

Curso de JavaScript

Fundamentos



Rafael Mesquita, Prof.

#12

Funções

Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Funções

Definição de Função

Uma **função** (também conhecida como método, rotina ou sub-rotina) é um bloco de código que executa uma tarefa ou um grupo de tarefas.

Vimos muitas funções embutidas como `console.log`, `prompt`, etc. Além das funções dos arrays: `push`, `pop`, `sort`, etc.

Uma definição de função é o **bloco de código reutilizável** sobre o qual falamos anteriormente e que executa tarefas.

Funções

Definição de Função

A sintaxe geral é:

```
function <nome da função> ( <parametros> ){  
    # Corpo da função  
    #Declarações...  
    return <valor>  
}
```

Nome da função: identifica uma função.

Parâmetros: também conhecidos como argumentos, são uma lista de variáveis que uma função aceita.

Corpo da função: é a parte central da definição onde o trabalho computacional real desaparece.

Return: a instrução de retorno opcional é usada para retornar valores para a chamada de função.

Funções

Função VS Procedimento

Guia JavaScript da Mozilla



“Uma função sempre retorna um valor, mas um procedimento pode ou não retornar um valor.”

Livro: JavaScript - O guia definitivo



“As funções sem valor de retorno são chamadas de procedimentos.”

Funções

Definição de Função

Exemplos de definições de funções:

exemplo_01.js × +

```
1 ▼ function mostrar_mensagem() {  
2   console.log("Mensagem dentro da função")  
3 }
```

exemplo_02.js × +

```
1 ▼ function calcular_produto(x, y, z) {  
2   console.log("Produto:", x*y*z)  
3 }
```

exemplo_03.js × +

```
1 ▼ function calcular_quociente(a, b) {  
2   return a / b  
3 }
```

Exemplos de chamadas de funções:

Funções

Chamada de Função

```
funcoes_chamadas.js × +  
  
1 ▼ function mostrar_mensagem() {  
2   | console.log("Mensagem dentro da função")  
3   | }  
4  
5 ▼ function calcular_produto(x, y, z) {  
6   | console.log("Produto:", x * y * z)  
7   | }  
8  
9 ▼ function calcular_quociente(a, b) {  
10  | return a / b  
11  | }  
12  
13 // Chamada da função mostrar_mensagem  
14 mostrar_mensagem()  
15  
16 // Chamada da função calcular_produto, passando 3 argumentos  
17 calcular_produto(4, 7.5, 2.3)  
18  
19 // Chamada da função calcular_quociente, recebendo o retorno de valor  
20 q = calcular_quociente(11, 5)
```

Console × Shell × +

Mensagem dentro da função
Produto: 69

Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Funções

Argumentos *default*

Os **argumentos padrões** são usados para definir valores padrão para parâmetros de função. Quando um parâmetro não é passado ao fazer uma chamada de função, o valor padrão será considerado e quando for passado, o valor padrão será substituído pelo valor passado. A sintaxe geral de definição de uma função com variáveis padrão:

```
function <nome da função> ( <parametro> = valor padrão){  
    // Corpo da função  
}
```

Funções

Argumentos *default*

```
funcoes_argumentos_default.js × +
1 // Argumentos padrões (default)
2 // Dois parâmetros obrigatórios, 1 variável padrão
3 ▼ function calcular_area(x, y, z = 1) {
4     area = x * y * z
5     return area
6 }
7
8 raio = 2
9 comprimento = 6
10 largura = 9
11
12 // Calcula área do círculo como raio x raio x pi
13 // Substitui o valor padrão de z (1) por pi = 3.14
14 area_circulo = calcular_area(raio, raio, 3.14)
15 console.log("Círculo", "\nRaio:", raio, "\nÁrea:", area_circulo)
16
17 // Calcula área do retângulo
18 // Passando comprimento e largura, apenas 2 argumentos
19 area_retangulo = calcular_area(comprimento, largura)
20 console.log("\n\nRetângulo", "\nComprimento:", comprimento,
    "\nLargura:", largura, "\nÁrea: ", area_retangulo)
```

Console × Shell × +

```
Círculo
Raio: 2
Área: 12.56

Retângulo
Comprimento: 6
Largura: 9
Área: 54
```

Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Funções

Escopo de Variáveis

Quando você declara uma variável fora de qualquer função, ela é chamada de **variável global**, porque está disponível para qualquer outro código no documento atual.

Quando você declara uma variável dentro de uma função, é chamada de **variável local**, pois ela está disponível somente dentro dessa função.

Observação: **let** permite que você declare variáveis limitando seu escopo no bloco.

Exemplo com variáveis globais

Funções

Escopo de Variáveis

New tab × funcoes_escopos_1.js × + ≡ ⋮

```
1 var a = 1 // variável global
2
3 ▼ function f1() {
4     b = 2 // variável global
5     console.log("f1() - a:", a)
6     console.log("f1() - b:", b)
7 }
8
9 ▼ function f2() {
10     console.log("f2() - a:", a)
11     console.log("f2() - b:", b)
12 }
13
14 f1()
15 f2()
16
17 console.log("a:", a)
18 console.log("b:", b)
```

Console × Shell × ⋮

```
f1() - a: 1
f1() - b: 2
f2() - a: 1
f2() - b: 2
a: 1
b: 2
```

Exemplo com variáveis: global e local

Funções

Escopo de Variáveis

New tab × funcoes_escopos_2.js × +

```
1 var a = 1 // variável global
2
3 ▼ function f1() {
4   var b = 2 // variável local
5   console.log("f1() - a:", a)
6   console.log("f1() - b:", b)
7 }
8
9 ▼ function f2() {
10  console.log("f2() - a:", a)
11  console.log("f2() - b:", b) // Erro
12 }
13
14 f1()
15 f2()
16
17 console.log("a:", a)
18 console.log("b:", b) // Erro
```

Console ×

Shell ×

```
f1() - a: 1
f1() - b: 2
f2() - a: 1
ReferenceError: b
```

Exemplo com variáveis: global, local e de bloco

Funções

Escopo de Variáveis

```
New tab × funcoes_escopos_3.js × +  
1 var a = 1 // variável global  
2  
3 ▼ function f1() {  
4   var b = 2 // variável local  
5 ▼ if (true) {  
6   let c = 3 // variável de bloco  
7   console.log("f1() - if - c:", c)  
8 }  
9   console.log("f1() - a:", a)  
10  console.log("f1() - b:", b)  
11  console.log("f1() - c:", c)  
12 }  
13  
14 f1()
```

Console × Shell × ⋮

```
f1() - if - c: 3  
f1() - a: 1  
f1() - b: 2  
ReferenceError: c
```


Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Funções

Expressão de Função

A palavra-chave **function** pode ser usada para definir uma função dentro de uma expressão.

A principal diferença entre uma expressão de função e a declaração de uma função é o nome da função, o qual pode ser omitido em expressões de funções para criar funções anônimas.

```
function [name]([param1], [param2], ..., [paramN]) {  
  // Corpo da função  
}
```

Funções

Expressão de Função

funcoes_expressao.js × +

```
1 // Expressão de função não nomeada
2 var q = function(n) { return n * n }
3 var resultado = q(4) // q recebe o valor 16
4 console.log(resultado)
5
6 // Expressão de função nomeada
7 var s = function somar(n1, n2) { return n1 + n2 }
8 resultado = s(3, 4)
9 console.log(resultado)
```

Console × Shell × +

16
7

Curso de JavaScript

Funções



Agenda:

- Definição de Função
- Chamada de Função
- Argumentos *default*
- Escopo de Variáveis
- Expressão de Função
- *Arrow Functions*



Funções

Arrow Functions

Uma expressão *arrow function* possui uma sintaxe mais curta quando comparada a uma expressão de função.

```
(param1, param2, ..., paramN) => { // instruções }
```

Funções

Arrow Functions

funcoes_arrow.js × +

```
1 // Expressão com arrow function
2 var q = (n) => n * n
3 var resultado = q(4) // q recebe o valor 16
4 console.log(resultado)
5
6 // Expressão com arrow function
7 var s = (n1, n2) => n1 + n2
8 resultado = s(3, 4)
9 console.log(resultado)
```

Console × Shell × +

16
7

Funções

Comparações

```
funcoes_comparacao.js × +  
1 // declaração da função  
2 ▼ function somar1(n1, n2) {  
3     return n1 + n2;  
4 }  
5  
6 // declaração da função como expressão  
7 const somar2 = function(n1, n2) { return n1 + n2; };  
8  
9 // declaração da arrow function  
10 const somar3 = (n1, n2) => n1 + n2;  
11  
12 // chamada das funções  
13 console.log(somar1(1, 2));  
14 console.log(somar2(1, 2));  
15 console.log(somar3(1, 2));
```

Console × Shell × +

```
3  
3  
3
```

Sobre mim



Rafael Mesquita, Prof.

Prof. Dr. Formado em
Ciência da Computação pela
Universidade Federal de
Lavras