COURSEWARE

Iteration

Contents

Overview

Iteration and loops allow us to loop over a block of code until a condition is met.
If the condition is not satisfied, the code will terminate out of its current execution.
Some of the common loops that are used within JavaScript include:

- For Loop
- While Loop
- Do While Loop
- Switch Cases

Whilst all of these provide somewhat of the same functionality, their implementation and executions differ from one another from syntax to condition execution.

It's important to remember that a loop might never run; if the condition is never satisfied the code is never executed.

Tutorial

For Loop

A for loop utilises a *counter* until a condition is met. It has three main features; ***the initialisation, the condition, and the iterator***. The general syntax is as follows:

```
for ([initial - expression]; [condition]; [iterator]) {
  statement;
}
```

Below is an example in which the variable `'i'` is incremented at each iteration.

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

Lets break this down...

1. Declare a variable `i` and initialise it to 0.
2. Set the condition to execute the loop *until* i is **not** < 10.
3. Increment the value of `i`
4. As the condition is satisfied, we are going to print out the value of `i` to the console
5. Repeat until the condition is no longer satisfied i.e. when `i=10`;

While Loop

While loops primarily work with boolean values; it will continue to execute until the condition is *no longer* `true`. The general syntax is as follows:

```
while (condition) {
  statement;
}
```

An example of this is below:

```
let puppies = 0;
let notEnoughPups = true;

while (notEnoughPups) {
  console.log("Another pup");
  puppies++;

  if (puppies > 238) {
    notEnoughPups = false;
  }
}
console.log("Oh so many puppies!");
```

Lets go through the code line by line:

1. Declare and initialise `puppies` to 0
2. Declare and initialise `notEnoughPups` to true
3. While `notEnoughPups` is true:
    - Print to the console and increment the value of `puppies`
    - Check whether the value of `puppies` is greater than 238
    - If the condition is satisfied then set `notEnoughPups` to false
    - If not, repeat again
4. The while loop will repeat 237 times before the final console.log() statement is executed.

It's important to remember that the while loop might not run at all, if the condition is not satisfied.

Do While Loop

Do While Loop is similar to a while loop however there is one major difference - we have the assurance that the code will run **at least once** given that the condition is checked *after* the statement. Therefore, even if the condition evaluates to false there will be at least one outputted value.
The general syntax is:

```
do {
  statement;
} while();
```

In practise:

```
let x = 0;
let canIRun = false;
do {
  x++;
  console.log(x);
} while (canIRun);
```

In this case:

1. Variable `x` is declared and initialised to 0
2. Boolean condition `canIRun` is declared and initialised to false to see the affects of the loop
3. Inside our `do` block - we increment the value of `x` and log it to the console.
4. Check if our condition in the `while` block is satisfied i.e. true

Note: Even though our boolean condition is not met; we still get an output of 1. It's important to note that in a ***do while loop*** the code will execute *at least* ONCE because our condition is checked AFTER the statement.

What will happen if we changed the boolean condition to true? What will the output be?

Switch Case

A Switch Statement evaluates an expression matching a case label and executes statements associated with that case. These can be effective as opposed to having a large if-else statement.
The general syntax is:

```
switch (expression) {
  case 0:
    statement;
  case 1:
    statement;
  case 2:
    statement;
  default:
    statement;
}
```

When an expression matches a case label, control is passed to that case. This will continue to execute statement until flow of control is terminated using either **break** or **return**, alternatively **continue** can be used.

If no case is matched to the expression, then control passes to the **default** case.

Break - Used to terminate a loop or switch case statement entirely and transfers control to the following statement.

Continue - Can be used to restart a loop or statement, it terminates the inner most loop and continues execution for the next iteration.

Return - Terminates the execution of the program and specifies a value to be returned.

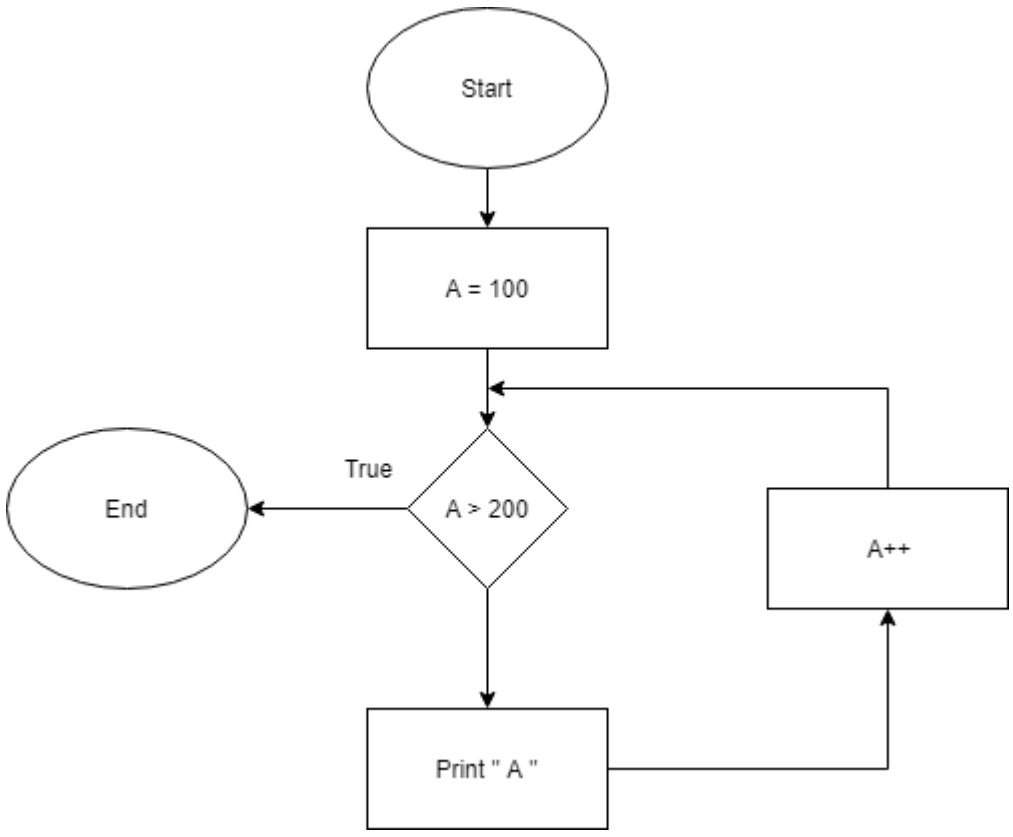
Let's look at an example:

```
let num = 5;
switch (num) {
  case 4:
    console.log(`Value is 4`);
    break;
  case 5:
    console.log(`Value is 5`);
    break;
  case 6:
    console.log(`Well...`);
  case 7:
  case 8:
  case 9:
  case 10:
    console.log(`Value is greater than 5`);
    break;
  default:
    console.log(`Sorry can't find the range you're looking for...`);
}
```

In this case, we have declared our switch statement which checks for an expression to match our variable. If it is found then it returns the statements within the correct case label. If a terminator is omitted then it continues to run until it finds the next `break` statement.

Exercises

1. Recreate the following flowchart as a loop



▶ Solution

2. Recreate the following flowchart as a loop.

▶ Solution

3. Create a method that can print out the numbers 1-10 10 times each.

▶ Solution

4. If you have used a while loop at any point in these exercises, replace them with for loops.

▶ Solution for 1
▶ Solution for 2

5. Write a switch case statement which uses the current day as its expression and matches with the relevant case. Criteria:

    - Omit a `break` statement if it is a weekday, until the last day
    - Use a default case to handle an invalid range.

▶ Solution