JavaScript Exercises and Consolidation

So far you have been introduced to the following JavaScript commands:

* **if...else if....else statement** - use this statement to select one of many blocks of code to be executed.
* **switch statement** – use this statement to select one of many blocks of code to be executed.
* **for loop** – use this statement to loop through a block of code a number of times.
* **while loop –** use this statement to loop through a block of code while condition is true.
* **do…** **while –** use this statement to loop through the block of code once, then continue to loop while condition is true.
* **break –** use this keyword to break the execution of a loop prematurely.
* **continue –** use this keyword to break the current iteration of a loop.

In the introductions to JavaScript you were given a program listing and the task was to type it in, make it work and understand how it works. In the real world of software development you will not be given the code you need to type in – it is your job to understand what the code needs to be. Normally you will be given a specification for the program you are going to write. The specification is one deliverable from the analysis and design phase of the project. As a professional software developer your job is largely to write programs that meet specifications.

In the following tasks you will be given a program specification, but not the actual code for the program. Using the JavaScript commands above and any other commands you have learnt so far you are going to write programs that meet these specifications. If you will need commands you have not yet encountered these will be introduced along the way.

**Task 1 – The greeting program.**

The client requires a program that can greet people by their name.

Write a program that:

1. Asks you to enter the first name of a person
2. Asks you to enter your last name of that person
3. Displays a message saying “Hello” to that person

Where the first name is “Bart” and the second name is “Simpson” the program should display:

Hello Bart Simpson

**Hints**

* Write your program in Notepad
* Save it as greeting.html in your public.html folder
* View/run the file in a browser
* Look back at previous tasks to see how we can get input from keyboard, or read online resources about the JavaScript function **prompt**
* Look back at previous tasks to see how we can display output on the screen, or read online resources about the javascript commands: **document.getElementByID**(“id”).**innerHTML** or **alert**

Here is a very simple program to help you to get started:

<!DOCTYPE html>

<html>

<body>

<script>

var input=prompt("Type something","");

alert(input);

</script>

</body>

</html>

**Task 2 – The adding program, version 1.**

The client requires a program that can add two whole numbers together.

Write a program that:

1. Asks you to enter a whole number (first number)
2. Asks you to enter another whole number (second number)
3. Displays the actual addition sum and the total of the numbers

Where the first number is 5 and the second number is 3 the program should display:

5 + 3 = 8

**Hints**

* Write your program in Notepad
* Save it as adding1.html in your public.html folder
* View/run the file in a browser
* **New Command**: JavaScript treats any input (from the keyboard) as a string characters; this could be letters or numbers. JavaScript cannot tell the difference between the number 53 and the character string “53”. For your program to work correctly you will need to use the new command: **parseInt.** ParseInt is a function that converts a string to an integer (whole number). The command takes the form:
  + *aVariable = parseInt(anotherVariable)*
  + So, where anotherVariable=”53” (the characters) aVariable will equal 53 (the number)

**Task 3 – The adding program, version 2.**

The client who requested the adding program now needs to be able to add not two, but ten numbers.

Write a program that:

1. Asks you to enter a whole number (first number)
2. Asks you to enter a whole number (second number)
3. Asks you to enter a whole number (third number)
4. Asks you to enter a whole number (fourth number)
5. Asks you to enter a whole number (fifth number)
6. Asks you to enter a whole number (sixth number)
7. Asks you to enter a whole number (seventh number)
8. Asks you to enter a whole number (eighth number)
9. Asks you to enter a whole number (ninth number)
10. Asks you to enter a whole number (tenth number)
11. Displays the total of all of the numbers

Where the numbers entered are 2,4,6,8,10,12,14,16,18,20 the program should display:

The total is: 110

**Hints**

* Write your program in Notepad
* Save it as adding2.html in your public.html folder
* View/run the file in a browser
* Use a loop e.g., **for loop**

**Task 4 – The adding program, version 3.**

The client who requested the adding program now needs to be able to add not two or ten, but any amount of numbers.

Write a program that:

1. Asks you to enter a whole number
2. Keeps asking for numbers until the word **exit** is typed instead of a number
3. Displays the total of all of the numbers

Where the input is: 1,2,3,4,5,exit the program should display:

The total is: 15

**Hints**

* Write your program in Notepad
* Save it as adding3.html in your public.html folder
* View/run the file in a browser
* Use a loop e.g., **do...while**
* Using a **for** loop will not work – can you figure out why?
* Use an **if** statement to check whether or not **exit** has been entered

**Stretch and Challenge**

These challenges are about making sure that the user has entered appropriate data into your program. This is known as **validation**.

**Challenge One**

Run the greeting program – do not enter a first name or a last name, just click OK each time you are asked to enter something. The program should just display **Hello**.

The client who requested the greeting program likes what you have done, but is not entirely satisfied. They have noticed that if you do not enter a first or last name the program still tries to greet you. Where a program does not accept empty input this is sometimes known as **required** or **mandatory** data. The client would like the program changed so that the first name and last name are required.

1. Asks you to enter the first name of a person – when a first name is not entered tells the user that the first name is required and asks for the first name again
2. Asks you to enter your last name of that person – when a last name is not entered tells the user that the last name is required and asks for the last name again
3. Displays a message saying “Hello” to that person

**Challenge Two**

Run the adding program version one – do not enter any numbers, just click OK each time. The program should display **NaN + NaN = NaN**

* Use online resources to discover what NaN means in JavaScript
* Modify the program so that if the input is left empty it asks the user again to enter a number

**Challenge Three**

If you have completed challenge two your adding version one program will not accept empty input. However, what happens is you type in letters instead of numbers? Does your program still work? Can you fix this?

**Challenge Four**

Look at all of the other programs you have written – improve these by using the validation techniques you have learnt in the previous challenges