HTML & CSS Level 1: Week 2
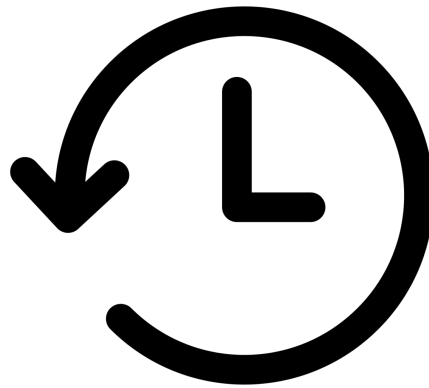
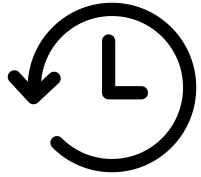May 12, 2015

Instructor: Devon Persing

Tonight

- Week 1 review and questions

- Restaurant website reviews

- Introduction to CSS

- The style cascade

- Prepping images for the web
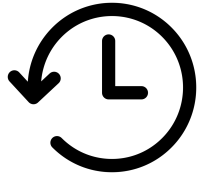
- Validating HTML and CSS

Review!

Review: Delivering content

- **HTML** structures content

- **CSS** creates style and layout

- **JavaScript** adds interactivity

Review: HTML documents

- `<!DOCTYPE html>` tells the browser it's serving an HTML file
- `<html>` tags wrap the whole document
- `<head>` tags wrap all of the metadata
- `<body>` tags wrap all of the content
- Most HTML elements have **opening and closing tags**, and some have **attributes**
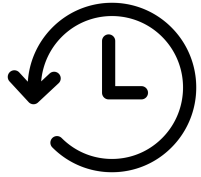
Review: HTML for content

- **Headings** create an outline: `<h1>...<h6>`

- **Paragraphs** and **lists** structure text: `<p>...<ul>...<ol>`

- **Images** embed jpegs, gifs, etc.: `<img src="" alt="">`

- **Links** connect pages: `<a href="">`
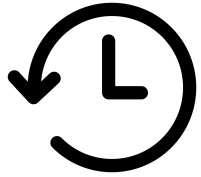
Review: Tidy file structures

- **HTML** files go in the main directory

- **CSS** and media go in subdirectories

- Your homepage is **index.html**

- Paths can be **absolute** or **relative**

- Use relative paths when possible (`img/kittens.png`, **not** `http://example.com/img/kittens.png`)
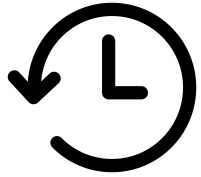
Review: Tidy file names

- **No spaces** in filenames

- **Capitalization** matters

- Use **letters**, **numbers**, **hyphens** and **underscores** only

- Filenames **start with a letter**
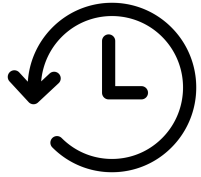
Review: **Good coding practices**

- **Standardize** your file structures and filenames

- Use *<!-- HTML comments -->* to leave yourself and others notes

- **Indent your code** so its readable

Questions?

?

Intro to CSS

{ }

# { } **Getting ready for our CSS**

1. Open `index.html` in your text editor.

2. After the `<title>...</title>` element, make a new line.

3. Type: `<style></style>`

4. Save your `index.html` file.

5. For now, we'll put all of our style rules inside `<style></style>`.

{ } **Cascading Stylesheets**

- **CSS** brings style, formatting, and layout to HTML

- Provides consistent and scalable ways to style **single elements**, **single pages**, or **entire sites**

- **Separates look and feel from content** so that sites can be restyled more easily
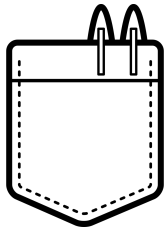
CSS overload alert

- There are **a lot** of things you can do with CSS

- We will **not get anywhere close** to covering all of them

- **Practice the basics** before getting fancy

- We'll cover **common CSS** for text styles, colors, positioning, layout, and some fun extras

{ } Anatomy of a CSS rule

`selector { property: value; }`

- **Selector** is the thing you want to style
- **Property** is what aspect you want to style
- **Value** is how you want to style it
- **Property** + **value** = **declaration**
- **Declarations** end in semicolons (;)

{ } **Example CSS rule**

```
h1 { font-size: 30px; }
```

- **Selector** is `h1` (yes, an `<h1>`!)
- **Property** is `font-size`
- **Value** is `30px` (30 pixels high)

{ } CSS comments

- Just like HTML, CSS can have **comments**

```
<style>

  /* I am a CSS comment! */


  h1 { /* I am also a CSS comment */

    color: #ff0000;

  }

</style>
```

{ } **Common font properties**

- **`font-size`**: a number followed by a measurement of how tall the element's text is, usually in ems (**`em`**) or pixels (**`px`**)

- **`font-family`**: the name of a typeface, or typefaces

- **`font-style`**: **`italic`**

- **`font-weight`**: **`bold`** | values of bold!

- **`line-height`**: a number followed by a measurement of how tall the element's line of is, usually in ems (**`em`**) or pixels (**`px`**)

{ } **Common text properties**

- **text-align: left | right | center | justify**

- **text-transform: capitalize | uppercase | lowercase** | some others

- **text-decoration: underline | overline | line-through** | some others

- **Note:** A lot of properties will take a value of **none** by default, or for overriding other styles

{ } **Colors**

- To set **text colors**, the property is `color`

- To set **background colors**, the property is `background-color`

- The value can be done a few ways:

  - Hex: `#ff0000`

  - RGB: `rgb(255,0,0)`

  - Also possible but not preferred: `red`

{ } **Multiple selectors & properties**

```css
/* style bullet AND numbered lists */
ul, ol {
    font-size: 1.2em;
    font-weight: bold;
    color: #39887c;
}
```

{ } The Web Inspector

Q: Want to see what's going on under the hood?

A: Of course you do.
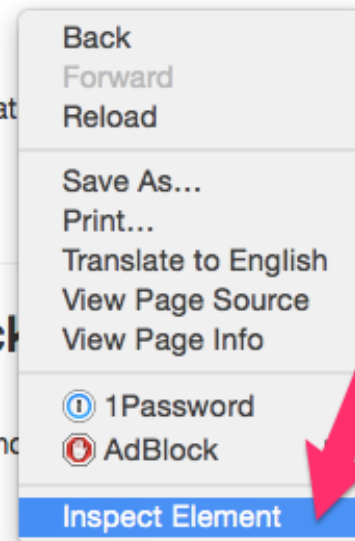
{ } Using styles in multiple places

- **Inline styles** are applied to only a single element (we'll talk about a better way to do this next week!)

- **Internal styles** are added in the head of a page and style only that page (what we've done so far)

- **External styles** are called into multiple pages, to style a whole site

{ } Making an external stylesheet

1. Create a **new file** in your text editor.

2. **Copy and paste** our styles from inside the `<style>...</style>` element into our new file.

3. Save our new file as `styles.css`, and put it in your `css` folder.

4. Save your `index.html` file.

5. Then...

{ } Linking to external CSS

```
<link href="css/styles.css"
rel="stylesheet">
```

- Tells an HTML page to go find and load the CSS file

- Goes inside the `<head>` element

- Needs to go in every page that should load the stylesheet

{ } The "cascading" part

The beauty of CSS is being able to create styles and then override them when you want to customize the look of your pages.

There are three big rules for determining how styles get applied:

- Styles are loaded from far to near.
- Styles are loaded top to bottom.
- Children are more specific than parents.

{ } Stylesheet "location"

- Styles that are "closer" to the elements they style take precedence
    - Browser defaults
    - External styles (in a `.css` file)
    - Internal styles (in the `<head>`)
    - Inline styles (in an element)

Least specific

Most specific
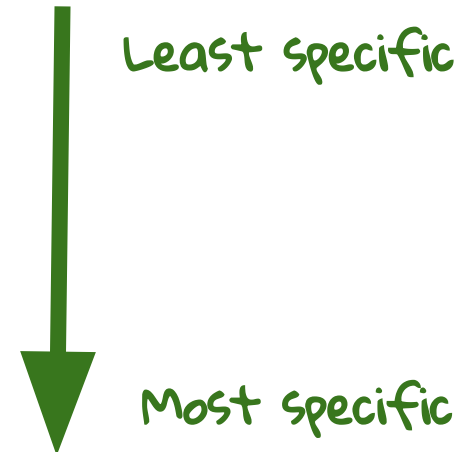
{ } **Top to bottom**

- If the same property is styled multiple times for the same selector, **the last one sticks**

```
p { color: #2f4251; }
p { color: #daa645; } /* overrides first! */
```

{ } **Children are specific**

- Children elements usually **inherit** styles from their parents but can **override** parents with their own styles

```
body { color: #60dd47; } /* parent of all
visible content */


p { color: #2f4251; } /* child of body */
```

{ } **Selectors can be more specific**

- If one style is **more specific** than another, it takes precedence

```
p { color: #2f4251; } /* paragraphs in general */

a { color: #e7c0c8; } /* links in general */

p a { color: #c4fe46; } /* links that are inside
paragraphs only */
```

{ } **Unordered list item styles**

- You can change things like:
  - The bullet type
  - How the bullet is spaced with the text

```
ul li {
    list-style-type: square;
}

ul li {
    list-style-image: url("an-image.png");
}
```

# **{ } Ordered list item styles**

- You can change positioning and even the type of numbering/lettering used

```
ol li {
    list-style-type: upper-roman;
    list-style-position: inside;
}
```
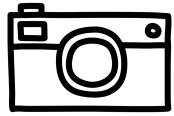
{ } "How will I remember all this?"

- **You won't.** I don't!

- Use online references like these:
    - Mozilla's [Getting Started with CSS](#) guide
    - CSS-Tricks' [CSS Properties Almanac](#)

- Use your Web Inspector to see load order for rules, and what's ignored
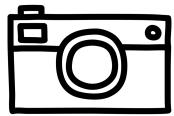
Images for the web
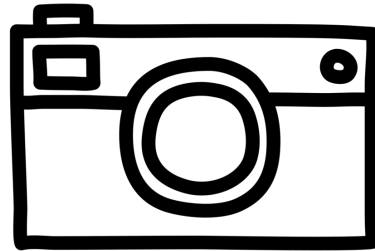
📷 Why "web-ready" images?

- Strips out **layers** and other **metadata** from your image

- Minimizes file sizes to **help load time** in the browser

- Optimized images for **RGB display** at the correct **resolution** for browsers
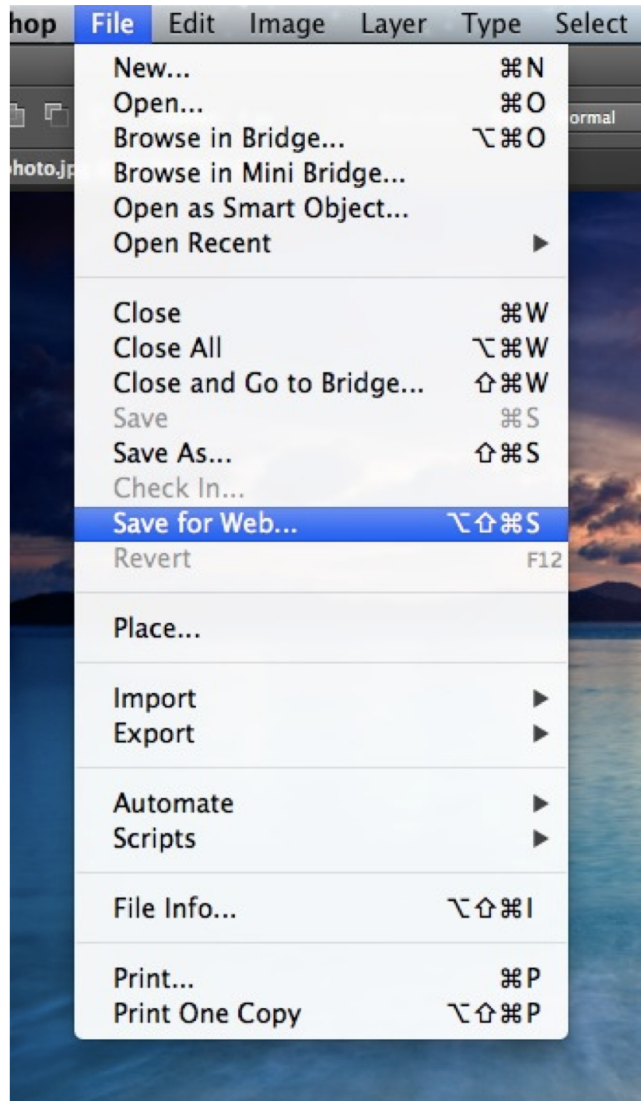
📷 **Web image types**

- **JPG** or **JPEG** is traditional for photos

- **GIF** is traditional for animation, illustrations, and transparency

- **PNG** was designed specifically for the web for any image type (including transparent images and animations)
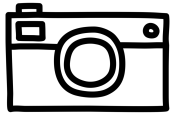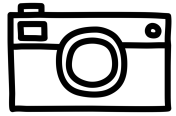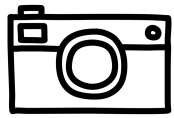
"Save for Web" in Adobe CS

- Adobe products have a **"Save for Web..."** or **"Save for Web and Devices..."** option
- Similar products will have a similar option

📷 Saving for web

1. Pick "Save for Web..." from the File menu

2. Pick a suitable file format (JPEG, GIF, etc.)

3. Adjust the image dimensions to be the max size you want to display

4. Save!

5. Add your image to your page with an `<img src="" alt="">` element

📷 Exceptions!

- If the **photo is the content**:

  - Usually we'll use a smaller image as a link, and load a larger resolution image on click

- Newer devices with **retina screens** may support larger resolution images

  - We can use Javascript or server-side code to display different images on different devices
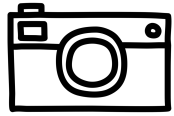
📷 **YouTube demos for images**

- [Saving for web from Photoshop](#)

- [Saving for web from Illustrator](#)

- Look for these on the class site!

Validating your code

Validating HTML and CSS

- **Validation** is an easy way to make sure your code is properly formatted to find and prevent errors

- HTML: html5.validator.nu

- CSS: jigsaw.w3.org/css-validator

"Homework" for next week

1. Style your fonts and backgrounds
2. Style your pages with an **external stylesheet**
3. **Validate** your HTML and CSS
4. *Optional***:** Create a web-friendly header image/logo for your site
5. *Optional reading: HTML and CSS* ch. 10-12

Next time

- Questions and review from Week 2

- HTML tables

- Web fonts

- Inline, block, and inline-block elements

- CSS box model (aka intro to layouts)

Questions? Comments?

- Visit dpersing.github.io/svc for:
  - Class slides
  - Code samples
  - Resources
- Email me: dep@dpersing.com
- Tweet at me: @devonpersing