

EXPLORANDO O MUNDO DO JAVASCRIPT



PARTICIPANTES

FABIO RIBEIRO-RA: N062290
GUSTAVO CAMPUS-RA: N050GD3
JUSTING VINICIUS-RA: R015440
KAUA KOKI-RA: G819675
PEDRO GORIN-RA: N087730
THIANES PIERRE-RA: N307EDO
PEDRO COSTA-RA: G7492H1

JOÃO VITOR RODRIGUES-RA: G72CHG5
PAULO VICTOR-RA: N0753DO
EDUARDO BORGES-RA: G804090
LUCAS SANTANA-RA: F3562l4
MAURO HENRIQUE-RA: R023893
CAMILA THOMAS-RA: G87ADC2
BEATRIZ OLIVEIRA-RA: N296429
GABRIEL DOS SANTOS-RA: G86DAB6



INTRODUÇÃO AO JAVASCRIPT

01

- JavaScript é uma linguagem de programação de alto nível e interpretada, usada principalmente para criar interatividade em páginas web.
- Criado em 1995 por Brendan Eich na Netscape, seu objetivo inicial era adicionar elementos dinâmicos aos sites.
- É uma das três principais tecnologias da web, junto com HTML e CSS.
- Atualmente, é uma linguagem versátil usada tanto no front-end (navegadores) quanto no back-end (servidores), além de aplicações móveis, desktop e até jogos.
- Popular por sua simplicidade e flexibilidade, JavaScript se tornou indispensável para desenvolvedores em diversas áreas.

JAVASCRIPT: HISTÓRIA

01

- Criação e Lançamento (1995):
- Foi criado por Brendan Eich, enquanto trabalhava na empresa Netscape Communications.
- Originalmente, o projeto se chamava Mocha, depois foi renomeado para LiveScript, e finalmente para JavaScript.
- Desenvolvido em apenas 10 dias, a linguagem foi inicialmente criada para adicionar interatividade aos websites.

02

- Competição e Padrões (1996-1999):
- Durante esse período, havia uma competição acirrada entre Netscape Navigator e Internet Explorer (da Microsoft). Isso fez com que JavaScript e outras tecnologias de web evoluíssem rapidamente.
- Em 1997, a linguagem foi padronizada pela ECMA International, levando ao surgimento do ECMAScript. JavaScript é uma implementação desse padrão.

03

- Crescimento e Modernização (2000-2010):
- O JavaScript passou por melhorias contínuas, especialmente com o lançamento do AJAX em 2005, permitindo a criação de aplicações web mais dinâmicas e responsivas.
- Surgimento de frameworks e bibliotecas como jQuery, que facilitavam o uso da linguagem.

04

- Era Moderna (2010-Presente):
- A criação do Node.js (2009) permitiu que JavaScript fosse usado para desenvolvimento back-end, expandindo suas capacidades além do navegador.
- Popularização de frameworks modernos como React, Vue e Angular, que revolucionaram o desenvolvimento front-end.
- Atualizações regulares do ECMAScript, garantindo que a linguagem continue evoluindo e adicionando novos recursos





FUNDAMENTOS DO JAVASCRIPT

01

Linguagem de Programação Interpretada:

- O código JavaScript é executado diretamente pelo navegador ou pelo ambiente Node.js, sem a necessidade de compilação prévia.


02

Tipagem Dinâmica:

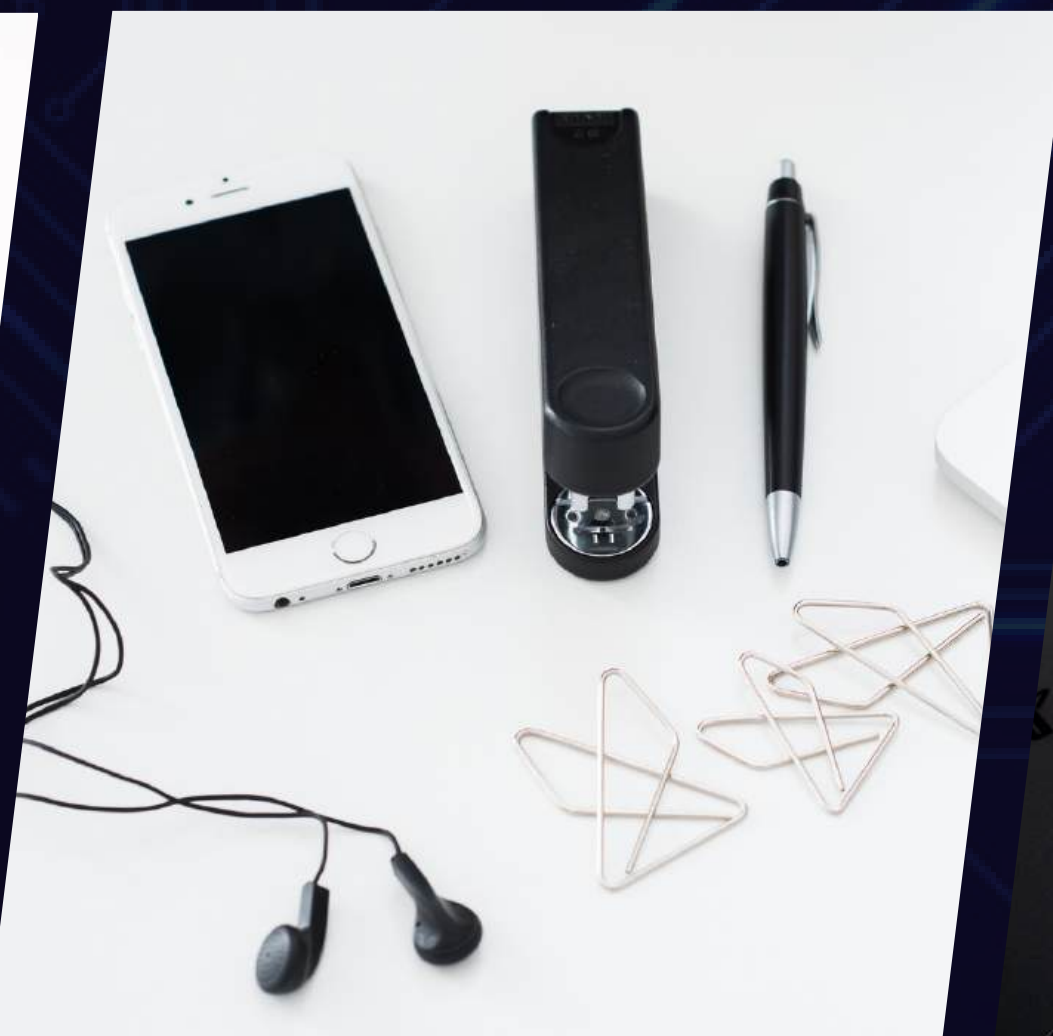
- A linguagem é dinamicamente tipada, o que significa que tipos de variáveis são determinados em tempo de execução, e não são fixos.

03

Orientada a Objetos, Funcional e Imperativa:

- Suporta vários paradigmas de programação, incluindo orientação a objetos, programação funcional e programação imperativa.
 - Usa prototipagem ao invés de classes tradicionais, uma característica única entre as linguagens de script.
- 

APLICAÇÕES DO JAVASCRIPT



01

Front-end Web:

- JavaScript permite adicionar interatividade a páginas web, como animações, validação de formulários e interação com APIs.
- Frameworks como React, Vue e Angular são amplamente utilizados para construir Single Page Applications (SPA), que oferecem uma experiência mais fluida aos usuários.

02

Back-end Web:

- Com Node.js, JavaScript passou a ser utilizado também no servidor, possibilitando o desenvolvimento de APIs, serviços web e aplicações completas.
- Ferramentas como Express.js ajudam a criar rapidamente servidores web robustos e eficientes.

03

Desenvolvimento Mobile e Desktop:

- React Native e Ionic permitem criar aplicativos móveis nativos para Android e iOS usando JavaScript.
- Electron é uma plataforma que permite criar aplicações desktop para Windows, macOS e Linux, utilizando tecnologias web como JavaScript, HTML e CSS.

04

Desenvolvimento de Jogos:

- JavaScript é usado para criar jogos para navegadores, utilizando bibliotecas como Phaser e frameworks como Babylon.js para jogos 3D.

05

Internet das Coisas (IoT):

- Com a ajuda de Node.js, JavaScript pode ser usado em dispositivos IoT para programar sistemas embarcados e dispositivos conectados.

IMPORTÂNCIA DO JAVASCRIPT

- Uma das linguagens mais populares do mundo:
- A cada ano, JavaScript continua no topo das pesquisas como a linguagem mais utilizada por desenvolvedores, devido à sua versatilidade e facilidade de aprender.
- Comunidade Ativa e em Crescimento:
- JavaScript possui uma das comunidades mais ativas e colaborativas, o que significa muitos recursos, tutoriais, frameworks e bibliotecas disponíveis.
- Essencial para a Web Moderna:
- Praticamente todos os sites modernos usam JavaScript, e ele é fundamental para criar experiências de usuário dinâmicas e responsivas.



SINTAXE BÁSICA E ESTRUTURA DE CÓDIGO EM JAVASCRIPT

Variáveis

Variáveis são utilizadas para armazenar dados que podem ser utilizados ao longo do código. Em JavaScript, você pode declarar variáveis usando var, let ou const. O var permite que a variável seja reatribuída e tem escopo global ou de função. O let também permite reatribuições, mas tem escopo de bloco, tornando-o mais seguro em termos de escopo. O const, por sua vez, é usado para declarar constantes, ou seja, valores que não podem ser alterados uma vez atribuídos.

```
var nome = "João";  
let idade = 25;  
const PI = 3.14;
```

TIPO DE DADOS

JavaScript suporta diversos tipos de dados, cada um com características específicas. Os mais comuns incluem:

String: Sequências de texto envolvidas em aspas, como "Olá, mundo!".

Number: Números que podem ser inteiros ou decimais, como 25 ou 3.14.

Boolean: Representa valores verdadeiros (true) ou falsos (false).

Array: Estruturas que podem armazenar múltiplos valores, como ["maçã", "banana"].

Object: Estruturas que armazenam dados na forma de pares de chave-valor, como { nome: "João", idade: 30 }.

```
//string  
let saudacao = "Olá, mundo!";
```

```
//number  
let idade = 30;
```

```
//boolean  
let isActive = true;
```

```
//array  
let frutas = ["maçã", "banana", "laranja"];
```

```
//object  
let pessoa = {  
  nome: "João",  
  idade: 30,  
  cidade: "São Paulo"  
};
```

OPERADORES

Os operadores em JavaScript são símbolos que realizam operações sobre variáveis e valores. Os operadores aritméticos são usados para realizar cálculos, como + para soma e - para subtração. Já os operadores de comparação são utilizados para comparar valores, retornando verdadeiro ou falso, como == para igualdade e != para desigualdade.

```
//Operadores Aritméticos:
```

```
let soma = 5 + 3; // 8
```

```
let subtracao = 10 - 4; // 6
```

```
let multiplicacao = 2 * 3; // 6
```

```
let divisao = 8 / 2; // 4
```

```
//Operadores de Comparação:
```

```
let igual = (5 == 5); // true
```

```
let diferente = (5 != 3); // true
```

```
let maior = (5 > 3); // true
```


ESTRUTURAS DE CONTROLE

As estruturas de controle permitem que você execute diferentes partes do código com base em condições. As condicionais (como if e else) permitem que você verifique se uma condição é verdadeira e execute um bloco de código se for o caso. Os loops (como for e while) permitem que você execute um bloco de código várias vezes, até que uma condição específica seja atendida.

```
//Condicionais:
let idade = 18;
if (idade >= 18) {
  console.log("Você é maior de idade.");
} else {
  console.log("Você é menor de idade.");
}

//Loops:
//for:
for (let i = 0; i < 5; i++) {
  console.log(i); // Imprime números de 0 a 4
}

//while
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

FUNÇÕES

Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizadas ao longo do programa. Elas podem receber parâmetros (valores de entrada) e retornar um resultado. Declarar uma função permite organizar o código em partes menores e mais gerenciáveis, facilitando a manutenção e a legibilidade.

```
function somar(a, b) {  
  return a + b;  
}  
let resultado = somar(5, 3); // resultado é 8
```

ESTRUTURA DE CÓDIGO

A estrutura básica de um arquivo JavaScript envolve a declaração de variáveis, a definição de funções e a chamada dessas funções. É comum iniciar com comentários para descrever o que o código faz, seguido pela declaração de variáveis e pela definição de funções. Por fim, as funções podem ser chamadas para executar o código desejado, permitindo que a lógica do programa seja executada de maneira ordenada.

```
const nome = "Maria"; // Declarar uma variável
let idade = 30;       // Outra variável

function apresentar() {
  console.log("Olá, meu nome é " + nome + " e tenho " + idade + " anos.");
}

apresentar(); // Chamar a função
```


OBJETOS E ORIENTAÇÃO A OBJETOS EM JAVASCRIPT

01

O que são Objetos?

Objetos em JavaScript são coleções de propriedades e métodos. Uma propriedade é uma associação entre uma chave (ou nome) e um valor, enquanto um método é uma função associada a um objeto. Objetos permitem que você agrupe dados e comportamentos relacionados.

```
let carro = {  
  marca: "Toyota",  
  modelo: "Corolla",  
  ano: 2020,  
  mostrarInfo: function() {  
    return `${this.marca} ${this.modelo} - ${this.ano}`;  
  }  
};  
  
// Acessando propriedades e métodos do objeto  
console.log(carro.mostrarInfo()); // Saída: "Toyota Corolla - 2020"
```

Criando Objetos com Construtores

Além de criar objetos diretamente, você pode usar funções construtoras para criar múltiplas instâncias de objetos semelhantes. Isso é útil para quando você precisa de vários objetos com a mesma estrutura.

```
function Pessoa(nome, idade) {  
  this.nome = nome;  
  this.idade = idade;  
  this.apresentar = function() {  
    return `Meu nome é ${this.nome} e eu tenho ${this.idade} anos.`;  
  };  
}  
  
// Criando instâncias do objeto Pessoa  
let pessoa1 = new Pessoa("João", 30);  
let pessoa2 = new Pessoa("Maria", 25);  
  
console.log(pessoa1.apresentar()); // Saída: "Meu nome é João e eu tenho 30 anos."  
console.log(pessoa2.apresentar()); // Saída: "Meu nome é Maria e eu tenho 25 anos."
```

03 Prototipagem

JavaScript utiliza protótipos para herança. Cada objeto pode ter um objeto protótipo a partir do qual herda propriedades e métodos. Isso permite que você compartilhe métodos entre várias instâncias de objetos.

```
function Animal(nome) {  
  this.nome = nome;  
}  
  
Animal.prototype.falar = function() {  
  return `${this.nome} faz um barulho.`;  
};  
  
let cachorro = new Animal("Rex");  
console.log(cachorro.falar()); // Saída: "Rex faz um barulho."
```


04

Orientação a Objetos

A orientação a objetos é um paradigma de programação que organiza o código em objetos que interagem entre si. Em JavaScript, você pode usar classes (introduzidas no ES6) para criar objetos de maneira mais intuitiva.

```
class Carro {  
  constructor(marca, modelo, ano) {  
    this.marca = marca;  
    this.modelo = modelo;  
    this.ano = ano;  
  }  
  
  mostrarInfo() {  
    return `${this.marca} ${this.modelo} - ${this.ano}`;  
  }  
}  
  
// Criando uma instância da classe Carro  
let meuCarro = new Carro("Honda", "Civic", 2021);  
console.log(meuCarro.mostrarInfo()); // Saída: "Honda Civic - 2021"
```

05

Herança

A herança permite que uma classe (subclasse) herde propriedades e métodos de outra classe (superclasse). Isso promove a reutilização do código e a criação de hierarquias de classes.

```
class Veiculo {
  constructor(marca, modelo) {
    this.marca = marca;
    this.modelo = modelo;
  }
}

class Carro extends Veiculo {
  constructor(marca, modelo, ano) {
    super(marca, modelo); // Chama o construtor da superclasse
    this.ano = ano;
  }

  mostrarInfo() {
    return `${this.marca} ${this.modelo} - ${this.ano}`;
  }
}

// Criando uma instância da classe Carro
let carro1 = new Carro("Ford", "Fiesta", 2019);
console.log(carro1.mostrarInfo()); // Saída: "Ford Fiesta - 2019"
```




CONCLUSÃO



JavaScript é uma linguagem essencial para o desenvolvimento web, permitindo a criação de aplicações interativas e dinâmicas. Nesta cartilha, exploramos seus conceitos fundamentais, como sintaxe básica, funções, programação funcional, objetos e orientação a objetos.

Sua versatilidade a torna ideal tanto para o front-end quanto para o back-end, sendo amplamente utilizada em frameworks e bibliotecas como React, Angular e Node.js. Ao dominar JavaScript, os desenvolvedores podem criar soluções inovadoras e eficientes, contribuindo para o avanço da tecnologia.





BIBLIOGRAFIA



Eloquent JavaScript: A Modern Introduction to Programming
Marijn Haverbeke
O'Reilly Media, 2018.
Disponível em: eloquentjavascript.net

JavaScript: The Good Parts
Douglas Crockford
O'Reilly Media, 2008.
ISBN: 978-0596805524

You Don't Know JS (book series)
Kyle Simpson
O'Reilly Media, 2014.
Disponível em: github.com/getify/You-Dont-Know-JS

MDN Web Docs - JavaScript
Mozilla Developer Network
Disponível em: developer.mozilla.org

JavaScript: The Definitive Guide
David Flanagan
O'Reilly Media, 2020.
ISBN: 978-1491952023

JavaScript.info
Ilya Kantor
Disponível em: javascript.info

