Why Different Approaches To Naming Classes

CSS is fairly unique in the world

- Separates content from appearance
- Applies appearance based on structure

Class names are important

- Better define your structure
    - And **communicate** that structure
- Like naming the rooms of your house
    - By purpose
    - Not just walls/floor/ceiling/doors

Different Needs

Different pages/sites have different needs

- News/Informational sites
- Govt Agencies/Universities/Companies
- Different Product Areas for Sales/Support

Different breakdowns of **Audience**, **Context**, and **Goals**

CSS by Structure

You COULD add styling by structure

- `div > ul > li > a`
- Pro: Added HTML that matches gets styling
- Con: Sometimes that isn't intended or desired
- Con: Different structures will have different specificities

What alternatives exist?

Semantic CSS

Give elements class names that describe the **purpose**

```
<div class="profile">
    <img class="avatar" src="avatar.png" alt="..."/>
    <span class="name">Jorts</span>
</div>
```

Using Semantic CSS

Approach: Use Cascading, use semantic class names

Pros:

- Original "intent" of CSS
- No conflicts between class names and visuals
    - Keeps code maintainable
- Easily expands to more HTML

Cons:

- Can expand past your desire
- "Style all EXCEPT these" gets harder

Semantic CSS Example

- Style headings, lists, paragraphs
- Add specific classes for specific but common purposes (menus, callouts, etc)

```
<nav class="site-nav">
  <ul class="menu">
    <li><a href="about.html">About</a></li>
    <li><a href="famous.html">Famous Cats</a></li>
    <li><a href="lolcats.html">LOLCats</a></li>
  </ui>
</nav>
```

See also: **http://www.csszengarden.com/**

Block__Element--Modifier (BEM)

A "fancy" Semantic CSS

Approach: Avoid wide rules, use semantic class names

Pros:

- Same benefits as Semantic
- Avoids unexpected side-effects

Cons:

- May have one-use classes
- What is a Block and Element may feel arbitrary

BEM Approach

"Block": area of content that gets styled

- Ex: `<nav class="menubar">`

"Element": subsection of that content

- Class name: `BLOCK__ELEMENT`
- Ex: `<ul class="menubar__menu">`

"Modifier": If an element has multiple variations

- Class name: `BLOCK__ELEMENT--MODIFIER`
- Ex: `<ul class="menubar__menu--open">`

BEM Benefits

- Reusable semantic structures
- Simplify structures by having a pattern to follow
- Minimize the specificity of rules
    - All selectors just 1 class
        - 2 in case of Modifier

See HTML of: **http://getbem.com/** for example

See **http://getbem.com/faq/** for good details

We will revisit BEM later in semester

BEM Casing Syntax

Technically BEM is not `kebab-case`

- It is up to three parts connected
- `BLOCK__ELEMENT--MODIFIER`
    - Max of ONE of each part!
- Ex: `primary-nav__link--active`
- `BLOCK`, `ELEMENT`, and `MODIFIER` each `kebab-case`
    - Then `__` and `--` separate THOSE parts

Whenever I say "class names must be `kebab-case`"

- Correctly done BEM is still permitted
- Or just do semantic kebab-case, no BEM

Utility First CSS

Approach: No semantics

- Name based on part of appearance

Pros:

- Once defined, easy to apply new content
- What you describe is what you get
- Less CSS

Cons:

- MANY more class names in HTML
- Design changes = Many HTML changes

Utility First Examples

Many common libraries: Tailwind CSS, etc

```
<ul class="flex">
  <li class="mr-6">
    <a class="text-blue-500 hover:text-blue-800" href="#">Active</a>
  </li>
  <li class="mr-6">
    <a class="text-blue-500 hover:text-blue-800" href="#">Link</a>
  </li>
  <li class="mr-6">
    <a class="text-blue-500 hover:text-blue-800" href="#">Link</a>
  </li>
  <li class="mr-6">
    <a class="text-gray-400 cursor-not-allowed" href="#">Disabled</a>
  </li>
</ul>
```

Utility First is hard for this course

Because utility first

- Involves an external library
- Or a ton of work

Utility First is HAPPY to "avoid class names"

- But I'm TEACHING class names

Starting out

Start with Semantic CSS

- Easy to shift to BEM
- Utility First avoids the concepts we are teaching
    - Both the good and the bad
- You WILL encounter Semantic
    - Need to understand it
- In future:
    - All approaches valid depending on needs

Summary - Approaches

- CSS are rules without a structure
- "Approach": How you organize your structures
- Three general common styles
    - Semantic
    - BEM (A fancy version Semantic)
    - Utility First
- Nothing formal, these are rough labels

Summary - Semantic

- Makes use of semantic class names
- Very reliant on structure
- Can easily have a rule apply in multiple places
    - Can be good or bad
- Easy to have same class names for different structures
- Good introduction to CSS

Summary - BEM

- A convention about naming semantic classes
- Goal is to "flatten" specificity
- Still can apply a rule to multiple places
    - More likely to be intended effect
- Fewer "broad" rules that apply multiple places
- An organized semantic approach

Summary - Utility First

- Discards semantics entirely
- Repetitive
- Controls precise look
- Less switching between HTML/CSS files
- Requires upfront investment to define standards

NOT USED IN THIS COURSE