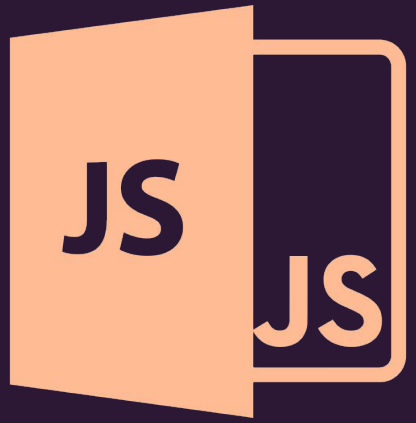
PROGRAMAÇÃO WEB

C/ YEHOSHUA OLIVEIRA



INTRODUÇÃO A JAVASCRIPT

O QUE É JAVASCRIPT?

I. Não é Java;
2. Anunciada em 1995 como uma linguagem de scripting para o navegador Netscape; Fonte
3. Hoje em dia é a principal linguagem usada na web;
4. A 7ª linguagem mais usada no mundo em Janeiro de 2021. Fonte

O QUE RAIOS É ECMASCRIPT?

- Especificação de uma linguagem de programação de propósito geral;
- ECMA-262 e ISO/IEC-16262.

LISTA (NÃO EXAUSTIVA) DE IMPLEMENTAÇÕES:

JScript · ActionScript · JavaScript

Quero *muito* ler a especificação

LISTA (EXAUSTIVA) DE VERSÕES DA ECMASCRIPT Fonte

- ES1 (1997);
- ES2 (1998);
- ES3 (1999);
- ES4 (2001);
- ES5 (abandonada);
- ES5.1 (2011);
- ES2015 (ES6);
- ES2016;
- ES2017;
- ES2018;
- ES2019;
- ES2020;
- ES.Next (?).

Quem define essas versões?
O que aconteceu com a ES5?

CARACTERÍSTICAS

1. Interpretada (e multiplataforma);
2. Tipagem fraca e dinâmica;
3. Sintaxe de chaves;
4. Multiparadigma (imperativo + funcional + OO via protótipo + orientado a eventos);
5. Funções são tipos de primeira classe;
6. Sensível à capitalização (*case sensitive*);
7. A especificação inclui API para lidar com coisas desde RegEx até Geolocalização. Fonte

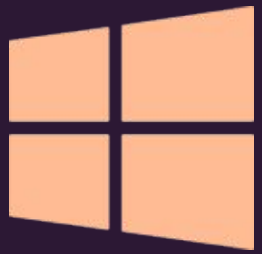
ADICIONANDO CÓDIGO JS À PÁGINA

É possível adicionar código JS de três formas:

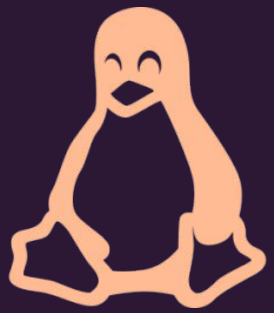
- Incluindo um arquivo separado;
- Adicionando código direto à página;
- Dentro de propriedades de evento em tags.

```
// index.html
<html>
  <head>
    <script src="script.js"></script>
  </head>
  <body>
    <button onclick="console.log('clicou')" />
    <script>
      console.log("Olá, mundo!");
    </script>
  </body>
</html>
```

TESTANDO CÓDIGO JS NO NAVEGADOR



[CTRL] + [SHIFT] + [I]



[CTRL] + [ALT] + [T]

[CTRL] + [SHIFT] + [C]

[CTRL] + [...] + [...]

Pra variar, nada na vida de usuário linux é fácil

MAS AFINAL, COMO SE PROGRAMA EM JS?

TIPOS DE DADOS Fonte

- Cadeia de caracteres (string);
- Numérico (number);
- Booleano (boolean);
- Função (function);
- Objeto (object);
- Nulo (null);
- Indefinido (undefined).

```
var nome = "João";
var idade = 12 // ; não é obrigatório

// Demonstrando tipagem fraca
nome = [1,2,3,'quatro']

// A partir da ES2015
const sobrenome = "Almeida"
let maioridade = 18
```

```
const verdade = true
const mentira = false

const funcao = function() {}
const objeto = {
  propriedade1: 1,
  propriedade2: true
}
```

OPERADORES ARITMÉTICOS [Fonte]

- Soma (+);
- Subtração (-);
- Divisão (/);
- Multiplicação (*);
- Módulo (%);
- Incremento (++);
- Decremento (--).

```
console.log(2 + 2)   // 4
console.log(5 - 9)   // -4
console.log(10 / 3) // 3.33335
console.log(3 * 10) // 30
console.log(7 % 4)   // 3

let n = 3;
console.log(n)    // 3
console.log(++n) // 4
console.log(n)    // 4
console.log(n++) // 4
console.log(n)    // 5
```

OPERADORES DE ATRIBUIÇÃO

- Soma (+=);
- Subtração (-=);
- Divisão (/=);
- Multiplicação (*=);
- Módulo (%=).

Fonte

```
let n = 1;
n += 5 // n = 6
n -= 2 // n = 4
n /= 2 // n = 2
n *= 4 // n = 8
n %= 5 // n = 3
```

OPERADORES DE COMPARAÇÃO [Fonte]

- Igual a (==);
- Diferente de (!=);
- Estritamente igual a (===);
- Estritamente diferente de (!==);
- Maior que (>);
- Menor que (<);
- Maior ou igual a (>=);
- Menor ou igual a (<=).

```
const nome = "João"
const idade = 35
const filhos = ["Maria",
"Gabriel"]

"João" == nome // true
idade > 35     // false
idade < 40     // true
idade >= 35    // true
filhos.length == 2 // true
```

OPERADORES LÓGICOS Fonte

- E (&&);
- Ou (||);
- Não (!);

```
const a = true
const b = false
console.log(!a) // false
console.log(a && b) // false
console.log(a || b) // true
console.log(a && !b) // true
```

ESCOPO DE VARIÁVEIS

Variáveis declaradas com `var`:

- São válidas dentro da função mais próxima;
- São processadas antes de qualquer outro código (*hoisting*):

```
nome = "Marcela"
var nome;

console.log(nome)
```

->

```
var nome;
nome = "Marcela"

console.log(nome)
```

ESCOPO DE VARIÁVEIS (CONT.)

Variáveis declaradas com `let` ou `const`:

- Têm escopo léxico (definido pelas chaves mais próximas).

Variáveis declaradas com `const`:

- São consideradas constantes - redefinições resultam em erro:

VETORES

- Elementos de vetores não precisam ser mesmo tipo;
- Têm métodos para se comportarem como listas, pilhas ou filas.

```
const arr = [
  'esta é uma string',
  'outra string',
  1, 3, 5
]
```

```
console.log(arr[1]) // outra string
console.log(arr[2] + arr[3]) // 4
console.log(arr.length) // 5
```

OBJETOS

- Podem conter propriedades arbitrárias;
- Podem herdar propriedades de outros objetos através do protótipo (não demonstrado aqui);
- Geralmente são criados através do literal de objetos:

OBJETOS (CONT.)

```
const obj = {
  a: 'A',
  b: 42,
  c: {
    a: ['a', 'b', 2]
  },
  d: function() {
    console.log('D!')
  }
}
```

```
console.log(obj.a)      // A
console.log(obj.b * 2) // 84
console.log(c.a[2])     // 2
obj.d() // D!
```

FUNÇÕES (A PARTE DIVERTIDA)

Funções em JS podem ser:

- Anônimas ou nomeadas;
- Passadas como parâmetros;
- Retornadas umas pelas outras;
- Armazenadas em variáveis.

```
function ola1(name) {
  console.log('Olá ' + name + '!')
}

const ola2 = function(name) {
  console.log('Olá ' + name + '!')
}

const multiplicar = function(a, b) {
  return a * b;
}
console.log(multiplicar(2, 5)) // 10
```

```
const cumprimentador = function(prefixo = "", sufixo = "") {
  return function(nome) {
    return prefixo + nome + sufixo;
  }
}

const olaMundo = cumprimentador("Olá, ", "!")
const helloWorld = cumprimentador("Hello ")

console.log(olaMundo("João")) // Olá, João!
console.log(helloWorld("Damien")) // Hello Damien
```

MANIPULANDO A PÁGINA COM JS

Quando o navegador interpreta uma página HTML, ele cria uma representação do documento em memória conhecido como Modelo de Objeto de Documento (do inglês, “Document Object Model”, ou *DOM*).

Quaisquer alterações a esse objeto são refletidas na página vista pelo usuário

```
const paragrafo = document.getElementById("paragrafo");

paragrafo.innerHTML = "Texto interessantíssimo!"
paragrafo.style.backgroundColor = "red"

paragrafo.classList.add("nova-classe")
paragrafo.classList.remove("classe-antiga")
```

INTERAÇÃO USANDO EVENTOS

- Pode-se criar funções JS para ser executadas quando ocorrem certos eventos na página;
- Cada evento tem argumentos e semânticas diferentes.

```
<html>
  <body>
    <button id="btn" onclick="console.log('clicou')" />
  </body>
</html>
```

```
const btn = document.getElementById("btn")

btn.onclick = function() {
  console.log('clicou')
}
```

LISTA (NÃO EXAUSTIVA) DE EVENTOS DO DOM

Tipo	Evento emitido quando...
onClick	...um elemento é clicado
onChange	...o valor de um elemento é alterado
onFocus	...um elemento recebe o foco do usuário
onKeyPress	...uma tecla é pressionada enquanto o elemento tem foco
onMouseOver	...o mouse entra sobre um elemento
onMouseOut	...o mouse sai de cima de um elemento
onSubmit	...um formulário é enviado

CRÉDITOS

- Imagem de fundo vetorizada por freepik - www.freepik.com
- Google Slides Code Syntax Highlighter


That's all, Folks!