IT2805 Assignment 4: JavaScript

Passing marks 65%

# Part 1 (50%)

Write all the JavaScript within the same provided file script.js. Write your code on the places indicated by the comments.

## Task 1.1 (3%)

Download the provided files index.html and script.js. Connect the JavaScript file with the HTML document using a relative path. Where in the HTML document would you include it, and why? Write your answers as a comment below your inclusion

## Task 1.2 (3%)

Below /\* Task 1.2 \*/ in script.js, write a for loop which prints the integers from 1 to 20 in the developer console. It shall not print the numbers 0 and 21. You can check the result by opening the developer console in your web browser.

## Task 1.3 (10%)

Now you will implement a simplified version of a common interview test, combining the for loop with an if statement. Write your code below /\* Task 1.3 \*/.

You will see the array 'numbers'. Use a for loop to go through that array. For each element in the array, print the number to the console using console.log. If the number is divisible by 3, instead of the number print the string 'eple' to the console. Should the number be divisible by 5, print the string 'kake' to the console instead of the number.

**Hint**: to check if a number is divisible by another number, use modulo (%). E.g. To check if the number 12 is divisible by 6, you can write 12%6, which will return 0.

Comment from the author: If a number is divisible by both 3 and 5, either 'eple' or 'kake' can be printed, this is dependent on your implementation.

## Task 1.4 (10%)

What is a page without a title? Well, what is a snake without its head? Pretty darn boring. Instead of simply adding the title in the HTML document, we will manipulate the DOM to add it. Sounds scary? It's not. In your HTML document index.html, you will see a h1 element with the id title. Use JavaScript to access this element and add the string ‘Hello, JavaScriptʼ to it. As you might have guessed, write your code below /\* Task 1.4 \*/

**Hint**: you can use .innerText to add text content to an element.

## Task 1.5 (12%)

We have manipulated the DOM by adding content to an element. Now we will use JavaScript to change a style attribute of two divs.

The provided HTML file contains three buttons and two divs. In script.js you will find three functions, one for each button. When clicked each button should activate its corresponding function, as listed below.

* Display: none -> changeDisplay()
* Visibility: hidden -> changeVisibility()
* Reset -> reset()

Below is a description of what each function should do. You will need to implement them correctly.

* changeDisplay() should change #magic's display attribute to none.
* changeVisibility() should change #magic's visibility attribute to hidden, and the display attribute to block.
* reset() should set #magic's display and visibility attributes to their default values, that is block and visible, respectively.

You can make changes to the HTML document if you need to do so.

Now open your webpage and look at the magic. If you have done it correctly, the div with number 1 should be removed from the document flow when you press the button 'display: none' and be invisible (but take up space in the document) when you press the 'visibility: hidden' button.

## Task 1.6 (12%)

Remember when you made an HTML list? Well, now you will populate that list using JavaScript. This way you can easily create dynamic webpages with your hands behind your back (this is not true).

In the HTML document, below <h2> Task 1.6 </h2>, you will find an empty unordered list with the id='tech'. In the JavaScript file, below /\* Task 1.6 \*/, you will find the array technologies. Loop through the array, and for each element, add it to that list in index.html

**Hint**: One way to solve this task is to combine each list element with the HTML tags <li></li>. You then have to concatenate string. In JavaScript you use the operator + to concatenate strings.

To solve this task, you can also use the JavaScript function appendChild(), which you can read about [here](https://www.w3schools.com/jsref/met_node_appendchild.asp). The important part is to create the list dynamically.

# Part 2 (50%)

Are you having trouble remembering what you have to do? Great news everyone! In this assignment you will make a todo list. Your submission for this part shall consist of two files, an HTML page named todo.html and a JavaScript file named todo.js. Go ahead and create them, and we will get along with this exciting exercise.

## Task 2.1 (5%)

The HTML page should consist of a title, one button for adding tasks, a correlating input field, an output element, and a list where the tasks are presented. Make sure to add IDs on the appropriate places.

When the user enters the page, the input field shall automatically get the focus. Connect your HTML page with the JavaScript file.

**Hint**: The button and input field could both be input elements. An input element, with type set to submit, could act as the button.

## Task 2.2 (10%)

Now you will start to implement some functionality. When the user writes a task in the input field and clicks the add button, the task shall be added to the empty list with a checkbox in front of it. Make a JavaScript function named "addTask()" that does this. Place the newest task on the top of the list.

**Hint** 1: Use eventlisteners to listen on button clicks.

**Hint** 2: You can remove the list bullet using CSS if you wish to do so.

**Hint** 3: Use preventDefault() so that the page does not reload itself if you use a form.

## Task 2.3 (10%)

Instead of simply showing the tasks in the DOM, we also want to store them as an object.

Create an empty list in todo.js called tasks. Then expand your function "addTask()" so that it adds a task object to the list.

The object should consist of at least one attribute to represent the text of the todo as a string.

You can check that this works by using console.log on the list 'tasks’.

## Task 2.4 (12%)

**Note**: this task can be completed using CSS, but you have to stick to only using JavaScript.

If the user checks the checkbox, the task should have a line through it. Should the user uncheck the checkbox, the task should be reverted to it's default state (no line through).

## Task 2.5 (13%)

We have not forgotten about the output element. Every time you update a task (check it, uncheck it or add it) the output element should be updated with the number of completed tasks and the number of tasks. E.g. if you have 7 tasks and 3 are completed, it should read '3/7 completed'.

# Submission

Put index.html, script.js, todo.html and todo.js in a zip folder and upload on Blackboard.