How can I/should I better combine numerous style declarations?

Asked 12 years, 11 months ago Modified 12 years, 11 months ago Viewed 2k times

1

Is it better/faster to (1) stack selectors in style declarations, (2) stack classes in HTML tags, or (3) stack duplicate declarations in unique selectors?

To clarify my question, I will show you examples of each. Each piece of code should accomplish the same end goal, but I'm not sure if one will ultimately be faster to load or whether it is just a matter of preference.

1 (class selectors are duplicated):

```
<style>
.one, .two, .three {color:white}
.one, .two {background:blue;height:30px}
.one, .three {width:800px}
.two, .three {font-size:16pt}
.one {font-size:10pt}
.two {width:650px}
.three {background:#333}
</style>
<div class="one"></div>
<div class="two"></div>
<div class="three"></div>
```

2 (inline classes are duplicated):

```
<style>
.white {color:white}
.bluebg {background:blue}
.heightThirty {height:30px}
.widthEight {width:800px}
.sizeSixteen {font-size:16pt}
.one {font-size:10pt}
.two {width:650px}
.three {background:#333}
</style>
<div class="one white bluebg heightThirty widthEight"></div>
<div class="two white bluebg heightThirty sizeSixteen"></div>
<div class="three white widthEight sizeSixteen"></div>
```

3 (declarations are duplicated):

```
<style>
.one {color:white; background:blue; height:30px; width:800px; font-size:10pt}
.two {color:white; background:blue; height:30px; font-size:16pt; width:650px}
.three {color:white; width:800px; font-size:16pt; background:#333}
</style>
<div class="one"></div>
<div class="two"></div>
<div class="three"></div>
```

I used to create sites using the third method, but I found it difficult to change values, if I decided to change the colour scheme of a site, for example. I would have to do a find/replace for `background:blue` and change it to something else (but find/replace can prove ineffective if I'm being less than consistent). Now I use a combination of the first and second method, with a preference for the first. I decide which elements should share style declarations, and group them together in the stylesheet.

The first method uses less characters (at least in this example), so I know it would make for a smaller filesized HTML file, but I'm not sure about loadtime; and there could be other things I'm unaware of.

Is there a method which ought to be used? I've pointed out the problem that can come with the third method, but are there other problems (with any three) that I don't know about?

html css load-time

Share Improve this question Follow

asked Dec 27, 2011 at 4:06
bozdoz
12.8k ● 8 ● 72 ● 99

3 Answers

Sorted by: Highest score (default)

2

I like the 3rd method and 1st method.

When working on most sites, I try to use the 1st service, without duplicating selectors in so many places (just a few) so that they remain in control.

When using 3rd method, it can be a nightmare to refactor, here comes the role of other languages that translate to CSS, namely SASS and LESS. Those languages allow you to store repeated parts in variables and functions (called "mixins" usually).

I think you'll love LESS http://lesscss.org/ a lot. I jumped to an existing project using it to create some special reusable UI widgets and all CSS was using LESS, and without prior knowledge it was easy enough to understand existing code, and write new parts. Recommended for new websites, while it or 1st method are fine for existing sites.

Share Improve this answer Follow

answered Dec 27, 2011 at 7:00
Meligy
36.5k ● 11 ● 87 ● 114

Hope that's what you are looking for from your question :) – Meligy Dec 27, 2011 at 8:11

I'll keep it open for a bit; I don't expect a definitive answer, so I'm interested in different opinions. – bozdoz Dec 28, 2011 at 3:14

1

This is a problem caused by the W3C's stubborn refusal to add variables to CSS. They argue that cascading styles provides a perfectly acceptable alternative, which it clearly doesn't in many cases, and that CSS is supposed to be simple enough that a lay person could pick it up without programming training—even though at this point you do need to be specially trained to handle its rapidly growing complexity.

Method #1

The problem with the first method is that, because aesthetic design isn't always logically structured, it makes it impossible to logically organize your styles and still use this method.

For example, I typically organize my styles by UI groupings & layouts and by broad sections of the site. There may be a global stylesheet for the entire site (or at least the front-end), and then there may be a stylesheet for a particular page layout or UI type. And within each stylesheet, there are different sections for different UI/layout elements.

The problem is, styles aren't grouped like that within a site design. Your navigation menu is very likely to share a color with your comment box; and your side menu is likely to share a width with your search widget; and you probably have several different classes of UI elements from different sections of the site using different layouts but which share attributes. Organizing your styles by their visual similarity would be chaos, not to mention impossible, since some elements might share 5 different properties with 5 different other elements.

Method #2

So this brings many developers to the second option. But because a site's aesthetics rarely correlate with semantics, they end up using class names like `blue`, `red3` or `primary-color`, `secondary-color`, or `width-1col` and `width-3col`, etc. But this is an equally poor option as it completely violates separation of content and presentation. Perhaps in your current design the search box, category menu, and community polls are all rendered in the right column and have the same width, but that may not always be the case. So if you redesign the site, you'll have to modify your HTML in addition to your CSS.

So what's the solution?

Most developers use a combination of the 3. They use option 1 when aesthetic organization lines up with site/UI organization. They use option 2 when aesthetic structure lines up with semantic structure. And they fall back on option 3 when 1 and 2 fail.

But there *is* a better way... fix the source of the problem: CSS implementation. Real world experience has taught many designers and developers that CSS design doesn't play out in real-life the way that the CSS working group intended it. To implement rich, complex designs while writing maintainable code, CSS needs to have selector inheritance, variables, and mixins.

Projects like Sass and LESS apply the hard lessons and knowledge learned from decades of software engineering to CSS. Modern programming languages don't have complex syntaxes simply to feed the egos of programmers. Programming languages have evolved alongside the discipline of software engineering because programmers have found that certain features and constructs are necessary to write maintainable/manageable code.

This gives developers the tools they need to write efficient, manageable CSS and HTML without changing the CSS specification. You simply write using SCSS or equivalent metalanguages and then let Sass convert it to CSS for you. Some people even pair it with Haml, but I haven't found the need to go that far.

Share Improve this answer Follow

answered Dec 27, 2011 at 7:52

Lèse majesté

8,035 ● 3 ● 34 ● 45

1

as my perspective this one is better

```
<style>
.main div{background:blue;width:800px;font-size:16pt;color:white;}
.one, .two {height:30px}
</style>
```

```
<div class='main'>
<div class="one" style='font-size:10pt'></div>
<div class="two" style='width:650px'></div>
<div class="three" style='background:#333'></div>
</div>
```

Share Improve this answer Follow

answered Dec 27, 2011 at 12:57

Sonal Khunt
1,894 ● 12 ● 20

You would suggest inline styles? – bozdoz Dec 27, 2011 at 19:17