Day 3 Notes

#coding-intro

In this note, we'll discuss methods of beautifying a webpage. Knowledge of how to create an HTML webpage is a strict prerequisite.

CSS

If HTML is the structure of a webpage, then CSS (**C**ascading **S**tyle **S**heets) is the layer of decoration that goes on top. While HTML can exist on its own (you can build a boring webpage if you'd like), it's impossible for CSS to be a standalone webpage.

Like HTML, CSS comes with its own special syntax. You can include a **CSS stylesheet** within an HTML file, or create a separate CSS document and import it. Here is an example of **included CSS**:

```
<!DOCTYPE html>
<html>
    <head>
        <style type="text/css">
            p {
                font-size: 12pt;
                    color: green;
            }
        </style>
    </head>
    <body>
        <h1>Hello world</h1>
        <p>Hello <b>world</b></p>
    </body>
</html>
```

> Remember that HTML isn't sensitive to indentation! It is common practice to indent each additional layer of tags to make your HTML easier to read.

To make that a separate file, we just take the contents in between the style tags and put it

in a CSS file (by convention, usually named `style.css`).

Finally, we include an import at the top of our HTML file:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel
    </head>
    <body>
        <h1>Hello world</h1>
        <p>Hello <b>world</b></p>
    </body>
</html>
```

A small aside: separate vs combined files

Having a separate CSS file is generally preferred for several reasons:

- Smaller and simpler files (HTML and CSS separate) are easier to work with than large and monolithic files (HTML + CSS combined).
- Separate CSS files promote the reuse of your styles across your website. This has a number of advantages, including making it much easier to maintain your CSS changes. One change is automatically propagated to all websites that include the file, versus having to make the change on every page.
- Experimentation is easier, since you can hang on to many different CSS files and change which one you want to include.
- Many people have written prebuilt CSS templates (for example, Bootstrap) which you can start with. On top of that, you can write your own CSS rules to override anything that you don't like about the template and include it on top of the template. This is the "cascading" part of CSS.

Rules and selectors

CSS is comprised of a list of **rule sets**. Each rule set is a set of curly braces `{}`. Associated with each rule set is one or more **selectors**. The selector tell the browser which tags the rule should be applied onto. Within the braces, we have **colon-separated property and value pairs**, which are the rules. These are written in plain English and their function is usually self-explanatory.

The simplest selector is a tag type (e.g. `body`, `p`, etc.). Anything associated with this tag or *nested within* this tag will have the rules applied. Let's take a look at this example:

```
p {
  color: green;
}

body {
  color: red;
  font-size: 64pt;
}
```

We have two different rule sets: one for paragraphs (`p`) and one for everything in the body of the webpage (`body`).

What if we have a paragraph in the body? Which rule set gets applied? It turns out that *both* are applied. You can think of it this way: the web browser applies the rules for body to everything under the body tag in the webpage. It then searches for more *specific* selectors (more details here) and then applies those. This is also part of the "cascade" feature of CSS!

Classes and IDs

Sometimes, you don't want all paragraphs to be styled the same way. In this case, you would use classes or IDs. They both refer to a similar idea: instead of having a selector be any old tag, you can create your own names. This is known as a class, and it looks like this in HTML:

```
<h1 class="normal"> Hello world </h1>
<h1 class="crazy"> Hello world </h1>
```

When we specify a selector for this class, we prefix it with `.` in our CSS ruleset.

```
.normal {
```

```
  color: blue;
}

.crazy {
  color: red;
  font-size: 100pt;
}
```

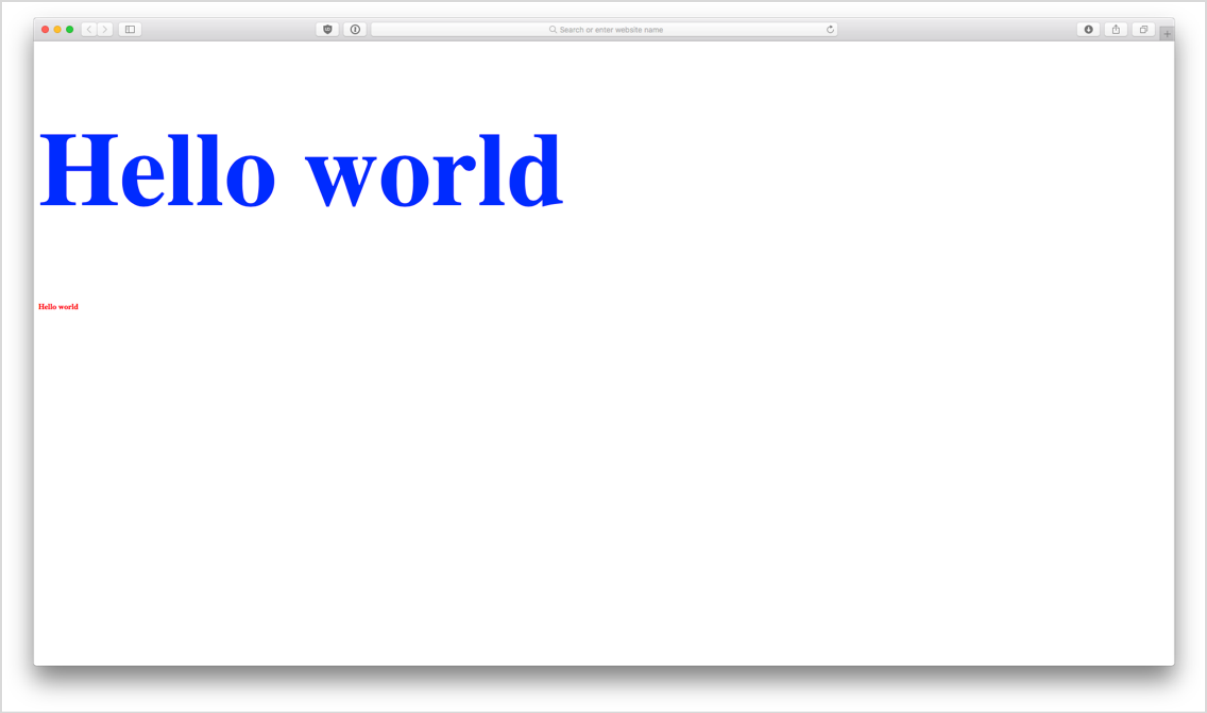
Class rules are more specific than the original tag (i.e. `h1`).

IDs are similar to classes, but they can only be used once. The selector is also prefixed with `#`. A tag can have both a class and an ID.

```
<h1 class="normal"> Hello world </h1>
<h1 class="crazy" id="tiny"> Hello world </h1>
```

```
#tiny {
  font-size: 10pt;
}
```

ID rules are more specific than class rules. For the example above, the "crazy tiny" header would obtain the red color from its class and font-size from its ID.



Result of our ID and class combo

Div and span tags

Unlike tags such as `<img>` or `<h1>`, the div and span tags do little to affect the layout of your page without including CSS. They are used primarily as an organizational tool for segmenting parts of your webpage.

Spans can be used inline in text, while divs usually contain multiple elements. Classes and IDs can also be applied to these tags. Here's an example of how they might be used.

```
<body>
    <div id="first">
        <h1> Hello world </h1>
        <h1> First div </h1>
    </div>
    <div id="second">
        <h1> Second div </h1>
        <p>Hello <span>world</span></p>
    </div>
</body>
```

```
#first {
    background-color: orange;
}

#second {
    background-color: yellow;
}

span {
    font-size: 200pt;
}
```

The resulting webpage looks like this:

Beautiful.

Resources

- CSS tutorial from w3schools
- Mozilla CSS reference