Extra notes - JS - Core - Part 2

- Dr Nick Hayward

A brief introduction to some of the core concepts for working with JavaScript.

Contents

Intro A few of the primary, core concepts for working with JavaScript. Many of these concepts are applicable to client-side design, web-stack mobile development, and web-stack desktop application development.

Functions and values So far, we've seen variables acting as groups of code and blocks, which act as one of the primary mechanisms for scope within our JS applications.

However, we can also use functions as values, effectively using them to set values for other variables, for example.

```
var a;

function scope() {
"use strict";
    a = 49;
  return a;
}

b = scope()*2;
console.log(b);
```

It's a useful and interesting aspect of the JS language, thereby allowing us to build values from multiple layers and sources.

More conditionals We've already briefly considered conditional statements using the `if` statement,

```
if (a > b) {
console.log("a is the best...");
} else {
console.log("b is the best...");
}
```

Switch statements in JS effectively follow the same pattern as `if` statements, but they are designed to allow us to check for multiple values in a more succinct manner. These statements enable us to check and evaluate a given expression, and then attempt to match a required value against an available `case` .

switch conditional The addition of `break` is important to ensure that only a matched case is executed, and then the application breaks from the switch statement. If not, execution after that case will continue. This is commonly known as **fall through** in programming. Be aware, however, that this may be an intentional feature of your code design as well. For example, we may wish to allow a match against multiple possible cases, therefore a premature break is not required within the code.

```
var a = 4;

switch (a) {
case 3:
  //par 3
  console.log("par 3");
  break;
case 4:
  //par 4
  console.log("par 4");
  break;
case 5:
  //par 5
  console.log("par 5");
  break;
case 59:
  //dream score
  console.log("record");
  break;
default:
  console.log("more practice");
}
```

ternary or conditional operator As a final point, there is also a more concise way to write our conditional statements using what is known as the **ternary** or **conditional** operator. It's best to consider this operator as a more concise form of the standard `if...else` statement. For example,

```
var a = 59;
var b = (a > 59) ? "high" : "low";
```

This is equivalent to the following standard `if...else` statement

```
var a = 59;

if (a > 59) {
  var b = "high";
} else {
  var b = "low";
}
```

References

- MDN
  - MDN - JS
  - MDN - JS Const
  - MDN - JS Data Types and Data Structures
  - MDN - JS Grammar and Types
  - MDN - JS Objects
- W3 - JS Object
- W3 - JS Performance