AIMERZ.ai
Aim.Act.Achieve

NOTES

Introduction to Function

Introduction to Function

Topics Covered

- Introduction to function
- Function declaration and invoking
- Function with parameter

What is a function?

A function is a block of code used to perform a specific task which can be reused throughout the program.

Advantages of a function

1. **Code Reusability**

   We can use a function multiple times in our code, whenever we need that particular functionality.

2. **Less Coding**

   By wrapping a specific task or logic in a function, we can reuse it instead of writing the same logic at multiple places in our code which will eventually increase the code size of the program. It also helps us in following the **DRY (Do Not Repeat Yourself)** principle of coding to stop the repetition of the same code.

Function declaration and invoking

Function declaration syntax

We can follow the given syntax for declaring a function in a javascript program-

```
JavaScript
function functionName() {
    // logic code
}
```

Let’s have a detailed look at function declaration syntax-

1. **function:**
   function is the keyword, which is used to declare a function

2. **functionName:**
   It will be the name of the function, like greetUser. We can use any name for the function but that should follow the same naming convention and rules like the variables.

3. **{}**
   This represents the function body where we will be writing all the logic code which we will execute when the function will be called or invoked.

Function call or invoke

If we want to execute the code inside the function body, then we have to invoke or call the function, which can be done by the function name followed by the parenthesis.

```
JavaScript
function functionName() {
    // logic code
}
// invoking or calling the function
functionName()
```

Example of function declaration and calling

```
JavaScript
function greetUser() {
  console.log("Aimerz");
}
greetUser();
// Output
// Aimerz
```

Parameter vs Argument

1. **Parameters**

   Function parameters are listed inside the parenthesis in the function definition. It is used to make functions more versatile and to customize the output.

2. **Arguments**

   Function arguments are the values which are received by the function when it is being invoked.

Example

```
JavaScript
function greetUser(name) {
  console.log("Welcome to " + name);
}
greetUser("Aimerz");
// Output
// Welcome to Aimerz
```

In the above example, name is the parameter and **"Aimerz"** is the argument passed to the greetUser function in which the argument **"Aimerz"** will be assigned to the name as its value.

Returning values from the function

We can use a **return statement** in javascript to specify the value that a function should return. When a **return statement** is executed within the function, it immediately **stops** the **execution** of the function and sends the specified value back to the caller.

Function without return keyword

If we are not using **return keyword** to return any value from the function, it will return **undefined** as its default value.

```
JavaScript
function greetUser(name) {
  console.log("Welcome to " + name);
}
const greeting = greetUser("Aimerz");
console.log(greeting);
// output
// Welcome to Aimerz
// undefined
```

In the above example, we are not using a **return keyword** inside the **greetUser function**. But when we are getting undefined if we are logging the value of greeting as it is the default return value of a function in the javascript.

Function with return keyword

```
JavaScript
function greetUser(name) {
  return "Hello " + name;
}
const greeting = greetUser("Aimerz");
console.log(greeting);
// output
// Hello Aimerz
```

In the above example, the function is returning the greeting message which we are being stored inside the greeting variable.

Function Declaration vs Function Expression

Function Declaration	**Function Expression**
• A function declaration must have a function name. • Function declaration does not require a variable assignment.	• A function expression is similar to a function declaration without the function name. • Function expressions can be stored in a variable assignment.

Note: More differences about them will be discussed in the class of hoisting in future class.

Syntax

```
JavaScript
// function declaration syntax
function functionName() {
  // logic code
}

// function expression syntax
const variableName = function () {
  // logic code
}
```

Example

```
JavaScript
// function declaration
function greetUser1(name) {
  return "Good Morning " + name;
}

// function expression
const greetUser2 = function (name) {
  return "Good Morning " + name;
};

const greeting1 = greetUser1("Prabir");
const greeting2 = greetUser2("Vishwa");
console.log(greeting1);
console.log(greeting2);

// output
// Good Morning Prabir
// Good Morning Vishwa
```

Function with parameter

While using the function we can pass parameters to it, so that we can make our function more versatile and can customize the output as per our requirement.

1. Single parameter

As we have seen in the above examples, we can pass one single parameter to our function by mentioning that one inside the parenthesis.

Let's take a new example to get more about of parameter

```javascript
function addNumber(secondNumber) {
  const firstNumber = 10;
  return firstNumber + secondNumber;
}

const sum = addNumber(20);
console.log(sum);

// Output
// 30
```

In the above example, we are getting **secondNumber** as a parameter and returning the sum of firstNumber which is defined inside the function and secondNumber from the function.

2. Multiple Parameter

We can also take multiple parameters in our function as per our requirement by separating them with commas.
In the above example, our function was only taking the secondNumber as a parameter, now let's take firstNumber and secondNumber both as a parameter.

```javascript
function addNumber(firstNumber, secondNumber) {
  return firstNumber + secondNumber;
}
```

```
const sum = addNumber(10, 20);
console.log(sum);

// Output
// 30
```

In the above example, we can see that we have passed two parameters in our function, similarly we can pass **multiple parameters** just by separating them with commas.
You can now try adding three numbers, by using thirdNumber as a parameter.

3. N number of parameters

In some cases,we are not aware of the number of parameters required or what we will get from the arguments, so in that type of case we can get any "n" number of parameters.
Let's assume in the above example, we want to use the same function to add two, three or ten numbers, so in such a case we are not sure about the number of parameters then we will use the "arguments" keyword to get all the parameters.

Let's take the example to understand more about arguments keyword

JavaScript
```javascript
function addNumber() {
  let sum = 0;
  for (let i = 0; i < arguments.length; i++) {
    sum += arguments[i];
  }
  return sum;
}

const sum1 = addNumber();
const sum2 = addNumber(5);
const sum3 = addNumber(5, 5);
const sum4 = addNumber(5, 5, 10);
console.log(sum1);
console.log(sum2);
console.log(sum3);
console.log(sum4);
```

```
// Output
// 0
// 5
// 10
// 20
```

In the above example, we are looping over the arguments using the for loop as arguments are similar to arrays which we will study later. We are using the length method of arguments to check the total length of arguments to loop.
We can see that we are passing zero, one, two and three arguments, and also can pass more as per our requirement even then our function is working properly. Just by this approach we do not need multiple functions to handle n number of arguments.

THANK YOU

AIMERZ.ai
Aim.Act.Achieve

Stay updated with us!

Vishwa Mohan

Vishwa Mohan

vishwa.mohan.singh

Vishwa Mohan