



JavaScript Avançado I

2.1 Escopos

Prof. Bruno no Augusto Teixeira

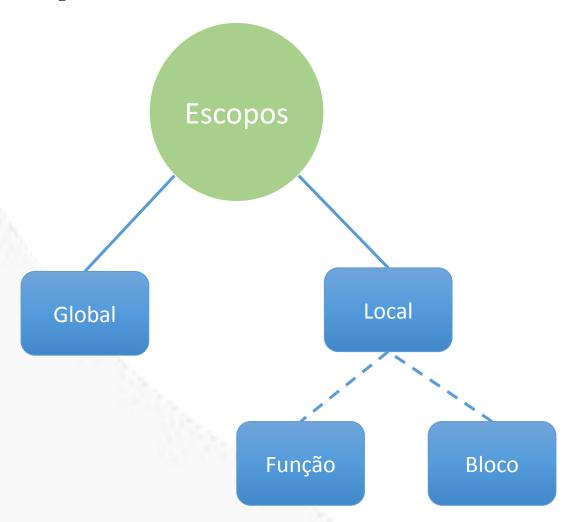
Nesta aula



☐ Escopos.

Escopos





Conclusão



- - ☑ Global.
 - ✓ Local.
 - ☑ Bloco.
 - ☑ Função.
 - ☑ Hoisting.

Próxima aula



Closures.



JavaScript Avançado I

2.2 Closures

Prof. Bruno no Augusto Teixeira

Nesta aula



☐ Closures.

Closures



Encapsulamento.



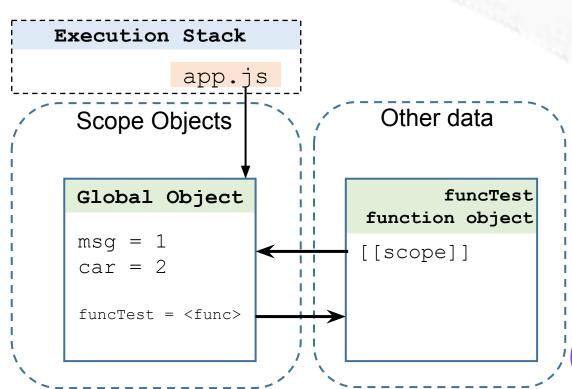
```
var msg = 1;
var car = 2;
```



```
var msg = 1;
var car = 2;

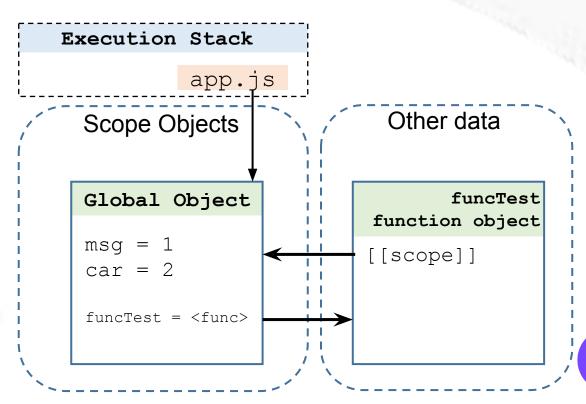
function funcTest() {
   var a = 1;
   var b = 2;
   var msg = 3;

   console.log("Dentro");
}
```

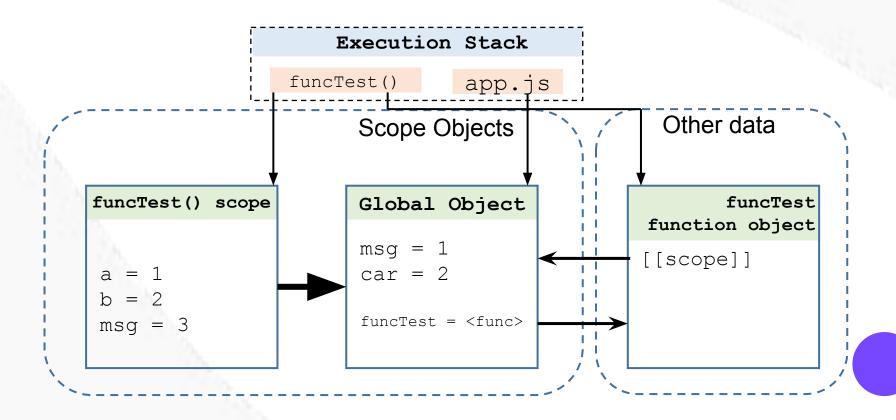




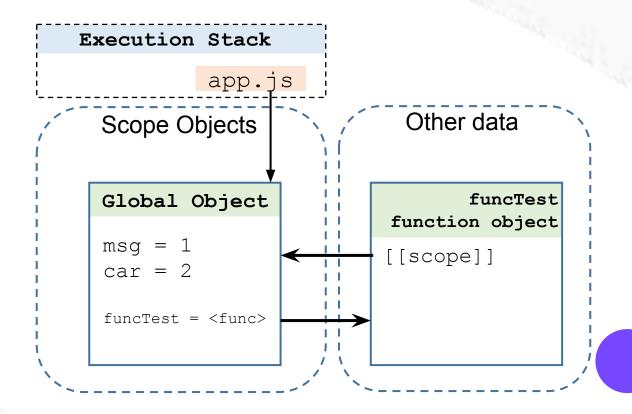
```
var msg = 1;
var car = 2;
function funcTest() {
  var a = 1;
  var b = 2;
  var msg = 3;
  console.log("Dentro");
console.log("Fora");
funcTest();
```





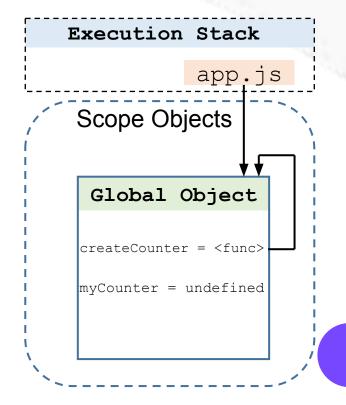






```
function createCounter(initial) {
 var counter = initial;
 function increment(value) {
   if (!isFinite(value) || value < 1) {</pre>
     value = 1;
    counter += value;
 function get() {
   return counter;
 return {
   increment: increment,
   get: get
 };
var myCounter = createCounter(100);
console.log(myCounter.get());
                               // print "100"
myCounter.increment(5);
console.log(myCounter.get());
                               // print "105"
```







```
Execution Stack
function createCounter(initial) {
                                                                            app.js
                                                    createCounter(100)
 var counter = initial
                                                                Scope Objects
 function increment (va
   if (!isFinite(value)
     value = 1;
   counter += value;
                                                                  Global Object
                               createCounter(100) scope
                               initial = 100
 function get() {
                                                                 createCounter = <func>
   return counter;
                               counter = 100
                                                                 myCounter = undefined
 return {
                               increment = <func>
   increment: incremen
                               get = <func>
   get: get
var myCounter = createCounter(100);
```

console.log(myCounter.get()); // print "100"

// print "105"

myCounter.increment(5);

console.log(myCounter.get());



```
Execution Stack
function createCounter(initial) {
                                                                             app.js
                                                     createCounter(100)
 var counter = initial
                                                                 Scope Objects
 function increment (va
   if (!isFinite(value)
     value = 1;
   counter += value;
                                createCounter(100) scope
                                                                   Global Object
                                initial = 100
 function get() {
                                                                  createCounter = <func>
   return counter;
                                counter = 100
                                                                  myCounter = {
                                                                   increment = <func>,
 return {
                                increment = <func>
                                                                   get = <func>
   increment: incremen
                                get = <func>
   get: get
var myCounter = createCounter(100);
```

console.log(myCounter.get()); // print "100"

// print "105"

myCounter.increment(5);

console.log(myCounter.get());

Closures

console.log(myCounter.get()); // print "100"

myCounter.increment(5);

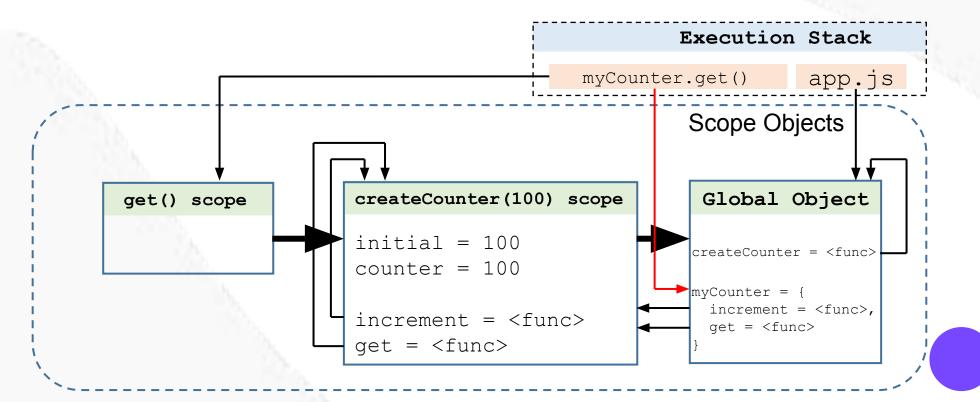
console.log(myCounter.get());



```
Execution Stack
function createCounter(initial) {
                                                                             app.js
 var counter = initial
                                                                 Scope Objects
 function increment (va
   if (!isFinite(value)
     value = 1;
   counter += value;
                                createCounter(100) scope
                                                                   Global Object
                                initial = 100
 function get() {
                                                                  createCounter = <func>
   return counter;
                                counter = 100
                                                                  myCounter = {
                                                                    increment = <func>,
 return {
                                increment = <func>
                                                                    get = <func>
   increment: incremen
                                get = <func>
   get: get
var myCounter = createCounter(100);
```

// print "105"







```
Execution Stack
function createCounter(initial) {
                                                                             app.js
 var counter = initial
                                                                 Scope Objects
 function increment (va
   if (!isFinite(value)
     value = 1;
   counter += value;
                                                                   Global Object
                                createCounter(100) scope
                                initial = 100
 function get() {
                                                                  createCounter = <func>
   return counter;
                                counter = 100
                                                                  myCounter = {
                                                                   increment = <func>,
 return {
                                increment = <func>
                                                                    get = <func>
   increment: incremen
                                get = <func>
   get: get
var myCounter = createCounter(100);
```

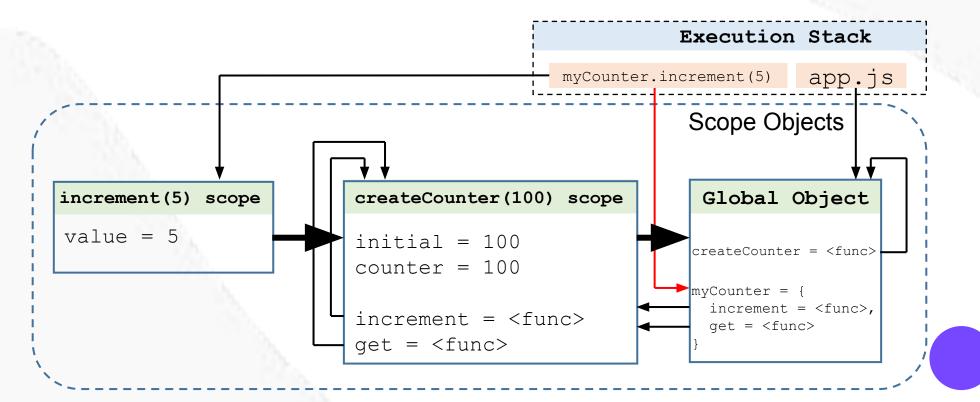
console.log(myCounter.get()); // print "100"

// print "105"

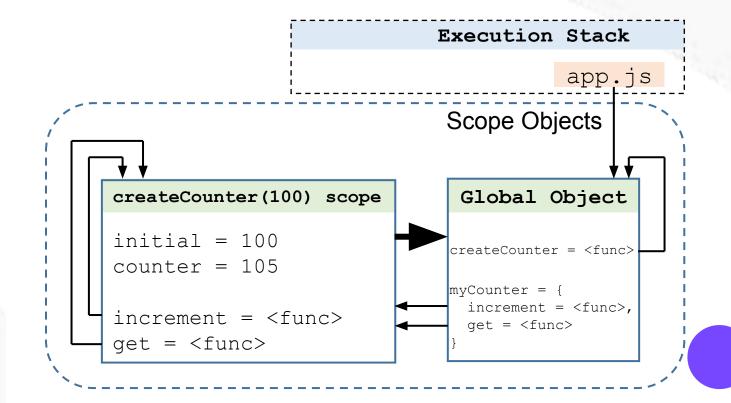
myCounter.increment(5);

console.log(myCounter.get());





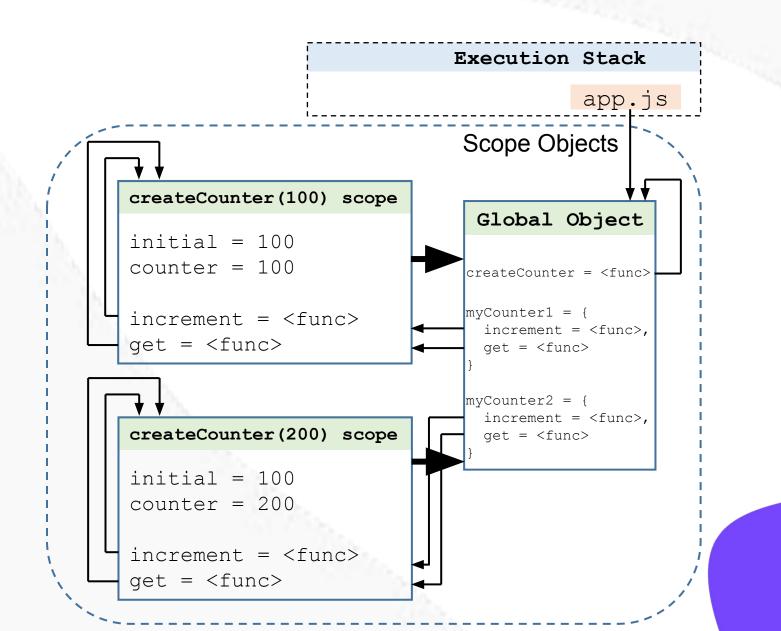






```
function createCounter(initial) {
   /* ... implementação*/
}

var myCounter1 = createCounter(100);
var myCounter2 = createCounter(200);
```





Conclusão



Próxima aula

iGTi

☐ Prototypes.



JavaScript Avançado I

2.3 Prototypes

Prof. Bruno no Augusto Teixeira

Nesta aula



Prototypes.

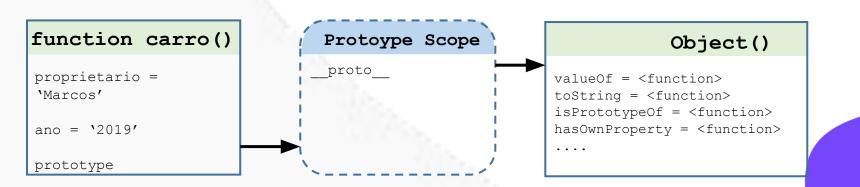
Prototypes



• Herança.

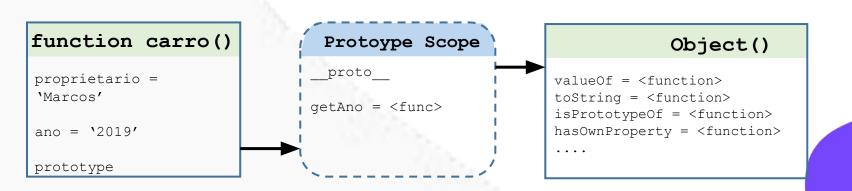


```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
```





```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
Carro.prototype.getAno = function () {
    console.log("Ano: " + this.Ano);
    return this.Ano;
};
```

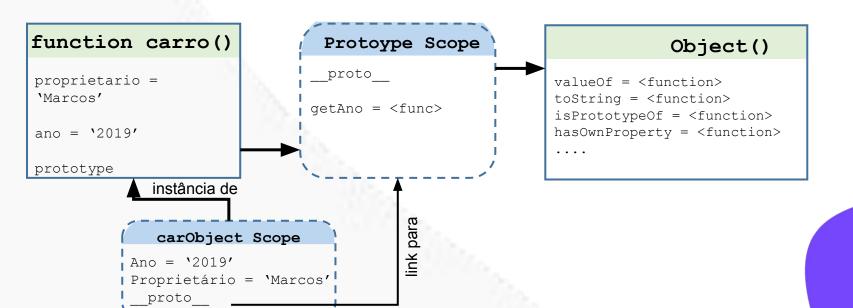




```
function Carro() {
           this.proprietario = 'Marcos';
           this.ano = 2019;
      Carro.prototype.getAno = function () {
           console.log("Ano: " + this.Ano);
           return this. Ano;
      let carObject = new Carro();
function carro()
                         Protoype Scope
                                                      Object()
                         proto
proprietario =
                                            valueOf = <function>
'Marcos'
                                            toString = <function>
                       getAno = <func>
                                            isPrototypeOf = <function>
                                            hasOwnProperty = <function>
ano = 2019'
prototype
```



```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
Carro.prototype.getAno = function () {
    console.log("Ano: " + this.Ano);
    return this.Ano;
};
let carObject = new Carro();
```





```
function Carro() {
     this.proprietario = 'Marcos';
     this.ano = 2019;
 Carro.prototype.getAno = function () {
     console.log("Ano: " + this.Ano);
     return this. Ano;
 let carObject = new Carro();
 carObject.toString();
function carro()
                               Protoype Scope
                                                                   Object()
                               proto
proprietario =
                                                       valueOf = <function
'Marcos'
                                                       toString = <function>
                             getAno = <func>
                                                       isPrototypeOf = <function>
                                                       hasOwnProperty = <function>
ano = ^{1}2019'
prototype
            instância de
             carObject Scope
          Ano = '2019'
         Proprietário = 'Marcos'
```

Conclusão



☑ Prototypes.

Próxima aula

igti

☐ IIFE – Funções Imediatas.



JavaScript Avançado I

2.4 IIFE – Funções Imediatas

Prof. Bruno no Augusto Teixeira

Nesta aula



☐ IIFE – Funções Imediatas.



Immediately Invoked Function Expression.



Function Declaration:

```
function myFunction () {
   /* código */
}
```

Function Expression

```
let myFunction = function() {
    /* código */
};
```



Immediately Invoked Function Expression.

```
( function () {} )();
```



- Poluição do escopo global.
- Privacidade de dados.
- Closures.
- Renomear variáveis.
- Capturar o objeto Global.

Conclusão



☑ IIFE.

Próxima aula

iGTi

☐ Proxy.



JavaScript Avançado I

2.5 Proxy

Prof. Bruno no Augusto Teixeira

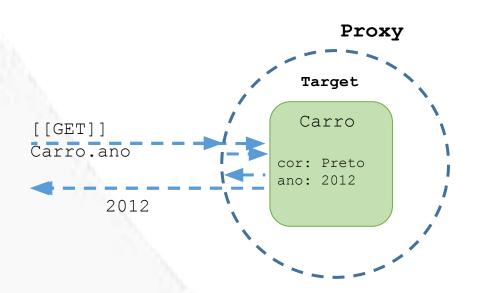
Nesta aula

iGTi

- ☐ Proxy.
- ☐ Reflect.

Proxy





Conclusão



- ✓ Proxy.
- ☑ Reflect.

Próxima aula

iGTi

☐ Curry.



JavaScript Avançado I

2.6 Curry

Prof. Bruno no Augusto Teixeira

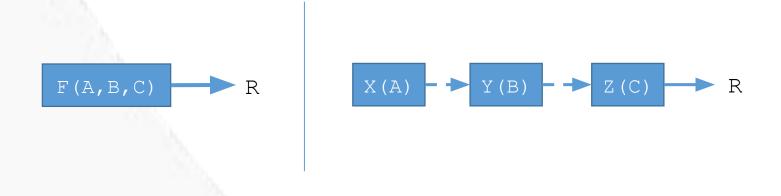
Nesta aula



☐ Curry.

Currying





Conclusão

igti

☑ Currying.

Próxima aula

igti

☐ JavaScript Assíncrono.