CSS 'schema' how-to

Asked 16 years, 2 months ago Modified 16 years, 1 month ago

Viewed 5k times

8

How does one go about establishing a CSS 'schema', or hierarchy, of general element styles, nested element styles, and classed element styles. For a rank novice like me, the amount of information in stylesheets I view is completely overwhelming. What process does one follow in creating a well factored stylesheet or sheets, compared to inline style attributes?

css styling

Share

Improve this question

Follow

edited Oct 8, 2008 at 14:43

warren

33.4k • 23 • 89 • 128

asked Oct 8, 2008 at 14:30

ProfK

51k • 126 • 414 • 796

See also: stackoverflow.com/questions/72911/… – Nathan Long Nov 19, 2008 at 16:13

8 Answers

Sorted by: Highest score (default)

6

I'm a big fan of naming my CSS classes by their contents or content types, for example a <ul> containing navigational "tabs" would have class="tabs". A header containing a date could be class="date" or an ordered list containing a top 10 list could have class="chart". Similarly, for IDs, one could give the page footer id="footer" or the logo of the website id="mainLogo". I find that it not only makes classes easy to remember but also encourages proper cascading of the CSS. Things like ol.chart {font-weight: bold; color: blue;} #footer ol.chart {color: green;} are quite readable and takes into account how CSS selectors gain weight by being more specific.

Proper indenting is also a great help. Your CSS is likely to grow quite a lot unless you want to refactor your HTML templates evertime you add a new section to your site or want to publish a new type of content. However hard you try you will inevitably have to add a few new rules (or exceptions) that you didn't anticipate in your original schema. Indeting will allow you to scan a large CSS file a lot quicker. My personal preference is to indent on how specific and/or nested the selector is, something like this:

```
    ul.tabs {
    list-style-type: none;
    }
        ul.tabs li {
        float: left;
        }
```

```
ul.tabs li img {
border: none;
}
```

That way the "parent" is always furthest to the left and so the text gets broken up into blocks by parent containers. I also like to split the stylesheet into a few sections; first comes all the selectors for HTML elements. I consider these so generic that they should come first really. Here I put "body { font-size: 77%; }" and "a { color: #FFCC00; }" etc. After that I would put selectors for the main framework parts of the page, for instance "ul#mainMenu { float: left; }" and "div#footer { height: 4em; }". Then on to common object classes, "td.price { text-align: right; }", finally followed by extra little bits like ".clear { clear: both; }". Now that's just how I like to do it - I'm sure there are better ways but it works for me.

Finally, a couple of tips:

1. Make best use of cascades and don't "overclass" stuff. If you give a <ul> class="textNav" then you can access its s and their children without having to add any additional class assignments. ul.textNav li a:hover {}
2. Don't be afraid to use multiple classes on a single object. This is perfectly valid and very useful. You then have control of the CSS for groups of objects from more than one axis. Also giving the object an ID adds yet a third axis. For example:

```
<style>
div.box {
float: left;
border: 1px solid blue;
padding: 1em;
}

div.wide {
width: 15em;
}

div.narrow {
width: 8em;
}

div#oddOneOut {
float: right;
}
</style>

<div class="box wide">a wide box</div>
<div class="box narrow">a narrow box</div>
<div class="box wide" id="oddOneOut">an odd box</d
```

3. Giving a class to your document <body> tag (or ID since there should only ever be one...) enables some nifty overrides for individual pages, like hilighting the menu item for the page you're currently on or getting rid of that redundant second sign-in form on the sign-in page, all using CSS only. "body.signIn div#mainMenu form.signIn { display: none; }"

I hope you find at least some of my ramblings useful and wish you the best with your projects!

Share Improve this answer edited Nov 19, 2008 at 1:17

Follow

answered Nov 19, 2008 at 1:08

Ola Tuvesson

5,181 ● 2 ● 30 ● 38

Are you kidding me? Most developers can indent correctly and you definitely should indent the properties in the selector (so you use 4 spaces before `float: left` etc.
– Luca Steeb Sep 28, 2016 at 9:52

No joke. CSS is not white space sensitive. Nor does the spec say anything about indentation. And I'm a stickler for formatting and standards. Not indenting the declarations is a personal choice; if you don't like it go ahead and indent them! I see a lot of people putting short (and sometimes not so short!) CSS blocks on a single line which I find annoying, though I sometimes do this in JS. Oh and I indent my CSS (and JS, and HTML) files with tabs, not spaces. Scream all you like. – Ola Tuvesson Sep 29, 2016 at 1:28

4

There are a number of different things you can do to aid in the organisation of your CSS. For example:

- Split your CSS up into multiple files. For example: have one file for layout, one for text, one for reset styles etc.
- Comment your CSS code.
- Why not add a table of contents?
- Try using a CSS framework like 960.gs to get your started.

It's all down to personal taste really. But here are a few links that you might find useful:

- http://www.smashingmagazine.com/2008/08/18/7-principles-of-clean-and-optimized-css-code/
- http://www.smashingmagazine.com/2008/05/02/improving-code-readability-with-css-styleguides/
- http://www.louddog.com/bloggity/2008/03/css-best-practices.php
- http://natbat.net/2008/Sep/28/css-systems/

Share Improve this answer Follow

edited Oct 9, 2008 at 7:30

answered Oct 8, 2008 at 14:46

Lee Theobald

8,567 ● 12 ● 51 ● 59

3

Think of the CSS as creating a 'toolkit' that the HTML can refer to. The following rules will help:

- Make `class` names unambiguous. In most cases this means prefixing them in a predicatable way. For example, rather than `left`, use something like `header_links_object2_left`.
- Use `id` rather than `class` **only** if you know there will only **ever** be one of an object on a page. Again, make the `id` unambiguous.

- Consider side effects. Rules like `margin` and `padding`, `float` and `clear`, and so on can all have unexpected consequences on other elements.
- If your stylesheet is to be used my several HTML coders, consider writing them a small, clear guide to how to write HTML to match your scheme. Keep it simple, or you'll bore them.

And as always, test it in multiple browsers, on multiple operating systems, on lots of different pages, and under any other unusual conditions you can think of.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:55

Marcus Downing

10.1k ● 12 ● 65 ● 86

I don't think using a style dependant name (i.e. calling styles "head_links_left") is a good idea. What if you decide to move the links in the left to the right ? You should change the style name ! That's not easy to mantain. I prefer to use "semantic" names, for example just "head_links". – Guido Oct 9, 2008 at 7:54

2

Putting all of your CSS declarations in roughly the same order as they will land in the document hierarchy is generally a good thing. This makes it fairly easy for future readers to see what attributes will be inherited, since those classes will be higher up in the file.

Also, this is sort of orthogonal to your question, but if you are looking for a tool to help you read a CSS file and see

how everything shakes out, I cannot recommend Firebug enough.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:44

Ryan

9,928 ● 7 ● 44 ● 57

I love FireBug's Inspect Element. In fact I love all of FireBug. – ProfK Oct 8, 2008 at 15:25

2

The best organizational advice I've ever received came from a presentation at An Event Apart.

Assuming you're keeping everything in a single stylesheet, there's basically five parts to it:

1. Reset rules (may be as simple as the `* {margin: 0; padding: 0}` rule, Eric Meyer's reset, or the YUI reset)
2. Basic element styling; this is the stuff like basic typography for paragraphs, spacing for lists, etc.
3. Universal classes; this section for me generally contains things like `.error`, `.left` (I'm only 80% semantic), etc.
4. Universal layout/IDs; `#content`, `#header`, or whatever you've cut your page up into.
5. Page-specific rules; if you need to modify an existing style just for one or a few pages, stick a unique ID

high up (body tag is usually good) and toss your overrides at the end of the document

I don't recommend using a CSS framework unless you need to mock something up in HTML *fast*. They're far too bloated, and I've never met one whose semantics made sense to me; it's much better practice to create your own "framework" as you figure out what code is shared by your projects over time.

Reading other people's code is a whole other issue, and with that I wish you the best of luck. There's some truly horrific CSS out there.

Share Improve this answer

Follow

answered Nov 19, 2008 at 1:41

One Crayon

19.2k ● 11 ● 36 ● 40

1

Cop-out line of the year: it depends.

How much do you need to be styling? Do you need to change the aspects of alomost every element, or is it only a few?

My favorite place to go for information like this is CSS Zen Garden & A List Apart.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:43

warren

33.4k ● 23 ● 89 ● 128

There are two worlds:

1. `The human editor perspective:` Where CSS is most easily understand, when it has clear structure, good formatting, verbose names, structured into layout, color and typesetting...
2. `The consumer perspective:` The visitor is most happy if your site loades quickly, if it look perfect in his browser, so the css has to be small, in one file (to save further connections) and contain CSS hacks to support all browsers.

I recommend you to start with a CSS framework:

- Blueprint if you like smaller things
- or YAML for a big and functional one

There is also a list of CSS Frameworks...

And then bring it in shape (for the browser) with a CSS Optimizer (p.e. CSS Form.&Opti.)

You can measure the Results (unpotimized <-> optimized) with YSlow.

edited Oct 8, 2008 at 14:52

answered Oct 8, 2008 at 14:47

Andre Bossard
6,281 ● 37 ● 53

1

A few more tips for keeping organized:

- Within each declaration, adopt an order of attributes that you stick to. For example, I usually list margins, padding, height, width, border, fonts, display/float/other, in that order, allowing for easier readability in my next tip
- Write your CSS like you would any other code: indent! It's easy to scan a CSS file for high level elements and then drill down rather than simply going by source order of your HTML.
- Semantic HTML with good class names can help a lot with remembering what styles apply to which elements.

Share Improve this answer

Follow

answered Oct 8, 2008 at 15:36

Evan Davis

36.5k ● 7 ● 52 ● 58