Week 2: HTML & CSS

INFSCI 2560
Web Technologies & Standards

Agenda for Today

- Introduction to HTML
- Break
- Introduction to CSS
- Activity 2 - Playing around with HTML & CSS

Bookmark these

CodePen http://codepen.io

Allows you to quickly test out code examples. No account needed and it is free.

MDN Documentation

Mozilla Developer Network is great reference for HTML & CSS documentation.

https://developer.mozilla.org/en-US/docs/Learn/HTML

https://developer.mozilla.org/en-US/docs/Web/CSS

Quick Check

What is the world wide web?

What is a web standard?

Why do we have them?

What is HTTP?

What is a status code?

Good article on web standards: https://www.smashingmagazine.com/2019/01/web-standards-guide/

HTTP Requests

```
GET /store/inventory

POST /store/order

GET /store/order/8402

DELETE /store/order/199

PUT /store/order/

PUT /store/order/2345
```

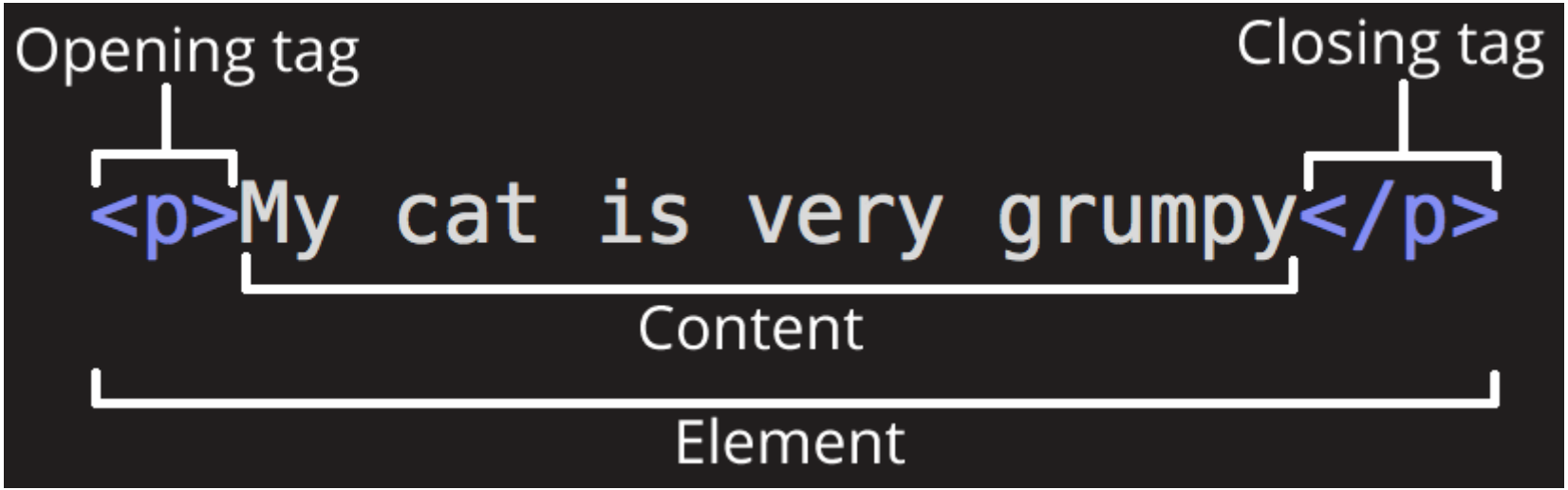
WHAT IS HTML

HTML - Structuring the Web

- HTML is a standard for expressing semi-structured documents
- This structure is made out of *tags*
  - Also called *elements*
- The HTML standard specifies the syntax (structure) and semantics (meaning) of HyperText Markup Language
  - Not a programming language. A *markup language*
- At its most basic, HTML is about *text*.

HTML Syntax - Elements

- **Tags/elements** have an opening tag and closing tag
- Element names and their meaning are defined by the HTML standard (and I don't have time to cover all of them)



HTML Syntax - Element Attributes

- You can add additional information to elements by added **attributes**
- Attributes and their meanings are defined by the HTML specification (and I won't cover all of them)
- Always include quotation marks

Attribute

```
<p class="editor-note">My cat is very grumpy</p>
```

Nesting Elements

- Elements can be placed inside other elements to add structure to text

```
<p>My cat is <strong>very</strong> grumpy.</p>
```

- Make sure nested tags are closed correctly!
- The example below is bad HTML

```
<p>My cat is <strong>very grumpy.</p></strong>
```

Anatomy of an HTML document

```
<!DOCTYPE HTML>
<html>
<head>
<title>Something</title>
</head>
<body>
    <p>
My cat is <strong>very</strong>
grumpy
    </p>
</body>
</html>
```

- The DOCTYPE declaration tells us what version of HTML we are using, this is the doctype for HTML 5
- The **<html>** element is called the **root** element and is the container for the whole document
- The **<head>** element is for containing information *about* the content, but not the content itself. Like the page title
- The **<body>** element is where the actual HTML content displayed to the users will live.
- At the end of the document you must be sure to close all of the open tags

Document Elements

- `<html>` - The root element of all HTML docs
- `<head>` - Container for meta information
- `<meta>` - General metadata tag
- `<title>` - Put the title of your page in here!
- `<link>` - External resource link. Mainly for linking to external CSS stylesheets
- `<style>` - Put your embedded CSS styles inside this element
- `<script>` - For embedding or linking to JavaScript Code
- `<body>` - ALL THE CONTENT GOES HERE!

Example

```
<!DOCTYPE html>
<html>
        <head>
        <title>Wayan's Short Stories</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="mystyle.css">
        <script>
                console.log("Greetings!")
        </script>
    </head>
    <body>
    ...
```

Block vs Inline Elements

- Block-level elements begin on new lines, but inline elements can start anywhere in a line.
- Inline elements may contain only data and other inline elements. You can't put block elements inside inline elements.
- The tag is a block-level element.
- The tag is an inline element.

https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

Block vs Inline Elements

`<a>`
`<abbr>`
`<acronym>`
`<audio>` **(if it has visible controls)**
`<b>`
`<bdi>`
`<bdo>`
`<big>`
`<br>`
`<button>`
`<canvas>`
`<cite>`
`<code>`
`<data>`
`<datalist>`
`<del>`
`<dfn>`
`<em>`
`<embed>`

`<i>`
`<iframe>`
`<img>`
`<input>`
`<ins>`
`<kbd>`
`<label>`
`<map>`
`<mark>`
`<meter>`
`<noscript>`
`<object>`
`<output>`
`<picture>`
`<progress>`
`<q>`
`<ruby>`
`<s>`
`<samp>`

`<address>`
Contact information.

`<article>`
Article content.

`<aside>`
Aside content.

`<blockquote>`
Long ("block") quotation.

`<details>`
Disclosure widget.

`<dialog>`
Dialog box.

`<dd>`
Describes a term in a description list.

`<div>`
Document division.

`<dl>`
Description list.

`<dt>`
Description list term.

`<fieldset>`
Field set label.

`<figcaption>`
Figure caption.

`<figure>`
Groups media content with a caption (see `<figcaption>`).

`<footer>`
Section or page footer.

`<form>`
Input form.

`<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
Heading levels 1-6.

`<header>`
Section or page header.

`<hgroup>`
Groups header information.

`<hr>`
Horizontal rule (dividing line).

`<li>`

https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

Basic Document Structure Tags

- `<h1>, <h2>, <h3>,<h4>,<h5>,<h6>` - Section headings where h1 is the highest and h6 is the lowest.
- `<p>` - Paragraph element. You will put most of your text content inside one of these
- `<address>` - For expressing the contact info of a person or organization.
- `<main>` - The primary container within the <body> element for document contents.
- `<article>` - Used to contain all of the independent content of the article.
- `<section>` - Used to either group different articles into different purposes or subjects, or to define the different sections of a single article.
- `<aside>` - For document content that is related, but not directly part of the content. LIke Footnotes.
- `<header>` - For content that belongs at the top of the document. Like navigation
- `<footer>` - For content that belongs at the bottom of the document. LIke author info.

See: https://developer.mozilla.org/en-US/docs/Web/HTML


<header></header>
<nav></nav>
<section id="sidebar">
</section>
Subscribe to RSS
Subscribe via Email
<section id="content"></section>
Story Title
Short description of story
Story Title
Short description of story
Story Title
Short description of story
<aside></aside>
Upcoming Events
Event A
Event B
Event C
<footer></footer>

Structuring Text

- - Indicate the importance of the text.
- <b> - Older HTML uses this to indicate bold
- - Indicate emphasis of the text.
- <i> - Older HTML uses this to indicate italics
- <pre> - Preformatted text that will be displayed exactly as written
- <code> - Used to indicate fragments of computer code.
- <cite> - Describe a reference to a cited work. Title, author or URL of work
- <blockquote>, <q> - Used for adding extended or short quotations.
- <abbr> - Use this to indicate abbreviations or acronyms
- <dfn> - Indicate that a term is being defined
- <sup>,<sub> - Superscript and subscript for

Example

```
<body>
        <h1>My Short Story</h1>
        <h2>By Jane Doe</h2>
        <Section>
        <header><h1>My Short Story</h1></header>
        <p>Once upon a time there were three bears and they
    were <em>super</em> hungry.</p>
        <p><strong>Mama Bear</strong> says: <q>I wish I had
    some honey, I <i>love</i> honey.</p>
        <footer><p>The End</p></footer>
        </section>
</body>
```

https://codepen.io/tedmonds/pen/GRKQOWZ

Linking with the Anchor element

- `<a>` - The basic element used for creating hyperlinks. The real action is in the attributes.
- `href` - This attribute is what contains the URL

- Anchor tags are used to create links
- The contents of an anchor tag is the link text
- The *href* attribute contains the URL for the link
- There are 3 kinds of href values
  - Absolute URLs - starts with a /
  - Relative URLs - no slash, relative to the current page
  - In-page linkings - using #id

Linking with the Anchor element

Absolute URL Examples:

```
<a href="/about/">About Me</a>
<a href="https://infsci2560-2023-activity2.glitch.me/">Activity 2</a>
```

Relative URL example:

```
<a href="week/2">Week 2 Page</a>
```

In-page Link:

```
<a href="#assignments">Assignments</a>
```

Adding Images

- <img/>
- *src* - Attribute with URL
- *alt* - Attribute with text description
- *width, height* - Set dimensions. Not recommended, but still often used

Examples:

Bad Alt Text

```
<img alt="image" src="penguin.jpg">
```

Good Alt Text

```
<img alt="A Rockhopper Penguin standing on
a beach." src="penguin.jpg">
```

- The IMG element is used to add images to your HTML document.
- This is a terminating element so it doesn't have a </img> because it has no content.
- Doesn't specify an image format, that is determined by the browser
- Most comment image types are
  - JPG
  - GIF
  - PNG
  - SVG
- The *alt* attribute is very important. Includes an alternative text description of the image.

Lists

- - Ordered list (1, 2, 3, ...)
- - Unordered list (*, *, *)
- - list Item
- <dl> - Description List
- <dt> - Definition/Description Term
- <dd> - Definition/Description details

```
<ol>
        <li>Item 1</li>
        <li>Item 2</li>
</ol>
<dl>
        <dt>Turkey</dt>
        <dd>A delicious, but
aggressive bird.</dd>
</dl>
```

- Use these elements to add lists to your document.
- If you use an ordered list, the browser will automatically add numbers to your list items.
- Unordered lists will use bullet points
- You can also create nested lists by creating a new list type inside of a list item.

Tables

- `<table>` - Container element for the table
- `<thead>` - Defines the header row
- `<tbody>` - Container for table
- `<tfoot>` - Defines a set of rows summarizing the columns of the table
- `<tr>` - Defines a new row
- `<th>` - Defines a cell as a header for a group of cells
- `<td>` - Defines a cell within a row

Let's Play: https://codepen.io/tedmonds/pen/RwbQjyR

Special Characters in HTML

- `"` - Quotes (“)
- `'` - Apostrophe (‘)
- `&` - Ampersand (&)
- `>` - Greater than (>)
- `<` - Less than (<)
- ` ` - Non Breaking Space

```
<p>In HTML, you define a paragraph
using the <p> element.</p>
```

```
<p>In HTML, you define a paragraph
using the &lt;p&gt; element.</p>
```

- <, >, “, ‘, and & are all special characters. They are HTML markup
- What if you want to include a less than or greater than sign in your document?
- We use entity references to create *literal* representations of these characters in our rendered HTML documents

About Whitespace and HTML

- HTML parsers ignore whitespace. If you have more than one space, it will be ignored.
- These two paragraphs will look the same:

```
<p>Dogs are silly.</p>
<p>Dogs        are
         silly.</p>
```

- This means you can put as much whitespace in your HTML code as you want.
- Why? To make it readable!

Comments

```
<!--- Comment text -->
```

Example:

```
<p>I'm not inside a comment</p>

<!-- <p>I am!</p> -->
```

- Comments are sections of your HTML document that are ignored by the browser
- They can't be seen in the rendered document and are invisible to the user (unless they "view source")
- Comments can span multiple lines
- It is always good to add comments to you computational document because they help you remember your thinking.

"Comments are love notes to yourself in the future." - Smart Person

Read More HTML on MDN

- There are TONS and TONS of HTML tags, it will take time to learn them all
- While many people have strong opinions, there is no one right way to structure an HTML document
  - But there are some specifically enforced structures
- Over time you will see there are some tags you use ALL the time.
  - P, div, strong, em, ul

- **Introduction to HTML** - Provides what we covered today and a whole lot more. Read this to learn about how to make semantically element HTML documents.
- **Multimedia and embedding** - One of the most powerful features of HTML is the media support in browsers. Read this to learn how to add media.
- **HTML Tables** - We only scratched the surface of HTML tables. There is so much more to talk about.
- **HTML Forms** - We didn't talk about forms at all today, but we will in the future. Get a jump start here.

Break

Cascading Style Sheets (CSS)

CSS - Styling the web

- HTML defines the *structure*
- CSS defines how it looks
- Here is an example

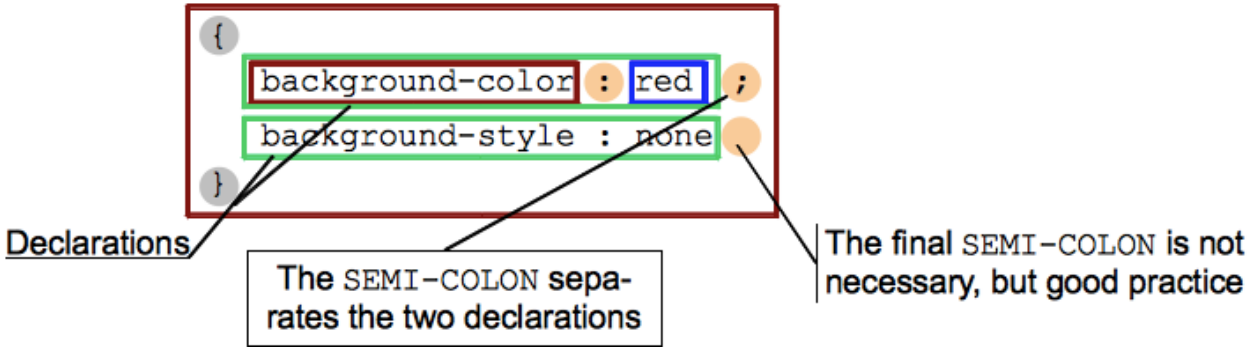

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

```css
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```

CSS concepts

- CSS is composed of two things:
  - *Properties*: Human-readable identifiers of stylistic features (font, color, background image) you want to change
  - *Values*: The specification of how you want to change the property of the stylistic feature (what font, what color, which background image)
- Together these form a *CSS declaration*
- You write CSS by writing a series of CSS declarations

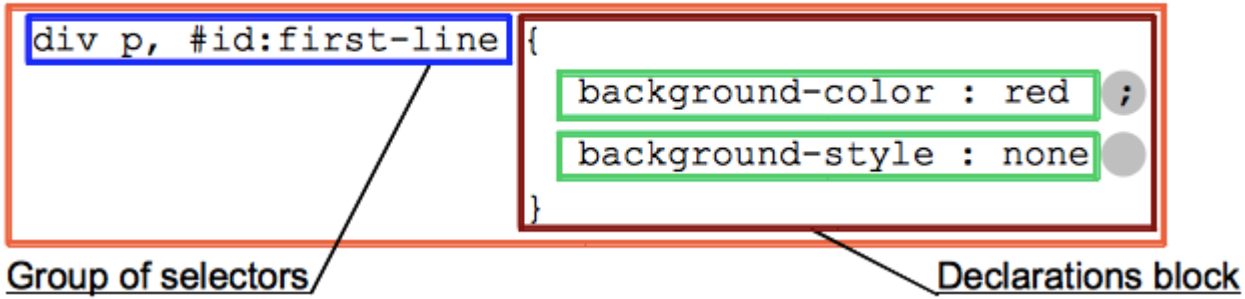
A CSS declarations block:



Selectors

- So if you use declarations to declare styles, how do we know when and where to apply them?
- We use *selectors* to select which HTML elements to apply the declared declarations ;)

A CSS ruleset (or rule):



Simple Selectors

```
/* All p elements are red */
p {
  color: red;
}
/* All div elements are blue */
div {
  color: blue;
}
/* The elements with the attribute
class="first" is bolded */
.first {
  font-weight: bold;
}
/* Element with attribute id=”rude”
THERE CAN BE ONLY ONE */
#rude {
  font-family: monospace;
  text-transform: uppercase;
}
```

- The most basic CSS Selectors are *element selectors*, *class selectors*, and *id selectors*
- The element selectors will style all HTML elements that match the named element in the selector (<p> or
- The class selector will style all elements whose *class* attribute equals the value after the period, . (...)
- The id select will style the element whose *id* attribute equals the value after the pound sign, # (<p id=”rude”>...)
  - HTML id attribute values should only identify a single element in the document

Combination Selectors

Name	Syntax	Selects
Selector list	`A, B`	Any element matching A and/or B (see Groups of selectors on one rule, below - **Group of Selectors** is not considered to be a combinator).
Descendant combinator	`A B`	Any element matching B that is a *descendant* of an element matching A (that is, a child, or a child of a child, etc.). the combinator is one or more spaces or dual greater than signs.
Child combinator	`A > B`	Any element matching B that is a *direct child* of an element matching A.
Adjacent sibling combinator	`A + B`	Any element matching B that is the next *sibling* of an element matching A (that is, the next child of the same parent).
General sibling combinator	`A ~ B`	Any element matching B that is one of the next *siblings* of an element matching A (that is, one of the next children of the same parent).

Styling Text

- Font Color - Use color names (or hex values) to specify the color of the text contained in the selected element.

```
p {
  color: red;
}
```

- Font Family - You can specify a series of fonts of your text. But make sure it is installed on the computer!
  - Use a comma separated list of increasingly generic fonts to ensure graceful degradation

```
p {
  font-family: Trebuchet MS, arial, sans-serif;
}
```

- Font Size - There are three units of measure in web design. Pixels (px), Points(pt) or em's (em or rem)
- Pixels and Points are absolute, *ems* and *rems* are relative to the font-size of the current element (16 pixels by default)

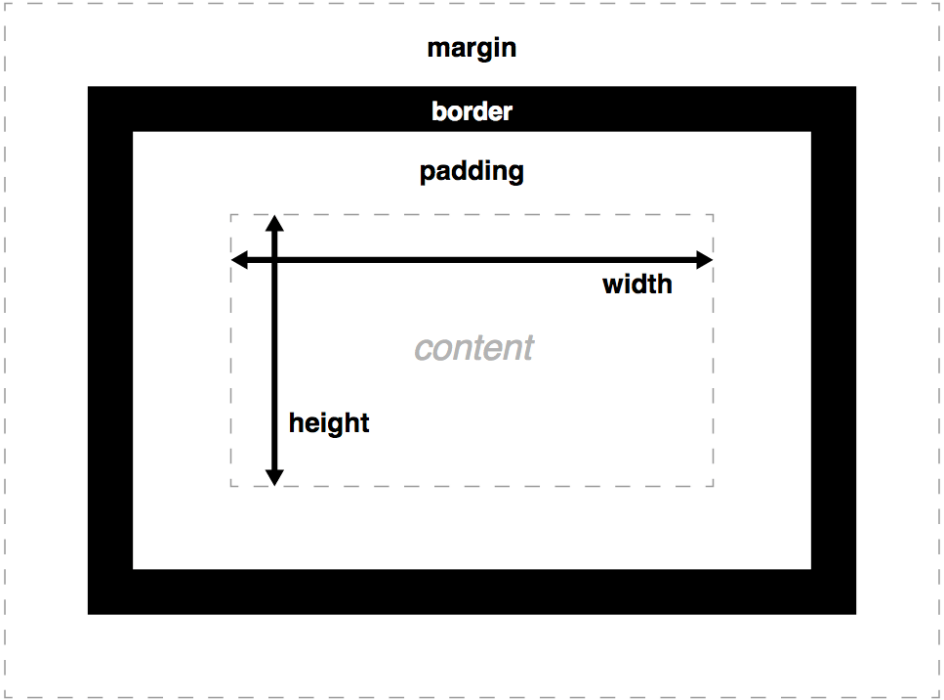
```
html {
  font-size: 10px;
}

h1 {
  font-size: 2.6pt;
}

p {
  font-size: 1.4rem;
  color: red;
  font-family: Helvetica, Arial, sans-serif;
}
```

Web Safe Fonts

Name	Generic type	Notes
Arial	sans-serif	It's often considered best practice to also add *Helvetica* as a preferred alternative to *Arial* as, although their font faces are almost identical, *Helvetica* is considered to have a nicer shape, even if *Arial* is more broadly available.
Courier New	monospace	Some OSes have an alternative (possibly older) version of the *Courier New* font called *Courier*. It's considered best practice to use both with *Courier New* as the preferred alternative.
Georgia	serif	
Times New Roman	serif	Some OSes have an alternative (possibly older) version of the *Times New Roman* font called *Times*. It's considered best practice to use both with *Times New Roman* as the preferred alternative.
Trebuchet MS	sans-serif	You should be careful with using this font — it isn't widely available on mobile OSes.
Verdana	sans-serif	

The Box Model

- Every element in the document has an invisible (or sometimes visible) box drawn around it
- You can use a set of CSS properties (see right) to change the default values of the boxes
- There are two kinds of boxes, block box and inline box
- Block boxes take up an entire line and stack on top of each other
  - Think about <p> or <li> make new lines
- Inline boxes flow around other boxes
  - Think how elements



Styling Lists

```
ul {
  padding-left: 2rem;
  list-style-type: none;
}

ul li {
  padding-left: 2rem;
  background-image: url(star.svg);
  background-position: 0 0;
  background-size: 1.6rem 1.6rem;
  background-repeat: no-repeat;
}
```

1. Line Item #1
2. Line Item #2
3. Line Item #3
4. Line Item #4
5. Line Item #5

Style Links

- **Link (unvisited)**: The default state that a link resides in, when it isn't in any other state. This can be specifically styled using the `:link` pseudo class.
- **Visited**: A link when it has already been visited (exists in the browser's history), styled using the `:visited` pseudo class.
- **Hover**: A link when it is being hovered over by a user's mouse pointer, styled using the `:hover` pseudo class.
- **Focus**: A link when it has been focused (for example moved to by a keyboard user using the `Tab` key or similar, or programmatically focused using `HTMLElement.focus()`) — this is styled using the `:focus` pseudo class.
- **Active**: A link when it is being activated (e.g. clicked on), styled using the `:active` pseudo class.

```
a {
  outline: none;
  text-decoration: none;
  padding: 2px 1px 0;
}
a:link {
  color: #265301;
}
a:visited {
  color: #437A16;
}
a:focus {
  border-bottom: 1px solid;
  background: #BAE498;
}
a:hover {
  border-bottom: 1px solid;
  background: #CDFEAA;
}
```

Read More CSS on MDN

- There is so, so much more to say about CSS
- It will take a long time and lots of practice before CSS makes sense
- Spend time reading the MDN Documentation, looking at examples, and playing around with different styles.

- **Introduction to CSS** - Covers much of what we talked about, but goes into more depth and provides more details
- **Styling text** - More about how to make your text beautiful!
- **Styling boxes** - Understanding the Box model is *crucial* for understanding how to fully use CSS effectively
- **CSS layout** - Once upon a time people used <tables> to layout their HTML documents. DON'T DO THAT! Read this module instead!
- **Use CSS to solve common problems** - Pointers to how to use useful things within the CSS documentation.

Don’t forget to do the quiz!

There is a quiz on the reading each week.

You should do it!

Activity Two - Playing around with HTML & CSS

1. Go to https://glitch.com/~infsci2560-2023-activity2
2. Read the *README.md* for instructions
3. Remix it
4. Follow directions to play around with HTML & CSS
5. Submit your remixed project url (the ~ one) on Canvas