skillcrush

CSS BEST PRACTICES

cheatsheet

INTRO

One of the challenges (and delights!) of coding is that there's getting things done and then getting them done RIGHT.

In that spirit, this worksheet will review CSS best practices including things you should always do when writing your CSS, things you should do your best to avoid and things you should never, ever, EVER do (even though there will be times you want to).

ALWAYS...

1. Keep Your Code Nice and Tidy

Like with HTML, practicing organization with your CSS will help you read through your code and more easily find errors.

To stay organized, here's what you can do:

Indent your CSS properties in your code blocks to help distinguish them from selectors.

NO INDENTS:

```
p.regular-blog-post {
color:black;
font-size:16px;
}
```

YAY INDENTS:

```
p.regular-blog-post {
    color:black;
    font-size:16px;
}
```

Create space in between each code block using a line break or two. They all start to run together pretty quickly - so let the page breathe!

NO SPACE:

```
p.regular-blog-post {
    color:black;
    font-size:16px;
}
article.regular-blog-post {
    width: 600px;
    float:left;
}
```

MUCH BETTER:

```
p.regular-blog-post {
    color:black;
    font-size:16px;
}

article.regular-blog-post {
    width: 600px;
    float:left;
}
```

Utilize uniform, descriptive naming conventions for your class names. If they represent similar ideas, then they should be named similarly. Like let's say you need two different sizes of columns, one that's small and one that's more medium sized. Ideally you'd name them something like `.col-small` and `.col-medium` instead of `.col-small` and `.column-medium` or `.medium-column`.

NO CONVENTION:

```

.full-size {
    width: 1200px;
    margin: 0 auto;
}

.half-column {
    width: 600px;
    float: left;
}

.third-width {
    width: 400px;
    float: left;
}
```

STANDARD CONVENTIONS:

```
.full-width {
    width: 1200px;
    margin: 0 auto;
}

.half-width {
    width: 600px;
    float: left;
}

.third-width {
    width: 400px;
    float: left;
}
```

And for bonus points, you can...

Alphabetize your CSS properties within each code block. Look at how nice that looks:

```
nav ul li a {
    color: #828282;
    font-size: 18px;
    padding-left: 80px;
    text-decoration: none;
    text-transform: uppercase;
}
```

2. Keep Your Code Short & Sweet

Why do we want things short and sweet? One reason is simply information processing time: the longer your document is, the longer it takes to process. And secondly, the longer your document, the more room for error there is. Think about all that room...just waiting for misspellings...and missing semicolons...*shudder.*

Here are a few things you can do to tighten things up:

Use classes for styles that repeat. If multiple parts of your site need to be 16px Roboto in dark gray, just make a class for it! No need to write it out more than you have to.

Choose short names for your classes and IDs. Fewer letters means less room for typos. Period. We could even do this with our column classes that we talked about – `.col-small` could be condensed to `.col-s`, and `.col-medium` could become `.col-m`.

And finally...

Combine your code blocks and declarations wherever possible. Admittedly, this tip is a bit more advanced, but let's talk about how it works.

Let's say your `<article>`s and your `<p>` tags are styled the same way. As you're starting out, they look like this.

```
article {
    color: black;
    font-family: 'Source Sans Pro', sans-serif;
    font-size: 16 px;
}

p {
    color: black;
    font-family: 'Source Sans Pro', sans-serif;
    font-size: 16 px;
}
```

BUT it turns out you can combine these two together, which reads like this:

```
article, p {
    color: black;
    font-family: 'Source Sans Pro', sans-serif;
    font-size: 16 px;
}
```

Now two CSS blocks with the same style information are just in one concise chunk!

Similarly, you can **shorten your declarations**. A declaration is just the set of property and value combinations – or arguments – that appear in each CSS block. In other words, everything inside the curly braces.

For example, here are some margin declarations for our sidebar:

```
.sidebar {
    margin-left:30px;
    margin-right:10px;
    margin-bottom:20px;
    margin-top:0px;
}
```

This code can be condensed into a single line, where each number represents top, right, bottom and left:

```
.sidebar {
    margin: 0 10px 20px 30px;
}
```

So much cleaner!

Of course, when you're just getting started, the most important thing is to understand your code. Go for correct first, and concise second!

3. Organize your CSS like you've organized your HTML

Your HTML will be organized in a hierarchical manner, because that's how the computer reads and interprets it.

```
<body>
     <header>
     </header>

     <main>
            <section>
            </section>

            <footer>
            </footer>
     </main>
</body>
```

```
body {}

header {}

main {}

section {}

footer {}
```

Since your CSS goes hand-in-hand with your HTML, go ahead and put it in the same order! Header styles go before the main body styles, and the footer styles go at the end.

In that same vein, start your CSS doc with your global styles – things like fonts and colors that apply to EVERYTHING, not just a single section. If it applies to everything, it goes at the top.

---

4. Use comments to organize sections of your code!

Oh, sweet comments. You are SO helpful for calling out important ideas or sections! A one-line comment looks like this:

```
/* Hello there! I am a comment. */
```

And a comment with many lines look like this:

```
/* Hello there!
I am a comment with many lines.
So there. */
```

To call out the code for your header area, you might write something like this:

```
/****** HEADER AREA ******/
```

or

```
/*==================== Header Area ================================*/
```

Comments like this make it so you can browse through your code and find things easily.

TRY YOUR BEST TO AVOID...

---

1. Negative margins.

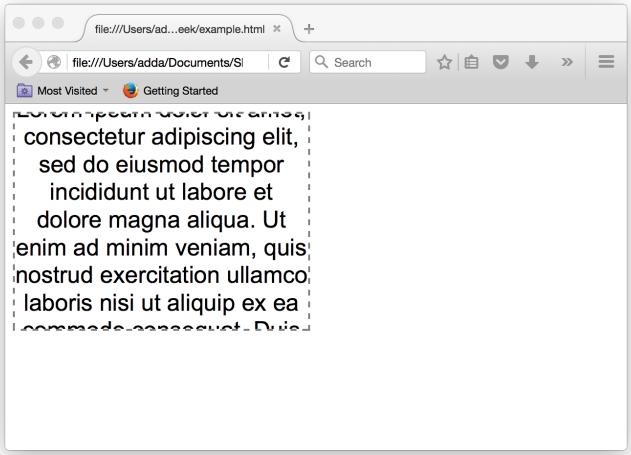
Margins are the space around your HTML element.

But sometimes you don't want any space, or you want the elements to be really close. But what you DON'T want is for those elements to overlap, which is what can happen with negative margins.

But before you go into the negative zone, think about other ways of getting your elements closer to one another, like setting your padding to zero or adjusting an adjacent margin setting. Negative margins are a last resort!

2. Fixed heights.

Here's something else you should avoid any time you can:



Fixed heights + too much text = scrollbars in the middle of your screen. Yikes!

Do whatever you can to avoid them.

3. Using borders on lots of elements.

Because it can throw off your measurements. And it can look super weird. If you're finding you need to use lots of lines to divide up your web page, take a moment to think about whether there are other ways to keep things organized. Create more spacing? Use a background color? Borders should be used sparingly, maybe for an element or two per page – if that.

4. Floating too many things.

Like we talked about, this will just wreak havoc on your site. Float a little, and only what you need.

5. Floating right.

It's one of those things that can easily throw everything off, especially when you start working with responsive layouts. If you need to float, stick with floating left if you can.

NEVER...

---

1. Never do with an image what you can do with CSS.

Like, if you want the background of your webpage to be blue, don't use an image of the color blue. Just change the background color!

Or if you have a button with lots of borders and shadows, take the time to re-create it in CSS!

2. Never Reuse Class Names.

This one might be obvious, but it's worth calling out. It is technically possible to reuse class names if you attach the class name to specific HTML tag selectors like so:

```
p.regular-blog-post {
    color:black;
    font-size:16px;
}

article.regular-blog-post {
    width: 600px;
    float:left;
}
```

Reusing class names is just ASKING for trouble. Think about the last time you had to meet someone who both had the same first name. "Wait, are you Jane Doe or Jane Dewey?" Ummm which one is the right Jane?

Don't torture your computer or–more importantly, *yourself*–when you can just give your classes different names.

3. Never overwrite your CSS.

Ok, this last one...it's a big one. It's also something I almost DON'T want to tell you about...because if you don't know about it, you won't do it! But just in case you've seen some stupid "trick" out there on the internet, let me be super clear: **NEVER OVERWRITE YOUR CSS.**

What do I mean by this?

CSS docs read from top to bottom. So, if you add, say, some new paragraph styles at the bottom of the document, it will overwrite the styling written about it at the top of the document.

So let's say I have some code for my `<h2>` tags that I want to change, I can technically just write the new code below the original code and it will overwrite it:

```
h2 {
    color: blue;
    font-size:72px;
}

h2 {
    color: purple;
    font-size:100px;
}
```

This example illustrates the absolute ridiculousness of doing this, but unfortunately, it won't always be this obvious. Often the new code you are writing is 100's if not 1000's of lines below the original.

So whenever you are editing styles, you must locate the original styles in your code and fix them at the source.