JavaScript

Functions

Pooja Godse
&
ALK Bilahari

Procareer Academy

Outline

- Function
- Types of functions
- Parameters
- Default Parameters
- Return types
- Scope of Variables

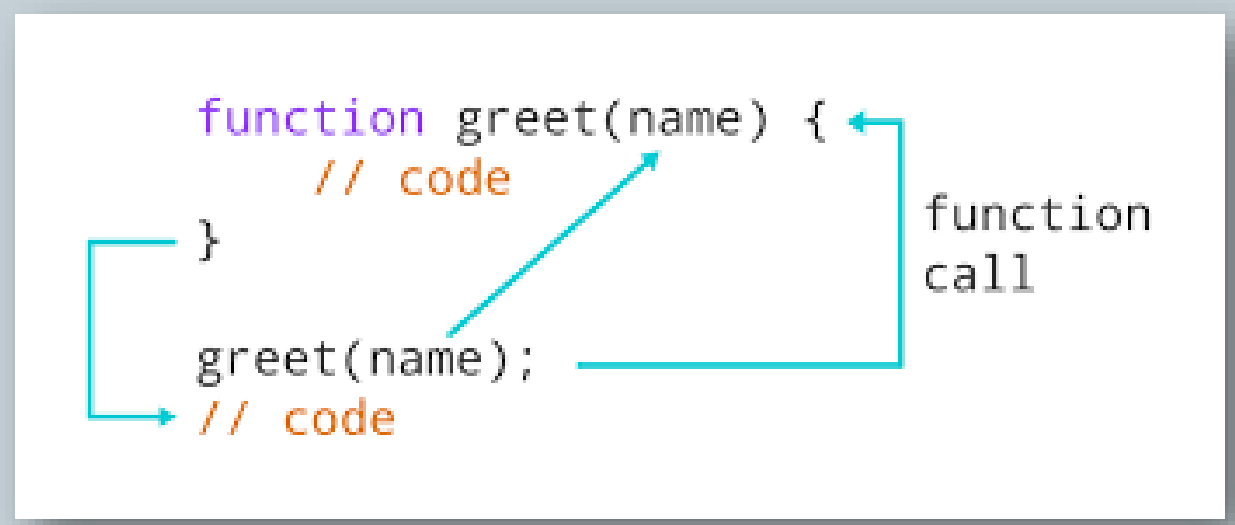
Recap

- Loops
  - for
  - while
  - do while
- Strings

Procareer Academy

Function

is a **block of JavaScript code** that is defined once but may be executed, or invoked, any number of times.



Procareer Academy

Types of Functions

JavaScript has two types of functions

1. User-defined functions
2. Built-in functions

User defined Functions

must be **defined or declared** and then it can be invoked anywhere in the program

Syntax for **Function Declaration**:

```
function _name(parameter 1, parameter 2 , ..., parameter n)
        {
        //statements to be executed
        }
```

Procareer Academy

User defined Functions

Example:

```
function multiply(num1, num2) {
    return num1 * num2;
    }
```

The code written inside the function body will be executed only when it is **invoked or called**.

User defined Functions

Syntax for **Function Invocation**:

function_name(argument 1, argument 2, …, argument n);

Example:

multiply (5,6);

Procareer Academy

User defined Functions: Parameters

- JavaScript functions are **parameterized**
- A function **definition** may include a list of identifiers, known as parameters.
- Parameters work as **local variables** for the body of the function.
- Function invocations provide **values, or arguments**, for the function's parameters.

User defined Functions

```
1.  function multiply(num1, num2) {
2.      if (num2 == undefined)
3.              {
4.              num2 = 1;
5.              }
6.  return num1 * num2;
7.  }
8.  console.log(multiply(5, 6)); // 30
9.  console.log(multiply(5)); // 5
```

Procareer Academy

Default Arguments

Example:

```
1. function fun( a, b)
2. {
3. console.log("Hello");
4.  }
5. fun();
```

Procareer Academy

Default Arguments

JavaScript introduces an option to assign default values in functions.

```
1. function multiply(num1, num2 = 1) {
2.     return num1 * num2;
3. }
4. console.log(multiply(5, 5)); //25
5. console.log(multiply(10)); //10
6. console.log(multiply(10, undefined)); //10
```

return

- Function can have an **optional** return statement.
- This statement should be the last statement in a function.
  - *let x = myFunction(4, 3);*

    function myFunction(a, b) {

    // Return the product of a and b

    *return a * b;*
  - *}*

Procareer Academy

Variable Scopes

- Variables can be **declared within** the function or **outside** the function.
- The **accessibility** of a variable is referred to as **scope**.
- JavaScript scopes can be of three types:
  - Global scope
  - Local scope
  - Block scope

Procareer Academy

Global Scope

```
1.//Global variable
2.var firstName = "Mark";
3.function fullName() {
4.//Variable declared without var has global scope
5.var lastName = "Zuckerberg";
6.console.log("Full Name from inside the function: " + firstName + " " +
lastName);
7.}
8.fullName();
9.console.log("Full Name from outside the function: " + firstName + " " +
lastName);
10.//Full Name from inside the function: Mark Zuckerberg
11.//Full Name from outside the function: Mark Zuckerberg
```

Local Scope

```
1.//Global variable
2.var firstName = "Mark";
3.function fullName() {
4.//Variable declared without var has global scope
5.lastName = "Zuckerberg";
6.console.log("Full Name from inside the function: " + firstName + " " + lastName);
7.}
8.fullName();
9.console.log("Full Name from outside the function: " + firstName + " " + lastName);
10.//Full Name from inside the function: Mark Zuckerberg
11.//Full Name from outside the function: Mark Zuckerberg
```

Global Scope

If a local variable is declared without using 'var', it takes a global scope.

```
1.//Global variable
2.var firstName = "Mark";
3.function fullName() {
4.//Variable declared without var has global scope
5.lastName = "Zuckerberg";
6.console.log("Full Name from inside the function: " + firstName + " " + lastName);
7.}
8.fullName();
9.console.log("Full Name from outside the function: " + firstName + " " + lastName);
10.//Full Name from inside the function: Mark Zuckerberg
11.//Full Name from outside the function: Mark Zuckerberg
```

Variable Scope

- In 2015, JavaScript introduced two new keywords to declare variables: let and const.
- Consider the below example:

```
1.function testVar() {
2.if (10 == 10) {
3.var flag = "true";
4.}
5.console.log(flag); //true
6.}
8.testVar();
```

Procareer Academy

Block Scope

Example

```
1.function testVar() {
2.if (10 == 10) {
3.let flag = "true";
4.}
5.console.log(flag); //Uncaught ReferenceError: flag is not defined
6.}
8.testVar();
```

'const' has the same scope as that of 'let' i.e., block scope.

Procareer Academy

Function Expressions

Function as expression

Procareer Academy

Function Expressions

- A function expression has to be stored in a variable and can be accessed using variable Name.
- Function Expression allows us to create an anonymous function which doesn't have any function name.
- The function keyword can be used to define a function inside an expression.

Function Expressions

Syntax for **Function Declaration**:

function functionName(x, y) { statements… return (z) };

Syntax for **Function Expression (anonymous)** :

let variableName = function(x, y) { statements… return (z) };

Syntax for **Function Expression (named)** :

let variableName = function functionName(x, y) { statements… return (z) };

Procareer Academy

Function Expressions : with out parameters

```
let sayHi = function() {
        alert( "Hello" );
        };
Console.log(sayHi);
```

```
function sayHi() {
alert( "Hello" );
}
let func = sayHi;
func();  // Hello
sayHi();// Hello
```

Note:
A function expression has to be defined first before calling it or using it as a parameter.

Procareer Academy

Function Expressions

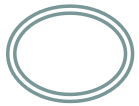
Code for Function Expression (anonymous)

```
var Sub = function(x, y){
      let z = x - y;
      return z;
  }
  console.log("Subtraction : " + Sub(7, 4));
```

Code for Function Expression (named)

```
var cMul = function Mul(x, y){
      let z = x * y;
      return z;
  }
  console.log("Multiplication : " + cMul(7, 4));
```

Procareer Academy



Questions?

Procareer Academy

Thank you.

Procareer Academy