

## Conteúdo do dia!

Site: [Geração Tech](#)  
Curso: Formação em Desenvolvedor Web - Online  
Livro: Conteúdo do dia!

Impresso por: JOÃO VITOR DE MELO FREITAS  
Data: quinta-feira, 18 jul. 2024, 22:36

# Índice

## 1. JavaScript

### 1.1. Arrays

### 1.2. Estruturas de Repetição

### 1.3. Exercício Prático de Estruturas de Repetição em JavaScript

# 1. JavaScript

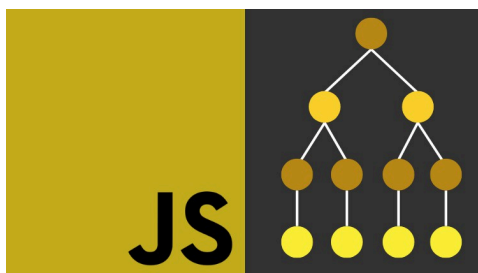
## Estrutura de Dados: Arrays em JavaScript

### Introdução

As estruturas de repetição são fundamentais em programação, permitindo que blocos de código sejam executados várias vezes de acordo com uma condição especificada.

Em JavaScript, as principais estruturas de repetição são `for`, `while` e `do/while`.

Elas são usadas para iterar sobre arrays, executar tarefas repetidas e simplificar o código que necessita de repetição.



## 1.1. Arrays


### Introdução

Olá, desenvolvedores! Continuando nosso tema sobre JavaScript, hoje vamos falar sobre mais uma estrutura de dados. Nos primeiros vídeos, discutimos três estruturas básicas: `String`, `Number` e `Boolean`, que representam texto, números e valores verdadeiros ou falsos, respectivamente. Mas existem outras estruturas de dados importantes que precisamos conhecer. Hoje, vamos aprender sobre `Arrays`.

### Problema com Múltiplas Variáveis

Imagine que você precise guardar vários nomes em variáveis. Inicialmente, você poderia fazer algo assim:

javascript

 Copiar código

```
let nome1 = "Gleudson"; let nome2 = "Márcio"; let nome3 = "Alessandro"; let nome4 = "Glauber";
```

Embora isso funcione, não é uma solução eficiente, especialmente se você tiver muitos nomes para armazenar. Manter várias variáveis torna o código mais complexo e difícil de gerenciar.


### Solução com Arrays

Um `Array` é uma estrutura de dados que permite armazenar múltiplos valores em uma única variável. Vamos ver como criar e usar `Arrays` em JavaScript.

### Criando um Array

Podemos criar um `Array` vazio assim:

javascript

 Copiar código

```
let nomes = [];
```

Ou um `Array` com valores:

javascript


 Copiar código

```
let nomes = ["Gleudson", "Márcio", "Alessandro", "Glauber"];
```

### Acessando Elementos de um Array

Cada elemento em um `Array` tem um índice, que começa em 0. Para acessar os elementos, usamos o nome do `Array` seguido do índice entre colchetes.

javascript


 Copiar código

```
console.log(nomes[0]); // Gleudson console.log(nomes[2]); // Alessandro
```

### Adicionando Elementos ao Array

Podemos adicionar elementos a um `Array` de várias maneiras. Uma delas é usando o método `push`.

javascript

 Copiar código

```
nomes.push("João"); console.log(nomes); // ["Gleudson", "Márcio", "Alessandro", "Glauber", "João"]
```

## Modificando Elementos do Array

Para modificar um elemento do **Array**, basta acessar o índice e atribuir um novo valor.

javascript


 Copiar código

```
nomes[1] = "Marcos"; console.log(nomes); // ["Gleidson", "Marcos", "Alessandro", "Glauber", "João"]
```

## Removendo Elementos do Array

Para remover elementos, podemos usar o método **splice**.

javascript

 Copiar código

```
nomes.splice(2, 1); // Remove o terceiro elemento (índice 2) console.log(nomes); // ["Gleidson", "Marcos", "Glauber", "João"]
```

## Arrays e Índices

Os índices são essenciais para acessar e manipular elementos em um **Array**. Cada elemento possui uma posição específica, começando do 0 até o tamanho do **Array** menos um.

## Exemplo Prático

Vamos criar um **Array** de nomes e acessá-los pelo índice:

javascript

 Copiar código

```
let nomes = ["Gleidson", "Márcio", "Alessandro", "Glauber"]; console.log(nomes[0]); // Gleidson console.log(nomes[2]); // Alessandro nomes.push("João"); console.log(nomes); // ["Gleidson", "Márcio", "Alessandro", "Glauber", "João"] nomes[1] = "Marcos"; console.log(nomes); // ["Gleidson", "Marcos", "Alessandro", "Glauber", "João"] nomes.splice(2, 1); console.log(nomes); // ["Gleidson", "Marcos", "Glauber", "João"]
```

dia 13 javascript vd 1



## Conclusão

Os **Arrays** são uma poderosa estrutura de dados em JavaScript que permitem armazenar e manipular múltiplos valores de maneira eficiente. Com **Arrays**, seu código se torna mais organizado e fácil de manter. Use-os sempre que precisar lidar com listas de dados.

Espero que essa aula tenha ajudado a entender como utilizar **Arrays** em seus códigos JavaScript. Vejo vocês no próximo encontro para continuar nosso aprendizado!

## 1.2. Estruturas de Repetição

### Introdução

Continuando nosso aprendizado sobre JavaScript, hoje vamos explorar as estruturas de repetição. Até agora, vimos as estruturas condicionais `if`, `else if`, `else`, `switch case` e o operador ternário. As estruturas de repetição permitem executar um bloco de código várias vezes, o que é extremamente útil quando precisamos realizar tarefas repetitivas.

### Estruturas de Repetição

#### for

A estrutura `for` é uma das mais comuns em JavaScript. Ela é usada quando sabemos com antecedência quantas vezes queremos repetir um bloco de código.

#### Sintaxe do `for`

javascript

 Copiar código

```
for (inicialização; condição; incremento) { // Código a ser executado }
```


- **inicialização:** Define a variável de controle do laço e seu valor inicial.
- **condição:** Testa a condição para continuar executando o laço.
- **incremento:** Atualiza a variável de controle a cada iteração.

### Exemplo Prático

Vamos criar um array de números e incrementar cada valor em 1.

#### Usando Estrutura `for`

javascript

 Copiar código

```
let numeros = [1, 2, 3, 4, 5]; for (let i = 0; i < numeros.length; i++) { numeros[i] += 1; } console.log(numeros); // [2, 3, 4, 5, 6]
```

Neste exemplo:

1. Inicializamos `i` com 0.
2. A condição `i < numeros.length` garante que o laço será executado enquanto `i` for menor que o comprimento do array.
3. O `i++` incrementa `i` em 1 a cada iteração.

### Cuidados com Laços Infinitos

É importante garantir que o laço tenha uma condição de parada para evitar laços infinitos, que podem travar o navegador ou a aplicação.

javascript

 Copiar código

```
for (let i = 0; i < numeros.length; i++) { console.log(i); // Certifique-se de que a condição de parada será alcançada }
```

### Usando `for` com Arrays

Podemos usar a estrutura `for` para iterar sobre arrays e modificar seus elementos de maneira eficiente.

javascript

 Copiar código


```
let numeros = [1, 2, 3, 4, 5]; for (let i = 0; i < numeros.length; i++) { numeros[i] += 1; } console.log(numeros); // [2, 3,
```

4, 5, 6]

## Exemplo Completo

Vamos ver um exemplo completo onde utilizamos um `for` para percorrer um array e incrementar cada valor em 2.

javascript

 Copiar código

```
let numeros = [1, 2, 3, 4, 5]; for (let i = 0; i < numeros.length; i++) { numeros[i] += 2; } console.log(numeros); // [3, 4, 5, 6, 7]
```

## Estruturas de Repetição Adicionais

Além do `for`, existem outras estruturas de repetição que você deve conhecer:

### `while`

O laço `while` executa um bloco de código enquanto a condição especificada for verdadeira.

#### Sintaxe do `while`

javascript

 Copiar código

```
while (condição) { // Código a ser executado }
```

#### Exemplo de `while`

javascript

 Copiar código

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

### `do...while`

O laço `do...while` é semelhante ao `while`, mas garante que o bloco de código será executado pelo menos uma vez.

#### Sintaxe do `do...while`

javascript

 Copiar código

```
do { // Código a ser executado } while (condição);
```

#### Exemplo de `do...while`

javascript

 Copiar código

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

dia 13 javascript vd 2



## Conclusão

As estruturas de repetição são fundamentais para automatizar tarefas repetitivas em JavaScript. A estrutura `for` é uma das mais usadas e versáteis, permitindo iterar sobre arrays e executar blocos de código múltiplas vezes de forma eficiente. Compreender e utilizar corretamente essas estruturas é essencial para desenvolver programas robustos e eficientes.

Espero que essa aula tenha ajudado a entender como utilizar estruturas de repetição em seus códigos JavaScript. Vejo vocês no próximo encontro para continuar nosso aprendizado!



## 1.3. Exercício Prático de Estruturas de Repetição em JavaScript

### Introdução

Olá, pessoal! Continuando nosso aprendizado em JavaScript, vamos colocar em prática tudo o que aprendemos sobre estruturas condicionais, de repetição e arrays. Nosso objetivo hoje é criar um formulário que permita selecionar uma data (dia, mês e ano) dinamicamente utilizando JavaScript.

### Estrutura HTML Básica

Vamos começar criando uma estrutura HTML básica para o nosso formulário:

html

 Copiar código

```
<!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <title>Formulário de Data</title> <script src="script.js"
defer></script> </head> <body> <form> <label for="dia">Dia:</label> <select id="dia"></select> <label for="mes">Mês:</label>
<select id="mes"></select> <label for="ano">Ano:</label> <select id="ano"></select> </form> </body> </html>
```

### Preenchendo os Campos com JavaScript

Vamos preencher os campos `select` de dia, mês e ano usando JavaScript.

#### Preenchendo os Dias

Vamos criar um laço de repetição para preencher os dias do mês:

javascript

 Copiar código

```
let diaSelect = document.getElementById('dia'); for (let dia = 1; dia <= 31; dia++) { let option =
document.createElement('option'); option.value = dia; option.textContent = dia; diaSelect.appendChild(option); }
```

#### Preenchendo os Meses

Para os meses, vamos usar um array para armazenar os nomes dos meses:

javascript

 Copiar código

```
let mesSelect = document.getElementById('mes'); let meses = ["Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho",
"Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"]; for (let i = 0; i < meses.length; i++) { let option =
document.createElement('option'); option.value = i + 1; // Os meses começam em 1 option.textContent = meses[i];
mesSelect.appendChild(option); }
```

#### Preenchendo os Anos

Vamos preencher os anos partindo do ano atual até 50 anos atrás:

javascript

 Copiar código

```
let anoSelect = document.getElementById('ano'); let anoAtual = new Date().getFullYear(); for (let ano = anoAtual; ano >=
anoAtual - 50; ano--) { let option = document.createElement('option'); option.value = ano; option.textContent = ano;
anoSelect.appendChild(option); }
```

### Código JavaScript Completo

Aqui está o código JavaScript completo para preencher os campos `select` de dia, mês e ano:

javascript



Copiar código

```
document.addEventListener('DOMContentLoaded', () => { // Preencher dias let diaSelect = document.getElementById('dia'); for (let dia = 1; dia <= 31; dia++) { let option = document.createElement('option'); option.value = dia; option.textContent = dia; diaSelect.appendChild(option); } // Preencher meses let mesSelect = document.getElementById('mes'); let meses = ["Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"]; for (let i = 0; i < meses.length; i++) { let option = document.createElement('option'); option.value = i + 1; // Os meses começam em 1 option.textContent = meses[i]; mesSelect.appendChild(option); } // Preencher anos let anoSelect = document.getElementById('ano'); let anoAtual = new Date().getFullYear(); for (let ano = anoAtual; ano >= anoAtual - 50; ano--) { let option = document.createElement('option'); option.value = ano; option.textContent = ano; anoSelect.appendChild(option); } });
```

dia 13 javascript vd 3



## Conclusão

Neste exercício, utilizamos estruturas de repetição para preencher dinamicamente os campos `select` de um formulário HTML com dias, meses e anos. Este tipo de interação entre HTML e JavaScript é fundamental para criar aplicações web dinâmicas e responsivas. Pratiquem bastante e explorem novas possibilidades!

Vejo vocês no próximo encontro para continuar nosso aprendizado! Valeu, pessoal!