

---

# Pair Programming with ChatGPT

## AI-Enhanced Coding for the Modern Developer

Copyright © 2023 by Michael D. Callaghan. All rights reserved.

No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, except for brief quotations used in critical articles and reviews.

Requests for permission to reproduce material from this book should be directed to [michael@walkingriver.com](mailto:michael@walkingriver.com).

This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition, including this condition, being imposed on the subsequent purchaser.

While every effort has been made to ensure the accuracy of the information contained in this book, the author and publisher make no representations or warranties, express or implied, about the completeness, accuracy, reliability, suitability or availability with respect to the information, products, services, or related graphics contained in the book for any purpose. Any reliance you place on such information is therefore strictly at your own risk.

In no event will the author or publisher be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from loss of data or profits arising out of, or in connection with, the use of this book.

If you find any errors or inaccuracies, please contact the publisher.

Cover design by Michael D. Callaghan

---

---

# Preface

## About This Book

This book is designed to be a fun, easy-to-read, and hands-on exploration of using Artificial Intelligence tools as your “pair programming” partner. Throughout the book, I hope you’ll see how you can use ChatGPT to make your own tasks easier and more effective.

Over the length of the book, we’ll cover the following topics:

- Using AI in Software Development
- Shell Scripting Commands
- Git Commands
- Common Algorithms
- Learning RxJS
- Angular’s HttpClient
- Regular Expressions
- Data Generation
- Agile Project Management
- Let’s Build an App!
- Unit Testing
- Other Considerations
- The Future of Software Development

My goal is to inspire you to come up with your own ideas and vastly improve your efficiency.

## Who is This Book For?

This book is for any software developer of any level, from “just starting out” to “ready to retire.” Though most of my examples use specific web technologies like HTML, JavaScript, TypeScript, and Ionic, you don’t need to be familiar with these languages and frameworks to get something out of this book.

---

---

## Following Along

Regardless of the type of software development you do, I'm sure you'll find value by following along in whatever language you prefer.

I'm pleased to offer you a list of most of the prompts I use throughout the book, in case you want to follow along and don't want to type them. You can sign up and download them free of charge here:

<https://walkingriver.gumroad.com/l/pair-programming-bonus>

---

*Those who sign up for my bonus materials will also be eligible for occasional sneak peeks and discounts on future books and other content.*

---

## Updates and Questions

If you ever have questions or just want to be updated about this and future books, feel free to sign up at the link above, or send an email to [michael@walkingriver.com](mailto:michael@walkingriver.com).

I'm also very active on Twitter, where you can find me posting as [@WalkingRiver](https://twitter.com/WalkingRiver).

## Books on Amazon

You can find the rest of my titles on Amazon at <https://amazon.com/author/mcallaghan>.

If you enjoy this book, I'd appreciate you leaving me a positive review on Amazon, which you can do here: <https://www.amazon.com/review/create-review?asin=B0C56TTL2V>

---

---

# Using AI in Software Development



*Image by Gerd Altmann on Pixabay*

I first heard about using AI for programming with the introduction of GitHub Copilot in mid-2022. I was intrigued by the idea that someone trained an Artificial Intelligence (henceforth “AI”) model from the source code inside every public GitHub repo. Further, that once the model had been trained, it could essentially understand my source code and make suggestions in real time. If this were real, I had to try it.

I registered for the free beta and waited. About a week later, I got the email accepting me into the beta program, along with a link to install a Visual Studio Code extension. I installed the extension and started playing with it. It seemed cool but I don’t do a lot of day-to-day coding on my personal machine and wasn’t ready to install Beta software on my work computer. So, I promptly forgot about it for a while.

---

---

Fast forward a few months. ChatGPT showed up and took the world by storm. Here was the first “approachable” AI with a conversational interface. People could ask it questions and expect a coherent answer.

Millions of people began experimenting with it, seeing how well it could write code simply by entering a brief description of the problem. While not perfect, its answers were surprisingly good. People began wondering whether we had entered a strange new world of software development, one where traditional coding, and software developers, would soon be obsolete.

## What We Will Cover

My objective in writing this book is to describe how AI can be used in the day-to-day activities of a typical software developer. Before we dive deep into these concepts, let’s take a high-level look at what the tools can and cannot do for us.

## What Can AI Tools Do?

1. Code generation: AI can be used to generate code based on a set of inputs and requirements, which can speed up the development process and reduce the chance of errors.
2. Test automation: AI can be used to automatically generate test cases and test scripts, which can save time and improve the quality of software.
3. Quality assurance: AI can be used to automatically evaluate the quality of code, which can help identify potential issues before they become problems.
4. UI/UX design: AI can be used to generate UI/UX designs, which can save time and improve the overall user experience.
5. Language Processing: AI can be used for natural language processing, which can be used in natural language interface for software development.
6. Predictive analytics: AI can be used to predict future trends, bugs and areas that need attention.
7. AI-powered search: AI can be used to search and recommend relevant code snippets, libraries and tools to developers.

I will be focusing on the first two items in that list, as they seem most relevant to software development today.

---

---

## What are Their Limitations?

1. Limited understanding of context: AI may not fully understand the context in which code is being written, which can lead to errors or inconsistencies.
2. Lack of creativity: AI can generate code or designs that are functional but may lack the creativity of a human developer.
3. Limited ability to handle complexity: AI may struggle with complex code or designs that involve many variables and interdependencies.
4. Lack of understanding of industry standards and best practices: AI may not be aware of the best practices and standards in a particular industry or field, which can lead to suboptimal results.
5. Lack of flexibility: AI may not be able to adapt to changing requirements or unexpected situations as well as a human developer.
6. Limited ability to handle unstructured data: AI may struggle with data that is not well-organized or does not conform to a specific format.
7. Limited ability to generalize: AI may struggle to generalize from examples and may not work well in situations where it has not been explicitly trained.
8. Data bias: AI models can perpetuate human biases if the training data is not diverse and representative of the population.
9. Dependence on large amounts of data: AI models often require large amounts of data to be trained, which can be difficult to acquire and process.
10. Lack of explainability: AI models can be opaque, making it difficult to understand how they arrived at a particular decision or output. As we will see, this is where ChatGPT shines.

## What is Covered in This Book

There are a lot of AI tools that we could choose. I won't pretend to know even half of them. The two with the biggest market penetration and clout seem to be ChatGPT from OpenAI and Copilot from GitHub. I will focus on ChatGPT-4 throughout the book.

You can check out my other titles to see similar discussions of other tools.

---

---

## ChatGPT

ChatGPT is a language model developed by OpenAI. It is a variant of the GPT (Generative Pre-trained Transformer) model, which is trained on a massive amount of internet text data to generate human-like text. The model is fine-tuned for specific tasks, such as language translation, text summarization, and conversation. With its ability to understand and respond to natural language inputs, ChatGPT can be used for a variety of applications, such as chatbots, virtual assistants, and language-based games.

People discovered very quickly that it can also be used to generate functioning computer code in a variety of languages. ChatGPT, as a powerful language generation model, can be used in several ways to help with software development:

1. Code generation: ChatGPT can be used to generate code snippets and even complete functions based on a set of inputs and requirements, which can speed up the development process and reduce the chance of errors.
2. Text generation: ChatGPT can be used to generate comments, documentation and even commit messages, which can save developers time and improve code readability.
3. Test case generation: ChatGPT can be used to generate test cases and test scripts for software, which can save time and improve the quality of software.

To use ChatGPT, you will need to sign up for an OpenAI API account, which you can do at <https://chat.openai.com/chat>. If you already have an account with OpenAI, you can log in with that. If not, you will need to register.

At the time of this writing, there are two tiers to the service: one is free, but you may find that it is frequently “at capacity,” which makes it hard to use.

There is also a paid tier, ChatGPT PLUS, which is currently \$20 USD per month. OpenAI has also announced plans for additional tiers, as well as a waitlist so you can get more information.

This book will cover GPT-4, which requires a PLUS subscription.

---

---

## How does it Compare to Tools Like Copilot?

ChatGPT and GitHub Copilot are both AI-powered tools that are designed to assist software developers in their work. Both tools use machine learning algorithms to provide developers with intelligent suggestions and assistance, but they are designed for different aspects of the development process.

One of the main differences between the two tools is their area of focus. ChatGPT is focused on natural language processing tasks, while GitHub Copilot is focused on code completion and suggestions. ChatGPT can be used to generate text responses or code, while GitHub Copilot can be used primarily to generate code snippets.

Another difference is the way the two tools are used. ChatGPT is accessed through a chat-like interface (hence the name), so developers tend to be more descriptive in using it to generate code. On the other hand, GitHub Copilot is integrated into the development environment and can be used directly inside your project.

## Controversy - Should Developers Use These Tools?

I thought I should get this out of the way early because there are those who feel the answer is a definite “no.”

One of the main concerns regarding ChatGPT’s training on a broad dataset from the internet is that it may propagate poor coding habits and security vulnerabilities. Since ChatGPT learns from a diverse range of sources, it might inadvertently pick up and promote suboptimal coding practices or security risks that exist in its training data. This could potentially lead developers to inadvertently incorporate these practices into their own code, possibly leading to security breaches or other complications.

Another potential risk is that developers might become overly reliant on ChatGPT’s suggestions, thereby diminishing the significance of their own understanding of the codebase. This dependency could deter developers from acquiring an in-depth knowledge of the code they’re working with, making it more challenging to maintain and troubleshoot issues in the future.

---



---

Moreover, even though OpenAI claims to use certain privacy techniques to protect the individuality of training data and ensure that ChatGPT doesn't remember specific documents, concerns about privacy and data security can arise. While ChatGPT is not designed to recall or generate sensitive data like usernames, passwords, or other personal information, there is always a risk that the AI could unintentionally generate such information if it were present in the training data.

## Licensing Issues

One of the larger criticisms and causes of the controversy stem from licensing issues with the data used to train the AI models. This can be a significant concern, as using code without proper permission or licensing can lead to legal issues. Some specific issues that can arise include:

1. Copyright infringement if the data used to train an AI model is protected by copyright.
2. Trade secret infringement if the data used to train an AI model constitutes someone else's trade secret.
3. If the data used to train an AI model contains personal information, using it without proper permission or anonymizing it may violate privacy laws.
4. If the data used to train an AI model is subject to a license or contract, using it in a way that violates those terms can lead to legal issues.

It's important for developers and organizations to be aware of these issues and to obtain proper permissions and licenses for any data used to train AI models. This can include obtaining licenses for open-source datasets or negotiating agreements with data providers to use proprietary data.

---

---

## No Respector of Licenses

When using AI to help write software, it is important to consider the open-source licenses of the code that may have been used to train the AI. Open-source licenses are agreements that dictate how the code covered by the license can be used, modified, and distributed. Different open-source licenses have different terms and conditions, so it is important to carefully review the licenses of any code that is used to train the AI and ensure that they are being complied with. This may involve obtaining the proper permissions, giving credit to the original authors, and making any required changes to the code. Failing to properly comply with open-source licenses can result in legal consequences, such as being sued for copyright infringement.

A concrete example may illustrate the issue more clearly. In one of my mobile apps, I use an algorithm I found to generate a deck of cards of an arbitrary size. The JavaScript code I found on GitHub (not using any AI) is clearly marked as “GPL Licensed.” As I understand it (and I’m not a lawyer), this means that I am free to use this algorithm in my own code, and can even sell the app. That’s allowed. However, I must include the GPL notice in the source code. I must also release my source code under the same license.

What if I had used an AI that suggested this algorithm to me and I was unaware of its license, and then proceeded to offer my app under a different license? Would I technically be in violation of the GPL from the original algorithm? If so, what risk would this expose me or my employer to? Unfortunately, these types of questions will probably need to be settled in courts. As I write this, there is at least one lawsuit pending regarding GitHub Copilot’s use of licensed software in its recommendations. I am not currently aware of any such lawsuits against OpenAI, but there very well could be.

## Ethical Considerations

What are the ethics of using AI to help with software development? Is it ethical to use an AI to help write software? Can the developer or company really claim that they are the proper authors of said software? This may seem odd, but I asked ChatGPT that very question. This is how it replied.

---

---

It is generally considered ethical to use AI to assist in the development of software, as long as the developers and the company are transparent about the role of the AI in the development process and give credit where it is due. The developer or company can claim authorship of the software but should also acknowledge the contributions of the AI used in the development process. Additionally, it is important to ensure that any AI models used in the development of software are trained on ethically sourced data and do not perpetuate any biases.

I also asked it whether there are any other ethical considerations to be aware of.

First, there is the issue of fairness and bias. It is important to ensure that the AI system is not making decisions based on biased data, as this could result in unfair treatment of certain individuals or groups. Additionally, there is the question of accountability. It is important to consider who is responsible for any errors or mistakes made by the AI system, and to ensure that there are appropriate safeguards in place to prevent such errors from happening. Finally, there is the issue of privacy. It is important to ensure that the personal data of users is collected, used, and stored in a manner that respects their privacy and complies with relevant laws and regulations.

## Conventions Used in This Book

As you see above, I am adopting the convention of printing the output from the AI in a different font and background, so that you can easily see distinguish my thoughts from the AI's.

When I show a prompt to ChatGPT, I will show it like this.

Output used verbatim from ChatGPT will be shown this way.

Also, any commands or generated source code will be shown this way.

---

---

# Shell Scripting Commands



*Image by cocoandwifi from Pixabay*

Shell scripting is a fundamental skill for anyone who wants to efficiently manage and automate tasks on a computer.

Let's investigate how ChatGPT does with shell scripting. A while back I wrote an article about what these symbols (`~|&||&&$!'()`?) mean in a bash (or similar) shell and how to use them. Can ChatGPT give me these types of commands from nothing but comments? Read on to find out.

ChatGPT is ideally suited for these sorts of things, given its conversational nature. Here is how it responded to my first prompt:

---

---

Check for a directory and print a message if it doesn't exist

You can use the **test** command (also known as **[** or **[[**) to check if a directory exists, and the **echo** command to print a message if it doesn't. Here is an example:

```
if test ! -d /path/to/directory; then
    echo "Directory does not exist"
fi
```

Alternatively, you could use the **if [ ! -d "/path/to/directory" ]; then** command with the same echo message.

Another option is to use **if [[ ! -d "/path/to/directory" ]]; then**

As you can see, ChatGPT offers an explanation with its answers and does not simply give you the code. As you will see, this can be a significant advantage to using ChatGPT.

## The find Command

The next thing I tried was to find all the PDF files anywhere in my home directory and below. Normally when I do that, I use the **find** command. Is that what ChatGPT would suggest? I entered the following comment, which led to two different options.

Find all PDF files in my home directory and print the path to each one

You can use the **find** command to search for all PDF files in your home directory and print the path to each one. Here's an example:

```
find ~ -name "*.pdf" -print
```

This will search for all files in your home directory (~) that end in **.pdf** and print the path to each one.

Alternatively, you could use **ls** command with wildcard,

```
ls ~/ -R | grep .pdf$
```

This will recursively list all files in your home directory, and filter only the files with **.pdf** extension.

As before, I appreciate that it provides an explanation and not just the command. If you are trying to learn, this is the better experience.

---

---

## Background Tasks

The next thing I wanted to do was to determine whether the AI knows how to use some of those symbols that often trip up beginners. Would they know that the greater than sign (**>**) is used to redirect output and that the ampersand (**&**) is used to run a command in the background?

I changed my comment as follows, and then gave it the same hint from above. As before, ChatGPT gave me multiple options, complete with thorough explanations.

Find all PDF files below my home folder and write the results to a file called pdfs.txt in the background.

You can use the **find** command to search for all PDF files below your home directory, and the **tee** command to write the results to a file called pdfs.txt in the background. Here's an example:

```
find ~ -name "*.pdf" -print | tee pdfs.txt &
```

This will search for all files in your home directory and its subdirectories that end in **.pdf** and print the path to each one. The output will be piped to the **tee** command, which will write it to a file called pdfs.txt in the current directory. The **&** at the end will run the command in background, so you can continue to use the terminal.

Alternatively, you could use **ls** command with wildcard,

```
ls ~/ -R | grep .pdf$ >> pdfs.txt &
```

This will recursively list all files in your home directory and its subdirectories, and filter only the files with **.pdf** extension, and write it to pdfs.txt file. The **&** at the end will run the command in background as well.

I had forgotten all about the **tee** command. I'm also impressed that it went a little farther with the **ls** command, using **grep** with a simple regular expression to find files that end in **.pdf**. It also assumed I would want to append to my **pdfs.txt** file if it already exists.

Though not exactly what I asked for, it actually gave me more.

## Output Pipe

Next, I wanted to see whether its know how to use the output of one command as the input to another.

---

---

Count the number of PDF files in my home folder in the background and print the result.

To count the number of PDF files in your home folder in the background and print the result, you can use the find command along with the wc command. Here's the command you can use in your zsh shell:

```
find ~/ -iname "*.pdf" -type f | wc -l &
```

Let's break down the command:

`find ~/ -iname "*.pdf"`: This command searches for files with the extension .pdf in your home directory (~). The -iname option is used to make the search case-insensitive.

`-type f` is used to limit the search to only regular files (excluding directories or other types of files).

`|`: This is a pipe symbol that redirects the output of the find command to the input of the next command.

`wc -l`: This command counts the number of lines in its input. By using the -l option, it counts the number of lines, which in this case corresponds to the number of PDF files found.

`&`: This ampersand symbol runs the command in the background.

After executing the command, it will print the number of PDF files found in your home folder. The command will continue running in the background, allowing you to continue using the shell.

## Redirect Errors to Ignore Them

Most people know how to use the `>` character to redirect the output of a command to a file. If the command encounters any errors, those will still display in the terminal, which can be confusing when running the command in the background.

I know it is possible to redirect errors to a special file called `/dev/null`, so that they are ignored completely. Unfortunately, I can never remember the syntax. I don't know why because it isn't that complicated. It's one of my weird blind spots, I suppose.

What's really cool about ChatGPT is that it remembers earlier parts of your conversation. For this one, I changed up the prompt and simply asked it to "rewrite the last command to ignore any errors." The explanation itself is also context-aware.

---

---

Rewrite the last command to ignore any errors

Certainly! To ignore any errors that may occur during the execution of the command and still count the number of PDF files in your home folder, you can use the following modified command:

```
find ~/ -iname "*.pdf" -type f 2>/dev/null | wc -l &
```

In this command, we've added 2>/dev/null to redirect the error output (file not found errors, permission denied errors, etc.) to the null device /dev/null, effectively discarding them.

With this modification, any errors that occur during the execution of the find command will be suppressed, and only the count of PDF files will be printed. The command will continue running in the background.

## \*nix System Administration

I started wondering whether ChatGPT could help me with some common system administration tasks, such as creating users, adding users to groups, and so forth.

### Add a New User Account

I decided to start with something simple, so I asked it to create a new user account for me.

ChatGPT again outdid itself, providing plenty of detail and explanation.

Create a new user account on macos

To create a new user account on macOS, you can use the dscl (Directory Service command line utility) command. Here is an example of how to create a new user account called "newuser" with a password of "newpassword":

```
sudo dscl . -create /Users/newuser
sudo dscl . -create /Users/newuser UserShell /bin/bash
sudo dscl . -create /Users/newuser RealName "New User"
sudo dscl . -create /Users/newuser UniqueID "501"
sudo dscl . -create /Users/newuser PrimaryGroupID 20
sudo dscl . -create /Users/newuser NFSHomeDirectory /Users/newuser
sudo dscl . -passwd /Users/newuser newpassword
```

This creates the account, sets the shell to /bin/bash, sets the RealName, UniqueID, PrimaryGroupID and home directory, and sets the password.

---



Alternatively, you can use the **sysadminctl** command to create a new user account and set the password. Here is an example:

```
sudo sysadminctl -addUser newuser -fullName "New User" -password  
newpassword -admin
```

This creates the account with the username "newuser", full name "New User", password "newpassword" and admin privilege.

You also can use the GUI by going to System Preferences > Users & Groups > click on the lock icon to unlock it > click on the "+" sign to add a new user.

Please note that these commands require administrative privileges and should be used with caution.

I tried the second command on my Mac and it worked flawlessly.

## Make an Existing User an Admin

Next, I asked for the command to make an existing user an admin. ChatGPT gave me a set of options.

Make user 'michael' a local administrator

To make a user "michael" a local administrator on macOS, you can use the **dscl** (Directory Service command line utility) command. Here is an example of how to make the user "michael" a local administrator:

```
sudo dscl . -append /Groups/admin GroupMembership michael
```

This command adds the user "michael" to the "admin" group. Members of the "admin" group have administrative privileges on the system.

Alternatively, you can use the **dseditgroup** command to add a user to the "admin" group. Here is an example:

```
sudo dseditgroup -o edit -a michael -t user admin
```

This command adds the user "michael" to the "admin" group.

You also can use the GUI by going to System Preferences > Users & Groups > click on the lock icon to unlock it > right click on the user you want to change > select "Advanced Options" > change the "Account type" to "Administrator".

---

Please note that these commands require administrative privileges and should be used with caution.

## Backup Files

Have you ever used **rsync** to back up files from one system to another? No? Of course not. No one does (ok, maybe some people do). It's hard to remember the syntax. That's why Dropbox and other such services exist.

ChatGPT gave me four different suggestions, with **rsync** being the first one. It also suggested using **scp**, which I had planned to ask about next.

Sync my home folder to a remote server

There are several ways to sync your home folder to a remote server on macOS. Here are a few options:

**rsync**: This command-line tool can be used to synchronize files and directories between two locations. Here's an example of how to use **rsync** to sync your home folder to a remote server:

```
rsync -avz -e ssh ~/ username@remote-server:/path/to/remote/folder/
```

This command will sync all files and directories in your home folder to the specified remote server. The **-a** option tells **rsync** to preserve file permissions and ownership, the **-v** option provides verbose output, and the **-z** option enables compression to speed up the transfer. The **-e** option specifies the remote shell to use, in this case **ssh**.

**scp**: This command-line tool can be used to copy files and directories between a local machine and a remote server. Here's an example of how to use **scp** to sync your home folder to a remote server:

```
scp -r ~/ username@remote-server:/path/to/remote/folder/
```

This command will copy all files and directories in your home folder to the specified remote server. The **-r** option tells **scp** to copy directories recursively.

Finder: You also can use Finder to sync your home folder to a remote server by mounting the remote folder as a local volume using SMB, NFS or FTP protocols, and then use the Finder to copy the files and folders.

Third party software: There are also third-party software like ChronoSync, Carbon Copy Cloner, and others that can be used to sync your home folder to a remote server.

---

Please note that these commands require a remote server with the appropriate permissions and access to be set up, and also make sure you have the correct login credentials to access the remote server.

## Securely Copy from Server to Local

Instead of backing up files, maybe there are some files on a remote server that you want to copy locally. You might use a command like this. ChatGPT had already suggested `scp` as one possible method to synchronize files to a remote server, and I fully expected it to get this one with ease.

### Securely Copy from Server to Local folder

To securely copy files from a remote server to a local folder on macOS, you can use the **scp** command. **scp** stands for "secure copy" and it uses the SSH protocol to transfer files.

Here's an example of how to use **scp** to copy a file called "file.txt" from the remote server "remote-server" to the local folder "~/Downloads":

```
scp username@remote-server:/path/to/file.txt ~/Downloads
```

This command will copy the file "file.txt" from the specified path on the remote server to the "Downloads" folder in your home directory.

You can also copy a directory and its contents by adding the `-r` option, which stands for "recursive"

```
scp -r username@remote-server:/path/to/directory ~/Downloads
```

You can also copy multiple files and directories by listing them, for example:

```
scp username@remote-server:/path/to/file1.txt /path/to/file2.txt  
/path/to/directory1/ ~/Downloads
```

You also can specify a different port number if the remote server uses a non-default port by using `-P` option:

```
scp -P 22 username@remote-server:/path/to/file.txt ~/Downloads
```

Please note that these commands require a remote server with the appropriate permissions and access to be set up, and also make sure you have the correct login credentials to access the remote server. Also, the **scp** command encrypts the data as it is transferred, providing an additional layer of security.

---

## Archiving Folders

Sometimes we want to back up a folder by creating an archive. We might want to email it or upload it to a public server, or just simply copy it to a flash drive. If the goal is to keep the files together, an archive is perfect. If only I could remember what that command syntax is.

I asked for both the commands to archive and unarchive a given directory.

Archive the src directory using tar and gzip

To archive the **src** directory using **tar** and **gzip**, you can use the following command:

```
tar -czvf src.tar.gz src/
```

This command creates a new archive file called **src.tar.gz** that contains the contents of the **src** directory. The **-c** option tells **tar** to create a new archive, the **-z** option tells it to use **gzip** to compress the archive, and the **-v** option tells it to display verbose information about the progress of the archive creation. The **-f** option specifies the name and location of the archive file.

You can also use **tar** with **gzip** with the following command

```
tar -zcvf src.tar.gz src/
```

It will produce the same result.

Please note that this command will create the archive file in the current directory, you can specify a different directory to save the archive file. Also, the original **src** directory will not be deleted, if you want to delete it after archiving you can add **--remove-files** option

```
tar -czvf src.tar.gz --remove-files src/
```

This command will archive the src directory and remove the original directory after archiving it.

This is the first time I have seen ChatGPT provide the exact same information as an alternative to its first suggestion. Notice that the first two commands are identical except for the ordering of parameters, which is irrelevant. It's literally the same command.

I do appreciate the third option it gave, though. I was not aware of the `--remove-files` option.

---

---

I then asked it to uncompress the tar file. I admit that I was not prepared for the tutorial it generated.

To uncompress a **.tar.gz** file, you can use the following command:

```
tar -xzvf src.tar.gz
```

This command tells **tar** to extract the contents of the **src.tar.gz** file, the **-x** option tells it to extract the archive, the **-z** option tells it to use **gzip** to decompress the archive, and the **-v** option tells it to display verbose information about the progress of the extraction. The **-f** option specifies the name and location of the archive file.

You can also use the following command

```
tar -xvf src.tar.gz
```

This command will also extract the contents of the **src.tar.gz** file, but you need to specify that the file is gzipped by adding the **-z** option

```
tar -xvzf src.tar.gz
```

This command will extract the contents of the archive to the current directory, if you want to extract the files to a different directory you can use the **-C** option

```
tar -xvzf src.tar.gz -C /path/to/directory
```

This command will extract the files to the directory **/path/to/directory**.

Please note that the extracted files will have the same permissions and ownership as the original files, if you want to change the permissions or ownership of the extracted files you can use the **--no-same-permissions** and **--no-same-owner** options.

```
tar -xvzf src.tar.gz --no-same-permissions --no-same-owner
```

This command will extract the files with the current user's permissions and ownership.

Also, after extracting the files you can delete the original archive file by adding **-k** option

```
tar -xvzf src.tar.gz -k
```

This command will extract the files and keep the original archive file.

It's important to note that tar.gz files are also known as tarballs, they are a combination of tar archive and gzip compression, the **.tar** file contains the files and the **.gz** file is the compression. It's important to use both options -x and -z to extract and decompress the file respectively.

---

---

Once again, it managed to get confused and essentially gave me the same command for two different options. Otherwise, I think it did a thorough job explaining things.

## GitHub SSH Key

I don't know about you, but I always have to look up how to create a new SSH key for GitHub using `git+ssh` protocol.

ChatGPT suggested that I use my email address as a comment and gave me a complete example. It also split the process into two commands instead a one, as I would have expected.

Create and copy a new ssh key for github to the clipboard

To create and copy a new SSH key for use with GitHub on macOS, you can use the **ssh-keygen** and **pbcopy** command. Here is an example of how to create a new SSH key and copy it to the clipboard:

Open the Terminal and generate a new SSH key:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

This command creates a new RSA key with a 4096-bit key size, and associates the email "your\_email@example.com" with the key as a comment.

When prompted, you can press Enter to accept the default location and file name for the key, or enter a new location and file name.

After the key is generated, you can copy the key to your clipboard using the **pbcopy** command:

```
pbcopy < ~/.ssh/id_rsa.pub
```

This command copies the contents of the file "`~/.ssh/id_rsa.pub`" to the clipboard.

Go to your GitHub settings, under SSH and GPG keys, click on "New SSH Key" button.

Give a title to the key, and paste the key on the "Key" field, press "Add SSH key" button.

You can test the connection by running **ssh -T git@github.com**

Please note that **pbcopy** command only works on macOS and is used to copy data to the clipboard. On other operating systems, the command may differ and you may need to use **xclip** or **xsel** to copy the key to the clipboard.

---

---

I love that ChatGPT goes above and beyond the actual command, providing examples, options, and even caveats.

## Commands Specific to macOS

So far, I've been impressed with what ChatGPT has been able to do. There are other commands that I have a hard time remembering, particularly commands on macOS that manipulate media files. Let's see how it did with some of those.

### Convert JPEG to PNG

Here is the first, asking it to convert a folder of JPEG files to PNG format.

Convert all jpg files in the current directory to png files

You can use the command-line tool **convert** from ImageMagick to convert all JPG files in the current directory to PNG files. Here is an example of how to use **convert** to convert all JPG files to PNG files:

```
for file in *.jpg; do convert "$file" "${file%.jpg}.png"; done
```

This command will loop through all files in the current directory that have a **.jpg** extension, run the **convert** command on each file, and save the output to a new file with the same name but with a **.png** extension.

Alternatively, you can use **find** and **mogrify** command from ImageMagick:

```
find . -name "*.jpg" -exec mogrify -format png {} +
```

This command will search for all **.jpg** files in the current directory and its subdirectories and change their format to **.png** and overwrite the original files.

Please note that this will overwrite the original JPG files. If you want to keep the original files, you can modify the command to save the output to a different location or with a different file name.

ChatGPT told me that the **convert** command comes from ImageMagick, which I didn't have (and possibly have never heard of). Deciding not to give up just yet, I asked it how to install ImageMagick.

---

---

How do I install ImageMagick?

To install ImageMagick on macOS, you can use the package manager Homebrew. Here are the steps to install ImageMagick using Homebrew:

Install Homebrew if you haven't already by running the following command in the terminal:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Once Homebrew is installed, you can use it to install ImageMagick by running the following command in the terminal:

```
brew install imagemagick
```

After the installation is complete, you can check that it was successful by running the following command:

```
convert -version
```

This will display the version of ImageMagick that you have installed.

Another way to install ImageMagick on macOS is to download the binary package from the official website and install it manually. You can find the binary package here:

<https://imagemagick.org/script/download.php#macosx>

Please note that you may need to add the ImageMagick binaries to your system's PATH for the **convert** and **mogrify** command to be recognized by the terminal.

## Extract Audio from Video Files

As expected, ChatGPT gave me a lot of context, after suggesting the command, and also explaining what I needed to understand about it.

Extract audio from all the mp4 files in the folder

You can use the command-line tool **ffmpeg** to extract the audio from all MP4 files in a folder on macOS. Here is an example of how to use **ffmpeg** to extract the audio from all MP4 files in the current directory and save it as an MP3 file:



```
for file in *.mp4; do ffmpeg -i "$file" -vn -acodec libmp3lame -ac 2 -q:a 4 -y "${file%.mp4}.mp3"; done
```

This command will loop through all files in the current directory that have a **.mp4** extension, run the **ffmpeg** command on each file, and save the audio output to a new file with the same name but with a **.mp3** extension.

This command uses the **-i** option to specify the input file, the **-vn** option to disable the video stream, the **-acodec** option to specify the audio codec, **-ac 2** to specify the number of audio channels, **-q:a 4** to specify the audio quality and **-y** to overwriting output files.

You can change the output file format and codec by changing the **mp3** and **libmp3lame** accordingly.

Please note that you may need to install **ffmpeg** in your system if it is not already installed. You can install **ffmpeg** using Homebrew by running the following command in the terminal:

```
brew install ffmpeg
```

Also, please note that this command will overwrite any existing files with the same name. If you want to keep the original files, you can modify the command to save the output to a different location or with a different file name.

The command ChatGPT gave is complete, but its explanation is misleading. Or maybe I simply read it wrong. The **-y** parameter will automatically overwrite any existing files on output, but the command is not destructive to the input files.

My final test was to try to extract audio using **avconvert**, another command on macOS. Apparently, this command no longer exists, which ChatGPT was happy to explain to me when I asked about it.

What is avconvert?

**avconv** is a command-line tool similar to **ffmpeg** that can be used to convert multimedia files. To extract the audio from a video file using **avconv** on macOS, you can use the following command:

```
avconv -i inputfile.mp4 -vn -acodec libmp3lame -ac 2 -q:a 4 -y outputfile.mp3
```

This command uses the **-i** option to specify the input file, the **-vn** option to disable the video stream, the **-acodec** option to specify the audio codec as **libmp3lame**, the **-ac 2** option to specify the number of audio channels, the **-q:a 4** option to specify the audio quality and the **-y** option to overwrite output files if they already exist.

---

You can change the input and output file name and format accordingly, also you can change the codec and quality options as you need.

Please note that you may need to install **avconv** in your system if it is not already installed. You can install **avconv** using Homebrew by running the following command in the terminal:

```
brew install libav
```

Please note that **avconv** and **ffmpeg** are similar but not exactly the same, and their syntax and options may differ slightly. Some options or codecs that work with **ffmpeg** may not work with **avconv**, and vice versa. Also, **ffmpeg** is more commonly used and has more active development than **avconv**, but both of them can be used to extract audio from a video file.

Note that it didn't answer the question that I asked, or so I thought. Confused and mildly curious, I then asked for clarification.

What is the difference between avconv and avconvert?

**avconv** and **avconvert** are actually the same thing, **avconv** is the command-line tool for the **libav** library, that is a fork of the **ffmpeg** library. **avconv** was the command-line tool for the **libav** library and it's the equivalent of the **ffmpeg** command-line tool for the **ffmpeg** library, but the **libav** project was discontinued in 2018 and now the **libav** developers recommend using **ffmpeg** instead of **avconv**. **avconvert** is a typo or a misspelling of **avconv**, and both commands can be used to perform the same tasks. The **avconv** tool has been replaced by **ffmpeg**, but you might find some old resources or scripts that use **avconv** or **avconvert**.

In summary, **avconv** and **avconvert** are the same command-line tool, but the **libav** project was discontinued in 2018 and now it's recommended to use **ffmpeg** instead of **avconv**.

What amazed me is that I thought **avconvert** was the current and correct command to use. Apparently, I was mistaken on both counts.

## Parting Thoughts

While using AI tools can potentially save time and effort, it is important to assess the quality of the generated commands and ensure they are safe and effective. The output from ChatGPT complete and provides appropriate context, but I expected that.

ChatGPT is an advanced AI model that is capable of understanding and responding to complex and nuanced questions. It can provide accurate and reliable answers to your questions, and its output is more detailed and comprehensive.

---

---

In addition to the difference in output quality, another aspect to consider is the area of expertise of each tool. ChatGPT is a general-purpose language model that has been trained on a vast amount of text data, making it a good choice when the subject matter may be unfamiliar, or when you need an explanation or clarification. It is good at providing a comprehensive understanding of a topic, and it can be used for a wide range of tasks, including language translation, text generation, and question answering.

As you will continue to see, it can help you with your coding tasks by suggesting snippets of code, providing documentation, and even completing your code for you.

---

---

# The Future of Software Development



*Image by NoName\_13 from Pixabay*

Is this the end of software development? Is it time to panic? Should we all pack up our laptops and learn a trade? I don't think so, no. Things will certainly change, but in my experience, these types of changes are usually positive things overall.

AI tools and technologies are having a significant impact on the field of software development. However, I don't believe that they will completely replace software developers in the near future.

As we have seen throughout this book, AI can certainly automate some aspects of software development, such as code generation, testing, and debugging. AI can also assist in the design process, as you saw earlier in the chapter on project management.

---

---

Eventually, I believe these tools will be able to help developers analyze code, predict bugs, and provide recommendations for improvements.

Software development is a complex and creative process that involves much more than just writing code. It requires domain knowledge, problem-solving skills, and an understanding of the users' needs. It's not just writing code, but also testing, deployment, maintenance, and updates. AI can certainly help with some of these aspects, but it cannot replace human intelligence and creativity.

In fact, as AI becomes more prevalent in software development, it is likely to create new opportunities and areas for growth. Developers will need to learn how to work with AI tools, understand their limitations, and use them effectively to enhance their own skills and productivity.

While AI will undoubtedly play a significant role in the future of software development, it is unlikely to completely replace the need for human developers. Rather, AI will complement and augment their skills, leading to more efficient and effective software development.

## **Should You Be Worried?**

Should junior developers and people just starting out worry that an AI will take their jobs? Again, I believe the answer is no.

If you are a new, junior level developer or someone who is just starting out in software development, you should not be overly concerned about AI taking your job.

Think of AI like power tools for software development. Just as power tools are designed to make a carpenter's job easier and more efficient, AI tools are designed to make a developer's job easier and more efficient. Power tools allow carpenters to work more quickly, accurately, and safely, and AI tools provide developers with similar benefits.

---

---

However, just as power tools don't replace the need for skilled carpenters, AI tools do not replace the need for skilled developers. Both carpenters and developers need to have a deep understanding of their craft, including the underlying principles, best practices, and techniques. They also need to have problem-solving skills and creativity to find new and innovative solutions to the challenges they encounter.

Furthermore, just as power tools require maintenance and upkeep to function properly, AI tools require ongoing development and improvement to stay relevant and effective. Developers need to stay up to date with the latest advancements in AI technology, learn how to use these tools effectively, and adapt their skills as needed.

With the right training, attitude, and mindset, you can build a rewarding and fulfilling career for yourself in software development.

I think we'll look back on this time as one of exciting growth and opportunity. Eventually we'll wonder how we ever got along without these tools.

## Amazon Review

If you enjoyed this book, I'd appreciate you leaving me a positive review on Amazon, which you can do here: <https://www.amazon.com/review/create-review?asin=B0C56TTL2V>

## Bonus Materials

If you haven't already done so, please don't forget to sign up for the book's bonus materials at <https://walkingriver.gumroad.com/l/pair-programming-bonus>

---

*Those who sign up for my bonus materials will also be eligible for occasional sneak peeks and discounts on future books and other content.*

---

---

## Updates and Questions

And remember, if you have questions or just want to be updated about this and future books, feel free to sign up at the link above, or send an email to [michael@walkingriver.com](mailto:michael@walkingriver.com).

I'm also very active on Twitter, where you can find me posting as [@WalkingRiver](#).

---

*And now, I urge you to go forth and create cool things.*

---