CSS Basics

CSS stands for Cascading Style Sheets. CSS is responsible for describing how elements should be displayed on a page by overriding your browsers default styling.

Useful Resources

CSS Demo CSS Documentation CSS Selector Reference CSS Diner

---

1. What is CSS?
    1. What do we mean by Cascading?
    2. Inline Styling
    3. Internal Styles using the `<style>` tag
    4. External Styles using the `<link>` tag
    5. Why a separate file?
2. CSS Syntax and Selectors
    1. Simple Selectors
    2. Attribute Selectors
    3. Pseudo-class Selectors
    4. Combinators and Multiple Selectors
3. CSS Properties
4. Exercises
    1. Basic CSS Styling
    2. Travel Site
    3. Sign up form
5. Homework

What is CSS?

Let's learn what CSS (Cascading Style Sheets) are exactly.

- CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts using a set of rules.
- It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers.
- CSS is independent of HTML and can be used with any XML-based markup language.

What do we mean by Cascading?

"Cascading" in the context of CSS means that because more than one stylesheet rule could apply to a particular piece of HTML, there has to be a known way of determining which specific stylesheet rule applies to which piece of HTML.

The rule used is chosen by *cascading* down from the more general rules to the specific rule required.

The most specific rule is chosen.

Let's Get Stylin'

1. In the "Front-End" directory you created yesterday, create a subdirectory called "day2".
2. Change directories to "day2" and create the file "index.html"
3. Create a new subdirectory called "assets".
4. Change directories to "assets" and create a new subdirectory call "stylesheets".
5. Changes directories to "stylesheets" and create a new file called "styles.css"
6. Open index.html and build out a basic html structure (HINT: type "html", then hit the "tab" button).
7. Add some elements and contents: a

   # tag, 3

   ### tags , each followed by a tag.

day2

assets

index.html

stylesheets

styles.css

index.html set-up

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
  <title>My Index Page</title>
</head>
<body>
  <h1>Welcome to My Page</h1>

  <h2>Sometimes it's nice to smile!</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
non
  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

  <h2>When you can't smile, sing!</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
non
  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>

  <h2>When you can't sing, dance!!</h2>
```

```
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
non
  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>
</body>
</html>
```

Not feeling wordy? In Sublime, type **"*****lorem", **hit the** tab button **and let it do the talking for you!

Inline Styling

We can actually write style inline with our HTML tags. This is done using the *style* attribute. Although this is not best practices, it can certainly come in handy when trying to test things out. It would be wise to move any inline styling to a separate stylesheet before making it live in production.

```
<h1 style="text-align: center;">Look Here!</h1>
<p style="height: 100px; color: red;">This is inline CSS</p>
```

Try it out! In your index.html, add some inline styling to your

tag and a tag.

Internal Styles using the `<style>` tag

This is really a compromise between inline styles and a separate style sheet. You can use a style tag in the `<head>` or `<body>` tag. Keeping it in the head tag is best practices without a solid use case.

Internal Style Example

```
<head>
  <style>
    h1 {
        height: 200px;
        background-color: black;
        color: white;
    }
  </style>
</head>
```

Internal Styling

In your index.html file, add an internal rule to style your

tags.

```
<head>
  <style>
    h2 {
      height: 200px;
      background-color: gray;
      color: white;
    }
  </style>
</head>
```

External Styles using the `<link>` tag

The `<link>` tag facilitates making your external style sheet available to your html file. The `<link>` tag should be placed in the `<head>` tag of your site. You should include your style sheets in order from generic to specific.

Basic anatomy of the link tag `<link>` There sevaral attributes for a `<link>` tag. Lets go over a few of the common ones.

- **type** - The common use of this attribute is to define the type of style sheet linked and the most common current value is text/css, which indicates a Cascading Style Sheet format.
- **rel** - This lets the browser know what type of file we are linking. In this case it will be stylesheet. Further Reading
- **href** - This attribute specifies the URL of the linked resource.

Further Reading on the `<link>` *tag*

External Styling

Using the tag in the "head" of your index.html file, let's link our html document with our "style.css" stylesheet.

***Remove all other styling from the document!**

```
<head>
  <!-- reference to your CSS page -->
  <link type="text/css" rel="stylesheet"
href="assets/stylesheets/styles.css" />
</head>
```

Why a separate file?

Although we can write CSS in our HTML pages directly via the `<style>` tag and using inline styles, it is best practices to keep you CSS in its own file with the extension **.css**.

Benefits of a separate CSS file

- We can apply the same styling to multiple HTML pages
- Keeps our site more organized. We know where to find our CSS quickly.
- Keeps you from writing the same code over and over again.

Discussion Topics *Re-enforce Separation of Concerns and DRY (Don't Repeat Yourself) concepts. Show some different browsers default styling*

CSS Syntax and Selectors

When writing CSS you are really creating rules. Each rule consists of a selector and a declaration block. The declaration block is is wrapped in curly braces `{}` . Each declaration and its value are separated by a colon (**:**) and end with a semi-colon (**;**)

CSS Rule Example

```
h1 {
  height: 200px;
  background-color: black;
  color: white;
}
```

- **h1** - represents the selector
- **height, background-color, and color** - represents the declarations

Simple Selectors

Simple selectors directly match one or more elements of a document, based on the type of element (or its class or id). Tid). These are the selectors you will use most often.

Further Reading on Simple Selectors with activities

Simple Selector Examples

```
/* targeting an element by it's name */
h1 {
  ...
}

/* targeting an element by a given class (class="myClass") */
.myClass {
  ...
}

/* targeting an element by a given id (id="myId") */
#myId {
  ...
}
```

```
/* The Universal Selector. Select all the things */
* {
  ...
}
```

Simple Selectors

In your "styles.css" document, **create a rule** that makes all

tags have text that is the color blue, with a beige background.

```
h1 {

  /* Your rule goes here!*/

}
```

In your "styles.css" document, **create a rule** that makes all

tags have text that is the color blue, with a beige background.

Don't forget the semi-colons at the end of each statement!

Simple Selectors

```
h1 {
  color: blue;
  background: beige;
}
```

Challenge!

Give your

tag text a larger font and different color.

Attribute Selectors

Attribute selectors are a special kind of selector that will match elements based on their attributes and attribute values.

Further Reading on Attribute Selectors with activities

Common Attribute Selector Examples

```
```

```
/* All elements with the attribute "data-beer"
 are bolded */
[data-beer] {
  font-weight: bold;
}

/* All elements with the attribute "data-beer"
   with the exact value "america" are given a blue
   background color */
[data-beer="american"] {
  background-color: blue;
  color: white;
}

/* All elements with the attribute "data-beer-profile",
   containing the value "hoppy", even among others,
   are given a red text color */
[data-beer-profile~="hoppy"] {
  color: red;
}
```

Discussion Topics *Talk about substring value arrtibute selectors and encourage students to play with the different selectors on their own.*

Pseudo-class Selectors

CSS pseudo-class is a keyword that represents the state something may be in. It is added to a selector separated by a colon (**:**). It allows you to style selected elements only when they are in certain state.

Further Reading on Pseudo-class Selectors with activities

Common Attribute Selector Examples

```
  /* Change what a link should look like when hovered over with the mouse
*/
  a:hover {
    color: darkred;
    text-decoration: none;
  }

  /* Change what a link should look like when its been visited */
  a:visited {
    color: blue;
    text-decoration: none;
  }
```

Discussion Topics *Talk about Pseudo-element selectors and encourage students to play with the different selectors on their own.*

Combinators and Multiple Selectors

This can get complicated pretty fast. Let's touch on the basics for now. combinators and multiple selectors save you time by allowing you to style multiple elements at once. They also let you get very specific on how they are targeted.

Further Reading on Combinators and Multiple Selectors with activities

Simple Combinators and Multiple Selectors examples

```
  /* target all <p> tags within a <div> and make the text red
  div p {
    color: red;
  }

  /* target all <div> and <p> tags on the page and make their text blue */
  div, p {
    color: blue;
  }
```

CSS Properties

There a ton of CSS properties. Please refer to links below for a deep dive into some of the properties and what they control.

Common Properties##

Common Property Reference

- **background** - Control all things related to the background including but not limited to color and image
  - **background-color:** - specifically control color of background
  - **background-image**: - set an image as background
- **color** - Controls the color of the text within an element. CSS Color Reference
  - Most standard colors (primary & secondary colors, white, black, grey) can be called by name. More specialized colors can be called using a hex code (ex.: black = #000000, white = #ffffff)

    https://www.w3schools.com/cssref/css_colors.asp

CSS Properties##

- **font-weight** - controls the boldness of the font. Common values are normal and bold
- **font-size** - specifies the size of the font. Usually in pixes or em units. CSS Font Size Reference
- **font-family** - specifies the font to use. CSS Web Safe Font Reference

CSS Properties##

- **height** - controls the height of the content area of an element
- **width** - controls the width of the content area of an element

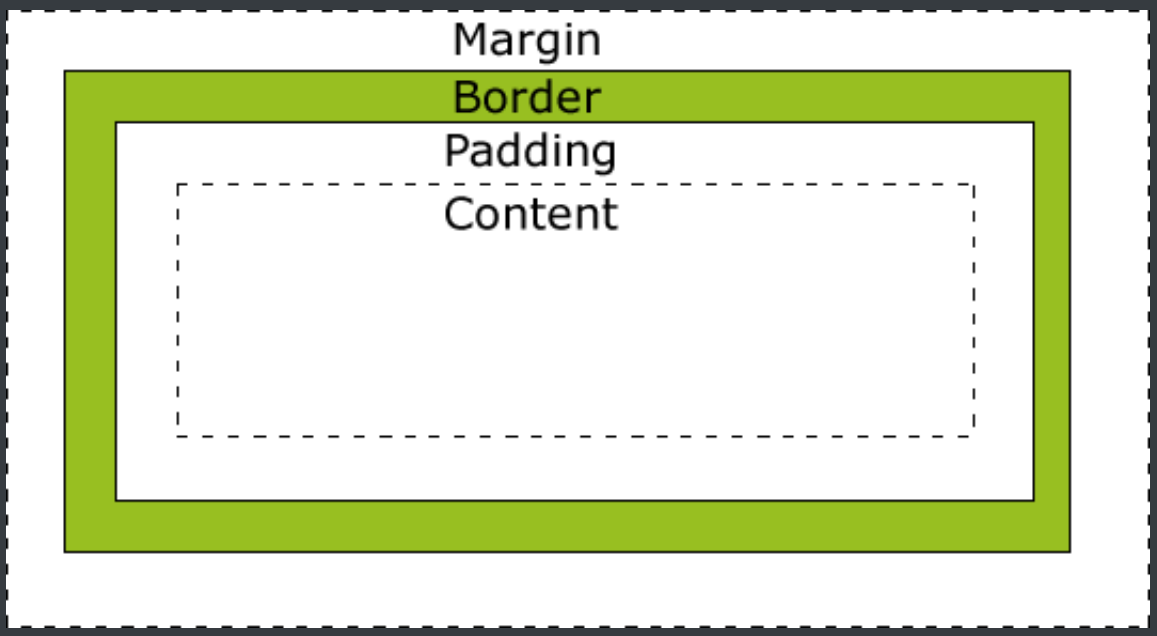
Like font sizes, *height* & *width* are commonly set in *pixel* or *em* units.

CSS Properties##

The Box Model

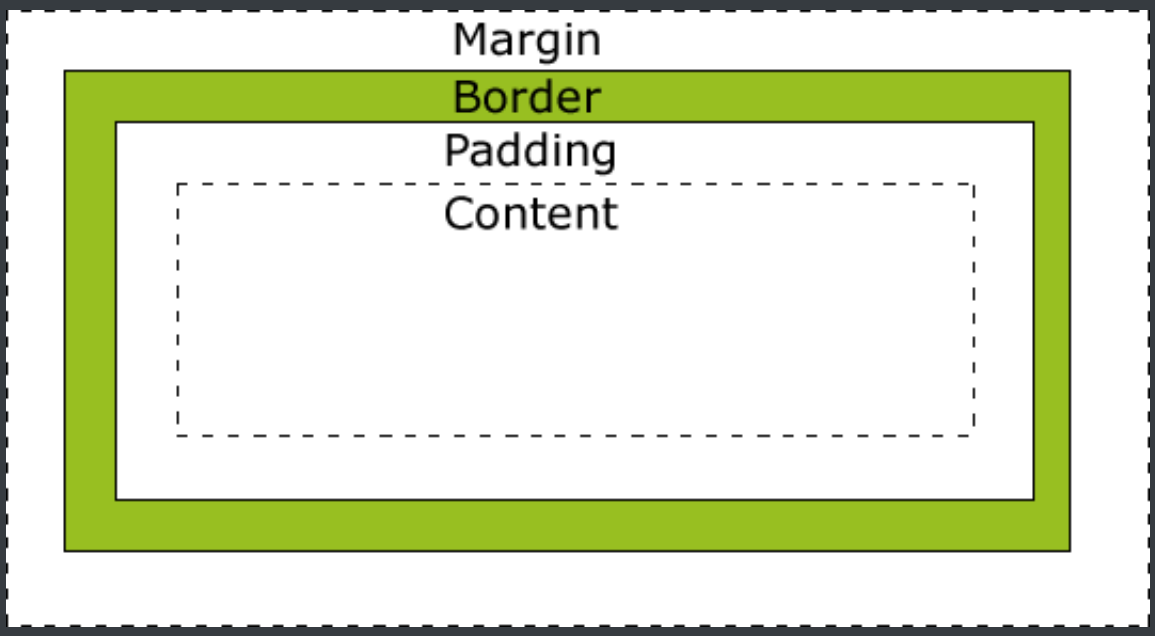
HTML elements are considered boxes. The term "box model" is used when refer to the design and layout of the elements in our HTML document.

The **box model** is a box that wraps around all HTML elements. It consists of margins, borders, and padding, all surrounding our content.



CSS Properties

- padding:
  - sets the padding space on all sides of an element between our content and the border.
  - CSS Padding Reference
- margin:
  - sets the margin on all sides of an element between our element and the next element.
  - CSS margin Reference
- border:
  - sets the border width on all sides of an element.
  - CSS border Reference



<Image Source: http://www.expression-web-tips.com/css-box-model/w3schools-box-model/>

CSS Properties

display and float properties

As we just learned, all elements on a web page follow the box model. We are going to learn about two properties that affect how our content behaves within the box:

- The **display** property determines how the box behaves.
- The **float** property specifies that an element should be placed along the left or right side of its container, allowing text and inline elements to wrap around it.** **

CSS Properties

display Property

CSS display Property

display: none

```
<h2>display: none:</h2>
<div>
This is an example of using the "none" value.
<p class="ex1">DISPLAY NONE</p>
The "none" value completely removes the element.
</div>
```

```
p {
    color: red;
}

.none {
    display: none;
}
```

This is an example of using the "none" value. The "none" value completely removes the element.

CSS Properties

display Property

display: inline

```
<div>
This is an example of using the "inline"
value. <p class="inline">DISPLAY INLINE</p> This is
the default value for the "display" property. It
displays an element as an inline element. Any height
and width properties will have no effect on the element.
</div>
```

```
<div>
This is an example of using the "inline"
value. <p class="inline">DISPLAY INLINE</p> This is
the default value for the "display" property. It
displays an element as an inline element. Any height
and width properties will have no effect on the element.
</div>
```

```
p {
    color: red;
}

.none {
    display: inline;
}

p {
    color: red;
}

.none {
    display: inline;
}
```

This is an example of using the "inline" value. DISPLAY INLINE This is the default value for the "display" property. It displays an element as an inline element. Any height and width properties will have no effect on the element.

CSS Properties

display Property

display: block

```
<div>
This is an example of using the
"block" value. <p class="block">DISPLAY BLOCK</p> It
displays an element as a block element. It starts
on a new line, and takes up the whole width.
</div>
```


```
<div>
This is an example of using the
"block" value. <p class="block">DISPLAY BLOCK</p> It
displays an element as a block element. It starts
on a new line, and takes up the whole width.
</div>
```


```
p {
    color: red;
}

.none {
    display: block;
}

p {
    color: red;
}

.none {
    display: block;
}
```

This is an example of using the "block" value.

DISPLAY BLOCK

It displays an element as a block element. It starts on a new line, and takes up the whole width.

CSS Properties

display Property

display: inline-block

```
<div>
This is an example of using the "inline-block" value.
<p class="inline-block">DISPLAY INLINE-BLOCK</p> This
value displays an element as an inline-level block container.
The element itself is formatted as an inline element, but
you can apply height and width values
</div>
```


```
<div>
This is an example of using the "inline-block" value.
<p class="inline-block">DISPLAY INLINE-BLOCK</p> This
value displays an element as an inline-level block container.
The element itself is formatted as an inline element, but
you can apply height and width values
</div>
```


```
p {
    color: red;
}

.none {
    display: inline-block;
}

p {
    color: red;
}

.none {
    display: inline-block;
}
```

This is an example of using the "inline-block" value. DISPLAY INLINE-BLOCK This value

displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values

CSS Properties

The text for this element will go left, while the image will float to the right. This position set by using the "float: right;". Other options for the float property are left, right, inherit, and initial.

float: right;

The text for this element will go left, while the image will float to the right. this position is set by using "float:right;". Other options for the float property are left, right, initial, and inherit.

CSS Properties

The text for this element will go left, while the image will float to the right. This position set by using the "float: right;". Other options for the float property are left, right, inherit, and initial.

"**float: initial**" - sets the element to its default position.

"**float: inherit**" - Inherits its value from its parent element.

CSS Properties

- positioning:
  - specifies the type of positioning method used for an element. Options are:
  - **static** - default; positioned according to normal the flow of the page
  - **relative** - Setting the top, right, bottom, and left properties will cause it to be adjusted away from its default position
  - **fixed** - position remains fixed, even when the page is scrolled

- **absolute** - positioned relative to the nearest positioned ancestor
- **sticky** - toggles between **relative** and fixed**, depending on the scroll position. As the page is scrolled, the position is relative until a specified offset of at least one property of top, right, bottom or left is reached, at which point the position becomes fixed.

CSS positioning reference

CSS Exercise

in index.html, add the following :

- Add an anchor tag around the first
  tag. Set the "href" attribute to "**https://https://www.techtalentsouth.com/**".
- Create a div with a class "sticky" above the second
  with the text "Sticky little divs!"
- Add an anchor tag around the second
  tag and set it's href to "https://www.youtube.com/watch?v=d-diB65scQU"**.**
- An anchor tag around the third
  tag and set it's href to "https://www.youtube.com/watch?v=Xj11GuedXDc"
- Add two
  tags to the bottom of your document.
- Give the first
  tag a **class = "square"** and the second
  tag a **class = "circle"**.
- Add a class to your third
  tag, set it equal to **"big-text"**.
- Add a
  tag that describes your favorite city. Within the
  tag, add a nested

tag with a link to picture of your favorite city. Set the width to 95px and height to 75px.

CSS Exercise con't##

In styles.css, add the following rules:

- Set the background color of your entire document to **"#efefe1"**.
- Targets all headers and changes their font-family to **"serif"**.
- Target all elements with an **"href"** attribute and make the color of their text **"orange".**
- create a rule for your **"sticky"** div and give it the following attributes - set the position to "sticky", set the top property to "0", give it a background color of blue and a solid, blue border 2px thick.
- Prevent the anchor tags to from changing colors after being clicked.
- Align all headers to the center of the document.
- Set the font-size of your elements with a class **"big-text"** to 50px.
- Set the div with the class **"square"** to display a red square on our document.
- Set the div with the class** "circle"** to display a blue circle on our document.
- Set your the image of your favorite city to float right of the your text.

PRO TIP* *https://developer.mozilla.org/en-US/docs/Web/CSS/YOUR_PROPERTY_HERE - Replacing YOUR_PROPERTY_HERE with something like 'color', 'background', or any other property you can think of should bring you to documentation on that specific property Discuss Custom Fonts with Google Fonts*

Homework

Easy

Instructions

1. Build out the pie chart seen in the image using only html and css (no images!)
2. Set the background color of each section to the color it is labeled

---

Medium

images for challenge cart-icon.png search-icon.png

Instructions

1. Build out the masthead seen in the image masthead.png
2. Use the 'cart-icon.png' and 'search-icon.png' images; everything else should just be html and css