GOMYCODE

The Switch Statement

Long chains of if-else statements can become monotonous very quickly when handling multiple scenarios. Luckily for us, there's an abbreviated version for this: the `switch` operator.

We have a variable inside the `switch`. This variable may have values. We jump to the `case` label for which the skipped argument is equal to the label value. Then we start executing the code.

Otherwise there's always our `default` which will handle any other scenario that is not covered by our cases.

```
// 1_Defining the function

function decodeColor( code ) {
    switch( code) {
        case 1:
                console.log( 'Red' );
                break;
        case 2:
                console.log( 'Yellow' );
                break;
        case "x":
                console.log( 'Green' );
                break;
        default:
                console.log( 'Unknown code' );
        }
}

// 2_Calling the function

decodeColor(1) // prints Red
decodeColor(2) // prints Yellow

decodeColor("x") // prints Green

decodeColor(3) // prints Unknown code
```

The Switch Statement

Ahaa! You must’ve noticed that we used a `break` there. Now why is the break so important?

Simply put, if there was no break at the end of the code segment belonging to a label, execution would continue on to the next code line. It does not matter that it is now under another label. Code just continues executing everything after the first valid scenario. Let’s try it in the code box, we will execute the same function as before but without the break(s).

This is why we have a `break` statement at the end of each code segment belonging to a label. The `break` statement jumps out of the closest `switch` statement, and our program continues execution after the `switch` statement.

```
// 1_Defining the function

function decodeColor( code ) {
    switch( code) {
        case 1:
                console.log( 'Red' );
        case 2:
                console.log( 'Yellow' );
        case "x":
                console.log( 'Green' );
        default:
                console.log( 'Unknown code' );
        }
}

// 2_Calling the function
```

```
decodeColor(2) // prints Yellow
// Green
// Unknown code
```

The Switch Statement

We can omit the `break` if we're going to use the `return` statement instead.

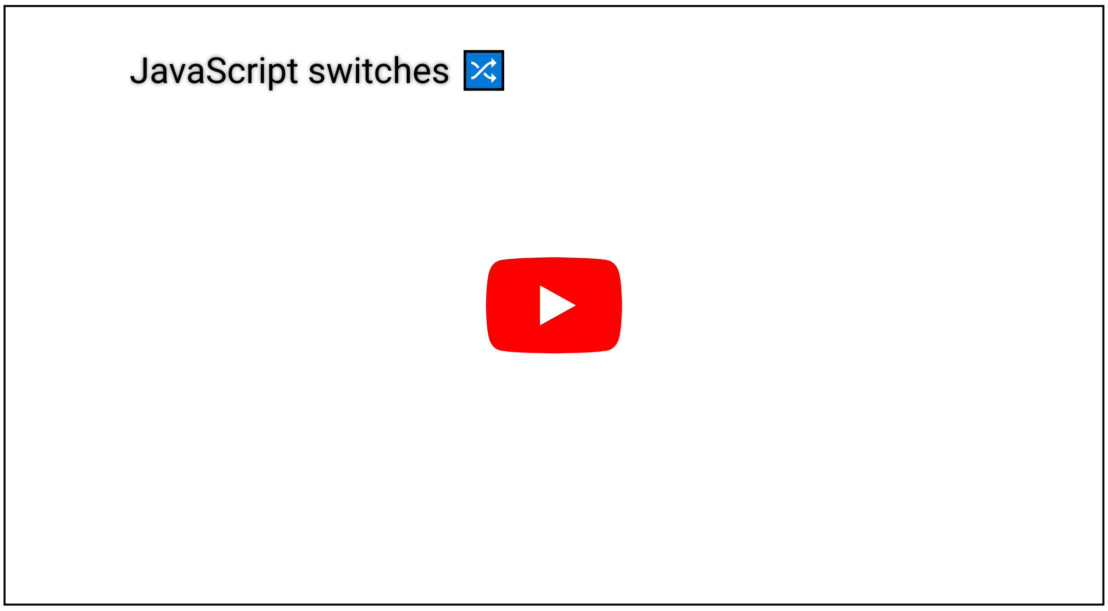
The `return` statement will only execute what's valid for a given scenario/case and then exits the `switch` statement without executing the rest of the lines inside of it (the switch statement and the rest of its cases).

```
// 1_Defining the function

function decodeColor( code ) {
    switch( code) {
        case 1:
                return 'Red';
        case 2:
                return 'Yellow';
        case "x":
                return 'Green';
        default:
                return 'Unknown code';
        }
}
// 2_Calling the function

console.log(decodeColor(2)) // prints Yellow
/*
PS: we now have to explicitly console.log the returned value of
our function to "see" it.
Merely executing the function won't print anything on the
console (although everything is working fine and as it should)
*/
```

The Switch Statement



‹ Previous

next ›