**15 Best practices of CSS3 for beginners.**   
  
Whether you are a beginner or an expert in Frontend Development it is important to follow some best practices in order to keep your cascading style-sheet (CSS) optimized and organized.

HTML5 has been around for many years now and has been stable and supported at least partially supported by most major browsers since 2014. Here I present list of best coding practices regarding HTML5 markup. Let’s get started:

1. **Don’t put everything in to one CSS file.**

**Case-01**: One large CSS file leads to fewer HTTP requests, which can improve performance.  
  
**Case-02:** Several smaller file leads to easier organization which will make development and maintenance cheaper and easier.

Now what should our strategy for both scenarios? That’s a million dollars question. Okay let me clear your confusion; we can follow the win-win rules

“We could use multiple style sheets to keep things better organized, and then compress them into one file before putting them on the site, to improve performance.”

1. **Follow top-down approach for organize your stylesheet**

It always makes sense to lay your stylesheet out in a way that allows you to quickly find parts of your code. I recommend a top-down format that tackles styles as they appear in the source code. So an example stylesheet might be ordered like this:

1. Generic classes (body, a, p, h1, etc.)
2. #header
3. #nav-menu
4. #main-content

It also helps if you keep track of different sections of the website in the stylesheet with comments.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | /\*\*\*\*\*\* main content \*\*\*\*\*\*\*\*\*/    styles go here...    /\*\*\*\*\*\* footer \*\*\*\*\*\*\*\*\*/    styles go here... |

1. **Make UI components**

Tried to make some prebuild UI component that you may use in future just simply plug a class name and have some properties.

1. **Comment your CSS**

For better readability and organized coding, code commenting play insane role. Code commenting helps code refactor in future and helps other developer to understand the project codebase. Code commenting is always good practice. Those developer who write code with proper commenting, have huge market value.

1. **How to write consistent CSS**

Consistent CSS is important for team collaboration as well as in large project. At the beginning part of the project you can set some rules for maintain throughout to your entire stylesheet. It will help you to write consistent CSS.

1. **CSS namespacing**

CSS namespacing helps you to create a structure that governs how CSS properties get written. If you follow the convention, you’ll be able to write CSS without being afraid of side effects.

Here’s a list of namespaces I use:

1. .l-: layouts

2. .o-: objects

3. .c-: components

4. .js: JavaScript hooks

5. .is-|.has-: state classes

6. .t1|.s1: typography sizes

7. .u-: utility classes

1. **Use BEM or Block elements modifier**

Have you worked on large websites that spans more than a few pages? If you did, you probably realized the horrors of not conforming to a robust CSS architecture. You probably would also have researched on ways to write maintainable CSS.

BEM convention is the good choice for well-maintained CSS architecture. If you’ve never heard of BEM before, it stands for block, element and modifier. It looks incredibly ugly when you first feast your eyes on it.

.block { /\* styles \*/ }

.block\_\_element { /\* styles \*/ }

.block--modifier { /\* styles \*/ }

I hated BEM to a point where I didn’t even give it a chance when I first got to know about it. I can’t remember what made me try BEM, but I realized how powerful it is to work with it.

|  |
| --- |
|  |

1. **CSS class nesting or complex selectors**

There are two major problems with using complex selectors. First, your increased specificity will not only make it harder to later rewrite existing rules, but also increase the time it takes for the browser to match selectors.

When your browser is trying to interpret selectors and decide which element it matches, they go from right to left. This is faster in terms of performance than doing the other way around. Let’s take the selector below as an example.

.deeply .nested .selector span {

...

}

Your browser will first start from the span. It will match all the span tags then go to the next one. It will filter out the spans that are inside a .selector class, and so on.

I don't recommend using tags for CSS selectors because it will match for every tag. While the difference can only be measured in a fraction of a millisecond, little things add up. More importantly, it’s good practice to reduce complexity and be specific.

1. **Use CSS preprocessor like SCSS, SASS and LESS**

For keeping your CSS DRY(Don’t repeat yourself), use CSS preprocessor like SCSS,SASS or LESS. The biggest benefit to using these preprocessor is that you can define your re-usable code at the top and reuse it by @include and @extend throught all of your CSS.  
  
CSS preprocessor will make your life easier to write modular based CSS, it’s helps to write more organized and more consistent cascading stylesheet. BEM naming convention with CSS namespacing and CSS preprocessor is perfect combo for robust CSS architecture.

|  |
| --- |
|  |

1. **Reduce redundancy**

Sometimes it’s hard to spot redundancy, especially when repeating rules don’t follow the same order in both selectors. But if your classes differ in just one or two rules it’s better to outsource those rules and use them as an extra class, instead of this.

1. **Vertically align any elements using only CSS**

For vertically align any elements modern CSS introduce CSS flex and grid system. Those are awesome tolls for dynamically centered any elements.

1. **Try to use shorthand**

There is lots of way, you can ménage your style but try to stick with best approach. Always try to use CSS shorthand, it will reduce your code as well as give your stylesheet a nice and good looking shape.

Example:   
.center—item{

margin-top: 0;

margin-bottom: 0;

margin-left: auto;

margin-right: auto;

}

Instead of this example we can write:  
.center—item{

margin: 0 auto;

}

1. **Use rem’s or em’s**

Using rem’s or em’s is more dynamic way instead of using pixels. Try to use rem’s or em’s rather than pixels.

1. **Avoid !important**

For avoid code collusion and if you don’t want to break normal flow of browser behavior with your CSS don’t ever use !important. It’s really hard to find-out conflict within !important and refactor the whole codebase in large scale application.

1. **Use mobile first development approach**

A mobile-first approach to styling means that styles are applied first to mobile devices.

Concern with:

* Min-Width Media Queries
* Progressive Enhancement (CSS & javaScript)
* Content (Not Device Width) Determines Breakpoints
* Relative Units & Percentage Widths

I hope you like my CSS3 coding guidelines and front end web development best practices. You can check here my **15 Best practices of HTML5 for beginners** for give your markup good shape. If you find any errors or mistakes then do let me know. Thanks for this journey & Happy Coding.