scottiBR

guilhermescotti

JavaScript The Tough Parts.

Guilherme Scotti

Eng. de Controle e Automação

Eng. Software UFMG

guilhermescotti

scottiBR

accenture

Back End

Full Stack

dito

Front End

Por que aprender sobre fundamentos da linguagem?

“True mastery means **understanding the core principles** and building up from them.”

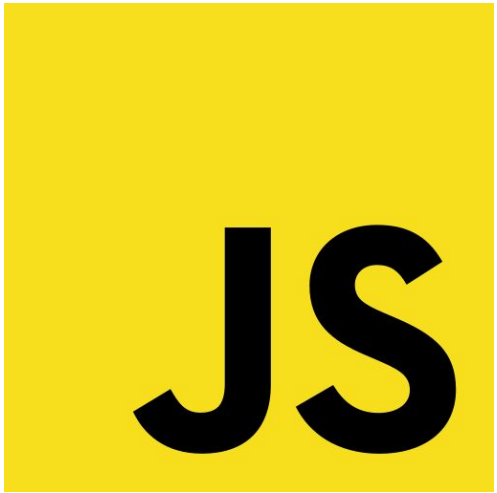
William Sentance

bit.ly/scottiQuiz

- Compilador
- Execution Context e Lexical Environment
- Hoisting
- Quiz II
- Lexical Scope
- Closure


NOW, LET'S CHECK
UNDER THE HOOD.

Hybrid Language



Compiled Language

Interpreted Language

C

C#

C++

Java™

```
        }).done(function(response) {
            for (var i = 0; i < response.length; i++) {
                var layer = L.marker(
                    [response[i].latitude, response[i].longitude]
                    // ,{icon: myIcon}
                );
                layer.addTo(group);

                layer.bindPopup(
                    "<p>" + "Species: " + response[i].species + "<
                    "<p>" + "Description: " + response[i].descript
                    "<p>" + "Seen at: " + response[i].latitude + '
                    "<p>" + "On: " + response[i].sighted_at + "</p
            }

            $('select').change(function() {
                species = this.value;
            });
        });
    }
$.ajax({
            url: queryURL,
            method: "GET"
        }).done(function(response) {
            for (var i = 0; i < response.length; i++) {
                var layer = L.marker(
                    [response[i].latitude, response[i].longitude]
                    // ,{icon: myIcon}
                );
                layer.addTo(group);
```

“JavaScript is a lightweight, **interpreted language**”

MDN Web Docs

“JavaScript is a **interpreter-agnostic** language”

ECMASCRIPT

V8

SpiderMonkey

Chakra

Nitro

“**Execution Context** is defined as the environment in which the JavaScript code is executed.”

Rupesh Mishra

- Compilation or Creation Phase
- Execution Phase

HOISTING

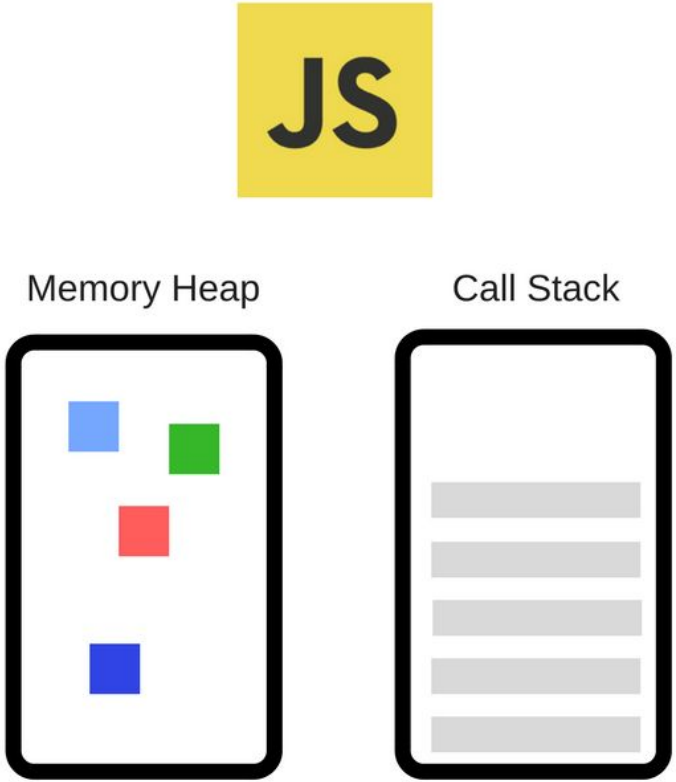
Declarations of variables and functions being

moved to top of your code


BUSTING THE
MYTHS

“**Hoisting** refers to the default behavior of Javascript to process and **put all variables and functions declarations into Lexical Environment** during compile phase of its execution context”

Maya Shavin



Lexical Environment is a **data structure** that holds identifier-variable mapping on the **memory heap**.

```
say("Hello"); //Hello
console.log(world); // world is undefined

var world = "World";

function say(hello) {
  console.log(hello);
}
```

JavaScript only hoists variable declarations,

initializations are not hoisted.

```
if (x === undefined) {
  console.log("x is not defined");
} else {
  console.log(x);
}

var x = "Defined";

//result "x is not defined"
```

Assignment of value to a variable happens **only at the execution phase**

Undefined X Reference Error

ES6

Arrow Functions

Const

Let

”**All declarations** in JavaScript function, var, let, const even classes, **are hoisted** at the compiler phase“

Sukhjinder Arora

```
console.log(x);


let x = "Hello";
```

Temporal Dead Zone (TDZ)

NOTE 13.2.1 let and const declarations define variables that are scoped to the running execution context Lexical Environment. The variables are created when their containing Lexical Environment is instantiated **but may not be accessed in any way until the variable's Lexical Binding is evaluated**. A variable defined by a Lexical Binding with an Initializer is assigned the value of its Initializer Assignment Expression when the Lexical Binding is evaluated, not when the variable is created. If a Lexical Binding in a let declaration does not have an Initializer the variable is assigned the value undefined when the Lexical Binding is evaluated.

”TDZ it’s a reserved memory space where declarations of ES6 **remain until they are initialized**“


CAN'T TOUCH THIS

Porque TDZ existe?

const undefined?

As far as I'm concerned the motivating feature for TDZs is to **provide a rational semantics for const**.

Allen Wirfs-Brock
Project Editor ES6

```
const words = "Hello TDC";
words = "Bye TDC";
// Type error
```

A language with **only let and var** would have been simpler then what we ended up with

Allen Wirfs-Brock
Project Editor ES6

```
var speaker = "Guilherme";
function name() {
  console.log(speaker);

  let speaker = "Scotti";
}
name();
```

Function Declaration === Function Expressions ?

```
function hello() {}
```

```
const hello = ()=>{}
```

```
hello();

world();


var hello = function() {
  console.log("Hello");
};
const world = () => console.log("world");
```

Vantagem do hoisting nas functions?

Mutual recursion

Clean Code

```
core execution code
/*...*/

function x(){}
/*... functions*/
```

Lexical Scope

bit.ly/scottiQuiz

```
function speaker() {
  var lastName = "Scotti";
}
speaker();
console.log(lastName);
//Reference error
```

```
function b() {
  var result = 3;
}
function a() {
  var result = 2;
  b();
}
var result = 1;
a();
```

Execution Environment

b()

result = 3

a()

result = 2

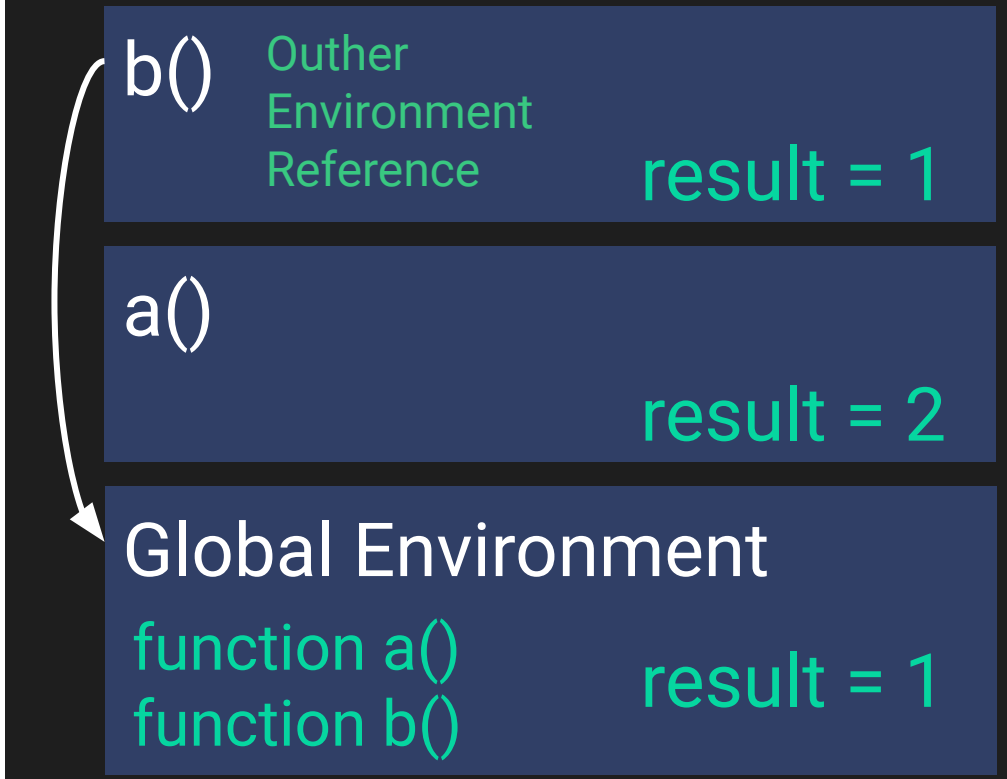
Global Environment

result = 1

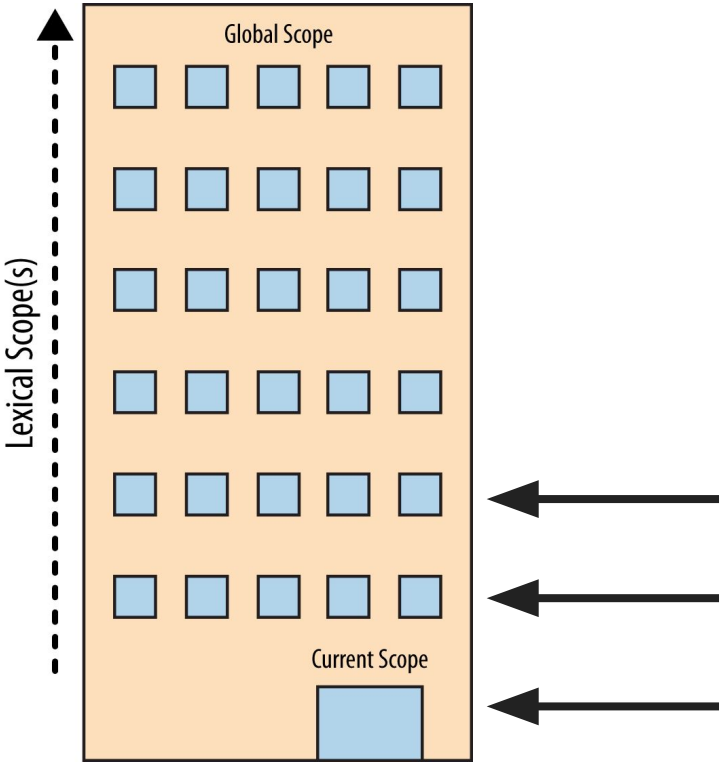
```
function b() {
  console.log(result);
}

function a() {
  var result = 2;
  b();
}

var result = 1;
a();
```



Outer Environment



Lexical Scope always reference over a variable,

not a value

```
let speaker = "scotti";

function callSpeaker() {
  console.log(speaker);
}

speaker = "Guilherme";
callSpeaker();
```

Call Stack

callSpeaker()

speaker = Guilherme

Global Environment

function callSpeaker()

speaker = Guilherme

What is **Closure**

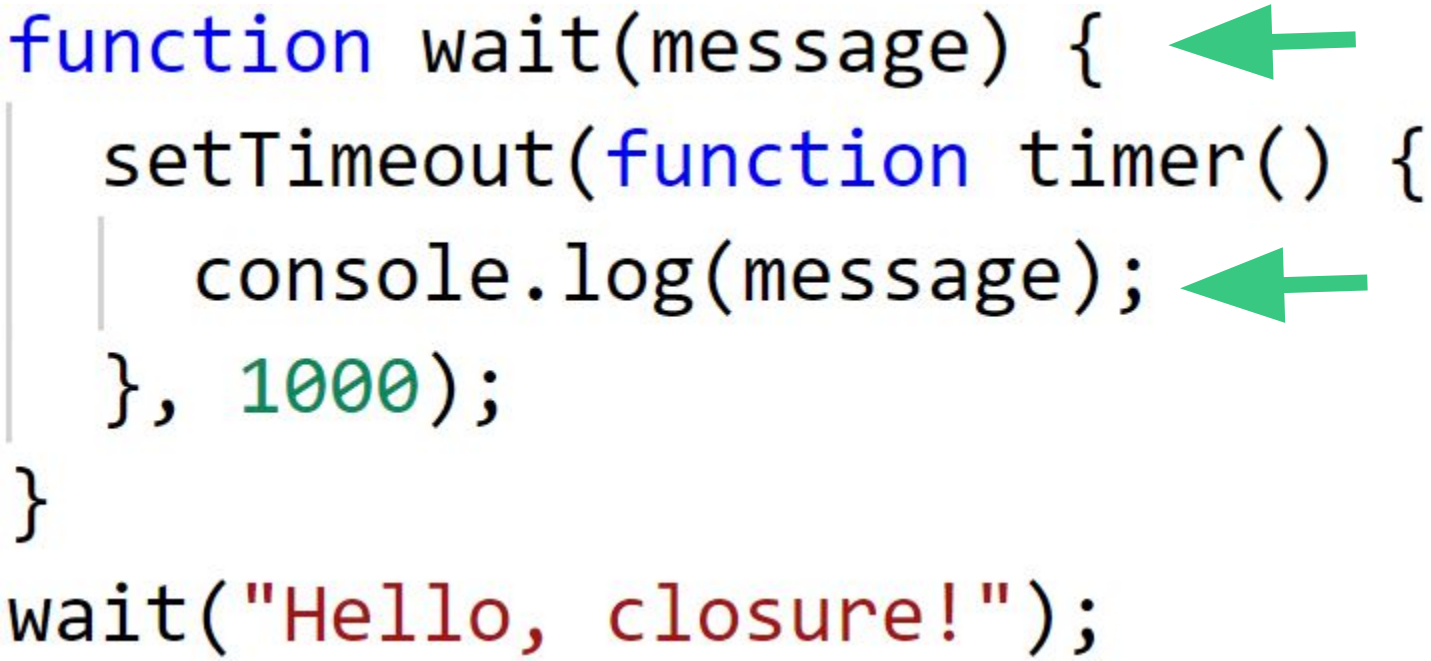
A closure is the **combination of a function and the lexical scope** within which that function was declared.

MDN


FUUUUSION... HA!

Closure is when **a function is able to remember and access its lexical scop**e even when that function is executing outside its lexical scope.

Kyle Simpson


function wait(message) {
setTimeout(function timer() {
console.log(message);
}, 1000);
}
wait("Hello, closure!");

```
function celebrityName(first) {
  var intro = "This is ";

  return function lastName(last) {
    return intro + first + " " + last;
  };
}
var intro = "Hello ";
const mjName = celebrityName("Michael");
mjName("Jackson");
```

```
function celebrityName(first) {
  var intro = "This is ";

  return function lastName(last) {
    return intro + first + " " + last;
  };
}
var intro = "Hello ";
const mjName = celebrityName("Michael");
mjName("Jackson");
//This is Michael Jackson
```

lastName(last)

intro = “This is”

first = “Michael”

last = Jackson

celebrityName(first)

intro = “This is”

first = “Michael”

lastName()

Global Environment

intro = “Hello ”

celebrityName()

mjName()

Slides e Referências

bit.ly/scottiSlides

scottiBR

FeedBack

bit.ly/scottiFeedbacks

guilhermescotti