Training day17 Report

(1 July 2024)

JavaScript Break and Continue

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

Break statement

The **break statement** is used to jump out of a loop. It can be used to “jump out” of a switch() statement. It breaks the loop and continues executing the code after the loop.

Example: This example shows the implementation of the JavaScript Break statement.

JavaScript

```
let content = "";
let i;
for (i = 1; i < 1000; i++) {
   if (i === 6) {
      break;
   }
   content += "Hello" + i + "\n"
}
console.log(content);
```

Output

```
Hello1

Hello2

Hello3

Hello4

Hello5
```

Continue statement

The continue statement “jumps over” one iteration in the loop. It breaks iteration in the loop and continues executing the next iteration in the loop.

Example: This example shows the implementation of the JavaScript Continue statement.

JavaScript

```
let content = "";
let i;
for (i = 1; i < 7; i++) {
    if (i === 4) {
        continue;
    }
    content += "World" + i + "\n";
}
console.log(content);
```

Output

```
World1

World2

World3

World5

World6
```

JavaScript nested-if statement

JavaScript allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement. A nested if is an if statement that is the target of another if or else.

Syntax:

```
if (condition1)
{
  // Executes when condition1 is true
  if (condition2)
  {
    // Executes when condition2 is true
  }
}
```

Functions in JavaScript

A **JavaScript function** is a block of code designed to perform a particular task. It encapsulates a set of instructions that can be reused throughout a program. Functions can take parameters, execute statements, and return values, enabling code organization, modularity, and reusability in JavaScript programming.
A JavaScript function is executed when "something" invokes it (calls it).

Example: A basic javascript function, here we create a function that divides the 1st element by the second element.

JavaScript

```
function myFunction(g1, g2) {
    return g1 / g2;
}
const value = myFunction(8, 2); // Calling the function
console.log(value);
```

Output:

4

Syntax: The basic syntax to create a function in JavaScript is shown below.

```
function functionName(Parameter1, Parameter2, ...)
{
    // Function body
}
```

To create a function in JavaScript, we have to first use the keyword *function*, separated by the name of the function and parameters within parenthesis. The part of the function inside the curly braces {} is the body of the function.
In javascript, functions can be used in the same way as variables for assignments, or calculations.

Function Invocation:

- Triggered by an event (e.g., a button click by a user).
- When explicitly called from JavaScript code.
- Automatically executed, such as in self-invoking functions.

Function Definition:

Before, using a user-defined function in JavaScript we have to create one. We can use the above syntax to create a function in JavaScript. A function definition is sometimes also termed a function declaration or function statement. Below are the rules for creating a function in JavaScript:

- Every function should begin with the keyword *function* followed by,

- A user-defined function name that should be unique,
- A list of parameters enclosed within parentheses and separated by commas,
- A list of statements composing the body of the function enclosed within curly braces {}.

Example: This example shows a basic declaration of a function in javascript.
JavaScript

```
function calcAddition(number1, number2) {
   return number1 + number2;
}
console.log(calcAddition(6,9));
```

Output:

15

In the above example, we have created a function named calcAddition,

- This function accepts two numbers as parameters and returns the addition of these two numbers.
- Accessing the function with just the function name without () will return the function object instead of the function result.

There are three ways of writing a function in JavaScript:

Function Declaration:

It declares a function with a function keyword. The function declaration must have a function name.

Syntax:

```
Function name(paramA, paramB) {
   // Set of statements
}
```

Function Expression:

It is similar to a function declaration without the function name. Function expressions can be stored in a variable assignment.

Syntax:

```
let name= function(paramA, paramB) {
   // Set of statements
}
```

Example: This example explains the usage of the Function expression.
JavaScript

```
const square = function (number) {
     return number * number;
};
const x = square(4); // x gets the value 16
console.log(x);
```

Output

16

Functions as Variable Values:

Functions can be used the same way as you use variables.

Example:

```
// Function to convert Fahrenheit to Celsius
function toCelsius(fahrenheit) {
    return (fahrenheit - 32) * 5/9;
  }

  // Using the function to convert temperature
  let temperatureInFahrenheit = 77;
  let temperatureInCelsius = toCelsius(temperatureInFahrenheit);
  let text = "The temperature is " + temperatureInCelsius + " Celsius";
```

Arrow Function:

Arrow Function is one of the most used and efficient methods to create a function in JavaScript because of its comparatively easy implementation. It is a simplified as well as a more compact version of a regular or normal function expression or syntax.

Syntax:

```
let function_name = (argument1, argument2 ,..) => expression
```

Example: This example describes the usage of the Arrow function.

JavaScript

```
const a = ["Hydrogen", "Helium", "Lithium", "Beryllium"];

const a2 = a.map(function (s) {
    return s.length;
});

console.log("Normal way ", a2); // [8, 6, 7, 9]

const a3 = a.map((s) => s.length);

console.log("Using Arrow Function ", a3); // [8, 6, 7, 9]
```

Output

```
Normal way  [ 8, 6, 7, 9 ]
```

Using Arrow Function [8, 6, 7, 9]

Function Parameters:

Parameters are additional information passed to a function. For example, in the above example, the task of the function *calcAddition* is to calculate the addition of two numbers. These two numbers on which we want to perform the addition operation are passed to this function as parameters. The parameters are passed to the function within parentheses after the function name and separated by commas. A function in JavaScript can have any number of parameters and also at the same time, a function in JavaScript can not have a single parameter.
Example: In this example, we pass the argument to the function.
JavaScript

```
function multiply(a, b) {
    b = typeof b !== "undefined" ? b : 1;
    return a * b;
}

console.log(multiply(69)); // 69
```

Output:

69