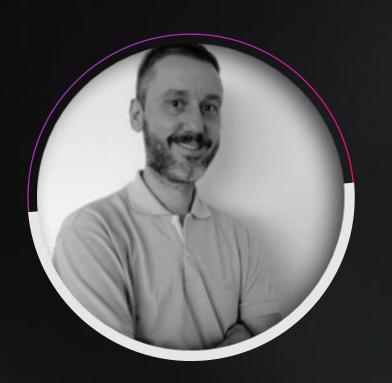




LÓGICA DE PROGRAMAÇÃO JAVASCRIPT



ALEX SANDER RESENDE DE DEUS

Professor

- Há 25 anos ensinando programação, é um apaixonado por tecnologia.
 Atualmente é coordenador de cursos na ETEC Albert Einstein do Centro
 Paula Souza.
- Na FIAP é professor da FIAP School e FIAP Corporate, lecionando C#, Banco de Dados e Desenvolvimento Mobile.

™ profalex.deus@fiap.com.br

AGENDA

Definição e histórico

1 AULA 1 Fundamentos Básicos da Programação - Tipos de dados, variáveis e estruturas básicas; Usando o GIT

2 AULA 2 Estruturas condicionais, operadores lógicos e switch case

3 AULA 3 Laços de Repetição: For, while,

4 AULA 4 Objetos nativos JavaScript - Date, String, Math

AGENDA

5 AULA 5 Arrays, Filter, Map
Reduce, Splite, for Each, for in, for of

6 AULA 6 Funções: tipos, declarações, escopo, retorno, clousers, call-back, IIFE, factory

7 AULA 7 DOM - Document Object Model

8 AULA 8 DOM - Document Object Model

AGENDA

9 AULA 9 Eventos

Objetos: Criação, prototype, getters, setters

10 AULA 10 Classes JavaScritp Assíncrono

AULA 6

FUNÇÕES JAVASCRIPT

Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.

É executada quando "algo" a invoca (chama).

Podemos dizer que uma função consiste em um objeto que possui código executável. É um recurso que fornece o encapsulamento no JavaScript, pois o que é definido dentro de uma função não pode ser acessado externamente.

Uma função JavaScript é definida com a palavra-chave function, seguida por um nome e depois parênteses ().

Os nomes das funções podem conter letras, dígitos, sublinhados e cifrões (mesmas regras das variáveis).

Os parênteses podem incluir nomes de parâmetros separados por vírgulas:

(parâmetro1, parâmetro2, ...)

O código a ser executado, pela função, é colocado entre chaves: {}

Os parâmetros da função são listados entre parênteses () na definição da função.

Os argumentos da função são os valores recebidos pela função quando ela é chamada.

Dentro da função, os argumentos (os parâmetros) se comportam como variáveis locais.

```
function name(parameter1, parameter2, parameter3) {
   // code to be executed
}
```

RETORNO DE FUNÇÃO

Quando o JavaScript atinge uma instrução return, a função para de ser executada.

Se a função foi chamada a partir de uma instrução, o JavaScript "retornará" para executar o código após a instrução de chamada.

O valor de retorno é "retornado" de volta para o "chamador":

```
function paraCelsius(f) {
  return (5/9) * (f-32);
}

alert(paraCelsius(80));
//Chama a função e exibe o resultado
```

```
<!DOCTYPE html>
    <html lang="pt-br">
        <head>
            <meta charset="UTF-8">
            <title>Funções</title>
       </head>
        <body>
            <script type="text/javascript" src="js/script.js"></script>
        </body>
    </html>
10
    var x = multiplicar(4, 3);
    alert (x);
    function multiplicar(a, b) {
      return a * b;
    //Neste exemplo a variável recebe o resultado
    //da função para exibir o valor depois.
```

FUNÇÕES AUTO-INVOCÁVEIS

Esse tipo de construção recebe o nome de função auto-invocável. Ou seja, ela executa assim que é definida. Em inglês, é também chamada de Immediately Invoked Function Expression (IIFE).

OBJETO ARGUMETS

Os Argumentos que uma função recebe são guardados em um objeto do tipo array chamado arguments. Quando você cria uma função, ela já vem com esse objeto embutido.

Então você pode usar diretamente esse objeto dentro de uma função para manipular os argumentos que ela recebe.

```
function frutas() {
  let texto="";
  for (var i = 0; i < arguments.length; i++) {
      texto += (arguments[i] + ",");
  }
  alert(texto);
}

frutas('Laranja', 'Mamão', 'Abacaxi', 'Melão', 'Banana', 'Maçã', 'Quiuí');</pre>
```

OPERADOR REST

A partir do ES6, surgiu o operador rest. Ele é representado por reticências Esse operador, quando colocado antes do último parâmetro nomeado de uma função, coleta todos os argumentos passados para esse parâmetro e coloca-os em um array.

No exemplo abaixo, o rest coletará, do segundo argumento em diante. Pois o primeiro, foi mapeado para o parâmetro a.

```
1  function coleta(a, ...b) {
2    alert(a);
3    alert(b);
4  }
5
6  coleta(1, 'teste', true, 'sim', 20);
```

PARÂMETRO PADRÃO

Outra novidade, surgida no mesmo pacote de especificações ES6, foi a dos parâmetros padrão. Vamos entender a utilidade deles.

Se eu declarar um parâmetro em uma função e não atribuir valor a ele, seu valor padrão será undefined.

Usando parâmetro padrão, ao não passar um argumento, o padrão será aplicado.

E passando um argumento, ele será atribuído normalmente ao parâmetro.

```
function retorna(a=2) {
       return a;
   //Passando o parâmetro a função
  //retornará o valor passado
  alert(retorna(3));
   //Sem passar o parâmetro, a função
  //retornará 2, que é o valor padrão
8
   alert(retorna());
```

OUTROS PARÂMETROS

Acabamos de ver como invocar funções passando como argumentos, valores primitivos. Mas, como funções são especiais, podem também receber objetos e outras funções como argumentos.

Posso criar objetos separadamente, e passá-los depois para dentro da função.

```
let pessoa = {
2
         nome: 'Gabriel',
        idade: 20
    };
5
    function mostra(argumento) {
         alert (`${argumento.nome}, ${argumento.idade}`);
    };
9
    mostra(pessoa);
10
```

ESCOPO DE FUNÇÕES

Escopos são limitadores de acesso em programação, e podem ser utilizados para estabelecer, por exemplo, que determinados dados não devem estar visíveis para além das fronteiras delimitadas por ele.

Funções criam escopo e, quando invocadas, retornam para o mundo externo, apenas aquilo que estiver indicado através da instrução return.

```
function minhaFuncao() {
   let minhaVariavel = 'Variável no escopo de função';
   return minhaVariavel;
}
alert(minhaFuncao());//Acessando a função a variável é retornada e exibida
alert(minhaVariavel);//Mas aqui não, pois a variável fica encapsulada na função
```

Por outro lado, ao declarar uma variável dentro da função, sem utilizar a palavra-chave var ou let, ela será criada no escopo global. Vamos para um exemplo, que é bem mais fácil de entender.

```
function minhaFuncao() {
    minhaVariavel = 'Variável no escopo de função';
    return minhaVariavel;
}
alert(minhaFuncao());//Acessando a função a variável é criada
alert(minhaVariavel);//E agora posso acessar diretamente
```

Ou seja, se você declara uma variável sem utilizar a palavra-chave var ou let, o JavaScript interpreta que essa variável deve ser criada no escopo global.

Esse é um detalhe sutil, mas que você deve tomar bastante cuidado. É altamente recomendável definir variáveis sempre usando var ou let, para que elas não vazem para fora das funções.

MOMENTO HANDS ON

Elabore um programa JavaScript que contem uma função denominada verificarIdade. Esta função recebe por parâmetro a idade de uma pessoa e retorna uma string informando se ela é maior ou menor de idade.

Elabore um programa em JavaScript que contém uma função denominada verificar Triangulo que recebe 3 valores por parâmetro. Verifique se estes valores formam um triângulo (um lado sozinho é sempre menor que a soma dos outros dois) e, se formarem um triângulo, escreva o tipo: Equilátero (3 lados iguais); Escaleno (3 lados diferentes); Isósceles (2 lados iguais)

Elabore um programa em JavaScript com uma função, denominada calcular Fatorial que recebe um valor por parâmetro e retorna o valor do fatorial calculado

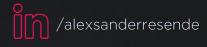
LEMBRE-SE DE VERSIONAR SEUS PROJETOS NO GIT

BIBLIOGRAFIA

- https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript
- https://developer.mozilla.org/pt BR/docs/Web/JavaScript/Reference/Global Objects/String
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math
- https://developer.mozilla.org/pt BR/docs/Web/JavaScript/Reference/Global Objects/Array
- https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM
- https://www.w3schools.com/
- http://docplayer.com.br/17393758-Javascript-eventos-e-objetos-nativos.html
- https://www.adalgisa-souza.appspot.com/javaScript/

OBRIGADO







Copyright © 2020 | Alex Sander Resende de Deus

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento,

é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

