Switch Conditional Statements

Authored by AllisonP

JavaScript Switch Statements

A *switch* statement allows a program to evaluate an expression by attempting to match the expression's value to a *case label*. If a match is found, the program jumps to the statement(s) associated with the matched label and continues executing at that point. Note that execution will continue sequentially through all the statements starting at the jump point unless there is a call to `break;`, which exits the switch statement. A switch statement looks like this:

```
switch (expression) {
    case label1:
        statement1;
        break;
    case label2:
        statement2;
        break;
    case label3:
        statement3;
```

Table Of Contents

```
        statement;
}
```

The program first looks for a `case` clause with a label matching the value of ***expression***, then transfers control to the matching clause and executes the associated statements. If no matching label is found, the program looks for the optional *default* clause and, if found, transfers control to that clause and executes the statements associated with it. If no `default` clause is found, the program continues executing after the end of the switch statement.

The `default` Clause

By convention, the `default` clause is always listed last. This is because the statements are checked sequentially, so you run into the following issues if you use the `default` label in an earlier clause:

- If the `default` case is listed *before* (above) a case that matches ***expression***, it will match the `default` case instead. This means the statements associated with the programmed match case won't be executed.
- If the `default` case doesn't have a break statement, any statements in the case label immediately following it will be executed.

The `break;` Statement

The break statement is optional, but you'll typically see one at the end of each `case` clause to ensure that the program breaks out of the switch statement once the statements associated with a matched case are executed. Once the flow of execution hits `break;`, it exits the switch statement and continues executing at the next line following the end of the switch statement; if the break statement is omitted, the program continues executing the next statement in the switch statement — even if its case label doesn't match ***expression***.

- EXAMPLE

Given an integer, n, such that $0 < n < 11$, do the following:

1. If n is equal to 2, print `A`.
2. If n is equal to 3, print `B`.
3. If n is equal to 4, print `C`.
4. If n is equal to 5, print `D`
5. For all the other values of n, print `E`.

Input Format

A single integer denoting n.

```
var input = "";
process.stdin.on('data', function (data) {
    input = data;
```

```
/**** Ignore above this line. ****/

function switchDemo() {
    var n = parseInt(readLine());

    switch (n) {
        case 2:
            console.log("A");
            break;
        case 3:
            console.log("B");
            break;
        case 4:
            console.log("C");
            break;
        case 5:
            console.log("D");
            break;
        default:
            console.log("E");
    }

    console.log("Exited switch.");
}
```

Input

4

Run

Output

Run the code above with the given input, and then try replacing that input with other integers and seeing how it changes. Note that, once reached, the `break;` statements transfer control back outside of the switch statement to the next line of code (in this example, there is no more code to execute).

Now, let's consider the same problem, but this time we'll remove all the `break;` statements from our code:

```
var input = "";
process.stdin.on('data', function (data) {
    input = data;
    switchDemo();
});
function readLine() { return input; }
/**** Ignore above this line. ****/

function switchDemo() {
    var n = parseInt(readLine());

    switch (n) {
        case 2:
            console.log("A");
        case 3:
            console.log("B");
        case 4:
            console.log("C");
        case 5:
            console.log("D");
        default:
```

```
    console.log("Exited switch.");
}
```

Input

4

Run

Output

Run the code above with the given input, and then try replacing that input with other integers and seeing how it changes. Observe that the statements execute sequentially, starting with the matching case.

Now, let's look at what happens if we don't parse the input as an integer:

```
var input = "";
process.stdin.on('data', function (data) {
    input = data;
    switchDemo();
});
function readLine() { return input; }
/**** Ignore above this line. ****/

function switchDemo() {
    // This will read n as an object.
```

```
    switch (n) {
        case 2:
            console.log("A");
            break;
        case 3:
            console.log("B");
            break;
        case 4:
            console.log("C");
            break;
        case 5:
            console.log("D");
            break;
        default:
            console.log("E");
    }

    console.log("Exited switch.");
}
```

Input

4

Run

Output

Run the code above with the given input, and observe that the code *does not* parse

(`===`) is made comparing the *case label* to the *expression value*.

Multi-Criteria Case

In the example below, we consider a similar problem in which there are multiple criteria for each case.

- EXAMPLE

Given an integer, n, such that $0 < n < 11$, do the following:

1. If n is equal to 2, print `A`.
2. If n is equal to 4, print `A`.
3. If n is equal to 6, print `A`.
4. If n is equal to 3, print `B`.
5. If n is equal to 5, print `B`.
6. If n is equal to 7, print `B`.
7. For all other values of n, print `C`.

Input Format

A single integer denoting n

```
process.stdin.on('data', function (data) {
    input = data;
    switchDemo();
});
function readLine() { return input; }
/**** Ignore above this line. ****/

function switchDemo() {
    var n = +(readLine());

    switch (n) {
        case 2:
        case 4:
        case 6:
            console.log("A");
            break;
        case 3:
        case 5:
        case 7:
            console.log("B");
            break;
        default:
            console.log("C");
    }

    console.log("Exited switch.");
}
```

Input

4

Run

Run the code above with the given input, and then try replacing that input with other integers and seeing how it changes.

- IF-ELSE CONDITIONAL STATEMENTS Recommended Article

If-Else Conditional Statements View