

# Aula 4: This

## Funções

# Objetivos

1. Apresentar a palavra “this” e seu uso
2. Como aplicar métodos para manipular seu valor

Aula 2 | Etapa 1:

# O que é

## Funções

# This: o que é?

```
const pessoa = {
  firstName: "André",
  lastName : "Soares",
  id       : 1,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  },
  getId: function() {
    return this.id;
  }
};

pessoa.fullName();
// "André Soares"

pessoa.getId();
// 1
```

A palavra reservada **this** é uma referência de contexto.

No exemplo, this refere-se ao objeto pessoa.

# This: o que é?

Seu valor pode mudar de acordo com o lugar no código onde foi chamada.

Contexto	Referência
Em um objeto (método)	Próprio objeto dono do método
Sozinha	Objeto global (em navegadores, window)
Função	Objeto global
Evento	Elemento que recebeu o evento

# This: o que é?

## Fora de função

```
JS playground.js •
JS playground.js
1 console.log(this);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)
$ node playground.js
{}
```

## No navegador

```
> this
< Window {0: global, window: Window, self: Window, document:
  document, name: "", location: Location, ...}
>
```

## Dentro de uma função

```
JS playground.js M x
JS playground.js
1 (function () {
2   console.log(this);
3 })();
4

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)
$ node playground.js
<ref *1> Object [global] {
  global: [Circular *1],
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  },
  queueMicrotask: [Function: queueMicrotask],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  }
}
```

# This: o que é?

## Dentro de um objeto (método)

```
1  const pessoa = {
2    firstName: 'Diego',
3    lastName: 'Vieira',
4    getFullName: function () {
5      console.log(`${this.firstName} ${this.lastName}`);
6    },
7  };
8
9  pessoa.getFullName();
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)
$ node playground.js
Diego Vieira

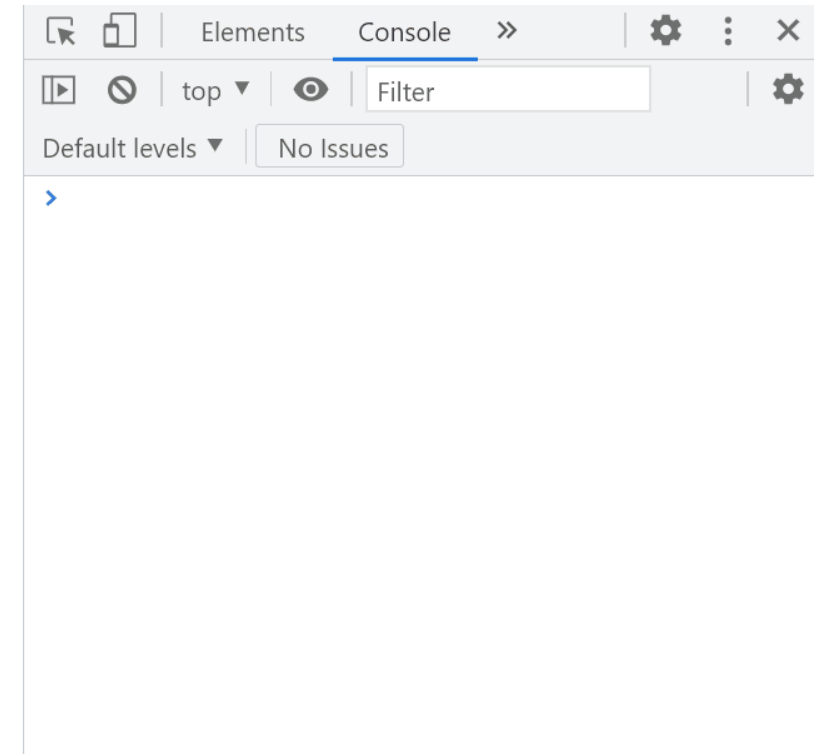
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)
$
```

# This: o que é?

## Em um evento no HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <button
      id="my-btn"
      onclick="console.log(this)"
    >click me!</button>
  </body>
</html>
```

click me!





Aula 4 | Etapa 2:

# Manipulando seu valor

## Funções



# Call

```
const pessoa = {  
  nome: 'Miguel',  
};  
  
const animal = {  
  nome: 'Murphy',  
};  
  
function getSomething() {  
  console.log(this.nome);  
}  
  
getSomething.call(pessoa);
```

```
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)  
$ node playground.js  
Miguel
```



# Call

```
const pessoa = {  
  nome: 'Miguel',  
};  
  
const animal = {  
  nome: 'Murphy',  
};  
  
function getSomething() {  
  console.log(this.nome);  
}  
  
getSomething.call(animal);
```

```
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)  
$ node playground.js  
Murphy
```

# Call

```
const myObj = {  
  num1: 2,  
  num2: 4,  
};  
  
function soma(a, b) {  
  console.log(this.num1 + this.num2 + a + b);  
}  
  
soma.call(myObj, 1, 5);  
// 12
```

É possível passar  
parâmetros para essa  
função separando-os  
por vírgulas.



# Apply

```
const pessoa = {  
  nome: 'Miguel',  
};  
  
const animal = {  
  nome: 'Godi',  
};  
  
function getSomething() {  
  console.log(this.nome);  
}  
  
getSomething.apply(pessoa);
```

```
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)  
$ node playground.js  
Miguel
```



# Apply

```
const pessoa = {  
  nome: 'Miguel',  
};  
  
const animal = {  
  nome: 'Godi',  
};  
  
function getSomething() {  
  console.log(this.nome);  
}  
  
getSomething.apply(animal);
```

```
Stephany@DESKTOP-KTK90HF MINGW64 /c/DIO (main)  
$ node playground.js  
Godi
```

# Apply

```
const myObj = {  
  num1: 2,  
  num2: 4,  
};  
  
function soma(a, b) {  
  console.log(this.num1 + this.num2 + a + b);  
}  
  
soma.apply(myObj, [1, 5]);  
// 12
```

É possível passar  
parâmetros para essa  
função dentro de um  
array.

# Bind

Clona a estrutura da função onde é chamada e aplica o valor do objeto passado como parâmetro.

```
const retornaNomes = function () {  
    return this.nome;  
};  
  
let bruno = retornaNomes.bind({ nome: 'Bruno' });  
  
bruno();  
// Bruno
```





DIGITAL  
INNOVATION  
ONE

# Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)