scottiBR

guilhermescotti

JavaScript The Tough Parts.

Guilherme Scotti

scottiBR

UFMG

guilhermescotti

Engenharia de Software

accenture

aSpect The u rts.

dito

Guilherme Scotti

Por que?

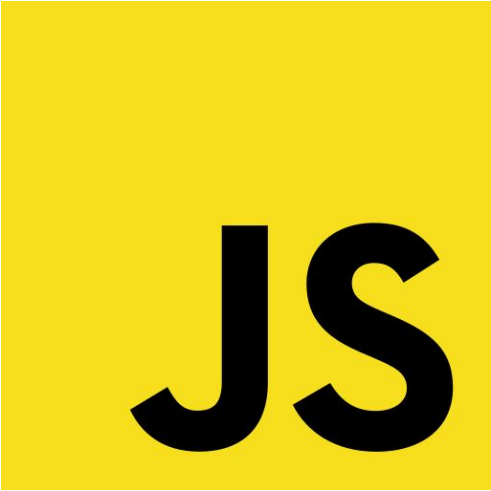
“True mastery means **understanding the core principles** and building up from them.”

William Sentance

bit.ly/scotti-quiz

- JavaScript Under the Hood
- Execution Context e Lexical Environment
- Hoisting
- Lexical Scope
- Closure

Hybrid Language



Compiled Language

Interpreted Language

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

MDN Web Docs

```
        }).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);
```

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

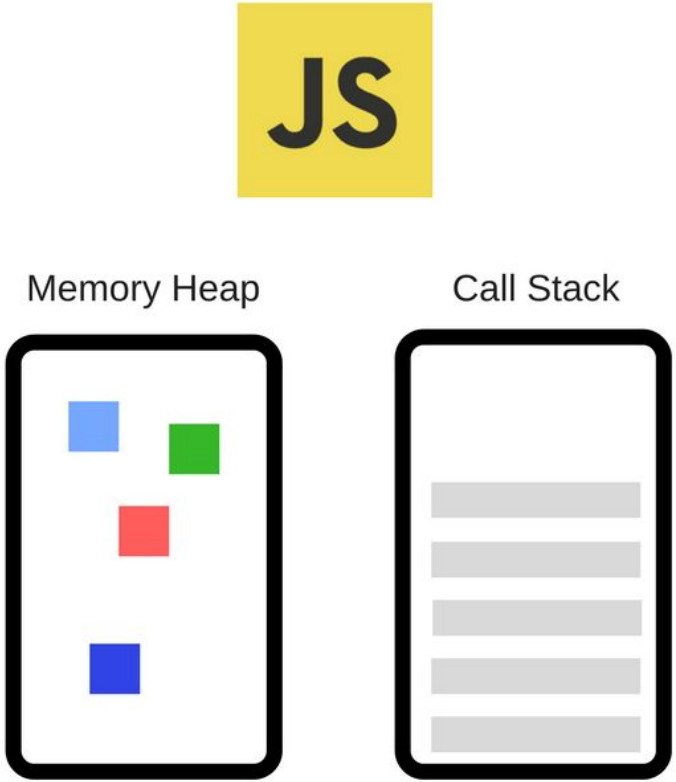
Declarations of variables and functions being

“**moved to top of your code**”

HOISTING

“**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);
}
```

ES6

Arrow Functions

Const

Let

Undefined X Reference Error

```
console.log(x);

let x = "Hello";
```

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

Sukhjinder Arora

Temporal Dead Zone

(TDZ)

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

Porque TDZ existe?

Allen Wirfs-Brock
Chief Project Editor ES6

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

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

```
const words = ["Hello", "TDC"];
words[0] = "bye";
// ['Bye','TDC'];
```

Allen Wirfs-Brock
Project Editor ES6

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

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

a() -> b() -> c() -> a()

```
const a = x => {
  if (x > 20) return x;
  const b = x => {
    const c = x => {
      return a(x * 2);
    };
    return c(x) + 1;
  };
  return b(x + 2);
};
```

```
function a(x) {
  if (x > 20) return x;
  return b(x + 2);
}
function b(x) {
  return c(x) + 1;
}
function c(x) {
  return a(x * 2);
}
```

Lexical Scope

```
var speaker = "Guilherme";
function name() {
  console.log(speaker);
  // ReferenceError or TDZ error
  let speaker = "Scotti";
}
name();
```

Closure

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

MDN

Closure is when a function is **able to remember and access its lexical scope**.

Kyle Simpson

```
      let
for (var i = 0; i < 5; ++i) {
  setTimeout(function() {
    console.log(i);
  }, 1000);
}
// 5 5 5 5 5  // 0 1 2 3 4
```

Slides e Referências

bit.ly/scottiSlides

scottiBR

FeedBack

bit.ly/scottiFeedbacks

guilhermescotti