Exercise 12b – Flow of Control

Objectives

In this exercise, you will investigate JavaScript flow of control statements, including branching and looping constructs.

Overview

This exercise is broken down into five separate sections, allowing you to explore flow of control statements in JavaScript.

This exercise is scheduled to run for **30 mins**.

Exercise Set Up

1. In VSCode, open the file called **FlowOfControl.html** from the **Starter** folder inside **Ex03** of the **Exercises** folder.

You will note that there are commented areas provided for each section of the exercise.

2. If you have rebooted your computer or stopped *live-server* from running since the last exercise, then use VSCode's integrated terminal to navigate to the **Exercises** folder and run the command **live-server**.
3. Open your browser (if it doesn't automatically) and navigate to **http://localhost:8080** and follow the path to open **Ex03→Starter→ FlowOfControl.html**.

Part 1 – if else statements

4. Under the comment //Part 1 - if else statement, declare a variable called age and initialise it to be 15.

Now, we are going to add a simple branching if statement. The if is one of the building blocks of programming, so it is essential we understand it. In this statement, we will check the age variable against a logical condition and let the code decide what it will do next.

5. Enter the following code to create the if below the variable declaration:

```
if (age <= 17) {
            console.log("Underage");
} else {
            console.log("Over 18");
}
```

6. Save the page and observe the output on the console of the developer tools (F12).

You should see the value **Underage** in the console. The value we have passed in has delivered a Boolean true result.

7. Change the value to **42** and check that the output has updated.

Part 2 - if else if statements

We now have a simple **if** statement in place. It is time to take our code a step further and add an **else if** caveat into our code. The **else if** is a further logical check delivering a Boolean value. Additional checks are examined in order; when a **true** is evaluated, the program leaves the **if** statement, so the order is important.

8. The code from the previous part has been slightly amended and needs to be included under the comment for part 2.

```
if (age <= 17) {
            console.log("Underage");
} else if (/*Remove me and insert your code here*/) {
            console.log("Insurable");
} else {
            console.log("out of range");
}
```

The **else if** statement is going to check if the age variable is between **18** and **65**, so we will need to use an **and** statement to achieve this.

9. Replace the code in the **else if** statement to achieve the desired result. Refer back to your notes or ask your instructor for help if you get stuck.
10. Save the page and observe the output in the browser.
11. Test that your code works by setting age to the following values:
    a) 10
    b) 50
    c) 80

Part 3 – if statements using short circuit

You have just successfully used an **and** operator, so let's give the **OR** a whirl. When we discussed relational operators in the course, we discussed that **||** was a short circuit operator. Consider that an **or** statement returns **true** *when one or all of its conditions are met*. If we use a short circuit operator within an **if** statement as soon as a Boolean **true** value is reached, the statement will exit. This is very useful as a programmer and can often mean the elimination of a number of additional if else sections.

Beneath the Part 3 comment please add the following code:

```
let value;
if (value === undefined || value === null || value == "") {
      console.log("value is mandatory");
      console.log(value);
}
```

12. Save the file and observe the output in the browser.

You should see the "**value is mandatory**" message and that value is **undefined** – pretty much as expected considering its undefined nature!

We are going to change the value of **value** to test what will happen.

13. Note down what you expect to happen before you test it:

Value	Result
Null	
" "	

Part 4 – for loops

Code needs to be used repetitively. Very often, you need to achieve the same activity more than once and loops are the most effective way of doing this. In this part of the exercise, you will investigate the for loop.

The for loop has three arguments: the counter, the condition and the iterator. You are going to code in a simple for loop, where the following properties need to be set:

parameter	value
counter	Variable name i set to 1
condition	While i is less than 10
Iterator	Each loop must add 1 to the value of i

14. Enter the following code, amended appropriately to achieve this:

```
for (counter; condition; iterator) {
            console.log(i * 1);
}
```

15. How many times do you expect the loop to execute? _____
16. Save the file and observe the output in the browser to check your assumptions.

Part 5 – while loops

for loops are best used when there is a *fixed number of iterations* that are required. The **while** loop is very useful when the end is not known. It can be used for iterator-based logic but does require a variable declaration external to the loop.

17. Enter the following code under the Part 5 comment.

```
let counter2 = 2;
let loopCount = 0;
while (counter2 < 100) {
      counter2 = ; //square of counter2
      //code to increment the loopCount by 1
      console.log("total : " +  counter2);
      console.log("loopCount : " + loopCount);
}
```

18. Replace the 2 comments with suitable code.

You need to provide the square of counter 2 and add code to increment the loop counter by 1. Call your instructor if you need help.

19. Once you have resolved this, save the code and observe the output in the browser.

Further activities

Only attempt the further activities, if there is sufficient time remaining. Your instructor can help you with these if you need it, but we hope you can start ranging out on your own.

1. Check what happens when you enter a value that does not match any of the or short circuit you created in Part 3.
2. Alter the for loop you created in Part 4 of the exercise to count down rather than up.