Switch statements

Another way of executing selective instructions is done through the switch statements.

There is an abbreviated version for a long chain of if-else statement: the `switch` operator. Switch is like a telephone operator. It puts you through some code in case the correct value is stored in your variable. There is also a default line, saying “the number you have dialed is invalid”.

```
function logLampColor( state ) {
  switch( state ) {
    case 1:
      console.log( 'Red' );
      break;
    case 2:
      console.log( 'Yellow' );
      break;
    case 3:
      console.log( 'Green' );
      break;
    default:
      console.log( 'Wrong lamp state' );
                }
}

logLampColor( 1 );
```

Try this code example out. Play around with it. Check what happens if you remove the `break` commands. Check what happens if you call this function with other arguments.

Now that you got a feel for the switch statement, let me explain it. You have a variable inside the switch. This variable may have values. We jump to the `case` label for which `state` is equal to the label value.

Then we start executing the code.

Why the `break` is so important?

If there was no `break` at the end of the code segment belonging to a label, execution would continue on the next code line. It does not matter that it is now under another label. Code just jumps through the labels.

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 your program continues execution after the `switch` statement.

We have two reasons not to include a `break` statement after a label:

- we use a `return` statement instead,
- we intentionally fall onto the next label (discouraged).

Example:

```
function getLampColor( state ) {
  switch( state ) {
    case 0:
    case 1:
      return 'Red';
    case 2:
      return 'Yellow';
    case 3:
      return 'Green';
                }
}
console.log(getLampColor( 0 ));
console.log(getLampColor( 1 ));
```

In this example, both cases `0` and `1` return `‘Red’`.