

HTML



CSS



HTML & CSS: LEVEL 1

Instructor: Sean Thompson

Week 2 -October 6, 2016

seanmarshallthompson@gmail.com



SESSION OVERVIEW

- Week 1 Review and questions
- Finish Week 1 - Images, Links, Organization, File Paths
- Web graphics overview
- Optimizing graphics and image formats
- Block vs. Inline Elements
- More HTML elements
- Introduction to CSS and styles



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
- `<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 CONTENT

- **Headings** create an header/outline: `<h1>... <h6>`
- **Paragraphs and lists** structure text: `<p>......`

QUESTIONS?

IMAGES

```

```

- Images `` do not have a closing tag.
- Images have 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>
```

- Defines an anchor or link that creates a path to other pages or websites.
- Have a **required href** attribute that says where the link should send the user.
- Anything inside **<a>** tags are clickable.
- Can be text, an image, or any valid HTML.
- **target="_blank"** -> opens link in a new tab.

FILE ORGANIZATION

- HTML files (.html)
- CSS files (.css)
- Javascript files (.js)
- Image files (.png, .jpg, .gif, etc)

FILE NAMING RULES

- NO spaces in filenames
- Capitalization matters (kittens.png is not the same as Kittens.png)
- Use only letters, numbers, hyphens (-) or underscores (_)
- Always start with a letter for file names.
- Your homepage is always **index.html**
- Recommended to use lowercase and hyphens (**about-us.html**)

FILE STRUCTURE

- On the web, folders are called **directories**. This is what dictates paths to resources such as html files, images, and CSS and Javascript files.
- HTML should usually go in the **main (root) directory**.
- Make **subdirectories** for media, CSS, and Javascript files.

FILE PATHS

- On the web, all resources have a Universal Resource Locator (**URL**)
- **Absolute paths** are URLs that always goes to the same place (e.g.
- <http://google.com>
- <https://www.svcseattle.com/assets/images/logo.gif>
- If someone renames or deletes the file, your link will be broken.
- Helpful read: <http://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/>

FILE PATHS (RELATIVE)

- Relative paths are URLs that go to a resource in relation to what page you're on. (e.g **css/layout.css**)
- Resources “local” to you should all be relative paths. (your images, fonts, CSS and JS files)

`` image is in the same folder as this file

`` image is in a subfolder named “images”

`` image is in a superior folder named “images”



WEB GRAPHICS



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.

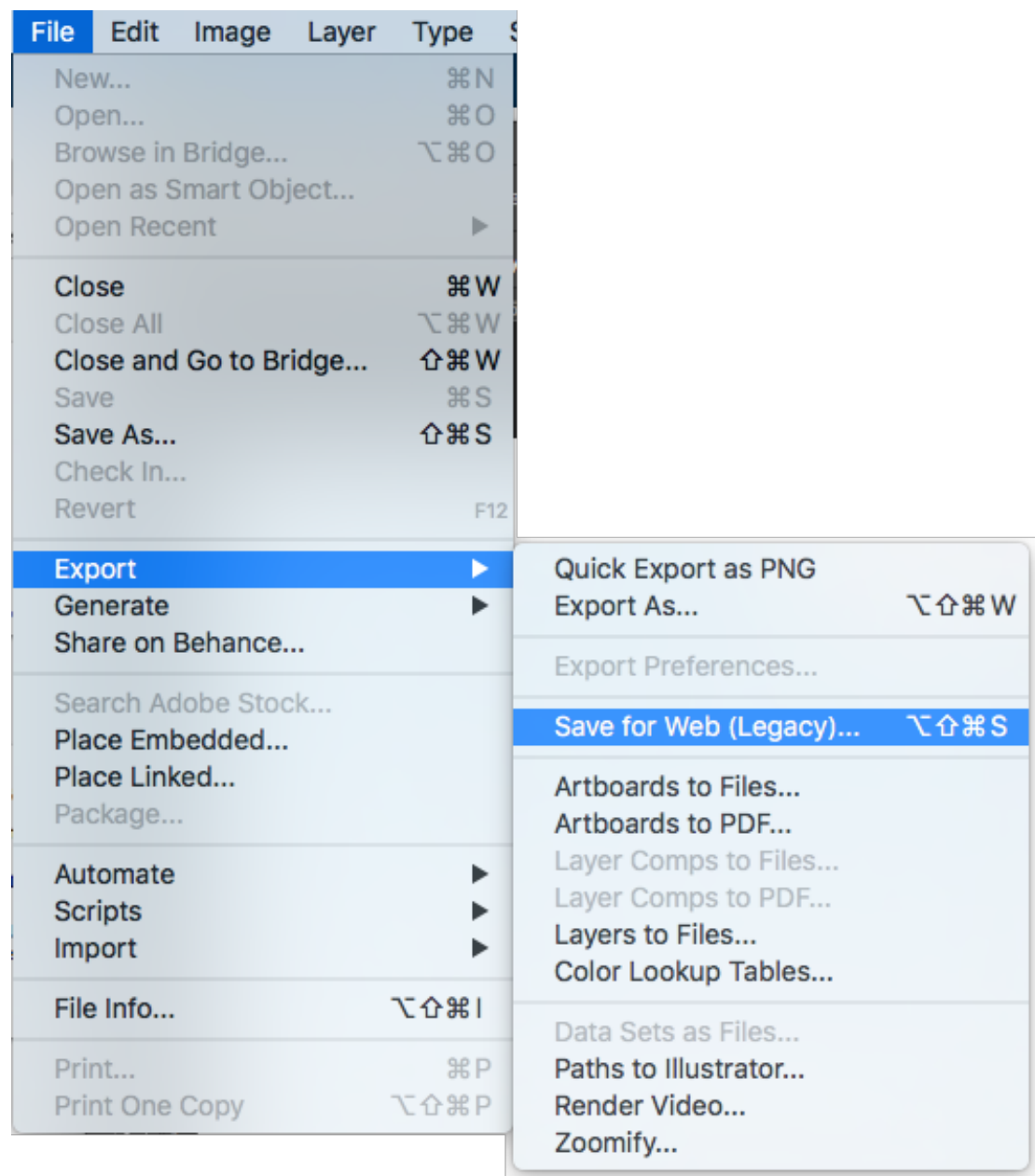


WEB IMAGE TYPES

- **JPG or JPEG (Joint Photographic Experts Group)**
is traditional for photos. Millions of colors, no transparency, no animation.
- **GIF (Graphics Interchange Format)** – 256 colors, can be animated, has transparency.
- **PNG 24 (Portable Network Graphic)** – Millions of colors, full alpha transparency, no animation.



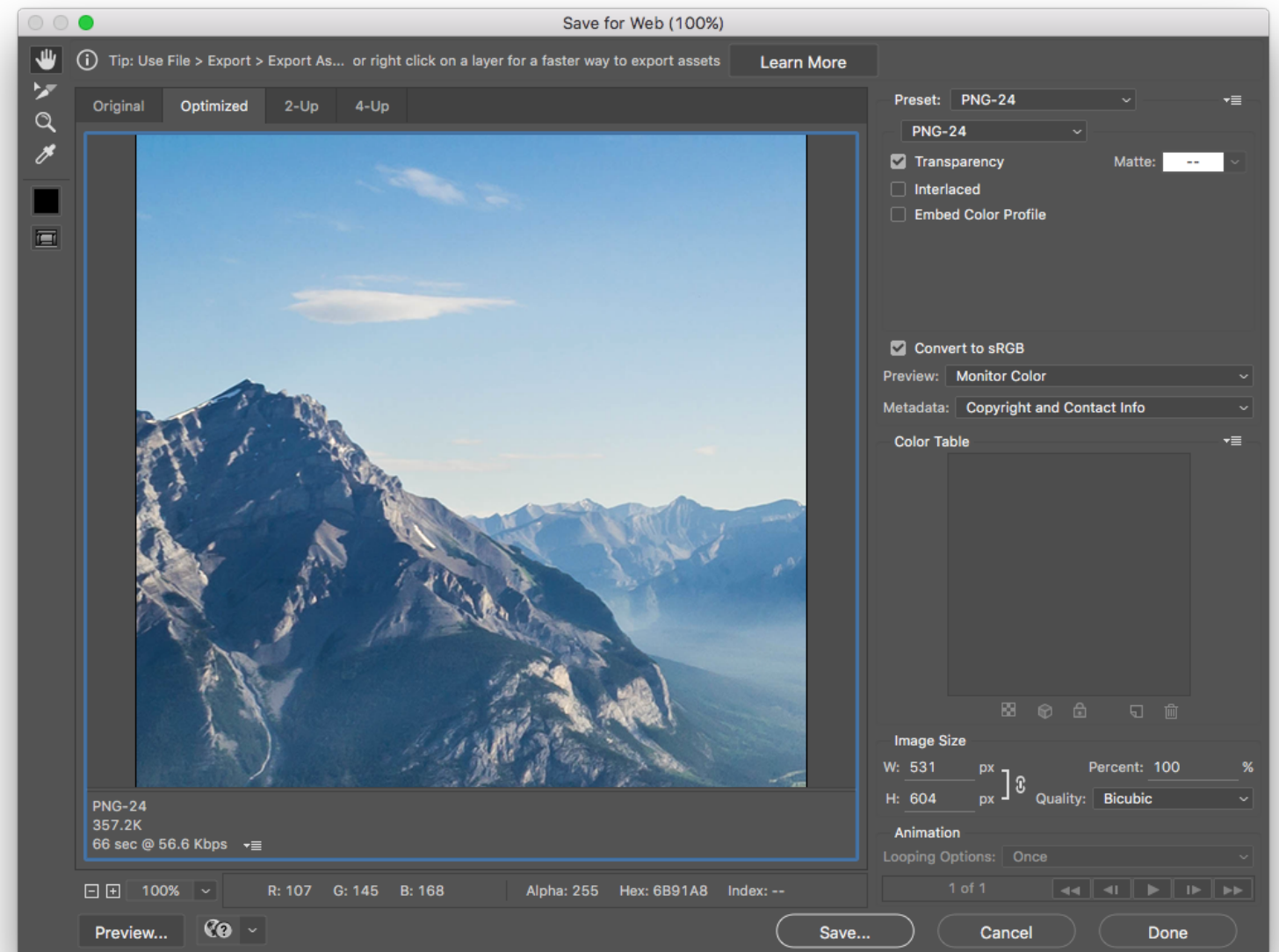
“SAVE FOR WEB” IN ADOBE CS



- Adobe products have a **“Save for Web...”** or **“Save for Web and Devices...”** option.
- Similar products have this option as well.

“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** n working with graphics for the web. **CMYK** is for print.
- Make your designers stick to these rules!
- Graphics for **Retina devices** need to be saved out at 2X their “normal” size.



YOUTUBE DEMOS

- Saving for Web in Adobe Photoshop
- Saving for Web in Adobe Illustrator



BLOCK VS. INLINE ELEMENTS

<https://www.impressivewebs.com/difference-block-inline-css/>

<> BLOCK & INLINE ELEMENTS

Block-level elements

- If no width is set, will expand naturally to fill its parent container
- If no height is set, will expand naturally to fit its child elements (assuming they are not floated or positioned)
- Can have margins and/or padding
- By default, will be placed below previous elements in the markup (assuming no floats or positioning on surrounding elements)
- Ignores the vertical-align property

<> BLOCK & INLINE ELEMENTS

Block-level elements we've learned:

- Headings (<h1>, <h2>, etc)
- Paragraphs (<p>)
- Lists (,)
- List items ()

<> BLOCK & INLINE ELEMENTS

Block-level elements

BLOCK ELEMENTS EXPAND NATURALLY 



AND NATURALLY DROP BELOW OTHER ELEMENTS 



<> BLOCK & INLINE ELEMENTS

Inline elements

- Flows along with text content
- Will ignore top and bottom margin settings, but will apply left and right margins, and any padding
- Will ignore the width and height properties
- If floated left or right, will automatically become a block-level element, subject to all block characteristics
- Is subject to the vertical-align property

<> BLOCK & INLINE ELEMENTS

Inline elements we've learned:

- Links (<a>)
- Coming soon: ****, ****, ****

<> BLOCK & INLINE ELEMENTS

Inline elements

INLINE ELEMENTS FLOW WITH TEXT

PELLENTESSQUE HABITANT MORBI TRISTIQUE SENECTUS
ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS.
VESTIBULUM **INLINE ELEMENT** VITAE, ULTRICIES
EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT
AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI
VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

<> BLOCK & INLINE ELEMENTS

“Inline-block” elements

- Inline/Block hybrid.
- Take up width and height like block-level elements.
- Flow with content around them.
- Can have margin and padding.
- Elements we know: **** elements.

<html>

(MORE) HTML ELEMENTS

<DIV> ELEMENTS

<div></div>

- <div> elements are generic **block elements**.
- Used to create **sections or groups** in HTML for layout.
- Can be used wrappers for other elements (including other divs!) for creating complex layouts.
- Have height and width
- **Building blocks of HTML layouts!**

<DIV> LAYOUT EXAMPLE

```
<div id="header"></div>
```

```
<div id="nav"></div>
```

```
<div id="section"></div>
```

```
<div id="footer"></div>
```


 ELEMENTS

- elements are generic **inline elements**.
- Can **nest inside** other block or inline elements.
- Used to **style unique inline content** or **content inside block elements**.
- Flow with content around them.

<> MORE INLINE ELEMENTS

- **em** elements are used to show *emphasis*. (like italic).
- **strong** elements are used to show **importance in context** (like bold)

<p>"Oh, great. Someone ate my only clean socks." </p>

<p>"Was it the cat?"</p>

<p>"No, it was the dog."</p>

CSS



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 will practice the basics before getting into advanced topics.
- We will cover **common 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.
 - e.g. update the font for the whole site in one line of code!

ANATOMY OF A CSS RULE

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

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

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

EXAMPLE CSS RULE

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

- **Selector** is the **p**. (<p> in the HTML)
- **Property** is the **font-size**.
- **Value** is **30px** (30 pixels high).
- All paragraph tags will have a font size of 14px.

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 how tall the element's text is, usually in ems (**em**) or pixels (**px**).
- **font-family:** the name of a typeface.
- **font-style:** (**normal**, or **italic** are most common)
- **font-weight:** **bold** (can also be values of 100, 200, up to 900 depending on the typeface).
- **line-height:** a number followed by a measurement of how tall the element's line of is, usually in ems (em) or pixels (px) (similar to **leading** in typography)

🛠️ COLORS

- To set **text color**, the property is **color**.
- To set **background colors**, the property is **background-color**.
- Color **value** can be: **HEX**, **RGB**, or **RGBA**.
 - Hex: **#ffffff**
 - RGB: **rgb(245, 245, 245)**
 - RGBA: **rgba(245, 245, 245, 0.8)** – (0.8 represents alpha/opacity)

```
p {color: #222222;}
```

```
div {background-color: rgba(0,0,0,0.5);}
```

WIDTH & HEIGHT

- Block elements have width and height by default, which you can override.
- You can set width and height of images with HTML attributes:

```

```

- **But** it's recommended to use CSS:

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

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

{ 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;  
}
```

{ CSS IN MULTIPLE PLACES

- **Inline styles** are applied to only a single element (best practice to avoid this if possible).
- **Internal styles** are added in the **<head>** of a page and style only that page. (what we've done so far)
- **External stylesheets** are called into multiple pages, and are declared in separate **.css** files. *Best practice.

{ EXTERNAL STYLESHEETS

- Create a **new file** in your text editor.
- **Copy and paste** your styles from inside the `<style>...</style>` element your 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 go find and load the CSS file.
- Goes inside the **<head>** element.
- Should go in every page that should load 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 three big rules for determining how styles get applied:

- Styles are loaded from far to near.
- Styles are loaded 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 defaults
- External styles (in a **.css** file)
- Internal styles (in the **<head>**)
- Inline styles (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 sticks.**

```
p { color: #2f4251; }  
ul{ color: #444444; }  
/* some other stuff */  
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 */
```

{ 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; } /* a nested in p */  
div p a { color: #a5dd5e; } /* a in p in div */
```

{ WEB INSPECTOR (AGAIN!)



PRACTICE TIME!

ASSIGNMENT

- Create the first page of a car review website.
- Use at least 4 <div> tags to create a basic header, navigation, main section, and footer, and give them a background color.
- Place the included logo in the header div.
- Resize the image to 700px in Adobe CS and **Save for Web**.
- Place the image in the main (2nd) div.
- Create a nav of **list** elements in the navigation div.
- Use at least one <h1>, <h2>, <h3> tag.
- Use an and tag in your design.

“HOMEWORK”

- Practice!
- Read MDN’s Introduction to CSS
 - [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting started](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started)

WOW! THAT WAS A LOT!

- QUESTIONS?
- Email me at: seanmarshallthompson@gmail.com