Chapter 52: CSS design patterns

These examples are for documenting CSS-specific design patterns like BEM, OOCSS and SMACSS.

These examples are NOT for documenting CSS frameworks like Bootstrap or Foundation.

Section 52.1: BEM

BEM stands for `Blocks, Elements and Modifiers`. It's a methodology initially conceived by Russian tech company Yandex, but which gained quite some traction among American & Western-European web developers as well.

As the same implies, BEM metholology is all about componentization of your HTML and CSS code into three types of components:

- **Blocks:** standalone entities that are meaningful on their own

  Examples are `header`, `container`, `menu`, `checkbox` & `textbox`

- **Elements:** Part of blocks that have no standalone meaning and are semantically tied to their blocks.

  Examples are `menu item`, `list item`, `checkbox caption` & `header title`

- **Modifiers:** Flags on a block or element, used to change appearance or behavior

  Examples are `disabled`, `highlighted`, `checked`, `fixed`, `size big` & `color yellow`

The goal of BEM is to keep optimize the readability, maintainability and flexibility of your CSS code. The way to achieve this, is to apply the following rules.

- Block styles are never dependent on other elements on a page
- Blocks should have a simple, short name and avoid `_` or `-` characters
- When styling elements, use selectors of format `blockname__elementname`
- When styling modifiers, use selectors of format `blockname--modifiername` and `blockname__elementname--modifiername`
- Elements or blocks that have modifiers should inherit everything from the block or element it is modifying except the properties the modifier is supposed to modify

Code example

If you apply BEM to your form elements, your CSS selectors should look something like this:

```
.form { }                          // Block
.form--theme-xmas { }              // Block + modifier
.form--simple { }                  // Block + modifier
.form__input { }                   // Block > element
.form__submit { }                  // Block > element
.form__submit--disabled { }        // Block > element + modifier
```

The corresponding HTML should look something like this:

```html
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input class="form__submit form__submit--disabled" type="submit" />
</form>
```