JavaScript Switch Statement

‹ Previous Next ›

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

The JavaScript Switch Statement

Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {
    case x:
        code block
        break;
    case y:
        code block
        break;
    default:
        code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Example

The getDay() method returns the weekday as a number between 0 and 6.

(Sunday=0, Monday=1, Tuesday=2 ..)

This example uses the weekday number to calculate the weekday name:

```
switch (new Date().getDay()) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    case 4:
        day = "Thursday";
        break;
    case 5:
        day = "Friday";
        break;
    case 6:
        day = "Saturday";
}
```

The result of day will be:

```
MondayMonday
```

Try it Yourself »

The break Keyword

When JavaScript reaches a **break** keyword, it breaks out of the switch block.

This will stop the execution of more code and case testing inside the block.

When a match is found, and the job is done, it's time for a break. There is no need for more testing.

> A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

The default Keyword

The **default** keyword specifies the code to run if there is no case match:

Example

The getDay() method returns the weekday as a number between 0 and 6.

If today is neither Saturday (6) nor Sunday (0), write a default message:

```
switch (new Date().getDay()) {
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
        break;
    default:
        text = "Looking forward to the Weekend";
}
```

The result of text will be:

```
Looking forward to the WeekendLooking forward to the Weekend
```

Try it Yourself »

The **default** case does not have to be the last case in a switch block:

Example

```
switch (new Date().getDay()) {
    default:
        text = "Looking forward to the Weekend";
        break;
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
}
```

Try it Yourself »

If default is not the last case in the switch block, remember to end the default case with a break.

Common Code Blocks

Sometimes you will want different switch cases to use the same code.

In this example case 4 and 5 share the same code block, and 0 and 6 share another code block:

Example

```
switch (new Date().getDay()) {
    case 4:
    case 5:
        text = "Soon it is Weekend";
        break;
    case 0:
    case 6:
        text = "It is Weekend";
        break;
```

```
    default:
        text = "Looking forward to the Weekend";
}
```

Try it Yourself »

Switching Details

If multiple cases matches a case value, the **first** case is selected.

If no matching cases are found, the program continues to the **default** label.

If no default label is found, the program continues to the statement(s) **after the switch**.

Strict Comparison

Switch cases use **strict** comparison (===).

The values must be of the same type to match.

A strict comparison can only be true if the operands are of the same type.

In this example there will be no match for x:

Example

```
var x = "0";
switch (x) {
    case 0:
        text = "Off";
        break;
    case 1:
        text = "On";
        break;
    default:
        text = "No value found";
}
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 »

‹ Previous Next ›