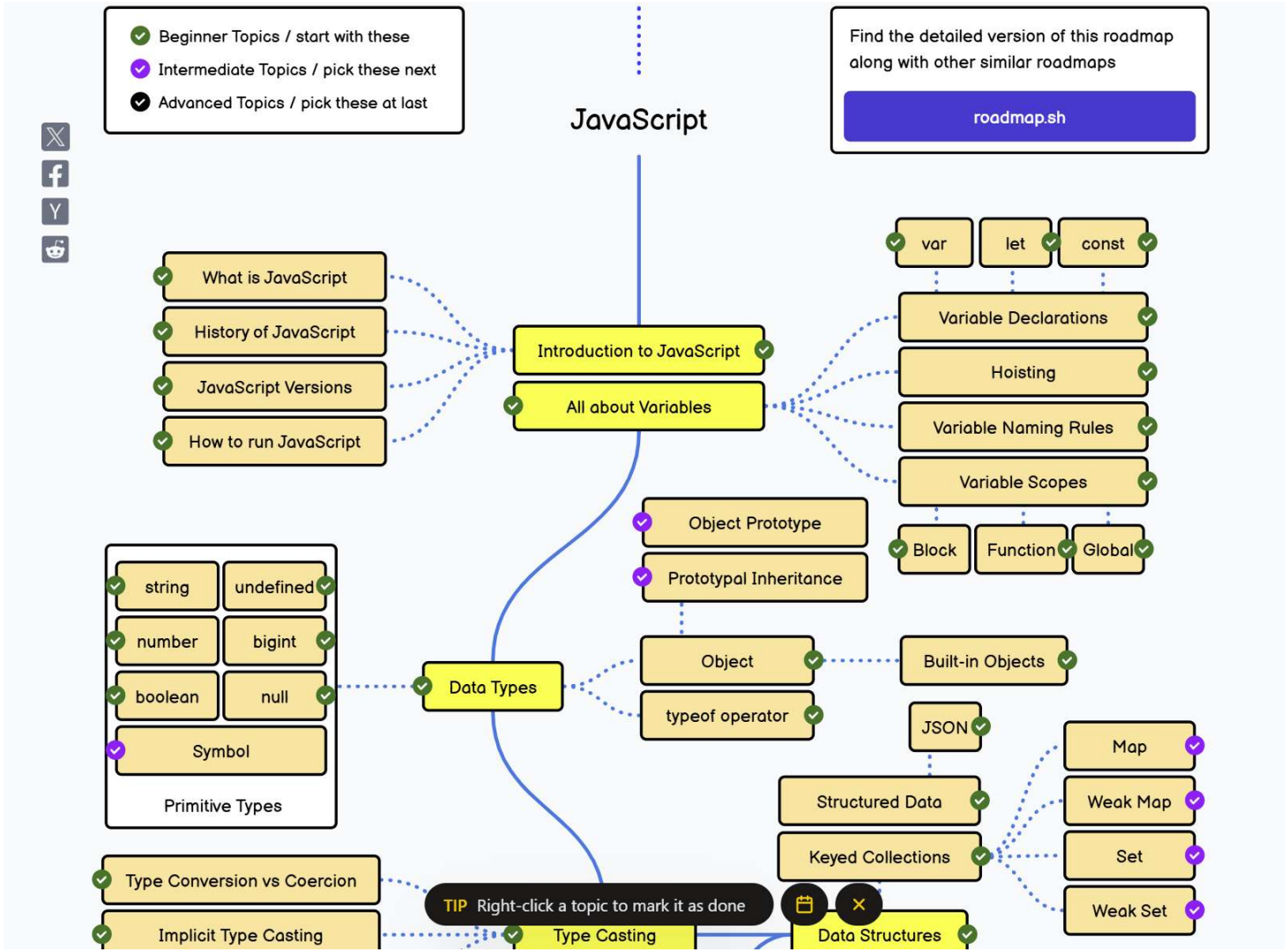
FPFtech
Escola Tecnológica

JavaScript

@fpftech.educacional



JavaScript Developer Roadmap: Step by step guide to learn JavaScript

O que é JavaScript?

FPFtech
Escola Tecnológica

- JavaScript é normalmente abreviado como JS.
- É uma das linguagens de promoção da principais tecnologias web, junto do HTML e CSS.
- Foi criada para dá interatividade para as páginas.

- JavaScript foi criado em 1995 com o nome Mocha.
- Recebeu o nome JavaScript em 1996.
- A escolha do nome "JavaScript" foi uma estratégia de marketing.
- Em 1997, JavaScript foi padronizado pela ECMA International sob o nome de ECMAScript.

ECMAScript

- O JavaScript foi padronizado pela organização EMCA International em 1997.
- Atualmente estamos na versão ECMAScript 2022 (ES13).
- O padrão ECMAScript é baseado na sintaxe da linguagem C.

- **Executado no Navegador**: JavaScript é uma linguagem de script que pode ser executada diretamente no navegador, sem a necessidade de compilação.

- **Codificar**: É possível escrever diretamente nas páginas HTML com a tag <script> ou em um arquivo separado e realizar a ligação utilizando a mesma tag <script>.

- **Orientada a Objetos**: Suporta programação orientada a objetos, embora de forma prototípica em vez de baseada em classes.
- **Interoperabilidade**: Pode ser integrado com HTML e CSS para criar páginas web dinâmicas e interativas.

- **Dinamicamente Tipada**: Não é necessário declarar explicitamente o tipo das variáveis; o tipo é determinado em tempo de execução.

- **Assíncrona e Event-Driven**: Utiliza eventos e callbacks para operações assíncronas, como requisições AJAX. Promises e async/await foram introduzidos para lidar com operações assíncronas de maneira mais eficiente.

Uso do JavaScript

FPFtech
Escola Tecnológica

- **JavaScript no Frontend**: No frontend, JavaScript é essencial para criar páginas web interativas, melhorando a experiência do usuário através de elementos dinâmicos.

- **JavaScript no Backend**: No backend, JavaScript, especialmente com Node.js, permite a criação eficiente de servidores e APIs para aplicações web.

DOM

Representação em Árvore

O DOM é uma estrutura hierárquica que representa os elementos da página web como nós em uma árvore, facilitando manipulações.

DOM

FPFtech
Escola Tecnológica

Manipulação pelo JavaScript: JavaScript permite a manipulação do DOM para modificar a estrutura, o estilo e o conteúdo da página, tornando-a interativa.

Eventos do Usuário: O DOM permite que páginas web respondam a eventos do usuário, como cliques e digitação, alterando dinamicamente a exibição.

Manipulação do DOM

Adição de Elementos: A manipulação do DOM permite adicionar novos elementos à página, criando dinâmicas interativas e engajadoras para os usuários.

Remoção de Elementos: Você pode remover elementos existentes da página para simplificar a interface e melhorar a usabilidade, ajustando a experiência do usuário.

Alteração de Elementos: A manipulação do DOM também permite alterar atributos e estilos de elementos, proporcionando uma experiência personalizada ao usuário.

Assíncrona e Event-Driven

Programação Assíncrona: JavaScript permite a execução de tarefas em segundo plano, essencial para aplicações web responsivas que não bloqueiam a interface do usuário.

Orientação a Eventos: A programação orientada a eventos em JavaScript permite responder a interações do usuário de forma eficiente, melhorando a experiência do usuário.

Chamadas de API: Executar chamadas de API de forma assíncrona previne bloqueios na interface, permitindo que a aplicação permaneça responsiva enquanto processa dados.

Formas de escrever

FPFtech
Escola Tecnológica

- Existem duas formas de escrever JavaScript.
  - **Incorporado**: Interno ao arquivo HTML.
  - **Externo**: Em um arquivo .JS e importado ao código HTML.

Interno/Incorporado

FPFtech
Escola Tecnológica

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    alert('Hello World!')
</script>
</body>
</html>
```

Externo

FPFtech
Escola Tecnológica

HTML

```
<body>
<script src="externo.js"></script>
</body>
```

JavaScript

```
alert('Hello World');
```

FPFtech
Escola Tecnológica

JavaScript Hello World

Hello World

FPFtech
Escola Tecnológica

- Crie um arquivo HTML chamado “hello-world.html”.
- Replique o código abaixo.

```
<body>
    <button onclick="alert('Hello World')">Hello World</button>
</body>
```

- Clique no botão.

Comentário de uma linha

- Os comentários de linha única são usados para adicionar notas ou explicações que ocupam apenas uma linha.
- Eles começam com duas barras (//).
- Tudo o que estiver à direita das barras na mesma linha será ignorado pelo interpretador JavaScript.

```
// Comentário de linha única
```

Comentário de Múltiplas linhas

- Os comentários de múltiplas linhas são usados para adicionar notas ou explicações que ocupam várias linhas.
- Eles começam com /* e terminam com */.
- Tudo o que estiver entre esses delimitadores será ignorado pelo interpretador JavaScript.

```
/*
* Comentário de Multiplas linhas
* */
```

Tipos de Variáveis: VAR

- **Escopo**: Funciona no nível de função ou global, ou seja, não tem escopo de bloco.
- **Hosting**: É "elevada" ao topo do seu escopo e inicializada como undefined. Isso significa que você pode usar a variável antes de declará-la, mas pode levar a bugs.
- **Mutabilidade**: Pode ser reatribuída.

```
var x : number = 10
```

Tipos de Variáveis: LET

FPFtech
Escola Tecnológica

- **Escopo**: Tem escopo de bloco, ou seja, só é acessível dentro do bloco onde foi definida.
- **Hoisting**: É "elevada", mas não inicializada, portanto, seu uso antes da declaração resulta em erro.
- **Mutabilidade**: Pode ser reatribuída, assim como var.

```
let x : number = 20;
```

Tipos de Variáveis: CONST

FPFtech
Escola Tecnológica

- **Escopo**: Igual ao let, com escopo de bloco.
- **Hosting**: Comportamento semelhante ao let, não inicializada durante o hosting.
- **Mutabilidade**: Não pode ser reatribuída. No entanto, objetos e arrays podem ter suas propriedades ou elementos modificados.

```
const x : number = 30;
```

Onde usar?

FPFtech
Escola Tecnológica

- **var**: Escopo de função ou global, hoisting, pode ser reatribuído.
- **let**: Escopo de bloco, hoisting (não acessível antes da declaração), pode ser reatribuído.
- **const**: Escopo de bloco, hoisting (não acessível antes da declaração), não pode ser reatribuído, mas permite a modificação de objetos e arrays.

Padrões de nomenclatura

Nome	Descrição	Usado para
camelCase	camelCase: Primeira palavra em minúsculo, próximas palavras com inicial maiúsculas.	Variáveis, funções, métodos e propriedades.
PascalCase	Todos as palavras começam com a letra maiúscula.	Classes, construtores e componentes.
snake_case	Palavras separadas por underscore, todas em minúsculas.	Normalmente não utilizado em JavaScript.
SCREAMING_SNAKE_CASE	Todas as letras maiúsculas, palavras separadas por underscore.	Constantes que representam valores fixos.

FPFtech
Escola Tecnológica

Tipos Primitivos

- **String**: Representa uma sequência de caracteres.
- **Number**: Representa números, tanto inteiros quanto de ponto flutuante.
- **Boolean**: Representa valores lógicos, que podem ser ‘true’ (verdadeiro) ou ‘false’ (falso).
- **Undefined**: Um tipo que indica que uma variável foi declarada, mas ainda não foi atribuída um valor.

Tipos Primitivos

```
// Undefined
var indefinido;
console.log(indefinido);

// Número
var numero : number = 1;
console.log(numero);

// Número grande
var numero_grande : bigint = 90000000000000000n;
console.log(numero_grande);

// String
var nome : string = "Douglas";
console.log(nome);

// Boolean
var ativo : boolean = true;
var inativo : boolean = false;
console.log(ativo);
console.log(inativo);
```

FPFtech
Escola Tecnológica

Tipos de Referência

- **Object**: O tipo mais genérico, que pode armazenar coleções de pares chave-valor.
- **Array**: Um tipo especial de objeto que armazena uma lista ordenada de valores.
- **Function**: Em JavaScript, funções são objetos de e podem ser atribuídos a variáveis.
- **Date**: Um objeto que representa datas e horas.

Tipos de Referência

```
// Object/Objeto
var objeto : {nome: string, sobrenome: string}  = {
    nome: "Maria",
    sobrenome: "Silva",
}
console.log(objeto);

// Array/Lista
var array : number[]  = [1, 2, 3];
console.log(array);

// Date/Datas
let data : Date  = new Date()
console.log(data);
```

JavaScript Object Notation (JSON)

- É um formato padrão baseado em texto.
- Foi criado para apresentar dados estruturados com base na sintaxe de objetos do JavaScript.
- É normalmente utilizado para transmitir dados na Web.

FPFtech
Escola Tecnológica

```json
{
  "slide": {
    "title": "Introdução ao JavaScript",
    "subtitle": "Os conceitos básicos para começar",
    "content": [
      "JavaScript é uma linguagem de programação usada no desenvolvimento web.",
      "Pode ser executada no navegador e no servidor.",
      "Permite criar páginas interativas e dinâmicas."
    ],
     "image": "https://example.com/images/javascript-logo.png",
    "notes": "Este slide apresenta uma visão geral para iniciantes em JavaScript."
  }
}
```

JSON

Operações

- **Aritméticas**: Soma (+), Subtração(-), Multiplicação(*), Divisão (/), Módulo(%), Exponenciação(**).
- **Concatenação**: Utilizar o operador mais (+) para integrar múltiplas strings.
- **Atribuição**: Atribuição simples (=), atribuição com adição (+=), atribuição com subtração (-=).
- **Comparação**: Igualdade (==), Identidade (===), Desigualdade (!=), Não identidade (!==), maior que (>), menor que (<) maior ou igual que (>=), menor ou igual que (<=).

Operações

Aritméticas

FPFtech
Escola Tecnológica

```
// Subtração
z = x - y;
console.log(z);

// Multiplicação
z = x * y;
console.log(z);

// Divisão
z = x / y;
console.log(z);
```

```
/*
* Operações Aritméticas
* */
let x :number = 10;
let y :number = 10;
let z;

//Soma
z = x + y;
console.log(z);
```

```
// Módulo
z = x % y;
console.log(z);

// Exponenciação
z = x ** y;
console.log(z);
```

FPFtech
Escola Tecnológica

Concatenação

- Você pode usar o operador + para concatenar/juntar strings.
- Interpolação: Interpolação em strings é uma técnica que permite inserir valores de variáveis ou expressões diretamente dentro de uma string.

FPFtech
Escola Tecnológica

Concatenação

```
let first_name : string  = "Douglas";
let second_name : string  = "Silva";
let name : string  = first_name + second_name;
console.log(name);

// Interpolação com template literals
let greatings : string  = `Hello, ${nome}!`;
console.log(greatings);
```

Atribuição

FPFtech
Escola Tecnológica

- As operações de atribuição são usadas para atribuir valores a variáveis. Você pode usar operadores de atribuição simples ou compostos.

Atribuição

```
// Declaração
let a;
let b;

// Atribuição simples
a = 10;
b = 10;
console.log('A:', a);
console.log('B:', b);

// Atribuição com adição
b += a;
console.log('A:', a);
console.log('B:', b);

//Atribuição com subtração
a -= b;
console.log('A:', a);
console.log('B:', b);

// Atribuição com divisão
b /= a;
console.log('A:', a);
console.log('B:', b);

// Atribuição com multiplicação
a *= b;
console.log('A:', a);
console.log('B:', b);
```

Comparação

FPFtech
Escola Tecnológica

- As operações de comparação são usadas para comparar valores e retornam um valor booleano (true ou false).

Comparação

```
let a : number = 10;
let b : number = 10;

// Igualdade
let igual : boolean = (5 == '5'); // igual é true
console.log('Igualdade: ', igual);

// Identidade
let identico : boolean = (5 === '5'); // identico é false
console.log('Identidade: ', identico);

// Desigualdade
let diferente : boolean = (5 != 6);
console.log('Desigualdade: ', diferente);
```

Comparação

```
// Não identidade
let naoIdentico : boolean  = (5 !== '5')
console.log('Identidade: ', naoIdentico);

// Maior
let maior : boolean  = (10 > 5);
console.log('Maior: ', maior);

// Menor
let menor : boolean  = (5 < 10);
console.log('Menor: ', menor);

// Maior ou igual a
let maiorOuIgualA : boolean  = (10 >= 5);
console.log('Maior ou igual a: ', maiorOuIgualA);

// Menor ou igual a
let menorOuIgualA : boolean  = (5 <= 10);
console.log('Menor ou igual a: ', menorOuIgualA);
```

Operadores Lógicos

FPFtech
Escola Tecnológica

- Os operadores lógicos são usados para combinar expressões booleanas e retornar um valor booleano (verdadeiro ou falso). Os principais operadores lógicos são:
- **AND (&&):** Retorna **true** se ambas as expressões forem verdadeiras. Caso contrário, retorna **false**.
- **OR (||):** Retorna **true** se pelo menos uma das expressões for verdadeira. Retorna **false** somente se ambas forem falsas.
- **NOT (!):** Inverte o valor booleano da expressão. Se a expressão for **verdadeira**, retorna **false**, e **vice-versa**.

Operadores Lógicos

```
// Operador: AND
// true, ambos são positivos
console.log(5 > 0 && 10 > 0);

// Operador: OR
// true, pelo menos um é positivo
console.log(5 > 0 || 10 < 0);

// Operador: NOT
// true, x não é igual a y
console.log(!(5 === 10));
```

Type Casting/Conversão de Tipo

FPFtech
Escola Tecnológica

- **Type casting** (ou conversão de tipo) é o processo de converter uma variável de um tipo de dado para outro em uma linguagem de programação.

FPFtech
Escola Tecnológica

Type Casting

```
let variavel : boolean = true;
console.log(typeof variavel); // boolean

variavel = String(variavel); // now value is a string "true"
console.log(typeof variavel); // string

let str : string = "123";
console.log(typeof str); // string

let num : number = Number(str); // becomes a number 123
console.log(typeof num); // number

let textoDecimal : string = "3.14";
let numeroDecimal : number = parseFloat(textoDecimal); // Converte para número decimal
console.log(numeroDecimal); // Saída: 3.14
```

Loop/Laços

- **FOR**: A instrução **FOR** cria um ciclo que consiste em três expressões opcionais, entre parênteses e separadas por ponto e vírgula, seguidas de uma instrução (normalmente uma instrução de bloco) a ser executada no ciclo.
- **WHILE**: A instrução WHILE permite executar um bloco de código repetidamente enquanto uma condição espeficiada for verdadeira.

FPFtech
Escola Tecnológica

FOR

```
let str : string = "";

for (let i : number = 0; i < 9; i++) {
    str = str + i;
    console.log(str);
}

console.log(str);

let int : number = 1;

for (let i : number = 0; i < 9; i++) {
    int = int + i;
    console.log(int);
}
```

```
const array1 : string[] = ["a", "b", "c"];

for (const element : string of array1) {
    console.log(element);
}

array1.forEach(
    element : string => console.log(element)
);
```

FPFtech
Escola Tecnológica

WHILE

```
// Declara uma variável contador iniciando em 1
let contador : number  = 1;

// Enquanto contador for menor ou igual a 5
while (contador <= 5) {
    console.log(contador);
    contador++;
}
```

Condicionais

FPFtech
Escola Tecnológica

- Em qualquer linguagem de programação, o código tem de tomar decisões e realizar ações em conformidade, dependendo de diferentes entradas.
- IF:
- TERNARY:

```ts
// Declara uma variável com um número
let numero : number  = 7;

// Verifica se o número é maior que 10
if (numero > 10) {
    console.log("O número é maior que 10");
} else {
    console.log("O número é 10 ou menor");
}
```

```ts
// Define a variável idade
let idade : number  = 20;

if (idade < 13) {
    console.log("Você é uma criança.");
} else if (idade >= 13 && idade < 20) {
    console.log("Você é um adolescente.");
} else if (idade >= 20 && idade < 60) {
    console.log("Você é um adulto.");
} else {
    console.log("Você é um idoso.");
}
```

```ts
const status : string  = numero >= 18 ? "O número é maior que 10" : "O número é 10 ou menor";
console.log(status);
```

Funções

FPFtech
Escola Tecnológica

- Funções em JavaScript são blocos de código reutilizáveis que realizam uma tarefa específica.
- Elas permitem que você agrupe instruções que podem ser executadas sempre que necessário, facilitando a organização e a manutenção do código.
- As funções podem receber entradas (chamadas de parâmetros) e podem retornar um valor após a execução.

Componentes de uma Função

FPFtech
Escola Tecnológica

- Palavra-chave **function**: Indica que você está declarando uma função.
- **Nome da função**: Um identificador que você usa para chamar a função.
- **Parâmetros** (opcionais): Variáveis que você pode passar para a função. Elas são definidas entre parênteses e podem ser usadas dentro do bloco de código da função.
- **Corpo da função**: O bloco de código que contém as instruções que a função executa.
- **return** (opcional): Uma instrução que encerra a função e pode retornar um valor para o local onde a função foi chamada.

Como as Funções Funcionam?

FPFtech
Escola Tecnológica

- **Declaração**: Você declara uma função usando a palavra-chave function, seguida pelo nome da função e os parâmetros.
- **Chamada**: Para executar a função, você a chama pelo nome e passa os argumentos necessários.
- **Execução**: Quando a função é chamada, o JavaScript executa o código dentro do corpo da função.
- **Retorno**: Se a função contém uma instrução return, ela retorna um valor que pode ser usado fora da função.

FPFtech
Escola Tecnológica

Funções

```
// Funções são blocos de código que podem ser reutilizados
// Funções podem receber parâmetros e retornar valores

// Funções podem ser declaradas de várias formas
function imprimirOla() {
    console.log("Olá");
}
imprimirOla();

// Funções podem receber parâmetros
// Parâmetros são variáveis que podem ser passadas para a função
function imprimirSoma(var1, var2) {
    console.log(var1 + var2);
}
imprimirSoma(1, 1);
```

Funções

```
// Funções podem retornar valores
// Retornar valores significa que a função pode devolver um valor para quem a chamou
function retornaASoma(var1, var2) {
    return (var1 + var2);
}

let soma = retornaASoma(1, 1);
console.log(soma);
```

Document Object Model (DOM)

FPFtech
Escola Tecnológica

- Com o HTML DOM, o JavaScript pode acessar todos os elementos de um documento HTML.
- Com o DOM é possível:
  - Adicionar, criar e alterar elementos HTML.
  - Remover estilos e elementos.

FPFtech
Escola Tecnológica

Acessar Elementos:

- Você pode acessar elementos do DOM usando métodos como:
  - **getElementById:** Permite selecionar um único elemento com ID específico.
  - **getElementsByClassName:** Retorna uma coleção de elementos que possuem uma classe específica.
  - **getElementsByTagName:** Permite selecionar todos os elementos de uma tag específica.
  - **querySelector:** Permite selecionar o primeiro elemento que corresponde a um seletor CSS.
  - **querySelectorAll:** Retorna todos os elementos que correspondem ao selector CSS.

Acessar elementos

```
/**
 * Seletores de Elementos
 * Os seletores de elementos são usados para acessar e manipular elementos HTML no DOM.
 */

// Retorna um único elemento com um ID específico.
const elementoPorId = document.getElementById('identificador');

// Retorna uma coleção de elementos que possuem uma classe específica.
const elementoPorClass = document.getElementsByClassName('classe-css');

// Retorna todos os elementos de um tipo específico (tag) no documento.
const elementoPorTagName = document.getElementsByTagName('tag');

// Retorna o primeiro elemento que corresponde ao seletor CSS
// Utiliza os mesmos padrões dos seletores CSS
let elementoPorQuery = document.querySelector('.classe-name');

// Retorna todos os elementos que correspondem ao seletor CSS
// Utiliza os mesmos padrões dos seletores CSS
const elementoPorQueryAll = document.querySelectorAll('#titulo');
```

Como modificar o DOM?

FPFtech
Escola Tecnológica

- **textContent**: É usada para obter ou definir o texto contido em um elemento, ignorando qualquer marcação HTML. Quando você usa textContent, o texto é tratado como texto puro.
- **innerHTML**: É usada para obter ou definir o conteúdo HTML de um elemento. Isso significa que você pode incluir tags HTML
- **style**: É usada para acessar e modificar os estilos do elemento.

Como modificar o DOM?

FPFtech
Escola Tecnológica

- **classList**: Fornece uma lista de classes CSS associadas a um elemento. Ela permite adicionar, remover e alternar classes.
- **getAttribute**: Usado para obter o valor de um atributo específico de um elemento.
- **setAttribute**: Usado para definir ou alterar o valor de um atributo específico de um elemento.
- **removeAttribute**: Usado para remover um atributo específico de um elemento.

FPFtech
Escola Tecnológica

```
<body>
    <div id="titleDiv">Título Principal</div>

    <p class="paragraph">Este é o primeiro parágrafo</p>
    <p class="paragraph">Este é o segundo parágrafo</p>

    <label for="inputNumber">Número: </label><br>
    <input
        type="text" name="inputNumber" id="inputNumber"
        placeholder="Digite um número">

    <script src="dom.js"></script>
</body>
```

DOM

FPFtech
Escola Tecnológica

```
let elemento = document.getElementById('titleDiv');

// textContent
let text = elemento.textContent; // obter o valor
elemento.textContent = "Novo texto"; // Definir valor

// innerHTML
text = elemento.innerHTML; // obter o valor
elemento.innerHTML = "<h1 id='titleH1'>Novo texto</h1>"; // Definir valor utilizando HTML

//style
elemento = document.getElementById('titleH1');
elemento.style.color = 'blue'; // Definir estilo
elemento.style.backgroundColor = 'red'; // Definir estilo
elemento.style.fontSize = '3rem'; // Definir estilo
```

dom

```
// classList

// Utiliza o primeiro elemento com a classe paragrafo
elemento = document.getElementsByClassName('paragraph')[0];
elemento.classList.add('text-warning'); // Adiciona classe ao elemento
elemento.classList.remove('text-warning'); // Remove classe ao elemento

// Seletores que retornam múltiplos elementos podem ser manipulados com um loop
elemento = document.getElementsByClassName('paragraph'); // Busca todos os elementos com a classe paragrafo
for (let i = 0; i < elemento.length; i++) {
    elemento[i].classList.toggle('text-warning'); // Alterna entre adicionar e remover a classe
}

elemento = document.getElementById('inputNumber'); // Busca a input pelo ID
attr = elemento.getAttribute('type'); // Busca o valor do atributo type da input nome
console.log(attr);
//Altera o atributo type da input para number
elemento.setAttribute('type', 'number');
elemento.removeAttribute('placeholder'); // Remove o atributo name da input nome
```

dom

Como modificar o DOM?

FPFtech
Escola Tecnológica

- **createElement**: Usado para criar um novo elemento HTML.
- **appendChild**: Usado para adicionar um novo nó (elemento) como filho de um elemento existente.
- **insertBefore**: Usado para inserir um novo nó antes de um nó existente dentro de um elemento pai.

preencher-uma-tag-select

```
<body>
    <div class="container" id="app">
        <!-- O conteúdo será criado via JavaScript -->
    </div>

    <script>
        // Obter referência do container onde colocaremos os elementos
        const container = document.getElementById('app');

        // Criar a label
        const label = document.createElement('label');
        label.htmlFor = 'frutas-select';
        label.textContent = 'Escolha uma fruta:';
        label.style.fontWeight = 'bold';
        label.classList.add('form-label');
        label.classList.add('m-1');
```

preencher-uma-tag-select

```
        // Criar o select
        const select = document.createElement('select');
        select.id = 'frutas-select';
        select.className = 'form-select';

        // Lista de frutas para popular os options
        const fruits = ['Maçã', 'Banana', 'Laranja', 'Uva', 'Abacaxi'];

        // Criar options e adicionar ao select
        fruits.forEach((fruit) => {
            const option = document.createElement('option');
            option.value = fruit.toLowerCase();
            option.textContent = fruit;
            select.appendChild(option);
        });

        // Adicionar o select ao container
        container.appendChild(select);

        // Inserir a label antes do select usando insertBefore
        container.insertBefore(label, select);
    </script>
</body>
```

Como modificar o conteúdo?

FPFtech
Escola Tecnológica

- **removeChild**: Usado para remover um nó filho de um elemento pai. Você precisa passar o nó filho que deseja remover como argumento.
- **remove**: Usado para remover um elemento do DOM. Ele não requer um elemento pai, pois o próprio elemento chama o método.

Eventos

FPFtech
Escola Tecnológica

- Os eventos em JavaScript são divididos em grupos, sendo eles:
  - Eventos de Mouse
  - Eventos de Teclado
  - Eventos de Formulário
  - Eventos de Janela
  - Eventos de Toque
  - Eventos de Arrastar
  - Eventos de Mídia

Eventos de Mouse

- **click**: Ocorre quando um elemento é clicado.
- **dblclick**: Ocorre quando um elemento é clicado duas vezes rapidamente.
- **mouseover**: Ocorre quando o ponteiro do mouse entra na área de um elemento.
- **mouseout**: Ocorre quando o ponteiro do mouse sai da área de um elemento.
- **mousedown**: Ocorre quando um botão do mouse é pressionado sobre um elemento.
- **mouseup**: Ocorre quando um botão do mouse é liberado sobre um elemento.

Eventos de Teclado

FPFtech
Escola Tecnológica

- **keydown**: Ocorre quando uma tecla é pressionada.
- **keyup**: Ocorre quando uma tecla é liberada.
- **keypress**: Ocorre quando uma tecla é pressionada (embora seja considerado obsoleto em alguns contextos).

Eventos de Formulário

FPFtech
Escola Tecnológica

- **submit**: Ocorre quando um formulário é enviado.
- **change**: Ocorre quando o valor de um elemento de formulário (como um campo de entrada ou uma caixa de seleção) é alterado.
- **input**: Ocorre quando o valor de um campo de entrada é alterado (inclui eventos de digitação).
- **focus**: Ocorre quando um elemento (como um campo de entrada) recebe foco.
- **blur**: Ocorre quando um elemento perde o foco.

Eventos de Janela

- **load**: Ocorre quando a página e todos os seus recursos (como imagens e scripts) foram completamente carregados.
- **resize**: Ocorre quando a janela do navegador é redimensionada.
- **scroll**: Ocorre quando a página é rolada.

Eventos de Toque (Touch Events)

- **touchstart**: Ocorre quando um toque é detectado na tela.
- **touchmove**: Ocorre quando um toque é movido na tela.
- **touchend**: Ocorre quando um toque é liberado.

Eventos de Arrastar (Drag Events)

FPFtech
Escola Tecnológica

- **drag**: Ocorre enquanto um elemento está sendo arrastado.
- **dragstart**: Ocorre quando o arrasto de um elemento começa.
- **dragend**: Ocorre quando o arrasto de um elemento termina.
- **dragover**: Ocorre quando um elemento é arrastado sobre outro elemento.
- **drop**: Ocorre quando um elemento é solto em um alvo de arrasto.

Eventos de Mídia

- **play**: Ocorre quando um vídeo ou áudio começa a ser reproduzido.
- **pause**: Ocorre quando a reprodução de um vídeo ou áudio é pausada.
- **ended**: Ocorre quando a reprodução de um vídeo ou áudio termina.
- **volumechange**: Ocorre quando o volume de um vídeo ou áudio é alterado.

FPFtech
Escola Tecnológica

Input - V1

```html
<div class="container">
    <form action="" method="post">

        <div class="mb-3">
            <label for="inputName" class="form-label">Name</label>
            <input type="text" class="form-control" name="inputName" id="inputName" oninput="ola()"
                placeholder="Digite seu nome" />
        </div>

        <h1 id="titleHello"></h1>

    </form>
</div>

<script>
    function ola() {
        const inputName = document.getElementById('inputName');
        const titleHello = document.getElementById('titleHello');
        titleHello.textContent = `Olá ${inputName.value.trim()}`;
    }
</script>
```

Input – v2

```html
<div class="container">
    <form action="" method="post">

        <div class="mb-3">
            <label for="inputName" class="form-label">Name</label>
            <input type="text" class="form-control" name="inputName" id="inputName" placeholder="Digite seu nome" />
        </div>

        <h1 id="titleHello"></h1>

    </form>
</div>

<script>

    const inputName = document.getElementById('inputName');

    inputName.addEventListener('input', () => {
        const titleHello = document.getElementById('ola');
        titleHello.textContent = `Olá ${inputName.value.trim()}`;
    });

</script>
```

```
<body>

    <div class="container">

        <button class="btn btn-info" id="btnButton">Event Listener</button>

        <div id="divResult" class="bg-primary text-center text-bg-primary fs-1 m-3"></div>

        <script src="eventos.js"></script>
    </div>

</body>
```

click

```
var divResult = document.getElementById("divResult");
var btnButton = document.getElementById("btnButton");

function click() {
    divResult.textContent = "Você clicou no botão!";
}

function mouseOver() {
    divResult.textContent = "Você passou o mouse por cima!";
}

function mouseOut() {
    divResult.textContent = "Você retirou o mouse!";
    setTimeout(() => {
        divResult.textContent = null;
    }, 3000)
}

btnButton.addEventListener("click", click);

btnButton.addEventListener("mouseover", mouseOver);

btnButton.addEventListener("mouseout", mouseOut);
```

click

FPFtech
Escola Tecnológica

Cadastro de Usuário

Nome completo:

Digite seu nome

E-mail:

nome@exemplo.com

Telefone:

(00) 00000-0000

Cadastrar

Lista de Usuários Cadastrados

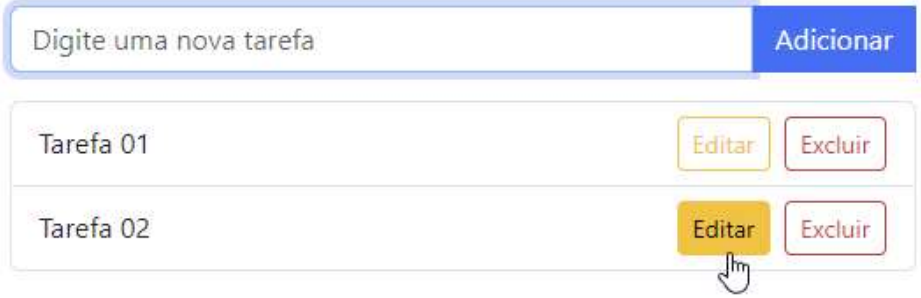
Nome	E-mail	Telefone

Cadastro de usuários

- Ao preencher todos os campos e clicar em cadastrar, os dados devem ser enviados para a tabela abaixo.
- O layout deve ser ajustado de acordo com a resolução.

FPFtech
Escola Tecnológica

Gerenciador de Tarefas



Gerenciador de Tarefas

- Funcionalidades:
  - Cadastra uma tarefa.
    - Ao cadastrar uma tarefa deve ser limpo o campo de entrada.
  - Alterar uma tarefa cadastrada.
  - Excluir uma tarefa cadastrada.

FPFtech
Escola Tecnológica

Cronômetro Digital

Cronometro

- Funcionalidades:
  - Iniciar á contagem.
  - Pausar a contagem.
  - Zerar a contagem.