# **Block\_\_\_Element--Modifier (BEM)**

A "fancy" Semantic CSS

Approach: Avoid wide rules, use semantic class names

#### Pros:

- Same benefits as Semantic
- Avoids unexpected cascades
- Makes Naming easier

#### Cons:

- May have one-use / more class names
- What are Block and Element non-obvious

# **BEM Style**

- BEM style is an option for this course
  - A classname is either BEM or it is not
- BEM style:
  - BLOCK, ELEMENT, MODIFIER each semantic
    - Each a meaningful name
  - BLOCK
  - LOCK--MODIFIER
  - .BLOCK ELEMENT
  - BLOCK\_\_ELEMENT--MODIFIER
- BEM is for humans
  - "It works" is talking about the computer

# **BEM Approach**

**Block**: area of content that gets styled

• Ex: <nav class="nav">

**Element:** subsection of that content

- Class name: BLOCK\_\_ELEMENT
- Ex:
- Usually not the "HTML element"

**Modifier**: If an element has multiple variations

- Class name: BLOCK\_\_ELEMENT--MODIFIER
- Ex:

### **BEM Casing Syntax**

Technically BEM is not kebab-case

- Up to three connected kebab-case parts
- 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

#### **BEM Block**

- BLOCK
- A "chunk" of the page
- Just a semantic label
  - Including being kebab-case
  - Multiple words connected by single hyphen
  - All lowercase
- No different than non-BEM Semantic labels
  - Benefit comes when we add elements
- Blocks can appear inside other blocks
  - Use when you will use/style parts inside

#### **BEM Element**

- BLOCK\_\_ELEMENT
  - Exactly two underscores!
  - Only two parts
    - o One Block
    - One Element
- ELEMENT is subblock
  - Might be actual semantic element type
  - Might be a semantic class name
    - Same kebab-case naming rules

#### **BEM Modifier**

- | BLOCK\_\_ELEMENT--MODIFIER
- Will be BLOCK\_ELEMENT as well!
  - Have both class names
- Exactly two underscores
- Exactly two hyphens (before MODIFIER)
- MODIFIER itself is kebab-case
- MODIFIER name is:
  - A semantic *state* 
    - ∘ open, active, etc
  - Or a variation
    - Same basic element, different version

## **Block Modifier, No Element**

A Block can have a modifier as well

- BLOCK—MODIFIER
- Will be BLOCK as well
  - Have both class names
- No underscores
- Exactly two hyphens (before MODIFIER)
- BLOCK and MODIFIER each kebab-case
- MODIFIER name is:
  - A semantic *state*
  - Or a variation

# Not Using BEM? Not a problem!

- Course requires for EVERY class name:
  - Semantic
    - Name describes block/contents
  - kebab-case
    - No underscores
- Outside of course
  - Lots of different rules sets (still rules!)
- Either way, want to use classes
  - Styled elements
  - Elements targeted by JS
  - Elements modified by JS

#### Common issues with the use of CSS classes

- When to have classes
- What name to use
- BEM or not?

These become more significant as we add JS

### When to have a class (Generally)

- Element selectors for "defaults"
  - ul> will look like this "by default"
  - <button> will look like this by default
  - Includes if such elements are added in future
- If you are styling HTML that you don't control
  - Use whatever works
- Otherwise, use a class
  - BEM will usually be a **single** class
  - Non-BEM still be based on a class
    - May be a scope (Ex: \_footer a)
  - BEM or not: may use attributes

# You generally want to style using a class name

- Ideally, styling a single class name
  - BEM is good for that
  - But BEM not required
- Not by general element type
  - Makes it harder to add elements to page
- Not by complex nested selectors
  - Specificity can be different
- Class names help explain your HTML structure

#### Rule of thumb for classes

- Will this element be styled?
  - Give it a class
- Will this element be interactive?
  - Give it a class
- Does this element collection represent a concept?
  - Give it a class

## Why do we want classes?

- All selectors using single classes
  - Same specificity
- Makes CSS more readable
- CSS breaks less often when HTML changes

```
main div div a {
   /* styling */
}
```

VS

```
.card__link {
  /* styling */
}
```

# Javascript will have similar issues

#### JS does

• ...not use specificity

#### BUT, JS does

- ...want to work after minor HTML changes
- ...want clarity on what an element represents

More on this when when we get to JS and the DOM