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/**

BEM

Not today:

- We will learn a naming style
- BEM (Block Element Modifier)

BEM is still semantic

- A pattern of how to name multiple related classes
- Reduces issues with similar names

But for now we will use semantic kebab-case names

- Make sure you know what I mean by **kebab-case**

Utility Classes

Semantic classes

- Named for what the element is/represents
- Not for the change the class imposes
    - Not for appearance

Utility classes

- Not named for what the element is/represents
- Named for what change the class imposes

Utility Class Examples

- `fade-in`
- `large`
- `fill-page-width`

These names usually describe some visual effect, but could name anything that is created by CSS.

The focus is on the effect created, not what the element itself *is* that makes it get that effect

Utility First / Atomic CSS

Approach: No semantics

- Focus on the added properties (Utility "first")
- Trying for 1 class = 1 CSS property

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

Similar to putting styles directly on an element

- Still allows Semantic HTML
    - But class names aren't semantic
- Some love it
    - Don't know CSS?
    - Know CSS but dislike it?
- Some hate it
    - Clutters up HTML
    - CSS class names have benefit

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
- Common styles
    - Semantic
        - BEM (A fancy semantic)
    - Utility First
    - Atomic 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 - 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