

# JavaScript

Declaração de variáveis

## JavaScript

JavaScript é uma linguagem de programação que permite a você implementar itens complexos em páginas web — toda vez que uma página da web faz mais do que simplesmente mostrar a você informação estática — mostrando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos 2D/3D animados, etc. — você pode apostar que o JavaScript provavelmente está envolvido. É a terceira camada do bolo das tecnologias padrões da web, duas das quais (HTML e CSS) nós falamos com muito mais detalhes em outras partes da Área de Aprendizado.

- HTML é a linguagem de marcação que nós usamos para estruturar e dar significado para o nosso conteúdo web. Por exemplo, definindo parágrafos, cabeçalhos, tabelas de conteúdo, ou inserindo imagens e vídeos na página.
- CSS é uma linguagem de regras de estilo que nós usamos para aplicar estilo ao nosso conteúdo HTML. Por exemplo, definindo cores de fundo e fontes, e posicionando nosso conteúdo em múltiplas colunas.
- JavaScript é uma linguagem de programação que permite a você criar conteúdo que se atualiza dinamicamente, controlar multimídias, imagens animadas, e tudo o mais que há de interessante. Ok, não tudo, mas é maravilhoso o que você pode efetuar com algumas linhas de código JavaScript.

## O código

Um exemplo básico de uma estrutura básica de um código é:

```
<script>  
    // Isto é o símbolo de comentário em JavaScript na linha  
    document.getElementById("demo").innerHTML = "My First  
    JavaScript";  
    /*  
    Isto é o símbolo de comentário em JavaScript em várias linhas  
    */  
</script>
```

Em alguns exemplos antigos de JavaScript pode haver o atributo `type` em `<script type="text/javascript">`. Esse atributo não é mais necessário, pois JavaScript é a linguagem default em HTML.

Ele pode aparecer tanto dentro do `<head>` quanto dentro do `<body>`. O jeito de escrever é igual para ambos os casos e ele pode aparecer em qualquer ordem no código, porém colocar os scripts no final do `<body>` aumenta a velocidade de carregamento do site, pois a interpretação do script diminui o carregamento dos elementos. Assim, se recomenda colocar todos os `<script>` no final do `<body>`.

Igualmente ao CSS, os códigos em JavaScript podem ser feitos externamente, com a extensão `.js` e linkados no `<head>` do HTML. Para isto, se usa o atributo dentro `src` do `<script>`, assim:

```
<script src="myScript.js"></script>
```

A vantagem disto são:

- Separar o código do HTML e o JavaScript;
- Facilita a leitura e a manutenção de ambos;
- JavaScript em cache aumenta a velocidade de carregamento da página.

## Possibilidade de saída do javascript - Tipos de dados

Para mostrar dados em JavaScript, há 4 maneiras. Cada uma tem o seu jeito de escrever e sua peculiaridade.

### Usando o innerHTML

Utiliza-se o atributo ID definindo um elemento em HTML. Ao chamar essa propriedade **id**, ele realiza a função descrita por ele.

### Usando o window.alert()

Esse método mostra o conteúdo em uma caixa de alerta. Seu jeito de escrever é:

```
window.alert(NÚMEROS + "TEXTO");
```

Como o objeto **window** é global, ele pode ser retirado e utilizado apenas **alert()**.

### Usando o console.log()

Esse método é utilizado **APENAS** para debug da página. Para ver esse conteúdo, é necessário apertar F12 na página e ir na janela **Console**. Como ele não aparece no site, ele só serve para achar erros na página. Seu jeito de escrever é:

```
console.log(NÚMEROS + "TEXTO");
```

## O código

A lógica de um código em JavaScript é feito por instruções (igualmente um código de programação padrão). Ele necessita que as instruções ocorram na ordem em que são escritos para serem realizados.

Cada instrução é separada por ponto e vírgula (;). Se são colocados em uma linha ou em várias, o código é lido igual, pois o separador é o ; e não a quebra de linha (**ERRO MAIS COMUM É A FALTA DE ; NO FINAL**)

Igualmente ao HTML, ele ignora vários espaços, então pode utilizá-los para organizar o código.

## Diferença de JavaScript e outros códigos

- HTML não possui lógica de programação que nem JS.
- HTML faz a estrutura do site e o JS monta a sua lógica para criar interações e dinamismo.
- PHP é executado no servidor, então sem um banco de dados ou WAMP server ele não funciona. O JS é executado dentro do navegador web.
- C# necessita ser compilado como todas as outras linguagens de programação. JS é apenas executado dentro do navegador e **NÃO POSSUI DEBUG**. Ou seja, JS roda o código ele estando certo ou não, muitas vezes dificultando a localização dos erros.

## Variáveis - Informações importantes

Podemos começar as variáveis sempre com:

- Uma letra (JS é case sensitive): `Variavel` `variavel`  
`VaRiAvEl` `VaRiavel`
- Cifrão (\$) **Igualmente no PHP:** `$Variavel` `$variavel`  
`$VaRiAvEl`
- Underscore: `_Variavel` `_variavel` `_VaRiAvEl`  
`VaRiavel`

### Problemas:

- Não possuem tipo de variáveis.
- Podem mudar de Número (inteiro ou real) para Texto (string).

## Data Types - Tipos de Variáveis

- Os tipos são dinâmicos.
- JavaScript é em inglês, então sem acentos.
- JavaScript é em inglês, então a casa decimal é ponto (.) e não vírgula.

Ou seja:  $\frac{1}{2} = 0.5$  e não 0,5

- JavaScript é em inglês, então vírgula é divisora de variáveis.  
(para a criação de arrays)

Ou seja: 0,5 são os valores 0 para a primeira posição e 5 para segunda

- JavaScript é em inglês, então vírgula é divisora de variáveis.

Ela também pode separar declaração de variáveis.

```
var NUMERO = NUMERO, TEXTO = 'TEXTO', OBJETO =
{Termo1:"TEXTO", Termo2:"TEXTO"};
```

- Ao invés de ter elas separadas.
- `var NUMERO = NUMERO;`
- `var TEXTO = 'TEXTO';`
- `var OBJETO = {Termo1:"TEXTO", Termo2:"TEXTO"};`

**Objetos:** Há duas formatações: **Vetor (array) com [ colchetes ]**

**Objeto com { chaves }** A diferença entre elas é o indexador.

Vetor (array) - **Numérico**

Objeto - **Nominal**

## Data Types - Tipos de Variáveis

### Jeito de escrever (Sintaxe)

```
var VARIABEL;
var NUMERO = NÚMEROS;
var TEXTO = 'TEXTO';
var TEXTO = "TEXTO";
var TEXTO = NÚMEROS + "TEXTO";
var TESTE = (VARIABEL1 == VARIABEL2);
falso)
var VETOR = [NÚMEROS, NÚMEROS, NÚMEROS];
var VETOR = ["TEXTO", "TEXTO", "TEXTO"];
var OBJETO = {Prop1:"TEXTO", Prop2:"TEXTO"};
var COMPLEXO = [1, 2, 3]+"<br>";
var FUNCAO = function MinhaFuncao() {};
```

### Tipo de variável

Undefined (não definida)

Number

String (Texto)

String (Texto)

String (concatenado)

Boolean (variável binária lógica, teste de verdadeiro ou falso)

Object (Array)

Object (Array)

Object

Object (Vetor + Texto)

Function

Os tipos das variáveis são dinâmicos. Isto é, ao introduzir uma nova informação, ela muda seu tipo. **Importante** lembrar que números decimais em inglês possuem o ponto (.) como divisor de casa decimal. Ou seja, se quero falar 1/2, tenho que colocar 0.5 e não 0,5. A vírgula é divisora de variáveis, ou seja, a cada vírgula é uma nova variável ou termo do objeto ou string. Por isto é possível declarar as variáveis da maneira abaixo, onde a vírgula separa as variáveis e o ponto e vírgula termina a instrução :

```
var NUMERO = NUMERO, TEXTO = 'TEXTO', OBJETO =  
{Termo1:"TEXTO", Termo2:"TEXTO"};
```

Ao utilizar concatenar informações, sendo **NÚMEROS + "TEXTO"**, qualquer operação matemática anterior ao primeiro string de texto poderá realizar operações matemáticas.

Exemplo:

|   |             |         |
|---|-------------|---------|
| <code>var x = 16 + "Volvo";</code>      | Irá mostrar | 16Volvo |
| <code>var x = 8 + 8 + "Volvo";</code>   | Irá mostrar | 16Volvo |
| <code>var x = "8" + 8 + "Volvo";</code> | Irá mostrar | 88Volvo |
| <code>var x = "Volvo" + 8 + 8 ;</code>  | Irá mostrar | Volvo88 |

Para o tipo objeto, é possível observar duas formatações, o Vetor com [ colchetes ] e o Objeto com { chaves }. Os arrays são vetores com informações onde se necessita utilizar um indexador para acessar seu conteúdo.

Isto é, para acessar um conteúdo, devo saber em qual parte do vetor ele está e chamar o indexador para acessá-lo. Igualmente em C, o indexador se inicia no 0.

Exemplo:

```
var VETOR = [1, 5, 10];
```

Esse vetor possui 3 células de dados nele, então meu indexador vai de 0 a 2 ou (index-1)

Ao chamar cada index dentro do colchete, é possível acessar o conteúdo, ou seja:

|             |         |                          |
|-------------|---------|--------------------------|
| VETOR = [0] | retorna | 1                        |
| VETOR = [1] | retorna | 5                        |
| VETOR = [2] | retorna | 10                       |
| VETOR = [3] | retorna | undefined (nada pois não |

existe)

Já o objeto não guarda apenas uma informação, ele guarda uma propriedade e uma informação. Cada informação, além de possuir um valor, possui também uma propriedade. Assim, para acessar o conteúdo de uma célula, devo chamar a variável seguida de um ponto e sua propriedade.

Exemplo:

```
var OBJETO_Pessoa = {Nome:"NOME", Sobrenome:"SB",  
Idade:25};
```

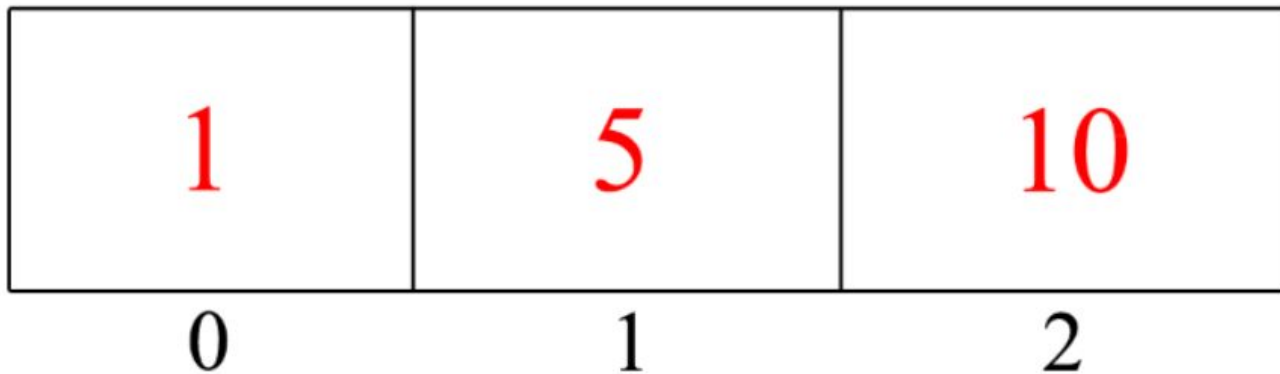
Assim, a diferença de um Vetor (Array) e de um Objeto é que o vetor trabalha com indexador numeral e o objeto trabalha com indexador nominal.

# Variáveis

## Objetos - Vetor (Arrays)

- Indexador numeral - como é em binário, começa com 0.

```
var VETOR = [1, 5, 10];
```



# Variáveis

## Objetos - Vetor (Arrays)

- Indexador numeral - como é em binário, começa com 0.

```
var VETOR = [1, 5, 10];
```

- Esse vetor possui 3 células de dados nele, então meu indexador vai de 0 a 2 ou (index-1).

```
VETOR = [0] retorna 1
```

```
VETOR = [1] retorna 5
```

```
VETOR = [2] retorna 10
```

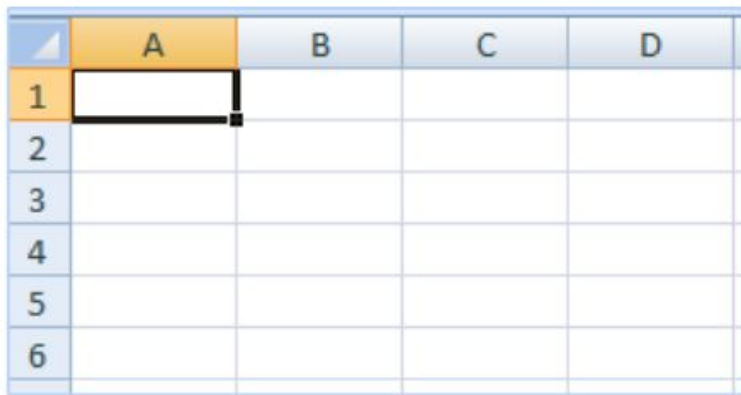
```
VETOR = [3] retorna undefined (nada pois não existe)
```



# Variáveis

## Objetos - Vetor (Arrays)

- Eles são muito utilizados no nosso dia a dia sem a gente reparar. Muitas vezes eles podem vir em formato de Matrizes (Arrays) e trabalhamos com dois indexs numéricos. Um exemplo disto é:
  - Excel ou qualquer outra planilha



|   | A | B | C | D |
|---|---|---|---|---|
| 1 |   |   |   |   |
| 2 |   |   |   |   |
| 3 |   |   |   |   |
| 4 |   |   |   |   |
| 5 |   |   |   |   |
| 6 |   |   |   |   |

A diferença é só que ele trabalha com o index numérico como linha e letras como coluna, mas seguem a mesma lógica.

Se usar só uma coluna, o número de cada linha é o index do vetor.

# Variáveis

## Objetos - Objetos

- Indexador nominal - como é uma palavra, chamamos de propriedade do objeto.

```
var OBJETO_Pessoa = {Nome:"NOME", Sobrenome:"SB", Idade:25};
```

|        |           |       |
|--------|-----------|-------|
| “Nome” | “SB”      | 25    |
| Nome   | Sobrenome | Idade |

# Variáveis

## Objetos - Objetos

- Indexador nominal - como é uma palavra, chamamos de propriedade do objeto.

```
var OBJETO_Pessoa = {Nome:"NOME", Sobrenome:"SB", Idade:25};
```

- Esse objeto possui 3 células de dados nele, então preciso saber as propriedades para acessar o conteúdo.

|                             |         |        |
|-----------------------------|---------|--------|
| OBJETO_Pessoa.Nome;         | retorna | "NOME" |
| OBJETO_Pessoa["Nome"];      | retorna | "NOME" |
| OBJETO_Pessoa.Sobrenome;    | retorna | "SB"   |
| OBJETO_Pessoa["Sobrenome"]; | retorna | "SB"   |
| OBJETO_Pessoa.Idade;        | retorna | 25     |
| OBJETO_Pessoa["Idade"];     | retorna | 25     |