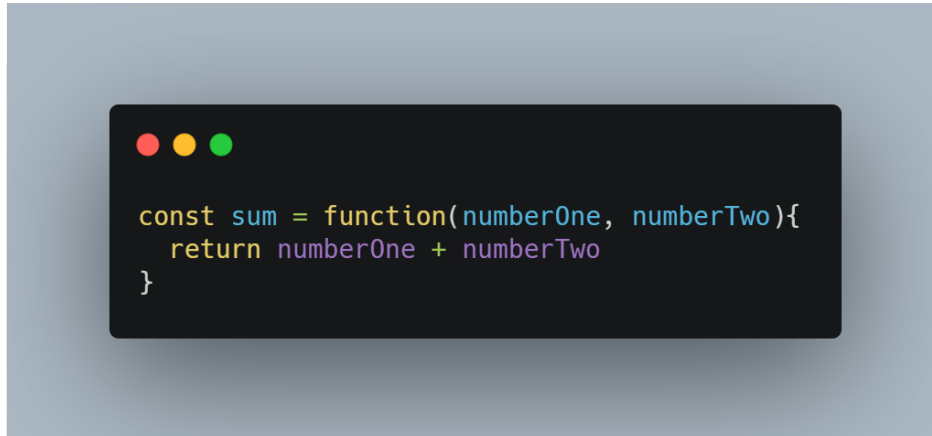


Mais sobre funções

Função como valor em constante

Neste exemplo vemos o conceito de função como um valor armazenado em uma constante e também chamamos esse tipo de função de “**função anônima**”, pois ela não recebe um nome na declaração.

[EXEMPLO]

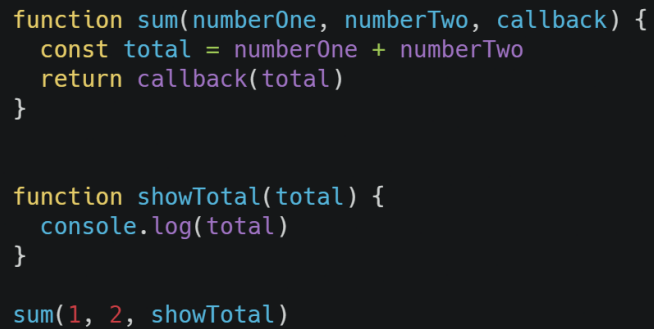


Referências

- <https://www.javascripttutorial.net/javascript-anonymous-functions/>
- https://www.w3schools.com/js/js_function_definition.asp

Callbacks e função como argumento

[EXEMPLO]



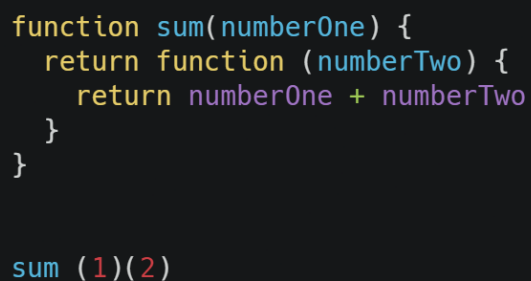
```
function sum(numberOne, numberTwo, callback) {  
  const total = numberOne + numberTwo  
  return callback(total)  
}  
  
function showTotal(total) {  
  console.log(total)  
}  
  
sum(1, 2, showTotal)
```

Referência

- https://developer.mozilla.org/en-US/docs/Glossary/Callback_function
- https://www.w3schools.com/js/js_callback.asp

Funções como retorno de funções

[EXEMPLO]



```
function sum(numberOne) {  
  return function (numberTwo) {  
    return numberOne + numberTwo  
  }  
}  
  
sum (1)(2)
```

Funções em arrays e objetos

[EXEMPLO 1]

```
const operations = {  
  sum: function (numberOne, numberTwo) {  
    return numberOne + numberTwo  
  },  
  subtract: function (numberOne, numberTwo) {  
    return numberOne - numberTwo  
  }  
}  
  
operations.sum(1, 2)  
operations.subtract(2, 1)
```

[EXEMPLO2]

```
const sum = function (numberOne, numberTwo) {  
  return numberOne + numberTwo  
}  
  
const subtract = function (numberOne, numberTwo) {  
  return numberOne - numberTwo  
}  
  
const operations = [sum, subtract]  
  
operations[0](1, 2)  
operations[1](2, 1)
```

Referências

- [Funções - JavaScript | MDN \(mozilla.org\)](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions)
- [Funções – JavaScript | Dev Content](#) (Bem explicativo, muito completo, tem um trecho sobre rest operator, fala sobre escopo tbm)

Arrow Function

```
// estrutura base de uma arrow function
const sum = (numberOne, numberTwo) => {
  return numberOne + numberTwo
}

// retorno implícito. Sem a necessidade do uso de "return"
const sum2 = (numberOne, numberTwo) => numberOne + numberTwo

// sem parenteses quando há apenas 1 só argumento
const print = whatever => console.log(whatever)
```

Referências

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>
- https://www.w3schools.com/js/js_arrow_function.asp

Checkpoint de Exercício

1. Crie uma função que recebe duas notas como argumento e retorna a média entre elas. Utilize arrow function e atribua a uma constante.
-

IF Ternário

- Outra forma de realizar if com bloco else
- Nos blocos do ternário devemos colocar expressões
- [EXEMPLO] Expressão com ternário

```
let x = 10
console.log(x < 10 ? 'é menor que 10' : 'é maior que 10')
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

Destructuring Assignment (Desestruturação)

- Funcionalidade do ES6+ que permite escrevermos códigos mais legíveis e robustos, fazendo mais em menos linhas
- Realiza uma atribuição de forma “resumida”
- Utilizado em objetos, arrays e parâmetros de funções

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Objeto

- [EXEMPLO] Sem destructuring assignment

```
const user = {  
  name: 'Érico',  
  idade: 29,  
}  
  
const name = user.name  
const idade = user.idade  
console.log(name, idade)
```

- [EXEMPLO] Com destructuring assignment

```
const { name, idade } = user  
console.log(name, idade)
```

- [EXEMPLO] Com destructuring assignment renomeando a variável

```
const { name: nome } = user  
console.log(nome)
```

Array

- [EXEMPLO] Sem destructuring assignment

```
const users = ['João', 'Juca']  
const primeiro = users[0]  
const segundo = users[1]  
console.log(primeiro, segundo)
```

- [EXEMPLO] Com destructuring assignment

```
const users = ['João', 'Juca']  
const [primeiro, segundo] = users  
console.log(primeiro, segundo)
```

- [EXEMPLO] Com destructuring assignment pulando elementos

```
const users = ['João', 'Juca']  
const [, segundo] = users
```

- `console.log(segundo)`

Parâmetros

- [EXEMPLO] Array sem DA

```
• const users = ['João', 'Juca']
• function getFirst(array) {
•   return array[0]
• }
• console.log(getFirst(users))
```

- [EXEMPLO] Array com DA

```
• const users = ['Érico', 'Juca']
• function getFirst([first]) {
•   return first
• }
• console.log(getFirst(users))
```

- [EXEMPLO] Objeto sem DA

```
• const user = {
•   name: 'João',
•   idade: 29,
• }
• function printName(user) {
•   console.log(user.name)
• }
• printName(user)
```

- [EXEMPLO] Objeto com DA

```
• const user = {
•   name: 'João',
•   idade: 27,
• }
• function printName({ name }) {
•   console.log(name)
• }
• printName(user)
•
```

- [EXEMPLO] Misturado sem DA

```
• const users = [
•   { name: 'João', role: 'Professor' },
•   { name: 'Juca', role: 'Estudante' },
•   { name: 'Márcia', role: 'Estudante' },
• ]
```

```

• { name: 'Pedro', role: 'Estudante' },
• { name: 'Matheus', role: 'Professor' },
• { name: 'Júlia', role: 'Estudante' },
• ]
• function getFirstName(array) {
•   return array[0].name
• }
• console.log(getFirstName(users))

```

• [EXEMPLO] Misturado com DA

```

• const users = [
•   { name: 'Érico', role: 'Professor' },
•   { name: 'Juca', role: 'Estudante' },
•   { name: 'Márcia', role: 'Estudante' },
•   { name: 'Pedro', role: 'Estudante' },
•   { name: 'Matheus', role: 'Professor' },
•   { name: 'Júlia', role: 'Estudante' },
• ]
• function getFirstName([{ name }]) {
•   return name
• }
• console.log(getFirstName(users))

```

Exercícios

1. Crie uma função que recebe uma string (com quatro possibilidades: “soma”, “subtrai”, “divide”, “multiplica”) e dois números. A função deve retornar a operação realizada informada pela string nos dois números.

Exemplo Entrada:

myFunction('soma')(3)(5)

Exemplo Saída:

8

2. Crie uma função que recebe um objeto usuário e imprime se ele é maior de idade, utilize os conhecimentos da aula para melhorar o código.

Exemplo Entrada:

{ name: 'Juca', idade: 28 }

Exemplo Saída:

Maior de idade