
Globant
we are ready

UI Engineering Studio. Day 6

Bootcamp: JS Functions

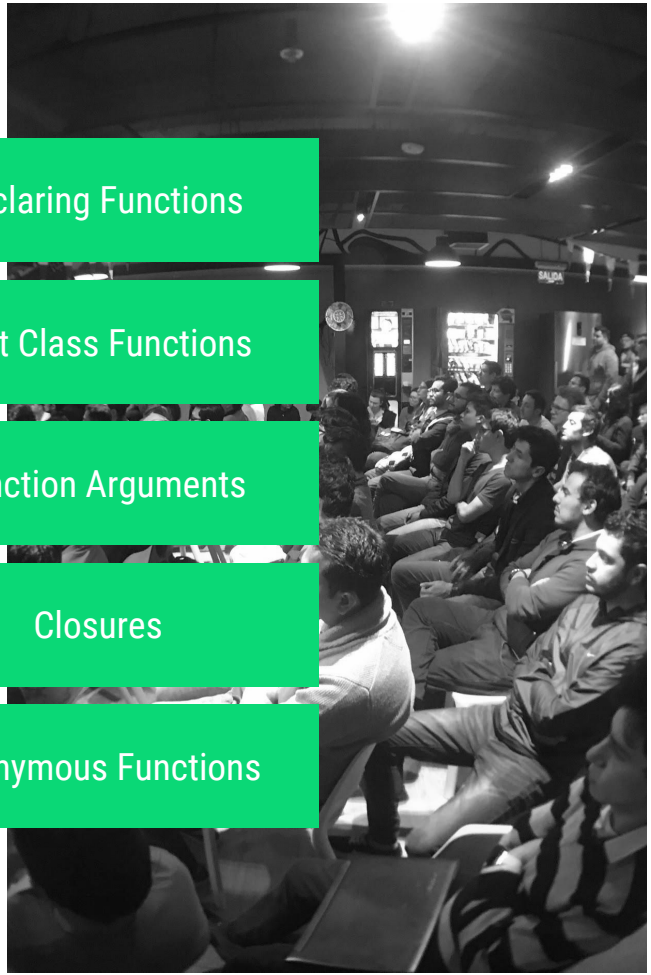
Globant

Functions

Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure—a set of statements that performs a task or calculates a value. To use a function, you must define it somewhere in the scope from which you wish to call it.

-MDN

- Declaring Functions
- First Class Functions
- Function Arguments
- Closures
- Anonymous Functions



UI Boot Camp: JS

Declaring Functions

A **function definition** (also called a **function declaration**, or **function statement**) consists of the function keyword, followed by:

- The name of the function.
- A list of parameters to the function, enclosed in parentheses and separated by commas.
- The JavaScript statements that define the function, enclosed in curly brackets, { }.

-MDN

Function declaration

```
function sumar(a, b) {
 return a + b;
}
```

Function expression

```
var restar = function(a, b) {
 return a - b;
}
```

UI Boot Camp: JS

First-class Function

A programming language is said to have **First-class functions** when functions in that language are treated like any other variable. For example, in such a language, a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.

```
var restarInvert = function(a, b) {
 return b - a;
}
function resta(a, b, restar) {
 return restar ? restar(a,b) : a - b;
}
resta(1, 2);
```

UI Boot Camp: JS Functions

Parameters

Function with parameters

```
function sumar(a, b) {
 return a + b;
}
```

Variable parameter number

```
function sumar(a, b, ...rest) {
 return rest[0];
}
```

Function without parameters

```
function procedure() {
 // ...
}
```

Default values for parameters

```
function sumar(a, b, c = 0) {
 return a + b + c;
}
```

UI Boot Camp: Basics JS Functions

Scope

Variables defined inside a function **cannot be accessed** from anywhere outside the function, because the variable is defined only in the scope of the function. However, **a function can access all variables and functions defined inside the scope in which it is defined.**

-MDN

```
// example 1
function add5(otherNumber) {
 var five = 5;
 return otherNumber + five;
}

console.log(add5(7)); // 12
console.log(five); // undefined

// example 2
var five = 5;

function add5(otherNumber) {
 return otherNumber + five;
}

console.log(add5(7)); // 12
console.log(five); // 5
```

UI Boot Camp: Basics JS

Closures

JavaScript allows for the nesting of functions and grants the **inner function** full access to **all the variables** and functions defined inside the outer function (and all other variables and functions that the outer function has access to). However, the outer function **does not have access** to the variables and functions defined inside the inner function.

```
var x = 10;

function foo() {
 var y = 20;
 function bar() {
   var z = 15;
   return x + y + z;
 }
 return bar();
}

console.log(foo()); // 45
```

UI Boot Camp: Basics JS

Anonymous functions

Anonymous functions are functions without a name. Usually used in function expressions or when using the callback pattern

```
var list = [1, 2, 3, 4];

var total =
list.reduce(function(total, number) {
 return total + number
}, 0);

console.log(total) // 10
```

UI Boot Camp: HTML

Homework: Calculator

https://www.sitepoint.com/dom-manipulation-vanilla-javascript-no-jquery/


Globant
we are ready