



JavaScript

Prof. Henrique Mota

O que é javascript

JavaScript é uma linguagem de script. Por ser uma linguagem de script, seu código deve ser executado dentro de um interpretador. O JavaScript, para ser interpretado, deverá ser executado dentro de um navegador (browser).

JavaScript - Uma linguagem cliente

Outro termo técnico que é comum de ser dito sobre linguagem JavaScript, é que ela é *client side*, ou seja, ela age no lado do cliente.

Vantagem disso: diminui seu processamento no lado do serviço, diminuindo a escala vertical desses servidores.



Princípios do javascript



- Sintaxe
- Orientação a Objetos
- Callback (é uma função com próprio contexto).
- Contexto (referencia – this)
- Variável de forma dinâmica
- Funções de invocação imediata - `(function(){})();`
- Manipulação do DOM

Estudar on-line



Curso on-line:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>

<https://www.w3schools.com/js/>

Testar os códigos:

<https://jsfiddle.net>

hello word, fugindo da Maldição.

index.html

```
<!DOCTYPE HTML>
```

```
<html>
```

```
  <body>
```

```
    <p>Antes do script...</p>
```

```
    <script> alert( 'Hello, world!' ); </script>
```

```
    <p>...Antes do script.</p>
```

```
  </body>
```

```
</html>
```

Melhorando o Código

index.html

```
<!DOCTYPE HTML>
<html>
<head>
  <script src="/path/to/script.js" > </script>
</head>
<body>
  <p>Antes do script...</p>
  <script> alert( 'Hello, world!' ); </script>
  <p>...Antes do script.</p>
</body>
</html>
```

Melhorando o Código

index.html

```
<!DOCTYPE HTML>
<html>
<head> </head>
<body>
  <p>pagina</p>
  <script src="/path/to/script.js" > </script>
</body>
</html>
```


Melhorando o Código



script.js

```
alert('hello world');
```

```
console.log('hello world');
```

Variáveis

```
<script>var variavel;</script>
```

```
<script>
```

```
  var variavel = 1      // Inteiro  
  console.log(typeof variavel);
```

```
  var variavel = "1"    // String  
  console.log(typeof variavel);
```

```
  var variavel = true   // Booleano.  
  console.log(typeof variavel);
```

```
</script>
```

Exemplos

Exemplo 1

```
1.<script type="text/javascript">
2.var nome = prompt('Digite seu nome: ');
3.alert(nome + ', seja bem vindo!');
4.</script>
```

Exemplo 2

```
1.<script type="text/javascript">
2./* Este é um script para cálculo de idade! */
3.
4.// Declara o ano atual para fazer o cálculo
5.var anoAtual = 2014;
6.
7.// Pede que o usuário digite o ano em que nasceu
8.var anoNascimento = prompt('Digite o ano em que você nasceu.');
```

```
9.
10.// Calcula a idade do usuário e armazena na variável idade
11.var idade = anoAtual - anoNascimento;
12.
13.// Mostra ao usuário a idade que ele possui
14.alert("Sua idade é: " + idade + " anos");
15.</script>
```

Condicionais

Estrutura:

```
<script>
  if (condicao) {
    executar operacao
  } else {
    executa outra operacao
  }
</script>
```

Exemplo:

```
<script>
  var a = 6;
  if (a > 6) {
    document.write('Maior que 6');
  } else {
    document.write('Menor que 6');
  }
</script>
```

Condicionais

< : Menor

```
<script>
  var a = 6;
  if (a < 6) {
    document.write('Menor que 6');
  } else {
    if (a > 6) {
      document.write('Maior que 6');
    } else {
      document.write('Igual a 6!');
    }
  }
</script>
```

Condicionalis

== : Igual

```
<script>
  var a = 6;
  if (a == 6) {
    document.write('Igual a 6');
  }
</script>
```

Condicionalis

`>=` : Maior ou igual

```
<script>
  var a = 6;
  if (a >= 6) {
    document.write('Maior ou igual a 6');
  }
</script>
```

Condicionais

!= : Diferente

```
<script>
  var a = 6;
  if (a != 6) {
    document.write('Diferente de 6');
  }
</script>
```


Operadores Lógicos

&& : E

```
<script>
  var a = 6;
  if ((a > 1) && (a < 6)) {
    document.write('Maior que 1 E menor que 6');
  }
</script>
```

|| : OU

```
<script>
  var a = 6;
  if ((a > 1) || (a < 6)) {
    document.write('Maior que 1 OU menor que 6');
  }
</script>
```

! : NAO

```
<script>
  var a = 6;
  if (!((a > 1) || (a < 6))) {
    document.write('Não é maior que 1 OU menor que 6');
  }
</script>
```

Condicionais

Instrução switch

```
<script>
  switch(variavel) {
    case 1:
      document.write('Opção 1');
      break;
    case 2:
      document.write('Opção 2');
      break;
    case 3:
      document.write('Opção 3');
      break;
    default:
      document.write('Padrão');
      break;
  }
</script>
```

Estruturas de Repetição

Estrutura:

```
<script>
    for (condicaoInicial; condicaoFinal;acaoExecutar) {
        executa bloco de código;
    }
</script>
```

Exemplo:

```
<script>
    for (i=0; i<= 10; i++) {
        document.write('Linha '+i);
    }
</script>
```

Estruturas de Repetição

Estrutura:

```
<script>
  while (condicao) {
    bloco de operação
  }
</script>
```

Exemplo:

```
<script>
  var var1;
  while (var1 <= 10) {
    document.write('linha '+var1);
    var1++;
  }
</script>
```

Estruturas de Repetição

Estrutura:

```
<script>
  do {
    bloco de operacao
  } while (condicao);
</script>
```

Exemplo:

```
<script>
  var var1;
  do {
    document.write('linha '+var1);
    var1++;
  } while (var1 <= 10);
</script>
```

Array

Estrutura:

```
<script>
  var myArray = [];

  myArray.forEach(function(value, key) {
    console.log(value, key);
  });
</script>
```

Exemplo:

```
<script>
var myArray = [0,1,2,3,4,5,6,7,8,9,10];

myArray.forEach(function(value, key) {
  console.log(value, key);
});
</script>
```

Criando Funções

Estrutura:

```
<script>  
  function minhaFuncao() {  
    /* Corpo da Função */  
  }  
</script>
```

Exemplo:

```
<script>  
  // chamando a função sem passar  
  parâmetros.minhaFuncao();  
  // chamando a função passando um parâmetro.  
  minhaFuncao(variavel);  
  minhaFuncao('literal');  
  // chamando a função passando mais de um parâmetro.  
  minhaFuncao(variavel1,variavel2);  
  minhaFuncao('literal1','literal2');  
</script>
```

Criando Funções

Estrutura:

```
<script>  
  var variavel;variavel = minhaFuncao();  
</script>
```

Exemplo:

```
<script>  
  // chamando a função sem passar  
  parâmetros.minhaFuncao();  
  // chamando a função passando um parâmetro.  
  minhaFuncao(variavel);  
  minhaFuncao('literal');  
  // chamando a função passando mais de um parâmetro.  
  minhaFuncao(variavel1,variavel2);  
  minhaFuncao('literal1','literal2');  
</script>
```


Entendendo mais de Arrays

Podemos declarar array de 2 formas:

Forma 1:

```
var ft = new Array('a','b');
```

Forma 2:

```
var ft = ['a','b'];
```

Manipulação de Arrays

`ft.push('c');` - adicionar um elemento no fim do array;

`ft.unshift('c');` - adicionar um elemento no inicio do array;

`ft.pop();` - remove ultimo elemento do array

`ft.shift();` - remove primeiro elemento do array

`ft.splice(índice, quantidade)` - remove elementos no meu array;

Entendendo mais de Arrays

Pegar um elemento do array pelo indice:

```
Arr[indice];
```

Exemplo:

```
var arr = [1,2,3,5,6];
```

```
var index = arr[1];
```

Entendendo mais de Arrays

Pegar um índice do elemento no array:

```
array.indexOf('elemento');
```

Exemplo:

```
var arr = [1,2,3,5,6];
```

```
var index = arr.indexOf(1);
```

Entendendo mais de Arrays

Como concatenar um array:

```
array1.concat(array2);
```

Exemplo:

```
var arr1 = [1,2,3,5,6];  
var arr2 = [11,12,13,15,16];
```

```
var result = arr1.concat(arr2);
```

Exercicio array

1. Criar um array de numeros
2. Fazer um for nesse array
3. Verificar dentro do for se o numero for par acrescenta o numero + 1 no final do array
4. Verificar dentro do for se o numero for impar retira o numero do array

Ajuda: Para achar o par numero $\% 2 == 0$

Objetos

Podemos declarar objetos de 2 formas:

Forma 1:

```
var obj = new Object();
```

Forma 2:

```
var obj = {} (Mais utilizada - sugar code)
```

Exemplo:

```
var obj = {  
  nome = 'da';  
  email = 'da@d'  
};
```

Para visualizar o Objeto

```
console.log(obj);  
console.log(JSON.stringify(obj));
```

Objetos

Opa, tem mais formas:

```
var obj = {};
```

```
obj.model = 'modelo';  
obj.name = 'name';
```

```
console.log(obj);  
console.log(JSON.stringify(obj));
```

Pava visualizar o Objeto

```
console.log(obj);  
console.log(JSON.stringify(obj));
```

Objetos

Mais uma:

```
var obj = {};
```

```
obj["model"] = 'modelo';  
obj["name"] = 'name';
```

```
console.log(obj);  
console.log(JSON.stringify(obj));
```

Pava visualizar o Objeto

```
console.log(obj);  
console.log(JSON.stringify(obj));
```


Objetos

Mais uma, legal. Atributos dos objetos podem conter funções:

```
var obj = {};
```

```
obj["model"] = 'modelo';  
obj["name"] = 'name';  
obj.start = function() {  
  console.log('I`m ready');  
}
```

```
console.log(obj.start());
```

Para visualizar o Objeto

```
console.log(obj);  
console.log(JSON.stringify(obj));
```

Objetos

A última:

Funções são objetos.

```
var object = function(arg1, arg2) {  
    this.primeiroNome = arg1;  
    this.segundoNome = arg2;  
}
```

```
var x = new teste("Joao", "Silva");  
console.log(x.primeiroNome);
```

DOM e a árvore

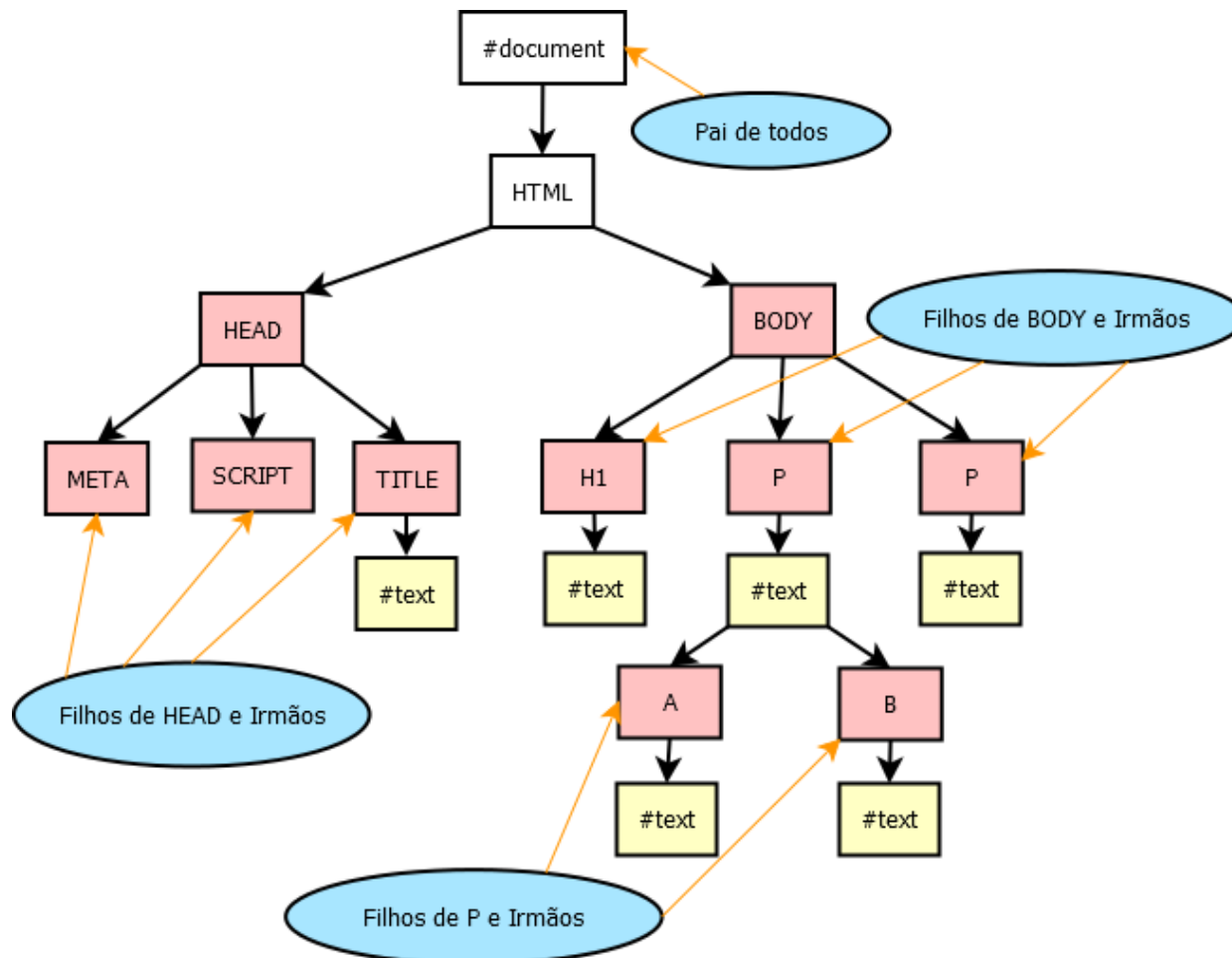
```
1 <!DOCTYPE html>
2
3 <html>
4
5 <head>
6
7 <meta charset="UTF-8">
8
9 <title>Demo</title>
10
11 <script src="meu_arquivo_javascript.js"></script>
12
13 </head>
14
15 <body>
16
17 <h1 id="id_h1" class="classe_h1">Sou um cabeçalho!</h1>
18
19 <p id="id_p" class="classe_p">
20 Um texto qualquer dentro de uma tag de parágrafo. Aqui também
21 temos outras tags, como <a href="#">um link<a>, ou um texto
22 <b>em negrito</b>.
23 </p>
24 <p id="id_p" class="classe_p">
25 Este é outro parágrafo.
26 </p>
27 </body>
28 </html>
```

DOM e a árvore

nodeName	id	class
▲#document		
└─html		
▲HTML		
▲HEAD		
#text		
META		
#text		
▲TITLE		
#text		
#text		
SCRIPT		
#text		
#text		
▲BODY		
#text		
▲H1	id_h1	classe_h1
#text		
#text		
▲P	id_p	classe_p
#text		
▲A		
#text		
#text		
▲B		
#text		
#text		
#text		
▲P	id_p2	classe_p2
#text		
#text		

DOM e a árvore

Existem elementos pai (**parent**), filhos (**childs**) e irmãos (**siblings**). Estes elementos são caracterizados na forma como estão na árvore, veja o mesmo exemplo na imagem abaixo:



DOM e a árvore

Localizando elementos (nós) na página

```
<p id="id_p" class="classe_p">Um texto qualquer</p>
```

```
var p = document.getElementById('id_p');
```

```
// Captura o evento load da página
```

```
window.onload=function(){
```

```
// Localiza o elemento com id "id_p"
```

```
var p = document.getElementById('id_p');
```

```
// configura a propriedade backgroundColor do elemento
```

```
p.style.backgroundColor='#0000FF';
```

```
}
```

DOM e a árvore

Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo</title>
<script src="meu_arquivo_javascript.js"></script>
</head>

<body>
<h1 id="id_h1" class="classe_h1">Sou um cabeçalho!</h1>
<p id="id_p" class="classe_p">
Um texto qualquer dentro de uma tag de parágrafo. Aqui também
temos outras tags, como <a href="#">um link</a>, ou um texto
<b>em negrito</b>.
</p>
<p id="id_p2" class="classe_p2">
Este é outro parágrafo.
</p>
</body>
</html>
```

DOM e a árvore

Exemplo:

```
window.onload=function(){  
  
  // Localiza o elemento com id id_p  
  var p = document.getElementById('id_p');  
  
  // Localiza os elementos a (links) dentro do p  
  var links = p.getElementsByTagName('a');  
  
  // Alerta o atributo href do primeiro link  
  alert(links[0].href);  
}
```


Contexto no Javascript

Toda função javascript tem um objeto associado a função, que é representada para palavra this. O ECMAScript chama isso ThisBinding.

Um evento que acontece toda vez que um código JavaScript é executado e um novo contexto de execução é estabelecido. O valor do this é constante e ele existe enquanto este contexto de execução existir.

```
function myFunc () {  
    console.log(this);  
}
```

```
myFunc();
```

Funções

JavaScript é uma linguagem funcional, por isso o grande uso de utilização de funções para realizar tarefas

O que é uma Função?

Uma função é um conjunto de instruções que podem ser executados sempre a tal função for chamada. As funções possibilitam a reutilização de código, já que você pode chamar a função várias vezes de dentro de seu código.

As funções podem, também, ter parâmetros, que permitem que você passe dados para a função. Funções também podem ter um valor de retorno, para que você possa retornar os resultados de uma operação (ou várias) para o código que a chamou.

Funções

Como Declarar Funções

Uma função deve ser declarada usando a palavra-chave `function` e, em seguida, definindo um nome (também conhecido como identificador).

Para declarar uma função:

- 1 - Nome da Função.
- 2 - Lista de argumentos para a função, entre parênteses e separados por vírgulas.
- 3 - Declarações JavaScript que definem a função, entre chaves `{ }`.

```
function nomeDaFuncao(parametros) {  
    // instrucoes  
}
```

Funções

Veja um exemplo mais prático:

```
/** * Soma dois valores */
```

```
function sum(a, b) {  
    return a + b;  
}
```

chamar a função:

```
var a = 1;  
var b = 2;
```

```
var result = sum(a, b);  
console.log(result);
```

Funções

Passando Objetos (Objetos são funções em Javascript)

```
function minhaFuncao(objeto) {  
    pessoa = "Zé";  
}
```

```
var pessoa = {  
    nome: "Zé",  
    idade: 1980  
};
```

```
var x, y;  
x = pessoa.nome;    // x recebe o valor "Zé" minhaFuncao(pessoa);
```

```
minhaFuncao(pessoa);  
y = pessoa.nome;    // y recebe o valor "Zé" //(a propriedade pessoa.nome foi alterada pela função)
```

Funções

Expressão de Função

A Expressão de Função, o nome não é obrigatório pois, na maioria dos casos, se trata de uma função anônima.

Exemplo:

```
var quadrado = function(x){  
    return x * x  
};  
  
console.log(quadrado(5)); //exibe 25
```

Funções

Qual a diferença declaração de função e Expressão de função ?

A diferença é que quando fazemos a Declaração de Função, permitimos que o parser analise previamente do que será executado enquanto a Expressão de Função é analisada em tempo de execução.

Funções de Callback

”Função de retorno – callback – é uma referência para um código executável, ou um pequeno código executável, que é passado como um parâmetro para um outro código.”

Wikipedia

```
var callMeBackFromTheFutureEvent = function(event) {  
    alert("Here is my answer!")  
}
```