JavaScript Function

A function is a block of code that performs a specific task.

Suppose you need to create a program to create a circle and color it. You can create two functions to solve this problem:

- a function to draw the circle
- a function to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

Types of functions

Function mainly are of 2 types,

- **In Built Functions:** These functions are provided by javascript itself like typeof, console.log etc.
- **User Defined:** These are the functions which are created by users.

In this tutorial, you will learn about user-defined functions.

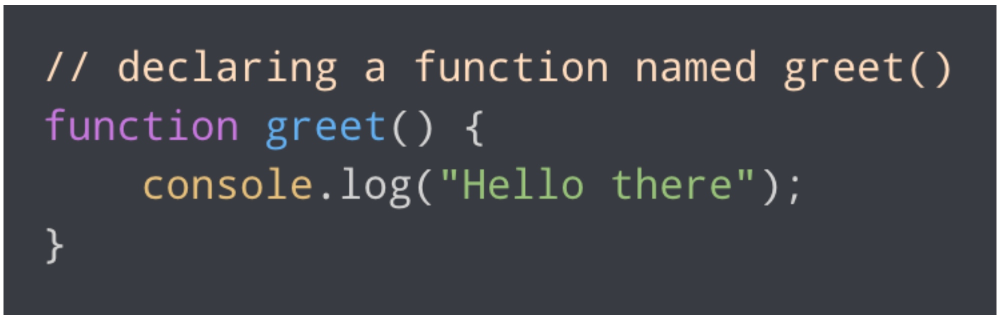
Declaring a Function

The syntax to declare a function is:

```
function nameOfFunction () {
    // function body
}
```

- A function is declared using the function keyword.
- The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.
- The body of function is written within {}.

For example

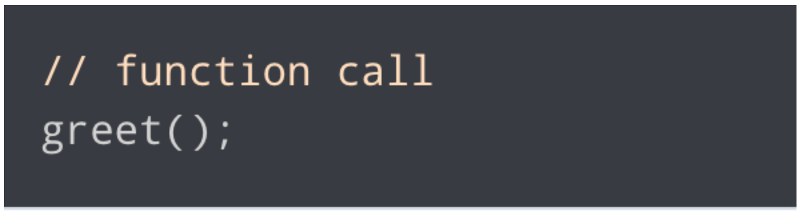


```
// declaring a function named greet()
function greet() {
    console.log("Hello there");
}
```

Calling a Function

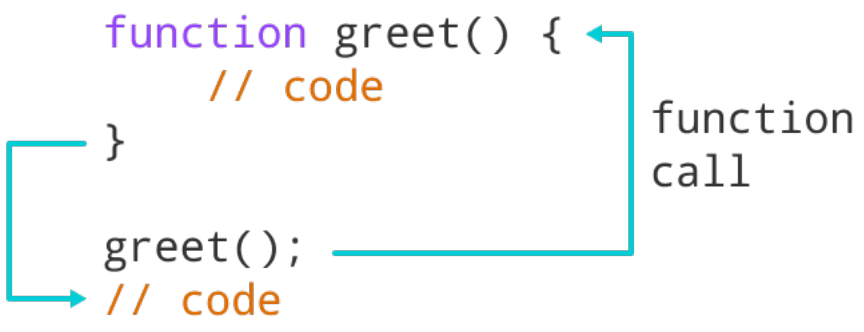
In the above program, we have declared a function named greet(). To use that function, we need to call it.

Here's how you can call the above greet() function.



```
// function call
greet();
```

Working of a Function

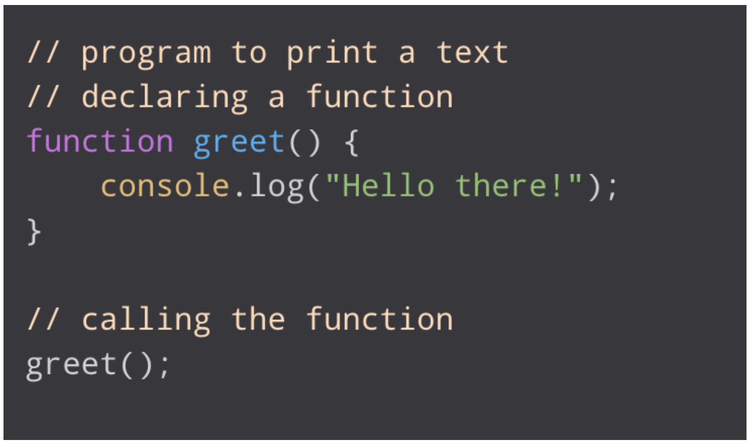


```
function greet() {
    // code
}

greet();
// code
```

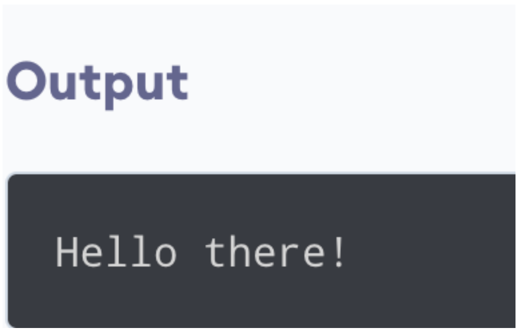
in JavaScript

Example 1: Display a Text



```
// program to print a text
// declaring a function
function greet() {
    console.log("Hello there!");
}

// calling the function
greet();
```



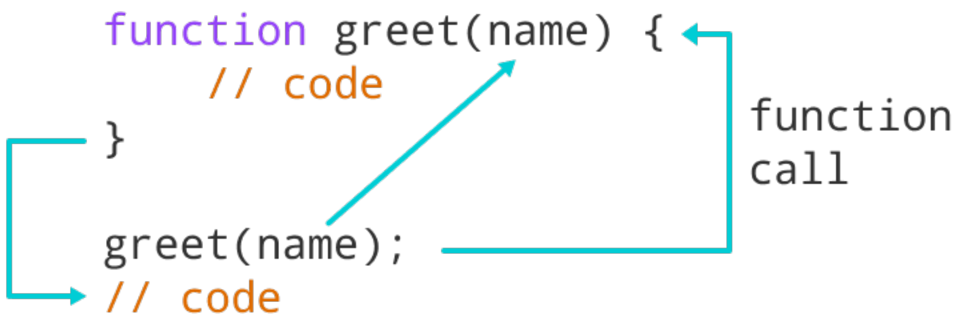
Output

```
Hello there!
```

Function Parameters

A function can also be declared with parameters. A parameter is a value that is passed when declaring a function.

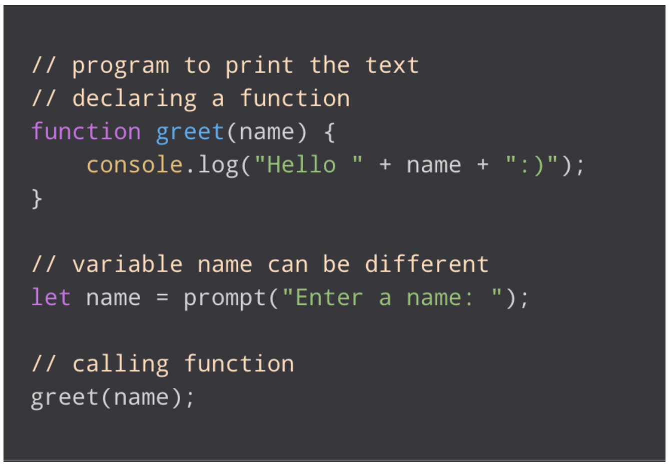
Working of JavaScript Function with parameter



```
function greet(name) {
    // code
}

greet(name);
// code
```

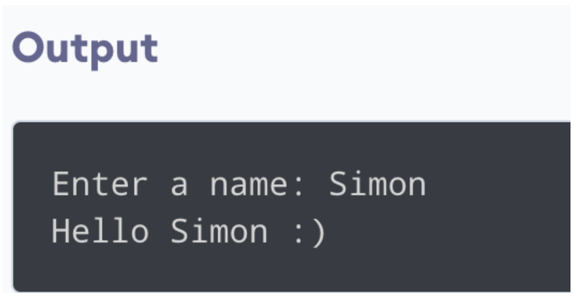
Example 2: Function with Parameters



```
// program to print the text
// declaring a function
function greet(name) {
    console.log("Hello " + name + ":)");
}

// variable name can be different
let name = prompt("Enter a name: ");

// calling function
greet(name);
```



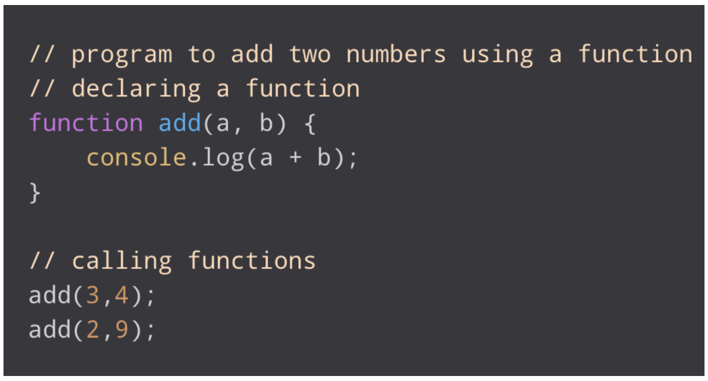
Output

```
Enter a name: Simon
Hello Simon :)
```

In the above program, the greet function is declared with a name parameter. The user is prompted to enter a name. Then when the function is called, an argument is passed into the function.

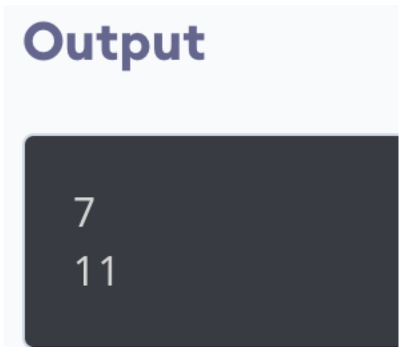
Note: When a value is passed when declaring a function, it is called parameter. And when the function is called, the value passed is called argument.

Example 3: Add Two Numbers



```
// program to add two numbers using a function
// declaring a function
function add(a, b) {
    console.log(a + b);
}

// calling functions
add(3,4);
add(2,9);
```



Output

```
7
11
```

In the above program, the add function is used to find the sum of two numbers.

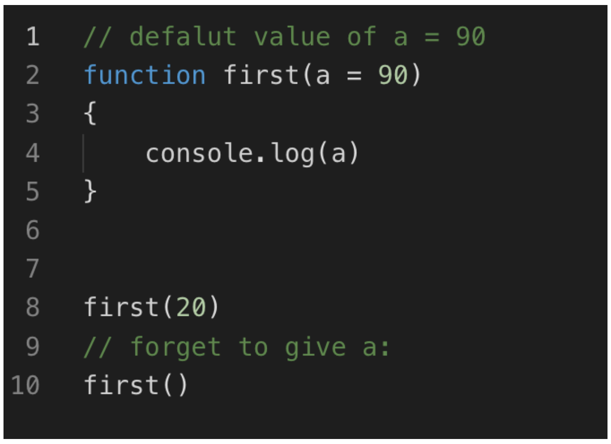
- The function is declared with two parameters a and b.
- The function is called using its name and passing two arguments 3 and 4 in one and 2 and 9 in another.

Notice that you can call a function as many times as you want. You can write one function and then call it multiple times with different arguments.

Default Parameter:

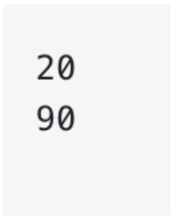
Default parameters are the parameters which will be initialised with any default value when the user forget to give parameter while calling the function

They are used to give any default value to your function parameter in case user forge to provide the parameter while calling a function



```javascript
// defalut value of a = 90
function first(a = 90)
{
    console.log(a)
}


first(20)
// forget to give a:
first()
```



```
20
90
```

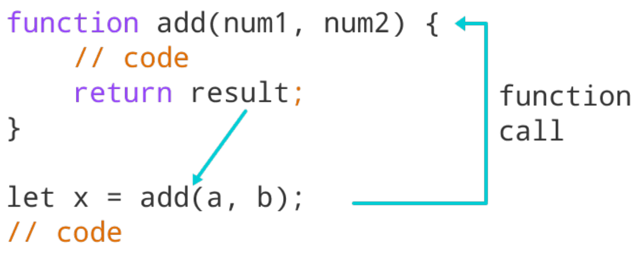
Function Return

The return statement can be used to return the value to a function call.

The return statement denotes that the function has ended. Any code after return is not executed.

If nothing is returned, the function returns an undefined value.

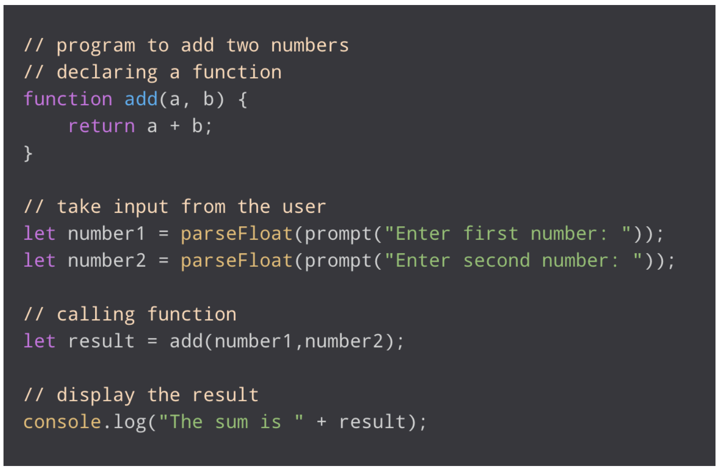
Working of JavaScript Function with return statement



```javascript
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

Example 4: Sum of Two Numbers

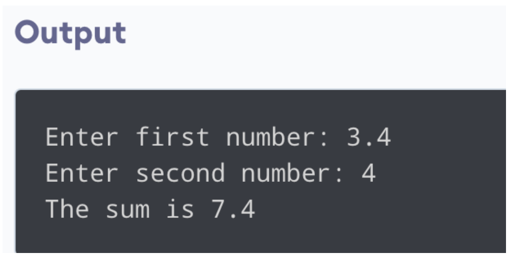


```
// program to add two numbers
// declaring a function
function add(a, b) {
    return a + b;
}

// take input from the user
let number1 = parseFloat(prompt("Enter first number: "));
let number2 = parseFloat(prompt("Enter second number: "));

// calling function
let result = add(number1,number2);

// display the result
console.log("The sum is " + result);
```



Output

```
Enter first number: 3.4
Enter second number: 4
The sum is 7.4
```

In the above program, the sum of the numbers is returned by the function using the return statement. And that value is stored in the result variable.

Benefits of Using a Function

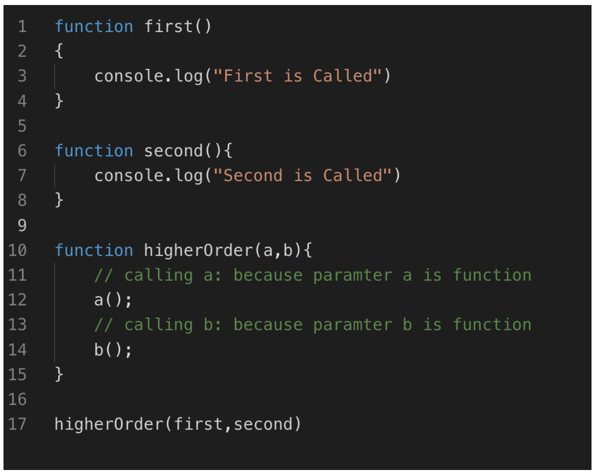
- Function makes the code reusable. You can declare it once and use it multiple times.
- Function makes the program easier as each small task is divided into a function.
- Function increases readability.

Higher Order Functions:

Higher order functions are the functions which takes another functions as an parameter.

As in the following examples the function named as higherOrder is taking 2 parameter a,b. When we are calling the function in line number 17, we are passing the functions first, second as parameter to it.

So higherOrder is an higher order function because it is taking 2 parameters and these are functions and it is calling them inside it's body.

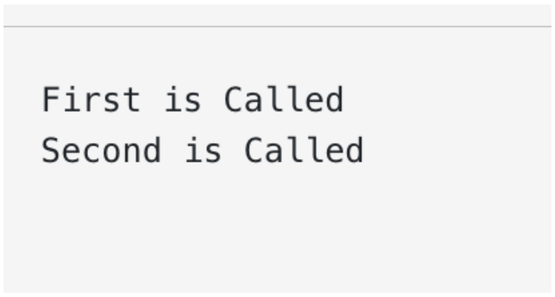


```javascript
function first()
{
    console.log("First is Called")
}

function second(){
    console.log("Second is Called")
}

function higherOrder(a,b){
    // calling a: because paramter a is function
    a();
    // calling b: because paramter b is function
    b();
}

higherOrder(first,second)
```

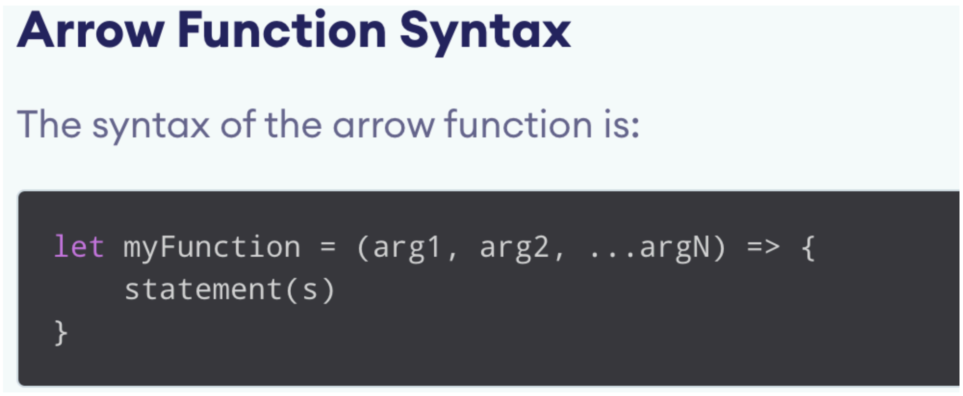


```
First is Called
Second is Called
```

Arrow Functions

In this tutorial, you will learn about JavaScript arrow function with the help of examples.

Arrow function is one of the features introduced in the ES6 version of JavaScript. It allows you to create functions in a cleaner way compared to regular functions. For example,



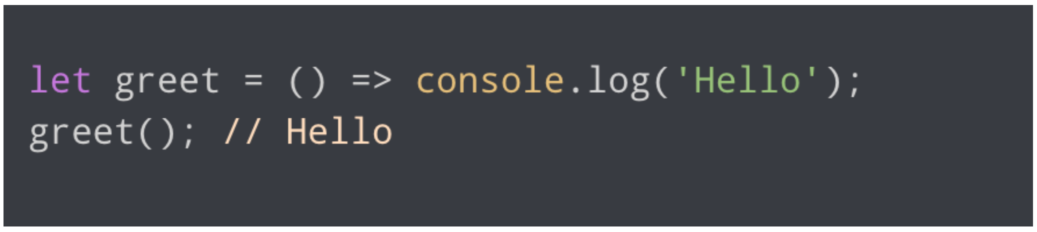
Arrow Function Syntax

The syntax of the arrow function is:

```javascript
let myFunction = (arg1, arg2, ...argN) => {
    statement(s)
}
```

- myFunction is the name of the function
- arg1, arg2, ...argN are the function arguments
- statement(s) is the function body

Example 1: Arrow Function with No Argument

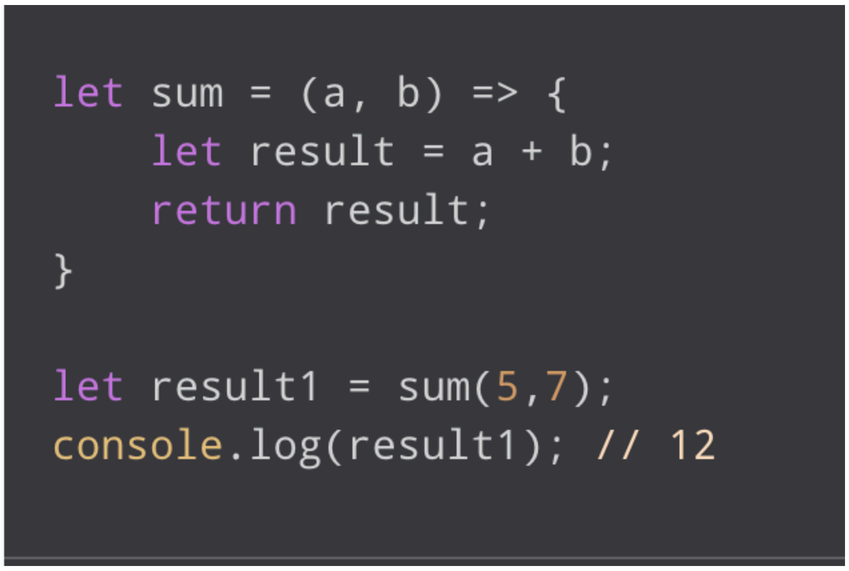


```
let greet = () => console.log('Hello');
greet(); // Hello
```

If a function doesn't take any argument, then you should use empty parentheses. For example,

Example 2: Multiline Arrow Functions

If a function body has multiple statements, you need to put them inside curly brackets {}. For example,



```
let sum = (a, b) => {
    let result = a + b;
    return result;
}

let result1 = sum(5,7);
console.log(result1); // 12
```