

Javascript

Lógica de Programação

Introdução a funções

As funções são blocos de comandos que permitem a execução de um conjunto de ações previamente definidas, por exemplo: somar múltiplos valores, inserir um novo elemento em uma página, verificar se o usuário preencheu um determinado campo no formulário, etc.

Entenda as funções como blocos que possuem em seu interior algum código que será executado, podendo ser reaproveitado quantas vezes você quiser. O código existente na função, não poderá ser acessado por ninguém que estiver fora dela.

Para facilitar o seu uso, as funções poderão receber parâmetros, eles representam valores que serão enviados para dentro delas, e utilizados em seu processamento.

As funções também podem ser usadas como valores, passadas como argumentos para outras funções e retornadas como resultados de outras funções. Isso torna as funções em Javascript uma parte fundamental do paradigma de programação funcional, que é amplamente usado na linguagem.

Function declaration

É a forma mais comum de declaramos uma função em Javascript. Para isso é utilizada a palavra-chave function seguida do nome desejado para a função.

Nesse tipo de declaração, ocorre o function hoisting, isso quer dizer que assim que a função for declarada, ela será elevada para o topo do código, procedimento comum às variáveis declaradas com var.

Criando uma função

```
function nomeFuncao(parâmetro1, parâmetro2,...) {
    //corpo da função
    return ...
}
//chamando a função
nomeFuncao(paramêtro1, parâmetro2,...);
```

Function expression

• • • • • •

Consiste na declaração de uma função diretamente em uma variável ou uma propriedade de um objeto.

Essas funções não possuem nome (função anônima) e, poderão ser utilizadas como argumentos para outras funções.

Arrow function

Consiste na declaração de uma Function Expression, de uma maneira um pouco mais resumida. Para isso devemos:

- Criar uma variável qualquer.
- Colocar o sinal de atribuição (=).
- Fazer a abertura e fechamento dos parênteses () para receber os parâmetros, caso necessário.
- Inserir o sinal de igual (=) seguido do sinal de maior (>), eles representam a arrow function.
- Abrir e fechar as chaves {} para o corpo da função.

```
const nomeVariavel = (parâmetro1, parâmetro2) => {

//corpo da função

return...
}
```

Arrays

Em Javascript, arrays são estruturas de dados que podem receber vários valores simultaneamente, inclusive de tipos diferentes. Os valores que compõem o array ficarão dentro de índices que poderão ser totalmente acessados e manipulados.

Para declararmos um array, podemos usar duas opções:

- Criando uma variável e atribuindo a ela o método construtor Array().
- Criando uma variável seguido da abertura e fechamento de colchetes [].

```
//declaração com construtor

const arrConstrutor = new Array(['Eu', 'Tu', 'Ele', ...);

//declaração literal

const arrLiteral = ['Eu', 'Tu', 'Ele', ...];
```

Métodos para arrays

Método	O que faz
Propriedade length	Retorna a quantidade de itens existentes no array.
unsfhit()	Insere um novo conteúdo no primeiro índice do array.
push()	Insere um novo conteúdo no último índice do array.
shift()	Remove o primeiro índice do array.

.

.

•

•

•

Método	O que faz
pop()	Remove o último índice do array.
splice()	Exclui e/ou insere elementos em um array partindo de um índice definido.
slice()	Copia elementos de um array para outro. Você deve indicar o índice inicial e o final. Não altera o array original.
indexOf()	Retorna o índice da primeira ocorrência de um elemento procurado no array. Retornará -1 se nada for encontrado.

•

•

Método	O que faz
lastIndexOf()	Retorna o índice da última ocorrência de um elemento procurado no array. Retornará-1 se nada for encontrado.
concat()	Permite que dois ou mais arrays sejam concatenados, gerando assim um novo array.
sort()	Organizar os conteúdos do array, colocando-os em ordem alfabética ou numérica.
reverse()	Inverte os índices de um array.

.

•

•

Método	O que faz
join()	Retorna uma string com os conteúdos de um array. É possível também, definir um separador para cada conteúdo.
includes()	Verifica se um determinado elemento está presente em um array. Ele retornará true ou false.
every()	Verifica se todos os elementos do array satisfazem uma mesma condição. Ele retornará true ou false.
some()	Executa um teste lógico verificando se pelo mesmo um conteúdo do array satisfaz a condição definida. Ele retornará true ou false.
find()	Executa um teste lógico e retornará o valor do primeiro elemento que satisfaça a condição definida.

. .

Percorrendo arrays

Podemos usar laços de repetição para percorrermos os elementos de um array, o mais comum deles é o **for**. Ele fará a iteração do array, tendo uma variável que será a responsável em garantir que o array seja inteiramente percorrido

```
const pessoas = ['Eu', 'Tu', 'Ele', 'Nós', 'Vós', 'Eles'];
//percorrendo o array
for(let i = 0; i < pessoas.length; i++){
   console.log((`Índice ${i} do array pessoas: ${pessoas[i]}`));
}</pre>
```

• • • • • •

Podemos também utilizar o laço **for in** que percorrerá cada índice do array de forma automática.

Ele deixará a sintaxe bem mais simples pois não precisaremos definir: onde o array começa, onde ele termina, e muito menos o incremento.

```
const pessoas = ['Eu', 'Tu', 'Ele', 'Nós', 'Vós', 'Eles'];
//percorrendo o array
for (const pessoa in pessoas){
    console.log((`Índice ${pessoa} do array Pessoas:
    ${pessoas[pessoa]}`));
}
```

Existe também o laço **for of**, ele percorrerá o array e retornará apenas os seus respectivos valores.

Ele não usa os índices, ele percorre diretamente todos os elementos do array.

```
console.log(`Pessoa: ${pessoa}`);
```

Temos também o laço **forEach**, ele percorrerá todo o array, e para cada elemento encontrado, executará uma função previamente definida

```
const pessoas = ['Eu', 'Tu', 'Ele', 'Nós', 'Vós', 'Eles'];
//percorrendo o array
pessoas.forEach( pessoa => {
    const idade = Math.ceil(Math.random()*50);
    console.log(`Nome do pessoa: ${pessoa} - Idade: ${idade}`);
})
```

Introdução objetos

Na linguagem Javascript, um objeto é uma coleção de propriedades que são usados para representar algo do mundo real, como por exemplo um usuário em alguma aplicação, um produto existente em uma loja virtual, ou até mesmo um herói em algum jogo.

Para criarmos um objeto, podemos declarar uma variável, abrir chaves { }, definir suas propriedades e seus respectivos valores, separando-as por vírgulas.

```
nome: 'Clark Kent',
  id: 12345,
  idade: 38,
  email: 'clark@planetadiario.com'
console.log(pessoa);
```

Utilizando factory function

Uma factory function é uma função que retorna um objeto, sendo uma forma muito comum de criarmos objetos em JavaScript.

```
function criarUsuario(id, nome, email,
senha) {
     id,
    nome,
    senha
```

```
//criando os objetos
const usuario1 =
criarUsuario(10,"Eu","eu@gmail.com","12345");
const usuario2 =
criarUsuario(11,"Tu","tu@gmail.com","password");
const usuario3 =
criarUsuario(12,"Ele","ele@gmail.com","admin");
console.log(usuario1);
console.log(usuario2);
```

Criando métodos

É possível também definir funcionalidades aos objetos, é o que chamamos de métodos.

A declaração é bem simples, basta definir um nome para o método seguido do bloco de instruções.

```
function criarUsuario(id, nome, email,
sėnha) {
    nome,
    mensagem: function() {
           console.log(`Usuário criado`);
```

```
//criando o usuário

const usuario1 =
criarUsuario(10,"Eu","eu@gmail.com","12345");

//chamando o método
usuario1.mensagem();
```

BIBLIOGRAFIA BÁSICA

- DUCKETT, Jon. Javascript & Jquery: desenvolvimento de interfaces interativas. Rio de Janeiro: Alta Books, 2015.
- LEPSEN, Edécio Fernando. Lógica de programação e algoritmos com JavaScript. São Paulo:
 Novatec, 2018.TERUEL.
- JAVASCRIPT.INFO. The Modern JavaScript Tutorial. 2022. Disponível em: https://javascript.info/.
- MDN WEB DOCS. Javascript. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript.
- LEARN.JS. Javascript. Disponível em: https://www.learn-js.org/.
- SOLOLEARN.JS. Introduction to Javascript. Disponível em: https://www.sololearn.com/learn/courses/javascript-introduction.
- BRAZILJS. Eloquente Javascript. Disponível em: https://github.com/braziljs/eloquente-javascript.

OBRIGADO



Copyright © 2023 | Professor Alexandre Russ

Todos os direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor/autor.

#