Why Different Approaches

CSS is fairly unique in the approach

- separates content from appearance
- applies appearance based on structure

Different Needs

Different pages/sites have different needs

- A newspaper wants a consistent appearance that mimics a newspaper
- A page may expect to have radically different layouts for mobile/desktop
- A site might have content areas from different sources they don't want to conflict
- A site might have general rules and specific overrides

Cascading

Categorization of rules (selectors that apply to elements in different parts of the structure) can cause or solve problems

- Pro: Added HTML that matches selector gets the styling with no effort
- Con: Sometimes that isn't intended or desired

Semantic CSS

Approach: Use Cascading, use semantic class names

Pros:

- Original "intent"
- 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="menubar">
  <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)

Approach: Avoid cascading, use semantic class names

Pros:

- Same benefits as Semantic
- Avoids unexpected side-effects

Cons:

- May have one-use classes
- Definitions of Block and Element may be arbitrary

BEM Approach

"Block": area of content that gets styled

- e.g. `<nav class="menubar">`

"Element": subsection of that content

- class name: `BLOCK__ELEMENT`
- e.g. `<ul class="menubar__menu">`

"Modifier": If an element has multiple states

- class name: `BLOCK__ELEMENT--MODIFIER`
- e.g. `<ul class="menubar__menu--open">`

BEM Benefits

- Still get the reuse of semantics by having the same semantic structures in HTML
- Minimize complex structures by having a pattern to follow
- Minimize the specificity of rules (no required combined classes in selectors), making it easier to handle overrides

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

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

Utility First CSS

Approach: No semantics, base on part of appearance

Pros:

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

Cons:

- Needs library or upfront investment
- 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

and I deny external libraries in this course

Starting out

I recommend Semantic CSS to start with

- Easy to shift to BEM
- You avoid the pain of Utility First in the class
    - Even if you want utility first, you WILL encounter Semantic
        - Need to understand it
- In future, all approaches valid depending on needs