

Advanced CSS Training

CSS Best Practices

Lesson 1, Activity 2: Considerations

In compiling this list of best practices, we've considered the following:

- *Cleaner code.* Does the practice result in cleaner, better organized, more reusable code?
- *Faster development.* Does the practice lead to faster development?
- *Speedier download.* Does the practice result in speedier downloads?
- *Greater control.* Does the practice give the developer greater control over the layout and/or the development process?
- *Accessibility.* Does the practice make your pages more accessible?

Each practice must meet at least one of these conditions.

Use CSS

By now, we hope that you're sold on this. Using CSS for formatting instead of deprecated HTML tags and attributes will result in cleaner, easier-to-maintain code that downloads faster. Using CSS for page layout instead of tables will also result cleaner, easier-to-maintain code that downloads faster and it will make your pages more accessible. And CSS gives you more control over the look and feel of your web pages.

Use External Style Sheets

In almost all cases, you should use external style sheets instead of embedded style sheets. And you should avoid using the `style` attribute at all. Using external style sheets results in better organized, easier-to-maintain code. Because the browsers can cache the style sheet, the code doesn't need to be redownloaded as a visitor navigates through your site.

Organize Your Style Sheets

Some people recommend using different style sheets for different types of content:

- **General styles.** These are usually simple type selectors, perhaps with some child, descendant, or attribute selectors included.
- **Special formatting styles.** These are generally class and id selectors for styles you're likely to reuse often (e.g, "callout" or "important").
- **Layout styles.** These are also generally class and id selectors for laying out your pages.
- **Specific styles.** These are styles specific to a section or individual page on your website.

While we agree that you should have a system for organizing your styles (and it should be well commented), we don't particularly care what the system is, just so long as it's simple to follow and well documented.

We also don't think it's all that important whether you use a single big style sheet broken up into sections or individual style sheets for each section. It depends on how your development team is structured and how people use your website. For example, if visitors tend to come to a certain section of your website and stay in that section, then it would make sense to have a style sheet specific to that section as there is no point in having them download all the styles for other sections of the website. On the other hand, if they're likely to jump from section to section, you might prefer to have the full download up front rather than have new style sheet downloaded each time the user visits a section.

ID the Body

Using an `id` attribute on the `body` provides a hook into the page for your CSS selectors. The `id` needs to be unique for that page, but different pages on your website can share the same `id`. For example, you might set the `id` to "products" for all the pages in the products section of your web site. You can then precede rules specific to your products pages with `#products`, like this:

```
#products h1 { /*declarations go here*/ }
```

Note that your `h1` elements will still inherit all the other `h1` properties set elsewhere unless you explicitly override them.

Some developers prefer to use the `class` attribute rather than the `id` attribute. You can do either or both.

Use CSS Shorthand

You should get used to using CSS shorthand. It will speed up your development time and make your pages download faster.

Combine CSS Rules

This is most easily explained with an example:

Don't do this:

```
ol {
  margin-left:10px;
  padding-left:10px;
}
ul {
  margin-left:10px;
  padding-left:10px;
}
```

Do this instead:

```
ol, ul {
  margin-left:10px;
  padding-left:10px;
}
```

It will download faster and it's easier to maintain.

Use a CSS Reset

We first discussed using CSS Resets in our Advanced CSS Page Layout lesson. We highly recommend this practice if you're using CSS at all heavily. It makes it much easier to create pages that work and look the same across browsers.

Use the "LoVe HAtE" Rule

The "LoVe HAtE" rule specifies the order you should use for rules affecting link pseudo-classes:

1. `:link` - unvisited links
2. `:visited` - visited links
3. `:hover` - links over which the cursor hovers
4. `:active` - links on which the user has clicked

This has to do with the specificity of the selectors and the fact that a link can be in multiple states simultaneously. Because all of these pseudo-classes have the same specificity, the rule that is defined last will take precedence. So, for example, if you define the `:visited` rule after the `:active` rule, then a link that is visited will never appear active.

Open CssBestPractices/Demos/LoVeHAtE.html in your browser to see this in action. The code is shown below:

Code Sample:

CssBestPractices/Demos/LoVeHAtE.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#working a:visited {
  text-decoration:underline;
  color:#f0f;
}
#working a:active {
  text-decoration:none;
  color:#f00;
}

#broken a:active {
  text-decoration:none;
  color:#f00;
}
#broken a:visited {
  text-decoration:underline;
  color:#f0f;
}
</style>
<title>The LoVe HAtE Rule</title>
</head>
<body>
<p>After visiting the linked site, return to this page and click and hold each link. The first link will change color and style. The second will not.</p>
<p>This is because, after the linked page has been visited, when the use clicks down, the link is both active and visited. Which ever rule is defined second will take precedence</p>
<ol>
<li id="working"><a href="http://www.cnn.com">CNN</a></li>
<li id="broken"><a href="http://www.cnn.com">CNN</a></li>
</ol>
</body>
</html>
```

Use HTML Lists for Navigation Menus

Navigation menus present a list of options and should be structured as HTML lists. This is covered in detail in [CSS Lists as Hierarchical Navigation](#).

Avoid CSS Hacks

If you look for solutions to problems you are having getting your pages to appear the same in different browsers, you're very likely to come upon many "CSS hacks". These are tricks that take advantage of browser bugs to get them to behave the way you want them to. One problem with using CSS hacks is that it is often unclear how the code will affect new browsers, future versions of existing browsers, or other devices that view web pages. Another problem is that these hacks can add a lot of complexity to your code, which runs counter to one of the main benefits of CSS - code simplification.

It doesn't take new CSS developers too long to discover that early versions of Internet Explorer are the most problematic of browsers. The most current versions of IE are some of the best at rendering to strict standards but versions 7 and below have some quirks. Microsoft introduced conditional comments with IE5 and they work great. We recommend you use them instead of CSS hacks. In fact, you might want to have a special style sheet devoted to Internet Explorer's quirkiness and use a conditional comment to load it.

Use a CSS Compressor

A CSS compressor is a program that minimizes a CSS style sheet by removing all extra white space. It may also do additional things to make the style sheet smaller. The purpose of a CSS compressor is to make the file size smaller so that it downloads more quickly. You should use one to compress your style sheets. There are many available for free on the web.

There is a simple web-based CSS compressor at <http://www.cssdrive.com/index.php/main/csscompressor>.

Yahoo! provides an excellent free compressor at <http://developer.yahoo.com/yui/compressor>. This one is Java-based and requires a small download. It can also be used for compressing JavaScript files.

Use a Master Style Sheet

This is one we don't feel so strongly about, but a lot of people recommend it so we thought we should give it a nod. Proponents suggest that a master style sheet helps to organize your code. An example is shown below:

```
@import url("reset.css");
@import url("general.css");
@import url("formatting.css");
@import url("layout.css");
```

We don't have anything much against using this, but we don't see that it adds much benefit either. Storing the style sheets in a single folder gives you a similar gain. Using a master style sheet also makes an additional download necessary. Mightn't it be better to call the pages directly from the HTML page?

Summary of Best Practices

Below is a table summarizing the practices discussed in this lesson. The right-most column contains a number between 1 and 3.

1. A 1 means we think you should absolutely follow this practice.
2. A 2 means we think it makes a lot of sense to follow this practice, but recognize that there might be some reasons not to.
3. A 3 means we think you should consider it, but don't consider it mandatory.

Summary of CSS Best Practices

Best Practice	Benefits	#
Use CSS	<ul style="list-style-type: none"> • Cleaner code • Faster development • Speedier download • Greater control 	1

Use External Style Sheets	<ul style="list-style-type: none"> • Cleaner code • Faster development • Speedier download 	1
ID the Body	<ul style="list-style-type: none"> • Greater control 	2
Use CSS Shorthand	<ul style="list-style-type: none"> • Speedier download • Cleaner code • Faster development 	2
Combine Rules	<ul style="list-style-type: none"> • Speedier download • Cleaner code 	2
Use a CSS Reset	<ul style="list-style-type: none"> • Greater control • Cleaner code • Faster development 	1.2
Use the "LoVe HAte" Rule	<ul style="list-style-type: none"> • Speedier download 	1
Use HTML Lists for Navigation Menus	<ul style="list-style-type: none"> • Accessibility 	1
Avoid CSS Hacks	<ul style="list-style-type: none"> • Cleaner code • Faster development 	2
Use a CSS Compressor	<ul style="list-style-type: none"> • Speedier download 	2
Use a Master Style Sheet	<ul style="list-style-type: none"> • Cleaner code 	3