USCS
UNIVERSIDADE DE SÃO CAETANO DO SUL

Linguagem de Programação

Prof. Renato Carioca Duarte

Unidade 03

Coleções

Coleções

- Em JavaScript, uma coleção é uma estrutura de dados que permite armazenar e organizar dados.
- As coleções podem ser usadas para armazenar uma variedade de dados, incluindo números, strings, objetos e funções.

- Existem vários tipos diferentes de coleções em JavaScript, incluindo:

Coleções

1. Array: Uma matriz é uma coleção de elementos ordenados.

Os elementos de uma matriz podem ser de qualquer tipo, incluindo números, strings, objetos e funções.

2. Map: Um mapa é uma coleção de pares de chave-valor, semelhante a um objeto.

A diferença entre um mapa e um objeto é que as chaves de um mapa devem ser únicas, enquanto as chaves de um objeto não precisam ser únicas.

Coleções

3. Set: Um conjunto é uma coleção de elementos únicos.

Os elementos de um conjunto podem ser de qualquer tipo, incluindo números, strings, objetos e funções.

4. Object: Um objeto é uma coleção de pares de chave-valor.

As chaves de um objeto podem ser de qualquer tipo, incluindo strings, números e objetos.

Os valores de um objeto podem ser de qualquer tipo, incluindo números, strings, objetos e funções.

Exemplos de Utilização de Coleções

1. Array - Para armazenar uma lista de itens:

Você pode usar uma matriz para armazenar uma lista de itens, como uma lista de compras ou uma lista de tarefas.

2. Map - Para armazenar pares de chave-valor:

Você pode usar um mapa para armazenar pares de chave-valor, como uma lista de produtos e seus preços ou uma lista de clientes e seus endereços.

Exemplos de Utilização de Coleções

3. Set - Para armazenar elementos únicos:

Você pode usar um conjunto para armazenar elementos únicos, como uma lista de números primos ou uma lista de palavras únicas em uma frase.

4. Object - Para armazenar dados sobre um usuário:

Você pode usar um objeto para armazenar dados sobre um usuário, como o nome do usuário, o endereço de e-mail do usuário e a senha do usuário.

Array

- Um array é um tipo de coleção
- Permite armazenar e organizar vários valores
- Estrutura indexada

- Arrays em JavaScript armazenam vários valores em uma única variável
- Utilização: armazenar, recuperar e manipular dados de forma eficiente

Array

- Elementos de um array podem ser:
  - Tipos primitivos de dados
  - Objetos
  - Outros arrays

- Acesso aos dados do array:
  - Índices numéricos (0-based)
  - Métodos, como forEach(), para percorrer os elementos

Array

- Um array é um tipo de objeto que permite armazenar e organizar vários valores em uma única estrutura indexada.
- Arrays em JavaScript são usados para armazenar vários valores em uma única variável.
- O array é uma estrutura de dados que contém um índice numérico e um elemento, que pode ser de qualquer tipo primitivo de dados, um objeto ou, até mesmo, um outro array.
- O acesso aos dados do array é feito através de índices ou métodos como o forEach

Array

- Os arrays em JavaScript são objetos especiais que têm algumas características distintas:

1. Indexação: Os elementos do array são acessados através de um índice numérico, começando do índice 0 para o primeiro elemento, 1 para o segundo elemento e assim por diante.

2. Tamanho dinâmico: Os arrays em JavaScript podem ser modificados após sua criação. Você pode adicionar ou remover elementos conforme necessário.

3. Tipos de dados heterogêneos: Diferentemente de algumas linguagens de programação, os arrays em JavaScript podem conter elementos de tipos de dados diferentes.

Array

Neste exemplo, criamos um array chamado `numeros` contendo cinco elementos numéricos.

Os elementos são acessados usando a notação de colchetes `[]`, indicando o índice do elemento que queremos acessar.

```
// Criando um array com alguns números
const numeros = [10, 20, 30, 40, 50];

// Acessando elementos do array pelo índice
console.log(numeros[0]); // Saída: 10
console.log(numeros[2]); // Saída: 30

// Acessando todos elementos do array pelo índice
for (let i = 0; i < numeros.length; i++) {
  console.log(numeros[i]);
}
```

Array

Arrays também podem conter outros arrays e objetos, permitindo criar estruturas de dados complexas para armazenar informações de forma organizada.

```
// Array de arrays (matriz)
const matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

// Array de objetos
const pessoas = [
  { nome: "João", idade: 30 },
  { nome: "Maria", idade: 25 },
  { nome: "Pedro", idade: 40 }
];
```

forEach

- A função `forEach` é um método disponível em arrays do JavaScript que permite percorrer cada elemento do array e executar uma função de callback para cada um deles.
- É uma forma concisa e funcional de iterar pelos elementos do array, sem a necessidade de utilizar um loop `for`.

```
array.forEach((elemento, índice, array) => {
  // código a ser executado para cada elemento do array
});
```

forEach

```
array.forEach((elemento, índice, array) => {
  // código a ser executado para cada elemento do array
});
```

- O parâmetro callback é uma função que será chamada para cada elemento do array.
- Ela pode receber até três argumentos:

  elemento: O elemento atual do array durante a iteração.

  índice (opcional): O índice do elemento atual no array durante a iteração.

  array (opcional): O próprio array que está sendo percorrido.

forEach

```
const numeros = [10, 20, 30, 40, 50];

numeros.forEach( (numero, index) => {
  console.log(`Elemento ${index}: ${numero}`);
});
```

- Neste exemplo, usamos o `forEach` para percorrer o array `numeros`.
- A função de callback recebe dois parâmetros: `numero`, que representa cada elemento do array, e `index`, que representa o índice do elemento no array.
- A função de callback imprime no console uma mensagem formatada com o índice e o valor de cada elemento.

Map

- O método `map` é usado para criar um novo array a partir de um array existente, aplicando uma função de callback a cada elemento do array original.
- Essa função de callback recebe o elemento atual, o índice do elemento e o próprio array como argumentos, e o valor retornado pela função será o elemento do novo array.
- O novo array terá o mesmo comprimento do array original.

```
const numeros = [1, 2, 3, 4, 5];

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

Filter

- O método `filter` é usado para criar um novo array contendo somente os elementos do array original que atendem a um determinado critério definido por uma função de callback.
- Essa função de callback também recebe o elemento atual, o índice e o array como argumentos e deve retornar `true` ou `false` para indicar se o elemento deve ou não ser incluído no novo array.

```
const numeros = [1, 2, 3, 4, 5];

const pares = numeros.filter((n) => n % 2 === 0);
console.log(pares); // Saída: [2, 4]
```

Find

- O método `find` é usado para encontrar o primeiro elemento do array original que atende a um critério definido por uma função de callback.
- Assim como nas outras funções, essa função de callback recebe o elemento atual, o índice e o array como argumentos e deve retornar `true` ou `false`.
- O valor retornado pelo `find` será o primeiro elemento do array que satisfaça a condição ou `undefined` se nenhum elemento atender ao critério.

```
const numeros = [1, 2, 3, 4, 5];

const primeiroPar = numeros.find((numero) => numero % 2 === 0);
console.log(primeiroPar); // Saída: 2
```

desestruturação de array

- A desestruturação de array é uma funcionalidade poderosa do JavaScript que permite extrair elementos individuais de um array e atribuí-los a variáveis separadas em uma única linha de código.
- Essa sintaxe simplifica o acesso aos elementos do array, tornando o código mais legível e conciso.

- A desestruturação de array é feita através do uso de colchetes `[ ]` e pode ser usada para declarar e atribuir valores a várias variáveis simultaneamente.
- A ordem em que as variáveis são declaradas corresponde à ordem dos elementos no array.

desestruturação de array

```
// Array com três elementos
const numeros = [10, 20, 30];

// Desestruturação do array
const [primeiro, segundo, terceiro] = numeros;

console.log(primeiro); // Saída: 10
console.log(segundo);  // Saída: 20
console.log(terceiro); // Saída: 30
```

Operador de Propagação

- O spread operator (operador de propagação) é uma funcionalidade do JavaScript introduzida no ECMAScript 6 (ES6) que permite copiar e combinar elementos de arrays e objetos de forma mais concisa.
- O spread operator é representado pelo uso de três pontos (`...`) seguidos de um array ou objeto, e pode ser aplicado em diversas situações para simplificar operações com estruturas de dados.

Operador de Propagação

- Para arrays:
  - Copiar arrays;
  - Concatenar arrays;
  - Adicionar elementos em uma nova cópia do array;
  - Transformar strings em arrays de caracteres, etc.

- Para objetos:
  - Copiar objetos;
  - Combinar objetos.

Copiando Arrays

```
const numeros = [1, 2, 3];
const copiaNumeros = [...numeros];

console.log(copiaNumeros); // Saída: [1, 2, 3]
```

Concatenando Arrays


```
const numeros1 = [1, 2, 3];
const numeros2 = [4, 5, 6];
const numerosConcatenados = [...numeros1, ...numeros2];

console.log(numerosConcatenados); // Saída: [1, 2, 3, 4, 5, 6]
```

Adicionando Elementos em uma Nova Cópia do Array


```
const numeros = [1, 2, 3];
const numerosComNovoElemento = [...numeros, 4];

console.log(numerosComNovoElemento); // Saída: [1, 2, 3, 4]
```

Operador Rest

- O operador Rest em JavaScript é representado pelo uso de três pontos (`...`) antes do nome de um parâmetro de função ou na desestruturação de arrays e objetos.
- Ele permite capturar um número indefinido de argumentos ou elementos em uma única estrutura de dados.

- Quando usado como parâmetro de função, o operador Rest permite agrupar um conjunto variável de argumentos passados para a função em um único array.
- Isso é especialmente útil quando você quer lidar com um número desconhecido de argumentos ou quando deseja evitar definir muitos parâmetros na função.

Operador Rest como parâmetro de função

```
function somar(...numeros) {
  let resultado = 0;
  numeros.forEach(numero => {
    resultado += numero;
  });
  return resultado;
}

console.log(somar(1, 2, 3, 4, 5)); // Saída: 15
console.log(somar(10, 20, 30));    // Saída: 60
```

- A função `somar` usa o operador Rest `...numeros` como parâmetro.
- Isso permite que você passe um número indefinido de argumentos ao chamar a função.
- Internamente, esses argumentos são agrupados em um array chamado `numeros`, que pode ser percorrido usando `forEach` para realizar a soma.

Operador Rest na Desestruturação de arrays

```
const numeros = [1, 2, 3, 4, 5];
const [primeiro, segundo, ...resto] = numeros;

console.log(primeiro); // Saída: 1
console.log(segundo);  // Saída: 2
console.log(resto);    // Saída: [3, 4, 5]
```

- Neste exemplo, o operador Rest `...resto` captura todos os elementos restantes do array após a atribuição das primeiras duas variáveis `primeiro` e `segundo`.

Map

- Um mapa é uma coleção de pares de chave-valor, semelhante a um objeto.
- A diferença entre um mapa e um objeto é que as chaves de um mapa devem ser únicas, enquanto as chaves de um objeto não precisam ser únicas.

Map

```
const myMap = new Map();
myMap.set("key1", "value1");
myMap.set("key2", "value2");

// Imprime "value1"
console.log(myMap.get("key1"));

// Imprime "value2"
console.log(myMap.get("key2"));
```

- Este código cria um mapa chamado myMap e adiciona dois pares de chave-valor ao mapa: key1 com o valor value1 e key2 com o valor value2.
- Em seguida, o código imprime o valor da chave key1 e o valor da chave key2.

Set

- Um conjunto é uma coleção de elementos únicos.
- Os elementos de um conjunto podem ser de qualquer tipo, incluindo números, strings, objetos e funções.
- Os conjuntos são semelhantes às matrizes, mas os elementos de um conjunto não precisam estar ordenados e não podem ser repetidos.
- Os conjuntos são uma estrutura de dados muito eficiente para armazenar elementos únicos.

Set

```
const mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(3);

// Imprime 1, 2, e 3
console.log(mySet);
```

- Este código cria um conjunto chamado mySet e adiciona três elementos ao conjunto: 1, 2 e 3.
- Em seguida, o código imprime os elementos do conjunto.

Dúvidas

??????

Questionário

Questionário Unidade 3 no classroom

Exercícios

1. Crie um array com os numeros 1,2,3,4,5 e imprima seu conteúdo (usando a console, p. ex).

2. Crie um array com os numeros 1,2,3,4,5, adicione um elemento ao final do array e imprima seu conteúdo (usando a console, p. ex).

3. Crie um array com os numeros 1,2,3,4,5, remova um elemento do início do array e imprima seu conteúdo (usando a console, p. ex).

4. Crie um array com os numeros 77,88,44,22,55 e imprima o índice do elemento 22 do array.

5. Crie um array com os numeros 77,88,44,22,55 e percorra um array e imprima cada elemento.

Exercícios

6. Criar um array de números 10, 11, 12, 13 e 14 e retornar um novo array com os números impares. Use filter.

7. Criar um array de números 10, 11, 12, 13 e 14 e retornar outro array com o quadrado destes números. Use map.