JENS OLIVER **MEIERT**

FRONTEND DOGMA

UPGRADE YOUR HTML HTML HTML HTML IV

FOREWORD BY SIMON **PIETERS**

Upgrade Your HTML IV

10 More Examples to Improve Your Markup

Jens Oliver Meiert

This book is available at http://leanpub.com/upgrade-your-html-4

This version was published on 2024-11-01

Tweet This Book!

Please help Jens Oliver Meiert by spreading the word about this book on Twitter!

The suggested hashtag for this book is #htmlupgrade.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#htmlupgrade

Contents

Foreword

Validating HTML was cool during the XHTML era, but developers largely forgot about this tool for quality since. Jens makes the case that it's an essential tool, and I agree. Quality improves because a conformance checker will catch mistakes such as typos or missing tags, or usage of elements or attributes that are obsolete.

Optimizing and minimizing HTML and checking if there are more appropriate elements or attributes to use is a helpful exercise, because it can improve accessibility while you learn about new features. HTML is also evolving—there are probably features you didn't know existed, or maybe you had the wrong idea about the appropriate use for an element that has been around forever. Your HTML skills should improve if you adopt a habit of being critical of the code you write or maintain, and read about the relevant features in MDN and the HTML standard.

Another aspect of consideration that Jens touches on is syntax. If you have the habit of writing HTML with XHTML-like syntax, you might be a bit uncomfortable with omitting the slash in void elements (e.g., `<br>`), or omitting tags. But the parsing rules for HTML are well-defined, and are interoperably implemented in browsers. So long as you understand and follow the rules, and check your work with a conformance checker, there is no problem with omitting tags. After doing it for a while, maybe you'll start to *like* the minimalist style, and go ahead and remove optional code.

Get ready to upgrade your HTML.

—Simon Pieters

Intro

Hi, and welcome! On your screen you're reading the fourth edition of *Upgrade Your HTML*, a book series I started in 2019, to critique and then minimize and optimize HTML largely found in the wild, somewhere on the Web.

I like doing that, and people like you like reading that (I hope), because HTML looks so easy but really isn't. HTML is a document language that comes with a syntax that is simple to learn, but rules and peculiarities that take a long time to master.

Like previous editions, this booklet won't teach you *all* rules and peculiarities. It won't even follow any strict didactic approach. But while it's a loose and light collection of HTML case studies, I've been paying attention to giving it some structure and expanding on some details. This is to ensure you can take something away from it, even if you've read other books of the series.

Just as the book is short, this introduction is short, too. Please enjoy *Upgrade Your HTML IV*.

—Jens Oliver Meiert

Who is this Jens person again? There's a section at the end of this book for that (*About the Author*). Jens has spent much of his life working with HTML and CSS, including developing compact HTML/CSS frameworks, defining quality HTML and CSS coding guidelines, and maxing out HTML and CSS optimization options. He loves HTML, and knows a thing or two about the language.

Acknowledgments

This book wouldn't have been possible without the help of Simon Pieters, who reviewed the manuscript and wrote the foreword of this book, Jad Joubran, who reviewed as well, and Kirsty MacRae, who edited the manuscript. I'm humbled and happy about having worked with you on this—thank you!

I also like to thank Geoffrey Crofte, Stefan Nitzsche, and Adrian Roselli for their friendly permission to discuss code they had discovered on their own journey on the Web. Great minds think alike!

0. Write HTML

This book starts with a “Chapter 0,” because I like to start with this point without depriving you of a chapter.

```
<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 div-slideshow">
  <h7>
    SOME TEXT I CHANGE TO <h7 style="color: white;">MAKE IT ANONYMOUS</h7>.
  </h7>
</div>
```

(Via and with friendly permission by Geoffrey Crofte.)

You already know what’s coming.

Those class names; really, those class names (👀 `div-slideshow`).

The `<h7>`; even *nested* `<h7>`.

The `<h7>`, no matter whether it’s nested.

There is no `<h7>` element in HTML. The buck stops with `<h6>`.

But isn’t this obvious?

As you can tell, it’s not. This is HTML that our peers are writing—that professional frontend developers are writing.

They write it because they had not been trained to validate their code. To validate, through the classic W3C validator (same core for living HTML as validator.nu), or one of the various packages. To validate, because that’s what we as professional frontend developers do. Everyone can write invalid HTML. Professional frontend developers write valid HTML—because only valid HTML *is* HTML.

1. Aim for a Good Balance

```
<nav id="r-skiplinks">
  <h6 class="r-sr-only">Bereichsnavigation</h6>
  <div class="r-skl-nav">
    <a class="r-skl" href="#r-search"><span>Zur Suche</span></a>
    <a class="r-skl" href="#r-usr-nav"><span>Zum User Menu</span></a>
    <a class="r-skl" href="#r-main-nav-rezepte"><span>Zum Hauptmenu / Rezepte</span>\
</a>
    <a class="r-skl" href="#r-main-nav-magazin"><span>Zum Hauptmenu / Magazin</span>\
</a>
    <a class="r-skl" href="#r-main-nav-community"><span>Zum Hauptmenu / Community</s\
pan></a>
    <a class="r-skl" href="#r-main-nav-videos"><span>Zum Hauptmenu / Videos</span></\
a>
    <a class="r-skl" href="#main"><span>Zum Content</span></a>
    <a class="r-skl" href="#r-footer"><span>Zum Footer</span></a>
  </div>
</nav>
```

What you just read is code from a German website, with “skip” links that are hidden by default, and shown when “tabbing” through the page. Nothing spectacular here; in fact, a good practice when there are large blocks of navigation between the user’s focus and the main content. (WebAIM, to name just one, has a good intro on skip links.)

Let’s look at the markup.

The `nav` container looks appropriate, given that the section offers navigation. That also holds in the case of in-document navigation.

The heading level appears dubious (is it an `h6`?), but you’ve seen me tread lightly around these. (After all, correcting a heading level boils down to changing two digits.)

Then it gets a little weird: A `div` element hosting a list of hyperlinks, which in turn contain `span` elements. The keyword here is certainly “list,” as in most if not all cases, this asks for a list element (usually one of `ul`, `ol`, or `dl`), as with:

```
<ul>
  <li><a href=#r-search>Zur Suche</a>
  <li><a href=#r-usr-nav>Zum Nutzer-Menü</a>
  <li><a href=#r-main-nav-rezepte>Zum Rezepte-Menü</a>
  <li><a href=#r-main-nav-magazin>Zum Magazin-Menü</a>
  <li><a href=#r-main-nav-community>Zum Community-Menü</a>
  <li><a href=#r-main-nav-videos>Zum Videos-Menü</a>
  <li><a href=#main>Zum Inhalt</a>
  <li><a href=#r-footer>Zum Footer</a>
</ul>
```

With this list change, I also snuck in three more changes:

1. Removing the `span` elements: Some of you have seen me do this in the past, working with the assumption that these aren't needed and that their effect can be accomplished by other means. (I'll repeat, too, how this assumption may be wrong. The use of helper elements can be an option.)
2. Removing the classes: That type of repetition does and must constitute a red flag—effectively, it's a violation of *Don't Repeat Yourself*. CSS has always offered the option to style based on the context; if you look at the original code, meaning a selector like `.r-skiplinks li` (or `.r-skl-nav li`, if we kept the `div` container class).
3. Fixing and rewording the navigation text: It contained some typos (menu, *Menü* in German, is written with an umlaut), and it was not as easy to parse as it could be (notice the *Zum Hauptmenu* [sic] references? this kind of frontloading slows down scanning and listening).

If you push the context argument (in 2.) further, you notice how this can be applied to the other classes, too: There's nothing preventing us from working with one class or ID for the whole section—or none, given how powerful CSS selectors are by now. (Haven't you ever wondered about classless development? I call it the "lost paradigm".)

This leads us to the following, final piece of code (with `aux` as a tentative name for auxiliary navigation):

```
<nav id=aux>
  <h6>Bereichsnavigation</h6>
  <ul>
    <li><a href=#r-search>Zur Suche</a>
    <li><a href=#r-usr-nav>Zum Nutzer-Menü</a>
    <li><a href=#r-main-nav-rezepte>Zum Rezepte-Menü</a>
    <li><a href=#r-main-nav-magazin>Zum Magazin-Menü</a>
    <li><a href=#r-main-nav-community>Zum Community-Menü</a>
    <li><a href=#r-main-nav-videos>Zum Videos-Menü</a>
    <li><a href=#main>Zum Inhalt</a>
```

```
11      <li><a href=#r-footer>Zum Footer</a>
12    </ul>
13  </nav>
```

Not that you planned to do this but—don't use the `hidden` attribute for navigation sections like this. When writing this chapter, I made the enthusiastic mistake of deeming this a possible use case for `hidden=until-found`. If you don't know it, `hidden=until-found` reflects a relatively recent addition to the previously Boolean `hidden` attribute. (The `until-found` value and corresponding "hidden until found" state indicate that "the element is hidden like [in] the hidden state but the content inside the element will be accessible to find-in-page and fragment navigation.")

That *seemed* to be useful here—but, as peers like Sara Soueidan and Scott O'Hara were quick to point out on a spontaneous Twitter poll, isn't true.

Now that this warning anecdote is out of the way, let's consider the method of hiding navigation. A CSS or accessibility book might want to approach this authoritatively, as I've been avoiding the topic in this series of HTML books. Not only does it go beyond the focus on HTML—there already are great articles on accessibly hiding elements. The A11Y Project's *Hide Content*, 18F's *Hidden Content*, and Orange's *Accessible Hiding* provide good information, to list a few.

But what happened to the "good balance" to aim for, in line with the topic of this chapter?

Web development requires balance, because there are many things that matter. Here, we value conformance, minimalism, code quality. But there's content quality, accessibility, usability, user experience, performance, SEO, and many more priorities. If we don't think in terms of balance, we start optimizing on one (or few) dimensions, and may even over-optimize. (See the coming "tip" section for more thoughts.)

When you look at the optimized code, then it still pretty much matches this book's priorities of valid and minimal and quality code. But for "skip" links, that isn't worth anything if it's done in a way that's not accessible; and it's also not worth much if we forget about building it so that it's usable. While I owe you more about defining and identifying paths to a good balance, understanding the idea of a balance is where it all starts.

Beware of over-optimization, or one-dimensional optimization. Web design and development cover many different areas, all of which we can optimize for: Think of the aforementioned factors such as SEO, performance, accessibility, usability and user experience, technical conformance, legal compliance, general code quality, &c.

The problem is that when we push too hard with one area or factor, not only does this get costly (it requires more and more work to identify ever smaller improvements), it will also at some point involve trade-offs that negatively affect other areas.

I tried to describe the problem in a post: *One-Dimensional Website Optimization Considered Harmful.* Of course, all I want to say is to be mindful, and to make conscious decisions. That should guard against over-optimization.

2. Fight Divitis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

3. Stop Closing Void Elements

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

4. Minimize Attributes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

5. Beware Metadata Madness

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

6. Question Table Buttons

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

7. Question Button Links

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

8. Try to Do Without

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

9. De-Duplicate Content

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

10. Challenge Yourself, Even When It's Art

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

Outro

What did we cover this time?

Write HTML. Check your HTML output on conformance—i.e., validate.

Aim for a good balance. Quality web development depends on many factors.

Fight divitis. Like not checking on conformance, it's one of the field's afflictions.

Stop closing void elements. You don't have to. Do it for the same reasons you're using JSON—you don't need XML for the job.

Minimize attributes. Find joy in that, too.

Beware metadata madness. There's nothing like too much of a good thing here.

Question table buttons and question button links.

Try to do without verbose form markup (and without default values).

De-duplicate content. De-duplicate content.

Challenge yourself, even when it's art. Aim to produce the highest-possible quality.

*
**

HTML isn't easy, or: Those who say HTML is easy, know little of HTML.

I'm grateful that you went through these 10 case studies with me. I'm grateful because we're connected by the understanding that there's more to HTML than knowing 10 tags (with 28.8% of the output being `div` elements).

Therefore, all I wish to do now is to thank you. See you around, whether in this series or elsewhere, or anywhere that's about minimal, quality HTML.

Feedback

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-4.

About the Author

Jens Oliver Meiert is an engineering lead and author who, after several years as a tech lead at Google, works as an engineering manager at LivePerson. He's an expert in web development, specializing in HTML and CSS minimization and optimization. Jens contributes to technical standards and regularly writes about the craft of web development on his website, meiert.com.

Other titles by Jens Oliver Meiert:

The Web Development Glossary 3K (2023)

> What is a BHO? CQRS? An EMD? What is Goanna? Hooking? Sharding? How about dynamic color, the phoenix server pattern, or the rules of ARIA? Covering more than 3,000 terms and concepts, and including explanations from Wikipedia and MDN Web Docs, *The Web Development Glossary 3K* provides an overview of web development unlike any other book or site.

Available at Apple Books, Kobo, Google Play Books, and Leanpub. (Try the glossary online at WebGlossary.info!)

The Little Book of Little Books (2021)

> *The Little Book of Little Books* consists of lovingly polished editions of *The Little Book of HTML/CSS Frameworks* (originally published in 2015), *The Little Book of HTML/CSS Coding Guidelines* (2015), and *The Little Book of Website Quality Control* (2016).

Available at Amazon, Apple Books, Kobo, Google Play Books, and Leanpub.

CSS Optimization Basics (2018)

> Are you unsure about your style sheets' quality, or whether you've maxed out your options? *CSS Optimization Basics* covers the necessary mindsets, discusses the main optimization methods, and presents useful resources to write higher-quality CSS.

Available at Amazon, Apple Books, Kobo, Google Play Books, and Leanpub.

On Web Development (2015)

> *On Web Development* bundles 134 articles and the last 11 years of technical writings by Jens Oliver Meiert (meiert.com). Freshly reordered and commented, the articles cover processes and maintenance, HTML and CSS, standards, as well as development and design in general; they include coding basics and principles, carefully scathing criticism, and tips and tricks and trivia.

Available at Amazon.

The Little Book of HTML/CSS Frameworks (2015)

> With the speed of web development today, it's little wonder that so many frameworks are available, since they come with a promise of saving development and design time. But using the wrong framework, or wrongly using the right framework, can be costly. This concise book shares higher-level ideas around web development frameworks that govern HTML and CSS code, whether you're looking at an external option or planning to build your own.

Available at O'Reilly.

About *Upgrade Your HTML IV*

Written by Jens Oliver Meiert.

Published by Frontend Dogma, c/o Jens Oliver Meiert, Apartado de correos 3, 36070 Pontevedra, Spain.

Editor: Kirsty MacRae
Reviewers: Jad Joubran, Simon Pieters

Contact +34-610859489 or info@frontenddogma.com for questions and more information.

Follow Frontend Dogma on Mastodon (or other networks).

[1.1.25]