
Session 15

Mohamed Emary

April 28, 2024

1 Operators In JavaScript

1.1 Arithmetic Operators

Arithmetic operators in JS are: +, -, *, /, **, and %.

The % operator returns the remainder of a division, for example if $x = 5$ and $y = 2$, then $x \% y$ is 1. When the remainder is 0, it means that y is divisible by x .

When + is used with strings, it concatenates them, for example:

```
1 var x = "Hello" + "World"; // x is "HelloWorld"
2 var y = "Hello" + 1 + 2;    // y is "Hello12"
3 var z = 1 + 2 + "Hello";    // z is "3Hello"
```

1.2 Assignment Operators

Assignment operators in JS are: =, +=, -=, *=, /=, **=, and %=.

1.3 Comparison Operators

Comparison operators in JS are: ==, ===, !=, !==, >, <, >=, and <=.

```
1 var x = 5;
2 var y = 5;
3 console.log(x >= y); // true
4
5 x = "5";
6 console.log(x == y); // true
7 console.log(x != y); // false
8
9 console.log(x === y); // false
10 console.log(x !== y); // true
```

So what is the difference between == and ===?

2 CONDITIONAL STATEMENTS

- `==` is used to compare values, while `===` is used to compare values and types.
- `console.log(x == y);` is true because JS converts the string to a number if possible.
- `console.log(x === y);` is false because `===` does not convert the types.

1.4 Logical Operators

Logical operators in JS are: `&&`, `||`, and `!` (AND, OR, and NOT).

Explanation:

- `&&` is true if all conditions are true and false if at least one condition is false.
- `||` is true if at least one condition is true and false if all conditions are false.
- `!` is used to reverse the result, so if a condition evaluates to true, `!` will make it false.

```
1 var x = 5;
2 var y = 10;
3 console.log(x > 3 && y < 20); // true
4 console.log(x > 3 || y > 20); // true
5 console.log(!(x > 3));        // false
```

Logical operators are commonly used in to make decisions in JS using conditional statements.

2 Conditional Statements

2.1 If Statement

The `if` statement is used to execute a block of code if a condition is true.

```
1 var x = 5;
2 if (x > 0) {
3   console.log("x is positive");
4 }
```

The statement `console.log("x is positive");` will only be executed if `x > 0`.

2.2 Else Statement

The `else` statement is used to execute a block of code if the same condition is false.

```
1 var x = -5;
2 if (x > 0) {
3   console.log("x is positive");
4 } else {
5   console.log("x is negative");
6 }
```

2.3 Else If Statement

The `else if` statement is used to specify new conditions if the previous conditions are false.

```
1 var skill = "HTML";
2 if (skill == "CSS") {
```

2 CONDITIONAL STATEMENTS

```
3   console.log("CSS");
4 } else if (skill == "HTML") {
5   console.log("HTML");
6 } else if (skill == "JavaScript") {
7   console.log("JavaScript");
8 } else {
9   console.log("Another skill");
10 }
```

Here if `skill` is not `CSS` it checks if it is `HTML`, and if not, it checks if it is `JavaScript` and if not, it prints `Another skill`.

2.4 Nesting If Statement

A nested if statement is an if statement inside another if statement.

```
1  var x = 10;
2  var y = 20;
3  if (x == 10) {
4    if (y == 20) {
5      console.log("x is 10 and y is 20");
6    }
7  }
```

2.5 Switch Statement

The `switch` statement is used to perform different actions based on different conditions.

```
1  var day = 3;
2  switch (day) {
3    case 1:
4      console.log("Monday");
5      break;
6    case 2:
7      console.log("Tuesday");
8      break;
9    case 3:
10     console.log("Wednesday");
11     break;
12   default:
13     console.log("Another day");
14 }
```

The `break` statement is used to break out of the switch block, because JS will execute the next switch case if a `break` is not found.

Switch statement has a better performance than if-else statement.

2.6 Nested Switch Statement

A nested `switch` statement is a `switch` statement inside another `switch` statement.

```
1  var day = 3;
2  var month = 4;
```

3 LOOPS

```
3 switch (day) {
4   case 1:
5     console.log("Monday");
6     break;
7   case 2:
8     console.log("Tuesday");
9     break;
10  case 3:
11    switch (month) {
12      case 4:
13        console.log("Wednesday, April");
14        break;
15      case 5:
16        console.log("Wednesday, May");
17        break;
18      default:
19        console.log("Another Month");
20    }
21    break;
22  default:
23    console.log("Another day");
24 }
```

2.7 Falsey Values

Falsey values in JS are values that are considered false when evaluated in a boolean expression. They include: `false`, `0`, `""`, `null`, `undefined`, and `NaN`.

```
1 var x = "";
2 var y = "Mohamed";
3 console.log(x && y); // prints nothing
4 console.log(x || y); // Mohamed
```

The first `console.log` prints nothing because `x` is falsey and we are using the `&&` operator so both conditions must be true to execute the statement, while the second `console.log` prints Mohamed because `y` is truthy and we are using the `||` operator so only one condition must be true to execute the statement.

The `&&` stops evaluating with the first false value, while the `||` stops evaluating with the first true value.

3 Loops

Loops are used to execute the same block of code multiple times.

3.1 For Loop

The `for` loop is used to execute a block of code a number of times.

Syntax:

3 LOOPS

```
1 | for (initialization; condition; step) {
2 |     // code block to be executed
3 | }
```

Example:

```
1 | for (var i = 0; i < 5; i++) {
2 |     console.log(i);
3 | }
```

The loop will print the numbers from 0 to 4.

`i++` is the same as `i = i + 1`, `i += 1`.

To print even numbers from 0 to 10:

```
1 | for (var i = 0; i <= 10; i += 2) {
2 |     console.log(i);
3 | }
```

These two code blocks will print forever (infinite loop):

<pre>1 for (;;) { 2 console.log("Hello"); 3 }</pre>	<pre>1 for (var i = 0; i < 5;) { 2 console.log(i); 3 }</pre>
---	---

While this one will cause an error because of a missing `;;`:

```
1 | for (;) {
2 |     console.log("Hello");
3 | }
```

3.2 While Loop

The **while** loop is used to execute a block of code as long as a condition is true.

Syntax:

```
1 | while (condition) {
2 |     // code block to be executed
3 | }
```

Example:

```
1 | var i = 0;
2 | while (i < 5) {
3 |     console.log(i);
4 |     i++;
5 | }
```

The loop will print the numbers from 0 to 4.

3.3 Do While Loop

The **do while** loop is a variant of the **while** loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

3 LOOPS

Syntax:

```
1 do {  
2   // code block to be executed  
3 } while (condition);
```

Example:

```
1 var i = 0;  
2 do {  
3   console.log(i);  
4   i++;  
5 } while (i < 5);
```

3.4 Using Loops With HTML Elements

You can use loops to manipulate HTML elements.

In HTML:

```
1 <ul id="list">  
2 </ul>
```

In JS:

```
1 var list = document.getElementById("list");  
2 for (var i = 0; i < 6; i++) {  
3   if (i % 2 === 0) {  
4     var item = "<li class='red'>Item " + i + "</li>";  
5   } else {  
6     var item = "<li class='green'>Item " + i + "</li>";  
7   }  
8   list.innerHTML += item;  
9 }
```

You can even use the classes to style the elements.

```
1 .red {  
2   color: red;  
3 }  
4  
5 .green {  
6   color: green;  
7 }
```

- Item 0
- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

Figure 1: Final Result

4 Summary

- Operators in JS include arithmetic, assignment, comparison, and logical operators.
- Arithmetic operators include `+`, `-`, `*`, `/`, `**`, and `%`.
 - `+` is used to concatenate strings.
 - `%` returns the remainder of a division.
- Assignment operators include `=`, `+=`, `-=`, `*=`, `/=`, `**=`, and `%=`.
- Comparison operators include `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, and `<=`.
 - `==` is used to compare values, while `===` is used to compare values and types.
- Logical operators include `&&`, `||`, and `!`.
- Conditional statements include `if`, `else`, `else if`, and `switch`.
 - `if` is used to execute a block of code if a condition is true.
 - `else` is used to execute a block of code if the same condition is false.
 - `else if` is used to specify new conditions if the previous conditions are false.
 - `switch` is used to perform different actions based on different conditions.
 - * `break` is used with `switch` to break out of a case, otherwise JS will execute the next cases till the end.
 - * `default` is used with `switch` to execute a block of code if no case is true.
- Falsey values are values that are considered false in a boolean expression, they include `false`, `0`, `""`, `null`, `undefined`, and `NaN`.
- Loops include `for`, `while`, and `do while`.
 - `for` is used to execute a block of code a number of times.
 - `while` is used to execute a block of code as long as a condition is true.
 - `do while` is used to execute a block of code once before checking the condition.