

Switch

Justin Besteman

December 30, 2016

Synopsis

Switch statements are a powerful tool for a coder to test multiple conditions.

Why

If a coder is testing only a few conditions then an if, else, or else if are perfect tools to use. But what if there are a large number of things to test? You could still use these tools but you better get some ice to cool off your fingers when you are done! A switch statement is a way to reduce the amount code that needs to be typed in such cases, and as a bonus the code is easy to understand at a glance.

Syntax

The switch statement is structured like a train track system where the rails switch (see what I did there?) based on particular cases. With this tool, you have a switch and cases that will be executed based on the condition of the switch. After each case, you must use the keyword break because if you don't then bad things happen to good people. A switch statement is block level code so without the break it would just continue and execute all the further conditional code for other cases after the first met case code was read.

Advanced

Switch statements have the ability to “Stack cases on top of each other.” (My mentor, Sean, taught me that!) This means you can have more than one condition or case that can be met for that code to execute. At the end of the switch statement, a coder may put the keyword “default” which is triggered if none of the cases are met.

Examples

```
1
2 // ----- Example 1 -----
3
4 var traffic_light = "green";
5
6 switch (traffic_light) {
7
8     case "green":
9         console.log("Go");
10        break;
11    case "yellow":
12        console.log("Speed up or slow down");
13        break;
14    case "red":
15        console.log("Stop, Stop, Stoopppp!!!");
16        break;
17
18 } // End of Switch Statement
19
20 // ----- Example 2 -----
21
22 var user_pick = 3;
23
24 switch (user_pick) {
25
26     case 1:
27         console.log("User 1");
28         break;
29     case 2:
30         console.log("User 2");
31         break;
32     case 3:
33         console.log("User 3");
```

```

34     break;
35 case 4:
36     console.log("User 4");
37     break;
38 case 5:
39     console.log("User 5");
40     break;
41
42 } // End of Switch Statement
43
44 // ----- Example 3 -----
45
46 // This shows how cases can be stacked
47
48 var user_pick = 3;
49
50 switch (user_pick) {
51
52     case "one":
53     case 1:
54         console.log("User 1");
55         break;
56     case "two":
57     case 2:
58         console.log("User 2");
59         break;
60     case "three":
61     case 3:
62         console.log("User 3");
63         break;
64     case "four":
65     case 4:
66         console.log("User 4");
67         break;
68     case "five":
69     case 5:
70         console.log("User 5");
71         break;
72
73 } // End of Switch Statement
74
75 // This is powerful because if a user entered an
76 // integer or a string for their choice, this code
77 // would still work. We are able to stack both the
78 // integer and string cases.

```

```

79
80 // ----- Example 4 -----
81
82 // This shows the default keyword
83
84 var user_pick = 3;
85
86 switch (user_pick) {
87
88     case "one":
89     case 1:
90         console.log("User 1");
91         break;
92     case "two":
93     case 2:
94         console.log("User 2");
95         break;
96     case "three":
97     case 3:
98         console.log("User 3");
99         break;
100    case "four":
101    case 4:
102        console.log("User 4");
103        break;
104    case "five":
105    case 5:
106        console.log("User 5");
107        break;
108    default:
109        console.log("Invalid input");
110        break;
111
112 } // End of Switch Statement
113
114 // Very awesome here. With this default
115 // statement, if the user's input doesn't
116 // match any case you can say it's not valid

```