A Jornada do Dev Pirata: Mestre dos Scripts

Aprenda JavaScript de forma fácil e prática. Torne-se o mestre dos scripts!

RIQUELMI DOS SANTOS

01 Introdução ao JavaScript

O que é JavaScript?

A linguagem que dá vida à web

JavaScript é a linguagem de programação que dá vida às páginas da web. É responsável por tornar os sites interativos, dinâmicos e inteligentes. Além do navegador, hoje também roda no servidor com **Node.js**.

02 Sintaxe

Entendendo a Sintaxe do JS

O JavaScript lê seu código linha por linha, de cima pra baixo. Você escreve instruções que dizem ao computador o que fazer.

➡ Exemplo de comentário e instrução simples:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
// Isso é um comentário
console.log("Olá, mundo!");
```

02 Tipos Primitivos

Tipos de Dados

O que você pode guardar nas variáveis?

Number: Números
String: Textos
Boolean: Verdadeiro ou falso
Undefined: Variável sem valor
Null: Ausência intencional de valor
NaN: Não é um número

➡ Exemplo real:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let idade = 25; // Number
let nome = "Ana"; // String
let ativo = true; // Boolean
let endereco; // Undefined
let carro = null; // Null
let resultado = "abc" * 3; // NaN
```

04 Declaração de Variáveis

Como Declarar Variáveis Corretamente

- **var — Antigo e problemático**
  - Tem escopo global ou de função.
  - Permite redeclarar e reatribuir.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
var nome = "João";
var nome = "Maria"; // Funciona, mas pode causar confusão
```

Como Declarar Variáveis Corretamente

- **let — O mais usado atualmente**
  - Tem escopo de bloco.
  - Permite reatribuir, mas não redeclarar.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let idade = 30;
idade = 31; // Ok
// let idade = 32; ❌ Não pode no mesmo escopo
```

Como Declarar Variáveis Corretamente

- **const — Para valores que não mudam**
  - Escopo de bloco.
  - Não permite redeclaração nem reatribuição.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
const PI = 3.14159;
// PI = 3.14; ❌ Erro
```

06 Hoisting

Hoisting

Variáveis "Içadas" pro Topo

O JavaScript lê primeiro as declarações e depois as execuções.

➡ Com var:

```
Jornada do Dev Pirata - Riquelmi Dos Santos.js

console.log(sobrenome); // undefined
var sobrenome = "Silva";
```

var sofre hoisting e vira undefined.

➡ Com let ou const:

```
Jornada do Dev Pirata - Riquelmi Dos Santos.js

console.log(nome); // ReferenceError
let nome = "Ana";
```

let e const também são içados, mas ficam em uma "zona morta" (TDZ) até a linha de declaração.

06 Operadores

Operadores

Operadores Aritméticos — Fazendo Contas

Servem para fazer cálculos:

- + soma
- - subtrai
- * multiplica
- / divide
- % pega o resto da divisão
- ** faz potência

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let total = 10 + 5; // 15
```

Operadores

Operadores de Comparação

Comparam valores e retornam true ou false:

- == igual (valor),
- === igual (valor e tipo),
- != diferente,
- !== diferente (valor e tipo),
- >, <, >=, <=.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let idade = 18;
console.log(idade ≥ 18); // true
```

Operadores

Operadores Lógicos

Juntam condições:

- && (e) — tudo precisa ser verdadeiro.
- || (ou) — uma condição verdadeira já basta.
- ! (não) — inverte o valor.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
if (temCarteira && temIdade) {
  console.log("Pode dirigir");
}
```

Operadores

Operador Ternário

Um if/else resumido:

- condição ? se verdadeiro : se falso

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let status = idade ≥ 18 ? "Maior" : "Menor";
```

07 Estrutura Condicional

Condições

Fazendo o Código Decidir

As condições permitem que o código execute algo se uma situação for verdadeira.

➢ **if / else**

- Se a condição for verdadeira, executa o bloco **if**. Senão, executa o **else**.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let idade = 18;

if (idade ≥ 18) {
  console.log("Maior de idade");
} else {
  console.log("Menor de idade");
}
```

Condições

Fazendo o Código Decidir

- **else if**
  - Usado para testar várias condições.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let nota = 7;

if (nota ⩾ 7) {
  console.log("Aprovado");
} else if (nota ⩾ 5) {
  console.log("Recuperação");
} else {
  console.log("Reprovado");
}
```

Condições

Fazendo o Código Decidir

- **Switch**
  - Útil quando há muitas opções.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let cor = "vermelho";

switch (cor) {
  case "vermelho":
    console.log("Pare");
    break;
  case "verde":
    console.log("Siga");
    break;
  default:
    console.log("Cor inválida");
}
```

Tratamento de Erros

Deixando o Código Seguro

Evita que o programa pare ao encontrar um erro.

➡ Exemplo com try...catch:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
try {
  let resultado = 10 / 0;
  console.log(resultado);
} catch (erro) {
  console.log("Ocorreu um erro:", erro);
} finally {
  console.log("Finalizado");
}
```

- **try**: Tenta executar.
- **catch**: Captura se der erro.
- **finally**: Executa sempre (com ou sem erro).

08 Estruturas de Repetição

Laços de Repetição

Fazendo Algo Várias Vezes

- **for**
  - Executa enquanto uma condição for verdadeira.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
for (let i = 1; i ⩽ 5; i++) {
  console.log(i);
}
```

Laços de Repetição

Fazendo Algo Várias Vezes

- **While**
  - Repete enquanto a condição for verdadeira.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let i = 1;
while (i ≤ 5) {
  console.log(i);
  i++;
}
```

Laços de Repetição

Fazendo Algo Várias Vezes

- **do...while**
  - Executa pelo menos uma vez, depois testa a condição.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let i = 1;
do {
  console.log(i);
  i++;
} while (i ≤ 5);
```

06 Funções

Funções

Reutilizando Código

Funções são blocos de código que executam uma tarefa e podem ser reutilizados.

- Declarando uma função:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
function saudacao() {
  console.log("Olá!");
}

saudacao(); // Executa a função
```

Funções

Reutilizando Código

- Com parâmetros:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
function soma(a, b) {
  console.log(a + b);
}

soma(5, 3); // Resultado: 8
```

- Retornando valores:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
function multiplicar(a, b) {
  return a * b;
}

let resultado = multiplicar(4, 5);
console.log(resultado); // 20
```

Escopo e Closures

Protegendo Dados

Escopo:

Define onde a variável pode ser acessada.

- Global: Fora de qualquer função ou bloco.
- Função: Só acessível dentro da função.
- Bloco: Usando let ou const dentro de {}.

➡ Exemplo de escopo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
function exemplo() {
  let nome = "Ana";
  console.log(nome);
}
exemplo();
// console.log(nome); // Erro, está fora do escopo
```

Escopo e Closures

Protegendo Dados

Closures:

Uma função que lembra seu escopo mesmo fora dele, protegendo dados.

- Usamos closures para manter valores privados.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
function contador() {
  let numero = 0;
  return function() {
    numero++;
    console.log(numero);
  }
}

const incrementar = contador();
incrementar(); // 1
incrementar(); // 2
```

10 Arrays

Arrays

Guardando Vários Dados

Arrays armazenam vários valores em uma única variável, como uma lista.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let frutas = ["Maçã", "Banana", "Laranja"];
console.log(frutas[0]); // Maçã
```

Métodos de Arrays

Trabalhando com Listas

Os arrays têm métodos prontos que ajudam a manipulá-los.

- push() — Adiciona no final
- pop() — Remove o último
- shift() — Remove o primeiro
- unshift() — Adiciona no início
- length — Mostra quantos itens tem

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let frutas = ["Maçã", "Banana"];
frutas.push("Laranja");
console.log(frutas); // ["Maçã", "Banana", "Laranja"]
```

11 Objetos

Objetos

Dados Completos e Organizados

Objetos guardam dados em formato chave e valor.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let pessoa = {
  nome: "Ana",
  idade: 25,
  cidade: "São Paulo"
};

console.log(pessoa.nome); // Ana
```

Acessando Dados de Objetos

Dados Completos e Organizados

Acessamos os dados de duas formas:

- Ponto: pessoa.nome
- Colchete: pessoa["nome"]

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
console.log(pessoa.cidade); // São Paulo
```

Métodos em Objetos

Funções Dentro de Objetos

Objetos podem ter funções dentro deles, chamadas de métodos.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let pessoa = {
  nome: "Ana",
  falar() {
    console.log("Olá, eu sou " + this.nome);
  }
};

pessoa.falar(); // Olá, eu sou Ana
```

12 Loop em Arrays e Objetos

Loop em Arrays e Objetos

Percorrendo Dados

for...of

- Para Arrays.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let frutas = ["Maçã", "Banana"];
for (let fruta of frutas) {
  console.log(fruta);
}
```

Loop em Arrays e Objetos

Percorrendo Dados

for...in

- Para Objetos.

➡ Exemplo:

Jornada do Dev Pirata - Riquelmi Dos Santos.js

```
let pessoa = { nome: "Ana", idade: 25 };
for (let chave in pessoa) {
  console.log(chave + ": " + pessoa[chave]);
}
```

12 Agradecimentos

OBRIGADO POR LER ATÉ AQUI

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github

.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.

https://github.com/RiquelmiDev/prompts-recipe-to-create-a-ebook