CMP-4011A: Web-Based Programming

Lab Session (Week 4) – Building Web Programs using JavaScript and the DOM

Learning Objectives

- Understand JavaScript syntax, grammar, variables, and control structures.
- Add an external JavaScript file to your web pages.
- Access, edit and modify elements in a web document using JavaScript and the DOM.
- Navigate the DOM tree with parent, children, and sibling properties.
- Create new elements, update, and remove elements dynamically.
- Track down errors using the browser console.

Background

The objective of this practical session is to familiarize ourselves with the fundamental features of the JavaScript language, and learn to navigate to and modify different elements on the page using DOM properties and methods. The selected tasks involve updating the home page of the Norwich Yoga Studio (created in weeks 1,2 and 3) dynamically with new content using JavaScript.

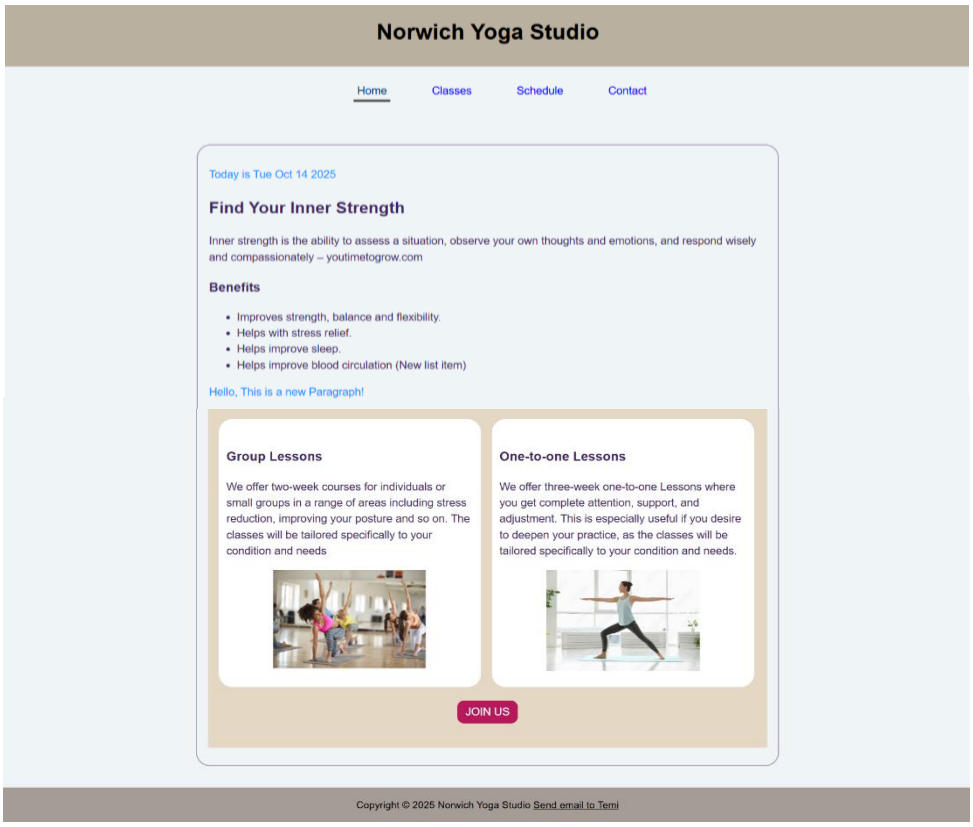

Figure 1.

Tools Required:

- Text editor (Visual Studio Code – follow a separate instruction on how to start VS code editor on the lab machines)
- Web Brower (Google Chrome)

The practical is quite open-ended and requires your active involvement in discovering and learning various aspects of the JavaScript language.

The main resources that will assist you are:

Lecture Notes

https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics - JavaScript Basics by MDN

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents - Manipulating Documents by MDN

Task 1: - Set up your development environment and folder structure – [10 minutes]

1.1 Open Visual Studio Code editor.

1.2 Create a new directory/folder in your Document directory (U: drive) called week4 and place it in the CMP4011A folder created in week 1.

1.3 For this exercise, copy yoga folder from week 3 and place it in the new week4 folder. This new yoga directory will be the project directory. The folder structure should look like this:

Task 2: - Add an external JavaScript file and write script to store data in variables – [20 minutes]

2.1. In the yoga folder, create a directory/folder called scripts, and create a file called apps.js in the scripts folder.

2.2. Within the head element of index.html, **add the JavaScript file (app.js) to the page using the <script> element** (place it after the link element). Use the *src* attribute to

specify the path to the script – *scripts/app.js*. Provide the <script> element with a 'defer' attribute so that the script will be executed AFTER the page loads.

```
<head>
      <link>...
      <script src="script/app.js" defer></script>
</head>
```

2.3. In apps.js file, using the *var* keyword, declare a variable called *myVariable* to store a string value – Welcome to CMP-4011A Web-Based Programming, and **print the value of the variable to the console** using the console.log() method.

```
var myVariable = "Welcome to CMP-4011 Web-Based Programming";
console.log(myVariable);
```

Save the file, refresh the page, and open the browser console (right click on the page ➔ Inspect ➔ Console or Ctrl+Shift+J) to see the value of myVariable**.**

2.4. Next, redeclare myVariable using the var keyword, update the value of myVariable to "Welcome to CMP-4011 Web-Based Programming Module"; and print the new value to the console. Save and refresh the page to see the new value.

2.5. In apps.js, declare a new variable letVariable to store the value - "This is a let variable"; and print the value to the console.

2.6. Redeclare the variable letVariable using the *let* keyword, update its value to "This is an updated let variable value"; and print the new value to the console. (Notice that let variables cannot be redeclared, however you can update its value by deleting the let keyword.) Save and refresh the page to see the new value.

2.7. Repeat steps 2.5 and 2.6 using the *const* keyword. (Notice that *const* variables can neither be redeclared nor updated). Save and refresh the page to see the new const value.

2.8. Using a *for loop*, print the myVariable value up to 5 times.

```
for (i=0; i<5; i++){
console.log(myVariable);
}
```

Save the file, refresh the page, and open the browser console to see the result. (Notice that there is only one output - Welcome to CMP-4011 Web-Based Programming Module, with a number 5 to indicate there are 5 identical outputs. This

is a normal behaviour of most browsers when identical outputs are generated in the console.)

2.9 Concatenate the index *i* to the variable name to print non-identical outputs.

```
for (i=0; i<5; i++){
console.log(myVariable + i);
}
```

Save the file, refresh the page, and open the browser console to see the result.

Task 3: - Update a paragraph element on the page with the current date – [30 minutes]

3.1 Add a paragraph element to index.html immediately after the opening tag of the main element. Add an id or class attribute (mypara) to enable you to reference the element in JavaScript. The paragraph will be used to specify the current date.

3.2 In app.js, create a date object using the new Date () constructor to obtain the current date and time, and store in a variable.

3.3 Use the toDateString() method to convert the current data and time to a readable string.

```
let myDate = new Date();
let todaysDate = myDate.toDateString();
```

3.4 Next, access the paragraph element (mypara) and update its content with the value of the current date to read as 'Today is Tue Oct 14 2025' (Figure1).

```
document.querySelector('.mypara').innerHTML = 'Today is ' + todaysDate;
```

Save and refresh the page the age to see the output.

3.5 Using the *setAttribute()* method (or add() method of the classList property), add a class name to the paragraph element in index.html to enable you change the text colour of the current date to *dodgerblue* in your stylesheet. [Hint: create a style rule for the specified class name within your CSS and add the color property and its value]. Save and refresh the page to see the effect.

Task 4: - Create new elements and update the DOM dynamically using JavaScript – [30 minutes]

4.1 Within app.js, create a new paragraph element with the text content "Hello, This is a new Paragraph!". The **new paragraph should be inserted just before the section element** in index.html. To create a new element, use the createElement() method of the document object, and use the textContent property to add text to the new paragraph as shown below.

```
let newPara = document.createElement('p');
newPara.textContent = "Hello, This is a new Paragraph!";
```

4.2 Use the insertBefore() to insert the new paragraph before the section element on index.html. The insertBefore() method inserts an element before a reference element as a child of a specified parent element –

```
referenceElement.parentNode.insertBefore(newElement, referenceElement);
```

First, we need to obtain the reference element - `let sectionElement = document.querySelector('section');` then use the insertBefore() method

```
let sectionElement = document.querySelector('section');
sectionElement.parentNode.insertBefore(newPara, sectionElement);
```

4.3 Render the content of the new paragraph in dodgerblue text colour. **Save and refresh the page.** The new paragraph and its content should look like Figure 1.

4.4 Using the insertBefore() method, insert a new list item (<li> element) with the content 'Helps improve blood circulation (New list item)' just before the last list item within the list of Benefits on index.html.

4.5 Using the removeChild() method of the DOM API, remove the last list item on the list of Benefits in index.html, leaving 4 items on the list instead of 5. [Hint: use the children property to return child elements].

Once you have completed this lab, show your code and output to the lab tutor who will sign the lab sign-off sheet if the lab has been satisfactorily completed.

You should upload the completed work to your OneDrive to avoid losing your work.

In any time that you have remaining, you should **meet with your coursework assessment group members** to continue planning your web project.

Temi

Oct. 2025

Appendix: Further References

The Script Element

In order to use JavaScript it either has to be included within a SCRIPT element, or (preferably) the script element should be used to provide a link to a separate JavaScript file stored with the extension .js

```
<!DOCTYPE html>
<html lang="en">
    <head>
            <title>JavaScript</title>
    </head>
    <body>
        <script type="text/javascript" src=myscript.js defer>

        </script>
    </body>
</html>
```

The Defer Attribute

Any JavaScript within the SCRIPT element will run as the page loads. However, by providing the SCRIPT element with a 'defer' attribute, the script will be executed AFTER the page loads IF IT IS CONTAINED IN A SEPARATE FILE.

JavaScript Basics

Types

JavaScript types include Numbers, Strings, Booleans, Objects, Undefined and Null.

Numbers

In JavaScript all numbers are represented as floating-point values.

```
var x = 100;
var y = 80;

var area = x * y; // = 8000
```

The fact that all numbers are floating point implies a need for care in handling values where precision is required.

Strings

A string is a sequence of Unicode characters and should be wrapped in a matching pair of either single or double quotes.

```
var firstWord = 'Hello';
var secondWord = 'World!';
var sentence = firstWord + ' ' + secondWord + '.';
// The + sign is the string-concatenation operator
```

Booleans

There are two possible Boolean values: true and false. Any comparison, using logical operators, will result in a Boolean.

```
5 === (3 + 2); // = true
var tall = true;
if (tall) {
    // shrink
}
```

The '===' is a comparison operator, meaning **equal in value and type**.

Functions

A Function is a specialised Object:

```
// Using the function keyword to create a new function:
function myFunctionName(arg1, arg2) {
    // statements.
}

// If the name is omitted then
// it is a nameless or anonymous function - these are used a great
      deal in JavaScript:

function(arg1, arg2) {
    // statements.
}

// Running a function is simply a case of referencing it
// and then adding a parenthesis (with any arguments):
     myFunctionName(); // No arguments
     myFunctionName('one', 'two'); // with arguments

// You can also run a function directly:
```

```
(function(){
    // A self-invoking anonymous function
})();
```

Note that variables declared within function with the keyword *var* are LOCAL in scope to the function.

Arrays

An Array is also a **specialised object** and can contain any number of data values of mixed types. To access data values within an array a ‘subscript’ or 'index' is used:

```
// Literal Array:
var days = ['monday', 'tuesday', 'wednesday'];

// Using the Array constructor:
var days = new Array('monday', 'tuesday', 'wednesday');

days[0]; // Access the 1st item of the array (monday)
days[2]; // Access the 3rd item of the array (wednesday)
```

Objects

An Object is a collection of named values (key - value pairs). It's similar to an array but a name may be given for each data value.

```
// 2 different ways of declaring a new Object,

// Object Literal :
var student = {
    name: 'Joe Lewis',
    age: 20,
    course: 'CSSH14'
};

// Using the Object constructor:
var student = new Object();
student.name = 'Joe Lewis',
student.age = 20,
student.course = 'CSSH14'
```

Selection

If..Else Statement

```
var width = 800;
var xpos = 780;

if ( xpos > width ) {
   xpos=0;
} else {
   xpos+=1;
}
```

Switch Statement

```
switch (new Date().getDay()) {
    case 1:
        day = "Monday"; break;
    case 2:
        day = "Tuesday"; break;
    case 3:
        day = "Wednesday"; break;
    case 4:
        day = "Thursday"; break;
    case 5:
        day = "Friday"; break;
    default:
        day = "Weekend"; break;
}
```

JavaScript Operators:

A complete reference to Javascript operators can be found at http://www.w3schools.com/js/js_operators.asp. You should not find too many surprises if you have previous Java experience.

Iterators (loops)

Looping has similar support to other languages. The while (indeterminate) and for (determinate) loops are common to most contemporary languages:

```
var weekDays = ['monday','tuesday','wednesday','thursday','friday'];

// WHILE
var i = 0;
```

```
while (i < weekDays.length) {
    alert(weekDays[i]);
    i++;
}

// FOR
for (var i = 0, i < weekDays.length; i++) {
    alert(weekDays[i]);
}
```

Javascript's **for..in** statement may not be so familiar. It iterates over the properties of an object, in *arbitrary* order. For each distinct property, statements can be executed:

```
// FOR..IN
for (day in weekDays) {
    alert(day);
}
```

Temi

Oct. 2025