



DE VOLTA PARA OS FUNDAMENTOS



BRUNO GARCIA

01

INTRODUÇÃO

Capítulo 1: Variáveis

Declaração de variáveis

Em JavaScript, você pode declarar variáveis usando três palavras-chave: *var*, *let*, e *const*.

var: Usada para declarar variáveis com escopo global ou de função.

let: Usada para declarar variáveis com escopo de bloco.

const: Usada para declarar constantes (não pode ser reatribuída).



variaveis.js

```
var nome = 'João';  
let idade = 30;  
const pais = 'Brasil';
```

Capítulo 1: Tipos de Dados

Tipos de Dados

JavaScript possui tipos de dados primitivos como:

String: Sequência de caracteres.

Number: Números inteiros ou de ponto flutuante.

Boolean: Valor lógico (verdadeiro ou falso).

Null: Ausência de valor.

Undefined: Valor não definido.

Symbol: Valor único e imutável.

Object: Estruturas de dados compostas.



```
tiposDados.js

1 let nome = 'Carlos';    // String
2 let idade = 25;         // Number
3 let isAdult = true;     // Boolean
4 let valorInexistente = null; // Null
5 let algoIndefinido;     // Undefined
6
```

02

Estruturas de Controle

Capítulo 2: Estruturas de Controle

Condicionais

As estruturas condicionais ajudam a tomar decisões com base em condições.

if: Verifica uma condição e executa um bloco de código se a condição for verdadeira.

else: Define o que fazer se a condição em if for falsa.

else if: Verifica uma condição alternativa.



```
1 let idade = 18;
2 if (idade ≥ 18) {
3     console.log('Maior de idade');
4 } else {
5     console.log('Menor de idade');
6 }
7
```

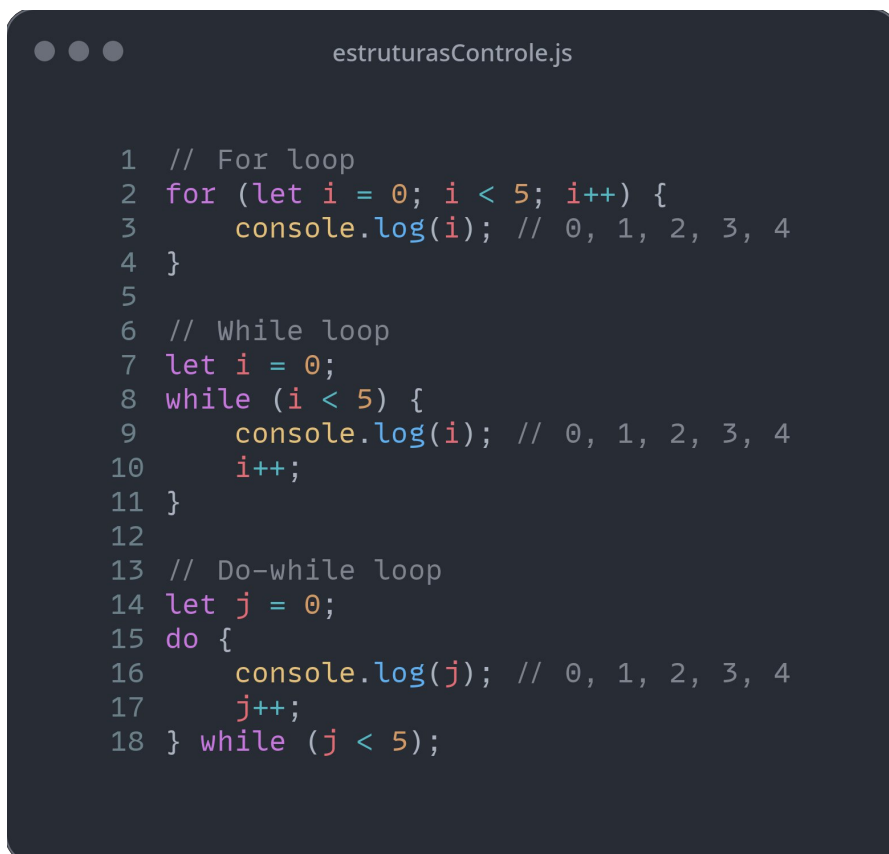
Capítulo 2: Laços de Repetição

Os laços são usados para repetir um bloco de código várias vezes.

for: Para executar um bloco de código um número específico de vezes.

while: Executa o código enquanto a condição for verdadeira.

do...while: Executa o código pelo menos uma vez, verificando a condição depois.

A screenshot of a code editor window titled 'estruturasControle.js'. The editor contains three JavaScript loop examples, each with line numbers 1 through 18. The first example is a 'for' loop (lines 1-4) that logs values 0 to 4. The second is a 'while' loop (lines 6-11) that also logs values 0 to 4. The third is a 'do...while' loop (lines 13-18) that logs values 0 to 4. The code is syntax-highlighted with colors: blue for keywords, green for variables, and orange for string literals.

```
1 // For loop
2 for (let i = 0; i < 5; i++) {
3     console.log(i); // 0, 1, 2, 3, 4
4 }
5
6 // While loop
7 let i = 0;
8 while (i < 5) {
9     console.log(i); // 0, 1, 2, 3, 4
10    i++;
11 }
12
13 // Do-while loop
14 let j = 0;
15 do {
16     console.log(j); // 0, 1, 2, 3, 4
17     j++;
18 } while (j < 5);
```

03

Funções

Capítulo 3: Funções

Declaração de Funções

Funções permitem agrupar um conjunto de instruções para serem executadas quando chamadas.

Exemplos:

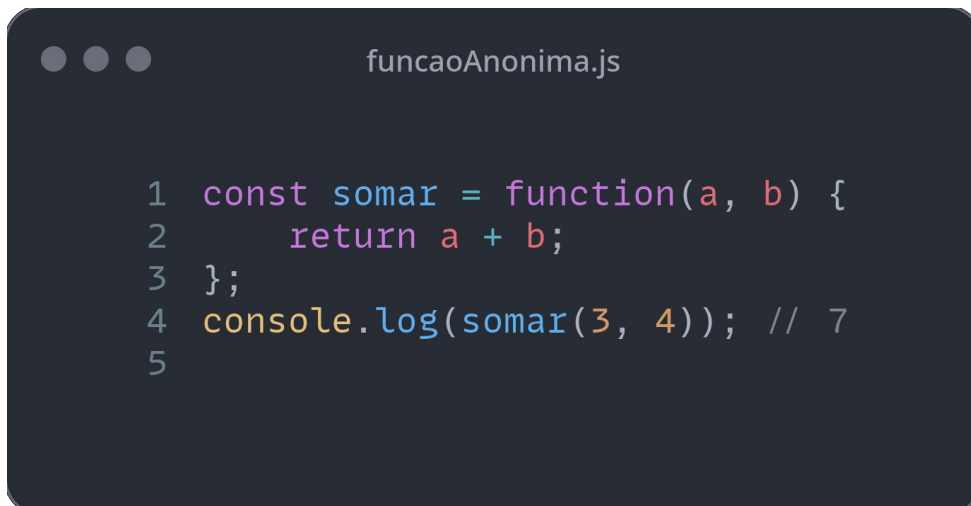


```
1 function saudacao(nome) {  
2     return 'Olá, ' + nome;  
3 }  
4  
5 console.log(saudacao('Maria')); // Olá, Maria  
6
```

Capítulo 3: Funções

Funções Anônimas

Funções que não têm nome, muitas vezes usadas em callbacks.



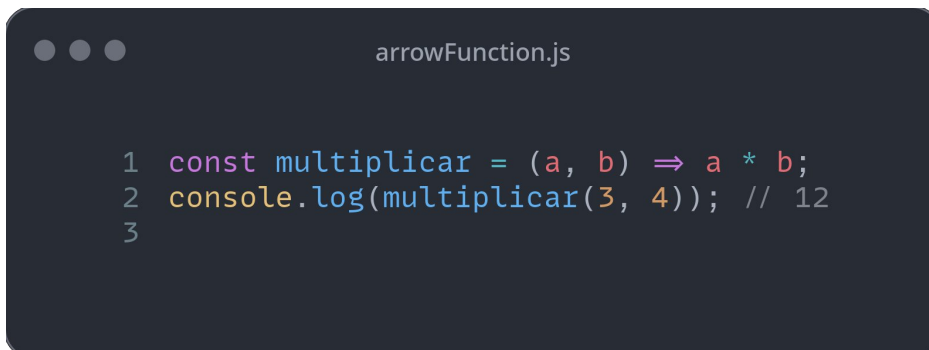
```
1  const somar = function(a, b) {  
2      return a + b;  
3  };  
4  console.log(somar(3, 4)); // 7  
5
```

Capítulo 3: Funções

Arrow Functions

Funções de flecha oferecem uma sintaxe mais curta para declarar funções.

Exemplos:



```
1 const multiplicar = (a, b) => a * b;
2 console.log(multiplicar(3, 4)); // 12
3
```

04

Arrays

Capítulo 4: Arrays

Criação de Arrays

Arrays são coleções de dados. Eles podem armazenar qualquer tipo de valor.

Exemplos:



```
arrays.js

1 let numeros = [1, 2, 3, 4, 5];
2 let frutas = ['maçã', 'banana', 'laranja'];
3
```

Capítulo 4: Arrays

Métodos de Arrays

Alguns métodos úteis de arrays incluem:

push(): Adiciona um item ao final do array.

pop(): Remove o último item do array.

shift(): Remove o primeiro item do array.

unshift(): Adiciona um item ao início do array.

Exemplos:

A screenshot of a code editor window titled 'arrays.js'. The code is as follows:

```
1 let frutas = ['maçã', 'banana', 'laranja'];
2 frutas.push('manga');
3 console.log(frutas); // ['maçã', 'banana', 'laranja', 'manga']
4
5 frutas.pop();
6 console.log(frutas); // ['maçã', 'banana', 'laranja']
7
```

05

Objetos

Capítulo 5: Objetos

Criando Objetos

Objetos são coleções de propriedades e métodos. Cada propriedade é composta por uma chave e um valor.

Exemplos:



```
1 let pessoa = {
2     nome: 'Ana',
3     idade: 28,
4     profissao: 'Desenvolvedora',
5     saudacao: function() {
6         return 'Olá, ' + this.nome;
7     }
8 };
9
10 console.log(pessoa.nome); // Ana
11 console.log(pessoa.saudacao()); // Olá, Ana
12
```


06

Manipulação de Strings

Capítulo 6: Manipulação de Strings

Métodos Comuns de Strings

length: Retorna o comprimento da string.

toUpperCase(): Converte a string para maiúsculas.

toLowerCase(): Converte a string para minúsculas.

substring(): Extrai uma parte da string.

Exemplos:

```
1 let texto = 'Hello, World!';
2
3 console.log(texto.length); // 13
4 console.log(texto.toUpperCase()); // HELLO, WORLD!
5 console.log(texto.toLowerCase()); // hello, world!
6 console.log(texto.substring(0, 5)); // Hello
```

07

Manipulação de Objetos

Capítulo 7: Manipulação de Objetos

Acessando Propriedades:

Você pode acessar as propriedades de um objeto de duas maneiras:

- ***Notação de ponto***: obj.propriedade
- ***Notação de colchetes***: obj['propriedade']

Exemplos:

```
1 javascript
2 let carro = {
3     modelo: 'Fusca',
4     cor: 'azul',
5     ano: 1969
6 };
7
8 console.log(carro.modelo); // Fusca
9 console.log(carro['cor']); // azul
```

08

Promises e Async/Await

Capítulo 8: Promises

Trabalhando com Promise:

Promises são usadas para lidar com operações assíncronas.

Exemplos:

promisses.js

```
let promessa = new Promise((resolve, reject) => {  
  let sucesso = true;  
  if (sucesso) {  
    resolve("Operação bem-sucedida!");  
  } else {  
    reject("Falha na operação.");  
  }  
});
```

```
promessa  
  .then((resultado) => {  
    console.log(resultado); // Operação bem-sucedida!  
  })  
  .catch((erro) => {  
    console.log(erro); // Falha na operação.  
  });
```

Capítulo 8: Async/Await

Usando Async/Await

`**async**` permite que você escreva código assíncrono de maneira mais legível, e `**await**` é usado para esperar que a promise seja resolvida.

Exemplos:

asyncAwait.js

```
async function minhaFuncao() {  
  let resultado = await promessa;  
  console.log(resultado);  
}
```

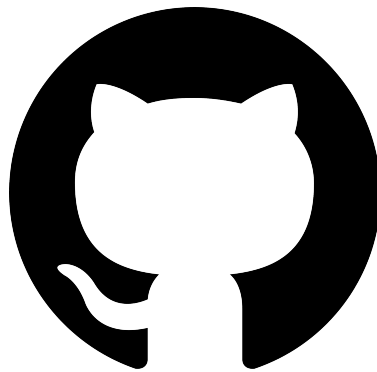
```
minhaFuncao();
```

Agradecimientos

Agradecimentos

Gostaria de expressar minha sincera gratidão ao tutor, Felipe Aguiar, pelo apoio, dedicação e orientação durante o processo de aprendizado.

Agradeço também à DIO, uma empresa fintech educacional de tecnologia, por proporcionar uma plataforma tão rica de aprendizado e por abrir portas para novas oportunidades. A DIO tem sido uma grande fonte de conhecimento e inspiração, e sou extremamente grato por ter acesso a um conteúdo de qualidade.



<https://github.com/brunog-infosec/prompts-recipe-to-create-a-ebook>