JENS OLIVER **MEIERT**
FRONTEND DOGMA

UPGRADE YOUR
HTML
HTML
HTML
HTML
III

FOREWORD BY
JUKKA K. **KORPELA**

Upgrade Your HTML III

10 More Examples to Improve Your Markup

Jens Oliver Meiert

This book is available at http://leanpub.com/upgrade-your-html-3

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

HTML was born simple. It was primitive, even when compared with word processors of the time. It has been extended in many ways, and HTML and related technologies are now a complex conglomerate that gets more complex. There is a reason for that: There are so many different things people want to do with it, from the simple display of facts to immersive applications.

In addition, there are many ways to do things with HTML. The simple way isn't always obvious, and sometimes it isn't even right. In his series of "Upgrade Your HTML" booklets, Jens Oliver Meiert describes how HTML code is so often pointlessly bloated and complex, and what to do with that. Typically, you'll learn that "upgrade" means simpler code, less code, and more useful code.

The famous quote from Einstein, "make things as simple as possible, but not simpler," is incorrect, as almost all famous quotes are. Einstein was actually referring to basic elements of a theory, not about things in general.

The popular slogan is still a useful guideline, especially in constructing things like web pages or applications. Use the simplest tools that let you do things you want or need to do, and use them as simply as possible, instead of looking for the "newest" or "most powerful"—or just copying code or design, or using software that generates unmanageable code.

Jens Oliver Meiert may at times take you too far, as if the "but not simpler" were not there. I think this is intentional and useful: You need to exaggerate in order to get far enough. His analyses might not cover all the possible good reasons for complexities in some HTML code.

The purpose is not to make you start deleting elements and attributes from existing code, but to use judgment in adding "features" into your code and even to reconsider what you might have used for no good reason, or for reasons that are no longer valid.

—Jukka K. Korpela

Intro

Upgrade your HTML. Why would you need to do that? HTML is old, HTML is easy! Can't you, can't we write HTML?

We're certainly *able to*, but we can't reliably put it into practice yet. Not from what we see in the millions of HTML documents and templates making millions of sites and apps out there.

That we're not reliably writing good HTML is the premise of this book series, that now enters its third year. Let's have a look at the introduction for the first part, released in October, 2019:

> […] *not* everyone writes good HTML. There are many reasons for that.
>
> The most important one is that HTML isn't actually that easy, because it really is complex. Want an example?
>
> HTML 5.2, which is the last HTML recommendation by the W3C […], contains 111 elements alone. *111.* How many do you know? How many does the average web developer know? What does it mean for their markup if they don't know all the elements?
>
> Want another example? When people talk about HTML elements like `html` or `a` or `p`, then most of the time they do mean *elements*. But what they then *talk about* is "tags." Google, as of September, 2019, finds 25 million occurrences of "html tags" alone. HTML elements, what people mean, but don't call by name? 2.4 million hits, a tenth. This begs the question how well developers understand HTML […].
>
> Another one? In a few years, HTML will celebrate its 30th birthday. Very cool! We will certainly have maxed out all options to reduce HTML payload to improve performance, wouldn't we? Well, no. One of the major options at our disposal to reduce HTML payload is not to write HTML that can be left out without a document turning invalid […]. Unfortunately, almost no one uses that option.

We still need to keep all HTML elements (and attributes, and rules) in mind. We still need to watch for the use of correct terminology. We still can't speak of HTML being mastered, let alone being exhausted yet.

Therefore, we can keep upgrading our HTML. Therefore, I can keep writing this series. Therefore, we can enjoy looking at the HTML on the Web and enjoy making it better.

This joy is important. The point of this book series is certainly to teach about the craft of writing HTML. But it's also about the joy of writing HTML, and of writing better HTML. If you've read one of the first parts of this series, you may have found some of the material joyful. If you haven't read one yet, I hope you will enjoy this third part. HTML is pretty cool—let's now review and work with 10 more examples.

—Jens Oliver Meiert

Who is this Jens guy 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.

Acknowledgments

This book wouldn’t have been possible without the help of Jukka K. Korpela, who was so kind to review the manuscript and contribute the foreword; of Jad Joubran for his technical reviews and comments; and of Kirsty MacRae for reviewing and editing. *Thank you.*

1. See the Forest for the Trees

```html
<ul class="nav__menu nav__menu--depth1 sub-menu">
  <li class="nav__menu-item nav__menu-item--depth1 menu-item menu-item-type-post_typ\
e menu-item-object-page menu-item-105848"><a href="https://www.whitehouse.gov/about-\
the-white-house/tours-events/" class="nav__link">White House Tours</a></li>
  <li class="nav__menu-item nav__menu-item--depth1 menu-item menu-item-type-post_typ\
e menu-item-object-page menu-item-105855"><a href="https://www.whitehouse.gov/about-\
the-white-house/presidents/" class="nav__link">Past Presidents</a></li>
  <li class="nav__menu-item nav__menu-item--depth1 menu-item menu-item-type-post_typ\
e menu-item-object-page menu-item-105862"><a href="https://www.whitehouse.gov/about-\
the-white-house/first-ladies/" class="nav__link">Past First Ladies</a></li>
  <li class="nav__menu-item nav__menu-item--depth1 nav__menu-item--has-submenu menu-\
item menu-item-type-post_type menu-item-object-page menu-item-has-children nav__menu\
-item--has-menu-5 menu-item-105995"><a href="https://www.whitehouse.gov/about-the-wh\
ite-house/the-grounds/" class="nav__link">The Grounds</a></li>
  <li class="nav__menu-item nav__menu-item--depth1 nav__menu-item--has-submenu menu-\
item menu-item-type-post_type menu-item-object-page menu-item-has-children nav__menu\
-item--has-menu-7 menu-item-106079"><a href="https://www.whitehouse.gov/about-the-wh\
ite-house/our-government/" class="nav__link">Our Government</a></li>
</ul>
```

This example is from the former U.S. administration's website for the White House. This is a technical book and not a political one, so let's look at the code. It handles 5 list items which, in turn, handle 12 words. The amount of code, because there's so much of it, makes it hard to tell what the content is all about. That's the point we're starting with:

If your HTML code vastly exceeds the amount of information it contains, it's a signal there's a problem (with the code).

In this example, most code comes from the number, length, and repetition of the various class names. Yes, we see BEM here, the CSS methodology we covered a little more in Chapter 5 of *Upgrade Your HTML II*. And yet we also see why the classic ground rule for ID and class names, to make them as long as necessary, but as short as possible still has value. It shows, too, why we hold "Don't Repeat Yourself" in high regard as an engineering principle: It keeps our code more focused and easier to understand.

What do we do? If you're new to this book series, I'll probably seem radical: We throw out everything that doesn't have an obvious reason for existence.

```
<ul class=nav>
  <li><a href=/about-the-white-house/tours-events/>White House Tours</a>
  <li><a href=/about-the-white-house/presidents/>Past Presidents</a>
  <li><a href=/about-the-white-house/first-ladies/>Past First Ladies</a>
  <li><a href=/about-the-white-house/the-grounds/>The Grounds</a>
  <li><a href=/about-the-white-house/our-government/>Our Government</a>
</ul>
```

If you compare this with the original code, you'll recognize the benefits of leaving out optional code; of not repeating ourselves; of keeping things simple. This is why we do this.

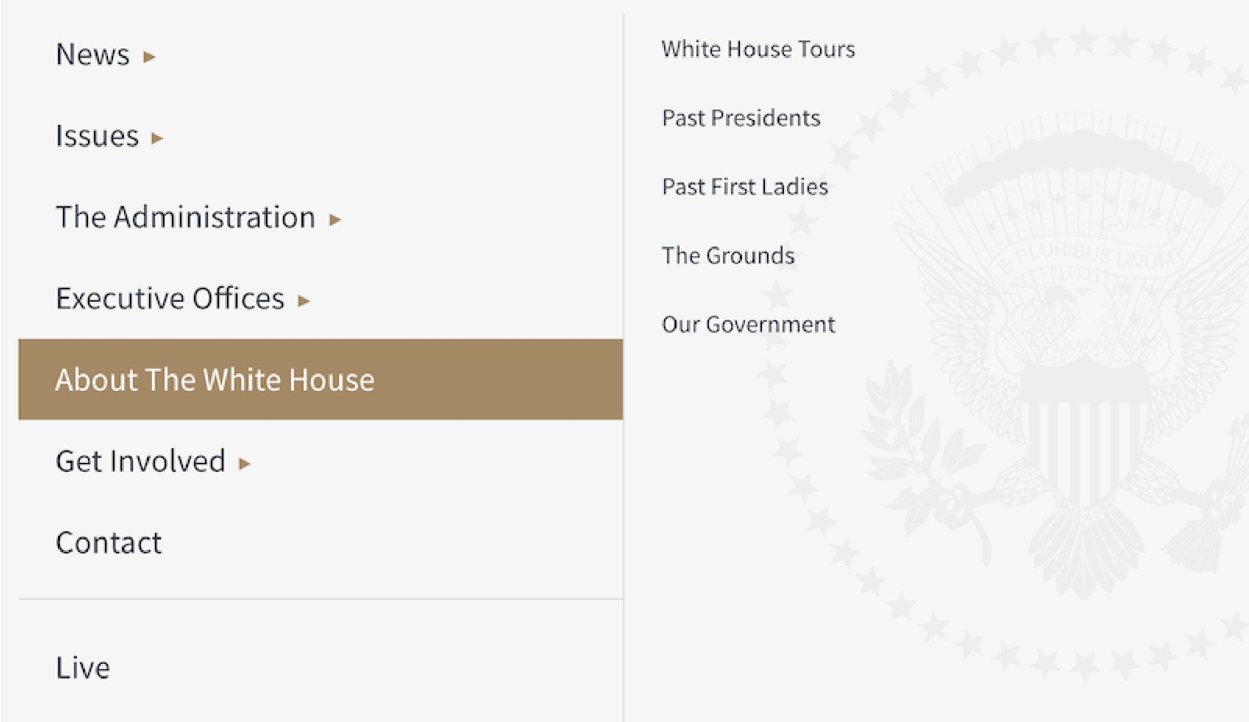
Yet then, question this. What happened here? Why? Are the underlying assumptions fair?

A more obvious change may be the truncating of link targets—all the links have been made relative to the root, stripping "https://www.whitehouse.gov". There can be cases where using the full URL is useful (like when a page is available on several domains, and you wish to lead users to the canonical one), but there are also alternatives for handling this (as with redirects). For same-site URLs, it's often more convenient, and less payload, to work with relative links.

A classic change for this book series, but one that not all readers may be familiar with, is the decision to strip all optional markup. You notice the missing quotes as well as the missing closing `</li>` tags. They are optional and would only increase page weight.

While it may be a matter of taste whether you want to keep optional code in your code base, at least for production it seems useful to remove it. Personally, I write HTML so that it doesn't include anything optional (exceptions would prove the rule).

Yet that leaves us with a drastic change: the aggressive consolidation of classes. They are all gone. Was this a reasonable change? To answer this, let's look at the navigation in question:



White House navigation.

What do you notice? What we're discussing is actually a subnavigation (so the `nav` class looks not entirely wrong, but off), whose styling is fairly straightforward. While we could go on now and discuss CSS naming methodologies and their pros, cons, and alternatives, I'd like to ask you to play with the idea of keeping it simple; or even simpler:

```
<ul>
  <li><a href=/about-the-white-house/tours-events/>White House Tours</a>
  <li><a href=/about-the-white-house/presidents/>Past Presidents</a>
  <li><a href=/about-the-white-house/first-ladies/>Past First Ladies</a>
  <li><a href=/about-the-white-house/the-grounds/>The Grounds</a>
  <li><a href=/about-the-white-house/our-government/>Our Government</a>
</ul>
```

Make use of element context, as with descendant combinators. Within the surrounding navigation (which could well use one or more classes), it looks entirely possible to do without *any* class for this submenu. Possible. Which is where I suggest we leave this chapter.

Don't Overuse Classes

In the first draft of this book, I had included a different Chapter 6: *Don't Overuse Classes*. It reminded me so much of this chapter that I thought not to axe it, but to add it to this one. So here's another view of the use of classes.

```
<div class="relative z-10 col-start-1 row-start-1 px-4 pt-40 pb-3 bg-gradient-to-t f\
rom-black sm:bg-none">
  <p class="text-sm font-medium text-white sm:mb-1 sm:text-gray-500">Entire house</p>
  <h2 class="text-xl font-semibold text-white sm:text-2xl sm:leading-7 sm:text-black\
 md:text-3xl">Beach House in Collingwood</h2>
</div>
```

In this particular case, let's *begin* with ambitiously optimized markup:

```
<div>
  <p>Entire house
  <h2>Beach House in Collingwood</h2>
</div>
```

(This optimization is ambitious because it seems likely that these are not the only `div`, `p`, and `h2` elements on the page, suggesting some hook, like a class, is still needed to style the markup.)

Now some of you may have recognized where these classes originate from—from Tailwind CSS. Tailwind is an approach to tightly couple structure and presentation based on the idea that both are easy to maintain and change together. (When this is the case, Tailwind seems to work well for developers. If neither structure nor presentation is easy to change, Tailwind is a disaster, because it violates the paradigm of separation of concerns promoted with the first paradigm of frontend development.)

The point is this: Even when the mentioned premises hold—easy maintenance of the markup (perhaps through components) and interlocked changes to both structure and presentation—, this makes for a ton of extra markup (329 bytes vs. 68 bytes).

It is, as always, *your call* to decide whether such an increase is acceptable, perhaps because you find advantages elsewhere. It is, as always, useful to make this a conscious decision. It's just that Tailwind comes with a formidable tax on markup size and markup understandability.

Tailwind counts as an HTML/CSS framework, and (developing) frameworks is a topic I love! In 2015 I wrote a small booklet about developing (but also using) them: *The Little Book of HTML/CSS Frameworks*. I believe it still shares useful thoughts on frameworks. You can access it for free at O'Reilly.

German readers, by the way, can obtain a relatively inexpensive yet also updated edition in German: *Das kleine Buch der HTML-/CSS-Frameworks*. I consider this a fine book on the matter—but what else am I to say.

2. Don’t Worry About Line Breaks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

3. Use the `i` Element, and Use It Appropriately

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

4. Keep an Eye on the Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

5. Don’t Hide Outlines

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

6. Know the Link Types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

7. Keep Your CSS DRY

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

8. Don’t Use Non-Standard Markup (Like `nobr`)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

9. Use Consistent Spelling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

10. Remember `figure` (and `figcaption`)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/upgrade-your-html-3.

Outro

What did we learn?

See the forest for the trees: Don't plant so many trees (write so much markup) that you miss the forest (the purpose). Also: Don't overuse classes—be mindful and frugal with all markup.

Don't worry about line breaks: You can use `br` elements.

Use the `i` element, and use it appropriately: If you're informed, there's no need to avoid `i`.

Keep an eye on the output: *You* should have the last word about your markup, not your tooling.

Don't hide outlines: Don't.

Know the link types: Be aware of the many different link types for `link`, `a`, `area`, and `form` elements—for example, `search`.

Keep your CSS DRY: Avoid repetition, not only in HTML.

Don't use non-standard markup (like `nobr`): If you do, you're not writing HTML.

Use consistent spelling: It makes code easier and more pleasant to read.

Remember `figure` (and `figcaption`): These elements have several use cases and come in handy.

⁂

HTML isn't easy. It's complex—that's why so few people write good HTML. May what we covered add to your knowledge, for you to write even better markup. See you in the next episode of *Upgrade Your 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-3.

About the Author

Jens Oliver Meiert is a web developer and author who, after several years as a tech lead at Google, works as an engineering manager at Jimdo. He’s an expert in web development, specializing in HTML and CSS optimization. Jens contributes to technical standards and regularly writes about his work and research, particularly 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 III*

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: Jukka K. Korpela, Jad Joubran

While this book has been produced with great care, the author, publisher, and contributors assume no liability for the up-to-dateness, correctness, and completeness of the information provided. Liability claims based on the use or non-use of this information are excluded, unless author, publisher, or contributors can be proven to have acted with intent or gross negligence. Use of the information in this book is on your own responsibility. When using code or content subject to open-source licenses or the rights of others, it is on you to ensure compliance with the respective licenses and rights.

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

Follow Frontend Dogma on Mastodon (or other networks).

[1.2.24]