Tools

The Basics:

Windows - Notepad
Mac - Text Edit

Downloads:

Google Chrome **Browser** (Free) - www.google.com/chrome/
Adobe Brackets (Free) - www.brackets.io

Our work over the next 6 weeks will be created and saved **locally** to the computer. No data will we leaving the computer to any **remote** computer.

If at some time in the future, you can contact a **web hosting service** to upload your files. This requires a month to month or yearly paid subscription for access to your account to **FTP** files to the **server**.

We will be learning the **HTML** and **CSS** markup languages. For more information, head to:
https://developer.mozilla.org/en-US/docs/Web

Homework:

browser:

local:

remote:

web hosting service:

FTP:

server:

HTML:

CSS:

The Basic HTML Structure:

```
<html>
<head>
   <title>Document</title>
</head>
<body>
</body>
</html>
```

<html> - Everything between opening and closing HTML tags is HTML code
<head> - This contains information about the page, which usually has the <title> within the head tag
<title> - The contents of the title tag are shown in the top of the browser in the tab for that page.
<body> - Everything between the opening and closing tags are shown in the main browser window. This is where most of your code will be placed.

(The word "tag" and "element" are both used interchangeably .
* is the opening tag, is the closing tag. The element is " and the text between ")*

Heading Tags

There are 6 levels of heading tags that are in 6 different sizes. H1 is the biggest and h6 is the smallest. They have opening and closing tags.

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Paragraph Tag

To create a paragraph, surround the text that makes up the paragraph with the ** ** tags. By default, the browser will show each paragraph on a new line with some space between it and the next paragraph like a block of text - this is known as a **block level element**.

Block Level and Inline Elements

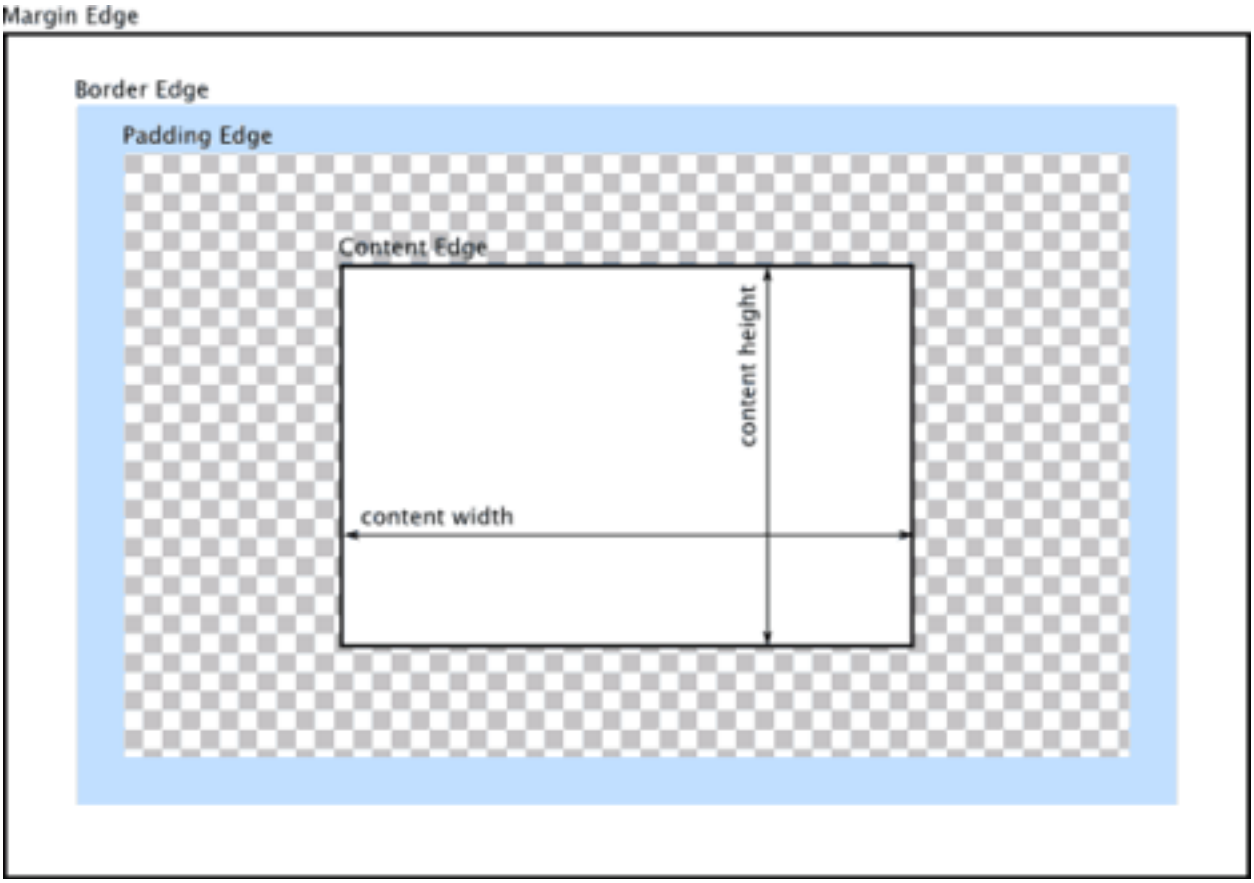
Block Level Elements

So far the h1-h6 tags and the paragraph tag the are block level elements. By default, block-level elements begin on new lines.

Inline Elements

An inline element occupies only the space bounded by the tags that define the inline element. The , and tags are inline.

The Box Model



An element is the sum of the widths of margin, padding, and border.
(There is an exception to this rule by way of CSS. The box-sizing property is used to alter the default CSS box model used to calculate widths and heights of elements.)

Read More - https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing

Sectioning Elements

The ** ** tags are for "divisions" in a page. They are often used as a container tag that can then be manipulated with CSS from classes and ids. Sectioning tags have no default styling applied to them.
Other tags that can be used: **<section>**, **<article>**, **<nav>**, **<header>**, **<footer>** including their closing tags.

Strong

Surrounding text within ** ** tags, can make the text appear bold. The **** tags can also be used.

Emphasis

Surrounding text within ** ** tags, can make text appear italic. The **** tags can also be used.

White Space

When writing your code, white space is ignored beyond one space.

```
<p>This is a line                    and another. </p>
```

is translated to the browser as:

This is a line and another.

Line Breaks

**
** or also written for older browsers **
, is used to break to a new line. Notice this a **self closing tag.

Horizontal Rules

<hr> tag is also a self closing tag that does not need an extra closing tag. It is also written **<hr />** for older browsers. This adds a line as wide as its containing element. This tag is an older tag and is not used as much these days. With the advancements in CSS, borders are often used as horizontal rules.

Lists

Unordered

To start and unordered list, use the ** ** tags. Inside those tags, each item needs to be surrounded by ** ** tags. Bullets are in front of each item.

```
<ul>
    <li>First Item</li>
    <li>Second Item</li>
    <li>Third Item</li>
</ul>
```

Ordered

To start and ordered list, use the ** ** tags. Inside those tags, each item needs to be surrounded by ** ** tags. Numbers are in front of each item.

```
<ol>
    <li>First Item</li>
    <li>Second Item</li>
    <li>Third Item</li>
</ol>
```

Links

Links are created using the **<a> ** tag. In the opening tag the **href** attribute is used to designate the address to go to:

```
<a href="http://www.somewhere.com">Somewhere Page</a>
```

The text between the opening and closing tag is known as the **link text**.
For local files in your project:

```
<a href="about.html">About Page</a>
```

Email Links

Email links are similar:

```
<a href="mailto:person@email.com">Email Me</a>
```

The **mailto:** attribute is used to trigger the email application to start a new email addressed to the email in the link.

Images

Formats:

JPG - is used for photographs that have millions of colors.
PNG - is used for logos, illustrations that have flat colors which is areas that are filled with exactly the same color.
GIF - Because of its small size, it is ideal for small navigational icons and simple diagrams and illustrations. Gifs can be animated.
SVG - Images that have a markup language of XML to make shapes and colors with support for interactivity and animation. This is a new file format.

Placeholder images

```
<img src="http://lorempixel.com/400/200">
```

Change the numbers to bigger or smaller values

DOCTYPE

There have been several versions of HTML. Web pages should start with a DOCTYPE declaration to tell a browser which version of HTML the page is using. The current version is HTML5 and it's declaration looks like this:

```
<!DOCTYPE html>
```

It is used as the very first tag before the opening <html> tag.

Character Encoding

```
<meta charset="UTF-8">
```

You should always specify a character encoding on every HTML page you serve. Not specifying an encoding can lead to security vulnerabilities.

Comment Tags

You can add comments to your code and it will not be rendered by the browser. Comment tags help you define certain sections of your code and is helpful when sharing code with a team and remembering where tags start and end.

For HTML comment tags:

Opening tag: <! --
Comment goes between
Closing tag: —>

CSS

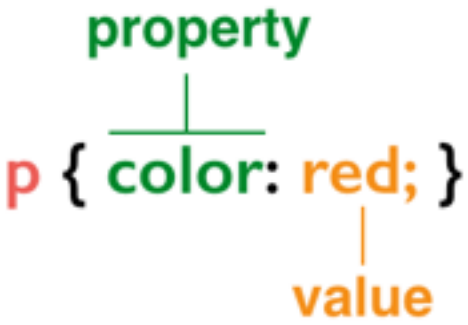
Connecting HTML and CSS

First create a new folder in you project called **css,** this to help you find exactly where the file is. Then create a new file in the css folder and name it: **style.css**
In the **<head>** section, below the **<title>** tag of your **index.html** page type:

```
<head>
<title>Website Name</title>
<link rel="stylesheet" href="style.css">
</head>
```

The **link** tag is used in an HTML document to tell the browser where to find the CSS file used to style the page. It does not need a closing tag.
rel - specifies the relationship between the HTML page and the file it is linked to.
href - specifies the path to the css file

CSS stands for Cascading Style Sheets. Like a waterfall that cascades down a down a mountain face, stylesheets cascade their styles throughout a site.
With CSS, you create rules that specify how the content of an (HTML) element should appear. Learning CSS involves learning the different properties you can use.



Selectors indicate which (HTML) element the rule applies to. The same rule can apply to more than one element if you separate the element name s with a comma:

```
h1, h2, h3 { font-family: Arial; }
```

All h1, h2 and h3 tags in your pages will all have the same font style.

Declaration indicates how the (HTML) elements should be styled.

Property indicates what you want to change. For example, color, font, width, height and border.

For declaring several rules, it is good coding practice to put each declaration on a separate line:

```
p {
font-family: Arial;
color: blue;
font-weight: bold;
}
```

For a list of all the properties and how they work:
https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

Some commonly used properties:

CSS comment Tags

:active	font-size	margin-(top,right, bottom, left)
background	font-weight	overflow
background-color	height	padding-(top,right, bottom, left)
background-image	font-family	text-align
border-(top, right, bottom, left)	font-size	text-decoration
box-shadow	font-weight	text-shadow
box-sizing	height	text-transform
color	:hover	:visited
display	letter-spacing	width
float	:link	
font-family	list-style-type	

Comments are seen if anyone looks at the source code behind the page.
In CSS, the comment tags are different from HTML:

```
/* This is a comment in CSS */
```

Typography

Changing the way the font looks comes in the form of fonts. References to fonts can come first from the users computer, or from a font repository like Google Fonts: https://www.google.com/fonts
On Google Fonts, each font listing has these buttons:

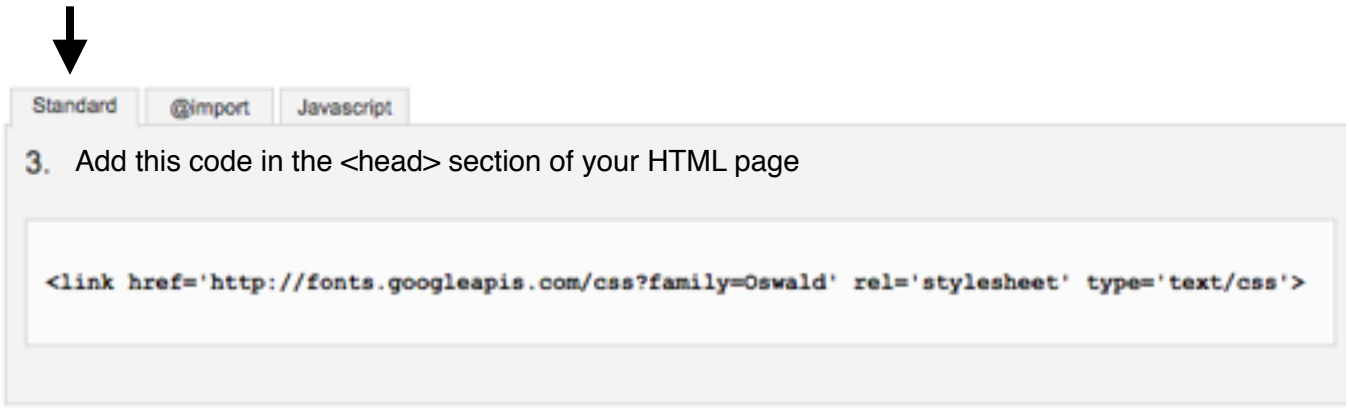
The middle button is the Quick-use button. On the next page you will see the styles to choose from. Notice the speedometer on the right. Fonts add weight or load time to your pages slowing down the time the page loads. Nobody like waiting for a site to load, so don't go crazy with fonts. Try to use common font styles like Arial and Times New Roman for body text that will load quickly from a commonly installed font.

Impact on page load time

Tip: Using many font styles can slow down your webpage, so only select the font styles that you actually need on your webpage.

The Standard Way

By clicking on the Standard tab, shows the code in a different way with HTML-like code with angle brackets. By using this version, you copy and paste into the **index.html** page right after the opening **<head>** tag.



```
<html>
<head>
Paste Code Here
<title>Website Title </title>
<meta charset="UTF-8">
  <title>Website Title</title>
  <link rel="stylesheet" href="css/style.css">
</head>
```

Why do we put the code there? When the browser renders the page it first sees there is an address to go and get a font reference. We want that to happen first then render the rest of the stylesheet, that comes after.

@import Way

Copy and paste the line of code into **style.css** as the very first line of code before anything else.

Again we put the code at the top of the stylesheet to load first, then the rest of the styles.

CSS Color

The 3 most common ways to add color in stylesheets are:

- Color Names
- Hexadecimal Values
- RGB Values

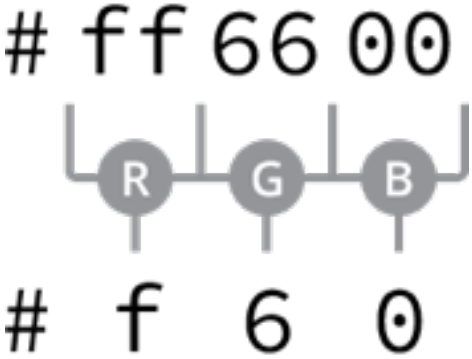
Color Names	Hex Values	RGB Values
black	#000000	rgb(0, 0, 0)
silver	#c0c0c0	rgb(192, 192, 192)
white	#ffffff	rgb(255, 255, 255)

Named Colors

There are 147 named colors. For the named list with it's Hex and RGB translation:
http://www.w3.org/TR/css3-color/#svg-color

Hexadecimal Values

There are over 16.7 million hexadecimal color values that consist of a **pound, or hash, #**, followed by a **three- or six- character figure**. The figures use the numbers **0 through 9** and the **letters a through f, upper or lower case**. These values map to the **red, green, and blue color channels**.



In **six-character** notation, the **first two** characters represent the **red** channel, the **third and fourth** characters represent the **green** channel, and the **last two characters** represent the **blue** channel.

In **three-character** notation, the **first** character represents the **red** channel, the **second** character represents the **green** channel, and the **last** character represents the **blue** channel **when each contain a repeating character**.
Adobe color: http://color.adobe.com is a great helper for HEX ad RGB values.

RGB & RGBa Colors

RGB color values are stated using the rgb() function, which stands for red, green, and blue. The function accepts three comma-separated values, each of which is an integer from 0 to 255. A value of 0 would be pure black; a value of 255 would be pure white.

RGBa

RGB color values may also include an **alpha, or transparency, channel** by using the rgba() function. The rgba() function requires a **fourth value**, which must be a **number between 0 and 1**, including decimals. A value of **0** creates a fully transparent color, meaning it would be **invisible**, and a value of **1** creates a **fully opaque** color. Any decimal **value in between 0 and 1** would create a **semi-transparent color**.

If we wanted our shade of orange to appear 50% opaque, we would use an RGBa color value of rgba(255, 102, 0, .5).

Hexadecimal color values remain the most popular as they are widely supported; though when an alpha channel for transparency is needed, RGBa color values are preferred.

Size

Pixels

Pixels have been around for quite some time and are commonly used with a handful of different properties. As an absolute unit of measurement, they don't provide too much flexibility. Pixels are, however, trustworthy and great for getting started.

The default base font size is 16 pixels which is referring to the body font size rendered by the browser. There are 96 pixels in an inch, but may vary depending on different devices.

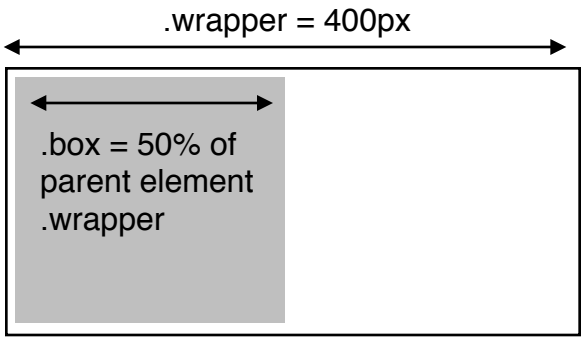
Example:

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

Percentages

This gets a little complex, with percentages. They are not fixed units of measurement, and rely on the length of another measurement. Percentages are known as “relative value”.

```
.wrapper {
   width: 400px;
}
.box {
  width: 50%;
  background: #999;
}
```



In the example above, the class .box is 50% of the total width of the class .wrapper. This is how percentages work, they must have a “parent” element to set their percentages off of.

Em

Em is another relative value and is calculated on an element’s font size. So, for example, if an element has a font size of 14 pixels and a width set to 5em, the width would equal 70 pixels (14 pixels multiplied by 5).

```
1em = 14px
2em = 28px
…
5em = 70px
```

When a font size is not explicitly stated for an element, the em unit will be relative to the font size of the closest parent element with a stated font size.

Positioning with Floats

Essentially, the float property allows us to take an element, remove it from the normal flow of a page, and position it to the left or right of its parent element. An <img> element floated to the side of a few paragraphs of text, for example, will allow the paragraphs to wrap around the image as necessary.

```
img {
  float: left;
}
```

The float property accepts a few values; the two most popular values are **left** and **right,** which allow elements to be floated to the left or right of their parent element.

The float comes with a few pitfalls. Often margin and padding property values aren't interpreted correctly, causing them to blend into the floated element; other properties can be affected, too.

The Clear Fix - both

The both value, however, will clear both left and right floats and is often the most ideal value.

Positioning with Inline-Block

we can also position content is by using the display property in conjunction with the inline-block value. The inline-block method is primarily helpful for laying out pages or for **placing elements next to one another within a line**.

Uniquely Positioning Elements

Every now and then we'll want to precisely position an element, but floats or inline-block elements won't do the trick. Floats, which remove an element from the flow of a page, often produce unwanted results as surrounding elements flow around the floated element. Inline-block elements, unless we're creating columns, can be fairly awkward to get into the proper position.

The relative value for the position property allows elements to appear within the normal flow a page, leaving space for an element as intended while not allowing other elements to flow around it; however, it also allows an element's display position to be modified with the box offset properties.

```
div {
  height: 100px;
  width: 100px;
}
.offset {
  left: 20px;
  position: relative;
  top: 20px;
}
```

Absolute Positioning

The absolute value for the position property is different from the relative value in that an element with a position value of absolute **will not appear within the normal flow of a document**, and the original space and position of the absolutely positioned element **will not be preserved**.

Additionally, absolutely positioned elements **are moved in relation to their closest relatively positioned parent element**. Should a relatively positioned parent element not exist, the absolutely positioned element **will be positioned in relation to the <body> element**.

Resources

For more information check this website:
http://learn.shayhowe.com/html-css/

Good Book:
HTML & CSS by Jon Duckett