week::three
Introduction to Cascading Style Sheets

Introduction to Cascading Style Sheets

What are they?

- CSS is the **presentation language** used in the design of a website.
- They are a **collection of rules** that define the appearance of elements in a Web page.
- Just like paragraph styles in the print world, CSS helps **maintain consistency** within and across Web pages.
- Sites can reference one (or more) common `.css` files that contain all styles for that site.
  - Make one change, and it alters the appearance of every page in the site.

What's cool about CSS

- Allow you to keep a pages' **structure** *separate* from its **appearance**.
- Excellent control over the **presentation** of type and layout.
- Help maintain **visual consistency** across pages in a website.
- Provide a number of **layout controls**, including margins and padding space, borders, background images, and others.
- You can create styles for **specific mediums**: One for Web, one for mobile, one for print, etc.
- When using a visual HTML editor like WordPress or Dreamweaver, style sheets can be applied similar to paragraph styles in InDesign.

Some problems with using CSS

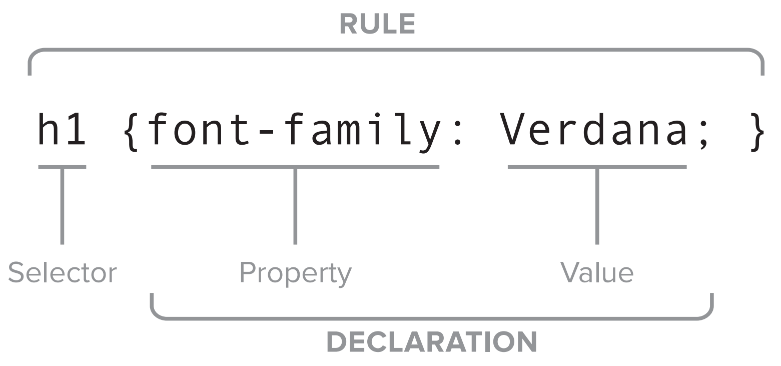
- Can be difficult to remember all options available to the designer.
- Simple layout is hard; Complex layouts are even more difficult.
- You often need to write CSS for specific device types (desktop, mobile, tablet, etc).
- Style sheets can be difficult to learn, design with, and debug across Web browsers.
- Some advanced options are not equally supported in all browsers.
  - Each browser's rendering engine has its own way of interpreting CSS rules.

Rule Structure

Rules — Selector & Declaration

Selector — Controls *what page elements* are altered by the declaration

Declaration — Property & Value pairs that declare *what* (property) will change and *how* (value).



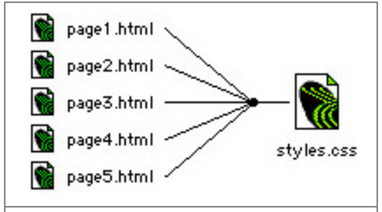
Where CSS Rules Are Stored: Embedded & External Style Sheets

Overview

- There are three main ways to store CSS rules: **Embedded**, **External**, and **Inline**.

External Style Sheets

- All pages within a site **reference a common, centralized CSS file** for style information.
- When a browser reads an HTML document with an external style sheet, the style sheet code is included in that page when displayed.
- Allow you to alter a single file to make appearance changes throughout a site.
- Best for websites with more than a single page.
- File extension is `.css`



Visual representation of HTML files linking to a common external style sheet.

Embedded Style Sheets

- Controls the appearance of **only the current page**.
- Style sheet is embedded in the `<head>` container of the HTML page.
- Site-wide type appearance changes are not possible when using this type.
- Good to use when a style rule is only used on the current page.

Inline Styles

- Style sheet information is written directly to a tag that precedes the type to be formatted, either to the enclosing element, or within a `<span>` tag.
- Allows for the overriding of Embedded or External style sheets.
- Makes it **very difficult** to make page or site-wide formatting changes.
- **Do not use inline styles unless absolutely forced to by the circumstances of your design/coding environment.**
  - When using some content management systems, for example.

Code Examples:

Link to an External Style Sheet – HTML version

```
<head>
  <link rel="stylesheet" href="styles.css" type="text/css" media="all" />
</head>
```

Link to an External Style Sheet — CSS version

```
<style type="text/css">
  @import url("reset.css");
  @import url("styles.css");
</style>
```

Embedded — CSS is applied directly in the HTML (not preferred)

```
<head>
  <style type="text/css">
    h2        { font-family: Verdana, Arial; font-size: 1.6em; font-weight: bold; }
    #bodytext { font-family: Verdana, Arial; font-size: .8em; line-height: 15px; }
  </style>
</head>
```

Tag Selectors

Overview

- Allows you to change the appearance of standard HTML tags.
  - Tags like: `<h1> <h2> <h3> <p> <body> <li> <td>`
- Tag selectors are **the first choice** among the different selector types, since paragraph and header tags give a document its structure and hierarchy. *Style with these first.*
- Change to tags are global across the entire page—changing the appearance of the `<p>` tag will change all paragraphs in your Web page.
- **Multiple Tags**—You can control multiple tags at once by separating them with a *comma*.

Code

```
p {
   color: red;
   font-size: 1.5em;
}

h1, h3, p {
   font-size: 2.275em;
   line-height: 1.5em;
}
```

Descendant Selectors

Overview

- Descendant selectors are the **most powerful type** of CSS rule, and the kind you will **use most often** when creating complete websites.
- They allow you to target and change specific areas of your page.
- It does this by using the document structure and hierarchy to create context.
  - For example, you can write a compound selector to say this: "Color the text red in a `<a>` tag that is inside of a `<ul><li>` list that is inside the `<nav>` tag with a class of 'primary'."

Code

- Example HTML:

```
<header>
   <h2>Some Headline</h2>
   <p>Some paragraph text with a <strong>strong element</strong>.</p>
</header>
<footer>
   <p>This is a footer <a href="#">with a link</a>.
   <p>This is also some text with a <strong>strong element</strong>.</p>
</footer>
```

- For example, if you wanted to change the appearance of the `<h2>` tag in ONLY the `<header>` section of your page, you would create the following selector:

```
header h2 { … }
```

- To change the appearance of hyperlinks (`<a>`) within paragraphs (`<p>`) only in the `<footer>` area:

```
footer p a { … }
```

- To change the appearance of the `<strong>` tag in paragraphs (`<p>`) the `<header>` only:

```
header p strong { … }
```

Classes

Overview

- Allow you to *selectively* apply formatting to an HTML page element.
- Classes can be **applied to multiple elements** within a page, making is useful to style multiple objects in a similar manner.
- Classes in CSS are identified by a single period (.) in front of the name.

When to Use

- To target specific areas of a page and style the HTML elements with (or within) that class with styles different than the rest of the page.
- Use when the hierarchy of HTML elements is not specific enough to style with CSS.
- Classes should be used before IDs—creating re-usable styles is always better than writing one-off code.

Code

```
<style>
   div.photogallery { border: 1px solid gray; }
   div.photogallery p { font-size: .9em; }
</style>

<div class="photogallery">
   <p>Some content goes here</p>
</div>
<div class="photogallery">
   <p>Yet some more content goes here</p>
</div>
<p>Some other content is here. It also looks different than the text above.</p>
```

ID Styles

ID Styles

- IDs should be applied to **only one element** within a page.
- They provide access to all of the same options as Classes.
- ID styles are identified by the pound sign (#) in front of the name.

When to Use

- When an element is the only one of its kind on a page and will always be the only one.
- Can be targeted in a URL by adding `#idname` to the end of a link to a page.
- In most circumstances, you will use these the *least* of the four selectors on this handout.

Code

```
<style>
   a#home { font-weight: bold; }
</style>

<nav>
  <ul>
      <li><a href="index.html" id="home">Home</a></li>
      <li><a href="about.html" id="about">About Us</a></li>
  </ul>
</nav>
```