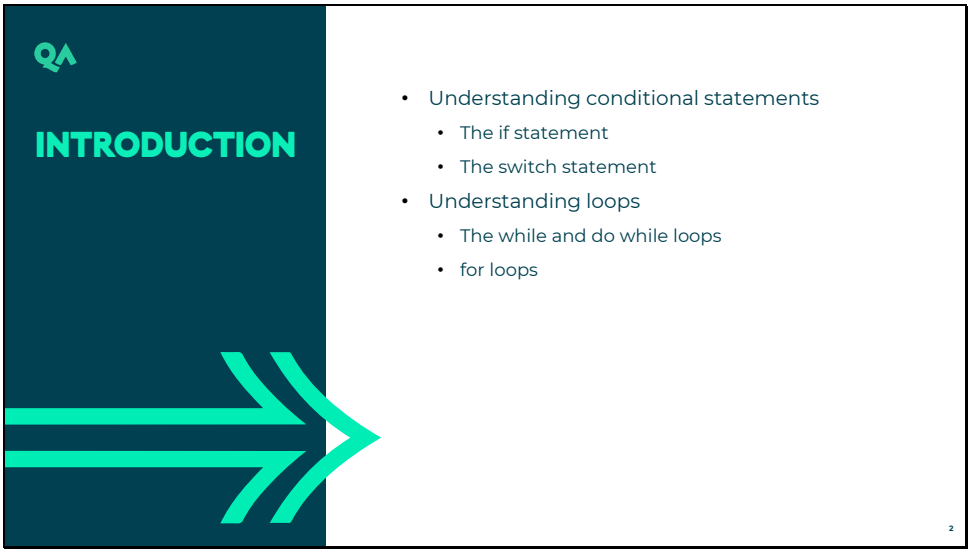
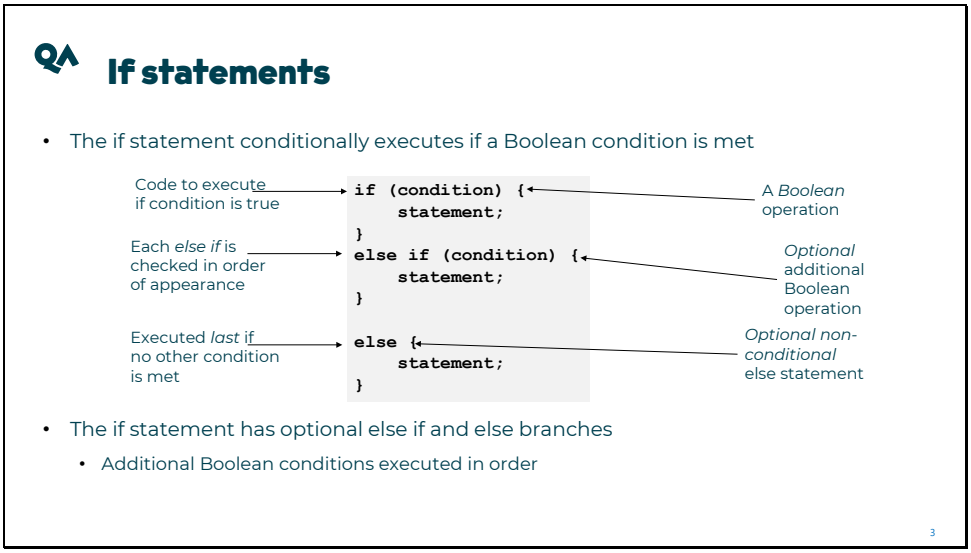
Slide 1


POWERING POTENTIAL
FLOW OF CONTROL
JavaScript Fundamentals

Slide 2


INTRODUCTION
Understanding conditional statements
The if statement
The switch statement
Understanding loops
The while and do while loops
for loops
2

Slide 3




JavaScript has a set of flow control statements, including `if`, `switch`, `while`, and `for`. The *condition* in each of these statements may be any expression that evaluates to either a Boolean or a number.

In the case of a numeric expression, the condition is considered to be `false` if the value of the expression is 0, and `true` in all other cases.

The *statement* shown in the syntax descriptions above may be any statement, including a statement block or a further flow control statement.

The curled braces are technically optional when there is only one line of code to be executed by the `if` statement. Therefore, the following code would be legal:

```
If (5 > 3)
      alert("true");
```

However, the following example would only consider the alert statement to be part of the conditional statements and not the variable declaration:

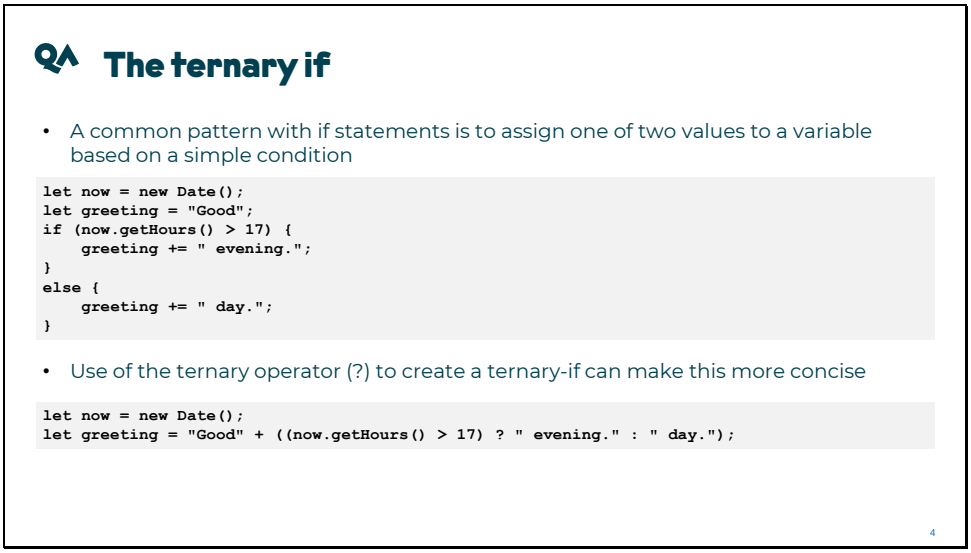
```
If (5 > 3)
      alert("true"); //conditional
      let x = 5; // not conditional
```

It is also possible to express an `if else` statement as a simple operator:

```
let result = (5 > 3) ? true : false;
```

This initialises the result variable to be a Boolean with a value of false.

Slide 4

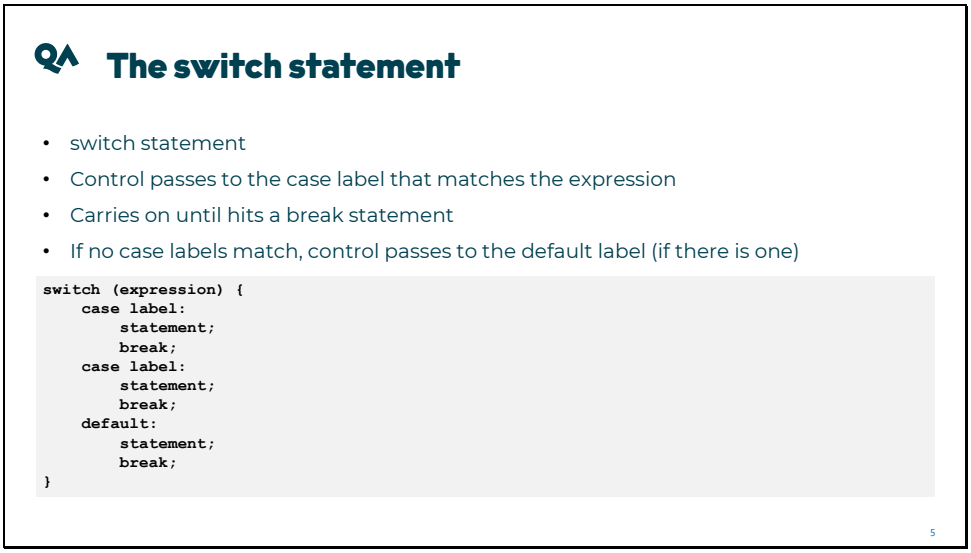



The ternary operator is an alternative to the standard `if` statement. Primarily used for variable assignments or when a standard if is too unwieldy, it has the following syntax:

```
test ? expression1 : expression2
```

***Test***
Any Boolean expression.
***Expression1***
An expression returned if *test* is **true**. May be a comma expression.
***Expression2***
An expression returned if *test* is **false**. More than one expression may be a linked by a comma expression.

Slide 5




The switch statement is useful when selecting an action from a number of alternatives. However, it is rarely used where a set of separate if statements make things more manageable and often more efficient. `Switch` compares the value of an integer test-expression with each of the case labels in turn. If a match is found, the statement(s) following the label are executed. Execution continues either until the end of the entire `switch` statement or until a `break` is encountered. If no match is found, control is passed to the statement(s) following the `default` label. The default label is optional – if there is no match and no default, the switch does nothing.

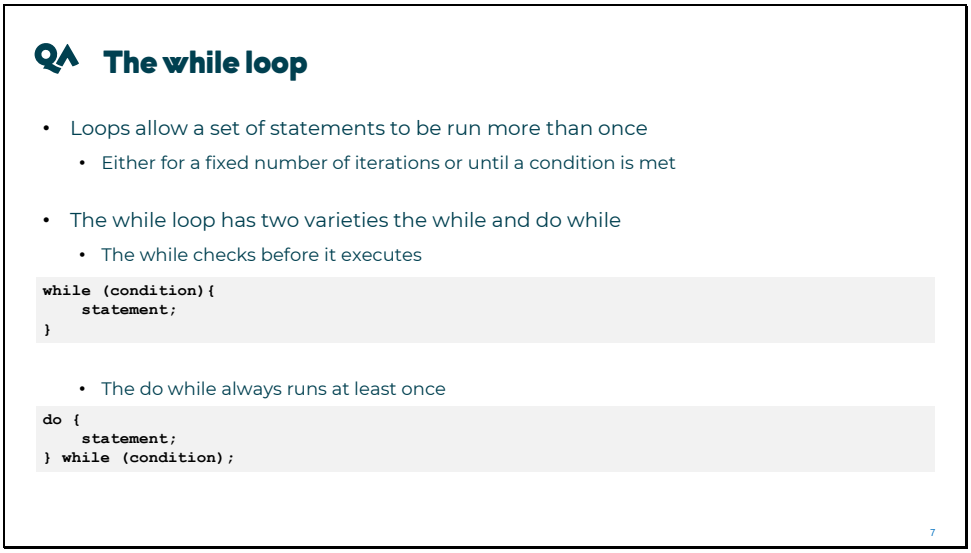
```
switch(weekDay)
{
        case "Mon":
        case "Tue":
        case "Wed":
        case "Thu":
        case "Fri":
            document.write("Go to work");
            break;
        case "Sat":
        case "Sun":
            document.write("Stay at home");
            break;
        default:
            document.write("Which planet are you on?");
            break;
}
```

Slide 6

## QuickLab 4a

- Experiment with conditional statements

Slide 7




All of the rules regarding brackets with the `if` statement apply to loops as well. Loops also provide two key loop control structures:

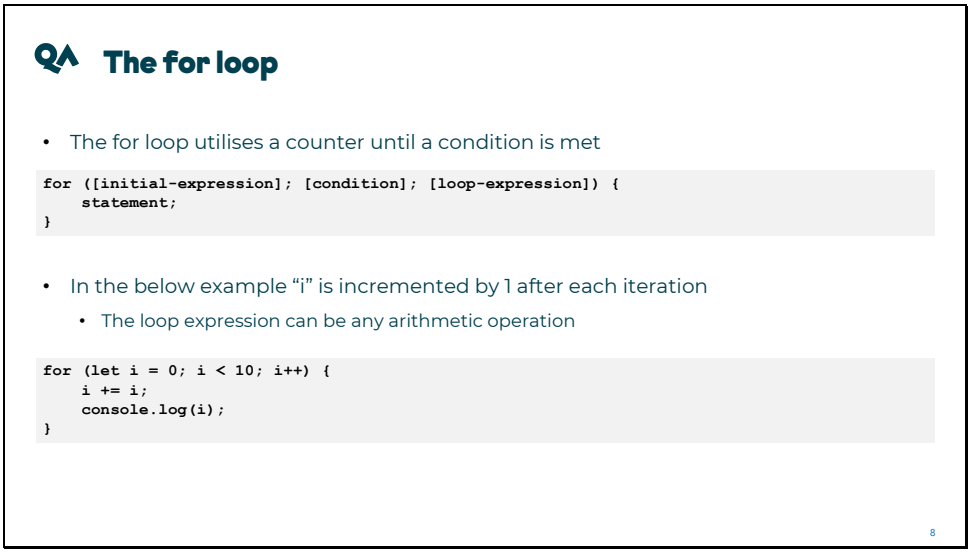
`break` – Exits the loop immediately.
`continue` – Jumps to the end of this iteration immediately.

So `while` and `do...while` loops repeat a statement (or statement block) repeatedly while the condition is `true`.

```
initial-expression;
while (condition)
{
      statement;
      loop-expression;
}
```

The `break` statement exits out of the loop immediately, and execution resumes at the statement immediately following the loop statement. The `continue` statement jumps to the end of the current iteration immediately; in the case of a `while` loop, execution will resume with evaluation of the *condition* expression at the top of the loop.

Slide 8




The `for` loop is a specialised form of a `while` loop;

```
for (initial-expression; condition; loop-expression)
{
	statement;
}
```

it is equivalent to writing:

```
initial-expression;
while (condition)
{
	statement;
	loop-expression;
}
```

All three of the expressions in a `for` statement are optional, and may be used in any combination; a `for` statement with none of the expressions present
(i.e. `for (;;)` ) creates an infinite loop.

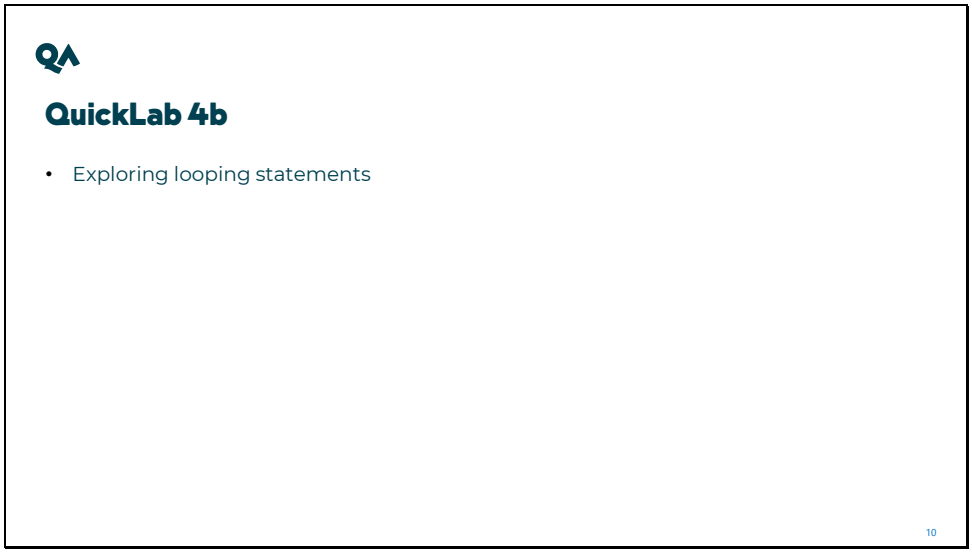
Using `continue` in a `for` loop causes execution to immediately jump to *loop expression* before then re-evaluating the *condition* and hence will effectively jump to the next iteration of the loop.

Slide 9

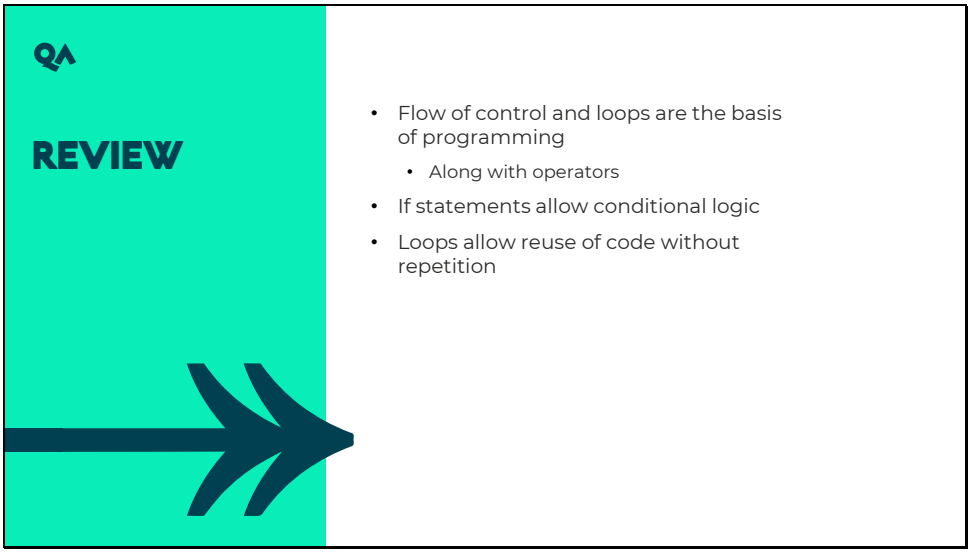
## Review

- Flow of control and loops are the basis of programming
  - Along with operators
- If statements allow conditional logic
- Loops allow reuse of code without repetition

Slide 10



Slide 11


REVIEW
• Flow of control and loops are the basis of programming
• Along with operators
• If statements allow conditional logic
• Loops allow reuse of code without repetition