# Instituto Federal de São Paulo

IFSP

EDUCAÇÃO PÚBLICA 100% GRATUITA

# Introdução à JavaScript


Professor Johnata Santicioli


# Sumários

- ✗ Introdução
- ✗ Sintaxe
- ✗ Operadores
- ✗ Variáveis
- ✗ Tipos de Dados


# Introdução


Estrutura | Estilo | Comportamento


# Introdução

- JavaScript é uma linguagem de programação *interpretada*. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do *lado do cliente* e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido.

Flanagan, David; Ferguson, Paula (2002). JavaScript: The Definitive Guide 4th ed. O'Reilly & Associates [S.l.]


# Introdução

- JavaScript:
- Interpretada pelo navegador
- Modelo de execução controlado por eventos
- Multiparadigma
- Baseado em objetos
- Tipagem dinâmica
- Case-sensitive


# Introdução

- O que pode-se fazer com JavaScript:
- Animações
- Verificar formulários
- Manipular elementos do documento HTML
- Manipular arquivos XML e JSON
- Server-side com node.js e deno.js


# Introdução

- Pode-se integrar o JavaScript no HTML das seguintes formas:

*@ Integração interna*

- No corpo da página (`<body>...</body>`)
- No cabeçalho (`<head>...</head>`)
- Em uma tag HTML

*@ Integração externa*

- Em um arquivo(.js) separado


# Introdução

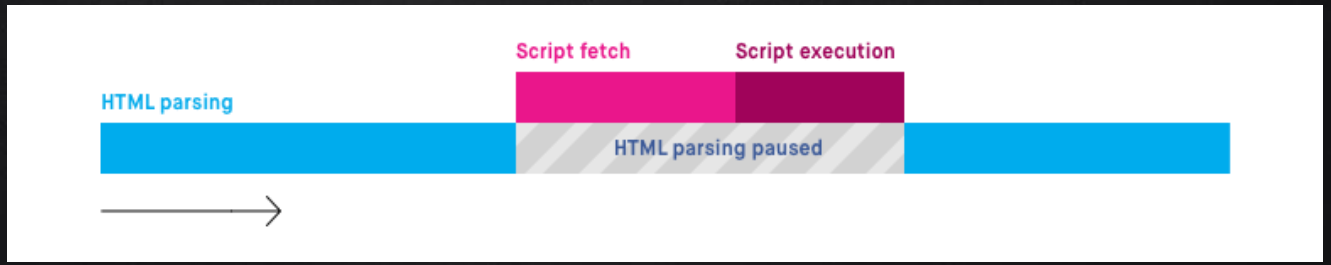
- Pode-se integrar o JavaScript no HTML das seguintes formas:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <script> alert("1-Olá Mundo!") </script> //no cabeçalho
  <script type="text/javascript" src="script.js"></script> //arquivo externa
</head>
<body>
  <input type="button" onclick="alert('3-Olá Mundo!')"> //na tag html
  <script> alert("2-Olá Mundo!") </script> //no corpo
  <script type="text/javascript" src="jquery.js"></script> //arquivo externa
</body>
</html>
```


# Introdução

- ✗ Execução normal
- ✗ Por que normalmente coloca-se os scripts no fim da página?

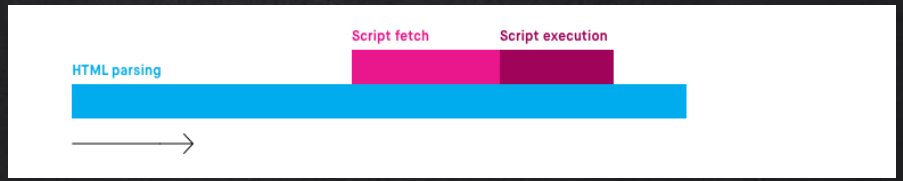
```
    ...
    <script type="text/javascript" src="script.js"></script>
</body>
```



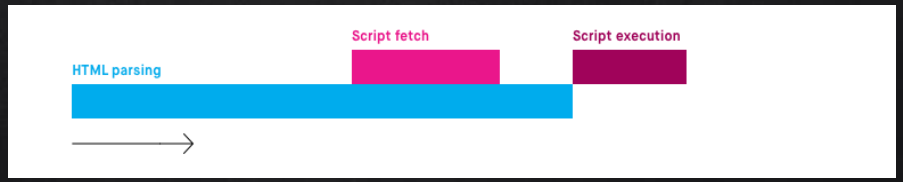

# Introdução

- Atributos async e defer

```
<script async src="script.js"></script> //só para arquivo externa
```




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




Saiba mais em: https://www.braziljs.org/p/diferencas-entre-async-e-defer


# Comentários

```
<script>
    //Comentário de linha

    /*
      Comentário
      de bloco
    */
</script>
```


# Tipo de Dados

*@ Tipos primitivos*

- Boolean (*lógico*)
- Null
- Undefined
- Number (*numérico*)
- BigInt (*número extensos e strings*)
- String (*caracteres*)
- Symbol (*símbolo único e imutável para chave de propriedade*)
- Function

*@ Tipos Objetos*

- Date, RegExp, Error, ... , objeto global e prototype
- Arrays


# Tipo de Dados

| Tipo | valor | Descrição | Exemplo |
|---|---|---|---|
| Boolean | `true` | Valor lógico verdadeiro | `var b = true;` |
| | `false` | Valor lógico falso | `var b = false;` |
| Number | `NaN` | **Not a Number** é o resultado de uma expressão com um operando que não pode ser convertido em valor numérico | `var a = NaN;`<br>`var a = 10 / "x";`<br>// a assume valor `NaN` |
| | `Infinity` | Representação de um valor infinito | `var a = Infinity;`<br>`var a = 10/ 0;`<br>// a assume valor `Infinity` |
| | `undefined` | conteúdo de variáveis não iniciadas | `var a;`<br>// a assume valor `undefined` |
| | `null` | representa o **não valor** , ou seja a inexistência de valor associado a uma variável | `var a = null;`<br>// x tem um não valor, ou seja `null` |


# Variáveis

- JavaScript é uma linguagem de tipagem dinâmica e fraca:
  - Não é necessário declarar o tipo de uma variável;
  - Todas as variáveis são objetos (referência);
  - A variável irá "alterar" o seu tipo de dado conforme os valores forem atribuídos:
    - Tipo de dado dinâmico:
      - var x; // x é indefinido
      - x = 5; // x é um número
      - x = "Johnata"; // x é uma string
      - x = true; // x é um valor lógico
      - x = null; // x é nulo

```
//template string
console.log(`Nome: ${x}`)
```


# Palavras reservadas

```
abstract
boolean break byte
case catch char class const continue
debugger default delete do double
else enum export extends
false final finally float for function
goto
if implements import in instanceof int interface
long
native new null
package private protected public
return
short static super switch synchronized
this throw throws transient true try typeof
var volatile void
while with
```

# Tipo de Dados

- Função typeof (operador) – retorna o tipo da variável ou constante

```
> a =9.5

> typeof(a)
'number'
```

| | |
|---|---|
| Object | 'object' |
| Array | 'object' |
| Function | 'function' |
| String | 'string' |
| Number | 'number' |
| Boolean | 'boolean' |
| null | 'object' |
| undefined | 'undefined' |

Atenção !

Atenção !


# Escopo de Variáveis e Constantes

- Declarando variáveis e constantes:
  - var x = 5 – usada para declarar tanto variáveis locais em funções como variáveis globais
  - let x = 5 – usada para declarar uma variável local de escopo de bloco
  - const x = 5 – usada para declarar uma constante local de escopo de bloco


# Escopo de Variáveis e Constantes

| keyword | const | let | var |
|---|---|---|---|
| global scope | NO | NO | YES |
| function scope | YES | YES | YES |
| block scope | YES | YES | NO |
| can be reassigned | NO | YES | YES |

# Operadores

- Operadores no JavaScript:

  - aritméticos: + , - , * , ** , / , %
  - atribuição: = , += , -= , *= , **= , /= , %= , ++ , --
  - relacionais: == , === , != , !== , < , <= , > , >=
  - lógicos: && , || , !
  - ternário: condição ? caso verdadeiro : caso falso

```
var salario = 1000
var bonus = salario * (salario > 1000 ? 0.10 : 0.15)
```


# Operadores Lógicos

O operador === e !== avalia os operandos e então os compara, sem realizar conversão de tipo.

```
> 1 === 1
  true
> 1 === '1'
  false
> 1 !== 1
  false
> 1 !== '1'
  true
>
```

Os operadores == e != são menos estritos. Se os valores dos operandos não forem do mesmo tipo, ele tenta algumas conversões de tipo e realiza a comparação novamente.

```
> 1 == 1
  true
> 1 == '1'
  true
> 1 != '1'
  false
> 1 != 1
  false
>
```

# Operadores e Prioridades

| Operador | Descrição |
|---|---|
| `. [] ()` | Acesso a propriedades, indexação, chamadas a funções e sub-expressões |
| `++ — - ~ ! new delete typeof` | Operadores unários e criação de objectos |
| `* / %` | Multiplicação, divisão, divisão módulo |
| `+ -` | Adição, subtracção, concatenação de *strings* |
| `<< >> >>>` | Deslocação de Bit |
| `< <= > >= instanceof` | Menor, menor ou igual, maior, maior ou igual, `instanceof` |
| `== != === !==` | Igualdade, desigualdade, igualdade estrita, e desigualdade estrita |
| `&` | AND bit a bit |
| `^` | XOR bit a bit |
| `\|` | OR bit a bit |
| `&&` | AND lógico |
| `\|\|` | OR lógico |
| `? :` | Operador condicional (ternário) |

# Referências



Você poderá estudar mais sobre html5  nas seguintes páginas:

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

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Grammar_and_types

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

https://www.braziljs.org/p/diferencas-entre-async-e-defer

https://www.alura.com.br/artigos/entenda-diferenca-entre-var-let-e-const-no-javascript

# THANKS!

## Até a próxima aula!!!

www.ifsp.edu.br
johnata.santicioli@ifsp.edu.br