# Difference between HTML & CSS

HTML (or Hypertext Mark-up Language) & CSS (Cascading Style Sheets) are both programming languages that work together to help make a website.



**Figure 1 – HTML code**

The HTML contains all the structural content (or skeletal bones like the meta, head, body tags (as identified in Figure 1), which are needed to help communicate to the web browser properly while the CSS is like the skin that contains all the elements that is needed to pretty the page up (by using different fonts, colours, margins & paddings adjustments etc). It allows you to vary your styling & formatting technique that can make it visually appealing for the user. (See Figure 2 below).

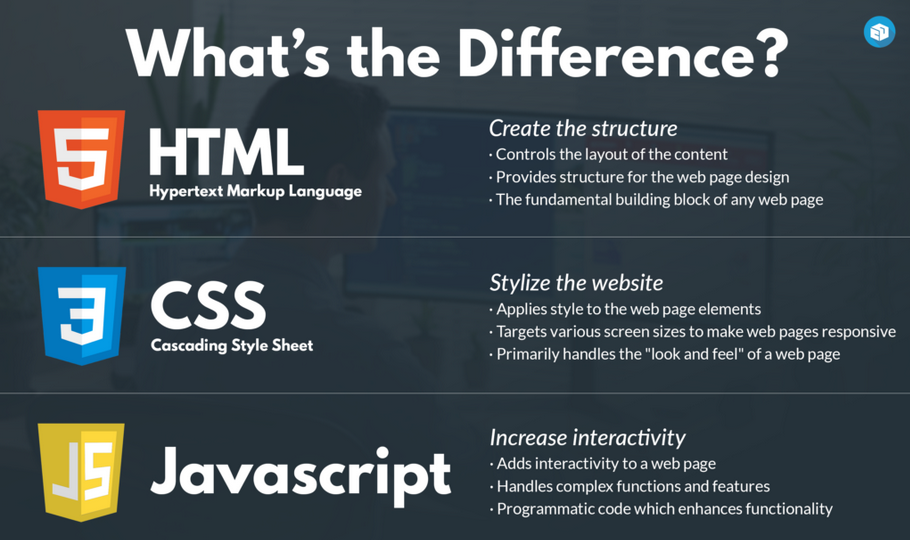


**Figure 2 – CSS code**

At a glance, both the HTML & CSS look similar, but they are not. The HTML gives instructions for what each element will do while the CSS works in the background listing the different styles & formatting techniques, we have imposed on each element to make it look different or the same to the rest of the sites content. They work in unison to give us the finished product.

# So why do we need JavaScript (JS) then?

Bryan Miller from [brytdesigns.com](https://brytdesigns.com/html-css-javascript-whats-the-difference/) defines the difference between all 3 different languages quite clearly in Figure 3 below.



**Figure 3 – HTML vs CSS vs JS**

JavaScript is a third language that works in unison with both HTML & CSS. It takes it up a level enhancing a websites functionality by making it more interactive to handle functions & complex calculations (that once coded correctly) removes the need for manual entries & human error while increasing through-put as data contained within can be retrieved & actioned automatically.

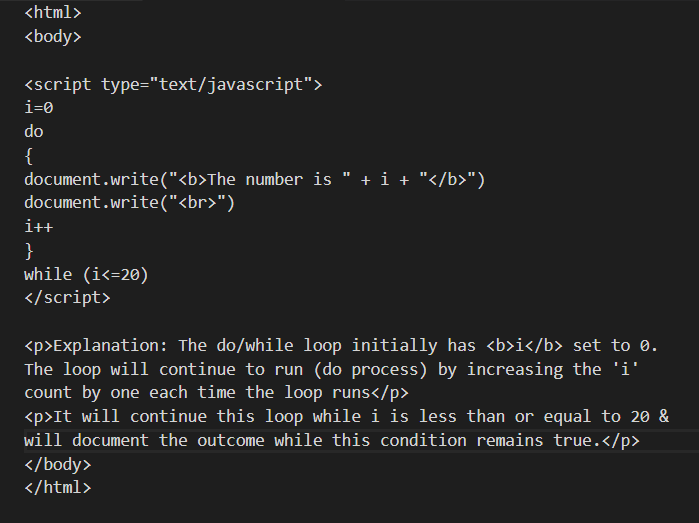
This could be something like purchasing a product online, creating a new account, identifying & paying for your order & then having the product shipped to the correct address. Then being able to retrieve your stored customer details again months down the track to be able to continue ordering more product off the same site.

# Explain Control Flow & Loops

Control Flow is the order of when a program command is executed in a system. It can be that you must complete this step first before continuing to the next step or utilise commands like **‘if/else’** statements that stipulate **‘if this is true, then complete this command, else if false – execute this instead.’**

The program could be stuck in a loop statement that continues going around & around in circles until the conditions are satisfied.

This could be something like starting at an initial number of 0. You could use a ‘**do/while loop’** that looks at checking the data & processing the outcome while checking, has it reached the condition = to ‘**20’** yet? No? Add an extra number to this & do the loop around again until the condition is true.



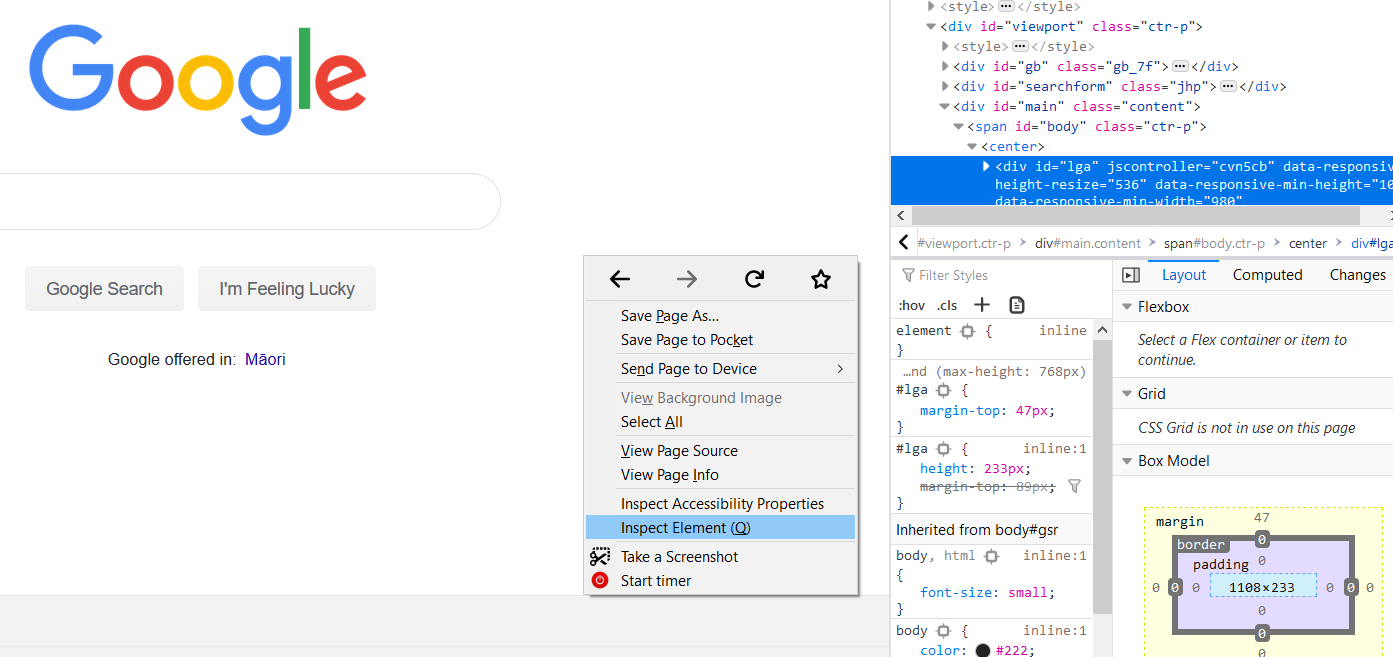
**Figure 4 – Do/While Loop**

# What is the DOM & how do you interact with it?

The Document Object Model (or DOM) is created when a web page is loaded into a browser. It provides a tree-like structure that stems from the document & contains objects from the page, which can be accessed & manipulated by JavaScript.

### Accessing the DOM

1. Open up any webpage
2. Either:
   1. Right-mouse click & select **INSPECT ELEMENT** or
   2. Press **F12** or
   3. Press **CTRL + SHIFT + i**
3. The **DOM** is the foreign jargon of code that gives you access to the **HTML, CSS & JS** content amongst other nifty features



**Figure 4 – Accessing the DOM**

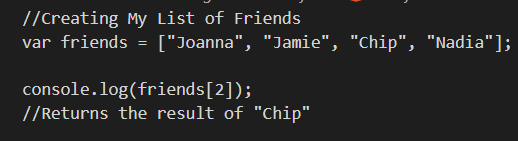
As a web developer, this can help you to visually identify the behaviour of each element you have with the current conditions that are in place & you can manipulate the DOM by making local changes to a site, but once refreshed, this content is lost.

It’s an easy way to view what you want your website to look like before committing changes fully in your HTML code.

You can also take any website, access the DOM & make changes to it that looks like you are a true hacker! Cool party trick!

# Accessing Data between Arrays & Objects

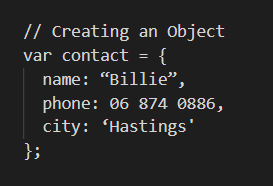
Arrays contain a list of objects (that stores a collection of data; like a list of your friends). See Figure 5 below where the data is contained in square brackets & separated by a comma).



**Figure 5 - Array**

This array called **friends** contains Joanna, Jamie, Chip & Nadia. To retrieve the data in an array, you refer to it by the index count which starts from **0**, where Joanna**[0]**, Jamie**[1]**, Chip**[2]** & Nadia**[3]** so when we run, **console.log** command, Chip is returned.

An **Object** is a collection of properties stored together in association with a **key:’value’** pair & are denoted by curly brackets **{ }** (See Figure 6).



**Figure 6 - Object**

Having access to an address book that stores a persons’ contact details (see Figure 6), you can access the content by either using: **. notation or [] notation** (I prefer .notation as it’s easier to read)

|  |  |
| --- | --- |
| .notation | [ ] notation |
| **contact.name** | **contact[name]** |
| **contact.phone** | **contact[phone]** |
| **contact.city** | **contact[city]** |

# What are Functions & Why are they Useful?

JavaScript uses Functions which are sets of instructions that execute command lines of code at given instances. It is useful as it:

1. frees the web developers from rewriting code time & time again
2. can be used more than once by ‘calling or invoking’ a function
3. means less code to sift through in the long run
4. can be utilised to lessen the monotonous tasks
5. calculates complex equations
6. can search through large data to find answers easily