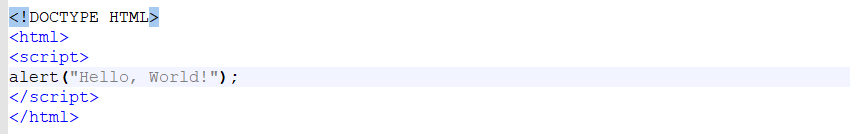
:



<!DOCTYPE HTML>

<html>

<script>

alert(“Mmmm pretzel bites”);

</script>

</html>

--------------------------------------------------------------

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Alert</h2>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

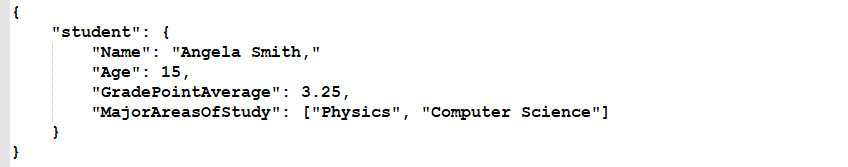
alert("I am an alert box!");

}

</script>

</body>

</html>

  
<!DOCTYPE HTML>

<html>

<body>

<script>

alert("Mmmm pretzel bites");

</script>

</body>

</html>

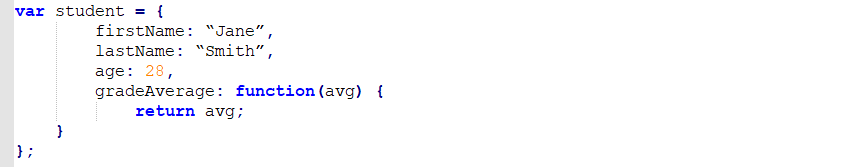


An example could look like this:



Other JavaScript code elements could call this “add” function by specifying its name and passing it two numbers. That could look like this:

https://techacademystorage.blob.core.windows.net/htmlandcss/function2.png



Here, we aren’t setting the property “gradeAverage” to a fixed number of 3.5. Instead, we are setting that property to the value returned by a set of code. Specifically, that code will take in a number (the variable “avg”) and set the value of the “gradeAverage” property to the value of that variable.

Executing that code could look like this:

https://techacademystorage.blob.core.windows.net/htmlandcss/method3.PNG

An “attribute” is a specification that assigns a property (name; characteristic) to something. Basically, an attribute is metadata (data that describes other data).

The element returned by document.getElementById has an Id attribute with a specific value assigned to it. The Id is a variable.

For an example:

https://techacademystorage.blob.core.windows.net/htmlandcss/h1.png

We have assigned the h1 element the Id “Header\_1”. To run and display (return) this element, we write:

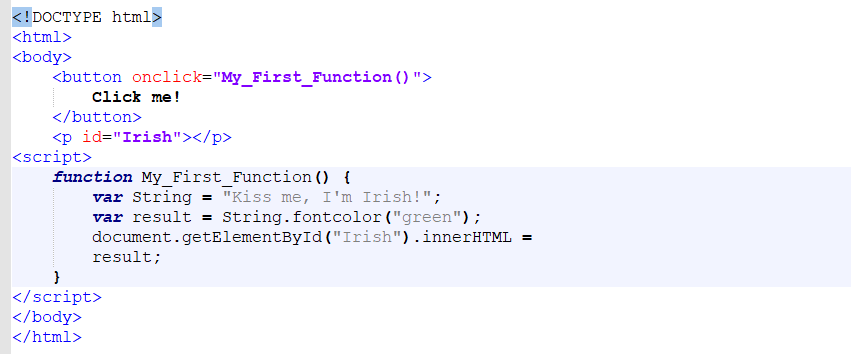
https://techacademystorage.blob.core.windows.net/htmlandcss/get_element.png

“innerHTML” gets or sets the HTML markup contained within the element.

Document.getElementById is used mainly to control or get information from an element within your code. If it can’t find the element with the specified Id, it will return “null.”

**CREATING A FUNCTION**

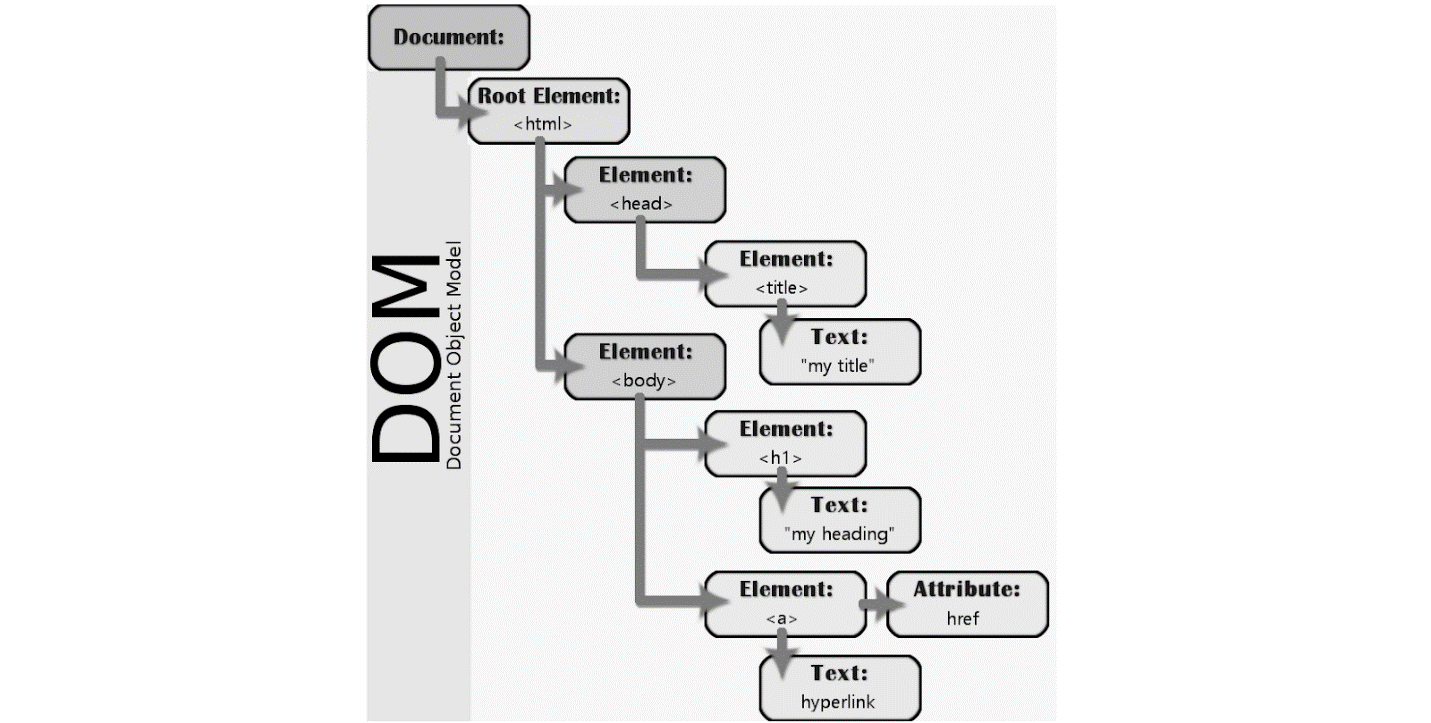
Let’s look at creating a function using the document.getElementById method:



The above code accomplished the following:

* Created a button element.
* Utilized the ID attribute and assigned the button element the value “Irish”.
* Returned the button element by calling the “Irish” value that we assigned earlier (when writing the ID attribute).

Everything inside our curly brackets { } is our function.



Suppose you wanted to find all paragraph elements in an HTML document. You might use code like this:

https://techacademystorage.blob.core.windows.net/htmlandcss/get_element2.png

The variable called “foo” would now contain a collection of all the paragraph tags in the document.

Many different programming languages can be used to work with the DOM. One of the most common is JavaScript; in fact, the code example above is actually JavaScript.

olor like:

https://techacademystorage.blob.core.windows.net/htmlandcss/color.PNG

2. Specify the color using its hex code (a way of specifying color using hexadecimal values). For example:

https://techacademystorage.blob.core.windows.net/htmlandcss/color2.PNG

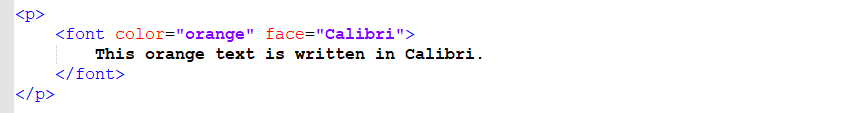
[**Here is a useful chart to find the hex code for various colors.**](https://www.rapidtables.com/web/color/html-color-codes.html)

For example: red would be written as

https://techacademystorage.blob.core.windows.net/htmlandcss/color3.PNG

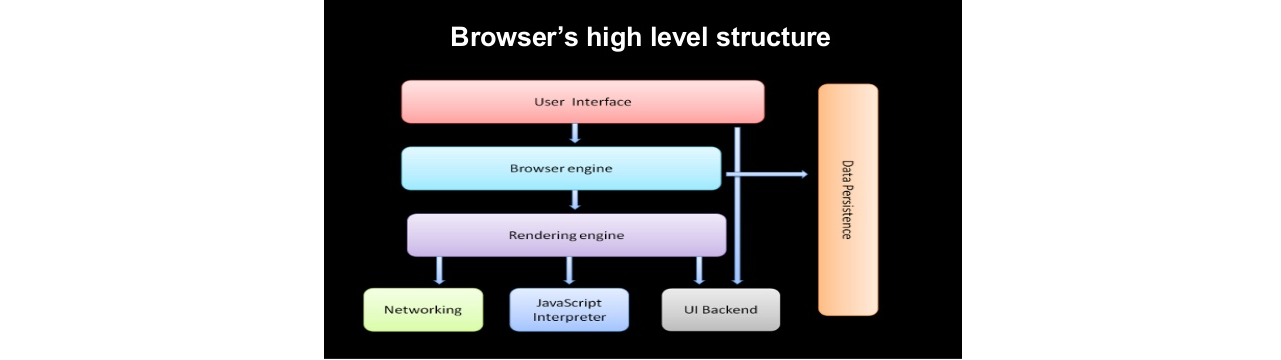
To assign a font color to an element (such as a paragraph) you write your code as follows:

https://techacademystorage.blob.core.windows.net/htmlandcss/color4.png



https://techacademystorage.blob.core.windows.net/htmlandcss/italics2.png

There are seven main components to any web browser:



1. Layout Engine: This takes input from the browser (URL bar, search box, mouse clicks, and keyboard input) and passes them to the rendering engine.

2. Rendering Engine: “Render” literally means “to cause to be; make”. A rendering engine takes HTML code and interprets it into what you see visually. For example: a <bold> tag would be interpreted by the rendering engine as a set of instructions to make the text inside the element bold.

3. User Interface: This is the visual presentation of controls in the browser, for instance the back and forward buttons.

4. JavaScript Engine: This engine takes JavaScript code, parses (analyzes and reads the code) it, executes it, and returns the results.

5. Network Layer: This is a function of the browser that happens behind the scenes and handles network functions such as encryption (concealing data by altering it into a secure format), requests, and all network settings such as HTTP.

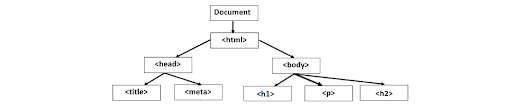
6. Storage: Browser’s must store some data which can include cached files and cookies.

7. Operating System Interface: The browser interacts with the operating system to display several elements of the page, like drop down boxes and some icons on the window (close, maximize, and minimize buttons).

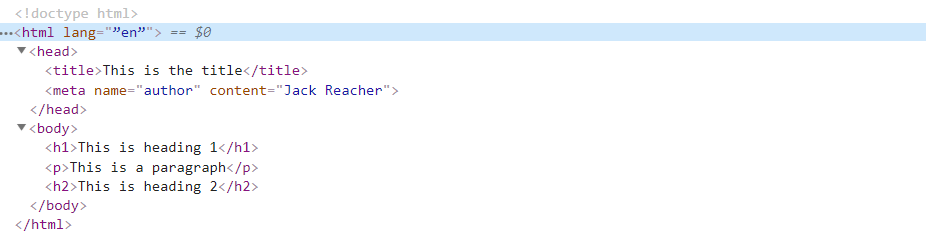
**RENDERING**

The rendering engine displays what you see on your screen. It receives HTML code and other items passed from a server and then creates the Document Object Model (DOM).

The rendering engine creates a DOM tree like this from the code received:



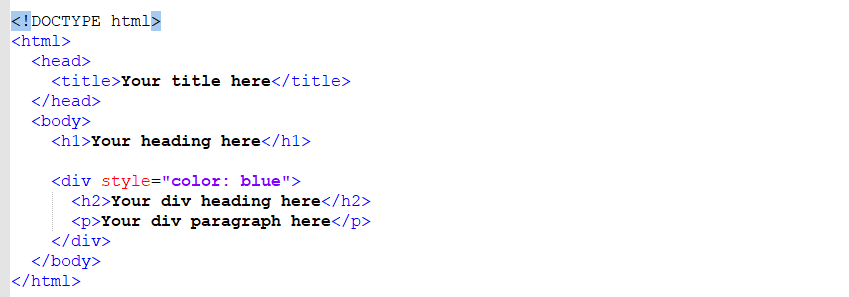
Though, technically, the DOM (render tree) would look like this:



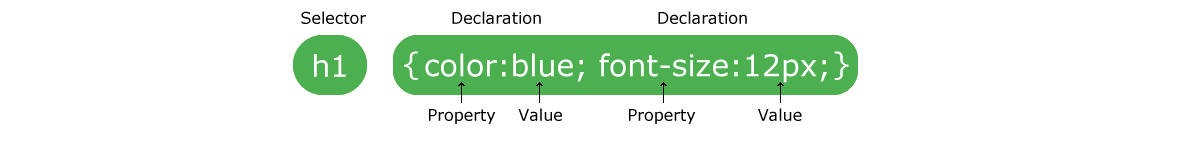
**DIV**

The <div> tags define a division or a section in an HTML document. The div element is often used as a container for other HTML elements to style them or make specific changes to the content within that particular div tag.

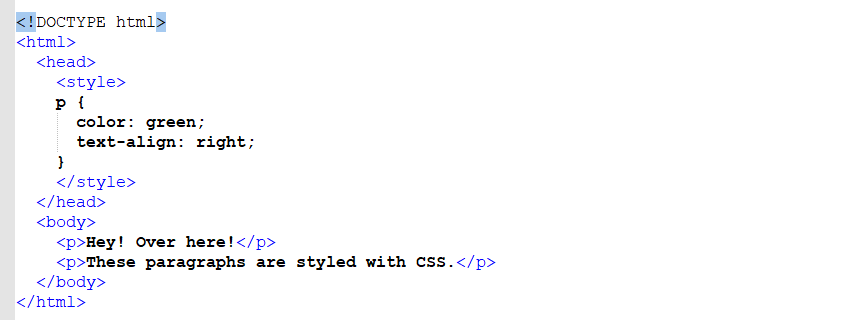
EXAMPLE:



All of the elements contained within the <div> element will be written in blue.



Review the following code:



The output of this code would be:

https://techacademystorage.blob.core.windows.net/htmlandcss/review2.PNG

**EXTERNAL STYLE SHEETS**

As we mentioned earlier, CSS can be saved in a different file than one’s HTML code. This is actually recommended. We do this in a similar way to how we did it with JavaScript:

https://techacademystorage.blob.core.windows.net/htmlandcss/stylesheet2.png

In HTML, there’s a <link> element that defines a link between a document and an external resource. It is common to use <link> tags to link to a style sheet.

“Rel” is short for “relationship”. Rel is an attribute (word used inside an element’s opening tag that controls its behavior) that specifies the relationship between two documents. For example: if you wanted to specify that you were linking to the help page on your website, you’d write:

https://techacademystorage.blob.core.windows.net/htmlandcss/rel.PNG

“Stylesheet” is a common value used following the rel attribute. The stylesheet value imports the specified stylesheet

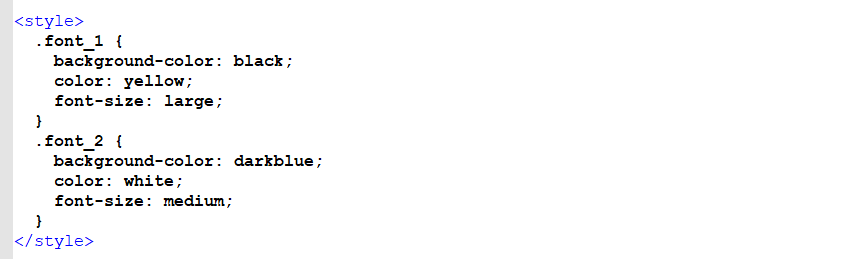
**CLASS ATTRIBUTE**

The HTML class attribute is used to define equal styles for all elements with the same class name. HTML elements with the same class attribute have the same format and style.

For example: we could utilize the class attribute if we wanted to have two different styles of paragraphs.

You assign a class name by writing .classname (a period followed by a class name of your choosing).

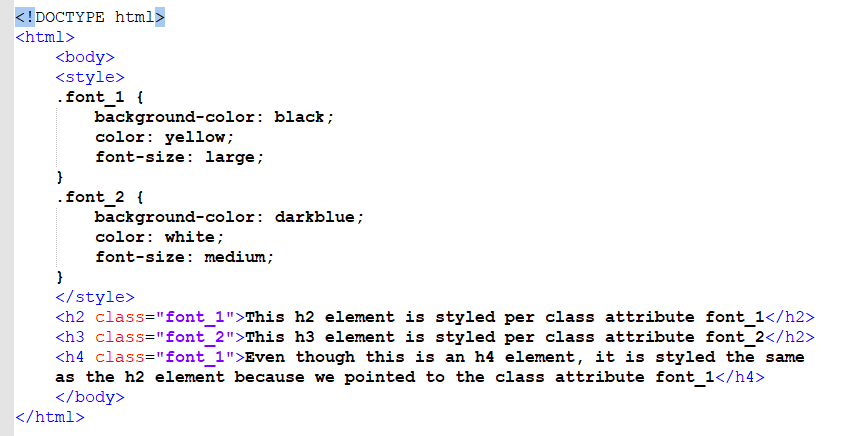
Here is an example of how to create two styles using the class attribute:



We created two class names: font\_1 and font\_2. The class attribute specified the class names.

By pointing to the class name “font\_1” or “font\_2” we can specify the styling of our font.

This is how:



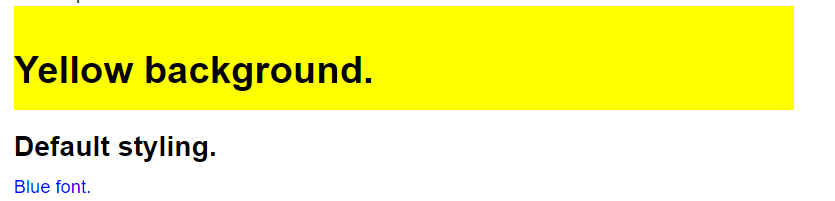
**ASSIGNMENT**

Complete the following:

1. Open the Basic\_HTML\_3.html file and add a line of text (e.g. a paragraph or heading).

2. Using CSS, set the color and alignment of the text.

Save your code and successfully run it in the browser.



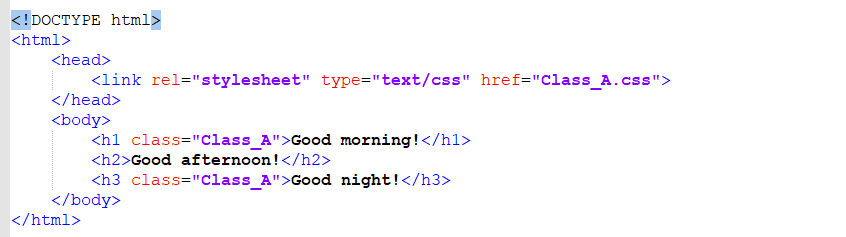
**CLASS IN EXTERNAL FILE**

Let’s try keeping our class in a separate file. To do this, we would first write a new css file as follows:

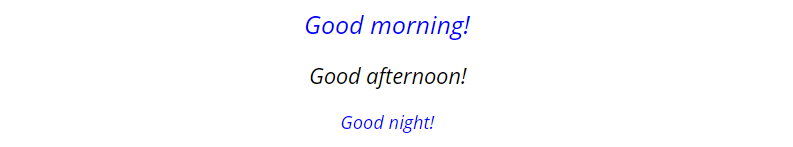


This code states that anything within the <body> element will be written in italics and centered on the page. We’ve also created are our class name (Class\_A) and stated that any element connected to that class name will have its text written in blue. Let’s say we saved the file as Class\_A.css.

Here is what the separate HTML file would look like:



The output of this code would look like this:



Notice that “Good afternoon!” is written in default black (not blue) because we didn’t utilize the class name with it

**ASSIGNMENT**

Complete the following actions:

1. Create a new CSS document and write your own class attribute that causes all <p> elements to have the following properties (you can set the values as you choose):

a. Color

b. Background color

c. Font-size

d. Text-align

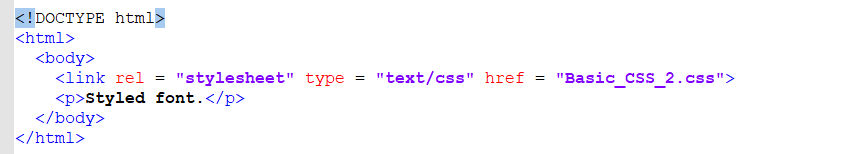
For example:



Save this file as Basic\_CSS\_2.css within your Basic\_HTML\_and\_CSS folder.

2. Create a new HTML document that minimally includes a <p> element. Save this file as Basic\_HTML\_5.html in your Basic\_HTML\_and\_CSS folder.

For example:



3. Successfully run the Basic\_HTML\_5.html file in the browser.

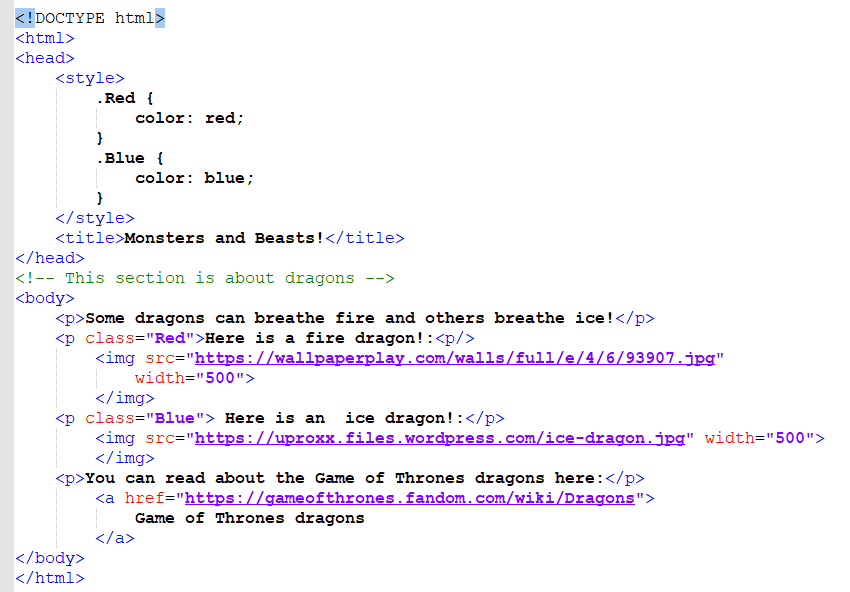
**COMMENTING CODE**

It is highly recommended that you write comments throughout your code that explains what each section of it is meant to do. This enhances the ability of other developers to read through and understand your code, and can also remind you of things in the future.

An HTML comment begins with <!– – and closes with – –>. HTML comments are visible to anyone that views the page source code, but are not rendered when the HTML document is rendered by a browser.

Meaning, you don’t see the comments when viewing the website in the browser but you can see them when you utilize Chrome’s dev tools.

An example of commenting HTML code is as follows:



The output of this code would be:

**HTML TABLES**

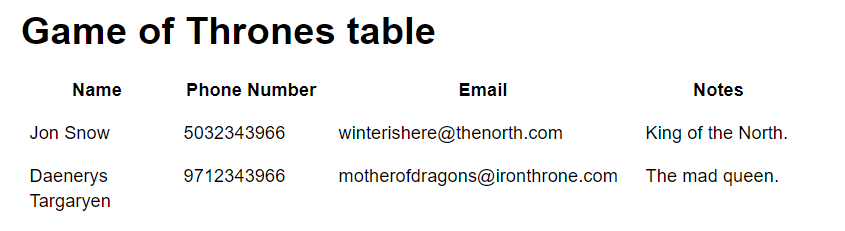
To create a table in HTML, we use the <table> tag

Rows are created with the <tr> tag (table row).

Table headers are created with the <th> tag. Table headers are bold and centered by default.

Table data (cells) are created with the <td> tag.

For example, look at this table:



To create this in HTML, we would write the following code:

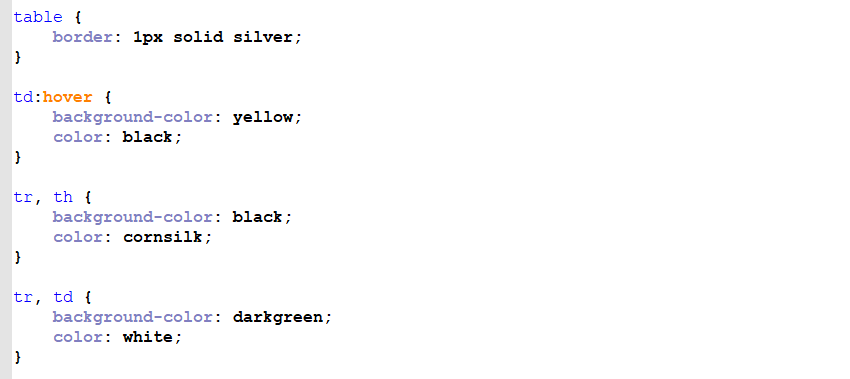


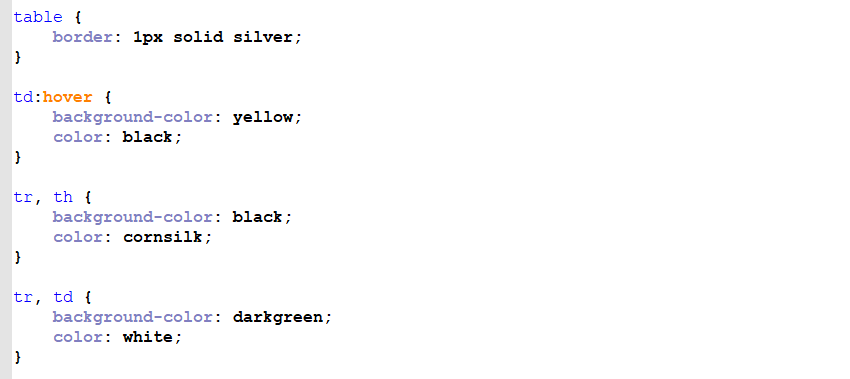
To create an ordered list, we use the <ol> tag. Each item on the list is contained within the list element <li>. In the earlier example, this would be written as:



To create an underordered list, we use the <ul> tag. The earlier example would be written as







https://techacademystorage.blob.core.windows.net/htmlandcss/styling7.png

**CSS LISTS**

We can also style lists with CSS. For example: the following CSS code can change an ordered list from 1., 2., 3. to Roman numerals:



And to make an unordered list from circular bullet points to squares, we write:



To add these effects to our code, we would first save the above CSS code as a separate .css file.

Then we would link our HTML document to the new CSS file.

Non-breaking Space

A common character entity used in HTML is the non-breaking space: **&nbsp;**

A non-breaking space is a space that will not break into a new line.

Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Examples:

* § 10
* 10 km/h
* 10 PM