HTML & CSS: FUNDAMENTALS OF DEVELOPMENT

Instructor: Beck Johnson

Week 2

SESSION OVERVIEW

- Week 1 review and questions
- Overview of CSS
- File organization
- Version control & code sharing with Git

REVIEW!

REVIEW: WEBPAGE COMPONENTS

HTML

Structures and organizes content

CSS

styles the markup and creates layout

JAVASCROIT

brings content and design to life

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 and styles
- `<body>` wraps the visible content
- Most HTML elements have **opening** and **closing tags**, and some have **attributes**

REVIEW: LAYOUT ELEMENTS

- `<header>` wraps header content
- `<footer>` wraps footer content
- `<nav>` indicates that everything inside is related to navigation
- `<section>` is used to define content sections

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="Cute kitten" />
```

- Does not have a closing tag ("self-closing")
- 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)

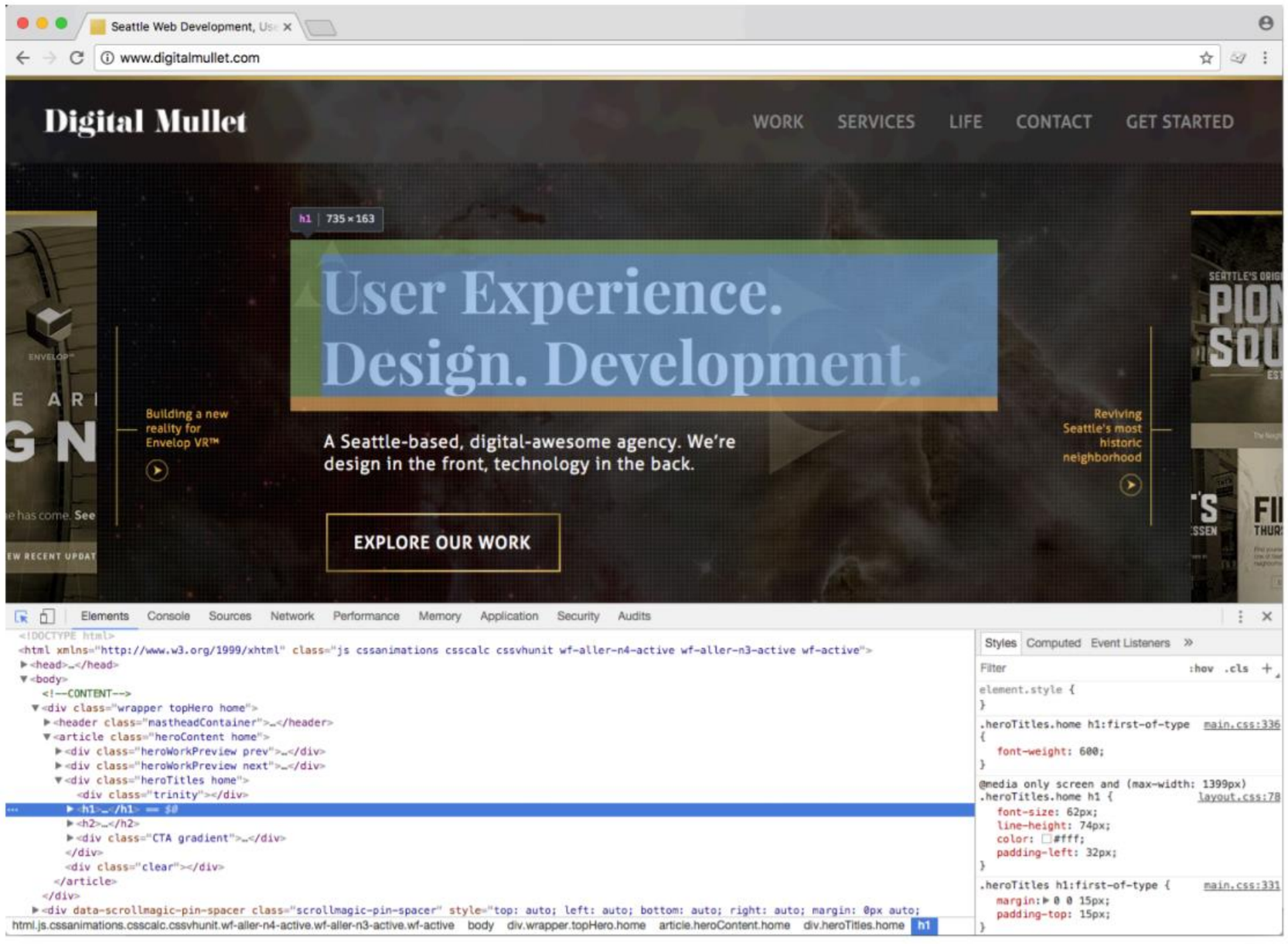
LINKS

```
<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

DEV TOOLS

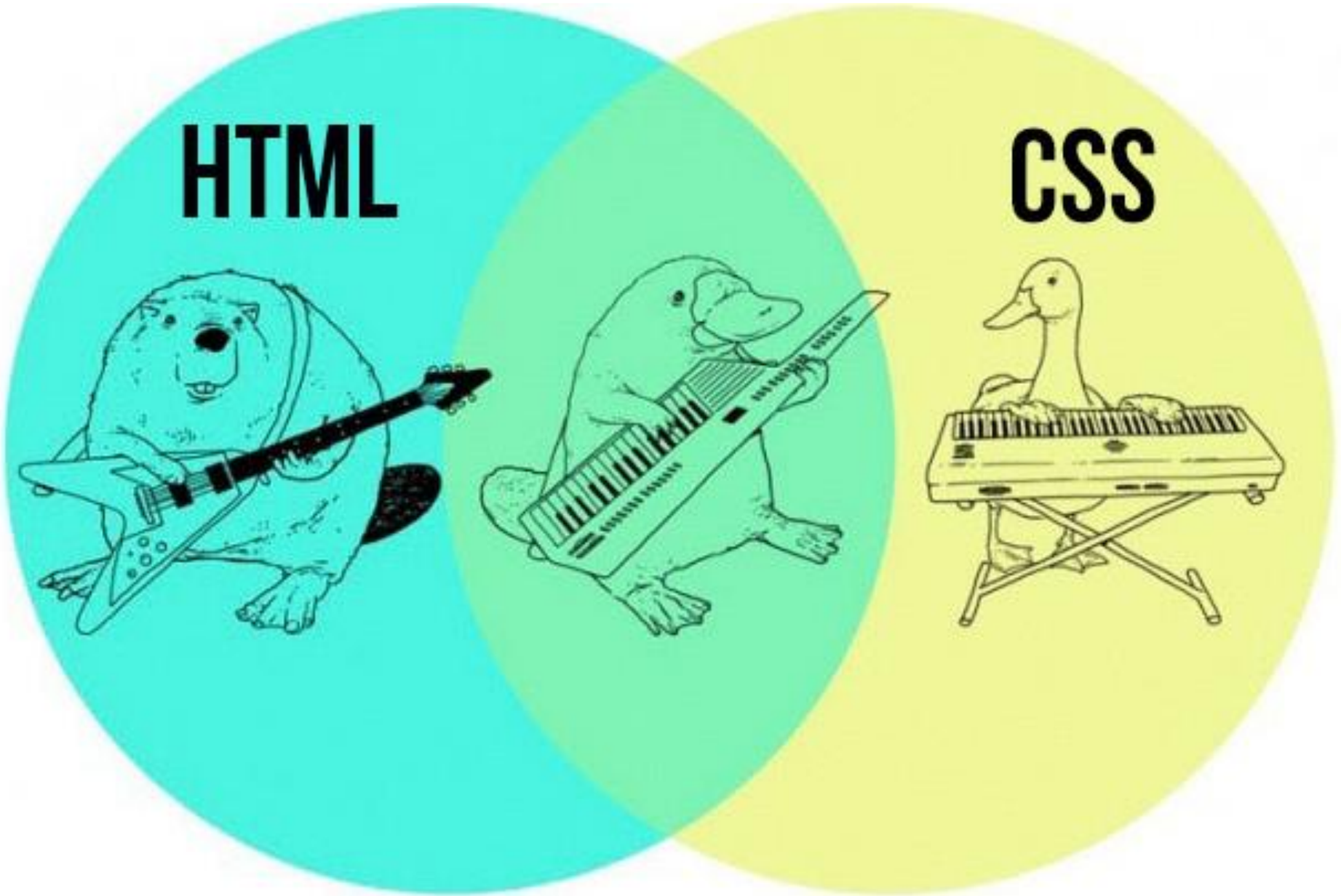
Right-click > Inspect, or hit the F12 key



QUESTIONS?

CSS

HTML + CSS = WEBPAGE



CASCADING STYLE SHEETS

- CSS is a language for specifying how documents are presented to users
- Ability to override the browser's default presentation styles with custom versions
- 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 cover 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 - for example, update the font for the whole site in one line of code!
- Separating presentation from content enforces style consistency and allows flexibility

CSS GOES WHERE?

CSS is a different type of language than the HTML we did last week, and has its own syntax.

- CSS can go directly in your HTML file, inside a `<style></style>` element

- You can also create a .css file that can be linked to your HTML page

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!

```
<style>
    p { color: blue; }
</style>
```

"All paragraphs will have blue text "

EXAMPLE CSS RULE

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

- `selector` is `p` (all `<p>` tags in the HTML)
- `property` is `color`
- `value` is `blue` (many color names are supported, or use the hex code `#0000ff`)

EXAMPLE CSS RULE

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

- Multiple `properties` can be defined for a single `selector`, each separated by a semi-colon (`;`)

{} COMMON FONT PROPERTIES

`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

```
p { line-height: 1.4em; }
```

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

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

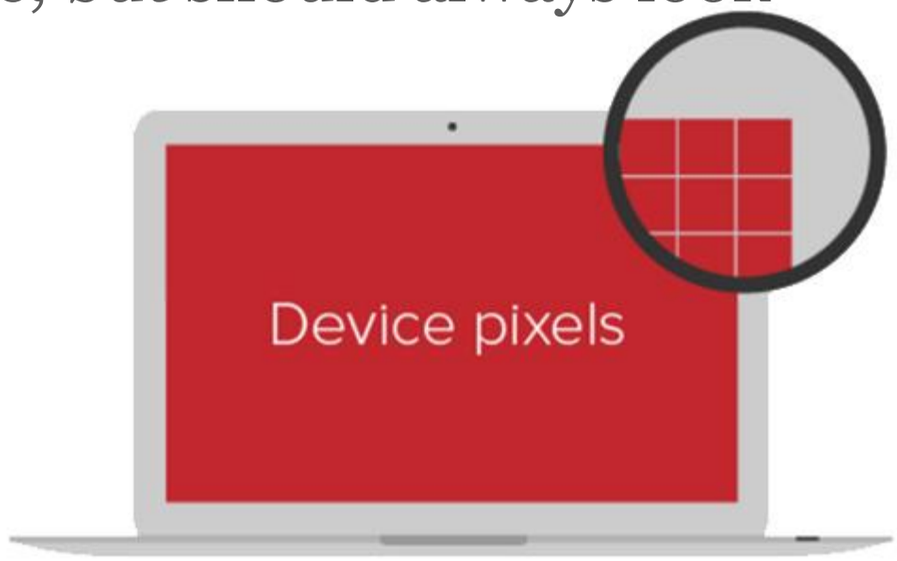
{} QUICK ASIDE ABOUT UNITS

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

- **px** is an abstract unit that isn't related to font height and isn't a physical unit of measurement

- Devices with more PPI (pixels per inch) may use several "device" pixels when displaying a 1px line

- That means that px size varies by device, but should always look "about the same"



{} QUICK ASIDE ABOUT UNITS

{} AH-EM

- **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

{} THAT WASN'T QUICK

Because em is relative, that means that if the parent's font size is increased, the children will get bigger too.

	`body { font-size: 100%; }`	`body { font-size: 120%; }`
`font-size: 1em`	The quick brown fox	The quick brown
`font-size: 12px`	The quick brown fox	The quick brown fox

{} COLORS

- `color:` changes the color of text
- `background-color:` sets the background color of an element
- Color `value` can be set using **names**, **HEX**, **RGB**, or **RGBA**

  - Name: `white`
  - Hex: `#ffffff`
  - RGB: `rgb(255, 255, 255)`
  - RGBA: `rgba(255, 255, 255, 0.8)`

{} COMMON FONT PROPERTIES

`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)

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

```
p {
    font-family: Arial, Helvetica, sans-serif;
}
```

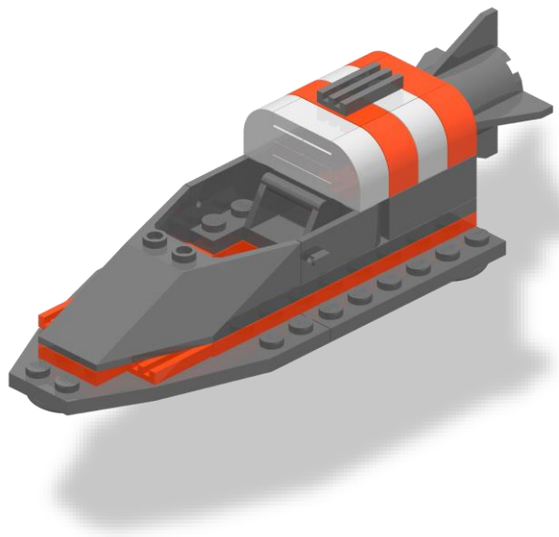
- The W3 has a list of "web safe" fonts that most people will have installed locally

{} FOUR LINK STATES

You can style a link differently depending on what **state** it's in

a:link

a:visited



a:hover

a:active

{} FOUR LINK STATES

```
a { color: blue; }

a:visited { color: gray; }

a:hover { color: purple; }

a:active { color: yellow; }
```

Let's inspect a live demo of how this looks

{} TEXT-ALIGN

You can change the alignment of text using the `text-align` property.

Values:

- `center`
- `left`
- `right`
- `justify`

```
h1 { text-align: center; }
```

{} MULTIPLE SELECTORS & PROPERTIES

- You can add multiple `selectors` to a CSS rule
- You can add multiple `properties` to a CSS rule

```
<style>
      ul {
            color: #ffffff;
       }

      ol {
            font-size: 16px;
            font-weight: bold;
            color: #444444;
       }
</style>
```

{} CSS COMMENTS

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

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

PRACTICE TIME!

PRACTICE

- Add a `<style></style>` section in the `<head>` on your homepage (index.html)

- Make some style changes using CSS
  - Consider changing font color, font family, font size, link color, text alignment, and background colors

FILE ORGANIZATION

FILE ORGANIZATION

- If you structure your site correctly, you are one step closer to faster updates
    - Structure should be not just for you, but for anyone who might use, need or want any of your files (images, scripts, stylesheets, etc)
- The next person to work on or look at your code will be able to understand what you've done and where to find things

FILE ORGANIZATION

Typical files in a website include:

HTML files (.html)
CSS files (.css)
Javascript files (.js)
Images (.png, .jpg, .gif)

- HTML should usually go in the **main** (root) directory

- Make **subdirectories** for media, CSS, and Javascript files

FILE NAMING RULES

- Use a consistent naming convention when naming files and folders
  - For example, always all lowercase, or words always separated by dashes, etc

- Capitalization matters - kittens.png is **not** the same as KITTENS.png

- Use only letters, numbers, hyphens (-) or underscores (_).

- No spaces in file names

- Your homepage is **index.html** by default

CODE ORGANIZATION

- Comment your files – especially if you have unfinished development code, or if you think you may forget why you made the decision you did

```
.viewmore {
    max-height: 2.85714286em; /* line-height of the paragraph x 2 */
}
```

- Indent your code (trust me)

CODE INDENTATION

The Javascript code on the right doesn't have consistent formatting, and is hard to read

```
    swapImages(true);

var t = false;
$(window).on('resize', function () {

          if (t !== false) {
    clearTimeout(t);
        }

 t = setTimeout(swapImages, 200);
});
});

function swapImages(flag) {
  $('img[data-lg-src]').each(function () {
     var $img = $(this);
   var a = $(window).width();
   if (flag) {
       $img.attr("data-sm-src", $img.attr('src'));
          }


    if (a >= 769) {
            $img.attr('src', $img.attr("data-lg-src"));
   } else if (a >= 481) {
             $img.attr('src', $img.attr("data-md-src"));
      } else {
           $img.attr('src', $img.attr("data-sm-src"));
          }
  });
  }
```

CODE INDENTATION

This code is indented, so it's easier to see the "if/else" logic

Comments are added to explain decisions

```
        swapImages(true);

        // On resize, swap in the correct image (after waiting for event thrashing to halt)
        var timer = false;
        $(window).on('resize', function () {

            if (timer !== false) {
                clearTimeout(timer);
            }

            timer = setTimeout(swapImages, 200);
        });
    });

    function swapImages(setMobileImages) {

        $('img[data-lg-src]').each(function () {

            var $img = $(this);
            var windowSize = $(window).width();

            if (setMobileImages) {
                $img.attr("data-sm-src", $img.attr('src'));
            }

            if (windowSize >= 769) {
                $img.attr('src', $img.attr("data-lg-src"));
            } else if (windowSize >= 481) {
                $img.attr('src', $img.attr("data-md-src"));
            } else {
                $img.attr('src', $img.attr("data-sm-src"));
            }
        });
    }
```


CSS

{} CSS IN MULTIPLE PLACES

So far, we've been making CSS changes directly on a single webpage, in the `<head>` element.

- These **internal styles** only apply to that page (but affect every element on that page that is styled)

{} CSS IN MULTIPLE PLACES

You can also add **inline styles** to a single element by using the `style` attribute in HTML markup

```
<p style="color: red">This paragraph is
special.</p>
```

- Inside the `style` attribute, use the same syntax as CSS (selector: value)
- Typically discouraged, because it can be hard to maintain

{} CSS IN MULTIPLE PLACES

The most common way to use CSS in “real life” is to use an **external stylesheet**.

- CSS lives in a separate .css file
- The **same** stylesheet can be included on multiple pages
- A single page can include **multiple** stylesheets

{} 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**

{} FAR TO NEAR

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

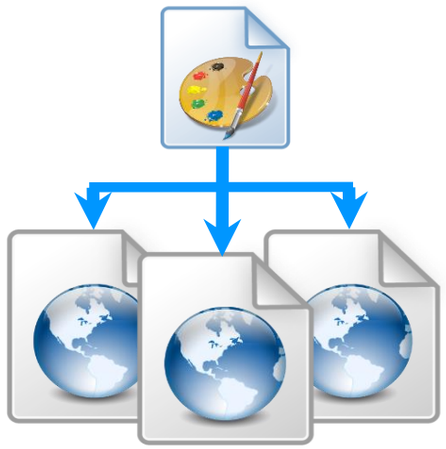
- Browser defaults
- External styles (in a .css file)
- Internal styles (in the `<head>`)
- Inline styles (directly on an element)

Less Specific

Most Specific

{} FAR TO NEAR

Browser default



External styles
(in a **.css** file)

Internal styles
(in the <head>)

Inline styles
(directly on an element)

Less specific to more specific

{} TOP TO BOTTOM

CSS rules are applied sequentially

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

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

b { color: #e7c0c8; } /* bold text in general */

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

PRACTICE TIME!

{} EXTERNAL STYLESHEETS

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

- Remember best practices for file organization

Save your new files as **styles.css**, and save in a new css folder.

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

Create a link to your new stylesheet on all of your webpages.

- Does everything still look the same?

VERSION CONTROL

VERSION CONTROL

In modern development, most websites are a team effort.

Version control systems (VCS) allow multiple people to work on the same file with less risk of overriding changes.

GIT

is an open-source, free VCS

GitHub.com is a free online hosting provider for code

- The website for this class is hosted by GitHub!

“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