# Painless CSS

Build Beautiful, Intuitive Websites

BILL MEI

This is a free sample of the book
*Painless CSS: Build Beautiful, Intuitive Websites*
by Bill Mei

Learn more at https://www.painlesscss.com/

# Bonus 3:
# Who determines all these rules?

I spoke a lot about "pre-determined rules" and "conventions". Who determines all these rules? Are the rules all listed somewhere for us to read? Will you get arrested by the Internet police if you break the rules?

The World Wide Web Consortium (W3C)[14] is the organization that publishes these rules, in the form of documents called ***specs*** (short for specifications), which you can think of as authoritative guidelines for how the HTML, CSS, and JavaScript languages should work.

If you are curious about what they look like and want to get a sense of what these documents contain, I've linked to them here:

- HTML: https://www.w3.org/TR/html/
- CSS: https://www.w3.org/TR/CSS/
- JavaScript (also known as ECMAScript[15]): https://www.ecma-international.org/ecma-262/9.0/index.html

The specs describe things like:

- Which tags exist and what are their names? (e.g. `<img>` is a real tag, but `<image>` is not.)

---

[14] https://www.w3.org/

[15] This might be the first time you've heard of "ECMAScript". What's the difference between JavaScript and ECMAScript? In short, JavaScript is one specific implementation of ECMAScript. You can read this article for a longer answer: https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5 Archived link: https://web.archive.org/web/20171214215715/https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5?gi=d413ea2379cc

- How do these tags behave by default? (e.g. is the tag inline, or block? See *Chapter 5* for what this means)
- What properties or attributes can you customize on these tags? (e.g. we talked earlier about how you can put an `href` property on an `<a>` tag, but you can't put an `href` property on a `<p>` tag.)
- And much, much more! The specs are comprehensive. There is nothing about how a tag works that is written outside the spec.

Once the W3C publishes a finalized spec, it's up to the browser vendors (e.g. Google Chrome, Mozilla Firefox, Internet Explorer, Safari) to implement the specified behavior in the browser software.

This is why websites sometimes look different when you view them in different browsers. The way that Microsoft decides to implement Internet Explorer will be different from how Google decides to implement Chrome, which will also be different from how Mozilla implements Firefox and how Apple implements Safari.

It was inaccurate for me to say the specs define the "rules". The spec is merely a guideline, not a set of "rules". Every company has different priorities, so they don't have to implement their browser exactly the way the spec describes. This is also why sometimes you will encounter bugs in one browser but not another one—it's because the browser was coded in a way that doesn't 100% match the spec; sometimes by mistake, sometimes intentionally.

## Should you learn all the rules?

Some tags can come with caveats that will result in unintended consequences, so should you read all the specs and memorize all the rules before using an HTML tag or CSS feature? In my opinion, no. While it might seem irresponsible to use a tag without fully understanding it, once you learn the basics, you'll find that tags generally behave the way you expect.

Most of the time, you can charge ahead by using a tag and seeing how it behaves. If something breaks, or surprises you by behaving differently, or you encounter a bug, then this is a wonderful opportunity to learn more about the tag by going directly to the documentation and reading it more thoroughly. This will let you

discover any secret rules you didn't know before and will allow you to slowly internalize more of what's in the spec.

The other reason why it's not generally necessary to read the whole spec, is that the spec is a *massive* document. If you printed it out, it would be over 1200 pages long[16]. These specs are continually changing too, so by the time you finish reading the whole thing, it will be obsolete. Usually, it's faster to search Google or Stack Overflow for answers and use the *Five Steps to Painless CSS* if you get stuck.

For example, let's say you're building a music player, and you're not sure what tag to use. You could search something like "html tag for playing music", and you'll get results that talk about the `<audio>` tag, along with information on how to customize and use it.

Since the specs are lengthy and technical, a better day-to-day reference is the Mozilla Developer Network (MDN)[17]. You can do a search there for the name of the HTML tag or CSS style you are trying to use, and you'll find detailed but easy to read documentation.

**Throughout this book, if I ever say something like "read the documentation", what I mean is to look up the word or concept on MDN.**

If you're truly stumped, know that *every* element will be fully documented, so if you can't find any answers using a web search, you can always fall back on reading the official spec (or MDN) to find what you need, meaning there is no need fear being unable to find the right answers via research.

## Do the rules change?

I mentioned earlier that the specs are continually changing—HTML, CSS, and JavaScript are living languages, and things are added and removed all the time as technology advances and the world changes.

---

16 And that's just HTML. The spec documents for CSS and JavaScript are also more than 1000 pages each.

17 https://developer.mozilla.org

Who gets to make these changes? Can you have your own `<banana>` tag?

## Change Process

The W3C has a process for proposing changes. Basically, you write your proposal in a format called an RFC (Request for Comments), and then you share your RFC with other people from the tech industry who will read it and then ask questions or provide suggestions.

After people have finished debating your RFC and all the feedback is addressed or incorporated, the community is said to have achieved consensus. Your proposal will then be approved as a final spec. The W3C will set a release date for the next version of the specs, which incorporates your proposal and all the other proposals that have been approved since the last version. Once that release date comes around, the browser vendors can then start implementing support for the new spec.

If you've ever heard of Firefox Beta, Firefox Nightly, Google Chrome Beta, or Google Chrome Canary, this is what these beta version browsers are for. If a proposal comes in and it looks like it has a high likelihood of getting accepted, then the browser vendor will often go ahead and ship the feature in a beta version. That way, you'll get a lot more feedback (and catch more bugs) from real users using the real feature to browse real websites.

This staged testing process ensures high quality in the proposals, because if during testing the proposal turns out to be a bad idea in the wild, or breaks websites, then the proposal will get rejected. The staged testing also gives website owners enough time to become aware of any upcoming changes and adapt if necessary, so people are not caught off-guard by their tools suddenly changing without notice.

Even if your proposal makes it into an official spec, remember that in the end the browser vendors decide if they want to follow it. If none of the vendors decide to follow the spec, then you still can't use your `<banana>` tag, because it won't work in any browser.

Thus, for changes to *really* happen, all browser vendors must diligently implement the new specs in a timely manner. Famously, this was something that Internet Explorer was terrible at doing in the 1990s—but because so many consumers used IE, the other vendors capitulated and followed the IE way, even if the

implementation was "wrong". This was never "fixed" because IE was the dominant browser at the time, so you couldn't fix it without breaking most websites which were designed specifically for IE.

Even though cross-browser bugs used to be a huge problem in the past, today it is less of an issue. Vendors are now collaborating to ensure cross-compatibility. The community has also come up with tools called ***polyfills***, which are backports of new features into older browsers, so that older browsers can have access to the newer features without waiting for support from the vendor.

What's incredible, is that almost *everything* you see in HTML, CSS, and JavaScript was proposed by someone like you at some point.[18] In CSS2[19], you couldn't easily draw animations. But designers and developers wanted to do animations, so it was proposed, tested, accepted, and eventually rolled out in CSS3. Now, you can do sophisticated animations in CSS which enables a whole suite of web applications that weren't possible before. HTML, CSS, and JavaScript are continually improving, but they can only improve through proposals and feedback from people like you! 🙂

## Change Politics

Who decides what emojis we have? There is an organization similar to the W3C called the Unicode Consortium; they also have a proposal process, set of specs, and a committee for moderating discussion and making the final decision on what emojis or Unicode characters are allowed.[20]

In 2015, a team of people from the community submitted a proposal to add electrical power symbols to Unicode which was accepted into Unicode 9.0. If you

---

[18] Except for the features that existed at the very beginning of course, but even those have undergone changes over time.

[19] That is, CSS version 2.

[20] Unicode is a standard that determines what characters can be typed from your keyboard, from A to Ż to 家.

want to learn more about what these proposals look like, you can read their write-up on GitHub[21].

Since the final power to make decisions that impact the future of the web are concentrated into the hands of a few committee members, these decisions are sometimes subject to controversy if the community has not come to a consensus and the committee decides to favor one group above another group.

Browser vendors and tech companies even hire their own lobbyists to try and influence community opinion and sway W3C committee members into supporting their interests. The most recent controversial example was when the W3C decided to roll out Encrypted Media Extensions, a DRM standard for HTML, which was met with objections from the EFF[22].

## The Future of the Web

Why do things change? As hardware improves and our relationship with the Internet evolves, the programming languages we use need to adapt.

HTML4[23] was developed in an era where you could only access the Internet through a desktop computer. When more people started to browse the Internet on their smartphones, the HTML language needed to change to support the smaller screens and lower computational power of phones. Thus, when HTML5 was released, a lot of the new features and additions were all about better support for webpages on mobile phones.

Remember that user agents are not just browsers! Your lightbulb or toaster can be a browser too. If you're designing a technologically advanced toaster with a screen

---

21 https://github.com/jloughry/Unicode#lessons_learnt

22 https://www.eff.org/deeplinks/2017/07/amid-unprecedented-controversy-w3c-greenlights-drm-web Archived link: https://web.archive.org/web/20181126033733/https://www.eff.org/deeplinks/2017/07/amid-unprecedented-controversy-w3c-greenlights-drm-web

23 That is, HTML version 4.

that tells you exactly how hot your toast is, you might want to propose a `<toasty>` tag for this purpose.

Today, browser vendors are working on proposing Virtual Reality (VR) related features for the specs. This is because progressively more people are purchasing VR hardware to access the Internet. Eventually, the demand will increase to a point where a spec change is needed, just as how we changed the spec to address increased demand for mobile-friendly websites when more people started buying cell phones.

This is the history of technology; new devices are invented, they bring new challenges, and we see a sort of race in the beginning as users are excited to experiment with the technology and startups are working hard to grow their business, but neither group spends much time thinking about standards.

If you're a startup working on VR, do you care about what some other competing startup is doing with their VR? No, you don't have time to sit down with them and work on a standard, you're just trying to ship products and make money.

In the 1990s, HTML was the new hot technology, so browser vendors just focused on capturing market share, not on standardization, leading to lots of cross-browser bugs for that era. Now that the technology is mature, they can spend more time collaborating on standardization, which is why cross-browser bugs are less common now.

Thus, the next new thing, like pancreas implants that share status updates of your insulin levels on the Internet, will have bugs because there hasn't been enough time for vendors to come together and solve all the issues.

Again, the only way to ensure that technology advances in the way that you want is to contribute to the community by submitting proposals or providing feedback on proposals. Short of that, you could donate to lobby groups such as the EFF or the Mozilla Foundation (if you think they support your interests) or convince a large tech company to back your ideas.