Aula 4: Arrays e Objetos em JavaScript

Prof. Karan Luciano

Desenvolvimento Web - Ensino Médio

14 de outubro de 2024

Sumário

O que são Arrays?

- Estruturas que armazenam uma coleção de dados ordenados.
- Cada elemento tem um índice, começando do zero.
- Podem armazenar qualquer tipo de dado: números, strings, objetos, etc.

Criando um Array

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

Acessando Elementos

```
1 console.log(frutas[0]); // Saida: 'maca'
2 console.log(numeros[2]); // Saida: 3
3
```

Métodos de Arrays

- `push()`: Adiciona um elemento ao final.
- `pop()`: Remove o último elemento.
- `shift()`: Remove o primeiro elemento.
- `unshift()`: Adiciona um elemento no início.

Exemplos de Manipulação

```
1  const lista = [];
2
3  // Adicionando elementos
4  lista.push('Item 1');
5  lista.push('Item 2');
6
7  // Removendo o ultimo elemento
8  lista.pop(); // 'Item 2' removido
9
10 // Adicionando no inicio
11 lista.unshift('Item 0');
12
13 // Removendo o primeiro elemento
14 lista.shift(); // 'Item 0' removido
15
```

O que são Objetos?

- Coleções de pares chave-valor.
- Representam entidades com propriedades e métodos.
- São fundamentais em JavaScript para modelar dados complexos.

Criando um Objeto

```
1 const pessoa = {
2     nome: 'Karan',
3     idade: 18,
4     profissao: 'Cientista da computacao'
5 };
6
```

Acessando Propriedades

```
1 // Notacao de ponto
2 console.log(pessoa.nome); // Saida: 'Karan'
3
4 // Notacao de colchetes
5 console.log(pessoa['idade']); // Saida: 18
6
```

Adicionando e Modificando Propriedades

```
1 // Adicionando uma nova propriedade
2 pessoa.altura = 1.85;
3
4 // Modificando uma propriedade existente
5 pessoa.idade = 18;
6
```

Métodos em Objetos

```
1  const calculadora = {
2      soma: function(a, b) {
3          return a + b;
4      },
5      subtrai(a, b) {
6          return a - b;
7      }
8  };
9
10 console.log(calculadora.soma(2, 3)); // Saida: 5
11 console.log(calculadora.subtrai(5, 2)); // Saida: 3
12
```

Loops em JavaScript

- Permitem executar um bloco de código múltiplas vezes.
- Principais tipos: `for`, `while`, `do...while`.
- Usados para percorrer arrays, objetos e executar tarefas repetitivas.

Loop for

```
1 for (let i = 0; i < 5; i++) {
2     console.log('Iteracao numero ' + i);
3 }
4
```

- Saída: Números de 0 a 4.

Percorrendo Arrays com `for`

```
1 const cores = ['vermelho', 'verde', 'azul'];
2
3 for (let i = 0; i < cores.length; i++) {
4     console.log(cores[i]);
5 }
6
```

- Saída: 'vermelho', 'verde', 'azul'

Loop while

```
1 let contador = 0;
2
3 while (contador < 3) {
4     console.log('Contador eh ' + contador);
5     contador++;
6 }
7
```

- Saída: 'Contador eh 0', 'Contador eh 1', 'Contador eh 2'

Loop do...while

```
1 let numero = 5;
2
3 do {
4     console.log('Numero eh ' + numero);
5     numero--;
6 } while (numero > 0);
7
```

- Saída: Números de 5 a 1.

Percorrendo Objetos com `for...in`

```
1  const aluno = {
2      nome: 'Karan',
3      idade: 18,
4      curso: 'Matematica'
5  };
6
7  for (let propriedade in aluno) {
8      console.log(propriedade + ': ' + aluno[propriedade]);
9  }
10
```

- Saída: Nome das propriedades e seus valores.

Percorrendo Arrays com `for...of`

```
1 const linguagens = ['JavaScript', 'Python', 'Java'];
2
3 for (let linguagem of linguagens) {
4     console.log(linguagem);
5 }
6
```

- Saída: 'JavaScript', 'Python', 'Java'

Percorrendo Arrays com `forEach`

```
1 const numeros = [1, 2, 3, 4, 5];
2 numeros.forEach((numero) => {
3     console.log(numero);
4 });
5
```

- Saída: 1, 2, 3, 4, 5

Usando `map` para Transformar Arrays

```
1 const numeros = [1, 2, 3];
2 const dobrados = numeros.map((numero) => numero * 2);
3 console.log(dobrados); // Saída: [2, 4, 6]
4
```

Agora vem a nata da aula!

Atenção total!

Sintaxe de Arrow Functions

```
1  // Funcao tradicional
2  function soma(a, b) {
3      return a + b;
4  }
5
6  // Arrow function
7  const soma = (a, b) => {
8      return a + b;
9  };
10
11 // Arrow function com retorno implicito
12 const soma = (a, b) => a + b;
13
```

Exemplos de Arrow Functions

```
1  // Sem parametros
2  const ola = () => console.log('Ola Mundo!');
3  ola(); // Saida: 'Ola Mundo!'
4
5  // Com um parametro
6  const quadrado = x => x * x;
7  console.log(quadrado(4)); // Saida: 16
8
9  // Com varios parametros
10 const multiplicar = (a, b) => a * b;
11 console.log(multiplicar(3, 5)); // Saida: 15
12
```

Arrow Functions e Arrays

```
1 const numeros = [1, 2, 3, 4, 5];
2 const dobrados = numeros.map(n => n * 2);
3 console.log(dobrados); // Saida: [2, 4, 6, 8, 10]
4
```

Arrow Functions e Objetos

```
1 const criarPessoa = (nome, idade) => ({
2     nome: nome,
3     idade: idade
4 });
5
6 const pessoa = criarPessoa('Karan', 18);
7 console.log(pessoa); // Saida: { nome: 'Karan', idade: 18
      }
8
```

Entendendo o `this` em Objetos

- `this` referencia o objeto atual em que um método está sendo chamado.
- Em métodos de objetos, `this` permite acessar outras propriedades e métodos do mesmo objeto.
- O valor de `this` pode variar dependendo do contexto de chamada.

Exemplo: Usando `this` em Métodos

```
1 const pessoa = {
2     nome: 'Karan',
3     saudacao: function() {
4         console.log('Ola, meu nome eh ' + this.nome);
5     }
6 };
7
8 pessoa.saudacao(); // Saida: 'Ola, meu nome eh Karan'
9
```

Exemplo: Sem Usar `this`

```
1 const pessoa = {
2     nome: 'Karan',
3     saudacao: function() {
4         console.log('Ola, meu nome e ' + pessoa.nome);
5     }
6 };
7
8 pessoa.saudacao(); // Saida: 'Ola, meu nome e Karan'
9
```

Cuidado com o Contexto do this

```
const pessoa = {
    nome: 'Maria',
    saudacao: function() {
        console.log('Ola, meu nome e ' + this.nome);
    }
};

const saudacao = pessoa.saudacao;
saudacao(); // Saida: 'Ola, meu nome e undefined'

```

Solução: Vincular o `this`

```
1 const saudacao = pessoa.saudacao.bind(pessoa);
2 saudacao(); // Saída: 'Ola, meu nome e Maria'
3
```

Arrow Functions e o this

```
1 const pessoa = {
2     nome: 'Joao',
3     saudacao: () => {
4         console.log('Ola, meu nome e ' + this.nome);
5     }
6 };
7
8 pessoa.saudacao(); // Saida: 'Ola, meu nome e undefined'
9
```

Resumo da Aula

- Aprendemos sobre arrays e como manipulá-los.
- Exploramos a criação de objetos e acesso a propriedades.
- Vimos como utilizar diferentes loops para iterar em estruturas de dados.
- Conhecemos as arrow functions e seus usos.
- Detalhamos o uso do `this` em objetos JavaScript.

Próximos Passos

- Praticar criando arrays e objetos personalizados.
- Experimentar o uso de arrow functions em diferentes contextos.
- Explorar o comportamento do `this` em funções e objetos.
- Resolver exercícios de manipulação de dados em JavaScript.

Obrigado!

Alguma pergunta?