University of Brighton

School of Computing, Engineering and Mathematics

Tutorial Workbook

CI435 Introduction to Web Development

Permitted exam material

Semester 2 Examination 2020/2021

Complete these details in the exam:	
Student ID	
Place number	

Workbook guidelines:

1 The workbook must be printed on paper before the exam. **Only hard copies of the workbook are permitted in the exam.**

2 You do not have to bring a workbook to the exam. It is **permitted but not mandatory**.

3 You must **not add any pages** to the workbook:

- Maximum 52 pages for the standard version
- Maximum 81 pages for the large print version

4 There are **no restrictions on font sizes** as long as you can read them.

5 The content in the **workbook is not marked**.

6 The workbook should contain your **code and notes from the tutorials**:

- e.g. text, code examples, diagrams, tables, drawings, etc.

7 It should **not** contain materials copied from third-party websites:

- e.g. screenshots of web pages are not OK (internet access is blocked for a reason)

8 You must fill in your **student number and place number** if you want to use your workbook in the exam.

9 The workbook will be **collected at the end of the exam** before you leave the exam room.

Note that it is impossible to cover every border case here. Please be sensible and use your own judgement if unsure about permitted content. Keep in mind that the exam is supposed to test your knowledge acquired in this module without access to outside information. Do not use the workbook to gain an unfair advantage or to circumvent restrictions on Internet access.

TUTORIAL 1

JAVASCRIPT BACKGROUND

1 Which person is credited with creating JavaScript?

2 Which organisation is responsible for standardising JavaScript?

3 JavaScript is also known by which name?

4 Describe what is meant by declarative and procedural languages. Which is JavaScript?

5 Describe what is meant by compiled and interpreted languages. Which is JavaScript?

6 Describe what is meant by weakly and strongly typed languages. Which is JavaScript?

7 Describe three common uses for JavaScript in Web development.

8 Describe two ways to activate the developer tools in Chrome.

9 Using the Chrome developer tools,
- where would you find out about page elements and their style?
- where would you inspect the different files making up a web page?
- where would you inspect console outputs of JavaScript code?
- where and how would you debug JavaScript code in a page?

10 Use the Chrome developer tools to explore your favourite website:
- does the page log any information to the console?
- how many JavaScript files does it load and execute?
- can you identify an interesting code section and debug it?

TUTORIAL 2

STATEMENTS, VARIABLES, DATA TYPES

You are welcome to use a code editor of your choice for the tutorials in this module. The editor used in lectures and learning materials will be Visual Studio Code. If you decide to use a different editor, make sure to not use any code generation features as this usually leads to bloated and unintelligible code.

1 Create a new folder in your web space, e.g. `ci435/sem2/session02`, and in it create a HTML page called `index.html` and an empty JavaScript file called `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Open the new page in the Chrome browser. Taking the path above, the URL would be:
`http://username.brighton.domains/ci435/sem2/session01/index.html`

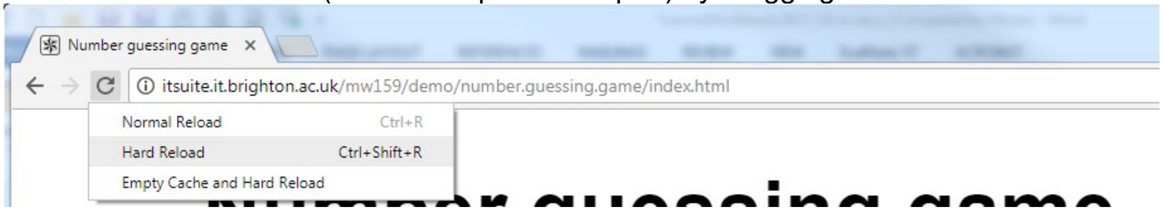
3 Activate the Chrome developer tools and locate the console window. Then add the following statement to your JS code to check that the HTML and JS the files are linked correctly:

```
console.log("Hello World!");
```

Save the JS file, then reload the page. The console should display the text `Hello World!`

HINT: Web browsers sometimes cache HTML, CSS and JS files for better performance. This can be a problem during development as changes to source files are not loaded from the server. To make deal with this problem, the Chrome developer tools offer additional reload options:
- `Hard Reload`
- `Empty Cache and Hard Reload`

These can be accessed (with developer tools open) by dragging down the Reload button:


Number guessing game
itsuite.it.brighton.ac.uk/mw159/demo/number.guessing.game/index.html
Normal Reload Ctrl+R
Hard Reload Ctrl+Shift+R
Empty Cache and Hard Reload

4 In your JS code, add statements for each of the following:

3.a define a variable named `a` and initialise it to the value of 5
3.b log the value of `a` to the console. HINT: use `console.log();`
3.c define a variable named `b` and initialise it to 3
3.d log to the console the value of `a` multiplied by `b`
3.e log to the console the value of `a` added to `b`, then modulo 3
3.f define a variable `s1` and initialise it to "Hello" and a variable `s2` and initialise it to your name
3.g print to the console the result of concatenating `s1` and `s2`
3.h improve the output of the previous step by adding a blank space between `s1` and `s2`
3.i log to the console the concatenated string `a` " squared is " `a*a`

5 In JavaScript a semi-colon after a statement is optional. Why should you always use it anyway?

6 In JavaScript variables can be declared with the keywords `var` and `let`. Explain the difference.

NOTE: Beware Microsoft smart quotes when copying code from Word and Powerpoint documents. Word and Powerpoint are configured by default to change ASCII quotes (such as "…" and '…') to smart quotes (such as “…” and ‘…’).
Smart quotes are not valid in JavaScript code and will cause problems.

TUTORIAL 3

CONTROL STRUCTURES

1 Create a new folder in your web space, e.g. `ci435/sem2/session03`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Print to the console first the expression `(1 == true)` and then the expression `(1 === true)`. Note the difference between using the simple and strict comparison operators `==` and `===`.

3 Define two variables `x` and `y` and initialise them to 3 and 5.

3.a Print to the console the following expressions:

- `x` less or equal to `y`
- `x` greater than 4 and `y` greater than 4
- `x` greater than 4 or `y` greater than 4
- not (`x` greater than 4 or `y` greater than 4)
- `x` not equal `y` and `y` greater than 0
- `x` times `y` less than 15
- `x` times `y` equal or less than 15

3.b Use an `if` statement that tests whether `x` is greater than `y` and then prints either "x is greater than y" or "x is less than or equal to y" to the console.

3.c Log the result of dividing `x` by `y` only if `x` is greater than `y`

3.d Add `x` to `y` while `y` is less than 100, then log the value of `y` to the console

HINT: use a `while` loop

4 Log the string "Hello `<number>`" to the console `y` times (i.e. Hello 1, Hello 2, Hello 3 ....)

HINT: use a `for` loop

5 Define a variable `day` and initialise it to a random number between 0 and 6. Then use a `switch` statement to test the value of `day` and print a corresponding day of the week to the console. (0=Monday, 1=Tuesday, ..., 6=Sunday)

Hint: use the following expression to generate a random number between 0 and 6:

```
var day = Math.floor(Math.random() * 7);
```

TUTORIAL 4

WORKING WITH STRINGS

1 Create a new folder in your web space, e.g. `ci435/sem2/session04`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Declare a variable `str` and assign it the following piece of text:
```
I'm learning JavaScript in my web development module.
```
HINT: Remember to escape any special characters.

3 Log the number of characters in string `str` to the console.

4 Find the position of the string `"JavaScript"` in string `str` and log it to the console.

5 Count the number of times the letter `"e"` appears in the string `str`.

HINT: Using a `for` loop and the `String.charAt()` method.

6 Find the position of all letters `"m"` in string `str` and log them to the console

HINT: The `String.indexOf()` takes an optional second parameter.

7 Reverse string `str` and print it to the console. The result should be:
```
.eludom tnempoleved bew ym ni tpircSavaJ gninrael m'I
```
HINT: Use a second string variable, a decrementing for loop, string concatenation and the `String.charAt()` method.

8 Declare a variable `url` and assign it the current web address. Test if the page was called up from a web server (`http:`), a secure web server (`https:`), the file system (`file:`) or any other source and print the result to the console.

HINT: 1) Use the `if / else if / else` construct and the `String.startsWith()` method
2) Use the following statement to get the current web address:
```
var url = window.location.href;
```

TUTORIAL 5

WORKING WITH ARRAYS

1 Create a new folder in your web space, e.g. `ci435/sem2/session05`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Declare an array with friend names, e.g.

```
var friends = ["Benjamin", "Lily", "Emily", " Gabriel"];
```

Then complete the following tasks, logging the resulting array to the console after each step:

2.a Add two new friends to the array called `Isabella` and `Charlotte`.

2.b Change the last element in the array to `Miles`

2.c Using the `splice()` method, replace the second and third friends in the array with a single new friend `Aria`

2.d Using the `pop()` method, remove the last element of the array and then insert it at index 0 using the `unshift()` method

2.e Reverse the process by using the `shift()` method to remove the first element of the array and the `push()` method to add it at the end of the array

2.f Using the `indexOf()` method, find the index of the name `Aria` in the array

2.g Using the `indexOf()` method, log to `true` the console if the array contains the name `Bob` or `false` otherwise. Try doing this in a single statement.

2.h Using the `sort()` method, sort the array in ascending alphabetical order

2.i Using the `join()` method, create a string of all names separated by a comma space (`", "`)

3 Declare a new variable `count`, then loop over the array `friends` and add the length of each name to the `count` variable. Log to the console the total number of characters in all names and the average numbers of characters per name.

TUTORIAL 6

FUNCTIONS

1 Create a new folder in your web space, e.g. `ci435/sem2/session06`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Write a function called `multiply` that takes two arguments `(a, b)` and returns the result of multiplying those arguments together. Use the new function to log to the console the result of multiplying two numbers, e.g. `354 * 875` and `9876.543 * 765.123`.

Try calling function `multiply` with one parameter or three parameters instead of the required two parameters and log the result to the console. What happens? How does JavaScript handle missing or surplus parameters? Research the special values `undefined` and `NaN`.

3 Write a function called `countChar` that counts how often a character `c` occurs in a string `s`. The function takes two arguments (`s` and `c`) and returns a number. Use the new function to log to the console the number of times the character `'l'` occurs in the word `'Hello'`.

HINT: since we are looking for a single character, we can use the `String.charAt()` method

4 Write a function called `countStr` that counts how often a search string `needle` occurs in a string `haystack`. The function takes two string arguments and returns a number. Use the new function to log to the console

a) the number of times the string `'cream'` occurs in the string
```
'I scream, you scream, we all scream for icecream!'
```

b) the number of times the string `'sea'` occurs in the string
```
'She sells sea shells on the sea shore. The shells that she sells are
 sea shells I'm sure. So if she sells sea shells on the sea shore, I'm
 sure that the shells are sea shore shells'
```

HINT: since we are looking for a string with multiple characters, we cannot use the `String.charAt()` method. Instead, try the `String.indexOf()` method and use the second parameter to control the start index for the search after each find. This is a tricky problem, so don't despair if things don't work out immediately.

5 Write a function called `rotate` that takes two arguments (`arr`, an array, and `num`, a number), which will modify the `arr` array by *popping* an element off the last element and *splicing* it at the start of the array. This should be repeated `num` times, effectively rotating the array. For example:
```
rotate(["Harry","Sarah","Oscar","Tina"], 2)  ->
["Oscar","Tina","Harry","Sarah"];
```

6 Sort the array `[12, 9, 33, 8, 71, 2, 41, 5]` first in alphabetical order, then in ascending and in descending numerical order, each time logging the result to the console. Use anonymous functions as arguments to the `Array.sort()` method for the numerical sorts.

HINT: Refer to the lecture slides on how to do this.

TUTORIAL 7

DOM 1 - INTRODUCTION, ACCESSING ELEMENTS AND ATTRIBUTES

1 Create a new folder in your web space, e.g. `ci435/sem2/session07`, and in it save a copy of the `index.html` and `index.js` files from this week's materials zip file. To complete the tasks below you should edit the `index.js` file.

NOTE: index.js initially looks like this:

```
window.addEventListener("load", function(){

    // add your tutorial code here

});
```

Don't worry about this piece of code, we'll talk about `Event` and `EventListener` next week. For now just add your code after the comment line and before the closing curly bracket `}`.

2 Select the element with the id `elem1` and log its content to the console. Do this twice, first using `querySelector()` then using `getElementById()`. Notice the difference in specifying the element ID in these two versions.

3 Practice your selector skills: Select the following elements then (a) loop over them and log to the console their content, and (b) log to the console the number of elements:

- all paragraph `<p>` elements
- all elements with the class `highlight`
- all list item `<li>` elements in the list with id `first`
- all list item `<li>` elements with the class `highlight`
- all list item `<li>` elements and all paragraph `<p>` elements with the class `highlight`

4 What are the advantages of `querySelector()` and `querySelectorAll()` over older methods like `getElementById()`, `getElementsByClassName()` and `getElementsByTagName()`?

5 `querySelector()` and `querySelectorAll()` are available for the `document` object as well as for DOM nodes. To test this, first select the `<ul>` element, then use `querySelectorAll()` on it to select all child nodes of class `highlight`. Log their number to the console.

6 The DOM provides node properties to access parent, sibling and child nodes. To test this, select the element with id `first` and then log to the console:

- the node type, node name and text content of its first child node (`.firstChild`)
- the node type, node name and text content of its next sibling node (`.nextSibling`)
- the node type and node name of its parent node (`.parent`)

HINT: To complete this question, first research the DOM node properties `.nodeType` and `.nodeName` at https://developer.mozilla.org/en-US/docs/Web/API/Node

TUTORIAL 8

DOM 2 - EVENTS, LISTENERS, MANIPULATING ELEMENTS

1 Create a new folder in your web space, e.g. `ci435/sem2/session08`, and in it save a copy of the `index.html` and `index.js` files from this week's materials zip file (note that the `index.js` file is empty at this stage). To complete the tasks below you should edit the `index.js` file.

2 Attach an initial `load` event listener to the `window` object, which hosts all the code to set up further event listeners on document elements once the DOM is fully loaded. If you are unsure how to do this, look at the JavaScript code from the previous tutorial.

3 Attach a `click` event listener to the button with id `btn3`. The event listener should log the text `"Clicked"` to the console.

4 Attach a `click` event listener to the button with id `btn4`. The event listener should get the paragraph with id `p4`, log its current text to the console, and replace it with `"Success"`.

5 Attach a `click` event listener to the button with the id `btn5`. This event listener should get all paragraph `<p>` elements and log their text to the console.

6 Investigate mouse events (e.g. `mousedown, mouseup, mousemove, click, etc.`) by registering a listener for each event on the `div` with id `div6`. The listeners should log event properties to the console including the event type and the mouse position.

HINT: To complete this question, first research `MouseEvent` and general `Event` properties at https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent and https://developer.mozilla.org/en-US/docs/Web/API/Event

7 Attach a `click` listener to the anchor element with id `a7`, which logs the link target (`href` attribute) to the console. The listener should prevent the default action of the anchor.

8 Attach a `click` event listener to all table cell `<td>` elements, which replaces the current cell text with `"Success"` when clicked.

TUTORIAL 9

DOM 3 - CREATING AND DELETING ELEMENTS

1 Create a new folder in your web space, e.g. `ci435/sem2/session09`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Create a paragraph element in your HTML with the id `p2` and a `title` attribute with some text. Add a `click` event listener to the paragraph that swaps the `title` attribute with the text content.

The code should turn a paragraph like this: `<p title="Title text">Content text</p>` into this: `<p title="Content text">Title text</p>` and back again.

3 Create an image element in your HTML with the id `img3`, then find two images of the same size (e.g. 300x300 pixels) and use one of them as the image source. Add a `click` listener to the image that swaps out the images shown in the image element.

HINT: You can do this by reading and manipulating the `src` attribute of the image.

4 Create a button in your HTML with id `btn4` and a `div` element with id `div4`. Add a `click` listener to the button that creates an unordered list `<ul>` from the array given below, with each array item being shown as a listitem `<li>` element. Append the list to the target `div4`.

```
var list_content = ["Apple", "Banana", "Orange", "Mango", "Papaya"];
```

5 Create a button in your HTML with id `btn5` and a `div` element with id `div5`. Add a `click` listener to the button that creates a table from the two-dimensional array given below. Append the table to the target `div5`.

```
var table_content = [
    ["Tom", 20, "BSc (Hons) Computer Science"],
    ["Mary", 19, "BSc (Hons) Computer Science (Games)"],
    ["Jerry", 21, "BSc (Hons) Digital Games Production"],
    ["Jane", 19, "BSc (Hons) Digital Media"],
    ["Olivia", 22, "BSc (Hons) Digital Media Development"],
    ["Jack", 20, "BSc (Hons) Software Engineering"]
];
```

6 Create a button in your HTML with id `btn6` and a `div` element with id `div6`. Add a `click` listener to the button that creates 10 paragraph elements with the text `"This is paragraph <n>"` and appends them to the target `div6`.

7 Create a button in your HTML with id `btn7` and add a `click` listener to it that removes all paragraphs from target `div6`. This should work together with the previous question and lead to two buttons that dynamically create and remove paragraph elements.

HINT: Study the slides how to efficiently remove all child nodes from an element.

TUTORIAL 10

WORKING WITH FORMS

1 Create a new folder in your web space, e.g. `ci435/sem2/session10`, and in it create a HTML page called `index.html` and a JavaScript file `index.js`. Link the JavaScript file to the HTML using a `<script>` element.

2 Create a form with a text input field with the id `fahrenheit`, and a paragraph element with the id `celsius`. Attach a `keyup` listener to the input field `fahrenheit` that reads its current value, converts it to degree Celsius and updates the paragraph with id `celsius` accordingly.

3 Create a form with the id `form3` containing an email input field with the id `input3` and a submit button, and a target `div` element with the id `div3`. Attach an event listener to the form that intercepts the `submit` event, retrieves the value in the email field, validates it as a University of Brighton email address, and displays any issues in the `div` element. Valid University of Brighton email addresses must be at least 20 characters long and end with `"@brighton.ac.uk"` or `"@uni.brighton.ac.uk"`.

If the entered data is **invalid** then an error message should be displayed in a `div` explaining the problem and the text field's background should turn red. If the number is **valid** then any previous error message should be cleared and the text field's background should turn green.

4 Create a form with the id `form4` containing a text input field with the id `input4` and two buttons with the id `btn4add`, and `btn4clear`. Also create an empty unordered list `<ul>` with the id `ul4` just below the form. Add a `click` listener to `btn4add` that retrieves the value in the text field and adds it as a new list item to the unordered list `ul4`. Add a `click` listener to `btn4clear` that removes all list items from the unordered list `ul4`.

5 Create a form with the id `form5` containing various input elements including:
- three `checkbox` elements with the values `"Donut", "Bagel"` and `"Burger"`
- three `radio` elements with the values `"Dog", "Cat"` and `"Fish"`
- a `select` element with three `option` elements `"Apple", "Banana"` and `"Orange"`
- a `textarea` element for multiline text entry

Attach a `submit` event listener to the form that prevents the default action and instead logs to the console what the user has entered/selected in the form.

Hints: 1) Make sure to give input elements a suitable `id` and/or `name` to be able to determine which values were entered/selected by the user.
2) Consult the lecture slides how to ensure that only one of the radio inputs can be selected at any time
3) Don't just log plain values; provide some context information to make log messages easy to understand.