CSS Architecture

Reset and Normalize

CSS libraries

Reset by Eric Meyer

- Idea: default browser settings are evil – they must die!
- Your first CSS file: `reset.css`
- Everything looks plain
- Seemed like a good idea (in 2011)
- Problems:
  - Additional effort
  - Different browsers render pages differently – still need browser testing
  - They don’t save time

Normalize by Nicolas Gallagher

- Idea: browsers render webpages differently; pick a standard and force them all to follow it
- Your first CSS file: `normalize.css`
- For better cross-browser consistency
- Fixes browser bugs (whenever possible)
- Leaves browser defaults intact (unlike reset.css)

Other CSS Selectors

Really esoteric stuff

Child and Sibling Selectors

- Child selectors
  - `div > h1 {…`
  - Any h1 that's a DIRECT descendent of a div tag only
- Adjacent sibling selectors
  - `h1 + p {…`
  - a p that's DIRECTLY after an H1
- General sibling selectors
  - `h1 ~ p {…`
  - Every p that comes after an H1 within the same parent

Attribute Selectors

`a[href] {…`

- any a tag with an href attribute

`a[href="home"] {…`

- any a tag with "home" as the href

`a[href~="home"] {…`

- any a tag with the whole word "home" somewhere in the href

`a[href^="http://"] {…`

- any a tag with an href that starts with "http://"

`a[href$=".pdf"] {…`

- any a tag with an href that ends with ".pdf"

`a[href*="main"] {…`

- any a tag with an href that contains with "main" anywhere in it

Remember…

- Browsers read from top-to-bottom (HTML, CSS, JS) - order matters!
- Specificity matters: descendent selectors – a powerful tool:

```
.main-content article .lead div img {
    padding: 5px;
    border: 1px solid #cccccc;
}
```

…will override any other IMG styles

…will be used in that situation only, never to be used again!

- Better…

```
.picture-frame{
    padding: 5px;
    border: 1px solid #cccccc;
}
```

Specificity

```
<div id="test">
  <p>
    <span>Lorem…
    </span>
    ipsum…
  </p>
</div>
```

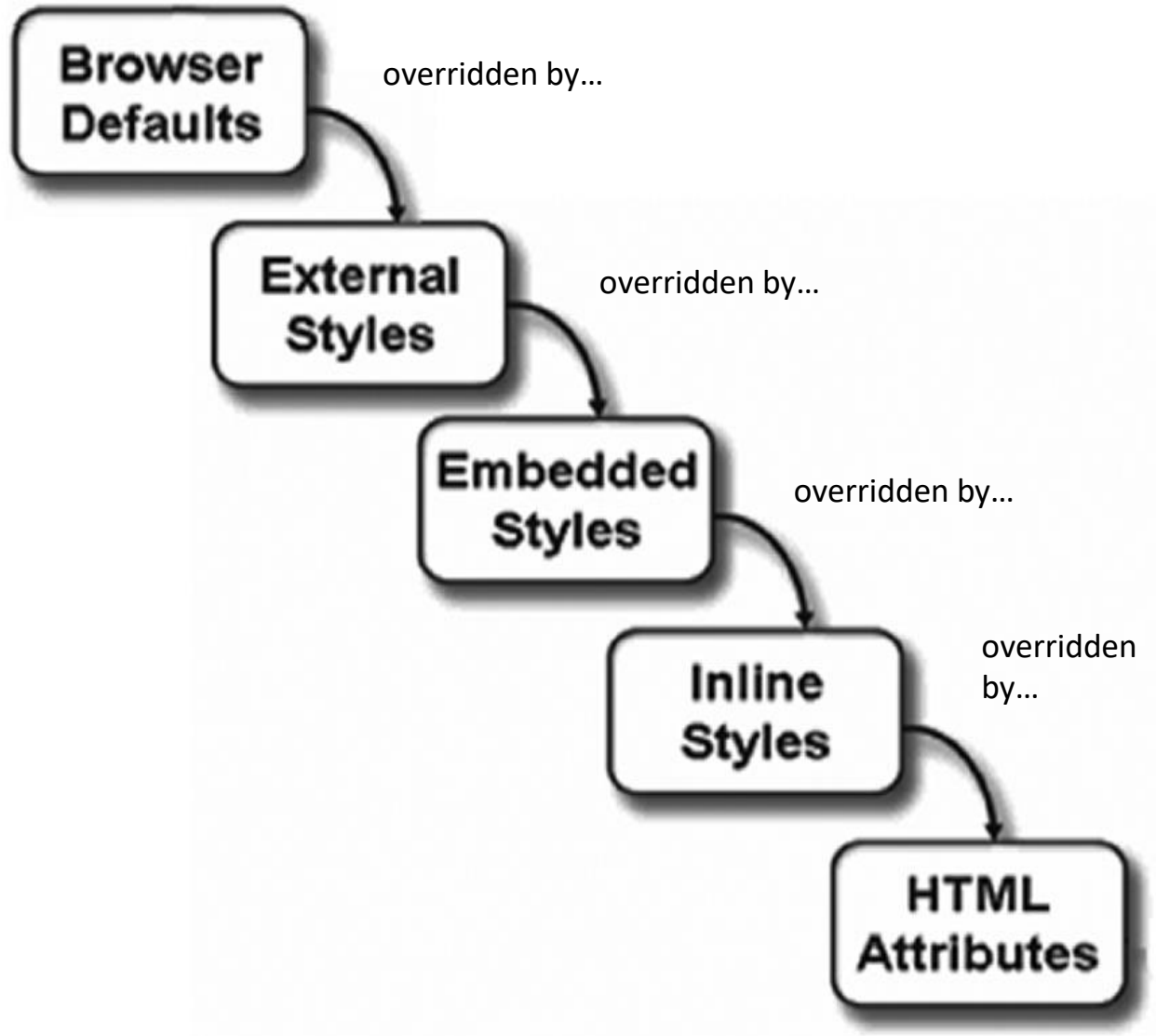
```
p { color: red }
div p { color: blue }
div#test p { color: green } <-- winner


span { background-color: yellow; }
p span { background-color: salmon; }
#test p span { background-color: silver; } <-- winner
```

The highest specificity wins

CSS – The Wild Wild West

- Multiple levels – external, embedded, inline
- Multiple CSS files – local and remote
- Intricately designed websites …thousands (tens of thousands?) of lines of CSS
- CSS Frameworks on top of CSS Frameworks



SMACSS
(Scalable and Modular Architecture for CSS)

- Pronounced "smacks"
- Style guide (not a library, not a template)
- Jonathan Snook – web developer, author, Canadian
  - Circa 2011
  - One of the first published/popular methodologies for CSS organization
  - New concepts introduced:
    - Groupings: Base, Layout, Module, State, Theme
    - "depth of applicability" and the importance of "shallow design"

SMACSS' structure

Create sections – defined by purpose

- By filename or by comment-sections

```
/* ********************
   Base styles
   ******************** */

body { ...
h1 { ...
/* ********************
   Layout styles
   ******************** */

...
```

Sections:

- Base
  - Defaults for the website
  - Exclusively element selectors
  - e.g. `p { color: #333333; }`
- Layout
  - Page "areas" – physical locations
  - e.g. `#hero { border: 5px…`
- Modules
  - Reusable, generic, simple
  - e.g. `.danger{ color: red; }`
- State
  - Inserted by JavaScript
  - e.g. `.is-current {…`
- Theme…

SMACSS' Depth of Applicability Rules

- Too much specificity --> too many rules, too much repeating yourself
- Shallow is better
- Follow the DRY principle! (Don't Repeat Yourself)
- Be *classy*
- Write small, modular rules, and lots of them
- In HTML, this is okay…

```
<div id="sidebar" class="loud special fancy">...
```

…but beware of *classitus*

Kostin's Opinionated SMACSS

- Base includes "normalize.css"
- Module
  - Module section is next after Base (not Layout)
  - Modules should do just about EVERYTHING!
  - Modules should be classes only (reusable and combine-able)
- Layout = shame
  - Layout is for page specific styling (unique things)
  - Use sparingly
  - Try to factor-out Layout rules – turn them into new Modules
  - Layout rules should be IDs only

Since SMACCS

More opinionated methodologies

Flavors

- SMACSS
  - https://smacss.com/
- BEM
  - https://en.bem.info/
- OOCSS
  - http://www.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/
- Point North
  - http://pointnorth.io/#base-browser-styling
- ITCSS
  - http://itcss.io/
- Title CSS
  - http://www.sitepoint.com/title-css-simple-approach-css-class-naming/
- Idiomatic CSS
  - https://github.com/necolas/idiomatic-css
- Atomic Design
  - http://patternlab.io/resources.html
- SUIT CSS
  - https://github.com/suitcss/suit/blob/master/doc/naming-conventions.md#u-utilityname
- Kickoff CSS
  - https://trykickoff.github.io/learn/css.html#namingscheme

…and endless advice: Snugug (http://snugug.com/musings/css-strategy)