User Manual

BBEdit™

Professional Code and Text Editor for the Macintosh

Bare Bones Software, Inc.

BBEdit™ 16.0

Product Design	*Jim Correia, Rich Siegel, Steve Kalkwarf, Patrick Woolsey*
Product Engineering	*Jim Correia, Seth Dillingham, Matt Henderson, Jon Hueras, Steve Kalkwarf, Rich Siegel, Steve Sisak*
Engineers Emeritus	*Chris Borton, Tom Emerson, Pete Gontier, Jamie McCarthy, John Norstad, Jon Pugh, Mark Romano, Eric Slosser, Rob Vaterlaus*
Documentation	*Fritz Anderson, Philip Borenstein, Stephen Chernicoff, John Gruber, Jeff Mattson, Jerry Kindall, Caroline Rose, Allan Rouselle, Rich Siegel, Vicky Wong, Patrick Woolsey*
Additional Engineering	*Polaschek Computing*
Icon Design	*Bryan Bell*
Factory Color Schemes	*Luke Andrews*
Additional Color Schemes	*Toothpaste by Cat Noon, and Xcode Dark by Andrew Carter. Used by permission.*
Additional Icons	*By icons8. Used under license*
Additional Artwork	*By Jonathan Hunt*
PHP keyword lists	*Contributed by Ted Stresen-Reuter. Previous versions by Carsten Blüm*

Published by:

Bare Bones Software, Inc.
P.O. Box 116
North Chelmsford, MA 01863 USA

phone: (978) 251-0500

website: **https://www.barebones.com/**

Sales & customer service: **sales@barebones.com**
Technical support: **support@barebones.com**

Copyrights, Licenses & Trademarks

cmark	*©2014 by John MacFarlane. Used under license; part of the CommonMark project*
LibNcFTP	*Used under license from and copyright © 1996-2010 Mike Gleason & NcFTP Software*
Exuberant Ctags	*©1996-2009 Darren Hiebert Source available here.*
PCRE2 Library	*Written by Philip Hazel and Zoltán Herczeg ©1997-2022 University of Cambridge, England*
Info-ZIP Library	*©1990-2009 Info-ZIP. Used under license.*
Quicksilver string ranking	*Adapted from sources and used under license.*
LetsMove	*Written by Andy Kim; adapted from sources.*
RSVerticallyCenteredTextFieldCell	*Written by Daniel Jalkut; ©2006 Red Sweater Software. Used under license.*
Lorem Ipsum generator	*Ported from this PHP implementation, originally by Josh Sherman. Used under license.*
mk_wcwidth	*By Markus Kuhn. Used with permission. Source available here.*
NSString+IATitlecase	*Copyright ©2016 Information Architects Inc. Used under license.*
UTF-8 validation	*Written by Daniel Lemire, Kendall Willets, and Zach Bjornson. Sources available; used under license.*
Verilog and VHDL support	*Written by Yasuhisa Kato, used with permission*
HTML Tidy Library	*©1998-2003 World Wide Web Consortium. ©2003-2015 by additional contributors. Used under license.*
sse2neon	*By John W. Ratcliff and others. Source available. Used under license.*
LSPKit	*By and copyright 2019 Christopher Atlan Source available. Used under license.*
JSON for Modern C++	*By and copyright ©2013-2021 Niels Lohmann. Source available. Used under license.*
NSJSONSerialization+Comments	*By and copyright ©2014 Alexander Blach Source available. Used under license*

BBEdit and “It Doesn’t Suck” are registered trademarks of Bare Bones Software, Inc.

[continued]

BBEdit® End User License Agreement:

You, the Licensee, assume responsibility for the selection of the BBEdit® program to achieve your intended results, and for the installation, use, and results obtained from the program. Your downloading and installing the program constitutes your acceptance of these terms and conditions. If you do not accept these terms and conditions, then do not download and install the program and contact Bare Bones Software, Inc., for a full refund if you have purchased a license.

License:

You may use the program and documentation and copy the program and documentation into any machine-readable or printed form for backup or support of your use of the program and documentation on any number of machines, provided that no copy of the program and documentation may be used by anyone other than you. You may use the program in this manner at no charge for a period of 30 days following your first launch of it; the program will be fully functional during this period. After this 30 day period, however, certain functions and features of the program will cease to operate. For so long as you are in compliance with the terms of this Agreement you may continue to use the program in this modified state and the terms of this Agreement will continue to apply; if you wish to restore full functionality and features of the program, you may purchase a license to do so and the terms of this Agreement will continue to apply.

You may not use or copy the program or documentation, or any copy thereof, in whole or in part, except as provided in this Agreement. You also may not modify or transfer (whether or not for consideration) the program or documentation, or any copy thereof, in whole or in part. If you use, copy, modify, or transfer the program or documentation, or any copy thereof, in whole or part, except as expressly provided for in this Agreement, your license is automatically terminated.

Whether or not you purchase a full functionality license to the program, from time to time Bare Bones Software, Inc., may make available to you those bug fixes, enhancements, error corrections, and other changes and additions to the program that it makes generally available to its other licensees at no charge, outside of major releases (collectively, “Updates”). Updates are considered to be minor releases. Bare Bones Software, Inc., owns the rights to these Updates, and Updates are included within the BBEdit program and are licensed to you for use under the terms of this Agreement.

From time to time Bare Bones Software, Inc., may make commercially available a subsequent major release to the version you have licensed (each, an “Upgrade”). You may obtain a license to that Upgrade, to which the terms of this Agreement shall also apply, and the Upgrade shall also be considered the “program” for these purposes. Bare Bones Software, Inc., owns the rights to all Upgrades as well. For clarity, you may use more than one version and release of the program under this license, but your use of all such versions and releases are subject to the terms of this Agreement.

Term:

The license is effective on the date you accept this Agreement and remains in effect until terminated as indicated above or until you terminate it. If the license is terminated for any reason, you agree to destroy the program and documentation, together with all copies thereof, in whole or in part, in any form, and to cease all use of the program and documentation.

Limited Warranty and Limitation of Remedies:

THE PROGRAM, DOCUMENTATION AND ANY SUPPORT FROM BARE BONES SOFTWARE, INC., ARE PROVIDED “AS IS” AND WITHOUT WARRANTY, EXPRESS AND IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL BARE BONES SOFTWARE, INC. BE LIABLE FOR ANY DAMAGES, INCLUDING LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF BARE BONES SOFTWARE, INC. IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY YOU OR ANY THIRD PARTY.

General Terms:

This Agreement can only be modified by a written agreement signed by you and Bare Bones Software, Inc. and changes from the terms and conditions of this Agreement made in any other manner will be of no effect. If any portion of this Agreement shall be held invalid, illegal, or unenforceable, the validity, legality, and enforceability of the remainder of the Agreement shall not in any way be affected or impaired thereby. This Agreement shall be governed by the laws of The Commonwealth of Massachusetts, without giving effect to conflict of laws provisions thereof. As required by United States export regulations, you shall not permit export of the program or any direct products thereof to any country to which export is then controlled by the United States Department of Commerce and its associated agencies and bureaus, unless you have the prior written approval of that Department, agency, or bureau. Use of the program and documentation by military and civilian offices, branches or agencies of the U.S. Government is restricted in accordance with the applicable Federal Acquisition Regulations (under which the program and documentation constitute “restricted computer software” that is “commercial computer software”) or Department of Defense Federal Acquisition Regulations Supplement (under which the program and documentation constitute “commercial computer software” and “commercial computer software documentation”) to that consistent with only those rights as are granted pursuant to the terms and conditions hereof.

Acknowledgment:

You acknowledge that you have read this Agreement, understand it, and agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between you and Bare Bones Software, Inc. which supersedes all proposals or prior agreements, oral or written, and all other communications between you and Bare Bones Software, Inc. relating to the subject matter of this Agreement.

Rev. Jan. 2021

Info-ZIP License

This is version 2009-Jan-02 of the Info-ZIP license. The definitive version of this document should be available at ftp://ftp.info-zip.org/pub/infozip/license.html indefinitely and a copy at http://www.info-zip.org/pub/infozip/license.html.

For the purposes of this copyright and license, “Info-ZIP” is defined as the following set of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly, Hunter Goatley, Ed Gordon, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath, Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der Linden, Igor Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs, Kai Uwe Rommel, Steve Salisbury, Dave Smith, Steven M. Schweda, Christian Spieler, Cosmin Truta, Antoine Verheijen, Paul von Behren, Rich Wales, Mike White.

This software is provided “as is”, without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the above disclaimer and the following restrictions:

- Redistributions of source code (in whole or in part) must retain the above copyright notice, definition, disclaimer, and this list of conditions.
- Redistributions in binary form (compiled executables and libraries) must reproduce the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution. The sole exception to this condition is redistribution of a standard UnZipSFX binary (including SFXWiz) as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal SFX banner has not been removed from the binary or disabled.
- Altered versions--including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, versions with modified or added functionality, and dynamic, shared, or static library versions not from Info-ZIP--must be plainly marked as such and must not be misrepresented as being the original source or, if binaries, compiled from the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases--including, but not limited to, labeling of the altered versions with the names “Info-ZIP” (or any variation thereof, including, but not limited to, different capitalizations), “Pocket UnZip,” “WiZ” or “MacZip” without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or the Info-ZIP URL(s), such as to imply Info-ZIP will provide support for the altered versions.

Info-ZIP retains the right to use the names “Info-ZIP,” “Zip,” “UnZip,” “UnZipSFX,” “WiZ,” “Pocket UnZip,” “Pocket Zip,” and “MacZip” for its own source and binary releases.

Contents

CHAPTER

1 Welcome to BBEdit

This chapter introduces you to BBEdit, a high-performance HTML and text editor for the Macintosh.

In this chapter

Getting Started

Thank you for selecting BBEdit, the premier text and HTML editor for the Macintosh.

If you are new to BBEdit, we recommend that you read at least Chapters 1 through 4 of this manual to familiarize yourself with the installation and basic operation of BBEdit. You may also wish to read or preview any other chapters that cover features you frequently use. After you have installed BBEdit, the best way to learn it is to use it.

If you have used earlier versions of BBEdit, we strongly recommend you take a few minutes now to review the current release notes for an overview of significant changes in this version:

`https://barebones.com/support/bbedit/current_notes.html`

What Is BBEdit?

BBEdit is a high-performance HTML and text editor. Unlike a word processor, which is designed for preparing printed pages, a text editor focuses on providing a means of producing and changing content. Thus, BBEdit does not offer fancy formatting capabilities, headers and footers, graphics tools, a thesaurus, or similar staples of feature-laden “office” software. Instead, it focuses on helping you manipulate text in ways that word processors generally cannot.

In service of this goal, BBEdit offers powerful regular expression–based (“grep”) search and replace, multi-file search, sophisticated text transformations, intelligent text coloring, and other features not usually found (or missed) in word processors.

BBEdit also has features that make it easier to edit specific kinds of text, such as source files for programming languages and HTML (Hypertext Markup Language) files for the World Wide Web. In fact, since the rise of the Web, BBEdit has been the tool of choice for Macintosh web designers who need more flexibility than visual web authoring tools can provide.

How Can I Use BBEdit?

Use BBEdit any time you need to create or edit web pages, source code, or text files of any kind. Whether you need to find (or change!) all the occurrences of some text in a set of files, or modify or reformat large text files of any sort, or quickly tweak a web page, BBEdit is the right tool for the job.

Development Environments

BBEdit found its initial following among the Macintosh programming community with its core editing- and development-oriented tools. Although we have added countless other features to BBEdit since its first incarnation, its source code editing capabilities are stronger than ever.

In addition to offering syntax coloring and function browsing for many different languages, BBEdit supports direct use of Perl, Python, and Ruby (as well as any other Unix scripting environment) and provides integrated support for the Git source control system. Chapter 14 provides more information on how to set up BBEdit for this type of work.

Writing HTML Documents

BBEdit is an ideal tool for preparing and editing HTML documents (web pages). In addition to many options for preparing text content, such as wrapping, case changes, and searching, BBEdit offers a powerful set of tools to make editing web pages easier. BBEdit’s Edit Markup command allows you to quickly add tags or modify existing tags, while the HTML Tools palette lets you access commands with just a click.

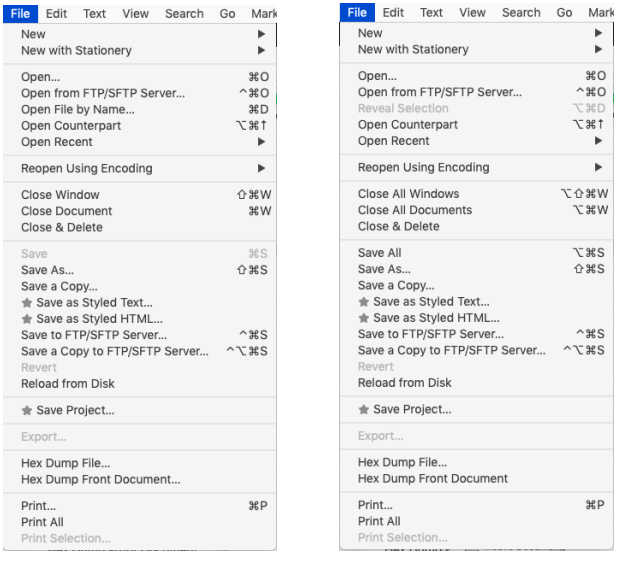
Using BBEdit, you can easily preview your work in most Macintosh web browsers, including Safari, Chrome, and Firefox, as well as via BBEdit’s native Preview feature and Windows browsers running under VMWare Fusion. For more information on using the HTML Tools to create, edit, and preview web pages, see Chapter 11.

Human Interface Notes

BBEdit enhances the behavior of its menus and dialogs as described in this section.

Dynamic Menus

IMPORTANT Many of BBEdit's pull-down menus are dynamic: if you hold down the Shift, Option, or Control key while a menu is open, you can see some of the items change. The illustration below shows what the File menu looks like normally (left) and when you hold down the Option key (right).



You can use the Shift, Option, or Control keys when you choose an item from a menu or when you use the Command-key equivalents.

Bypassing Options Dialogs

You may have noticed that commands that require additional settings to be made before they are performed appear on the menu with ellipses after their names. To bypass this step and use the command with its most recent settings, hold down the Option key while selecting the menu item. For example, "Zap Gremlins…" in the Text menu becomes "Zap Gremlins" when the Option key is pressed and, when chosen, will zap gremlins in the frontmost text document using the current settings.

Keyboard Shortcuts for Commands

Many of BBEdit's commands have keyboard shortcuts. BBEdit lets you reassign the shortcuts for any menu item, clippings entry, or script to suit your own way of working.

To change the keyboard shortcut for any menu command as well as any available scripts and text filters, go to the Menus & Shortcuts settings panel.

Contextual Menus

When you Control-click on selected text or at the insertion point in a text window, BBEdit's contextual menu will display a set of commands relevant to that location or text, as well as some appropriate standard commands (such as Cut/Copy/Paste, or Check Spelling) so you do not have to hunt around in the menu bar for them.

Dialog Box and Sheet Key Equivalents

You can use key equivalents to click buttons or select options in most of BBEdit's dialog boxes and sheets. Certain keys have the same meaning in all dialogs and sheets:

- Pressing either the Return or Enter key is the same as clicking the default button.
- Typing Command-period or pressing the Escape key is the same as clicking the Cancel button.
- You can use the Cut, Copy, Paste, Clear, and Select All commands (either from the Edit menu or with their Command-key equivalents) in any text field.

Feature Highlights

BBEdit 16 offers many powerful features for editing and processing text and code, and for managing your work. Here are some highlights:

- Wide range of source code editing enhancements via built-in support for the Language Server Protocol (LSP)
- Integrated support for multiple Notebooks to reduce "untitled text" clutter!
- New "Repeat Last Command" command
- Built-in language support for Go, R, Lisp, Rust, TOMML, Pixar USD, and more
- Enhancements to many built-in text transforms
- Support for expanding Emmet abbreviations

as well as all the powerful core features BBEdit is known for, including:

- Pattern Playground windows provide an interactive interface for experimenting with grep patterns
- Built-in support for Git
- 'Extract' command to gather & transform matching text from multiple files
- Support for the EditorConfig settings file convention
- Grep and Markdown syntax "cheat sheets"
- A selection of carefully crafted pre-installed color schemes
- Dedicated column selection and sorting commands
- Powerful Find Differences command compares files & folders
- Info popup offers live document statistics and file permissions adjustment

- Automatically preserves unsaved documents upon quit
- Project document-based website configurations (with support for content deployment)
- In-window Live Search to highlight and quickly jump between matches
- Direct editing of files within Zip archives—plus multi-file search & replace!
- Simplified settings and configuration management
- Text completion for easy insertion of words, syntax elements, and clippings
- 'Preview in BBEdit' supports filters, template pages and stylesheets
- Find and Multi-File Search windows provide a convenient interface to BBEdit's legendary search and replace capabilities

Info on New Features

In addition to these major features, BBEdit 16 also contains a vast number of behavioral and interface refinements, as well as performance enhancements and bug fixes. If you've been using an older version of BBEdit, we strongly recommend that you take a few minutes now to review the current release notes, which are available in the BBEdit Support section of our website.

```
https://www.barebones.com/support/bbedit/current_notes.html
```

Discussion Group

We maintain a public Google Group where our customers can discuss and share knowledge about using BBEdit.

```
https://groups.google.com/group/bbedit
```

Support Services

If you need information about using BBEdit (or any of our other products) the Support area of our website offers up-to-date details:

```
https://www.barebones.com/support/
```

You'll find a wide range of information there, including:

- Frequently Asked Questions (FAQ) — Information and answers for commonly encountered questions and problems. We strongly recommend you check the BBEdit FAQs before resorting to any other means of inquiry.
- Product Updates — The latest maintenance versions of our products are always available for download.

as well as access to language modules, sample scripts, developer info, and other materials.

How to contact us

If you have a registered copy of BBEdit (or any other Bare Bones product), and you can't find the information you need on our website, or if you encounter any problems with the software, please use the contact form on our website or send email to:

```
support@barebones.com
```

You can choose the "Contact Us" command in the Help menu to automatically create a pre-addressed email message to our support address (above), which contains the application and OS versions in its body. We will be grateful for a meaningful subject line and an appropriately detailed description of your question, issue, or feature request.

Note We do not offer telephone support. Please refer to the support resources available on our website for information and assistance, or contact us via email.

News and Notifications

You can sign up for our news and notification list by choosing the "Sign Up for News" command in the Help menu (of non-App Store builds).

Please fill in your name and email address, and we'll add you to our news notification list and let you know when important and/or interesting things are happening. The information you provide here is protected by our privacy policy:

```
https://www.barebones.com/company/privacy.html
```

CHAPTER

2 Installing BBEdit

This chapter tells you how to install BBEdit on your Macintosh. It also describes the files BBEdit creates, where it puts them, and how to install or remove optional components of BBEdit.

In this chapter

Basic Installation

BBEdit is supplied as a single application file. Specific system requirements and installation instructions are described below, and the organization of BBEdit's supporting files is described in subsequent sections.

System Requirements

IMPORTANT BBEdit 15 requires macOS 14 or later, and runs under macOS 26 "Tahoe" and on Apple Silicon-based Macs. The software will not run on any earlier version of macOS.

Installing BBEdit

When you download BBEdit, you will receive a standard disk image (".dmg") file. Your web browser may automatically mount the disk image once the download is complete; otherwise, you should double-click on the disk image file to mount it. Once the disk image is mounted, drag the "BBEdit" application over the adjacent icon for the Applications folder and drop it there to copy BBEdit onto your Mac's hard drive. You can then dismount (eject) the disk image and discard the ".dmg" file.

IMPORTANT If you have subscribed to BBEdit in the Mac App Store, you must install the software via the App Store application.

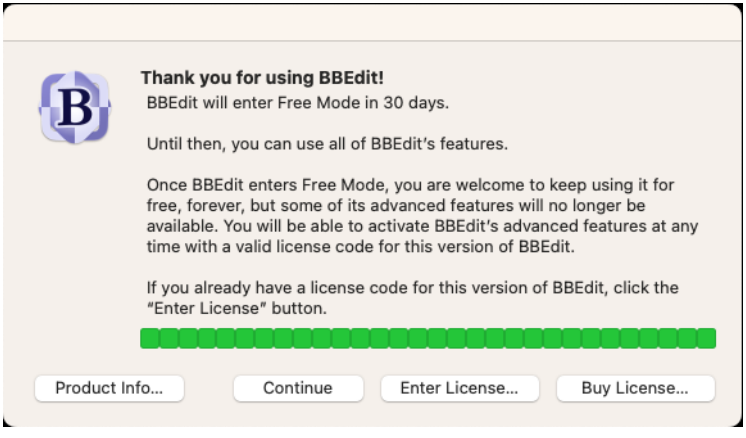
Automatic Relocation

If you launch BBEdit from any location other than your Mac's main "Applications" folder (/Applications/) or the account-specific equivalent (~/Applications/), BBEdit will offer to automatically relocate itself into the main "Applications" folder.

Note This behavior does not apply to copies of BBEdit obtained from the Mac App Store.

Evaluation Period

The first time you launch BBEdit after installation, it will display the following informational dialog:



When you first launch BBEdit, the app will enter a 30-day period, during which all of its features are available. Once that period has ended, you may keep using BBEdit for free, forever, with no nag screens or unsolicited interruptions, but some of its advanced features will no longer be available. (To learn more about what's different in Free Mod e, please see our handy comparison chart, or you may re-enable all of BBEdit's exclusive features at any time by purchasing a license.)

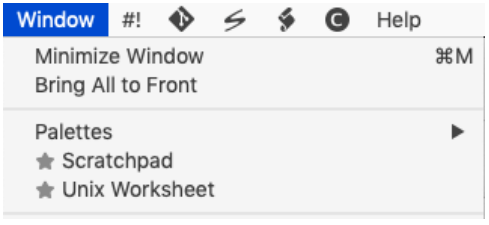
You may click "Continue" to start using BBEdit immediately, "Enter License..." to present an activation sheet into which you can enter your name and product serial number, or "Buy License..." to immediately send you to our online store where you can purchase a new license (or if BBEdit detects a license for an older version, it will instead send you to the upgrade verification page of our online store).

Note This behavior does **not** apply to copies of BBEdit obtained from the Mac App Store.

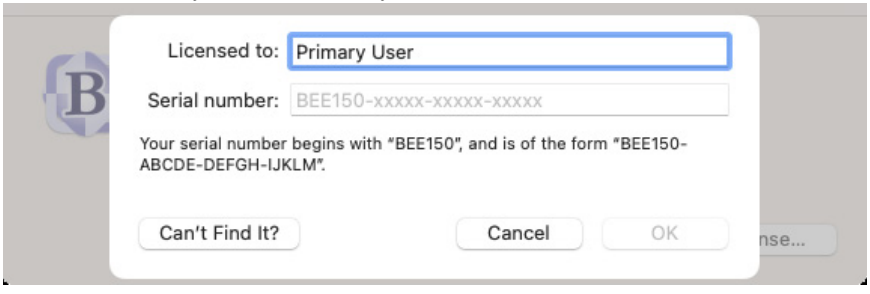
Activating BBEdit

While running in evaluation mode, BBEdit will operate with full functionality for up to 30 days. Once the demo period has ended, BBEdit will **remain** permanently functional with a revised feature set that includes its powerful text editing capabilities but not its web authoring tools or other exclusive features; these features will present a reminder dialog instead of functioning when chosen.

When the product is unlicensed, all menu commands corresponding to these exclusive features are badged with a “Star” icon, as shown here in the Window menu. BBEdit's exclusive features may be re-enabled at any time with a purchased license.



To activate the demo, click “Enter License...”and enter your name and the unique product serial number that you received with your order into the activation sheet:



Note We recommend you copy your BBEdit serial number and paste it into the activation sheet to avoid transcription errors.

Once you enter the serial number, your copy of BBEdit will be activated, and all demo restrictions will be removed.

IMPORTANT **In order to activate BBEdit 16, you must have a valid BBEdit 16 product serial number (one beginning with a prefix of “BEE160-” or “BEC160-”). You cannot activate the application with a serial number from any older version of BBEdit.**

If you have not yet purchased the product and thus do not have a serial number, leave this space blank. BBEdit will operate in a fully functional manner for 30 days (starting upon its initial launch), after which you must purchase a license and enter a valid BBEdit 16 serial number in order to continue using the software’s full functionality.

If BBEdit has already been activated, you may review the active serial number at any time by choosing the License command from the BBEdit menu to bring up the license info dialog. In order to change activation to a different serial number, click Edit License to bring up the activation sheet and enter the new information.

Note This behavior does not apply to copies of BBEdit obtained from the Mac App Store, which you may instead activate by purchasing a subscription within the app.

(Once you have entered a valid serial number, BBEdit will figure out whether you have Yojimbo installed and if so, it will offer to save your serial number there as a convenience.)

Checking for Updates

BBEdit offers the option to automatically check for updates; this behavior is controlled by the "Automatically check for updates" option in the Application settings panel. You can also directly check for updates at any time by choosing Check for Updates in the BBEdit (application) menu.

In order to update BBEdit when future maintenance releases become available, you need only apply the update when prompted. (Alternatively, you may quit BBEdit, and manually replace your existing copy with the updated version.) The first time you launch a newer version of the software, BBEdit will prompt you for any further actions which may be needed, such as updating the command line tools.

Note This behavior does not apply to copies of BBEdit obtained from the Mac App Store.

Upgrading from a Previous Version

IMPORTANT If you are upgrading from any version prior to BBEdit 8.5, in addition to installing the current application, you will need to manually copy over any items you wish to use from your existing "BBEdit Support" folder into BBEdit's application support folder. You should **not** simply rename your existing "BBEdit Support" folder. (See "BBEdit's Supporting Folders" on page 35.)

Please carefully read the remainder of this chapter, since the organization of BBEdit's supporting files has changed considerably. We have provided specific suggestions and tips for transferring your customized support items in each category.

Note This behavior does not apply to copies of BBEdit obtained from the Mac App Store.

Converting from TextWrangler

If you launch BBEdit with no existing settings, the application will instead look for any existing TextWrangler settings and migrate them.

In addition, if a TextWrangler settings data folder exists (at "/Users/<username>/Library/TextWrangler/` which is the default for recent versions), then BBEdit will copy that folder to "/Users/<username>/Library/BBEdit/".

Finally if a TextWrangler application support folder exists (at "/Users/<username>/Library/Application Support/TextWrangler/" BBEdit will copy this folder to "/Users/<username>/Library/Application Support/BBEdit/".

BBEdit's Supporting Folders

IMPORTANT In version 12.6 and later, BBEdit's **actual settings file** and its **prefs data folder**, together with other supporting files and folders are by default now stored within an application-specific subfolder of your login account's "Containers" folder, and the location of this folder is:

```
/Users/USERNAME/Library/Containers/com.barebones.bbedit/
```

where "USERNAME" is the short name of your login account.

Depending on when and how you installed BBEdit, its settings and supporting items may be contained entirely within the above application-specific container folder or selected items, such as BBEdit's application support folder, may remain outside the container.

Chief among these items is BBEdit's **application support folder**, whose default location remains:

```
/Users/USERNAME/Library/Application Support/BBEdit/
```

and this folder may contain items to extend BBEdit's capabilities, such as clipping sets (or items), language modules, scripts, and more. These items are organized into subfolders according to their purpose, as described below.

If this folder does not exist when BBEdit starts up, BBEdit will create it together with a number of standard subfolders, to which you can add any appropriate items. None of these folders are essential for doing basic tasks with BBEdit, and you can remove any or all of them that you don't use.

You can open BBEdit's (active) application support folder in the Finder at any time by choosing the corresponding entry in the Folders submenu of the BBEdit (application) menu.

Application Support Folder Contents

BBEdit's application support folder contain various subfolders, each of which holds a specific type of support item.

You can relocate BBEdit's application support folder to Dropbox or iCloud Drive as described later in this chapter; however, you cannot independently relocate this folder or any of its subfolders. We also recommend that you do not try to share scripts between BBEdit and other applications, nor should you attempt to store BBEdit's application support folder on any remote (server) volume.

Quick Access to Subfolders

The Folders submenu in the BBEdit (application) menu lists all potential subfolders of BBEdit's application support folder as well as entries that provide access to other select locations, such as "System Diagnostic Folders" for quick access to the diagnostic logs generated by macOS.

You can select any subfolder's name to open that folder in the Finder (creating it first, if necessary).

You can also access all subfolders via the Folders tab of the Setup window.

Attachment Scripts

This folder does not exist by default, but you may create it at any time. The Attachment Scripts folder contains AppleScripts which are run at specific points: when BBEdit starts or quits; and when documents are open, saved, and closed.

Clippings

BBEdit will automatically create this folder if it does not exist. The Clippings folder contains clipping items. These items are text files which appear in the Clippings menu and palette, and whose contents can be inserted into a document by choosing them directly, or via text completion. Clippings may also contain special placeholders which insert varying or context-sensitive information—for example, a date or the name of the current file. (See Chapter 12 for more information on creating and using clippings.)

Color Schemes

BBEdit will automatically create this folder when needed. The Color Schemes folder stores any custom color schemes which you have saved within the Text Colors settings panel (or which you have download and copied over). Each scheme is stored within a separate ".bbColorScheme" file.

Completion Data

This folder does not exist by default, but you may create it. The Completion Data folder contains tags files (or aliases to tags files) which can provide additional text completions for editing documents in the corresponding languages.

These tags files should be in the format generated by 'bbedit --maketags', and must be placed in subfolders corresponding to their languages.

Each subfolder should have the exact name of its language as that language appears in the list of installed languages (or on the Languages popup menu).

For example, the subfolder containing a Python tags file must be named "Python", and the subfolder containing a tags file for ANSI C must be named "ANSI C".

Custom Keywords

BBEdit will automatically create this folder if it does not exist. This folder provides an easy, supported way to add keywords to any installed languages via text files.

In order to do so, place one or more files containing the keywords that you wish to be colored inside this folder. Each file's name should map to the appropriate language, e.g. ".js" for JavaScript files. You can have multiple keyword files mapped to the same language, if you wish.

Each file should contain UTF-8 text (no BOM) with one keyword per line. Keyword lookups are case-sensitive if the corresponding language is case sensitive, or case-insensitive otherwise.

All keywords within a keyword file are colored using the "Language Keywords" color.

Custom keyword files can also use an Emacs mode line to specify their language type. This option addresses the case in which you may want to supply keywords for a language that doesn't have any filename extension mappings. For example, ".php" maps to the "PHP in HTML" language type, which means that embedded PHP keywords need to be in a keywords file that maps to the "PHP" language. Such a keyword file's contents would look like this:

```
-*- mode: php; -*-
keyword_one
keyword_two
...etc...
```

HTML Templates

BBEdit does not create this folder by default, but one may exist from previous versions. This folder contains template files which are used by the New HTML Document command. In order to use these templates elsewhere, you may either choose this folder to be the Templates & Includes folder for the website configuration within a specific project, or you can copy the template files into an already-designated site templates folder. Please see Chapter 11 for more information on BBEdit's HTML tools.

Upgrading You should move or copy over any customized template or include documents that you wish to preserve.

Language Modules

BBEdit does not create this folder by default, but will do so if necessary. The Language Modules folder allows you to add syntax coloring and function navigation support for additional languages by installing language modules.

IMPORTANT **Please do not attempt to extract or modify the language modules contained in the BBEdit application bundle.**

A list of additional modules from third-party developers is available on our website, or you may create your own compiled or codeless language modules (see "Codeless Language Modules" on page 406).

Upgrading You should move or copy over any compatible third-party language modules that you wish to preserve.

Menu Scripts

This folder does not exist by default, but you may create it. The Menu Scripts folder contains AppleScripts that are attached to BBEdit menu items. (For more details on using menu scripts, please see "Attaching Scripts to Menu Items" on page 359.)

Upgrading You should move or copy over any menu scripts that you wish to preserve.

Notes.bbnotebookd

BBEdit will automatically create this bundle. This bundle contains all the data and metadata necessary for BBEdit to preserve and present your stored notes in the default Note window (For complete details on working with notes, please see "The Notes window and Notebooks" on page 242.)

Packages

This folder does not exist by default, but you may create it. The Packages folder contains pre-packaged sets of supporting items. (For information about creating packages, please see “Packages” on page 407.)

Readme.txt [file]

This file contains an abbreviated description of the default contents of BBEdit’s application support folder.

Scratchpad [file]

BBEdit automatically creates this file, which contains the data for BBEdit’s main Scratchpad. Removing this file will result in the loss of your Scratchpad data.

Scripts

BBEdit will automatically create this folder if it does not exist. The Scripts folder may contain AppleScript files, Automator workflows, text factories, and executable Unix files (scripts). Items placed in this folder will appear in the Scripts menu (left), and you may place items within subfolders (up to four levels deep) to organize them.

You may run these items from the Scripts menu, the floating Scripts palette, or via assigned key equivalents. (You may use the Menus & Shortcuts settings panel to assign a key equivalent to any item in the Scripts menu.)

BBEdit runs such items by simply loading the item and calling it directly, without providing any inputs. (Naturally, AppleScript scripts and Automator actions may query BBEdit for more information, and Unix scripts may obtain information from the environment variables that BBEdit sets, while text factories will use their stored target list if any.)

Upgrading If you are upgrading from BBEdit 8.5 or 9, the first time you launch BBEdit 15, it will automatically copy all of your existing Unix scripts into this folder.

If you are upgrading from any version prior to 8.5, you must instead manually move or copy over any customized scripts that you wish to preserve. Note also that scripts written for use with such older versions of BBEdit may no longer work. (Please see Chapters 13 and 14 for more details and tips on modifying your existing AppleScripts and Unix filters & scripts.)

Setup

BBEdit will automatically create this folder if it does not exist. The Setup folder contains configuration data such as: stored file filters, FTP/SFTP bookmarks, key bindings, and grep patterns. Thus, if you have relocated your BBEdit application support folder into either your Dropbox or iCloud Drive folder, these items will be synchronized.

The Setup folder may contain any or all of the following data files.

BBEdit Settings Backup.plist

An automatically-created “snapshot” of your current BBEdit prefs options. You should not attempt to directly edit the contents of this file.

Enabled Clipping Sets.plist

BBEdit stores information about clipping set activation in this file. You should not attempt to directly edit the contents of this file.

File Filters.filefilters

BBEdit stores all user-defined file filter patterns in this file. You should not attempt to directly edit the contents of this file; instead, please use the Filters panel of the Settings window to add, modify, or remove stored grep patterns.

FTP Bookmarks.xml

BBEdit stores user-defined FTP and SFTP bookmarks in this file. You should not attempt to directly edit the contents of this file; instead, please use the Bookmarks panel of the Settings window to add, modify, or remove bookmarks.

Grep Patterns.xml

BBEdit stores user-defined search patterns in this XML file. You should not attempt to directly edit the contents of this file; instead, please use the Patterns panel of the Settings window to add, modify, or remove stored grep patterns.

Upgrading If you have created any custom grep patterns in a previous version of BBEdit, these patterns will be imported; otherwise, BBEdit will create a set of factory default patterns.

Menu Shortcuts.xml

BBEdit stores keyboard shortcuts for menu commands in this XML file. You should not attempt to directly edit the contents of this file.

Not Menu Shortcuts.xml

BBEdit stores other keyboard shortcuts in this XML file. You should not attempt to directly edit the contents of this file.

Shutdown Items

This folder does not exist by default, but you may create it at any time. The items in this folder are opened when you quit BBEdit. Usually, this function is used to run scripts of some sort.

Shutdown items are run after all windows have been closed, and only if BBEdit is actually quitting. Thus, if you wish to run any items as the immediate result of a Quit command, you should write a menu script attached to BBEdit•Quit.

Note In some previous versions of BBEdit, shutdown items were run before all windows were closed, and were run whenever the application was told to quit (either by the Quit menu command or via the scripting interface), regardless of whether it actually quit or not.

Upgrading You should move or copy over any shutdown items that you wish to preserve.

Startup Items

This folder does not exist by default, but you may create it at any time. When launched, BBEdit will open any items it finds in this folder.

If the items present are documents of a type that BBEdit knows how to handle (such as text files or projects), BBEdit will open them directly. If you place a compiled AppleScript in this folder, BBEdit will execute the script. If you place a folder alias here, BBEdit will open a disk browser window based at that folder.

If you place other types of items in this folder, BBEdit will ask the Finder to open them. If you often edit HTML files, for instance, you may want to place an alias to your Web browser (or your visual HTML editor) in the BBEdit Startup Items folder so that it will start up automatically whenever you run BBEdit.

Upgrading You should move or copy any file or application aliases that you wish to preserve. If you have any AppleScripts startup items, please see the preceding upgrade note for the Scripts folder about script compatibility.

Stationery

This folder does not exist by default, but you may create it at any time. The Stationery folder contains stationery files for use with BBEdit's New with Stationery command. Stationery files may be placed within subfolders (up to four levels deep) to organize them.

You can hide, or show, all items included from the global folder by using the menu item "Hide/Show Library Stationery".

Upgrading You should move or copy over any stationery documents that you wish to preserve.

Text Filters

This folder does not exist by default, but you may create it at any time. The Text Filters folder contains executable items, such as compiled AppleScripts, Automator workflows, text factories, and Unix filters, which you may apply to the frontmost document via the Apply Text Filter command in the Text menu

When you apply such an item, BBEdit will pass either the selected text (if any) or the contents of the entire document (or the clipboard) on STDIN to Unix executables and filters, as a reference to a 'RunFromBBEdit' entry point in AppleScripts, as text input to Automator workflows, and as a source to text factories. (An AppleScript script intended for use as a text filter must have a 'RunFromBBEdit' handler.)

AppleScript scripts and Automator workflows should return a string which BBEdit will use to replace the selection range, Unix filters should write to STDOUT, and the text emitted by a text factory will replace the selection range.

Paste Using Filter

You may also apply any available text factory or Unix filter to the current contents of the clipboard via the Paste Using Filter submenu of the Edit menu.

Note Due to internal restrictions, the Paste Using Filter command does **not** allow the use of AppleScript filters or Automator actions.

Unix Worksheet.worksheet

BBEdit automatically creates this file, which contains the data for BBEdit's global Unix worksheet window. (Choose Show Unix Worksheet in the Window menu to open this worksheet.) Removing this file will result in the loss of your global Unix worksheet data.

Superseded App Support Folders

Upgrading BBEdit 15 does not use the Text Factories or Unix Support subfolders, though these folders may exist if they were created by a prior version. Instead, the first time you launch BBEdit 15 after upgrading from a substantially older (pre-10.0) version, it will copy all existing Unix scripts into the Scripts folder, and all existing text factories and Unix filters into the Text Filters folder.

BBEdit Backups

When the “Keep historical backups” option (in the Text Files settings pane) is enabled, BBEdit will preserve backups in the “BBEdit Backups” folder. (This option is enabled by default whenever the “Make backup before saving” option is enabled.)

You can access this folder at any time by choosing Document Backups in the Folders submenu of the BBEdit (application menu), or via the list in the Folders pane of the Setup window.

The backup folder will contain one folder for each day's backup files. The format of the dated folder name is static and non-localized: YYYY-MM-DD, and each day’s backup folder will contain all of the backup files made on that day, each file itself being named in a time-stamped format.

NOTE The current default location of this folder is “~/Library/Containers/com.barebones.bbedit/Data/Documents/BBEdit Backups/” In versions prior to BBEdit 14, the default location of the “BBEdit Backup” folder was “~/Documents/BBEdit Backups/” and if such a folder exists, BBEdit will continue to use it. You should **not** however attempt to change this folder’s location but allow the app to manage it.

Settings Files and Folders

When you start up BBEdit, it may create the files and folders noted in this section.

BBEdit Settings File

All of BBEdit’s basic settings settings are stored within its settings file:

`/Users/USERNAME/Library/Containers/com.barebones.bbedit/Data/Library/Settings/com.barebones.bbedit.plist`

which is created and maintained using standard system services.

In addition to the settings documented in Chapter 10, you may adjust additional expert settings settings outside of BBEdit by issuing suitable “defaults write” commands. For a complete list of available expert settings settings, please see the “Expert Settings” page on our website:

`https://www.barebones.com/support/bbedit/ExpertSettings.html`

You can access this page on our website at any time, or use the “Expert Settings” link in the Help book, or click the “Expert Settings Help” button in the Expert prefs pane to open that page in your default browser.

Transferring Settings Settings

Since macOS does not support relocation of an application's core settings, you cannot directly sync these settings. You can however export BBEdit's core settings file to other machines to 'seed' them with your preferred settings settings.

You can transfer your existing settings settings (including your license) as follows

First, make sure BBEdit is not running on either machine, then open the Terminal utility app on the source machine and enter the following command (you can copy & paste it):

```
defaults export com.barebones.bbedit ~/Desktop/BBEditSettings.plist
```

Next, transfer the resulting file ("BBEditSettings.plist") to the Desktop on the target machine by any standard means (e.g. via AirDrop, iCloud Drive, or a USB thumb drive).

Then, launch the Terminal app on your new Mac and run the following command to apply the transferred prefs (again, you can copy & paste it):

```
defaults import com.barebones.bbedit ~/Desktop/BBEditSettings.plist
```

Note This procedure applies only to BBEdit's settings settings, not the contents of its application support folder or any files and/or other data that you may want to preserve.

BBEdit Settings Data Folder

By default, BBEdit stores your ancillary settings data within a subfolder of its sandbox container folder, but in pre-existing installations, this folder may still be found at:

`/Users/USERNAME/Library/BBEdit/`

The standard contents of this folder are as follows.

Auto-Save Recovery

BBEdit will automatically create this folder. The Auto-Save Recovery folder contains information which BBEdit can use to recover the contents of unsaved documents after a crash, or to restore them at launch.

WARNING Please exercise caution as removing items from this folder can cause data loss.

Document State.plist

BBEdit stores state information for individual documents in this file.

Recent Files & Favorites

This folder is no longer used and may be deleted.

Recent Folders & Favorites

This folder is no longer used and may be deleted.

Save Application State.appstate

BBEdit stores application state info in this file.

Saved Sources.xml

BBEdit stores all user-defined search sources in this file.

Sleep State.appstate

BBEdit stores application state info in this file.

IMPORTANT BBEdit 14.5 and later no longer support locating BBEdit's **prefs data folder** in Dropbox or iCloud Drive. If BBEdit detects such a folder when starting up, it will instead copy that folder's contents to the appropriate local location, and use the latter going forward.

Sharing Application Support Info via Dropbox

If you use Dropbox, you may relocate your BBEdit application support folder to your active Dropbox folder, and BBEdit will use its contents from Dropbox rather than the default location within your account's local "Library" folder. In this way, you can easily share supporting files among multiple BBEdit installations.

In order to do this:

- Quit BBEdit if it is running.
- Move your BBEdit application support folder (/Users/USERNAME/Library/Application Support/BBEdit/) into a folder named "Apps" within your Dropbox folder, so that its final location is "/Users/USERNAME/Dropbox/Apps/BBEdit/. (Since Dropbox does not create an "Apps" folder by default, you may need to do so.)
- Relaunch BBEdit.

Note Though BBEdit will still honor the original location of /Users/USERNAME/Dropbox/Application Support/BBEdit/ we recommend switching to the new, standard location detailed above.

Sharing Application Support & Backups via iCloud Drive

If you use iCloud Drive, you may relocate your BBEdit application support folder to iCloud Drive, and BBEdit will use its contents from that location rather than the default location within your account's local "Library" folder (~/Library/Application Support/BBEdit/). In this way, you can share BBEdit's supporting files among multiple machines.

Due to restrictions imposed by App Sandboxing, we have made some changes in how BBEdit interacts with iCloud Drive when looking for support folder items. In particular, BBEdit is unable to locate or make use of items within your top-level iCloud Drive folder.

Transition Instructions for Previous iCloud Users

If you **previously used** 'iCloud Drive/Application Support/BBEdit/' to store BBEdit's app support folder, please follow these instructions:

1. **Look in your iCloud Drive folder. If you see a folder named "BBEdit" with the BBEdit application icon imprinted on it, skip to step 4.**
2. **Start BBEdit. If necessary, allow sandbox access.**
3. **Quit the application.**
4. **You should see a new "BBEdit" folder within iCloud Drive.**
5. **Open the "BBEdit" folder within iCloud Drive, and use the Finder's "New Folder" command to create a new folder inside of it. Name the new folder "Application Support".**
6. **Open up iCloud Drive/Application Support/BBEdit/, and copy its contents into the new folder you've just created.**

To summarize, what you must do is move the **contents** of 'iCloud Drive/Application Support/BBEdit/'**into** 'iCloud Drive/BBEdit/Application Support/'.

The next time you start BBEdit, it will use your support folder contents from the new location.

Setup Instructions for iCloud Users

If you **have not previously used** iCloud Drive to share your BBEdit app support folder, we recommend you instead use Dropbox for this purpose (if possible); otherwise, you may start doing so by following these instructions:

1. **Look in your iCloud Drive folder. If you see a folder named "BBEdit" with the BBEdit application icon imprinted on it, skip to step 5.**
2. **Start BBEdit. If necessary, allow sandbox access.**
3. **Quit the application.**
4. **You should see a new "BBEdit" folder within iCloud Drive.**
5. **Open the "BBEdit" folder within iCloud Drive, and use the Finder's "New Folder" command to create a new folder inside of it. Name the new folder "Application Support".**

6. **Make a new window in the Finder, and then use the Finder's “Go to Folder” command and enter “~/Library/Application Support/BBEdit/” in the panel that appears.**

7. **Copy the contents of this folder into the “Application Support” folder that you created in step 5.**

The next time you start BBEdit, it will use the application support folder contents from the new location.

CHAPTER

3 Working with Files

This chapter discusses how to use BBEdit to manipulate text files.

In this chapter

Launching BBEdit

To launch BBEdit, double-click the BBEdit application icon or a BBEdit document. Holding down the following keys at launch has the indicated effects, overriding any startup options set in the Application settings pane. When one of these key combinations is applied, BBEdit will beep after it finishes launching.

Modifier	Function
Option	Suppress startup items only.
Shift	Disable all external services and startup items, and skip reopening all documents **except** those which contain unsaved changes.
Command-Control-Shift	Disable all external services and startup items, and optionally discard auto-recover information (which will result in the loss of any unsaved changes).

Startup Items

When launched, BBEdit will look for a folder named Startup Items in the its application support folder (see “Sharing Application Support Info via Dropbox” on page 44). If this folder is found, BBEdit will open any items it finds in the folder.

If the items present are documents of a type that BBEdit knows how to handle (such as text files or projects), BBEdit will open them directly. If you place a compiled AppleScript in this folder, BBEdit will execute the script. If you place a folder alias here, BBEdit will open a disk browser window based at that folder.

If you place other types of items in this folder, BBEdit will ask the Finder to open them. If you often edit HTML files, for instance, you may want to place an alias to your Web browser (or your visual HTML editor) in the BBEdit Startup Items folder so that it will start up automatically whenever you run BBEdit.

Sandboxing

BBEdit is now a “sandboxed” application.

“App Sandboxing” is a term that refers to a collection of security technologies built in to macOS. Sandboxing is intended to protect you and your data by limiting the operation of applications to their intended use, which in turn makes it harder for malicious software or accidental misuse to cause data loss or damage to your computer.

One of the core concepts of App Sandboxing is that BBEdit, as a sandboxed application, is not allowed to use any of your files or folders without your explicit permission. You can grant this permission in one of a small number of ways, including (but not necessarily limited to):

- Asking BBEdit to open a file from the Finder by double-clicking it;
- Dragging a file from the Finder on to BBEdit's application icon;

- Using BBEdit's “Open…” command to choose a file or folder.

However, as an advanced developer tool, BBEdit frequently requires access to files or folders that you may not have specifically asked it to open, for safe and legitimate reasons; including (but again not limited to):

- If you have located BBEdit's application support folder in Dropbox or iCloud Drive, BBEdit needs to be able to figure out where the folder actually is, and be able to use that location;
- When opening a document, BBEdit needs to know:
  * whether the file is under revision control with Git;
  * whether there are one or more .editorconfig files in the active file’s path (since such files control the active file’s editing options);
  * whether any ctags (“tags”) data is available to support BBEdit's completion and “Find Definition” features;
- When previewing a file with unsaved changes, BBEdit needs to be able to write a temporary file out in the same directory, so that relative links within the file remain correct;
- If you have turned on “Make Backups” in the Text Files settings, BBEdit needs to be able to write backup and auto-recovery files in the same directory as the file being saved;
- The “Open File by Name” feature needs to be able to examine all possible directories that may contain files for which you're searching, including some that you may never have used before in BBEdit;
- Many dialog boxes in BBEdit which allow you to choose a file or folder give you the choice of directly entering a file path;

...and many more similar cases.

Without unrestricted access to your files and folders, many of BBEdit's most useful features, from the basic to the most powerful, won't work at all; or they may misbehave in unexpected ways. At the very least, this hinders your ability to get work done.

In order to resolve this fundamental conflict between security and usability, we have devised a solution in which BBEdit requests that you permit it the same sort of access to your files and folders that would be available to a non-sandboxed version.

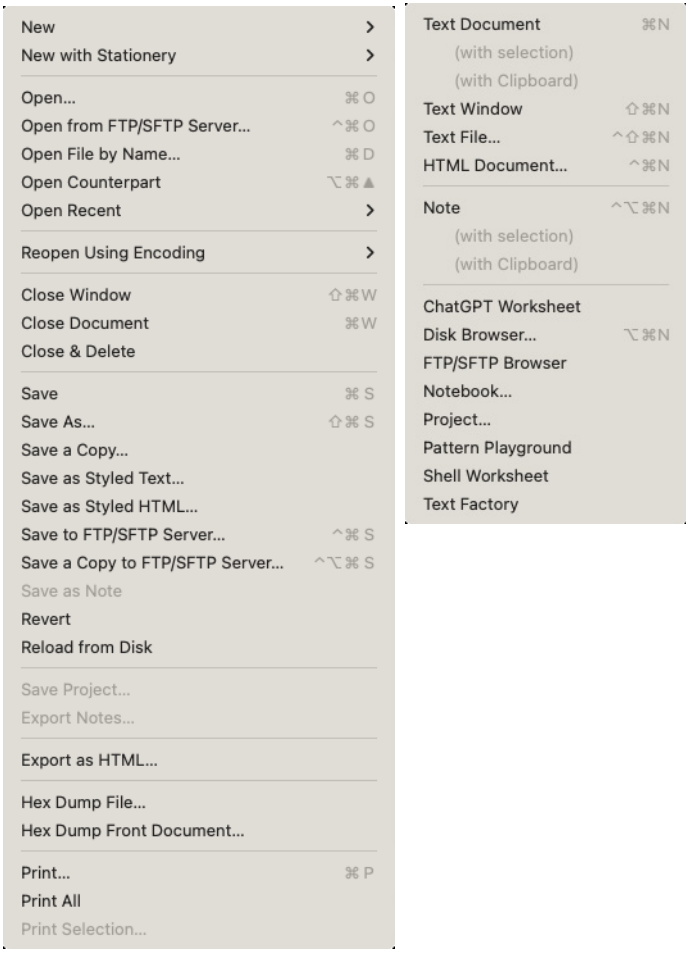
For this reason, the first time you start BBEdit, it will prompt you to allow this access. The prompt will not be repeated; so if you decline to allow this access and later reconsider, you may go to the Application settings pane, and click on the “Allow” button in the “Sandbox Access” section.

This in no way compromises your security or that of your computer while using BBEdit, but does allow BBEdit to function at its fullest potential.

If you have been using a previous non-sandboxed version, your existing settings and support folder contents are unaffected by this change.

Creating and Saving Documents

To create a new text document or special-purpose window within BBEdit, pull down the File menu and open the New submenu. Since BBEdit uses different kinds of documents for specific purposes, you will see several options, as follows:



The available commands and their effects are as follows:

- Text Document: Opens an empty text document.
- (with selection): Opens a new text document containing any text selected in the active document and having the same display font, saving you the trouble of copying and pasting it.
- (with Clipboard): Opens a new text document and automatically pastes the contents of the current clipboard into it.

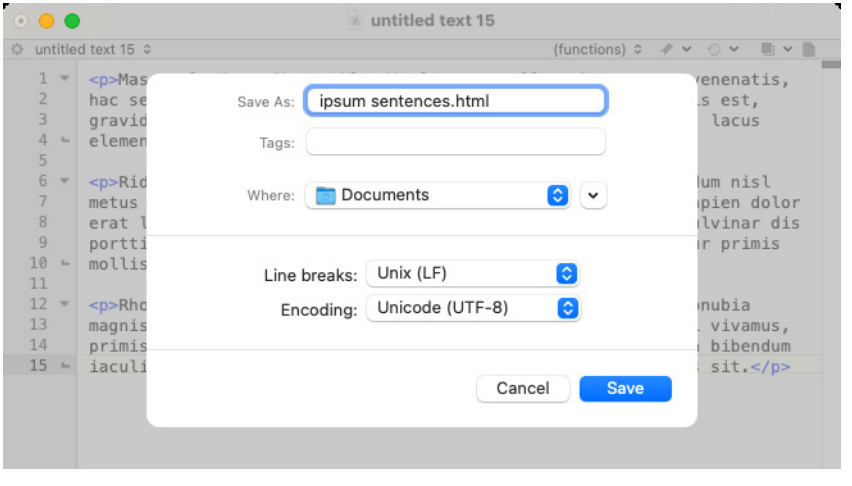
- HTML Document: Brings up a dialog with options for creating a new HTML document (see Chapter 11 for more information on working with HTML documents).
- Text Window: Opens a new text window (see “Text Windows” later in this chapter for more information).
- Project: Opens a new project window (see “Using Projects” later in this chapter for more information).
- Disk Browser: Opens a new disk browser (see Chapter 9 for more information).
- FTP/SFTP Browser: Opens a new FTP/SFTP browser (see later in this chapter for more information).
- Shell Worksheet: Opens a new shell worksheet using your default shell.
- Text Factory: Opens a new Text Factory window (see “Text Factories” in Chapter 5 for more information).

You can also create a new text document by selecting text in any application which supports the system Services menu, and choosing the New Window with Selection command from the BBEdit submenu of the Services menu. BBEdit will open a new text window containing a copy of the selected text.

When you want to save a new text document:

1 **Choose the Save or Save As command from the File menu.**

BBEdit opens a standard Save sheet:



2 **Give the file a name.**

3 **Change the automatically-provided filename extension (if necessary).**

BBEdit will automatically provide a filename extension based on the current document’s language.

4 **Change any desired options (see below).**

5 **Click Save to save the file.**

You can also create a new document from the selected text in any open window with BBEdit's contextual menu. Simply Control-click the selected text and choose New (with selection) or Save Selection from the menu that appears. Depending on which command you choose BBEdit will either create a new editing window containing the selected text, or display the Save dialog and allow you to create a new file containing the selected text. The new file will use the same options (see "File Saving Options," below) as those of the original parent document.

Saving a Copy of a File

You can save a copy of a file with BBEdit's Save a Copy command in the File menu. Just like the Save As command, the Save a Copy command displays a Save dialog and lets you choose a name and location for the file. However, unlike the Save As command, where BBEdit will start working with the new file you saved in place of the original, when you use Save a Copy, you create a new file in the designated location, but keep working with the original file.

For example, say you are editing a document called Test.c and use the Save a Copy command to save a document called Backup-Test.c. The next time you choose the Save command, BBEdit saves the changes to Test.c and not to Backup-Test.c.

File Saving Options

BBEdit's Save dialog is the standard Macintosh Save dialog with these additions:

Line Breaks

The Line Breaks menu let you choose what kinds of line breaks BBEdit writes when you save the file. Choose:

- Unix line breaks (ASCII 10) for most purposes, including use with modern Mac applications, or for files being saved to a Unix file server. This is the default option.
- Legacy Mac OS line breaks (ASCII 13) if you will be using the file with other applications which expect this format.
- Windows line breaks (ASCII 13/10) if the file resides on a Windows file server or if you will be sending it to someone who uses a Windows- or DOS-based system.

Encoding

BBEdit lets you save documents using any character set encoding supported by macOS, including a variety of Unicode formats (see "Saving Unicode Files" on page 65). To select an encoding, choose its name from the Encoding pop-up menu. The list of available encodings is controlled by your settings (see "Text Encodings Settings" on page 275).

When you select an encoding that requires a Unicode file format, you can also choose "Unicode" as an option from the Line Breaks pop-up menu in this dialog. (Unicode has its own line-ending standard.)

Note You can choose which encodings appear in the Encoding pop-up menu in the Text Encodings settings pane.

Encoding and File Type Codes

BBEdit no longer uses the “TUTX”, “utxt”, or “UTF8” HFS file type codes to determine the text encoding of a file’s contents.

File State

If you modify a document’s window position or display settings and then save the document, BBEdit stores state information, which it will use to reopen that document in the same manner.

BBEdit captures only those settings which are fundamental to the document (window position, selection range, folds, splitter setting), or any settings which vary from the global settings. (The latter ensures that changes to the global settings are never inappropriately overridden by stored display options derived from prior global or default settings settings.)

For example, say BBEdit’s default display font is Menlo, and you open (or create), save, and close a document which uses that font. If you then change BBEdit’s default display font to Monaco before reopening that document, the document will display in Monaco.

Note The above example uses the display font option for illustration, but the same principle applies to any document display option which derives from BBEdit’s global settings.

EditorConfig

BBEdit supports the ‘EditorConfig’ settings file convention. You can learn more about this convention at the EditorConfig project website:

`http://www.editorconfig.org/`

BBEdit supports most, but not all, of the core EditorConfig properties listed here:

`https://github.com/editorconfig/editorconfig/wiki/EditorConfig-Properties`

except that the ‘end_of_line’ and ‘max_line_length’ properties are **not** supported.

In addition to the core EditorConfig properties, there are some BBEdit-specific additions. First, BBEdit supports the following keys which originated as Emacs variables:

- coding: similar to ‘charset’, but allows you to specify any IANA character set name.
- mode: allows you to explicitly specify the language. Many of the Emacs-style mode names work, as long as they correspond to supported languages in BBEdit. In addition, any installed language in BBEdit may be expressed as a mode name by lowercasing its name and replacing spaces with dashes. For example, “Ruby in HTML” becomes ‘ruby-in-html’; or “Strings File” becomes ‘strings-file’.
- make-backup-files: set to 1 or 0, determines whether BBEdit makes a backup of the file when saving.
- backup-inhibited: if present and set to 1, will explicitly suppress the creation of backup files when saving.

Note An applicable EditorConfig ‘coding’ or ‘charset’ value will override any encoding xattr applied to an individual file.

Finally, BBEdit supports some keys which are explicitly specific to its own settings. All of these keys have names that begin with ‘x-’ in order to prevent collisions with any future core keys. These correspond directly to individual document settings, and if present will override BBEdit’s global settings:

- x-typographers-quotes: Use Typographer’s Quotes
- x-balance-while-typing: Balance While Typing
- x-soft-wrap-text: Soft Wrap Text
- x-soft-wrap-mode: (string) Must be one of ‘CharacterWidth’, ‘WindowWidth’, or ‘PageGuide’
- x-soft-wrap-limit: (integer) if the wrap mode is ‘CharacterWidth’, specifies the number of characters
- x-font-name: (string) the display font name
- x-font-size: (integer) the display font point size
- x-show-invisibles: Show Invisibles
- x-show-spaces: Show Spaces
- x-show-tabs: Show Tabs
- x-show-line-breaks: Show Line Breaks

Unless otherwise noted, these application-specific keys are all Boolean flags.

Emacs Local Variables

Emacs (the popular Unix text editor) supports a convention in which you can define Emacs-specific settings in a block of text near the end of the file, or in the first line of the file. This convention helps maintain consistency when sharing files among a group of people, or across multiple systems.

For general information on Emacs variables, please see the GNU Emacs manual:

`https://www.gnu.org/software/emacs/manual/html_node/emacs/Specifying-File-Variables.html`

BBEdit will read and honor the “coding”, “tab-width”, and “x-counterpart” variables in any file which contains an Emacs variable block, and adjusts the value of the “coding” variable if you change the document’s encoding by using the Encoding popup.

If a file contains an Emacs variable block (or line) having a “mode” variable, BBEdit will attempt to match the mode name against all currently recognized languages, before attempting to match the file name suffix or guess based on the file's contents.

You may add an Emacs variable block (or lines) to any document either directly, or by selecting the Emacs Variable Block command from the Insert submenu of the Edit menu.

Here is an example variable block from a plain text file:

```
Local Variables:
coding: ISO-8859-1
tab-width: 8
End:
```

You may also add the BBEdit-specific variable “make-backup-files” to control whether or not BBEdit should back up a given file. For more details, please see “Controlling Backups with Emacs Variables” on page 277.

Saving with Authentication

BBEdit supports saving files that require administrator privileges, if you possess the necessary user and password information to enable this. For example, you can edit and save files that are owned by, and only readable by, the “root” user. Authenticated saving is particularly useful in conjunction with the “Show Hidden Items” option in the Open dialog, which allows you to see and open files in hidden folders (like /bin and /usr).

When you open a file for which you do not have write privileges, BBEdit will display a locked padlock icon in the status bar. To edit the file, click the padlock icon. BBEdit will prompt you to confirm whether you wish to unlock the file. (Option-click the padlock icon to skip the confirmation dialog.)

When you are finished editing, simply choose Save from the File menu. BBEdit will prompt you to authenticate as a user with administrator privileges. Type a suitable user name and password to save the file.

Saving Compressed Files as bz2 or gzip

BBEdit transparently supports opening, browsing, and saving files compressed in the ‘bz2’ and ‘gzip’ formats. To save a file with gzip compression, simply append a filename extension of “.bz2”, “.gz”, or “.gzip” when creating it (or doing a Save As of an existing document). (For more information on these formats, issue the commands ‘man bz2’ or ‘man gzip’ in the Terminal.)

Saving as Styled Text or HTML

If syntax coloring is active, BBEdit can generate rich text (RTF) or HTML markup which replicates the visual appearance of the current document’s contents. You can save a file containing RTF via the Save as Styled Text command or a file containing HTML via the Save as Styled HTML command (both in the File menu), or copy the current selection to the clipboard as styled text via the Copy as Styled Text command, or as HTML markup via the Copy as Styled HTML command (both in the Edit menu).

Crash Auto-Recovery

IMPORTANT BBEdit automatically saves auto-recovery information for all unsaved open documents at the specified interval. When you relaunch BBEdit after a system or application crash, BBEdit will reopen and restore the contents of any documents for which recovery information is available.

BBEdit's auto-recovery mechanism can help minimize the chance of data loss in the event of unexpected system or application crashes. However, it may not protect against extraordinary events, and it will not protect against hardware failures or any other events that render your disk unreadable. You should always manually save a document after making any significant changes to it, and we strongly recommend that you take appropriate measures to back up your important files and other data.

Opening Existing Documents

There are several ways to open existing documents with BBEdit.

- Double-click any file with a BBEdit document icon.
- If BBEdit is running, choose the Open or Open Recent command from the File menu.
- Select the name of a file in a BBEdit editing window; then use the Open Selection command in the File menu.
- Double-click a file name in a browser's file list (see Chapter 9, "Browsers").
- Drag a file's icon to the Windows palette (see Chapter 6, "Working with Windows").
- Drag a file's icon into the sidebar of any editing window (see Chapter 4, "Window Anatomy).
- Drag a file's icon to the BBEdit icon or to an alias of the icon.
- Select a file in the Finder, and use the Open File command from the BBEdit submenu of the Services menu.

BBEdit can natively open all files which it or the system recognize as plain text files. By default, BBEdit will not attempt to display the contents of files which are not recognizable as text files, such as images or PDF files, but it is still possible to open some such files in a "raw" condition as if they contained only text.

NOTE You can adjust how BBEdit should handle PDF files via its expert settings. (See the "Expert Settings" page of our website for complete details.)

Front Window versus Separate Windows

Since BBEdit supports opening multiple documents into a single text window, you must decide whether the application should work in this manner, or whether it should instead open each document into its own window. (A document may represent either a file which you open for editing, a text document created by the New Document command on the File menu, or any similar item, such as a text document created via the scripting interface.)

By default, BBEdit will open all new documents into a single text window, but you can instead configure it to open each document into its own text window by turning off the option "Open documents into the front window when possible" in the Application settings panel.

Choosing the Encoding for a Document

When you open a document, BBEdit will automatically examine its contents for any indication of the proper encoding, and attempt to handle it appropriately. If BBEdit cannot determine the proper encoding, and you opened the file with the Open command, it uses the encoding specified in the Read As pop-up menu on the Open dialog. Otherwise, it will try to employ the encoding(s) specified in the“If the file’s encoding can’t be guessed, use” settings list in the Text Encodings settings panel.

Note You can choose which encodings appear in the Read As pop-up menu by using the Text Encodings settings panel.

Here are the details of the steps that BBEdit goes through to determine the proper encoding for a file:

1. **If the file is well-formed HTML or XML, BBEdit looks for an “encoding=” or <meta charset=> directive.**
2. **If the file contains an Emacs variable specifying its encoding, BBEdit will use that encoding.**
3. **If you have opened the file with BBEdit before, BBEdit will use the file’s stored encoding info (if any).**
4. **If the file contains a UTF-8 or UTF-16 (Unicode) byte-order mark (BOM), BBEdit opens it as that type of Unicode file.**
5. **If the file has a resource that contains font information (such as a ‘styl’ resource) and that resource specifies a multi-byte font, BBEdit opens the file as a Unicode file.**
6. **If you are opening the file with the Open command, BBEdit uses the encoding specified Read As pop-up menu on the Open dialog.**
7. **If the file contains no other cues to indicate its text encoding, and its contents appear to be valid UTF-8, BBEdit will open it as UTF-8 without recourse to the below settings option.**
8. **Finally, it uses the encoding chosen for the option “If the file’s encoding can’t be guessed, use” from the pop-up menu in the Text Encoding settings panel.**

To change the encoding for a file after opening it, use the Text Encoding popup in the document’s status bar.

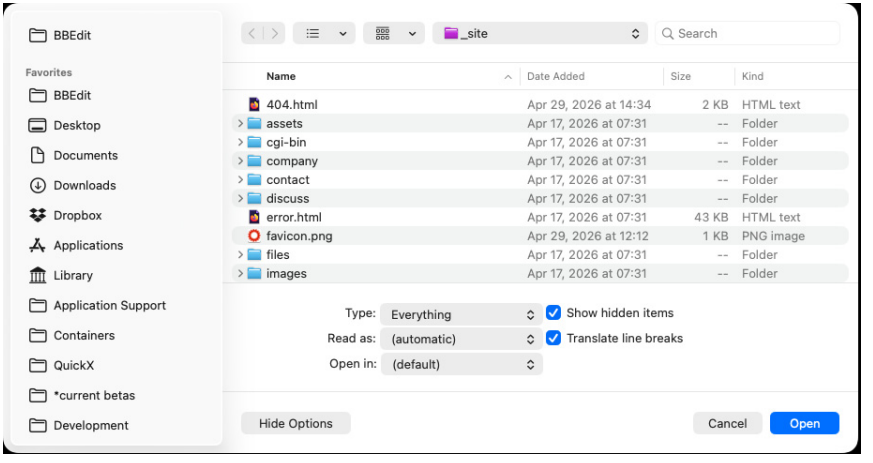
Note If an encoding change results in the conversion of a document’s contents from a single-byte script to a multi-byte script, BBEdit will mark the document as being “dirty” or changed.

Using the Open Command

To open a file with the Open command:

1 **Choose Open from the File menu.**

BBEdit displays the Open dialog box:



2 **Select the file you want to open.**

You can select (or deselect) multiple files by holding down the Command key or the Shift key as you click the files.

3 **Change any desired options (see below).**

4 **Click Open to open the file.**

You can use the options described below when you open a file.

Show Hidden Items

Turning this option on will both cause BBEdit to make any hidden items visible **and** allow BBEdit to open all files present (whether or not they appear to be text files).

File Type Filtering

BBEdit now separates file type filtering from invisibles filtering in the Open panel, via a new "Type:" menu (behind the Options button) which allows you to choose which file types to display. (“Show hidden items” will modify this.)

Read As

When opening a file, you can tell BBEdit what encoding to use by choosing it from this pop-up menu. Usually, BBEdit will correctly auto-detect the encoding, but if it does not, you can try applying the Reopen Encoding command with an appropriate encoding. Chapter 5 includes more information on encodings.

Open In

When opening one or more files, you can use the options on this pop-up menu to override your default document opening settings. These options have the following effect:

- (default): BBEdit will open the selected documents according to your settings settings.
- Front Window: BBEdit will open all of the selected documents into the frontmost text window. If there are no text windows open, or the frontmost text window contains an active sheet, this option will be disabled.
- New Window: BBEdit will open all of the selected documents into a new text window.
- Separate Windows: BBEdit will open each of the selected documents into its own text window.

Translate Line Breaks

When this option is selected, BBEdit translates Windows or Unix line breaks when opening a file. Otherwise, BBEdit leaves the original line breaks untranslated.

Unlike the other options in the Open dialog, the setting of this option is not preserved between uses of the Open command, since in general you will only need to use this operation temporarily, e.g. to read in a particular file.

Reload from Disk

When you choose this command, BBEdit will compare the contents of the current document in memory to those of its file on disk, and reload the document from its file if they differ. This is useful in situations where the file may have changed without BBEdit noticing, which can happen if, e.g. the "Automatically refresh documents" option in the Application settings panel is turned off, or if the file is on a shared disk and has been modified from another workstation.

When the active document has been opened from a remote (FTP/SFTP) server, the Reload from Disk command will become Reload from Server and choosing this command will cause BBEdit to fetch a fresh copy of the file from the server, and then refresh the document's contents.

Opening and Editing Files within Zip Archives

BBEdit transparently opens and displays the contents of any Zip-compressed archives (".zip") both directly and during multi-file search. In addition, you can directly edit the contents of files within such an archive and save changes directly back to those files. (Zip archives must be in the format created by the Finder's "Compress" command, or by applying `ditto -k` from the command line.)

Note If the Zip archive contains only one top-level item, and that item is a folder, BBEdit will "hoist" the rest of that package's contents and not display the top-level item.

Opening Compressed Archives, Tarballs, and Binary plists

BBEdit transparently opens and displays the contents of many types of compressed file archives (".bz2", ".gz", ".gzip, and ".xz" files), as well as tarballs (".tar" files) and binary plists (".plist" files), both directly and during multi-file search.

This is especially useful for viewing and working with system log files and similar automatically-generated files, as well as system and application settings files.

If you make any changes to such a file and save it, BBEdit will automatically re-compress or re-convert the file on save.

Opening Hidden Files

Turn on the “Show Hidden Items” option in the Open dialog to display hidden files (including both files whose invisible attribute has been set, and those whose names begin with a period) or files from a folder which is normally hidden by the system.

Opening Images

You may open image files of any common format (GIF, JPG, PNG, etc.) into BBEdit in order to view their contents.

Launch without reopening files

When you launch BBEdit with the Shift key down, an alert will now appear asking you to confirm whether you want BBEdit to skip reopening of all previously opened documents. (The application would formerly do this without prompting and post an OS notification, which quite often could be missed or suppressed.)

Using the Open from FTP/SFTP Server Command

See “Accessing Remote Servers” on page 66.

Using the Open Selection Command

The Open Selection command lets you quickly invoke the Open File by Name command to search for any file that is referenced in the text of a document. It is particularly useful for opening include files or any document referenced by another file.

To open a file whose name is referenced in the text of a document:

1. **Select the file name within the body of the document.**
2. **Choose Open Selection from the File menu.**

   If a suffix of the form “.x” follows the name, BBEdit will automatically expand the selection to include the suffix.

   BBEdit will display the Open File by Name window, pre-populated with the selected text.
3. **Click Open or type Return in the Open File by Name window.**

BBEdit also understands the Unix-style line number and character offset specifications “:line:offset” that can be appended to a file name, and will honor them when opening a file. If the specified file is already open, this command will simply select the designated location within the file. (These specifications are frequently generated by Unix command line tools.)

For example, selecting the text “main.cp:210” and choosing Open Selection will bring up the Open File by Name window prefilled with that search string, and when you click Open, BBEdit will then open the file “main.cp” and automatically select line 210. Likewise if you apply Open Selection to the text “foo.cp:398:43” and invoke Open File by Name, BBEdit will open the file “foo.cp” and automatically position the insertion point at the specified location.

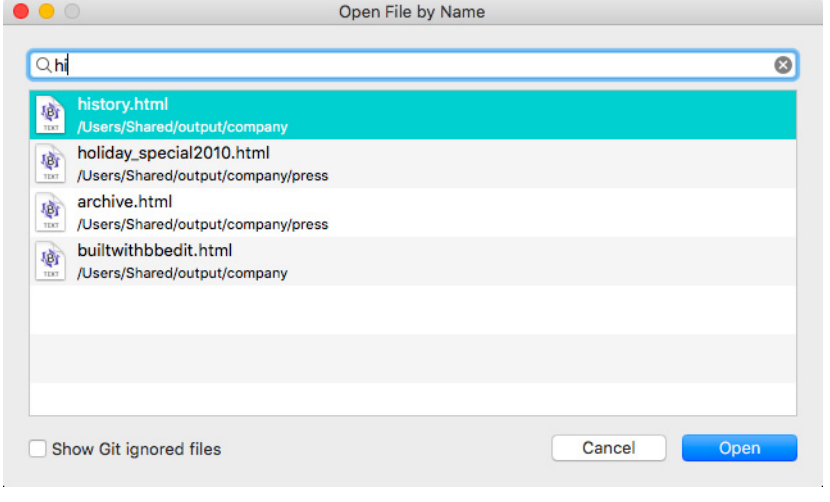
In searching for the requested file, BBEdit will look in the following locations, in order:

- If a project document is open, BBEdit will start its search within the project, looking first at discrete file entries and then (if necessary) searching any folders contained within the project. (If multiple projects are open, BBEdit will use the frontmost project.)
- If the Open Selection is being done from a shell worksheet, BBEdit will search the shell’s current working directory, followed by any subdirectories within it.
- Otherwise, BBEdit will look first in the same folder as the file containing the selected file name, and then in any subfolders within that folder.

In some cases, there may be more than one file with the same name in the various folders BBEdit looks in. Normally, BBEdit opens the first such file it encounters, and then stops.

Using the Open File by Name Commands

If there is no selection, or there is no text display view in the front window, Open Selection becomes Open File by Name. Choosing this command brings up the Open File by Name window.)



Activating the Open File by Name window, or choosing the menu command, will place keyboard focus in the search box and select its contents, so that you can just start typing. (To clear an existing entry, click the “clear” widget at the right-hand edge of the field.)

As you type, BBEdit will search for files matching the current string as well as wildcard matches, and present a list of possible matches in the bottom panel of the window. If the string you enter contains wildcard characters (see below) then BBEdit will treat it as a wildcard pattern. If the string you enter does **not** contain wildcards, BBEdit will instead use it as a basis for casting a wide net.

BBEdit will look for matches in the following locations (in order of settings):

- If a project document is open, BBEdit will start its search within the project, looking first at discrete file entries and then (if necessary) searching any folders contained within the project. (If multiple projects are open, BBEdit will use the frontmost (Z-order) project.)
- If Xcode is running, BBEdit will search all files in the active project (if any), then files in the system framework includes.
- If there is a disk browser open, BBEdit will search within its current root directory.
- If there is a shell worksheet open, BBEdit will search the shell's current working directory, followed by any subdirectories within it.
- Otherwise, BBEdit will look first in the same folder as the file containing the selected file name, and then in any subfolders within that folder.

You can navigate the list of potential matches by using the up and down arrow keys or the mouse pointer, and open any listed file by selecting it and typing Return or Enter, or clicking the Open button.

If BBEdit does not locate any potential matches, you can still search for the file, as before. (The search will skip locations where such a file would have already been found, i.e. the frontmost project.)

If you type a string which appears to be an absolute or a home-relative path (e.g. "/path/to/some/file.txt" or "~/Documents/some/file.txt", BBEdit will cease searching and when you type Return or Enter, or click the "Open" button, BBEdit will attempt to open the file at that path, if it exists.

If you type a string which appears to be a URL, BBEdit will attempt to open it directly, or hand it off to an application that can. (BBEdit supports a number of schemes, including 'file', 'http', 'ftp', and 'sftp'.)

If you type in an unqualified partial path, e.g. "sys/errno.h", BBEdit will check the path components and only display files whose immediate ancestry matches what you entered. In this example, it would list "/usr/include/sys/errno.h" but not "/usr/include/errno.h".

If you type in an absolute path, or a home-directory-relative path, e.g. "/usr/include/errno.h" or "~/.bash_profile", BBEdit will show the file if it exists at that location.

BBEdit also maintains a search history in the Open File by Name window: when you open a matched item, BBEdit will store the string you used, and the search history (magnifying glass) popup lists these recently used strings.

You may use the following wildcards as part of a search string:

Wildcard	Meaning
?	Any single character
*	Any number of characters
#	Any numeric character
\	Escapes one of the above; for example, \? enters a question mark. To enter a literal backslash, use \\.

Using the Related Files Command

You can use this command or its default key equivalent of Option-Command-Up arrow (configurable via the Menus & Shortcuts settings panel) to switch between related files (from source to header and vice versa). In addition to intrinsic counterparts (e.g. C/C++ style header/source mapping, or HTML to CSS mapping) and includes detected by the active language module's function scanner (if any), you can explicitly define a relationship by setting a value for the (BBEdit-specific) “x-counterpart” variable in a file’s Emacs variables. For example, if your file contains the following as part of its variable block:

```
-*- x-counterpart: ExampleStrings.R; -*-
```

when you type Option-Command-Up arrow, BBEdit will look for the file “ExampleStrings.R”.

Using the Open Recent Command

The Open Recent submenu contains a list of files, folders, and other relevant items (such as projects) which you have opened recently. To open any of these items, just choose it from the Open Recent submenu, or you may directly select the “Open Recent” command to bring up the “Search Recent Items” panel and use that to refine your search and select any desired item -- or items).

(To set the number of items displayed in the Open Recent list, use the “Remember the [N] most recently used items” option on the Application settings panel.)

Using the Reopen using Encoding Command

The Reopen using Encoding submenu contains a list of all available text encodings. To reopen the current text document and have its contents interpreted using a different encoding, choose the desired encoding from the Reopen using Encoding submenu. This command will only be available if the current document is unmodified.

Quitting BBEdit

By default, whenever you quit BBEdit or BBEdit automatically quits because of a system shutdown, restart, or user account logout, BBEdit will attempt to restore as much of its state as possible when starting back up. Thus, you may not be prompted to save new or unsaved documents, since BBEdit will automatically preserve the contents of all open documents before it exits.

You can control whether BBEdit should preserve and restore unsaved changes via the "Restore unsaved changes" option in the Application settings panel existing documents. If this option is on, BBEdit will automatically preserve unsaved changes. If this option is off, BBEdit will instead prompt you to save each document which has unsaved changes.

An International Text Primer

All Macs have extensive support for working with international text, including Unicode. If you have enabled additional text input methods in the International section of the System Settings, you will see the Input menu on the right-hand side of the menu bar. This menu allows you to change keyboard layouts or script systems as you work.

Note Actually, even if you have never used a non-Roman script system before, you may still have used this menu, if you have ever chosen an alternate keyboard layout such as Dvorak, or a keyboard layout for a Roman language such as French. However, since the Roman script is suitable for several languages, choosing one of these keyboard layouts still leaves you in the Roman script.

International Text in BBEdit

As a text editor, BBEdit supports only one font per document window, though it can display all available characters in the active font, including Unicode characters.

BBEdit supports editing in almost any language which uses left-to-right text input methods. To start entering text in any supported language, choose a suitable input method from the Input menu. The icon for that method will appear in the menu bar in place of either the American flag (for the U.S. English layout) or the icon for your usual Roman keyboard layout.

If you have turned off the "Try to match keyboard with text" option in the Options dialog of the International section of the System Settings), you may also need to select a suitable display font via the Font panel. (We recommend leaving this option on, so that BBEdit can automatically switch to the correct input method when you change document windows.)

You can use international text throughout BBEdit—for example, in the Find window, in the HTML Tools, and everywhere else you would use Roman text. Likewise, BBEdit will provide the necessary style information so that if you copy and paste, or drag and drop, international text into another application, that application will have enough information to handle the text correctly (assuming it is capable of doing so).

BBEdit remembers the encoding used in a document when you save it, so the next time you open it, you will not need to choose the font. However, you may not be able to read files which do not have this stored information, for instance, files downloaded from the Internet, until you choose an appropriate encoding for them.

When performing a search, BBEdit respects any available information about each file's encoding. If a file does not contain any information about its encoding, BBEdit will use the default encoding set in the Text Encodings Settings panel.

Unicode

Unicode is an international standard for character encoding, which includes an extensive selection of characters from Roman, Cyrillic, Asian, Middle Eastern, and various other scripts. For more background information or complete details on Unicode, the Unicode Consortium website is the best place to look.

```
https://unicode.org/main.html
```

BBEdit fully supports and makes extensive use of Unicode, in addition to all other OS-supported text encodings. In particular, BBEdit internally represents all documents as Unicode, regardless of their on-disk encoding.

Saving Unicode Files

BBEdit lets you save documents that use character set encodings other than Mac Roman, even multi-byte character sets. When saving a file, you can choose to save text composed in any script with any encoding. In addition to the standard character set encodings, BBEdit also lets you save the files in a variety of plain Unicode files:

- **Unicode (UTF-8)**: UTF-8 **without** a byte-order mark
- **Unicode (UTF-8, with BOM)**: UTF-8 **with** a byte-order mark (BOM)
- UTF-16 Little-Endian
- UTF-16 Little-Endian, no BOM
- UTF-16
- UTF-16, no BOM

IMPORTANT The naming convention BBEdit follows for UTF-8 documents has changed from that used by versions before 9.5: the encoding name "Unicode (UTF-8)" refers to files **without** a byte-order mark (BOM), while the specific name "Unicode (UTF-8, with BOM)" refers to files which have a BOM.

Here are details about what each of the above options means:

- UTF-8: UTF-8 encoding is a more compact variant of Unicode that uses 8-bit tokens where possible to encode frequently used sequences from the file. (This format makes it easier to view and edit content in non-Unicode-aware editors.)
- UTF-16: UTF-16 encoding always uses 16-bit tokens.
- BOM: When saving Unicode files, you may include a byte-order mark (BOM) so that the reading application knows what byte order the file's data is in. However, since many applications do not correctly handle files which contain BOMs, you may wish to use an encoding variant without a BOM for maximum compatibility. (For purposes of recognition when you use this option, the UTF-16 BOM is FEFF, and the UTF-8 BOM is EFBBBF.)

- Little-Endian: Since UTF-16 uses two bytes to represent each character, this leaves the question of which of the two bytes comes first—whether it is “little-endian” or “big-endian.” By default, BBEdit writes UTF-16 big-endian (the standard). By choosing one of the “Little-Endian” (or “byte-swapped”) encodings, you can write little-endian files instead, which some Windows software requires.

Opening Unicode Files

When opening files, BBEdit will ordinarily determine the format of a file based on its file type and content, and automatically process Macintosh text, Unicode, and UTF-8. However, some files are structured such that BBEdit is unable to correctly determine their format based on their type or contents. The cases that we know of are:

- UTF-8 files which lack a byte-order mark and do not contain any encoding specification **or** any extended characters. (If a UTF-8 has a byte-order mark, BBEdit will correctly interpret its contents as UTF-8.)
- Byte-swapped Unicode files which were written without a byte-order mark (usually by broken Windows software);
- Unicode files which lack a byte-order mark.

If you know that a file you are trying to open is in Unicode but it displays as gibberish on your screen, close its window without saving. Then try reopening the file, using the Open As pop-up menu in the Open dialog to specify whether to treat the file as Unicode, byte-swapped (little-endian) Unicode, or UTF-8.

If you attempt to open a document which cannot be represented by either its declared encoding or any recognizable encoding, BBEdit will present an alert to warn you. Also, if BBEdit encounters such a file during a multi-file search, it will log a warning.

Accessing Remote Servers

BBEdit can open files directly from, and save them to, any available FTP or FTP-S server. It can also open and save files directly via SFTP (**S**SH **F**ile **T**ransfer **P**rotocol). In order to access a server via SFTP, that server must be running a compatible version of sshd. (A great many machines, including Mac systems for which “Remote Login” is turned on in the Sharing panel of System Settings, satisfy these criteria.)

Aside from choosing the SFTP checkbox in the Open from…/Save to… dialogs, or the FTP/SFTP Browser, opening and saving files via SFTP works just like it does when using ordinary FTP. A file opened via SFTP will appear in the Open Recent submenu with an “sftp:” URL, and you can send a “get URL” event to BBEdit with an “sftp” URL as well.

Opening Files from Remote Servers

To directly open files from any FTP, FTP-S,or SFTP server, choose Open from FTP/SFTP Server from the File menu. BBEdit will open an new FTP/SFTP Browser window. Like other browser windows, FTP/SFTP browsers will remain open until you close them, and once connected, they will maintain a persistent connection to the server for as long as they remain open

Configuring Connection Settings

Enter the server's name in the "Server:" field, or choose a local server advertised by Bonjour by clicking the Bonjour popup menu to the right of the "Server:" field, specify your user name and password in the appropriate fields, enable the "SFTP" option if appropriate, and optionally choose an identity file.

In case the server or service you are using requires this, it's simpler than modifying your SSN config file ('~/.ssh/config') although that of course remains an option as well.

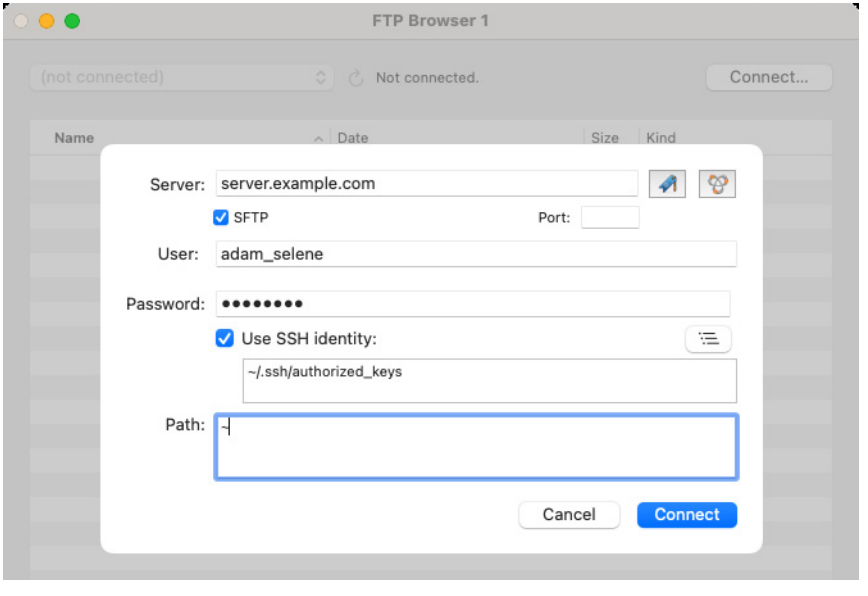
If you enable "Use SSH Identity" and choose a file, BBEdit will display a warning indicator next to the file's path if the file's permissions are not 0644' i.e. readable by owner only. (This is necessary because SSH will refuse to use the private key file if its permissions are not 0644.) Should you subsequently move or delete the identity file, BBEdit will display a warning indicator next to the file's path if it's not found.

The connection panel for the Open from FTP/SFTP Server command provides a popup menu for choosing the security model, as follows:

```
FTP: FTP protocol, no transport security

SFTP: SSH file transfer protocol, SSH transport security

FTPS (Explicit): FTP protocol, TLS transport security
  (negotiation requires server "AUTH TLS" support and
  negotiation)

FTPS (Implicit): FTP protocol, TLS transport security
  (connection opened immediately; requires server support)
```

Connecting to the Server

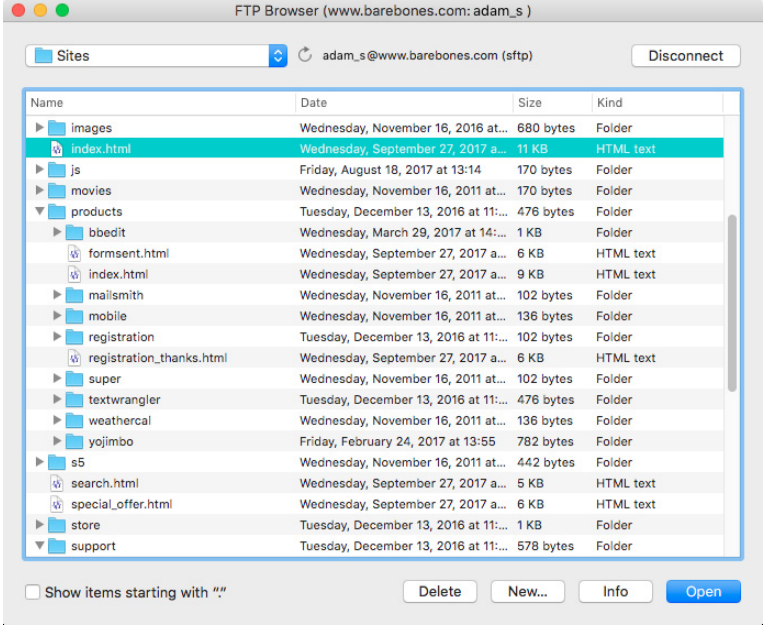
Once you have verified the connection options are correct, you may click the Connect button or press the Return or Enter keys to connect to the server.



Alternatively, you can choose a bookmark from the Bookmarks pop-up menu to fill in stored info for the server, user name, password, and connection options. You can choose Add Bookmark... from the directory popup in an FTP/SFTP Browser window to create a bookmark for the current server and directory, or create arbitrary bookmarks via the Bookmarks panel of the Setup window.

You can also add, delete, or modify bookmarks via the Bookmarks panel of the Setup window:

Once you have connected to the server, you can open files by double-clicking them, or selecting them and clicking the Open button. You can double-click a folder to change directories. If you hold down the Option key when opening a folder, it will open in a new FTP/SFTP Browser window. You can select a range of files and directories by Shift-clicking, and you can select (and deselect) multiple items one at a time by Command-clicking them.



To refresh the directory listing, click the circular arrow icon (located in the center of the toolbar). The checkbox below the listing labeled Show Files Starting with "." tells BBEdit whether to display hidden or admin files in the chosen directory, such as .login, .forward, and .signature. (Starting a file name with a period is a convention used by macOS and other Unix-like systems to make that file invisible in most directory listings.)

Once you have selected a file and opened it, BBEdit displays the file in a text editing window. The navigation bar displays the URL of the file on the server, not the pathname of the file on your hard drive as it does for local files.

NEW You can drag items from an FTP browser window to other applications. BBEdit will include a URL in the drag event for each selected item in a form that applications which accept URLs may be able to use.

You can use the Info button to examine the size, modification date, and if applicable, file system permissions of the selected file. You can edit the file's name and click the Rename button to rename the file on the server; you can also make changes to the permissions and click the Set button to change them. (Take care not to set the permissions such that the file becomes inaccessible to you!)

You can directly create a new file (or folder) on the server by clicking the New button, or remove files from the server by selecting them and pressing the Delete button.

Specifying Alternate Ports

BBEdit allows you to open an FTP or SFTP connection to any alternate port (i.e. any port other than the default). To specify an alternate port, just type the desired port number into the "Port" field of the connection sheet.

Specifying a Login Path

You can optionally specify the server directory that BBEdit logs into by typing or pasting a suitable path into the "Path" field of the connection sheet.

Storing Passwords

As long as your account's login keychain is unlocked, BBEdit will use it to store the password for each server that you access, and to automatically fill in the corresponding password whenever you enter a server and user name pair for which there is a keychain entry. If your keychain is locked, you will need to retype your password every time you use the FTP browser.

Using SSH Key Files

In order to connect to an SFTP server which requires SSH keys (or certificates) rather than passwords, you must first create an appropriate entry for that server in your local account's .ssh/config. You may then type the server name, or shortcut name, into the Server field of the FTP/SFTP Browser and connect without entering a password.

For example, here is an entry you can insert to specify the key for a given server and account:

```
# for a specific user account "myusername@server.example.com"

Host server.example.com

    User myusername

    IdentityFile path/to/my/public/key

   # replace the latter with the actual path to your public key
  file
```

Transfer Formats

When you open a file from an FTP or SFTP server, BBEdit downloads the file "raw" (in binary mode) and then performs a standard line ending conversion upon opening the (local temp) file.

Working with URL Clippings

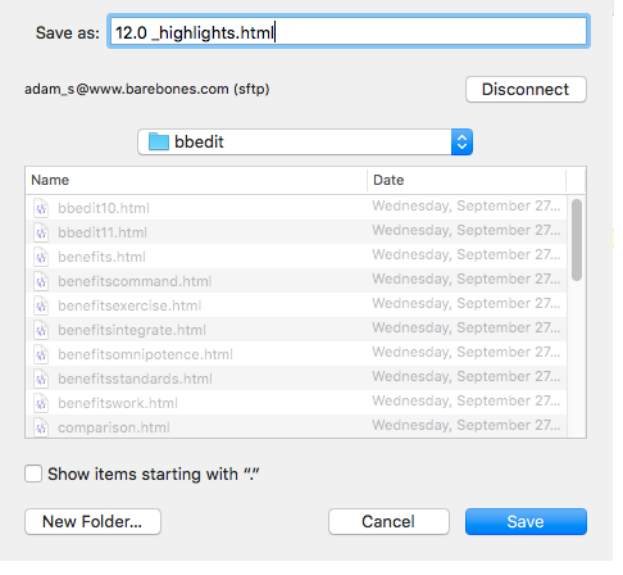
BBEdit also supports FTP and SFTP URL clippings. You can make a clipping of the FTP or SFTP URL for a file, add the clipping to a project, and double-click it, and BBEdit will open the specified file for editing. If the clipping contains the URL for a directory, BBEdit will open a new FTP/SFTP Browser at that location. Alternatively, you can double-click an FTP or SFTP clipping in a disk browser, or drop one on BBEdit's icon in the Finder, with the same results as just described.

Dragging the window proxy icon from the editing window of a file open from an FTP or SFTP server will create a clipping containing that file's URL.

Saving Files to FTP/SFTP Servers

After you have edited a file opened from an FTP or SFTP server, pressing Command-S or choosing Save from the File menu saves the new version back to the server. If you want to save the file in a different directory or under another name, choose Save to FTP/SFTP Server to open the Save to FTP/SFTP Server sheet (shown below).

This sheet works much like the standard Save sheet for a local file, with the addition of fields and controls similar to those in the FTP/SFTP browser allowing you to select or specify connection info, and to navigate and obtain info about other files.



Note When you save a file to an FTP server using either Save or Save to FTP/SFTP Server, and that file has Unix (LF) or Windows (CR+LF) line endings, BBEdit uploads the file in binary mode, preserving its line endings exactly as they are on your local machine. However, if the file has Legacy Mac (CR) line endings, it is uploaded in text mode so that the server can convert the line endings as appropriate.

In contrast, since SFTP has no concept of transfer modes, there is no line ending conversion during either download or upload.

Finally, you can use Save a Copy to FTP/SFTP Server to upload a copy of your current file to an FTP server while keeping your local file open. This is especially useful when you maintain website content on your local hard drive and only need to upload changes made in one or two files to the server.

"Safe" Save Behavior for Remote Files

Starting in BBEdit 15.0, file uploads made by saving a document via the built-in FTP/SFTP support are "safe" by default: the file is uploaded to a temporary file in the destination directory. Once that is successful, the existing copy (if any) is deleted, and the temporary upload file is renamed. This adds a measure of protection against various things that can go wrong during file uploads, though there is still no substitute for keeping **both** your local files **and** the files on your remote server backed up.

Note File uploads during site deployment are not made "safe" in this fashion; these work the same as before.

Using BBEdit with the Command Line

BBEdit's Command Line Tools

You can use the "bbedit" command line tool to open files into BBEdit via the Unix command line. (In case you have not already done so, you can choose "Install Command Line Tools" from the BBEdit (application) menu at any time to install the current version of the command line tools.)

To open a file in BBEdit from the command line, type

```
bbedit filename
```

where *filename* is the name of the file to be opened. To launch BBEdit without opening a file (or activate it, if it is already running), type

```
bbedit -l
```

In addition to files, you can also specify FTP or SFTP URLs to files or directories, to have BBEdit open the specified files, or an FTP/SFTP Browser for each directory. You will be prompted to enter passwords if necessary.

You can also pipe STDIN to the "bbedit" tool, and it will open in a new untitled window in BBEdit: for example,

```
ls -la | bbedit
```

If you just type

```
bbedit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to BBEdit.

The complete command line syntax for the "bbedit" tool is

```
bbedit [ -<short-form switches> --<long_form_switches> ]
 [ -e <encoding_name> ] [ -t <string> ]  [ +<n> ]
 [ file (or) <S/FTP URL> ... ]
```

See the "bbedit" tool's man page ('man bbedit') for a complete description of the available switches and options.

The 'x-bbedit' URL Scheme

BBEdit now supports using the 'x-bbedit' URL scheme for opening files, and optionally specifying destination line and column numbers. The syntax for this scheme is:

```
x-bbedit://open?url=file:///path/to/some/file
```

where '/path/to/some/file' is the actual path to the file you want BBEdit to open.

You can also specify a specific line to select when opening the file; for example:

```
x-bbedit://open?url=file:///path/to/some/file&line=5
```

or if you also include add a column position, then BBEdit will place the insertion point before the indicated character on the previously-specified line:

```
x-bbedit://open?url=file:///path/to/some/file&line=5&column=42
```

Try clicking on this URL, and you'll see that it opens one of your local files in BBEdit, even though you're viewing it in a browser window:

```
x-bbedit://open?url=file:///etc/hosts
```

An 'x-bbedit' URL is guaranteed to launch BBEdit when opened from another application, unlike 'file' or 'editor' URLs, which are at the mercy of the OS to decide which application should open any given file.

Using Projects

If you frequently work with many related files, you may want to create a project for them. A project is a special kind of BBEdit file that contains references to other files and folders, including aliases and URL clippings.

You can use a project to gather and directly edit related files, as well as to serve as an organizational or navigational aid. For example, you can perform multi-file searches on the contents of projects, or process a project's contents with a text factory.

About Project Documents

All project documents which have been saved to disk have a top-level item in the Project pane of the sidebar which reflects the project itself; all other items in the project are subordinate.

Clicking on the top-level item will open a pane which displays configuration and settings for the project itself.

You can use the "Workspace" settings to specify which directory represents the root of the workspace; that is, the directory containing all of the files which are functionally important to the project.

This concept is (intentionally) vague, since the notion of a workspace's functional root may vary depending on what the project is. For example, a software project for macOS might use the directory containing the Xcode project document; or a Node.js development project root would be determined by the location of its `package.json` file.

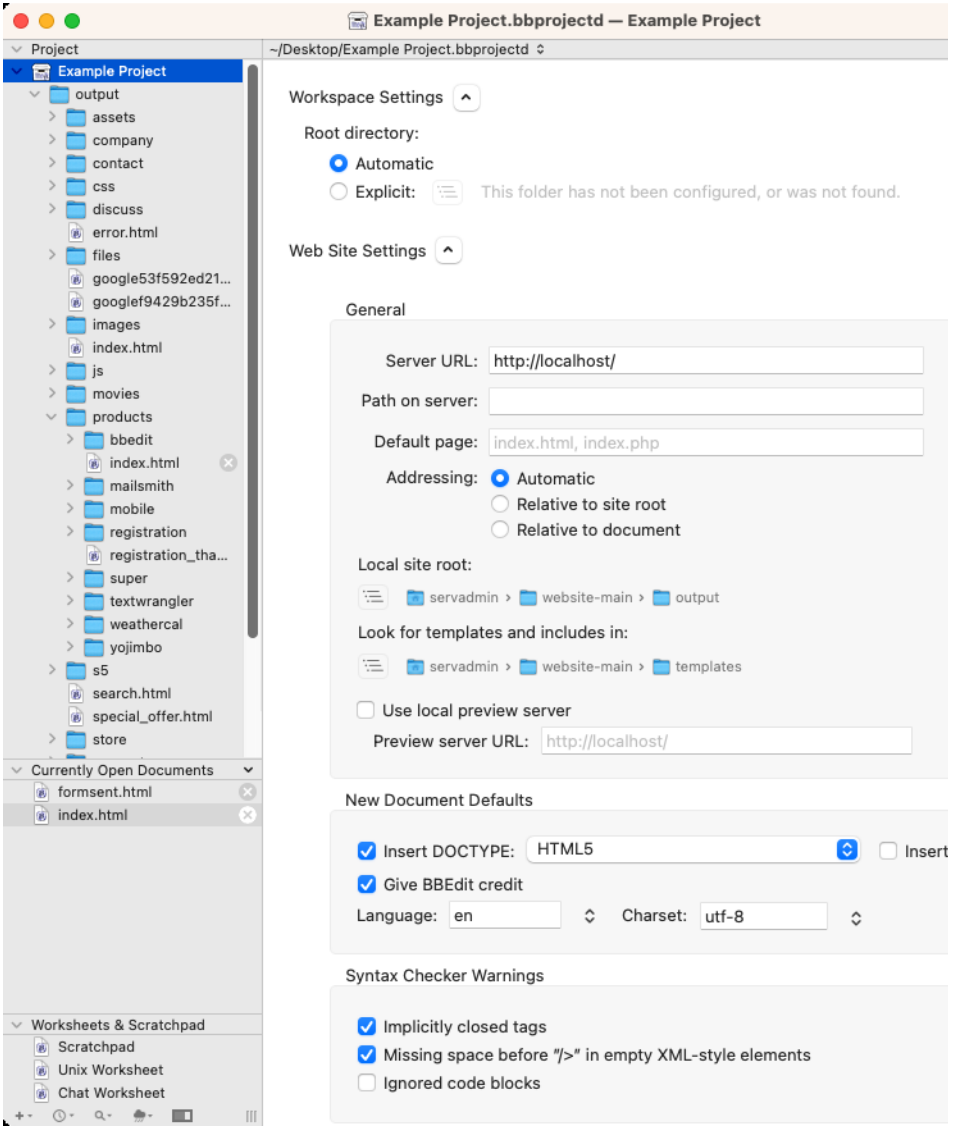
If you set a project's workspace root to "Automatic", BBEdit will attempt to guess the workspace root based on various rules (which are intentionally undocumented because they are subject to change). In the simplest case, if the project has only one item at the top level, and it's a directory, then BBEdit will assume that directory to be the project's root. If the project has multiple top-level directories, then their common ancestor (if any) is the workspace root.

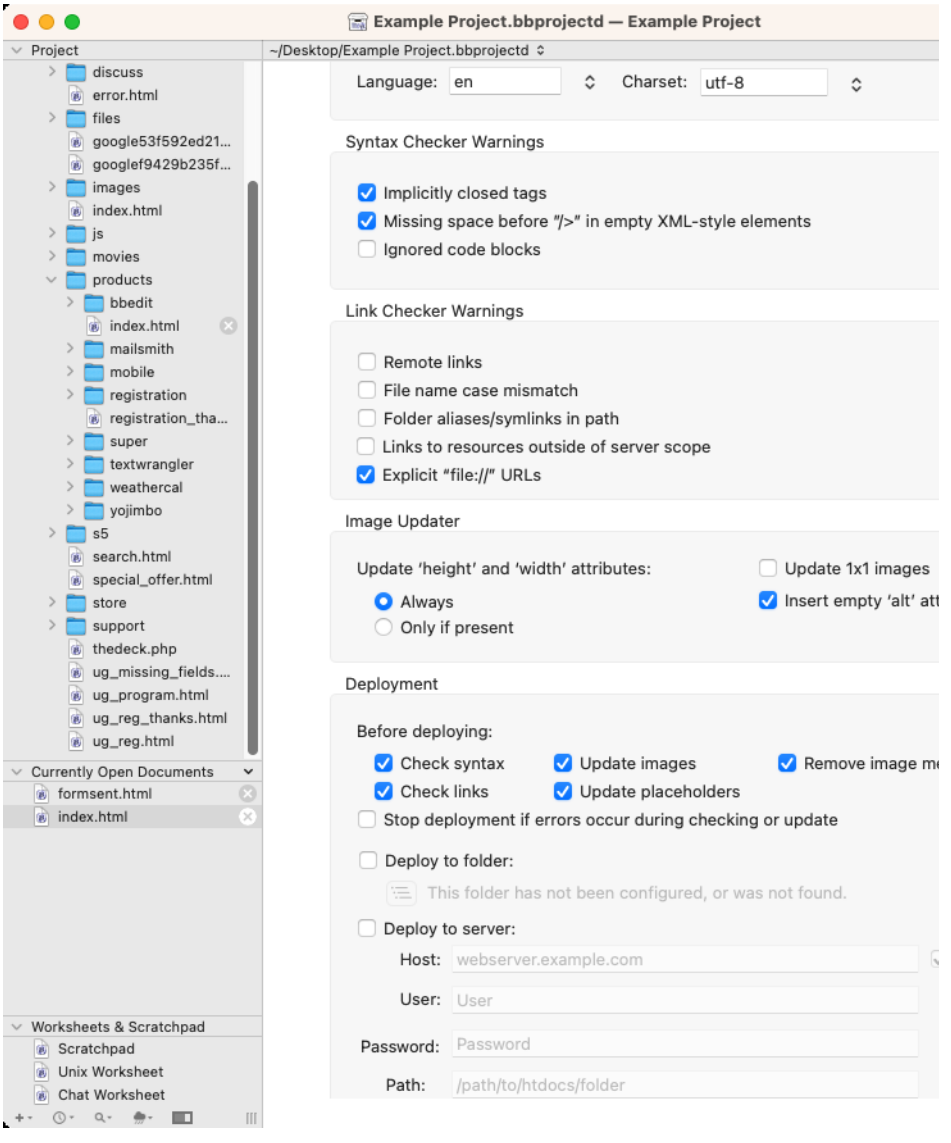
For greater flexibility, or if the "Automatic" setting does not select the appropriate workspace root, you can select an explicit root directory. This does **not** have to be a directory in the project; it can be anywhere.

The workspace root setting (whether automatic or explicit) is also relevant to BBEdit's Git support, and to its Language Protocol Server (LSP) support.

If the workspace root contains a Git working copy, BBEdit will use the workspace root as the working directory for commands on the Git menu. Otherwise, BBEdit will attempt to locate the nearest ancestor which contains a Git working copy, and use that instead.

BBEdit will also relay the project's workspace root (along with the workspace roots of any other open projects) to any running language servers, as long as they advertise support for workspaces. (Not all of them do.) In language servers that support it, providing the project's workspace will improve the completeness and accuracy of code completion and other features which rely on LSP-based code indexing.


Example Project.bbprojectd — Example Project
Project
~/Desktop/Example Project.bbprojectd
Example Project
output
assets
company
contact
css
discuss
error.html
files
google53f592ed21...
googlef9429b235f...
images
index.html
js
movies
products
bbedit
index.html
mailsmith
mobile
registration
registration_tha...
super
textwrangler
weathercal
yojimbo
s5
search.html
special_offer.html
store
Currently Open Documents
formsent.html
index.html
Worksheets & Scratchpad
Scratchpad
Unix Worksheet
Chat Worksheet
Workspace Settings
Root directory:
Automatic
Explicit:
This folder has not been configured, or was not found.
Web Site Settings
General
Server URL:
http://localhost/
Path on server:
Default page:
index.html, index.php
Addressing:
Automatic
Relative to site root
Relative to document
Local site root:
servadmin › website-main › output
Look for templates and includes in:
servadmin › website-main › templates
Use local preview server
Preview server URL:
http://localhost/
New Document Defaults
Insert DOCTYPE:
HTML5
Insert
Give BBEdit credit
Language:
en
Charset:
utf-8
Syntax Checker Warnings
Implicitly closed tags
Missing space before "/>" in empty XML-style elements
Ignored code blocks


Example Project.bbprojectd — Example Project
Project
~/Desktop/Example Project.bbprojectd
discuss
error.html
files
google53f592ed21...
googlef9429b235f...
images
index.html
js
movies
products
bbedit
index.html
mailsmith
mobile
registration
registration_tha...
super
textwrangler
weathercal
yojimbo
s5
search.html
special_offer.html
store
support
thedeck.php
ug_missing_fields....
ug_program.html
ug_reg_thanks.html
ug_reg.html
Currently Open Documents
formsent.html
index.html
Worksheets & Scratchpad
Scratchpad
Unix Worksheet
Chat Worksheet
Language: en
Charset: utf-8
Syntax Checker Warnings
Implicitly closed tags
Missing space before "/>" in empty XML-style elements
Ignored code blocks
Link Checker Warnings
Remote links
File name case mismatch
Folder aliases/symlinks in path
Links to resources outside of server scope
Explicit "file://" URLs
Image Updater
Update 'height' and 'width' attributes:
Always
Only if present
Update 1x1 images
Insert empty 'alt' att
Deployment
Before deploying:
Check syntax
Update images
Check links
Update placeholders
Stop deployment if errors occur during checking or update
Deploy to folder:
This folder has not been configured, or was not found.
Deploy to server:
Host: webserver.example.com
User: User
Password: Password
Path: /path/to/htdocs/folder

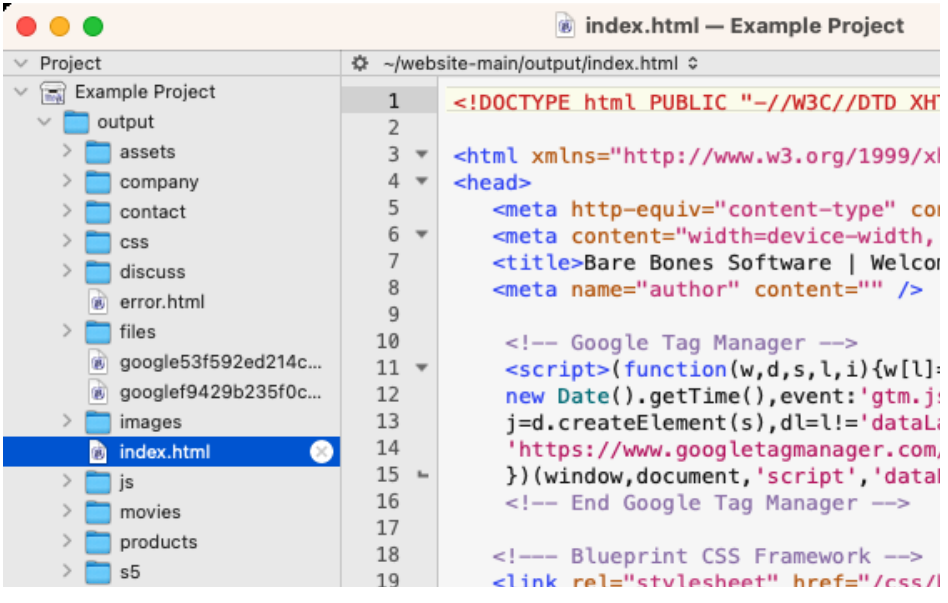
Creating a Project

To create a new project, choose the Project... command from the New submenu in the File menu. You will need to decide where to place the project file on disk; thereafter, the project document will autosave as necessary.

Alternatively, you can create a project from the frontmost editing window by choosing the Save Project command in the File menu.

To add files or folders to a project, drag them from the Finder into the project window, or click the Add button. When you click Add, BBEdit presents the Open dialog in which you can choose one or more files and folders to add. The files and folders you select will be added directly to the project's file list, and you can drag them to reorganize their positions. You can also add a file by dragging its icon from a text window's navigation bar or sidebar to the group window, or by dragging a file entry from any disk browser or results browser.

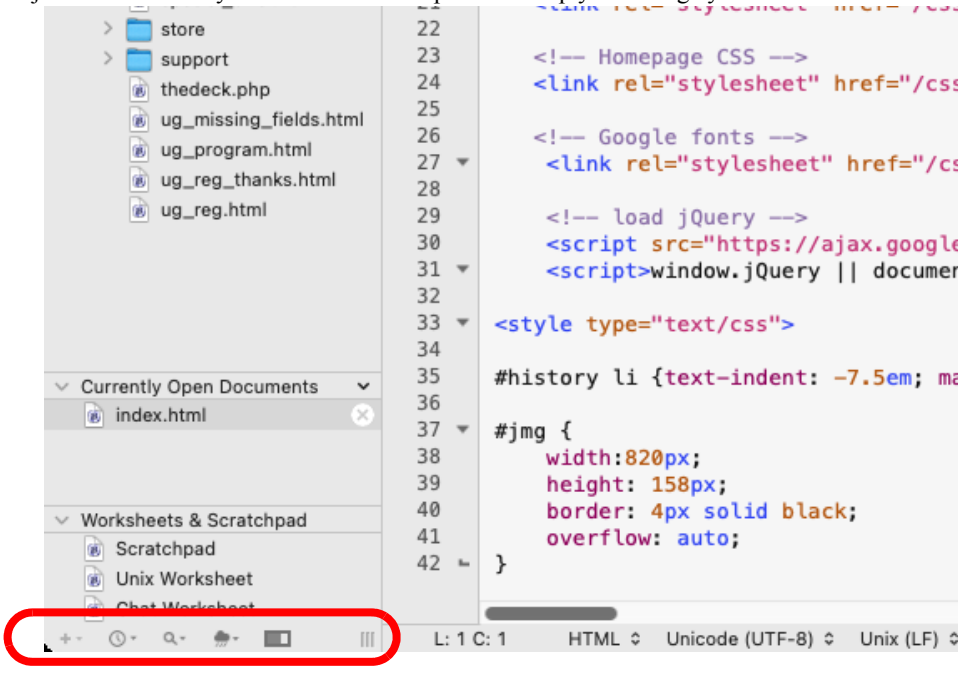
You can display the contents of a folder by clicking the adjacent disclosure control. Each folder's listing is a "live" representation of its contents on disk.



In addition to file and folder references, BBEdit supports URLs in projects. You can drag a URL (text or clipping file) to a project window, and the URL will be saved in the group. If you subsequently open the item, BBEdit will either open that URL directly if it is an FTP or SFTP URL, or hand the URL off to a system-designated helper.

Project Commands

Projects offer a variety of commands and options to help you manage your files.

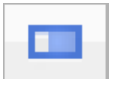


The buttons and popup menus at the bottom of the sidebar are commands that give you quick access to project functions. You can use these commands to add new items, create new files and folders, open existing files and folders, reveal them in the Finder or navigate to them in the Terminal, limit the kinds of files which appear in the project list, and navigate through your disks and folders.

You can resize the Currently Open Documents pane by dragging its top boundary upward or downward, or collapse either the Currently Open Documents pane or the Project pane by double-clicking its header bar. (If you collapse the Project pane, the Currently Open Documents pane will expand upward to fill the available space.)

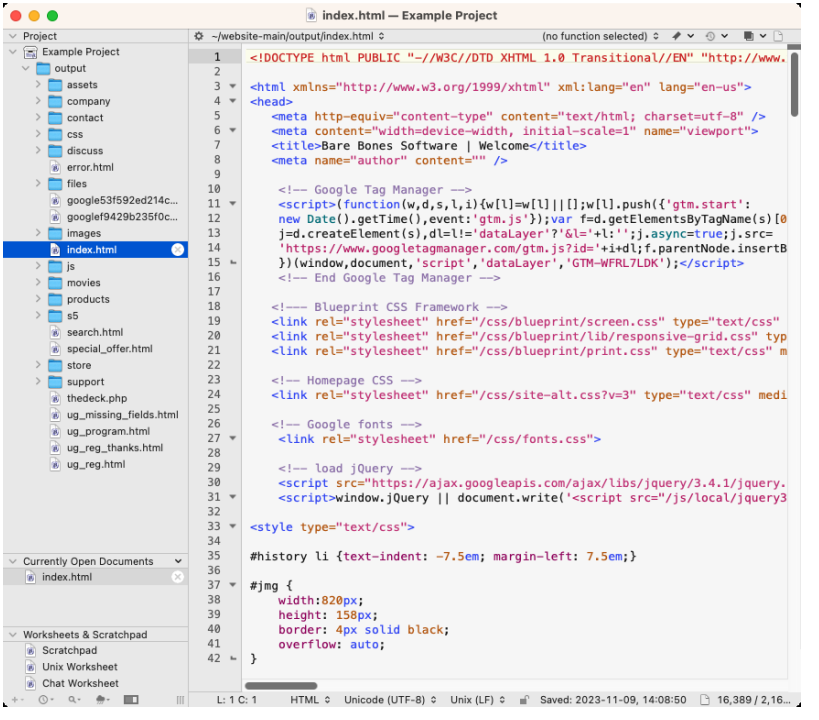
Note The "Recent Documents" section in the sidebar is gone; in its place, you can select and reopen files via the Recent (clock) popup in the action bar at the bottom of the sidebar.

The following table explains each button.

Icon	Meaning
	The Add (plus sign) popup menu allows you to add items to the project's file list, or immediately create a new text file or folder.(Note: The "Add Items" dialog contains a "Show hidden items" option to allow you to directly see and select invisible items.)
	The Recent (clock) popup menu at the bottom of the sidebar allow you to select and reopen any recently-open files.
	The Filter (magnifying glass) popup menu allows you to apply options to control what items BBEdit should display in the project's file list.
	The Site (cloud) popup menu allows you to assign website configuration settings to the current project via the Project Settings sheet, apply website-related commands (e.g. Check Site Syntax), and deploy the contents of the designated site folder to a remote server via FTP or SFTP.
	Click the Toggle Editor button to collapse or expand the project's text editing pane. (This button has the same effect as choosing the View/Hide Editor command in the View menu.)
	Drag the "grip strip" handle to expand or reduce the width of the sidebar. (This has the same effect as clicking and dragging on the separator line between the sidebar and editing panes.)

Using projects

To open a text file within a project, just click on the file and BBEdit will display it in the editing pane. If you click on an FTP or SFTP URL clipping, BBEdit will open the remote file (or open an FTP Browser if the clipping points to a directory). Otherwise, BBEdit tells the Finder to open the file.



If you added nested folders, they appear in the project with disclosure triangles, as in a Finder list view. Click any folder's disclosure triangle to reveal the files and folders inside that folder.

You can use the project browser pane or the document list popup in the navigation bar to select any open file in the project.

You can use a project as the basis of a multi-file search. See Chapter 7, "Searching," for more information.

Creating Files and Folders within a Project

When you choose either the New Folder or New Text Document command (via the Action menu or the contextual menu), BBEdit will display a standard Save panel in which you can specify the location of the item you wish to create. BBEdit will then determine where to place the item in the file list based on its location and/or the starting point of the command: within the initially selected collection (if any), or at the top level of the project, or adjacent to another item already in the project.

Removing Files from a Project

To remove a directly added file or folder from a project, Control-click or right-click on the desired item and choose Remove from the contextual menu.

You cannot directly remove files and folders which are part of a “live” folder listing from a project. However, you can move the original item(s) to the Finder trash by selecting them and choosing Move to Trash from the contextual menu.

Contextual Menu Commands

If you select one or more items in the file list and bring up the contextual menu, BBEdit will offer commands to open the selection, copy the items’ paths (in the same fashion available via the “Copy Path” submenu), or apply selected Git commands (for items contained within a suitable working copy).

Script Access to Project Contents

You can access the items contained in a project document via AppleScript. Note that BBEdit enforces a strict containment hierarchy for such items. For instance, if you ask for “every item of project document 1”, BBEdit will return a list of every item that’s present at the top level of the project. You can recurse to explore the items contained within folders or collections.

Project-Specific Settings

BBEdit now supports applying custom color schemes to projects, instaprojects, and notebooks for project- (and notebook-) windows. Settings applied here will override the global settings (including, subject to setting, any language-specific overrides).

For project documents that have been saved to disk, the color scheme settings are in the project settings visible when clicking on the first item in the sidebar (corresponding to the project itself). For notebooks and instaprojects (created by opening a folder), settings are available via the small gear in the action bar at the bottom of the sidebar.

Using Stationery

BBEdit supports creating new files based on stationery documents. A stationery document serves as a template that, when opened, results in a new, untitled document containing all the text within the stationery document. In other words, you do not edit the stationery document itself but instead use its contents as the starting point for a new document.

To create a stationery document, you need only save the current file into the “Stationery” subfolder of BBEdit’s “Application Support” folder.

You can create new documents from a stationery document by choosing the New With Stationery command from the File menu, and selecting the desired stationery document.

You can also assign a keyboard shortcut directly to any available stationery document in the Menus & Shortcuts settings panel (in the File -> New with Stationery section).

Manually Sorting the Stationery Submenu

By default, items in the Stationery submenu are displayed in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their names. For example, “00)Web template” would sort before “01)HTML Template”. For such files, the first three characters are not displayed in BBEdit. You can also insert a divider by including an empty folder whose name ends with the string “-***”. (The folder can be named anything, so it sorts where you want it.)

Hex Dump for Files and Documents

Choose the “Hex Dump File” command to generate a hex dump representation from a file that you choose. Choose “Hex Dump Front Document” to generate a hex dump representation of the frontmost document as it exists in memory. (The “Hex Dump Front Document” command is now also available for image files, though the “Encoding” option will be disabled since it is not applicable.)

Please bear in mind that the outcome of performing the Hex Dump command against a disk file may differ from the result obtained by using it against an open document, since when a document is open in memory line-break translation and character set encoding conversions may have taken place, even **without** any explicit edits having been made.

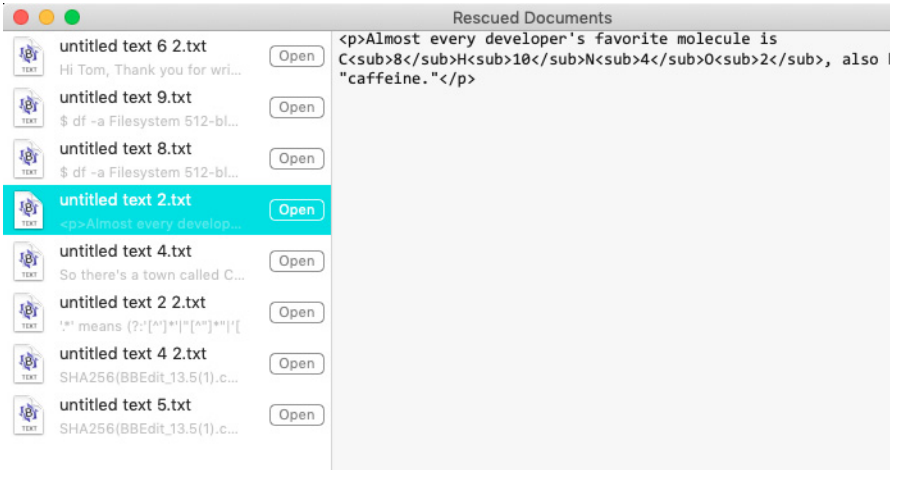
Making Backups

BBEdit can automatically make a backup copy of each document you edit before saving it. To enable this feature, turn on the “Make backup before saving” option in the Text Files settings panel. For complete details on how this feature works, and optional behaviors, please see “Make backup before saving” on page 277.

Rescuing Untitled Documents

Have you ever made a new document, typed some important info into it, and then later on, when you go to close that document (either alone or as part of closing a bunch of documents), you clicked the “Don’t Save” button? And then, an *ohnosecond* later, you realize you’ve made a terrible mistake?

BBEdit has a new feature to protect your data: in the Text Files settings, there is an option: “Rescue untitled documents when discarding changes”. When this option is on (as it is by default), and you close an untitled document (one that has never been saved to disk), and click “Don’t Save”, BBEdit will save a snapshot of that document’s contents to disk. If you realize you need that text back, it’s there -- just choose the “Rescued Documents” command in the Window menu, and BBEdit will open a Rescued Documents window that you can use to browse all available snapshots and open any desired snapshot into a new document.



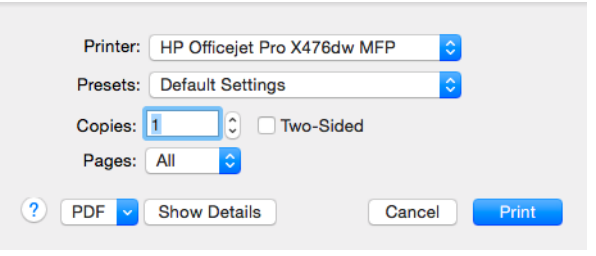
BBEdit can also do some housekeeping for you, if you like. By default it will clean up old data after a week, but you can adjust the interval from 1 to 365 days; or disable the cleanup altogether. (When cleaning up old items, BBEdit will move them into the Trash.)

Handling Images

The image inspector shows information about any text that is recognized from the image (using OS facilities provided for the purpose).

Printing

To print a document, choose the Print command from the File menu. BBEdit will display a standard print sheet in that document's window.

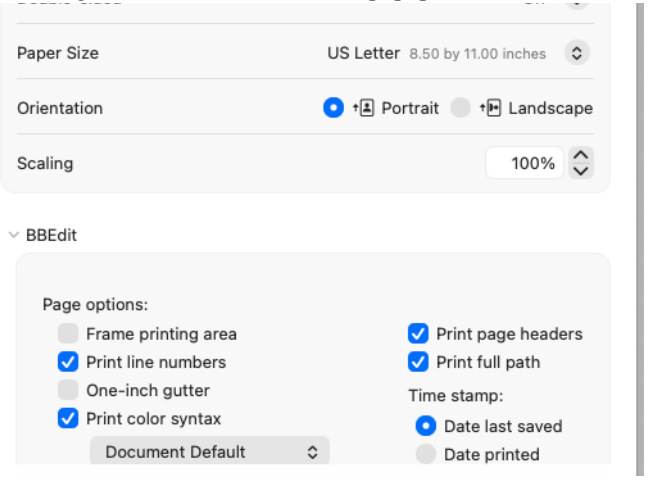


To print one copy of the active document without displaying the print sheet, press and hold the Option and Shift keys and choose the Print One Copy command from the File menu.

To print only the selected range of text within the active document, choose the Print Selection command from the File menu.

Printing Options

You can access BBEdit's application-specific printing options for the current document by choosing the "BBEdit" sheet via the popup in the center of the Print panel



Note You can set defaults for these options, as well as the printing font, in the Printing panel of BBEdit's Settings window.

Page Options:

These options control how the printed pages will be laid out.

Frame printing area

When this option is selected, BBEdit draws a frame around the printed text.

Print line numbers

When this option is selected, BBEdit prints line numbers along the left edge of the paper.

One-inch gutter

When this option is selected, BBEdit leaves a one-inch margin along the left edge of the paper. Use this option if you usually put your pages in three-ring binders.

Print color syntax

When this option is selected, BBEdit will print the document in color, with an added option to allow use of an alternative color scheme when printing, if desired.

The default is to use the document's current color scheme, but if you routinely work in Dark Mode **and** print things out, selecting an alternative color scheme will generally provide a better outcome.

The printing color scheme can be changed per-document via the Print panel, per-project via the project document settings, and globally in the Printing settings pane.

Note BBEdit will not use the color scheme's background color when printing, so as to avoid excessive resource consumption. Also, when printing on paper, a color scheme with a light background will generally give more legible results than one with a dark background.

Print page headers

When this option is selected, BBEdit prints the page number, the name of the file, and the time and date printed in a header at the top of each page.

Print full pathname

When this option is selected, BBEdit prints the full pathname of the file in the header.

Time stamp

The Time Stamp option lets you choose whether the date that appears in the header is the date that the file was last modified or the date that the file was printed.

Tail Mode Monitors Changes

BBEdit now supports “Tail Mode”, which when enabled will cause the insertion point to automatically move to the end of the file if its contents changed while the file was already scrolled to that position.

This behavior can be quite useful for monitoring log files, and thus it is enabled by default for all files whose suffixes map to the “Log File” language. You can also enable or disable this mode on a per-file basis via any of the usual mechanisms, including the Text Display command in the View menu, the Text Options command, or the Text Options popover. (This option is available for **any** locally opened file, regardless of its language type.)

The “Tail Mode” option is persistent, so a file for which you have manually enabled (or disabled) Tail Mode will maintain this status when closed and reopened.

Note If you want to disable “Tail Mode” for the “Log File” language, there is an expert settings command available:

```
defaults write com.barebones.bbedit "EditorTailMode_Log File" -
bool NO
```

CHAPTER

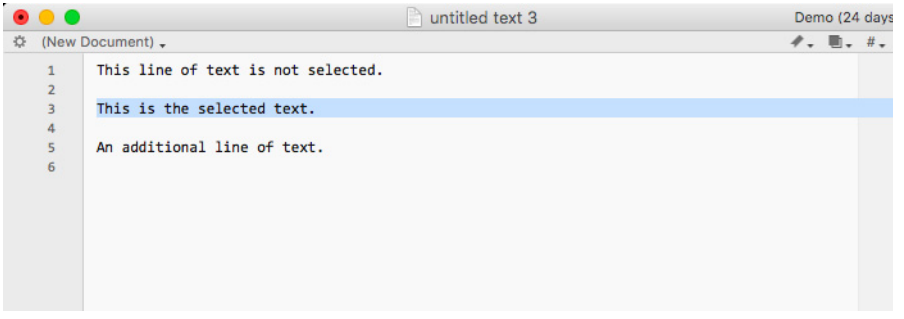
4 Editing Text with BBEdit

This chapter describes the basics of editing text with BBEdit, wrapping text, text manipulations, and file comparison.

In this chapter

Basic Editing

BBEdit behaves like most Macintosh word processors and text editors. Characters that you type in an active window appear at the insertion point, a vertical blinking bar. You can click and drag the mouse to select several characters or words, and (by default) the selected text is highlighted using the system highlight color, which you can set in the Appearance panel of the System Settings.



If you select some text and then type, whatever you type replaces the selected text.

To delete selected text, press the Delete key or choose Clear from the Edit menu. If you have a keyboard with a numeric keypad on it, you can press the Clear key on the keypad to delete the selected text.

In addition to clicking and dragging to select text, you can use the selection commands in the Edit menu.

To select…	Choose this from the Edit menu…
All text	Select All
No text (deselect)	(click anywhere in the document, or type any arrow key)
Line containing insertion point	Select Line
Paragraph containing insertion point	Select Paragraph
Matches found via the “Display instances of selected text” feature	Highlighted Matches (in the Select submenu)
Matches found via the Live Search command (Only available while keyboard focus is in the Live Search bar.)	Live Search Results (in the Select submenu)

You can then cut, copy, or perform any other action that affects the selected text.

Note BBEdit defines a paragraph as a block of text surrounded by blank lines (lines containing no characters other than tabs or spaces). The beginning and end of the document also mark the beginning and end of paragraphs.

Cut, Copy, and Paste

BBEdit supports the standard Cut, Copy, and Paste commands to move (or copy) text from one place to another, as well as a number of more specialized abilities.

To move text, please follow these steps:

1. **Select the text you want to move.**
2. **Choose Cut from the Edit menu.**

   BBEdit removes the text from the window and stores it on the clipboard.
3. **Use the scroll bars to move to the new place for the text if necessary; then click to set the insertion point where the text is to be inserted.**
4. **Choose Paste from the Edit menu.**

   You can paste the contents of the clipboard as many times as you want in any BBEdit window or in any other application.

Pasting inserts the text stored on the clipboard at the insertion point. If there is a selection, pasting replaces the selection with the contents of the clipboard.

Adding Selected Text to the Clipboard

To place selected text on the clipboard without deleting it from the document, just choose Copy from the Edit menu.

To add selected text to the existing contents of the clipboard, hold down the Shift key as you choose the Cut or Copy command. When you hold down the Shift key, BBEdit changes these commands to Cut & Append and Copy & Append.

Copying and Pasting Indented Text

If you need to copy and paste indented text, you can instead choose ‘and Match Indentation’ from the Paste submenu (or type its default shortcut of Cmd-Shift-Opt-V), and this command will attempt to indent the pasted text to the same level as the line on which you paste (or if that line is empty, the most recent non-empty line).

To paste text and have BBEdit immediately select that text, click on the Paste submenu of the Edit menu and choose the ‘and Select’ command, or use its key equivalent of Option-Command-V.

Pasting File Paths and URLs

After you have used the Copy command within the Finder (or in any other application that allows copying file references to the clipboard), you can select the File Paths or File URLs command to have BBEdit paste the appropriate file paths (or URLs) into the current document as text.

Pasting HTML

Some applications put a specific ‘HTML’ data type on the clipboard; you can use this command to paste such data into an editing window as text.

Note If the HTML on the clipboard does not contain any line breaks, BBEdit will run that HTML through its “gentle hierarchical” formatter before insertion.

macOS Sharing Mechanics

BBEdit 16 adds support for the macOS standard sharing mechanics, via the context menu in text selections, and on the File menu when text is selected.

Multiple Clipboards

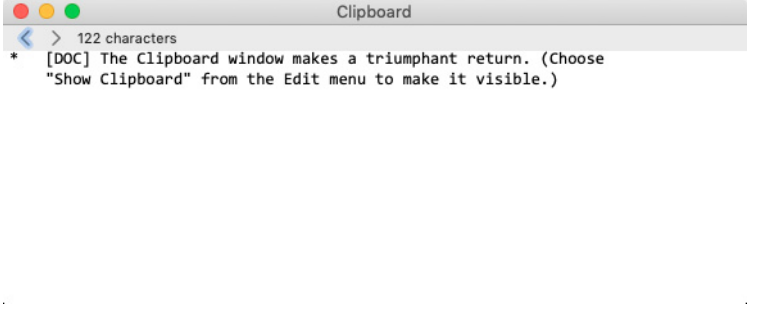
BBEdit supports six separate clipboards. Each time you use the Cut or Copy command, BBEdit automatically switches to the next clipboard (wrapping back around to the first clipboard after the sixth). This way, the last six things you copied or cut are always available for pasting—sort of a “clipboard history.”

By default, the Paste command pastes text from the most recently used clipboard, so if you do nothing special, BBEdit works just like any other Macintosh program. However, by using the Previous Clipboard command in the Edit menu you can access the previous clipboard contents. Next Clipboard moves forward through the clipboard history.

Once you have selected a clipboard, the next Cut, Copy, or Paste command will use the clipboard you chose. (Subsequent Cut or Copy commands will advance to the next clipboard; Paste never advances automatically.)

Alternatively, you can select the Previous Clipboard command in the Paste submenu (or use its key equivalent of Shift-Command-V. This command replaces the pasted text with the contents of the previous clipboard. The previous clipboard becomes current and will be used for any further paste operations; repeated applications of the command cycle backward through the available clipboards.

You can also directly view and navigate through each of BBEdit’s clipboards via the Clipboard window, which you can open via Show Clipboard in the Edit menu.



Drag and Drop

Another way to move text from one place to another is by “drag and drop.” If you drag and drop text from one window to another, BBEdit copies the text to the target window without removing it from the original window.

In addition, you can drag and drop an item from the Finder onto an editing window in BBEdit. If the item is a text file, the file’s contents are inserted. If the item is a folder, a listing of the item’s contents is inserted. If you hold down the Command key while dragging a folder, the path of the item is inserted instead.

Multiple Undo

BBEdit provides the ability to undo multiple edits, one action at a time. The number of edits that may be undone is limited only by available memory. The practical limitation is determined by the extent of the edits and the amount of free memory.

BBEdit also supports multiple Redos. If you have not made any changes after performing an Undo, you can redo each action, in order, by choosing that Redo command from the Edit menu or typing Shift-Command-Z. However, once you perform a new action, you cannot redo any actions that you undid before you made that change.

There is also a Clear Undo History menu command (Control-Command-Z), which will clear the undo history for the current editing window. This command can be useful if you have performed many operations on a file and wish to recover memory stored by Undo state information (in the rare event that should become necessary). You can also script this operation via the “clear undo history” scripting command (see the scripting dictionary for details).

Repeat Last Command

When you need to repeatedly perform the same action, consider using the Repeat Last Command command (in the Edit menu). This command literally repeats the previous menu command, including base menu commands as well as items in the Scripts, Clippings, Apply Text Filter, and Stationery submenus.

Please bear in mind that choosing Repeat Last Command repeats only the selection of the previous menu command, not necessarily its effects. If the state of the active document or the application has changed so that the previous command is no longer enabled, the ‘Repeat’ command will be disabled. For example, if you repeat the Close Document command, BBEdit will close a different document this time (since the previously open document would have been closed by the initial command).

Repeatable commands are those which also appear in the “Commands” window. Therefore, items in the BBEdit menu are not repeatable, nor is Repeat Last Command itself, nor is the “Commands” command.

Some menu commands will not affect Repeat Last Command’s state, specifically: commands in the Window menu, Close commands, and the Next/Previous Document commands. This allows you to (for example) apply a text transformation command to different documents.

The default keyboard shortcut for Repeat Last Command is Command-Y. As always you can adjust this in the “Menus & Shortcuts” settings.

Appearance Modes and Color Schemes

BBEdit no longer alters its own appearance based on the active color scheme. Instead, the scheme that you select within the Text Colors settings pane (see page 273) will remain in effect until the prevailing appearance changes.

Thus, you can select one color scheme to be used in Light Mode, a different color scheme to be used in Dark Mode, and whenever you change the OS appearance in the General system settings, BBEdit will select the desired color scheme. (This plays particularly well with the “Auto” appearance setting in macOS Catalina.)

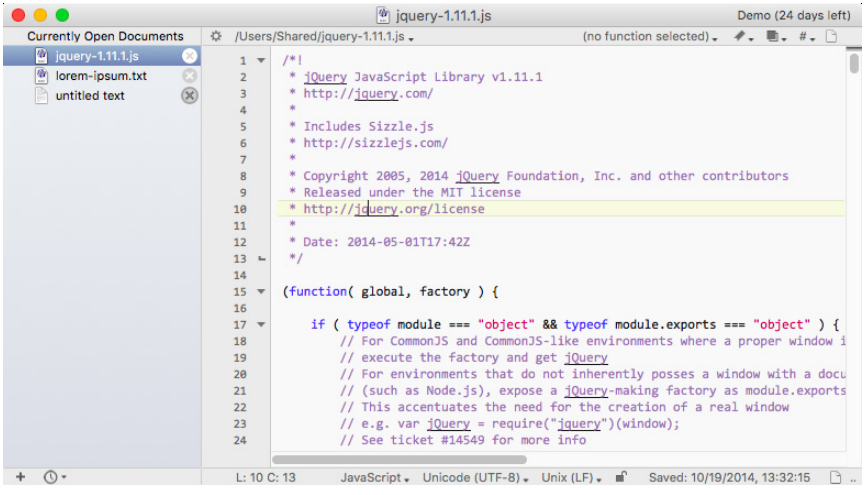
The factory default color schemes are “BBEdit Light” for Light Mode, and “BBEdit Dark” for Dark Mode.

In the Appearances settings pane (see page 250), you may either choose a persistent mode or instruct BBEdit to follow the system’s appearance mode. The factory default appearance mode is “Dark”, which will cause BBEdit to use Dark Mode (and the associated color scheme, per the above) all the time.

If you choose “Light”, BBEdit will be in Light Mode all the time. If you opt to “Use system appearance” then BBEdit's appearance will change whenever the system’s appearance mode changes. (Again, this works nicely with the “Auto” setting in macOS Catalina.)

Window Anatomy

BBEdit text windows have the same controls you are familiar with from other Macintosh applications (for example, text windows are resizable and zoomable, and have both vertical and horizontal scroll bars). Some additional elements which may be less familiar are the navigation bar, the sidebar, the split bar, and the status bar.



In addition, BBEdit will automatically display marks in the vertical scroll bar for elements which are highlighted in the text at various times-such as matches for selected text, Live Search results, spelling errors, and differences.

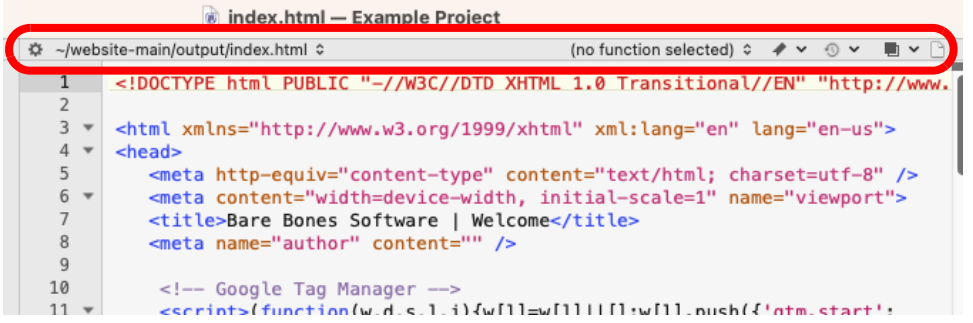
IMPORTANT You can choose whether BBEdit should display all new and opened documents in the frontmost window, or open each document into a new text window, by setting the “Open documents into the front window when possible” option in the Application settings panel (see page 254).

Full Screen Mode

You can expand the current editing window into full screen mode by clicking the green “Zoom” button, or typing the default key shortcut of Control-Command-F, while you can exit full screen mode by typing the default shortcut again, or by pressing the Escape key, or by moving the mouse to the top of the screen to display the menu bar and clicking the double-arrow control in the top right corner.

The Navigation Bar

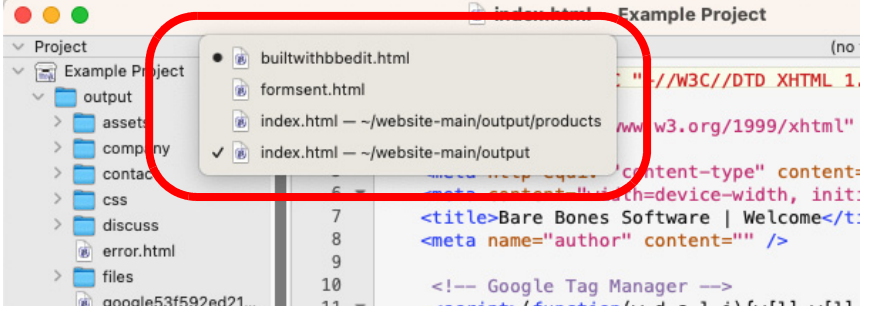
The navigation bar is a panel at the top of a text window which provides controls for selecting the active document and for moving to specific points with the current document. To hide the navigation bar, choose Hide Navigation Bar in the View menu, or turn off the Navigation Bar options in the Appearance settings panel.



You can also use the options in the Appearance settings panel to hide or show individual items on the navigation bar.

Choosing the Active Document

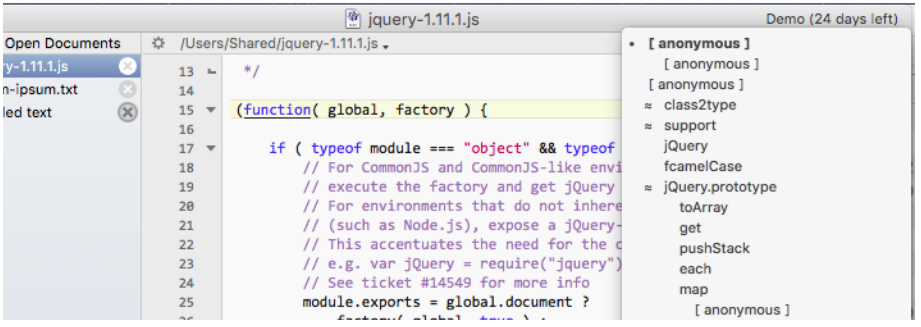
You can click on any document in the sidebar to make that document active or choose Previous Document/Next Document from the View menu. You can also choose a specific document from the navigation bar's file popup to make it active, as shown here:



The Previous Document/Next Document commands select documents by the order in which they were most-recently used, rather than alphabetical order.

Function Navigation

The Function popup menu lists the functions defined in a source code file or various specific tags present within an HTML document. If the current document's language does not support function scanning, the function popup will not be displayed in the navigation bar.



The following indicators appear in the function popup to show the type of function.

Indicator	Meaning
•	The function containing the insertion point
†	C/C++ typedef
◊	C/C++ "#pragma mark" directive
italic name	C/C++ function prototype
1-6	Heading level (in HTML files)
tag name	Tag name for the indicated name or ID attribute value (in HTML files)

Manually Defined Functions

For code written in several languages, including C/C++, PHP, Python, and Ruby, you can manually add customized entries to the function popup menu by inserting suitable "mark" directives within a document. These directives are also often known as comment callouts, and they are generally not case-sensitive (with some exceptions as detailed below.)

In C/C++ documents, BBEdit recognizes "#pragma mark" directives. In HTML and Markdown documents, BBEdit recognizes 'mark' directives placed within HTML comments. For other languages, each directive consists of a line comment followed by a space and the desired 'mark' directive, plus the desired marker string.

The complete set of 'mark' directives includes:

```
mark, fixme, fix-me, note, nyi, review, todo, to-do, xxx, ???, !!!
```

For example, to add an function popup entry named "My item":

In C/C++:

```
#pragma mark this is a named marker
```

In HTML:

```
<!-- mark this is a named marker -->
```

In Markdown:

```
<!-- MARK this is a named marker -->
```

Note In Markdown, callouts are case-sensitive and must be all upper case to be detected.

In JavaScript:

```
// #mark this is a named marker
```

In PHP

```
// #mark this is a named marker
```

```
// #MARK: this is a named marker
```

Note Callout markers may be uppercase, lowercase, or mixed case, and may have a single, optional colon at the end, before the space and the text to appear in the function popup.

In Python:

```
#mark this is a named marker
```

```
# #mark this is a named marker
```

Note In Python files, each directive must be separated from the preceding content by at least one empty line.

Recent File Revisions

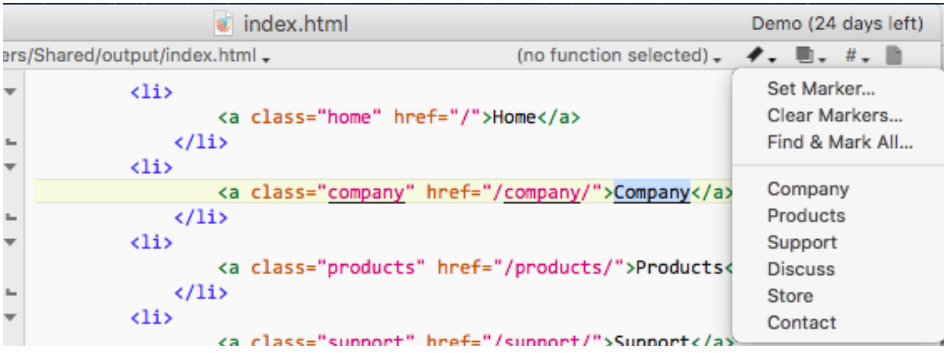
The navigation bar now includes a popup control which lists recent Git revisions as well as any recent filesystem-recorded versions of the current document's file. You can use this control as a shortcut to compare a file's current contents against a recent revision.

Each group of revisions (if applicable) displays the six (6) most recent versions; if there are more then there will be a command which presents an interface for choosing from the entire list.

Revisions are listed in reverse chronological order, so that the most recent ones are at the top of the list.

Navigation with Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Marker popup menu to move quickly to the specified section of the file.



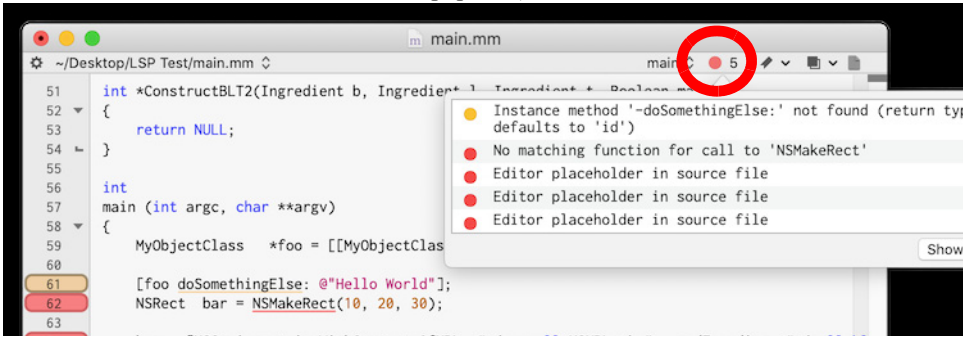
For more information on working with markers, please see “Using Markers” on page 134.

Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if BBEdit supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu.

Opening Document Issues

For documents in languages having a running LSP server, the navigation bar will display a document issues item to the right of the function popup with a colored (or gray) dot and a number. The dot is green if the server returned an empty list (“no issues”) yellow if there are warnings, red if there are errors, and the number indicates the total number of issues found by the language server in the file. If the dot is gray and shows a dash (-), the server has not (yet) returned any diagnostics information for this file.

Clicking on the indicator will open a popover which shows all of the issues in the file; clicking on an issue within the list will select that issue in the document. (Double-click an item to select the location and dismiss the popover.)



Opening Related Files

You can use the Related Files popup next to the Marker popup to quickly open and/or switch back and forth between a file and its counterpart (source file to header, or vice versa).

This button has the same effect as Open Related Files in the File menu (see page 63) but in addition to defined and derived relations, it will also list files that are in the same directory as the active document's disk file.

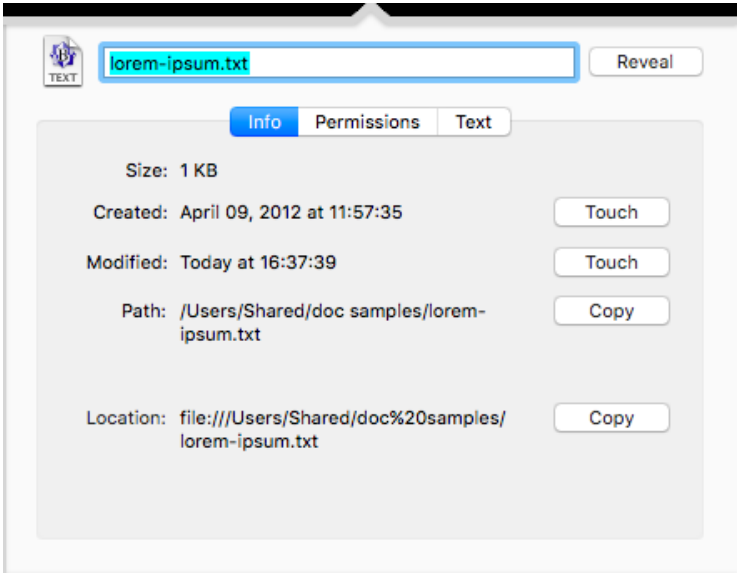
Selecting Colors

When you select a color from the color panel, BBEdit will ask an available language server (if one is engaged for the current source file) to provide an appropriate textual representation of the color. If the server returns multiple representations, BBEdit will present the completion panel so that you can pick one; if only a single answer comes back, BBEdit will insert it.

The Document Info Panel

Clicking on the document icon in a window's navigation bar (below) will open a spring-loaded info panel (next page) which displays basic information about the current document's file.

The document info panel also allows you to rename the current file (for local files only), to "touch" its creation/modification dates, to copy its path in filesystem or URL format, or to change its permissions (via the options presented in the Permissions tab). To dismiss this panel, click outside it, switch to another window or application, or press the Escape key.



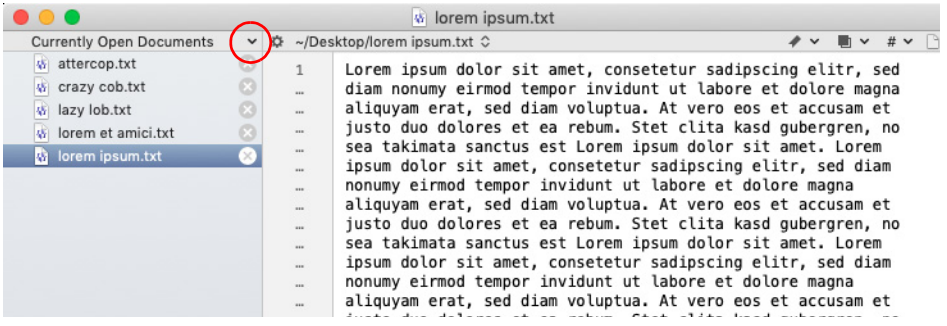
Key Equivalents for Navigation Bar Menu Items

You can assign key equivalents to the controls on the navigation bar from the Navigation Bar entry in the Menus & Shortcuts settings panel. So, for example, you can assign a key equivalent to Open Function Menu, then press that key combination and use the arrow keys to navigate the current document's function list directly from the keyboard.

If the current document has a corresponding disk file, the navigation bar will display that file's full path (or as much of the path as space permits). If the document has not been saved to disk, the toolbar displays "(New Document)" instead of a file name.

The Sidebar

If BBEdit is configured to open documents into the front window, it will display a sidebar down the left-hand side of each editing window which lists all the documents currently open in that window. To hide (or show) the sidebar, choose the Show Sidebar (or Hide Sidebar) command in the View menu, or type its default key equivalent of Command-0 (Command-Zero). Click any document's name in the "Currently Open Documents" list to make that document frontmost in the text window.

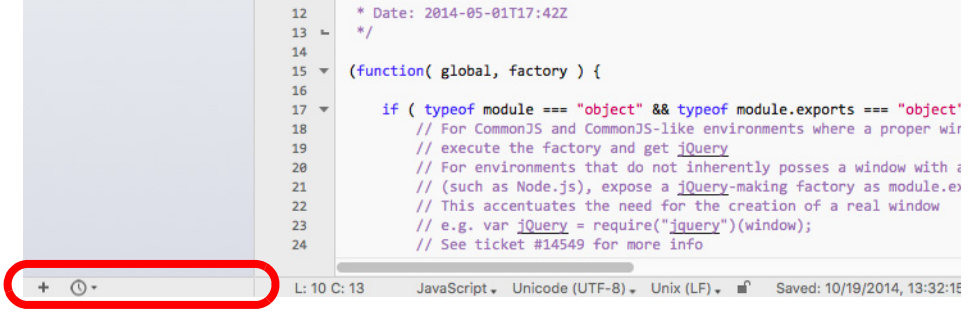


BBEdit will display a dark ring around the "close" button of each open documents which contains unsaved changes. In addition, the title bar and status bar icons of such documents will change to indicate their unsaved condition.

Dragging a document's name from the list has the same effect as dragging its proxy icon in the navigation bar.

You can also drag documents within the list to manually reorder them, or click on the Arrange popup (encircled in red above) to rearrange all documents present based on their names, modification dates, or history order ("Most Recently Opened" or "Least Recently Opened").

There are also two controls below the sidebar, which you can employ to perform various additional actions. (These controls are very similar to the controls in a project window's sidebar.)



To open an existing file into the current text window, you may click the Add (plus) button, choose Open from the File menu, or drag and drop the file from the Finder into that window's sidebar.

To create a new document, you may choose New Text Document in the New submenu of the File menu, or Control-click in the sidebar and choose New Test Document (or New HTML Document) from the contextual menu.

To move a single document from the current text window into its own text window, just Control-click on that document in the file list and choose the Move to New Window command in the contextual menu. To move multiple documents, select them and choose Move to New Window to create a new text window containing all the selected documents.

Alternatively, you can choose the Move to Window command in the View menu or to the contextual menu in the sidebar. This command will open a dialog in which you can choose an extant window for the active document (or selected sidebar documents, as appropriate), and upon confirmation moves the document(s) to the designated window and makes it active. A search box is available to filter candidates. Note that moving all of a window's documents to another window may result in the active window closing.

To coalesce multiple unsaved documents, select the Gather Untitled Documents command in the View menu. This command will collect all untitled (never saved) documents into the active window, removing them from other windows (and possibly closing those windows in the process).

To close a document, choose Close Document from the File menu, click on the close box next to its name in the list, or Control-click on it in the list and select Close in the contextual menu. You can also choose the Close Others command in the contextual menu to close all documents *except* for the selected document.

To reopen any recently open file, click on the Recent (clock) popup and select the desired file.

To move a document from one text window to another, drag its name from the first text window’s sidebar into the second text window’s sidebar. You can select and move multiple documents at once.

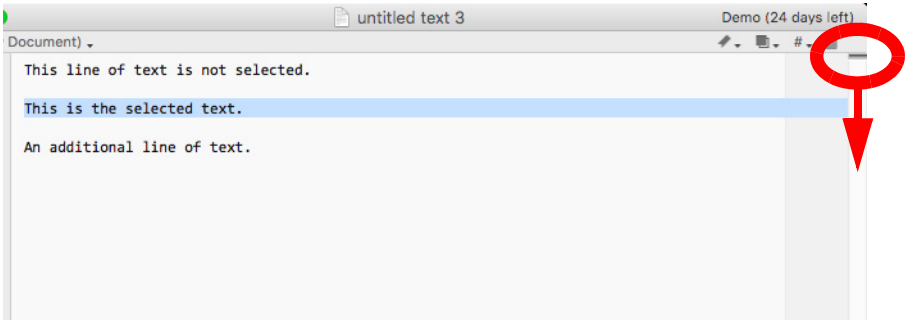
To save the current document, choose Save from the File menu or the action menu. To save multiple documents at once, select them, then Control-click on them and select Save in the contextual menu. To save all documents in the window at once, hold down the Option and Shift keys and choose Save All in Window in the File menu.

The Split Bar

Every text window and every browser text pane has a split bar, a small black bar above the scroll bar, that lets you split it into two active view regions. Splitting a text pane lets you view and edit a document’s content in two places at the same time. Each region is independently scrollable.

Note Scrolling the non-active split region does not automatically change view focus.

To split the text pane, simply drag the split bar down and let go.

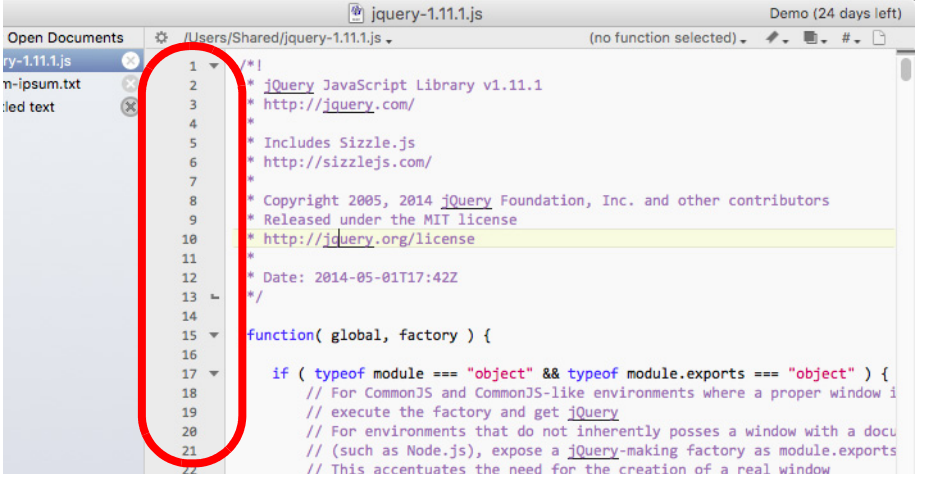


To collapse the text pane back down to a single region, drag the split bar (starting from anywhere along its length, not just at its right end) back up to its original position.

Tip Double-clicking the split bar unsplits a split text pane or restores the last-used split position. If the text pane has never been split, it will be split 50-50. To force a 50-50 split for a previously split text pane, Option-double-click the split bar when it is in its original position.

The Gutter and Folded Text Regions

The gutter is the vertical bar directly to the left of the text area, and immediately to the right of the line number display bar (not shown), which contains indicators for folded and foldable regions (automatically-generated folds).



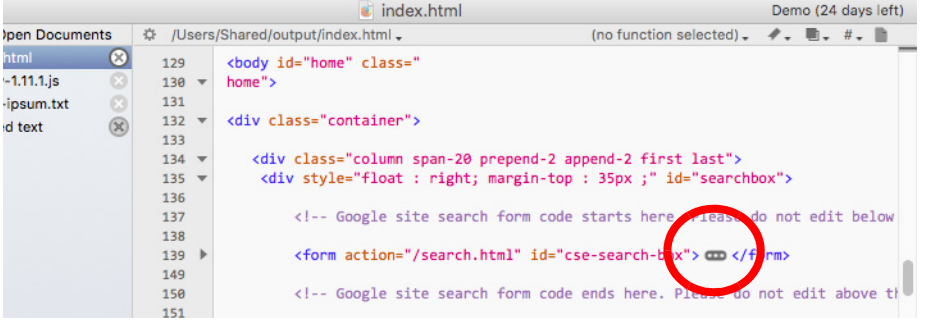
Folding Controls

The triangular controls displayed in the gutter are disclosure triangles; you can click on them to fold or expand regions within the document. If there are nested folds present, Option-clicking on the outermost fold will expand or collapse that fold and all subordinate folds.

You can also employ the commands on the View menu to expand or collapse folds, or fold manually-selected ranges of text. (See “The View Menu” on page 106.)

The linear bars displayed in the gutter are range end indicators. They show where each foldable range ends.

When you fold a range, BBEdit displays a fold indicator within the document to represent that range. To expand the range, you can either click on the disclosure triangle or double-click the fold indicator.



If you expand a range by double-clicking the fold indicator, the entire range will remain selected after expansion.

The Status Bar

The status bar is located directly to the left of the horizontal scrollbar. The status bar displays the current cursor position and the document's save status, and contains popup menus showing the language, text encoding, and line break format of the current document

You can use the options in the Text Status Bar group of the Appearance settings panel to hide or show individual items on the status bar, and if you disable these options, BBEdit will hide the status bars.

Cursor Position

This section of the status bar shows the current line and character position of the insertion point. In addition, clicking on the cursor position display will open a popover that you can use to go to another line. (The popover will remain active until you click outside of it or press the Escape key, allowing you to browse the file by line number.)

NOTE You can now assign a key shortcut to the Cursor Position status bar item; you can use this to bring up the alternative "Go to Line" interface (i.e. assign Command-J to use this popover as an alternative to the Go menu's Line Number command.

Language

The Language popup menu displays the language mapping for the current document. You can change this mapping by choosing a different language from the popup.

Text Encoding

The Text Encoding popup menu displays the encoding used to open the current document. You can change the encoding in which the document will be saved by choosing a different encoding from the popup.

To choose an arbitrary encoding, even one not currently displayed, choose Other from the popup and pick your desired encoding from the resulting list.

Line Break Type

The Line Break Type popup menu shows the line break format of the current document's disk file. You can change the line break format with which the file will be saved by choosing it from the popup.

Document Lock State

The padlock icon immediately to the left of the Document Save Date indicates whether the document is currently writeable or locked.

Document Save Date

The Document Save Date section of the status bar displays the date and time that the document was last saved (if applicable).

Document Statistics

This section of the status bar dynamically displays the number of characters, words, and lines in the document or the active selection (if any). (This icon will be white when BBEdit is displaying statistics for the whole document, and green when it's displaying statistics for the current selection.)

You may also click on the statistics section at any time to display the Text tab of the document information panel which will present details for both the whole document and the current selection range (if any).

Magnification

The Magnification popup menu displays the current document's magnification level and allows you to change it. The default magnification level is 100%.

Key Equivalents for Status Bar Items

You can assign key equivalents to the items on the status bar from the Status Bar entry in the Menus & Shortcuts settings panel. For example, you can assign a key equivalent to the Line Breaks popup, then press that key combination and use the arrow keys to select the desired line break option directly from the keyboard.

The View Menu

This menu contains commands which you can use to toggle the display of navigational elements in text windows, to fold and expand regions of text, to select documents, and to get information on documents and files.

Text Display

This submenu contains commands which control various text formatting and display options. You can set key equivalents for any of these commands under the Text Display entry of the View menu in the Menus & Shortcuts settings panel. You can also adjust many of the same options via the Text Options command in the Edit menu.

Show/Hide Fonts

This command toggles display of the standard system font panel, which you can use to set the font, font size, text style, and tab spacing for the active document.

IMPORTANT The chosen display style will be used for *all* text in the window; BBEdit does not support the use of selective text styles.

Note The font changes you make by using this command affect only the active document. To set the default font, size, style, and tab width for all documents, use the "Default Font" option in the Editor Defaults settings panel.

Soft Wrap Text

This command toggles the use of soft wrapping in the current document. (See "Soft Wrapping" on page 125.)

Show/Hide Page Guide

This command toggles display of the page guide in the current document. (See "Page guide" on page 122.)

Show/Hide Tab Stops

This command toggles display of tab stops in the current document. (See "Tab stops" on page 122.)

Show/Hide Line Numbers

This command toggles display of line numbers in the current document. (See "Line numbers" on page 122.)

Show/Hide Gutter

This command toggles display of the gutter in the current document. (See "The Gutter and Folded Text Regions" on page 102.)

Show/Hide Invisibles

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 123.)

Show/Hide Spaces

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 123.)

Show/Hide Navigation Bar

Choose this command to hide or show the navigation bar in the frontmost text window. (See “The Navigation Bar” on page 94.)

Show/Hide Editor

Choose this command to hide or show the editing pane within a project window.

Show/Hide Sidebar

Choose this command to hide or show the sidebar within the frontmost text window. (See “The Sidebar” on page 100.)

Show/Hide Open Documents

Choose this command to hide or show the “Open Documents” section within a project window’s sidebar.

Show/Hide Worksheet & Scratchpad

Choose this command to hide or show the “Worksheet & Scratchpad” section within a project window’s sidebar.

Balance

This command locates the pair of parentheses, braces, brackets, or smart (curly) quotes that surround the insertion point or the current selection. If there are unmatched delimiters within this area, BBEdit beeps. You can also double-click a delimiter character to invoke this command.

When syntax coloring is active for a document, Balance (including auto-balance) will ignore balance characters that appear inside strings or comments.

Balance & Fold

This command behaves identically to Balance (above) except that in addition to locating the paired delimiters, BBEdit will also generate a fold range including the delimiters and all the text they contain.

Fold Selection

This command generates a fold range from the currently selected text. You can use Unfold Selection (below) or double-click the fold indicator to expand the fold. When there is no active selection, this command is disabled.

Unfold Selection

This command will expand any text folds in the selection range. When there is no active selection, this command is disabled.

Collapse Enclosing Fold

This command will collapse the auto-generated fold that most closely surrounds the current insertion point (or the start of the selection range).

Collapse All Folds

This command will collapse all automatic fold points in the current document.

Expand All Folds

This command will expand all text folds in the current document, whether automatically generated or manually created.

Collapse All Folds

This command will collapse all automatically generated fold regions in the text, whether or not they are contained within other folds. (This is distinct from "Collapse Top-Level Folds", which collapses the top-level folds but leaves any nested folds open.)

Collapse Folds Below Level

This command presents a submenu listing all available fold levels within the current document. Choosing any level will collapse all of the automatically generated fold regions in the text that are below that fold level. So, for example, choosing Collapse Folds Below Level 1 will leave the top-level folds open, but will collapse all of the folds below the top level, whether or not they are contained by other folds.

Re-Center Selection

This command performs the same operation as the Emacs "recenter" command (Control-L), and scrolls the line containing the insertion point to the center of the view area (or as close to it as possible). If the selection range is non-empty and longer than the visible area, choosing this command repeatedly will alternate between centering the start and end of the selection range.

Previous Document/Next Document

You may use these commands to switch between documents within the frontmost text window. (By default, BBEdit selects documents in most-recently viewed order, but you can choose to have it select documents in name order via an expert settings. For details, see the "Expert Settings" page in BBEdit's built-in Help.)

Move to New Window

Choose this command to open the active document of the frontmost text window or project window into its own text window. If the frontmost text window contains only one document, this command will be disabled.

This command will also be available in the contextual menu when you right-click on any open file in the sidebar.

Open in Additional Window

Choose this command to open the active document of the frontmost text window or project window into an additional text window, while leaving it open in the current window.

This command will also be available in the contextual menu when you right-click on any file in the sidebar.

Open in New Window

This command will be available in the contextual menu when you right-click on any item in the Project pane.

Merge Windows

Choose this command to collect all open documents from all other editing windows into the frontmost editing window, and then close all other editing windows, thus merging all open documents into the active window. (This command has no effect on project windows.)

Show in Finder

Choose this command to open a Finder window which will display the active document's file. If the active document is not associated with a file, this command will be disabled. Using this command is the same as clicking (without dragging) the document proxy icon in the navigation bar.

If the selected text in a document is the name of a file, hold down the Option key as you open the File menu and choose the Reveal Selection command to have BBEdit open a Finder window which will display that file.

Show in Project List

When there are one or more projects open, choose this command to locate the frontmost project which contains the active document, and reveal that document's file in the project's file list.

Go Here in Terminal

This command is enabled when the active document has a corresponding disk file. Choose this command to open a Terminal window with the current working directory set to the document's parent directory.

Go Here in Disk Browser

This command is enabled when the active document has a corresponding disk file. Choose this command to open a disk browser in the document's parent directory.

Cursor Movement and Text Selection

BBEdit gives you several ways to move the insertion point and change the selection. You can click and drag using normal Macintosh text selection techniques or you can use various keys on the keyboard.

Clicking and Dragging

You can select text in an editing window in the normal Macintosh fashion, by clicking and dragging. Holding down the Shift key while clicking or dragging extends the selection.

	No Modifier	Shift
Click	Move insertion point	Extend selection
Double-click	Select word	Extend selection to word
Triple-click	Select line	–none–

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

You can hold down the Command or Option keys when clicking or double-clicking to trigger special actions:

	Option	Command
Click	–none–	Open URL
Double-click	Look up selected word in programming reference	–none–

BBEdit also allows you to select entire lines by clicking in the left margin of an editing window. (If you have enabled line number display via the Show Line Numbers option in the Appearance settings panel, you can click in the line number as well.) You can click and drag to select multiple lines, double-click to select an entire paragraph, or double-click and drag to select a range of paragraphs.

Arrow Keys

You can use the arrow keys to move the insertion point right, left, up, and down, and augment these movements with the Command, Option, and Control keys:

	No Modifier	Option	Command	Control
Up	Up one line	Up one screen	Start of document	Edit: Lines: Move Line Up*
Down	Down one line	Down one screen	End of document	Edit: Lines: Move Line Dn*
Left	Left one character	Left one word	Start of line	Previous case transition or word boundary
Right	Right one character	Right one word	End of line	Next case transition or word boundary

Holding down the Shift key extends the selection. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

Note *: By default, the Control-Up arrow and Control-Down arrow key shortcuts invoke the Move Line Up and Move Line Down commands respectively. If you clear either or both these shortcuts via the Menus & Shortcuts settings pane, then these gestures will instead scroll the text view up or down. Alternatively, you may assign custom key shortcuts for text view scrolling in the Miscellaneous section of the Menus & Shortcuts settings pane.

CamelCase Navigation

BBEdit supports CamelCase navigation. CamelCase (also “camel case”) is the practice of writing intercapitalized compound words or phrases; it is used as a standard naming convention in several programming languages, and as an automatic link creation method in wiki content.

You can move from one part of a CamelCase word to the next by holding the Control key down and tapping the right (or left) arrow key to jump to the next (or previous) transition from lower-case to upper-case or the next word boundary, whichever comes first.

Delimiter Handling

When the insertion point is inside of an opening delimiter, or immediately outside a closing delimiter, BBEdit will highlight the matching delimiter, as appropriate.

If the insertion point is immediately inside of a container element within an HTML document, delimiter matching will highlight the corresponding pair of opening and closing elements. (As part of the HTML tTools, this feature is only available during evaluation or with a paid license.)

The default color is the same one for highlighting live search matches; this may be adjusted in the respective color scheme via the “Text Colors” settings pane.

This feature may be controlled via the “Highlight matched delimiters” option in BBEdit's “Editing” settings pane.

Rectangular Selections

By holding down the Option key as you drag, or holding down the Shift and Option keys while clicking, you can select all text lying within a specified rectangular area (column). You can then perform all of the normal editing operations on this “rectangular selection,” such as Cut, Copy, Paste, or drag and drop, as well as text transformations such as Change Case, Shift Left, Shift Right, Convert Spaces to Tabs (and vice versa), Increase Quote Level, Decrease Quote Level, Strip Quotes, and Zap Gremlins.

BBEdit offers two additional commands in the Edit menu: Select Up and Select Down. These commands facilitate rectangular selection via the keyboard. (Their default key equivalents are Control-Shift-up arrow and Control-Shift-down arrow, which can be changed as usual in the Menus & Shortcuts settings panel.) Starting from either an existing selection that does not cross a line boundary, or an existing rectangular selection, the Select Up command will extend the selection range up, or Select Down will extend it down, within the same column, thus creating (or extending) a rectangular selection.

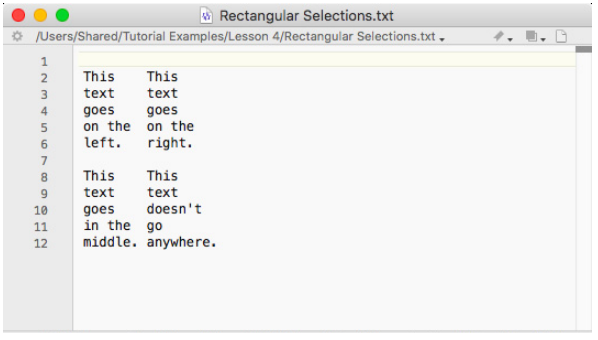
IMPORTANT **BBEdit now allows you to make a rectangular selection within any document, even if the “Soft wrap text” option is enabled.** (Note that the rectangular selections are made in the actual text, not in the visual representation, so if a rectangular selection crosses a wrapped line, the wrapped portion of that line will not be highlighted.)

Working with Rectangular Selections

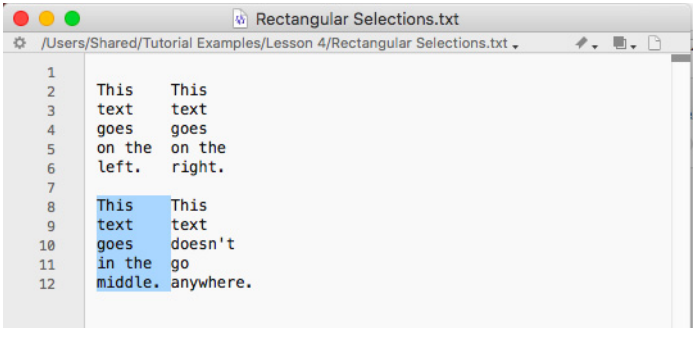
Commonly, while working with text, you will be performing actions on a line-by-line basis; for example, when making a selection, you will start by selecting the contents of one line before moving on to the next. However, if you need to deal with tabular data, it can be useful to think in terms of rectangles or blocks of text that include parts of several lines. This is where you can make use of BBEdit’s ability to manipulate rectangular selections.

Example: Moving a Column

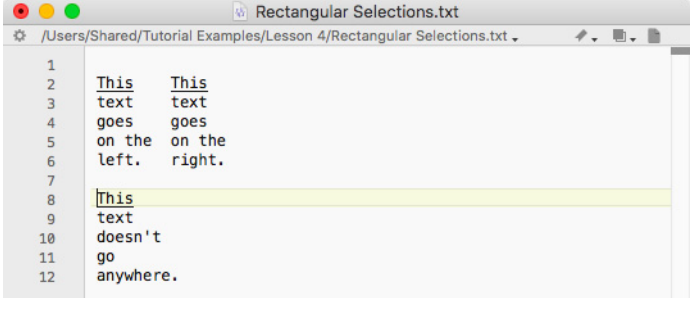
Consider you have the document shown below, and you want to move only the bottom left column (the one that says “This text goes in the middle”) and move it in between the top left and top right columns. To do this using standard selection methods, you would have to perform five separate cut-and-paste operations. However, by using rectangular selections, you can move the whole column in one operation.



To start, hold down the Option key while dragging over the bottom left column, until you get a selection that looks like this:



Choose Cut from the Edit menu (or press Cmd-X) to cut the selected text out of the document and place it on the Clipboard.

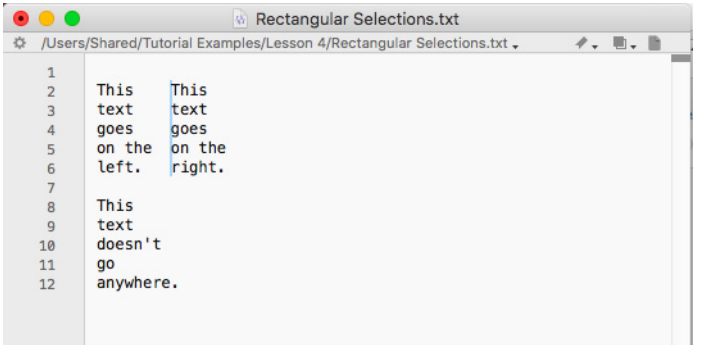


Next, you must paste in the text you just cut. You can do this in either of two ways:

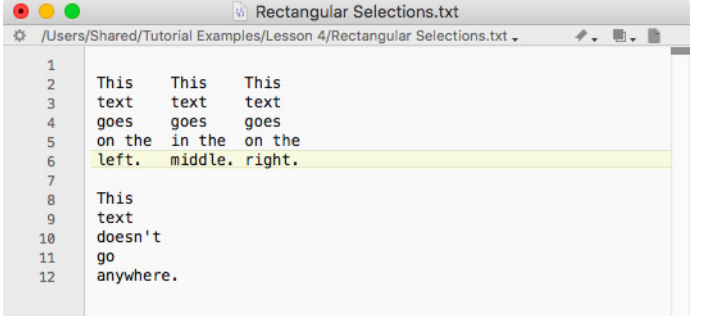
- Use the Paste Column command, which will “paste down” from the current insertion point. This allows you to directly insert text without needing to make a rectangular selection first.

- Make a rectangular selection as described below, and then use the standard Paste command. This procedure is less efficient for moving columnar data than using the Paste Column command, but it allows you to select and replace a region of text as well as simply inserting text.

To manually make a rectangular selection prior to pasting text, position the arrow pointer just to the left of the top right column, press and hold the Option key, press the mouse button, and drag straight down until you have a very thin vertical selection just to the left of the whole column, as shown below.



Now, paste the text you previously cut back in, and the task is finished.



Filling Down

When you apply the Paste Column command and the pasted text contains no line break (or only a single line break at the end), BBEdit will perform a “fill down”, placing a copy of the pasted text on each line within the selected column.

Further Details

Some word processors also provide support for rectangular selections which works a little differently than BBEdit's, so you may wish to keep this difference in mind. Typically, when you copy a rectangular selection of text to the clipboard in these programs, they handle that piece of text differently than text copied from a line-by-line selection. Then, when you paste, the text will be entered in a block, even when you have not made a rectangular selection to paste into.

BBEdit does not do this. Instead, when you copy a rectangular selection to the clipboard, BBEdit turns the selection into a series of individual lines, which is why you must make a rectangular selection before pasting, so BBEdit will know it should paste the text in block fashion. Though this method does require an extra step, it is more flexible, because you can select a set of lines and then paste it as a block, or vice versa.

Scrolling the View

You can control how keyboard scrolling behaves via custom key shortcuts or accelerated scrolling as follows.

Edit View Scrolling

You can assign custom key shortcuts to scroll the active document in any of the four available directions: Up, Down, Left, and Right, via the shortcut actions provided in the "Edit View Scrolling" submenu of the Miscellaneous section in the Menus & Shortcuts settings pane.

Accelerated Scrolling

When clicking the arrows in a scroll bar, you can use the Command and Option keys to accelerate the scrolling. These shortcuts also apply if you use a mouse with a built-in scroll wheel.

Modifier	Scroll Speed
none	Normal
Command	2x accelerated
Option	3x accelerated
Command+Option	6x accelerated

The Delete Key

The Delete key deletes the character to the left of the insertion point. If you have selected text, the Delete key deletes all the text in the selection. You can use the Command and Option keys to modify the way the Delete key works:

Modifier	Action
none	Deletes character to the left of the insertion point
Option	Deletes to the beginning of the word to the left of the insertion point
Command	Deletes to the beginning of the line
Command+Option	Deletes to the beginning of the document

Holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

The Numeric Keypad

Some keyboards have a numeric keypad on the right side. Normally, you use the keys on the keypad to enter numbers.

To toggle the behavior of the keypad between moving the cursor and entering numbers, hold down the Option key and press the Clear key in the upper-left corner of the keypad. (This key is also labeled Num Lock on some keyboards.)

When keypad navigation is active, BBEdit will perform the following actions:

start of line **7**	up **8**	Scroll up **9**
left **4**	show selection **5**	Right **6**
end of line **1**	down **2**	Scroll down **3**

You can use the Shift key with the keys on the numeric keypad to extend a selection. You can use the Command and Option keys with the 2, 4, 6, and 8 keys as you would the arrow keys.

Line Number Command

To move the insertion point to a specific line, use the Line Number command in the Go menu. When you choose this command, BBEdit opens a Go To Line panel in the frontmost document. Type the number of the line you want to move to and click Go To to have BBEdit bring that line into view with the insertion point at its start.

You can drag the panel around by clicking anywhere other than the input field); use Return or Enter to confirm the current input, and Escape or Command-Period to dismiss the panel.

The Line Number panel includes a summary of all supported input formats; it will accept both absolute values, relative inputs, and character offsets. Entering a value prefixed with +/- will add that value to the current line number.

For example, with the insertion point in line 100, “+75” will move to line 175; “-75” will go to line 25. (When you enter an unsigned number, BBEdit will move to the specified line number.)

In addition, you can enter a line number of the form “xx:yy”, in which "yy" is a character offset into the destination line. If the character offset exceeds the number of characters on the line, BBEdit will place the insertion point at the end of the specified line.

Alternatively, you can jump to an absolute character offset, by using the ‘line:column’ syntax but leaving the ‘line’ blank or specifying it as zero. For example, entering “0:1500” or “:1500” will cause BBEdit to place the insertion point before the 1500th character in the document. (The range syntax works too; so you could use “0:12-0:56” to select characters 12 through 56.)

Note The Line Number command honors the Use “Hard” Lines in Soft-Wrapped Views option in the Editing settings panel.

Function Keys

If your keyboard has function keys, you can use the following key equivalents for cutting and pasting, to scroll, and to move the insertion point.

	No Modifier	Option	Command	Shift
del	forward delete	delete to end of word	delete to end of line	
Home	scroll to top of document		move insertion point to start of document	
End	scroll to end of document		move insertion point to end of document	
Pg Up	scroll page up			
Pg Dn	scroll page down			

Note Holding down the Command and Option keys as you press the forward delete key deletes to the end of the document.

Resolving URLs

To resolve a URL (Uniform Resource Locator), you can Command-click anywhere in the URL text, or Control-click to bring up the contextual menu and choose Open URL from the menu. BBEdit will examine the URL and launch the appropriate helper application. If the URL is not valid or the helper application cannot be found, BBEdit will beep.

Note Some Web browsers cannot resolve URLs if the request is sent when the browser is starting up. If your Web browser does not properly resolve the URL, wait until the browser has finished starting up and then try again.

Bare Bones Software gratefully acknowledges John Norstad for providing the URL parsing code.

Touch Bar Actions

On any machine having a Touch Bar, BBEdit will present dedicated keys within the Touch Bar whenever any of the following window types is active:

- Editing windows: Keys to toggle the sidebar's visibility and navigate to the previous or next document.
- FTP/SFTP browser windows: Keys to "go up", "go down" (into selected folder), and "reload".
- "Preview in BBEdit" windows: Keys corresponding to each button in the Preview window's tool bar.
- Text factory windows: Keys to add or remove actions, and to run the factory.

Text Completion

BBEdit can either automatically offer completions for words and symbols as you edit, or offer completions only when you manually request them. You can control when BBEdit offers completions via the “Show text completions” option in the Completion settings panel:

- After a delay in typing: If you pause briefly while typing, BBEdit will figure out the possible completions for what you just typed and display them.
- Only manually: BBEdit will only display possible completions when you invoke the Complete command (see below).

Note This feature is also known as “autocomplete” or “autocompletion”.

You can also enable (or disable) text completion on a per-language basis by adding a custom language setting in the Languages settings panel.

Completions are derived from a variety of sources, including (in no particular order):

- clippings (both language-specific and universal);
- ctags symbols computed by running the current document through 'ctags';
- ctags symbols found in 'tags' files in the current document's hierarchy;
- predefined names for the source code language at the point of completion;
- language-specific completions (both predefined and derived from the current document’s content);
- dictionary words provided by the system spelling service

Each completion item has an associated completion symbol which indicates its source; for a complete listing of completion symbols, see the Completion Symbols table on the following page.

Invoking Completion

You may trigger a completion at any time (whether or not automatic completion is enabled) by using the “Complete” command in the Edit menu or pressing its default key equivalent of F5. (You can change this equivalent via the Menus & Shortcuts settings panel.)

Note Text completion treats clippings in the same way that the "Insert Clipping" command used to (and still does). So, the behavior of F5 should be indistinguishable if you were used to using it to complete clippings.

You can also optionally use the Escape key to invoke text completion. By default, this key is not used for completion, but you may enable it by turning on the corresponding option in the Keyboard settings panel.

Note Because the Escape key has a special meaning when the “Use Emacs key bindings” option is on (in the Keyboard settings panel), if this option is active and you choose to use the Escape key as a completion trigger, you will have to press Escape twice to invoke completion.

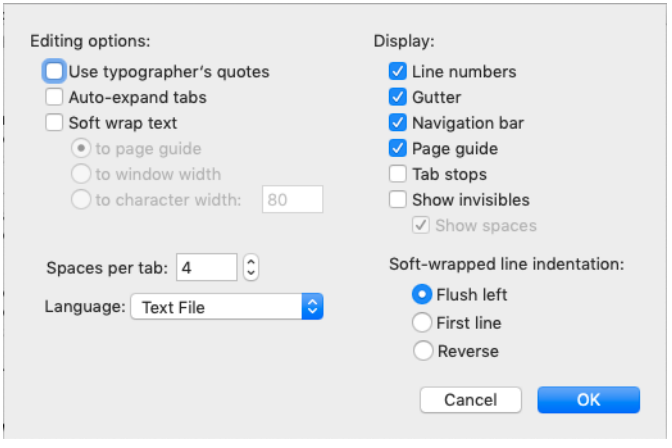
Completion Symbols

Item Type	Badge Shape	Symbol Color	Letter (Symbol)
Global			
`Clipping`	Circle	Black	C
`Spelling word`	Square	Black	w
Content-derived			
`Class`	Square	Light purple	C
`CSS property`	Square	Dark purple	p
`Enumeration name`	Square	Orange	E
`Enumeration value`	Square	Orange	e
`External variable`	Square	Light green	V
`Function`	Square	Light blue	*f*
`Function prototype`	Square	Light blue	F
`Global variable`	Square	Light green	G
`HTML attribute name`	Square	Dark purple	A
`HTML attribute value`	Square	Dark purple	V
`HTML element`	Square	Dark purple	<
`IVar`	Square	Light green	v
`Include file`	Square	Dark purple	#
`Language keyword`	Square	Blue	K
`Local variable`	Square	Light green	L
`Macro`	Square	Light red	#
`Member`	Square	Light green	v
`Method`	Square	Light blue	M
`Namespace`	Square	Light purple	N
`Predefined symbol`	Square	Blue	k
`Static type`	Square	Orange	T
`Struct`	Square	Orange	S
`Union`	Square	Orange	U

Text Options

You can use the Text Options command to change the way BBEdit edits text and the way it displays text and additional elements in its windows. When you choose this command, BBEdit will drop a Text Options sheet in the current text window.

The controls on the Text Options sheet are divided into two parts: the Editing options on the left control the way BBEdit behaves while you type, and the Display options on the right control the appearance of the BBEdit window.



Note You can also change many of these options using the commands in the Text Display submenu of the View menu.

Changes you make in the Text Options sheet affect only the active document or window. To set options which will apply to all text windows you open, use the Editor Defaults and Appearance settings panels.

Editing Options

These options control the way BBEdit behaves as you type text in the active document window. Changes you make here affect only that document. To change the default editing options for documents that you will open in the future, use the Editor Defaults settings panel.

Use typographer's quotes

When this option is on, BBEdit will automatically replaces straight quotes (" ') with typographer's quotes (“ ” ‘ ’) in the current document. If you need to type a straight quote when this option is selected (or to type a typographer's quote when the option is not selected), hold down the Control key as you type the " or ' key.

Note We recommend against using this option if you are editing HTML content, email content, or program code.

Auto-expand tabs

When this option is selected, BBEdit inserts an appropriate number of spaces when you press Tab, rather than inserting a tab character.

Additionally, when there are only spaces (and tabs) between the insertion point and the start of the current line (or the first non-whitespace character on the line), BBEdit will delete a tab stop's worth of spaces when you press Delete (Backspace).

Soft wrap text

When this option is selected, BBEdit soft-wraps the text in the file to the right margin that you choose: the page guide, the window width, or a specific number of characters. The page guide is an arbitrary visual boundary whose width you can set in the Appearance settings panel. (See “Soft Wrapping” on page 125 to learn how wrapping works in BBEdit.)

Spaces per tab

This option allows you to control how many spaces are equivalent to a tab within the active document.

Language

The Language menu lets you specify which source code language the file uses. The file’s language setting affects how BBEdit performs syntax coloring and parses function names for the function popup menu. BBEdit generally determines the file’s language from its filename extension, using the mapping table in the Languages settings panel.

For example, “.cp” files are C++, and “.m” files are Objective-C. You can use this menu to override those settings for a specific file. To quickly check the language for a file, look at the Languages popup in the status bar, or choose the Text Options command and look at the Languages popup in the resulting options sheet.

Display Options

These options determine which controls appear in the frontmost text window, regardless of whether that window contains one or more documents. Changes you make here affect only that window. To change the display characteristics for text windows that you will open in the future, use the Appearance settings panel.

Line numbers

This option displays line numbers along the left edge of the window.

Gutter

This option shows or hides the gutter in the window.

Navigation bar

This option shows or hides the navigation bar in the window.

Page guide

This option shows or hides the page guide in the window.

Tab stops

This option shows or hides tab stop indicators in the window.

Show invisibles

This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and "gremlins" (other invisible characters). BBEdit uses these symbols:

Symbol	Meaning
∆	tab
•	space
●	non-breaking space
¬	line break
¶	page break
¿	other non-printing or special characters

If you turn on Show Invisibles, the Show Spaces option will become available, allowing you to enable display of the visually "noisy" space characters if you desire.

Syntax Coloring

When this option is selected and the editing window contains a document in a programming language BBEdit recognizes, BBEdit displays keywords and other language elements in color.

BBEdit uses several methods to determine what language (if any) to use for a particular file. The primary way to activate syntax coloring in a document is simply to save it with a file name extension that indicates what programming or markup language the file contains. For example, if you save your file with ".html" at the end of the file name, BBEdit will color your HTML tags and anchors. Other common suffixes are ".tex" for TeX files and ".c" for C files.

For any file whose name does not have an extension, or whose name has an extension that does not match any of the mappings in BBEdit's Languages settings panel, BBEdit will attempt to guess what language the file contains and apply the appropriate syntax coloring. If BBEdit guesses wrong (or is unable to guess), you can resort to the Language popup in the status bar or the Language popup menu in the Text Options sheet, either of which gives you the ability to manually select *any* installed language to be applied to the document, regardless of its name. If the file is saved with "BBEdit" state, the manual language selection will persist and override any suffix mapping.

By default, BBEdit recognizes over 20 different languages and several dozen suffix mappings. You can add new suffixes to map to existing languages or (by installing third-party language modules) add syntax coloring support for new languages as well. All the specific languages that BBEdit recognizes, and the suffixes or extensions it expects for them, are listed in the Languages settings panel, and suffix mappings can also be changed there. You can choose the colors that BBEdit uses for syntax coloring in the Text Colors settings panel.

Note BBEdit will recognize and syntax-color VBScript embedded within HTML via the <%…%> and <SCRIPT>…</SCRIPT> tags.

How BBEdit Wraps Text

BBEdit wraps text in one of two ways: soft wrapping or hard wrapping.

Soft wrapping is like the word wrapping found in most word processors. When the insertion point reaches a right margin as you type, the word processor automatically moves the insertion point to the beginning of the next line. You never need to type a hard line break (that is, press the Return key) at the end of a line, but only to start a new paragraph. If you place the insertion point in the middle of a paragraph and start typing, the text reflows so that words that are pushed out beyond the right margin end up on the next line. Usually, you use soft wrapping when you are editing memos, mail messages, and other prose. It is also useful for HTML documents. With soft wrapping, you generally do not have to scroll the window horizontally to see all the text in the file.

Unlike soft wrapping, hard wrapping requires a line break at the end of every line. When soft wrapping is turned off, BBEdit lets you type as far as you like on a line, and never automatically moves the insertion point to the beginning of the next line. You have to manually type a line break to start a new line. You usually use hard wrapping to write programs, tabular data, resource descriptions, and so on. With hard wrapping, each line of source code or data appears on its own line in the window, although you may have to scroll the window horizontally to see the entire line if it is long.

Note When you use the Hard Wrap command on a rectangular selection, lines will be padded with spaces as necessary.

Tip If you open a file in BBEdit that appears to consist of a few very long lines, you should select the soft wrapping option for that file.

This table summarizes the commands to soft-wrap and hard-wrap text. The sections that follow give details about using the wrapping commands.

To do this…	Do this…
Soft-wrap text as you type	Choose Soft Wrap Text from the Text Display submenu of the View menu or select the Soft Wrap Text option from the Text Options sheet
Convert hard-wrapped text to soft-wrapped text	Use the Remove Line Breaks command in the Text menu, and activate soft wrapping
Convert soft-wrapped text to hard-wrapped text	Use the Add Line Breaks command in the Text menu
Hard-wrap text to a specific margin, reflowing paragraphs as needed	Use the Hard Wrap command in the Text menu

Note Users of very old versions of BBEdit or BBEdit Lite will note that the Wrap while Typing option (which hard-wrapped text automatically by inserting a Return when you reach the right margin) has been relegated to the dustbin of history. It has been superseded by soft wrapping.

Soft Wrapping

To turn on soft wrapping within the current document, you may do either of the following:

- Choose Soft Wrap Text from the Text Display submenu of the View menu.
- Select the Soft Wrap Text option from the Text Options sheet. (Choose Text Options from the Edit menu to open this sheet.)

You can also specify whether BBEdit should wrap text at the Page Guide, the edge of the window, or a specific character position.

IMPORTANT Soft wrapping and rectangular selection are mutually incompatible. When soft wrapping is enabled, dragging the mouse performs normal (non-rectangular) selection even if the Option key is held down.

To make soft wrapping the default for new windows, select the Soft Wrap Text option in the Editor Defaults settings panel. You can also use the settings in that panel to specify the default wrapping margin.

To "freeze" the current line endings and hard-wrap the text at the current soft wrapping settings, use the Add Line Breaks command to insert a line break at the end of each line.

While BBEdit prefers to break lines at white space when soft-wrapping, lines will be broken as close as possible to the designated wrap width if they do not contain any white space. This way, long URLs and other extended strings of characters are visible without requiring horizontal scrolling.

Soft Wrapping with Indentation

You can control how BBEdit indents soft wrapped text by means of the Soft Wrapped Line Indentation option in the Editing settings panel. Choose Flush Left to have all lines of each paragraph below the first wrap flush to the left margin of the window. Choose First Line to have all subsequent lines of a paragraph wrap to the same indent level as its first line. Choose Reverse to have all subsequent lines of each paragraph wrap indented one level deeper than its first line.

Exporting Soft-Wrapped Text

BBEdit will not insert hard line breaks into softwrapped files upon saving them. If you wish to add hard line breaks to a softwrapped file, use the Hard Wrap or Add Line Breaks command.

Soft Wrapping in Browsers

Use the Text Options command from the Edit menu to control soft wrapping (and other display options) for files viewed in a browser window.

Soft Wrapping and Line Numbers

The settings Use "Hard" Lines in Soft-Wrapped Views controls how line numbers are displayed when you use soft wrapping. If this option is turned on, the line number bar,

display, and Line Number commands in editing views will use line numbers that correspond to "hard" line breaks in the document, rather than to soft-wrapped line breaks. To restore the behavior of previous versions of BBEdit, turn this settings off.

Hard Wrapping

The easiest way to hard-wrap text is to type a line break (by pressing the Return key) whenever you want to start a new line. If you are editing program source code, it is generally best to turn off soft wrapping altogether.

To turn off soft wrapping for the active window, do one of the following:

- Choose Soft Wrap Text from the Text Options popover in the navigation bar.
- Deselect the Soft Wrap Text option from the Text Options sheet box by choosing Text Options from the Edit menu.

To turn off soft wrapping for new windows, deselect the Soft Wrap Text option in the Editor Defaults settings panel.

BBEdit provides two ways to convert soft-wrapped text into hard-wrapped text. The first is a simple technique that uses a single command; the second is a bit more complicated but gives you much more control over wrapping.

Hard-Wrapping Soft-Wrapped Text

To convert soft-wrapped text to hard-wrapped text, use the Add Line Breaks command in the Text menu. This command inserts a line break at the end of every line of the text as it appears in the window. If your wrapping margin is the edge of the window, you will get different results depending on the width of the window.

If the current document contains a selection range, Add Line Breaks will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

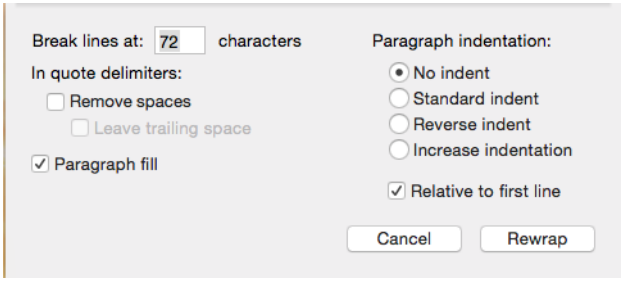
Note The Add Line Breaks command does not turn off soft wrapping.

Hard Wrapping and Filling Text

The Hard Wrap command in the Text menu offers more flexibility for hard-wrapping text than the Add Line Breaks command. Whereas Add Line Breaks merely “freezes” the line breaks displayed in a document by inserting line breaks, the Hard Wrap command allows you to wrap text to any arbitrary width, while also reflowing or indenting paragraphs.

If the current document contains a selection range, Hard Wrap will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

When you choose the Hard Wrap command, BBEdit opens a sheet in the frontmost document:



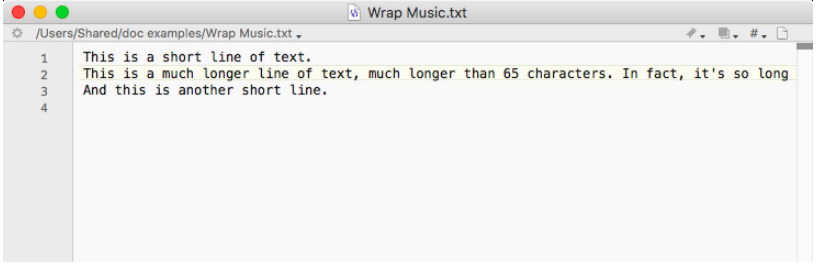
The controls in the left half of the sheet control the maximum width of lines after hard wrapping, how to treat quote delimiters (if present), and whether wrapped lines should be consolidated to fill paragraphs to the specified width. The controls in the right half determine how paragraphs should be indented.

The “Break Lines at” setting let you specify the wrapping margin.

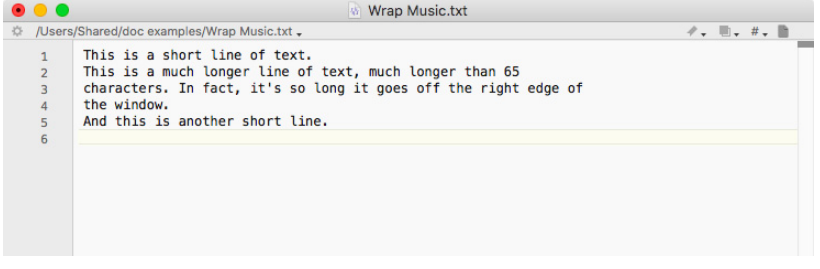
If the text contains Internet-style quotes (one or more “>” characters at the beginning of each line) and the “Remove spaces” option is selected, BBEdit will remove the leading and trailing spaces from each line, or if the “Leave trailing space” option is also selected, BBEdit will remove the leading spaces while leaving trailing spaces.

If the Paragraph Fill option is selected, BBEdit forms the lines into paragraphs before wrapping the lines. An example is the best way to illustrate this option.

Suppose you start with this text:

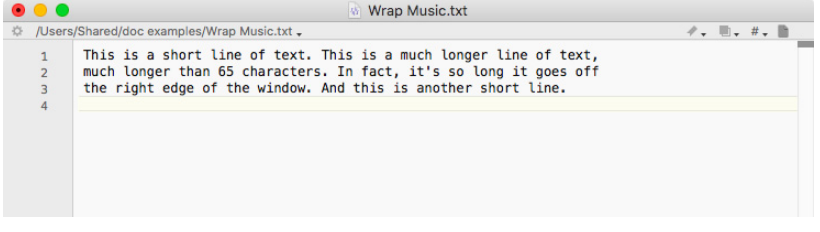


This is what happens when you wrap to 65 characters with Paragraph Fill off:



BBEdit breaks the long line at a width of 65 characters (twice, because the line was so long) and leaves the short lines alone.

This is what happens to the same text when you wrap with Paragraph Fill on:



BBEdit joins all the lines together to form a single paragraph and then wraps the text to a width of 65 characters.

The Paragraph Indentation buttons let you indent paragraphs after they have been wrapped.

- Flush Left does not indent paragraphs at all.
- First Line indents all lines in the paragraph by one tab stop.
- Reverse places the first line in the paragraph flush against the left edge of the window and indents all other lines in the paragraph by one tab stop.

Mark the Relative to First Line checkbox to make any paragraph indents relative to the original indent of the first line of the selection or document. If you want paragraph indents to be relative to the left margin of the document, make sure this checkbox is not marked.

Click the Wrap button to perform the Hard Wrap command, or cancel to dismiss the sheet.

Tip If you hold down the Option key as you choose the Hard Wrap command, BBEdit uses the last Hard Wrap settings to perform the operation, without displaying a sheet.

The Insert Submenu

In addition to typing, you can use the commands in the Insert submenu of the Edit menu to insert text into the active window. These commands let you insert the contents of other files, folder listings, page break characters, time stamps, and Emacs variable blocks.

Inserting File Contents

The File Contents command inserts the contents of one or more files into the document you are editing. When you use this command, BBEdit displays an Open sheet in which you can choose the files to insert. To select more than one file hold down the Shift key or Control key as you click the files. BBEdit then inserts the contents of the selected files at the insertion point or replaces the selected text. If you select more than one file, the files will be inserted in alphabetical order, according to file name.

If you click the Options button in the lower left corner, the Open sheet will also offer two options named “Include separators” and “Ensure line break after each inserted file”. If you enable the former option, BBEdit will include a separator which consists of a dashed line and the file's name between each inserted file's contents, while the latter option will do just what its name implies.

IMPORTANT There is a bug in the macOS file selection panel which can prevent some files from being inserted and it is not currently possible to correct this problem within the application itself.

One workaround is to scroll through the entire list of files in the panel before selecting and inserting them; this will allow the OS to correctly provide each selected file to BBEdit when you apply the File Contents command.

Alternatively, you may select the desired files in a Finder window and then drag & drop those files into an empty editing window.

Inserting File & Folder Paths

The File/Folder Paths command inserts the full path information for the selected files and folder into the document you are editing. When you use these commands, BBEdit displays a sheet that lets you select the files and/or folders. BBEdit inserts the path information at the insertion point or replaces the selected text.

Inserting a Folder Listing

The Folder Listing command inserts a textual listing of a folder hierarchy. When you use this command, BBEdit displays a sheet that lets you select a folder to insert, and options to control whether the generated listing should be hierarchical (the original style), or flat (which lists the full path of every item) and whether to include hidden (invisible) items. These option settings will remain selected until you change them, and will also apply if you drag a folder into a document.

Tip You can also drag a folder's icon from the Finder into a document to insert a folder listing.

Inserting a Page Break

To insert a page break, choose the Page Break command from the Insert submenu of the Edit menu. This will place a form feed character (ASCII 12) at the location of the insertion point. BBEdit uses this character to indicate the start of a new page when printing.

Inserting Time Stamps

To insert the current time, choose Short Time Stamp or Full Time Stamp from the Insert submenu of the Edit menu. These commands will insert short and long forms (respectively) of the current date and time at the location of the insertion point.

Inserting an Emacs Variable Block

To insert an Emacs variable block describing the option settings for the current document, choose Emacs Variable Block from the Insert submenu of the Edit menu. This will bring up a sheet which you can use to review and confirm the desired options. (Since depending on what options are set, the resulting block can be rather verbose, you may wish to prune the resulting text.)

These options specified in this block will take precedence over saved document state when BBEdit opens the document. (Inserting these explicit settings can be useful when sharing the document with others.

Inserting Lipsum

To insert any desired quantity of "lipsum" placeholder text, choose the "Lorem Ipsum" command from the Insert submenu of the Edit menu. This will bring up a sheet which you can use to confirm or set the desired options.

`https://en.wikipedia.org/wiki/Lorem_ipsum`

Comparing Text Files

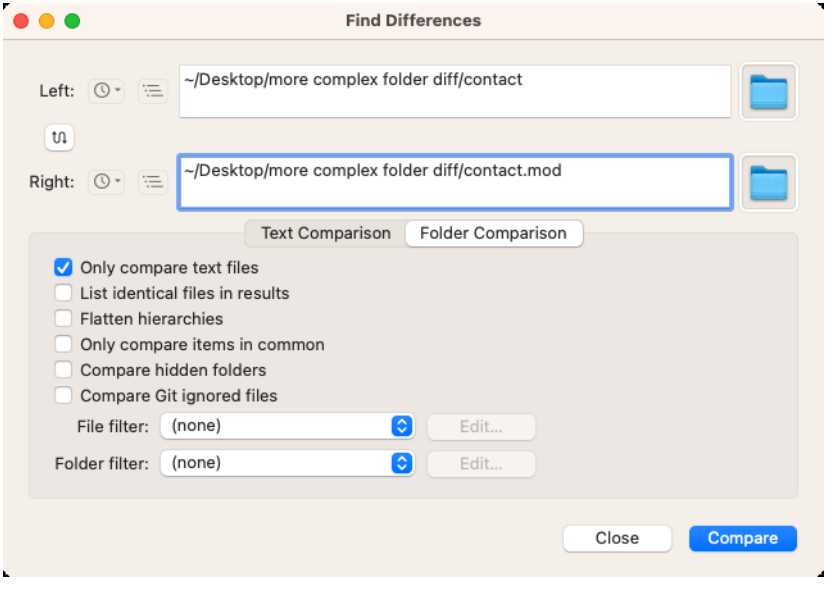
If you have ever had to reconcile changes between two different versions of a file, or even larger numbers of documents, you know how laborious this task can be. BBEdit's Find Differences command is a powerful tool for doing such comparisons faster and more effectively. Using Find Differences, you can compare any two files, or the contents of two folders. You can also specify options to eliminate minor variations in document content, such as different amounts of white space, from being considered.

If you have two or more text documents open, choose the Compare Two Front Windows command on the Search menu to quickly compare the topmost two documents. (BBEdit will always place the topmost window's document on the left, so changing the placement of the documents being compare is as simple as changing the front-to-back order of the windows involved.)

To compare two arbitrary files or folders:

1 **Choose the Find Differences command from the Search menu.**

BBEdit opens the Find Differences window



2 **Use the Left and Right popup menus to select the files (you want to compare.**

If the files you want to compare are already open, they will appear in the popup menus; otherwise, you can select them by clicking the standard item selection button next to either popup menu, or by dragging the icon of any desired file or folder icons from the Finder into the Left or Right field.

You can also select recently opened files from the Recent Files item on the Left and Right popup menus, or drag files (or folders) from a project or editing window's sidebar, title bar, or navigation bar, or from the Finder, into the "Left" and "Right" path fields or their adjacent image wells.

The image well to the right of each path shows a file (or folder) icon if the path refers to an item on disk; if the item indicated by the path does not exist on disk (i.e. an unsaved document), the image well will instead display an alert icon.

When choosing a file (or folder), the selection sheet presents a “Show hidden items” option behind the “Options” button.You can enable this option to more easily select invisible files (or folders).

3 **Select the Compare options that apply.**

When the Case Sensitive option is selected, BBEdit distinguishes uppercase from lowercase letters; deselect this option if you want BBEdit to consider uppercase and lowercase letters the same.

When Ignore Curly Quotes is selected, BBEdit treats typographers’ quotes the same as straight quotes.

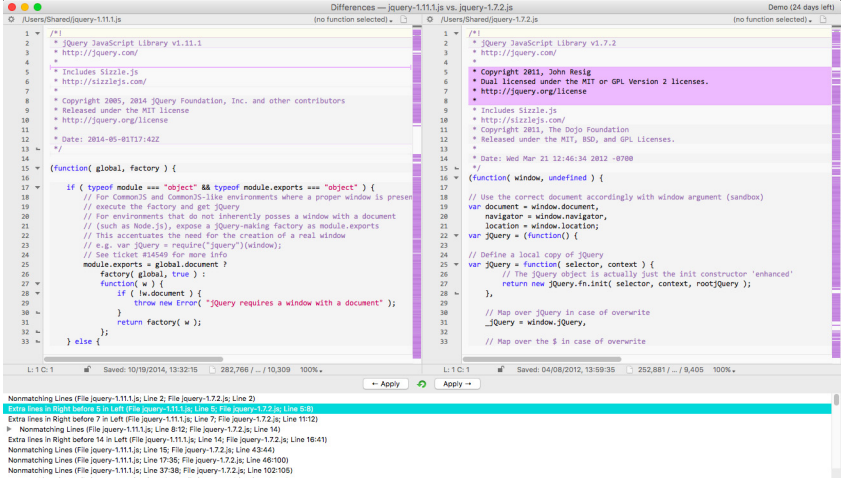
When Ignore Blank Lines is selected, BBEdit will skip all differences that consist entirely of blank (empty) lines.

When one or more of the Ignore Spaces options is selected, BBEdit will ignore the corresponding presence of whitespace at the specified positions while comparing files, or if the “All” option is selected, BBEdit will ignore all whitespace differences.

4 **Click Compare to perform the comparison.**

Alternatively, you can use the ‘bbdiff’ command line tool to specify two files (or folders), and have BBEdit perform a Find Differences on them.

If the two documents are different, BBEdit opens a Differences window which contains both documents.



The Differences window lists all the differences between the left-hand and right-hand documents. To see the differences in context, click a line in the Differences window; BBEdit scrolls and selects that spot in both files.

The range of lines within each file which belong to the selected difference are highlighted with the current color scheme’s “Differences” color, while all other differences within file are drawn with a light grey background.

Reviewing and Applying Differences

To view and apply individual differences within a line or region (i.e. sub-line differences), just click on the triangle to the left of that difference to expand the list and select the appropriate character difference.

Use the Apply to Left and Apply to Right buttons in the Differences window to transfer the differing text from the new file to the old file, or vice versa. After you use one of these buttons, BBEdit italicizes the entry in the Differences window to indicate that you have already applied that change.

You may also apply all differences by clicking in the differences list, then choosing Select All in the Edit menu, and using the Apply to Left or Apply to Right button to apply the differences to the desired file.

If a Differences window is open and is the frontmost window, you can select the Compare Again command in the Find Differences submenu of the Search menu to have BBEdit recompare the two selected files and refresh the list of differences accordingly.

Alternatively, you can click the Compare Again button (with the circular arrow icon) between the Apply to Left and Apply to Right buttons to perform the same function.

Preserving a List of Differences

When a Differences window is active, you can save the currently listed differences to a plain text file by using the Export command in the File menu.

Comparisons by Other Means

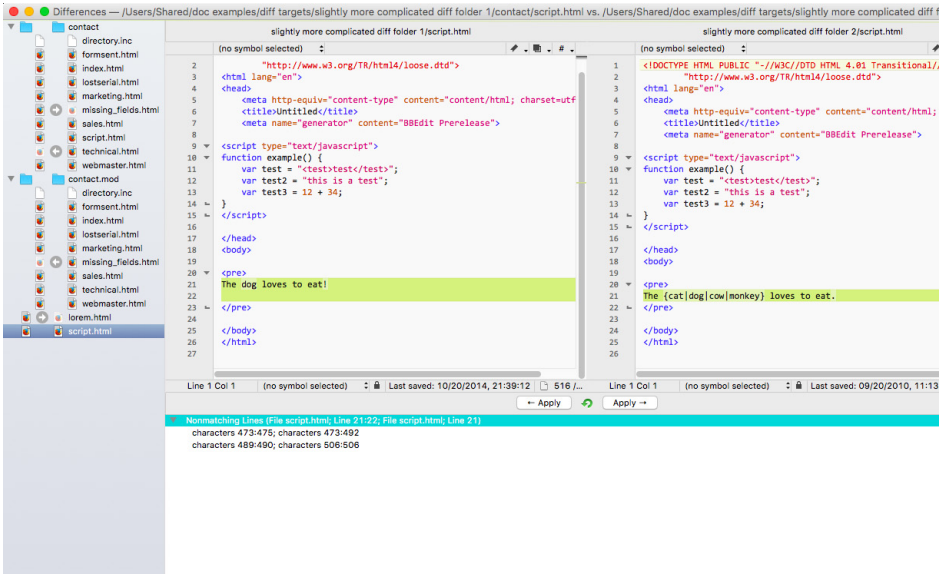
Comparisons performed by means other than the Find Differences dialog will use the settings currently specified in that dialog, rather than reverting to factory defaults. These means include the contextual menu Compare command, the Compare Against Previous Version and Compare Against Disk File commands, and various forms of source control revision comparisons.

Compare Against Disk File

You can use the Compare Against Disk File command to compare the contents of the active document against the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Multi-File Compare Options

You can compare multiple files at once by selecting two folders in the Find Differences dialog; BBEdit lists all the files in each folder, and displays a directional arrow icon to denote files which exist only in one folder.



You can select any file pair to view their differences (as for a single pair of files), or for files which exist only in one folder, click the arrow icon to copy the existing file into the corresponding location in the second folder. In addition, you can click any file's icon to ask the OS to open it, or Option-click to reveal that file in the Finder, or Command-click to open that file into a separate editing window within BBEdit.

When performing a multiple file comparison, you can specify the additional options described below.

List identical files

Normally, when you compare folders using the Find Differences command BBEdit will display all the files present within each folder in the sidebar of the differences window, including pairs of identical files which have an equal sign "=" between them for ease of identification.

If you are comparing very large folders, however, the large number of identical file pairs can make the differing pairs hard to find. When you deselect the List Identical Files checkbox, BBEdit will display only differing and/or unpaired files in the sidebar listing.

Flatten hierarchies

Normally, BBEdit retains the hierarchy of the files being compared in a folder. In other words, when comparing folders, it looks in each subfolder of the first folder you select and tries to match it with a file of the same name in the same subfolder of the second folder, and so on down for all subfolders. If you choose Flatten Hierarchies, BBEdit considers the files in the folders as a single flat list, allowing a file in one folder to match a file of the same name in the other folder, regardless of whether they are in the same subfolder in both hierarchies.

Only compare items in common

If this option is set, BBEdit will only list items in the results that exist in both of the folders being compared. (This option is also available to the scripting interface.)

Compare hidden folders

If this option is set, BBEdit will examine the contents of invisible folders when comparing folders.

Only compare text files

If this option is set, BBEdit does not include non-text files when comparing folders.

Use file filter

File filters allow you to select files for comparison with great precision. If either file in a compared pair matches the filter, the files are eligible for comparison; if *neither* file matches the filter, the files will not be compared. See Chapter 7, "Searching," for more information on creating, editing, and using file filters.

Note When comparing folders with the Find Differences command, BBEdit applies any specified file filter to the contents of the resulting "Only in new" and "Only in old" lists, so that only those files that match the filter criteria will appear in the lists.

Use folder filter

Folder filters allow you to select (or exclude) the contents of specific folders for comparison. See Chapter 7, "Searching," for more information on creating, editing, and using folder filters.

Using Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Mark popup menu to move quickly to the specified section of the file. (The navigation bar must be visible in order to access the Mark popup menu. Choose Show Navigation Bar from the View menu to display the Navigation bar if it's hidden.)

Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if BBEdit supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu in the document window.

Setting Markers

To set a marker:

1. **Select the text you want to mark.**
2. **Choose the Set Marker command from the Mark popup menu, or Control-click the selected text and choose Set Marker from the contextual menu.**

   BBEdit opens a sheet so that you can name the marker. If you have selected a range of text, the sheet will contain the first characters of the selection.
3. **Click Set to set the marker.**

Tip If you hold down the Option key as you choose Set Marker, BBEdit sets the marker using the leading characters of the selected text as the name of the marker, without displaying a dialog box.

Clearing Markers

To clear a marker:

1. **Choose the Clear Markers command from the Mark popup menu.**

   BBEdit displays the list of markers.
2. **Select the marker you want to delete.**
3. **Click Clear to clear the marker.**

BBEdit also offers a Clear All Markers command, which clears all the markers in the document in one fell swoop. You can access this command by holding down the Option key and using the Mark popup menu.

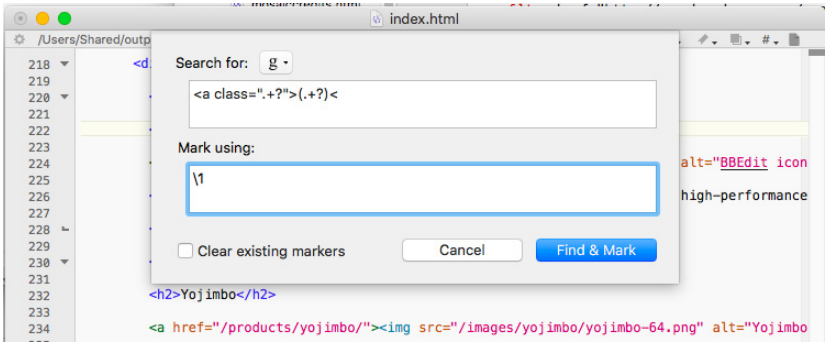
Using Grep to Set Markers

You can use the Find & Mark All command in the Mark popup menu to mark text that matches a grep pattern. To learn more about using grep patterns, see Chapter 8, “Searching with Grep.”

To use a grep pattern to mark text:

1 **Choose the Find & Mark All command from the Mark submenu.**

BBEdit opens the Find & Mark All sheet.



2 **Type the pattern in the Search For field and the marker names in the Mark With field.**

You can also choose stored patterns from the Patterns popup menu.

3 **Click Find & Mark to mark the matching text.**

BBEdit searches the current document for text that matches the pattern and marks it the way you specified.

Speaking & Spell Checking Text

Speaking Text

The Start Speaking command in the Edit menu will speak the selected text in the document using the system's current speech settings, or if there is no selection range it will speak the entire document.

If the frontmost document is empty, this command will be disabled. When speaking is in progress, this command reads "Stop Speaking" and selecting it will stop the in-progress speech.

Spell Checking Text

The commands in the Edit menu's Spelling submenu let you check the spelling of the text in your documents using the system's built-in spelling checker.

Check Spelling As You Type

To have BBEdit automatically check spelling as you type for the current document, select Check Spelling as You Type in the Spelling submenu. To have BBEdit always check spelling as you type, turn on the corresponding option in the Editor Defaults settings panel.

When BBEdit encounters a word which is either misspelled or not in the checker's dictionary, it will draw a heavy red underline beneath the word. You can either type a correction, or Control-click on the word and select a suggested correction from the contextual menu.

To skip the identified word and continue checking, use the Check Spelling command again. To ignore all further instances of the word, Control-click on it and choose Ignore Spelling from the contextual menu. To add the word to the dictionary, Control-click on it and choose Learn Spelling from the contextual menu.

Manual Spell Checking

Choose the Find Next Misspelled Word command from the Spelling submenu, or type its key equivalent (Command-;) to start checking a document's spelling. BBEdit will check every word in the document in order, starting from the current insertion point.

To check the spelling of all words in the document at once, choose the Find All Misspelled Words command, or type its key equivalent (Option-Command-;). BBEdit will draw an underline under every questioned word in the document. You can then correct the spelling of any questioned word by typing, or by using the contextual menu to select a suggested correction or to skip, ignore, or add the word to the dictionary.

To clear the underline from all questioned words, choose the Clear Spelling Errors command.

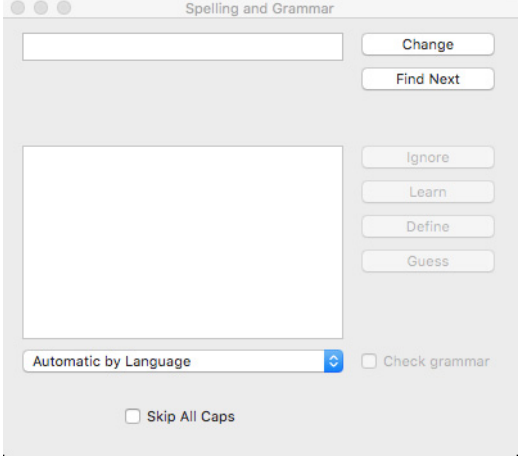
To remove any previously learned word from the dictionary, place the insertion point within that word, then Control-click (or right-click) on that word and select the Unlearn Spelling command.

The Spelling Panel

In addition to allowing you to correct, ignore, or learn identified words, the Spelling panel allows you to choose what spelling behavior or specific dictionary the spelling service should employ. To use the Spelling panel:

1 **Choose the Show Spelling Panel command from the Spelling submenu.**

BBEdit opens the Spelling panel.



2 **Set spelling options.**

Choose a dictionary to use by selecting it from the Dictionary popup menu. Select Skip All Caps to avoid checking words consisting of only capital letters. (Note that these settings persist across runs of the application.)

3 **Click Find Next to begin checking.**

BBEdit scans the document, and stops at the first misspelled or unrecognized word. This word is displayed in the text field to the left of the Correct button. Possible corrections for the questioned word are listed in the Guess box above.

4 **If the questioned word is misspelled, choose the correct spelling from the Guess list or type it yourself in the Correct field.**

5 **Click one of the Spelling panel's action buttons to handle the questioned word.**

Click Ignore to ignore further instances of the questioned word, without adding it to the active dictionary.

Click Guess to display a list of possible corrections.

Click Find Next to ignore this instance of the questioned word and continue checking.

Click Correct to replace this instance of the questioned word with the text in the adjacent text field.

Click Learn to add the questioned word to the active dictionary.

Writing Tools (macOS Text Processing Commands)

BBEdit supports for the “Writing Tools” feature introduced in macOS 15.2 as part of the Apple Intelligence suite. Thus, the corresponding commands on the Writing Tools submenu of the Edit menu are operational.

CHAPTER

5

Text Transformations

This chapter describes the range of powerful text transformation commands offered by BBEdit. Beyond applying individual commands to the current document, you can define and save Text Factories, which are sequences of commands that can be applied to one or more documents.

In this chapter

Text Menu Commands

BBEdit provides a variety of commands which you can use to transform text in different and useful ways. Most of these commands are situated in the Text menu, and described in this section. You can also use BBEdit's built-in search and replace capabilities to transform text, as detailed in Chapters 7 and 8 of this manual.

Unless otherwise specified, each of these commands will be applied to the active text selection in the frontmost document range, or if there is no active selection, to the entire contents of the document.

Hold down the Option key when selecting any command from the menu in order to quickly re-invoke it with its last-used option settings. (These “short form” commands are also available in the Menus & Shortcuts settings panel, so that you can set key equivalents for them.)

Apply Text Filter

When you choose this command, BBEdit will present a submenu listing all currently available text filters. (These filters consist of all suitable items contained in the Text Filters subfolder of BBEdit’s application support folder. See “Text Filters” on page 40.)

When you choose a filter, BBEdit will pass either the selected text (or the contents of the active document, if there is no selection) on STDIN to Unix executables or filters, as a string to text factories, as a reference to a ‘RunFromBBEdit’ entry point in AppleScripts, as text input to Automator workflows, and as a source to text factories. (An AppleScript script intended for use as a text filter must have a ‘RunFromBBEdit’ handler.)

AppleScript scripts and Automator workflows should return a string which BBEdit will use to replace the selection range, Unix filters should write to STDOUT, and the text emitted by a text factory will replace the selection range.

Apply Text Filter <last filter>

When you choose this command, BBEdit will reapply the most recently used text filter.

Apply Text Transform

When you choose this command, BBEdit will open a dialog which allows you to choose any text factory action, and apply that action directly to any desired set of files and/or folders.

You may also click the “Save as Text Factory...” button to save the chosen action and options directly as a text filter file.

Run Unix Command

When you choose this command, BBEdit will open a sheet into which you can enter a Unix command, and choose where the output goes; either replacing the frontmost document’s selection (or contents) as a text filter would, or placing it into a new document.

This can be extremely useful for applying quick (or simple) “one shot” Unix commands that aren't worth the effort of writing and saving a text filter.

As with a text filter, BBEdit will provide the current selection’s contents on standard input, or if there is no selection, then it will instead provide the entire contents of the frontmost document. This command will also remember previously-used commands, which you can select in the usual history menu adjacent to the text field.

This panel now also provides a “Save as Script” or “Save as Text Filter” button to save a script (or text filter, depending on the selected output option) directly into the “Scripts” or “Text Filters” subfolder within BBEdit’s app support support folder. (You may also save the file to any other location you desire, but those are the locations of greatest immediate relevance.)

Run Unix Command <recent command>

When you choose this command, BBEdit will re-run the selected recently-used Unix command.

Exchange Characters

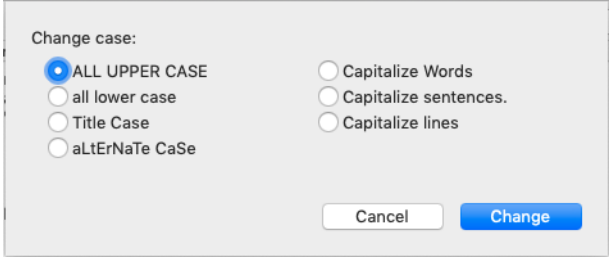
This command swaps two characters according to the following rules:

- If there is no selection and the insertion point is not at the beginning or end of a line or of the document, this command transposes the two characters on either side of the insertion point.
- If the insertion point is at the beginning of a line or document, this command transposes the two characters following the insertion point.
- If the insertion point is at the end of a line or document, this command transposes the two characters before the insertion point.
- If there is a selection, this command transposes the characters at *either end* of the selection.

If you hold down the Option key as you choose this command, Exchange Characters becomes Exchange Words. Exchange Words behaves like Exchange Characters except that it acts on entire words rather than individual characters.

Change Case

This command lets you change between uppercase and lowercase characters, or capitalize word, line, or sentence starts. When you choose the Change Case command, the following sheet appears:



The radio buttons let you choose how to change the case of the text. The following table explains the function of each option in this dialog.

This button…	Changes the text like this…
ALL UPPER CASE	Every character changes to uppercase.

This button…	Changes the text like this…
all lower case	Every character changes to lowercase.
Title Case	Applies several different transforms in order to produce a valid English language title.
Alternate Case	Makes the first letter of the selected range lowercase and then alternates between upper and lower case to the end of the processed range, resetting at sentence boundaries. (This format is sometimes known as "sPoNgEcAsE".)
Capitalize Words	The first character of every word changes to uppercase; all other characters change to lowercase.
Capitalize sentences	The first character of every sentence changes to uppercase; all other characters change to lowercase.
Capitalize lines	The first character of every line changes to uppercase; all other characters change to lowercase.

In addition to using the Change Case sheet, you can also select individual case change actions from the Change Case submenu immediately below the Change Case... command.

Shift Left / Shift Right

These commands indent or outdent the selected text by one tab stop. If there is no selection, this command works on the current line. Hold down the Shift key while choosing these commands, to have BBEdit indent or outdent the text by one space instead of one tab stop.

BBEdit also converts spaces to tabs (or vice versa) on the fly as you shift text. For example, if the selected text is indented one tab stop and you apply Shift Left One Space, the tab will be converted to spaces and the text will be outdented one space. If you then apply Shift Right One Space, the spaces will be converted back to a single tab.

Note If the "Auto-Expand Tabs" option is enabled, Shift Left/Right will not insert literal tabs but will instead insert (or remove) space runs of the appropriate width.

Un/Comment Lines & Un/Comment Block

These commands allow you to selectively comment and uncomment sections of code in various programming languages, using line or block comments respectively. Choose a range of text and apply the desired command to add or remove line (or block) comments to that text, depending on its initial comment state. If there is no selection, these commands are disabled.

If you apply Un/Comment Lines and **any** lines within the selection range are uncommented, then BBEdit will prefix all lines in the selection range with the current language's line comment delimiter -- including lines which are already prefixed. If instead **all** lines are prefixed with the line comment delimiter, then BBEdit will remove the first occurrence of the line comment delimiter on each line, thus returning everything to its previous state. This behavior is consistent with the "toggle comments" behavior of Xcode and others.

The one condition under which BBEdit will automatically override your chosen command is if you attempt to apply line comments to text whose language type does not include them (e.g. HTML or XML), or block comments to text whose language does not support them (e.g. Perl or CSS). In that situation, these commands will behave identically and apply the appropriate comment format.

You can use the Options sheet of the Installed Languages list in the Languages settings panel to modify or set comment strings for any available languages.

Note If you have set custom comment delimiters for HTML in the Languages settings panel, those delimiters will be honored when you use the Un/Comment command. However, they will not affect the operation of the HTML-specific comment commands on the Markup menu.

Hard Wrap

This command wraps long lines by inserting hard line breaks and can reflow (fill) paragraphs if desired. See "How BBEdit Wraps Text" on page 124 for more information.

Add Line Breaks

This command inserts a hard line break at the end of each line of text as displayed. See "How BBEdit Wraps Text" on page 124 for more information.

Remove Line Breaks

This command removes line breaks (line feeds) and spaces from sections of text. Use this command to turn text that has hard line breaks into text that can be soft-wrapped. See "How BBEdit Wraps Text" on page 124 for more information.

Convert to ASCII

This command is no longer present; instead, the Zap Gremlins command using the "Replace with code" and "Use ASCII equivalent" performs the same conversions.

Educate Quotes

This command converts straight quotes (" and ') to typographer's quotes (“ ” and ‘ ’).

Note You should not use this command to prepare text for use in a web page or an email, as typographer's quotes in the Mac character set will generally not be properly displayed by applications on other platforms.

Straighten Quotes

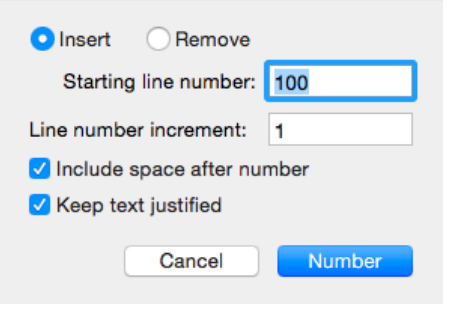
This command performs the reverse of Educate Quotes; it converts typographer's quotes (“ ” and ‘ ’) to straight quotes (" and ').

Reformat Document/Selection

This command is enabled when either the current document's language module supports reformatting intrinsically, or if the language module has a configured running language server which reported the 'documentFormattingProvider' capability. Choosing this command will supply the document's entire contents (or the contents of the current selection) to the formatter, which in turn will do whatever violence to the text it deems appropriate.

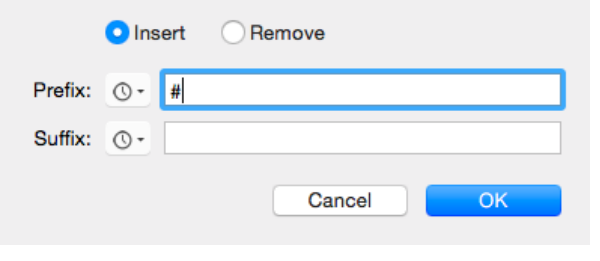
Add/Remove Line Numbers

This command displays a sheet which allows you to add or remove line numbers for each line of the selected text or of the document. You can set the starting number and numbering increment, as well as whether to include a trailing space, and whether to right-justify the inserted numbers, by choosing the appropriate options.



Prefix/Suffix Lines

This command displays a sheet which allows you to insert (or remove) the specified prefix and/or suffix strings on each line of the selected text or of the document.

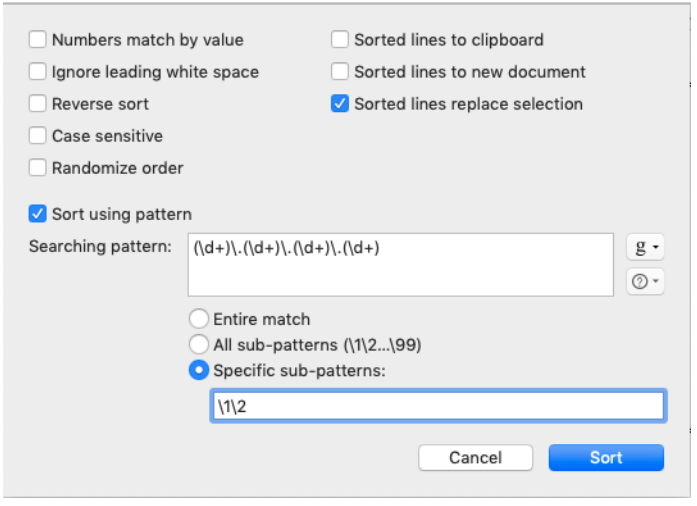


If you define both a prefix and a suffix string, BBEdit will apply them to the text at the same time.

Note When using the “add prefix”, “add suffix”, “remove prefix”, or “remove suffix” scripting commands, the string direct parameter is required.

Sort Lines

This command displays a sheet which allows you to sort lines of text in Unicode collation order. The sorted lines can be copied to the clipboard, be displayed in a new untitled window, replace the selection within the original document, or any combination of the three.



There are also options for ignoring white space at the beginning of lines, taking case distinctions into account, sorting strings of digits (integers) by numerical value instead of lexically, sorting in descending rather than ascending order, or sorting randomly.

IMPORTANT If you need to sort lines in strict character code order (e.g. in order for case sensitivity to take precedence), you may do so by turning on the “Case sensitive” option and turning off the “Numbers match by value” option.

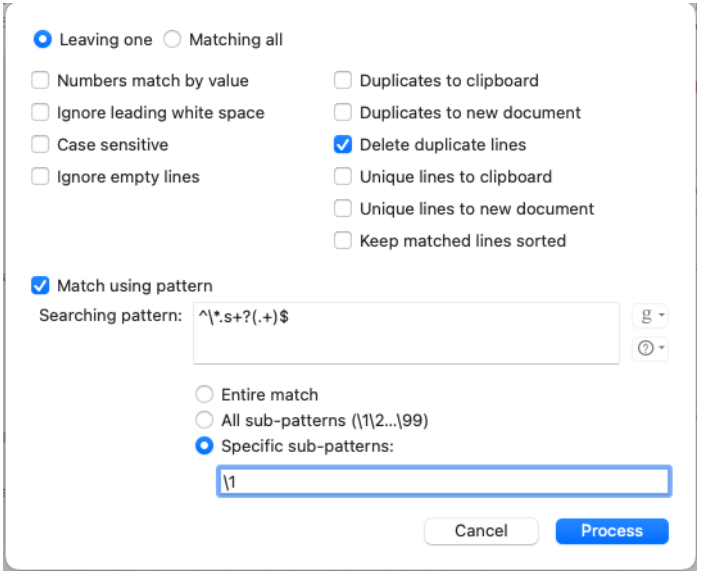
By checking the Sort Using Pattern option, you can specify a grep pattern to further filter the lines to be sorted. If the pattern contains subpatterns, you can use them to control the sort order based on the contents of the strings they match. When you sort using a grep pattern, the Case Sensitive option controls the case sensitivity of the pattern match in the same manner as the equivalent option in the Find dialog.

For example, suppose you are sorting a list of cities together with their two-letter state abbreviations, separated by a tab character. The pattern and subpatterns shown in the figure will sort the results first by city name and second by state abbreviation. Changing the contents of the Specific Sub-Patterns field from “\1\2” to “\2\1” will instead sort the results by state first and by city second.

IMPORTANT When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start ^ and line end $ operators as necessary. Also, keep in mind that the pattern will only be tested against a single line at a time. So, if the pattern matches one or more sets of multiple lines within the document, but does not match any individual lines, BBEdit will not sort the contents of the document.

Process Duplicate Lines

This command displays a sheet which allows you to locate duplicate lines within a body of text and operates on them in various ways.



The Matching All option processes all duplicate lines; Leaving One ignores the first of each set of duplicate lines and processes only the additional ones.

The Numbers Match by Value and Ignore Leading White Space options allow you to choose whether strings of digits should be evaluated numerically or compared as strings, and whether white space at the beginnings of lines should be considered.

The Match Using Pattern option allows you to use a grep pattern to further filter the lines to be processed. You can enter a pattern in the Searching Pattern field, or choose a stored pattern from the popup menu. The Match Using: radio buttons control what part of the specified pattern should be used to determine duplication.

IMPORTANT When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start ^ and line end $ operators as necessary.

The options on the right-hand side of the sheet allow you to specify how duplicate lines should be handled once they have been identified. You can copy duplicate lines to the clipboard (Duplicates to Clipboard), copy them to a new document (Duplicates to New Document Window), and/or delete them from the current document (Delete Duplicate Lines). You can likewise specify how to handle the lines that are *not* duplicated by choosing Unique Lines to Clipboard and/or Unique Lines to New Document).

Additionally, you may choose the option to “Keep matched lines sorted” which does essentialy what its name indicates, namely that after processing, any remaining lines will be sorted according to the various sorting/matching criteria specified elsewhere in the options. This can be useful for saving a step in cases where the existing order of the input lines is not important, and you need for the de-duplicated lines to be sorted in order.

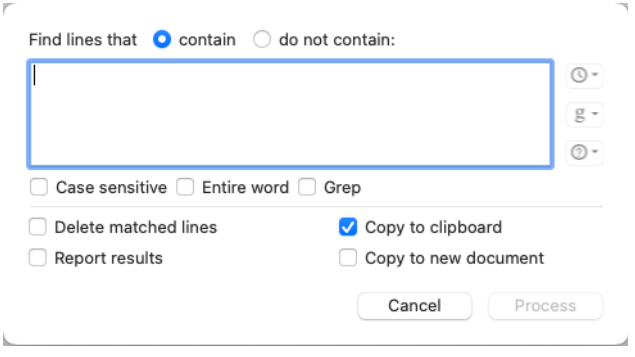
Since each of these options is an independent checkbox, you can select any combination of them that you wish. For example, selecting both Delete Duplicate Lines and Unique Lines on Clipboard would delete the duplicate lines from the document and copy them to the clipboard for pasting elsewhere.

Additionally, Process Duplicate Lines gets a new option to include duplicate counts in the duplicated-lines output. Each duplicate line will be prefixed by the number of duplicates that were removed.

Finally, when using Process Duplicate Lines and the "Leaving one" option is selected, BBEdit will preserve the first occurrence of the duplicated line, and subsequent ones will be discarded. This provides some protection against the vagaries of the system Unicode collator.

Process Lines Containing

This command displays a sheet which allows you to search the active window for lines containing a specified search string and then removes those lines or copies them to the clipboard. The options above the text field control how the search is performed and the options below the text field control what happens to lines which contain matches.



To specify a search pattern, enter it in the Find Lines Containing field. If you do not want BBEdit to match text when the letters in the text differ from the letters in the search string only by case (upper-case versus lower-case), select Case Sensitive.

To search using a grep pattern, select Use Grep and enter the pattern in the text field. You can also select a predefined search pattern from the Patterns popup menu.

Note If the selection ends on a trailing line break, BBEdit will omit that line break from the search string copied into the text field.

The checkboxes on the right of the sheet control the way lines containing the specified search pattern will be processed. By selecting the appropriate combinations of these options, you can achieve the effect of applying various editing commands to each line:

- Setting both Copy to Clipboard and Delete Matched Lines on is equivalent to applying the Cut command.
- Setting Copy to Clipboard on and Delete Matched Lines off is equivalent to applying the Copy command.

- Setting Copy to Clipboard off and Delete Matched Lines on is equivalent to applying the Clear command.

The Copy to New Document option opens a new, untitled document containing copies of all lines matching the search pattern, whether or not they are deleted from the original window. By using this option and turning Copy to Clipboard off, you can collect all matching lines without affecting the previous contents of the clipboard.

The Report Results option causes BBEdit to display a dialog reporting the total number of lines matched, regardless of their final disposition. With all of the other options turned off, this can be useful for pretesting the extent of a search operation without affecting the clipboard or the contents of the original window.

Finally, you can invert the match test, such that this command will collect all lines which do **not** match the provided search string or pattern by selecting the "do not contain" option (or if scripting, by providing the 'inverting test: true' parameter).

Remove Blank Lines

This command allows you to remove all blank (lines containing only whitespace) and empty (no characters between line breaks) lines from the document being processed, without needing recourse to a grep pattern or trying to mis-apply the Process Lines Containing command.

Canonize

This command allows you to perform batch search and replace operations which are governed by a pre-defined file. The transformation file is itself a list of paired search and replace strings or grep patterns, one pair per line. Each search string (or pattern) is separated from its replace string (or pattern) by a literal tab, so if you want your searches or replacements to match or insert tabs, you must use the special character "\t" within the corresponding strings.

Blank lines are allowed, and transformation files may be commented: all the text on a line which follows a literal hashmark character"#" will be ignored so if you want your search or replace strings to match or insert hashmarks, you must escape them by prepending a backslash:"\#".

There is one case in which a line may contain only a single string, namely when you wish to case-insensitively normalize the spelling of all occurrences of a word.

Here is an example transformation file:

```
void    VOID
MyAncientClassName  MyModernClassName   # class rename for the new world order
# this line is a comment
\#include   \#import    # convert from C++ to Objective-C
noErr   # normalize capitalization from "noerr", "NOERR", "NoErr", etc
```

When using the Canonize command, you can specify whether the search is to be case-sensitive or not; and whether the searches should match on word boundaries, or not. (These behave identically to the "Case Sensitive" and "Match Words" options in the Find and Multi-File Search windows.)

Note Even when you specify that the Canonize command should be "Case Sensitive", when BBEdit encounters a single word on a line, it will perform a case-insensitive search for that word, since the purpose of this construction is to normalize all occurrences of the word to use the same (provided) capitalization.

Canonize Using Grep Patterns

You can now use transformation files which contain grep patterns with the Canonize command, by either enabling the "Grep" option in the "Canonize" dialog, or by adding a suitable mode line to the transformation file.

There are some restrictions to be aware of when the "Grep" option is in use: all lines must contain two columns (except for blank lines or comment lines); and the first column must be a valid Grep search pattern, while the second column may be a literal string or a valid Grep replacement pattern.

When using Grep, BBEdit will check every search pattern for valid Grep syntax before beginning the Canonize operation. If any pattern is invalid, BBEdit will report an error for that line in the Canonize data file (including the PCRE error) and will not begin the operation.

NOTE Whether you are using Grep or not, since the hashmark character "#" is used to comment lines, you must backslash-escape it "\#" in any string or pattern which needs to use it literally.

Mode Lines in Canonize Transformation Files

You can now add mode lines to your Canonize transformation files. The mode line must be a comment line and occur on the first or second line of the file. It may contain the following variables (in addition to any other mode-line variables, which Canonize will ignore):

x-bbedit-canon-case-sensitive

x-bbedit-canon-match-words

x-bbedit-canon-grep

If any of these are present, then the mode line will override the settings in the Canonize dialog box (or Text Factory action settings). In that case, any absent mode settings will default to "off".

Here are some examples:

```
# -*- x-bbedit-canon-grep: 1; -*-
```

Meaning: Use grep; "match words" and "case sensitive" are off.

```
# -*- x-bbedit-canon-case-sensitive: 1; x-bbedit-canon-match-
words: 1; x-bbedit-canon-grep: 0; -*-
```

Meaning: "case sensitive" and "match words" are on; "use grep" is off.

You may use "1" and "0" for "on" and "off" respectively; or "t" and "f", or "y" and "n".

We recommend that you always use a mode line; this makes it easier to share Canonize transformation files and guarantee that their behavior will be consistent irrespective of the individual user's settings.

Text Merge

The "Text Merge" command on the Text menu provides a new way to transform text in files, by employing a file containing a Grep search and replace pattern along with a table of substitution values.

Each match of the search pattern is processed using the replace pattern, and the table of substitution values is used to provide substitutions for each respective capture group reference in the replacement.

Here is a simple contrived example to illustrate.

Consider a file consisting (entirely) of the following:

```
a1
b2
c3
d4
e5
f6
g7
h8
i9
j10
```

If you wanted to transform the first letter (“a”, “b”, “c”, etc) to the corresponding NATO alphabet name (“alpha”, “baker”, “charlie”, etc) and each number to the corresponding spelled-out word, the search pattern to match these would be: `(\w)(\d+)`.

To replace each letter with the NATO spelling, and each number with the full spelling, the replacement pattern would be: `\1 \2`.

The corresponding replacements can be expressed as a list of pairs:

```
alpha      one
baker      two
charlie    three
delta      four
echo       five
foxtrot    six
golf       seven
hotel      eight
india      nine
juliett    ten
```

So instead of needing to perform sequential grep replacements with different values for different matches, Text Merge provides the mechanics to do the replacements sequentially. The instructions for the merge operation are specified in the "merge file"; when you choose the "Text Merge" command from the Text menu, select the merge file to apply to the active document.

Note The ordering of the pairs in the example provided above is purely incidental. Pairs are used in the order that they are listed in the merge file, consistent with the order that each Grep match is found (starting at the beginning of the document), not in any intrinsic alphabetic (or other) ordering.

Increase and Decrease Quote Level

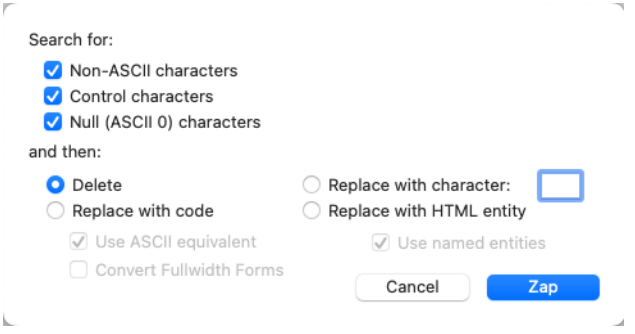
These commands respectively insert or delete a standard Internet quote character (">") from each line of the selected hard-wrapped text, or for the current line if there is no selection.

Strip Quotes

This command removes all Internet-style quoting from the selected hard-wrapped text, or from the current line if there is no selection.

Zap Gremlins

This command displays a sheet which allows you to remove or replace various non-printing characters, often known as “gremlins”. Use this command when you have a file that may contain extraneous control characters, or any non-ASCII characters, which you wish to identify or remove.



The checkboxes in the “Search for:” section of the sheet determine which types of characters the Zap Gremlins command affects, while the radio buttons below determine what action(s) to perform on all gremlins which are found.

Non-ASCII characters

When this option is selected, Zap Gremlins zaps *all* characters in the file that do not fall in the 7-bit (or ASCII) range. Examples of such characters include special Macintosh characters such as bullets (•) and typographer’s quotes (“ and ”, ‘ and ’), as well as all multi-byte characters. In general, such special characters are those that you type by holding down the Option key.

Control characters

When this option is selected, Zap Gremlins zaps a specific range of invisible low-ASCII characters, also known as control characters. Control characters can cause compilers and other text-processing utilities to malfunction, and are therefore undesirable in many files.

Null (ASCII 0) characters

When this option is selected, Zap Gremlins zaps all instances of the null character (ASCII 0). Like other control characters, nulls can cause many programming tools and text-processing utilities to malfunction. This specific option is included in case you want to remove only nulls without affecting other control characters that may be present in a file.

Delete

This option removes the zapped character completely from the text. It is useful if you are only interested in destroying gremlins and you do not care where they were in the text.

Replace with code

This option causes BBEdit to handle each gremlin character according to its value as follows:

BBEdit will convert certain eight-bit Mac Roman characters (characters whose decimal values are greater than 128 and less than 255) to 7-bit (printable ASCII range) equivalents. Converted characters include umlauted and accented vowels, ligatures, typographer's quotes, and various specialized punctuation forms. This conversion may entail expansion to multiple characters; for example, in the case of ligatures.

When the *Use ASCII equivalent* option is also enabled, BBEdit will convert all extended Roman characters to their closest ASCII equivalent.

Otherwise, BBEdit will convert all other gremlins present to escaped hexadecimal format. The escape code is formed via the same convention used by the C programming language: \0x followed by the character code in hexadecimal (base 16). This option is useful for identifying both the value and the location of gremlin characters. Later, you can search for occurrences of \0x to locate the converted characters. (Searching for the grep pattern of "\\0x.." will select the entire character code for easy modification or deletion.)

When the *Convert fullwidth forms* option is also enabled, BBEdit will convert all Unicode fullwidth forms: U+FF01 through U+FF5E (inclusive) to their ASCII equivalents.

Replace with <character>

This option replaces the gremlin with the character you type in the text field next to the radio button. It is useful for identifying the location of gremlins, but not their value. The replacement character can be specified not only as any typeable character, but also by using any of the special characters defined for text searches, including hex escapes. (See "Special Characters" on page 179.)

Note In some cases, this option could be counterproductive, since hex escapes (\xNN) can themselves be used to insert unprintable characters.

Convert Escape Sequences

This command transforms text by replacing various commonly used character escape sequences with their actual characters:

- Control characters: this will replace the same set of character escape sequences that BBEdit's text searching uses: \n, \r, \t, \f, and \\.
- Hex escapes: \xNN and \x{NNNN}, where each N is a valid hex digit.
- Unicode escapes: JavaScript-style \uNNNN and \u{NNNN}, where each N is a valid hex digit.
- HTML entities: any valid and well formed HTML character entity, named or numeric.
- Percent escapes: URL-encoded character sequences (interpreted as UTF-8).

Convert Spaces to Tabs

This command displays a sheet which allows you to set the number of consecutive space characters which should be converted into tabs. This transformation is useful when you are copying content from many online sources, which use spaces to line up columns of text. If you do not use a monospaced font, columns usually will not line up unless you first convert spaces to tabs.

Convert Tabs to Spaces

This command displays a sheet which allows you to set the number of consecutive spaces which should replace each tab. This command is useful when you are preparing text for use in a program which has no concept of tabs as column separators, for email transmission, and similar purposes.

Normalize Line Endings

This command converts a document containing mixed line endings to have a uniform set of line endings.

If you open a file which contains a mixture of Mac, Unix, and DOS/Windows line endings, the “Translate Line Breaks” option may not suffice to properly convert the document for viewing and editing. After conversion, the document may appear to not have any line breaks at all (this usually happens if the first line break in the file is a Mac line break, and all the rest are Unix), or to have an invisible character at the beginning of each line.

Should this happen, use Normalize Line Breaks to convert the remaining line endings, and save the document. Once you have done this, the document’s line endings will be consistent, and BBEdit’s line-break translation will suffice when you next open it.

Normalize Spaces

This command converts a document containing mixed space characters to have a uniform set of plain ASCII spaces.

When used from the Text menu, scripting interface, or text factory, this command transforms turns printable nonbreaking spaces into ASCII “space” characters. This is useful for cleaning up text copied from web pages, which is frequently rendered using nonbreaking spaces of various sorts.

Precompose Unicode

This command will convert decomposed Unicode pairs (such as a letter followed by a combining accent or diaresis) into a single Unicode character, where possible. (This capability is also available as a text factory action as well as via the AppleScript interface.

Decompose Unicode

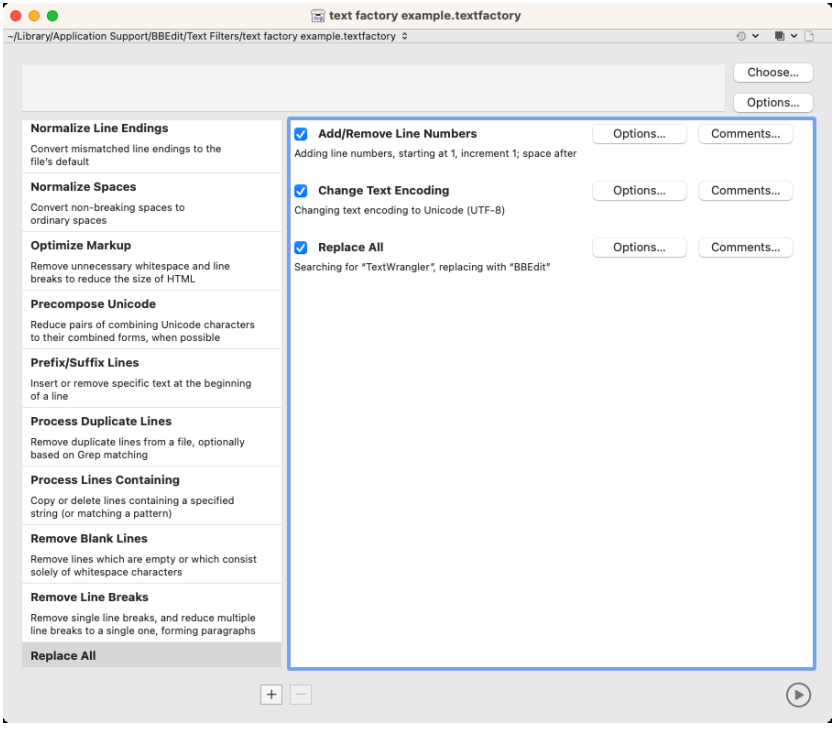
This command will convert single (individual) Unicode characters (such as an accented letter) into an Unicode combining pair consisting of a base character plus a suitable accent or diaresis, where possible. (This capability is also available as a text factory action as well as via the AppleScript interface.

Strip Diacriticals

This command replaces composed diacritical forms in the text with the base character. Thus, “á” becomes “a”, “ç” becomes “c”, and so forth. This capability is available via the Text menu, "Apply Text Transform", text factories, and the AppleScript interface.

Text Factories

A text factory document enables you to apply BBEdit's powerful text transformation commands in the order and fashion that you decide, to whatever collection of files you choose. So, for example, if you routinely need to process a folder full of server logs by reducing them to lines which don't contain "error", prefixing each line with a line number, and converting each file's line endings to Unix, you can assemble and save a text factory to do that work for you, and apply it at any time.



Creating and Configuring Text Factories

To create a new text factory, choose the Text Factory command from the New submenu of the File menu. BBEdit will create a text factory document. You may save a text factory document to disk any time by using the Save command, or clicking the Save button at the bottom of the window.

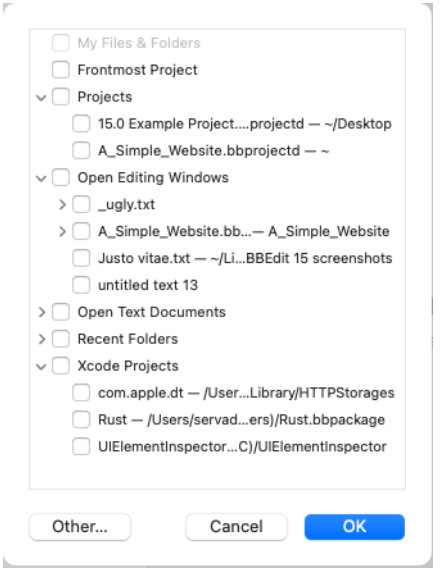
Text factories placed in the Text Filters subfolder of BBEdit's application support folder will appear in the Apply Text Filter and Paste Using Filter submenus, and you may apply such a factory to process the current selection (if any) or the contents of the frontmost document (or clipboard).

Text factories placed into the Scripts subfolder will appear in the Scripts menu, and you may run such a factory to process any designated set of target files and folders.

Note Text factory documents are macOS "property list" files, which are distinguished from editable text files by having a filename extension of ".textfactory". In order for text factory files to be recognized as such, you must use this filename extension when saving a text factory document.

Choosing Targets

At the top of a text factory window is a summary area which displays information about the target files and folders you have chosen for processing. Click Choose to present a sheet containing a list of selected and available target items.

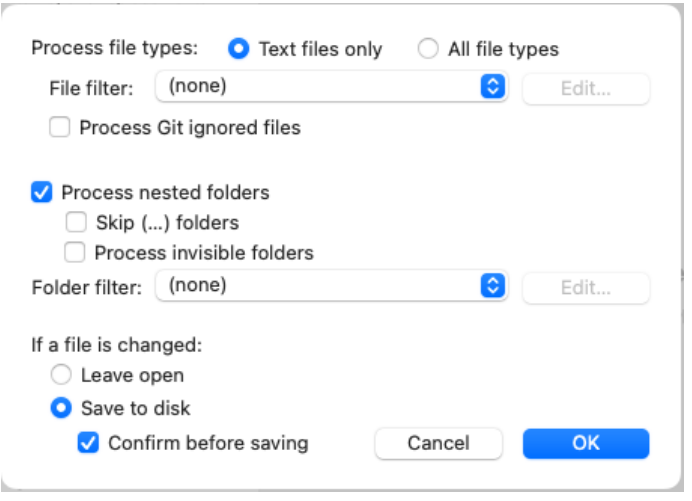


Available targets include:

- individual files and folders
- the contents of the frontmost project (only)
- the contents of any open project documents
- the contents of any open multi-document editing windows
- open text documents
- the files in any recently-searched folder
- the files listed in any results browser (such as a search results browser, an HTML syntax errors browser, or a compile errors browser)
- the files and folders contained in any open project
- any Finder "Smart Folders" which you have defined (BBEdit will automatically list such items from the "Saved Searches" folder for your login account)

To select (or deselect) an item or a set of items as a target, click the checkbox next to its name. To add a file or folder to the list, click Other and select it in the resulting Choose Object dialog.

Click Options to select additional options for controlling which target items will be processed. To process all the files in subfolders of each target folder, mark the “Process nested folders” checkbox (this option is on by default). If you also want the factory to process Git ignored files, mark the “Process Git ignored files” checkbox



You can also choose to process only text files or to process all file types. If you have graphics or other types of files in the target folders, you should restrict processing to only text files. This setting works in addition to any file filter that you apply (see “File Filters” on page 185) and will take effect *before* the filter.

The last group of options controls how BBEdit treats processed documents. Choose “Leave open” to have BBEdit leave all the documents open so that you can inspect the results of the operation. Choose “Save to disk” to have BBEdit automatically save changes to each file after processing it. When the “Confirm before saving” option is enabled, you will have an opportunity to approve the changes before BBEdit saves them to disk. You should not turn this off unless you are sure that the actions being applied are doing what you want.

Defining Actions

Each action in a text factory contains the following elements:

- a checkmark to enable or disable the action. (This option is primarily intended as a troubleshooting aid, but may also be useful in other contexts.)
- a list of all the actions BBEdit supports in the lefthand sidebar; you can select and add any desired operation by clicking the Add “+” button or by double-clicking the action. You are not limited to a single use of an action; for example, you can apply multiple Replace All operations in a single factory.
- an Options... button, which is used to configure any settable options for the command. If the command has no settable options, this button is disabled.

- a Comments button, which brings up a dialog you can use to record comments about the configuration or purpose of the action.
- Add ("plus") and Remove ("minus") buttons. Clicking the Remove button in an action will remove the selected action(s); clicking the Add button will insert a new action after the selected action. Hold down the Option key while clicking the Add button to create a new action which duplicates the action next to that button. If there is only one action in the factory, the Remove button will be disabled.
- a summary line which describes the chosen command and any parameters.

Note It is up to you to make sure that your actions do not work at cross purposes. For example, it would not be useful to have an "Educate Quotes" operation followed immediately by a "Straighten Quotes" operation.

You can re-order actions by clicking the line containing an action (in any place not occupied by a button or popup menu) and dragging it to a new location.

Additionally, you can copy and paste actions both within and between text factory windows. When pasting, the items on the clipboard will be inserted before the first selected item if there is one; otherwise the pasted items are added to the end of the list.

Each operation available on an action's pop-menu behaves similarly to its counterpart command on the Text menu. When choosing and configuring operations, you should keep the following differences in mind.

- Change Line Endings and Change Text Encoding will only affect the line endings and text encoding of the file that BBEdit writes out when you choose Save to Disk from the factory's Options sheet. Neither operation changes any file contents in memory, so they will have no visible effect if the document is left open (either by choosing the Leave Open button in the Options sheet, or by opting to leave the document open when confirming a save).
- When you use the Run AppleScript Filter operation, your script should be written with an entry point named "ApplyTextTransform". The input parameter to this entry point is a Unicode string containing the file's contents. This entry point should return the file's contents as a Unicode string (or something which can be directly coerced to one):

```
on ApplyTextTransform (fileData)

-- do something to fileData

return fileData -- or some reasonable facsimile thereof

end
```

- The "Remove Blank Lines" action will remove only blank (lines containing only whitespace) and empty (no characters between line breaks) lines from the document being processed.
- The "Replace All" action now provides a search history popup which is shared with the Find window's search history for ease in re-using externally developed and tested patterns.
- There is now a "Run Unix Command" action, which functions similarly to "Run Unix Script" but the script text is stored within the factory proper, making it more portable.

- The “Run Unix Filter” action can reference any executable file; such files need not contain a shebang line.
- Unix filters run from text factories get their input on STDIN, and should write any content they want passed to the next filter stage to STDOUT. (Unlike Unix filters run from the Shebang menu, text factories do not pass a temp file in argv[1].)
- When you use the “Run Unix Filter” operation, you can choose to have BBEdit convert the file’s contents and the output to UTF-8 by setting the “Use UTF-8 for I/O” option. If you do not set this option, BBEdit will use the system encoding instead.

Duplicating or Disabling Actions

You can select any number of existing actions within a text factory and then bring up the contextual menu to Cut, Copy, or Paste those actions to a different location within the factory, or enable/disable all of the selected actions.

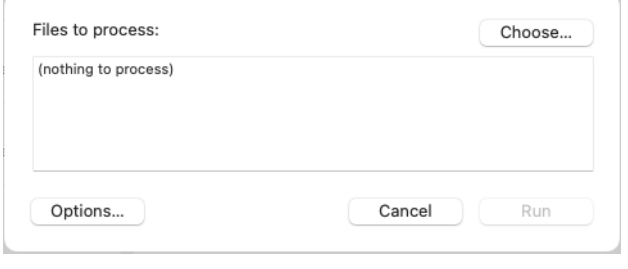
Applying Text Factories to Files

Once you have configured a text factory, you may click the Apply button (the triangular control) in the lower right-hand corner of the window to run that factory.

If you have already defined target(s), then BBEdit will apply the actions in the order you specified, to each file in the target set. This processing happens in the background, so you can keep using BBEdit while it’s underway (similar to a multi-file search operation).

If you have not yet defined any target(s), BBEdit will prompt you to choose a target and will apply the actions you specified to each file in the target set process, but the text factory will **not** remember the items you selected. However, if you change any of the scanning options, this will mark the text factory document as having unsaved changes

To run a stored text factory, choose it from the Scripts menu. If the text factory has any saved targets, BBEdit will apply the text factory to them. Otherwise, BBEdit will display the Run Text Factory dialog so you may click the “Choose” button and then select a set of target items to process.



Applying Text Factories to Open Documents

You can apply a stored text factory to the current document, or to the contents of the current selection, by choosing it from the Apply Text Filter submenu of the Text menu, or by double-clicking it in the Text Filters palette. (Alternatively, you can apply such a factory to the current contents of the clipboard by choosing it in the Paste Using Filter submenu of the Edit menu.)

HTML Processing Actions

Text factories support the following actions based on the equivalent Markup menu commands. (See Chapter 11 for details.)

- Translate Text to HTML
- Translate HTML to Text
- Format Markup
- Optimize Markup

Dedicated Text Processing Actions

In addition to offering text processing actions based on the commands in BBEdit's Text menu, text factories support the following action(s) which are not directly accessible via menu command:

- Delete Non-Matching Text

Delete Non-Matching Text

This operation works similarly to using the Extract button in the Find window; but rather than creating a new document with the extracted text, the document's contents are replaced with the results.

Thus, using this operation will delete any text which was not matched by the search string (or pattern), and the remaining text in the document will consist of all matches of the search string (or pattern), separated by line breaks. As with the Extract operation, this is not terribly useful for literal matches, but quite powerful when using the "Grep" search option together with a suitable replacement pattern.

Automator Actions

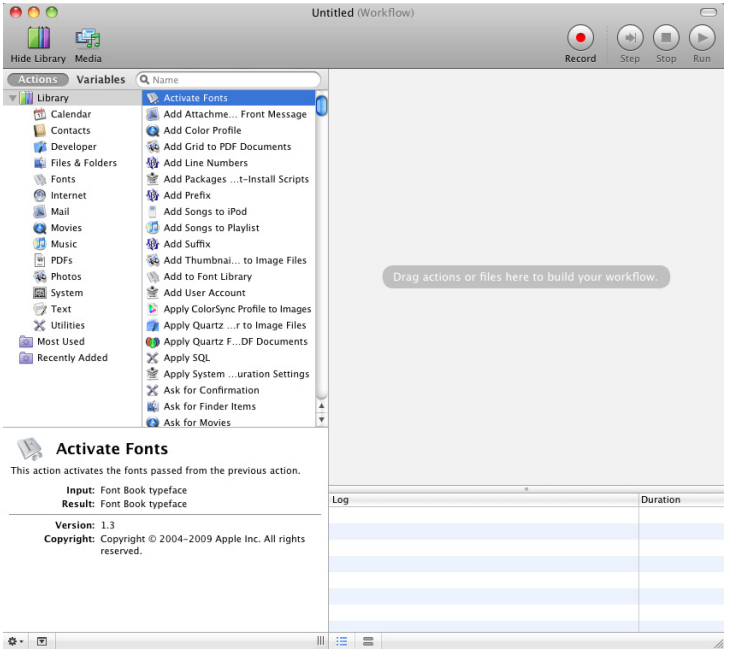
Automator is a system feature which enables you to create and reuse actions to easily perform common or time-consuming tasks. The Mac OS X Automation website offers a good overview of Automator.

`http://www.macosxautomation.com/automator/`

Using BBEdit with Automator

BBEdit offers Automator actions which correspond to most of the text transformation commands on the Text menu. In order to use any of BBEdit's actions, BBEdit must be running.

After you launch Automator, it will create an Untitled workflow as shown below. Choose any application in the Library section to see what actions it offers. To add an action to your workflow document, you can double-click on the action, or drag it into the right-hand panel. After you add an action to your workflow, you can set options (if any) in the action's tile. To expand or collapse an action tile, click the triangle next to the action's name.



The Automator actions which BBEdit offers allow you to make use of BBEdit's unique transformations in the context of a larger Automator workflow. Please keep in mind that BBEdit currently does not include any actions which support working with files (other than opening them), as it's expected that you will be applying BBEdit actions to work on text acquired from other sources in the workflow. If your main need is to apply BBEdit's text transformations to files or sets of files, you would be better served by using a Text Factory.

Available Actions

Add/Remove Line Numbers

This action will prepend a line number to each line processed, or remove the leading line number from each line processed.

Input: (Text)
Result: (Text) Text with leading line numbers added or removed.

Change Case

This action will change the case of the provided text according to the specified rules.

Input: (Text)
Result: (Text) Text with the case transformation applied.

Convert Spaces to Tabs

This action will convert each run of N space characters into a tab character.

Input: (Text)
Result: (Text) Text with runs of space characters converted to tabs.

Convert Tabs to Spaces

This action will convert each tab character into the specified number of spaces.

Input: (Text)
Result: (Text) Text with tab characters converted to runs of spaces.

Convert to ASCII

This action will convert the input text into an ASCII-only representation.

Input: (Text)
Result: (Text) Text with non-ASCII characters “translated” into ASCII

Delete Lines Containing

This action will return all lines in the input string which do not match the specified string.

Input: (Text)
Result: (Text) Text with the complement of the matching lines.
Related Actions: Extract Lines Containing

Educate Quotes

This action will convert ASCII quotes into their typographer equivalent.

Input: (Text)
Result: (Text) Text with ASCII quotes converted to typographer’s quotes
Related Actions: Straighten Quotes

Extract Lines Containing

This action will return all lines in the input string which contain the specified string.

Input: (Text)
Result: (Text) Text with the matching lines.
Related Actions: Delete Lines Containing

Get Contents of BBEdit Document

This action will get the contents of the frontmost text document in BBEdit according to the options you set.

Requires: A text document must be open.
Result: (Text) A BBEdit document object
Related Actions: Set Contents of BBEdit Document, New BBEdit Document

New BBEdit Document

This action creates a new BBEdit document from the input text.

Input: (Text)
Result: (com.barebones.bbedit.document-object) A BBEdit document object
Related Actions: Get Contents of BBEdit Document, Set Contents of BBEdit Document

Normalize Line Breaks

This action will change all line breaks into the format specified.

Input: (Text)
Result: (Text) Text with the chosen line breaks.

Prefix/Suffix Lines

This action will insert (or remove) the specified prefix and/or suffix strings on each line processed.

Input: (Text)
Result: (Text) Text with the specified strings inserted on (or removed from) each line.

Remove Duplicate Lines

This action will eliminate duplicates from the provided lines according to the specified rules.

Input: (Text)
Result: (Text) Unique Lines of Text
Related Actions: Sort Lines

Remove Prefix

This action will remove the supplied string from the beginning of each line processed.

Input: (Text)
Result: (Text) Text with the specified prefix removed from each line.
Related Actions: Add Suffix, Remove Suffix

Remove Suffix

This action will remove the supplied string from the end of each line processed.

Input: (Text)
Result: (Text) Text with the specified suffix removed from each line.
Related Actions: Add Suffix, Remove Prefix

Search and Replace

This action will search the input string, and replace all matches with specified string.

Input: (Text)
Result: (Text) Text with the specified replacements done.

Set Contents of BBEdit Document

This action will replace or insert the passed text into the frontmost text document in BBEdit according to the options you set.

Requires: A text document must be open.
Input: (Text)
Result: (com.barebones.bbedit.document-object) A BBEdit document object
Related Actions: Get Contents of BBEdit Document, New BBEdit Document

Sort Lines

This action will sort the provided lines according to the specified rules.

Input: (Text)
Result: (Text) Sorted Lines of Text.
Related Actions: Remove Duplicate Lines

Straighten Quotes

This action will convert typographer quotes into their ASCII equivalent.

Input: (Text)
Result: (Text) Text with typographer's quotes converted to ASCII quotes
Related Actions: Educate Quotes

Zap Gremlins

This action will convert each non-ASCII character as specified in the configuration.

Input: (Text)
Result: (Text) Text with non-ASCII characters removed.
Related Actions: Convert to ASCII

Other Transforms

BBEdit also offers a variety of other tools for transforming text, including columnar selections and manipulation, the Extract command for gathering text from single or multi-file search operations, and the ability to apply Text Filters to the contents of the clipboard.

Columnar Text Manipulations

The Columns submenu of the Edit menu contains commands to help you work more easily with column-delimited text files.

The three basic commands: Cut Columns, Copy Columns, and Clear Columns work similarly to their top-level analogues (Cut, Copy, and Clear). The columns to be cut, copied, or cleared are determined by the selection range: for example, to cut a single column, click in the middle of it and choose Cut Columns. You can cut, copy, or clear multiple columns by selecting text across them.

The New Document with Columns command will create a new untitled document containing the text of the selected column(s) within a recognizable column-delimited document.

The Rearrange Columns command offers you an easy way to rearrange the columns in a column-delimited text document. To use this command, just invoke it to bring up a dialog which will list all distinguishable columns present within the current document, then drag the columns into whatever order you wish, and click the Apply button to have BBEdit rearrange them.

Extract

The Extract command (available in the Search menu, as well as the Find and Multi-File Search windows) will find all instances of the search pattern (or string; see below) in the current document or search set, and collect each occurrence into a new untitled text document, separated by line breaks.

When performing an Extract operation, you can optionally enter a grep replacement pattern in the "Replace" field of the Find or Multi-File Search window to transform the extraction results. For further details as well as an example, please see "Extract" on page 193.

Alternatively, if you perform an Extract with the "Grep" option disabled, BBEdit will extract the entire contents of every line which *contains* the match, rather than just the exact text of the match.

Paste Using Filter

The Paste Using Filter submenu lists all the text factory and Unix filters available in the "Text Filters" subfolder of BBEdit's application support folder. When you choose a filter, BBEdit will process the text on the clipboard through the selected filter before inserting that text into the current document.

You can re-invoke the most recently used filter by choosing the 'Paste Using Filter <FILTERNAME>' command in the Paste submenu of the Edit menu, and optionally assign this command a key shortcut for quicker access in the Menus & Shortcuts settings panel.

Note Due to internal restrictions, the Paste Using Filter command does **not** allow the use of AppleScript filters or Automator actions.

CHAPTER

6 Windows & Palettes

This chapter describes the commands in the Window menu. These commands allow you to arrange and access editing and browser windows quickly, and also to access BBEdit's extensive set of tool and utility palettes.

In this chapter

Window Menu

The Window menu provides easy, centralized access to all of BBEdit's tool and utility palettes, in addition to offering commands that you can use to access and organize editing and results windows on screen.

Minimize Window

This command puts the frontmost window into the Dock. Click the window icon in the Dock to restore the window. Hold down the Option key and this command becomes Minimize All Windows.

Bring All to Front

This command brings all un-minimized BBEdit windows to the front.

Notes

This command will open BBEdit's Notes window (or bring this window to the front if it was already open).

For complete details on the Notes window, please see “The Notes window and Notebooks” on page 242

Palettes

The Palettes submenu provides quick access to all of BBEdit’s numerous tool palettes and utility windows. Choosing an item from this submenu toggles display of the corresponding palette.

Character Inspector

This command opens a palette which displays the character values of the selected text in several standard formats, as well as the offsets of the selected character(s). The offsets are expressed relative to the beginning of the document; so the first character offset is zero.

You may also select the displayed character values and either copy them or drag & drop them into a document.

Further, if the selected range of text looks like a color specification, the Character Inspector panel will display a color swatch for the corresponding color. (For example, try selecting “#EE11DD” or “rebeccapurple”.)

Clippings

BBEdit’s powerful Clippings feature provides an easy way to store and access frequently used text of any sort. For details on using Clippings, please see “About Clippings” on page 331.

Note The Set Shortcut button in the Clippings, Filters, and Scripts palettes allows you to assign a key equivalent to the currently selected item. You can use combinations of the Command, Shift, Option, and Control keys, plus any single other key, to create such equivalents, except that any equivalent must contain either the Command or Control keys (or both). You can also map Function keys directly to items, with or without the use of a modifier.

Colors

This command opens the system color picker, which you can use to insert hex color values into HTML and XML files.

Minimap

This palette shows a scaled-down view of the active (frontmost) text document.

You may click or drag within the minimap to make a corresponding selection change in the active text document; and likewise, changing the selection in the active document will update the displayed selection in the minimap.

(For greater visibility, an empty selection range in the active document is shown as a highlighted bar across the Minimap.)

The Minimap will also highlight (using the Page Guide color) the range of text currently visible in the active document. Scrolling the active document will move the highlighting in the Minimap.

The Minimap palette gets a cursor position display, and a popup menu button displaying the name of the function containing the start of the selection range. Clicking on the function display will open a function menu, from which you can choose another location in the displayed file.

Markdown Cheat Sheet

This command opens a floating window showing common Markdown constructions. Double-clicking on an item will insert it into the active document; you can also drag an item to insert it wherever you wish.

Scripts

The Scripts palette displays all the scripts currently installed in the Scripts subfolder of BBEdit's application support folder. See Chapter 2, "Scripts", for more information about using scripts in BBEdit.

Text Filters

The Text Filters palette displays all the text filters currently present in the Text Filters subfolder of BBEdit's application support folder. See Chapter 2, "Text Filters", for more information about using text filters in BBEdit.

Windows

The Windows palette displays the names of all open windows ordered by name and kind, and optionally displays an hierarchical list of all documents open within an editing window or project. (You can expand or collapse this list by clicking the disclosure triangle to the left of the parent window's name.)

You can open a file by dragging its icon from the Finder or from a project window into the Windows window.

Any document whose icon is darkened indicates that document's contents have been modified but not yet saved.

To bring any window or document to the front, click its name in the Windows palette, or to close any window or document, click the circular "close" box to its right.

"Hovering" the mouse over a window name displays a tool tip showing the full window title; this is useful for names that have been truncated with ellipses (…) because they are too long to fit within the width of the window.

HTML Markup Tools

The main HTML Markup Tools palette is a comprehensive listing of BBEdit's numerous HTML markup commands. See "HTML Tools Palette" on page 324 in Chapter 11 for details on what these commands do.

Several other HTML palettes are available, each with a specific focus: CSS, Entities, and Utilities. For more information on these tools, please see Chapter 11, "BBEdit HTML Tools".

Rescued Documents

This command opens BBEdit's Rescued Documents window.

Note Documents created by any in-app process, such as a text transformation (e.g. "Process Lines Containing" with results to a new document), Extract, or similar operations are **not** eligible for automatic rescue.

Show Scratchpad

This command opens BBEdit's persistent Scratchpad window. The scratchpad provides a workspace where you can store text to manipulate without needing to manage files. It's ideal as a 'worktable' where you can accumulate and modify text from one source (by performing transforms, manual edits, or batches of copy/paste operations) before gathering the text and using it elsewhere.

You can use the Save a Copy command in the File menu to save the scratchpad's contents into a stand-alone text file.

Show Unix Worksheet

This command opens BBEdit's persistent Unix Worksheet window.

Save Default <type of >Window

The Save Default Window command stores the position and size of the front window in BBEdit's settings, and BBEdit will create all new windows of the same type with the stored position and size.

By default, new windows always stack down and right 20px. If you have saved a default window size, and that window is of full screen height, new windows will just stack to the right, and preserve their saved height.

Each type of window has its own default position and size. For instance, the default position and size for project windows is different from the default position and size for text windows.

Window position and size settings are also keyed to the active screen configuration, so if you frequently switch screen layouts (as when connecting an external display to a portable), you can save separate default window settings which will be applied depending on which screen configuration is active.

Cascade Windows

The Cascade Windows command cascades all open editing windows in the default fashion: each successive window will be moved incrementally down and to the right (as described above).

Arrange

The Arrange command presents a submenu of multiple window arrangements to cascade, stack, or tile windows as indicated by their titles. You may choose any arrangement to immediately apply it, or assign a keyboard shortcut to any specific arrangement that you use frequently in the "Menus & Shortcuts" settings panel.

Move to [Display]

When BBEdit is running on macOS 10.15 or later with more than one display available, an additional command will become available in the Window menu: Move to [Display], where [Display] is the name of an eligible attached display. You may choose this command to move the front window to the indicated display, or hold the Option key down to move all open windows to the indicated display.

Cycle Through Windows

This command sends the front window behind all the other windows. Hold the Shift key down when choosing this command to Cycle Through Windows Backwards, i.e. to bring the rearmost window to the front.

Exchange with Next

This command makes the second window the active window. Choose this command repeatedly to alternate between the front two windows.

Synchro Scrolling

When you have two or more windows open, Synchro Scrolling makes both files scroll when you scroll one. This feature is useful to look over two versions of the same file.

Synchro Scrolling is also enabled by default within Differences windows, and when this option is active, scrolling either document within the window will scroll the other document as well. You can turn Synchro Scrolling off within the active Differences windows, or you can disable this behavior off for all new Differences windows by turning it off and then selecting the “Save Default Differences Window” command in the Window menu.

Window Names

The last items in the Window menu are the names of all the open documents, browsers, and other editing windows. Choose a window’s name from this menu (or use its numbered Command key equivalent, if applicable) to bring that window to the front.

Tip You can also use the Windows palette to quickly select any open window.

Zoom (key equivalent only)

There is no longer a Zoom command in the Window menu, but the key equivalent Command-/still works. Zoom will produce the same effect as clicking a window’s zoom box: it makes the active window larger if it is small, or returns it to its original size if it was previously enlarged by a Zoom command.

When zooming windows, BBEdit will move the window as little as possible (consistent with maximizing the window’s size). This behavior is similar to what the Finder does when zooming a window.

Workspaces

Introducing “workspaces”, a way to switch between arrangements of open documents and windows. A workspace includes the same application state that is saved and restored across quit and relaunch, but can be activated at any time while the application is running. This is useful for (for example) switching between working setups for different clients, or for different types of projects (writing vs programming vs web development).

The applicable menu commands reside on the BBEdit application menu. To create a new workspace, choose Save Workspace and give the workspace a name. To activate a previously saved workspace, choose it from the Workspaces submenu. You can overwrite an existing workspace by specifying the same name.

When switching to a different workspace, BBEdit will ask you to save any unsaved documents which correspond to existing files on disk (or remote FTP/SFTP items). You must save these before switching workspaces. Untitled (never-saved) documents are not affected by workspace changes, and will remain open (and unsaved) when you switch workspaces.

Saved workspaces reside in the “Workspaces” subfolder of BBEdit’s application support folder, accessible via the Folders submenu in the application (BBEdit) menu. If you wish to remove an inactive workspace, deleting the corresponding file will accomplish this. Existing workspaces can be renamed by renaming the corresponding file (whose extension must be preserved when doing so).

BBEdit does *not* update a workspace as you open or close documents or move windows around. If you make significant changes and would like to update the workspace, use “Save Workspace” and overwrite the current workspace. (For convenience BBEdit will fill in the name of the current workspace when you use “Save Workspace”.)

CHAPTER

7 Searching

This chapter describes how to employ BBEdit's powerful Find and Multi-File Search commands. It tells you how to search for text in the active window or within a set of files, and explains how to construct and employ file filters. BBEdit can also do advanced pattern, or grep, searching. To learn about pattern searching, you should read this chapter first and then read Chapter 8, "Searching with Grep."

In this chapter

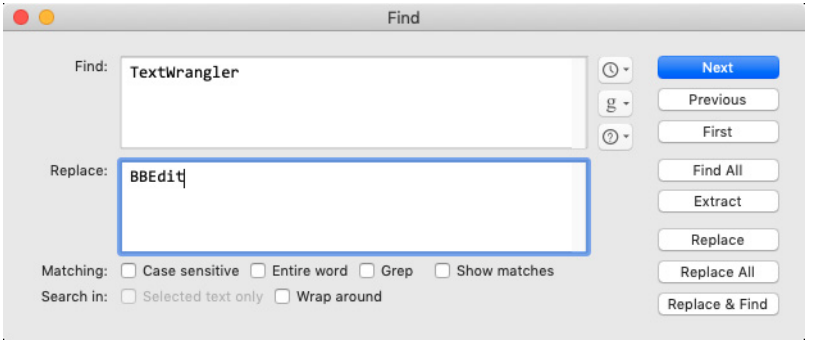
Search Windows

BBEdit's Find and Multi-file Search windows provide a consistent modeless interface to BBEdit's powerful text search and replace capabilities.

Basic Searching and Replacing

This section describes the basic steps for searching and replacing text in a document. Later sections in this chapter cover more advanced techniques. To search and replace text in the front document, follow these steps:

1 **Choose Find from the Search menu. BBEdit opens the Find window.**



2 **Type the string you are looking for in the Find text field.**

You can use special characters in the Find text field to search for tabs, line breaks, or page breaks. See "Special Characters" later in this section.

3 **Type the replace string (if any) in the Replace text field .**

BBEdit persistently remembers the pairs of search and replace terms that you have most recently used. If you want to repeat a previous search or replace, you can choose the appropriate entry from the Search History popup menu at the right of the Find text field to fill in the Find and Replace fields.

Note The size of both the search and replace terms is limited only by available memory.

4 **Turn on any options that you want to apply to your search.**

For more info about these options, see "Search Settings" later in this section.

5 **Click one of the buttons along the right side of the dialog box.**

The following table explains what each of the buttons does.

This button…	Does this…
Next	Finds the first occurrence of the text in the active window after (below) the current insertion point.
Previous	Finds the first occurrence of the text in the active window before (above) the current insertion point.
First	Always finds the first match in the document, irrespective of where the insertion point or selection range are.
Find All	Finds all the occurrences of the search string and displays the results in a search results browser.

This button…	Does this…
Replace	Replaces the current selection with the replace string, or if there is no selection, finds the next match and replaces it.
Replace All	Replaces every occurrence of the search string in the active window with the replace string.
Replace & Find	Replaces the current selection with the replace string, then finds the next occurrence of the text in the active window.

Once you have entered a search string (and also, if desired, a replace string), you can also use the commands in the Search menu to find and replace matches (see "Search Menu Reference" later in this chapter). The table below summarizes the most common commands you can use at this point.

This command…	Does this…
Find Next	Finds the next occurrence of the search string. To reverse the search direction, hold down Shift.
Replace	Replaces the selection with the replace string.
Replace All	Replaces all occurrences of the search string within the document with the replace string.
Replace to End	Replaces every occurrence of the search string from the current insertion point to the end of the document with the replace string.
Replace & Find Again	Replaces the selection with the replace string and looks for the search string again.

Search Settings

The checkboxes in the Find window lets you control how BBEdit searches your document for the indicated text.

Case Sensitive

When this checkbox is selected, BBEdit treats upper- and lowercase letters as different letters. Otherwise, BBEdit treats upper- and lowercase letters as if they were the same.

Entire Word

When this checkbox is selected, BBEdit matches the search string only if it is surrounded in the document text by word-break characters (white space or punctuation). Otherwise, BBEdit matches the search string anywhere in the text.

Grep

When this checkbox is selected, BBEdit treats the search and replace strings as grep patterns. Otherwise, BBEdit searches the document for text that matches the search string as it appears literally, and will replace any matched text with the replace string. To learn more about pattern searching see "Searching with Grep" on page 201.

When this option is enabled in the Find and Multi-File Search windows, BBEdit will validate the pattern as you edit it. An indicator button will display the pattern's validation status, and you can click this button for more information about the specific error.

Show Matches

When this checkbox is selected, BBEdit will automatically perform live match highlighting within the current document. Otherwise, BBEdit will find matches only when you invoke a search command.

Further, when this option is enabled, a status line will appear at the bottom of the Find window to report the number of matches.

Selected Text Only

When there is a selection active within the frontmost document and this checkbox is enabled, BBEdit will search (or replace) only within the selected text. Otherwise, BBEdit will search the entire document with no regard for the existence of any selection.

(This option will now become available ONLY while there is a selection within the active document, and BBEdit will automatically disable it if you clear the existing selection, or re-enable it if you make a new selection.)

Wrap Around

When this checkbox is selected, BBEdit will continue searching from the beginning of the document if a match is not found (or from the end of the document if searching backwards). Otherwise, BBEdit stops searching when it reaches the end of the frontmost document (or the beginning if searching backwards).

Searching Git Ignored Folders

By default, BBEdit will avoid scanning Git ignored folders during multi-file searches. Generally this is fine (and may significantly improve search performance), but it also ignores exceptions that may be specified in the `.gitignore` file. To address this when needed, BBEdit provides a setting in the Multi-File Search window's Options panel to search in Git ignored folders. (This option is off by default, for consistency with previous versions.)

Live Match Highlighting

NEW As you edit the search string in the Find window, BBEdit will automatically highlight any matches for it in the currently active document (usually the one immediately behind the Find window). This affords you a basic preview of the text that will be targeted by a Find All or Replace All operation.

(This behavior also works with grep patterns, provided the pattern within the "Find:" field is valid.)

You can disable this match highlighting by turning off the "Show matches" option within the Find window, which may be useful if you find live matching to be a distraction, or in cases where displaying live matches takes a noticeable amount of time.

Special Characters

You can use the following special characters to search for line breaks and other non-printing characters, as well as hexadecimal escapes to search for any desired 8-bit character.

Character	Matches…
\r or \n	line break
\t	tab
\f	page break (form feed)
\xNN	hexadecimal character code NN (for example, \x0D for CR)
\x{*NNNN*}	any number of hexadecimal characters *NN…* (for example, \x{0} will match a null, \x{304F} will match a Japanese Unicode character)
\\	backslash (\)

The form of a hex escape is "\xNN", where "N" is any single hex digit [0-9,A-F]. The "x" may be upper or lower case. (You can use the Character Inspector palette to find the hex value for any selected character or string.) You can perform a literal search for any character, including a null, using this option. Malformed escapes are treated as literal strings.

Note When the "Grep" option is off, BBEdit will use the "Keywords" color from the active text color scheme to highlight backslash escapes within text in the "Search" and "Replace" fields of the Find and Multi-File Search windows.

Swap Search & Replace Fields

Both the Find and Multi-File Search windows now possess a button which you can press to exchange the contents of the "Find" and "Replace" fields. (This action also has a factory default key shortcut of Control-Shift-F, which you may adjust via the Menus & Shortcuts settings pane if desired.

Multi-File Searching

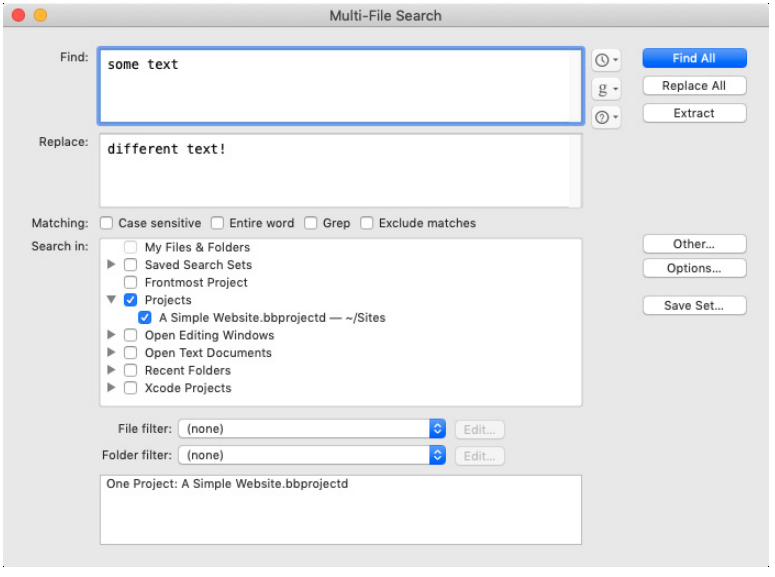
The main difference between single-file searching and multi-file searching is that to perform a multi-file search, you must specify the files to be searched. BBEdit gives you a great deal of flexibility in how to do this. You can search all the files in a given folder or defined website, in a project, in open editing windows, or in an existing search results browser. For even greater control, you can select a diverse set of search sources, or apply BBEdit's advanced file filtering capabilities.

When you start a search, BBEdit will display a search progress window and return control, so that you can continue working. You can perform more than one multi-file searches at a time; each search will have its own progress window. Closing a search's progress window or clicking Cancel in the progress window will stop the operation, and BBEdit will display a search results browser containing any matches found up to that point.

Starting a Search

To search for a string in multiple files, do the following steps:

1 **Choose Multi-File Search from the Search menu, or type Shift-Command-F, to open the Multi-File Search window (if it is not already open).**



2 **Type the string you are looking for in the Find text field.**

3 **Type the replace string (if any) in the Replace text field.**

Be sure to read the section "Multi-File Replacing" later in this chapter if you want to perform replace operations.

4 **Turn on any search options that you want to apply to your search.**

To learn more about these options, see "Search Settings" earlier in this chapter.

5 **Drag a folder to the search target area to search its contents, or select any of the available search sources in the Sources list to specify the set of files to search.**

See "Specifying the Search Set" later in this chapter for more information.

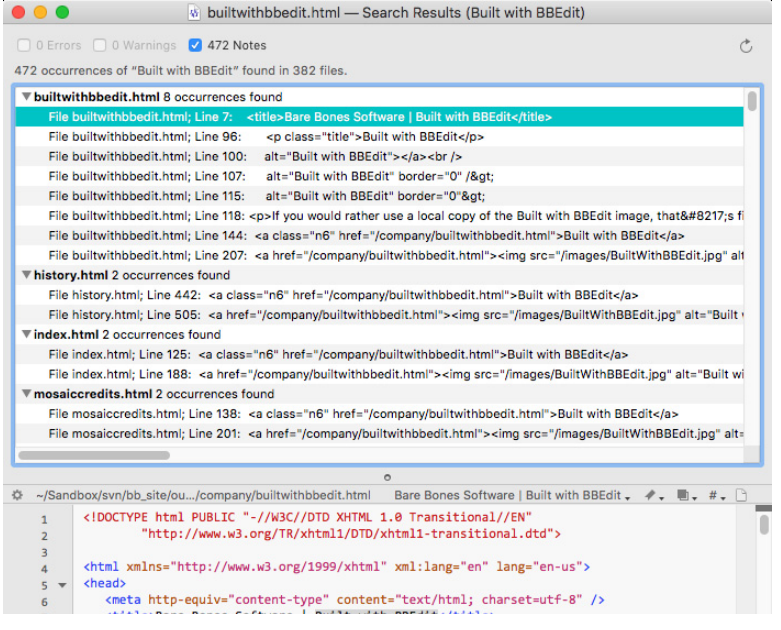
6 **Click one of the buttons along the right side of the dialog box to begin the search.**

The table below tells you what each of the buttons does.

This button…	Does this…
Find All	Finds all occurrences of the search string in all the files in the selected search sources. BBEdit displays the results in a search results browser.
Replace All	Finds all occurrences of the search string in all the files in the selected search sources and replaces them with the replace string.
Options	Brings up the Search Options sheet.
Save Set	Creates an entry under the “Saved Search Sets” heading in the search sources list which you can later choose to reselect the same search sources.
Other	Select arbitrary file(s) or folder(s) to add to the search sources.

Find All and Multi-File Search Results

When you perform a Find All search, either on a single file or across multiple files, BBEdit will open a search results browser which lists every occurrence of the search string in the selected file(s)



The information at the top of the window tells you how many matches BBEdit found in the set of files you specified, as well as specifying whether there were any error conditions or warnings generated during the search. You can display or hide any combination of errors, warnings, and matches, by checking the appropriate options.

The top pane lists each line that contains the matched text. Every match is identified by file name and line number.

Click any match in the list of occurrences to have BBEdit display the part of the file that contains the match in the text pane.

IMPORTANT You can edit text directly within a search results browser, or double-click any line that contains a match to open the corresponding file at the point of the match.

After you have opened a file, you can use the Find Again, Replace, Replace All, and Replace & Find Again commands in the Search menu to continue searching it, as if you had chosen a File by File search. See the next section for information on File by File searching.

To find only files whose contents do **not** contain the search string, select the Exclude Matches option.

You can also re-run the search with the same settings by clicking the Reload (circular arrow) button at the top right-hand side of the results browser.

Note You can use a search results window as the basis of another multi-file search. See "Specifying the Search Set" below.

Search Results Browser Reuse

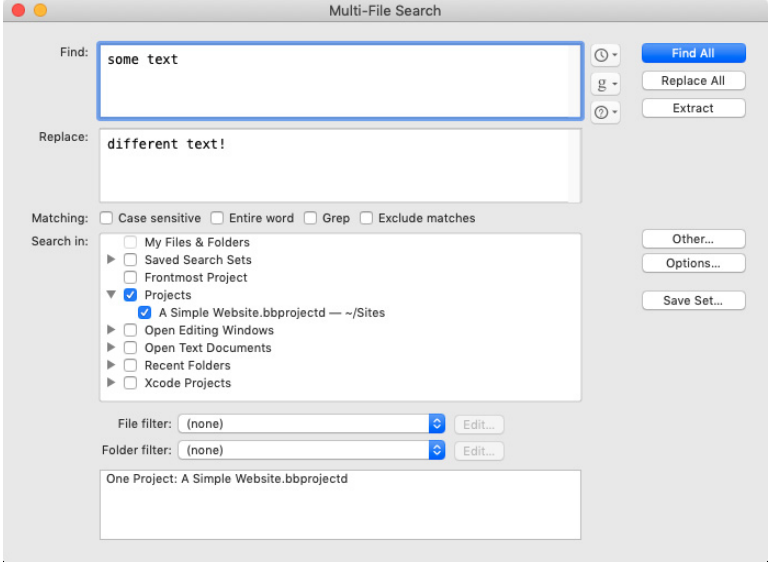
When you run a multi-file search, BBEdit will look through all existing search results browsers for the same search (based on the search & replace strings or patterns, search options, and the items being searched).

If a suitable match is found, BBEdit will reuse that window for the search results, rather than creating a new one.

Note There is an expert settings option that you can set to require BBEdit to create a new search results browser every time.

Specifying the Search Set

To specify which files and folders BBEdit should examine when performing a multi-file search, just select items from the Search In list of the Multi-File Search window.



You can choose multiple sources for a multi-file search, and you can mix different types of sources. Available sources include:

- specified individual files
- the files in any selected or recently-searched folder
- open text documents
- the frontmost project
- the files listed in any results browser (such as a search results browser, an HTML syntax errors browser, or a compile errors browser)
- the files and folders contained in a project
- the files and folders contained within any Zip archives
- any Xcode projects in your home directory
- any “Smart Folders” which you have saved in the Finder (BBEdit will automatically list any such items present in the “Saved Searches” folder for your login account)

To add a file, folder, or other suitable item to the Search In list, click Other in the Multi-File Search window, and choose the item using the resulting selection sheet. (You can select multiple items to be added.)

To designate any item in the list as a search source, click on the box next to its name, or double-click on the name, to add a checkmark. To deselect a search source, click the box next to its name, or double-click its name, to turn off the checkmark. To select a single source only, and deselect all other sources, Command-click on the checkbox next to the desired source’s name. To remove a search source from the list, click on the minus sign (–) to the right of its name. (Doing so removes only the entry from the list, **not** the original item.)

BBEdit will display a summary of the selected sources in the information box at the bottom of the Multi-File Search window. Here are some common scenarios.

Searching the files in a folder

To search the files in a folder, click on the box next to the folder’s name, or double-click its name, in the Sources list. If the folder you want to search is not in the Sources list, click the Other button at the right of the dialog and pick the folder using the resulting selection sheet.

You can also drag a folder from the Finder directly into the search items box of the Find & Replace dialog to choose it as the source.

Searching the frontmost project

You can directly search all files contained within the frontmost project document. To choose the frontmost project, click the box next to the Frontmost Project item, or double-click on the item in the list.

If the active document is also a project document, you can directly target it by choosing the Search in [project name] command (in the Search menu).

Searching all open documents

You can choose any or all open text documents as search sources. This option allows you to search documents that have not yet been saved to a file, or which contain unsaved changes. To choose all open documents, click the box next to the Open Text Documents item, or double-click on the item in the list.

Searching the contents of compressed archives

You can control whether BBEdit should search within the contents of compressed archives (“.bz2”, “.gz”, and “.zip”) via the “Search compressed files” option in the Multi-File Search window's “Options” sheet. When this option is off, BBEdit will skip all bz2, gz, and Zip while searching, even if they may contain compressed text files.

Searching the files contained in a results browser

If a previous multi-file search found many files that contain your search string, you may want to narrow the search. To search the files listed in any results browser window, click the box next to that browser’s name, or double-click on its name, in the Sources list. You can also click the box next to the Results Browsers item, or double-click on this item, to search the files listed in all results browsers.

Searching the files in a project

If the files you are working with are all listed in a BBEdit project, you can choose that group in order to search the files. To chose a project, click the box next to that group’s name, or double-click on its name, in the Sources list.

Note The Choose a Folder dialog will display any packages it encounters as folders (rather than just as single files, the way they appear in the Finder). This allows you to navigate their internal structure just as you would any other folder. Similarly, you can drag a package from the Finder into the path box in the Find & Replace dialog and it will be treated as a true folder rather than as a single file.

Saved Search Sources

You can use the Saved Search Sources popup menu to store specific sets of search sources for later reuse. To save a set of search sources, choose Remember this Set from the popup menu and give the set a name in the resulting dialog. To select a saved set of search sources, choose that set’s name from the pop-menu.

If you hover the pointer over a saved search set in the Multi-File Search window’s source list, BBEdit will display a tooltip listing the items contained in the search set.

Note Search sets are a deprecated feature; we recommend that you use projects instead.

Searches from ‘paths’ files

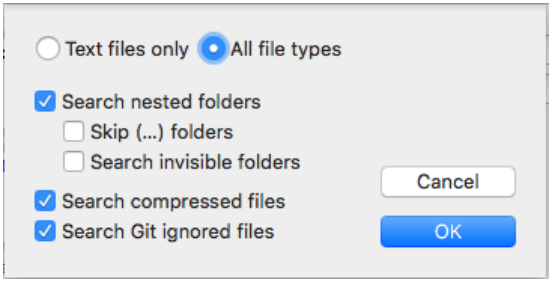
You can use the ‘paths’ files created by the FindAnyFile utility app as a list of files to be searched.

```
FindAny File  https://findanyfile.app/
```

You can drag a ‘paths’ file to the Multi-File Search window or use the “Other” button to select such file(s) as needed. (You can search multiple paths files in a single search, if desired.)

Multi-File Search Options

Click the Options button to display the search options sheet.



You can choose to search only text files or to search all file types. If you have image files or other non-text files in search source folders, it may be a good idea to restrict the search to only text files. This setting is applied in addition to any file and/or folder filters (see the following two sections) and in fact takes effect *before* the filter(s).

To search the contents of all subfolders within the folders you choose, select the “Search nested folders” option in the resulting sheet. You can also choose to skip any folders whose names are enclosed in parentheses here by selecting the “Skip (…) folders” option, or whether to search the contents of invisible folders by selecting the “Search invisible folders” option. In addition, you can control whether BBEdit should search compressed files and/or Git ignored files by enabling (or disabling) the corresponding options.

You can further restrict which files from the chosen sources BBEdit will search by applying a file filter. See “File Filters” (below) for more details.

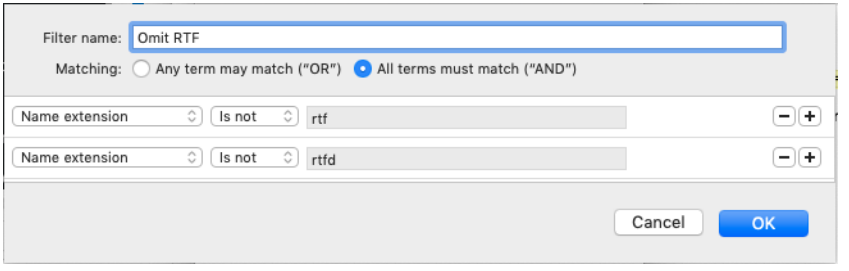
File Filters

If you do not want to search every file in the set you selected, but want to search the contents of only those files that meet certain criteria (such as those created on a certain date, or only those created by BBEdit and not some other program, or those that are HTML or Perl documents), you can use a file filter.

To apply a file filter, just choose it from the “File filter” popup menu in the Multi-File Search window. If none of the available filters meets your needs, you can define a new one, or create a temporary filter.

New Filter

To define a new saved file filter, select New… from the popup menu. BBEdit will ask you for a filter name, and then display the Edit Filter dialog (below). You can also define new file filters in the Filters panel of the Setup window (see page 280).



Note If the Setup window is open, any filters you define in the Multi-File Search window will not be available in the Filters panel of the Setup window until you close and reopen the Setup window.

The Edit Filter dialog lets you specify multiple criteria that determine whether a given file is selected by the filter. You can choose whether these criteria are exclusive (that is, whether a file must meet *every* listed test to be selected) or inclusive (that is, whether a file that meets *any* of the tests is selected) using the "Any term may match (OR)" or "All terms must much (AND)" radio buttons at the top of the dialog.

To add a test, click the Add (+) button, and a new row will appear in the dialog.

Within each row (criterion), the left-hand popup lets you specify which attribute of a file you wish to test. BBEdit lets you test a file's name, the name of its enclosing folder, its creator or type, its creation and modification date (or both date and time), or its Finder label, visibility, or the programming or markup language it is written in. You can also test the content of a file, using the "Contents" criterion.

The center popup lets you choose the test to be applied to the selected attribute. The available options here change depending on what attribute you selected. If you choose Visibility in the first column, for instance, your only choices are whether the file is or is not visible, However, if you choose File Name in the first column, the middle column lets you test to see if the name does or does not exactly match, contain, begin with, or end with a particular string. You can also test file names to see if they match wildcard or Grep patterns.

Note In wildcard patterns, the asterisk (*) and question mark (?) characters have special meanings. The asterisk matches any number of characters, such that "*.c" matches any file whose name ends with ".c". The question mark matches a single character, so that "foo?" matches "food", fool", "foot", and many other words. Both the asterisk and the question mark can be used anywhere in a wildcard pattern, and any number of either can be used in a single pattern.

Grep patterns, also known as regular expressions, are a powerful method of selecting file names based on classes of text or repeating text. They are covered in great detail in the next chapter.

The right-hand text field specifies the match criterion. For example, when filtering by File Name, you type the text you want the name to match, contain, begin with, or end with (or not). When filtering by Language, you choose a supported language from a popup menu.

You can add any number of criteria using the Add (+) button. To delete any criterion, click the Remove (-) button next to it.

Click Save to save the file filter and use it for this search. BBEdit will ask you to name the filter, and it will then appear in the Filters popup menu in the Find & Replace dialog (and in the Filter panel of the Setup window). Click Cancel to discard any changes you have made to the filter. (Hold the Option key when you click Cancel to skip the confirmation alert.)

Filtering by Name

In order to provide the greatest possible flexibility, BBEdit offers several different criteria for filtering based on file names

File Name: Tests the complete string corresponding to the file name.

File Name Root: Tests only the "root" portion of the file name. Given a name of the form "foo.txt", the root is the string which occurs before the period, in this case "foo".

File Name Extension: Tests only the file name extension. In the above example, the extension is "txt". (Note that the extension does not include a period.)

Filtering by Finder Tag

File (and folder) filters include a new term "Any Finder tag". You can enter the name of a single Finder tag in the string field, and if the file has one or more Finder tags applied and any of those tags matches the specified tag name (using any available string test), the file (or folder) will match that criterion.

Temporary Filters

Choose "(current criteria)" from the popup menu in the Find & Replace dialog to reuse the last set of criteria applied (either from using a saved filter, or from using the Edit button to define criteria). Thus, you can use filter criteria on the fly, without the need to create and store a throwaway filter.

Editing and Deleting Filters

To edit a file filter you have already defined, choose it from the Filters popup menu, change it as desired, and click Save. Since each filter must have a unique name, saving it will replace the old version of the filter. To delete a filter entirely, visit the Filters panel of the Setup window. (You can also create or modify filters there.)

Folder Filters

The Multi-File Search window provides the ability to filter out folders irrespective of their contents. In this way, you can restrict a multi-file search to only those folders whose contents you explicitly wish to search.

Folder filters are applied **after** any automatic filtering (such as ".gitignore" processing; skipping of invisible folders, "node_modules"; and so forth).

There are some things to be aware of:

- Filters can be used interchangeably for both files and folders; but not all terms make sense for folders. The following in particular will behave predictably but not necessarily usefully:

- The “Any Path Component” and “Every Path Component” criteria can be used to test any (or every) component of the file or filter's path using any of the available string tests.
- The “File Contents” term will never match when it occurs in a folder filter.
- Using the “Language” term to filter a folder may not make sense, but is nonetheless applied as though filtering a file. So if a folder’s name happens to map to an available language, the term may match.
- The “Legacy HFS file type” and “Legacy HFS file creator” terms will always return a zero value for folders, and therefore may not match.
- If a folder matches a filter, there is no assurance that any of its subfolders will be processed, **unless** they match the same filter. Conversely, if a folder fails to match a filter, **none** of its contents (files or subfolders) will be examined.
- If a folder (for multi-file search or other operations) does **not** match a specified folder filter, BBEdit will still examine its descendant folders (but not files). Thus, BBEdit will process the contents of descendants which **do** match the folder filter (which includes applying filtering to their respective descendants).

  This contrasts with the previous behavior, in which BBEdit would skip a folder which did not pass a folder filter **as well as all of its descendants**. (The old behavior was intentional, but potentially confusing.)

For example, if you have a directory structure like this:

```
MyDirectory/
MyDirectory/Foo/
MyDirectory/Foo/Grumble/
MyDirectory/Foo/Mumble/
MyDirectory/Bar/
MyDirectory/Bar/Foo/
```

and you’re using a folder filter of the form ‘Name is “Foo”’, when you perform a multi-file search of “MyDirectory”, then BBEdit will search only the top-level folder “Foo” but **not** either of its subfolders (“Mumble” or “Grumble”), because they do not match the filter.

Likewise, BBEdit will not search either “Bar” or its subfolder “Foo”, because “Bar” and all of its contents are excluded by their failure to match the filter.

For this reason, if you want to limit searches to folders *and their contents* by name, your best bet is to use the “Path” term: ‘Path contains “Foo”’.

Given the above, this would allow the search to examine “MyDirectory/Foo/”, “MyDirectory/Foo/Grumble/”, and “MyDirectory/Foo/Mumble/”. Note, however, that again, “MyDirectory/Bar/” would not match this term, and so neither it nor “MyDirectory/Bar/Foo/” would get searched. (Since filters can contain multiple terms, adding a ‘Path contains “Bar”’ term would allow BBEdit to search “MyDirectory/Bar/”.)

Searching SCM Directories

When scanning folders for various purposes (multi-file search, Find Differences, and other batch operations), BBEdit ignores all directories which contain administrative data for source-control management (SCM) tools: CVS, .svn, .git, .hg, .bzr. This behavior prevents any inadvertent modifications to such data which might otherwise occurs during a multi-file search or other batch operation. If you must search the contents of such directories, you can enable BBEdit to do so by issuing the following Terminal command:

```
defaults write com.barebones.bbedit SkipSCMAdminDirsWhenScanningFolders
-bool NO
```

Note The "Search Invisible Folders" option no longer enables BBEdit to search within such directories.

Multi-File Replacing

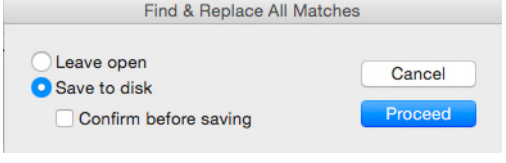
If you want to replace only some occurrences of text in multiple files, you can simply search those files, select the instances you want to change in the search results browser to open the files to those points, and perform the replacements individually. However, BBEdit can also change *all* occurrences of a string in a group of files with one command.

Globally replacing text in more than one file works the same as replacing it in a single file. The only possible complication is that, if you make a mistake, it can have much wider consequences. If you are not sure what effect a replace operation will have, test it out on a few sample files (or a copy of your data) first!

To do a multi-file search and replace, replacing all occurrences:

1. **Enter your desired find and replace strings in the Multi-File Search window as described in the section "Multi-File Search."**
2. **Choose the files to be searched as described in "Specifying the Search Set".**
3. **To start the operation, click Replace All in the Multi-File Search window, choose the Replace All command, or type its key equivalent of Option-Command-R.**

   BBEdit displays the Find & Replace All Matches dialog:



This is what each of its options does:

This option…	Replaces all occurrences of the search string with the replace string and…
Leave open	Leaves all the files open so that you can inspect the replacements. If there are many files that contain the search string, BBEdit may run out of memory.

This option…	Replaces all occurrences of the search string with the replace string and…
Save to disk	Saves each file with the changes.
Confirm before saving	When this option is enabled, you will have an opportunity to approve changes to each file being searched before BBEdit saves that file to disk. You should *not* turn this option off until you are sure that the replace operation is doing what you want.

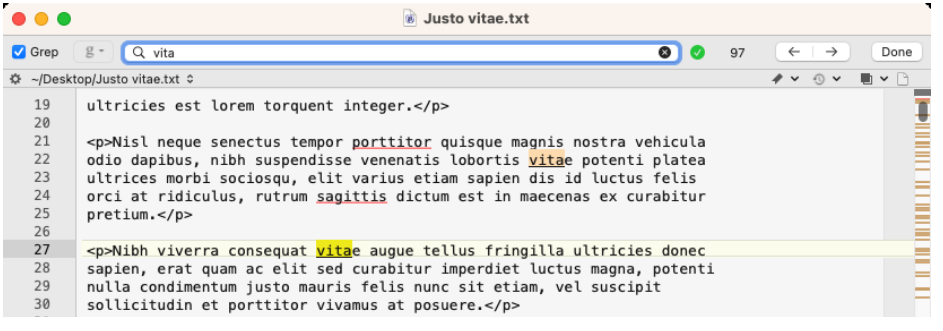
Live Search

The Live Search command performs an incremental search. In other words, it shows the matching text as you type the search string, so you only have to type until you find the text you want.

Live Search always searches in the text view of the frontmost window; if that window has no text view, the Live Search command will be disabled.

To use Live Search:

1. **Choose Live Search from the Search menu, or type Option-Command-F.**
2. **Type the string you are looking for into the Live Search field.**



   As you type, BBEdit selects the first occurrence of what you have typed so far.

3. **To find the next occurrence of the matching text, click the Next (right) arrow, or type Return or Enter.**
4. **To find the previous occurrence of the matching text, click the Previous (left) arrow, or type Shift-Return or Shift-Enter.**

If Emacs key bindings are enabled, you can also type Control-S to start a Live Search, and then type Control-S or Control-R to search forward or backward respectively.

To clear the most recent word of the search string, you can type Option-Delete, or click on the "delete" button (the "X") within the search field to delete the entire search string.

To cancel Live Search, you may click the "Done" button in the search bar or type the Escape key.

Live Search also supports grep pattern matching when the “Grep” option is enabled, and stored patterns are available under the “Saved patterns (“g”) popup. If the entered pattern is invalid, BBEdit will display an alert icon which you can click on to see the error.

Search Menu Reference

This section describes all of the commands in the Search menu.

Find

Opens the Find window (or the Find & Replace dialog). See “Basic Searching and Replacing” on page 176.

Multi-File Search

Opens the Multi-File Search window. See “Multi-File Searching” on page 179 and “Multi-File Replacing” on page 189.

Search in [Document’s Folder]

If a text document is active in the front window and that document is associated with a file on disk, this command will be enabled, and selecting it will open the Multi-File Search window with the search source pre-set to the parent folder of the document’s file, and that folder’s name and path will appear in this command’s menu name.

Search in [Project or Disk Browser]

If the frontmost window is a project or disk browser, this command's name will reflect the name of the project, or the name of the disk browser’s current root directory.

Choosing this command will open the Multi-File Search window with the search source pre-set to the current project, or the disk browser’s root folder.

If the Multi-File Search window is frontmost, this command will target the project or disk browser which is closest to the front (Z-order).

This command is also available in the Action (‘gear’) menu of projects and disk browsers.

Live Search

Opens the Live Search bar. You can use this feature to interactively search for text strings, as described in the previous section.

Find Next/Previous

Searches the current document for the next occurrence of the search string. Hold down the Shift key to find the previous occurrence.

Find First

Always finds the first match within the current document, irrespective of where the insertion point or selection range are.

Find All

Finds all instances of the search string in the current document or search set, and displays a search results browser.

Find & Select All

As its name implies, this command will select all matches for the search string (or pattern), and you can then perform any normal action such as typing to replace the selected matches, or pressing the Delete key to delete them.

You may also invoke this command via the menu bar or key shortcut when the Find window is frontmost, and BBEdit will apply it to the active document.

Extract

Finds all instances of the search string in the current document or search set, and collects each occurrence into a new untitled text document, separated by line breaks.

When performing an Extract operation, you can optionally enter a grep replacement pattern in the “Replace” field of the Find or Multi-File Search window to transform the extraction results.

For example, consider this sample text:

```
!_TAG_FILE_FORMAT   2   /extended format; --format=1 will not append ;" to lines/
!_TAG_FILE_SORTED   1   /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_PROGRAM_AUTHOR    Universal Ctags Team    //
!_TAG_PROGRAM_NAME  Universal Ctags /Derived from Exuberant Ctags/
!_TAG_PROGRAM_URL   https://ctags.io/   /official site/
!_TAG_PROGRAM_VERSION    0.0.0    /a1e9cbe/
```

To extract just the unique part of each field name (without the leading “!_TAG_”), match it with this pattern: “_TAG_(.+?)\s”. The first capture group is the unique part of the field name, so use “\1” in the "Replace:" field, and then click the Extract button.

The resulting extracted text will be:

```
FILE_FORMAT
FILE_SORTED
PROGRAM_AUTHOR
PROGRAM_NAME
PROGRAM_URL
PROGRAM_VERSION
```

Performing extraction with Grep substitutions will allow you to complete many extraction operations in a single step, rather than requiring you to apply multiple Replace All operations to the extracted text.

Search for [selected text] in [location]

When you invoke this command, BBEdit will start a multi-file search for the selected text (or the word surrounding the insertion point, if applicable), in the locations most recently used in the Multi-File Search window.

Find Selected Text/Previous Selected Text

Uses the selected text as the search string and finds the next occurrence of the selected text. Hold down the Shift key to find the previous occurrence of the selected text.

When you invoke this command, BBEdit will add the current search string to its Search History list of recently-used search strings.

Tip You can also hold down the Option and Command keys as you double-click on a selection to search for the next occurrence of the selected text.

Use Selection for Find

Sets BBEdit's search string to the currently selected text but does not perform a search. When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Find (grep)

When you hold down the Shift key, Use Selection for Find becomes Use Selection for Find (grep). This command sets BBEdit's search string to the currently selected text and turns on the Grep option, but does not perform a search. When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Replace

Sets BBEdit's replace string to the currently selected text but does not perform a search operation. When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Replace (grep)

When you hold down the Shift key, Use Selection for Replace becomes Use Selection for Replace (grep). This command sets BBEdit's replace string to the currently selected text and turns on the Grep option, but does not perform a search operation. When you invoke this command, BBEdit will add the current search& replace strings to its Search History list.

Replace

Replaces the selected text (usually an occurrence of the search string) with the replace string.

Replace All

Replaces **all** occurrences of the search string in a file with the replace string, or, starts a multi-file search & replace operation.

Replace All in Selection

This command is enabled only when there is a selection in the frontmost text document (or in the text document immediately behind the Find window). Choosing it will perform a Replace All upon the selected range of text, i.e. it has the same effect as enabling the Selected Text Only option in the Find window.

Replace to End

Replaces each occurrence of the search string from the current insertion point (or the start of the current selection range) to the end of the document.

Replace & Find Again

Replaces the selected text with the replace string and searches for the next occurrence of the search string.

Find Differences

Finds the differences between two files, or all of the files contained in two folders. See "Comparing Text Files" in Chapter 4 for more details.

Compare Two Front Windows

Performs a Find Differences between the active documents within the two frontmost text windows, using the same settings currently active for the Find Differences command.

This command will be enabled **only** if: there are two or more editing (or project) windows open, and a text document is active in each of the frontmost two windows, and neither window is blocked by a sheet or modal dialog.

Compare Against Disk File

Performs a Find Differences between the contents of the front document and the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Compare Against Previous Version

Provides access to previous versions of the current document using the system's built-in file versioning capability. You can compare and integrate changes from any prior version into the current document as desired.

Apply to New

Applies the currently selected difference to the "New" version of two files which are being compared. See "Comparing Text Files" for more details.

Apply to Old

Applies the currently selected difference to the “Old” version of two files which are being compared. See “Comparing Text Files” for more details.

Compare Again

Find the differences between two files, using the same settings that were used in the last time you used the Find Differences command. See “Comparing Text Files” for more details.

Find Definition

Looks up definitions for the selected word using ctags information if available. If there is no selection, BBEdit will attempt to determine the symbol name by inspection of the text around the insertion point, rather than requiring you to type a name. (See “Ctags for Enhanced Language Support” in Chapter 14 for more details about working with ctags.)

Find in Documentation

Performs a search for the selected symbol using an appropriate language-specific online resource. As for Find Definition, if there is no selection, BBEdit will attempt to determine the symbol name by inspection around the insertion point.

For example, Find in Documentation in a PHP document will look up the selected symbol on php.net; in a Ruby document, it will use the ‘ri’ interactive reference; in a Unix Shell Script, it will open the appropriate Unix man page.

For languages which don’t have a pre-defined resources, lookups will performed on the Apple Developer Connection website.

You can modify the URL template which BBEdit uses to perform the lookup for a particular language by bringing up the Options sheet for that language in the Languages settings panel and editing the template directly. In the template, use “__SYMBOLNAME__” to indicate where the selected symbol name should be placed in the lookup string.

Dash Integration

BBEdit will now provide docset information to Dash when you use Find in Documentation. Most of BBEdit’s factory-supplied languages specify a default docset list, derived (where applicable) from information provided by the developer.

In addition, BBEdit’s custom language settings (in the Languages settings pane) also provide a tab for Dash intergration, within which you can customize (or set up) the list of Dash docsets that BBEdit should request when using Dash.

Find References to Selected Symbol

This command will be available for any document written in a language backed by an LSP server.

When a word is selected, or the insertion point is in the middle of a word, this command will ask the server to return all occurrences of places where this symbol is used. The results are presented in a standard results window.

In addition, right-clicking in or on a selected word will add a “References” submenu to the resulting contextual menu, showing the references (if any) to that symbol.

Find Symbol in Workspace

If the active document's language has an LSP server running, this command (and its counterpart in the contextual menu) will open a modal panel so that you can search for symbols.

Each search request goes to the server, which is in complete control of what (if any) results are returned.

Go Menu Reference

This section describes all of the commands in the Go menu.

Line Number

When you choose this command, BBEdit opens the Line Number sheet. Type in a line number and the frontmost text window will jump to display that line.

This command does not follow the usual convention of applying the last-used setting when invoked with the Option key pressed. Instead, if you select a number within the current document, then hold down the Option key and choose “Line Number”, BBEdit will go directly to the correspondingly numbered line.

Note This command honors the “Use ‘Hard’ Line Numbering in Soft-Wrapped Text Views” option in the Editing settings panel.

Center Line

Will move the insertion point to the beginning of the middle or center line of the displayed text.

Named Symbol

When you choose this command, BBEdit opens a sheet displaying all named symbols within the current document, i.e. all available functions and markers. You can navigate this list with the arrow keys, or type to filter the displayed list of symbols.

Functions

When you choose this command, BBEdit opens a floating window which lists the functions in the active document (if any, i.e. provided its language type supports function scanning, and the document contains recognizable functions). You can filter the list by typing a partial function name into the search box.

Reveal Start/End

When you choose one of these commands, BBEdit will move the insertion point to a position immediately before the start or immediately after the end of the current function, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts settings panel.

Go to Previous/Next

When you choose one of these commands, BBEdit will select the name of the previous or next function in the document, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts settings panel.

Go to Declaration/Definition

These commands both require a running language server for the document's language.

Choosing either command will ask the language server for the location of the respective symbol declaration/definition, based on the location of the insertion point (or start of the selection range).

NOTE If you are using a C-family language for which 'clangd' is the language server, these commands are effectively synonymous, and will alternate (if appropriate) between the declaration and the definition of the selected symbol when the same command is chosen repeatedly. This is an intentional behavior in 'clangd' itself, and is not a bug in either BBEdit or in 'clangd'.

Alternatively, when you right-click on a word in a document served by an LSP server, BBEdit will ask the server for declarations and definitions of the word. If one (of each) is found, the contextual menu will contain Go to Declaration and Go to Definition, as appropriate. (Please note that BBEdit is **not** responsible for any nonsense results returned by the language server; it can only do what it's told.)

Markers

When you choose this command, BBEdit opens a floating window which lists any markers associated with the active document. (For information about setting markers, see "Using Markers" on page 134). You can filter the list by typing a partial marker name into the search box.

Jump Points

This command opens a floating window which displays the current document's jump history (implemented in previous versions of BBEdit) and provides a means to navigate this history in a non-linear fashion.

Previous

When you choose this command, BBEdit will go to the last selection you made in the document which was outside the current view (an automatic jump mark), or the last location you marked with the Set command (see "Set" on the preceding page). If the current document does not contain any jump marks, this command is disabled.

Next

When you choose this command after navigating to an earlier jump mark, BBEdit will go to the next later jump mark, or return to the most recent position of the insertion point. If you have not jumped back to a jump mark, this command is disabled.

Set

Choose this command to define the current insertion point location or selection range as a manual jump mark within the active document. You can navigate to jump marks using the Jump Back and Jump Forward commands.

Previous/Next Error

If an error browser is open, this command will open the listed error which came before or after the selected error. See Chapter 9 for more information on error browsers.

Previous/Next Placeholder

When you apply a clippings item that contains multiple #INSERTION# cookies, the second and subsequent cookies are replaced with special jump placeholders. You can also manually insert jump placeholders at any desired points within a clipping or a document.

These placeholders are strings of the form “<#...#>”, where the content “...” between the two # signs may be any alphanumeric text, or empty.

You can use the Go To Previous Placeholder and Go to Next Placeholder commands to jump back and forth between these special strings from the keyboard. For example, you might use this command when filling in the parameters of a function call, or a series of tag attributes. For additional details, see “Selection and Insertion Placeholders” on page 339.

If the “Use Tab key to navigate Placeholders” option in the Keyboard settings panel is turned on, you can press the Tab key (or Shift-Tab) to navigate between placeholders.

In additional to jump placeholders, you can also insert “optional” placeholders of the form <#?#>. When the “Go to Next Placeholder” command would select such a placeholder, BBEdit will place the insertion point at the specified position and remove the optional placeholder.

Commands...

The Commands... command brings up a modal panel which lists every action that you can perform by selecting a menu item within BBEdit: menu commands, clippings, scripts, stationery, text filters, as well as open text documents and recent files.

Type in the search box to narrow down the list; when you do this, results are ranked by how closely you matched what you entered.

You can use wildcards (not grep, but conventional file name wildcards like “*”, "?", and so forth) if you like.

Use the Up- and Down-arrow keys to move between matches; Return or Enter will run the highlighted command. This works whether keyboard focus is in the search box or on the list.

You can dismiss the panel without doing anything by pressing Escape or Command-period.

The right-hand column will display the currently assigned keyboard equivalent, if there is one. Double-clicking on the equivalent, or in the space where it would be, will allow you to change the key equivalent: press the keys that you would like to use for that command.

CHAPTER

8 Searching with Grep

This chapter describes the Grep option in BBEdit's Find command, which allows you to find and change text that matches a set of conditions you specify. Combined with the multi-file search and replace features described in Chapter 7, BBEdit's grep capabilities can make many editing tasks quicker and easier, whether you are modifying Web pages, extracting data from a file, or just rearranging a phone list.

In this chapter

What Is Grep or Pattern Searching?

Grep patterns offer a powerful way to make changes to your data that "plain text" searches simply cannot. For example, suppose you have a list of people's names that you want to alphabetize. If the names appear last name first, you can easily put these names in a BBEdit window and use the Sort tool. But if the list is arranged first name first, a simple grep pattern can be used to put the names in the proper order for sorting.

A grep pattern, also known as a regular expression, describes the text that you are looking for. For instance, a pattern can describe words that begin with C and end in l. A pattern like this would match "Call", "Cornwall", and "Criminal" as well as hundreds of other words.

In fact, you have probably already used pattern searching without realizing it. The Find window's "Case sensitive" and "Entire word" options turn on special searching patterns. Suppose that you are looking for "corn". With the "Case sensitive" option turned off, you are actually looking for a pattern that says: look for a C or c, O or o, R or r, and N or n. With the "Entire word" option on, you are looking for the string "corn" only if it is surrounded by white space or punctuation characters; special search characters, called metacharacters, are added to the search string you specified to indicate this.

What makes pattern searching counterintuitive at first is how you describe the pattern. Consider the first example above, where we want to search for text that begins with the letter "C" and ends with the letter "l" with any number of letters in between. What exactly do you put between them that means "any number of letters"? That is what this chapter is all about.

Note Grep is the name of a frequently used Unix command that searches using regular expressions, the same type of search pattern used by BBEdit. For this reason, you will often see regular expressions called "grep patterns," as BBEdit does. They're the same thing.

Recommended Books and Resources

Mastering Regular Expressions, 3rd Edition

by Jeffrey E.F. Friedl. O'Reilly & Associates, 2006. ISBN 0-596-52812-4

Although it does not specifically cover BBEdit's grep features, *Mastering Regular Expressions* is an outstanding resource for learning the "how-to" of writing useful grep patterns.

BBEdit Talk

The BBEdit Talk discussion group covers a wide range of topics and questions about using BBEdit, which frequently include searching and the use of grep patterns.

```
https://groups.google.com/g/bbedit
```

Note BBEdit's grep engine is based on the PCRE library package, which is open source software, written by Philip Hazel, and copyright 1997-2004 by the University of Cambridge, England. For details, see: https://www.pcre.org/

Writing Search Patterns

This section explains how to create search patterns using BBEdit's grep syntax. For readers with prior experience, this is essentially like the syntax used for regular expressions in the Perl programming language. (However, you *do not* need to understand anything about Perl in order to make use of BBEdit's grep searching.)

Most Characters Match Themselves

Most characters that you type into the Find window match themselves. For instance, if you are looking for the letter "t", Grep stops and reports a match when it encounters a "t" in the text. This idea is so obvious that it seems not worth mentioning, but the important thing to remember is that these characters *are* search patterns. Very simple patterns, to be sure, but patterns nonetheless.

Escaping Special Characters

In addition to the simple character matching discussed above, there are various special characters that have different meanings when used in a grep pattern than in a normal search. (The use of these characters is covered in the following sections.)

However, sometimes you will need to include an exact, or literal, instance of these characters in your grep pattern. In this case, you must use the backslash character \ before that special character to have it be treated literally; this is known as "escaping" the special character. To search for a backslash character itself, double it \\ so that its first appearance will escape the second.

For example, perhaps the most common "special character" in grep is the dot: ".". In grep, a dot character will match any character except a return. But what if you only want to match a literal dot? If you escape the dot: "\.", it will only match another literal dot character in your text.

So, most characters match themselves, and even the special characters will match themselves if they are preceded by a backslash. BBEdit's grep syntax coloring helps make this clear.

Note When passing grep patterns to BBEdit via AppleScript, be aware that both the backslash and double-quote characters have special meaning to AppleScript. In order to pass these through correctly, you must escape them in your script. Thus, to pass \r for a line break to BBEdit, you must write \\r in your AppleScript string.

Wildcards Match Types of Characters

These special characters, or metacharacters, are used to match certain types of other characters:

Wildcard	Matches…
.	any character except a 'hard' line break
^	beginning of a line (unless used in a character class)
$	end of line (unless used in a character class)

Being able to specifically match text starting at the beginning or end of a line is an especially handy feature of grep. For example, if you wanted to find every instance of a message sent by Patrick, from a log file which contains various other information like so:

```
    From: Rich, server: barebones.com
    To: BBEdit-Talk, server: lists.barebones.com
From: Patrick, server: example.barebones.com
```

you could search for the pattern:

```
^From: Patrick
```

and you will find every occurrence of these lines in your file (or set of files if you do a multi-file search instead).

It is important to note that ^ and $ do not actually match return characters. They match zero-width *positions* after and before returns, respectively. So, if you are looking for "foo" at the end of a line, the pattern "foo$" will match the three characters "f", "o", and "o". If you search for "foo\r", you will match the same text, but the match will contain four characters: "f", "o", "o", and a return.

Note ^ and $ do not match the positions after and before soft line breaks.

You can combine ^ and $ within a pattern to force a match to constitute an entire line. For example:

```
^foo$
```

will only match "foo" on a line by itself, with no other characters. Try it against these three lines to see for yourself:

```
foobar
foo
fighting foo
```

The pattern will only match the second line.

Other Positional Assertions

BBEdit's grep engine supports additional positional assertions, very similar to ^ and $.

Escape	Matches
\A	only at the beginning of the document (as opposed to ^, which matches at the beginning of the document and also at the beginning of each line)
\b	any word boundary, defined as any position between a \w character and a \W character, in either order
\B	any position that is *not* a word boundary
\z	at the end of the document (as opposed to $, which matches at the end of the document and also at the end of each line)
\Z	at the end of the document, or before a trailing return at the end of the doc, if there is one

Examples (the text matched by the pattern is underlined)

Search for: `\bfoo\b`

Will match: `bar foo bar`

Will match: `foo bar`

Will not match: `foobar`

Search for: `\bJane\b`

Will match: `Jane's`

Will match: `Tell Jane about the monkey.`

Search for: `\Afoo`

Will match: `foobar`

Will not match: `This is good foo.`

Character Classes Match Sets or Ranges of Characters

The character class construct lets you specify a set or a range of characters to match, or to ignore. A character class is constructed by placing a pair of square brackets […] around the group or range of characters you wish to include. To exclude, or ignore, all characters specified by a character class, add a caret character ^ just after the opening bracket [^…]. For example:

Character Class	Matches
`[xyz]`	any one of the characters x, y, z
`[^xyz]`	any character except x, y, z
`[a-z]`	any character in the range a to z

You can use any number of characters or ranges between the brackets. Here are some examples:

Character Class	Matches
`[aeiou]`	any vowel
`[^aeiou]`	any character that is not a vowel
`[a-zA-Z0-9]`	any character from a-z, A-Z, or 0-9
`[^aeiou0-9]`	any character that is neither a vowel nor a digit

A character class matches when the search encounters any *one* of the characters in the pattern. However, the contents of a set are only treated as separate characters, not as words. For example, if your search pattern is [beans] and the text in the window is “lima beans”, BBEdit will report a match at the “a” of the word “lima”.

To include the character] in a set or a range, place it immediately after the opening bracket. To use the ^ character, place it anywhere except immediately after the opening bracket. To match a dash character (hyphen) in a range, place it at the beginning of the range; to match it as part of a set, place it at the beginning or end of the set. Or, you can include any of these character at any point in the class by escaping them with a backslash.

Character Class	Matches
`[]0-9]`	any digit or]
`[aeiou^]`	a vowel or ^
`[-A-Z]`	a dash or A - Z

Character Class	Matches
`[--A]`	any character in the range from - to A
`[aeiou-]`	any vowel or -
`[aei\-ou]`	any vowel or -

Character classes respect the setting of the Case Sensitive checkbox in the Find window. For example, if Case Sensitive is on, [a] will only match “a”; if Case Sensitive is off, [a] will match both “a” and “A”.

Matching Non-Printing Characters

As described in Chapter 7 on searching, BBEdit provides several special character pairs that you can use to match common non-printing characters, as well as the ability to specify any arbitrary character by means of its hexadecimal character code (escape code). You can use these special characters in grep patterns as well as for normal searching.

For example, to look for a tab or a space, you would use the character class [\t] (consisting of a tab special character and a space character).

Character	Matches
\r	‘hard’ line break
\n	‘hard’ line break
\t	tab
\f	page break (form feed)
\a	alarm (hex 07)
\cX	a named control character, like \cC for Control-C
\b	backspace (hex 08) *(only in character classes)*
\e	Esc (hex 1B)
\x*NN*	hexadecimal character code *NN* (for example, \x0D for CR)
\x{*NNNN*}	any number of hexadecimal characters *NN…* (for example, \x{0} will match a null, \x{304F} will match a Japanese Unicode character)
\\	backslash

Use \r to match a line break in the middle of a pattern and the special characters ^ and $ (described above) to “anchor” a pattern to the beginning of a line or to the end of a line. In the case of ^ and $, the line break character is not included in the match.

Other Special Character Classes

BBEdit uses several other sequences for matching different types or categories of characters.

Special Character	Matches
`\s`	any whitespace character (space, tab, carriage return, line feed, form feed)
`\S`	any non-whitespace character (any character not included by \s)
`\w`	any word character (a-z, A-Z, 0-9, _, and some 8-bit characters)
`\W`	any non-word character (all characters not included by \w, including line breaks)
`\d`	any digit (0-9)
`\D`	any non-digit character (including line breaks)

A “word” is defined in BBEdit as any run of non-word-break characters bounded by word breaks. Word characters are generally alphanumeric, and some characters whose value is greater than 127 are also considered word characters.

Note that any character matched by \s is by definition not a word character; thus, anything matched by \s will also be matched by \W (but not the reverse!).

Quantifiers Repeat Subpatterns

The special characters *, +, and ? specify *how many times* the pattern preceding them may repeat. {}-style quantifiers allow you to specify exactly how many times a subpattern can repeat. The preceding pattern can be a literal character, a wildcard character, a character class, or a special character.

Pattern	Matches
`p*`	zero or more p's
`p+`	one or more p's
`p?`	zero or one p's
`p{`*`COUNT`*`}`	match exactly *COUNT* p's, where *COUNT* is an integer
`p{`*`MIN`*`,}`	match at least *MIN* p's, where *MIN* is an integer
`p{`*`MIN`*`,`*`MAX`*`}`	match at least *MIN* p's, but no more than *MAX*

Note that the repetition characters `*` and `?` match *zero or more* occurrences of the pattern. That means that they will *always* succeed, because there will always be at least zero occurrences of any pattern, but that they will not necessarily select any text (if no occurrences of the preceding pattern are present).

For this reason, when you are trying to match more than one occurrence, it is usually better to use a + than a *, because + *requires* a match, whereas * can match the empty string. Only use * when you really mean "zero or more times," not just "more than once."

Try the following examples to see how their behavior matches what you expect:

Pattern	Text	Matches
`.*`	Fourscore and seven years	Fourscore and seven years
`[0-9]+`	I've been a loyal member since 1983 or so.	1983
`\d+`	I've got 12 years on him.	12
`A+`	BAAAAAAAB	AAAAAAA
`A{3}`	BAAAAB	AAA *(first three A's)*
`A{3,}`	BAAAAB	AAAA
`A{1,3}`	BAAAAB	AAA on the first match, the remaining A on the second match
`c?andy`	andy likes candy	"andy" on the first match, "candy" on the second
`A+`	Ted joined AAA yesterday	"AAA" on the first match; "a" from "yesterday" on the second

Combining Patterns to Make Complex Patterns

So far, the patterns you have seen match a single character or the repetition of a single character or class of characters. This is very useful when you are looking for runs of digits or single letters, but often that is not enough.

However, by combining these patterns, you can search for more complex items. As it happens, you are already familiar with combining patterns. Remember the section at beginning of this discussion that said that each individual character is a pattern that matches itself? When you search for a word, you are already combining basic patterns.

You can combine any of the preceding grep patterns in the same way. Here are some examples.

Pattern	Matches	Examples
`\d+\+\d+`	a string of digits, followed by a literal plus sign, followed by more digits	4+2 1234+5829
`\d{4}[\t ]B\.C\.`	four digits, followed by a tab or a space, followed by the string B.C.	2152 B.C.
`\$?[0-9,]+\.\d*`	an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits	1,234.56 $4,296,459.19 $3,5,6,4.00000. *(oops!)*

Note again in these examples how the characters that have special meaning to grep are preceded by a backslash (\+, \., and \$) when we want them to match themselves.

Creating Subpatterns

Subpatterns provide a means of organizing or grouping complex grep patterns. This is primarily important for two reasons: for limiting the scope of the alternation operator (which otherwise creates an alternation of everything to its left and right), and for re-inserting the text matched by that portion of the pattern when performing replacements.

A subpattern consists of any simple or complex pattern, enclosed in a pair of parentheses. You can optionally specify a simple string to identify a subpattern, making it a named subpattern.

Pattern	Matches
`(p)`	the pattern *p* and remembers it
`(?P<`*NAME*`>p)`	the pattern *p* and remembers it by the specified string *NAME*

You can combine more than one subpattern into a grep pattern, or mix subpatterns and other pattern elements as you need.

Taking the last set of examples, you could modify these to use subpatterns wherever actual data appears:

Pattern	Matches	Examples
`(\d+)\+(\d+)`	a string of digits, followed by a plus sign, followed by more digits	4+2 1234+5829
`(\d{4})[\t ]B\.C\.`	four digits, followed by a tab or a space, followed by the string B.C.	2152 B.C.
`\$?([0-9,]+)\.(\d*)`	an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits	1,234.56 $4,296,459.19 $3,5,6,4.00000.

Using Backreferences in Subpatterns

What if we wanted to match a series of digits, followed by a plus sign, followed by the exact same series of digits as on the left side of the plus? In other words, we want to match "1234+1234" or "7+7", but *not* "5432+1984".

Using grouping parentheses, you can do this by referring to a backreference, also known as a captured subpattern. There are two kinds of backreferences: numbered backreferences, and named backreferences. You can use both types of backreference within the same grep pattern.

Each subpattern within the complete pattern is numbered from left to right, starting with the opening parenthesis. Later in the pattern, you can refer to the text matched within any of these subpatterns by using a backslash followed by the number of that subpattern; this is a numbered backreference. Unlike numbered backreferences, which are automatically identified from the pattern, named backreferences are only available after you define them.

Pattern	Matches…
`\1, \2, …, \99`	the text of the nth subpattern in the entire search pattern
`(?P=`*`NAME`*`)`	the text of the subpattern *NAME*

Names may include alphanumeric characters and underscores, and must be unique within a pattern.

Here are some examples of numbered backreferences:

Pattern	Matches	Examples
`(\d+)\+\1`	a string of digits, followed by a plus sign, followed the same digits	7+7 1234+1234
`(\w+)\s+\1`	double words, or, a pair of identical character runs separated by whitespace	the the ti**re re**turn *(oops!)*
`(\w)(\w)\2\1`	a word character, a second word character, followed by the second one again and the first one again	abba

We will revisit subpatterns in the section on replacement, where you will see how the choice of subpatterns affects the changes you can make.

Using Alternation

The alternation operator | allows you to match any of several patterns at a given point. To use this operator, place it between one or more patterns x|y to match either x or y.

As with all of the preceding options, you can combine alternation with other pattern elements to handle more complex searches.

Pattern	Text is…	Matches…
`a|t`	A cat	each "a" and "t"
`a|c|t`	A cat	each "a", "c", and "t"
`a (cat|dog) is`	A cat is here. A dog is here. A giraffe is here.	"A cat is", "A dog is"
`A|b+`	Abba	"A", "bb", and "a"
`Andy|Ted`	Andy and Ted joined AAA yesterday	"Andy" and "Ted"
`\d{4}|years`	I've been a loyal member since 1983, almost 16 years ago.	"1983", "years"
`[a-z]+|\d+`	That's almost 16 years.	"That", "s", "almost", "16", "years"

The "Longest Match" Issue

IMPORTANT When creating complex patterns, you should bear in mind that the quantifiers +, *, ? and {} are "greedy." That is, they will always make the longest possible match possible to a given pattern, so if your pattern is E+ (one or more E's) and your text contains "EEEE", the pattern matches all the E's at once, not just the first one. This is usually what you want, but not always.

Suppose, for instance, that you want to match an HTML tag. At first, you may think that a good way to do this would be to search for the pattern:

```
<.+>
```

consisting of a less-than sign, followed by one or more occurrences of a single character, followed by a greater-than sign. To understand why this may not work the way you think it should, consider the following sample text to be searched:

```
<B>This text is in boldface.</B>
```

The intent was to write a pattern that would match both of the HTML tags separately. Let's see what actually happens. The < character at the beginning of this line matches the beginning of the pattern. The next character in the pattern is . which matches any character (except a line break), modified with the + quantifier, taken together, this combination means one or more repetitions of any character. That, of course, takes care of the B. The problem is that the next > is also "any character" and that it *also* qualifies as "one or more repetitions." In fact, all of the text up to the end of the line qualifies as "one or more repetitions of any character" (the line break does not qualify, so grep stops there). After grep has reached the line break, it has exhausted the + operator, so it backs up and sees if it can find a match for >. Lo and behold, it can: the last character is a greater-than symbol. Success!

In other words, the pattern matches our entire sample line at once, *not the two separate HTML tags in it as we intended.* More generally, the pattern matches all the text in a given line or paragraph from the first < to the *last* >. The pattern only does what we intended when there is only one HTML tag in a line or paragraph. This is what we meant when we say that the regular quantifiers try to make the longest possible match.

Non-Greedy Quantifiers

IMPORTANT To work around this "longest match" behavior, you can modify your pattern to take advantage of *non-greedy* quantifiers.

Quantifier	Matches…
`+?`	one or more
`*?`	zero or more
`??`	zero or one
`{`*`COUNT`*`}?`	match exactly *COUNT* times
`{`*`MIN`*`,}?`	match at least *MIN* times
`{`*`MIN`*`,`*`MAX`*`}?`	match at least *MIN* times, but no more than *MAX*

Astute readers will note that these non-greedy quantifiers correspond exactly to their normal (greedy) counterparts, appended with a question mark.

Revisiting our problem of matching HTML tags, for example, we can search for:

```
<.+?>
```

This matches an opening bracket, followed by one or more occurrences of any character other than a return, followed by a closing bracket. The non-greedy quantifier achieves the results we want, preventing BBEdit from "overrunning" the closing angle bracket and matching across several tags.

A slightly more complicated example: how could you write a pattern that matches all text between <B> and </B> HTML tags? Consider the sample text below:

```
<B>Welcome</B> to the home of <B>BBEdit!</B>
```

As before, you might be tempted to write:

```
<B>.*</B>
```

but for the same reasons as before, this will match the entire line of text. The solution is similar; we will use the non-greedy *? quantifier:

```
<B>.*?</B>
```

Writing Replacement Patterns

Subpatterns Make Replacement Powerful

We covered subpatterns earlier when discussing search patterns and discussed how the parentheses can be used to limit the scope of the alternation operator. Another reason for employing subpatterns in your grep searches is to provide a powerful and flexible way to change or reuse found information as part of a search-and-replace operation. If you do not use subpatterns, you can still access the complete results of the search with the & metacharacter. However, this precludes reorganizing the matched data as it is replaced.

Pattern	Inserts…
`&`	the text matched by the entire search pattern
`\1, \2, …, \99`	the text matched by the nth subpattern of the entire search pattern
`\P<`*`NAME`*`>`	the text matched by the subpattern *NAME*

Note A pattern can recall up to 99 backreferenced subpatterns.

Using the Entire Matched Pattern

The & character is useful when you want to use the entire matched string as the basis of a replacement. Suppose that in your text every instance of product names that begin with the company name "ACME" needs to end with a trademark symbol (™). The following search pattern finds two-word combinations that begin with "ACME":

```
ACME [A-Za-z]+
```

The following replacement string adds the trademark symbol to the matched text:

```
&™
```

For example, if you start with

```
ACME Magnets, ACME Anvils, and ACME TNT are all premium
products.
```

and perform a replace operation with the above patterns, you will get:

```
ACME Magnets™, ACME Anvils™, and ACME TNT™ are all premium
products.
```

Using Parts of the Matched Pattern

While using the entire matched pattern in a replacement string is useful, it is often more useful to use only a portion of the matched pattern and to rearrange the parts in the replacement string.

For example, suppose a source file contains C-style declarations of this type:

```
#define Util_Menu 284
#define Tool_Menu 295
```

and you want to convert them so they look like this, Pascal-style:

```
const int Util_Menu = 284;
const int Tool_Menu = 295;
```

The pattern to find the original text is straightforward:

```
#define[ \t]+.+[ \t]+\d+[^0-9]*$
```

This pattern matches the word "#define" followed by one or more tabs or spaces, followed by one or more characters of any type, followed by one or more tabs or spaces, followed by one or more digits, followed by zero or more characters that are *not* digits (to allow for comments), followed by the end of the line.

The problem with this pattern is that it matches the entire line. It does not provide a way to remember the individual parts of the found string.

If you use subpatterns to rewrite the above search pattern slightly, you get this:

```
#define[ \t]+(.+)[ \t]+(\d+)[^0-9]*$
```

The first set of parentheses defines a subpattern which remembers the name of the constant. The second set remembers the value of the constant.

The replacement string would look like this:

```
const int \1 = \2;
```

The sequence \1 is replaced by the name of the constant (the first subpattern from the search pattern), and the sequence \2 is replaced by the value of the constant (from the second subpattern).

Our example throws out any comment that may follow the C-style constant declaration. As an exercise, try rewriting the search and replace patterns so they preserve the comment, enclosing it in (*…*) style Pascal comment markers.

Here are some more examples:

Data	Search for	Replace	Result
4+2	(\d+)\+(\d+)	\2+\1	2+4
1234+5829	(\d+)\+(\d+)	\1+\1	1234+1234
2152 B.C.	(\d{4})[\t]B\.C\.	\1 A.D.	2152 A.D.
1,234.56	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	1,234 dollars and 56 cents
$4,296,459.19	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	4,296,459 dollars and 19 cents
$3,5,6,4.00000	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	3,5,6,4 dollars and 00000 cents

Case Transformations

Replace patterns can also change the case of the original text when using subpattern replacements. The syntax is similar to Perl's, specifically:

Modifier	Effect
\u	Make the next character uppercase
\U	Make all following characters uppercase until reaching another case specifier (\u, \L, \l) or \E
\l	Make the next character lowercase
\L	Make all following characters lowercase until reaching another case specifier (\u, \U, \l) or \E
\E	End case transformation opened by \U or \L

Here are some examples to illustrate how case transformations can be used.

Given some text:

```
mumbo-jumbo
```

and the search pattern:

```
(\w+)(\W)(\w+)
```

the following replace patterns will produce the following output:

```
\U\1\E\2\3      MUMBO-jumbo
\u\1\2\u\3      Mumbo-Jumbo
```

Note that case transformations also affect literal strings in the replace pattern:

```
\U\1\2fred                  MUMBO-FRED
\1MUMBLE\2\3                mUMBLE-jumbo
```

Finally, note that \E is not necessary to close off a modifier; if another modifier appears before an \E is encountered, that modifier will take effect immediately:

```
\Ufred-\uwilma              FRED-Wilma
```

Examples

The example patterns in this section describe some common character classes and shortcuts used for constructing grep patterns, and addresses some common tasks that you might find useful in your work.

Matching Identifiers

One of the most common things you will use grep patterns for is to find and modify identifiers, such as variables in computer source code or object names in HTML source documents. To match an arbitrary identifier in most programming languages, you might use this search pattern:

```
[a-z][a-zA-Z0-9]*
```

This pattern matches any sequence that begins with a lowercase letter and is followed by zero or more alphanumeric characters. If other characters are allowed in the identifier, add them to the pattern. This pattern allows underscores in only the first character of the identifier:

```
[a-z_][a-zA-Z0-9]*
```

The following pattern allows underscores anywhere *but* the first character, but allows identifiers to begin with an uppercase or lowercase letter:

```
[a-zA-Z][a-zA-Z0-9_]*
```

Matching White Space

Often you will want to match two sequences of data that are separated by tabs or spaces, whether to simply identify them, or to rearrange them.

For example, suppose you have a list of formatted label-data pairs like this:

```
User name:      Bernard Rubble
Occupation:     Actor
Spouse:         Betty
```

You can see that there are tabs or spaces between the labels on the left and the data on the right, but you have no way of knowing how many spaces or tabs there will be on any given line. Here is a character class that means “match one or more white space characters.”

```
[ \t]+
```

So, if you wanted to transform the list above to look like this:

```
User name("Bernard Rubble")
Occupation("Actor")
Spouse("Betty")
```

You would use this search pattern:

```
([a-z ]+):[ \t]+([a-z ]+)
```

and this replacement pattern:

```
\1\("\2"\)
```

Matching Delimited Strings

In some cases, you may want to match all the text that appears between a pair of delimiters. One way to do this is to bracket the search pattern with the delimiters, like this:

```
".*"
```

This works well if you have only one delimited string on the line. But suppose the line looked like this:

```
"apples", "oranges, kiwis, mangos", "penguins"
```

The search string above would match the entire line. (This is another instance of the "longest match" behavior of BBEdit's grep engine, which was discussed previously.)

Once again, non-greedy quantifiers come to the rescue. The following pattern will match "-delimited strings:

```
".+?"
```

Marking Structured Text

Suppose you are reading a long text document that does not have a table of contents, but you notice that all the sections are numbered like this:

```
3.2.7     Prehistoric Cartoon Communities
5.19.001  Restaurants of the Mesozoic
```

You can use a grep pattern to create marks for these headings, which will appear in the Mark popup menu. Choose Find & Mark All from the Mark popup menu in the navigation bar. Then, decide how many levels you want to mark. In this example, the headings always have at least two digits and at most four.

Use this pattern to find the headings:

```
^(\d+\.\d+\.?\d*\.?\d*)[ \t]+([a-z ]+)
```

and this pattern to make the file marks:

```
\1 \2
```

The ^ before the first search group ensures that BBEdit matches the numeric string at the beginning of a line. The pattern

```
\.?\d*
```

matches a (possible) decimal point and a digit sequence. The other groups use the white space idiom and the identifier idiom. You can use a similar technique to mark any section that has a section mark that can be described with grep.

Marking a Mail Digest

You can elaborate the structured text technique to create markers for mail digests. Assume that each digest is separated by the following lines:

```
From: Sadie Burke <sadie@burke.com>
Date: Sun, 16 Jul 1995 13:17:45 -0700
Subject: Fishing with the judge
```

Suppose you want the marker text to list the subject and the sender. You would use the following search string:

```
^From:[ \t]+(.*)\r.*\rSubject:[ \t]+(.*)
```

And mark the text with this replacement string:

```
\2 \1
```

Note that for the sequence \r.*\r in the middle of the search string, the \r before "Subject" is necessary because as previously discussed, the special character . does not match a 'hard' line break. (At least, not by default. See "Advanced Topics," below, for details on how to make dot match *any* character, including line breaks.)

Rearranging Name Lists

You can use grep patterns to transform a list of names in first name first form to last name first order (for a later sorting, for instance). Assume that the names are in the form:

```
Junior X. Potter
Jill Safai
Dylan Schuyler Goode
Walter Wang
```

If you use this search pattern:

```
^(.*) ([^ ]+)$
```

And this replacement string:

```
\2, \1
```

The transformed list becomes:

```
Potter, Junior X.
Safai, Jill
Goode, Dylan Schuyler
Wang, Walter
```

Advanced Grep Topics

BBEdit's PCRE-based grep engine offers unparalleled syntactical power. The topics below cover areas that show how grep can effectively match very complicated patterns of text—matches which were impossible to achieve with older versions of BBEdit. However, with this power comes complexity.

If you aren't yet familiar with grep, it is possible that the topics covered in this section will not make much sense to you. That's OK. The best way to learn grep is to use it in real life, not by reading example patterns. In many cases, the basic grep syntax covered previously in this chapter will be all that you need.

If you are an experienced user of grep, however, many of the topics covered below will be of great interest.

Matching Nulls

To search for null charcters (ASCII value zero), you can use an hex escape:

```
\x00 or \x{0}
```

Backreferences

The following charts explain the rules BBEdit uses for determining backreferences.

In Search Patterns

Modifier	Effect
\0	A backslash followed by a zero is an octal character reference. Up to two further octal characters are read. Thus, "\040" will match a space character, and "\07" will match the ASCII BEL (\x07), but "\08" will match an ASCII null followed by the digit 8 (because octal characters only range from 0-7).
\1-9	A backslash followed by a single decimal digit from 1 to 9 is always a backreference to the *Nth* captured subpattern.
\10-99	A backslash followed by two decimal digits, which taken together form the integer N (ranging from 10 to 99), is a backreference to the *Nth* captured subpattern, *if* there exist *N* capturing sets of parentheses in the pattern. If there are fewer than *N* captured subpatterns, the grep engine will instead look for up to three octal digits following the backslash. Any subsequent digits stand for themselves. So, in a search pattern, "\11" is a backreference only if there are AT LEAST 11 sets of capturing parentheses in the pattern. If not, BBEdit interprets this as 11 octal (9 decimal) and matches a tab. "\011" always matches a tab. "\81" is a backreference if there are 81 or more captured subpatterns; otherwise, BBEdit will report an error, or, if your pattern contains at least 8 captured subpatterns, you may re-write this as \081, representing the 8th subpattern "\08" followed by the literal character "1".

In Character Classes

Modifier	Effect
OCTAL	Inside a character class, a backslash followed by up to three octal digits generates a single byte character reference from the least significant eight bits of the value. Thus, the character class "[\7]" will match a single byte with octal value 7 (equivalent to "\x07"). "[\8]" will match a literal "8" character.

In Replacement Patterns

Modifier	Effect
NNN+	If more than two decimal digits follow the backslash, only the first two are considered part of the backreference. Thus, "\111" would be interpreted as the 11th backreference, followed by a literal "1". You may use a leading zero; for example, if in your replacement pattern you want the first backreference followed by a literal "1", you can use "\011". (If you use "\11", you will get the 11th backreference, even if it is empty.)
NN	If two decimal digits follow the backslash, which taken together represent the value N, and if there is an Nth captured substring, then all three characters are replaced with that substring. If there is not an Nth captured substring, all three characters are discarded—that is, the backreference is replaced with the empty string.
N	If there is only a single digit N following the backslash and there is an Nth captured substring, both characters are replaced with that substring. Otherwise, both characters are discarded—that is, the backreference is replaced with the empty string. In replacement patterns, \0 is a backreference to the entire match (exactly equivalent to "&").

POSIX-Style Character Classes

BBEdit provides support for POSIX-style character classes. These classes are used in the form [:CLASS:], and are *only* available inside regular character classes (in other words, inside another set of square brackets).

Class	Meaning
alnum	letters and digits
alpha	letters
ascii	character codes 0-127
blank	horizontal whitespace
cntrl	control characters
digit	decimal digits (same as \d)
graph	printing characters, excluding spaces

Class	Meaning
lower	lower case letters
print	printing characters, including spaces
punct	punctuation characters
space	white space (same as \s)
upper	upper case letters
word	"word" characters (same as \w)
xdigit	hexadecimal digits

For example: [[:digit:]]+ is the same as: [\d]+

POSIX-style character class names are case-sensitive.

It is easy to forget that POSIX-style character classes are only available inside regular character classes. The pattern [:space:], without enclosing square brackets, is just a character class consisting of the characters ":", "a", "c", "e", "p", and "s".

The names "ascii" and "word" are Perl extensions; the others are defined by the POSIX standard. Another Perl extension supported by BBEdit is negated POSIX-style character classes, which are indicated by a ^ after the colon. For example, to match any run of non-digit characters:

```
[[:^digit:]]+
```

Non-Capturing Parentheses

As described in the preceding section "Creating Subpatterns", bare parentheses cluster and capture the subpatterns they contain. The portion of the matching pattern contained within the first pair of parentheses is available in the backreference \1, the second in \2, and so on.

Opening parentheses are counted from left to right to determine the numbers of the captured subpatterns. For example, if the following grep pattern:

```
((red|white) (king|queen))
```

is matched against the text "red king", the backreferences will be set as follows:

```
\1  "red king"
\2  "red"
\3  "king"
```

Sometimes, however, parentheses are needed only for clustering, not capturing. BBEdit now supports non-capturing parentheses, using the syntax:

```
(?:PATTERN)
```

That is, if an open parenthesis is followed by "?:", the subpattern matched by that pair of parentheses is not counted when computing the backreferences. For example, if the text "red king" is matched against the pattern:

```
(?:(red|white) (king|queen))
```

the backreferences will be set as follows:

```
\1  "red"
\2  "king"
```

Perl-Style Pattern Extensions

BBEdit's grep engine supports several extended sequences, which provide grep patterns with super-powers from another universe. Their syntax is in the form:

(?*KEY…*)

in other words, an open parenthesis followed by a question mark, followed by a KEY for the particular grep extension, followed by the rest of the subpattern and a closing parenthesis. This syntax—specifically, an open parenthesis followed by a question mark—was not valid in older versions of BBEdit, thus, none of these extensions will conflict with old patterns.

We have already seen one such extension in the previous section of this document—non-capturing parentheses: (?:…). The remainder are listed in the chart below, and discussed in detail afterward.

Extension	Meaning
(?:…)	Cluster-only parentheses, no capturing
(?#…)	Comment, discard all text between the parentheses
(?imsx-imsx)	Enable/disable pattern modifiers
(?imsx-imsx:…)	Cluster-only parens with modifiers
(?=…)	Positive lookahead assertion
(?!…)	Negative lookahead assertion
(?<=…)	Positive lookbehind assertion
(?<!…)	Negative lookbehind assertion
(?()…|…)	Match with if-then-else
(?()…)	Match with if-then
(?>…)	Match non-backtracking subpattern ("once-only")
(?R)	Recursive pattern

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

Search for: `foo(?# Hello, this is a comment)bar`
Will match: foobar

Pattern Modifiers

The settings for case sensitivity, multi-line matching, whether the dot character can match returns, and "extended syntax" can be turned on and off within a pattern by including sequences of letters between "(?" and ")".

Modifier	Meaning	Default
`i`	case insensitive	according to Case Sensitive checkbox in Find window
`m`	allow ^ and $ to match at \r	on
`s`	allow . to match \r	off
`x`	ignore most white space and allow inline comments in grep patterns	off

i — By default, BBEdit obeys the "Case Sensitive" checkbox in the Find window (or the corresponding property of the search options when using the scripting interface). The (?i) option overrides this setting.

m — By default, BBEdit's grep engine will match the ^ and $ metacharacters after and before returns, respectively. If you turn this option off with (?-m), ^ will only match at the beginning of the document, and $ will only match at the end of the document. (If that is what you want, however, you should consider using the \A, \Z, and \z metacharacters instead of ^ and $.)

s — By default, the magic dot metacharacter . matches any character except return ("\r"). If you turn this option on with (?s), however, dot will match *any* character. Thus, the pattern (?s).+ will match an entire document.

x — When turned on, this option changes the meaning of most whitespace characters (notably, tabs and spaces) and #. Literal whitespace characters are ignored, and the # character starts a comment that extends until a literal return or the "\r" escape sequence is encountered. Ostensibly, this option intends to let you write more "readable" patterns.

Perl programmers should already be familiar with these options, as they correspond directly to the -imsx options for Perl's m// and s/// operators. Unadorned, these options turn their corresponding behavior on; when preceded by a hyphen (-), they turn the behavior off. Setting and unsetting options can occur in the same set of parentheses.

Example	Effect
`(?imsx)`	Turn all four options on
`(?-imsx)`	Turn all four options off
`(?i-msx)`	Turn "i" on, turn "m", "s", and "x" off

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern, the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

In other words, all four of the above patterns will match without regard to case. Such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at the top level, the right-most setting is used.

If an option change occurs inside a subpattern, the effect is different. An option change inside a subpattern affects *only* that part of the subpattern that follows it, so, if the "Case Sensitive" checkbox is turned on:

Search for: `(a(?i)b)c`
Will match: `abc` *or* `aBc`

and will not match anything else. (But if "Case Sensitive" is turned off, the "(?i)" in the above pattern is superfluous and has no effect.) By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example:

Search for: `(a(?i)b|c)`

matches "ab", "aB", "c", and "C", even though when matching "C", the first branch is abandoned before the option setting.

These options can also be set using the clustering (non-capturing) parentheses syntax defined earlier, by inserting the option letters between the "?" and ":". The scope of options set in this manner is limited to the subpattern contained therein. Examples:

Search for: `(?i:saturday|sunday)`
Will match: `SATURDAY` *or* `Saturday` *or* `SUNday` (and so on)

Search for: `(?i:foo)(?-i:bar)`
Will match: `foobar` *or* `FOObar`
Will not match: `FOOBAR` *or* `fooBAR`

Positional Assertions

Positional assertions "anchor" a pattern, without actually matching any characters. Simple assertions have already been described: those which are invoked with the escape sequences \b, \B, \A, \Z, \z, ^ and $. For example, the pattern \bfoo\b will only match the string "foo" if it has word breaks on both sides, but the \b's do not themselves match any characters; the entire text matched by this pattern are the three characters "f", "o", and "o".

Lookahead and lookbehind assertions work in a similar manner, but allow you to test for arbitrary patterns to anchor next to. If you have ever said to yourself, "I would like to match 'foo', but only when it is next to 'bar'," lookaround assertions fill that need.

Positive lookahead assertions begin with "(?=", and negative lookahead assertions begin with "(?!". For example:

```
\w+(?=;)
```

will match any word followed by a semicolon, but the semicolon is not included as part of the match.

```
foo(?!bar)
```

matches any occurrence of "foo" that is *not* followed by "bar". Note that the apparently similar pattern:

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion (?!foo) is always true when the next three characters are "bar". A lookbehind assertion is needed to achieve this effect.

Positive lookbehind assertions start with "(?<=", and negative lookbehind assertions start with "(?<!". For example:

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=Martin|Lewis)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is different compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. (Lookbehinds in conjunction with non-backtracking [a.k.a. "once-only"] subpatterns can be particularly useful for matching at the ends of strings; an example is given in the section on once-only subpatterns below.)

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches “foo” preceded by three digits that are not “999”. Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, and then there is a check that the same three characters are not “999”. This pattern does not match “foo” preceded by six characters, the first of which are digits and the last three of which are not “999”. For example, it does not match “123abcfoo”. A pattern to do that is:

```
(?<=\d{3}...)(?<!999)foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three characters are not “999”. Assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of “baz” that is preceded by “bar” which in turn is not preceded by “foo”, while

```
(?<=\d{3}(?!999)...)foo
```

is another pattern which matches “foo” preceded by three digits and any three characters that are not “999”.

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Conditional Subpatterns

Conditional subpatterns allow you to apply “if-then” or “if-then-else” logic to pattern matching. The “if” portion can either be an integer between 1 and 99, or an assertion.

The forms of syntax for an ordinary conditional subpattern are:

if-then:	`(?(condition)yes-pattern)`
if-then-else:	`(?(condition)yes-pattern|no-pattern)`

and for a named conditional subpattern are:

if-then:	`(?P<`*NAME*`>(condition)yes-pattern)`
if-then-else:	`(?P<`*NAME*`>(condition)yes-pattern|no-pattern)`

If the condition evaluates as true, the “yes-pattern” portion attempts to match. Otherwise, the “no-pattern” portion does (if there is a “no-pattern”).

If the “condition” text between the parentheses is an integer, it corresponds to the backreferenced subpattern with the same number. (Do not precede the number with a backslash.) If the corresponding backreference has previously matched in the pattern, the condition is satisfied. Here’s an example of how this can be used. Let’s say we want to match the words “red” or “blue”, and refer to whichever word is matched in the replacement pattern. That’s easy:

```
(red|blue)
```

To make it harder, let's say that if (and only if) we match "blue", we want to optionally match a space and the word "car" if they follow directly afterward. In other words, we want to match "red", "blue", or if possible, "blue car", but we do not want to match "red car". We cannot use the pattern:

```
(red|blue)( car)?
```

because that will match "red car". Nor can we use:

```
(red|blue car|blue)
```

because in our replacement pattern, we want the backreference to only contain "red" or "blue", without the " car". Using a conditional subpattern, however, we can search for:

```
((blue)|(red))(?(2) car)?
```

Here's how this pattern works. First, we start with "((blue)|(red))". When this subpattern matches "blue", \1 and \2 are set to "blue", and \3 is empty. When it matches "red", \1 and \3 are set to "red", and \2 is empty.

Next comes the conditional subpattern "(?(2) car)?". The conditional test is on "2", the second backreferenced subpattern: if \2 is set, which in our case means it has matched the word "blue", then it will try to match " car". If \2 is not set, however, the entire conditional subpattern is skipped. The question mark at the end of the pattern makes this conditional match optional, even if \2 is set to "blue".

Here's an example that uses an assertion for the condition, and the if-then-else form. Let's say we want to match a run of digits of any length, followed by either " is odd" or " is even", depending on whether the matched digits end with an odd or even digit.

```
\d+(?(?<=[13579]) is odd| is even)
```

This pattern starts with "\d+" to match the digits. Next comes a conditional subpattern, with a positive lookbehind assertion as the condition to be satisfied. The lookbehind assertion is true only if the last character matched by \d+ was also in the character class [13579]. If that is true, we next try to match " is odd"; if it is not, we try to match " is even". Thus, this pattern will match "123 is odd", "8 is even", and so on, but will not match "9 is even" or "144 is odd".

Once-Only Subpatterns

With both maximizing (greedy) and minimizing (non-greedy) repetition, failure of what follows normally causes the repeated item to be reevaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it to fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern "\d+foo" when matching against the text "123456bar".

After matching all 6 digits and then failing to match “foo”, the normal action of the grep engine is to try again with only 5 digits matching the \d+ item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be reevaluated in this way, so the matcher would give up immediately on failing to match “foo” the first time. The notation is another kind of special parenthesis, starting with “(?>”, as in this example:

```
(?>\d+)bar
```

This kind of parentheses “locks up” the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

In most situations, such as in the example above, the time saved by using once-only subpatterns is insignificant—a few small fractions of a second, at most. With some complicated grep patterns or with humongous lines of text, however, you can save tremendous amounts of time using once-only subpatterns.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both \d+ and \d+? are prepared to adjust the number of digits they match in order to make the rest of the pattern match, (?>\d+) can only match an entire sequence of digits.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of a line of text. Consider a simple pattern such as:

```
abcd$
```

when applied to a long line of text which does not match (in other words, a long line of text that does not end with “abcd”). Because matching proceeds from left to right, the grep engine will look for each “a” in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as:

```
^.*abcd$
```

the initial .* matches the entire line at first, but when this fails (because there is no following “a”), it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for “a” covers the entire string, from right to left, so we are no better off. However, if the pattern is written as:

```
^(?>.*)(?<=abcd)
```

there can be no backtracking for the .* item; it can match only the entire line. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the whole match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time (literally millions or even billions of years, in some cases!). The pattern:

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in <>, followed by either ! or ?. When it matches, it runs quickly. However, if it is attempts to match this line of text:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. So long, in fact, that it will effectively "freeze" BBEdit. This is not really a crash, per se, but left to run on its own, it might take years before it finally fails. (We are not sure, frankly, because much like determining how many licks it takes to get to the center of a Tootsie Pop, we do not feel like waiting long enough to find out.)

The reason this takes so long to fail is because the string can be divided between the two repeats in a large number of ways, and all have to be tried before the grep engine knows for certain that the pattern will not match. (The example used [!?] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.) If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?]
```

sequences of non-digits cannot be broken, and failure happens quickly.

Recursive Patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested, balanced parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (among other things). It does this by interpolating Perl code in the expression at run time, and the code can refer to the expression itself. Obviously, BBEdit's grep engine cannot support the interpolation of Perl code. Instead, the special item (?R) is provided for the specific case of recursion. The following recursive pattern solves the parentheses problem:

```
\(((?>[^()]+)|(?R))*\)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (that is, a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it tries to match against this line of text:

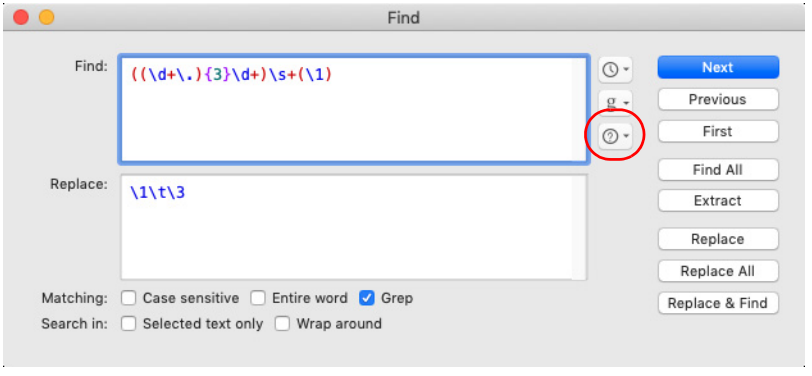
```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

it yields "no match" quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the + and * repeats can carve up the subject, and all have to be tested before failure can be reported.

Search Window Grep ‘cheat sheet’

The Find and Multi-File Search windows, as well as pattern playground windows, now offer “cheat sheets” for both the “Find” and “Replace” fields to serve as an at-a-glance grep syntax reference without needing to leave BBEdit or even change to a different window.

Click the Cheat Sheet button (circled) to display the grep cheat sheet:



and you can select any entry to insert the corresponding pattern into the active field.

abc123	literal text
\d	Any digit
\D	Any non-digit
.	Any character
\.	Period
[abc]	Character class (only a, b, or c)
[^abc]	Character class (anything but a, b, or c)
[a-z]	Character class (letters from a to z)
[0-9]	Character class (numbers from 0 to 9)
[0-9,a-f]	Character class (any hex digit)
[PARTY*PODS]	Character class (that’s a literal asterisk!)
[:xdigit:]	Character class (POSIX hex digit)
[:punct:]	Character class (POSIX punctuation)
[:^punct:]	Character class (anything except punctuation)
\w	Any “word” (alphanumeric) character
\W	Any “non word” (punctuation, space, etc) character
{x}	Repeated X times
{x,y}	Repeated X to Y times
{x,}	Repeated at least X times
*	Repeated zero or more times
+	Repeated one or more times
?	Repeated zero or one times
\s	Any whitespace character (includes line breaks)
\S	Any non-whitespace character
\n	Character escape: new line
\t	Character escape: tab
\\	Character escape: literal backslash
\xNN	One-byte hex escape: general form
\x{NNNN}	Two-byte hex escape: general form
^	Match must occur at the beginning of a line
$	Match must occur at the end of a line
(abc)	Subpattern
(a(bc))	Nested subpattern
\1	Reference to subpattern #1
\2	Reference to subpattern #2
(?P<foo>abc)	Create named subpattern “foo” matching “abc”
(?P=foo)	Reference to named subpattern “foo”
ab|cd	Alternation (matches “ab” or “cd”)
x(?=abc)	“abc” must occur after “x”
x(?!abc)	“abc” must not occur after “x”

Pattern Playgrounds

Pattern Playground window offer an interactive interface for experimenting with the behavior of grep patterns. This makes the process of creating complicated patterns much less trial-and-error, since you can see exactly what will match, and how, before committing to any irreversible actions.

You can choose “Pattern Playground” in the Search menu to open a playground window, or make another playground window at any time by choosing Pattern Playground in the New submenu of the File menu. (You can have multiple playground windows open at the same time.)

Using a Pattern Playground

Here’s a brief overview of how you can make use of a pattern playground:

The “Search pattern” field is where you’ll do most of your work. You can choose an existing pattern from the Saved Patterns popup (which offers the same set of saved patterns as provided in the Find window and elsewhere), or start typing in a pattern from scratch. BBEdit will check the pattern ‘live’ as you type and if the pattern is valid, BBEdit will search whatever text is displayed in the “Contents of” section (see below).

The Search History popup provides a (window-local) history of patterns in which you have successfully used the “Next” button and you can use this menu to easily switch between different iterations of a working pattern.

The “Search results” and “Next” button will become enabled whenever a pattern matches anything in the “Contents of” section. The results line will show you how many matches there are for the specified pattern. The “Next” button will highlight either the next match to the entire pattern, or if you have selected a specific capture group (see below), the next match to the selected capture group only.

You can click the “Use for Find” button to transfer the current search (and if applicable, replace) patterns to BBEdit’s global search state, thus setting up the Find and Multi-File Search windows and turning on the “Grep” search option.

The “Capture groups” list will show you all of the defined capture groups in the Grep pattern, including the implicit “\0” capture group which represents the entire pattern.

NOTE If your pattern contains positional assertions, "\0" may not represent the entirety of the pattern match!

The “#” column always shows the capture group number. If the pattern contains any named capture groups (e.g. (?P<foobar>)), the “Name” column will display the name of the corresponding capture group (e.g. “foobar”).

The list is populated for any valid pattern. If you perform a search, however, more data gets filled in. The “Text” column will display the text which is actually contained by the capture group; so by selecting different groups, you can see how the pattern internally matches your text, and then adjust the pattern as desired.

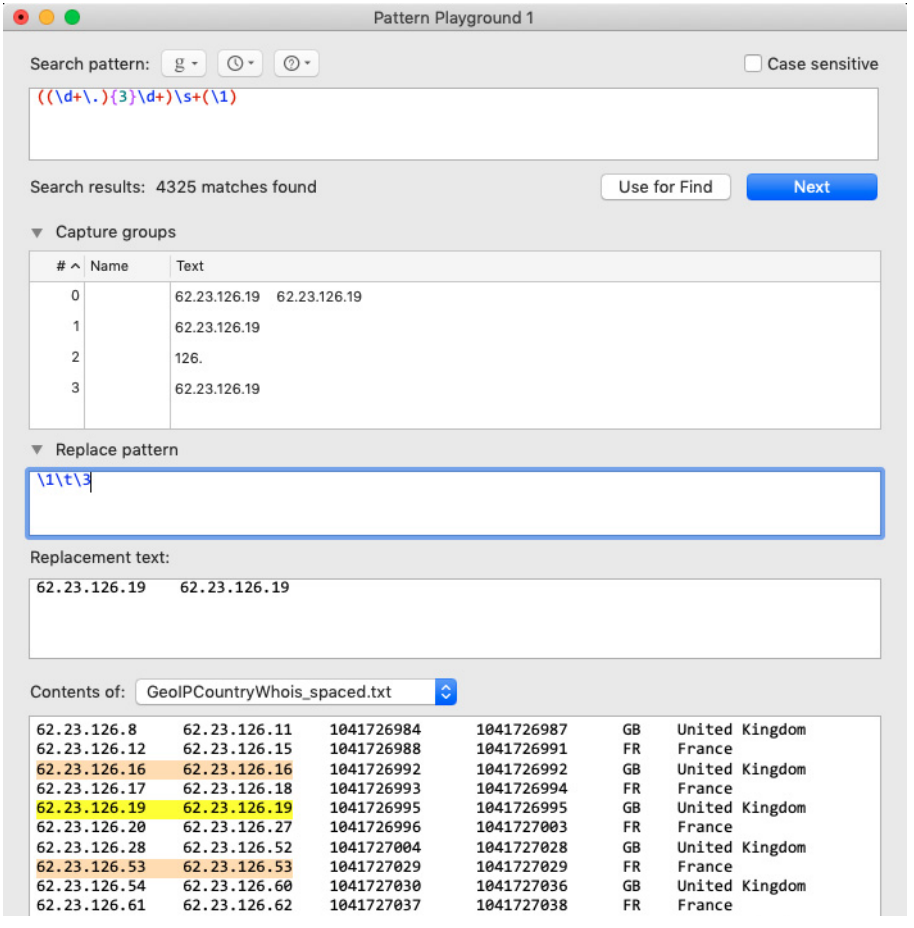
You can further use the “Replace Pattern” field to experiment with substitions. As you edit, the “Replacement Text” field will show the results of applying the replacement pattern to the current match.

The “Contents of” section shows a popup menu which lists all open documents, in alphabetical order. When you choose a document, the text area will fill with a copy of that document’s contents. (The text area is not editable, but any changes you make to the actual document in its own window will be reflected in the text area.)

The first item on the document menu is “Scratch Space”. This makes the text area at the bottom editable (it isn’t normally), so you can paste in text from somewhere else for testing, and make small edits. You should not rely on this area as a general purpose document editor.

When you have entered a search pattern that generates matches, BBEdit will now highlight all non-selected matches to that pattern in the same manner as the Live Search command, while the selected (active) match will be highlighted as usual.

You can click the “Next” button to proceed to the next match.



Additional notes and behaviors

The pattern tester is connected to the equivalent Search menu commands, so that choosing Find Next, or typing its key shortcut will do the same thing as the “Next” button.

When a pattern playground window is active, the “Use Selection for Find” command will change to “Use Pattern for Find”. This allows use of the appropriate key shortcut for the “Use for Find” button.

The Capture Groups and Replacement Pattern sections within a pattern playground window are collapsible.

If you make edits to a pattern, and then close the Pattern Playground window without either saving it or clicking “Use for Find”, BBEdit will prompt you, and give you a chance to save the pattern.

CHAPTER

9 Browsers, Notes and Workspaces

Browsers are special kinds of windows that let you see a lot of information about files at once. Browsers typically have two panes: one pane lets you select a file, the other displays detailed information about the file (often its contents). If you have performed a Find All search, you have already seen an example of a BBEdit browser.

In this chapter

Browser Overview

All BBEdit browsers share the same basic structure and behavior. All browsers have a results bar, a sidebar or results list, and a text pane. You can either edit files directly in any browser window, or open them separately.

List Pane

The top pane of a browser lists the items available in the browser. This pane shows different information for different kinds of browsers:

Browser	File pane contains
Disk browser	Sidebar lists files and folders in the current directory
Search results	File and line number of each match
Error results (or) general results	File, line number, and status message for each condition

You can open both files and folders from the sidebar pane. When you double-click a folder name, BBEdit replaces the file list pane with the contents of the folder. When you double-click a file name, BBEdit opens the file in an editing window. If the file list pane also included a line number, BBEdit scrolls to that line.

Controls above the list may allow you to determine what kinds of items are displayed in the list. For example, in disk browsers, there is a popup menu that lets you choose to display text files, all files, or other types of files, and another that lets you return the browser to a parent directory of the current folder. In error browsers, checkboxes allow you to hide or show all errors, warnings, or notes.

To remove items from the display list, select them and press the Delete key, or choose Clear from the Edit menu.

In results browsers, you may Control-click on items in the list to bring up the contextual menu with relevant commands—primarily “Copy”.

Navigation Bar

The browser navigation bar is like the navigation bar in editing windows. Some browsers have additional buttons and controls in the status area as well.

The standard items in this bar should already be familiar to you, since they appear on BBEdit document windows by default. See “Window Anatomy” in Chapter 4 for an explanation of these standard BBEdit functions.

Text View Pane

When you click on a file name in the list pane, BBEdit displays that file in the text view pane, and you can edit the file just as if it were open in a document window.

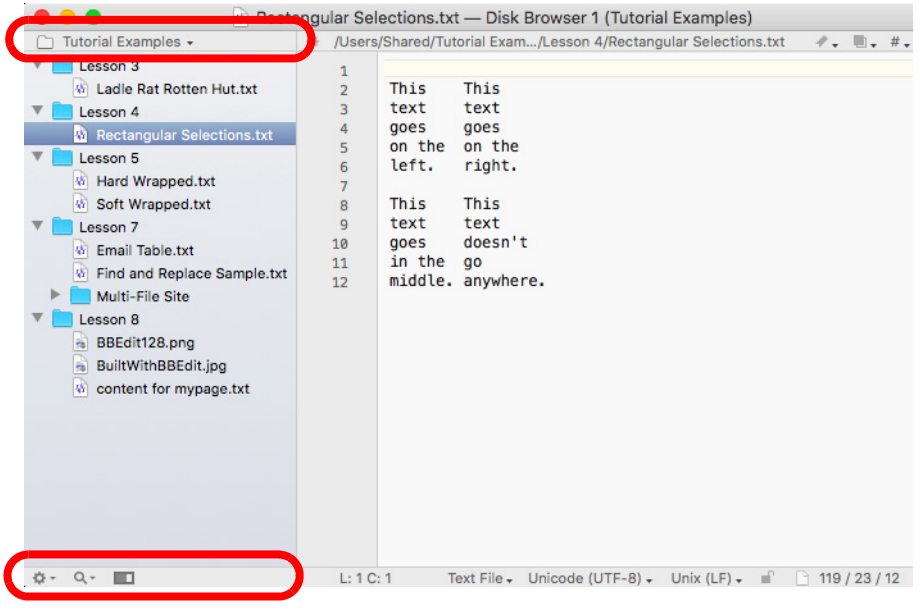
Splitter

You can change the size of the file list pane or the text view pane by dragging the double line that separates them. Double-clicking on the splitter bar will collapse the text view pane completely, and double-clicking on it again (in the bottom of the browser window) will restore the text pane to its previous proportions. You can also choose the Hide Editor or View Editor commands in the View menu to hide or display the text view pane.

Disk Browsers

Use a disk browser to explore the contents of a disk or a folder without opening each file one at a time.

To open a disk browser, pull down the File menu and choose Disk Browser from the New submenu. BBEdit will prompt you to choose a directory to start browsing in, and you can select any desired (and permitted) location:



The name and path of the file (if any) and directory currently being viewed are displayed in the title bar of the window. The sidebar pane displays all the items in the current folder. Click on a file in the sidebar pane to open it in the text pane, or double-click to open the file into a text window.

Disk Browser Controls

The menus at the top and bottom of the sidebar pane let you create new files and folders, open existing files and folders, reveal them in the Finder or navigate to them in the Terminal, limit the kinds of files to show in the list pane, and navigate through your disks and folders.

Directory Menu

The Directory popup menu at the top of the sidebar pane always shows the currently active folder. You can use this menu to “back out” of any folder you are currently in to a higher-level folder (as you can by Command-clicking the name of a folder in the Finder).

Action Menu

The commands on the Action (gear) popup menu at the bottom of the sidebar pane allow you to open the selected items, reveal them in the Finder, copy their paths, navigate to their location in the Terminal, move them to the Trash, or create a new file or folder.

Filter Menu

The Filter (magnifying glass) popup menu at the bottom of the sidebar pane lets you specify what kinds of files BBEdit should display:

- All Available: All files which BBEdit recognizes, including its own document types. This includes text files, project documents, text factories, compressed archives, and so on.
- Text Files Only: Only files which BBEdit recognizes as text files.
- Everything: All items present, including invisible files and folders.

You can also select a file filter to further limit what files BBEdit should display. (You can define additional file filters in the Filters panel of the Setup window.)

Disk browser and project windows now present a separate “Hidden Items” file filtering option (via the magnifying glass at the bottom of the sidebar); this option separates file type filtering from visibility filtering, so (for example) you can show only dotfiles

(and dotfolders) that are text files.

Toggle Editor Button

Click this button to collapse or expand the browser’s text view pane. (This button has the same effect as choosing the View/Hide Editor command in the View menu.)

Contextual Menu Commands

If you select one or more items in the sidebar pane and bring up the contextual menu, BBEdit will offer a variety of commands including those available from the Action menu.

Dragging Items

You can select and drag files and folders from a disk browser’s sidebar to any location, either within BBEdit or elsewhere, which can accept file or folder drags. For example, you can drag a file from a disk browser to a project window to add it to that project, or to an editing window to insert its contents, or to a folder in the Finder to copy or move it.

Using the List Pane in Disk Browsers

The list pane of a disk browser displays disks, files, and folders. When you are at the computer level, the list shows all mounted volumes.

When you click a folder or disk in the list pane, BBEdit displays the names of all the files it can open in the text pane, subject to the criteria specified by the Show and Filter menus.

When you click a file name in the list pane, BBEdit displays that file in the text pane.

To open a folder or disk and display its contents in the sidebar pane, you can either double-click it, or Select it and press Command-Down Arrow.

To go up one level to the enclosing folder or disk, you can either choose the enclosing folder from the directory popup menu, or press Command-Up Arrow

You can also use Quick Look to examine any non-text file by selecting it and pressing the spacebar.

Note When the list pane has input focus, the browser window's AppleScript "selection" property will return a list of the files currently selected. See "Getting and Setting Properties" on page 369 for further details.

Viewing Image Files

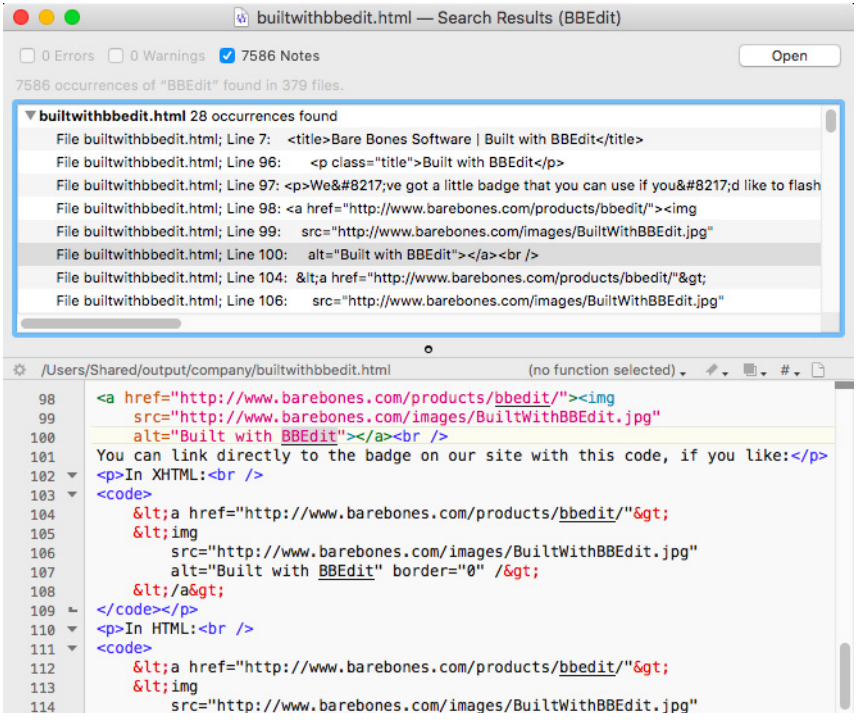
BBEdit supports viewing image files within disk browsers and projects, as well as ordinary window sidebars. The image view includes metadata details, and a "Remove Metadata" button is available for deleting image metadata if desired. (BBEdit will create a macOS version snapshot of the image file before doing so, if possible.) BBEdit will attempt to pick a background color (shade of gray) that provides reasonable contrast with the image. If desired you can adjust the background color using the slider adjacent to the image, and BBEdit will remember this setting per-image.

The File Info item (in the navigation bar) and "Get Info" command (from the menu bar or contextual menu) will show an image thumbnail and the image metadata in respective tabs in the info popover.

For non-image files, the "Metadata" tab in the "Get Info" will display (in a vaguely human-readable presentation) the file system and/or Spotlight metadata available for the item. You can also use Quick Look to examine any non-text file by selecting it and pressing the spacebar.

Search Results Browsers

If you selected the Batch Find option when performing a multi-file search, BBEdit displays every occurrence of the search string in the searched files in a search results browser.



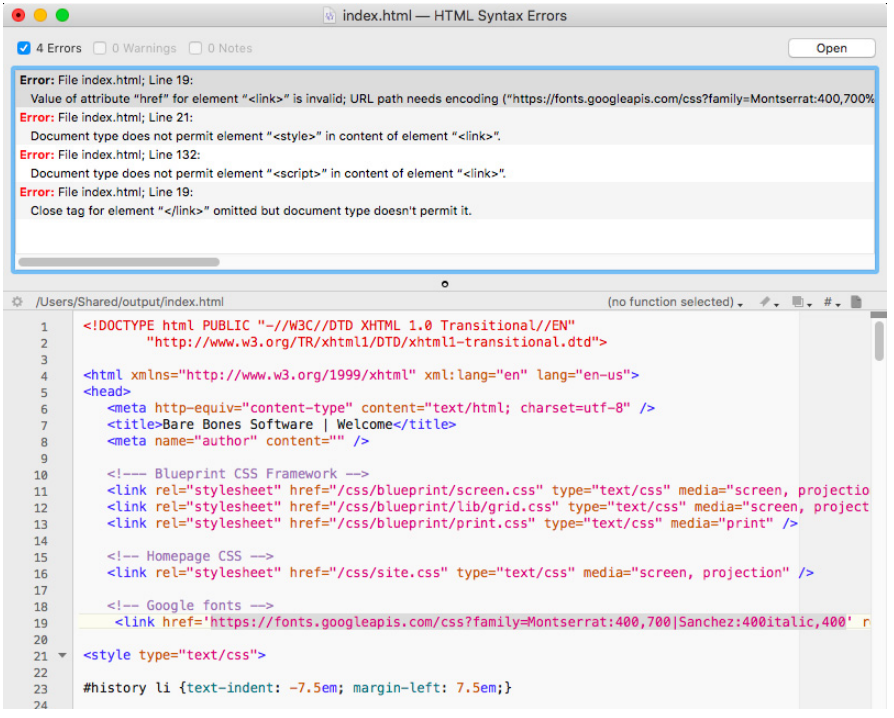
The items at the top of the window tell you how many matches BBEdit found in the set of files you specified, as well as whether any error conditions or warnings were generated during the search. The list pane lists each line that contains the matched text. Every match is identified by file and line number. To choose whether to display the search errors, warnings, and results, use the checkboxes at the top of the browser.

To open the file which contains a particular match, just click on that match in the results list. After you have opened a file, you can use the Search menu commands to continue searching it. (See Chapter 7 for more information on searching.)

The Open button opens the selected items using BBEdit. To open the selected items using the Finder, hold down the Option key while clicking the Open button.

Error Results Browsers

When you use the HTML syntax checker, link checker, or update tool, BBEdit will open an error results browser to display any errors generated by the command. BBEdit will also open an error results browser to list errors generated by Perl or Python scripts.



Each entry in the list pane corresponds to an error, warning, or note. You can use the checkboxes for each type of item to suppress or display the associated results as desired.

If you click on a entry in the sidebar, BBEdit will open the corresponding file in the text display pane and select the section of text related to the error.

The Notes window and Notebooks

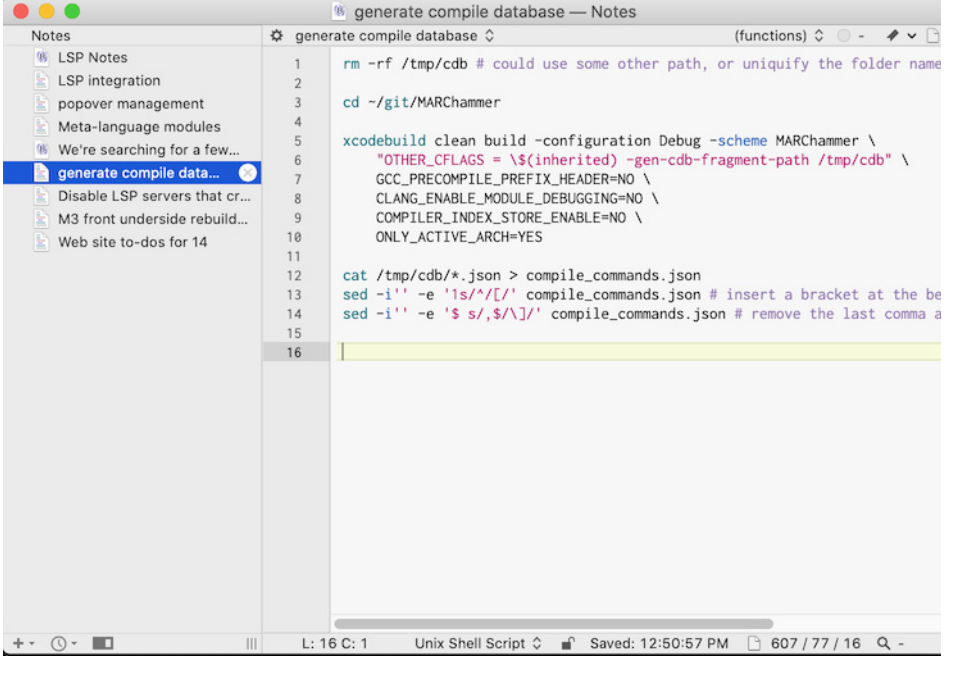
BBEdit now offers both a dedicated, default Notes window and the ability to create multiple notebooks to help you easily store and manage information.

To get started, just select the Notes command in the Window menu to open the global Notes window and you can then create individual notes via the “New Note” command (or a variety of additional options).

BBEdit can also create multiple notebooks using the Notebook... command in the New submenu of the File menu and you can then employ standard gestures to open them from the Finder. Notebooks can reside anywhere you like. If using a file sharing service (Dropbox, iCloud Drive, etc) you should take care to ensure that the notebook is open in only one running BBEdit at a time. Any additional notebooks that you create are supplementary to the built-in shared “Notes” notebook. Open notebooks, like projects, will also automatically be listed in the search target list of the Multi-File Search window.

Notes are mostly like ordinary text documents, except that you don’t have to remember to save them or even make up a name if you don’t want to. BBEdit keeps notes all together in a “notebook”. Notes exist on disk as text files; there’s no secret file format involved, and by default BBEdit treats notes as Markdown documents, but you can change a note’s language type in the usual fashion if doing so would be convenient.

Although notes are stored as text files, some of BBEdit’s commands for working with text files are disabled or modified when a note is active, in order to avoid confusion. (For example, you can’t relocate a note file since that would break lots of things, but you *can* use “Save a Copy” to save a copy of a note’s contents as an ordinary text file somewhere else.



Project-Specific Settings

BBEdit now supports applying custom color schemes to projects, instaprojects, and notebooks for project- (and notebook-) windows. Settings applied here will override the global settings (including, subject to setting, any language-specific overrides).

For project documents that have been saved to disk, the color scheme settings are in the project settings visible when clicking on the first item in the sidebar (corresponding to the project itself). For notebooks and instaprojects (created by opening a folder), settings are available via the small gear in the action bar at the bottom of the sidebar.

Making Notes

There are many ways to make a note, so you can use whatever fits your workflow and style:

- Choose “Note” from the “File → New” menu. This makes a new empty note, into which you can type or paste text.
- Choose “Note (with selection)” from the “File” → New menu. If you have a range of selected text in a BBEdit text document, this command is enabled and choosing it will create a new note with the selected text.
- Choose “Note (with Clipboard)” from the “File → New” menu. If this command is enabled, choosing it makes a new note with the contents of the Clipboard.
- Choose “Save as Note” from the File menu. This command is available for untitled text documents (which have never been saved to disk).
- Right-click on some selected text in a BBEdit editing window and choose “New Note (with selection)” from the contextual menu.
- Right-click on some selected text in another application, and choose “New Note in BBEdit” from the Services submenu of the contextual menu; or from the Services submenu of the ? menu.
- Open the Notes window (on the Window menu), and use the “+” button at the bottom of the sidebar, or the contextual menu in the sidebar, to add a new empty note.
- With the Notes window open and its sidebar visible, drag a file or some text into the sidebar. If you drag a file, the original file is left undisturbed.
- After installing the command line tools, pipe some text into `bbedit --note`; BBEdit will create a new note from the text. Additional details are available in the `bbedit(1)` man page.

Working with Notes

The “Notes” command on the Window menu opens the Notes window, which displays all of the notes that you’ve created. Drag notes in the sidebar to rearrange them; you can also use the “+” menu or contextual menu in the sidebar to create Collections to organize your notes.

When you create a note, and as you edit it, BBEdit will change the note's title based on the text at the beginning of the note. If you would like to make a note's name persistent, just right-click on that note in the sidebar and choose the "Rename Note" command from the contextual menu. The name you apply there will become that note's name forever (or until you "Rename Note" again), and subsequent edits to the note's content won't affect the note's name.

To permanently delete a note, use the "Remove" command in the Notes window's sidebar contextual menu. You can right-click on any note to remove it; or on a selected range of notes; or on a selected range of collections and notes. (Removing a collection will remove all of the notes within it.) You'll need to confirm that you wish to permanently delete the note(s), since the operation is not undoable. When you delete a note, BBEdit will place the note's backing file in the Trash, so if you change your mind you can open the file and recover its contents.

When a note is the active document, or is open in a window by itself, the "Close & Delete" command on the File menu will change to "Remove Note" and choosing it will close the note and remove it from its notebook. (As always you can assign a keyboard equivalent to this command via the "Menus & Shortcuts" settings pane.)

If for some reason you need to export all of your notes, use the "Export Notes" command on the File menu. This will create a folder with all of your notes in it, with subfolders for collections. This command is always available if any notes exist.

Notebook windows now also present a text search field at the bottom of the sidebar. You can use this field to search for string matches in note names as well as contents.

Use the "Notes" item in the list of search sources in the Multi-File Search window to have BBEdit search your notes when doing a multi-file search (or replace, or extract).

You can quickly go to a note using the "Open File by Name" window; enter all or part of a note's title, and matching notes (and other files) will appear in the results.

You can arrange notes in a notebook's sidebar by clicking on the popup indicator on the right-hand side of the sidebar header. There are options for arranging by name, modification date, or creation date.

Workspaces

BBEdit now offers "workspaces" as a convenient way to switch between arrangements of open documents and windows. A workspace includes the same application state that is saved and restored across quit and relaunch, but can be activated at any time while the application is running. For example, this is useful when switching between working setups for different clients, or for different types of projects (writing vs programming vs web development).

The applicable menu commands reside on the BBEdit application menu. To create a new workspace, choose "Save Workspace" and give the workspace a name. To activate a previously saved workspace, choose it from the "Workspaces" submenu. You can overwrite an existing workspace by specifying the same name.

When switching to a different workspace, BBEdit will ask you to save any unsaved documents which correspond to existing files on disk (or remote FTP/SFTP items). You must save these before switching workspaces. Untitled (never-saved) documents are not affected by workspace changes, and will remain open (and unsaved) when you switch workspaces.

Saved workspaces reside in the "Workspaces" application support folder, accessible via the Folders submenu on the application (BBEdit) menu.

If you wish to remove an inactive workspace, deleting the corresponding file will accomplish this. Existing workspaces can be renamed by renaming the corresponding file (whose extension must be preserved when doing so).

If there are any saved workspaces, BBEdit's Dock menu will have a Workspaces submenu, so that you can easily switch via the Dock if desired.

BBEdit does **not** update a workspace as you open or close documents or move windows around. If you make significant changes and would like to update the workspace, use "Save Workspace" and overwrite the current workspace. (For convenience, BBEdit will fill in the name of the current workspace when you use "Save Workspace".)

Addtionally, BBEdit will prompt you to update a workspace when switching, if necessary. You can control this behavior in the Application settings, using the “When changing workspaces” group of settings.

CHAPTER

10 Settings

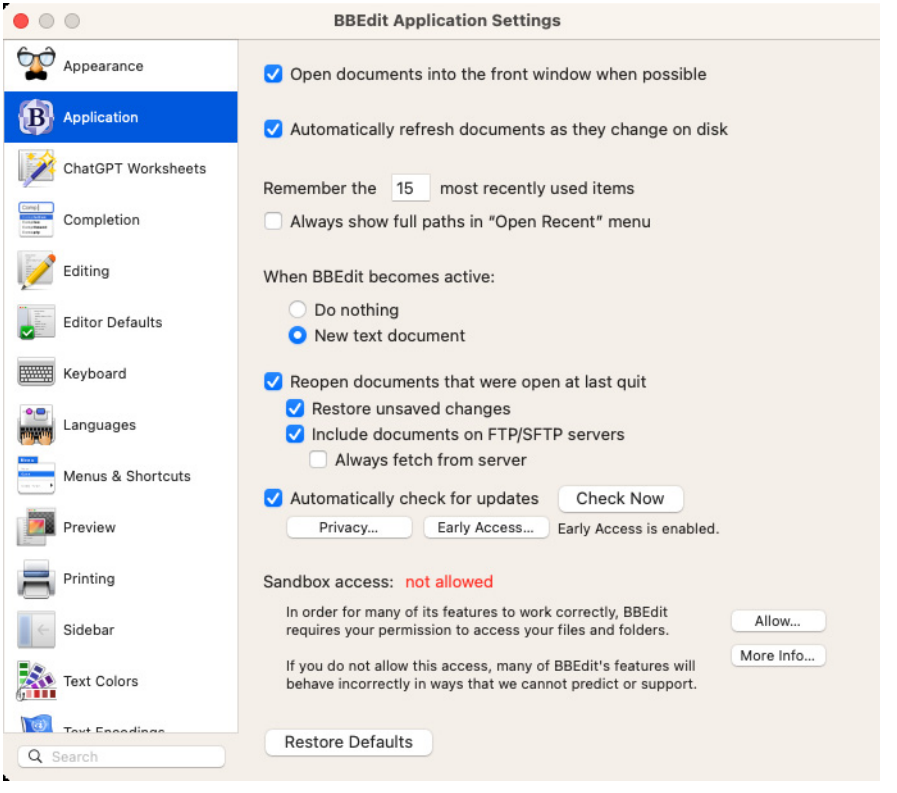
You can use the Settings command to customize much of BBEdit's behavior. You can decide which windows are open when you launch BBEdit, set the default options for windows, set the default options for searches, and so on. This chapter describes BBEdit's extensive settings options.

In this chapter

The Settings Window

The Settings window provides control over many aspects of BBEdit's behavior. You can decide which actions BBEdit should perform when you launch it, set default options for editing behavior, examine and set or modify keyboard shortcuts, create and apply text color schemes, and so on.

To open the Settings window, choose the Settings command from the BBEdit menu.



To select a settings panel, click its name in the list at the left side of the window. The text area at the top of the Settings window gives you a brief description of the options provided by the currently displayed settings panel.

BBEdit's Settings window is non-modal: you can leave it open and change settings settings while you work, or close it at any time by clicking its close button or by choosing Close Window from the File menu. Any changes you make to settings options take effect immediately unless otherwise indicated.

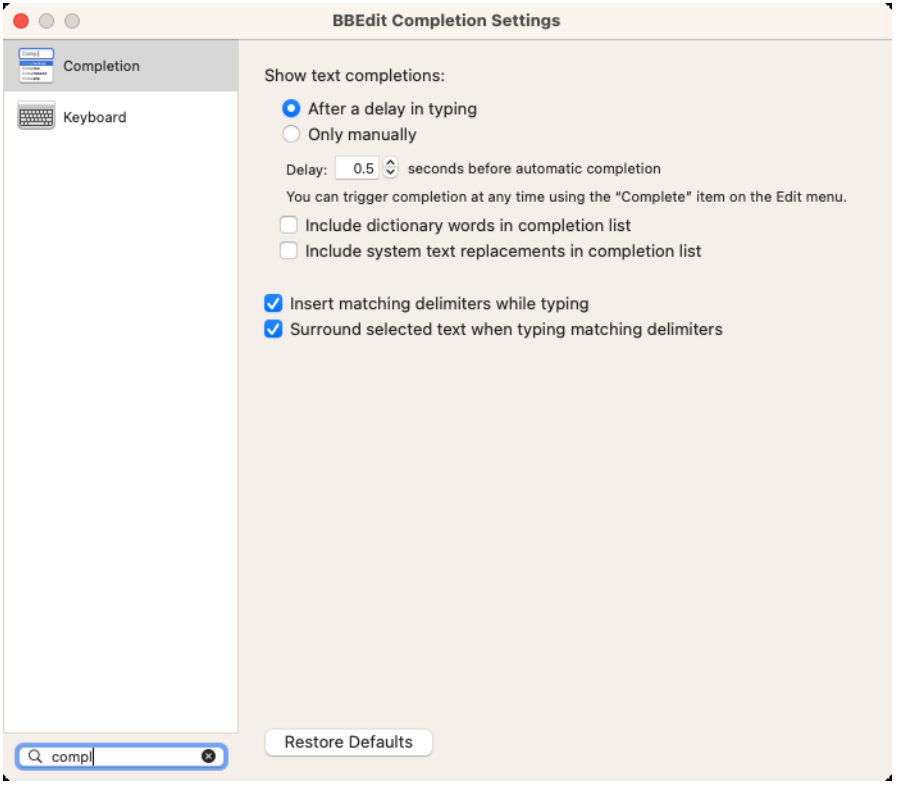
IMPORTANT BBEdit employs the standard system settings mechanism to store your settings settings. Accordingly, you can modify settings options directly by issuing "defaults write" commands. However, if you choose to modify your settings by means of "defaults write" commands other than those documented in this manual or the "BBEdit Expert Settings" page of our website without explicit advice from Bare Bones Software technical support, you take responsibility for any adverse effects.

If you discard your BBEdit settings file, you will have to re-activate the application with your product serial number, in addition to re-selecting any customized settings options you may have chosen

IMPORTANT In keeping with changes to macOS, the Settings command is now named "Settings" and the title of the Settings window will likewise change to "Settings".

Searching the Settings

You can perform keyword searches to quickly locate settings options in the Settings window. To do this, just click in the search field below the list of settings panels, and type a word or partial word into the field. As you type, BBEdit will search for instances of the current term and display all the panels which contain it. You can then select any of the listed panels to view and change the options within it. For example, here is the Settings window with an active, partial search for the term “completion”:.



Restore Defaults

Each of BBEdit’s settings panels (except the Language panel) contains this button, which you can click to reset all settings options within the current panel to their factory default settings.

Appearance Settings

The Appearance settings pane let you choose what appearance mode BBEdit should use.

Application appearance

This option offers you three choices to control BBEdit's appearance:

- Use system appearance: follow the appearance set in the "General" pane of the system settings. If the "Automatic" option is enabled then BBEdit's appearance will change as the system's does.
- Light: always use the Light appearance, even if the system's appearance is set to Dark (whether manually or automatically).
- Dark: always use the Dark appearance, even if the system's appearance is set to Light (whether manually or automatically).

Editor color scheme

This option allows you to easily select any existing color scheme. The scheme chosen here will be mirrored in the Text Colors settings pane, but is more convenient for "one-stop shopping" when you are experimenting with the application's overall appearance.

You may also click the arrow control below this popup to be taken directly to the Text Colors settings pane.

Status bar item size

This option controls whether the text and icons displayed in the navigation bar, the status bar, and in the sidebars of editing, project, and differences windows are of normal, large, or extra-large size.

This option is set to "Normal" by default, while the "Large" setting will cause these elements to occupy more space and use larger text, in order to enhance their readability, and the "Extra Large" setting does the same except even larger.

List display font size

This option controls the size of the system font used to display text in browser list panes, including disk browsers, search results browsers, FTP/SFTP browsers, etc. In addition, the various list-based palettes (Scripts, Text Filters, Clippings, Jump Points, Markers, and Functions) will use this option to determine the font size of their list content.

To decrease the font size, click on the slider control and move it to the left, or to increase the font size, click on the slider control and drag it to the right. (The default size is 11 point.)

Application icon

This option allows you to control which icon BBEdit presents in the Finder (if possible) and in the Dock (while the application is running). You may choose between “Default” (the factory default icon), “Classic” (the icon used in BBEdit 11-13), “Legacy” (the icon used in versions of BBEdit shipped in the 20th century), and “TextWrangler” (the TextWrangler app icon).

Navigation Bar Items

When any of the listed options are on, BBEdit displays the navigation bar (see page 94). You can also show or hide the navigation bar independently for each text window. This option is on by default.

Text options

When this option is on, BBEdit displays the Text Options popover in the navigation bar (see page 94).

Document navigation

When this option is on, BBEdit displays the Document popup menu in the navigation bar (see page 95).

Marker menu

When this option is on, BBEdit displays the Marker popup menu in the navigation bar (see page 96).

Related Files button

When this option is on, BBEdit displays the Related Files button in the navigation bar (see page 97).

Included files menu

When this option is on, BBEdit displays the Included Files popup menu in the navigation bar (see page 98).

Document status

When this option is on, BBEdit displays the document status icon in the navigation bar (see page 98). This icon serves as a proxy for the document file; you can Command-click on it to reveal the current file in the Finder, or drag it anywhere the original file can be dragged. In addition, you can click on this icon to display the File Info popover.

Function menu

When this option is on, BBEdit displays the Function popup menu in the navigation bar (see page 95). The related options below control how items appear in the menu.

Sort items by name

If this option is on, BBEdit sorts the items in the Function popup menu by name. Otherwise, items appear in the same order in the menu as they appear in the file. This option is off by default.

Show function prototypes
When this option is on, BBEdit displays the names of function prototypes as well as function definitions in the Function popup menu. Otherwise, the menu does not include entries for function prototypes. This option is on by default.

Editing Window
These options control additional elements which BBEdit can display in editing windows.

Line numbers
If this option is on, BBEdit displays line numbers along the left edge of the window.

Gutter
When this option is on, BBEdit displays the gutter (see page 102). You can show or hide the gutter independently for each text window. This option is on by default.

Tab stops
If this option is on, BBEdit displays tab stops as vertical grid lines within the content area of text windows, using the tab width set in the Editor Defaults panel.

Guide Contrast
You can use this sliding control to adjust the contrast level of the page guide display region. (See "Tab stops" on page 252.)

Page Guide at N characters
When this option is on, BBEdit displays the page guide at the specified character width. The page guide is a visible boundary indicator, whose color and contrast you can adjust (see page 275). This option is on by default.

Text Status Bar
When any of the listed options are on, BBEdit displays the status bar (see page 101). You can show or hide the status bar independently for each text window.

Cursor position
When this option is on, BBEdit displays the current location (line and column) of the insertion point, or the endpoint of the current selection range in the status bar (see page 104).

Language
When this option is on, BBEdit displays the Language popup menu in the status bar (see page 104).

Text encoding
When this option is on, BBEdit displays the Text Encoding popup menu in the status bar (see page 104).

Line break type
When this option is on, BBEdit displays the Line Break Type popup menu in the status bar (see page 104).

Document lock state

When this option is on, BBEdit displays a padlock icon in the status bar to represent the document's current lock state. You can click on this icon to lock (or unlock) the current document (if possible).

Document save date

When this option is on, BBEdit displays the current document's last saved date and time (if applicable) in the status bar.

Document statistics

When this option is on, BBEdit displays an item in the status bar which shows the number of characters, words, and lines in the document (and, if there's a selection, the number of characters, words, and lines in the selection range).

Instances of selected text

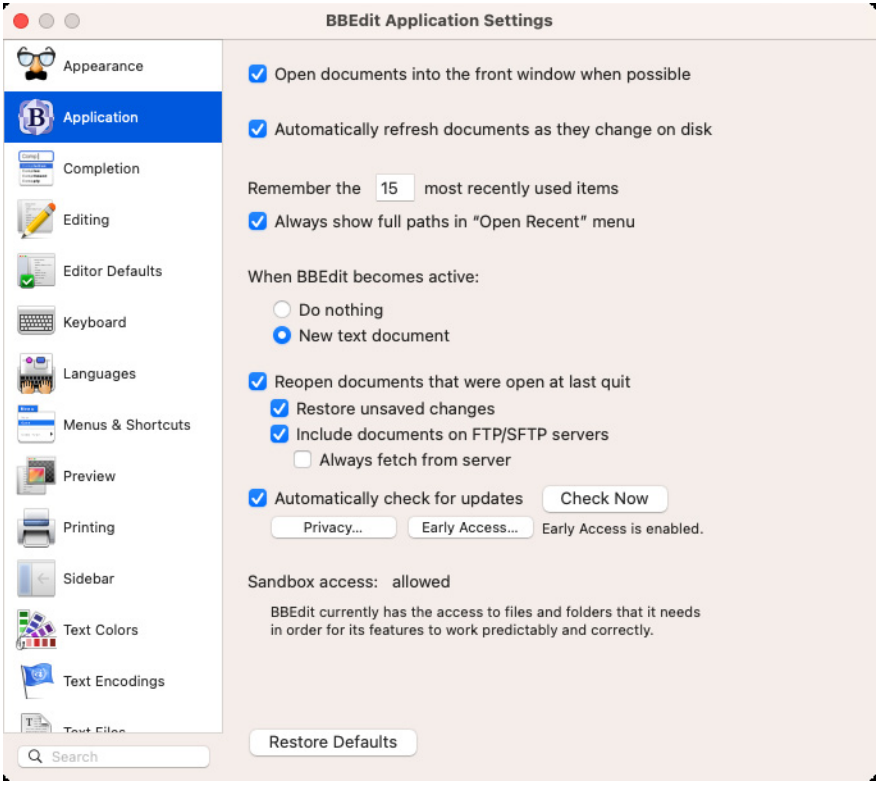
When this option is on, BBEdit displays either the number of matches for the selected text within the document, or the word surrounding the insertion point if applicable. (If the search takes long enough, BBEdit will display a progress indicator instead.)

Text magnification

When this option is on, BBEdit displays the Magnification popup in the status bar (see page 105).

Application Settings

The Application settings control how BBEdit checks for updates, when open files are verified, what action BBEdit performs at startup, and various other global settings.



Open documents into the front window...

This option controls whether BBEdit should attempt to open newly created or opened documents into the frontmost window (if possible), or whether each document should open directly into a separate text window.

This option is active by default, and while it is, BBEdit will handle documents in the following manner.

When you open an existing document, BBEdit will look for a project document which contains the document's file. If there is one, BBEdit will open the document that project's window (and bring the project window to the front). Otherwise, BBEdit will open the document into the frontmost editing window (and bring that window to the front if it is not already there.)

When you create a new document (via the File menu), BBEdit will never open that document in a project document's window: it will either use the frontmost editing window (if one is available), or make a new editing window if necessary. If you wish to explicitly create a new document explicitly within a project, you may use the “New Document...” within the project’s action menu (or contextual menu) to do so.

Automatically refresh documents as they change on disk

This option controls whether BBEdit checks if documents (files) have changed on disk while they’re open. If an open document has changed on disk, and there are no unsaved changes, BBEdit will automatically reload the document. If a document has changed on disk and also has unsaved changes, BBEdit will ask whether you want to reload the document from disk or keep the unsaved changes. This option is on by default.

The effects of the Revert command (from the File menu), and of a file Reload (which occurs when a document is reloaded by a refresh action) are both undoable.

Remember the N most recently used items

This text field lets you choose how many files appear on the Open Recent sub-menu of the File menu, and how many folders appear on the folder search popup menu in the Find Differences folder lists.

Always Show Full Paths in “Open Recent” Menu

Check this option to have BBEdit always display full paths in the Open Recent menu. If this option is off, BBEdit will only display path info when it’s needed to distinguish between files with the same name.

When BBEdit becomes active

This settings controls what BBEdit does when you launch it, or activate it when there are no open windows (e.g. by clicking its Dock icon while the application is already running). To override any of these actions when launching BBEdit, hold down the following modifiers.

Modifier(s)	Function
Option	Suppress startup items only.
Shift	Suppress all external services and startup items, and only reopen documents which contain unsaved changes.
Command-Control-Shift	Disable all external services and startup items, and optionally discard auto-recover information (which will result in the loss of any unsaved changes).

Do Nothing

Choose this option to prevent BBEdit from opening a new text editing window.

New text document
Choose this option to have BBEdit open a new, empty text editing window.

Reopen documents that were open at last quit
When this option is on, BBEdit will remember what documents (as well as disk browsers and FTP/SFTP browsers) were open when you choose the “Quit” command, and will attempt to reopen those documents the next time you launch it. This option is on by default.

Restore unsaved changes
When this option is on, BBEdit will preserve the contents of any unsaved document contents when you quit (including untitled documents) and restore those documents the next time you launch it. If you prefer the traditional Quit behavior, turn this option off.

Include documents on FTP/SFTP servers
When this option is on, BBEdit will attempt to reopen documents from remote servers when you launch it.

By default, BBEdit will save enough information about every document opened via its own FTP/SFTP support to reopen those documents at next startup **without** needing to immediately connect to the remote server(s).

Restore unsaved changes
If you prefer to have BBEdit always reconnect and reload all files previously opened from remote server(s), turn this option on.

Automatically check for updates
This option controls whether BBEdit automatically looks to see if a newer version is available. Regardless of the setting of the checkbox, you can manually check for an update at any time by clicking the Check Now button.

The version checking mechanism used by BBEdit protects your privacy. It works by requesting information about the currently available version from Bare Bones Software’s web server. The server will log the date, time and originating address of the request, and which versions of the OS and BBEdit you are using. This information is used to guide the future development of BBEdit; it is not personalized and will not be disclosed. Click the Privacy button to view our posted privacy policy.

Sandbox access
This option reports whether BBEdit has access to the files and folders necessary in order for it to work normally. If sandbox access is “allowed”, then BBEdit will function normally.

If sandbox access is “not allowed” then many of BBEdit’s features will behave incorrectly in ways we cannot predict or support; however, you may grant BBEdit access by clicking the “Allow...” button and selecting your Mac’s internal (boot) drive.

You may also click the “More Info...” button to display more information about sandboxing and its access requirements.

AI Chat Worksheets Settings

All queries and responses sent between BBEdit worksheets and the chosen chat service API are transmitted securely, and directly between BBEdit and the API endpoint. Bare Bones Software does not receive or retain any data transmitted thereby. Your conversations with the API service are governed by the API service provider's own privacy and content policies.

Configuring AI Chat Services

The AI Chat Worksheets settings provides a popup to let you choose what service and default model BBEdit should use, as well as an option you can set to control the character width at which lines within responses should be wrapped.

BBEdit will prompt you for your API key the first time you send a query from a worksheet or you may enter its in advance here. (You can also use the provided button to clear the API key should that become necessary.)

The popup menu for model selection in the AI Worksheet settings now contains an "Other..." action which makes an API request to the service (if supported) requesting a list of the available models, and then provides a searchable list.

Completion Settings

The Completion settings control BBEdit's text completion behaviors, including both the completion popup and automatic insertion of delimiters, which consist of parentheses, brackets, single and double quotes, as well as various language-specific elements.

Show text completions

This option lets you choose when BBEdit will display text completions: automatically after a short delay while typing, or manually upon typing a trigger key (F5 by default) or choosing the Complete command in the Edit menu.

Note This feature is also known as "autocomplete" or "autocompletion".

Include dictionary words in completion list

When this option is on, BBEdit will include dictionary words in the text completion list. This option is off by default.

Include system text replacements in completion list

When this option is on, BBEdit will include any system-wide Text Replacement triggers (as configured in the "Keyboard" system settings) which begin with the text you typed in the completion popup.

Note This feature is also known as "autocorrect" or "autocorrection".

Insert matching delimiters while typing

When this option is on, typing any opening delimiter will cause BBEdit to immediately insert the appropriate closing delimiter. This option is on by default.

Surround selected text

When this option is on, typing an opening delimiter will cause BBEdit to surround the selected text with a matched pair of delimiters. This option is on by default.

Editing Settings

The Editing settings control various general editing behaviors.

Display instances of selected text

When this option is on and you make a selection (that doesn't consist entirely of whitespace or punctuation), BBEdit will display all occurrences of the selection within the current document via either underlining or highlighting, and you can use the Search menu commands Next Occurrence of <string> and Previous Occurrence of <string> to navigate the occurrences. This option is on by default.

You can control whether BBEdit underlines or highlights occurrences via the corresponding radio buttons, or adjust the duration of the delay before BBEdit applies highlighting via the “Delay” control.

Note Any navigation you undertake via this feature is independent from the Find and Live Search commands, except that BBEdit will add the selected string to its search history for convenient future reuse.

Show tick marks in scroll bars

You can use these options to control whether BBEdit shows tick marks for instances of selected text, spelling errors, Live Search results, and/or language server diagnostics (“Issues”) in the active document’s scroll bar.

NOTE Diagnostics may be generated by a language server if one is active for the document's language.

Show issues

When this option is on, BBEdit will display LSP-generated diagnostics when available.

Highlight insertion point

When this option is on, BBEdit will highlight the line that currently contains the insertion point. (This behavior is now independent of the selected color scheme.)

Use “hard” lines in soft-wrapped views

When this option is on, the line number bar, position display, and Line Number commands in editing views will use line and character position numbers that correspond to the “hard” line breaks actually present in the document, rather than the soft-wrapped line breaks.

Additionally, when this option is on, line selection commands and gestures, including the Select Line command, triple-clicking, and click selection in the left margin, will treat only "hard" line breaks as line boundaries.

Insertion point

You can use this option to choose your desired insertion point cursor style. The available options are "Vertical bar" (the default), "Block", and "Underline".

Note If you use the Underline cursor, we recommend that you change the style for "Display instances of selected text" to "Highlight".)

Line spacing

This control allows you to adjust the amount of space between lines of text in editing views. The default value is consistent with previous versions of BBEdit.

Extra vertical space ("overscroll") in text views

This option allows you to specify how much empty space BBEdit should leave after the end of each document's content: none, a half-window, or a full window. (This behavior is also sometimes called "overscroll".)

Allow pinch-to-zoom to change magnification

When this option is on, you can use touchpad pinch gestures to increase or decrease the magnification level of the frontmost (active) document. This option is on by default.

Enable single-click line selection

When this option is on, you can select lines by single-clicking on them in the line number bar or gutter. This option is on by default.

Cut/Copy entire line for insertion point

When this option is on, the Cut and Copy commands will act upon the entire line which contains the insertion point, in order to make life easier for Windows refugees.

This option is off by default, since standard macOS behavior is to only enable the Cut and Copy commands when there is actually a selection.

When opening a document, collapse folds

When this option is on, BBEdit will automatically collapse all folds below the specified level. This option is off by default.

Invisibles display: tabs and line breaks

This option allows you to specify alternative characters for tabs and for line breaks when the Show Invisibles option is enabled. You can click the arrows next to the respective edit fields to open the standard macOS character panel. There are some restrictions: you can't use Roman letters and numbers, for example, nor can you use whitespace or literal line breaks. You may use punctuation, since that is generally more readable. If you try a character and find it doesn't suit your taste, please try something else.

The factory defaults are “∆” (U+2206, INCREMENT) or Option-J on US keyboards) for tabs, and “¬” (U+00AC, NOT SIGN) or option-L on US keyboards) for line breaks.

Editor Defaults Settings

The Editor Defaults settings control the behavior of newly created document windows and documents without saved state information. Many of the options in this panel parallel options provided in the Text Options popover in the navigation bar (or the Text Options sheet). The difference is that the options in the Text Options sheet and the Text Options popover control only the behavior of the *active* (frontmost) document, while the Editor Defaults settings control the behavior of all *new* documents.

Auto-indent

When this option is selected, pressing the Return key in new windows automatically inserts spaces or tabs to indent the new line to the same level as the previous line.

Further, while this option is enabled, the “New Line Before Paragraph” and “New Line After Paragraph” commands will indent the inserted line by the same amount as the line on which the command was invoked.

Tip To temporarily invert the sense of the Auto Indent option while typing, hold down the Option key as you press the Return key.

Balance while typing

When this option is selected, BBEdit flashes the matching open parenthesis, brace, bracket, or curly quote when you type a closing one. This option is useful when you are editing source files, to ensure that all delimiters are balanced.

Use typographer’s quotes

When this option is on, BBEdit will automatically substitutes curly (or typographer’s) quotes (“ ” ‘ ’) for straight quotes (" ') in any new documents you create.

Tip To type a straight quote when this option is selected (or to type a curly quote when the option is deselected), hold down the Control key as you type a single or double quote.

Note You should avoid using typographer’s quotes when creating or editing any plain-text documents such as email message content or source code.

Auto-expand tabs

When this option is on, BBEdit inserts an appropriate number of spaces instead of a tab character every time you press the Tab key.

Show invisible characters

This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and gremlins (invisible characters). BBEdit uses these symbols to represent non-printing characters:

Symbol	Meaning
Δ	tab
•	space
◦	non-breaking space
¬	line break
¶	page break
|	zero-width character
¿	other non-printing characters

Show Spaces

When this option is on (and Show Invisibles is also active), BBEdit will display placeholder characters for spaces. Turn this option off to suppress the display of spaces (reducing visual clutter when you are displaying invisible characters).

You can also customize the glyphs used as placeholders for tabs or line breaks within the Editing settings pane.

Note Non-breaking spaces (typed by pressing Option-space) will not be displayed with a placeholder.

Check spelling as you type

When this option is on, BBEdit will automatically check spelling as you type, and underline any potentially misspelled words. Turn this option off to prevent BBEdit from automatically checking spelling.

You can turn on automatic spell checking for the active document only by choosing Check Spelling as You Type from the Text menu. (See “Check Spelling As You Type” on page 137.)

Default font

This option controls the standard font and font size which BBEdit uses to display the contents of text windows. To change this option, click Select to bring up the standard Font panel, and choose the desired font and size. If the “Inconsolata” font is available on your Mac then BBEdit will use it as the default font; otherwise, it will default to an appropriate choice. (Menlo, the OS’s default monospaced font is the fallback choice.)

Further, the maximum allowable font size is 144pt, to eliminate any potential issues caused by accidental miskeying in the Font panel’s Size field (which on its own will allow one to specify an arbitrarily large size).

Note You can also adjust the default tab width on a per-language basis. To do so, select a language entry in the Languages settings panel, click “Options” to bring up the language options sheet, and enter the desired tab width in the Editing section of this sheet.

Override document setting

When this option is enabled, it will cause the default font to always prevail over any previously saved font settings associated with a given text file. That way, if you save changes to a file, close it, and then change the font setting, the new font setting will be in effect the next time you open that file, rather than the old one.

Spaces per tab

This option controls the default number of spaces that BBEdit uses to represent the width of a tab character.

Magnification

This option sets the default magnification for new text documents. Changes will take effect the next time a document is opened (or created), and will not affect existing documents, i.e. documents which are already open.

Soft wrap text To

When this option is selected, BBEdit soft-wraps the text in the file to the right margin that you choose: the Page Guide, the window width, or a specified number of characters, as selected by the options below the checkbox.

Soft-wrapped line indentation

This option lets you specify how BBEdit should indent soft wrapped text: flush with the left edge of the window (“Flush left”), at the same indent level as the first line of the paragraph (“First line”), or indented one level deeper than the first line of the paragraph (“Reverse”).

Note You can now change a document's soft-wrapped line indentation using the Text Options sheet panel or popover (gear) menu in the navigation bar; you can also adjust this behavior on a per-language basis by creating a custom language setting in the Languages settings pane.

Keyboard Settings

The Keyboard settings control BBEdit's response to the use of various special keys, including the ability to recognize Emacs key bindings

Use Tab key to navigate Placeholders

When this option is on, BBEdit will jump to the next placeholder in the document (if any) when you press the Tab key, or the previous placeholder if you press Shift-Tab. This behavior is equivalent to the Go to Previous/Next Placeholder commands in the Search menu (see page 200). For additional details, see "Selection and Insertion Placeholders" on page 339. This option is on by default.

"Home" and "End" Key Behavior

There are three potential choices for this option:

Scroll to Beginning and End of Document

Choose this setting to have the Home and End keys perform these respective actions. This is the default setting, which reflects the standard key motion behavior in Macintosh applications.

Move Cursor to Beginning and End of Current Line

Choose this setting to have the Home and End keys perform these respective actions instead. This option may be useful for those accustomed to Windows editing key behavior.

Progressive (BRIEF Compatible)

Choose this option to have the Home and End keys behave as follows on successive presses:

- the first press will move the insertion point to the beginning (or end) of the current line;
- the second press will move the insertion point to the beginning of the first line (or the end of the last line) in the current page of text, without scrolling;
- the third press will move the insertion point to the beginning (or end) of the document.

The behavior is progressive within a specific time period. After the period expires, or if you change the selection range by other means, the behavior state resets, so the next press of Home or End will behave as in the first step described above.

The factory default timeout period is ten seconds. (There is an expert settings option to control this period; please see the "Expert Settings" page of BBEdit's online Help for details.)

Enter key generates Return

When this option is on, BBEdit will generate a line break when you press the Enter key.

When this option is off, pressing the Enter key will bring the current insertion point (or selection range) into view.

Note Pressing the Enter key in a Unix worksheet will always execute the current line (or the selected lines).

Allow Tab key to indent text blocks

When this option is on, you can press the Tab key to invoke the Shift Right command, or Shift-Tab to invoke the Shift Left command; this may be useful for those accustomed to Windows editing key behavior. When this option is off, pressing Tab will insert a tab character in the normal manner. This option is off by default.

Enable Shift-Delete for forward delete

When this option is on, holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

Enable macOS "Help" key

When this option is on, pressing the Insert key present on some PC-style keyboards will open BBEdit's Help book. (This frequently happens by accident.) This option is off by default.

When auto-indenting, remove leading white space from indented line

When this option is on, if the current line is not indented but there is whitespace followed by text to the right of the insertion point, when you type Return, BBEdit will insert a line break and strip the following whitespace, leaving the remaining text also left-aligned.

Allow Page Up and Page Down keys to move the insertion point

When this option is off, scrolling the view by typing Page Up or Page Down does not affect the position of the insertion point. (This is the standard behavior for Mac applications.) When this option is on, BBEdit will move the insertion point to the same relative position within the window in each new screenful of text displayed by Page Up or Page Down. This option may be useful for those accustomed to Windows editing key behavior.

Option-Up arrow and -Down arrow move by paragraphs

When this option is on, the Option-Up arrow and Option-Down arrow key shortcuts will advance to the previous or next paragraph, instead of the previous or next screen. (This option is off by default.)

Note Since a paragraph is defined as the region between two hard line breaks (including the beginning or end of the document), please note that if soft wrapping is **not** enabled, setting this option basically causes Option-Up Arrow and Option-Down arrow to move to the beginning or end of the current line (or the next non-empty line, if you're already at the end of the current line).

Emulate Emacs key bindings

If turned on, this option allows you to use the basic Emacs navigation keystrokes to move around in editing views. It is not a full Emacs emulation mode; rather, it is more of a comfort blanket for individuals with Emacs key bindings hard-wired into their muscle memory. See Appendix B, “Editing Shortcuts,” for a list of the Emacs commands BBEdit supports.

Display status window

When both this option and “Emulate Emacs key bindings” are on, BBEdit will display a small palette which shows Emacs shortcuts as you type them.

Enable meta sequences

This option is off by default so the Escape key can be used to exit full screen mode when running on OS X 10.10 or later.

If you turn this option on, BBEdit will intercept presses of the Escape key and use these to emulate a meta key to generate Emacs command sequences, at the cost of no longer being able to use the Escape key to exit full screen mode.

Allow the Escape key to trigger text completion

This option is off by default. When this option is on, pressing the Escape key will trigger text completion (also known as “autocomplete”) in the same manner as pressing the default key shortcut of F5.

Note This option works at cross purposes to both Emacs emulation and using the Escape key to exit full screen mode; that is, if you turn on Escape key completion triggering, the Escape key will serve only to trigger a completion, and will not function as either the Emacs meta key nor will it exit full-screen mode.

Languages Settings

The Languages settings allow you to configure how BBEdit maps file names to language types (e.g. “.html” to HTML), and allows you to apply customized behavior and display parameters to any installed language.

Installed Languages

Click the “Installed Languages” button within this panel to see a complete list of installed languages, together with the language module version number (if applicable) and filename extension(s) associated with each language. (This list includes both languages intrinsically supported by BBEdit, and those added via installed language modules.)

You can now control whether any given language should appear in the Languages menu (as used in the status bar and the “Text Options” sheet) by using the check boxes in the “Menu” column.

Default Language

These two options allow you to choose the default language for new and existing documents. The “New documents” option applies to created but unsaved documents, while the “Unmapped files” option affects files that already exist, but which do not have any pre-existing metadata (such as a mode line, EditorConfig file, or filename extension) to identify their language type.

Extensions Mappings

BBEdit includes a set of default file extension mappings which cover the most common usages for each supported language, while each language module ordinarily contains extension mappings for the language it supports.

You may add (or remove) additional extension mapping via the Extensions list. To add a mapping, click the Extensions tab, then click the”Add” (+) button below the list, click in the Extension column and type the desired filename extension, then select the associated language via the adjacent popup. (You can also edit existing mappings in the same manner.)

Note You can use wildcards in the suffix to indicate single characters (?), any number of characters (*), or a single digit (#). For example, “page.#html” could map to a different language from “.html”.

For convenience, you can also add custom filename extension mappings by dragging files into this list. If BBEdit can guess a file's language based on its contents, it will; otherwise you can change the language mapping using the popup menu in the list. If a filename extension is already known to BBEdit, the dropped file will be ignored, as will duplicate filename extensions when dropping multiple files.

Custom Settings

By default, BBEdit will apply your active settings settings within each language.

If you wish to modify how BBEdit treats documents having a particular language, e.g. to have BBEdit use a specific tab width or a custom color scheme, you may add a custom language settings.

To create such a settings,select the “Custom Settings” tab, then click the plus (+) button below the list of Language Customizations, and select the desired language from the resulting popup. When you do so, BBEdit will display a language options sheet which contains the following sections:

- General: In this section, you can view or change the comment-start and comment-end strings used by the Un/Comment command on the Text menu for the selected language, or to view or change the Reference URL Template used by the Find in Documentation command.
- Editor: In this section, you can view or change the default display and editing options used for documents in the selected language. (These options parallel the options provided by the Text Options command.)
- Display: In this section, you can view or change the default items which appear on the navigation bar and status bar for documents in the selected language. You can also choose any available color scheme to use for syntax coloring of documents in the selected language in each of Light and Dark appearance modes.

To remove an existing language settings, select the desired entry in the list of Language Customizations, and click the minus (-) button below the list. Once you have removed the entry, BBEdit will again apply its active global settings settings to all documents with that language.

Note A language settings can specify an alternate display font and font size, so, for example, you could use one font for Markdown, a second font for Objective-C, a third font for HTML, and so on.

Further, a language settings can now control completion behavior and you can employ this option as desired, either to turn completion off by default and then re-enable it for specific languages, or (more likely) turn off automatic completion for Markdown and other content-oriented (versus code-oriented) languages.

JavaScript-specific Settings

If you create language-specific settings for JavaScript, you can now control whether anonymous items should appear in the function menu via the corresponding option in the JavaScript tab.

Python-specific Settings

If you create language-specific settings for Python, you can now explicitly control a number of related behaviors via options in the Python tab, including:

- which Python binary to use (‘python’, ‘python3’, or any separate item of your choosing);
- whether to allow the document's shebang line “#!” to override the choice of interpreter; and
- whether to use the ‘flake8’ tool for syntax checking (if available).

When using a custom interpreter, you must specify it in one of two ways:

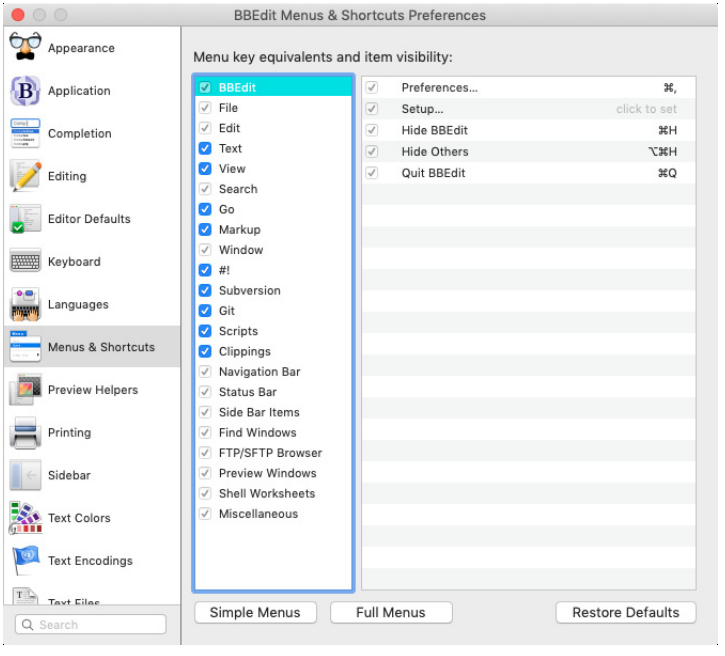
- via a simple command (e.g. ‘python’) which will look for the interpreter in your PATH (the explicit ‘python’ and ‘python3’ options will do this), or
- via a fully qualified path to the executable (e.g. ‘/usr/local/bin/python3/).

Note You cannot use a partial path, since GUI applications have no useful notion of a working directory and so there is no basis for resolving a relative path.

Whether you enter a full path or just a command name, BBEdit will validate that the item you specified exists before enabling the “OK” button.

Menus & Shortcuts Settings

The Menus & Shortcuts settings allow you to show or hide whole menus or individual commands. You can also assign key equivalents to commands and various window elements, as well as to clippings and scripts.



Menu Key Equivalents and Item Visibility

This section of the settings panel displays a hierarchical list of each menu and menu command available within BBEdit.

You can hide any menu or command which is not necessary for BBEdit to function, by turning off the checkbox next to that item's name. (The checkbox is disabled for necessary items, such as the File menu and the Quit command.)

You can assign or change the keyboard shortcut (key equivalent) for any menu command, as well as items on the Text Options popover, and the Markers and Line Breaks popup menus, by double-clicking on the right-hand portion of that command's list item (at the "click to set" label) and typing the desired key equivalent.

To clear the key equivalent from a menu command, double-click on the right-hand portion of that command's list item and press the Delete key.

Click Restore Defaults to restore all key equivalents to their factory default values (as listed in Appendix A).

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Note The OS may preempt certain key combinations, such as Command-Tab.

Simple Menus/Full Menus

You can click the Simple Menus button to simplify BBEdit's menu structure by hiding its "advanced" commands (i.e. the commands which are available with a full BBEdit license, as well as during the initial evaluation period). The result is a menu structure that is very similar to TextWrangler's.

Clicking the Full Menus button will "undo" the changes made by the Simple Menus button by making all menus and commands visible (even ones that might have previously been manually hidden).

Each of these buttons will leave your custom keyboard equivalents intact.

Restore Defaults

Clicking the Restore Defaults button will reset all keyboard equivalents to their factory defaults, and will make all menus and items visible.

Preview Settings

The Preview Helpers settings panel lists all web browsers on your machine which are available to preview HTML documents.

Web browsers available for previewing

This list displays all the available (installed) web browsers known to BBEdit. Browsers are listed by name and version number, in the same form as they appear in the Preview In submenu of the Markup menu.

The browser list includes each individual browser application that is available on your Mac. For example, if you have Firefox 101.0 and Firefox 78.0 on your hard disk, both applications will be listed and available for previewing.

You can use the plus (+) and minus (-) buttons to respectively to add a browser to the list or to remove an entry.

NOTE If you click the "Restore Defaults" button, BBEdit will ask the system for a list of all installed applications which claim to support HTTP/HTTPS URLs, and will add these applications to the list if they are not already present. If using this button does not add a browser which you know is available, you can add it directly with the Add button. (Sometimes, the system may not properly inform BBEdit of every browser application present.)

Previewing in Windows browsers through VMWare

BBEdit recognizes the application stubs created by VMWare Fusion so that you can preview documents in Internet Explorer or other Windows-based browsers while running VMWare Fusion. Any documents you wish to preview must be readable by the running Windows virtual machine via either a shared folder or VMWare Fusion's "Mirrored Folder" option.

If the browser list does not include these VMWare-hosted applications, you can add them manually as described above.

Markdown Processor

This popup allows you to select which built-in, or installed and recognized, Markdown processor BBEdit should employ to convert Markdown content to HTML for previewing and export.

BBEdit has built-in support for 'cmark' as well as 'classic' Markdown (Gruber's original 'Markdown.pl' script). In addition, if you have Discount, MultiMarkdown, Pandoc, or Cmark-gfm installed, BBEdit can use these as well. (If a particular tool is not installed, its corresponding menu entry will be disabled.)

Finally, the "Custom" option provides the means to specify your own rendering command and arguments, according to the following rules:

- The command name must be available in your $PATH. (You can specify a fully qualified path, but this is discouraged.)
- The command arguments must be separated by spaces. (Because they are separated by spaces, the arguments must not contain spaces.)
- The tool you're using will be run as a BBEdit text filter, so it must read the Markdown to be processed from standard input, and output the HTML to standard output.

Printing Settings

The Printing settings control BBEdit's default document printing behavior.

Printing font

This option specifies the default font BBEdit uses for printing when the "Print using document's font" option is turned off.

Use document's font

When this option is on, BBEdit uses the document's display font and tab settings when printing.

Use custom font

When this option is on, BBEdit uses the specified font and tab settings when printing. To change these settings, click Select to bring up the standard Font panel, where you can choose a font and font size. (The currently chosen options appear in the display box.)

Frame printing area
When this option is on, BBEdit draws a box along the edges of the printed text.

Print page headers
When this option is on, BBEdit prints the page number, the name of the file, the time and date printed in a header at the top of each page.

Print full path
When this option is on, BBEdit prints the full path of the file being printed in the header.

Print line numbers
When this option is on, BBEdit prints line numbers along the left edge of the paper.

1-inch Gutter
When this option is on, BBEdit leaves a one-inch margin along the left edge of the paper. Use this option if you usually store printed pages in three-ring binders.

Print color syntax
If this checkbox is on, BBEdit prints all colorized text within the document in color. You should generally use this option only on color printers, as colorized text may come out in difficult-to-read dithered shades of gray on black-and-white printers.

Time stamp
This option let you choose whether the date that appears in the printed page header is the date that the file was last modified or the date that the file was printed.

Wrap printed text to page
This option lets you choose whether a document's contents should always be soft-wrapped upon printing, or only if soft wrapping is enabled for that document.

Sidebar Settings
The Sidebar settings let you control when BBEdit should display the sidebar on editing windows, and adjust other aspects of its behavior (see page 107).

Automatically show sidebar
This option controls whether BBEdit should display the sidebar only as needed (i.e. when there are two or more documents open in the front window, always, or never.

Further, if this option is set to "as needed", BBEdit will store the current sidebar visibility of the window as part of the default window layout. Otherwise, if the sidebar settings is set to "Always" or "Never", then BBEdit will set the default sidebar state to be visible ("Always") or hidden ("Never"), respectively.

Note

Reshape window when showing or hiding sidebar

When this option is on, BBEdit will reshape the entire window when the sidebar appears (or is hidden) to maintain the editing pane at a consistent size. (This option has no effect over windows in full screen mode.)

Add documents to sidebar in alphabetical order

When this option is on, BBEdit will add newly created and/or opened documents to the sidebar list in alphabetical order, if possible. If this option is disabled, BBEdit will always add documents at the end of the list.

Show icons in sidebar lists

When this option is on, BBEdit displays icons for files and folders in the sidebar's file list. This option controls the display of icons in editing windows, project documents, and disk browsers.

Show Finder tag in sidebar lists

When this option is on, BBEdit displays the Finder tags for files and folders in the sidebar's file list. This option controls the display of tags in editing windows, project documents, and disk browsers.

Allow keyboard focus

When this option is on (as it is by default, or if you previously set the expert settings), the sidebar lists in projects, multi-document editing windows, and differences windows can acquire keyboard focus (by clicking or tabbing into them), and you can then use conventional keyboard navigation gestures to change the selection.

The selection highlighting will additionally reflect whether or not the sidebar has keyboard focus; the background always reflects the activation state of the window.

Whenever you click in the sidebar to select something, keyboard focus will go there (assuming the corresponding settings option in the "Sidebar" pane is turned on, as it is by default now). Any single click will select a new file, but **does not** move keyboard focus to the editing pane. You can double-click on an item, press Return or Enter, or use the Tab key to move keyboard focus as desired.

NOTE When this setting is turned on, Differences windows gain keyboard navigation in the sidebar, which was not previously possible.

Text Colors Settings

The Text Colors settings let you adjust the default colors that BBEdit applies to both general properties (including the foreground and background text colors, spelling errors, spaces and invisible characters, and highlight colors) and language-specific syntax elements

IMPORTANT You can select and edit color schemes within the Text Colors settings pane. A central concept is that there is **always** a color scheme in effect, which may be a factory color scheme, a custom scheme that you've downloaded, or a scheme that you have created yourself.

If you are upgrading from an older version of BBEdit which had custom settings in effect, BBEdit will write those settings out into a color scheme file in your “Color Schemes” application support folder, and then make that color scheme active. You won't lose any settings, and as a bonus your color scheme can be used with any modern version of BBEdit (11.0 or later).

Next, any changes you make to the active color scheme within the Text Colors settings pane will automatically change the color scheme file on disk.

BBEdit will however ask Time Machine to save a version snapshot before making any changes to the color scheme file.

You can open the scheme file and use the Compare Against Previous Versions command (in the Search menu) to evaluate the changes. BBEdit will also automatically make a backup copy of the scheme file into the “BBEdit Backups” folder (which is located within BBEdit’s sandbox container folder).

If the selected color scheme was one of the factory default schemes, BBEdit will first make a copy of that scheme into the “Color Schemes” support folder, and then apply your changes to that copy, and select the copy as the active scheme.

To create a new color scheme, bring up the Text Colors settings pane and select the scheme you'd like to start with (if it's not already the active scheme), then click the “New...” button. BBEdit will propose a name for the new scheme, which you may of course edit as desired. Click “OK”, and BBEdit will automatically create your new color scheme in the “Color Schemes” support folder, and then make it active. From there you can edit the scheme by changing settings within the Text Colors settings pane.

Finally, if you open a color scheme file directly, it will (once again) open as text. This makes hand editing and inspection simpler than was possible before. (Please note that though manual edits to the scheme file will not take effect immediately; you can switch away from and then back to the scheme in question to cause it to reload.)

Selecting and Saving Color Schemes

BBEdit offers several built-in color schemes, which you may use as-is or as the basis for a custom scheme. To create a new color scheme document using the current settings, click the “New...” button, and you can then apply, install, or save that scheme. To load a saved color scheme, choose it in the Color Scheme popup menu. (As in prior versions, any color scheme you create will be stored in the “Color Schemes” subfolder of BBEdit’s application support folder.)

You can further associate a saved color scheme with any language via the Custom Language Settings list in the Languages settings panel. (See “Languages Settings” on page 265.)

BBEdit will also import any BBColors files which you place in the “Color Schemes” folder, and automatically convert them.

How to Change an Element’s Color

The color bars show the colors that BBEdit uses to display different interface and language elements. To change the color for any element, click the adjacent color box to open the system color picker which you can use to select a new color. To restore all colors and options to their default settings, click the Restore Defaults button.

Language-Specific Colors

The center section of the Text Colors panel contains groups of language-specific syntax coloring options, which you can adjust to specify the colors BBEdit uses to display the corresponding language elements.

Though the available set of languages and elements is too extensive to list in total, here are some common elements:

- **Comments** include all text set off by a language’s designated comment marker(s).
- **Strings** (and **Numbers**) are defined by each individual language’s specification.
- **Language keywords** are those terms defined in a language’s specification
- **Predefined symbols** are terms which are not language keywords, but which are predefined by a language's reference implementation, or which are part of a language's standard library/framework support, or which have other special meaning to developers writing code in that language.
- **ctags symbols** are any words or elements identified in an associated ctags file.

while in general, most elements’ natures should be clear from their display names (e.g. **Preprocessor directives**).

Global Colors

The General group within the Text Colors panel contains options which control global colors used within any language such as the foreground (text) and background (window) colors and the color of the underline used by the spelling checker to mark questioned words.

Plain text

This option controls the foreground text color used within editing windows (and other content display views).

Document background

This option controls the background color used within editing windows (and other content display views).

Selected text (active) or (inactive)
These options control the color BBEdit uses to highlight words selected in an active or an inactive document, respectively.

Insertion point line highlight words
This option controls the color BBEdit uses to highlight the line containing the insertion point.

Difference highlight
This option controls the color BBEdit uses to highlight differing regions within a Differences window. (BBEdit automatically derives the color used to display differences within a line from this color by darkening or lightening it as needed).

If a custom color scheme which does not include a Differences color is active, BBEdit will instead use a dark gray or light gray, depending on the scheme's background color.

Sub-line difference hightailed
This option provides explicit control over the color BBEdit uses to display character ranges within a single top-level (or line-level) difference, rather than using a derived color.

If the chosen color scheme does not contain an explicit color for sub-line differences, BBEdit will derive it from the Differences color (as in versions past).

Spelling error highlight
This option controls the color BBEdit uses to highlight misspelled words.

Invisible spaces
This option controls the color BBEdit uses to display spaces when the Show Invisibles and Show Spaces display options are active.

Other invisibles
This option controls the color BBEdit uses to display invisible characters other than spaces when the Show Invisibles display options is active.

Use custom selection highlight colors
Turn this option on to have BBEdit use custom highlight colors. You can choose the primary and secondary highlight colors. (This option is off by default.)

Text Encodings Settings

The top of the Text Encodings settings panel contains an alphabetical list of every character set encoding available in the system, and allows you to choose which of these encodings BBEdit includes in its menus. These menu are:

- The Read As popup menu in the Open dialog
- The Encoding popup menu in the Options dialog within the Save dialog
- The Encoding popup in the status bar
- The character set popup menus in various dialogs (e.g. New HTML Document)

- The encoding selection popup menus in this settings panel

To include an encoding for display, select it and click Enable. To remove an encoding from display, select it and click Disable. To include all encodings or remove all but the required encodings, click the Enable All or Disable All buttons respectively.

(All available Unicode encodings are permanently enabled and cannot be turned off.)

Tip To keep the length of the encoding menus manageable, you should add only those encodings which you use frequently.)

Default text encoding for new documents

BBEdit uses the encoding specified by this option for new documents which do not contain an intrinsic encoding specification.

If file's encoding can't be guessed, try

If BBEdit cannot determine a file's proper encoding by examination, it will try opening the file using the encodings) contained in this list, in the order they appear.

Text Files Settings

The Text Files settings control how BBEdit opens and saves files, including whether to make backups.

Line breaks

This option controls what kind of line breaks BBEdit writes when creating a new file. You can choose:

- Unix line breaks (ASCII 10) for general use. This is the default option.
- Legacy Mac line breaks (ASCII 13) if you will be using the file with older (or Classic) Macintosh applications.
- Windows line breaks (ASCII 13/10) if the file will reside on a Windows server or if you are sending it to someone who uses a Windows system

Ensure file ends with line break

When this option is on, BBEdit will add a line break at the end of the file if there is not already one present.

You can also adjust this option on a per-language basis by adding custom language settings. (See "Languages Settings" on page 265).

Strip trailing whitespace

When this option is on, BBEdit will trim all trailing non-vertical whitespace from the document file before writing it out.

You can also adjust this option on a per-language basis by adding custom language settings. (See "Languages Settings" on page 265).

Backups

These options control whether BBEdit should make backup copies of edited files, and the manner in which it does so.

Make backup before saving

Turn this option on to have BBEdit automatically make a backup copy of each file that you save. BBEdit creates a single backup file for each file that you save in the same folder as that file. This option is global and backups can no longer be made on a per-file basis. However, you can exclude individual files from being backed up by adding an Emacs variable to them (see “Emacs Local Variables” on page 54).

When this option is on, and you close a document with unsaved changes and elect to discard those changes (“Don't Save”), BBEdit will automatically save a snapshot of the document's contents in the same directory as the document, and the snapshot file’s name will follow the Emacs convention “#foo.txt#” (or if the “Preserve file name extension” (see below) is on, the snapshot's name will be “#foo#.txt”).

Keep historical backups

When this option is on, BBEdit will preserve backups in the “BBEdit Backups” folder, whose default location is “~/Library/Containers/com.barebones.bbedit/ Support/BBEdit Backups/” and the “Preserve File Name” option (see below) will automatically be turned on and locked.

Within the backup folder will be one folder for each day's backup files. The format of the dated folder name is static and non-localized: YYYY-MM-DD. Inside of each day’s backup folder will be all of the backup files made on that day, each named using a timestamped format, and BBEdit will keep a running total of the amount of disk space occupied by all backed up files.

If need be, you can quickly access this folder by clicking the “Go” button adjacent to the Space used” report, or by choosing Document Backups in the Folders submenu of the BBEdit (application menu), or via the list in the Folders pane of the Setup window.

NOTE The previous location of the backup folder was “~/Documents/BBEdit Backups/” and if such a folder exists, BBEdit will continue to use it.

Preserve file name extension

By default, the backup files which BBEdit creates are named in accordance with current system conventions (which themselves follow the old Emacs convention): the backup file takes the name of the original with a tilde appended; for example, “foo.html~” is the backup of “foo.html”.

If you want backup files to have the same filename extension as the originals, turn on this option to have BBEdit place the tilde after the “base” name of the file; for example, “foo~.html”.

Controlling Backups with Emacs Variables

You may also use an Emacs variable to control whether or not a given file is backed up. There are two ways to do this:

Absolute: If the variable line/block contains a “make-backup-files” variable, that variable’s value will override the global “Make Backup Before Saving” settings.

```
-*- make-backup-files: 1 -*-     --> always back up this file
-*- make-backup-files: 0 -*-     --> never back up this file
```

If the first letter of the variable's value is "y", "t", or "1", the value is "yes", otherwise it's "no". These are all synonymous:

```
make-backup-files: yes
make-backup-files: y
make-backup-files: true
make-backup-files: t
make-backup-files: 1
```

Inhibit: If the variable's line/block contains a "backup-inhibited" variable, and its value is true (see above), then the file will never be backed up, even if "Make backup before saving" is turned on in the global settings.

Rescue untitled document contents...

When this option is on (as it is by default) and you close an untitled document (one that has never been saved to disk), and click "Don't Save", BBEdit will preserve a snapshot of that document's contents. If you later need to recover the contents of that document, just choose the Rescued Documents command in the Window menu to open a Rescued Documents browser, which allows you to review and restore the contents of any previously-discarded documents.

Remove rescued items after N days

When this option is on, BBEdit will automatically perform some housekeeping for you by cleaning up rescued data. By default, BBEdit will clean up old data after a week, but you can adjust the interval from 1 to 365 days; or disable the cleanup altogether. (BBEdit will clean up old items by moving them into the Finder's Trash.)

Text file name extensions

This list allows you to add filename extensions for files that BBEdit does not intrinsically know about, but which should be treated as (plain) text files. You can use the "+" and "-" buttons to manipulate the list; or drag one or more file(s) to the list to add its extension.

Note If a file already has a language mapping in the Languages settings pane, BBEdit already knows that it's a text file so you need not add it here.

Expert Settings

In addition to the ordinary settings options which you can adjust in its Settings window, BBEdit supports a number of expert settings options which you can adjust by issuing an appropriate 'defaults write' command.

The Expert Settings pane provides background information about BBEdit's expert settings options, and a Expert Settings Help button which you can press to open the "Expert Settings" page of the Help book.

Expert Settings Help page

You can find a complete, current listing of these options in the “BBEdit Expert Settings” page on our website:

```
https://www.barebones.com/support/bbedit/ExpertSettings.html
```

which you can open at any time by pressing the “Expert Settings Help” button.

Expert Settings pane

Alternatively, you can use the list pane within the Expert Settings window to scan or search a list of all available expert settings options, their current values, and their default values. Settings which have been changed appear in boldface in the list, or you can enable the “Only show non-default values” option to display only values which differ from the factory defaults.

Selecting one or more items from the list of expert settings will enable the standard Copy command. Choosing this command will copy formatted “defaults write” command(s) to the clipboard, from which you can paste them into a shell worksheet or a Terminal window for editing and execution. (The copied commands will reflect the default values for the respective settings.)

Finally, if necessary you can click the Restore Defaults button in this pane to reset **all** expert settings to their factory defaults.

Website configurations

Website configurations are no longer stored in BBEdit’s settings; instead, you may apply or modify site configurations on a per-project basis via the Web Site Settings dialog accessible via the Sites (cloud) popup menu within a project window. For complete details on website configurations, please see “Configuring Websites” on page 283.

The Setup Window

The Setup window allows you to manage several types of configuration info which BBEdit uses, including FTP/SFTP bookmarks, file filters, and grep search patterns. (In older versions, most of this information was managed through the Settings window.)

Folders

The Folders panel lists all potential subfolders of BBEdit’s application support folder. You can click on any folder’s name to open that folder in the Finder (creating it first, if necessary).

In addition, each folder is a drop target, so you can drag a file (or files) onto it and BBEdit will move that file to the appropriate directory, or if you hold down the Option key, then BBEdit will instead copy that file into the chosen folder.

You can also access all of BBEdit’s application support subfolders via the Folders submenu of the BBEdit (application) menu.

Patterns

The Patterns panel lists all the grep patterns (regular expressions) you have stored via the Grep pattern popup in the Find and Multi-File Search windows. These patterns are also available in most commands which allow you to specify grep patterns, such as the Process Lines commands in the Text menu.

You may click the plus (+) button to create a new pattern, double-click any pattern item to edit its stored options (or rename it), or select a pattern and click the minus (-) button to remove it.

Bookmarks

The Bookmarks panel lists any bookmarks you have created for FTP and SFTP servers. You may click the plus (+) button to create a new bookmark, double-click any bookmark item to edit its stored options (or rename it), or select a bookmark and click the minus (-) button to remove it.

Further, there is a “go to” arrow to the right of each bookmark and clicking on that arrow will cause BBEdit to open the bookmarked location in an FTP/SFTP browser

Additionally, the Patterns list of the Setup window now contains “Import” and “Export” buttons which can be used to export patterns for sharing with others, or import patterns provided by others. “Export” will export only the selected patterns; if none are selected then it will export all patterns.

Clippings

The Clippings panel lists all the clipping sets (folders) present within the “Clippings” subfolder of BBEdit’s application support folder.

Filters

The Filters panel lists all the file filters you have defined for use with multi-file searches, Find Differences, and disk browsers. You may click the plus (+) button to create a new filter, double-click any filter item to edit its stored options (or rename it), or select a filter and click the minus (-) button to remove it. (For more information on using file filters in searches, see Chapter 7.)

This panel makes it easier to access these folders since by default the OS hides your account’s local “Library” folder in the Finder.

CHAPTER

11 BBEdit HTML Tools

This chapter describes the use of BBEdit's HTML Tools, a powerful suite of utilities for creating and maintaining HTML documents and entire websites.

In this chapter

Introduction to the HTML Tools

IMPORTANT Please be sure to read both this introduction and the next section, "Configuring Websites," before attempting to create web pages using these tools.

Already the most powerful set of utilities ever created for web developers, BBEdit's built-in HTML commands are more powerful than ever. These commands streamline the process of creating HTML documents, help you check for common usage errors, and speed up development time, without sacrificing flexibility or forcing you to work within the limits of visual editing tools.

BBEdit's HTML and CSS editing features and this chapter are written with the assumption that you already understand HTML. If you do not, we suggest one or more of the references listed below. None are published by or otherwise affiliated with Bare Bones Software, Inc., but other BBEdit users have found them useful for HTML usage and design issues.

Recommended Books

HTML for the World Wide Web with XHTML and CSS: Visual QuickStart Guide (6th Edition), Elizabeth Castro. Peachpit Press, 2006. ISBN: 0-32143-084-0

Cascading Style Sheets: The Definitive Guide (2nd Edition), Eric A. Meyer. O'Reilly and Associates, 2004. ISBN: 0-596-00525-3

Recommended Online Resources

HTML Help by The Web Design Group
`https://www.htmlhelp.com/`

The Bare Bones Guide to HTML by Kevin Werbach (no relation to Bare Bones Software)
`http://werbach.com/barebones/`

The W3 Consortium site
`https://www.w3.org/`

evolt.org — Browser Archive
`https://browsers.evolt.org/`

WebMonkey by Wired.com
`https://www.webmonkey.com/`

What You Need

Before you start, make sure you have the following available:

- A modern Web browser for previewing your pages. Safari is the obvious choice since it's supplied with the system, but you may want to try Firefox or Google Chrome as well. (These are the most widely used browsers but they often do not display pages the same way.)
- A general-purpose file transfer client such as CyberDuck, Fetch, or Transmit. While BBEdit does have built-in support for opening and saving files via FTP and SFTP, such dedicated applications are naturally more powerful, and of course also allow you to upload things other than text files. You will find them useful in creating and managing your website.
- Access to a web server, either your own or someone else's, where you will publish your pages on the web. (Your Internet service provider can help you find the answers to questions about using their server facilities, or obtaining your own domain name, setting up your own dedicated server, and so on.)

You will also want to be familiar with BBEdit's basic capabilities. The other chapters in this manual will help you learn more about editing and searching text using BBEdit.

Configuring Websites

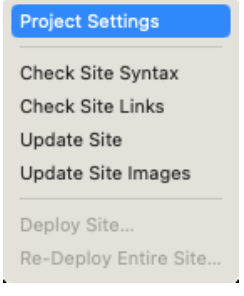
Before you start creating a new website, or making changes to an existing site's contents, we recommend you prepare a project document to manage your files and provide BBEdit some important details about them.

Creating a Project Document

To start, if you do not already have a local folder which contains working (draft) copies of your HTML documents, please create such a folder and put any HTML documents which belong to the site you are working on in this folder. (You can either move these files from elsewhere on your Mac's hard drive, or download them from your website.)

Once you have created a website content folder, launch BBEdit and create a new project document by choosing "Project Document" from the New submenu of the File menu, then add the local folder which contains your site's contents to the project's sidebar list, and save the project file to any desired location.

Next, click on the Project Settings (cloud) action menu in the project's sidebar, and choose the Web Site Settings command:



to open the project settings pane in which you can specify key details about the current website, and (if desired) configure deployment options that BBEdit can use to upload your site's contents to a remote FTP or SFTP server.

All of BBEdit's HTML editing commands which generate or operate on links, such as Edit Markup and Check Links, will recognize and take account of these site settings to ensure the accurate construction and editing of links within the files you edit. (Please refer to later sections of this chapter for further details on using these commands.)

Entering Web Site Settings

The site settings pane contains several sections, including: General, New Document Defaults, Syntax Checker Warnings, Link Checker Warnings, Image Updater, and Deployment, and the contents of each section are described below.

General Settings

The General section of the site settings sheet allows you to configure all the basic properties associated with the current website. (In many cases, this may be the only section of the setting sheet that you need to complete.)

Server URL

Enter the URL of your web server here, such as “https://www.example.com/” in the figure. BBEdit uses this information to determine which links are on (local to) your server.

Path on server

Enter the server path of your site’s main page here. For example, if your website is at “https://shared.example.com/foo/bar/”, you would enter “https://shared.example.com/” for the Web Server Name (as noted above) and “foo/bar/” for the Site Path on Server.

Default page

Specify the default name used by your server for the document that is sent to a web browser when a browser accesses a directory without specifying a file name. Examples include “index.html”, “index.php”, “index.shtml”, and “index”.

This field may also contain a comma-separated list of default file names. Thus, if for example you want both “index.html” and “index.php” to be recognized as default page targets when checking links, you would enter “index.html, index.php” in that field. (Please note this is purely an illustrative example as BBEdit will recognize both of those names by default.

Addressing

You may use this option to specify how BBEdit should generate links for anchor and image tags (and other tags with URI attributes) within all files contained by the current site. The available methods are:

- Automatic: When the linked file is in the same folder as the document (or a descendant of that folder), generate the link relative to the document. Otherwise, generate the link relative to the site root directory.
- Relative to site root: Always generate the link relative to the site root directory.
- Relative to document: Always generate the link relative to the document.

Local Site Root

Click the button to the left of this option and use the standard folder navigation dialog to select the root folder containing your local copy of the website’s content. To open the current root folder in the Finder, double-click on the graphical path below this option.

Look for templates and include files in

Click the button to the left of this option and use the standard folder navigation dialog to select the local folder that contains your HTML document templates and include files. To open the current templates folder in the Finder, double-click on the graphical path below this option.

Use local preview server

If you have a web server running on your Mac, you can preview HTML pages through it by activating this option, and entering the base URL for your preview server. (Depending on your needs, you can activate and manage your Mac's built-in Apache web server via the command line, or install the macOS "Server" application (available in the Mac App Store), or install and use a separate server package such as MAMP.)

Preview server URL

When you are configuring a new website, if the local site root folder is located within the "Sites" folder of your home folder (~/Sites/), BBEdit will create an appropriate local http URL and enter it in this field. Otherwise, you can specify the local http URL for your site root folder.

Note If your web content folders are not located within ~/Sites/, or if you want to use virtual domains, need to enable PHP, etc., you must modify your machine's Apache config file accordingly, since BBEdit's site configurations cannot directly enable such an arrangement.

New Document Defaults

The New Document Defaults section of the site settings sheet allows you to specify the default properties that BBEdit should use when you create a new file within the current website folder.

Insert DOCTYPE

When this option is checked, BBEdit will insert the selected DOCTYPE into the created file. (This option is on by default.)

Insert XML declaration

When this option is checked, BBEdit will insert a suitable XML version declaration into the created document. (This option is off by default.)

Give BBEdit credit

When this option is checked, BBEdit will insert a 'meta name=generator' tag containing a notice with its own name and version number.

Language and Charset

These popup menus allow you to specify the default language type and character set declarations which BBEdit should place into the created document.

Syntax Checker Warnings

This section of the project settings pane allows you to specify the default options that BBEdit should use when you perform a syntax check. (These options match those available within the stand-alone Check Site Images command.)

Link Checker Warnings

This section of the project settings pane allows you to specify the default options that BBEdit should use when you perform a link check. (These options match those available within the stand-alone Check Site Links command.)

Image Updater

The Update section of the site settings sheet allows you to specify the default options that BBEdit should use when you apply the Update Site Images command. (These options match those available within the stand-alone Site Images command.)

Deployment

The Deployment section of the project pane allows you to specify the default actions that BBEdit should perform before deploying (uploading) the contents of the current site root folder to the designated FTP or SFTP server, and/or to any desired local folder (for testing purposes).

To save time, BBEdit tracks the modification dates of uploaded files, and will only upload files which have changed since the last deployment. (Any change to a file's content -- not the modification date -- will trigger a reupload.)

In BBEdit, the “Deployment” settings group in web site projects has been split into three new sections:

1.”Deployment Settings”, which includes the pre-deployment processing options.

2.”Deployment Location (Testing)”, which are the previous deployment location settings, carried over from existing projects.

3.“Deployment Location (Production)”, which is an additional deployment location.

The previous “Deploy Site” and “Re-Deploy Entire Site” commands (available in the Markup Update as well as in the “cloud” menu in the project’s action bar) have been renamed tp “Deploy Site to Testing” and “Re-Deploy Entire Site to Testing”. This pair of commands uses the “Deployment Location (Testing)” settings for upload.

A new pair of commands within the Update submenu of the Markup command (and the “cloud” popup in the project’s action bar) are respectively named “Deploy Site to Production” and “Deploy/Re-Deploy Entire Site to Production”. These function similarly, but will use the “Deployment Location (Production)” settings.

There is no functional difference between the “Testing” and “Production”deployments, except for the configured locations. In this fashion, you could (for example) use a local folder and/or server on your LAN for testing, and a remote file system and/or a public-facing server for production.

Before Deploying

When one or more of these options are checked, BBEdit will perform the specified actions before deploying the contents of the current site root folder.

(The “Remove image metadata” option will remove all metadata from the uploaded image without affecting the local original.)

Stop Deployment if Errors Occur...

When this option is checked and any of the specified deployment actions reports an error, BBEdit will halt and display a corresponding error browser.

Upload Settings

These fields allow you to specify the FTP or SFTP server to which BBEdit should upload the content of the current site root folder.

Server

This field should contain (only) the hostname of the desired FTP or SFTP server, for instance: “host.example.com”. If this host is an SFTP server, you should also enable the “SFTP” option below, or leave this option disabled for an FTP server.

(You may optionally include a port specification by appending it to the hostname with a colon, e.g. “host.example.com:8080”.)

User

This field should contain the user name of the server account that you wish to use.

Password

This field should contain the password for the server account that you wish to use. (BBEdit will store this password in your login keychain.)

Path

If you need to upload to a directory other than the default (home) directory for the specified server account, you may specify the path to that directory on the server in this field.

Deploying Site Content

Once you have created and configured a website project, you may deploy (upload) the contents of the current site root folder to the FTP or SFTP server designated in the Deployment Settings section.

The “Deploy Site” command for configured projects will now:

- Generate and upload HTML for Markdown (and Textile, if any such are in use) files.
- Perform placeholder and include processing on HTML documents (including HTML generated from Markdown and Textile, if applicable).

IMPORTANT In a significant change from previous versions, the transformations that BBEdit applies to all site content files (e.g. placeholders, includes, Update Images) **no longer affect** the site files on disk. This change vastly improves usability when the site files are under source control, since there are no more random batch changes to files as they are uploaded.

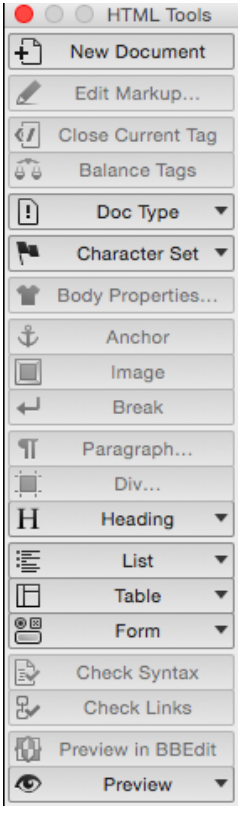
Note When you deploy a site, BBEdit will upload any non-HTML files which are preprocessed into HTML (such as Markdown or Textile) using each file’s base name with a file name extension of “.html”. Thus, for instance, BBEdit will upload “foo.md” as “foo.html”.

If however the site already contains a file whose unmodified name exactly matches the uploaded name, BBEdit will not upload the preprocessed file but instead report an error. So if your site contains the files “foo.md” and “foo.html” within the same directory, BBEdit will upload “foo.html” and report an error for “foo.md”

Creating and Editing HTML Documents

There are three ways to use BBEdit's HTML Tools commands: via the HTML Tools floating palette, via the Markup menu, and directly via the Edit Markup command and markup panel. These methods are functionally equivalent in most respects.

Many people find it easy to use the HTML Tools through the palette. There are three basic types of buttons on the HTML Tools palette:



- Those you simply click to perform an action or bring up a settings dialog before performing an action—for example, New Document, Close Current Tag, or Preview in BBEdit.
- Those that provide popup menus containing related options—for example, Heading, List, Table, and Form.
- Those which bring up BBEdit's markup panel to fill in attributes and values—for example, Anchor, Image, or Div.

The second means of using the HTML Tools is from BBEdit's Markup menu. This allows you to make your own choice between the drag and drop convenience of palettes, and the less screen-intensive menus; either way, you will still be able to access all of BBEdit's capabilities. Most common tags, as well as many utility functions, are available though items in the Markup menu or one of its submenus. Key equivalents (if assigned) are displayed next to the menu item. (You can change or set key equivalents for menu commands in the Menus & Shortcuts settings panel.)

Most of BBEdit's HTML Tools commands apply to the frontmost document—either at the current insertion point, or on the current selection range, as appropriate. Some utility functions, however, can operate on many documents. The Tool Descriptions section provides more details on what each command does.

The third way to use BBEdit's HTML Tools is by choosing the Edit Markup command to bring up BBEdit's markup panel, in which you can directly add or edit tags and their corresponding attributes and attribute values.

Creating a New Document

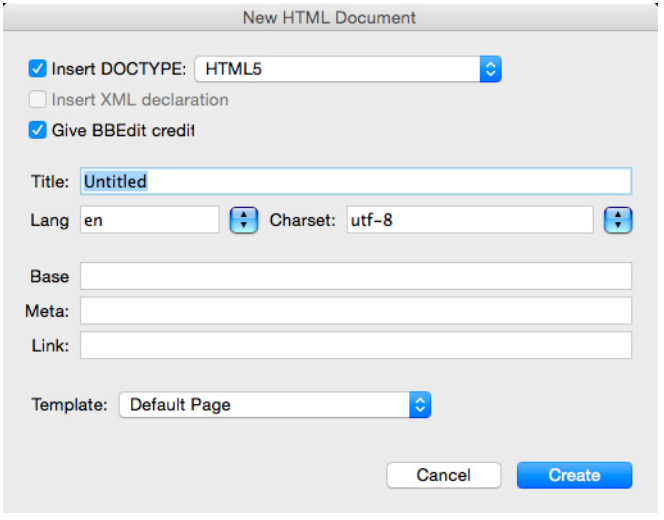
You can create an HTML document simply by taking any text file and adding HTML markup to it, but there's a better way. BBEdit includes a New Document command to create the basic skeleton of an HTML document for you.

To create a new HTML document, you can do either of the following:

- Choose New from the File menu and then choose HTML Document from the New submenu.

- Click the New Document button in the HTML Tools palette.

In either case, the following dialog appears:



In many cases, you can simply specify a title for the document and click OK, ignoring the other options. However, we suggest that you fill out this dialog as completely as possible. The function of each field is described below.

Insert DOCTYPE

Choose the type of this HTML document from the popup menu to have BBEdit insert an SGML prolog containing the desired document type. This information is largely ignored by browsers; however, HTML syntax checkers (such as the one built into BBEdit) use it to determine which constructs are legal according to the HTML standard you select. Available DOCTYPEs include:

- HTML5
- HTML 4.01 (Transitional, Frameset, and Strict versions)
- XHTML 1.0 (Transitional, Frameset, and Strict versions)
- XHTML 1.1

Insert XML declaration

Choose this checkbox to have BBEdit insert an XML declaration. If the DOCTYPE selected in the popup menu below is not an XML-based type (that is, is not an XHTML version), this checkbox will be disabled.

Give BBEdit credit

This option generates a <META NAME="generator" CONTENT="BBEdit [VERSION]"> tag in the document, indicating that you used BBEdit to create it.

Title

Enter the HTML title for the document (which can be different from the file name) here. This text will appear in the title bar of a browser's window when this document is opened.

Lang

This option indicates the language this document is written in. This information can be used by search engines and translation software to help Web users find pages in their own language.

Charset

This option indicates the character set used by the document. If you do not specify a character set, the character set chosen in the user's browser will be used.

Note You can choose which character sets appear in this popup menu by using the Text Encodings settings panel.

Base

Enter the URL for this document's BASE tag. The BASE tag indicates the actual location of the document on a server, and all relative URLs specified in the document will be resolved by the browser relative to this location. No BASE tag is created if you leave this field blank.

Meta

Enter the META tag to be included at the top of the document here, if any. (META tags can be used for "client-pull" techniques, for indicating search keywords, and for a wide variety of other purposes.)

Link

If you want to use a LINK tag to specify a relationship between this document and other documents, an email address, style sheet, or other information about the document, enter the desired information in this field.

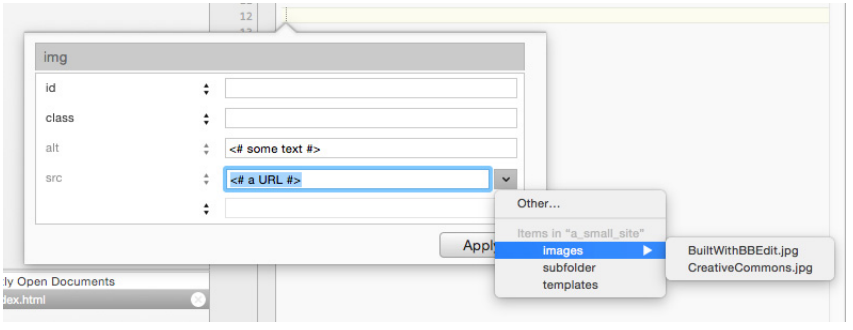
Note If you use a template to create the HTML document, the template must include the #META# and #LINK# placeholders to indicate the location at which this information should be inserted into the generated document.

Template

This popup menu displays the templates specified in the "Templates & Includes" folder associated with the selected website. (In order to appear in this menu, template files need only have a filename extension which maps to an HTML-like language, e.g. HTML, XML, Ruby in HTML, or PHP in HTML, or be a text file whose contents appear reasonably HTML-like.) Selecting a template other than Default will use the specified template to create a new document, potentially ignoring some or all of the settings specified in this dialog.

File Addressing

Many HTML tags require you to specify the pathname or URL of a file or folder, such as to identify a base address, style sheet, or hypertext link. When you edit such a tag in BBEdit's markup panel, you can type or paste the path directly or click the File button to the left of the attribute value field to bring up an Open sheet in which you can select the desired file, or (if the active document has been saved to disk) select the desired file from the popup menu list of directories and files within the current document's parent folder.



In general, URLs may be constructed in any of the following three ways:

- *Full addressing* specifies the complete URL, including the scheme ("http:"), the server's domain name, and the complete directory path leading to the file within that server.
- *Root addressing* specifies just the file's location within its host server.
- *Relative addressing* specifies the file's location relative to that of the HTML document referring to it.

For example, if the website resides on a server named "www.example.com" in directory "foo/bar," and you are creating a document in that directory named "index.html" with a link to file "target.html" in subdirectory "flapdoodle", the full address would be

```
https://www.example.com/foo/bar/flapdoodle/target.html
```

the root address would be

```
/foo/bar/flapdoodle/target.html
```

and the relative address would be

```
flapdoodle/target.html
```

Using the Check Syntax Command

You can use BBEdit's Check Syntax command (see page 317) to validate your HTML documents to the specification defined in their <!DOCTYPE> prolog. BBEdit will apply HTML5 rules when checking any document that does not contain a <!DOCTYPE> specification,

Note that an HTML document may display the way you expect it to in a browser and still contain invalid HTML. Browsers are designed to be lenient in the markup they accept, so you can get away with a certain amount of "sloppy" markup. However, producing well-formed (syntactically correct) HTML documents is the best way to assure that your document will display in some reasonable fashion in a wide variety of Web browsers, even those you have not tested the page in.

Syntax Checking Partial Documents

BBEdit can check the syntax of either complete or partial HTML documents. You may find the ability to check partial documents useful if you are preparing template sections for inclusion into other documents, whether this is done locally or via a server-side mechanism.

In order to check the syntax of a partial HTML document, the document must contain a balanced portion of the content tree. For example, you can check a partial document which contains a set of paragraphs or a table; you cannot check a partial document which contains an unclosed <BODY> or <DIV>.

Additionally, for partial documents which do not contain a DOCTYPE, you can specify one by means of a "#bbpragma doctype=" comment, which specifies what the root or parent element of the partial page's content is. For example, if your partial document consists of <BODY> content:

```
<!-- #bbpragma doctype="-//W3C//DTD XHTML 1.0 Transitional//EN"
root_element="body" encoding="utf-8" -->
```

In this comment, you must use either the public identifier text for the DOCTYPE you wish to check against (see above) or the "display name" for the document type used in the New HTML Document dialog.

Finally, you may also specify an "encoding=" attribute to declare the encoding of the partial document.

Ignoring Sections of Documents

You can mark sections of HTML documents to have BBEdit ignore these sections when performing a syntax check. This can be useful for purposes such as checking documents which must from necessity incorporate non-standard markup to support old browsers, or which contain customized server constructs. To mark a section, enclose it with "#bbpragma ignore_errors=" directives, as follows:

```
<!-- #bbpragma ignore_errors="on" -->

ignored markup

<!-- #bbpragma ignore_errors="off" -->
```

Note that when you check a document containing ignored sections, BBEdit's syntax parser still runs through the markup contained in these sections; it simply does not report errors encountered there. You should thus be mindful of the following conditions:

- The presence of fragmentary tags or similarly malformed content in an ignored section can cause a syntax check to fail.

- An error may still be generated for an unclosed element which resides within an ignored section, if its lack of closure results in an error cascade which continues beyond that section.
- If you terminate a document inside of an ignored section, an error will be generated.

Using the W3C Syntax Checker

BBEdit now offers support for using the W3C HTML checker service. This is on by default, and improves the correctness and accuracy of syntax checking in HTML5 documents. XML documents and non-HTML5 document types (as determined by the 'DOCTYPE' declaration) continue to use the built-in syntax checker.

Settings to control this are available in the "Syntax Checker" section of a project document's settings, as well as in the HTML language-specific settings (which you can customize via the "Custom Settings" tab in BBEdit's "Languages" settings).

Use of the W3C service involves sending your HTML files securely via HTTPS to the W3C checker service. Bare Bones Software never sees your HTML code, and the W3C service does not retain any data submitted to the syntax checker.

If you prefer, you can run the checker locally. To do this, you will need to follow some steps:

- 1.Download and install the Docker application for macOS: `https://www.docker.com/products/docker-desktop/`

  (If you don't already have a Docker account, you can set one up and the free tier will suffice.)
- 2.Using Homebrew or a similar package manager, install the 'docker' command-line tool: '`brew install docker`'.

  Alternatively, you can create a symlink named '`docker`' somewhere in '$PATH' to '`/Applications/Docker.app/Contents/Resources/bin/docker`'.
- 3.In BBEdit, configure the checker URL (in project settings or globally, as appropriate) to:

  `http://localhost:8888/`

  Steps 1-3 need only be done once.
- 4.To start the server, run this command:

  ```
  docker run -it --rm -p 8888:8888 ghcr.io/validator/
  validator:latest
  ```

This will also install and/or update the checker, as needed. To learn more about the W3C syntax checker, visit the page:

`https://validator.nu/about.html`

If you believe that the W3C syntax checker is inappropriately reporting an error or warning, we recommend you file a bug report against the checker:

https://validator.nu/about.html#issues

Export as HTML

When using the Export as HTML command (in the File menu), BBEdit will examine the generated output, and if that output lacks a basic HTML document structure, it will apply a simple structure. This is useful for HTML renderers which don't do this on their own (e.g. most Markdown renderers).

Format Customization

The “Pretty Print” option of BBEdit’s Format command is implemented internally using a Dreamweaver-style source format profile. Advanced users may override the factory format profile by placing an appropriately constructed file in BBEdit’s app support folder.

Previewing Pages

BBEdit’s Preview commands allow you to view your pages in one or more web browsers. You can display an automatically-updated page preview directly within BBEdit by choosing the Preview in BBEdit command, use the Preview in <Selected Browser> command to use the current default browser, or choose a specific browser from the Preview In submenu. You can also preview the page in all running browsers or in a text-only format.

NEW Further, when you preview a document, BBEdit will perform placeholder and include processing on that document so that the live preview reflects exactly what the results of site deployment (see page 286) would be.

In addition to static previews, BBEdit also supports live local previewing of web pages through the Apache web server built into your Mac. This capability enables you to easily preview pages which are built using server-side technologies, for example, DHTML or PHP.

To enable live previewing for any website project, you must turn on the “Use local preview server” option and enter an appropriate “Path on server” in that site’s configuration (see page 284). In addition, your Mac must be running a local web server, such as the OS’s built-in Apache server or a third-party server such as MAMP.

Once you have done so, whenever you preview a file from that website project using any of the Preview In commands, BBEdit will have the selected browser load that file’s corresponding page through the built-in web server.

Applying Preview Filters

BBEdit supports passing documents through a preview filter before displaying them in its built-in Preview window. You can use the “Preview Filter” popup in a Preview window’s navigation bar to route the document’s contents through the text filter of your choice before display. The default choice is “(language default)”; in this case, the preview contains the language module’s default HTML conversion, as before.

As an example, you could use BBEdit's preview filter support to override the built-in default Markdown-to-HTML conversion with something tailored more closely for your own needs, e.g. MultiMarkdown. After installing the MultiMarkdown package, you could create a symlink (or alias) from MultiMarkdown's 'mmd' helper script at "/usr/local/bin/mmd" to "~/Library/Application Support/BBEdit/Preview Filters/mmd", at which point you could choose "mmd" from the Preview Filter menu in the preview window.

Note If you wish to assign a keyboard equivalent for opening the Preview Filters menu, you can do so in the Menus & Shortcuts settings (look under "Preview Windows").

Creating and Using Preview Filters

Preview filters may consist of any of these three types:

- An AppleScript, with an entry point named "FilterTextForBBEditPreview". This entry point will receive a 'unicode text' object which is the document's contents. If there is no "FilterTextForBBEditPreview" entry point, the script's run handler will be called with the text. The script should return a 'unicode text' result.
- A Unix executable or a symlink to any such item. (For example, a copy of the 'multimarkdown' binary.)
- A Unix script; for example, a Perl, Python, Ruby, or shell script. (Any such script should contain a shebang line.)

Both Unix scripts and Unix executables will receive the document's contents as UTF-8 text on STDIN and should return UTF-8 text (ordinarily, in the form of an HTML document) to STDOUT. BBEdit will then display that output in the Preview window.

Preview filters may reside in one of two places: a "Preview Filters" folder within BBEdit's application support folder, or within an installed package's "Contents/Preview Filters/" directory. (So, for example, if someone supplied a BBEdit package for MultiMarkdown, it might conceivably contain a 'multimarkdown' executable that you could use immediately.)

By default, BBEdit will use the preview filter named "DefaultFilter_<language name>" (if such an item exists) to process all files whose language type is "<language name> before previewing them. In addition, BBEdit's Preview window will remember the Preview Filter selection on a per-document basis.

Unlike default preview templates and CSS (see below), the filename extension of the preview filter is not significant; so the following examples will all work:

- DefaultFilter_Markdown (a compiled executable)
- DefaultFilter_Markdown.pl (a Perl script)
- DefaultFilter_Markdown.scpt (an AppleScript)

The default preview filter can also be a symlink or alias to a filter elsewhere.

Applying Templates and Custom CSS

You can apply document templates and customized CSS to pages displayed by the Preview in BBEdit command. In order to do this:

- Place a fully structured HTML document in "~/Library/Application Support/BBEdit/Preview Templates/". This document may can contain anything you like but should define the basic structure and appearance of your desired page. Within the document, place this single placeholder: #DOCUMENT_CONTENT#.
- Make a new text document and add some content to it; you may add tagged content, however, this document should not have a complete HTML tag structure.
- Choose Preview in BBEdit to preview the document. BBEdit's Preview window will display a row of items in the preview bar, including "Template:" and "CSS:". In the Template popup menu, you can choose the template that you saved in the first step. When you do so, BBEdit will replace the #DOCUMENT_CONTENT# placeholder in that template with the contents of the document that you previewed.

Thus, you can use this technique to preview a fragmentary document without having to replicate the chrome defined in the template into that document.

Further, if you place a valid CSS document in "~/Library/Application Support/BBEdit/Preview CSS", that document will be available in the Preview window's "CSS:" popup menu and choosing it will apply that CSS to the Preview window's contents.

Default Language-Specific Templates

You can set a language-specific preview template to use in "Preview in BBEdit" windows. This is done by creating a file in the "Preview Templates" subfolder of BBEdit's app support folder whose name is of the form "DefaultTemplate_LANGUAGE.html", where "LANGUAGE" is the actual name of the language in the Document Languages popup menu. So for Markdown, for example, the default preview template would be named "DefaultTemplate_Markdown.html". The default template will apply when there is no previous template setting for that document.

Previewing Code and Text

BBEdit supports previewing documents which are not HTML (or HTML generators) via the Preview in BBEdit command. For example, if you preview a C++ document, you will see an HTML rendering of that document as BBEdit displays it in the editing window. (The HTML displayed in the Preview window is the same HTML markup that you can generate by using the Copy as Styled HTML or Save as Styled HTML commands.) This is useful in situations where you want to typeset your code, and wish to experiment with different page templates and styles.

When previewing documents in a particular language, you can specify a default CSS file for BBEdit to use by placing an appropriately named CSS file in the "Preview CSS" folder of BBEdit's application support folder. The name of that file should follow the pattern "DefaultCSS_*.css", in which the "*" is replaced by the name of the language. For example, to designate a default CSS file for previewing Markdown, you should name that file "DefaultCSS_Markdown.css".

Printing Previewed Pages

When a BBEdit preview window is frontmost, you can use the Print commands to print a copy of the displayed page.

Note Due to limitations of the WebKit rendering engine which BBEdit employs, the format of the printed output may not exactly match the screen rendition.

`~/Library/Application Support/BBEdit/SourceFormat.profile`

Exercising Emmet

If you have installed Emmet (see below), you can select the Expand Emmet Abbreviation command (in the Edit menu) to ask Emmet to expand the abbreviation on the line containing the insertion point, based on the line's contents and the position of the insertion point.

Note This feature requires that Emmet be installed, which you can do using 'npm install emmet'. That means you must also have Node installed, which you can do using Homebrew (use 'brew install node') or some other utility. As long as 'node' and 'npm' are in your '$PATH' and thus are usable from Terminal, BBEdit will be able to run Emmet.

Markdown Indentation

In Markdown documents, headings are now indented according to their level. H1 ("# this is an H1") is not indented, H2 indented by one space, etc.

The foldable section associated with a heading now encompasses lower level headings and their sections. So, an H1's foldable section will include an H2's section that immediately follows it.

These headings will be listed in the function popup with indentation to match this nesting, like an outline.

Additionally, blockquote sections (indicated with a '>') and list items “belong” to the heading section immediately before them,and act like nested documents. The result is that any headings within those subsections are indented below the parent section. Here’s a small example:

```
```

This is an H3

> This is a blockquote

>

> # This is an H1 in a blockquote

>

> This section will be nested in the function popup

> below/within the H3 at top of the example.

```
```

HTML Tool Descriptions

This section describes all of the HTML Tools commands as they appear in the hierarchical Markup menu. For a description of the tools as they appear on the palette, see the HTML Tool Palette Index, which appears after the tool descriptions.

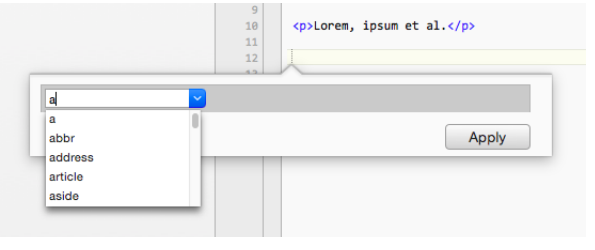
Note Tools that create tags insert the tag at the insertion point unless otherwise specified. Such tools also place an end tag automatically where appropriate.

You should already be familiar with HTML before using BBEdit's HTML tools. BBEdit's markup panel will help you associate correct attributes with each tag, and provide shortcuts to help you enter information; however, it does not (and cannot) know what intent or the final results of your markup will be. There is no substitute for knowing HTML.

Edit Markup

BBEdit presents a context-sensitive markup panel for creating and editing HTML tags. When you choose the Edit Markup command from the menu, or invoke it by pressing Control-Command-M, BBEdit will open the markup panel, in which you can select any tag that is valid in the current HTML context of the insertion point and then add attributes and attribute values to that tag.

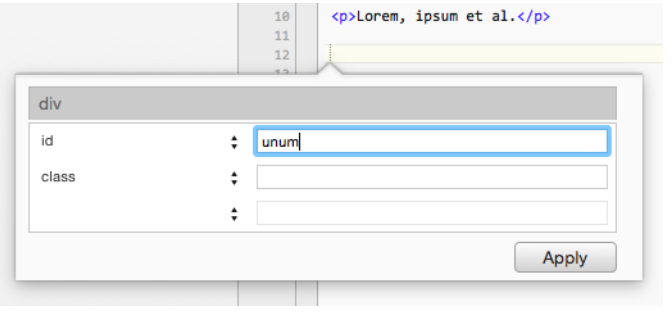
For example, if the insertion point is positioned directly inside the document's body section (delineated by the <body> and </body> tags), the markup panel will open in tag selection mode, and you can either use type-ahead or the arrow keys to select among any available tag:



If you have selected a tag which does not require attributes or you don't wish to add attributes, you may click Apply or press the Return key to enter that tag into the document. Choosing, for example, 'audio' will insert an <audio> tag at the insertion point, while choosing 'article' will insert an <article></article> tag pair (with any additionally chosen attributes) and leave the insertion point positioned between the two tags for easy content entry. You can close the markup panel without taking any action by typing the Esc key.

If you select a tag which has required or optional attributes, just press the Tab key to move the insertion point through the markup panel's fields, where you can choose or type the desired attributes and attribute values. To pre-insert all recommended attributes for a tag, you may either click the Fill (gear) control in the upper right corner of the markup panel or type Control-Command-M while the panel is open.

For example, here is the markup panel showing the DIV tag selected with pre-inserted attributes:

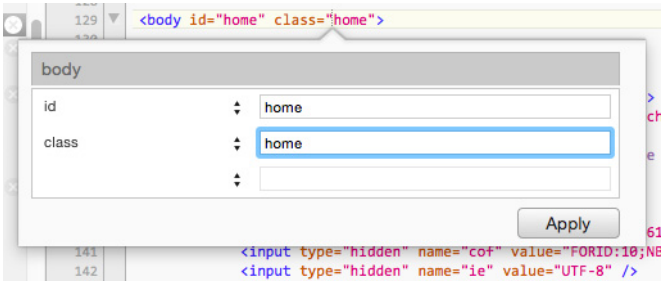


BBEdit's Edit Markup command also works within CSS. If the insertion point is within a CSS selector or declaration, BBEdit will display a sheet containing suitable options for editing the property (when possible). Invoking Edit Markup within a CSS context, but when the insertion point is not within a CSS selector will cause BBEdit to insert a new skeleton rule set.

BBEdit's markup editing capabilities are also available through the contextual menu. Simply Control-click at any point in your markup, and BBEdit will present all valid tags or attributes for the context of the insertion point within the Insert Tag submenu of the contextual menu.

Editing Existing Tags

You may also edit existing tags by placing the insertion point within them and choosing Edit Markup in the Markup menu or the HTML Tools palette (or by typing Control-Command-M). When you do this, BBEdit will bring up the markup panel pre-filled with all existing attributes and attribute values of that tag. For example, here is the markup panel invoked against an existing BODY tag:



Edit Markup also works with CSS. Choose Edit Markup while the insertion point is within a selector's property or value, and BBEdit will display a context-appropriate dialog for editing many common properties.

Close Current Tag

The Close Current Tag command inserts a closing tag to match the nearest opening tag preceding it. If the closing tag is placed on a new line, it will use the same indent level as the opening tag. For instance, if the insertion point is preceded by a <P> (Paragraph) tag plus some text content, Close Current Tag will insert a matching </P> tag to close the paragraph.

Note If you frequently work with HTML documents, you may want to assign a key equivalent to this command in the Menus & Shortcuts settings panel.

Balance Tags

When Balance Tags is chosen, BBEdit expands the selection to encompass the content of next outermost set of enclosing tags. The easiest way to understand how this works is to see it in action. Place the insertion point in an HTML document's <TITLE> element and choose Balance Tags. The title will be selected, since it lies between the tags <TITLE> and </TITLE>. Choose Balance Tags again, and BBEdit selects everything between <HEAD> and </HEAD>, the next set of enclosing tags outside the <TITLE> element. Choosing the command once more will select everything between <HTML> and </HTML>.

Use this command to quickly select an element for editing or just to check to see whether all your nested elements are formed correctly. If BBEdit sounds the system alert beep when you expect it to select text, it cannot find a matching set of tags around the selected text.

Document Type

The Document Type submenu allows you to select the desired document type (DTD) for the current document. (If the document already contains a DOCTYPE declaration, that option will be checked.) When you select a document type, BBEdit will insert the corresponding declaration into the document.

Character Set

The Character Set submenu allows you to select the encoding (character set) of the current document. When you select an encoding, BBEdit will insert the appropriate encoding declaration into the document

Note You can specify which encodings appear in the menu via the Text Encodings settings panel.

CSS submenu

This submenu allows you to create, edit, and format Cascading Style Sheet markup. BBEdit has built-in support for CSS. When you are editing stand-alone CSS files or HTML files with embedded CSS, syntax coloring is available, and the Function menu lists CSS selectors, as well as CSS files referenced by @import directives and <link> tags. Choose an external stylesheet from the Function menu and BBEdit will open that stylesheet for editing.

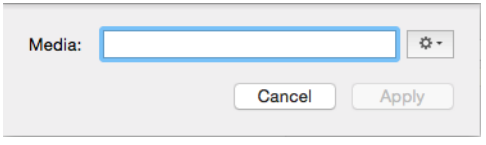
IMPORTANT BBEdit supports all CSS 2.1 properties. Although there are no explicit dialog editors for many such properties, you can create or modify them using the Tag Maker command, or by using Insert Property from the contextual menu.

The CSS function parser supports the following syntax for laying a mark in the function menu:

```
/* bbmark string to appear in the menu */
```

@media

The @media directive allows you to control which stylesheet should be used for different output media, e.g. screen versus printer. You can use this dialog to add media rules, or given an existing media rule, you can use the CSS editing dialogs or Edit Markup to edit the rulesets within.

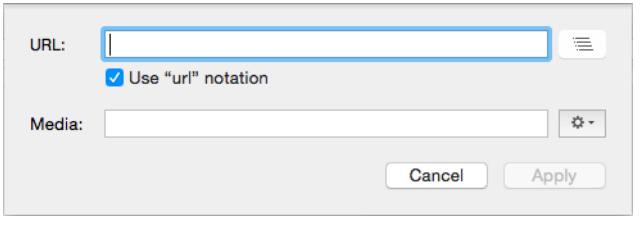


For example:

```
@media print {
    body { font-size: 10pt; }
}

@media screen {
    body { font-size: 13px; }
}

@media screen, print {
    body { line-height: 1.2; }
}
```

@import

The @import directive instructs a web browser to load an external style sheet. This dialog box allows you to select a file (or drag and drop one from the Finder) and choose whether to use the optional “url” notation for specifying the location of the style sheet. (Remember that @import must come before other CSS rules inside a <style> tag or in a stand-alone CSS document; otherwise it will not work.)



Format

The CSS Format command will reformat your CSS markup for easier reading.

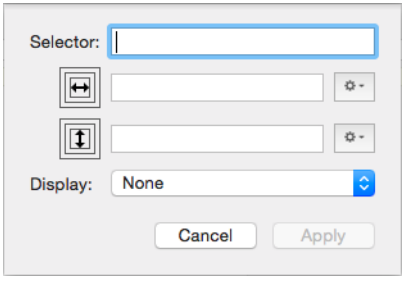
In stand-alone CSS files, if there is a selection range, only the selected text is formatted. If there is no selection range, the whole file will be formatted.

In HTML files with embedded CSS, if there is a selection range, only the selected text will be formatted. If there is no selection range, BBEdit will format all CSS in the <style></style> tag pair that encloses the insertion point. If the insertion point is outside a <style></style> tag pair, or if the selection range spans a <style></style> tag pair, the formatter will simply beep.

When formatting CSS embedded into HTML, BBEdit will indent the CSS based on the indent level of the opening <style> tag, plus one additional tab stop for better readability. BBEdit's CSS markup tools (listed below) use the same rules for formatting as the Format command.

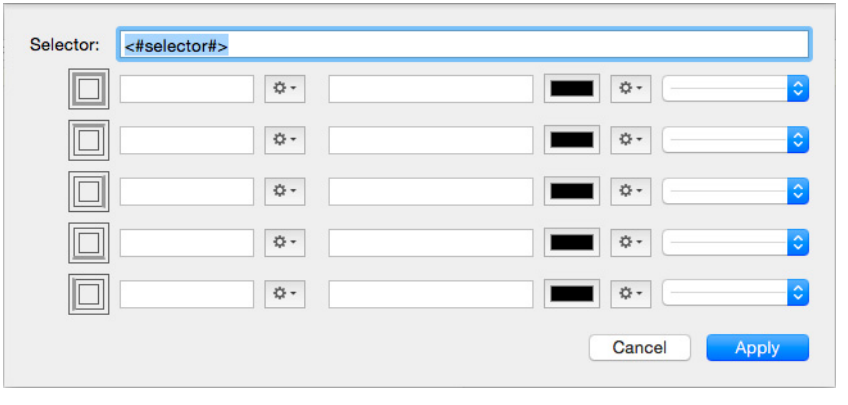
Box

The Box sheet allows you to specify a selector's width, height, and display properties.



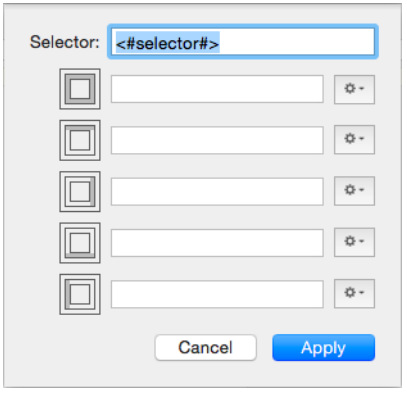
Border

The Border sheet allows you to edit the border-width, border-color, and border-style properties for a selector. The first row lets you specify values that apply to all four sides. The color buttons let you select colors using the system color picker; the popup menus next to them let you select colors by name. The icons on the left side of the dialog represent (from top to bottom), the entire border, top, right, bottom, and left.



Padding/Margins

These identical sheets allow you to edit the padding and margin properties. In both cases, the icons on the left in the dialogs represent the entire box, top, right, bottom, and left, respectively.



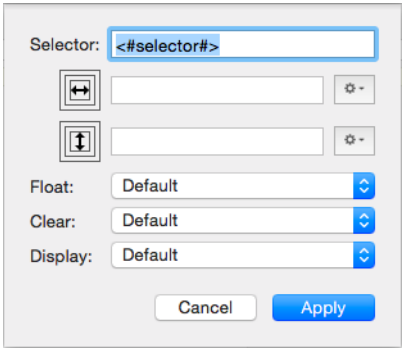
When you are working with the individual margin fields in these dialogs as opposed to the overall value, they behave the same way CSS value replication does:

- If right is missing, it takes on the value of top
- If bottom is missing, it takes on the value of top
- If left is missing, it takes on the value of right

so an empty field has special meaning - it means “replicate the related value”. If you want to specify a value for any given side, you must enter it explicitly.

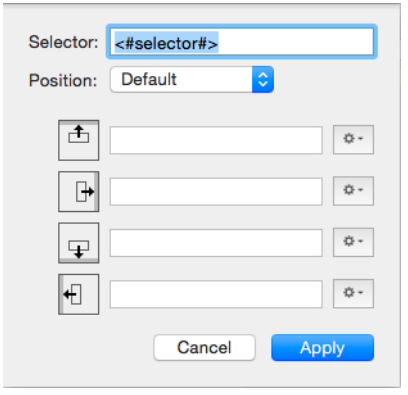
Layout

The Layout sheet allows you to edit the page layout properties of a selector.



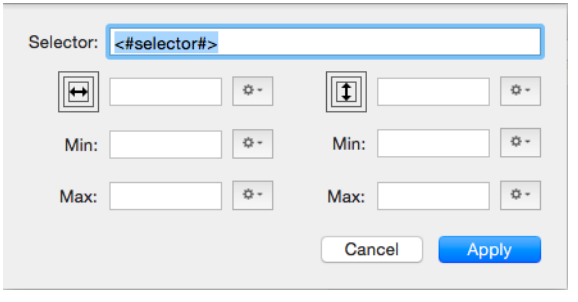
Position

The Position sheet allows you to edit the positioning properties of a selector.



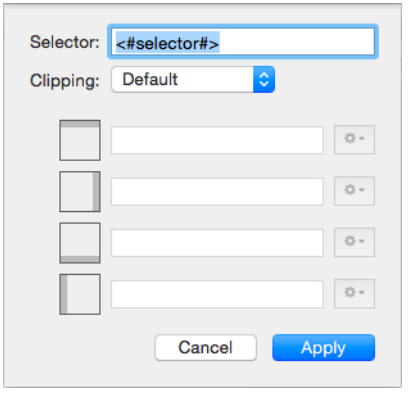
Size & Constraints

The Size & Constraints sheet allows you to edit the size and constraint properties of a selector.



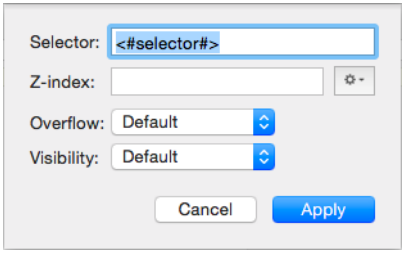
Clipping

The Clipping sheet allows you to edit the clipping properties of a selector.



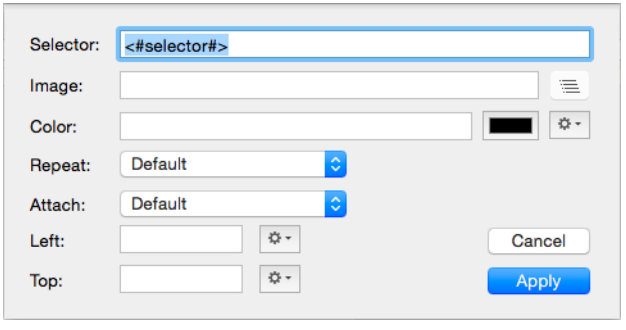
Effects

The Effects sheet allows you to edit the z-index, overflow, and visibility properties of a selector.



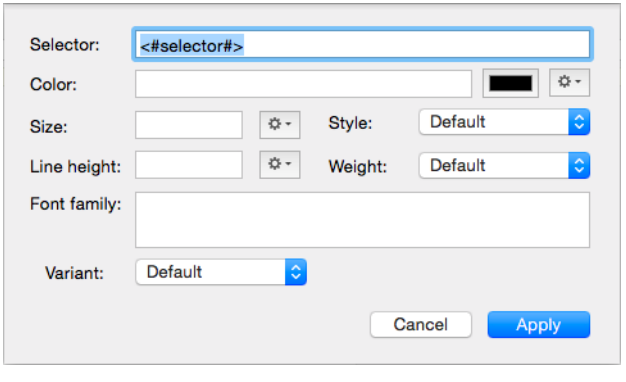
Background

The Background sheet allows you to edit background-image, background-color, background-repeat, background-position, and background-attachment properties. The Image field allows you to select an image file by clicking the File button, or by using drag and drop from the Finder.



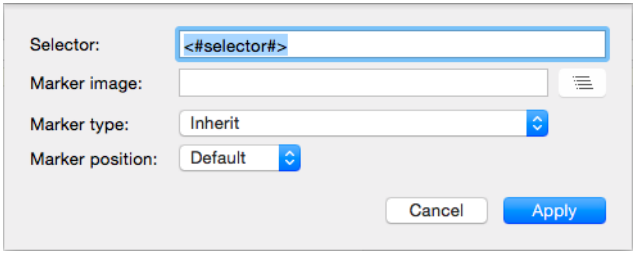
Font

The Font dialog allows you to edit values for the following font properties: color, font-size, line-height, font-family, font-style, font-weight, and font-variant. Note that BBEdit will parse the “font:” shortcut property, but never generates it; instead, BBEdit generates exploded values for font-style, font-variant, font-family, and font-weight.



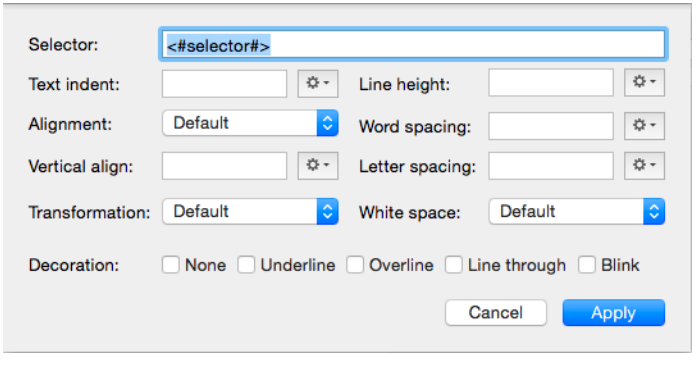
List

The List sheet allows you to edit the following list properties: marker image and addressing format, marker type, and marker position.



Text

The Text sheet allows you to edit the following text properties: text-decoration, text-indent, text-align, vertical-align, line-height, text-transform, word-spacing, letter-spacing, and white-space.



Body Properties

This command selects the BODY tag in the current document, and brings up the markup to edit it.

Head Elements

The commands in this submenu bring up the markup panel at an appropriate point within the current document's HEAD tag.

Base

The BASE tag determines the default location of documents referenced in the current document. The recommended attribute for this tag is 'href" whose value is the document or folder that all relative links within the document should be considered relative to. You can enter the Href by typing it into the supplied field or by clicking the Folder button to select a folder on your hard disk.

Link
The LINK tag tells the browser about a document related to the current document. The most common use for this tag is to point to an externally defined CSS stylesheet document. You can choose to indicate whether this link defines a REL (forward relation) or REV (reverse relation), the type of the relation (for example, a stylesheet), the media, the URL of the referenced file (including full, root, or relative addressing), and optional Type, Hreflang, Target, and Charset attributes.

Meta
META tags are used to define browser-specific or optional information that is not a part of the HTML specification. You can choose to create either a NAME or an HTTP-EQUIV variation of the META tag. The latter is frequently used for "client-pull" applications in browsers that support it, but more generically it makes most browsers behave as if the specified line was received as part of the HTTP protocol header. After choosing the type of META tag and the desired value of the NAME or HTTP-EQUIV field, enter a value for the tag's CONTENT attribute and optionally its SCHEME attribute.

Script
This command begins a section of client-side script code (by default, JavaScript, although some browsers support other scripting languages). You can choose to execute a script contained in an external file by entering a URL in the Source field (click File to choose the file using an Open dialog). You can also enter values for the TYPE of script, the script LANGUAGE it is written in, and the character set or CHARSET of the script.

Noscript
This command begins a section of HTML to be displayed only if the web browser does not support client-side scripting; often used to provide alternate content following a <SCRIPT> block. A matching </NOSCRIPT> tag is also inserted.

Style
This command begins a stylesheet declaration.

Block Elements
This submenu lets you add HTML elements that behave as paragraphs or other types of text blocks. Since many block elements can be containers for other elements, most have an explicit or implicit ending tag (for example, </P> to close a paragraph) as well as an opening tag (for example, <P>). If text is selected when one of these commands is chosen, these opening and ending tags are placed before and after the selection.

Paragraph
This command begins a new paragraph element. You can specify alignment, ID, a class (for obtaining formatting cues from a stylesheet), and inline CSS style information.

Div
This command begins a new division. A division is a generic block of text containing one or more paragraphs (or other block elements) that all have some type of structural attribute in common. Use <DIV> when no predefined block type is appropriate. You can specify alignment, ID, a class (for obtaining formatting cues from a stylesheet), and inline CSS style information.

Horizontal Rule
This command inserts a <HR> tag. You can specify the alignment, the thickness (size) of the rule, its width, and whether it contains a three-dimensional "shade."

Heading
This command inserts a heading of any level and allows you to specify the alignment of the heading.

H1 through H6
These commands insert a heading of the specified level. (The alignment attribute used will match the last one chosen when using the Heading tool.)

Address
This command inserts an <ADDRESS> block. The formatting of this element is browser-dependent but it is usually used to indicate that a block of text is a postal address.

Blockquote
This command inserts a block quote—that is, several lines of text that have been quoted from another document. (Most browsers display this as indentation, leading many authors to use this tag to indent a section of text, although stylesheets are a more correct and flexible way to accomplish this.) You may optionally indicate the document being quoted, if it is available on the Web, using the Cite field.

Center
This command inserts a block centering tag. This tag, while permitted in current HTML specifications, is deprecated since it includes no information about the content being centered. It is generally considered better form to use stylesheets or P or DIV tag ALIGN attributes instead.

Deleted Text
This command inserts a block formatted to indicate that the enclosed text has been deleted (usually with a horizontal line through it—that is, "struck out"). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Inserted Text
This command inserts a block formatted to indicate that the enclosed text has been inserted (usually by underlining the text). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Noscript
This command begins a section of HTML to be displayed only if the Web browser does not support client-side scripting; it is often used to provide alternate content following a <SCRIPT> block. A matching </NOSCRIPT> tag is also inserted.

Preformatted
This command begins a section to be reproduced with line breaks as specified in the HTML document. (Normally, browsers convert line breaks to white space for display, breaking lines only at <P> or
 tags.) Most browsers use a monospaced font for this type of block.

Lists

These commands add numbered or bulleted lists to your HTML documents. If text is selected, the selected text is converted to a list, with each line (terminated by a line break) becoming a list item.

List

This command inserts a list. You can choose the type of list (unordered, ordered, definition, menu, or directory) and also the type of marker for an unordered ("bulleted") list. You can also suggest a compact display format for the list. When converting existing text to a list, you can choose to ignore blank lines in the text being converted, to mark up only list items (and not insert the list header), and whether to indent the list items. When converting text to a definition list, DT (term) and DD (definition) tags are applied to alternating lines in the selection.

Unordered/Ordered/Definition/Menu/Directory

These commands convert the selected text to the indicated type of list, or insert a new list (as with the List command) using the options set in the last List dialog displayed.

List Items

This command converts selected text to list items (one line becomes one item), or inserts an <LI> tag if no text is selected.

Tables

The commands on this submenu all have to do with building HTML tables. HTML tables are frequently used for layout purposes as well as for the display of tabular data, although strictly speaking their use for layout should be avoided as much as possible.

Table

This command inserts <TABLE> and </TABLE> tags around the selected text. You can also specify border, width, spacing, padding, frame, ruling, alignment, and background color.

Row

This command inserts <TR> and </TR> tags around the selected text (if any). You can specify the desired horizontal and vertical cell alignment and a row background color. If horizontal alignment to a specific character is specified, you can also indicate the character that determines alignment and the character offset to the first alignment character in the line.

TD, TH

These commands inserts a table data cell element or a heading cell element, respectively. (Both have the same options, though most browsers render TH elements differently from TD elements.) You can specify the width and height of the cell, the number of rows or columns it should span, its vertical and horizontal alignment (including alignment to a character and the offset to the first such character), whether the text in the cell should be permitted to wrap, the background color of the cell, and the scope of the header information

in this cell, if any. You can also specify the axes, an abbreviated version of the cell's content, and which header cells contain information about the current cell. Many of the less familiar and infrequently used attributes have use in certain applications such as speech accessibility. To provide maximum accessibility for tabular data, we suggest you consult the appropriate HTML version specification.

Caption

This tag specifies a caption for a table. You may also optionally specify the caption's vertical alignment.

Colgroup, Col

These tags are used to define column and column groups. Browsers that understand HTML 4 tables can, for example, be told to format a number of columns the same way, or to place rules between column groups, using this construction. The contents of a column group may be one or more <COL> elements (or none at all, if the SPAN attribute is used). You can specify the span of the column or group, its desired width, and its vertical and horizontal alignment. Cells within this column group may inherit some or all of these attributes depending on the attributes of the individual <TD> or <TH> elements.

THead, TFoot, TBody

These tags define an optional table section element, which can be used independently of the <TH> tag. (The latter indicates that particular cells should be displayed in a heading "style", which is usually displayed by browsers as boldface.) <TH> may be used anywhere in a table that a "heading look" is desired. In contrast, these three related tags define the logical divisions of a table. Browsers might hold the table's header or footer fixed on the screen while scrolling a lengthy body up and down, for instance. All three tags allow you to select vertical and horizontal alignment, which may be inherited by cells inside the element depending on the attributes of <TR>, <TD>, and <TH> tags.

Create Table Shell

This command presents a sheet which offers various options for creating a prefabricated HTML table structure.

Convert to Table

This command provides a quick way to convert tab-delimited or comma-delimited lines of text to tables. You must specify the delimiter to be used (either tabs or commas), and you can optionally have the entire first row of the table or the first cell of each row converted to <TH> rather than <TD> elements. If One Cell Per Line is marked, each cell will be placed on its own line in the resulting HTML; otherwise cells will be placed on a single line.

Forms

This submenu contains commands that help you build HTML forms, which are used for accepting user data for processing by a client-side script or a server-side CGI program (or other server-side technology, such as Active Server Pages).

Form

This tag defines a form. The Method can be either GET (encoding the form data in the URL) or POST (sending the form data separately after the HTTP transaction header). The Action should be the URL of the CGI program (or other server-side script, such as ASP). Enctype and Accept-Charset define the encoding type and character set for the transaction (usually, you will not need to use these fields). Use the On Submit and On Reset fields to enter the names of JavaScript handlers to be used for the Submit and Reset buttons, respectively. The Target field sets the frame to be used for the page returned by the CGI.

Button

This tag creates a form button. Choose a type (Submit, Reset, or Button), specify a name and value for the form element, and set optional attributes such as Disabled, Tab Index (the order in which the button will be reached by the Tab key), and Access Key (the key the user can press to activate the button in the browser). (The latter two options are HTML 4 features and may not work on all popular browsers.) You can also enter the names of JavaScript onFocus and onBlur handlers for the button.

Field Set, Legend

In HTML 4, you can group your form's fields and other controls into sets of related fields by using the FIELDSET container. Within the FIELDSET container, the LEGEND tag is used to define a title for the field set. Browsers differ in how they represent field sets visually, but some browsers may draw a rectangle around the related controls as in dialog boxes. In this case the Align attribute of the LEGEND tag can be used to set the alignment of the legend relative to the visual representation of the field set. (Browsers that do not support these tags will ignore them, and the contents of the LEGEND container will be displayed as any other text.)

Input

This tag defines an input field, which can be a text or password input, various types of buttons, and even files, images or hidden fields. Specify the name and the default value of these fields, and, if applicable, their size, maximum length, tab index, access key, and disabled or read-only attributes. (Disabled, Read Only, Tab Index, and Access Key are HTML 4 features and may not be supported by all popular browsers.) You may also specify handlers for the JavaScript onFocus, onBlur, onSelect, and onChange handlers.

Label

HTML 4 allows you to specify that text next to a control is a Label, and in browsers that understand the tag, clicking the label associated with a button activates the corresponding control. BBEdit lets you create a <LABEL> tag, specifying the name of the control it should be associated with, an optional keyboard equivalent to activate the control, and onFocus and onBlur JavaScript handlers.

Select

This tag defines a scrolling list or popup menu. Enter the name of the control, the number of items to display (leave the size blank for a popup menu rather than a scrolling list), and whether the list allows multiple items to be selected. Optionally mark the control as disabled and specify onFocus, onBlur, and onChange handlers.

Option Group

Using the <OPTIONGROUP> tag, you can create submenus in popup menus in browsers that support them. All <OPTION> tags within an <OPTIONGROUP> container are displayed as items cascading from the specified submenu label. (In browsers that do not understand <OPTIONGROUP>, users will see a simple straight list of all defined options.)

Option

This tag defines an option in a popup menu or a scrolling list. Enter the desired label and value for the option, and mark the Selected checkbox to make the option the default or initial choice.

Text Area

This tag defines a scrolling text area field for entering large amounts of data. You can specify the name of the file, its size in rows and columns, and optional HTML 4-only attributes such as Disabled, Read Only, Access Key, and Tab Index. You can also specify script handlers for onFocus, onBlur, onSelect, and onChange events.

Inline Elements

Inline elements are HTML elements that can appear as part of a paragraph, such as anchors, images, applets, client-side scripts, image maps, and more.

Anchor

This command inserts an HTML anchor (<A>) tag. Anchors can either be hyperlinks or be used as the target of hyperlinks to provided multiple targets on a single page. The anchor must have an associated URL in the HREF field to be a link; it must have a name in the Name field to be a target. The Target field is used to specify which frame the linked page should appear in.

Image

This command inserts an <IMG> tag to display an image. As with the anchor tag, you can select the Source by choosing a file or typing a URL.

After choosing an image file, you can specify alternate text (which will appear in browsers that do not support images or for users who are surfing with image-loading turned off), enter the Size of the image, select the amount of horizontal and vertical Space for wrapping around the image, and choose the thickness of the border and the image's alignment. (Image height and width should be specified whenever possible to speed layout of the page in the browser; BBEdit will enter these values for you automatically when you choose an image file.)

If you drag and drop an image file into an HTML document, BBEdit will automatically generate and insert an image tag at the drop point with the image's actual dimensions pre-filled.

NOTE Dropping an image file into a non-HTML document (e.g. a document whose type is "Text File") will instead generate a Markdown-style image link.

Applet

This command inserts the <APPLET> tag for specifying a Java applet. You will need to specify the location the folder that contains your main Java class file (the codebase) as well as the name of the main class file. If the file is in a .ZIP or .JAR archive, you can specify its name here as well. If you will control the applet via a client-side script, enter a name for it. You should always enter the desired size for the applet's display area. You can also specify alignment and white space around the applet, along with ALT text to be displayed if the applet cannot be used.

Object

The <OBJECT> tag is a generic tag for including almost any type of data in a page, including images and Java applets. (It can also be used to insert ActiveX controls and data intended to be used by browser plug-ins.) However, this tag may not be supported in all popular browsers. For this reason we suggest using <IMG> and <APPLET> for those types of objects and use <OBJECT> only for embedding other types of data, such as that used by plug-ins. For an example of this, see the Web Design Group's HTML Help reference page:

`https://www.htmlhelp.com/reference/html40/special/object.html`

The <OBJECT> tag, like the <IMG> and <APPLET> tags, allows you to reserve screen space in the browser window, recommend an amount of white space between the object and surrounding text, align the object, set its border, specify alt text to be displayed if the object cannot be displayed, and so forth. You will also need to specify at least the codebase and class ID of the object for ActiveX controls, and fill in the Data field for embedded objects such as Shockwave animations which will be handled by browser plug-ins. The Standby field can be used to tell browsers a text message to be displayed while the object is loading. For more information on the <OBJECT> tag, consult the HTML 4 specification.

Param

To pass parameters to a Java applet, ActiveX control, or browser plug-in, the <PARAM> tag can be used between the <OBJECT> and </OBJECT> (or <APPLET> and </APPLET>) tags. Each parameter to be passed to the object requires a separate <PARAM> tag. You must specify the name and value of each parameter; the actual parameter names and values required will vary depending on the object being embedded.

Script

This tag begins a section of client-side script code (by default, JavaScript, although some browsers support other scripting languages). You can choose to execute a script contained in an external file by entering a URL in the Source field (click File to choose the file using an Open dialog). You can also enter values for the TYPE of script, the script LANGUAGE it is written in, and the character set or CHARSET of the script. Mark the DEFER checkbox to add a DEFER attribute.

Note You can choose which character sets appear in the Charset popup menu by using the Text Encodings settings panel.

Map

This tag embeds a client-side image map in the document. You must enter a name by which the map can be referenced in the Use Map attribute of the Image tag. Individual clickable areas within the image map are provided by the <AREA> tags inserted between the <MAP> and </MAP> tags.

Area

This tag defines a clickable area within a client-side image map. Each clickable area requires a separate <AREA> tag. You will need to specify the document to be loaded when the area is clicked (or mark the No HREF checkbox to cause clicks in the area to be ignored), along with its Target frame if the page is being used in a frameset. You can choose the desired map shape (rectangular, circular, polygonal, or the default URL) using the Shape popup menu and enter the desired coordinates of the shape in a comma-separated list in the Coords field. (For rectangles this is in the order left, top, right, bottom; for circles it is in the order X, Y, radius. For polygons this should be a comma-separated list of coordinates in X, Y form.) You can also set the tab index of the field for keyboard control on browsers that support it. You may also wish to specify JavaScript onFocus and onBlur handlers.

Break

This command enters a line break tag,
, into the document. If multiple lines are selected, a line break tag will be inserted after each.

Font

This tag selects the font, size, and/or color for the selected text. This tag is deprecated and should generally not be used; stylesheets are a more flexible and more content-oriented way of achieving this end.

Base Font

This tag selects the default font, size, and/or color for the text in this document. Like <FONT>, this tag is deprecated; it is considered better form to use stylesheets.

Bidirectional Override

This command inserts a <BDO> tag to note that the enclosed text is in a language that should be rendered in a different direction (either left-to-right or right-to-left) than the default text order for the document's primary language. You can specify the desired text order and the language, so that savvy browsers can switch fonts or script systems to display the text correctly.

Quotation

This command marks the selected text as a quotation. Use this only for short quotes within a paragraph; use <BLOCKQUOTE> for quotations consisting of a paragraph or more of text.

Span

This command marks the selection as belonging to a certain class of information—such as a book title—usually so that its text style can be retrieved from a stylesheet. (In contrast with <DIV>, which marks paragraph-level classes, <SPAN> marks character-level classes.) You will be prompted for an ID for this span, a class name (which should correspond to a stylesheet entry), and inline style information. All are optional.

The Span command can also create nested span elements. This means that in order to edit an existing span element (since they can be nested), you must place the insertion point within the open tag of the desired instance.

Subscript

This command marks the selected characters as a subscript (lowered below the baseline).

Superscript
This command marks the selected characters as a superscript (raised above the baseline).

Phrase Elements
Phrase elements are HTML tags that mark sentences or phrases within a block element (such as a paragraph) with certain content-related styles, such as emphasis, strong emphasis, citation, and so on. Indirectly this determines the displayed format of the enclosed text (although exactly what "emphasis" and so on mean is left up to the browser or the stylesheet).

Abbreviation
The enclosed text is an abbreviation.

Acronym
The enclosed text is an acronym.

Citation
The enclosed text is a citation of another document.

Computer Code
The enclosed text is computer source code.

Deleted Text
This command inserts a block formatted to indicate that the enclosed text has been deleted (usually with a horizontal line through it—that is, "struck out"). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Defined Term
The enclosed text is term defined in a clippings.

Emphasis
The text should be displayed with visual emphasis (most browsers interpret this as italic text).

Inserted Text
This command inserts a block formatted to indicate that the enclosed text has been inserted (usually by underlining the text). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Input Text (Kbd)
The enclosed text is text to be entered on a computer keyboard (used in instructions).

Sample Output
The enclosed text is sample output from a computer program (used in instructions).

Strong Emphasis
The text should be displayed with strong emphasis (most browsers interpret this as boldface).

Variable
The text is a placeholder in an instruction or tutorial, and should be replaced with an actual value of the appropriate type before actually performing the indicated operation.

Font Style Elements
Like Phrase Elements, Font Style Elements mark relatively short pieces of text within a block element. However, they are concerned more with the appearance of the text than its structural function in the document.

Big
This displays the enclosed text in a larger font than usual.

Small
This displays the enclosed text in a smaller font than usual.

Bold
This displays the enclosed text in boldface type.

Italic
This displays the enclosed text in italic type.

Strike-Through
This displays the enclosed text in a strike-through style.

Teletype Text
This displays the enclosed text in a monospaced font, as on a computer terminal or teletype.

Underline
This displays the enclosed text in an underlined style.

Frames
The commands in the Frames submenu help you design documents that use frames. The first document loaded by the browser contains at least one <FRAMESET> tag and one or more <FRAME> tags, which specify the number and sizes of the desired browser window subdivisions and indicate the URLs of the files to be loaded into each.

Frame Set
This defines a frame set, a series of one or more frames. You indicate whether the frame set divides the browser window vertically (ROWS) or horizontally (COLS), and then indicate the size of each frame in a comma-separated list, using * to tell the browser to use whatever space is left over from the other specified frames.

Frame sets can be nested. For example, if you want to create a framed Web page with three rows, with the middle row divided into two independent columns, you would first define a frame set consisting of three rows. Instead of defining the second row with a <FRAME> tag, however, you would open another <FRAMESET> tag there, this time to specify the two columns for the middle frame (which would then be specified by <FRAME> tags).

Frame

This defines a frame in a frame set document. You will need to specify the URL of the file to be displayed in this frame (either using the button, or by drag and drop). If the frame will be targeted by links in another frame, you will also need to give the frame a name. You can optionally specify a long description for the frame, choose whether the frame can be scrolled, and indicate whether the user should be able to resize the frame. You can also set margins and borders for the frame. (Borders are the visible lines between frames. Margins determine how far each frame's content appears from its border or from the window edge.)

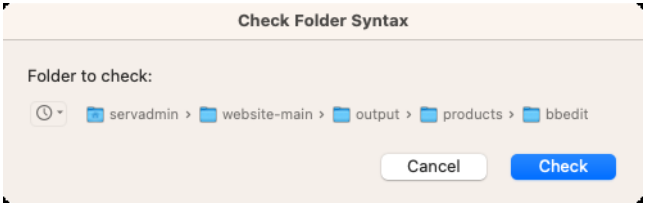
No Frames

HTML markup included between the <NOFRAMES> and </NOFRAMES> tags is displayed by browsers that do not support frames. This is where you should include a link to a non-frame (or text-only) version of the site. Although many current browsers support frames, some users do not like the feature and intentionally disable it in their browser.

Check

The Check submenu contains utilities for finding errors in your HTML markup and document links. You can run syntax or link checks on the current document, a specified folder, or any currently available website configuration. You can also perform a Balance Tags operation on the current document.

When you are checking a folder or a site using these tools, a dialog like the one below appears.



The popup menu to the right of the path strip includes all of the websites which are available within open projects; choosing any of these selects its designated site root folder for checking. The Other entry on the popup menu displays an Open dialog, allowing you to navigate to and choose any other desired folder. (You can also drag a folder from the Finder directly into the path strip.)

If a Check Syntax or Check Links operation generates any errors or warnings, BBEdit will display an error results browser listing. For more details on the error results browser format, please refer to Chapter 9.

Syntax

This command invokes BBEdit's syntax checker, which validates your HTML document to the specification defined in the <!DOCTYPE> prolog at the top of the document. Errors are displayed in an error results browser. Scroll through the list at the top of the window to see the errors that have been found; click to see the text that caused the error in the lower part of the window. Double-click an error message to open the file for editing.

In addition, the syntax checker (and the link checker) will now check for the use of explicit "file:///" URLs as attribute values. (Such references are problematic because they are nonportable and many web browsers will refuse to render them even locally, for security reasons.)

Links

This command causes BBEdit to scan your document, or a folder of documents, looking for links and object references (such as images and Java applets) that cannot be resolved. Note that BBEdit only looks at pages contained within your site's root folder as defined in the General section of the Site Settings sheet, not at any links that go offsite.

This command will now warn you to the presence of explicit "file:///" URLs; projects offer an option you can adjust to control this behavior when using "Check Site Links". (This option is on by default.)

Update

The Update submenu contains commands for updating IMG tags, includes, and placeholders in the current document, the selected folder, or the current site. BBEdit displays a results browser after the operation so you can see what was changed.

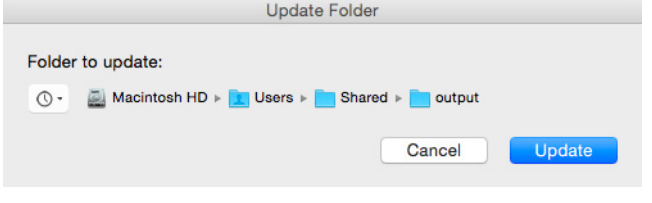
For more details on the results browser format that BBEdit uses, please refer to Chapter 9.

Choose the Document, Folder, or Site command from the Update submenu to update all includes and placeholders in the selected set of file(s). (Appendix C contains information regarding the use of placeholders.)

Choose the Document Images, Folder Images, or Site Images command from the Update submenu to update the HEIGHT and WIDTH attributes of image tags (and optionally to insert empty ALT attributes when missing) in the selected set of file(s).

You can also update the dimensions of any specific image by Control-clicking (or right-clicking) inside an image tag and choosing Update Image Dimensions in the contextual menu.

When you are updating a folder or a site using either of these sets of tools, a dialog like the one below appears.



The popup menu to the right of the path strip includes all currently-open projects which contain website configurations; choosing any of these selects its designated site root folder for updating. The Other entry on the popup menu displays an Open dialog, allowing you to navigate to and choose any other desired folder. (You can also drag a folder from the Finder directly into the path box.)

Includes

The Includes submenu contains commands for inserting one-time include directives, “persistent” include directives, and placeholders. See Appendix C for more information.

Choosing Include or Persistent Include prompts you to choose a file using an Open dialog and then inserts the appropriate markup. Choosing Placeholders displays a scrolling list of available placeholders; selecting one and clicking Insert places it into the document.

Utilities

The Utilities submenu contains commands for automatically editing the current HTML document for ease of editing and for consistency.

Format

This command formats the current HTML document for easier editing. The appearance of the document in a browser is generally not affected (except in the case of Document Skeleton). You can choose from among the following:

- Pretty Print: A balanced format suitable for general use
- Strict Hierarchical: All nested HTML structures are indented according to their depth

Note The “Strict Hierarchical” formatting method is disabled for HTML documents because it changes whitespace in ways that will adversely affect document layout and presentation.

- Plain: Places each tag on a separate line with no indenting
- Compact: Absolute minimum white space
- Gentle Compact: A slightly more human-readable version of Compact
- Document Skeleton: A hierarchical view with all non-tag content removed
- Don’t Reorganize: Allows normalizing of case, quote, and entity settings, as well as encoding entities within attributes, without otherwise changing the existing structure

You can also have the Format command operate on the whole document or only the selection, normalize the tags to uppercase or lowercase, normalize quote marks around attributes, and encode special characters, or entities, found in attributes.

If you choose the “Format…” command (with the ellipsis), BBEdit displays a dialog allowing you to choose the formatting options. If you choose the Format command (without the ellipsis), BBEdit uses the previous options.

Optimize

This command reformats the document to use the absolute minimum of characters while remaining syntactically valid. You will have difficulty editing your document in this format (in fact, if you do not have Soft Wrap turned on in the text options, you might think most of your document has vanished, because the command strips out all line breaks), but rest assured that your document will appear the same in your browser as it always has. Use one of the Format commands discussed above to put your page back into an editable format if you need to make changes. This command also applies the various Cleaner tools automatically.

Translate Text to HTML

This command allows you to translate plain text to HTML . The resulting sheet presents flexible options for converting paragraphs and translating extended characters to HTML entities. You can also choose to convert only the contents of the current selection or create a new document containing the results. (See "HTML Translation" later in this chapter for more information.)

Translate HTML to Text

This command allows you to translate HTML to plain text. The resulting sheet presents options for removing tags, converting tag-delimited paragraphs, and translating HTML entities to extended characters. You can also choose to convert only the contents of the current selection or create a new document containing the results. (See "HTML Translation" later in this chapter for more information.)

Remove Comments or Markup

This command removes all HTML comments or HTML tags, respectively, from the selection. Note that removing comments will not remove comment markers around client-side scripts like JavaScript, where they are required for proper functioning of the page on older browsers, but will remove the comment markers used by placeholders and indexes, making these items difficult to update in the future.

Comment, Uncomment

Note These commands have both been superseded by the Un/Comment command in the Text menu.

Raise Tag Case/Lower Tag Case

These commands convert all HTML tags in the document to either upper case or lower case.

Tidy

The Tidy commands provide various options for reformatting and cleaning up your HTML documents. To perform these commands, BBEdit makes use of code from the HTML Tidy Library project.

`https://www.html-tidy.org/`

Clean Document

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit presents a sheet in which you may select various options for modifying the frontmost document's tags. The available options are:

- Remove bogus markup: This option controls whether BBEdit should strip out surplus presentational tags and attributes, replacing them by style rules and structural markup as appropriate. (This option works well on the HTML saved by Microsoft Office products.)
- Insert blank line before
: This option controls whether BBEdit should output a line break before each
 element.
- Discard empty paragraphs: This option controls whether BBEdit should discard empty paragraphs. If you do not set this option, BBEdit will replace each empty paragraph with a pair of
 elements as HTML4 precludes empty paragraphs.
- Discard comments & optional tags: This option controls whether BBEdit should remove all comments and optional tags.
- Enclose block text: This option controls whether BBEdit should insert a <P> element to enclose any text it finds in any element that allows mixed content for HTML transitional but not HTML strict.
- Enclose text: This option controls whether BBEdit should enclose any text it finds in the body element within a <P> element. This is useful when you want to take existing HTML and use it with a style sheet.
- Escape <CDATA>: This option controls whether BBEdit should convert <![CDATA[]]> sections to normal text.

Reflow Document

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit presents a sheet in which you may select various options for reformatting the current document's tag structure. The available options are:

- Input is XML: This option controls whether BBEdit should use the Tidy XML parser rather than the error-correcting HTML parser.
- Indent block-level tags: This option controls whether BBEdit should indent block-level tags.
- Indent attributes: This option controls whether BBEdit should begin each attribute on a new line.
- Wrap ASP: This option controls whether BBEdit should wrap text contained within ASP pseudo elements, which look like: <% ... %>.
- Wrap Attributes: This option controls whether BBEdit should wrap attribute values, for easier editing.

- Wrap JSTE: This option controls whether BBEdit should wrap text contained within JSTE pseudo elements, which look like: <# ... #>.
- Wrap PHP: This option controls whether BBEdit should line wrap text contained within PHP pseudo elements, which look like: <?php ... ?>.
- Wrap script literals: This option controls whether BBEdit should wrap string literals that appear in script attributes. BBEdit will wrap long script string literals by inserting a backslash character before the line break.
- Wrap sections: This option controls whether BBEdit should line wrap text contained within <![...]> section tags.

Convert to XHTML

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit converts the contents of the frontmost document to XHTML.

If no DOCTYPE or namespace was present, BBEdit will set them as appropriate. If a DOCTYPE or namespace was given, it will checked for consistency with the content of the document, and in the event of an inconsistency, the corrected values will appear in the output. Entities will be written using the named form, and the original case of tags and attributes will be preserved.

Convert to XML

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit converts the contents of the frontmost document to well-formed XML. Any entities not defined in XML 1.0 will be written as numeric entities to allow them to be parsed by a XML parser, and the original case of tags and attributes will be preserved.

Check Accessibility

This option instructs BBEdit to check the frontmost document's compliance to various WCAG accessibility guidelines, at various levels of strictness.

Preview

The Preview commands provide various options for previewing your HTML documents in a web browser.

Preview in BBEdit

Choosing this command will open a live content preview window within BBEdit which is linked to the document that was frontmost when you chose the command. You can go back from the preview window to its corresponding source document by clicking on the document icon button in the preview window, or by choosing the Show Document command from the Markup menu.

The preview window uses WebKit (the standard OS-provided content display engine), and automatically updates whenever you modify the document. Closing the document will also close the preview window. (You can of course have multiple preview windows open on multiple documents.)

The preview window will not automatically display changes made in any related files, such as images or linked CSS files. However, you can use the Refresh BBEdit Preview command (see below) to update the preview window's display of both the source document and all related files.

If you choose this command when the current document is a Markdown source file, BBEdit will run that file through the Markdown script and generate a preview window which reflects how that file would be rendered in a web browser. The contents of the preview window will update as you edit the file.

Finally, the Save a Copy command is now enabled for live HTML preview windows; this allows you to save a copy of the rendered HTML of the document being previewed. (This can be useful in cases where the HTML was generated by a Markdown renderer or preview filter.)

Show Inspector

The toolbar of every preview window contains a 'Show Inspector' (or 'Hide Inspector') button, which when clicked will toggle display of the WebKit inspector pane on (or off).

Refresh BBEdit Preview

This command works in conjunction with the Preview in BBEdit command. How it behaves depends on the situation in which it's invoked:

- If the front window is a BBEdit Preview window, its associated HTML file will be reloaded, together with any related files which were changed behind the preview window's back (e.g. images, linked CSS files).
- If the front window is a text document, and there exists a preview window previously created by a "Preview in BBEdit" command on that document, then the associated preview window will be reloaded.
- If the front window is a text document, and any preview windows are open, the frontmost preview window will be reloaded, even if it does not necessarily belong to the front document.

BBEdit Preview windows also contain a "Reload" button, which has the same effect as this command. Finally, whenever you save a CSS file, BBEdit will automatically refresh all open BBEdit Preview windows.

Preview in <Selected Browser>

This command will display the name of the current default web browser, and choosing it will cause BBEdit to display the frontmost document in that browser.

You can choose a browser from the Preview In submenu, or select Preview in <Selected Browser> to use the last chosen browser. You can also preview the page in all running browsers or in a text-only format.

Preview in

The Preview in command provides a submenu listing all installed web browsers and versions. You may add browsers which are not listed by using the Preview Helpers settings panel (see page 269).

You can preview the frontmost document in any available browser by choosing that browser in the menu. Making such a choice will also cause BBEdit to use that browser as the default until you select a different browser.

Alternatively, you can choose the New Text Window item to generate a text-only rendering within BBEdit, or the All Running Browsers item to preview the current document in all running browsers.

The HTML Tools Palette

The main HTML Tools Palette is the place from which you will probably access the HTML Tools most frequently. You can invoke the HTML Tools palette at any time by selecting it from the Palettes submenu in the Window menu. BBEdit remembers which palettes you had open when you quit, so if you open the HTML Tools palette, it will remain open until you close it again, even on subsequent uses of BBEdit.

HTML Tools Palette Tips

A list of all the tools that are available on the HTML Tools palette appears below. In most cases, their behavior corresponds obviously with the tool descriptions in the previous section. In the few cases where there are significant differences, these are noted.

Some palette buttons are actually popup menus (indicated by a small downward-facing triangle on the right side), and clicking such a button will bring up a popup menu of options you can choose.

HTML Tools Palette

Tool	Menu-Based Equivalent
New Document	File > New > HTML Document
Edit Markup	Markup > Edit Markup
Close Current Tag	Markup > Close Current Tag
Balance Tags	Markup > Balance Tags
Doc Type	Markup > Document Type
Character Set	Markup > Character Set
Body Properties	Markup > Head Elements > Body Properties
Anchor	Markup > Inline Elements > Anchor
Image	Markup > Inline Elements > Image
Break	Markup > Inline Elements > Break

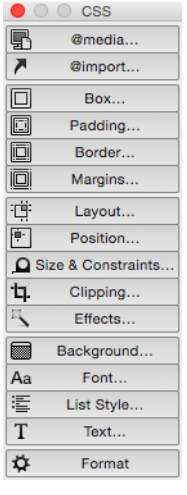
Tool	Menu-Based Equivalent
Paragraph	Markup > Block Elements > Paragraph
Div	Markup > Block Elements > Div
Heading	Markup > Block Elements submenu
List	Markup > Lists > List, Markup > Lists submenu
Table	Markup > Tables > Table, Markup > Tables submenu
Form	Markup > Forms submenu
Check Syntax	Markup > Check > Document Syntax
Check Links	Markup > Check > Document Links
Update	Markup > Update submenu
Preview in BBEdit	Markup > Preview in BBEdit
Preview	Markup > Preview in submenu

More CSS and/HTML Palettes

In addition to the main HTML Tools palette, BBEdit includes three other palettes that may be useful to HTML authors. These are the CSS palette, the Entities palette, and the Utilities palette, and you can display (or close) each of these palettes via the Palettes submenu in the Window menu.

CSS palette

Click any of the buttons in this palette to invoke the corresponding CSS editing dialog.

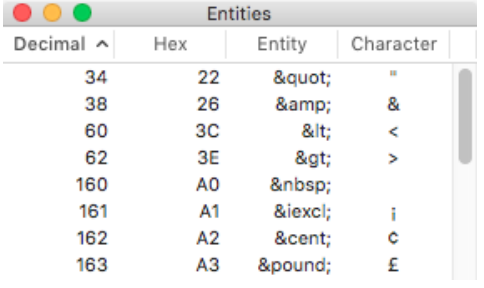


Font Style Elements palette

Click any of the buttons in this palette to invoke the corresponding command.

HTML Entities palette

In HTML, extended characters must be encoded as entities, since different computers define the extended ASCII characters differently. The HTML Entities palette lists these entities.



Entities can be inserted by name (“©” is the copyright symbol, ©) or number (“&169;” for ©) by choosing the desired method from the small popup menu at the top of the HTML Entities palette. (We suggest inserting entities by name, since they are more readable, unless browser compatibility requires use of the decimal versions.)

Double-click an entity name to insert it into the active document, or click once to select the desired entity and then click Insert.

The list of entities presented in the HTML Entities palette is sortable by decimal value, name (case-insensitive, so “&ecaute;” and “É” are grouped together), or base character (sorted by the character position after all diacriticals have been stripped, so that all “a”s are grouped, and so on). Click on a column label to set the sort order accordingly. (The default is to sort by decimal value.)

Utilities palette

Click any of the buttons in this palette to invoke the corresponding command.

HTML Translation

BBEdit provides the Translate Text to HTML and Translate HTML to Text commands to help you quickly transform existing content. Here is more information about how the conversion options provided by these commands behave.

Convert Paragraphs

When converting text to HTML, BBEdit finds paragraphs in the same way as the Paragraph command, and then adds opening and closing paragraph tags around them.

When converting HTML to text, BBEdit will (if necessary) place hard line breaks around each paragraph in the resulting text.

HTML Entities

When this option is set, the Translate Text to HTML command will convert all extended characters in the current document into HTML entities, using either names or the code (in decimal or hexadecimal). You can specify whether the tool should ignore < and >. This is useful when translating text already marked up as HTML. You can also specify that all Unicode text should be converted to entities.

Remove Tags

When this option is set, the Translate HTML to Text command will remove all HTML tags and comments.

Templates

In addition to providing many facilities for creation and markup of individual documents, the HTML Tools also incorporate a Template facility, which can be used to quickly create (or revise) a set of HTML documents that share a common format, structure, or content. You can design a skeleton document, make a template from it, and then use that template over and over again to produce new pages ready to fill with content, or to insert into existing text documents to provide an uniform structure or appearance. Templates may also employ placeholders and include files (see Appendix C), adding even more power to this useful function.

Template Setup

By default, older versions of BBEdit created a folder named HTML Templates within their application support folder. If this folder already exists, you may continue to use it as your templates folder; otherwise, you may set up your own template folder(s) wherever you like. (See “Look for templates and include files in” on page 284.) If you plan to maintain multiple sets of templates for different projects, you may find this option very useful.

Using a Template

A template is a simple text file that contains boilerplate text or HTML content that will form the foundation for the document you are creating. Template files must have the file name suffix ".tmpl" in order to be recognized.

When creating a template file, you can convert or reuse an existing document, or you can write one from scratch. Simply rename the file by adding the suffix ".tmpl" to it, and then move or copy it into your active templates folder.

Templates are always invoked using the New Document tool. The Template option appears as a popup menu at the bottom left of the New Document dialog, and all template files in the Templates folder appear in this menu. (The "Default" setting is not a template per se, but rather a directive to create a blank HTML document framework containing whatever Title, Base, Meta, Link, SGML Prologue (and so on) values you specify. It is always available, regardless of the contents in your Templates folder.)

Once you have specified the appropriate settings and chosen Create, BBEdit will open an new Untitled window containing the full text of the selected template file. Note that the template file itself is never changed by this action; rather, its contents are simply copied into the new document.

Note Templates can make full use of placeholders and include files, which are fully documented in Appendix C.

CHAPTER

12 Clippings and Cheat Sheets

This chapter describes BBEdit's powerful Clippings command and the use of clippings in preparing "cheat sheets". Clippings provide you an easy way to store and enter any sort of frequently used text, including program code, HTML markup, or just about anything else. Clippings can be language-sensitive, and their optional ability to run scripts and insert the results, further extends their flexibility and usefulness.

Cheat sheets are based on clipping placeholders, and intended to serve as a quick "training wheels" reference for pieces of text that you can never quite remember how to type.

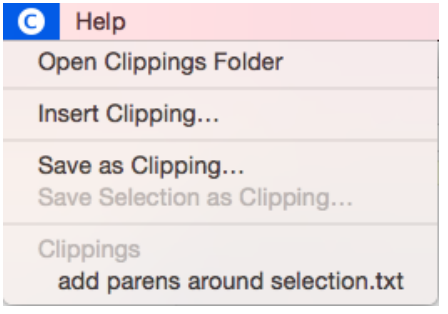
In this chapter

About Clippings

Clippings provide you an easy way to store and enter any sort of frequently used text, including program code, HTML markup, or just about anything else. Clippings can be language-sensitive, and their optional ability to run scripts and insert the results, further extends their flexibility and usefulness.

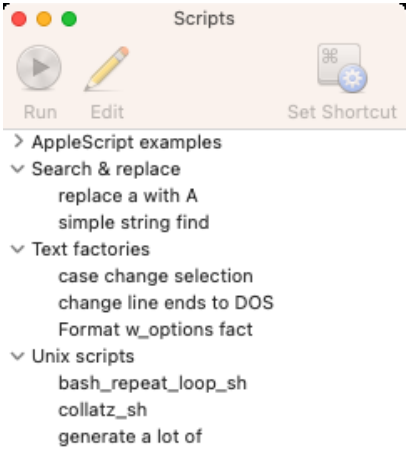
The Clippings Menu

The Clippings menu contains commands which you can use to insert and manage text clippings. The Clippings menu also presents the contents of all available clipping sets. You can choose any available clipping to insert its contents into the active document, or use the Insert Clipping command**. (**See “Inserting Clippings” on page 334.)



The Clippings Palette

Choosing the Clippings command from the Palettes submenu of the Window menu opens the Clippings palette, shown below. This window hierarchically displays the contents of all clipping sets available for the frontmost document Clipping item names that are too long to fit within the width of the window are truncated with ellipses (…).



“Hovering” the mouse over such a truncated name displays a tool tip showing the full name. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Names that fit entirely within the window without truncation do not display a tool tip.

Managing Clipping Sets

Although BBEdit does not install any clipping sets by default, you can find a variety of customer-contributed clipping sets on our website:

`https://www.barebones.com/support/bbedit/clippings_library.html`

or create your own.

Installing New Clipping Sets

To install a new clipping set, just copy its folder into the “Clippings” folder within your local BBEdit application support folder.

Language Sensitivity of Clipping Sets

IMPORTANT BBEdit no longer limits clipping use to the items contained within a single “active clipping set”; instead, all available clippings are available at all times, **unless** a particular clipping set’s name maps to an installed language. In that case, the clippings from that set are available only when the effective language in the active document matches the clipping’s language. (For example, clippings from a clipping set “JavaScript.js” will by default only be available within JavaScript documents or content areas.)

You can override this default behavior by manually enabling clipping sets for any desired set of languages via the Clippings pane of the Setup window. Select one or more clipping sets, and click “Edit Enabled Languages” (or double-click the selected items) to edit the languages for which the set(s) are to be enabled. Within the “Edit Enabled Languages” panel, you can select multiple languages and turn them on or off at once.

The “Universal Items” clipping set no longer enjoys special behavior; instead, like all other clipping sets, it is automatically enabled for all languages by default, while any clippings that you place loose in the top level of the Clippings folder will automatically be available at all times.

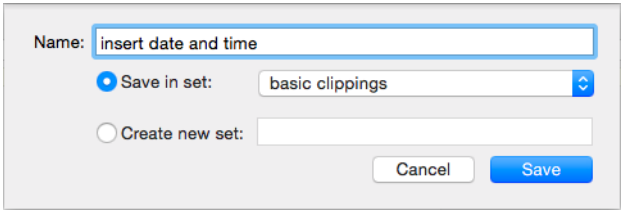
These clipping selection and activation rules are intended to provide maximum flexibility while automatically doing the right thing as often as possible.

Manually Sorting Clipping Sets

By default, the Clippings menu displays clipping sets and clipping items in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name: for example “00)Web template” would sort before “01)HTML Template”. The first three characters of such names are not displayed in the menu. You can also insert a divider by including an empty folder whose name ends with the string “`-***`”. (You can use anything you want for the rest of the name, to make it appear where you want it in the menu.)

Creating and Editing Clippings

You can create a clipping by typing or pasting any desired text, or text and substitution placeholders, into a BBEdit document window and then choosing Save As Clipping from the Clippings menu. BBEdit will display a sheet in which you can name your clipping, and assign it to any existing or new clipping set.



You can also create a clipping from the current selection by choosing Save Selection as Clipping from the Clippings menu, and using the clipping creation sheet as described above. Using this command does not affect the name or location of the document from which you create the clipping.

If you wish to further organize clippings within a set, choose Open Clippings Folder from the Clippings menu. You can create multiple levels of subfolders inside the Clippings folder, to better organize different types of content. The first level of such subfolders appear in the Set popup menu of the Clippings palette, allowing you to reveal only the group of clippings you wish to work with at a given time. (Any clippings not placed in a subfolder are always shown in the Clippings palette.)

You can edit a clipping by selecting it in the Clippings palette and then clicking the “Edit...” button, or by opening the clipping file directly from its location within the “Clippings” subfolder of BBEdit’s application support folder.

You can also hold down the Shift key while selecting a folder node within the Clippings menu to open that folder in the Finder.

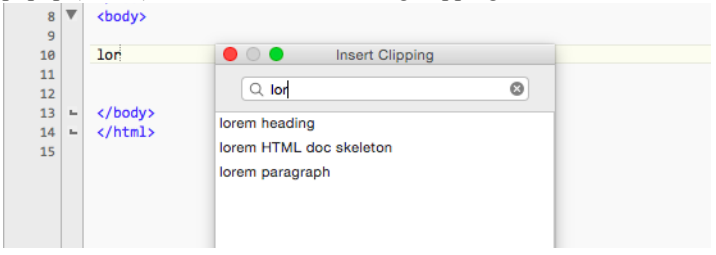
Inserting Clippings

The quickest way to insert clippings is to use BBEdit’s text completion feature. Just type a clipping item’s name (or the beginning of a name), then either pause or invoke the Complete command by pressing F5 (or choosing it in the Edit menu) and BBEdit will display any matching clippings, as well as other available completions, in the completion popup.

You can select a clipping in the popup by using the up- and down-arrow keys and type Return or Tab to insert that clipping, replacing the word or partial word.

You can also select and insert a clipping by choosing the Insert Clipping command from the Clippings menu. You can either select a clipping by navigating the list with the up and down arrow keys, or type in a complete or partial string to filter the list of available clippings.

- If there is a single match, BBEdit replaces the word with the contents of the matched clipping.
- If there are multiple matches, BBEdit brings up the Insert Clipping popup (below) and lists all the matching clippings.



  You can continue typing to further narrow the list and select a clipping, or use the up- and down-arrow keys to select a clipping and type Return to insert that clipping. You can also insert any listed clipping by double-clicking its name in the panel.

- If there is no match, BBEdit brings up the Insert Clipping panel with focus in the filter field, but does not filter the clipping list.

  You can type in the filter field to narrow the list and select a clipping, or use the up- and down-arrow keys to select a clipping, and then type Return to insert the selected clipping. You can also insert any listed clipping by double-clicking it. BBEdit replaces the word with the contents of the clipping and closes the panel.

You can also use wildcards with the Insert Clippings palette's search box. (The palette's interpretation of the pattern is strict; "ab*" will only match clippings whose names begin with "ab", whereas a non-wildcard "ab" will match any clipping whose name contains "ab".

Typing Shift-Return and Option-Return in the Insert Clippings palette's search field will behave as the same modifiers do when choosing an item from the Clippings menu: Shift-Return will reveal the clipping file in the Finder, while Option-Return will open it for editing.

Note: If there is a word or partial word before the insertion point, BBEdit looks for a clipping of the form 'clipping name **begins with** partial word'. However, when there is no word or partial word, BBEdit filters items in the Insert Clipping panel based on whether their names **contain** the currently-entered string. This makes it easier to filter clipping sets which contain many items having common prefixes.

You can use the Clippings menu to insert any listed clipping at the insertion point, or in place of the current selection, by choosing its name in the menu.

You can use the Clippings palette to insert any listed clipping by double-clicking its name in the window. Alternatively, you can click a clipping's name to select it and then click the Insert button, or drag the clipping directly to the desired location in a document window or text field.

When you insert a clipping, BBEdit always reads the clipping file from disk—if a clipping's file is open and has unsaved changes, those changes will not be used.

Assigning Key Equivalents to Clippings

The Set Shortcut button in the Clippings palette (see page 332) lets you assign key equivalents for easy access to frequently used clippings. To assign a key to a clipping:

1. **Select the desired clipping in the Clippings palette.**
2. **Click the Set Shortcut button to activate key entry mode**

   A highlighted rectangular area will become active to the right of the desired clipping.
3. **Type the desired key equivalent.**

   You can use any combination of the Command, Shift, Option, and Control keys in the key equivalent, provided that it must include at least the Command or Control key to be valid. You can also use function keys, with or without additional modifiers.
4. **Click anywhere within the Clippings palette to end key input.**

Note If you try to assign a key equivalent that is already used elsewhere, BBEdit warns you that there is a conflict and asks you whether you want to reassign that key equivalent to the new item.

To remove a key equivalent from a clipping:

1. **Select the clipping in the Clippings palette.**
2. **Click the Set Shortcut button to activate key entry mode.**
3. **Type the Delete key.**
4. **Click anywhere within the Clippings palette to end key input.**

Clipping Substitution Placeholders

When you insert a clipping containing a placeholder into an editing window, BBEdit replaces the placeholder with appropriate substitution text. This is similar to the operation of BBEdit's HTML Templates and Update features. The following table shows the placeholders you can use in a clipping:

Placeholder	Replaced by…
#BASENAME#	The name of the file stripped of its rightmost period-delimited portion. For example, if the file is named "test.html", the base name is "test", while if the file is named "test.foo.html", the base name is "test.foo".
#BLOCK#	Inclusion of this placeholder guarantees that the inserted text will begin and end with a line break.
#CLIPBOARD#	Contents of the current clipboard
#DATE#	Current date, formatted according to your Format settings in the International panel of the System Settings
#DATETIMEXXX#	Inserts a localized, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below)
#DATETIME_GMT XXX#	Inserts the universal, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below)
#FILE#	File name of the document into which the item is inserted
#FILE_EXTENSION#	The filename extension for the file (determined as the rightmost period-delimited portion of the filename, without the period). For example, whether the file is named "test.html" or "test.foo.html", the filename extension is "html"
#FULLPATH#	If the current document exists on disk, this item will insert its fully qualified path
#FUNCTION#	If the item is being inserted into a source file, the name of the current function
#GMTIME YYY#	The current GMT time formatted according to the parameters YYY (see "Time Formats" below)
#INDENT#	When used in a clipping with multiple lines, causes every line after the first to be indented to the same whitespace level as the line in which the item was inserted (see the supplied WML clippings for examples)
#INLINE#	Strips all trailing vertical white space from the item before insertion

Placeholder	Replaced by…
#INSERTION#	Marks the place where BBEdit will place the insertion point after inserting the item; if multiple #INSERTION# placeholders are used, the second and subsequent occurrences are replaced with a placeholder "<##>", which can be used with Go to Next/Previous Placeholder in the Go menu
#LIPSUM [options]#	Lipsum text generated according to the options provided (see "Lipsum Options" below)
#LOCALTIME YYY#	The current local time formatted according to the parameters YYY (see "Time Formats" below)
#LOCALE#	This is the "short" locale code corresponding to the "Language" option in the New HTML Document dialog box, e.g. 'en', 'de', 'x-klingon', and the like.
#NAME#	The long name of the active user account. (There is no intrinsic placeholder for the short name, but you can use #inline##system whoami# to obtain it.)
#PATH#	If the current document exists on disk, this item will insert either its tilde-abbreviated path (for files within your home directory) or its fully qualified path
#PLACEHOLDERSTART #label#PLACEHOLDER END#	Inserts a placeholder "hop" point which you can go to by using Next/Previous Placeholder.
#SCRIPT filename#	Result of running the specified AppleScript
#SELECT#	Selected text
#SELECTIONORINSERTION#	If there was a selection when the clipping was expanded, it will be put at this position; otherwise, the insertion point will remain here.
#SELECTIONORPLACEHOLDER label#	If there is an active selection, the placeholder will be replaced with the selected text. If there is no selection, a placeholder named with the specified text ("label") will be inserted into the document. This placeholder is particularly useful when building clippings for insertion via both BBEdit's auto-completion mechanism and the clippings palette (or direct key equivalent).
#SELSTART# and #SELEND#	Mark a range within the inserted material to be selected after the insertion. You can use multiple pairs of these placeholders within a single clipping.
#SYSTEM shell_script#	Given the full path to a shell command or script, BBEdit will run that command or script and insert the result.

Placeholder	Replaced by…
#TIME#	Current time, formatted according to your Format settings in the International panel of the System Settings
#UUID#	A 128-bit UUID (universally unique identifier), formed by combining a value unique to the computer on which it was generated (usually the Ethernet hardware address) with a value representing the number of 100-nanosecond intervals since October 15, 1582

Placeholders are not case-sensitive. If you want to include a literal placeholder in a clipping, escape the first # with a backslash, as in \#DATE#.

Note BBEdit no longer supports and will ignore the old expert settings key "ClippingsIgnoreTrailingReturns". Instead, use the #INLINE placeholder to ensure that BBEdit ignores any trailing line breaks within a clipping item.

Selection and Insertion Placeholders

You can use multiple #SELSTART#/#SELEND# pairs together with any number of #INSERTION# placeholders.

Example:

Suppose you have defined the following clipping which contains an insertion placeholder:

```
typedef struct #SELECT#

{

#INSERTION#

} #SELECT#, *#SELECT#Ptr, **#SELECT#Handle;
```

If the selected text in your editing window is "MyStruct" and you insert this clipping, BBEdit will insert the following in the editing window:

```
typedef struct MyStruct

{

|

} MyStruct, * MyStruct Ptr, ** MyStruct Handle;
```

(where the vertical bar marks the position of the insertion point).

Example:

Suppose you have defined the following clipping which contains multiple pairs of selection placeholders:

```
MyFancyFunction(#selstart#arg1#selend#,
  #selstart#arg2#selend#, #selstart#arg3#selend#);
```

When you insert this clipping, BBEdit will place the following text in the editing window:

```
MyFancyFunction(arg1, <#arg2#>, <#arg3#>);
```

and the string “arg1” will be selected. You can then use the Go To Next/Previous Placeholder commands from the Go menu to hop to the other arguments and enter the desired values.

Jump Placeholder Format

When you apply a clippings item that contains multiple #INSERTION# cookies, the second and subsequent cookies are replaced with special jump placeholder strings. These strings have the form “<#...#>” where the content “...” between the two # signs is either alphanumeric text, or empty.

You can also directly create and insert jump placeholders at any desired points within a document.

Older versions of BBEdit generated temporary placeholders of the form “#•#” for clippings containing multiple instances of #INSERTION#. If you have any existing clippings which directly employ the old placeholder format, you will need to modify them to use the supported placeholder format.

In addition to jump placeholders, you can also insert “optional” placeholders of the form <#?#>. When the “Go to Next Placeholder” command would select such a placeholder, BBEdit will place the insertion point at the specified position and remove the optional placeholder.

Optional-Argument Placeholder Format

Optional-argument placeholders have the form “<#* ... #>” where the content “...” between the two # signs is either alphanumeric text, or empty. These placeholders may be used to represent optional arguments generated during completion. You can select such placeholders in the usual fashion, but additionally, if you delete a selected optional-argument placeholder with the Backspace key, BBEdit will also delete any leading whitespace back to a preceding comma (if there is one).

Date Formats

The #DATETIME XXX# and #DATETIME_GMT XXX# placeholders allow you to insert the corresponding date and time values with flexible formatting. In order to use these placeholders, you must substitute XXX with an ICU date/time format string. ICU is the mechanism used by macOS for date formatting. For full details, please refer to the section “Formatting Dates and Times” in the ICU documentation:

```
https://unicode-org.github.io/icu/
```

Examples:

#DATETIME EEE, MMM d, yy 'at' h:mm a#

produces:

Tue, Jul 3, 18 at 5:48 PM

#DATETIME_GMT EEE, MMM d, yy 'at' h:mm a#

produces:

Tue, Jul 3, 18 at 9:49 PM

#DATETIME EEEE 'at' h 'o''clock' a#

produces:

Tuesday at 5 o'clock PM

Time Formats

The #GMTIME YYY# and #LOCALTIME YYY# placeholders offer you the option to insert the specified time value with flexible formatting.

In order to use these placeholders, you must substitute YYY with a time format using the same expansion options offered by the ‘strftime’ routine (see ‘man strftime’ for further details).

Examples:

#LOCALTIME %r %z on %A# produces: 06:50:13 PM -0400 on Monday

#GMTIME %r %z# produces: 10:50:13 PM +0000

Lipsum Options

The #LIPSUM [options]# placeholder offers you the ability to insert generated lipsum text in flexible quantities.

The option syntax is of this form:

#LIPSUM [wordlist] [units] [units-count] [max-line-length]#

where “wordlist” and “units” are each a single character, and case matters:

- “p’ for paragraphs
- “s” for sentences
- “w” for words
- “S” for startup
- “B” for bacon
- “N” for normal

The case sensitivity allows these to appear in any order, and they can be separated by spaces, commas, semicolons, periods, slashes, colons, or dashes. Thus, the following examples are equivalent and all will generate ten sentences of Startup lipsum with the lines wrapped to 72 characters:

#LIPSUMS,s 10 72#

#LIPSUM Ss 10 72#

#LIPSUM s:S 10 72#

The “wordlist” specifier is optional; if it is not present then BBEdit will generate “normal” lipsum.

A #lipsum#` placeholder may contain zero, one, or two numbers. If no numbers are provided then there is no line breaking and only one unit (word, line, paragraph) is generated.

If one number is provided, then that number of units is generated.

If two numbers are provided, the first is the number of units to be generated, and the second is the maximum line length.

Thus, if you wish to break lines, you must also specify the number of units that should be generated.

SThe #script filename# placeholder is a powerful option which allows you to insert variable or conditional content from a clipping, by invoking any compiled AppleScript or Unix shell script.

The script itself can either be located in the same folder as the clipping that invokes it (in which case you need only specify its name, such as “MyDateScript”) or you can supply a full pathname to a script on any mounted volume in either POSIX or classic Mac format. An instance of a placeholder referencing the former would be

```
#script /Users/me/example/folder/Script.scpt#
```

and an instance referencing the latter would be:

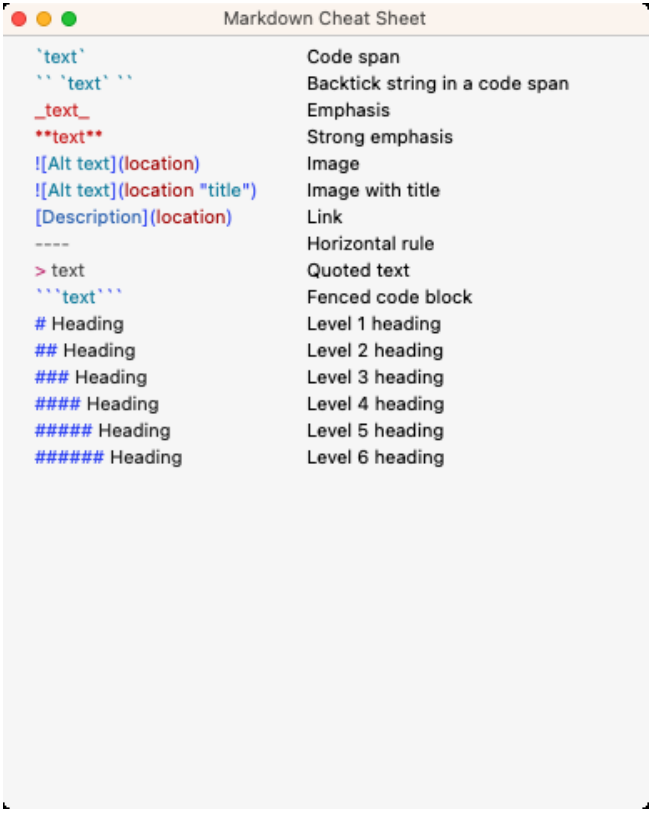
```
#script Hard Drive:Users:me:example:folder:Script.scpt#
```

The script must return a text string (or a value that can be coerced to a string). This result string can itself contain additional clippings placeholders, which will be interpreted before the item is inserted in the current document.

WARNING Note that this makes it possible for one script to invoke another. You must take care to not create a script execution loop, which could hang the application!

About Cheat Sheets

Cheat sheets are based on clipping placeholders, and intended to serve as quick "training wheels" references for pieces of text that you can never quite remember how to type or are overly complicated to type. The built-in examples are for Markdown and Clippings placeholders but of course you can create your own cheat sheets.



To make a cheat sheet available, place it in the "Cheat Sheets" subfolder of BBEdit's application support folder.

To use a cheat sheet, choose it in the "Cheat Sheets" submenu of the Window menu. This will cause BBEdit to open a floating window which displays the contents of the cheat sheet. The left-hand column shows the text of the "cheat", and the right-hand column shows a brief description.

Double-clicking on the item in the cheat sheet will insert it into the active text document, if there is one. You can also drag the item from the cheat sheet to wherever you'd like to insert it.

When inserting the item from the cheat sheet, BBEdit will expand any clippings placeholders present in the text, if the cheat sheet author specified that this should be done. Otherwise, the text is inserted literally. (For example, items in the Clippings cheat sheet itself are inserted literally, to speed the process of authoring clippings.)

Cheat sheets **may** display a preview of the text that will be inserted, along with the full text of the description (which is helpful if it is long). The Clippings cheat sheet does this as well.

Cheat sheets can now specify a policy for handling of backslash character escapes in their contents. This is done by adding an “Options” dictionary, containing a single key: ‘backslashEscapePolicy’. The value of `backslashEscapePolicy` is an integer, with one of three supported values: `0` for default processing (applies if absent) which will process only backslash escapes corresponding to control characters or hex escapes; `1` to process all backslash escape sequences including `\\` for a literal backslash; or `2` to suppress all backslash escape processing and treat the text literally.

An example usage to suppress processing:

Options": { "backslashEscapePolicy": 2 }

The `Options` dictionary may be present at the top level of the cheat sheet, **or** within an individual entry in the `Hints` array.

If both are present, any conflicting settings are resolved in favor of the individual entry.

CHAPTER

13 Scripting BBEdit

BBEdit offers access to nearly all of its features and commands via AppleScript. This chapter provides a brief overview of AppleScript, discusses BBEdit's scripting model, and explains how you can use scripts within BBEdit.

An excellent way to learn how to script BBEdit is to look at the scripts others have written for it, or to turn on recording in your script editor while you perform actions in BBEdit. The BBEdit Talk discussion group is also a good resource for learning more about scripting.

```
https://groups.google.com/g/bbedit
```

IMPORTANT Regardless of whether you are new to scripting BBEdit or are familiar with scripting previous versions, we strongly recommend that you carefully review the sections "BBEdit and AppleScript" and "Working with Scripts" in this chapter.

In this chapter

AppleScript Overview

If you are familiar with AppleScript, you should have little difficulty scripting BBEdit. It has a robust and highly flexible object model. If you do not know much about scripting, though, read on for an introduction to the necessary concepts.

About AppleScript

AppleScript is an English-like language which you can use to write scripts that automate the actions of applications, and exchange data between applications. Although AppleScripts can manipulate applications' user interfaces by taking advantage of the system's GUI Scripting capability, this is not their primary function. Rather, scripts talk directly to a application's internals, bypassing its user interface and interacting directly with its data and capabilities.

If you want to insert some text into a document, emulating a user typing into an editing window is not the most efficient way of accomplishing this. With AppleScript, you just tell the application to insert the text directly. If you want the application to save the frontmost document, you need not mime choosing Save from the File menu, but rather just tell the application to save its frontmost document.

Note AppleScript is actually a specific language which resides atop the general Open Scripting Architecture (OSA) provided by macOS. Although AppleScript is by far the most common OSA language, there are others, including a JavaScript variant. All OSA languages are capable of accomplishing similar things, although the actual commands used differ from one language to the next. In this chapter, we will focus exclusively on AppleScript, since it is the standard scripting language, but you should bear in mind that there are other options.

Scriptable Applications and Apple Events

Since AppleScripts must have direct access to an application's internal data structures, any application that will be used in an AppleScript must be designed to allow this access. We say such applications are *scriptable*. BBEdit is scriptable, as are many, many other programs. However, it is important to note that not *every* application is scriptable, and AppleScripts are not the best solution for automating applications that are not.

What goes on in an application that is scriptable? The foundation of AppleScript is something called the *Apple Event*. Macintosh applications are designed around an event loop; they go around in circles waiting for you, the esteemed user, to do something (choose a menu command, press some keys, and so on). These actions are passed to the application by the operating system in the form of an *event*. The application decodes the event to figure out what you did, and then performs an appropriate operation. After an event has been handled, the application goes back to waiting for another one. (At this point, the Mac OS may decide to give some time to another application on your computer.)

Apple Events are special events that applications send to each other, enabling a feature called *inter-application communication* (IAC). (It's a mouthful, but it just means applications can talk to each other.) Apple Events are also the way AppleScripts tell applications what to do, and which data to retrieve. So to be scriptable, an application must first support Apple Events.

Apple Events in their naked form are raw and cryptic things—bits of hieroglyphics only a programmer could love. So a scriptable application also has a *scripting dictionary*. The scripting dictionary tells any application that lets you write AppleScripts, such as the standard Script Editor, the English-like equivalent for each Apple Event and each event's parameters.

It is important to note that because Apple Events were originally designed to allow applications to communicate with each other, AppleScripts automatically inherit the ability to talk to more than one application. It is common in the publishing industry, for instance, to write scripts that obtain product information from a FileMaker Pro database and insert it into an InDesign file. This integration is one of the Macintosh's primary strengths.

You use AppleScript's *tell* verb to indicate which application you are talking to. If you are only sending one command, you can write it on one line, like this:

```
tell application "BBEdit" to count text documents
```

If you are sending several commands to the same application, it is more convenient to write it this way:

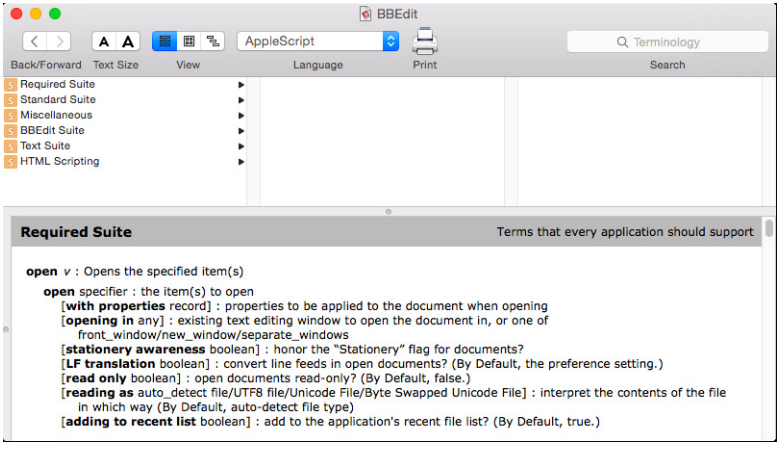
```
tell application "BBEdit"
    count text documents
    repeat with x from 1 to the result
        save text document x
    end repeat
end tell
```

The Script Editor automatically indents the lines inside the *tell* block for you so you can more easily follow the organization of the script.

Reading an AppleScript Dictionary

To display an application's AppleScript dictionary, you can simply drag that application onto the Script Editor icon, or use the Script Editor's Open Dictionary command. As we noted earlier, all scriptable applications include a dictionary that tells AppleScript how to convert English-like commands into the Apple Events actually expected by the application. The Script Editor uses this same information to display a sort of "vocabulary guide" that helps you write your scripts.

We will naturally use BBEdit's dictionary, shown below, to illustrate how to read a dictionary.



(You will probably want to make the window bigger if you have room on your screen.)

Down the left side is a list of every event and object supported by the application. An event is a verb—it tells the application what to do. A class is a noun: a piece of data, or a structured collection of data, inside the program. In BBEdit, for instance, classes are things like files, windows, the clipboard, browsers, and so on.

Suites

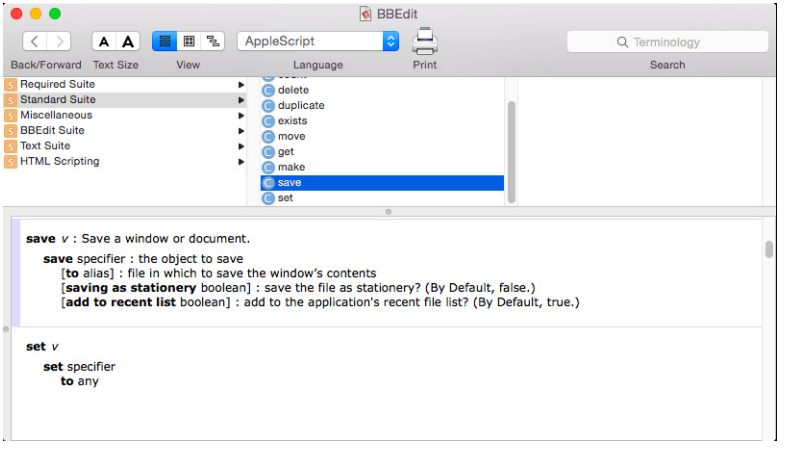
The first thing you will notice is that the events and classes are divided into *suites.* A suite is just a collection of related events and classes. Apple, for instance, has decreed that all applications should support particular events, which together are called the Required Suite. Another Apple-defined suite is the Standard Suite: if an application offers certain functions which Apple considers to be common, it should use these standard terms, so that scripters do not need to learn a new term for each application they work with. After that, it is a free-for-all—each developer is free to organize their events and classes however they think best.

In addition to the Required and Standard suites, BBEdit has a Miscellaneous suite, a BBEdit Suite, a Text suite, and an HTML Scripting suite.

Within each suite, events—verbs—are displayed in normal text, while classes—nouns—are italicized. Most commands sent to BBEdit will start with one of the verbs. (In some cases, *get* might be implied.)

Events

Let's look more closely at one of the events—*Save* is a good one to start with. It is shown below.



The right side of the window shows the syntax of the selected event, as well as a brief description of its function. The boldface words are keywords; they must be included exactly as shown or the script will not compile. The normal text tells you what kind of information goes after each keyword. For example, after *save* you must give a reference; the italicized comment next to that line indicates that it is a reference to the window to be saved. In other words, some window object, which in BBEdit would be *window 1* for the frontmost window, or *window "Text File"* if you want to specify a window by name. (we will show you how to figure all that out in a moment—you have to look at the window class's dictionary entry.)

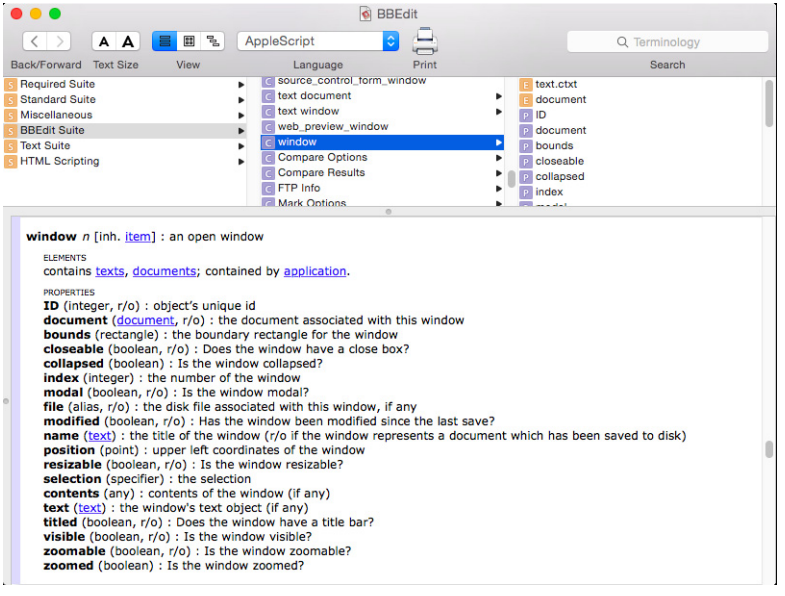
Anything in square brackets is optional. Most of the rest of the *save* event is optional, in fact. The basic event just saves the frontmost window to the same file from which it was opened. However, you can also optionally include the word *to* followed by a file reference. (You specify a file simply by using the word *file* followed by the path name of the file, as in *file "Hard Disk:Users:BBSW:Documents:My file".)* If you specify a file to save the window to, the text will be saved into that file instead of the file it came from—like using Save As instead of Save.

The last three optional parts of the *save* event are denoted as boolean. That means they take either a true or a false value. In AppleScript, there are a couple of different ways to specify boolean values. You can write *saving as stationery true* to tell BBEdit to save the file as a stationery document. Or you can write *with saving as stationery.* You will notice that the last two parameters default to true if you do not specify them as false. To do that, you would use *add to recent list false* or *without add to recent list*. Whichever way you write it, you will notice that when you compile the script, AppleScript rewrites it using "with" or "without". Since that is the syntax AppleScript seems to like best, that is probably the one you should get used to thinking in.

Let's take a look at another one: the prosaic *get.* Select *get* from BBEdit's dictionary listing and take a quick look at its class definition.You use *get* to retrieve information from an application. You must specify a reference to the object you want to retrieve, and you can specify a *coercion*—a condition that tells AppleScript to treat one type of data as if it were another—by adding the *as* clause. However, after that is the *Result:* line, which we have not seen before. This line tells you what type of value the command returns. (This value is placed in the AppleScript system variable called *the result.) Get* can retrieve any kind of object, so it can return anything, as indicated here. Other events might return a specific type of result, or none at all. *(Save* did not have a *Result:* line in its dictionary entry, which means it does not return a result.)

Classes and the Class Hierarchy

Let's look at a typical class definition: *window* will do nicely. It is in the BBEdit Suite, toward the bottom.



All windows in BBEdit belong to this class. A class defines a particular kind of object; a particular example of an object belonging to the class is said to be an instance of that class, or just an object of that class. So here we are looking at the class itself; each individual window object has all these properties.

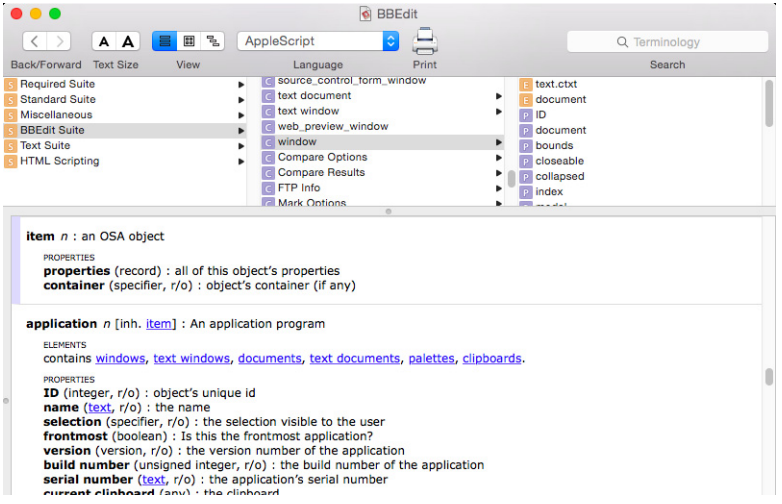
After a tag line that tells you about the class ("an open window") comes the plural form. AppleScript lets you refer to windows either singly or as a group, so it needs to know what the plural of every term is. For example, try this little script:

```
tell application "BBEdit" to count windows
```

The result of this script is the total number of window objects currently displayed by BBEdit.

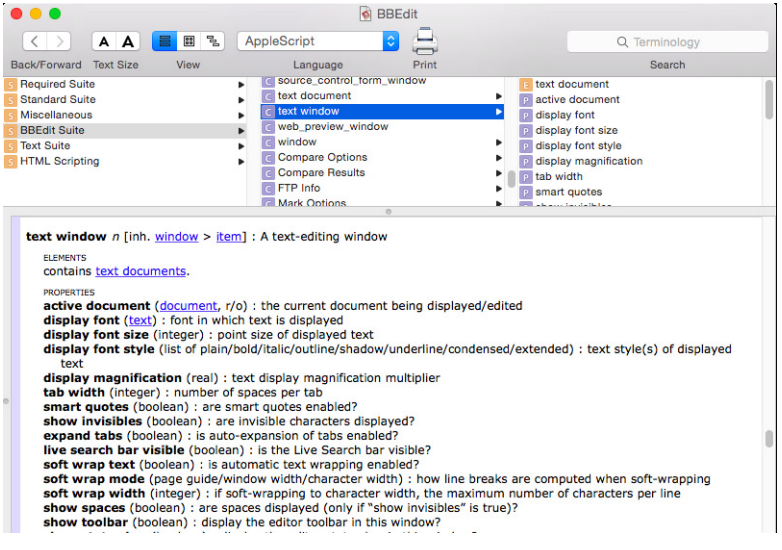
After the plural form comes a list of properties. Some objects do not have properties—for example, a string—but many applications do. An object's properties are merely a collection of data that describes that particular object. For example, as you look down the list of window properties, you will see that every window has a name, every window has a position, every window has bounds (the area of the screen it covers), and so on.

The first item on the list, though, is <*inheritance*> *item*. This tells you that a window is a kind of item, and that it therefore has all the properties of an item. Take a quick look at *item's* class definition, shown below.



You will see three properties: *properties, ID*, and *container*. The first entry *properties* is a record containing all the object's properties. In other words, because a window is an item, it has, in addition to all its listed properties, another property which returns all the other properties as a record—a single piece of data that can be stored in a variable. Every class in BBEdit is part of a hierarchy with the *item* class at the top, so every object in BBEdit "inherits" the *properties* property. This catch-all property can be handy for making exact duplicates of objects, among other uses.

You may realize that BBEdit has several kinds of windows; you can see their classes listed in the dictionary: differences window, disk browser window, project window, text window, tool window, and the like. Let's look at *text window:*



You can see that a text window inherits all the properties of the *window* class. And, since the *window* class inherits all the properties of the *item* class, this means that the *text window* class also has the *properties* property defined by the *item* class.

To make explicit what you might have already gathered, classes in AppleScript form a hierarchy. That is, classes can be based on other classes. Such a class is called a *subclass,* and the class on which a subclass is based is referred to as its *parent class.* (In AppleScript, classes can only have one parent. Multiple inheritance is a feature found in more complex languages.)

The idea of a class hierarchy makes it easier for us to add new features to BBEdit, since when we want to create a new kind of window, half the work is already done. However, when scripting, you may need to flip back and forth between two or more class definitions to find all the properties of the object you are working with. (This is, technically speaking, a limitation of Apple's Script Editor. There is no reason the inherited properties could not automatically be included in a subclass listing by a smarter editor, for example, Script Debugger, which does this.)

Now that we have the class hierarchy under control, let's look at the properties themselves more closely. we will stick with the *text window* class at this point.

Properties of an object are referred to using the preposition *of.* For example, the following line of script returns the font of the frontmost text window.

```
tell application "BBEdit" to get display font of
text window 1
```

Note In this specific example, you can just write *get display font of window 1*. AppleScript will figure out that window 1 is more specifically a text window, and therefore has a *display font* property, even though the generic *window* class does not have any such property. All the properties of the object are available even if you did not use its specific class name. However, in most cases, you should specify exactly the object you want; this distinction is especially important when dealing with text documents (content) versus text windows (display elements).

You can set the properties using the *set* event, like so:

```
tell application "BBEdit" to set display font of text
window 1 to "Courier New"
```

Let's go back to the *window* class for a moment. Most of the properties of this class are marked with the abbreviation *[r/o].* That stands for Read-Only. In other words, you can only *get* these properties, not *set* them.

Recordable Applications

Once an application accepts Apple Events, it actually makes a good deal of sense for an application to be designed in two parts: the user interface that you see, and the "engine" that does all the work. (An application designed this way is sometimes said to be *factored*.) The user interface then communicates with the engine via Apple Events.

The design of the Apple Event system makes it possible to "record" events into a script. This feature not only lets you automate frequently performed tasks with little hassle, it also can be an enormous aid in writing larger and more complicated scripts, because the application tells you what events and objects to use for the kind of task you record.

Because of the important recording functionality they enable, applications that have been factored and use Apple Events to let the two halves communicate are said to be *recordable*. It is important to note that not all scriptable applications are recordable.

Saving Scripts

Any AppleScript can be saved in what's called a *compiled script file*. A compiled script file contains the actual Apple Events; by generating these events when you save the file, the operating system does not have to convert your English-like commands into events each time you run the script, which means it loads faster. When double-clicked in the Finder, a compiled script file automatically opens in the Script Editor, where it can be run. A script can also be saved as a stand-alone application, or *applet*, in which case double-clicking the script's Finder icon automatically runs the script. Both types of files can be saved with or without the English-like *source code;* if you save it without the source code, other users you give the script to will not be able to make any changes to it (of course, you should also keep a copy of the script *with* the source for yourself).

Using Scripts with Applications

Although you can place a script applet in the global Scripts menu, or in any folder, and use it any time you need it, many applications (including BBEdit) provide a special menu that lets you launch compiled scripts intended specifically for use with that one application. Since you do not have to save them as applets, they take up less disk space and launch more quickly. They also show up only in the application you use them with, rather than cluttering your global Scripts menu.

Some applications go even further, allowing you to define scripts to be run when certain things happen in the program. For example, an application might let you define a script to be executed when the user chooses *any* menu item. The script might then perform some pre-processing, and then exit by telling the application whether to continue with the menu command or to cancel it. As a simple example, a script might check to see what printer is selected when the user chooses the Print command. If it is the expensive color dye-sublimation printer, on which printing a page costs several dollars, the script could remind the user of that fact and confirm their intention (through an alert) before continuing with the print operation.

An application that supports such a feature (or any method of integrating user-written scripts seamlessly into its user interface) is said to be *attachable,* because the scripts become "attached" to the features of the program. (BBEdit is attachable; more details about using this feature are provided later in this chapter.)

Scripting Resources

Covering all the details you might need to write your own AppleScripts is not something we can reasonably do in this manual. AppleScript, despite its deceptively simple English-like syntax, is a sophisticated object-oriented language with many subtleties. For this reason, we suggest you consult supplemental documentation and resources if you are a beginning scripter.

A good place to start is with someone else's script: find a script that does *almost* what you want it to and repurpose it. Even if you cannot find a script that does anything close to what you want, reading others' scripts is a good way to learn how AppleScript "thinks" and how BBEdit's particular AppleScript implementation behaves.

In addition to the basic AppleScript documentation included with the system, you may find the following resources useful in your quest to understand scripting.

Books

AppleScript: The Definitive Guide (Second Edition), Matt Neuberg. O'Reilly and Associates, 2006. ISBN: 0-596-10211-9

Discussion Groups

BBEdit Talk

`https://groups.google.com/g/bbedit`
The BBEdit Talk discussion group is an excellent place to ask BBEdit-specific scripting questions.

Mac Scripting

`http://listserv.dartmouth.edu/scripts/wa.exe?A0=MACSCRPT`
Unofficial list covers AppleScript and other Macintosh scripting languages, with occasional forays into peripheral topics.

Websites

AppleScript: The Language of Automation

`http://www.macosxautomation.com/applescript/`

MacScripter.Net

`http://macscripter.net/`
A good selection of AppleScript-related news and topics, including the "AppleScript FAQ" and discussion forums.

ScriptWeb

`http://www.scriptweb.com/`
This site covers all scripting languages, not just AppleScript. Also, it has an extensive directory of scripting additions.

Software

Script Debugger

`https://www.latenightsw.com/`

Despite its name, Script Debugger is more than a debugger; it is actually an enhanced replacement for Apple's Script Editor, featuring variable monitoring, step/trace debugging, an object browser for an application's objects, and much more.

Using AppleScripts in BBEdit

BBEdit has been scriptable for years, and we have continually worked to refine its level of scripting support. In addition to providing extensive script access to its commands and data, BBEdit is both attachable *and* recordable.

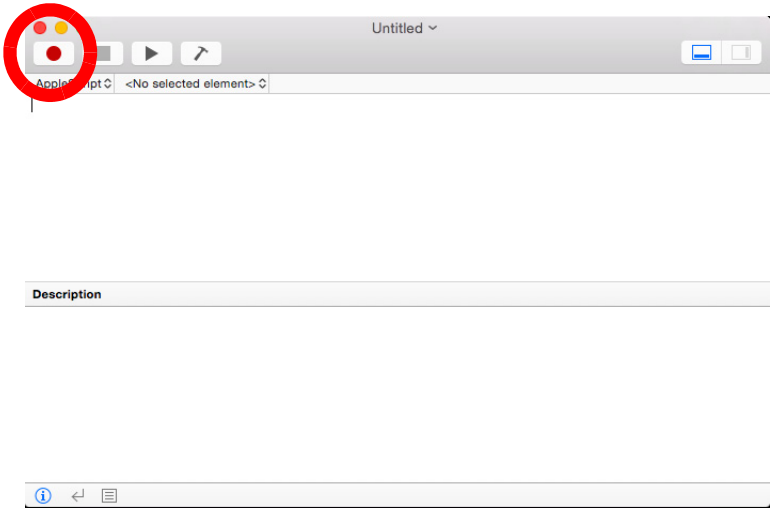
This section describes how you can create and employ AppleScripts within BBEdit via recording and BBEdit's various scripting facilities, while the following section covers BBEdit's scripting commands and other issues related to preparing scripts for use.

Recording Actions within BBEdit

Any language is easier to read than to write, easier to understand than to speak. AppleScript is no different. That's because, even though all the commands it uses are English words arranged in ways that more or less make grammatical sense, you still have to know (or find out from the application's dictionary) exactly which words to use, and what order they should go in. But it is easy to get started making scripts by recording them.

First, launch both BBEdit and the Script Editor.

When you launch the Script Editor, a new, blank script window appears. Click the Record button, circled in the illustration below.



Now switch to BBEdit and perform your task. Remember that the Script Editor is recording *everything* you do in every recordable application you are running, not just BBEdit. If you do something in the Finder, for instance, that will get recorded too. Since almost everything you do is recorded, remember that if you make an error, and then Undo it, your recorded script will faithfully make the same mistake and undo it when you run it later. It will be possible to fix minor errors later, but things always go more smoothly if you do not make any mistakes, so take your time and try to do it right the first time.

Now switch back to the Script Editor and click the Stop button. After a brief pause, your script is compiled and ready for use. Try clicking the Run button to see it work. (It might not work correctly. If you recorded a search and replace operation changing every "cat" to "dog", you already changed the document while recording the script, and of course the script will not do anything when you run it.)

Finally, save the script in the BBEdit Scripts folder so that it shows up in BBEdit's script menu. Choose Save As from the File menu, and then use the Script Editor's Save dialog to put the script in your BBEdit Scripts folder. Now try selecting it from the script menu in BBEdit.

The Scripts Menu

The Scripts menu (left) in BBEdit's menu bar contains several commands. It also lists all AppleScripts (as well as Automator actions, text factories, and Unix scripts) present in the Scripts folder within BBEdit's application support folder, providing a quick way to access frequently used scripts. You can place scripts within subfolders (up to 4 levels deep) of the Scripts folder to organize them.

Note AppleScripts written for use in as BBEdit filters or scripts should be saved as compiled (data fork) script files, not script applications.

In addition to the list of available scripts, the Scripts menu provides the following commands.

Open Script Editor

Choose this item to switch to the system's default AppleScript editor. If the script editor is not running, BBEdit launches it.

Open Scripting Dictionary

Choose this item to switch to your preferred AppleScript editor and open BBEdit's scripting dictionary for viewing. If the script editor is not running, BBEdit launches it.

Open Scripts Folder

Choose this item to open the Scripts folder which is located within BBEdit's application support folder. (See "Scripts" on page 38.)

Running and Editing Scripts

Choose the item corresponding to any script to run that script. To edit a script, you may either open the script file directly from within the "Scripts" subfolder of BBEdit's application support folder, or select the desired script in the Scripts palette and click the "Edit..." (pencil) button.

The Scripts Palette

The Scripts command, located in the Palettes submenu of the Window menu, opens a palette listing all available scripts. Names that are too long to fit within the width of the window are truncated with ellipses (…).

"Hovering" the mouse over such a truncated name displays a tool tip showing the full name. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Names that fit entirely within the window without truncation do not display a tool tip.

Organizing Scripts

Items in the Scripts menu and Scripts palette are displayed in alphabetical order by default, but you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "00)Save All" would sort before "01)Close All.") For names of this form, the first three characters are not displayed in the window.

You can also insert a divider into the Scripts menu by including an empty folder whose name ends with the string "–***". (The folder can be named anything, so it sorts where you want it.)

Note Any dividers you add will appear in the Scripts menu, but not the Scripts palette.

Attaching Scripts to Menu Items

BBEdit lets you attach scripts to menu items. By this, we mean that you can write scripts that BBEdit automatically calls before or after performing a menu command. For example, if you want BBEdit's Open from FTP/SFTP Server command to launch your favorite FTP client, you can simply attach a script to that menu item. Scripts can return a value that tells BBEdit whether to continue with the command that was selected, or to cancel the operation (in which case only the script is executed).

Scripts attached to BBEdit menu items must be stored in the Menu Scripts folder of BBEdit's application support folder. These files should be compiled scripts, not script applications. Scripts are named to indicate which menu item they go with: first the name of the menu (or the submenu) upon which the item is immediately located, then a bullet "•" (Option-8) character, then the name of the menu item. For example, to attach a script to the Open from FTP/SFTP Server menu item, you would name it "File•Open from FTP/SFTP Server", while to attach a script to the New Document menu item, you would name it "New•Text Document".

Some of BBEdit's menus have icons rather than names. BBEdit uses the following names for its icon menus: "#!" [the 'Shebang' menu], "Compiler", and "Scripts". Furthermore, the New With Stationery submenu is named "Stationery" for purposes of attachability.

When you choose a menu command which has an attached script, BBEdit will pass the menu name and command (item) name to the script's MenuSelect handler, if it has one. If the script contains no MenuSelect handler, BBEdit executes the script's run handler.

The script's MenuSelect handler can tell BBEdit to skip performing the chosen command by returning "true", or have it continue on and perform the command by returning "false". If MenuSelect returns "false", BBEdit will call the script's PostMenuSelect handler, if it has one, after it performs the menu command.

Here is a simple example, which adds a confirmation dialog to the Save command (addressed as "File•Save"). Note that we test the menu and item names to make sure the script is attached to the Save command—if it is attached to some other command, it does nothing.

```
on menuselect(menuName, itemName)
  if menuName = "File" and itemName = "Save" then
    set weHandledCommand to true
    display dialog "Are you sure you want to save?" ¬
    buttons {"No", "Save"} default button 2
    if button returned of the result is "Save" then
    -- the application should do its work
    set weHandledCommand to false
    else
    -- we handled the command, app does no work,
    -- postmenuselect doesn't get called
    display dialog "The document was not saved." ¬
      buttons {"OK"} default button 1
    end if
    return weHandledCommand
  end if
end menuselect

on postmenuselect(menuName, itemName)
  -- this is called after the application has processed
  -- the command
  display dialog "The document was saved." ¬
    buttons {"OK"} default button 1
end postmenuselect
```

Attaching Scripts to Events

IMPORTANT BBEdit offers script attachability, which means you are not limited to menu commands but can attach scripts directly to the app which will take effect whenever selected application and/or document events occur.

To access these events, your attachment scripts must contain function names which correspond to the names of the events' attachment points. Except when otherwise noted, all of the following considerations apply:

- Every function takes a single argument which is a reference to the object in question: the application for application entry points, or the document being opened/closed/saved/etc for document entry points.
- Any function associated with an attachment point whose name contains 'should' must return a Boolean result: 'true' or 'false'. If it returns 'true', the operation will continue. If it returns 'false' or throws an error (see below) then the operation will be cancelled. So, for example, 'applicationShouldQuit' returning 'true' will allow the application to quit; returning 'false' will not.
- If an attachment script causes a scripting error and does not handle it within the script itself, BBEdit will report the error. In the case of functions which are used to allow a 'should' action, this will prevent the action from occurring.

Here are the available attachment points:

Application attachment points

- *applicationDidFinishLaunching*: called when the application has completed startup.
- *applicationShouldQuit*: called when you choose the Quit (or the application receives a 'quit' event for any other reason).
- *applicationDidQuit*: called when the application has finished shutting down and is about to exit.
- *applicationDidSwitchIn*: called when BBEdit has been brought to the foreground.
- *applicationWillSwitchOut*: called when BBEdit is being put into the background. You could use this (for example) to save outstanding changes to the front document.
- *applicationWillChangeWorkspace:* called when BBEdit will be switching to a new workspace.
- *applicationDidChangeWorkspace:*called when BBEdit has just switched to a new workspace.

NOTE In BBEdit 13.1 and later, the attached script will run **before** the app goes into the background, rather than **after** (as in previous versions). You should also avoid using 'show dialog' or similar verbs during *applicationWillSwitchOut*, because that will leave the resulting item on screen until you switch back to BBEdit (and in the event you have also defined an attachment for *applicationDidSwitchIn*, that will likewise run so you'll **really** be in the soup).

Document attachment points

- *documentDidOpen*: called when a document has been opened and is ready for use. (Since BBEdit supports multiple types of documents, your script should allow for the argument to be a document of any type.)
- *documentShouldClose*: called when the application is preparing to close a document.
- *documentDidClose*: called when the application has closed a document.
- *documentShouldSave*: called when the application is trying to determine whether a given document should be saved.
- *documentWillSave*: called when the application is about to begin saving a document. (note that this will only be called after a successful return from a 'documentShouldSave'.
- *documentDidSave*: called after a document has been saved successfully.
- *documentWillUnlock*: called when BBEdit is going to make a document writeable. (For example, when you click the pencil to unlock a document)
- *documentDidUnlock*: called when BBEdit has successfully made a document writeable.
- *documentWillLock*: called when BBEdit is going to make a document read-only.
- *documentDidLock*: called when BBEdit has successfully made a document read-only.
- *documentWillInsertTextForDroppedImageFile:* called when you drop an image file into a BBEdit editing view

Using Attachment Scripts

Scripts attached to events must be stored in the "Attachment Scripts" folder of BBEdit's application support folder (see page 36).

You can write one script to handle each attachment point, or one script to handle the attachment points for an entire class of objects, or one script to handle all of the attachment points for the entire application.

You can also mix and match scripts to meet specialized needs: for instance, by using one script to implement a particular attachment point for documents, and a second script to handle the remaining attachment points.

BBEdit associates scripts to attachment points by means of the script's file name. There are three ways to specify a script's role:

1 **<ObjectClass>.<entryPoint>**

2 **<ObjectClass>**

3 **<ApplicationName>**

The first form is the most specific: the 'ObjectClass' may be either "Application" or "Document", while the 'entryPoint' is one of the attachment points described above appropriate to that object class.

For example, a script which implemented only the *documentDidSave* attachment point should have the name “Document.documentDidSave.scpt” and contain a subroutine named ‘documentDidSave’, thus:

```
on documentDidSave(myDoc)

    -- do something useful and appropriate

end documentDidSave
```

Note Adding the filename suffix ‘.scpt’ is not mandatory, but you should follow the current system conventions suggested when creating scripts with the AppleScript Editor (or any other script editor such as Script Debugger).

The second form allows you to implement all of the attachment points for a single object class in a single script file, if desired.

For example, you could create a script named “Application.scpt” containing subroutines for as many of the application attachment points as you wish:

```
on applicationDidFinishLaunching

    -- do something relevant

end applicationDidFinishLaunching

on applicationShouldQuit

    -- hello world

    return (current date as string contains "day")

end applicationShouldQuit
```

Likewise, to implement all of the attachment points for the Document class, you could create a script named “Document.scpt”, and put subroutines in it for the document attachment points:

```
on documentDidSave

    -- do something relevant

end documentDidSave

    ...

on documentWillClose

    ...

end documentWillClose
```

The third form allows you to write a single all-encompassing script which contains subroutines for all of the attachment points in the application. To do this, name the script "BBEdit.scpt" and include whatever subroutines you wish to implement. For example:

```
on applicationShouldQuit

    -- hello world

    return (current date as string contains "day")

end applicationShouldQuit

on documentWillClose

    ...

end documentWillClose
```

When figuring out which script to run, BBEdit will first look for a script whose name exactly matches the attachment point, e.g. "Document.documentShouldSave.scpt". If there is no such script, BBEdit will then look for a script whose name matches the object class at the attachment point, e.g. "Document.scpt". Finally, if there are no scripts with either an exact or a class match, BBEdit will look for an application-wide script: "BBEdit.scpt".

Note You do **not** have to implement attachment subroutines for all attachment points, or for all classes—only the ones you need. If there is no attachment script or subroutine, BBEdit proceeds normally.

Using an Attachment Script to Perform Authenticated Saves

BBEdit supports a special attachment point for the Document class: *documentShouldFinalizeAuthenticatedSave*. This attachment point will be called whenever an authenticated save is necessary (for text documents only).

The following sample script illustrates how to use this facility (the comments are important, so please read them!):

```
on documentShouldFinalizeAuthenticatedSave(theDocument,
tempFilePath, destinationPath)

    -- on input: tempFilePath points to the contents
    -- of the document written to a temp file, ready
    -- to move to the destination; destinationPath is
    -- where the file should be copied.

    -- on exit: if the operation succeeded, delete the
    -- temp file (or else the application will assume
    -- the operation failed) and return YES for success

    -- this is pretty straightforward:
    --   "cp tmpFilePath destinationPath"

   do shell script "cp" & " " & quoted form of tempFilePath
& " " & quoted form of destinationPath with administrator
privileges

    -- now remove the temp file, this indicates to
    -- the application that we did the work

    do shell script "rm" & " " & quoted form of tempFilePath

    return true

end documentShouldFinalizeAuthenticatedSave
```

Using an Attachment Script to Handle Dropped Images

BBEdit 14 and later offer a new script attachment point, to provide additional control over the text generated when you drop an image file into a BBEdit editing view.

The entry point for the script is '*documentWillInsertTextForDroppedImageFile*' and the parameters are (listed in the order supplied):

- a reference to the BBEdit document on which the image file is being dropped;
- a reference to the image file being dropped;
- the text generated by BBEdit for the drop, if any. (This will always be present, but may be the empty string.

Unlike the other 'will' attachment points, 'documentWillInsertTextForDroppedImageFile' should return a value. This can be one of:

- the proposed text, which inserts what BBEdit would have inserted anyway had this script not been present;

- the 'missing value' pseudo-value, which will abort the drop operation (and insert no text);
- some customized string, generated at the script's sole discretion.

Since BBEdit will wait for this function to return, any work that it does must take minimal time (or be done asynchronously, perhaps within an 'ignoring application responses' block).

Here is a small example script to describe and illustrate. In this case, the script asks Acorn to open the image file being dropped, while otherwise leaving BBEdit's default behavior unaffected.

```
on documentWillInsertTextForDroppedImageFile(theDocument,
theImageFile, proposedText)
 tell application "Acorn" to open theImageFile
 return proposedText
end documentWillInsertTextForDroppedImageFile.
```

Filtering Text with AppleScripts

The Text Filters folder in BBEdit's application support folder contains executable items, such as compiled AppleScripts, Automator workflows, and Unix filters, which you may apply to the active document via the Apply Text Filter submenu of the Text menu.

When you apply such an item, BBEdit will pass either the selected text (or the contents of the active document, if there is no selection) as a reference to a 'RunFromBBEdit' entry point within your AppleScript, and your script should return a string which BBEdit will use to replace the selected text (or the contents of the document). If your script does not contain a 'RunFromBBEdit' entry point, BBEdit will call its run handler, again passing a reference to the current selection range.

BBEdit's Scripting Model

This section provides a high-level overview of BBEdit's scripting model that will, where appropriate, contrast the current scripting framework against older versions of BBEdit, and suggest how you can modify your existing scripts for compatibility.

IMPORTANT Because BBEdit's scripting dictionary changes whenever we add features, it should be considered the definitive reference in any situation where it and this document differ. We have found Script Debugger from Late Night Software to be an excellent tool for browsing and navigating BBEdit's scripting dictionary, as well as for preparing and testing scripts.

```
https://www.latenightsw.com/
```

Script Compatibility

Since BBEdit's scripting model has changed over time, scripts prepared for much older versions may need revision in order to work properly. For example, since BBEdit allows multiple documents to be open within a single text window, you may need to revise existing scripts which presume documents and windows are identical.

Distinguishing Between Script Elements

Because different applications handle different types of data, you should be aware that the actual data, or the interface items, referred to by a particular name may not be consistent from application to application. The following sections describe how several common elements are handled in BBEdit.

Applying Commands to Text

Since BBEdit supports opening multiple documents within a single text window, all scripting commands which operate on text must specifically target the text contents of a window, or a document within that window, rather than the window itself.

For example, you may use:

```
count lines of text of text window 1
```

or:

```
count lines of active document of text window 1
```

but not:

```
count lines of window 1
```

Documents vs. Windows

In substantially older versions of BBEdit, the object classes *document* and *window* could be used interchangeably, and generally had the same properties listed in the scripting dictionary. This is no longer the case.

The class *window* corresponds to a window (of any type—text or otherwise) on screen, and thus the properties of the *window* class refer strictly to properties of a window on screen. If a document is associated with a window, the document is accessed as the *document* property of the window:

```
document of text window 1
```

The class *document* refers to a document, and as with a window, the document's properties pertain strictly to the condition of a document (that is, something that can be saved to disk and opened later). Note that this does not mean a document must be saved to a file, only that it could be.

As a rule, documents and windows are associated with each other, but it is important to remember that there is not a one-to-one correspondence between windows and documents. For example, the About box is a window which has no document associated with it. Furthermore, in current versions of the application, there is no such thing as a document with no associated window.

Here is a general overview of the object classes used in BBEdit:

Classes of Windows

- *window*: the basic window class contains properties that can be fetched and set for any window on screen: position, size, and so forth.
- *palette*: the palette class refers to windows that float above all others on the screen; the HTML tools palette, scripts list, and so on.
- *text window*: the text window class provides properties which are specific to text-editing windows as on-screen entities. These properties pertain mostly to the display of text in the window: *show invisibles*, *auto_indent*, and so on. In addition to the text-editing-specific properties, the basic window properties are also accessible.
- *project window*: provides a way to reference windows corresponding to open projects. A group window does not present any properties beyond the basic *window* class, but provides a way to differentiate project windows from other types of window.
- *disk browser window*: provides a way to reference windows corresponding to open disk browsers. A disk browser window does not present any properties beyond the basic *window* class, but provides a way to differentiate disk browser windows from other types of window.
- *results browser*: provides a way to reference results generated by a batch operation. A results browser does not present any properties beyond the basic *window* class, but provides a way to differentiate results windows from other types of window.
- *search results browser*: a subclass of results browser, referringspecifically to the results of a single-file Find All command or a multi-file search.

Classes of Document

As with windows, there are various classes of document:

- *document*: the basic document class contains properties that apply to any sort of document: whether it has unsaved changes, the alias to the file on disk, and so on.
- *text document*: text documents contain information specific to text files opened for editing in BBEdit.

- *project document*: refers to a document corresponding to an open project. A project document does not present any properties beyond the basic *document* class, but provides a way to differentiate project documents from other types of document.

"Lines" and "Display_lines"

The "line" element refers to a "hard" line, that is, a stream of characters that begins at the start of file or after a line break, and which ends at the end of file or immediately before a line break. This is consistent with the semantics of "line" in hard-wrapped documents, and these semantics also apply within soft-wrapped documents.

The "display_line" element refers to a line of text as displayed on screen (bounded by soft and/or hard line breaks).

The "startLine" and "endLine" properties of a text object always refer to the "hard" start and end of lines. In other words, if a text object crosses multiple soft-wrapped lines, the startLine and endLine properties will be the same.

Both "startDisplayLine" and "endDisplayLine" properties are part of the text object class. These serve the same purpose as the startLine and endLine semantics for soft-wrapped views in older versions of BBEdit.

Getting and Setting Properties

One significant improvement in BBEdit's new scripting framework is the ability to get and set multiple properties of an object with a single scripting command. Every object has a property called *properties*. This property returns a record which contains all of the properties which can be fetched for that object. For example, the script command

```
properties of text window 1
```

will return a result like this one:

```
{{id:55632400, container:application "BBEdit", bounds:{31, 44,
543, 964}, closeable:true, collapsed:false, index:1,
modal:false, file:alias "Hard
Disk:Users:Shared:doc_examples:index copy.html", modified:false,
name:"index copy.html", position:{31, 44}, resizable:true,
selection:"", contents:"..."}
```

Conversely, to set one or more properties at once is very easy:

```
set properties of text window 1 to { show invisibles: true, show
spaces : true, soft wrap text : true }
```

Only the properties specified will be changed. The rest will not be modified.

It is important to note that when setting properties in this fashion, you can only set modifiable properties. If you attempt to set any read-only properties, a scripting error will result:

```
set properties of text window 1 to { show invisibles: true,
modal: false, expand tabs: true }
```

The above script command will turn on Show Invisibles and then report a scripting error, since *modal* is a read-only property.

Performing Actions

The following sections provide basic information on how to perform various common actions via AppleScript.

Scripting Searches

The ability to script searches presents you with a very powerful tool, since you can prepare a script which instructs BBEdit to perform a whole series of search or search and replace operations.

Consider the scripting command below:

```
tell application "BBEdit"

find "BBEdit(.+)$" searching in document of text window 1 ¬
  options { search mode: Grep } with selecting match

end tell
```

In substantially older versions, the *find* command always operated on the front window; however, you must now explicitly specify the text to be searched, either by specifying an explicit tell target, or by supplying a *searching in* parameter. So the following scripts are equivalent:

```
tell application "BBEdit"
    find "BBEdit" searching in document of text window 1
end tell
```

and

```
tell application "BBEdit"
    tell document of text window 1
        find "BBEdit"
    end tell
end tell
```

Note that either the tell-target or the *searching in* parameter must resolve to something that contains text. As a shortcut, you can specify a window, and if the window contains text, the search can proceed. You can also specify a text object:

```
find "Search Text" searching in (lines 3 thru 5 of document of
text window 2)
```

Please also bear in mind that the defaults for parameters not specified in the *find* command are independent of those visible within the user interface (that is, the Find and/or Multi-File Search windows).

When performing a *find*, BBEdit will return a record describing the results of the search. This record contains a Boolean which indicates whether the search was successful, a reference to the text matched by the search, and the text string matched by the search. Given the first example above, the results might look like this (after reformatting for clarity):

```
{found:true,
found object:characters 55 thru 60 of text window 1 of
application "BBEdit",
found text:"BBEdit"}
```

Scripting Single Replaces

To do a single find and replace via AppleScript, you can write:

```
tell application "BBEdit"

set result to (find "BBEdit" searching in text of ¬
    text window 1 with selecting match)

    if (found of result) then
       set text of (found object of result) to "Replacement"
    end if

end tell
```

When performing a grep search, you cannot just replace the matched pattern with a replacement string; the grep subsystem needs to compute the substitutions. The *grep substitution* event is provided for this purpose; given a preceding successful Grep search, it will return the appropriate replacement string. So if you perform a grep search, the script would look like:

```
tell application "BBEdit"

set result to find "BBEdit(.+)$" searching in text of ¬
    text window 1 options {search mode:grep}

    if (found of result) then
         set text of (found object of result) to ¬
             grep substitution of "\\1"
    end if

end tell
```

Note that when using a backslash "\" character in AppleScript, it needs to be "escaped" by means of another backslash; thus, in the above example, "\\1" used in the script, will become the grep replacement string "\1" when passed to BBEdit.

Scripting Multi-File Searches

In BBEdit, a multi-file search is a simple extension of the *find* scripting command. To search a single file or folder for all occurrences matching the search parameters, specify the file or folder as the *searching in* parameter of the search.

For example, to find all occurrences of "index.html" in a website, one might use the following scripting command:

```
find "index.html" searching in (alias "Files:WebSite:")
```

Likewise, to find JavaScript line comments:

```
find "//.+$" searching in (alias "Files:WebSite:") ¬
    options {search mode: Grep}
```

To search in a single file:

```
find "crash" searching in (alias "Files:WebSite:index.html")
```

Scripting the Clipboard

BBEdit has multiple clipboards. These are fully accessible via the scripting interface. Due to operating system constraints, most clipboard operations require BBEdit to be frontmost.

Here are some examples:

```
count clipboard
```

- Returns the number of clipboards supported by the application

```
clipboard 1
```

- Returns {index:1, contents:"Files:WebSite:", length:14, is multibyte:false, display font:"ProFont", display font size:9, style:{plain}}

```
clipboard 1 as text
```

- Returns "Files:WebSite:"

```
clipboard 1 as reference
```

- Returns clipboard 1 of application "BBEdit"

```
current clipboard
```

- Returns the current clipboard as a record (you can coerce it to reference or text or get individual properties)

To set the text in a given clipboard to literal text:

```
set contents of clipboard 3 to "foobar"
```

To set the text in a clipboard to text represented by an object specifier:

```
set contents of clipboard 3 to selection of window 2
```

To copy the contents of one clipboard to another:

```
set contents of clipboard 5 to clipboard 3
```

or, to set the current clipboard to the contents of a different clipboard, (thus making it exportable to the system clipboard):

```
set current clipboard to clipboard 3 as text
```

or finally, with even less typing involved:

```
set current clipboard to clipboard 5
```

To make any clipboard the current clipboard, select it:

```
select clipboard 5
```

Scripting Text Factories

You can apply a text factory to a file via the AppleScript interface. The minimum invocation is:

```
apply text factory <file reference> to <reference>
```

The "to" parameter can be a single reference or a list of references, as for the multi-file "find" or "replace" events.

Optional parameters include "filter", "saving", "recursion", "text files only", "search invisible folders", all with the same meanings as in the multi-file "replace" event.

Setting Text Encodings

When specifying the encoding to use for opening or saving a file, you may either use the encoding's internet name, or its exact display name (as shown in the Read As popup menu).

For example:

```
open {file "Hard Disk:Users:Shared:example.txt"} reading as
"Western (ISO Latin 1)"

open {file "Hard Disk:Users:Shared:example.txt"} reading as
"iso-8859-1"
```

Arranging Documents and Windows

BBEdit provides considerable control for handling windows and documents both directly and via AppleScript.

Opening Documents

The "open" command supports additional options, which allow you to override your window handling settings on a case by case basis:

```
open aFileList opening in <value>
```

As in previous releases, <value> may be a reference to an existing text window. However, you may instead specify "front_window", "new_window", or "separate_windows", which have the following effect:

- front_window: All files in aFileList are opened in the frontmost text window. (If there is no text window open, BBEdit will create a new one.)
- new_window: All files in aFileList are opened into a new text window.
- separate_windows: Each file in aFileList is opened into its own text window.

Moving Documents

The "move"command can be used to move text documents between text windows. For example:

```
tell application "TextWrangler"
    if (count of text windows) > 0 then
        select text window 1
        repeat while (count of text windows) > 1
            set ct to count documents of text window 2
            repeat with i from 1 to ct
                move document 1 of text window 2 to text window 1
            end repeat
        end repeat
    else
        beep
    end if
end tell
```

Referencing Documents

Previously, documents were indexed inside of multi-document windows by their display order in the sidebar. This meant that "document 1" of the application might not be the active document, which in turn required scripts to make special provisions to deal with the presence of multiple documents in a single window.

In order to handle this, BBEdit 8.0 introduced the "active document" property, which you could always use to specify the currently active document of a given text window. For example:

```
active document of text window 1 of application "BBEdit"
```

Although BBEdit still supports the "active document" property, this is no longer necessary. Instead, if a text window is frontmost, the following references:

```
document 1 of application "BBEdit"

document 1 of text window 1 of application "BBEdit"

active document of text window 1 of application "BBEdit"
```

all resolve to the same document. The side effect of this change is that if you wish to access documents within a text window by index, that index is:

a) not related to the visual ordering of documents in the sidebar, and,

b) documents' indexes may change over time

This situation is effectively no different than handling documents which are contained in individual text windows, i.e. the index will change over time when you select different windows. If your script needs to keep a permanent references to a particular document, you should refer to that document by its id rather than its index.

Common AppleScript Pitfalls

Here are some things to watch out for when scripting BBEdit with AppleScript.

The Escape Issue

AppleScript uses the backslash character as an escape character. You can use \n or \r to specify a literal line break or \t to indicate a tab character. More importantly, you can use \" or \' to include a quote mark or apostrophe in a string that is delimited by quotes or apostrophes. If you want to specify a literal backslash, you must write \\ i.e. a pair of backslashes.

That's not all that confusing until you start writing AppleScripts that call on BBEdit's powerful grep searching capability. BBEdit *also* uses the backslash as an escape character. If you want to search for an actual backslash in a document, you have to tell BBEdit to search for \\. However, if you do that in AppleScript, you must keep in mind that AppleScript will first interpret the backslashes before passing them to BBEdit. To pass one backslash to BBEdit from AppleScript, you must write two in AppleScript.

So to tell BBEdit to search for a single literal backslash from an AppleScript, you must write no fewer than *four* backslashes in the script. Each pair of backslashes is interpreted as a single backslash by AppleScript, which then passes two backslashes to BBEdit. And BBEdit interprets those two backslashes as a single one for search purposes. (This proliferation of backslashes can make your scripts look a bit like a blown-over picket fence.)

The Every Item Issue

When writing a script that loops through every item of a BBEdit object (for example, every line of a document), do not do it like this:

```
repeat with i in every line of text document 1
   -- do stuff here...
end repeat
```

This forces BBEdit to evaluate “every line of document 1” every time through the loop, which will slow your script significantly. Instead, write

```
set theLines to every line of text document 1
repeat with i in theLines
   -- do stuff here...
end repeat
```

Working with macOS Shortcuts

This release of BBEdit introduces expanded support for macOS “Shortcuts”, via additional actions provided in the Shortcuts application. A “Transform Text” operation allows invocation of "one shot" operations of many kinds, and transforms are provided for extracting matching lines, deleting matching lines, sorting lines, and text replacement.

Although accessors are provided for getting and setting the contents of the active BBEdit document (or its selection range), the Shortcuts support is intended to make BBEdit's powerful text transformations available to workflows outside the application -- after all, BBEdit already supports (and will continue to support) text factories as well as text filters, any of which can purpose-built for operating on text within the application.

CHAPTER

14 Working with Development Tools

This chapter describes how to set up BBEdit to work with development environments. BBEdit offers an arsenal of capabilities in support of development tasks, beginning with syntax coloring and function browsing support for numerous languages, as well as support for Exuberant Ctags. BBEdit also offers direct integration with the system-supplied Perl, Python, and Ruby environments, as well as shell scripts and other Unix scripting tools, the Git source control system, and a number of commercial AI systems. Additionally, you can invoke BBEdit from the command-line via optional tools, or employ shell worksheet windows to store and execute frequently used commands.

In this chapter

Configuring BBEdit for Development Environments

By default, BBEdit will display a separate menu for files maintained under the Git source control system if this tool is installed on your Mac in any of its standard locations. You can enable or disable display of any tool's menu by checking or unchecking its entry in the Menus & Shortcuts settings panel.

Syntax Coloring

Syntax coloring is the practice of drawing keywords and other language elements in colors which differ from the standard text color to add emphasis and improve the readability of your code. BBEdit offers built-in syntax coloring support for a wide range of programming languages and other types of structured content. You can adjust BBEdit's default text colors or define color schemes in the Text Colors settings panel, or assign a color scheme to a specific language in the Languages settings panel.

Color schemes now provide an option that you can set to control the color BBEdit should use for live search matches (via the Find window or Live Search bar): one setting for the selected match, and one setting for all others. You can adjust this behavior using the corresponding color setting in the Text Colors settings pane. (Absent an explicit setting, BBEdit will use the previous defaults.)

Ctags for Enhanced Language Support

In addition to its native function browsing capability, BBEdit supports the use of information from ctags-format 'tags' files for navigating source code files. BBEdit does not include a tag file generator; we recommend Universal Ctags:

```
https://ctags.io
```

Using ctags

BBEdit allows you to use tags files as text completion sources, and will recognize any tags files associated with your documents.

You may place tags files generated via ctags in the Completion Sources folder of BBEdit's application support folder (see page 36) for use as text completion sources.

If one or more tags files are found in the same directory as the front document, or in any parent directory up the chain from the front document, you can employ the ctag information by selecting a word, and either choosing Find Definition from the Search menu, or by bringing up the Definitions submenu of the contextual menu.

If you choose Find Definition, BBEdit will use any available ctags information (either from completion sources or associated with the current document) to find definitions of the selected word, and open a sheet from which you can choose the desired definitions to view. Select a definition to open it, or use the Show All button to open a search results browser showing all of the available definitions.

If you use the contextual menu, the Definitions submenu will contain a list of the available definitions. Select a definition to open it, or choose Show All to open a search results browser showing all of the available definitions.

Tag File Discovery

BBEdit does not rely solely on directory scanning to discover tags files but instead uses Spotlight whenever possible. Thus, any file whose name is “tags” or whose name ends in “.tags” or “.ctags” (see below) is eligible, and if it resides in the ancestor directory hierarchy of the document, its symbols will be available for code completion and syntax coloring. Since “tags” is a filename extension, you can have multiple tags files available for the same directory hierarchy, e.g. “macOS 10.14 SDK.tags” and “Project Sources.tags”.

BBEdit exports the UTI 'com.barebones.bbedit.ctags-data', which conforms to 'public.utf8-plain-text', for files whose extensions are “tags” and “ctags”. This UTI drives the Spotlight support.

If you have disabled Spotlight on your local disk (or for the directory tree containing your source files) or if your Spotlight index is incomplete, BBEdit will discover “tags” files the old-fashioned way, and the old limitations will apply (only files named “tags” will be discovered, and so you can have only one tags file at any level in the directory tree).

Tag File Generation and Updating

Because individual workflows and setups vary widely, BBEdit does not attempt to generate or update ctags information; it only uses existing tags files.

- You can generate tags files for whatever fields you wish. BBEdit requires the signature (`--fields=+S`) to build the definitions menu. For our own source code, we include class members (`+a`) and function implementations (`+m`) in addition to the required fields.
- BBEdit looks for tags files starting in the same folder as the current document, and crawls upwards from there.

In order to keep your tags up to date, we recommend incorporating a suitable script into your build system, or employing some other mechanism such as a cron task. Xcode’s documentation contains information on creating build phases that run shell scripts and similar solutions should be possible for other IDEs.

For example, we use the below Python script to update the tags for our source code whenever a build is started in Xcode. Since ctags is efficient, this script only takes a few seconds to run, and will update the tags even if one or more files does not compile.

```
#!/usr/bin/python
import os
import sys

TAGS_TEMP_FILE = '/tmp/tags'

projectDir = os.environ['SRCROOT']
projectName = os.environ['PROJECT_NAME']

ctagsExecutablePath = "/usr/local/bin/ctags"

baseArgs = '--excmd=number --tag-relative=no --
fields=+a+m+n+S -f /tmp/tags -R'
appendArg = '--append'

os.chdir('/')

sourceDir = os.path.join(projectDir, "ApplicationSource")
tagsFile = os.path.join(projectDir, 'tags')

# create the project's tags in '/tmp'
if os.access(sourceDir, os.F_OK):
    buildTagsCommand = ''''%s' %s '%s' ''' %
(ctagsExecutablePath, baseArgs, sourceDir)
    output = os.popen(buildTagsCommand).read()

# move it where it goes
os.rename(TAGS_TEMP_FILE, tagsFile)
```

Tag Files as Completion Sources

You can add tags files to specific locations to make symbols available as completion data sources when editing in desired languages. In particular:

- When you build a (coded) language module, if you place a file named “tags” in the language module’s “Resources” directory, BBEdit will use those tags as completion sources.
- You can generate a tags file (using exuberant ctags or “bbedit --maketags”) and place the resulting file in Application Support/BBEdit/Completion Data/<language name>/, where “<language name>” is the name of the language as it appears in the list of installed languages (or on the Languages popup menu).

So, for example, if you were to generate a tags file for the 10.14 SDK so that you could add completion data when editing Objective-C files, the file would go in Application Support/BBEdit/Completion Data/Objective-C/.

Tags files can be given any appropriate name, so you can have multiple tags files for a single language, and they will all be examined when generating completions.

Locating Unix tools via PATH

When locating Unix tools for various purposes, BBEdit will honor your account's PATH environment variable (provided it is available and not empty). This should result in more predictable outcomes when using aftermarket installations of open-source tools as well as for alternative installations of tools included with the system (such as Python).

If you modify your PATH, you must quit and relaunch BBEdit for those changes to take effect. Note also that if your PATH contains entries relative to the current working directory, those entries are not likely to work since $PWD is undefined for a GUI application, though absolute paths will work.

Switching Between Related Files

When editing any source file which has any related files (such as headers), you can press the Related Files button in the navigation bar or type Control-Option-up arrow to switch to its counterpart file, or vice versa. (BBEdit uses the suffix mapping options in the Languages settings panel to determine how a particular file type should be treated.)

Language Server Protocol Support (LSP) Basics

BBEdit has built-in support for the Language Server Protocol, (occasionally referred to as "LSP", not to be confused with Lightspeed Pascal).

This protocol defines a communication standard by which outboard programs ("servers") can respond to requests from an editor ("client") with appropriate responses based on the content and location of files written in the language(s) that the server supports.

NOTE Despite the term "server", nothing is transmitted over a network, nor does any data otherwise leave your computer by means of LSP support.

By using LSP when a suitable server is available on your computer, BBEdit can provide improved text completion suggestions, provide enhanced navigation both within and between files, and enhance or enable other features and behaviors.

Feature Support

All of BBEdit's core features work just fine without a language server, so if you never install a server, BBEdit will behave just as it always has.

The following features are built using LSP support:

- If a language server is running for the document's language, BBEdit will ask the server for completions rather than attempting to compute them on its own.
- If a language server supports the "signature help" feature, BBEdit enables the "Show Parameter Help" command on the Edit menu; choosing this will open a panel providing assistance for filling in function parameters at the current insertion point (if applicable).

- If a language server reports issues (errors and warnings) for a file in which you're editing, ranges corresponding to those issues get highlighted according to their severity, and the corresponding lines are highlighted in the line number bar. Click on a highlight in the line number bar to open a popover showing issues on that line; double-click on an item in the popover to select the relevant range of text and dismiss the popover. The popover has a button, "Show All". If enabled this will open a results window showing all of the issues in the current file.

  Use the "Show/Hide Issues" command on the View → Text Display menu to toggle the display of issues in the text view.

  The "Editing" settings pane has a setting to show tick marks in the scroll bar to indicate lines containing issues reported by the language server.

  When issues are available, there's an item to the right of the function popup with a colored (or gray) dot and a number. The dot is green if the server returned an empty list ("no issues") yellow if there are warnings, red if there are errors, and the number indicates the total number of issues found by the language server in the file. If the dot is gray and shows a dash (-), the server has not (yet) returned any diagnostics information for this file. Click on the indicator to open a popover which shows all of the issues in the file; clicking on an item in the list will select it in the document. (Double-click an item to select the location and dismiss the popover.)

- If a language server claims that it can format documents, the "Reformat Document" command sends the document's text to the server, along with information about the tab width, whether or not "Auto-Expand Tabs" is turned on, whether "Strip trailing whitespace" is turned on, and whether "Ensure file ends with line break" is turned on. The server is expected to use all of this information to generate correctly formatted text, but is not required or guaranteed to do so.

  Likewise, if the server claims that it can format a range of text in a document, the "Reformat Selection" command sends the selected range of text to the server for reformatting. Note that some servers may support range formatting but be very fussy about the range of text that is selected for formatting.

- Command-double-click on a word will direct the request to an appropriate language server and perform the equivalent of "Go to Definition", if possible.

- When right-clicking on a word in a documents served by an LSP server, BBEdit will ask the server for declarations and definitions of the word. If one (of each) is found, the contextual menu will contain "Go to Declaration" and "Go to Definition", as appropriate. (We disclaim all responsibility for nonsense results returned by the language server, we can only do what we're told.)

- The “Go” menu contains two commands: “Go to Declaration” and “Go to Definition”. These both require a running language server for the document’s language.

  Choosing either command will ask the language server for the location of the respective symbol declaration/definition, based on the location of the insertion point (or start of the selection range).

  NOTE: If you are using a C-family language for which clangd is the language server, these commands are effectively synonymous, and will alternate (if appropriate) between the declaration and the definition of the selected symbol when the same command is chosen repeatedly. This is an intentional behavior in clangd itself, and is not a bug in either BBEdit or in clangd.

- “Find References to Selected Symbol” on the Search menu will ask the server to return all occurrences of places where the symbol is used. The search symbol is determined based on the selection range, when a word is selected or the insertion point is in the middle of a word. The results are presented in a standard results window.

  In addition, right-clicking in or on a selected word will add a “References” submenu to the resulting contextual menu, showing the references (if any) to that symbol.

- “Find Symbol in Workspace” command on the Search menu will open a modal panel so that you can search for symbols. Each search request goes to the server, which is in complete control of what (if any) results get returned.

- The “(Go to) Named Symbol” command will present symbols returned by an available language server, if possible.

- “Find Definition” will ask the language server to locate the requested symbol, if available.

- Symbol Renaming

Code Actions

BBEdit offers support for LSP “Code Actions” when available. If the server supports these at all, they will be offered contextually based on the insertion point/selection range when right-clicking in a text document. BBEdit will ask the server what it supports, and the server may respond with a list of operations that are appropriate for that location in your source file.

If the server returns such a list, BBEdit will present the operations on the “Code Actions” submenu of the contextual menu.

Not all language servers support this feature; and not all servers even document what code actions they **do** support. And even then, the list of available operations for any arbitrary point in code is not guaranteed. If you don’t seeing the operations you expect when right-clicking in a text document, we recommend filing a bug report with the server developer.

Expanded Syntax Coloring Support

BBEdit 16 offers expanded syntax coloring support, via the Language Server Protocol's "semantic token coloring" feature. When supported by the active language server, this allows the server to provide information on specific ranges of the source code that supplements BBEdit's built-in syntax coloring for the given language.

In cases where the server provides coloring information for a given range of text, BBEdit will always prefer that information over its own keyword lookups (including ctags data).

The presentation and availability of this feature varies by the language and server implementation; some servers do not support semantic token coloring at all, and some do not color the full range of possible symbols.

Due to the requirements of the protocol and variations in individual server behavior, there may be an observable delay before the coloring adjusts when changes are made in the document. This delay is not under the application's control.

Symbol Renaming

When using a language server that supports symbol renaming (specifically, the `textDocument/rename` command), you can right-click on a symbol, and if the server thinks you should be allowed to rename the symbol, a "Rename" command will appear on the contextual menu.

When you choose this command, BBEdit will prompt you to enter the new symbol name, and confirming the entry will perform the rename operation. Symbol renames potentially may affect multiple files; if so BBEdit will open them as needed.

Changes performed by a symbol rename are undoable in each file, if desired.

Language Server Installation & Configuration

In order for a language server to be usable by BBEdit, it must be installed and (in some cases) configured according to the server developer's instructions. Most of the time, a server may be installed using Homebrew (or other package manager) or npm, but may also be built from source.

In order to provide more complete and timely information than is possible in this manual, we have posted and will maintain comprehensive information about language server installation and configuration within the Support section of our website:

`https://www.barebones.com/support/bbedit/lsp-notes.html`

BBEdit and the Unix Command-Line

This section describes BBEdit's facilities for interacting with the Unix command-line: shell worksheets for issuing commands *to* the Unix shell and the "bbedit", "bbdiff", and "bbfind" command-line tools for invoking BBEdit *from* the command-line.

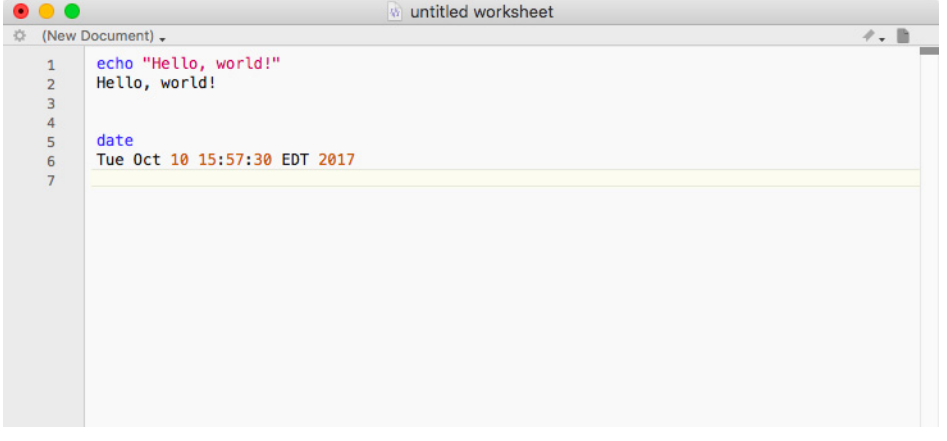
Shell Worksheets

BBEdit allows you to store and execute Unix command-lines by means of a "shell worksheet". Choose Shell Worksheet in the New submenu of the File menu to open a new worksheet using your default Unix shell ('bash' under macOS 10.14 or 'zsh' under macOS 10.15 and later).

Shell worksheets are stored in a private document format which is not text-based. This format allows BBEdit to store auxiliary data in the worksheet file's data fork, thus ensuring that worksheets can safely be stored in version control systems that are not resource fork aware.

Using Worksheets

You can type, delete, and edit text in a worksheet window just as in an ordinary BBEdit document window. To invoke a Unix command, type the command, then press the Enter key or Control-Return, or click in the status area at the bottom-left of the worksheet window. (Keep in mind that Enter and Return are different keys; pressing Return by itself inserts a line break instead of executing a command.) You can execute more than one command at a time by selecting multiple lines and pressing Enter or Control-Return. The output will appear in the worksheet window below the line or lines containing the commands executed. Unlike a terminal, this does not have to be at the end of the document: you can type commands anywhere in the worksheet window, or place the insertion point back in a previously executed command to run it again.



If the selection range is non-empty, only the exact text selected will be executed; if there is just an insertion point, the entire line containing it will be executed (even if it is not at the end of the line).

Note Command-Return will no longer execute commands within shell worksheets by default since BBEdit instead uses this shortcut as the default key equivalent for the New Line After Paragraph command. You can however use Control-Return for the same purpose, or alternatively, you can manually reassign this key shortcut to the new "Send Command" placeholder in the Shell Worksheet group of the Menus & Shortcuts settings pane.

The status area at the bottom-left of the worksheet window shows the name of the Unix process currently executing (or the name of the shell itself when no process is running). This can be useful for seeing what is going on when a process hangs or takes a long time to complete. You can kill the currently running process by typing Control-C or Command-Period in the worksheet window. Also, clicking in the status area sends the currently selected text (or the line containing the insertion point) as a command to the Unix shell.

Keep in mind that shell worksheets are not terminal windows. If you have ever used MPW, you will probably feel right at home using shell worksheets. If you are only familiar with terminal emulators, however, you will find that shell worksheets work quite differently. Command line editing gestures do not work, nor will any Unix commands that expect to be dealing with terminals. (For example, try running "emacs" in a shell worksheet.)

When you drag files or folders into a worksheet window, the behavior is different than when dragging these items into an ordinary document window. An unmodified drag of a single file or folder will insert the POSIX-style path of that item at the drop location. Additionally, rather than selecting the inserted text as in a normal editing window, the insertion point will be left at the end of the current line, so you can easily continue entering additional information or execute the line as a command.

Dragging multiple files and folders will produce a set of paths for those items, with spaces for separators rather than line breaks. This makes it easier to add arguments to a line for immediate execution as part of a command.

If you hold down the Command key while dragging, it will cause the file's contents to be inserted (or a folder listing, if the item you are dragging is a folder).

The default working directory for new worksheet windows is the user's home directory. This directory is also used as the search directory for any Open Selection or Open File by Name operations executed from within the worksheet. New shell windows are colored using the "Unix Shell Script" language.

New shell worksheets initially run in normal user mode as the currently logged-in user, but if you invoke 'sudo' within a shell worksheet, BBEdit will automatically prompt you (if necessary) to enter your password.

WARNING If you are not familiar with Unix command-line tools, we strongly urge you to obtain and read an introductory guide to using a Unix shell. Command-line tools can be very useful, but if used incorrectly, they can render files, or even your entire system, unusable.

Default Worksheet Stationery

When creating a new worksheet window, BBEdit will look for a worksheet stationery file named "Default Worksheet Stationery". This file is located in the Stationery folder of BBEdit's application support folder. (See Chapter 2 for more information regarding BBEdit's application support folder.) If the default worksheet stationery exists, you will see the contents of this file in every new worksheet window you create.

BBEdit ships with a default worksheet stationery file that provides a small tutorial on using worksheet windows. When you grow tired of seeing this tutorial in every new worksheet, you can either remove the “Default Worksheet Stationery” file from the Stationery folder, or replace it with one of your own.

Exporting a Worksheet’s Contents

When a (non-empty) shell worksheet is active, the Export command in the File menu will become Export to Text, and you can select this command to save a text-only representation of the active worksheet’s contents.

Installing the Command Line Tools

You can install BBEdit’s command line tools: “bbedit”, “bbfind”, “bbdiff” and “bbresults” at any time by choosing “Install Command-Line Tools” from the BBEdit (application) menu and entering your login password to approve each required step.

(If older versions of the tools are installed, choosing this command will update them; it will not overwrite existing versions of the tools with older versions.)

The “bbedit” Command Line Tool

You can use the “bbedit” command line tool to open files into BBEdit via the Unix command line.

To open a file into BBEdit from the command line, type

```
bbedit filename
```

where *filename* is the name of the file to be opened. You may also specify a complete FTP or SFTP URL to a remote file or folder to have BBEdit open the file, or an FTP/SFTP browser to the folder.

To launch BBEdit without opening a file (or to activate the application if it is already running), type

```
bbedit -l
```

You can also pipe STDIN to the “bbedit” tool, and it will open in a new untitled window in BBEdit: for example,

```
ls -la | bbedit
```

If you just type

```
bbedit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to BBEdit.

The basic command line syntax for the “bbedit” tool is

```
bbedit [ -<short-form options> --<long-form options> ] [ -e
  <encoding_name> ] [ -t <string> ] [ +<n> ] [ file (or) <S/FTP
  URL> ... ]
```

See the “bbedit” tool’s man page (“man bbedit”) for a complete description of the available switches and options.

The "bbdiff" Command Line Tool

You can use the "bbdiff" command line tool to apply BBEdit's Find Differences command to a pair of files or folders specified on the Unix command line.

To invoke the Find Differences command from the command line, type

```
bbdiff oldfile newfile
```

or

```
bbdiff oldfolder newfolder
```

where *oldfile* and *newfile* are the names of the files, or *oldfolder* and *newfolder* are the names of the folders, to be compared. You can also specify options for how the Find Differences command will be applied, which correspond to those available in the dialog.

The complete command line syntax for the "bbdiff" tool is

```
bbdiff [ --<options> ] [ OLDFILE NEWFILE | OLDFOLDER NEWFOLDER
  ]
```

See the "bbdiff" tool's man page ("man bbdiff") for a complete description of the available switches and options.

Invoking "bbdiff" as an External Helper

When using "bbdiff" as an external diff helper for any other program such as Git, you should invoke it with the --wait option.

The "bbfind" Command Line Tool

You can use the "bbfind" command-line tool to access BBEdit's powerful multi-file search from the Unix command line.

To perform a multi-file search from the command line, type

```
bbfind search-string search-path
```

where *search-string* is your search string (or pattern) and *search-path* is a list of path(s) to search. You can also specify options which control how the search should be performed; these options correspond to those available in the Multi-File Search window.

If no search paths are specified on the command line, "bbfind" will attempt to read them from standard input. This makes it easy to process the output of other tools such as "find". For example:

```
`find . -name "*.py" -print | bbfind blah`
```

takes the paths printed by "find" and searches those files.

By default, "bbfind" expects that input will be separated by Unix newlines (\n). If instead, the input is being generated programmatically and contains "NUL"-separated paths, you can specify the "-0" option. Again using "find" as an example input source:

```
`find . -name "*.py" -print0 | bbfind blah -0`
```

The complete command line syntax for the “bbfind” tool is

```
bbfind search-string [-cEghInRSvVwZ0 --<long_form_switches>
  search-path ]
```

See the “bbfind” tool’s man page (“man bbfind”) for a complete description of the available switches and options.

The “bbresults” Command Line Tool

You can use the “bbresults” command-line tool to pass error results to BBEdit from the Unix command line.

This tool reads data from STDIN which is expected to be typically formed Unix error messages and passes that data to BBEdit, which will create a results browser to provide convenient navigation of errors and warnings.

For example:

```
proselint --demo | bbresults
```

or

```
flake8 foobar.py | bbresults --pattern flake8
```

or even

```
grep -n void *.c | bbresults
```

See the “bbresults” tool’s man page (“man bbresults”) for a complete description of the available switches and options as well as information on how to employ this tool.

You can learn more about `proselint` here:

`https://github.com/amperser/proselint`

Unix Scripting: Perl, Python, Ruby, Shells, and more!

BBEdit provides robust integration with numerous Unix scripting environments, including Perl, Python, Ruby, and shell scripts.

Using Unix Scripts

BBEdit works directly with the native Perl, Python, and Ruby environments provided by macOS, and supports similar integration with shell scripts and any other Unix scripting language.

BBEdit’s Unix scripting features are accessed via the Shebang menu: “#!”. (Why “Shebang”? Because executable Unix scripts traditionally start with the two-character sequence “#!”. Some people pronounce these two characters “hash-bang,” others say “sharp-bang,” but the most common pronunciation is simply “shebang”.)

The “shebang line” is the first line of the script, and includes a Unix-style path to the interpreter for the language—for example, “#!/usr/bin/perl”, or “#!/usr/local/bin/python”. Further, when you save any new file which begins with “#!”, BBEdit will automatically mark the resulting file as executable.

While BBEdit does not entirely depend upon the accuracy of the shebang line (if your script file has an accurate language mapping), it is always a good practice, and sometimes necessary, to specify the full path to the executable in the shebang line.

Dealing With Quarantined Scripts

By default, whenever you save a file that begins with a shebang line (“#!”), macOS will “quarantine” that file so that it cannot be run from the command line or by other applications. You can address this by granting BBEdit sandbox access; but if you declined to do so the first time you used BBEdit, then the quarantine will occur.

Note The help (question mark) button in the alert will take you to a page on our web site which explains the issue as well.

Language Resources

Perl is an acronym for Practical Extraction and Report Language (or alternatively, Pathologically Eclectic Rubbish Lister) and was developed by Larry Wall. If you are interested in learning Perl, the quintessential Perl references are:

Learning Perl (4th Edition), by Randal L. Schwartz & Tom Phoenix. O’Reilly and Associates, 2005. ISBN: 0-596-10105-8

Programming Perl (3rd Edition), by Larry Wall, Tom Christiansen, Jon Orwant. O’Reilly and Associates, 2000. ISBN: 0-596-00027-8

The following are excellent Internet resources for the Macintosh implementation of Perl, and Perl in general:

Perl.com from O’Reilly and Associates
https://www.perl.com/

Perl Mailing Lists
http://lists.cpan.org/

Python is a portable, interpreted, object-oriented programming language, originally developed by Guido van Rossum. If you are interested in learning Python, consider the following Internet resources as a starting point:

Python home page
`https://www.python.org`

Ruby is an interpreted scripting language with an emphasis on object-oriented programming, which has fast become a favorite of Web developers. Ruby was created by Yukihiro Matsumoto. If you are interested in learning Ruby, consider the following books:

Programming Ruby: The Pragmatic Programmer's Guide (2nd Edition), by Dave Thomas, with Chad Fowler and Andy Hunt. Pragmatic Bookshelf, 2004. ISBN: 0-9745140-5-5

Ruby Cookbook, by Lucas Carlson & Leonard Richardson. O'Reilly and Associates, 2006.
ISBN: 0-596-52369-6

Internet resources for Ruby:

Ruby home page
`https://www.ruby-lang.org/`

Setting Environment Variables for GUI Apps

BBEdit reads your account's command-line environment directly; thus, you need not employ any special mechanisms to pass environment settings to it.

Line Endings, Permissions and Unix Scripts

To execute scripts, the script interpreter for any given language requires source code to be encoded with native line endings, i.e. Unix line breaks for Perl and most other shell scripting languages. BBEdit will warn you if you attempt to run a script which does not have Unix line endings.

Additionally, to execute scripts anywhere outside of BBEdit (e.g. in the Terminal), the system requires that the script file have 'execute' permissions set. Thus, when you first save any script file which contains a shebang (#!) line, BBEdit will automatically set execute permissions for your login account (a+x, as modified by the umask) on that file.

Configuring Perl

BBEdit can make full use of the system's default Perl install or any additional installations with no need for further configuration.

Search Paths

By default, Perl looks for modules in its standard library path and in the current directory. You may also use modules from other locations by specifying their paths in the PERL5LIB environment variable.

Configuring Python

BBEdit can make full use of the system's default Python install or any additional (newer) installations with no need for further configuration.

Configuring Ruby

BBEdit can make full use of the system's default Ruby install with no need for further configuration. However, if you wish to install and work with multiple versions of Ruby, you will need to specify the appropriate version in your scripts' shebang lines.

Shebang Menu

The commands in this menu allow you to run Unix scripts directly within BBEdit.

Check Syntax

Checks the syntax for the frontmost window. Errors are displayed in a standard BBEdit error browser (see Chapter 9, "Browsers," for more details on working with error browsers). This command is only available for Perl and Python scripts.

Run

Runs the script in the frontmost window by default. If this script has an associated disk file, BBEdit will automatically set the current working directory to the directory containing the script file, and any output from the script (on STDOUT) will be displayed in BBEdit's "Unix Script Output" window. By default, errors for Perl and Python scripts are displayed in an error browser, while errors for other languages will be written into a new document.

When you apply the Run (or Run with Options) command to a file which contains a "shebang line", BBEdit will no longer attempt to run the file using the appropriate interpreter for the language (e.g. Python), but will instead ask the shell to run that file directly.

Thus, if you run a file like this:

```
#!/usr/bin/env python
import sys
print(sys.version)
```

BBEdit will always run the script using whatever Python binary the system would have used on the command line (or internally via the Run in Terminal command.

On the other hand, if you omit the shebang line ("#!"):

```
import sys
print(sys.version)
```

then BBEdit will make a best guess as to which Python binary ('python') to use, based on the previously documented rules.

If your script needs to perform any sort of interactive keyboard I/O function (such as 'input' in Python), the Run command will fail, because BBEdit is not a terminal emulator. In this case, you must instead use the Run in Terminal command, as described below.

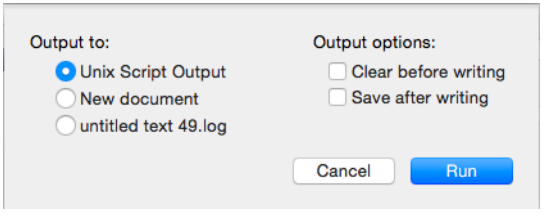
Note BBEdit will only run the file directly using the shell when using the Run (or Run with Options) or "Run in Terminal" commands. When you invoke the Check Syntax or Run in Debugger commands, BBEdit will continue to use its best guess for the language interpreter.

This behavior is controlled by an expert settings, and you may force BBEdit to ignore the shebang line by entering the following Terminal command:

```
defaults write com.barebones.bbedit AlwaysUseShebangLineForWindowRun -bool NO
```

Run with Options...

Displays the Run sheet, which allows you to set options before running the script in the frontmost window.



Output to: Choose to display output in a new document, to direct it to the Unix Output file, or to write it to an equivalently-named log file ("{script name}.log") in BBEdit's "Logs" folder, which you may access at any time via the Folders submenu of the BBEdit (application) menu.

Output Options: Set the "Clear before writing" option to clear the output file before writing.

Note Since BBEdit now autosaves output logs, there is no longer a separate option to save after writing.

Run in Terminal

This command will run the script in a new Terminal window, regardless of the settings in the Run a Script dialog.

If your script needs to perform any sort of interactive keyboard I/O function (such as 'input' in Python), the Run command will fail, because BBEdit is not a terminal emulator.

In such cases, you must instead use the Run in Terminal command to run your code in a Terminal session in order to enable proper I/O handling. (Please make sure you have first granted sandbox access via the Application settings pane.)

Run in Debugger

Runs the script in the interpreter's debugger, regardless of whether the Use Debugger option is set for the Run command; also, any output options set in the Run command will be ignored. The Run in Debugger command is only available for Perl and Python.

Run File

Runs a script from an arbitrary file rather than from a BBEdit window. The Run a Script File dialog appears. You can select a file by clicking the File button or by dragging a file to the path box at the top of the dialog from the Finder. The options are the same as the ones described above for the Run a Script dialog.

Script Logging Behavior

The Unix Script Output log, as well as file-specific logs resulting from the execution of any Shebang (#!) menu script, are now autosaved, so you need not explicitly save changes to those logs before closing them.

If a script run from the Shebang (#!) menu produces no output, rather than doing nothing (and potentially creating confusion), BBEdit will generate placeholder text to indicate that the script produced no output.

Further, if a script produces no output, the designated log will so note. However, if the log is not already open, BBEdit will not open it. If the log is open, BBEdit will update the log's window, but will not bring it to the front.

Show POD/Show Module Documentation

When the frontmost document is a Perl file and you invoke the Show POD command, BBEdit will process the document contents using by the command-line 'pod2text' tool and display the result in a new text window.

Note POD stands for Plain Old Documentation, and is the standard Perl documentation format.

When the frontmost document is a Python file, the name of this command will change to Show Module Documentation, and if you invoke it, BBEdit will display the module documentation.

Filters and Scripts

Before you begin using Unix filters and scripts with BBEdit, you should locate and familiarize yourself with the Text Filters and Scripts folders, which resides within BBEdit's application support folder. (See Chapter 2 for details.).

The contents of the Text Filters and Scripts subfolders are presented respectively in the Apply Text Filters submenu and the Scripts menu, as well as the Text Filters and Scripts floating palettes.

Document State

For convenience, BBEdit sets some runtime environment variables to provide information about the front document's state right before a Unix filter or script is run:

Variable	Description
BB_DOC_LANGUAGE	Name of the document's current language (not set if language is "none")
BB_DOC_MODE	Emacs mode of the document's current language
BB_DOC_NAME	name of the document
BB_DOC_PATH	path of the document (not set if the document is unsaved)
BB_DOC_SELEND	(zero-based) end of the selection range (not set if not text document)
BB_DOC_SELEND_COLUMN	(one-based) de-tabbed column number of BB_DOC_SELEND
BB_DOC_SELEND_LINE	(one-based) line number of BB_DOC_SELEND
BB_DOC_SELSTART	(zero-based) start of the selection range (not set if not text document)
BB_DOC_SELSTART_COLUMN	(one-based) de-tabbed column number of BB_DOC_SELSTART
BB_DOC_SELSTART_LINE	(one-based) line number of BB_DOC_SELSTART

Note Selection ranges and other offsets are expressed in **characters,** not bytes.

Filters

Text filters operate on the selected text of the frontmost document, or on the whole document if there is no selection, or on the current contents of the clipboard (if invoked via Paste Using Filter).

BBEdit will pass either the selected text (if any) or the contents of the entire document as input to the filter on STDIN as UTF-8 text (no BOM), while any output generated by the filter on STDOUT will replace the selection (or document contents), and anything written to STDERR will be logged in a separate document.

Note This method represents a change from the method used by much older versions, where BBEdit wrote a temporary file and passed it on `argv[0]`. Thus, if you have any existing Unix filters (in the "Text Filters" folder) which were based on that method, you **will** need to modify those filters to accept input from STDIN before you can use them.

There two ways to apply filters: to the current document through the Apply Text Filters submenu in the Text menu or via the Text Filters palette, or to the current contents of the clipboard via the Paste Using Filter submenu of the Edit menu.

To open the Text Filters palette, select it from the Palettes submenu in the Window menu. You can run a filter by selecting it from the list and clicking the Run button, or you can simply double-click the filter name in the list.

You can also hold down the Shift key while selecting a folder node from the menu to open that folder in the Finder.

Passing Arguments to a Filter

Text filter scripts may present a dialog box allowing the user to specify arguments to the filter (which the filter script may then use to modify its operation on the input). This is done by creating a Cocoa nib file in Xcode, and placing it in a specific location relative to the script being run (namely, "../Resources/<script base name>.xib"").

For complete details on using this capability, please see the Developer section of our website:

```
https://www.barebones.com/support/bbedit/
```

Scripts

Scripts do not operate on the text of the frontmost window, but rather run directly. You can also run scripts from the Scripts menu or the Scripts palette, and edit the selected script or assign a keyboard shortcut to it by clicking the corresponding buttons in the Scripts palette.

Additional Notes

In addition to the features detailed above, BBEdit offers some additional options which it may help you to be aware of.

Setting Menu Keys for Scripts

The Filters and Scripts palettes both have a "Set Shortcut" button at the top. Select a filter or script in the list and click this button to set a keyboard shortcut for the selected item. You may also assign key equivalents to scripts or filters within the Menus & Shortcuts settings panel.

Manually Sorting the Text Filters and Script Menus

By default, items in the Apply Text Filters submenu and the Scripts menu display in alphabetical order. However, you can force items to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "00)Foo" would sort before "01)Bar.") For such files, the first three characters are not displayed in BBEdit. You can also insert a divider by including an empty folder whose name ends with the string "-***". (The folder can be named anything, so it sorts where you want it.)

Canceling Filter or Script Execution

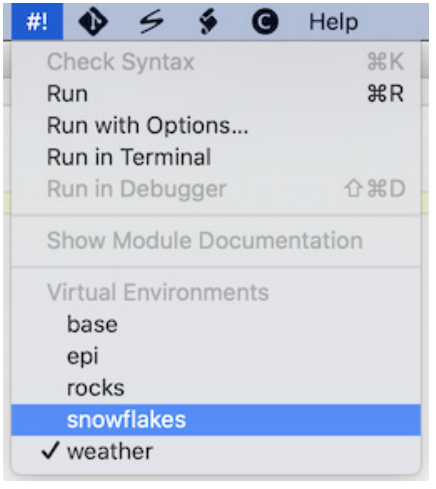
You can press the Cancel button in the progress dialog or type Command-. (Command-period) to cancel a task directly from within BBEdit. Since BBEdit must kill the spawned Unix process with a SIGINT, any unflushed data in open filehandles (including STDOUT and STDERR) will be lost unless the script takes measures to prevent this.

Opening Filters and Scripts

If you press and hold down the Option key while selecting any file-backed action (such as a script or text filter), BBEdit will reveal that action's parent file in a Finder window from which you can easily open it for editing.

Working with Anaconda

If you are using Anaconda or 'conda'/'miniconda' , BBEdit will list your available environments in the “Shebang” (#!) menu, and you can change the active environment (which has effect only within BBEdit) by choosing it from the menu.



The following notes and limitations apply:

- Scripts and filters that you run from the “Shebang” (#!) menu’s commands, as well as from the Scripts menu, Apply Text Filter, and text factories, will use the additional environment variables contributed by the current virtual environment.

  This includes, but is not by any means limited to '$PATH', and so switching virtual environments may alter the behavior of Unix script operations in ways that only make sense if you keep this in mind.

- Shell Worksheets do not currently participate in virtual environment awareness or switching.

Working with Git

BBEdit offers integrated support for the Git distributed version control system.

```
https://www.git-scm.com/
```

Configuring Git

In order to enable BBEdit's Git integration, the 'git' command-line tool must be available in any standard location, such as the OS default installation in '/usr/bin/git'. Provided this condition is met, BBEdit will display a Git menu in the menu bar, and you can perform various operations when the active document corresponds to a file that is in a local Git working copy, or apply appropriate commands to any such working copy.

BBEdit's Git integration is not intended to replace a full-featured Git client, but rather to enable you to easily perform common tasks on the file(s) you're working on without having to switch to the command-line or a dedicated GUI client.

Command-Line Integration

You can use the "bbdiff" command-line tool as an external diff tool for Git by specifying it in your Git config (~/.gitconfig) as appropriate, e.g.

```
[difftool "bbdiff"]
```

```
cmd = /usr/local/bin/bbdiff --wait --resume "$LOCAL" "$REMOTE"
```

Git Commands

You can use most Git commands on either the active document, or the selection in a results browser or disk browser; however, these commands are always available:

- Commit Staged Changes...
- Commit Working Copy...
- Show Working Copy Status...
- Fetch from Remote...
- Pull from Remote...
- Push to Remote...
- Open Log File

Check Out Branch
Displays a dialog allowing you to select and check out any available branch.

Add
Schedules the current file, or all files selected in the sidebar, for addition to the repository. (You must perform a Commit to add the file(s) to the repository.) This command will not be available if the selected files are already part of the repository.

Discard Changes
Discards local changes to the active document's file, reverting it back to BASE. Has no effect if there are no local modifications to the file.

Remove
Removes the current file from the working copy.

Remove from Index
Removes the current file from the index.

Stash
Performs `git stash` on the current file's working copy.

Apply Stash
Performs `git stash apply` on the current file's working copy.

Pop Stash
Performs `git stash pop` on the current file's working copy.

Commit
Commits the local file to the repository. BBEdit will display a Git commit window into which you can enter your commit message. This window will be prefilled with form text below the insertion point, consisting of a special separator line and a listing of the files which are about to be committed and their status. This separator line, and all lines below it, will be removed and will not appear as part of the commit message.

Commit Staged Changes
Commits all currently staged changes to the repository. BBEdit will display a Git commit window into which you can enter your commit message. This window will be prefilled with form text below the insertion point, consisting of a special separator line and a listing of the files which are about to be committed and their status. This separator line, and all lines below it, will be removed and will not appear as part of the commit message.

Commit Working Copy
Commits all changed files in the current file's working copy or the selected working copy to the Git repository.

Show Working Copy Status
If invoked while a document is active, BBEdit will show the status of the working copy to which that document belongs; otherwise, BBEdit will present a standard folder selection dialog in which you can choose the desired working copy.

The status results window includes a Reload button, which you can click to refresh the results.

Compare Revisions
Displays a sheet which allows you to select any previous/future revision of the current file in the repository and perform a Find Differences between that revision and the local revision.

Compare Arbitrary Revisions
Displays a dialog which allows you to choose any two revisions of the current file and perform a Find Differences on those revisions.

Compare with Staging/Compare with Previous/Compare with Head
These are convenience commands which allow you to compare the current file with BASE, PREV or HEAD revisions respectively.

Pull Current Branch from Remote
Pulls the current branch from its configured remote repository.

Push Current Branch to Remote
Pushes the current branch to its configured remote repository.

Fetch from Remote
Fetches all branches and/or tags from any configured repositories.

Pull from Remote
Incorporates changes from a configured remote repository into the current branch.

Push to Remote
Updates remote refs using local refs to the configured remote repository.

Show Blame
Brings up a temporary document showing what revision and author last modified each line of the current file.

When this command is invoked, the resulting annotation will place the insertion point at the beginning of the line that was selected in the source document (if there was one).

Show Revision History
Brings up a temporary document which contains the revision history of the current file.

Open Log File
Opens BBEdit's Git log file

Git Revision List Indicators
Git revision lists (as used by Compare Revisions. Compare Arbitrary Revisions, and the file version menu in the navigation bar) get indicators for revisions that have annotations (such as commit tags or special status as branch head revisions). Further details are available via hover tooltips.

Working with AI Chat Worksheets

BBEdit supports a new type of document: the AI Chat wsorksheet, which provides an interface for conversational exchanges with AI chat services, and you can create such worksheets from the File => New submenu as with other document types.

In order to use this feature, you will need a account and an API key for ChatGPT, Claude (https://claude.ai/), or Ollama (https://www.ollama.com/).

You can also add your own services and endpoints, as long as they use an API that is compatible with OpenAI's (or close enough that BBEdit can't tell the difference).

Getting Started with ChatGPT

If you already have a ChatGPT account, you can generate an API key here:

```
https://platform.openai.com/account/api-keys
```

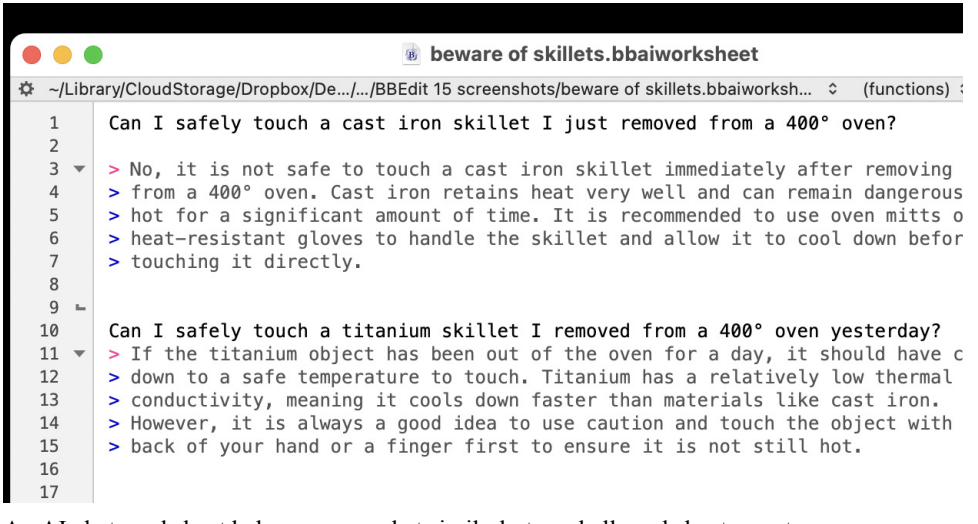
If you need an account, start here instead:

```
https://chat.openai.com/auth/login
```

(Please note that we cannot provide account or technical support for ChatGPT or any other AI chat service.)

Please further note that use of the ChatGPT API is not free, but it's not terribly expensive, either, and we are not collecting any money from this feature.

Using AI Chat Worksheets



An AI chat worksheet behaves somewhat similarly to a shell worksheet: you type something in, and press any of Control-Return, the Enter key, or Edit => Send Command to send it to the chat service. After some period of time, the service will send a response which BBEdit will insert into the worksheet window.

If you wish to cancel your request before the response arrives, you may press Command-period or Control-C.

Responses from the chat service are automatically quoted, as long as the worksheet's language is set to “Markdown”. If you change the worksheet's language, this quoting will not occur.

(Since worksheets are Markdown, you can apply the “Preview in BBEdit” to a worksheet to visualize its contents.)

Because chats depend heavily on history, a worksheet saves your prompts and the server’s responses. Thus, the document size will grow over time and context is preserved, even if you delete previous prompts and responses from the text area.

You can choose a default service and model within the AI Chat Worksheets” settings pane. There's a predefined list available in a status bar popup, but the model is specified in an edit field so you can write in your own model if it's not listed. (Choosing a nonexistent or unsupported model may result in unexpected outcomes.)

The status bar item for AI worksheets shows the service name along with the model name. You can switch services in an AI chat worksheet, and also clear the chat/response history if desired. (This may be necessary in cases where enough history accumulates that the service rejects or limits request lengths.)

NOTE Changing services requires clearing the history.

CHAPTER

15 Language Modules and Packages

Language modules are special files that you can install to add support for syntax coloring, and optionally, function browsing, for programming languages beyond those built in. Many people have prepared language modules for BBEdit, and these modules are available from various websites (including our own).

Packages are collections of related supporting items, such as filters, scripts, and language modules, which you can add to BBEdit to extend it.

This chapter describes the basic procedures for installing and using language modules and packages, and provides references to information about producing such items.

In this chapter

Language Modules

Language modules are add-on items which provide syntax coloring and function browsing for programming languages that BBEdit does not natively support.

There are two types of language modules: coded, and codeless. Coded language modules must be prepared according to the requirements of BBEdit's language module interface. (See Appendix D.) Codeless language modules are text documents prepared in a specific plist format. (See below.)

After you install a language module and relaunch BBEdit, syntax coloring and function browsing will be available for the language(s) supported by that module. To verify that a language module is active, or to modify or add file suffix mappings for the language(s) it provides, use the Languages settings panel (see page 265).

Installing Language Modules

To install a language module, move or copy the module file into the Language Modules subfolder of your BBEdit application support folder. If no such folder exists, you may create it.

After installing a new language module, you must quit and relaunch BBEdit in order to use it.

To remove an installed language module, you must remove the item's file from the Language Modules subfolder of your BBEdit application support folder, then quit and relaunch BBEdit.

Overriding Existing Modules

Language modules can override existing language definitions, including the built-in definitions. If there is more than one module present which supports a given language, BBEdit will use the module with the most recent modification date.

Codeless Language Modules

A codeless language module (CLM) is a specially-formatted text file which allows you to describe the properties of a source code language via a set of basic parameters. BBEdit will then use these parameters to perform syntax coloring and function navigation for the specified language.

Codeless language modules are written as "property lists" (or "plists"), which is anXML format that macOS uses for many purposes. You can create or edit codeless language module files with BBEdit itself, with Xcode, or with a third-party editor such as PlistEdit Pro.

`https://www.fatcatsoftware.com/plisteditpro/`

You can find complete specifications for creating codeless language module in the Developer Information section of our website.

`https://www.barebones.com/support/develop/clm.html`

Validating Codeless Language Modules

You can validate a codeless language module by opening it and choosing the Check Syntax command in the Shebang (#!) menu. This will cause BBEdit to perform the same check it does when loading modules at startup, and will report anything untoward.

Code-Based Language Modules

BBEdit also supports producing code-based language modules to handle more complex languages or document formats. You can find complete specifications for creating code-based language modules in the Developer Information section of our website.

`https://www.barebones.com/support/develop/`

Language Module Compatibility

IMPORTANT You will **not** be able to use any third-party language modules which do not support Unicode text, or which were built in CFM format. If BBEdit encounters such a module, it will not load that module, and will log a message to the system console.

Contact the developers of such a module, or visit the Bare Bones Software website (see above) for more information on the availability of updated modules.

Packages

BBEdit supports installing “packages” of items to extend its functionality. A package is simply a pre-defined group of the sort of items you can individually place into BBEdit’s application support folder (and subfolders) to extend BBEdit; however, the package format makes it easier to handle and install sets of related items.

Each package is a folder whose name must ends in “.bbpackage”, and the items within this folder must conform exactly to the following requirements.

The package folder must contain the following item:

- Contents [folder]

The “Contents” folder may contain these two items (which are currently not required, and are reserved for future use):

- Resources [folder]
- Info.plist [file]

The “Contents” folder may also contain any or all of the following subfolders:

- Clippings
- Language Modules
- Preview CSS
- Preview Filters
- Preview Templates
- Scripts
- Text Filters

The items contained within each subfolder will behave as though they were present within the subfolder of the same name inside BBEdit’s application support folder. (These subfolders are all optional, though obviously a package which contains none of them will be of no benefit.)

In order to use a populated package, you should place it within the “Packages” subfolder of BBEdit’s application support folder (~/Library/Application Support/BBEdit/Packages/).

In addition to a “Preview Filters” directory, packages can also contain “Preview Templates” and “Preview CSS” directories, within their “Contents” directory. Items placed within these folder will follow the same rules as those placed in the global “Preview Templates” and “Preview CSS” folders within BBEdit’s application support folder and will appear on the appropriate menus in preview windows.

APPENDIX

A

Command Reference

This appendix provides a quick reference for key assignments and a comprehensive list of the commands that are available from BBEdit's user interface.

In this appendix

Keyboard Shortcuts for Commands

Many of BBEdit's commands have pre-defined keyboard shortcuts. BBEdit also lets you reassign the shortcuts for any menu command, clipping item, text filter, or script to suit your own way of working.

To change the keyboard shortcut for any menu command, you can use the Menus & Shortcuts settings panel. (See "Assigning Keys to Menu Commands" on the following page.)

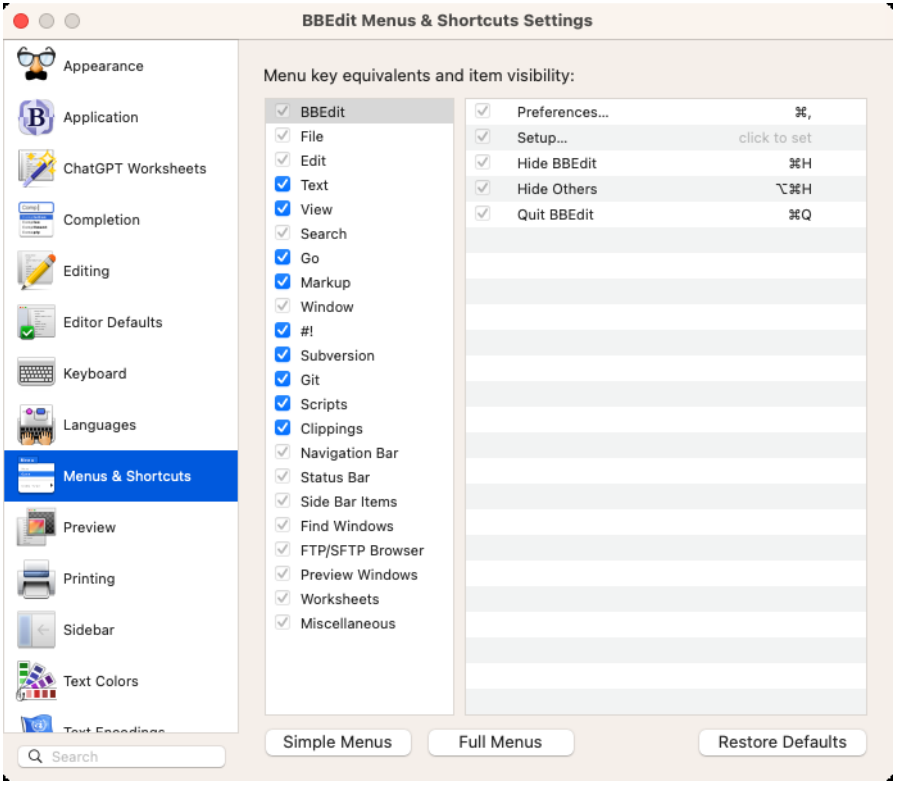
Many other BBEdit features can have keyboard shortcuts assigned as well. Here's how to set them:

Feature	Set Keys in...
Menu commands	Menus & Shortcuts settings panel
Clippings	Clippings palette
Text filters	Menus & Shortcuts settings panel, or the Text Filters palette
Scripts	Menus & Shortcuts settings panel, or the Scripts palette
Stationery	Menus & Shortcuts settings panel

To display any of BBEdit's floating palette windows, use the Palettes submenu in the Window menu.

Assigning Keys to Menu Commands

You can assign your own keyboard shortcuts (key equivalents) to any of BBEdit's menu commands, as well as items on the Text Options popover, and the Markers and Line Breaks popup menus, by choosing Settings from the BBEdit menu to bring up the Settings window, then selecting the Menus & Shortcuts settings panel.



To set the key equivalent for a menu command, locate and select the entry for the command under the appropriate menu section, then double-click on the right-hand part of the line containing that command, and type the desired keystroke. To remove an existing key equivalent from a command, double-click on the existing key combination and press the Delete key. To restore all key equivalents to their default values (as listed in this Appendix), click the Restore Defaults button.

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Listing by Menu and Command Name

BBEdit Menu

About BBEdit	
Settings	Cmd-,
Setup	
Folders	*(submenu)*
Save Workspace...	
Workspaces...	*(submenu)*
License	
Check for Updates...	*(submenu)*
Install Command Line Tools...	
Services	*(submenu)*
Hide BBEdit	Cmd-H
Hide Others/Show All	Cmd-Opt-H
Quit BBEdit	Cmd-Q

File

New	*(see next page)*
New With Stationery	*(submenu)*
Open…	Cmd-O
Open from FTP/SFTP Server...	Cmd-Ctl-O
Open File by Name...	Cmd-D
Reveal Selection	Cmd-Opt-D
Open Counterpart	Cmd-Opt-uparrow
Open Recent	*(submenu)*
Reopen Using Encoding	*(submenu)*
Close Window	Cmd-Shift-W
Close All Windows	Cmd-Opt-W
Close Document	Cmd-W
Close All in Project	Cmd-Ctl-W
Close All Documents	Cmd-Opt-Shift-W
Close & Delete	
Save	Cmd-S
Save All	Cmd-Opt-S
Save As…	Cmd-Shift-S
Save a Copy…	
Save as Styled Text…	
Save as Styled HTML…	
Save to FTP/SFTP Server…	Cmd-Ctl-S
Save a Copy to FTP Server…	Cmd-Opt-Ctl-S
Save as Note	
Revert	
Reload from Disk	
Save Project	
Export	
Hex Dump File	
Hex Dump Front Doc...	
Print…	Cmd-P
Print All	Cmd-Opt-P
Print One Copy	Cmd-Opt-Shift-P
Print Selection	

Edit

Undo <action>	Cmd-Z
Redo <action>	Cmd-Shift-Z
Clear Undo History	Cmd-Ctl-Z
Repeat <last command>	Cmd-Y
Cut	Cmd-X
Cut & Append	Cmd-Shift-X
Cut Line	Cmd-Ctl-X
Cut Line & Append	Cmd-Ctl-Shift-X
Copy	Cmd-C
Copy & Append	Cmd-Shift-C
Copy Line	Cmd-Ctl-C
Copy Line & Append	Cmd-Ctl-Shift-C
Copy as Styled Text	
Copy as Styled HTML	
Paste	Cmd-V
Paste	*(submenu)*
Paste Using Filter	*(submenu)*
Clear	
Select All	Cmd-A
Select	*(submenu)*
Complete	F5
Show Symbol Help	Cmd-Ctl-F5
Show Parameter Help	Ctl-F5
Expand Emmet Abbreviation	Shift-F5
Lines	*(submenu)*
Columns	*(submenu)*
Insert	*(submenu)*
Copy Path	*(submenu)*
Show Clipboard	
Previous Clipboard	Ctl-[
Next Clipboard	Ctl-]
Text Options…	Cmd-Opt-,
Normalize Options…	
Spelling	*(submenu)*
Start Speaking	
Send Command	*Ctl-Return*

Start Dictation...	*(OS-managed)*
Emoji & Symbols	*(OS-managed)*

File --> New

Text Document	Cmd-N
(with selection)	
(with Clipboard)	
Text Window	Cmd-Shift-N
Text File	Cmd-Ctl-Shift-N
HTML Document...	Cmd-Ctl-N
Note	Cmd-Opt-N
(with selection)	
(with Clipboard)	
ChatGPT Worksheet	
Disk Browser	Cmd-Opt-N
FTP/SFTP Browser	
Notebook...	
Project...	
Pattern Playground	
Worksheet	
Text Factory	

Edit --> Paste

and Select	Opt-Cmd-V
Previous Clipboard	*Shift-Cmd-V*
and Match Indentation	Opt-Shift-Cmd-V
Column	Ctl-Cmd-V
Using Last Filter	
HTML	
HTML and Select	
File Paths	
File URLs	

Edit --> Select

Word	
Line	Cmd-L
Paragraph	Cmd-Opt-L
Up	Ctl-Shift-Up arrow
Down	Ctl-Shift-Dn arrow
Highlighted Matches	
Live Search Results	
Clear List Selection	Cmd-Shift-A

Edit --> Lines

New Line Before Paragraph	Cmd-Shift-Return
New Line After Paragraph	Cmd-Return
Move Line Up	Ctl-Up arrow

Move Line Down	Ctl-Dn arrow
Delete Line	Ctl-Shift-Delete
Duplicate	Cmd-Shift-D

Edit --> Columns

Cut Columns
Copy Columns
Clear Columns
New Document with Columns
Rearrange Columns

Edit --> Insert

File Contents...
File/Folder Paths...
Folder Listing...
Emacs Variable Block...
Lorem Ipsum...

(various clipping-derived commands)

Edit --> Copy Path

Copy Path
Copy Full Path
Copy Shell-Escaped Path
Copy URL
Copy Name

Edit --> Spelling

Find Next Misspelled Word	Cmd-;
Find All Misspelled Words	Cmd-Opt-;
Clear Spelling Errors	
Check Spelling as You Type	
Show/Hide Spelling Panel	Cmd-Shift-;

Text

Apply Text Filter	*(submenu)*
Apply Text Filter [last]	
Apply Text Transform	CtlOpt-Shift-A
Run Unix Command	
Run Unix Command	*(submenu)*
Exchange Characters	
Exchange Words	(Option)
Change Case…	
Change Case	*(submenu)*
Shift Left	Cmd-[
Shift Left One Space	Cmd-Shift-[
Shift Right	Cmd-]
Shift Right One Space	Cmd-Shift-]
Un/Comment Lines	Cmd-/
Un/Comment Block	Cmd-Opt-/
Hard Wrap…	Cmd-\
Hard Wrap	Cmd-Opt-\
Add Line Breaks	
Remove Line Breaks	
Educate Quotes	
Straighten Quotes	
Reformat Document	
Reformat Selection	
Add/Remove Line Nums	
Add/Remove Line Numbers	(Option)
Prefix/Suffix Lines…	
Prefix/Suffix Lines	(Option)
Sort Lines…	
Sort Lines	(Option)
Process Duplicate Lines…	
Process Duplicate Lines	(Option)
Process Lines Containing…	
Process Lines Containing	(Option)
Canonize...	
Text Merge	
Increase Quote Level	
Decrease Quote Level	
Strip Quotes	
Zap Gremlins…	
Zap Gremlins	(Option)
Convert Spaces to Tabs…	
Convert Spaces to Tabs	(Option)
Convert Tabs to Spaces…	
Normalize Line Endings	
Precompose Unicode	
Decompose Unicode	
Strip Diacriticals	

View

Text Display	*(submenu)*
Window Appearance	*(submenu)*
Balance	Cmd-B
Balance & Fold	Cmd-Shift-B
Fold Selection	
Unfold Selection	
Collapse Enclosing Folds	
Collapse Top-Level Folds	
Collapse All Folds	
Collapse All Folds Below Level	*(submenu)*
Re-center Selection	
Expand All Folds	
Previous Document	Cmd-Opt-[
Next Document	Cmd-Opt-]
Move to New Window	Cmd-Opt-O
Open in Addtl Window	
Merge Windows	
Get Info	
Show in Finder	
Show in Project List	
Go Here in Terminal	
Go Here in Disk Browser-	
Enter Full Screen	

View->Text Display

Show/Hide Fonts	Cmd-T
Soft Wrap Text	
Show/Hide Page Guide	
Show/Hide Tab Stops	
Show/Hide Line Numbers	
Show/Hide Gutter	
Show/Hide Issues	
Show/Hide Invisibles	
Show/Hide Spaces	
Show/Hide Tabs	
Show/Hide Line Breaks	
Actual Size	
Zoom In/Zoom Out	
Split Text View	

View -> Window Appearance

Show/Hide Navigation Bar	
Show/Hide Editor	
Show/Hide Sidebar	Cmd-0
Show/Hide Open Docs	
Show/Hide Worksheet & Scratchpad	

Search->Find Differences

Compare Two Front Windows	
Compare Against Disk File	
Compare Against Previous Version…	
Apply to Left	Cmd–left arrow
Apply to Right	Cmd–right arrow
Compare Again	

Search

Find…	Cmd-F
Multi-File Search	Cmd-Shift-F
Search in [Document's Folder]	
Search in [Project or Disk Browser]	
Live Search	Cmd-Opt-F
Find Next	Cmd-G
Find Previous	Cmd-Shift-G
Find All	Cmd-Opt-G
Find & Select All	Cmd-Opt-Ctl-G
Extract	Cmd-Ctl-G
Find Selected Text	
Find Previous Selected Text	Cmd-Shift-H
Search for [selected text] in [location]	
Use Selection for Find	Cmd-E
Use Selection for Find (grep)	Cmd-Shift-E
Use Selection for Replace	Cmd-Opt-E
Use Selection for Replace (grep)	Cmd-Opt-Shift-E
Replace	Cmd-=
Replace All	Cmd-Opt-=
Replace All in Selection	Cmd-Ctl-=
Replace to End	Cmd-Shift-=
Replace & Find Next	
First Occurrence of Selected Text	
Next Occurrence of Selected Text	Ctl-G
Previous Occurrence of Selected Text	Ctl-Opt-G
Find Differences…	
Find Differences	*(submenu)*
Find Definition	Cmd-hyphen
Find in Documentation	Cmd-Shift-hyphen
Find References to Selected Symbol	
Find Symbol in Workspace	

Go

Line Number…	Cmd-J
Line Number	Cmd-Opt-J
Center Line	Cmd-Shift-J
Named Symbol	Cmd-Ctl-J
Functions	
Reveal Start	
Reveal End	
Go to Previous	
Go to Next	
Go to Declaration	
Go to Definition	
Markers	
Jump Points	
Previous	
Next	
Set	
Previous Error	Cmd-Ctl–uparrow
Next Error	Cmd-Ctl–dnarrow
Previous Conflict	
Next Conflict	
Previous Placeholder	Ctl-Shift-`
Next Placeholder	Ctl-`
Commands...	Cmd-Shift-U

Markup

See "Markup" on page 420.

Window

Minimize Window	Cmd-M
Minimize All Windows	Cmd-Opt-M
Bring All to Front	
Palettes	(*submenu)*
Rescued Documents	
Show Scratchpad	
Show Unix Worksheet	
Save Default [type of] Window	
Cascade Windows	
Tile Two Front Windows	(Option)
Arrange	(*submenu)*
Move to [Display]	
Cycle Through Windows	Cmd-`
Cycle Through Windows Backwards	Cmd-Shift-`
Exchange With Next	
Synchro Scrolling	
(Open windows)	

Window -> Palettes

- Show/Hide Palettes
- Character Inspector
- Clippings
- Colors
- Minimap
- Scripts
- Text Filters
- Windows
- HTML Markup Tools
  - CSS
  - Entities
  - Font Style Elements
  - Utilities

Shebang (#!)

Check Syntax
Run
Run with Options...
Run in Terminal
Run in Debugger
Show Module/POD Documentation

Git

Check Out Branch...
Add
Discard Changes...
Remove...
Remove from Index... (Option)
Stash...
Apply Stash...
Pop Stash...
Commit...
Commit Staged Changes...
Commit Working Copy...
Show Working Copy Status...
Compare Revisions...
Compare Arbitrary Revisions...
Compare with Staging
Compare with Previous
Compare with Head
Fetch from Remote...
Pull from Remote...
Push to Remote...
Show Blame
Show Revision History
Open Log File

Scripts

Open Script Editor
Open Scripting Dictionary
Open Scripts Folder
(Installed scripts & factories)

Clippings

Open Clippings Folder
Insert Clipping…
Save as Clipping…
Save Selection as Clipping…
(Available clippings)

Help

Search
BBEdit Help
User Manual
Change Notes
Service and Support
Sign Up for News

Toolbar *(in-window)*

Text Options	(popup menu)
Soft Wrap Text	
Show/Hide Page Guide	
Show/Hide Tab Stops	
Show/Hide Line Numbers	
Show/Hide Gutter	
Show/Hide Invisibles	
Show/Hide Spaces	
Use Typographer's Quotes	
Auto-Expand Tabs	

Navigation Bar *(in-window)*

Open Text Options	
Open Files Menu	Ctl-Opt-F
Open Function Menu	Ctl-Opt-N
Open Related Items Menu	Ctl-Opt-I
Open Issues Panel	Ctl-Opt-C
Open Marker Menu	Ctl-Opt-M

Markers	(popup menu)
Set Marker…	
Set Marker	(Option)
Clear Markers…	
Clear All Markers	(Option)
Find & Mark All…	
Find & Mark All	(Option)

Status Bar *(in-window)*

Open Language Menu	
Open Text Encodings Menu	
Open Breaks Menu	Ctl-Opt-B
Line Breaks	(popup menu)
Macintosh	
Unix	
DOS	

Miscellaneous Commands

Zoom Window	
Zoom All Windows	
Zoom Window Full Screen	
Zoom All Windows Full Screen	
Open URL	(Cmd-click in URL)

Markup

Edit Markup…	Cmd-Ctl-M
Close Current Tag	
Balance Tags	Cmd-Opt-B
Document Type…	
Character Set...	
CSS	*(see below)*
Body Properties…	
Head Elements	*(see below)*
Block Elements	*(see next column)*
Lists	*(see next column)*
Tables	*(see page 422)*
Forms	*(see page 422)*
Inline	*(see page 422)*
Phrase Elements	*(see page 423)*
Font Style Elements	*(see page 423)*
Frames	*(see page 423)*
Check	*(see page 423)*
Update	*(see page 424)*
Includes	*(see page 424)*
Utilities	*(see page 424)*
Tidy	*(see page 424)*
Preview in BBEdit	Cmd-Ctl-P
Refresh BBEdit Preview	
Show Document	
Preview in <Default Browser>	
Preview in	*(see page 424)*

Markup --> CSS

@import…
@media…
Box…
    Padding…
    Border…
    Margins…
Layout…
    Position…
    Size & Constraints…
    Clipping…
    Effects…
Background…
Font…
List Style…
Text…
Format

Markup --> Head Elements

Base…
Link…
Meta…
Script…
Noscript
Style…

Markup --> Block Elements

Paragraph…
Paragraph (Option)
Div…
Horizontal Rule…
Horizontal Rule (Option)
H1
H2
H3
H4
H5
H6
Address
Blockquote…
Center
Deleted Text…
Inserted Text…
Noscript
Preformatted

Markup --> Lists

Unordered
Ordered
Definition
Menu
Directory
List Item
List Items

Markup --> Tables

Table…	Cmd-Ctl-T
Row…	
Row	(Option)
TD…	
TD	(Option)
TH…	
TH	(Option)
Caption	
Colgroup…	
Col…	
THead…	
TFoot…	
TBody…	
Create Table Shell...	
Convert to Table…	

Markup --> Forms

Form…
Button…
Field Set
Legend…
Input…
Label…
Select…
Option Group…
Option…
Text Area…

Markup --> Inline

Anchor…	Cmd-Ctl-A
Image…	Cmd-Ctl-I
Applet…	
Object…	
Param…	
Script…	
Map…	
Area…	
Break…	Cmd-Ctl-B
Break	
Font…	
Base Font…	
Bidirectional Override…	
Quotation	
Span…	
Subscript	
Superscript	

Markup --> Phrase Elements

Abbreviation
Acronym
Citation
Computer Code
Defined Term
Deleted Text…
Emphasis
Inserted Text…
Input Text (Kbd)
Sample Output
Strong Emphasis
Variable

Markup --> Font Style Elements

Big
Small
Bold
Italic
Strike-Through
Teletype Text
Underline

Markup --> Frames

Frame Set…
Frame…
No Frames

Markup --> Check

Document Syntax	Cmd-Ctl-Y
Document Links	Cmd-Ctl-K
Folder Syntax…	
Folder Links…	
Site Syntax…	
Site Syntax	(Option)
Site Links…	
Site Links	(Option)

Markup --> Update

Document	Cmd-Ctl-U
Folder…	
Site	(Option)
Document Images…	
Document Images	(Option)
Folder Images…	
Site Images…	
Site Images	(Option)
Re-Deploy Entire Site	(Option)

Markup --> Includes

Persistent Include…
Include…
Placeholders…

Markup --> Utilities

Format…	Cmd-Opt-Shift-F
Format	
Optimize	
Translate Text to HTML	
Translate HTML to Text	
Remove Comments	
Remove Markup	
Raise Tag Case	
Lower Tag Case	

Markup --> Tidy

Clean Document...
Clean Document
Reflow document...
Reflow document
Convert to XHTML
Convert to XML
Check Accessibility...
Check Accessibility

Markup --> Preview In

New Text Window
in All Running Browsers
(Installed browsers)

Listing by Default Key Equivalent

Key	Command
Cmd-0	View: Show/Hide Files
Cmd-A	Edit: Select All
Cmd-B	Text: Balance
Cmd-C	Edit: Copy
Cmd-D	File: Open Selection *or* File: Open File by Name
Cmd-E	Search: Use Selection for Find
Cmd-F	Search: Find…
Cmd-G	Search: Find Again
Cmd-H	Search: Find Selection *or* BBEdit: Hide BBEdit
Cmd-J	Go: Line Number…
Cmd-L	Edit: Select Line
Cmd-N	File: New: Text Document
Cmd-O	File: Open…
Cmd-P	File: Print…
Cmd-Q	BBEdit: Quit BBEdit
Cmd-S	File: Save
Cmd-T	View: Text Display: Show/Hide Fonts
Cmd-V	Edit: Paste
Cmd-W	File: Close Document/Close Window
Cmd-X	Edit: Cut
Cmd-Y	Edit: Repeat Last Command
Cmd-Z	Edit: Undo
Cmd-,	BBEdit: Settings
Cmd-`	Window: Cycle Through Windows
Cmd--	Search: Find Definition
Cmd-;	Text: Find Next Misspelled Word
Cmd-[	Text: Shift Left
Cmd-]	Text: Shift Right

Key	Command
Cmd-/	Un/Comment Lines
Cmd-=	Search: Replace
Cmd-\	Text: Hard Wrap…
Cmd–left arrow	Search: Apply to New
Cmd–right arrow	Search: Apply to Old
Cmd-Return	New Line After Paragraph
Cmd-Ctl-A	Markup: Inline: Anchor…
Cmd-Ctl-B	Markup: Inline: Break…
Cmd-Ctl-F	(Switch to/out of full screen mode)
Cmd-Ctl-I	Markup: Inline: Image…
Cmd-Ctl-J	Go: Named Symbol…
Cmd-Ctl-K	Markup: Check: Document Links
Cmd-Ctl-L	Markup: Lists: List…
Cmd-Ctl-M	Markup: Edit Markup...
Cmd-Ctl-N	File: New: HTML Document…
Cmd-Ctl-O	File: Open from FTP/SFTP Server…
Cmd-Ctl-P	Markup: Preview in BBEdit
Cmd-Ctl-S	File: Save to FTP/SFTP Server…
Cmd-Ctl-T	Markup: Tables: Table…
Cmd-Ctl-U	Markup: Update: Document
Cmd-Ctl-V	Edit: Paste Column
Cmd-Ctl-W	File: Close All in Project
Cmd-Ctl-X	Edit: Cut Line
Cmd-Ctl-Y	Markup: Check: Document Syntax
Cmd-Ctl-Z	Edit: Clear Undo History
Cmd-Ctl-,	Edit: Document Options
Cmd-Ctl-=	Search: Replace All in Selection
Cmd-Ctl–Down arrow	Go: Previous Error
Cmd-Ctl–Up arrow	Go: Next Error

Key	Command
Cmd-Ctl-Shift-C	Edit: Copy Line & Append
Cmd-Ctl-Shift-N	File: New: Text File...
Cmd-Ctl-Shift-S	File: Save a Copy to FTP Server…
Cmd-Ctl-Shift-X	Edit: Cut Line & Append
Cmd-Opt-B	Markup: Check: Balance Tags
Cmd-Opt-D	File: Reveal Selection
Cmd-Opt-E	Search: Use Selection for Replace (grep)
Cmd-Opt-F	Search: Live Search
Cmd-Opt-G	Search: Find All
Cmd-Opt-H	BBEdit: Hide Others
Cmd-Opt-J	Go: Line Number
Cmd-Opt-L	Edit: Select Paragraph
Cmd-Opt-M	(Minimize all windows)
Cmd-Opt-N	File: New: Disk Browser
Cmd-Opt-O	View: Move to New Window
Cmd-Opt-P	File: Print All
Cmd-Opt-R	Compiler: Run
Cmd-Opt-S	File: Save All
Cmd-Opt-V	Edit: Paste & Select
Cmd-Opt-W	File: Close All Windows
Cmd-Opt-,	Edit: Text Options
Cmd-Opt-;	Edit: Find All Misspelled Words
Cmd-Opt-[	View: Previous Document
Cmd-Opt-]	View: Next Document
Cmd-Opt-=	Search: Replace All
Cmd-Opt-/	Un/Comment Block
Cmd-Opt-\	Text: Hard Wrap
Cmd-Opt–up arrow	File: Open Counterpart

Key	Command
Cmd-Opt-Shift-E	Search: Use Selection for Replace (grep)
Cmd-Opt-Shift-F	Markup: Utilities: Format…
Cmd-Opt-Shift-P	File: Print One Copy
Cmd-Opt-Shift-V	Edit: Paste : Match Indentation
Cmd-Opt-Shift-W	File: Close All Documents
Cmd-Shift-A	Edit: Select None
Cmd-Shift-B	View: Balance & Fold
Cmd-Shift-C	Edit: Copy & Append
Cmd-Shift-D	Edit: Duplicate
Cmd-Shift-E	Search: Use Selection for Find (grep)
Cmd-Shift-F	Search: Multi-File Search
Cmd-Shift-G	Search: Find Previous
Cmd-Shift-J	Search: Go to Center Line
Cmd-Shift-N	File: New: Text Window
Cmd-Shift-S	File: Save As…
Cmd-Shift-U	Go: Commands...
Cmd-Shift-V	Edit: Paste Previous Clipboard
Cmd-Shift-W	File: Close Window *{special}*
Cmd-Shift-X	Edit: Cut & Append
Cmd-Shift-Z	Edit: Redo
Cmd-Shift-`	Misc.: Cycle Through Windows Backwards
Cmd-Shift--	Search: Find in Documentation
Cmd-Shift-;	Text: Show Spelling Panel
Cmd-Shift-[	Text: Shift Left One Space
Cmd-Shift-]	Text: Shift Right One Space
Cmd-Shift-Return	New Line Before Paragraph

Key	Command
Ctl-`	Go: Next Placeholder
Ctl-[	Edit: Previous Clipboard
Ctl-]	Edit: Next Clipboard
Ctl-Tab	(switch focus to next pane)
Ctl-Down arrow	Edit: Lines: Move Line Down
Ctl-Up arrow	Edit: Lines: Move Line Up
Ctl-Opt-C	Navigation Bar: Open Counterpart Menu
Ctl-Opt-F	Navigation Bar: Open Files Menu
Ctl-Opt-I	Navigation Bar: Open Includes Menu
Ctl-Opt-M	Navigation Bar: Open Marker Menu
Ctl-Opt-N	Navigation Bar: Open Function Menu
Ctl-Shift-`	Go: Previous Placeholder
Ctl-Shift-Delete	Edit: Delete Line
Ctl-Shift-Dn arrow	Edit: Select Down
Ctl-Shift-Up arrow	Edit: Select Up
Shift-F5	Edit: Expand Emmet Abbreviation

APPENDIX

B

Editing Shortcuts

In BBEdit you can perform many editing functions (including word selection or deletion) directly from the keyboard. Chapter 4 contains complete details on BBEdit's text editing features. This appendix provides a quick reference to available keyboard and mouse shortcuts for word selection and deletion.

In this appendix

Mouse Commands

	No Modifier	Shift
Click	move insertion point	extend selection
Double-click	select word	extend selection to word
Triple-click	select line	–none–

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

Holding down the Command or Option keys as you click or double-click triggers special actions:

	Option	Command	Command/ Option
Click	–none–	Open URL	–none–
Double-click	–none–	–none–	find next instance of the selected text

Arrow and Delete Keys

You can use the arrow keys to move the insertion point right, left, up, and down. You can augment these with the Command and Option keys to move by word, line, or screens, or with the Shift key to create or extend selections. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

You can hold down the Control key while using the arrow keys to scroll through editing windows without moving the position of the insertion point.

Key	Modifier	Action
(left/right) Arrow		Move 1 character left/right
(left/right) Arrow	Option	Move 1 word left/right
(left/right) Arrow	Command	Move to beginning/end of line
(left/right) Arrow	Control	Jump to the previous/next character transition from lower case to upper case OR the next word boundary
(up/down) Arrow		Move up/down 1 line in file
(up/down) Arrow	Command	Move to top/bottom of file
(up/down) Arrow	Option	Move to previous/next screen page
(up/down) Arrow	Control	Edit: Lines: Move Line Up/Down*
[any Arrow key]	Shift	Make or extend a selection range
Delete		Deletes selection range, or character preceding (to the left of) the insertion point.
Delete	Command	Deletes all characters backwards to beginning of line
Delete	Option	Deletes all characters back to beginning of word
Delete	Shift	(same as Forward Delete)
Forward Delete		Deletes selection range, or character after (to the right of) the insertion point
Forward Delete	Command	Deletes all characters forward to end of the current line
Forward Delete	Option	Deletes all characters forward to end of word
Forward Delete	Shift	(same as Forward Delete alone)

Note *: By default, the Control-Up arrow and Control-Down arrow key shortcuts invoke the Move Line Up and Move Line Down commands respectively. If you clear either or both these shortcuts via the Menus & Shortcuts settings pane, then these gestures will instead scroll the text view up or down.

Emacs Key Bindings

The Keyboard settings panel contains an option labelled Use Emacs Key Bindings. When this option is on, BBEdit will accept the following Emacs-style keyboard navigation commands. The Escape key is used in lieu of the Emacs “Meta” key; to type these key equivalents, press and release the Escape key followed by the specified letter key—for example, to type “Esc-V” press and release the Escape key and then type the letter V.

Key Sequence	Action
Ctl-A	beginning-of-line (Move insertion point to start of current line)
Ctl-B	backward-char (Move insertion point backward 1 place)
Ctl-D	delete-char (Delete forward 1 character)
Ctl-E	end-of-line (Move insertion point to end of current line)
Ctl-F	forward-char (Move insertion point forward 1 place)
Ctl-G	keyboard-quit (cancel pending arguments)
Ctl-H	backspace (Delete)
Ctl-K	kill-line (Cut)
Ctl-L	recenter (Scrolls the current view so the selection is centered on screen)
Ctl-N	next-line (Move insertion point down one line)
Ctl-O	open-line (Inserts line break without moving insertion point)
Ctl-P	previous-line (Move insertion point up one line)
Ctl-R	isearch-backward (Live Search backward)
Ctl-S	isearch-forward (Live Search forward)
Ctl-T	transpose-chars (Exchange Characters)
Ctl-U	universal-argument (*See note below*)
Ctl-V	scroll-up (Page down)
Ctl-W	kill-region (Cut)
Ctl-Y	yank (Paste)
Ctl-_	undo (Undo)

Key Sequence	Action
Ctl-X Ctl-C	save-buffers-kill-emacs (Quit)
Ctl-X Ctl-F	find-file (Open file)
Ctl-X Ctl-S	save-buffer (Save current document)
Ctl-X Ctl-W	write-file (Save As)

Esc-<	beginning-of-buffer (Move insertion point to start of document)
Esc->	end-of-buffer (Move insertion point to end of document)
Esc-Q	fill-paragraph (Hard Wrap with current settings)
Esc-T	transpose-words (Exchange Words)
Esc-V	scroll-down (Page up)
Esc-W	copy-region-as-kill (Copy)
Esc-Y	yank-pop (Paste Previous Clipboard)

Using universal-argument

The universal-argument command (Ctl-U) does not work quite the same way as it does in Emacs. In BBEdit, it is a simple repeat-count. For example, if you type Ctl-U, then a 3, and then Ctl-N, the insertion point will move down three lines. There is no visual feedback as you type the number, and no way to backspace or otherwise edit the number. If you make a mistake, the best you can do is type Ctl-G (keyboard-quit) and start over.

vi Key Bindings

BBEdit now offers basic keyboard emulation for `vi` which you can enable by setting the corresponding option in BBEdit's "Keyboard" settings. This enables a basic set of 'vi' navigation and editing commands and modes.

NOTE Please note that ':q' will close the active document rather than quitting BBEdit.

When 'vi' emulation is enabled, mode and command status are displayed in the status area immediately below the text area (and just to the right of the cursor position). If you want that space back, you can hide it via BBEdit's "Appearance" settings.

APPENDIX

C Placeholders and Include Files

This appendix lists the placeholder tokens used by BBEdit templates and include files, and describes the use and capabilities of include files.

In this appendix

Placeholders

Placeholders are processed under the following circumstances:

- When a new HTML document is created from a template, the placeholders in the template are replaced with their current values. (The new document receives the substituted text; the original template file is not modified.)
- When the Update Document command (part of the HTML Tools) is invoked, any placeholders in the documents being updated are replaced with their current values. (Since the placeholders are replaced, subsequent updates *do not* update the substituted text.) Although this command is part of the HTML Tools, it can be used in any document whenever you want to use placeholders.
- When a file is included in another file using the #bbinclude directive (or a related directive), any placeholders in the included file are replaced with their current values before the text is included. (The include file itself is not changed, only the included text is substituted.) All of the above methods of invoking placeholders can also invoke included files, which can have placeholders of their own.

Note The placeholders described in this chapter are only available for use with the HTML Tools' Update command. They cannot be used with BBEdit's Clippings command, nor can Clippings placeholders be used in include or template files.

BBEdit supports the following placeholders. Placeholders are not case-sensitive.

Placeholder	Replaced By…
#ABBREVDATE#	Abbreviated date—e.g., Sun, Aug 16, 2009
#BASE#	The BASE tag as entered using the New HTML Document command
#BASENAME#	The name of the file stripped of its rightmost period-delimited portion. For example, if the file is named "test.html", the base name is "test", while if the file is named "test.foo.html", the base name is "test.foo".
#BASE_URL#	The value of the BASE URL specified in an HTML document's header (useful if you want to refer to the document's location on the server)
#CHARSET#	The character set specified in the New HTML Document command
#COMPDATE#	Compact Date format—e.g., 16-Aug-09
#CREATIONDATE#	The creation date of the current file—e.g., 16-Aug-09
#CREATIONTIME#	The creation time of the current file, formatted according to your Format settings in the International panel of the System Settings
#DATETIME XXX#	Inserts a localized, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below)
#DATETIME_GMT XXX#	Inserts the universal, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below)
#DIRPATH#	The path on the server as specified in the General panel of the Site Settings sheet. Strips any leading slash from the path string
#DOCSIZE#	The size of the current document plus included images in bytes
#DOCTITLE#	The title of the current document as extracted from the <TITLE> tag
#DONT_UPDATE#	Marks a document so that the HTML Update tool will ignore it during processing
#FILE_EXTENSION#	The filename extension for the file (determined as the rightmost period-delimited portion of the filename, without the period). For example, whether the file is named "test.html" or "test.foo.html", the filename extension is "html".
#FILENAME#	The file name of the current file
#GENERATOR#	Generator name used for "Give BBEdit Credit" in New HTML Document function (e.g., "BBEdit 15")

Placeholder	Replaced By…
#GMTIME YYY#	The current GMT time formatted according to the parameters YYY (see "Time Formats" below)
#LANGUAGE#	The language specified in the New HTML Document command, in format (space)lang="en"
#LINK#	The LINK tag as entered using the New HTML Document command
#LIPSUM [options]#	Lipsum text generated according to the options provided (see "Lipsum Options" below)
#LOCALPATH#	The full local path to the current file
#LOCALTIME YYY#	The current local time formatted according to the parameters YYY (see "Time Formats" below)
#LONGDATE#	Long Date format—e.g., Sunday, September 27, 2014
#MACHINE#	The machine name as specified in the Sharing section of the System Settings.
#META#	Any META tag entered using the New HTML Document command
#MODIFIEDDATE#	Modification date of the current file—e.g., 16-Aug-09
#MODIFIEDTIME#	Modification time of the current file, in the format specified in the International section of the System Settings
#MONTHDAYNUM#	Numeric value of the day of the month
#MONTHNUM#	Numeric value of the current month
#PATH#	Path to access your documents from the web server, as specified in your project's website configuration
#PREFIX#	As #DIRPATH# but does not strip the leading slash of the path
#REAL_URL#	The real URL for the current document in its current location
#RELATIVE#	The relative path from the current file back up to the Local Server Root (inserts a path of the form "../../" to tell the browser to "back up" to the site's root directory)
#ROOT#	Path to the Local Site Root, as specified in your project's website configuration
#ROOTPATH#	The current file's path relative to the Local Server Root specified in your project's website configuration
#SERVER#	URL of your Web server, as specified in your project's website configuration

Placeholder	Replaced By…
#SHORTDATE#	Short Date. Day, month, year—e.g., 08/16/09
#SHORTUSERNAME#	Returns the login (short) name instead of the full user name
#TIME#	Current time, according to your Format settings in the International panel of the System Settings
#TITLE#	The title of the current document as extracted from the <TITLE> tag; synonymous with the #DOCTITLE# placeholder
#USERNAME#	The owner name (for the currently logged in user)
#YEARNUM#	The current year—e.g., 2009

Default Date and Time Formats

The following placeholders, when used without explicit date or time format strings will now insert a generically formatted date and time: #DATETIME#, #DATETIME_GMT#, #DATETIME_UTC#, #LOCALTIME#, and #GMTIME#.

Date Formats

The #DATETIME XXX# and #DATETIME_GMT XXX# placeholders allow you to insert the corresponding date and time values with flexible formatting.

In order to use these placeholders, you must substitute XXX with an ICU date/time format string. ICU is the mechanism used by Mac OS X for date formatting. For full details, please refer to the section “Formatting Dates and Times” in the ICU documentation:

`https://unicode-org.github.io/icu/userguide/format_parse/datetime/`

Examples:

#DATETIME EEE, MMM d, yy 'at' h:mm a#

produces:

Tue, Jul 3, 09 at 5:48 PM

#DATETIME_GMT EEE, MMM d, yy 'at' h:mm a#

produces:

Tue, Jul 3, 09 at 9:49 PM

#DATETIME EEEE 'at' h 'o"clock' a#

produces:

Tuesday at 5 o'clock PM

Time Formats

The #GMTIME YYY# and #LOCALTIME YYY# placeholders offer you the option to insert the specified time value with flexible formatting.

In order to use these placeholders, you must substitute YYY with a time format using the same expansion options offered by the ‘strftime’ routine (see ‘man strftime’ for further details).

Examples:

#LOCALTIME %r %z on %A#

produces:

06:50:13 PM -0400 on Monday

#GMTIME %r %z#

produces:

10:50:13 PM +0000

Lipsum Options

The #LIPSUM [options]# placeholder offers you the ability to insert generated lipsum text in flexible quantities.

The option syntax is of this form:

#LIPSUM [wordlist] [units] [units-count] [max-line-length]#

where “wordlist” and “units” are each a single character, and case matters:

- “p’ for paragraphs
- “s” for sentences
- “w” for words
- “S” for startup
- “B” for bacon
- “N” for normal

The case sensitivity allows these to appear in any order, and they can be separated by spaces, commas, semicolons, periods, slashes, colons, or dashes. Thus, the following examples are equivalent and all will generate ten sentences of Startup lipsum with the lines wrapped to 72 characters:

#LIPSUMS,s 10 72#

#LIPSUM Ss 10 72#

#LIPSUM s:S 10 72#

The “wordlist” specifier is optional; if it is not present then BBEdit will generate “normal” lipsum.

A #lipsum#` placeholder may contain zero, one, or two numbers. If no numbers are provided then there is no line breaking and only one unit (word, line, paragraph) is generated.

If one number is provided, then that number of units is generated.

If two numbers are provided, the first is the number of units to be generated, and the second is the maximum line length.

Thus, if you wish to break lines, you must also specify the number of units that should be generated.

Using the #RELATIVE# Placeholder

When dealing with large websites that have multiple content folders, it is often useful to specify relative rather than absolute paths for linking documents. The #RELATIVE# placeholder allows you to easily generate relative references in templates and include files by providing a virtual path that uses the ".." construction to "back up" the hierarchy to the root directory of the site.

To use this placeholder, write your links as if they were all relative to the top of your website, including #RELATIVE# as the first "directory" in the path. For example, consider that you have the following file structure, where each page includes a file which references the separate GIF image.

```
My_Web_Site:
    Folder1:
        File1.html
    Folder2:
        File2.html
        File3.html
    Folder3:
        Folder4:
            Folder5:
                File4.html
    Graphics:
        Buttons:
            my_footer_button.gif
```

If you write a relative link as follows:

```
<img src="#relative#Graphics/Buttons/my_footer_button.gif"
alt="">
```

and then run the Update command, the following links will be generated.

In File1.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File2.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File3.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File4.html,

```
../../../Graphics/Buttons/my_footer_button.gif
```

Include Files

An include file, or just an “include,” is a special form of placeholder whose substitution happens to be the contents of another file. If you have used C or certain other programming languages, you may already be familiar with the concept. Using includes, you can reuse standard bits of text content or HTML markup in several templates or clippings entries without having to revise all of those individual files whenever you revise the included text.

Include File Locations

BBEdit looks for include files first in the same directory as the document containing the directive, then in the same directory as the document into which the processed document is being inserted, and finally in the HTML Templates folder specified by the configuration for the current website in the General section of the Site Settings sheet.

Simple Includes

A simple include takes the following form:

```
#bbinclude "filename"
```

where filename is the full name of the file whose contents you wish to include. When such an include is present in a template or clippings entry, it is replaced with the contents of the specified file when the template is used to build a new document, or when the clippings entry is inserted. (The original template or clippings file is not changed.)

Imagine that you have ten different templates, each of which contains your name, address, phone number, email address, and a copyright statement with the current year in them. Rather than pasting this info into all ten templates, you can create a file named “address.html”, put it in your Templates folder, and include this statement:

```
#bbinclude "address.html"
```

in each of the templates, at the appropriate point. Later, when the new year arrives, or you move, you only have to update one file, not all ten templates. (You could use the #YEARNUM# placeholder for the year and only need to update the include file when you move!)

Headers and footers are probably the most common uses for include files, but any template or clippings entry may use as many include statements as you wish. Included files themselves may also use #bbinclude directives, up to 16 levels deep.

Persistent Includes

Simple includes are appropriate for use situations where you want the inclusion to happen only once. Once the file has been included, however, it cannot be changed in any automated fashion. Since the #bbinclude directive is replaced by the included text, the Update tool cannot tell the included text is any different from any other text.

Includes become even more powerful, however, when you can update existing files to incorporate revised include text at a later date. For example, suppose you generate several dozen HTML documents using a template that uses an #bbinclude directive to insert a standard footer containing your email address. Later, you change your email address. After you change it in the footer document, only *new* HTML files you create from the template will include your new address. What you would really like to be able to do is update all the files you have *already* created to include the revised footer.

Since this capability is needed primarily in website maintenance, BBEdit lets you embed the include directive in an HTML comment. An “end bbinclude” comment is also required. The included text is inserted *between* the two comment markers, but the comments themselves remain in place. The comments are not shown in the browser. This is known as a *persistent* include.

A persistent include looks like this:

```
<!-- #bbinclude "filename" -->
<!-- end bbinclude -->
```

The first time a persistent include is processed, it is handled much like a simple include. However, since the include directives remain in place, and because they mark the beginning and end of the inserted text, the Update tool can “rip out” the obsolete included text and replace it with the updated file. Using persistent includes and the Update Folder or Update Site commands, you can easily make these sorts of changes to entire sites in moments.

IMPORTANT Any changes you have made to the included text after its initial inclusion will be discarded when the persistent include is updated, even if you have not changed the include file.

Inline versus Block Includes

By default, BBEdit places included content in the document as a block, to ensure that it does not occupy the same line as the include directives. However, if you wish to override this behavior and have BBEdit place the included content inline, you may do so by adding the special option #bbincludeoptions#="inline=true" to the include directive.

```
<!-- #bbinclude "filename" #bbincludeoptions#="inline=true" -->
<!-- end bbinclude -->
```

Include Files with Variables

Include files can be extended even further through the use of variables, which provide a means of inserting arbitrary text when the included file is processed, so that not all instances of the included file are exactly the same. Variables are essentially placeholders that you make up yourself. Some possible uses are to insert names, taglines, alt strings for images, or file names (for files other than the current document) into documents.

Note A variable name consists of a string of alphanumeric characters, enclosed in number signs (the ‘#’ character). Spaces are not allowed in variable names, but underscores may be used to represent word breaks.

Variables can be placed anywhere in an include file, just like placeholders. When you include that file in a document, you specify the variable names and values with it. Consider an include file named "footer.html", which contains the following

```
<HR>
<IMG SRC="#MY_GRAPHIC#" ALT="#MY_ALT_DESC#">
<H1>#MY_TITLE#</H1>
<BIG>This document copyright 1998-2009 by Sid Zookim.</BIG>
```

In your document, the Include reference would look like this:

```
…
<BODY>
…
<!-- #bbinclude "footer.html"
#MY_GRAPHIC#="test1.gif"
#MY_ALT_DESC#="a test image"
#MY_TITLE#="A Test Title"
-->
<!-- end bbinclude -->
…
</BODY>
…
```

Note that the values of placeholders are specified *inside* the HTML comment of a persistent include, using a #PLACEHOLDER#=”Value” syntax. The quote marks around the value are mandatory; if you need to include a quote mark in the actual value, escape it with a backslash.

Including AppleScripts

BBEdit allows included files to be compiled AppleScript scripts. The script should contain an “on include” handler which is passed two parameters: a reference to the file from which the script is being called, and a record containing one field for each variable passed in the #bbinclude directive. Scripts can of course also retrieve information from BBEdit, other scriptable applications, or the system. The handler’s return value is inserted into the file that included it.

Given the HTML document below:

```
<html>
<head>
    <title>Include Test</title>
    <meta name="generator" content="BBEdit 15">
</head>
<body>

<!-- #bbinclude "foo.script" #x#="3" #author#="JEK"-->

<!-- end bbinclude -->

</body>
</html>
```

The following script inserts three lines: the first containing the file’s path, the second containing the parameter “x” passed to it in the #bbinclude directive, and the third containing the parameter “author.”

```
on include(f, vars)
   set s to f as text
   set s to "File Path:  " & s & return & return as text
   set s to s & "x:  " & x of vars & return & return as text
   set s to s & "Name:  " & author of vars & return as text
   return s
end include
```

The resulting document might look like this:

```
<html>
<head>
    <title>Include Test</title>
    <meta name="generator" content="BBEdit 15">
</head>
<body>

<!-- #bbinclude "foo.script" #x#="3" #author#="JEK"-->
File Path:  Boot:Desktop Folder:incl_test.html

x:  3

Name:  JEK
<!-- end bbinclude -->

</body>
</html>
```

Including Unix Scripts

BBEdit lets you include scripts written in Perl, Python, Ruby, or any other Unix scripting language. The complete path name of the file being processed is passed to the script as its first argument, and any variables in the include statement are passed as additional arguments. (For Perl, all these can be retrieved by your script via @ARGV.)

Any text sent to STDOUT by the script will be taken as the value of the #bbinclude operation and inserted into the HTML file. If an error occurs while running the script, the STDERR output, if any, will be inserted into the file along with the STDOUT, and a single line indicating the error will be added to the error browser.

For example, enter this directive in your HTML file:

```
<!-- #bbinclude "foo.pl" #length#="2" #width#="3" -->
<!-- end bbinclude -->
```

Then use this source code for “foo.pl”, and save it in the same folder as the HTML file, or in the “Templates and Includes” folder specified in this website’s configuration:

```
#!/usr/bin/perl -w
my $file = shift @ARGV;
my %args = @ARGV;
my $area = $args{"length"} * $args{"width"};
print "Filename: $file\n";
print "Area: $area\n";
```

When you run the Update command, BBEdit will place the file name in the script’s variable $file and the “length” and “width” variables in the associative array (hash) %args.

After the update, the BBEdit file will look like this:

```
<!-- #bbinclude "foo.pl" #length#="2" #width#="3" -->
Filename: Mac HD:Desktop Folder:sample.html
Area: 6
<!-- end bbinclude -->
```

In addition, BBEdit will pass information about the current HTML Tools settings to the script in the following environment variables:

```
BBEditServerURL
BBEditServerPath
BBEditDefaultFileName
BBEditTemplateDirectory
BBEditRootDirectory
BBEditLowercaseTags
BBEditLowercaseAttributes
BBEditAlwaysQuoteAttributes
```

To access these in your Perl code, use the %ENV environment variable hash. For example, this line of Perl will print the web server name specified by this website's configuration:

```
print $ENV{BBEditServerURL};
```

Here's an example Python include script.

```
#!/usr/local/bin/python
import os
import string
import sys

print "Hello Python World!"
print "==================="
print "File being updated: ", sys.argv[1]
print

userVariables = {}
for i in range(2, len(sys.argv), 2):
   userVariables[sys.argv[i]] = sys.argv[i+1];

print
print "Dumping the user variables passed to the script"
print "==============================================="
print
keys = userVariables.keys();
keys.sort()
for k in keys:
   print "%-30s %s" % (k, userVariables[k])

print
print "Dumping the environment variables set by BBEdit"
print "==============================================="
print
for k, v in os.environ.items():
   if (string.find(k, 'BBEdit') == 0):
      print "%-30s %s" % (k, os.environ[k])
```

Other Include Notes

IMPORTANT Some old versions of BBEdit supported the use of “#include” as an alternative to “#bbinclude”. However, this syntax made it difficult to mix BBEdit includes and Microsoft Active Server Page (ASP) directives, so it is no longer supported. If you have existing documents which use this syntax, simply change “#include” and “end include” to “#bbinclude” and “end bbinclude” to continue using them.

APPENDIX

D Codeless Language Modules

The information previously contained in this appendix is superseded by the Codeless Language Module Reference on our website:

`https://www.barebones.com/support/develop/clm.html`

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z