

Flow of Control

PROGRAMMING WITH JAVASCRIPT

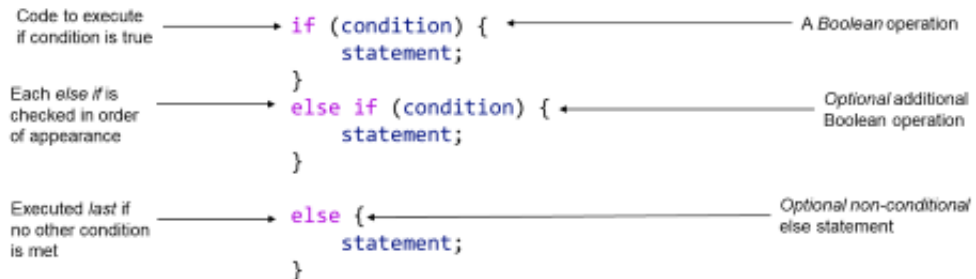


Introduction

- Understanding flow of control statements
 - The if statement
 - The switch statement
- Understanding loops
 - The while and do while loops
 - for loops

If statements

- The if statement conditionally executes if a Boolean condition is met



- The if statement has optional else if and else branches
 - Additional Boolean conditions executed in order

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
    var x = 5; // not conditional
  
```

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

```

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

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

The ternary if

- A common pattern with if statements is to assign one of two values to a variable based on a simple condition

```
let now = new Date();
let greeting = "Good";
if (now.getHours() > 17) {
  greeting += " evening.";
}
else {
  greeting += " day.";
}
```

- Use of the ternary operator (?) to create a ternary-if can make this more concise

```
let now = new Date();
let greeting = "Good" + ((now.getHours() > 17) ? " evening." : " day.");
```

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.

The switch statement

- switch statement
- Control passes to the case label that matches the expression
- Carries on until hits a break statement
- If no case labels match, control passes to the default label (if there is one)

```
switch (expression) {
  case label:
    statement;
    break;
  case label:
    statement;
    break;
  default:
    statement;
    break;
}
```

3

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;
}
```

The while loop

- Loops allow a set of statements to be run more than once
 - Either for a fixed number of iterations or until a condition is met
- The while loop has two varieties the while and do while
 - The while checks before it executes

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

- The do while always runs at least once

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

← Note the semi-colon

6

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.

The for loop

- The for loop utilises a counter until a condition is met

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

- In the below example "i" is incremented by 1 after each iteration
 - The loop expression can be any arithmetic operation

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

7

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.

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

Exercise

- Exploring statements