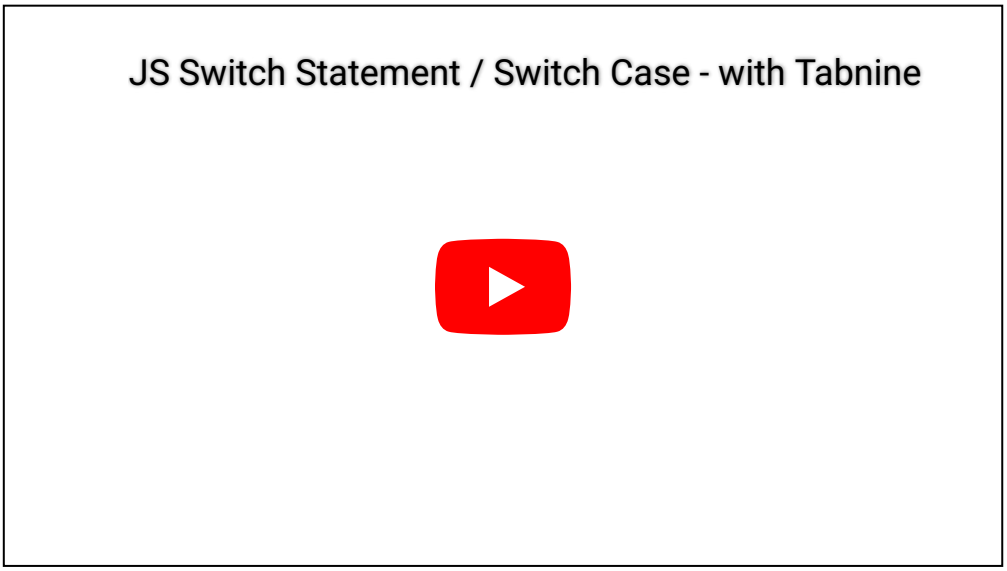
How to Use switch Statements in JS

7 min read I December 6, 2020

A switch statement is a form of a conditional statement. It will execute a code block if a certain condition is met.

Switch statements are used when you want different actions or results to occur for different situations. They are similar to using **if...else** statements to handle multiple alternative outcomes in the code.

The switch statement matches an expression's value to **cases** by order. It will either execute the code for a matching case, or an *optional* **default** code if no matches are found.



Basic switch example

Below is a basic switch statement example:

```
const numOfItems = 2;
switch (numOfItems) {
  case 3:
    console.log('Get one free item!!');
    break;
  case 2:
  case 1:
    console.log('The fourth item is free!!');
    break;
  default:
    console.log('Today only, 3 + 1 free!!');
}
// Expected output: The fourth item is free!!
```

In the example above, const *numOfItems* represents the number of items in a user's cart, two items in this case.

The switch statement uses the **strict equality** operator (===) to evaluate each condition in turn. Conditions are defined using the case *keyword*. The switch statement begins with the top condition, checking *numOfItems* against the value 3. Since

following code being executed until a break statement is encountered. In the example above while case 2 has no associated code defined, the code in *case 1* is executed as it occurs after the matching case has been encountered. This results in "*The fourth item is free!!*" being printed to the console. The break statement that follows *stops* the procedure; if omitted, the program will continue to execute the following statements until it encounters either the next break statement or the *end* of the switch statement. In the example above, if the second break was omitted the text "*Today only, 3 + 1 free!!*" would have also been printed to the console.

It is possible to use the same result code for multiple cases, as seen in the example above – both cases 2 and 1 print the text 'The fourth item is free!!' to the console.

Any other value, (e.g. 0, 'x', and even 4) will result in the default case executing. In these cases, the final **console.log()** statement will execute, printing "*Today only, 3 + 1 free!!*" to the console.

Note: Passing this code a value of 4 creates a problem in how the cases are evaluated. This issue is explored below in the section ***Using switch with conditions other than strict equality***.

Before we explore this problem, let's take a look at the syntax for switch statements.

Syntax – switch Statements

The switch statement receives an *expression*, which is evaluated once. It then checks for matching values in the different **cases** provided. When a match is found, the *statements* associated with that case are executed. If no match

provided. When a *default* clause is included, the statements associated with the *default* clause are executed when no match is found.

```
switch (expression) {
    case value1:
        statement(s) to execute
        break;
    case value2:
        statement(s) to execute
        break;
    case valueN:
        statement(s) to execute
        break;
    default:
        statement(s) to execute
}
```

expression: an expression, which will be evaluated once at the beginning of the process. The value of this expression is compared against the following cases.

case values (1 thru N): each case needs to have a specified

strictly compared (===) to the expression, and when there is a match the associated statements for the case are executed.

break (*optional*): the break statement stops the execution process and *exits* the switch statement.

Once a **case value** comparison returns **true**, *all of the following statements are set for execution*. The break statement prevents all of the statements from being executed, stopping execution when the break statement is encountered. By placing this statement at the end of each set of statements for a case, you can ensure only the statement(s) associated with the specific matched case are executed.

default (*optional*): this statement will be executed if all other comparisons returned **false**. This clause is usually found at the bottom of a switch statement, but it is not mandatory.

Note: If the default clause is not the last clause in the switch statement, make sure to add a break statement at the end of the statements intended to be executed for the *default* case.

Using switch with conditions other than strict equality

If you wish, you can use comparison operators other than *strict equality*. The comparison expression used must evaluate to a *Boolean* value (therefore logical operators can be applied as well).

For example, let's say that in the example above we want to print the text "*Get one free item!!*" to the console whenever the number of items is equal to or greater than 3. A first iteration of this approach might look like the following: *case >= 3:*

Unfortunately, this code will result in an error – Uncaught SyntaxError: Unexpected token '>=' .

Parentheses will not help in this situation, as the issue lies in defining the left-hand-side of the >= operand. To resolve this, we need to instead provide a complete comparison expression that can evaluate to a *Boolean* value.

The following example demonstrates how to achieve this type of relative comparison:

```
1.  const numOfItems = 2;
2.  switch (true) {
3.    case (numOfItems >= 3):
4.      console.log('Get one free item!!');
5.      break;
6.    case (numOfItems >= 1):
7.      console.log('The fourth item is free!!');
8.      break;
9.    default:
10.     console.log('Today only 3 + 1!!');
11. }
12. // Expected output: The fourth item is free!!
```

In the code above, whenever the case's expression evaluates to **true**, there is a match and the case's statement(s) will be executed. In the case of *numOfItems = 4*, both of the first two cases return **true**. However, due to the break statement at the end of the first case, the second case will not be read and its statement will not be executed.

switch vs if...else

switch statements and if...else statements are in many ways functionally equivalent. Both types of statements can produce similar results, and in many cases programmers simply choose

are many conditions to choose from, but the choice of statement is still largely a matter of personal preference.

The above switch statement rewritten, as an if...else statement, would match the example below:

```
if (numOfItems >= 3) console.log('Get one free item!!');
else if (numOfItems >= 1) console.log('The fourth item is free!!');
else console.log('Today only 3 + 1!!');
```

switch with no break statements example

Let's say you are writing a code that issues prizes for a contest in a game. In this game, the first place winner gets all of the prizes, while each successive lower ranking results in one less prize being granted. Code to implement this could resemble the following:

```
const rank = 1;
switch (rank) {
  case 1:
    console.log('Invisibility ring, ');
  case 2:
    console.log('Healing potion, ');
  case 3:
    console.log('Treasure chest, ');
  default:
    console.log('5 Gold bars');
}
// Expected output:
// Invisibility ring,
// Treasure chest,
// Healing potion,
// 5 Gold bars
```

Note that the example above uses **no** break statements. As

value matches the *expression*, all of the following statements are executed. According to the code above, a user who came in 4th or after will only receive the *5 gold bars* (due to the default clause), while by comparison a user in 2nd place will receive the *healing potion*, the *treasure chest*, and the *5 gold bars*.

Related Articles

JavaScript – How to Use if...else Statements

JavaScript – A Short Introduction to Loops

JavaScript – How to Use Classes

Related Posts

How to Change CSS Using JavaScript

7 min read I December 21, 2020

Cascading Style Sheets (CSS) is used to design the layout of a webpage. CSS is a set of structured rules describing the layout

How to Use the Assignment Operator in JavaScript

2 min read | November 30, 2020

The Assignment operator is used to assign a value to a variable. Assignment (=) The assignment operator simply assigns the value of the right operand to

How to Use the for...in Statement in JavaScript

3 min read | December 2, 2020

The for...in statement is used to loop through an object's properties. Basic for...in loop example The following code demonstrates a basic example using the for...in statement