Introduction to CSS

Jerry Cain
CS 106AX
November 13, 2019
Introducing CSS

• Recall that there are three technologies that are typically employed when implementing interactive web pages.
  – **HTML**: which dictates the structure and content of a web page
  – **JavaScript**: which implements animations and user interactivity and otherwise control the behavior of the elements
  – **CSS**: short for *Cascading Style Sheets*, controls layout, formatting, and presentation

• Any nontrivial web application will require a large amount of HTML, CSS, and JavaScript.
  – As web applications grow in complexity, it’s important to decouple the HTML, CSS, and JavaScript as much as possible that that changes to an HTML document never break a CSS rule or compromise the execution of JavaScript event handler.
  – Web applications that successfully separate content, presentation, and interactivity are easier to modify and maintain.
CSS Declarations

• Web designers control the presentation of a page using one or more CSS declarations, each of which is structured like this:

```
property-name: property-value;
```

• The property name is one of several CSS keywords (567 according to https://www.w3.org) that identify some detail related to presentation.

• The set of possible property values depends on the name.
  - `background-color` can take on any legal JavaScript or CSS color, e.g. `green`, `rgb(85, 172, 238)`, or `#E98725`.
  - `text-align` governs the horizontal alignment of text and can be set to `left`, `right`, `center`, or `justify`.
  - `display` controls how an element is displayed and can be bound to `inline`, `block`, `inline-block`, `flex`, and `none`. 
CSS Declarations: Photo Wall

- The easiest way to include a CSS declaration is to include a style attribute as part of a tag, as with:

  ```html
  <div style="text-align: center;">
    <img src="david-goliath.jpg" style="height: 144px; border: dotted 1px;"/>
    <img src="elsa-anna.jpg" style="height: 144px; border: dashed 1px;"/>
    <img src="bert-ernie.jpg" style="height: 144px; border: ridge 1px"/>
  </div>
  ```

- The above HTML fragment produced the following:
CSS Declarations

• In the last example, each `img` tag is decorated with its own `style` attribute.
  – All values share the same `height` property value, so the displayed images are scaled to the specified height and are vertically aligned.
  – Unless otherwise specified, image elements are `inline-block`, which means they flow left to right as text does (that’s the `inline` part) and their width and height can be specified (that’s the `block` part).
  – Each value specifies its own `border` property value. The border width is 1px everywhere, but each specifies its own border style.

• The `div` tag is styled so all text—or rather, all inline elements like `img`—under its jurisdiction are horizontally centered.

• You don’t need to memorize all 500+ property names and the spectrum of possible property values for each. You just need to be able to read CSS declarations and understand them.
CSS Declarations: The Bad

- While adding **style** attributes to HTML tags is relatively straightforward, it’s rarely the best approach.
- One obvious drawback is that the styling is isolated to just the one tag. In the images example, we repeated portions of the **style** attribute value for each of the three **img** tags.
  - If the image height needs to change—maybe you decide it should be 96 pixels instead of 144—then you’d need to change the property value in three separate places.
  - It’s a form of code replication—yes, CSS is code!—that’s borderline forbidden no matter the language.
- A more serious drawback is that placing CSS in an HTML file violates a core web software engineering principle that HTML, CSS, and JavaScript remain separate.
  - The engineer writing the HTML shouldn’t be responsible for inlining CSS style attributes if CSS isn’t their expertise.
CSS Rules

• To mitigate the code replication problem, we define styles that apply not to a single element but to an entire document. The most common practice for doing so is defining a set of CSS rules, each of which takes the following form:

```
selector {
    property-name-1: property-value-1;
    property-name-2: property-value-2;
    property-name-3: property-value-3;
    etc...
}
```

• To help mitigate both problems, CSS declarations are typically placed in separate files and referenced by the HTML file that depends on them. These files are called *stylesheets*.

• HTML and CSS can’t be completely decoupled, since each impacts the other. We rely on the two mechanisms above to make a web application’s architecture as modular as possible.
CSS Rules

• The selector portion of the rule can take on many forms, the most basic of which take on the following structure:
  – A hash followed by the DOM id of an element, which states the rule should be applied to the element with that id.
  – A period followed by a name, which defines a rule that HTML tags adopt by defining a class attribute whose value includes the name. So, `<span class="answer incorrect final">43</span>` would ensure the 43 of interest would be presented in **bold**, angry **red**.
  – The name of an HTML tag, which specifies that all elements of that type follow the rule. The rule on the right would mandate that all images be 72px square.

```css
#keyboard {
  font: 12px Courier;
  color: #080019
}

.incorrect {
  font-style: bold
  color: red
}

img {
  width: 72px;
  height: 72px;
}
```
Photo Wall, Take II

- Here is an implementation of the photo wall application that makes proper use of CSS rules and stylesheets.

File: images.html

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Photo Wall, Take II</title>
    <link rel="stylesheet" type="text/css" href="images.css"/>
  </head>
  <body>
    <div id="photo-wall">
      <img src="david-goliath.jpg" class="portrait dotted-frame;"/>
      <img src="elsa-anna.jpg" class="portrait dashed-frame;"/>
      <img src="bert-ernie.jpg" class="portrait ridge;"/>
    </div>
  </body>
</html>
```

File: images.css

```css
#photo-wall {
  text-align: center;
}

.portrait {
  height: 288px;
  border-width: 1px;
}

.dotted-frame {
  border-style: dotted;
}

.dashed-frame {
  border-style: dashed;
}

.ridge-frame {
  border-style: ridge;
}
```
The Cascade of CSS

- The selector component is often a waterfall of selectors that telescope to identify a specific category of HTML elements.

```css
#keyboard span.highlighted {
  color: #CC3333; /* Persian red */
}

#lampset span.highlighted {
  color: #FF9999; /* Light salmon pink */
}
```

- The first rule applies to `span` elements that:
  - are below the element with id "keyboard" in the DOM tree
  - include "highlighted" in its list of `class` attribute values
- The second rule also applies to `spans` satisfying the same two constraints, but replace "keyboard" with "lampset"
- The `cascade` is the algorithm browsers use to decide what rules apply to what elements, particularly in the face of conflicts.
CSS and JavaScript

- JavaScript can be used to modify the presentation of existing elements.
- The JavaScript standard recognizes that per-element style manipulation is so common, that it exposes the `class` attribute via `element.classList`.
- The `classList` attribute stores its class names in an array-like object. So, the element representing:

  `<span class="answer incorrect final">43</span>`

would include a `classList` attribute with three classes. The only three `classList` methods of interest are illustrated here:

```javascript
function regradeQuestion(elem) {
    if (elem.classList.contains("incorrect")) {
        elem.classList.remove("incorrect");
        elem.classList.add("correct");
    }
}
```
The End