Switch Scope

In this lesson, you will see two ways to use the switch statement and how those ways can affect the scope of variable declaration.

A `switch` statement is similar to an `if` statement. The condition set between the two parentheses must be met to reach one of the cases. If not, the `default` case will be entered.

```
function switchFunction(a: number): void {
    switch (a) {
        case 1:
            let variableInCase1 = "test";
            console.log(variableInCase1);
            break;
        case 2:
            let variableInCase2 = "test2";
            console.log(variableInCase2);
            break;
        default:
          console.log("Default");
    }
}
switchFunction(1);
switchFunction(2);
switchFunction(3);
```

Not adding the keyword `break` will fall through the `case` underneath. A quick modification of the previous example shows that the value `1` will now print `test2` twice because the value `1` gets into the `case 2`.

```
function switchFunction(a: number): void {
    switch (a) {
        case 1:
            let variableInCase1 = "test";
            console.log(variableInCase1);
        case 2:
            let variableInCase2 = "test2";
            console.log(variableInCase2);
            break;
        default:
          console.log("Default");
    }
}
switchFunction(1);
switchFunction(2);
switchFunction(3);
```

The `switch` statement requires curly brackets after the colon and after the `break` statement. Otherwise, variables defined within the parent scope are shared. This is not a constraint with TypeScript, but it is with ECMAScript.

```
function switchFunction(a: number): void {
    switch (a) {
        case 1:
            let variableInCase1 = "test";
            console.log(variableInCase1);
            break;
        case 2:
            let variableInCase2 = "test2";
            console.log(variableInCase2);
            break;
    }
}

switchFunction(1);
switchFunction(2);
```

The previous code can be problematic. For example, let's change the variable to use the same name. The scope is expanded to the whole switch statement. The following code does not compile in TypeScript.

```
function switchFunction(a: number): void {
    switch (a) {
        case 1:
            let variableInCase1 = "test";
            console.log(variableInCase1);
            break;
        case 2:
            let variableInCase1 = "test2";
            console.log(variableInCase2);
            break;
    }
}
switchFunction(1);
switchFunction(2);
```

To avoid stumbling into a situation where variables are shared across cases, it is suggested to use curly braces for each case.

```
function switchFunction(a: number): void {
    switch (a) {
        case 1: {
            let variableInCase1 = "test";
            console.log(variableInCase1);
            break;
        }
        case 2: {
            let variableInCase1 = "test2";
            console.log(variableInCase1);
            break;
        }
    }
}
switchFunction(1);
switchFunction(2);
```

It’s a good practice to always use curly brackets around each `case`. Adding brackets makes the scope explicit and ensures that code from one `case` does not impact other cases.

Back

TypeScript Scope is JavaScript Scope

Next

The Multiple Methods of Declaring a S...

Completed

Report an Issue