JAVASCRIPT

AINT151 WEB TECHNOLOGIES

WHAT IS IT?

- **Dynamic** Scripting language (Not to be confused with the Java programming language)
- Manipulates **HTML** via the **DOM** (Document Object Model)
- **Widespread**; billions of devices with the ability to run JavaScript.
- Loads of API's to do interesting things! (WebGL, Geolocation, Audio & Video, Games!)

MULTIPLE ENVIRONMENTS

ADDING JAVASCRIPT TO A WEB PAGE

Link a .js file in our web page source

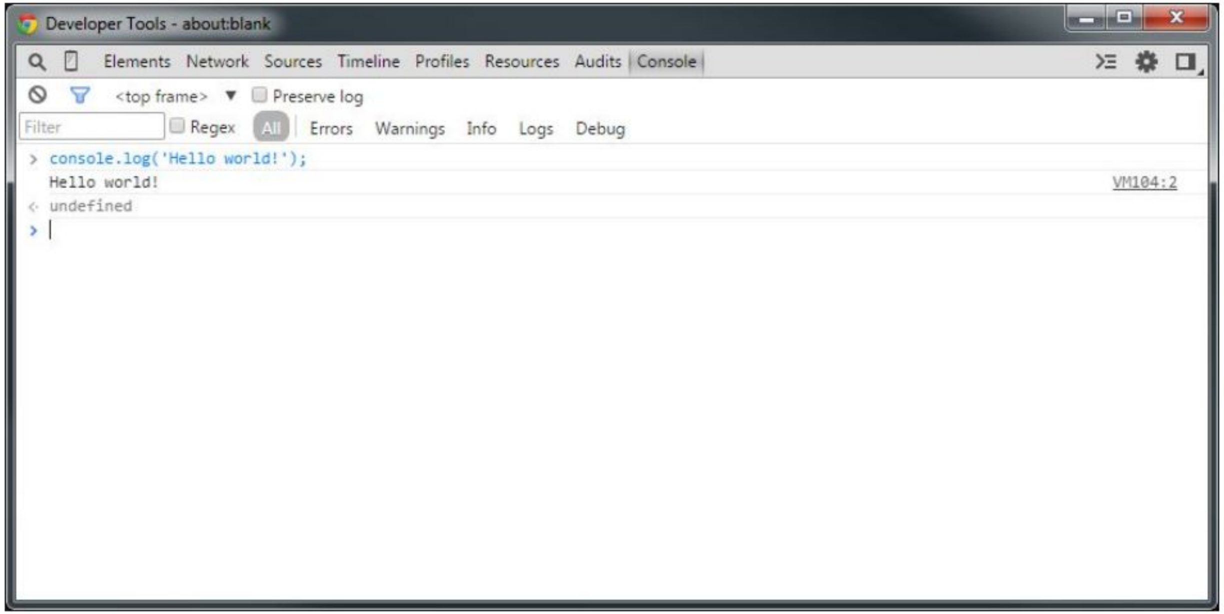
- <script src="js/main.js"></script>

Directly embed JavaScript in a <script> tag

- <script>
  console.log('hello world!');
  </script>

And at “**runtime**” via Browser Dev-tools **(great for testing).**

DEV TOOLS - SUPER USEFUL!


Developer Tools - about:blank
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Filter
Regex
All
Errors Warnings Info Logs Debug

```
> console.log('Hello world!');
  Hello world!                                                    VM104:2
<· undefined
>
```

JAVASCRIPT SYNTAX

VARIABLES

```
var helloworld = ‘hi’;
```

- Declared using the **var** keyword.
- No data type required
- Can be Numbers, Strings, Arrays & Objects
- Variable names are case sensitive!

```
var helloworld = 8;
```

```
var helloworld = [1, ‘hello’, 3];
```

```
var helloworld = {name: ‘hello’};
```

VARIABLES - let

```
let helloworld = ‘hi’;
```

- Declared using the **let** keyword.
- Used in the same way as **var**
- Keeps a variable to a certain data type
- Type cannot be changed after declaration
- Can be Numbers, Strings, Arrays & Objects

```
let helloworld = 8;
```

```
let helloworld = [1, ‘hello’, 3];
```

```
let helloworld = {name: ‘hello’};
```

CONDITIONALS

```
if (helloWorld == true) {
     console.log('Hello');
}
else if(helloWorld == false) {
     console.log('World');
}
else {
     console.log('HelloWorld');
}
```

- Always starts with **if**
- Optionally add more conditions with **else if**
- If no conditions are true, optionally include **else** as final code block
- The final **else** does not have a condition
- The **condition** is within the **()** brackets
- Code contained within **{}** brackets will run if the **condition** evaluates to true
- The conditions are evaluated one at a time, from top to bottom
- Only one **if/else if/else** code block will run

https://www.w3schools.com/js/js_comparisons.asp

https://www.w3schools.com/js/js_if_else.asp

LOOPS

```
while (helloWorld === true) {
     console.log('Hello');
}
//prints 'Hello' until helloWorld is false
```

```
for(var count = 0; count < 3; count++) {
     console.log(count);
}
//output
0
1
2
```

Loops run a block of code repeatedly

- While loops
  - Run the code block **while** a condition is true
- For loops
  - Run the code block **for** a set amount of times
  - Includes a "counter" variable that stores the current **loop count**

https://www.w3schools.com/js/js_loop_for.asp
https://www.w3schools.com/js/js_loop_while.asp

CONSOLE LOG

```
var myVariable = “Hello”;
console.log(myVariable);
// output:
// Hello
```

```
var numberA = 2;
Var numberB = 3;
console.log(numberA + numberB);
// output:
// 5
```

Output debug messages to a web browser’s console window

NOTE: Press f12 in your web browser to open the developer tools, including the console window

- console.log() will output anything within the **()** brackets:
    - console.log(“Hello”);
    - console.log(5);
    - console.log(myVariable);
- You can also evaluate conditions and equations

FUNCTIONS

Functions are custom code blocks that run a set of instructions

- Need to be “called” from somewhere else in the code
- Can **return** data after running (optional)
- Can be given data input, called **parameters** (optional)

```
function Add(numberA, numberB) {
     return numberA + numberB;
}
// calling the method
var answer = Add(2, 3);
// output the result
console.log(answer);
```

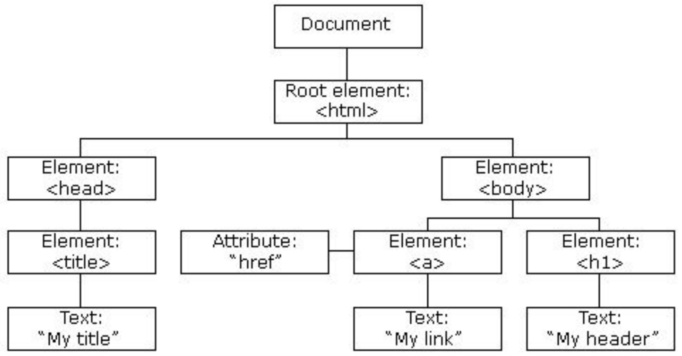
```
function Equal(numberA, numberB) {
     return numberA == numberB;
}
// calling the method
var answer = Equal(2, 3);
// output the result
console.log(answer);
```

THE DOM

DOCUMENT OBJECT MODEL

- When a web page is loaded into a browser, the browser forms a data model of the page
- This is called the Document Object Model
- JavaScript can manipulate the DOM to access and change any element

The DOM is a Tree hierarchy of all the HTML elements in a page



DOM HTML EXAMPLE

```
<html>
        <p id=”paragraph”>Hello</p>
</html>

<script>
var  element = document.getElementById("paragraph");

element.innerHTML = “Good Morning!”;
</script>
```

Get Element By Id
A common way to get a tag is to give it an id attribute
The DOM is accessed using **document**
getElementById returns a **HTML element** with the **id**

We can change the text or styling of the tag

https://www.w3schools.com/js/js_htmldom_methods.asp

DOM CSS EXAMPLE

```
<html>
	<p id=”paragraph”>Hello</p>
</html>

<script>
var  element = document.getElementById("paragraph");

element.style.color = “#009900”;
element.style.fontSize = “x-large”;
</script>
```

DOM style object
Use the **style** object to access CSS styles using JavaScript

Adjust CSS any properties dynamically on the selected elements:
Colours, fonts, background images etc

https://www.w3schools.com/js/js_htmldom_css.asp

EVENTS

WHAT IS AN EVENT?

- Events are triggered whenever something happens within the web page
- Our JavaScript can react to these events and execute code.
- Mouse clicks, form input, and page navigation can all be “listened” to.
- There are 2 ways to register an event handler for an element:
    - HTML Attribute - onclick
    - Event Listeners
- Event listeners offer the best option

https://www.w3schools.com/js/js_htmldom_events.asp

onclick EVENT LISTENER

Setting up an event listener function from a HTML tag using the **onclick** attribute

Easy to set up, may be hard to maintain if the page has lots of dynamic content

```
<p onclick=”Clicked()”>clickable tag</p>
<script>
function Clicked(){
        console.log(“Clicked tag”);
}
</script>
```

DOM EVENT LISTENER

Here we get an element by its id using the DOM, then add an event listener to it.

This method is slightly more complex, but can be easier to maintain for pages with dynamic content

```
<p onclick=”Clicked()”>clickable tag</p>
<script>
var myButton = document.getElementById("myButton");
myButton.addEventListener("click", Clicked);
function Clicked() {
        console.log(“Clicked tag”);
}
</script>
```