


Prof. Marcos Nava

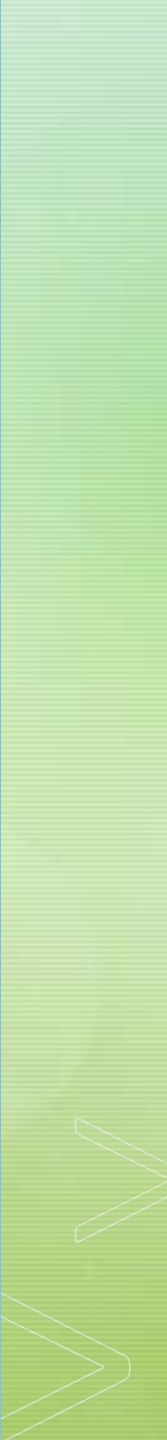


# Programação de Scripts





# Agenda da Aula

- O que são as funções para o Javascript
  - Funções
  - Hoisting
  - Loops
- 

# O que são funções para o JavaScript?

- Para o JavaScript as funções são cidadãos de primeira classe.
- Elas possuem uma importância muito maior do que outras linguagens.
- Você pode criar funções como em qualquer outra linguagem.
- Você pode também:
  - Armazenar funções em outras variáveis
  - Armazenar em vetores e objetos
  - Passar funções como parâmetros
  - Uma função pode retornar uma função
  - Você pode ter funções anônimas e arrow functions

# Funções

Como qualquer outra linguagem você pode criar funções:

```
function fun1()  
{  
    console.log('Eu sou uma função!');  
}  
fun1();
```

Saída:  
Eu sou uma função!

# Funções

Você pode armazenar funções em variáveis, vetores ou objetos:

```
const fun2 = function () { }
```

```
const array = [function (a,b) { return a + b}, fun1, fun2]  
console.log(array[0](2, 3));
```

```
const obj = { }  
obj.falar = function() { return 'Tudo bem?'}  
console.log(obj.falar())
```

Saída:

5  
Tudo bem?

# Funções

Você pode armazenar funções em variáveis, vetores ou objetos:

```
function run(fun)
{
    fun();
}

run(function() { console.log('Executando...')})
```

Saída:  
Executando...

# Funções

Uma função pode até mesmo retornar uma função:

```
function soma(a,b)
{
    return function(c) {
        console.log(a + b + c);
    }
}

soma(2, 3)(4)
const cincoMais = soma(2,3);
cincoMais(4);
```

Saída:

9  
9

# Funções

Você deve passar os parâmetros conforme a necessidade:

```
function imprimirSoma(a, b)
{
    console.log(a + b);
}

imprimirSoma(2, 3);
imprimirSoma(2);
imprimirSoma(2, 10, 4, 2, 5);
imprimirSoma();
```

Saída:

5  
NaN  
12  
NaN



# Funções

Você pode criar valores padrão e retornar resultados:

```
function soma(a, b = 1)
{
    return a + b;
}
console.log(soma(2, 3));
console.log(soma(2));
console.log(soma());
```

Saída:

5  
3  
NaN

# Funções

Você pode criar arrow functions:

```
const imprimirSoma = function (a, b)
{
    console.log(a + b);
}
imprimirSoma(2, 3);
const soma = (a, b) => {
    return a + b;
}
console.log(soma(2, 3));
```

Saída:

5  
5

# Funções

Quando uma arrow function simplesmente retorna um valor você pode deixar implícito:

```
const subtracao = (a, b) => a - b;  
console.log(subtracao(2,3));
```

Saída:

-1

# Funções

Veja alguns exemplos de arrow function conforme o número de parâmetros

```
const exe1 = () => false;
```

```
const exe2 = umso => false;
```

```
const exe3 = (um,dois) => false;
```

# loops

Geralmente você tem os mesmos loops da linguagem java ou C:

- while
- for

A algumas particularidades são a criação de variáveis dentro deles.

Lembre-se que temos o let e o var para a declaração de variáveis.

# Loops com var

Vejamos primeiro exemplo com var:

```
for(var i = 0; i < 10; i++)  
{  
    console.log(i)  
}  
console.log('i =', i)
```

Saída:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
i = 10
```

# Loops com var

O que acontece aqui?

```
const funcs = [];  
for(var i = 0; i < 10; i++)  
{  
    funcs.push(function () {  
        console.log(i)  
    });  
}  
funcs[2]();  
funcs[8]();
```

Saída:

10  
10

# Loops com let

O mesmo primeiro exemplo com var agora usando let:

```
for(let i = 0; i < 10; i++)  
{  
    console.log(i)  
}  
console.log('i =', i)
```

Saída:

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

Erro!!!!!!



# Loops com let

O que acontece aqui, se agora usamos o let?

```
const funcs = [];  
for(let i = 0; i < 10; i++)  
{  
  funcs.push(function () {  
    console.log(i)  
  });  
}  
funcs[2]();  
funcs[8]();
```

Saída:

2  
8

# Hoisting

A criação de variáveis em JavaScript podem sofrer um efeito chamado hoisting (elevação):

```
console.log('a = ', a);  
var a = 2;  
console.log('a = ', a);
```

Saída:

```
a = undefined  
a = 2
```

# Hoisting

Então, quando usamos o var temos a elevação da criação da variável, não seu valor. Mas com let isso não ocorre.

```
console.log('a = ', a);  
let a = 2;  
console.log('a = ', a);
```

Erro!!!

Saída: