HTML & CSS: LEVEL 1

Instructor: Beck Johnson

Week 2 - March 2 2017

SESSION OVERVIEW

- Week 1 review and questions
- Image format overview
- Optimizing web images
- Introduction to CSS

REVIEW!

REVIEW: WEBPAGE COMPONENTS

- **HTML** structures and organizes content
- **CSS** stylizes the content and creates layout
- **Javascript** adds interactivity

REVIEW: HTML DOCUMENTS

- `<!DOCTYPE html>` tells the browser it's serving an HTML file using HTML5 standards
- `<html>` wraps the whole document
- `<head>` wraps the metadata
- `<body>` wraps the visible content
- Most HTML elements have **opening** and **closing tags**, and some have **attributes**

REVIEW: HTML CONTENT

- **Headings** create an header/outline

  `<h1>…<h6>`

- **Paragraphs** and **lists** structure text

  `<p>`
  `<ul>`
  `<ol>`

- **Images** and **links** both require **attributes** to work

IMAGES

```
<img src="kitten.jpg" alt="picture of a kitten" />
```

- Does not have a closing tag
- Two required **attributes**:
  - **src** is where the file lives (local or external)
  - **alt** is a description of the image (used for screen readers, search engines, etc)

LINKS

```
<a href="http://google.com">Google</a>
```

- Creates a link to other pages or websites
- The `href` attribute says where the link should go
- Anything inside `<a>` tags is clickable

QUESTIONS?

WEB IMAGES

WEB-READY IMAGES

- Minimize file sizes to help load times in browser
- **Optimizes images for RGB displays** with correct **resolution** for browsers
- **Flattens** layers and removes metadata from graphics

WEB IMAGE TYPES

JPG/JPEG

- millions of colors
- uses a compression algorithm called **lossy**

GIF

- 256 colors - fewer colors mean a smaller file
- animation and off-on transparency

PNG

- millions of colors
- full alpha transparency

JPGS

JPG pros:

- small file size
- rich colors

JPG cons:

- image distortion
- no transparency

GIFS

GIF pros:

- small file size
- transparency
- animations

GIF cons:

- few colors

PNGS

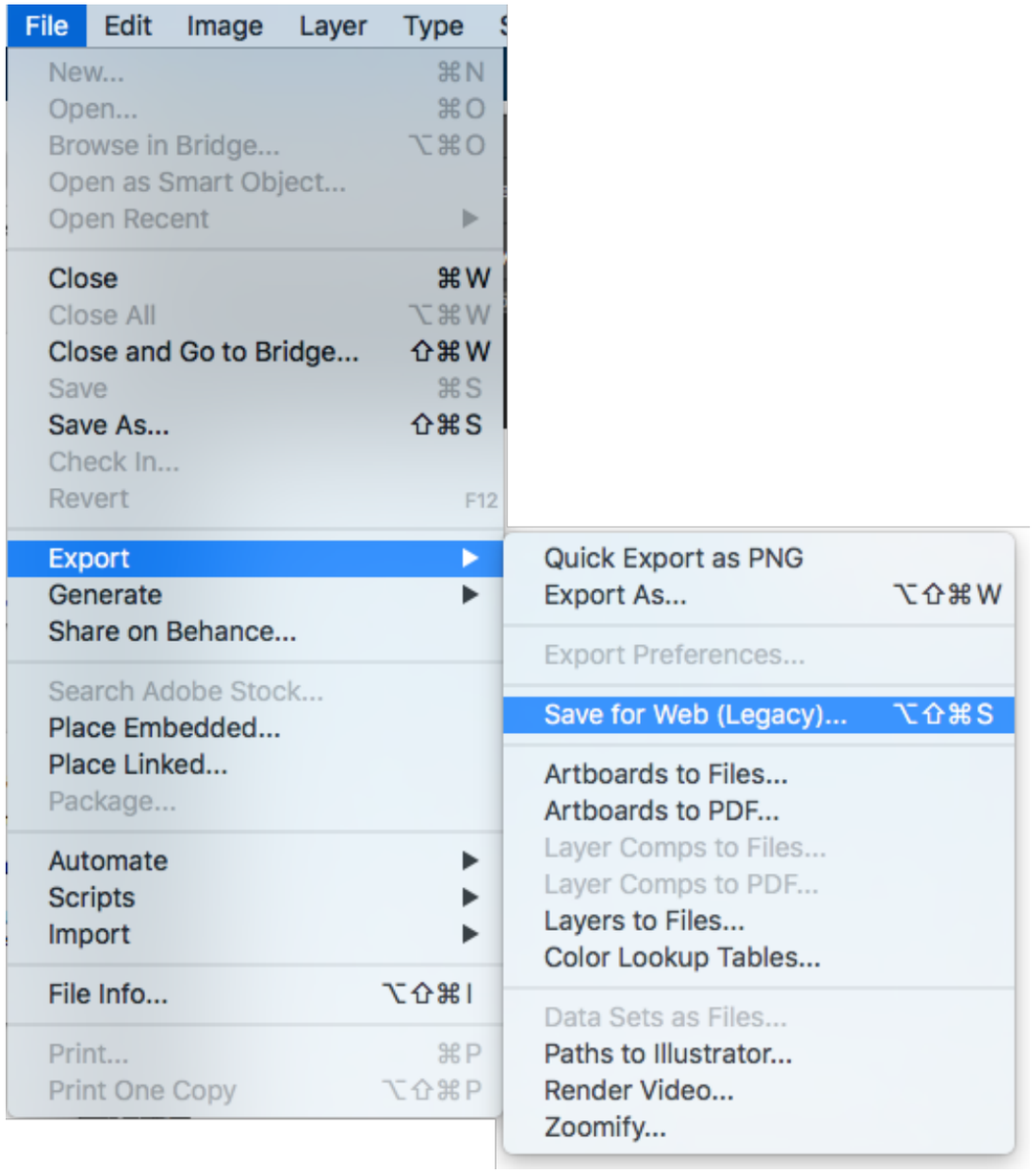
PNG pros:

- any amount of transparency
- best image quality

PNG cons:

- large file size
- IE 7&8 don't support transparency

“SAVE FOR WEB” IN ADOBE CS



Adobe products have a “**Save for Web...**” or “**Save for Web and Devices...**” option

“SAVE FOR WEB” IN ADOBE CS

- Click **File > Export > Save for Web… (**or **Export As)**
- Choose a format **(JPEG, PNG 24, or GIF)**
- Adjust image size to max size display
- Save to your images directory.

SOME “GOTCHAS”

- Best practice to work in 72 PPI in graphic editor programs. (keeps file sizes down)
- Always work in **RGB** when working with graphics for the web. **CMYK** is for print
- Graphics for **Retina devices** need to be saved out at 2X their “normal” size

CSS

CASCADING STYLE SHEETS

- Language for specifying how documents are presented to users
- We can override the browser's default presentation styles with our own.
- Provides consistent and scalable ways to style single elements, single pages, or entire websites
- Separates look and feel from content/markup

CASCADING STYLE SHEETS: FAIR WARNING

- There is **A LOT** you can do with CSS.
- We won’t get anywhere close to covering everything!
- We'll practice the basics before getting into advanced topics
- We will cover basic CSS for text styles, colors, positioning, layout, and a couple of extras

WHY USE CSS?

- Helps you avoid duplication by keeping styles in one place (one external stylesheet)
- Makes style maintenance easier
- Allows you to make a site-wide change in one place for example, update the font for the whole site in one line of code!
- Separating presentation from content enforces style consistency and allows flexibility

ANATOMY OF A CSS RULE

```
selector { property: value; }
```

- **Selector** is the **thing** you want to style
- **Property** is the **attribute** you want to style
- **Value** is how you want to style it
- **Values** always end in semicolons (;)

ANATOMY OF A CSS RULE

So!

```
p { color: blue; }
```

"All paragraphs will have blue text "

EXAMPLE CSS RULE

```
p { font-size: 14px; }
```

- **Selector** is **p** (<p> in the HTML)
- **Property** is **font-size**
- **Value** is **14px** ("em height" of 14 pixels high)

CSS COMMENTS

```
<style>
  /* I am a CSS comment! */

h1 { /* I am also a CSS comment */
   color: #ff0000;
   }
</style>
```

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

{} COMMON FONT PROPERTIES

- **`font-size:`** a number followed by a measurement of the height of that element's text in ems (em) or pixels (px)

- **`font-family:`** the name of a typeface installed on the user's computer

- **`font-style:`** normal by default - can also be italic or oblique

- **`font-weight:`** normal by default - can also be **bold**, or values of 100, 200, etc depending on the typeface

- **`line-height:`** a number followed by a measurement of the height of a line of that element, in ems (em) or pixels (px)
  - similar to leading in typography

{} QUICK ASIDE ABOUT UNITS

The two standard units for sizing in CSS are **px** and **em**

- **px** isn't a constant length, and isn't related to font height or a physical unit of measurement (like inches)
  - The requirement is that a 1px line must display without anti-aliasing
  - So px varies by device, but always looks "about the same"

- **em** refers to the height of the letter 'm' of the font being used
  - This unit of measurement is a description of the **relative** size between this element and its parent
  - So `h2 { font-size: 2em }` means the header is 2 times as big as the letter 'm' of the default font in your html document

{} COLORS

- `color:` changes the color of text
- `background-color:` sets the background color of an element
- Color **value** can be set using **HEX**, **RGB**, or **RGBA**
  - Hex: `#ffffff`
  - RGB: `rgb(245, 245, 245)`
  - RGBA: `rgba(245, 245, 245, 0.8)`

```
p { color: #222222; }
h1 { background-color: rgba(0,0,0,0.5); }
```

{} WIDTH & HEIGHT

All elements have a height and width, which can be changed via CSS.

You can set width and height of images directly with HTML attributes:

```
<img src="example.jpg" width="300" height="200" />
```

But it's better to use CSS:

```
img { width: 300px; height: 200px; }
```

{} MULTIPLE SELECTORS & PROPERTIES

- You can add multiple **selectors** to a CSS rule
- You can add multiple **properties** to a CSS rule
- Example: style all ordered and unordered lists:

```
ul,
ol {
    font-size: 16px;
    font-weight: bold;
    color: #444444;
}
```

{} WEB INSPECTOR TIME

{} CSS IN MULTIPLE PLACES

- **Inline styles** are applied to only a single element

- **Internal styles** are added in the `<head>` of a page and style only that page

- **External stylesheets** are called into multiple pages, and are declared in separate .css files

{} EXTERNAL STYLESHEETS

- **Copy and paste** your styles from inside `<style></style>` in index.html into a new file.

- Save your new files as **styles.css,** and save it in your **css** directory/folder.

- **Remove** the `<style></style>` tags from index.html

{} LINKING TO EXTERNAL STYLESHEET

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

- Tells the browser to find and load the styles.css file from the css directory
- The `rel` attribute stands for "relation" - in this case, this link's relationship to the document is "stylesheet"
- This tag goes inside the **`<head>`** element
- Should be on every page that needs the styles

{} 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 **3 rules** for determining how styles get applied:

- Styles are applied from far to near
- Styles are applied from top to bottom
- Children elements are more specific than parents

{} STYLES "LOCATION"

Styles that are "closer" to the elements they style take precedence.

- Browser default
- External styles (in a **.css** file)
- Internal styles (in the **<head>**)
- Inline styles (directly on an element)

Less Specific

Most Specific

{} TOP TO BOTTOM

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

```
p { color: #2f4251; }

p { color: #daa645; } /* this one wins */
```

{} CHILDREN ARE SPECIFIC

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

```
body { color: #2f4251; } /* parent */

p { color: #daa645; } /* child */
```

All text in the body that is NOT a paragraph will be dark gray. Paragraphs will be mustard-colored

{} SELECTORS CAN BE MORE SPECIFIC

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

```
p { color: #daa645; } /* all paragraphs */

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

p a { color: #c4fe46; } /* links in paragraphs */
```

{} WEB INSPECTOR (AGAIN!)

PRACTICE TIME!

HOMEWORK

- Add a stylesheet to the website you created last week
- Decide how you want it to look, and start making those changes using CSS
    - Consider changing font color, size, link color, and background
- Using what we learned about image formats, decide if the images you included last week are the best choice for how you're using them
- Re-save images in an optimized format using Photoshop

“HOMEWORK”

- Practice!

- Optional: read chapters 10-12 and chapter 16 of HTML and CSS: Design and Build Websites

- Check out the CSS Zen Garden for inspiration on how simply changing CSS can change the entire look and feel of a page