JavaScript Avançado I

Capítulo 1. Mapa de Eventos

Prof. Bruno Augusto Teixeira

JavaScript Avançado I

1.1 Manipuladores de Evento

Prof. Bruno no Augusto Teixeira

Nesta aula

- Manipuladores de Evento;
- Arquitetura do Evento;
- *Interface Event.*

Manipuladores de Evento

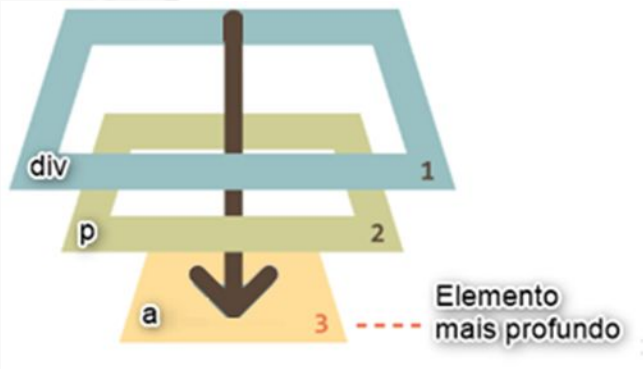
- Manipuladores:
  - Manipuladores in-line;
  - Propriedades dos elementos;
  - Métodos DOM 2:
    - addEventListener();
    - removeEventListener().
- Argumentos de eventos.

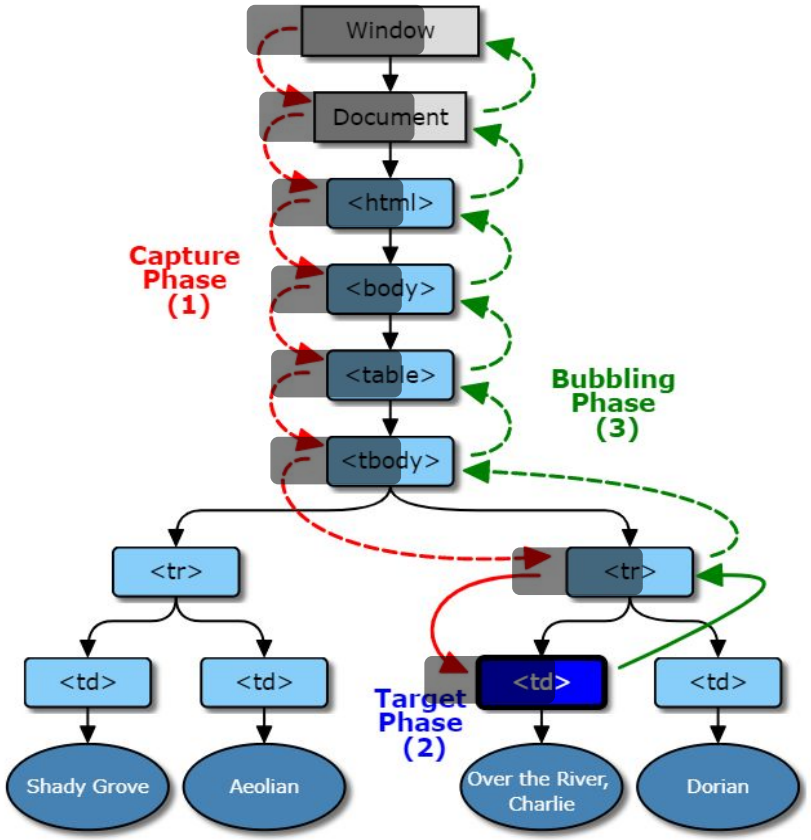
Arquitetura do Evento

- Fases dos Eventos:
  - Capturing;
  - Target;
  - Bubbling.

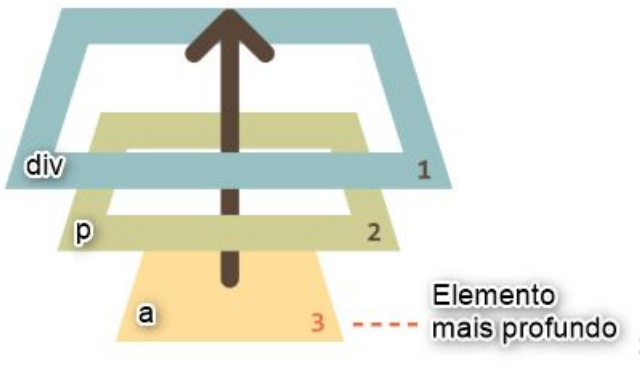
- Interrupção da propagação.

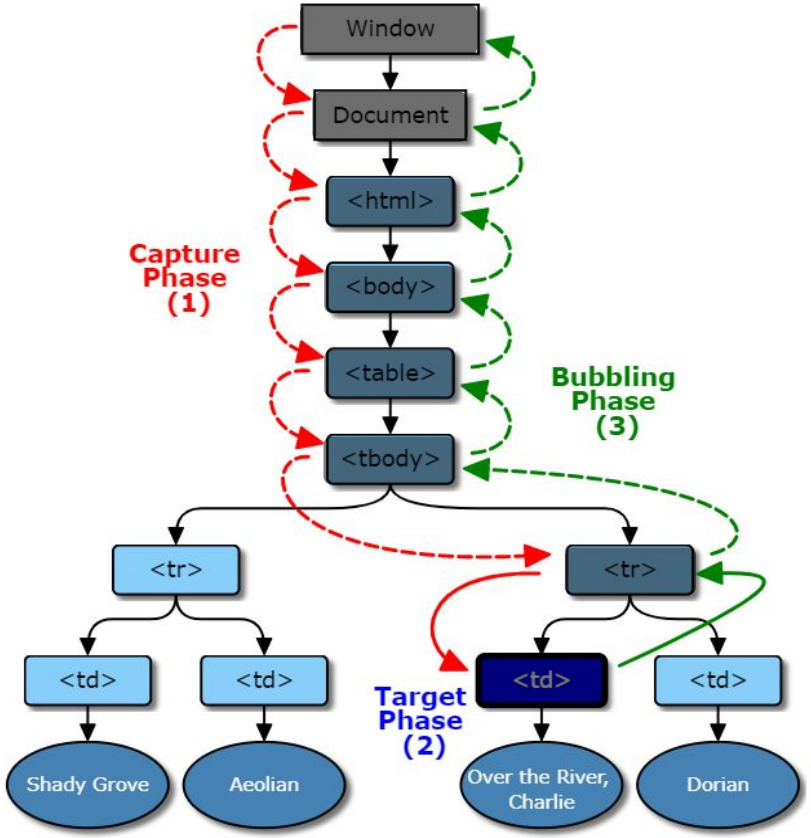
Capturing


div
1
p
2
a
3
Elemento
mais profundo

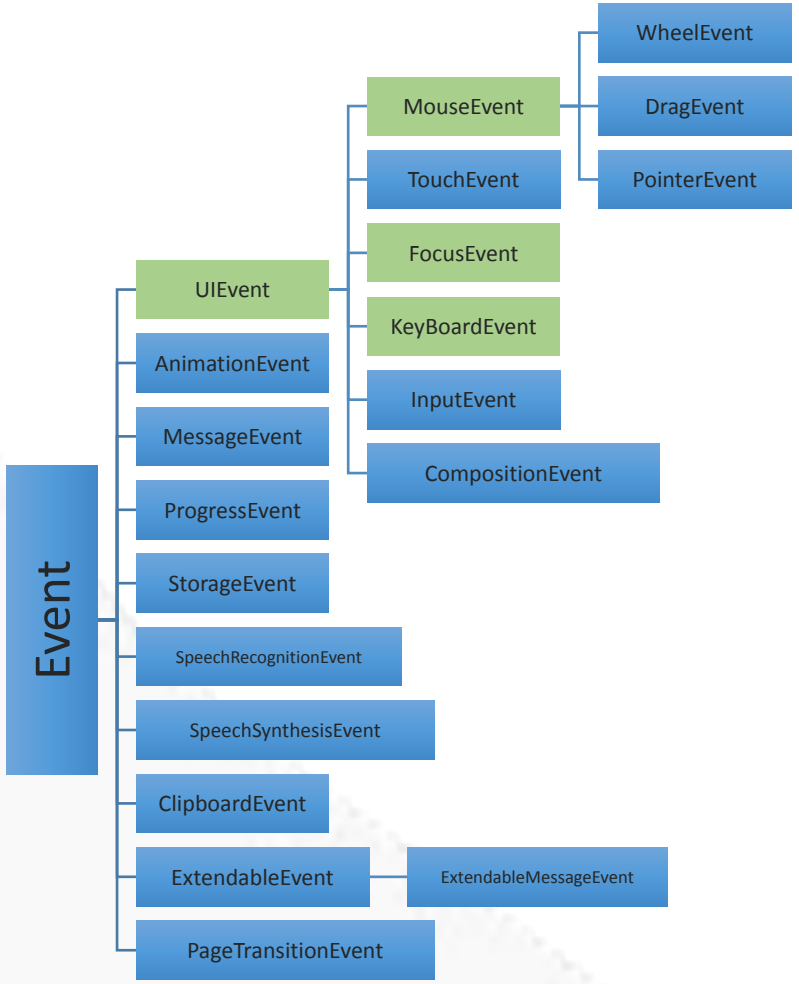

Window
Document
<html>
<body>
<table>
<tbody>
<tr>
<tr>
<td>
<td>
<td>
<td>
Shady Grove
Aeolian
Over the River, Charlie
Dorian
Capture Phase (1)
Bubbling Phase (3)
Target Phase (2)

Bubbling


div
1
p
2
a
3
Elemento
mais profundo


Window
Document
<html>
<body>
<table>
<tbody>
<tr>
<tr>
<td>
<td>
<td>
<td>
Shady Grove
Aeolian
Over the River, Charlie
Dorian
Capture Phase (1)
Bubbling Phase (3)
Target Phase (2)

Interface Event



Conclusão

- ☑ Manipuladores de Evento;
- ☑ Arquitetura do Evento;
- ☑ *Interface Event*.

Próxima aula

- UIEvent – Eventos da Interface de Usuário.

JavaScript Avançado I

1.2 UIEvent – Eventos da Interface de Usuário

Prof. Bruno no Augusto Teixeira

Nesta aula

- UIEvent.

UIEvent

- Eventos:
  - *load, unload, abort, error, select e resize.*

- Propriedades:
  - *view e detail.*

Conclusão

- ☑ UIEvent.

Próxima aula

- MouseEvent – Eventos do Mouse.

JavaScript Avançado I

1.3 MouseEvent – Eventos do Mouse

Prof. Bruno no Augusto Teixeira

Nesta aula

- MouseEvent.

MouseEvent

- Eventos:
  - *Mousedown, mouseup, mouseover, mouseout, mousemove.*
  - *Click, dblclick, mouseenter, mouseleave, e contextmenu.*
- Ordem dos Eventos:
  - *Mousedown -> mouseup -> click.*
- Propriedades:
  - *ScreenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, metaKey, button, buttons, relatedTargets.*

MouseEvent

- Mouseover e mouseout:

  - RelatedTarget.

- Mouseenter e mouseleave.

Conclusão

- ☑ MouseEvent

Próxima aula

- KeyboardEvent – Eventos do Teclado,

JavaScript Avançado I

1.4 KeyboardEvent – Eventos do Teclado

Prof. Bruno no Augusto Teixeira

Nesta aula

- KeyboardEvent.

KeyboardEvent

- Eventos:
  - *Keydown e keyup.*
- Ordem dos Eventos:
  - Keydown, beforeinput, input e keyup.
- Propriedades
  - *Key, code, location, altKey, shiftKey, ctrlKey, metaKey, repeat, isComposing.*

Conclusão

- ☑ KeyboardEvent.

Próxima aula

- FocusEvent – Eventos de Foco.

JavaScript Avançado I

1.5 – FocusEvent – Eventos de Foco

Prof. Bruno no Augusto Teixeira

Nesta aula

- FocusEvent.

FocusEvent

- Eventos:
  - *Focusin, focusout, blur e focus.*

- Ordem dos Eventos:
  - Focusin -> focus.
  - Focusout -> focusin -> blur -> focus.

- Propriedades:
  - *RelatedTarget.*

Conclusão

- ☑ FocusEvent

Próxima aula

- JavaScript: Funções.

JavaScript Avançado I

Capítulo 2. JavaScript: Funções

Prof. Bruno no Augusto Teixeira

JavaScript Avançado I

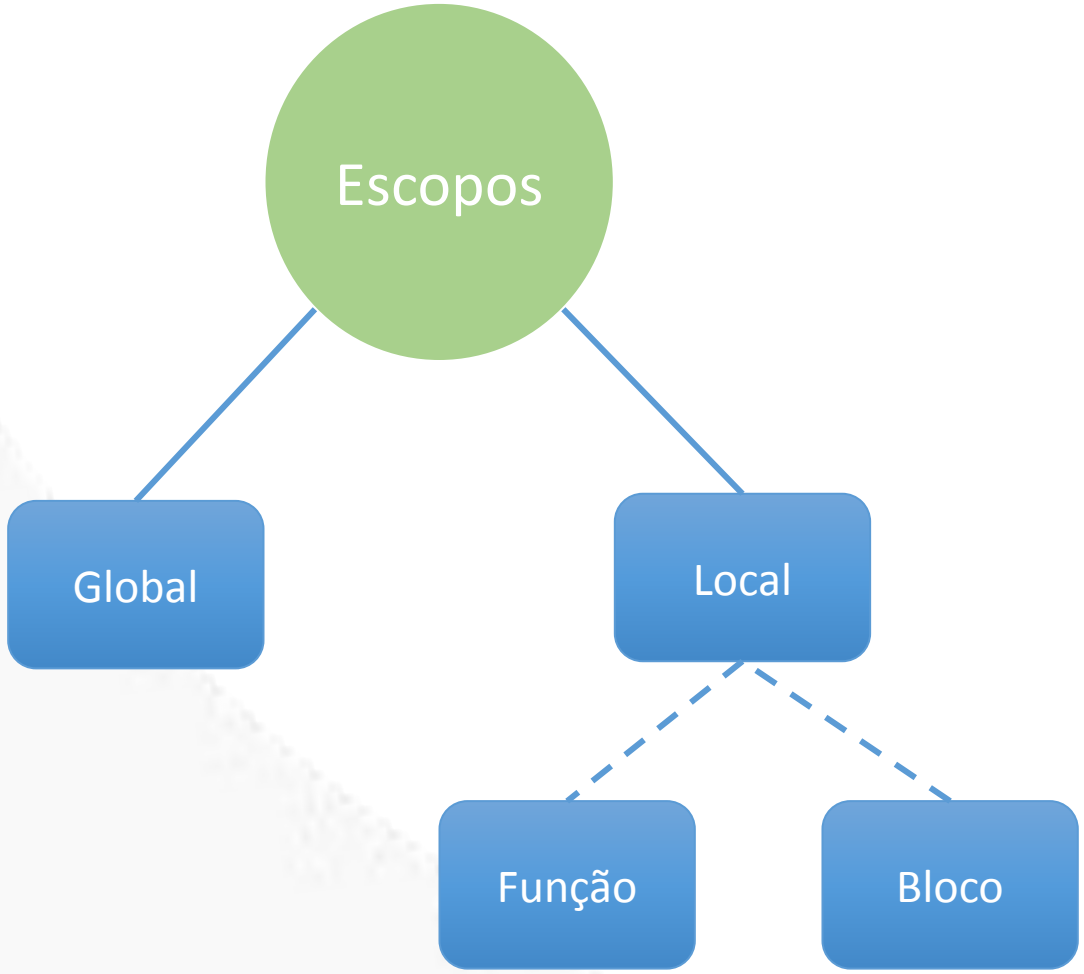
2.1 Escopos

Prof. Bruno no Augusto Teixeira

Nesta aula

- Escopos.

Escopos



Conclusão

- ☑ Escopos:
  - ☑ Global.
  - ☑ Local.
  - ☑ Bloco.
  - ☑ Função.
  - ☑ Hoisting.

Próxima aula

- Closures.

JavaScript Avançado I

2.2 Closures

Prof. Bruno no Augusto Teixeira

Nesta aula

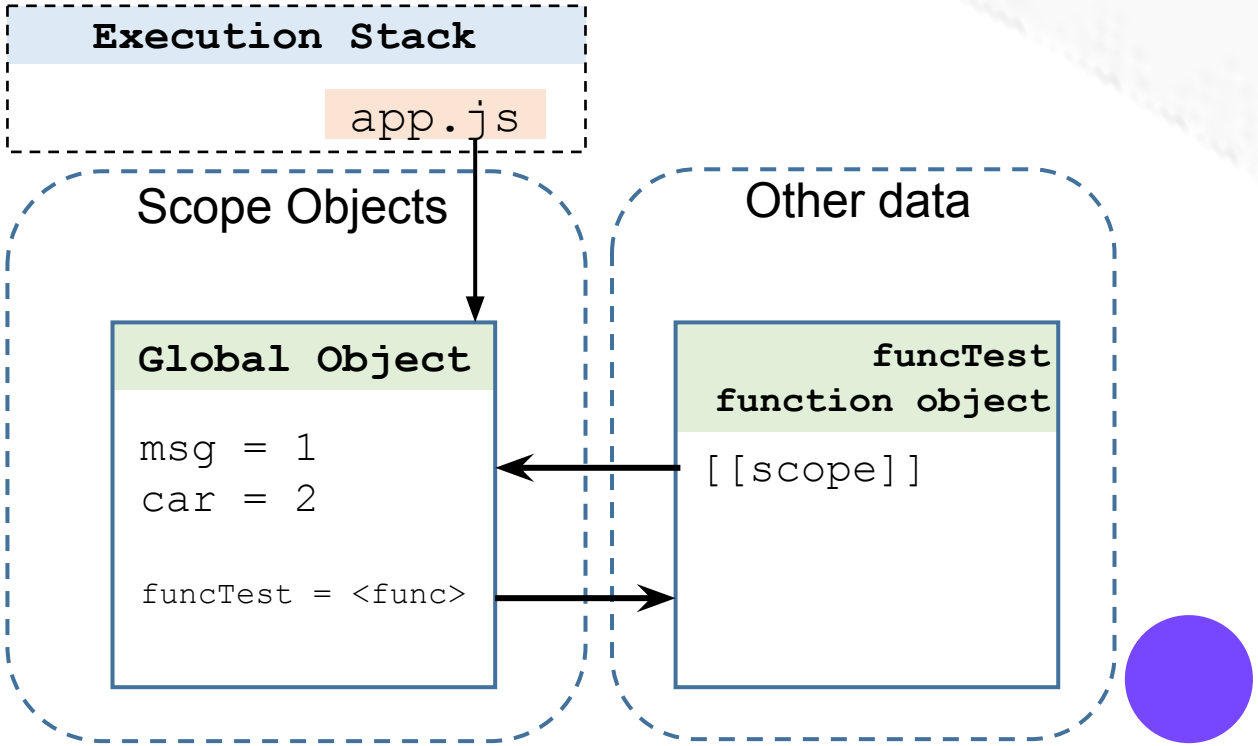
- Closures.

Closures

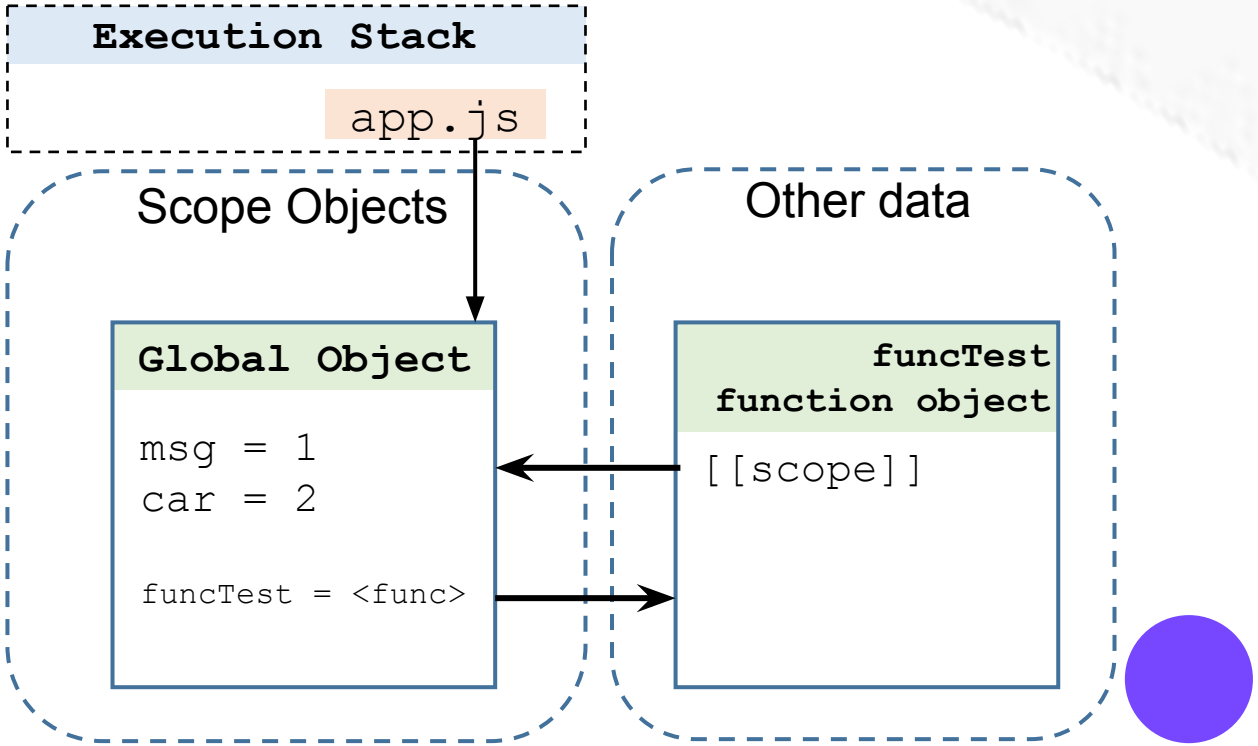
- Encapsulamento.

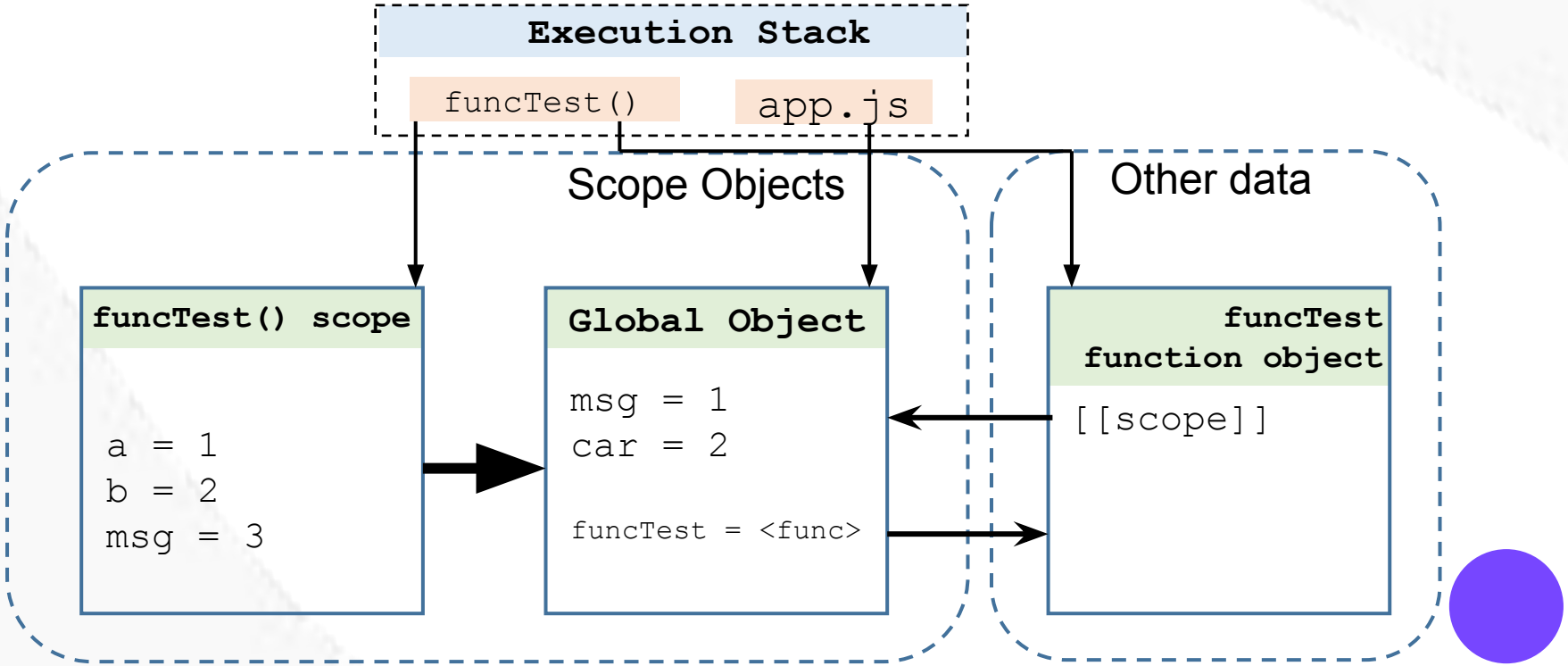
```
var msg = 1;
var car = 2;
```

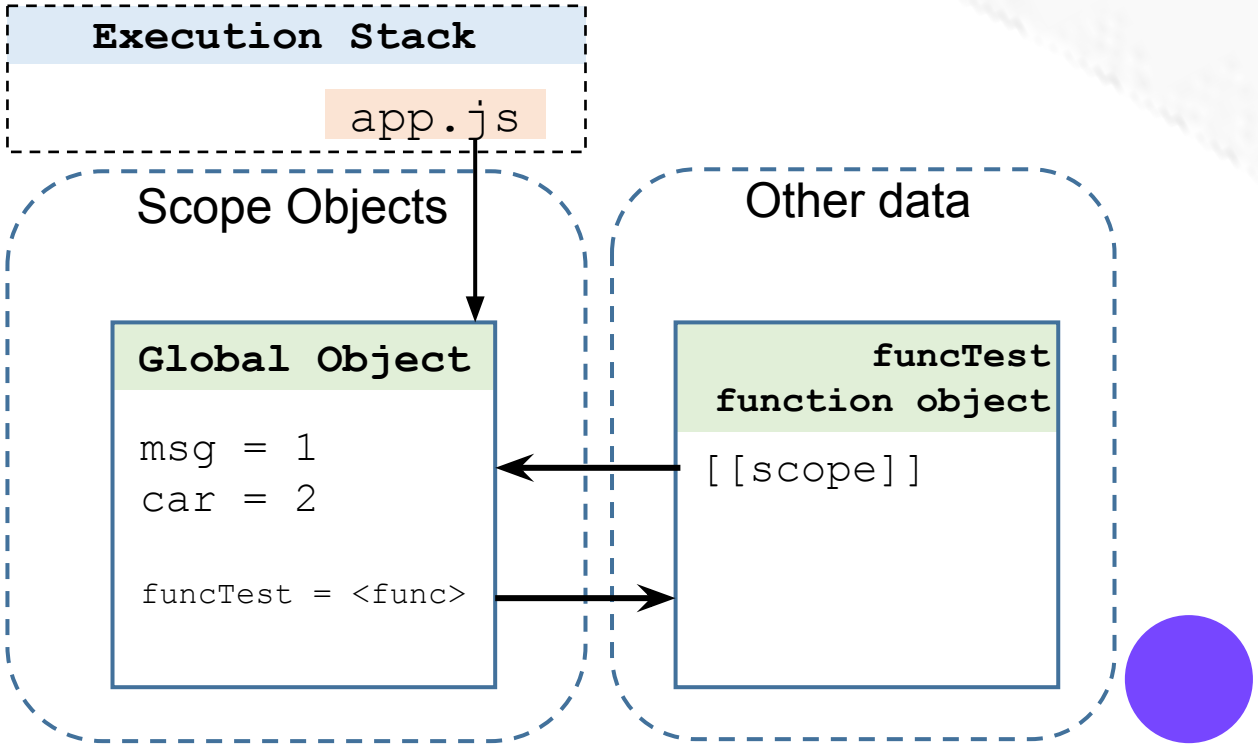
```
var msg = 1;
var car = 2;

function funcTest() {

  var a = 1;
  var b = 2;
  var msg = 3;

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



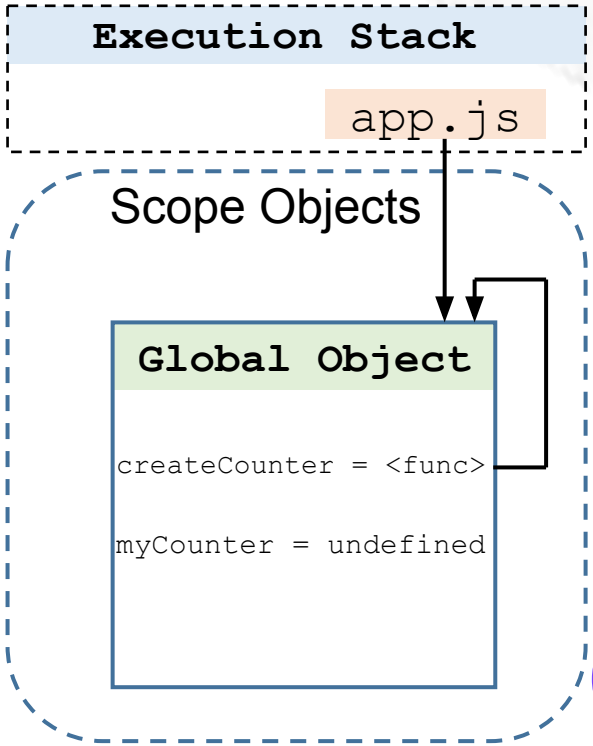
```
var msg = 1;
var car = 2;

function funcTest() {

  var a = 1;
  var b = 2;
  var msg = 3;

  console.log("Dentro");
}

console.log("Fora");

funcTest();
```



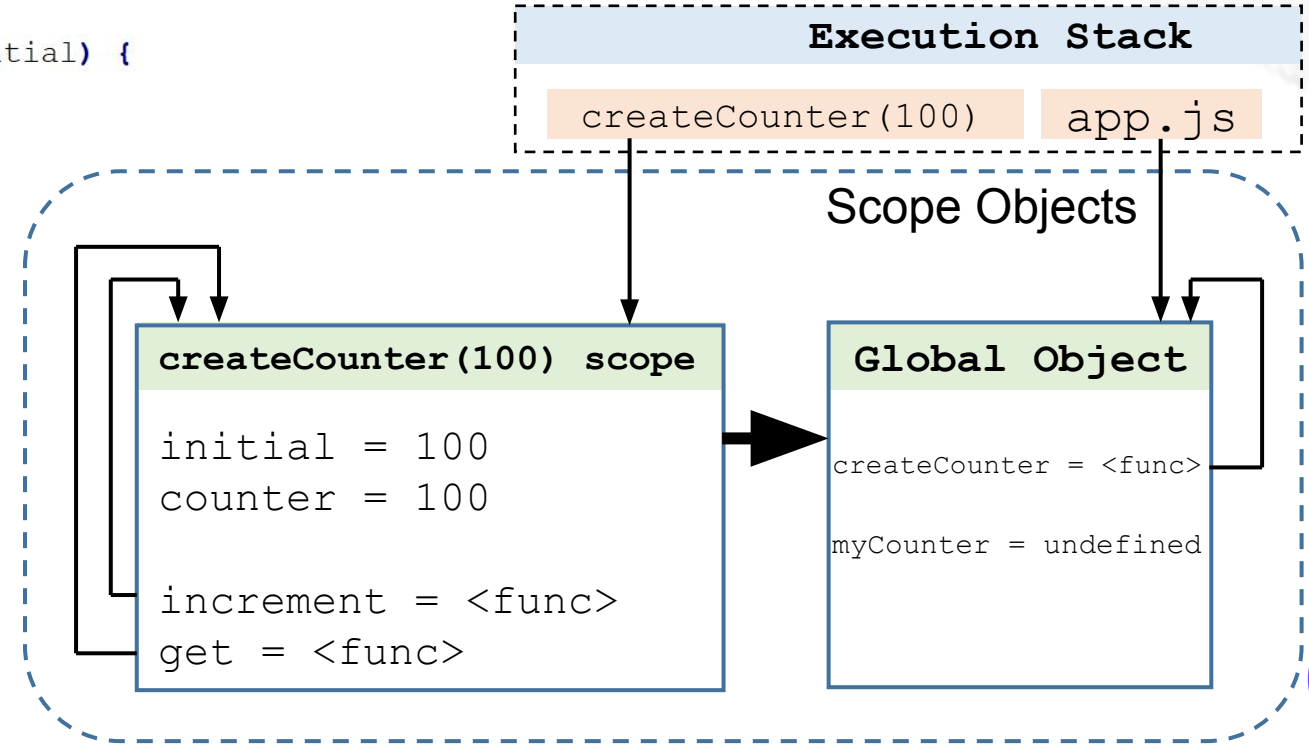

Execution Stack
funcTest()
app.js
Scope Objects
Other data
funcTest() scope
a = 1
b = 2
msg = 3
Global Object
msg = 1
car = 2
funcTest = <func>
funcTest
function object
[[scope]]


Execution Stack
app.js
Scope Objects
Other data
Global Object
msg = 1
car = 2
funcTest = <func>
funcTest
function object
[[scope]]

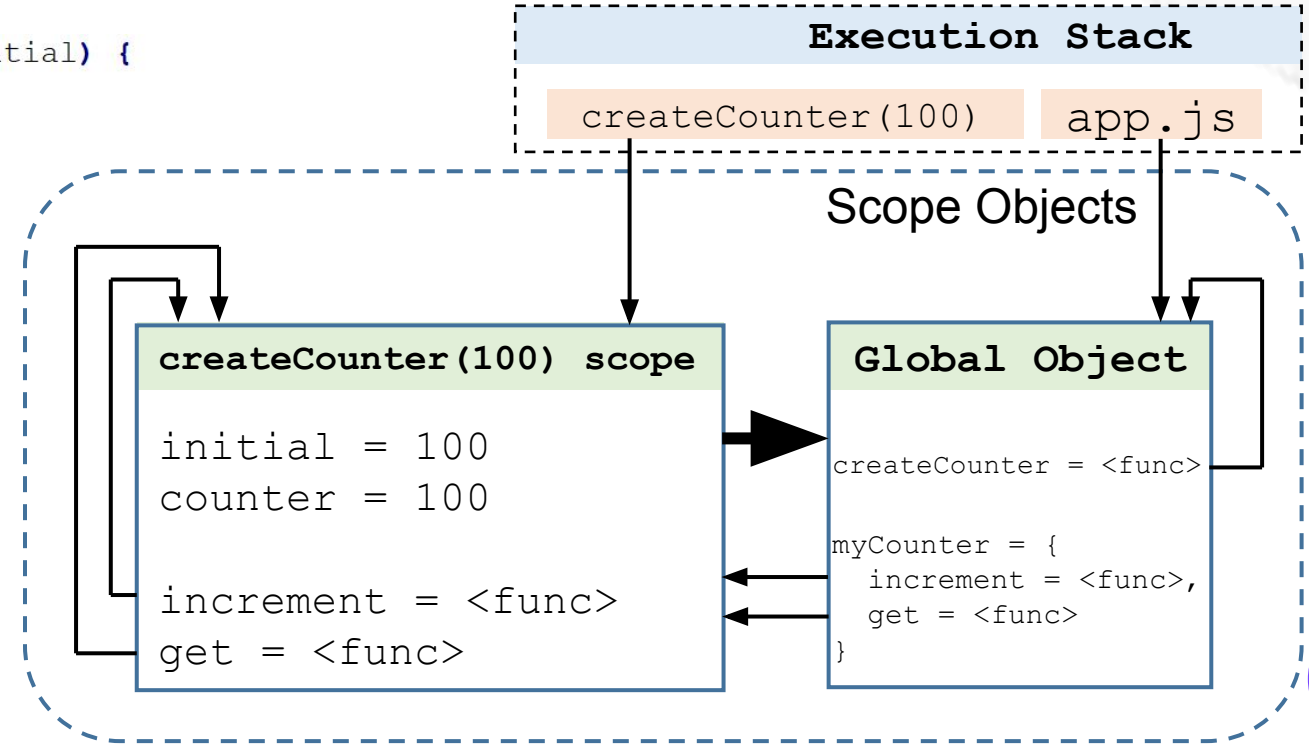
```
function createCounter(initial) {

  var counter = initial;

  function increment(value) {
    if (!isFinite(value) || value < 1){
      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"
```



```
function createCounter(initial) {

  var counter = initial

  function increment(va
    if (!isFinite(value
      value = 1;
    }
    counter += value;
  }

  function get() {
    return counter;
  }

  return {
    increment: incremen
    get: get
  };
}

var myCounter = createCounter(100);

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

myCounter.increment(5);
console.log(myCounter.get());   // print "105"
```

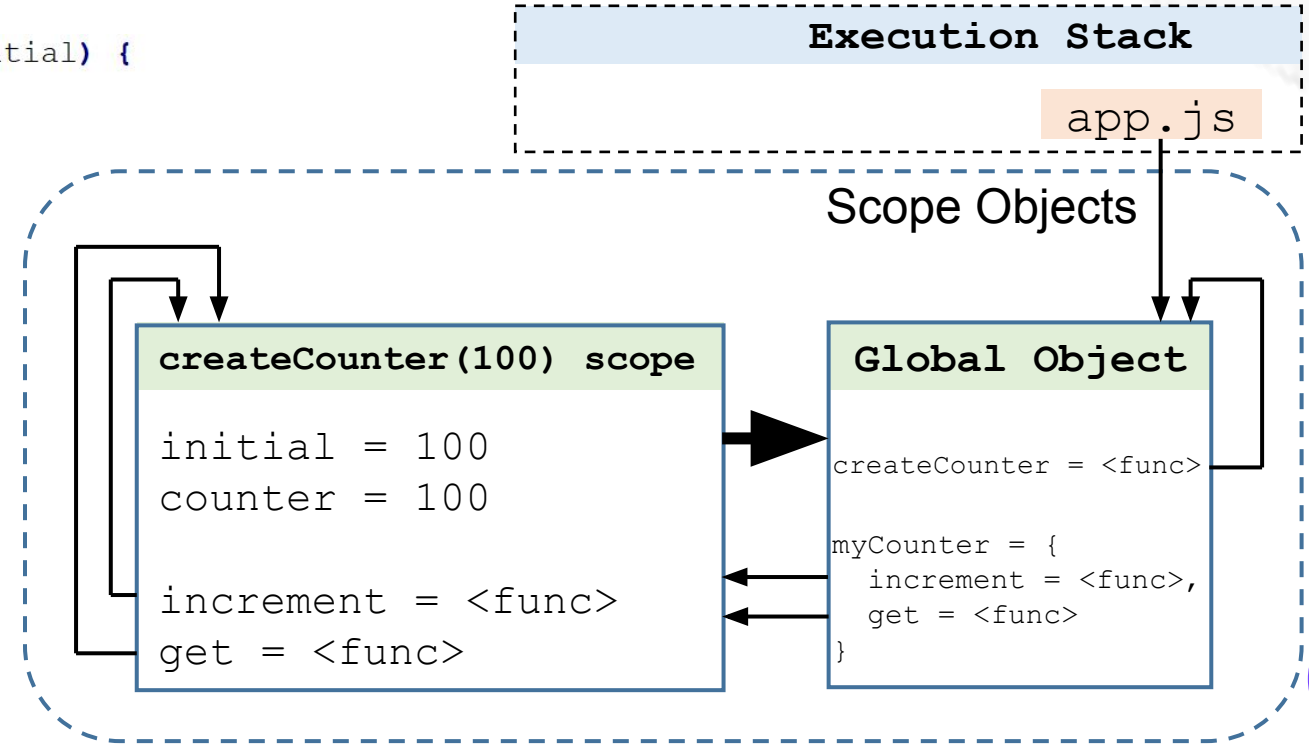


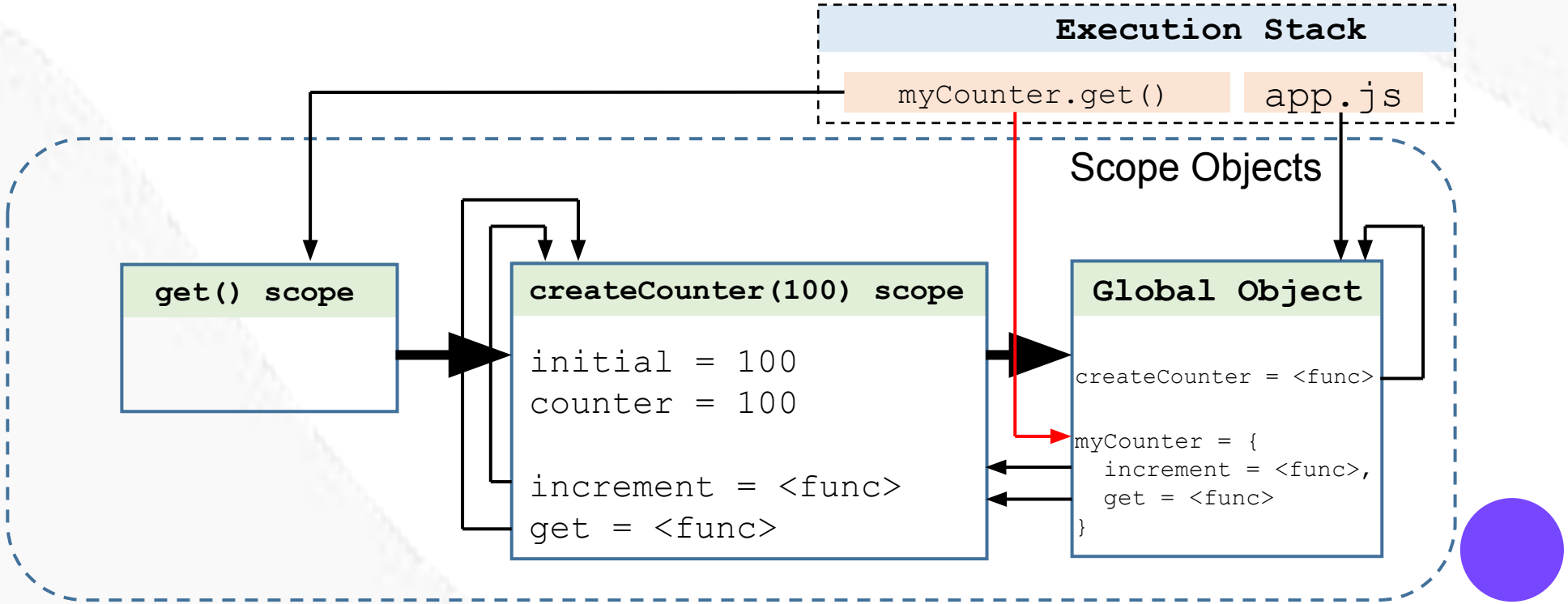
```
function createCounter(initial) {

  var counter = initial

  function increment(va
    if (!isFinite(value
      value = 1;
    }
    counter += value;
  }

  function get() {
    return counter;
  }

  return {
    increment: incremen
    get: get
  };
}

var myCounter = createCounter(100);

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

myCounter.increment(5);
console.log(myCounter.get());    // print "105"
```



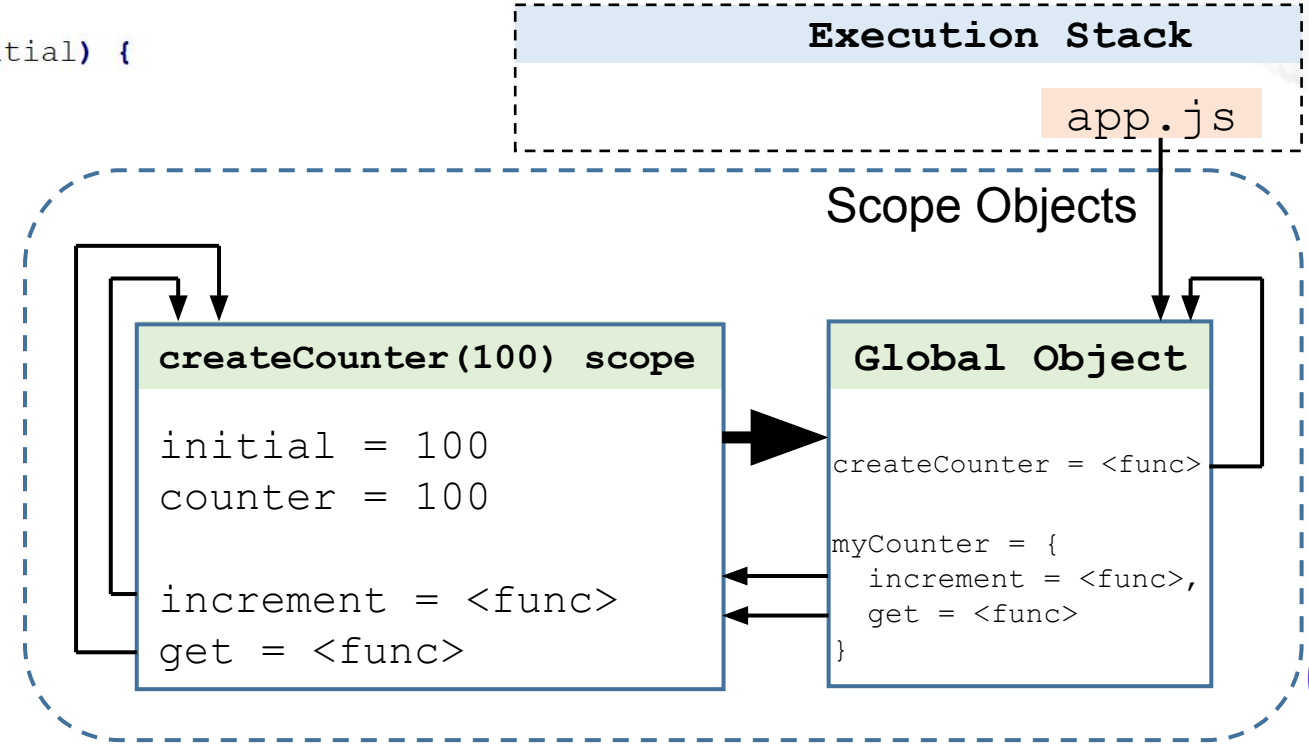
Closures

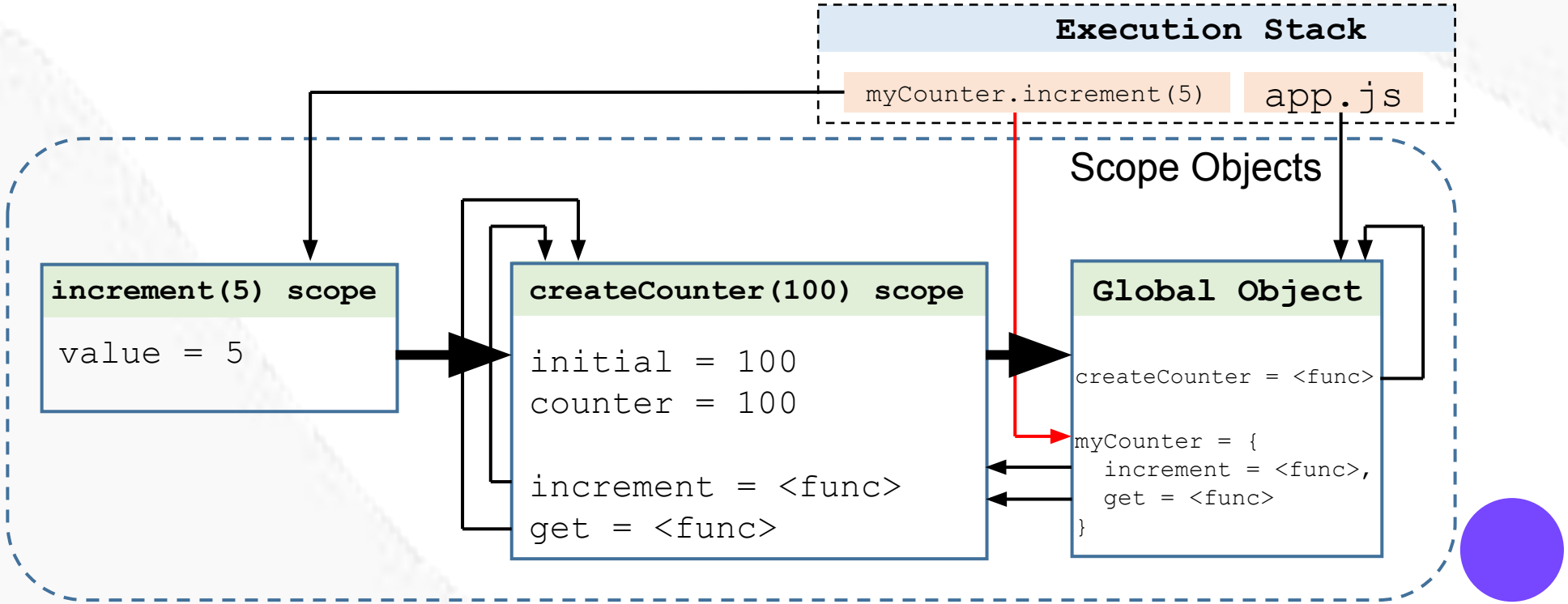
```
function createCounter(initial) {

  var counter = initial

  function increment(va
    if (!isFinite(value
      value = 1;
    }
    counter += value;
  }

  function get() {
    return counter;
  }

  return {
    increment: incremen
    get: get
  };
}

var myCounter = createCounter(100);

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

myCounter.increment(5);
console.log(myCounter.get());    // print "105"
```

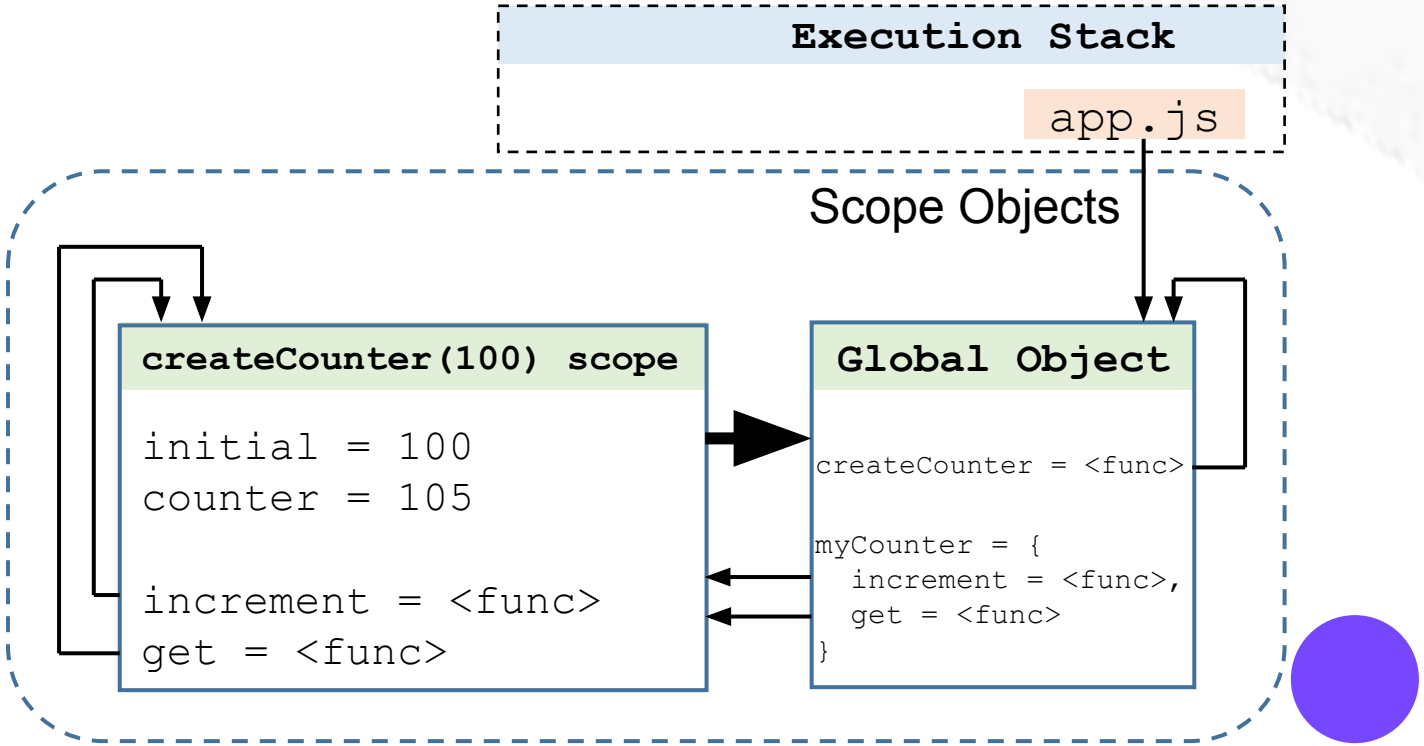



Execution Stack
myCounter.get()
app.js
Scope Objects
get() scope
createCounter(100) scope
initial = 100
counter = 100
increment = <func>
get = <func>
Global Object
createCounter = <func>
myCounter = {
increment = <func>,
get = <func>
}

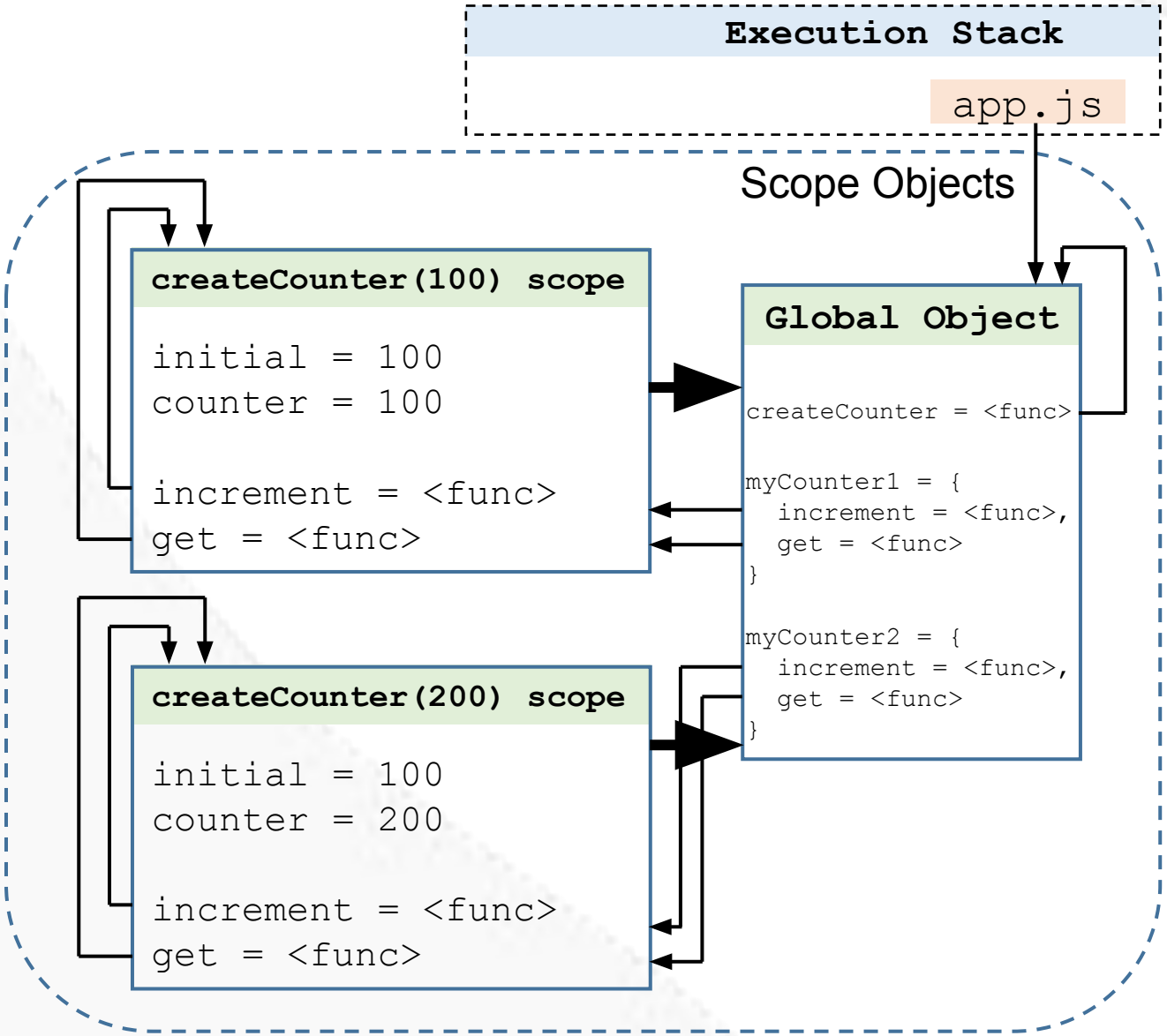
```
function createCounter(initial) {

  var counter = initial

  function increment(va
    if (!isFinite(value
      value = 1;
    }
    counter += value;
  }

  function get() {
    return counter;
  }

  return {
    increment: incremen
    get: get
  };
}

var myCounter = createCounter(100);

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

myCounter.increment(5);
console.log(myCounter.get());    // print "105"
```




Execution Stack
myCounter.increment(5)
app.js
Scope Objects
increment(5) scope
value = 5
createCounter(100) scope
initial = 100
counter = 100
increment = <func>
get = <func>
Global Object
createCounter = <func>
myCounter = {
increment = <func>,
get = <func>
}


Execution Stack
app.js
Scope Objects
createCounter(100) scope
initial = 100
counter = 105
increment = <func>
get = <func>
Global Object
createCounter = <func>
myCounter = {
increment = <func>,
get = <func>
}

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


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


Execution Stack
app.js
Scope Objects
createCounter(100) scope
initial = 100
counter = 100
increment = <func>
get = <func>
Global Object
createCounter = <func>
myCounter1 = {
increment = <func>,
get = <func>
}
myCounter2 = {
increment = <func>,
get = <func>
}
createCounter(200) scope
initial = 100
counter = 200
increment = <func>
get = <func>

Conclusão

- ☑ Closures.

Próxima aula

- Prototypes.

JavaScript Avançado I

2.3 Prototypes

Prof. Bruno no Augusto Teixeira

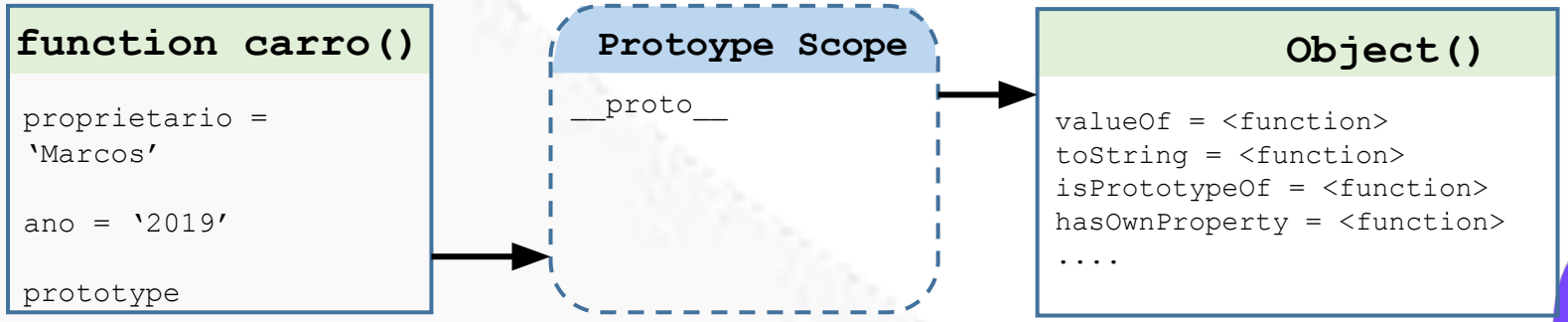
Nesta aula

- Prototypes.

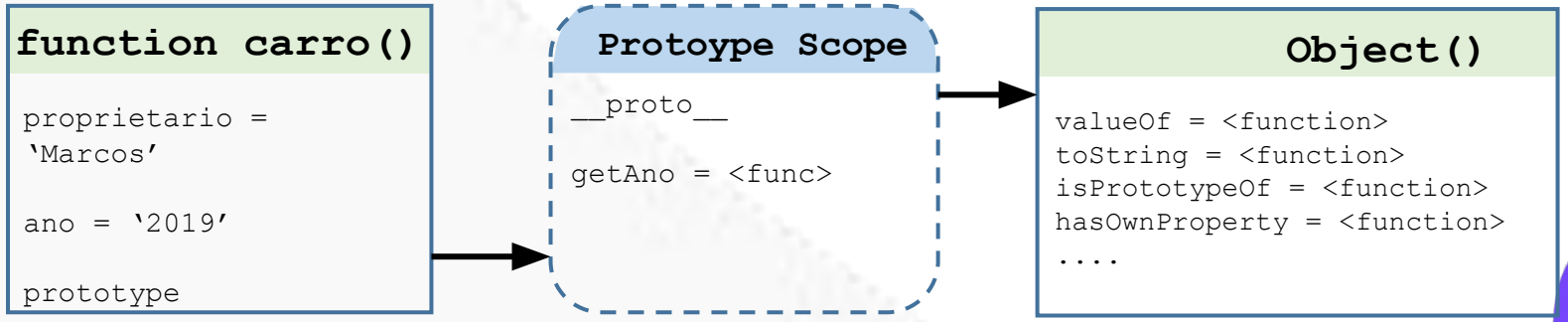
Prototypes

- Herança.

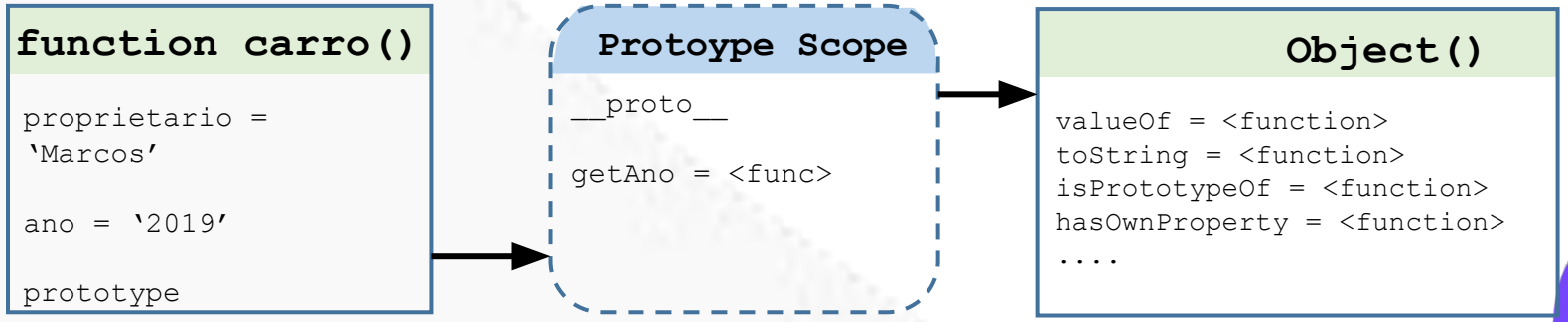
```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
```



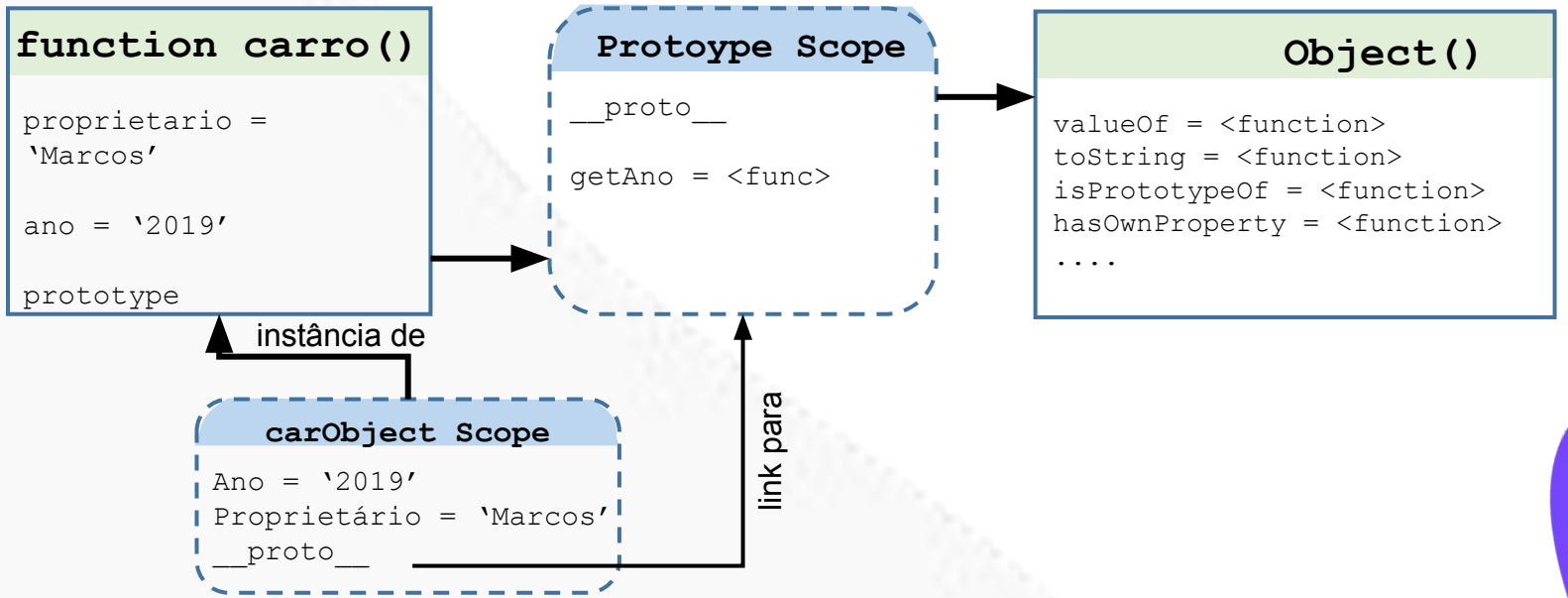
```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
Carro.prototype.getAno = function () {
    console.log("Ano: " + this.Ano);
    return this.Ano;
};
```



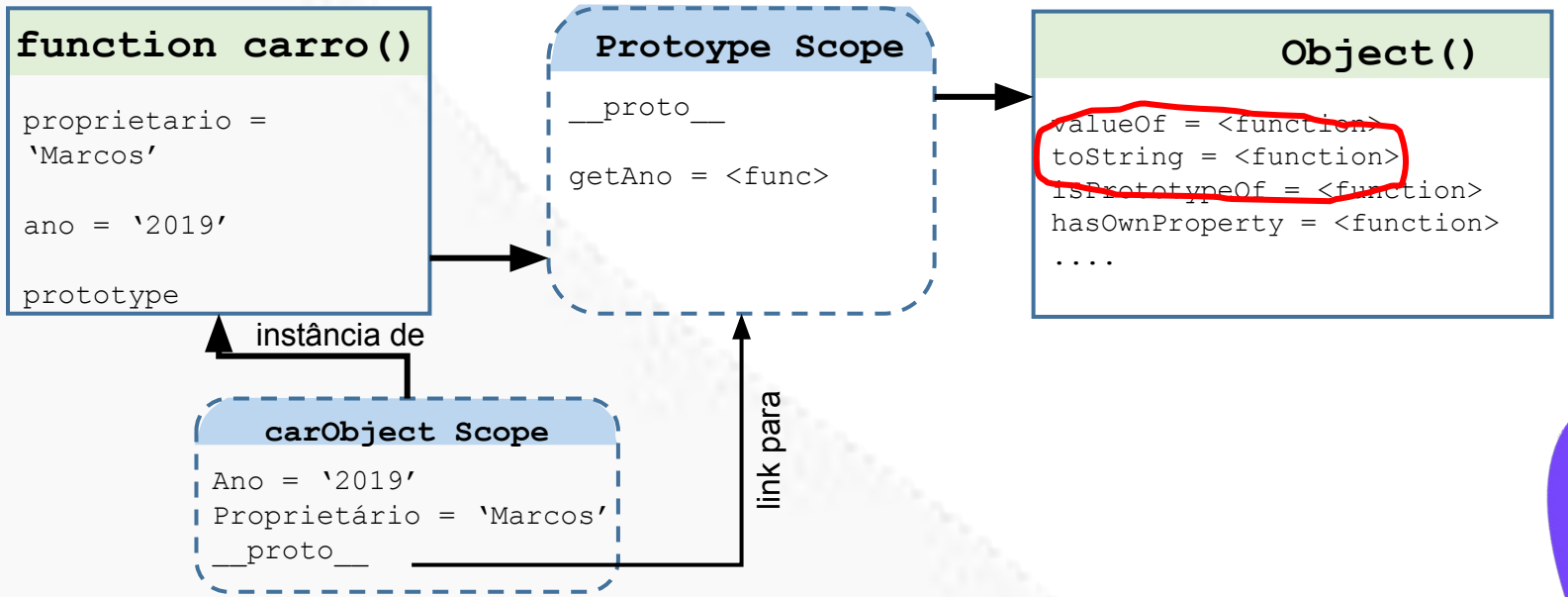
```
function Carro() {
    this.proprietario = 'Marcos';
    this.ano = 2019;
}
Carro.prototype.getAno = function () {
    console.log("Ano: " + this.Ano);
    return this.Ano;
};
let carObject = new Carro();
```



```
function Carro() {
    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();
```



Conclusão

☑ Prototypes.

Próxima aula

- IIFE – Funções Imediatas.

JavaScript Avançado I

2.4 IIFE – Funções Imediatas

Prof. Bruno no Augusto Teixeira

Nesta aula

- IIFE – Funções Imediatas.

IIFE

Immediately Invoked Function Expression.

IIFE

Function Declaration:

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

Function Expression

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

IIFE

Immediately Invoked Function Expression.

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

IIFE

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

Conclusão

☑ IIFE.

Próxima aula

- Proxy.

JavaScript Avançado I

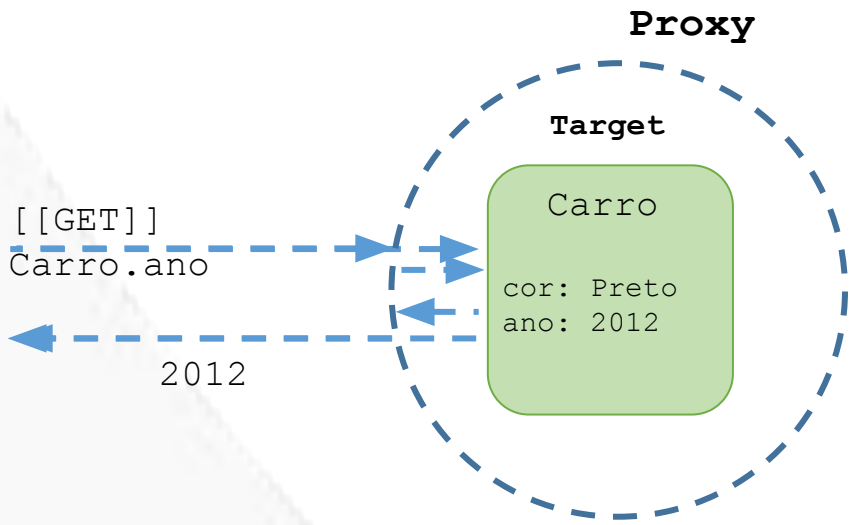
2.5 Proxy

Prof. Bruno no Augusto Teixeira

Nesta aula

- Proxy.
- Reflect.

Proxy



Conclusão

- ☑ Proxy.
- ☑ Reflect.

Próxima aula

- Curry.

JavaScript Avançado I

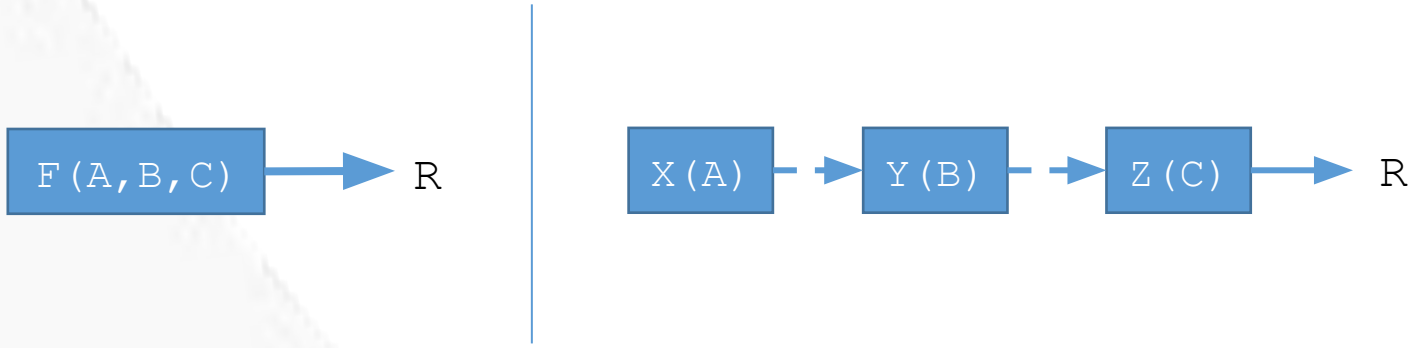
2.6 Curry

Prof. Bruno no Augusto Teixeira

Nesta aula

- Curry.

Currying



Conclusão

☑ Currying.

Próxima aula

- JavaScript Assíncrono.

JavaScript Avançado I

3. JavaScript Assíncrono

Prof. Bruno Augusto Teixeira

JavaScript Avançado I

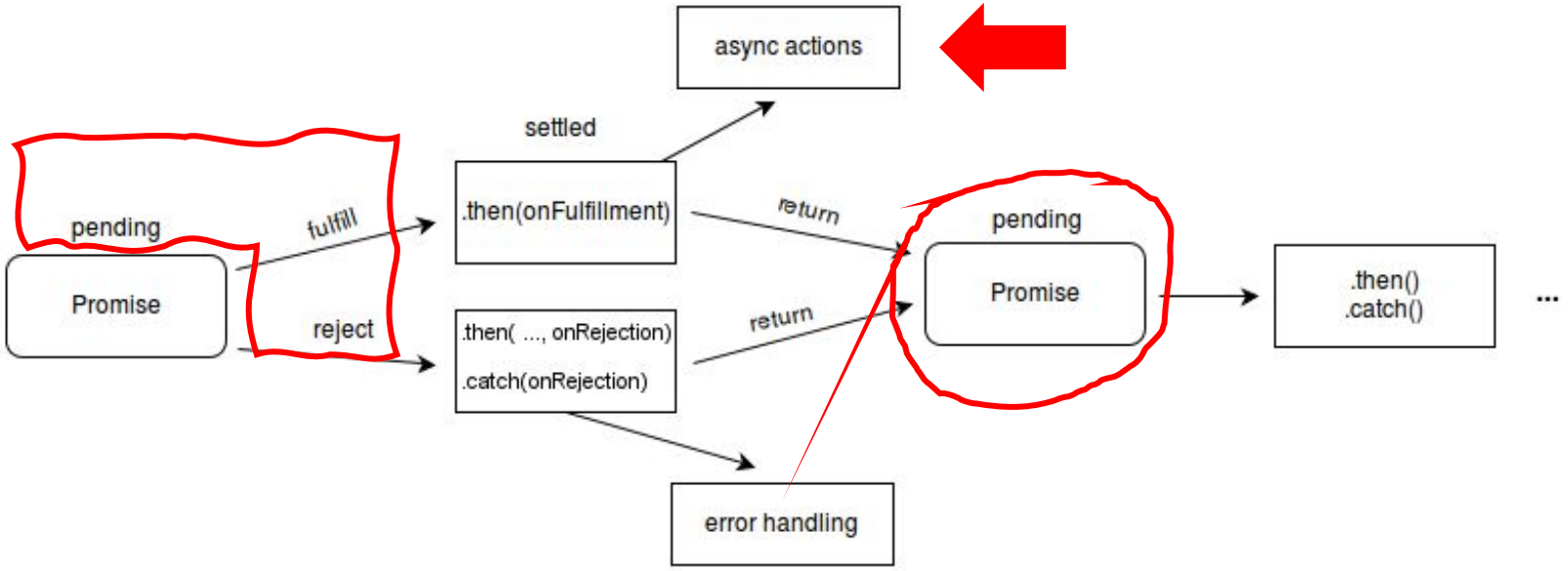
3.1 Promises

Prof. Bruno Augusto Teixeira

Nesta aula

- Promises.

Promises


async actions
settled
pending
fulfill
.then(onFulfillment)
return
pending
Promise
.then()
.catch()
...
Promise
reject
.then(..., onRejection)
.catch(onRejection)
return
error handling

Conclusão

- ☑ Promises.

Próxima aula

- Promises API.

JavaScript Avançado I

3.2 Promises API

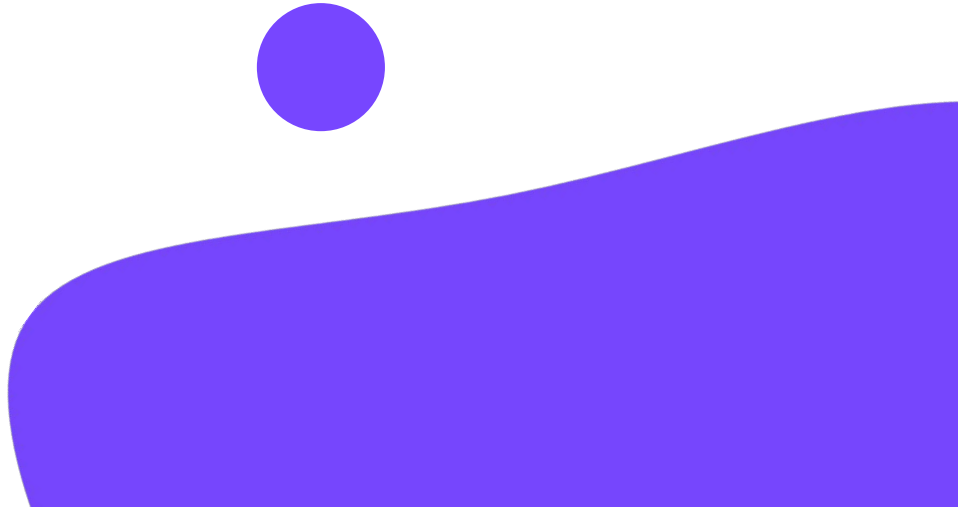
Prof. Bruno Augusto Teixeira

Nesta aula

- Promises API.

Promises API

- `Promise.resolve`
- `Promise.reject`
- `Promise.all`
- `Promise.allSettled`
- `Promise.race`
- `Promise.any`



Conclusão

☑ Promises API:

- `Promise.resolve`
- `Promise.reject`
- `Promise.all`
- `Promise.allSettled`
- `Promise.race`
- `Promise.any`

Próxima aula

- Event Loop.

JavaScript Avançado I

3.3 Event Loop

Prof. Bruno Augusto Teixeira

Nesta aula

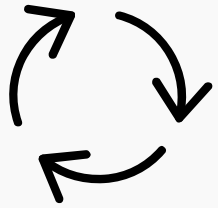
- Event Loop.

Event Loop

STACK

```
console.log('Início');

setTimeout(function getLog() {
    console.log('Aguarde');
}, 5000);

console.log('Fim');
```

BACKGROUND



TASK QUEUE

CONSOLE

Event Loop

STACK

console.log('Início');

```
setTimeout(function getLog() {
    console.log('Aguarde');
}, 5000);
```

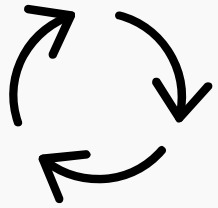
console.log('Fim');

console.log('Aguarde')

getLog()

BACKGROUND

timer getLog()



TASK QUEUE

getLog()

CONSOLE

Início
Fim
Aguarde

Event Loop

STACK

console.log('Início');

setTimeout(function getLog() {
    console.log('Aguarde');
}, 5000);

console.log('Fim');

Promise.resolve("Success")
.then((value) => console.log(value)
, (value) => {});

console.log('Aguarde')

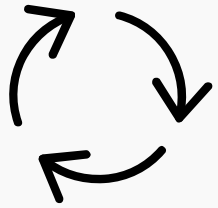
console.log('Success')

BACKGROUND

timer getLog()

TASK QUEUE

getLog()



MICROTASK QUEUE

Promise.resolve()

CONSOLE

Início
Fim
Success
Aguarde

Conclusão

- ☑ Event Loop.
- ☑ Microtask e Macrotask.

Próxima aula

- Iterators.

JavaScript Avançado I

3.4 Iterators

Prof. Bruno Augusto Teixeira

Nesta aula

- Iterators.

Iterators

```
interface Iterator {
  next() {
      //...
      return {
          value: <value>,
          done: <boolean>
      };
  };
}
```

Conclusão

- ☑ Iterators.

Próxima aula

- Generators.

JavaScript Avançado I

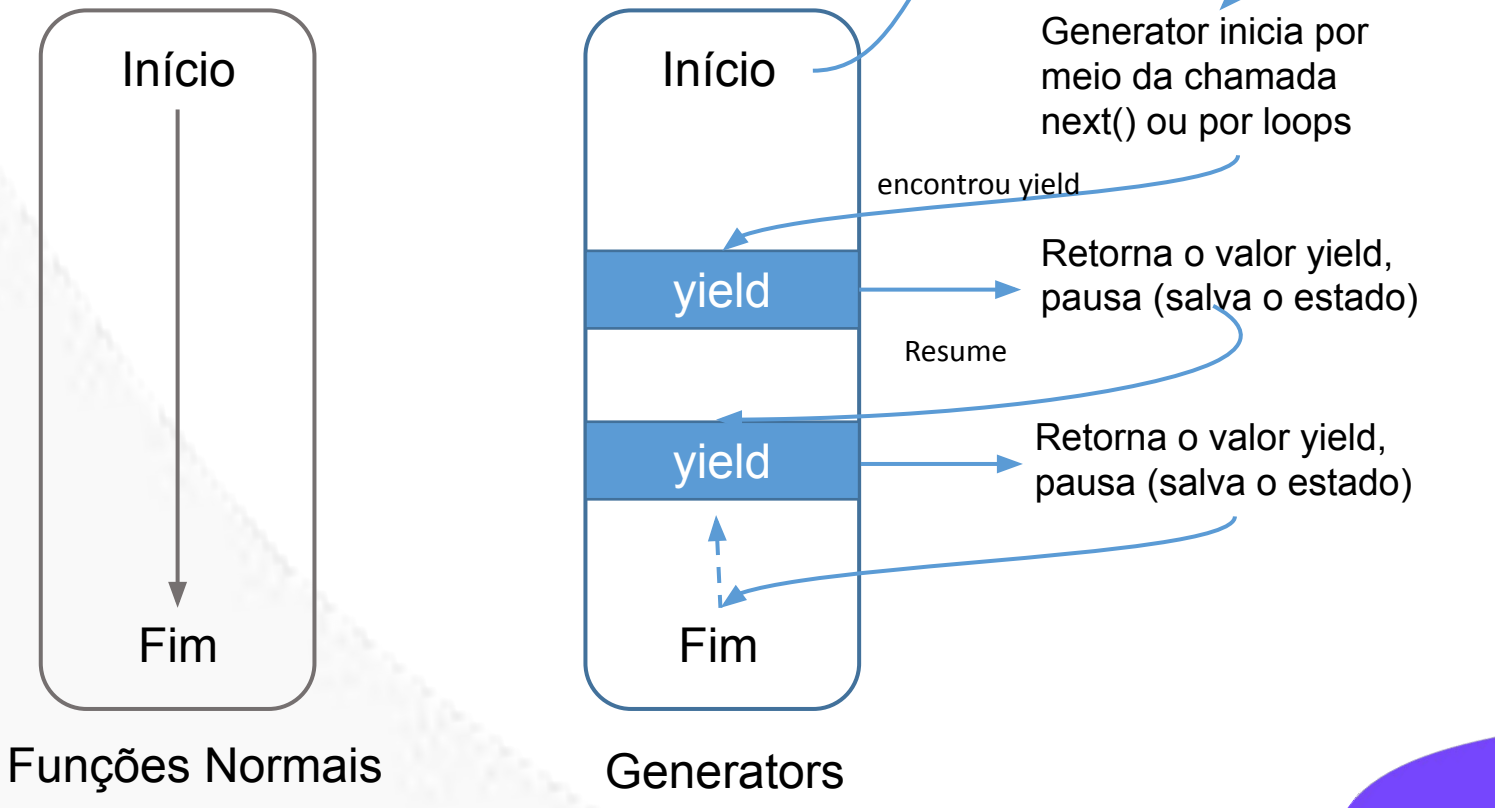
3.5 Generators

Prof. Bruno Augusto Teixeira

Nesta aula

- Generators.

Generators


Início
Fim
Funções Normais
Início
Retorna
Object Generator (Iterator)
Generator inicia por
meio da chamada
next() ou por loops
encontrou yield
yield
Retorna o valor yield,
pausa (salva o estado)
Resume
yield
Retorna o valor yield,
pausa (salva o estado)
Fim
Generators

Conclusão

- ☑ Generators.

Próxima aula

- ECMAScript.

JavaScript Avançado I

4. ECMAScript

Prof. Bruno Augusto Teixeira

JavaScript Avançado I

4.1 Contexto Histórico

Prof. Bruno Augusto Teixeira

Nesta aula

- Contexto Histórico.

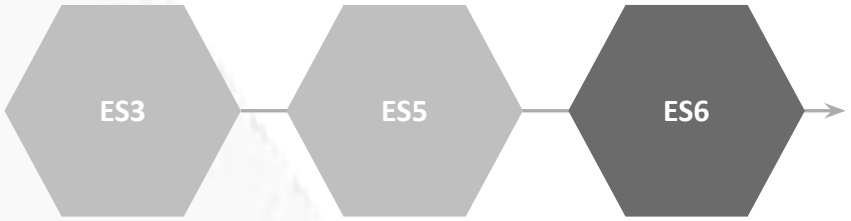
ECMAScript

ES3

ECMAScript

- Strict mode

ECMAScript



- Strict mode
- Classes
- Let + const

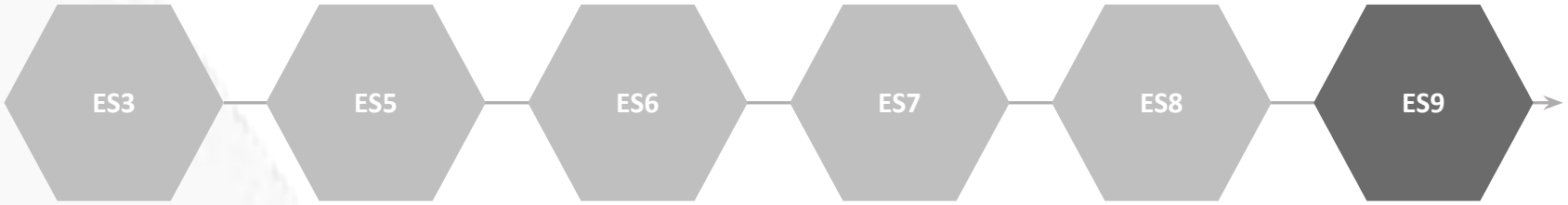
ECMAScript

- Strict mode
- Classes
- Let + const
- Operador de exponenciação
- Isolamento de código

ECMAScript

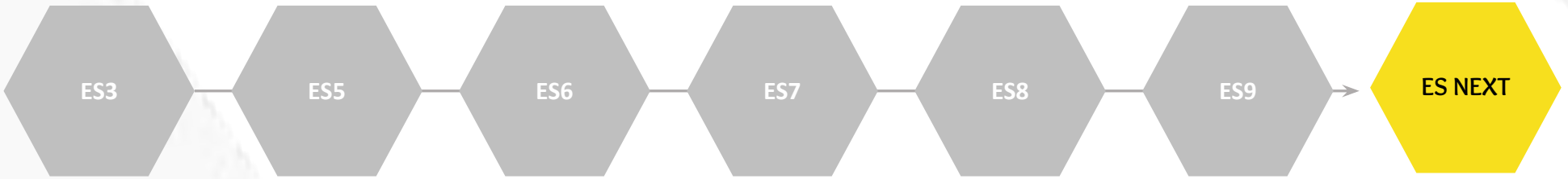
- Strict mode
- Classes
- Let + const
- Operador de exponenciação
- Isolamento de código
- Tuplas
- Traits
- String Padding

ECMAScript



- Strict mode
- Classes
- Let + const
- Operador de exponenciação
- Isolamento de código
- Tuplas
- Traits
- String Padding
- Iteração assincona
- Propriedades Rest\Spread

ECMAScript



- Strict mode
- Classes
- Let + const
- Operador de exponenciação
- Isolamento de código
- Tuplas
- Traits
- String Padding
- Iteração assincona
- Propriedades Rest\Spread

Conclusão

☑ Contexto Histórico.

Próxima aula

- ECMAScript 2015.

JavaScript Avançado I

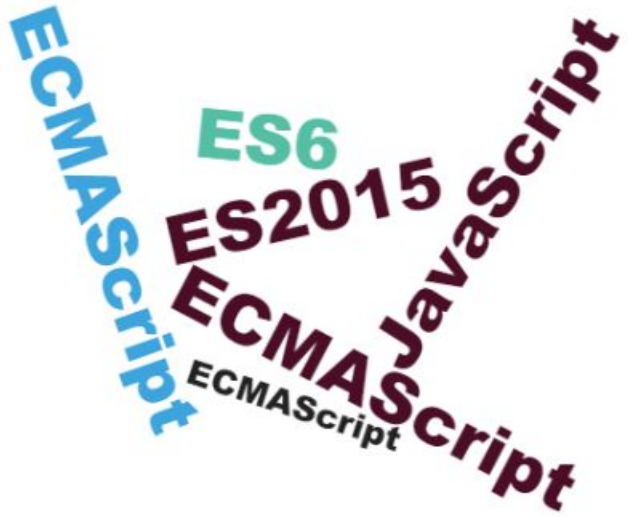
4.2 ECMAScript 2015

Prof. Bruno Augusto Teixeira

Nesta aula

- ECMAScript 2015.

ECMAScript 2015



ES6

- Let + Const.
- Arrow functions.
- Classes.
- Template Strings.
- Destructing.
- Default + rest + spread.

Conclusão

- ☑ ES6.

Próxima aula

- ES7.

JavaScript Avançado I

4.3 ES7

Prof. Bruno Augusto Teixeira

Nesta aula

- ES7.

ES7

- Operador de exponenciação.
- Array.prototypes.includes

Conclusão

- ☑ ES7.

Próxima aula

- ES8.

JavaScript Avançado I

4.4 ES8

Prof. Bruno Augusto Teixeira

Nesta aula

- ES8.

ES8

- String padding
- Trailing commas
- async + await

Conclusão

- ☑ ES8.

Próxima aula

- ES9.

JavaScript Avançado I

4.5 ES9

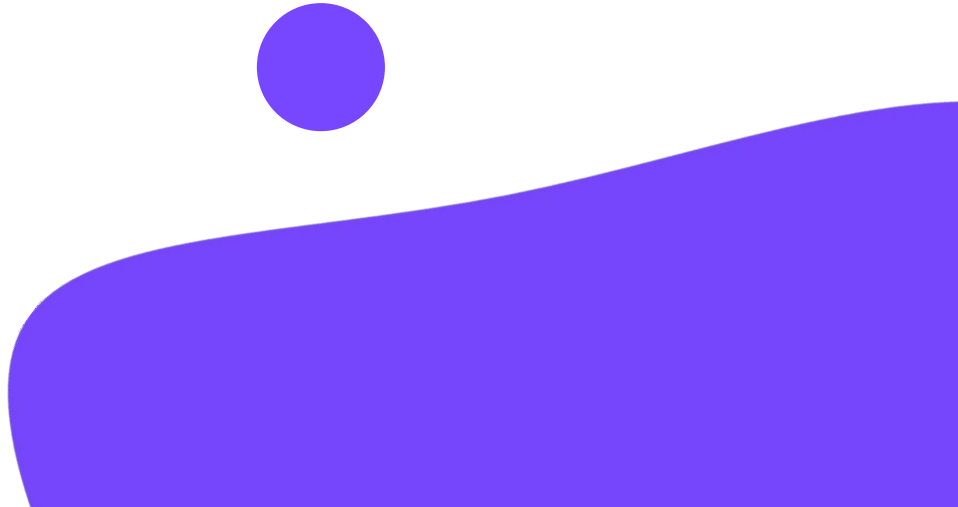
Prof. Bruno Augusto Teixeira

Nesta aula

- ES9.

ES9

- Promises.prototype.finally()
- Iteração assíncrona.
- Propriedades Rest + Spread



Conclusão

- ☑ ES9.

Próxima aula

- ES10.

JavaScript Avançado I

4.5 ES10

Prof. Bruno Augusto Teixeira

Nesta aula

- ES10.

ES10

- Array.Flat()
- String.trimStart()
- String.trimEnd()
- Array.sort()

Conclusão

☑ ES10.

Próxima aula

- ES11.

JavaScript Avançado I

4.6 ES11

Prof. Bruno Augusto Teixeira

Nesta aula

- ES11.

ES11

- BigInt
- Private Methods
- Nullish coalescing Operator
- globalThis
- Promise.allSettled
- Optional Chaining
- Dynamic import

Conclusão

☑ ES11.

Próxima aula

- Bibliotecas.