

Modelo de documento markdown ↩️

1. INDEX

1. Introdução

1. [Objetivo da função seta](#)
2. [Pre-requisitos.](#)
2. [Sintaxe da function seta \(arrow function\)](#)
3. [Diferença entre arrow function \(=>\) e function\(\) em javascript](#)
4. [Referências](#)
5. [Histórico](#)

2. CONTEÚDO

1. Introdução


1. Objetivo

1. Objetivo da função seta:

1. A arrow function tem por finalidade tornar a sintaxe mais fácil de ser lida e expressa, é útil quando usado como função anônima para interagir com coleções, porque ela usa o **this** da coleção e não seu próprio **this**.
2. Uma expressão arrow function possui uma sintaxe mais curta quando comparada a uma expressão de função (function expression) e não tem seu próprio **this**, **arguments**, **super** ou **new.target**. Estas expressões de funções são melhor aplicadas para funções que **não sejam métodos**, e elas não podem ser usadas como construtoras (**constructors**)

2. [BACK]

2. Pre-requisitos:

1. Conhecimento da [linguagem html](#).
2. Conhecimento de [funções javascript](#).
3.  [BACK]

2. [Sintaxe da função seta \(arrow function\)](#):

1. Sintaxe básica:

1. Um parâmetro com a expressão simples, o retorno não é necessário:

1. param => expression

1. param

1. O nome de um argumento.
2. Nenhum argumento precisa ser indicado com ().
3. Para apenas um argumento, os parênteses não são necessários.
(como `foo => 1`)

2. **statements** or expression

1. Várias instruções precisam ser colocadas entre colchetes.
2. Uma única expressão não requer colchetes.
3. A expressão também é o valor de retorno implícito da função.

2. Vários parâmetros requerem parênteses. Com a expressão simples, o retorno não é necessário:

1. `([param[, param]]) => { statements }`
2. `(param1, paramN) => expression`

3. As instruções multilinhas requerem colchetes e retorno:

1. Código javascript:

```
param => { let a = 1;
          return a + param;
        }
```

4. Vários parâmetros requerem parênteses. As instruções multilinhas requerem colchetes e retorno:

1. Código javascript:

```
(param1, paramN) => { let a = 1;
                     return a + param1 + paramN;
                   }
```

2. Sintaxe avançada:

1. Para retornar uma expressão literal de objeto, são necessários parênteses em torno da expressão:

1. Código javascript:

```
params => ({foo: "a"}) // returning the object {foo:
```

```
"a"}
```

2. [Rest parameters](#) are supported:

1. Código javascript:

```
(a, b, ...r) => expression
```

3. [Default parameters](#) are supported:

1. Código javascript:

```
(a=400, b=20, c) => expression
```

4. [Destructuring](#) within params supported:

1. Código javascript:

```
([a, b] = [10, 20]) => a + b; // result is 30  
({ a, b } = { a: 10, b: 20 }) => a + b; // result is  
30
```

5. [Exemplo javascript](#) para contar o tamanho de cada string de um array:

1. Código javascript:

```
const materials =  
['Hydrogen', 'Helium', 'Lithium', 'Beryllium'];  
  
/**  
 * Array.prototype.map()  
 * Cria uma nova matriz com os resultados da chamada  
de uma  
 * função fornecida em cada elemento desta matriz.  
 * EXEMPLO DE USO:  
 */  
result = materials.map(material => material.length);
```

```
console.log(result);  
// expected output: Array [8, 6, 7, 9]
```

6. A lista de parâmetros para uma função sem parâmetros deve ser escrita com um par de parênteses.

1. Código javascript:

```
() => { statements }
```

3. Diferenças e limitações das arrow functions :

1. Não tem vínculos próprios com **this** ou **super** não deve ser usado como **methods**.
2. Não tem **arguments**, nem **new.target** palavras-chave.
3. Não é adequado para **call**, **apply** e **bind** métodos, que geralmente dependem de estabelecer um **scope**.
4. Não pode ser usado como **constructors**.
5. Não pode usar **yield**, dentro de seu corpo.

4.  **BACK**

3. Diferença entre arrow function (=>) e function() em javascript

1. Arrow-Function incluída na es6 não é apenas uma **sugar syntax** de function, as duas sintaxes tem suas peculiaridades. Abaixo citarei algumas delas e quais as vantagens em usar as Arrow functions.

1. O que é arrow function?

1. Arrow function é uma função no javascript, porem ela traz algumas diferenças quando comparada com funções normais, vejamos abaixo:

1. Lexical this

1. Ele captura o valor de **this** do contexto vinculado, ou seja do escopo em que ele se encontra, por exemplo:

1. Código JavaScript

```
1  function Mensagem() {  
2  
3  this.mensagem = 'Passou';  
4  
5  // Traditional function  
6  this.logado = function() {  
7      setTimeout(function()
```

```

    { //setTimeout=Executa um bloco específico uma
      vez depois de um determinado tempo
8      console.log(this.mensagem);
9      }, 30);
10   };
11   }
12
13   var msg = new Mensagem();
14   msg.logado(); // undefined

```

2. A função *Mensagem* ao ser instanciada, retornara um objeto com 2 propriedades, *mensagem* e *logado*. Que não logou a *mensagem* desejada porque o **this** na linha 8 trata-se da função anônima *logado*. Para funcionar como esperado, basta trocar a função na linha 7 por uma **arrow function**.

1. Código JavaScript

```

1   function Mensagem() {
2
3   this.mensagem = 'Passou';
4
5   // Traditional function
6   this.logado = function() {
7       setTimeout(() => {
8           console.log(this.mensagem);
9           }, 30);
10  };
11  }
12
13  var msg = new Mensagem();
14  msg.logado(); // "Passou"

```


3. Neste caso funciona como esperado porque o **this** dentro da arrow function é uma referência do **this** da função acima no caso a função *Mensagem* assim o **this** dentro da função *logado* será o mesmo que o da função *Mensagem*.

2. .

2. ...


3. 

4. REFERÊNCIAS


1. [Arrow functions descrição](#)
2. [Arrow functions examples](#)
3. [Arrow functions syntax](#)
4. [Arrow functions sintaxe avançada](#)
5. [Arrow functions usada em coleções](#)
6. [Arrow functions vs Functions](#)
7. [Arrow functions - https://developer.mozilla.org/](https://developer.mozilla.org/)
8. [Arrow function expressions](#)
9. [comparing traditional functions to arrow functions](#)
10. [Syntactic Sugar](#)
11. [ BACK]

5. HISTÓRICO

1. 15/02/2021

- ☒ Criado a versão 0.0.1 deste documento;
- ☒ Escrever objetivo de arrow function
- ☒ Escrever a diferença entre arrow function e funções normais.
- ☒ Escrever o tópico sintaxe de arrow function;
- ☒ Conferir os links do documento
- ☒ Inserir os botões de retorno para o topo.
- [ BACK]

2. 16/02/2021

- ☒ Fazer revisão do texto do documento para saber se os textos estão claros.
- [ BACK]

