TGIS 503: WEB-BASED GIS

Week 1 – Introduction to Web GIS

October 1, 2019

Dr. Emma Slager

TONIGHT'S SCHEDULE

- Intro and welcome
- Syllabus
- Intro to web development and web mapping
- Basics of HTML
- Break
- Basics of CSS
- Break
- Setting up your environment
- Hello World!
- Lab 1 Intro & Open Work Time

MISCELLANEOUS OPPORTUNITIES & ANNOUNCEMENTS

- Cascadia Users of Geospatial Open Source, Fall Fling (basically a one-day conference): https://cugos.org/2019-fall-fling/
  - Free, register at the link
  - Sunday, Oct. 6, 8:30am-4pm, UW Seattle, Bill and Melinda Gates Center for CS & Engineering

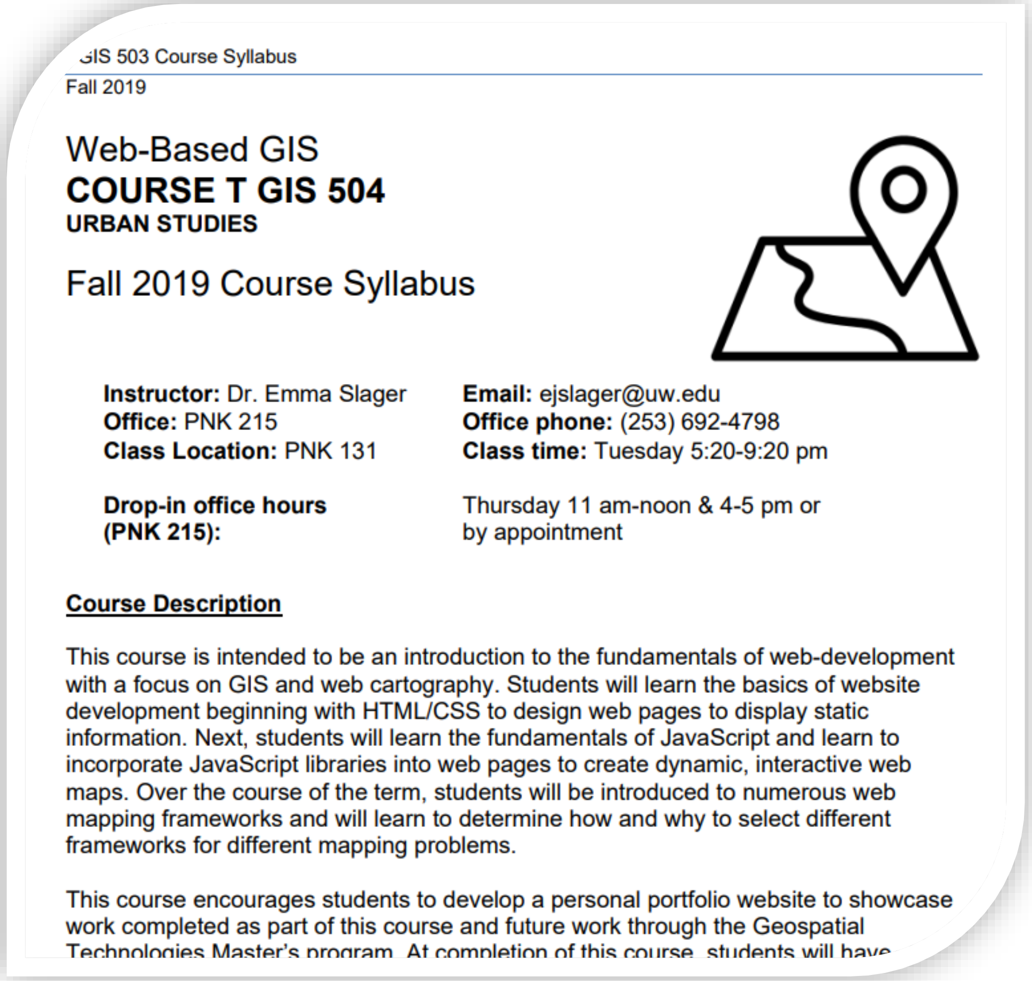
- UW Libraries Storytelling Fellows, Podcasting class, https://docs.google.com/forms/d/e/1FAIpQLSdi38Mm0Bj7YmGITYI5zDGLNg9lSQ_ImhyC2bM9UEDdDXpekQ/viewform
  - Apply by Oct. 11
  - Free, online, sessions that happen on Sunday evenings

INTRODUCTIONS

- Name and pronouns
- Where you’re from
- Tell us a story about your name (e.g. what it means, how you got it)

SYLLABUS & CLASS SCHEDULE

- SYLLABUS ON Canvas
  - Office Hours: Thurs 11-noon & 4-5, OR BY APPOINTMENT
- General Schedule of Assignments:
  - Lab every 2 weeks, new labs introduced after you submit
  - Reading Responses every week (on Canvas)
- Most Importantly:
  - If you're having problems, come see me ASAP



GIS 503 Course Syllabus

Fall 2019

Web-Based GIS

COURSE T GIS 504

URBAN STUDIES

Fall 2019 Course Syllabus

Instructor: Dr. Emma Slager	**Email:** ejslager@uw.edu
Office: PNK 215	**Office phone:** (253) 692-4798
Class Location: PNK 131	**Class time:** Tuesday 5:20-9:20 pm
Drop-in office hours (PNK 215):	Thursday 11 am-noon & 4-5 pm or by appointment

Course Description

This course is intended to be an introduction to the fundamentals of web-development with a focus on GIS and web cartography. Students will learn the basics of website development beginning with HTML/CSS to design web pages to display static information. Next, students will learn the fundamentals of JavaScript and learn to incorporate JavaScript libraries into web pages to create dynamic, interactive web maps. Over the course of the term, students will be introduced to numerous web mapping frameworks and will learn to determine how and why to select different frameworks for different mapping problems.

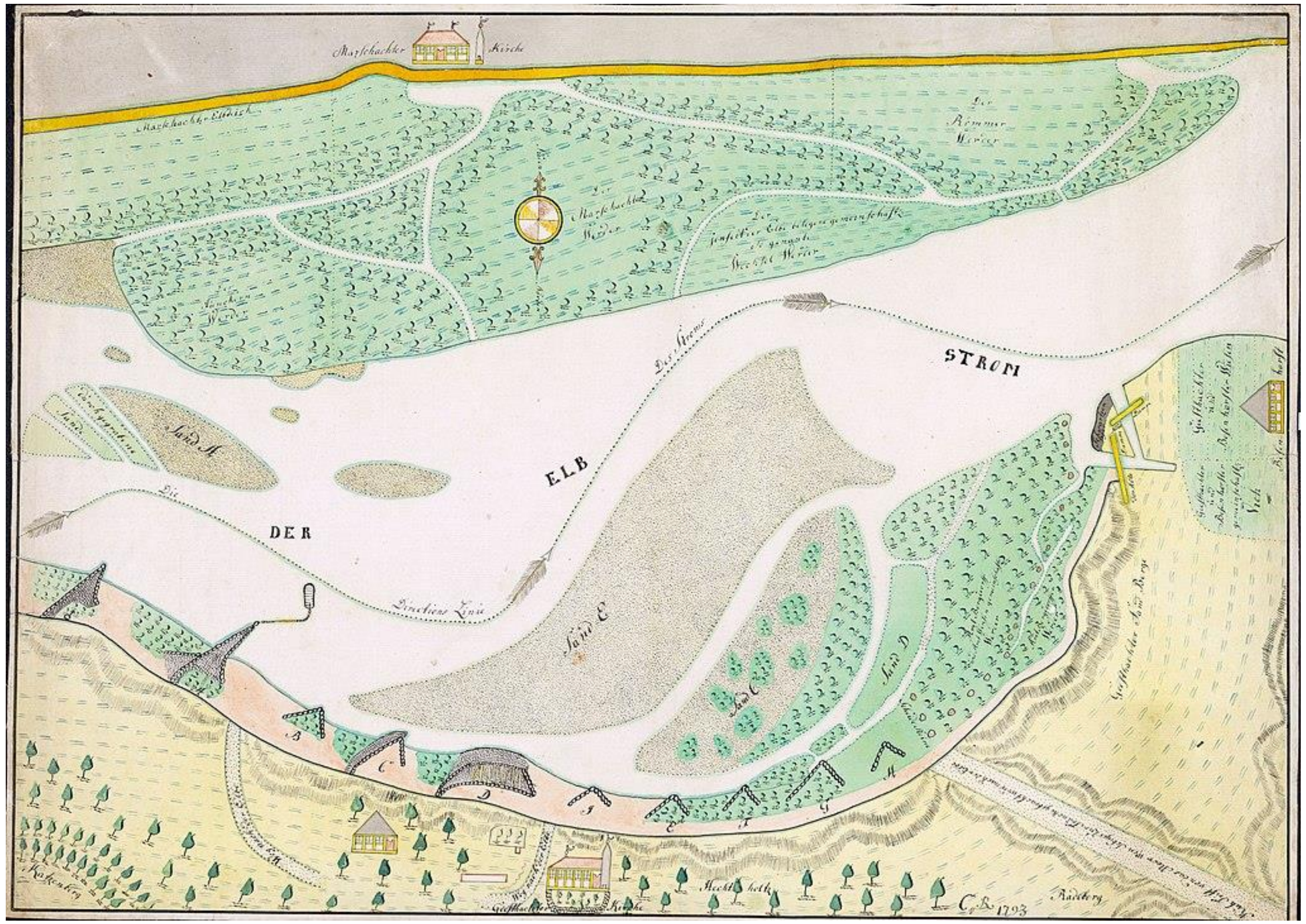
This course encourages students to develop a personal portfolio website to showcase work completed as part of this course and future work through the Geospatial Technologies Master's program. At completion of this course, students will have

WEB MAPPING: A PARADIGM SHIFT

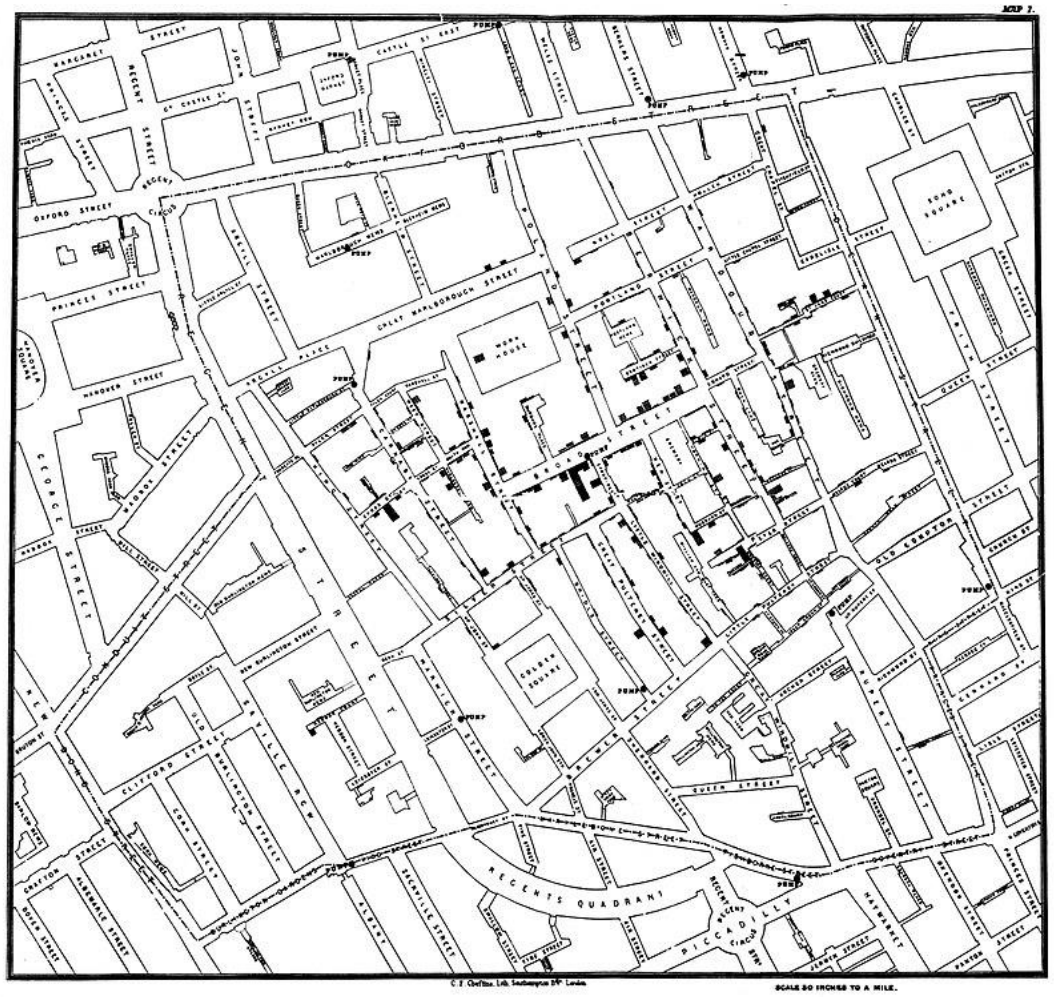
- Muehlenhaus says…

  thematic mapping : 19th century cartography ::

  web mapping : contemporary cartography



1793 map of Hamburg, Germany. Public Domain, http://resolver.sub.uni-hamburg.de/goobi/PPN611997878



John Snow's 1854 map of cholera cases in London. Public Domain, http://matrix.msu.edu/~johnsnow/images/online_companion/chapter_images/fig12-5.jpg

WEB MAPPING: A PARADIGM SHIFT

- Muehlenhaus says…

  thematic mapping : 19th century cartography ::

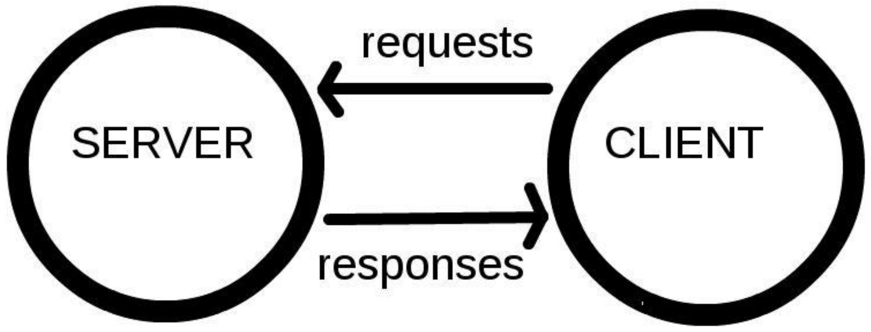
  web mapping : contemporary cartography
- Maps are now:
  - Interactive
  - Multimedia
  - Immersive
  - Easier than ever to make (but still very hard to make well)

WEB MAPPING EXAMPLES

- Let’s look at some examples made by students in last year’s cohort:
  - https://kmiles94.github.io/Capstone_Final/twocolumn1.html
  - https://hrharo.github.io/capstone/pnw_deglaciation_var2.html
  - https://rachel-saunders.github.io/#maps
  - https://tacomajapantown.github.io/dailypaths.html

WEB DEVELOPMENT 101

- The server is where website files and data live
- Your browser downloads the files it needs to show you the page
- Other data is downloaded as needed



- To display a webpage the client gets the:
  - HTML
  - CSS
  - JS

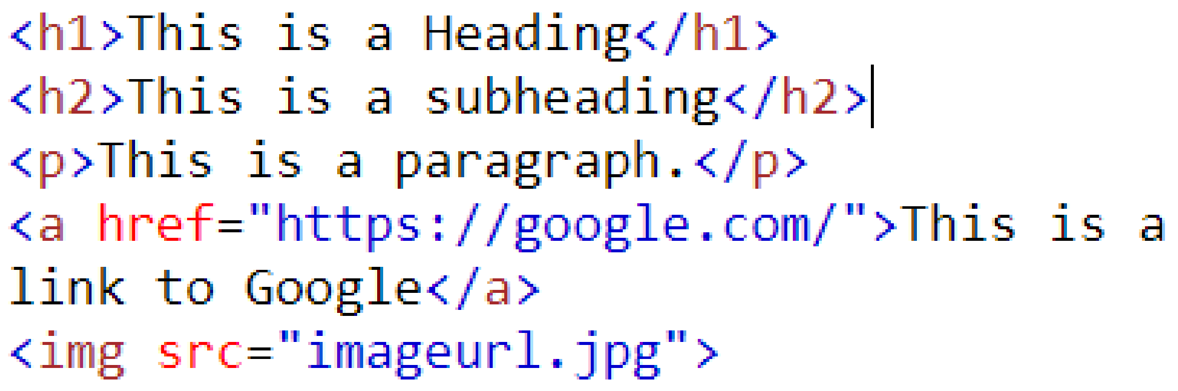
This is our focus for the class

HTML/CSS/JS – WORKING TOGETHER

- HTML is the *content* of a site
  - Provides all of the raw material that makes up the webpage, such as text, images, and hyperlinks.
- CSS is the *style* of a site
  - A set of rules that dictate how to display HTML elements to add colors, fonts, define image sizes, etc.
- JavaScript is the *functionality* of a site
  - Defines what happens when the user interacts with the webpage in different ways, such as loading the page, clicking a button, etc.

HTML –HYPERTEXT MARKUP LANGUAGE

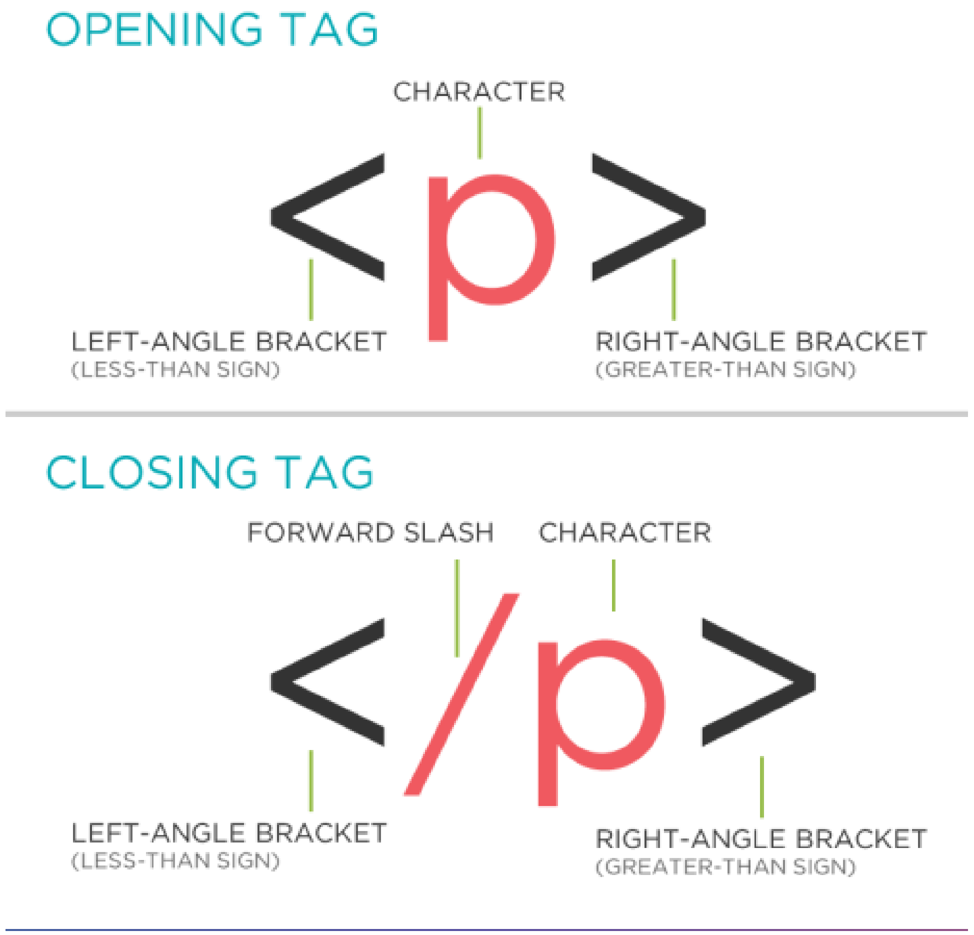
- Is the CONTENT of the webpage
- Made up of a series of *elements*, which are denoted with *tags*
- An element is an individual component of HTML, such as a heading, paragraph, a link, an image, etc.



```
<h1>This is a Heading</h1>
<h2>This is a subheading</h2>
<p>This is a paragraph.</p>
<a href="https://google.com/">This is a
link to Google</a>
<img src="imageurl.jpg">
```

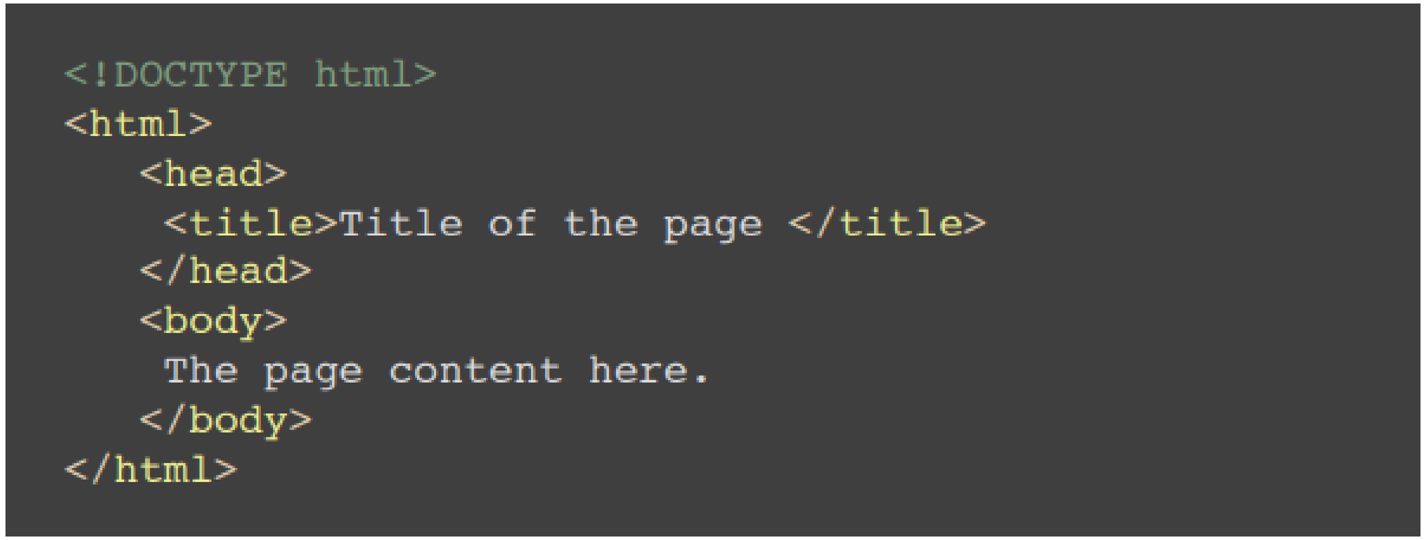
HTML ELEMENTS

- Most elements require opening and closing tags
- **Container elements** can contain other elements or content
  - E.g. <p>This is a paragraph with <strong>emphasis</strong> on one word</p>
- A **standalone element** cannot contain anything else. These elements do not have closing tags
  - E.g. forced line break <br/> or image <img/>



Live example: https://www.w3schools.com/code/tryit.asp?filename=G8EP245E9O8T

HTML STRUCTURE

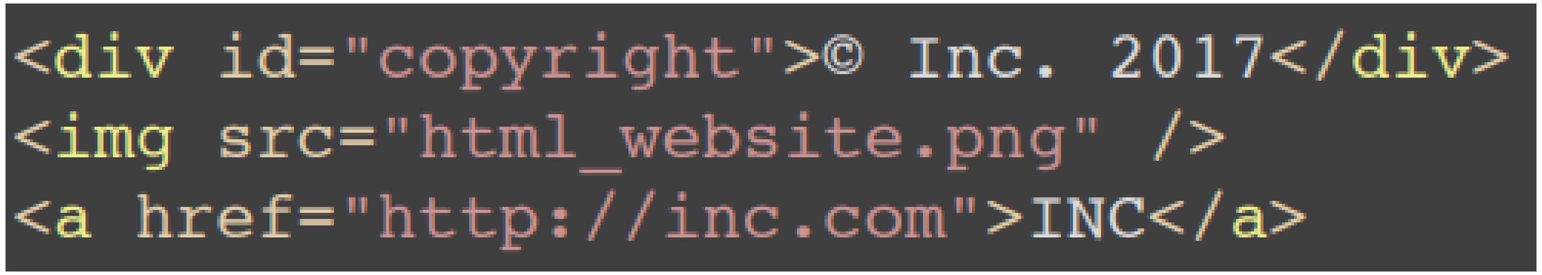


```
<!DOCTYPE html>
<html>
   <head>
    <title>Title of the page </title>
   </head>
   <body>
    The page content here.
   </body>
</html>
```

- The head contains the title of the page and metadata about the page. Metadata is not visible to the user but has many purposes.
- The body contains all of the content of the page that is visible to the user.
- All elements “nest” inside one another. Whichever opens first closes last.
- Note the layout & spacing of the different element tags makes the code easier to read

MORE ABOUT ELEMENTS

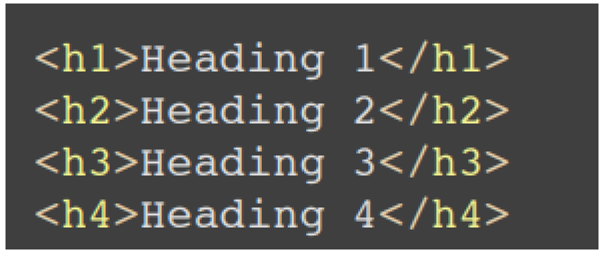
- Elements *can* have **attributes**, such as a class, id, or source
- Attributes are placed inside an opening tag, before the right angle bracket, and they are followed by an equals sign
- **Values** are assigned to a given attribute, and they must be contained inside quotations marks



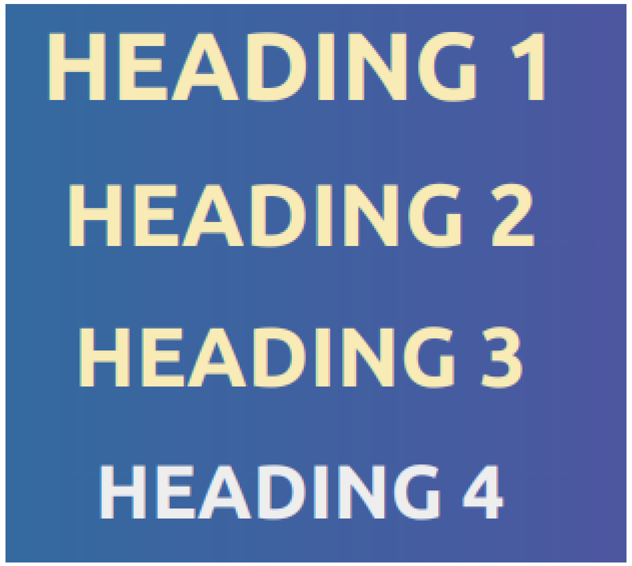
```
<div id="copyright">© Inc. 2017</div>
<img src="html_website.png" />
<a href="http://inc.com">INC</a>
```

ELEMENT: HEADING

- Tag: <h1></h1>
- Headings are numbered 1-6, with 1 being the largest and 6 being the smallest

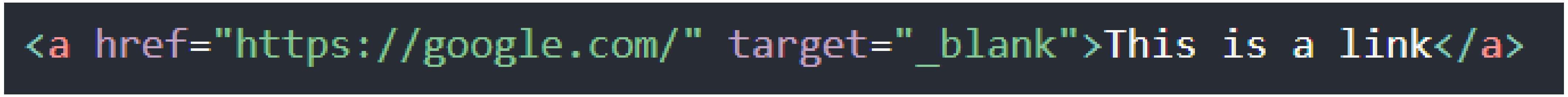


```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
```



ELEMENT: LINK

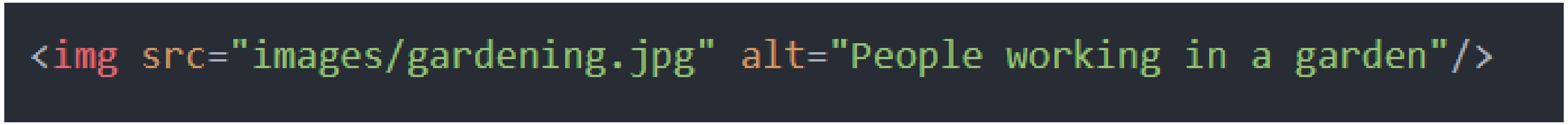
- Tag: <a></a>
- The <a> tag surrounds text or images to turn them into links
- Necessary attribute: **href** (stands for "hypertext reference")
- Common optional attribute: **target**. This can be used to open a link in a new tab.



```html
<a href="https://google.com/" target="_blank">This is a link</a>
```

ELEMENT: IMAGE

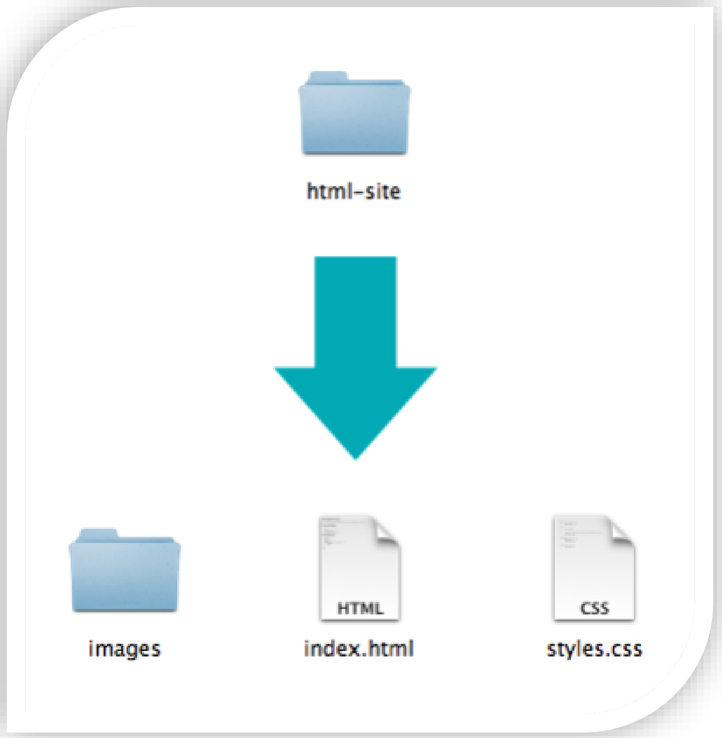
- Images have three components:
  - Tag: **<img/>** [note: does not have a closing tag]
  - Source attribute: **src="images/gardening.jpg"**
  - Alternate text attribute: **alt="People working in garden"**
    - The value of the alt attribute should describe the image. This is if users for some reason cannot view images (because of a slow connection, and error in the src attribute, or if the user uses a screen reader).



```html
<img src="images/gardening.jpg" alt="People working in a garden"/>
```

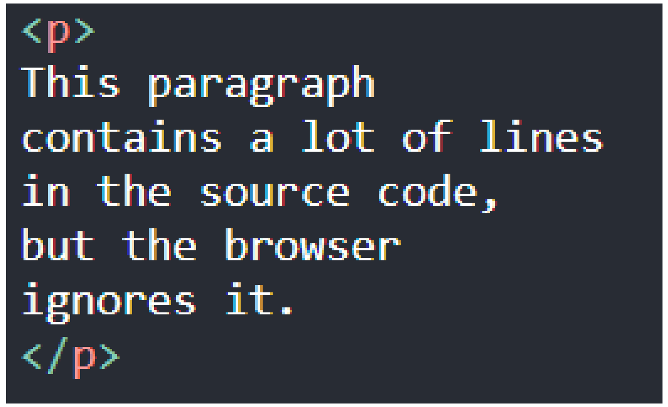
RELATIVE AND ABSOLUTE PATHS FOR LINKS AND IMAGES

- Relative
  - Used when you have created a directory to hold all of the images for your site
  - Links within the same directory need no path information. “filename.jpg”
  - Subdirectories are listed without preceding slashes. “images/filename.jpg”
  - *This is the best practice for storing and referencing all images on your site*
- Absolute
  - Refer to a specific location of a file, including the domain. “http://www.images.google.com”
  - Typically used when pointing to a link that is not within your own domain.



ELEMENT: PARAGRAPH

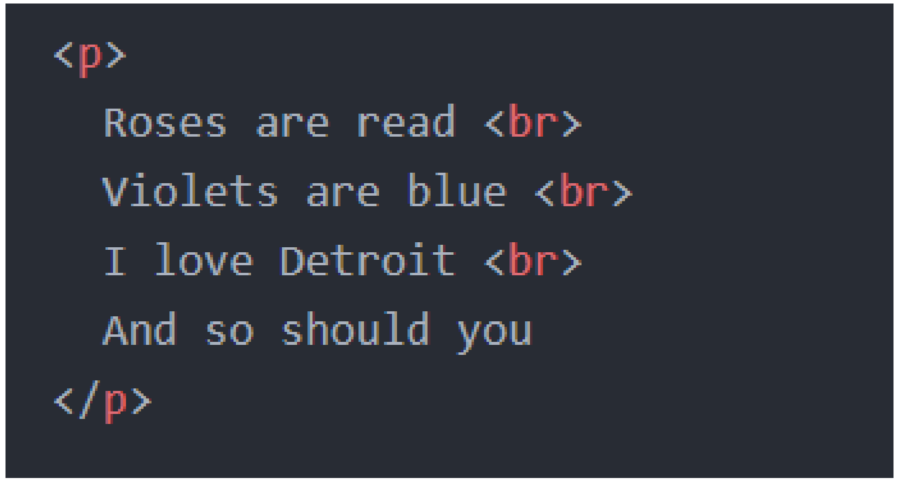
- Tag: <p></p>
- This is the most common text element.
- The browser automatically adds a margin (white space) before and after a paragraph
- The browser will ignore any extra spaces and extra lines inside a paragraph tag when the page is displayed.



```html
<p>
This paragraph
contains a lot of lines
in the source code,
but the browser
ignores it.
</p>
```

ELEMENT: LINE BREAK

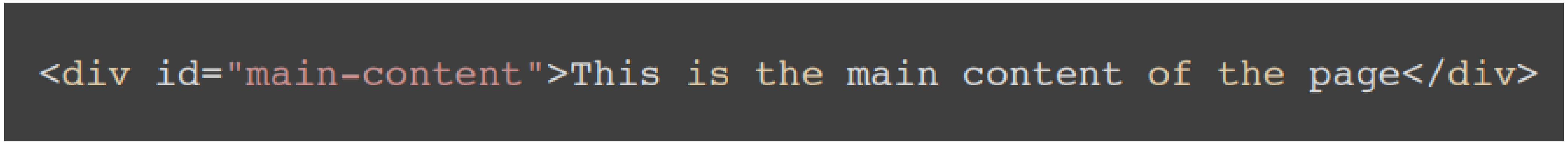
- Tag: <br/> [**does not need a closing tag, the slash is optional]**
- Use a line break if you want to start a new line without starting a new paragraph.
- An example where you might do this is in formatting a poem.
- In general, avoid line breaks when paragraphs will do.



```
<p>
  Roses are read <br>
  Violets are blue <br>
  I love Detroit <br>
  And so should you
</p>
```

ELEMENT: DIV

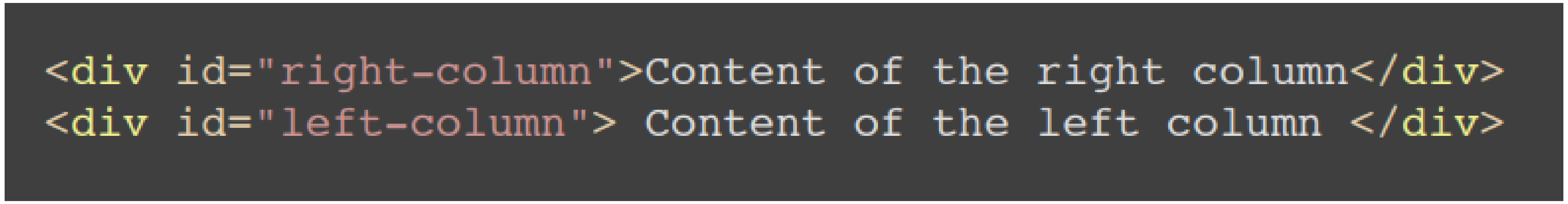
- Tag: <div></div>
- Widely used to break apart a web page into elements, each with its own layout attributes (we'll get more into this when we learn about CSS). It's useful when you want to apply a special style to a particular element or group of elements.



```
<div id="main-content">This is the main content of the page</div>
```

ELEMENT: ID

- Tag: <id></id>
- Widely used to style an element that is used only once
- In the below example, the right column and the left column can be styled differently because they have different IDs.



```
<div id="right-column">Content of the right column</div>
<div id="left-column"> Content of the left column </div>
```

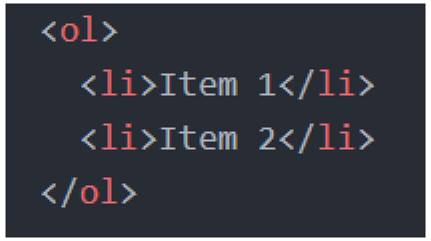
ATTRIBUTE: CLASS

- Note this is an attribute, not an element with its own tag!
- Classes are used when you want to style an element that appears multiple times.
- In the below example, all the blog entries can be styled the same because they have the same class:

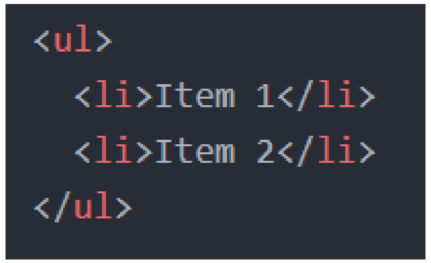
```html
<div class="blog-class">Here is blog entry number 1 </div>
<div class="blog-class">Here is blog entry number 2 </div>
```

ELEMENT: LISTS

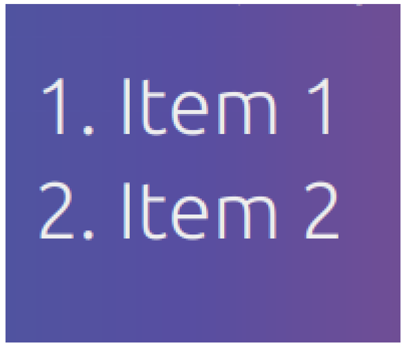
- Tag: <li></li>
- Lists can be ordered (numbered) or unordered (bulleted). Ordered list items are enclosed in the <ol> tag, and unordered list items are enclosed in the <ul> tag.



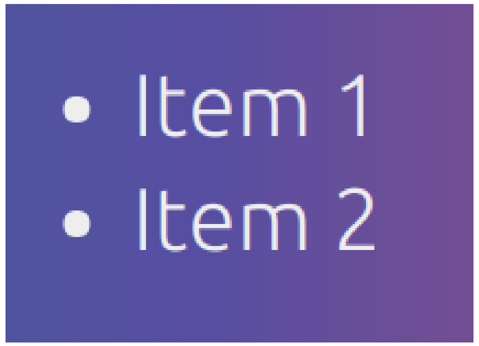
```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```



```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```



1. Item 1
2. Item 2



- Item 1
- Item 2

FORMATTED TEXT

- While some basic styling is possible in HTML, we will do most of this in CSS. However, you may sometimes wish to format snippets of text inside a paragraph or header tag:

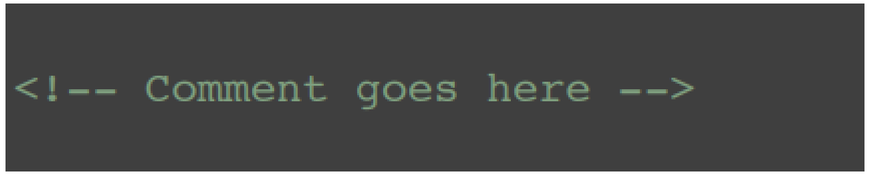
```html
<p>This is your <b>first</b> quarter of grad school.</p>
```

- Common formatting tags include:

<b>	**Bold text**	<ins>	Inserted text
<strong>	**Important text**	<del>	~~Deleted text~~
<i>	*Italic text*	<sub>	Subscript $_{\text{text}}$
<em>	*Emphasized text*	<sup>	Superscript $^{\text{text}}$

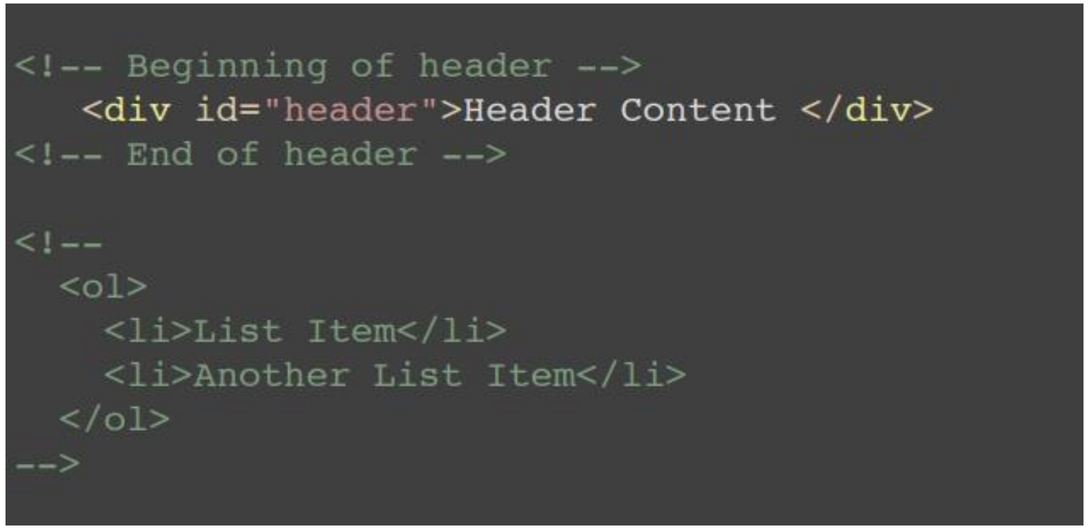
COMMENTING

- You can add comments to your code that will not be seen by the browser but are visible when you view the code



```
<!-- Comment goes here -->
```

- Comments are frequently used to organize code into sections so that you (or someone else) can easily understand the code. It can also be used to 'comment out' sections of code to hide it from the browser, which is helpful in testing and development.



```
<!-- Beginning of header -->
   <div id="header">Header Content </div>
<!-- End of header -->

<!--
  <ol>
    <li>List Item</li>
    <li>Another List Item</li>
  </ol>
-->
```

TRY IT OUT

https://tinyurl.com/503html

- Modify each of the elements.
- Add a list.
- Add commented out text.

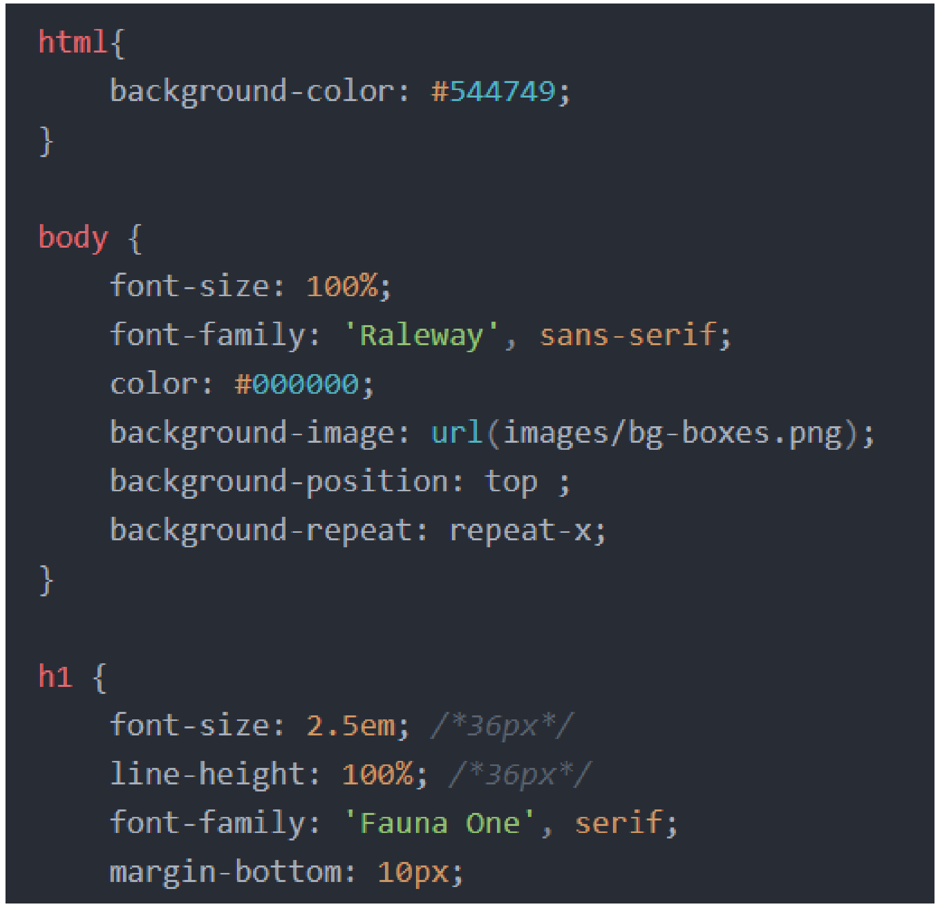
BREAK

HTML/CSS/JS – WORKING TOGETHER

- HTML is the *content* of a site
  - Provides all of the raw material that makes up the webpage, such as text, images, and hyperlinks.
- CSS is the *style* of a site
  - A set of rules that dictate how to display HTML elements to add colors, fonts, define image sizes, etc.
- JavaScript is the *functionality* of a site
  - Defines what happens when the user interacts with the webpage in different ways, such as loading the page, clicking a button, etc.

CSS – CASCADING STYLE SHEETS

- Is the STYLE of the webpage
- A set of rules that dictate how to display HTML elements
- Allows you to style the elements on your page by changing properties like their **color, size,** and **position**



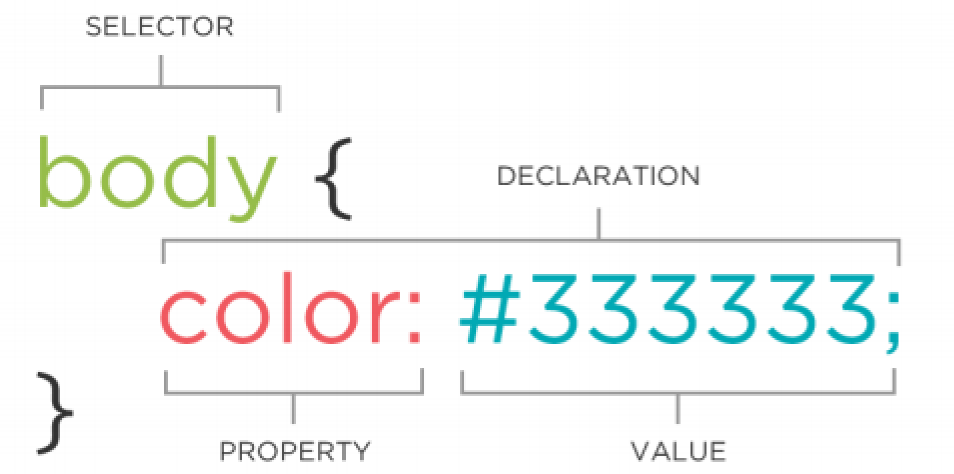
```css
html{
    background-color: #544749;
}

body {
    font-size: 100%;
    font-family: 'Raleway', sans-serif;
    color: #000000;
    background-image: url(images/bg-boxes.png);
    background-position: top ;
    background-repeat: repeat-x;
}

h1 {
    font-size: 2.5em; /*36px*/
    line-height: 100%; /*36px*/
    font-family: 'Fauna One', serif;
    margin-bottom: 10px;
```

CSS SYNTAX

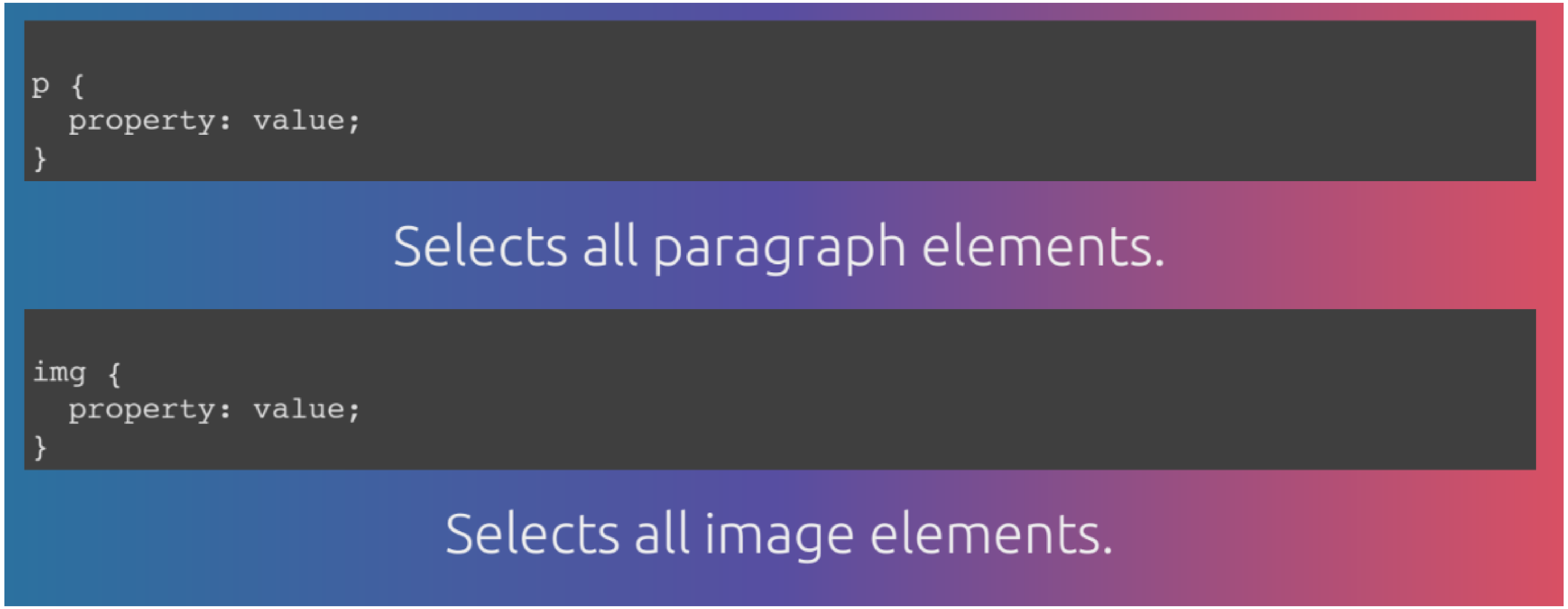
- You reference the element you want to style with a **selector**
- You define the **property** you want to change, followed by a colon :
- You give a **value** to the defined property, followed by a semi-colon ;
- The property and value together are known as the **declaration.** The declaration is enclosed in curly brackets. Multiple declarations can be included in one declaration block, but they must be separated by a semi-colon.



Live example: https://codepen.io/ejeans/pen/RwbmqGq

SELECTING BY ELEMENT

- Most commonly, we'll select by HTML element



```
p {
  property: value;
}
```

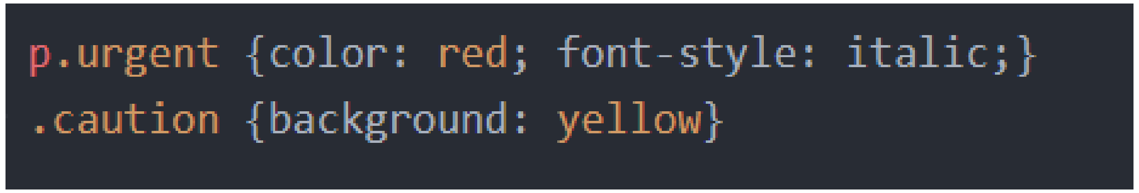
Selects all paragraph elements.

```
img {
  property: value;
}
```

Selects all image elements.

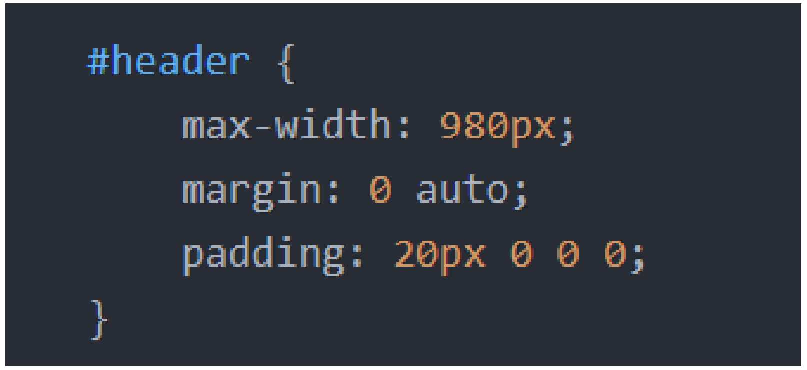
OTHER SELECTORS

- You can select by **class** using dot notation



```
p.urgent {color: red; font-style: italic;}
.caution {background: yellow}
```

- You can select by **ID** using hash notation



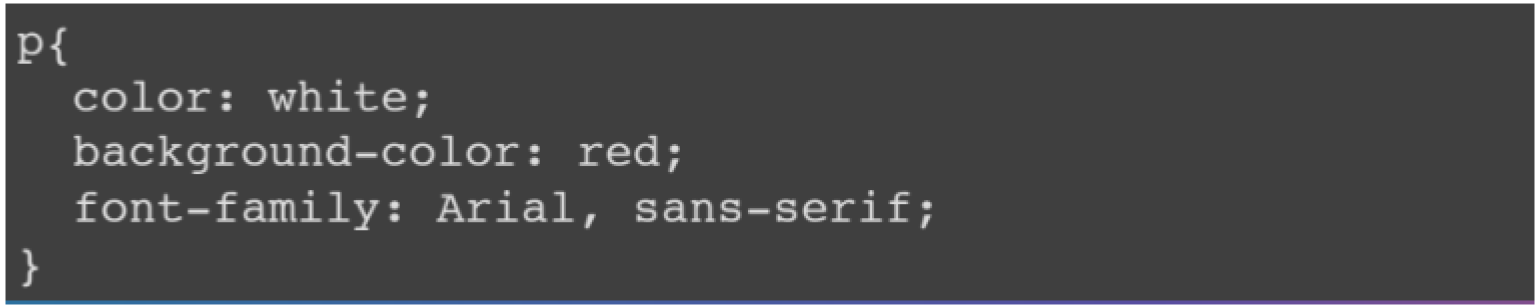
```
#header {
    max-width: 980px;
    margin: 0 auto;
    padding: 20px 0 0 0;
}
```

PROPERTIES & VALUES

- There are hundreds of available CSS properties, and we'll look individually at some of the most common. You can find a full reference list here: https://www.w3schools.com/cssref/default.asp
- Many CSS properties have self-explanatory names:
  - background-color
  - font-family
  - font-size
  - color
  - width
  - height
- Different properties can take different values. Values can be keywords (specific to each property), numbers, colors, or length values.

PROPERTY VALUES

- Each property can have one or more common separated values

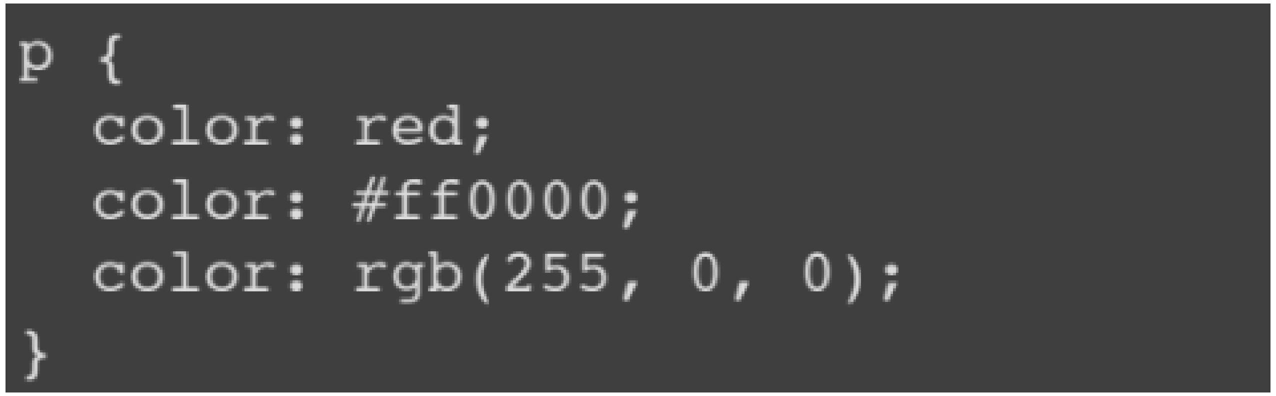


```
p{
  color: white;
  background-color: red;
  font-family: Arial, sans-serif;
}
```

- The “cascade” in Cascading Style Sheets means the browser will attempt to render the values in the order they are listed, but if the font Arial isn’t available (i.e. not built in to your specific browser), it will move on to a generic sans-serif font instead.

PROPERTY: COLOR

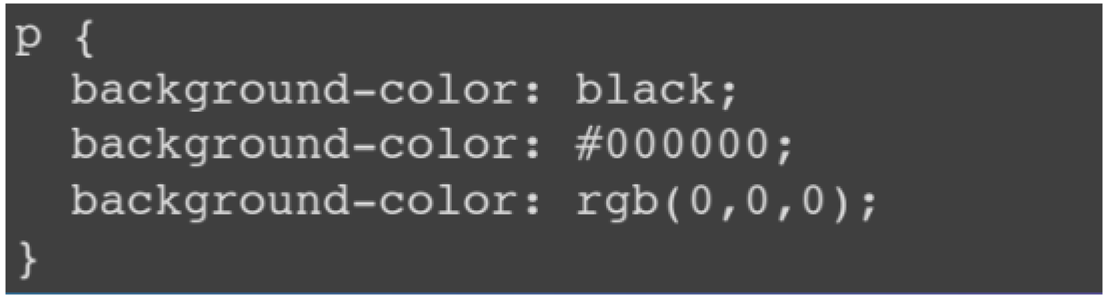
- Changes the color of text
- There are a few different ways to define the value of a color:
  - Name (there are 17 standard, named colors), e.g. aqua, lime, red
  - RGB value (three numbers ranging from 0-225 or three percents ranging from 0-100%. First number is the level of red, second is the level of green, third is the level of blue.), e.g. the RGB for black is (0,0,0)
  - Hexadecimal value (similar to RGB but each color is defined by a pair of digits ranging from 00-FF), e.g. "pure" red is (FF0000)



```
p {
  color: red;
  color: #ff0000;
  color: rgb(255, 0, 0);
}
```

PROPERTY: BACKGROUND-COLOR

- Changes the color of an element's background



```
p {
  background-color: black;
  background-color: #000000;
  background-color: rgb(0,0,0);
}
```

- What's another element we might change the background color of?

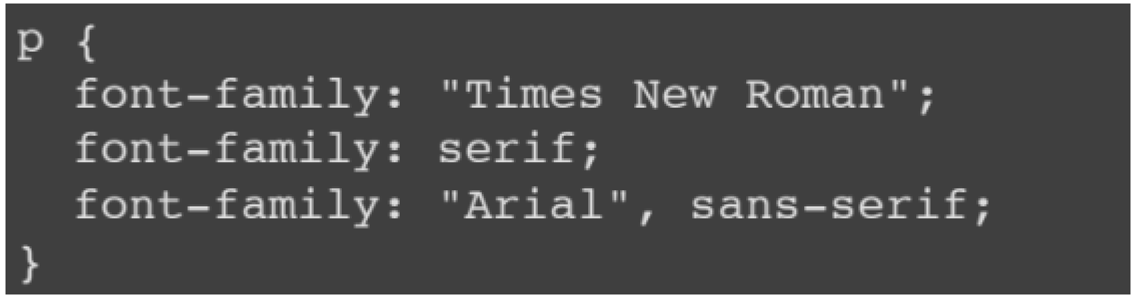
COLOR REFERENCE

- There are many different tools for finding the RGB value or hexadecimal value of the precise color you want. Here's one: https://www.w3schools.com/colors/colors_picker.asp

- Just as you should be aware of colorblindness when you select map colors, you should keep in mind colorblind-safe choices when designing web pages. Here is a good, brief, intro guide: https://davidmathlogic.com/colorblind/

PROPERTY: FONT-FAMILY

- Defines which font is used

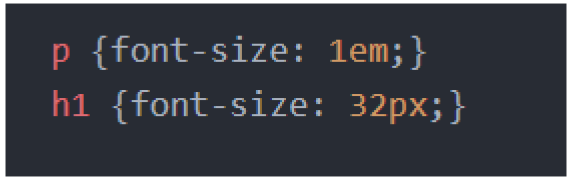


```
p {
  font-family: "Times New Roman";
  font-family: serif;
  font-family: "Arial", sans-serif;
}
```

- Not every browser has every font built in, so we always use generic back-ups like "serif", "sans-serif", "cursive", "monospace"

PROPERTY: FONT-SIZE

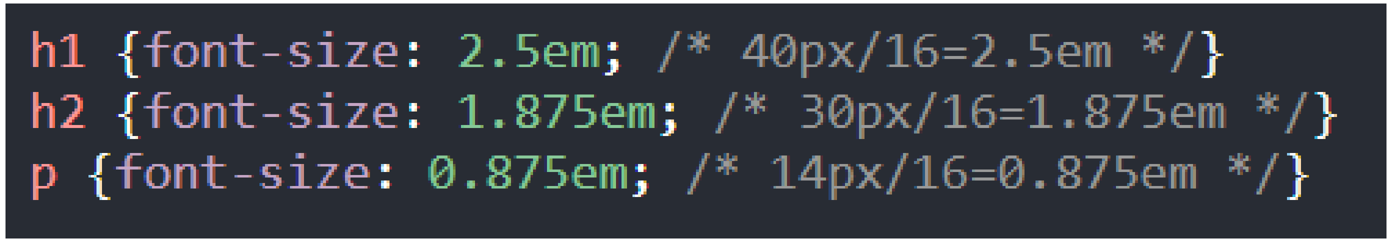
- Specifies the size of the font
- Expressed as a number followed by a two-letter abbreviation of the units
- Units can be absolute or relative



```css
p {font-size: 1em;}
h1 {font-size: 32px;}
```

FONT-SIZE REFERENCE

- You can use absolute length values (inches, centimeters, points, etc.) to define your font-size, but this is **strongly** discouraged because many computer displays have not built-in reference to "real world" measurements.
- More common length values:
  - Pixels (px): equivalent to a cell on the screen display. There are about 96px in an inch. The default text size in most desktop web browsers is 16px.
  - Em-height (em): this is the recommended way to set font-size as it is the easiest to resize for web and mobile browsers. 1em is equal to the current font size. So 1em = 16px in default settings. It functions similarly to percent.

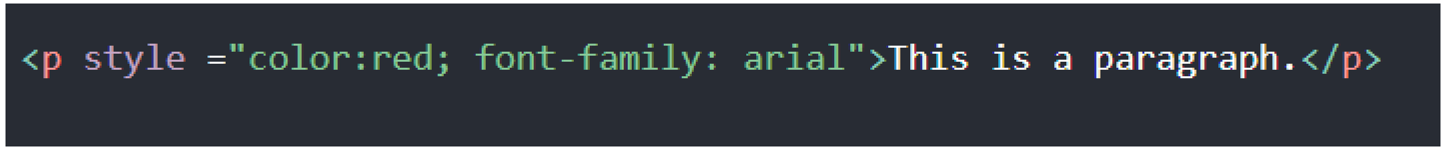


```css
h1 {font-size: 2.5em; /* 40px/16=2.5em */}
h2 {font-size: 1.875em; /* 30px/16=1.875em */}
p {font-size: 0.875em; /* 14px/16=0.875em */}
```

INCORPORATING CSS IN HTML

Three ways:

1. **Inline styles**
2. Embedded style sheets
3. External style sheets



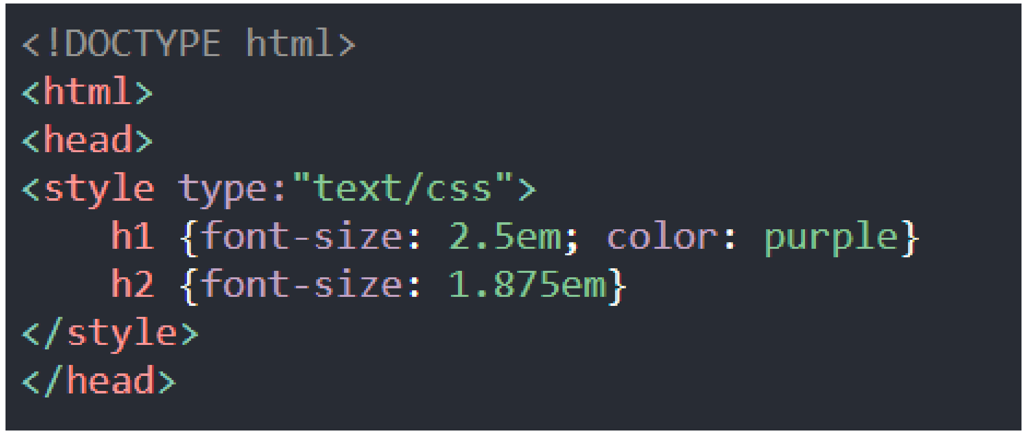
```html
<p style ="color:red; font-family: arial">This is a paragraph.</p>
```

This is strongly discouraged because it is highly inefficient.

INCORPORATING CSS IN HTML

Three ways:

1. Inline styles
2. **Embedded style sheets**
3. External style sheets



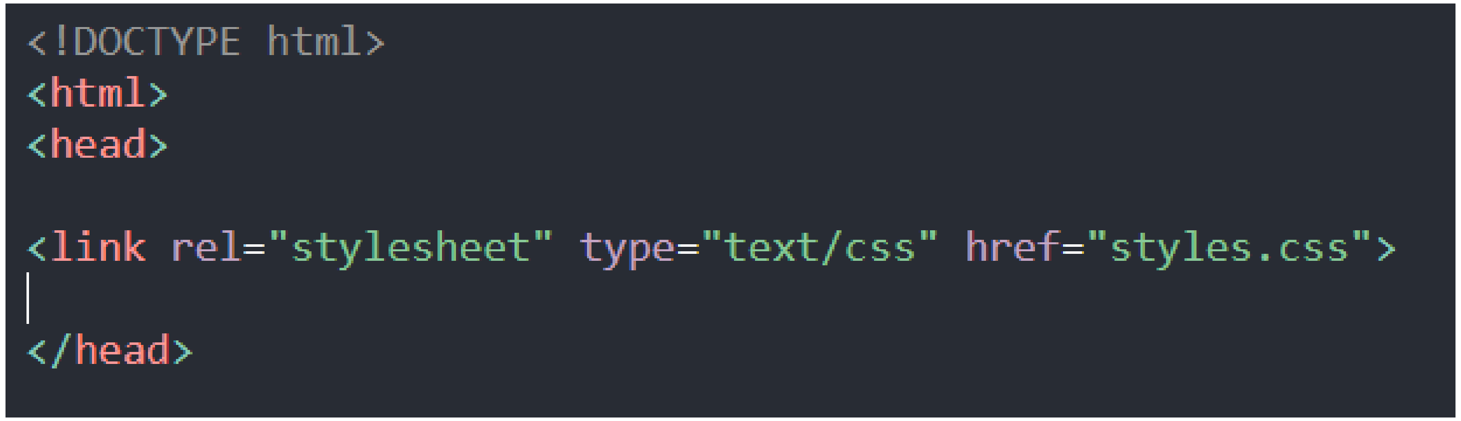
```
<!DOCTYPE html>
<html>
<head>
<style type:"text/css">
    h1 {font-size: 2.5em; color: purple}
    h2 {font-size: 1.875em}
</style>
</head>
```

This is better than in-line styling because it keeps all of your CSS in one, easy-to-find place, which makes it easier to edit.

INCORPORATING CSS IN HTML

Three ways:

1. Inline styles
2. Embedded style sheets
3. **External style sheets**



```
<!DOCTYPE html>
<html>
<head>

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

</head>
```

This is the best practice because all pages in a website that use a single style can be updated simultaneously by editing a single style sheet. Additionally, external style sheets are cached by web browsers, which can reduce bandwidth usage.

TRY IT OUT

https://codepen.io/ejeans/pen/RwbmqGq

- Change some of the fonts
- Change the color of the background
- Add a selector for the img and give it the following declaration:

```
width: 100%;
```

- If you finish quickly and want a challenge, create a div to hold the body content and center the div. Center all of the text within the div.

BREAK

- After the break, we'll start building a website using a text editor. If you plan to use a personal computer, please be sure to download a text editor such as
  - Atom: https://atom.io/
  - Notepad++: https://notepad-plus-plus.org/
  - Sublime: https://www.sublimetext.com/
- I also recommend downloading Google's Chrome browser if you do not have it.

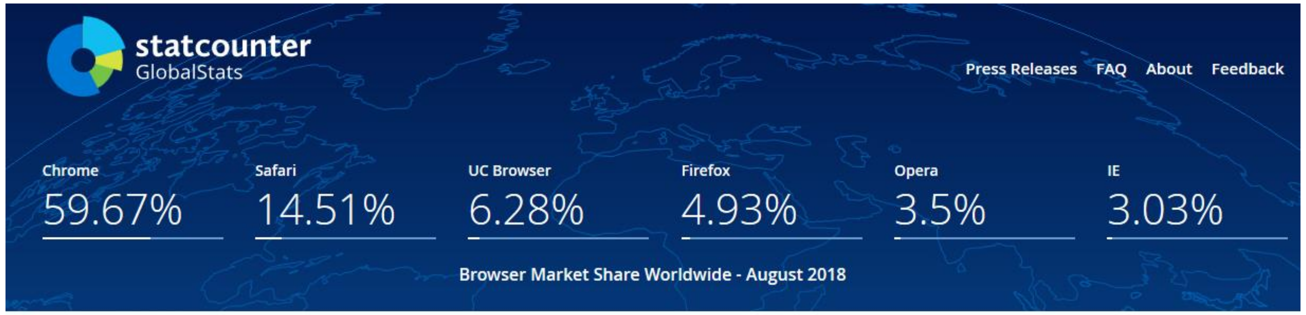
DEVELOPING WEBPAGES

- You need:
  - A text editor
  - A browser

- Chrome/Firefox/Edge have developer tools to show you what’s happening

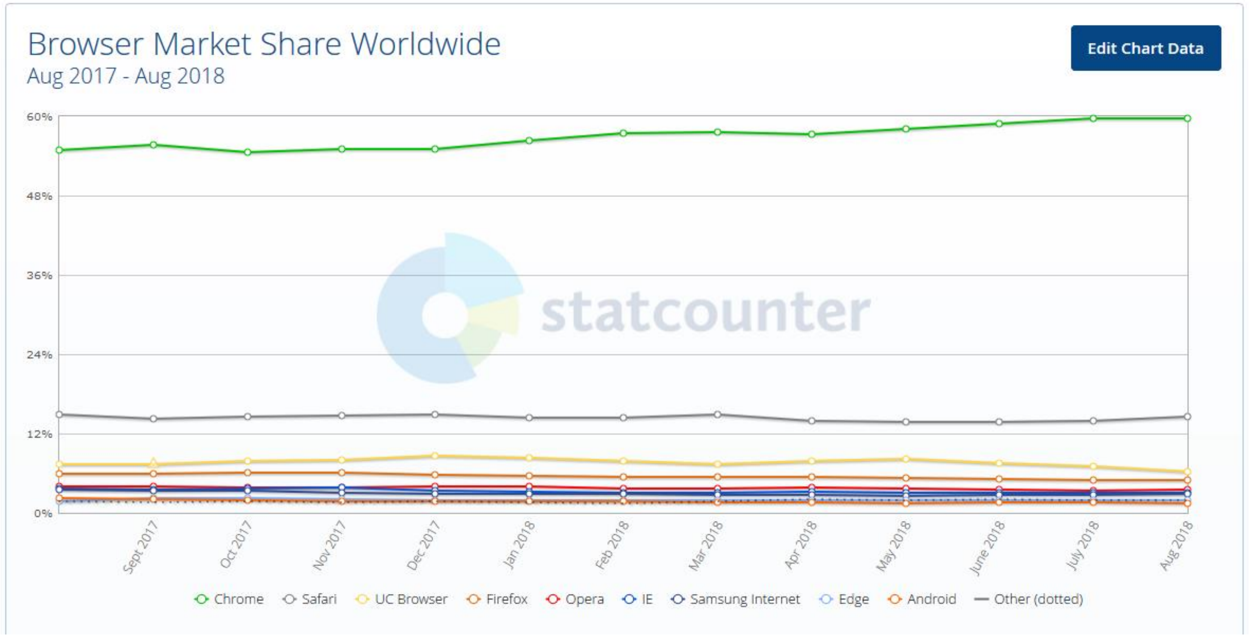
SETTING UP A DEV ENVIRONMENT

- Pick a text editor
  - **Atom**
  - Notepad++
  - Sublime Text
  - Others?
- Pick a browser
  - Chrome/Firefox are standard
  - Safari/Edge
  - IE is dying (and is a hassle)



Chrome	Safari	UC Browser	Firefox	Opera	IE
59.67%	14.51%	6.28%	4.93%	3.5%	3.03%

Browser Market Share Worldwide - August 2018



http://gs.statcounter.com/browser-market-share

MY FIRST HTML

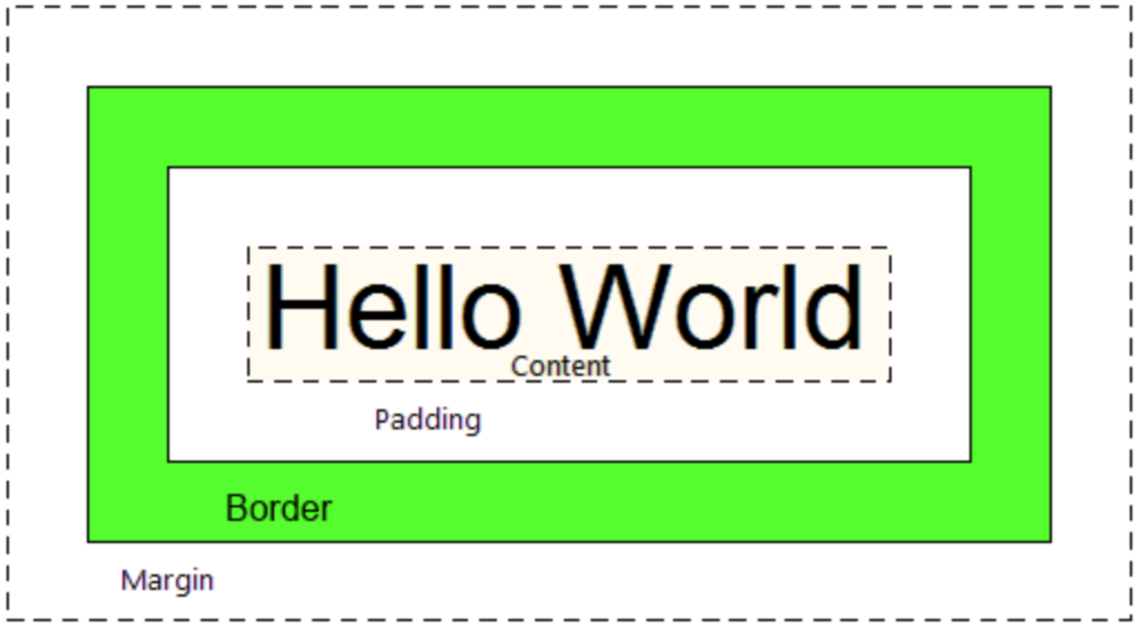
- Make an html file (Traditionally, the landing page is index.html)
- Don't forget to include the REQUIRED HTML tags
  - Doctype
  - Html
  - Head
  - Body

- Let's make it!
- Hello World!

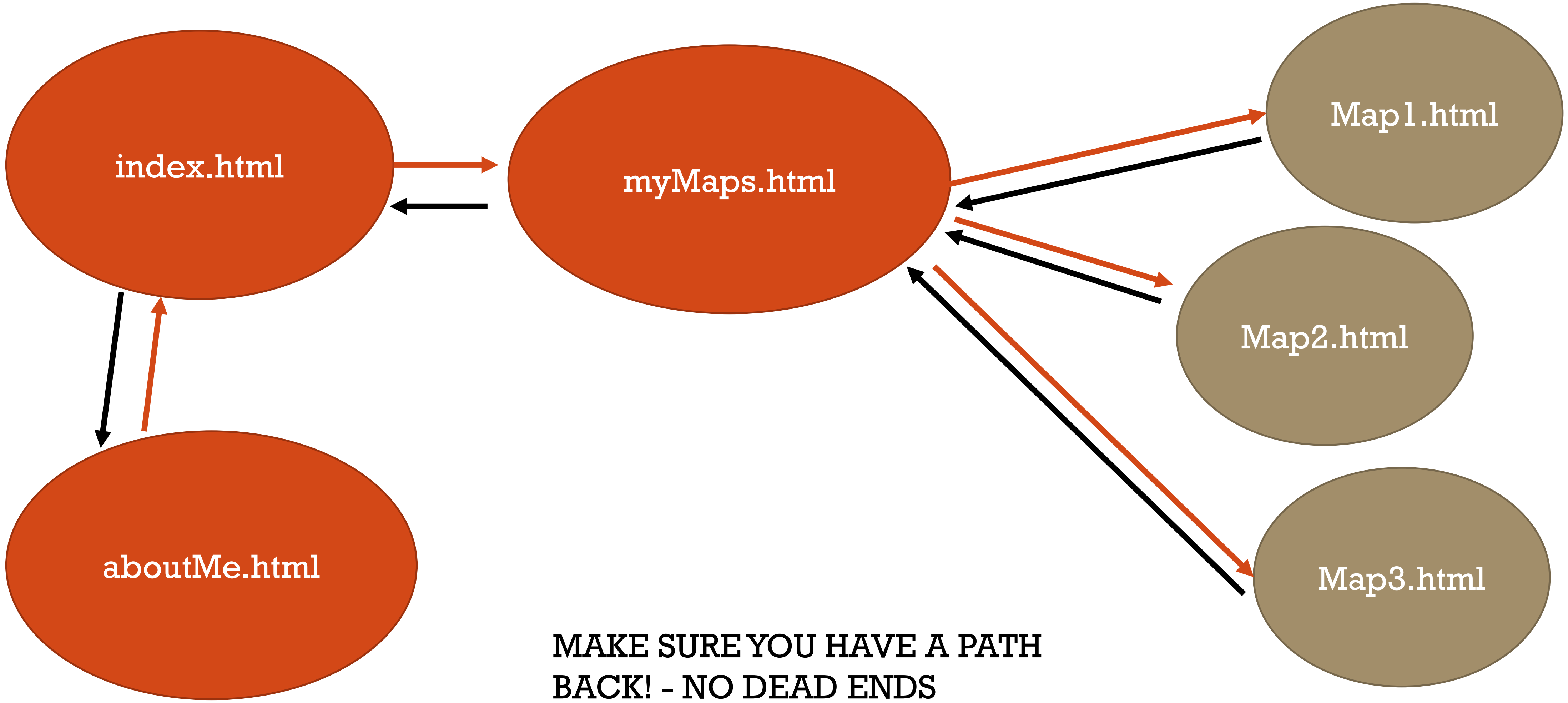
ADDING SOME OTHER CONTENT

- Add a picture
- Add some filler text (Lorem Ipsum)
- Add some styles!
  - In-line
  - In an external style sheet
  - CSS Padding/Margins

- Make another Page
- Hyperlink!
- Navigation PLANNING



NAVIGATION PLANNING - BEGINNING UX DESIGN



MAKE SURE YOU HAVE A PATH
BACK! - NO DEAD ENDS

LAB 1 INTRO

- Labs are written as Product Specification Sheets (Specs). Will become more robust as the quarter progresses. Document contains:
  - Introduction to the problem
  - Some associated background literature that might help you on your way
  - Minimum Viable Product (MVP) is the minimum required for the lab
  - Bonus requirements
  - Rubric for Grading
  - Additional Links or help
- NOTE: These labs DO NOT provide step-by-step instructions on how to accomplish the lab. This course is intended to teach you enough web-development so that you can research on your own

SOURCES/LINKS

- https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works
- https://www.w3schools.com/html/default.asp
- https://www.itonlinelearning.com/blog/how-do-html-css-and-javascript-work-together/#