Theory: Switch

15 minutes 0 / 5 problems solved

You already know how to write conditions using the `if` operators. This construction is great for simple cases, but when it is used with too many branches it becomes way too complicated. There is another operator called `switch` that works better when you need to check multiple potential values. In this topic, you will learn how to use it.

§1. Switch syntax

Imagine you are helping to write a program for a train ticket app. You need to show the ticket price, which varies depending on the person's destination. Using the `if` operator, you can do it like this:

```
const cityTo = "Paris";

if (cityTo === "Berlin") {
  console.log("The price is $100");
} else if (cityTo === "Paris") {
  console.log("The price is $120");
} else(cityTo === "London") {
  console.log("The price in $150");
}
```

As a result, you will see the text "The price is $120" in the console.

Even though there aren't too many destination options, the example above is still difficult to read and repetitive. It is not the best idea to process values using the `if` operator: in such cases, it is better to use the `switch` operator:

```
const cityTo = "Paris";

switch (cityTo) {
  case "Berlin":
    console.log("The price is $100");
    break;
  case "Paris":
    console.log("The price is $120");
    break;

  case "London":

    console.log("The price is $150");

    break;

}
```

The result will be the same.

The `switch` statement is used to perform specific actions depending on the value of the variable. It matches the value or expression in the parentheses (`cityTo`) with each case clause ("Berlin", "Paris", and "London"). If the case clause is equal to the expression, the corresponding operator is executed (shown "The price is $120" in the console). For each `case`, the operator checks the value in strict equality using type checking.

Please pay attention to the `break` operator at the end of each case. We will look at this in more detail later.

§2. Processing similar cases

Let's say that the prices for a ticket to London and a ticket to Rome are the same. Now you can combine these cases to handle them together:

Current topic:

Switch

Topic depends on:

✕ Conditional operators

Topic is required for:

Break and continue

Table of contents:

↑ Switch

§1. Switch syntax

§2. Processing similar cases

§3. Break operator

§4. Conclusion

Feedback & Comments

```
const cityTo = "Rome";

switch (cityTo) {
  case "Berlin":
    console.log("The price is $100");
    break;
  case "Paris":
    console.log("The price is $120");
    break;

  case "London":

  case "Rome":

    console.log("The price is $150");

    break;

}
```

As a result, for both London and Rome you will see the text "The price is $150" in the console.

What if you have the same price for all cities except Berlin, Paris, and London? It would be strange to list the names of all these cities with identical prices. In this situation, it is better to use the `default` case of the `switch` statement. Here's an example:

```
switch (cityTo) {
  case "Berlin":
    console.log("The price is $100");
    break;
  case "Paris":
    console.log("The price is $120");
    break;
  case "London":
    console.log("The price is $150");

    break;

  default:

    console.log("The price is $90");

    break;

}
```

If `cityTo` is equal to `Berlin`, you will get a message "The price is $100" in the console, just like before. If `cityTo` contains "Stockholm ", "Athens", or "Helsinki", you will see "The price is $90".

This way, the `default` case works if a few cases don't match. The `default` case can be placed in any part of the `switch` statement, but by convention it is better to add it to the last clause.

§3. Break operator

In all previous examples, you probably noticed the `break` keyword added at the end of each code snippet. As a result, the program compared each expression and, when it was *true*, processed the associated block and stopped calculating the remaining cases. The situation would be different if we forgot to insert the `break` keyword:

```
const cityTo = "Berlin";

switch (cityTo) {
  case "Berlin":
    console.log("The price is $100");
  case "Paris":
    console.log("The price is $120");
  case "London":
    console.log("The price is $150");

  default:

    console.log("The price is $90");

}
```

When the browser finds the correct statement, it starts executing it until it finds the `break` keyword. Without it, the browser will think that we don't want to stop the program, so it will continue to process other blocks of code even if they correspond to other `cases`. Then, the result of executing the code in the console will look like this:

```
The price is $100
The price is $120
The price is $150
The price is $90
```

That being said, please, do not forget to use the `break` keyword after each code block to avoid errors!

§4. Conclusion

Now you know that it is much more preferable to use the `switch` statement instead of `if` to handle several different variable values. Switch helps avoid unnecessary repetition and makes your code easier to read. It also gives you more options for managing the handler block using the `break` and `default` keywords.

Report a typo

27 users liked this theory. **0** didn't like it. **What about you?**

😍 🙂 😐 ☹️ 😡

Start practicing

Comments (1) Hints (0) Useful links (0) Show discussion