# 1- O que é JavaScript

## História do JavaScript:

JavaScript é uma linguagem de programação interpretada que tem se tornado um dos pilares do desenvolvimento web moderno. Originalmente concebida para adicionar interatividade às páginas da web, o JavaScript evoluiu para uma linguagem completa, capaz de desenvolver complexas aplicações web, mobile, e até mesmo servidores com o Node.js.

- **Versatilidade**: O que diferencia o JavaScript de outras linguagens é sua capacidade de rodar tanto no cliente (navegador) quanto no servidor (Node.js), tornando-o uma ferramenta essencial para full-stack developers.
- **Facilidade de Aprendizado com Profundidade Técnica**: Apesar de ser amigável para iniciantes, o JavaScript possui profundidade suficiente para construir aplicações complexas e escaláveis, sendo usado por gigantes da tecnologia.

### Um pouco da história:

- **Nascimento e Objetivos Iniciais**: Criado por Brendan Eich em 1995, na Netscape, o JavaScript foi inicialmente projetado para tornar o conteúdo da web mais dinâmico. Naquela época, a internet era majoritariamente estática, somente HTML e CSS, e o JavaScript surgiu como uma solução para trazer interatividade.
- **Evolução e Padronização (ECMAScript)**: Com o passar dos anos, a linguagem sofreu várias modificações e melhorias. Para padronizar a linguagem, foi criada a especificação ECMAScript, que é atualizada anualmente com novas funcionalidades e melhorias.
- **A Era Node.js e o JavaScript no Servidor**: O lançamento do Node.js em 2009 foi um marco para o JavaScript, permitindo seu uso fora dos navegadores e trazendo capacidades de desenvolvimento backend para a linguagem.

## Características Técnicas do JavaScript:

- **Tipagem Dinâmica e Flexibilidade**: O JavaScript é uma linguagem de tipagem dinâmica, o que significa que as variáveis podem armazenar diferentes tipos de dados e mudar esses tipos em tempo de execução. Isso traz uma grande flexibilidade na codificação, mas também exige uma

compreensão clara de conceitos como coercion (conversão automática de tipos) e truthiness (como valores são interpretados em contextos booleanos).
- **Funcionalidades Modernas**: Ao longo dos anos, o JavaScript incorporou conceitos de programação funcional e orientada a objetos, oferecendo recursos como closures, funções de alta ordem, promessas para programação assíncrona, classes, e módulos.

JavaScript não é apenas uma ferramenta para adicionar efeitos ou interatividade simples em sites; é uma **linguagem robusta, capaz de construir sistemas complexos e performáticos**. Seu papel no desenvolvimento web moderno é indiscutível, e seu domínio é essencial para qualquer desenvolvedor que deseja estar na vanguarda da tecnologia.

---

# 2- Variáveis

Variáveis são usadas para armazenar informações que serão referenciadas e manipuladas em um programa de computador. Elas também fornecem uma maneira de rotular dados com um nome descritivo, para que nossos programas possam ser entendidos de maneira mais clara e concisa.

## Declaração de Variáveis: var, let ou const?

No JavaScript, temos 3 jeitos de declarar variáveis, no começo é meio confuso, e no final fica mais confuso ainda. Porém vou explicar de uma forma que você vai entender fácil fácil:

A diferença delas basicamente está no **escopo**, basicamente escopo é em que lugar a variável está inserida e de onde ela pode ser acessada.

**`var`: O vovô das variáveis. Era tipo o único jeito de declarar uma variável antigamente. O lance é que ele é meio bagunçado, porque não se importa muito com onde você o declara. Pode criar umas paradas estranhas, como você declarar ela dentro de uma função e acessar fora. É tipo o cara que não respeita as regras de onde deve estar, por isso você não vai estar utilizando var durante o andamento desse curso.**

```
if (true) {
    var exemploVar = "Eu sou global!";
}
console.log(exemploVar);
// Funciona, porque var é global
```

**`let`: O "mano mais novo" do `var`. O `let` é mais organizado, sabe? Ele respeita os limites do bloco onde foi declarado. Se você declara dentro de um `if`, por exemplo, fora dele ninguém conhece. É tipo aquele amigo que só fica na dele, na sua área.**

```
if (true) {
  let exemploLet = "Só existo aqui dentro, beleza?";
}
console.log(exemploLet);
// Isso vai causar um Erro! O 'let' não deixa ver fora do bloco
```

**`const`: Const vem de constante, o mais firmeza dos três. `const` é tipo o `let`, mas ainda mais rígido. Uma vez que você dá um valor pra ele, não pode mudar mais. É ideal pra quando você tem valores que não quer que mudem nunca, tipo constantes mesmo.**

```
const exemploConst = "Sou firmeza, não mudo.";
exemploConst = "Vou tentar mudar"; // Erro! O 'const' não deixa mudar
```

**Basicamente, é isso aí: `var` é o avô, meio sem regra; `let` é o cara mais na dele, que fica no seu espaço; e `const` é o firmeza que não muda nunca.**

# Regras de Nomenclatura de Variáveis

**Assim como existe a lista de nomes proibidos para filhos no Brasil, no JavaScript temos várias regras para dar nome às nossas variáveis:**

**Precisa começar com Letra, Sublinhado (_), ou Cifrão ($):**

**Não dá pra colocar número no início, `let 1nome` não vai rolar, mas `let nome1` tá valendo.**

```
let _idade = 20;
let $dinheiro = "pouco";
let temDúvidas = "deixe um comentário";
```

**Exemplo que vai dar erro:**

```
let 1chance = "não rola, parceiro";
```

Zero Espaços, Zero Problemas: No mundo do JavaScript, espaço é inimigo. `let meu sonho` **vai dar ruim, mas** `let meuSonho` **tá tranquilo.**

```
let minhaTrajetoria = "mudou o jogo";
```

**Exemplo que vai dar erro:**

```
let minha trajetória = "vai dar bug";
```

Case Sensitive: `let nome` **e** `let Nome` **são dois caras diferentes. O JavaScript não mistura as coisas, cada um no seu quadrado.**

```
let vidaDeDev = "uma";
let VidaDeDev = "outra";
```

**Sem Palavras Reservadas: Tem umas palavras que são tipo área VIP no JavaScript, só pra ele.**

**Coisas como `if`, `break`, `let`... essas palavras são do sistema, não dá pra usar como nome de variável.**

```
let if = "não pode, mano";
let let = "isso aqui vai dar erro...";
```

## Tipos de Dados no JavaScript

**Apesar de ser uma linguagem dinamicamente tipada, o js possuí vários tipos de dados, vamos passar sobre cada um deles rapidamente, pois isso é um conceito que você aprenderá melhor na prática, principalmente sofrendo com bus:**

**String: Sabe aquele texto, frase, palavra? Isso é uma string. Tudo que fica entre aspas, pode ser simples ou duplas:**

```
let nome = "JavaScript";
let frase = 'mudou minha vida';
```

**Number: Aqui é o mundo dos números. Não importa se é inteiro, decimal, positivo ou negativo.**

```
let idade = 20;
let pi = 3.14;
```

Boolean: Só tem dois valores: `true` **ou** `false`**. É o tipo "sim ou não" do JavaScript.**

```
let amoCodar = true;
let odeioCodar = false;
```

Undefined: Tipo, quando você cria uma variável e não dá nenhum valor pra ela, ela fica `undefined`**, ou seja, indefinida.**

```
let vazio; console.log(vazio); // vai mostrar 'undefined'
```

Null: Parecido com `undefined`**, mas aqui é quando você quer deixar claro que a variável tá vazia, sem nada mesmo.**

```
let semValor = null;
```

**Object: Objeto é tipo uma coleção de dados. Pensa num pacote que tem várias coisas dentro, cada coisa com seu nome (vamos nos aprofundar em objetos mais para a frente).**

```
let pessoa = {
  nome: "Dev",
  idade: 20,
  linguagemFavorita: "JavaScript"
};
```

**Array: Array é uma lista de coisas, que podem ser números, strings, objetos, até outros arrays.**

```
let linguagens = ["JavaScript", "Python", "Ruby"];
```

**Symbol: É um tipo mais raro, usado para criar identificadores únicos. Não é tão comum para iniciantes, mas é bom saber que existe.**

```
let id = Symbol('id');
```

# Exercícios com variáveis

1. **Crie 3 variáveis contendo:**

- **Seu nome;**
- **Sua idade;**
- **Comida favorita;**

**Após declarar as variáveis e atribuir os valores, utilize o `console.log` para formar uma frase como:**

**“Meu nome é …, tenho …. anos e minha comida favorita é …..”**