Modern Web Development

1

INTRODUCTION TO JAVASCRIPT

A quick revision

- Modern websites are build with a variety of web technologies:
  - HTML: defines the structure of the website
    - What is a paragraph, a heading, a list, etc
  - CSS: defines the look of the website
    - How the HTML elements will be rendered, what colours, fonts etc.
  - JavaScript (JS): defines the behaviour of the website
    - How the HTML elements and CSS rules change/behave in response to user actions.
    - With JS it is possible to change the content of any HTML element and change the rules of any CSS.
    - In essence with JS we write programs that create HTML and CSS!

It all starts with HTML

- The simplest way to add JS to our website is to write the code using the `<script>...</script>` HTML element
- This is similar to the way that we insert CSS in HTML, using the `<style>...</style>` HTML element.
- We can place the `<script>` in the `<head>`, but it is vastly preferred to have it as the last child of the `<body>` tag.
- Like with CSS, we can also have JS written externally to a file. We can use the `<style src="myCode.js">` to link to the JS file.

Let's Start!

This is the basic structure of an HTML document that uses CSS and JS

```
<!DOCTYPE html>
<html>
    <head>
        <!--External CSS rules->
        <link rel="stylesheet" type="text/css"
              href="mystyle.css">
        <!--Internal CSS rules->
        <style>
            /* CSS Rules go here */
        </style>
        <!--External JS rules->
        <script src="myCode.js"></script>
    </head>
    <body>
        <!-- All HTML is bellow here ->

        <script>
            /* internal JS code goes here */
        </script>
    </body>
</html>
```

External or internal JS?

- Placing scripts in external files has some advantages:
  - It separates HTML and code
  - It makes HTML and JavaScript easier to read and maintain
  - Cached JavaScript files can speed up page loads
  - Have the same JS code available for multiple HTML pages
- To add several script files to one page - use several script tags

JS output

- JavaScript can "display" data in different ways:
  - Writing into the HTML output using `document.write()`.
  - Writing into an alert box, using `window.alert()`.
  - Writing into the browser console, using `console.log()`.
  - Printing to a printer, using `window.print()`.
  - Writing into an HTML element, using `innerHTML`.

document.write().

Writing into the HTML output using `document.write()`.

```
<!DOCTYPE html>
    <html>
    <head></head>
    <body>

        <h2>My First Web Page</h2>
        <p>My first paragraph.</p>

        <p>Never call document.write after the
            document has finished loading.
            It will overwrite the whole document.
        </p>

        <script>
            document.write("<h1>hello</h1>");
        </script>

    </body>
</html>
```

window.alert()

Using window.alert()

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>

        <h1>My First Web Page</h1>
        <p>My first paragraph.</p>

        <script>
            window.alert("Some message");
        </script>

    </body>
</html>
```

Using console.log()

You will need to:

- Right-click at your window
- Select “inspect”
- Click from the top bar “Console”

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>

        <script>
            console.log(5 + 6);
        </script>

    </body>
</html>
```

JavaScript Print

JavaScript does not have any print object or print methods.

You cannot access output devices from JavaScript.

The only exception is that you can call the window.print() method in the browser to print the content of the current window.

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>

    <button onclick="window.print()">
        Print this page
    </button>

    </body>
</html>
```

Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

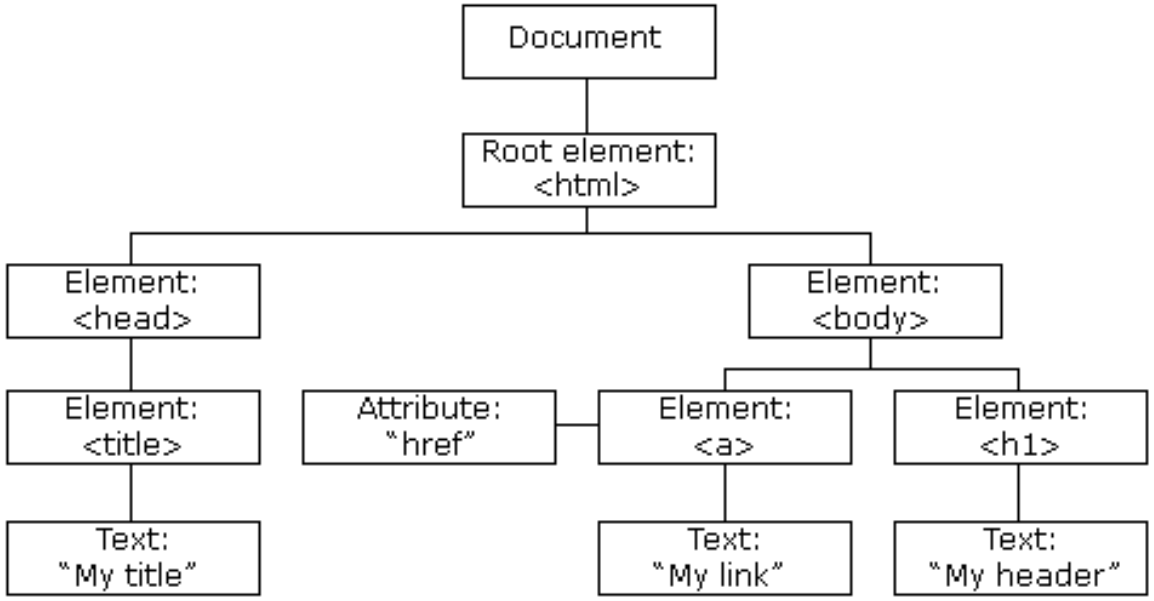
```
<!DOCTYPE html>
<html>
    <head></head>
    <body>

        <h1>My First Web Page</h1>
        <p>My First Paragraph</p>

        <p id="demo"></p>

        <script>
            document.getElementById("demo")
            .innerHTML = 5 + 6;
        </script>

    </body>
</html>
```

The Document Object Model (DOM)

- So far we have seen that in order to use JS we write something like:
  - `document.getElementById("demo").innerHTML = 5 + 6;`
- This is known as manipulating the DOM
- DOM is the way that a browser allows the user to have access to HTML elements, and CSS rules
- When a browser loads a page, it converts the HTML and CSS to a really big JS object (named document)

The DOM is Object-Oriented

- This is an Object-Oriented approach which you will discuss later on in Object-Oriented Programming
- For now we can thing of the DOM as a variable that contains all the HTML and CSS of a page.



JS variables

- A variable is a named container that holds a value.
- In contrast to JAVA that has many types of variables
  - int, float, String, char, short, byte, boolean
- In JS there is only one type:
  - var
- This is why JAVA is called “strongly typed” language.

JS var

Variables must have unique names, but a variable may change the type it holds during the program

JS var have a global scope and can be accessed from anywhere of the program (even at functions)

```
var price1 = 5;
var price2 = 6;
var total = price1 + price2;
console.log(total); // will print 11
...
total = “hello”;
console.log(total); // will print hello
```

Finding HTML elements in the DOM

- We can find elements in the DOM by searching the document object with one of the flowing ways:
  - By tag name (e.g. body, p, h1)
  - By class (any class defined with the class attribute)
  - By id (and id defined with the id attribute)

Accessing the DOM

```
<!DOCTYPE html>
<html>
   <head></head>
   <body>

      <h1>Hello</h1>

      <p id="007">I am a spy</p>
      <h2 class="aClass">a heading 2</h2>

      <script>
         //get many HTML elements by a name
         var byName =
         document.getElementsByTagName("h1");

         //get a single HTML element by its id
         var byID = document.getElementByID("007");

         //get many HTML elements by class name
         var byClass =
         document.getElementsByClass("aClass");
      </script>
   </body>
</html>
```

Changing HTML elements

- Once an HTML element is accessed, we can change its content with JS:
  - `var x = document.getElementByID("007");`
  - `x.innerHTML = "new content";`
- Or change its attribute with JS:
  - `document.getElementById(id).attribute = new value`
  - `x.src="file1.html";`
- Or change its CSS property with JS:
  - `document.getElementById(id).style.property = new style`
  - `x.style.display="block";`

JS functions

- Functions are defined with the `function` keyword
  - Function definition
  - ```
    var myFunction = function(a, b) {
      return a * b;
    }
    ```
  - Function call (use)

    ```
    var x = myFunction(4, 3);
    ```
    ```

Alternative JS functions

- Functions are defined with the `function` keyword
  - Function definition
  - ```
    function myFunction(a, b) {
      return a * b;
    }
    ```
  - Function call (use)

    ```
    var x = myFunction(4, 3);
    ```
    ```

JS events

- HTML events are **"things"** that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.
- Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked
- To add an event, we use the appropriate event attribute, and give it as value, the name of the function we wish to execute

Common HTML Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JS events

With event attributes

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <p id="para"></p>
        <button onclick="doThis">clickMe</button>



        <script>
            var doThis = function(){
                document.getElelemtById("para")
                .innerHTML="new text";
            }
        </script>
    </body>
</html>
```

JS events 2

With event listeners

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <p id="para"></p>
        <p id="interactive">click me</p>



        <script>
            var doThis = function(){
                document.getElelemtByID("para")
                .innerHTML="new text";
            }
            var pEl = document.
                        getElelemtByID("interactive");
            pEl.addEventListener("click", doThis);
        </script>
    </body>
</html>
```