Aula 3

o que vamos aprender nessa aula:

- Laço de repetição.
- Map
- Objetos

Laços de Repetição em JavaScript

Em JavaScript, os laços de repetição são estruturas que permitem executar um bloco de código repetidamente com base em uma condição. Existem diferentes tipos de laços de repetição em JavaScript, sendo o mais comum o laço `for`.

O laço `for` é usado para executar um bloco de código um determinado número de vezes. Ele consiste em três partes: a inicialização, a condição de continuação e a atualização. Veja a sintaxe do laço `for`:

```
for (inicialização; condição; atualização) {
  // bloco de código a ser executado
}
```

- A inicialização é executada apenas uma vez no início do laço e é usada para definir uma variável de controle ou contador.
- A condição é avaliada antes de cada iteração do laço. Se a condição for verdadeira, o bloco de código é executado. Se a condição for falsa, o laço é encerrado.
- A atualização é executada após cada iteração do laço e é usada para atualizar a variável de controle ou contador.

Exemplos de Laço `for` sem Array

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

Neste exemplo, o laço `for` é executado 5 vezes. A cada iteração, o valor de `i` é impresso no console.

Exemplos de Laço `for` com Array

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

for (let i = 0; i < numeros.length; i++) {
  console.log(numeros[i]);
}
```

Neste exemplo, o laço `for` é usado para percorrer cada elemento do array `numeros`. A cada iteração, o elemento correspondente é impresso no console.

Exemplos de Laço `for` com Matriz

```
let matriz = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

for (let i = 0; i < matriz.length; i++) {
  console.log(matriz[i]);
  for (let j = 0; j < matriz[i].length; j++) {
    console.log(matriz[i][j]);
  }
}
```

Neste exemplo, o laço `for` é usado para percorrer cada elemento da matriz. O primeiro laço `for` itera pelas linhas da matriz, e o segundo laço `for` itera pelos elementos de cada linha.

Os laços de repetição são fundamentais na programação em JavaScript, permitindo executar blocos de código repetidamente. Ao entender como funcionam os laços `for` e como aplicá-los em diferentes situações, podemos criar algoritmos mais eficientes e manipular dados de forma estruturada.

`Map` em JavaScript

O `Map` é uma estrutura de dados em JavaScript que permite armazenar pares de chave-valor. É uma coleção de elementos onde cada elemento é composto por uma chave única e um valor associado a essa chave.

Importância do Map

O `Map` é uma estrutura muito útil em JavaScript, pois nos permite organizar e acessar dados de forma eficiente. Algumas das principais características e benefícios do `Map` são:

- Permite armazenar dados de forma estruturada, associando uma chave a um valor.
- Garante a unicidade das chaves, evitando duplicações.
- Permite acessar facilmente os valores associados a uma chave específica.

Exemplos de uso do Map

Exemplo 1: Armazenando informações de alunos

```
let alunos = new Map();

alunos.set('João', 18);
alunos.set('Maria', 20);
alunos.set('Pedro', 19);

console.log(alunos.get('Maria')); // 20
```

Neste exemplo, criamos um `Map` chamado `alunos` para armazenar as idades dos alunos. Cada aluno é representado por uma chave (nome) e um valor (idade). Podemos acessar a idade de um aluno específico usando o método `get`.

Exemplo 2: Contagem de palavras

```
let texto = 'Este é um exemplo de texto com várias palavras repetidas';

let palavras = texto.split(' ');
let contador = new Map();

for (let i = 0; i < palavras.length; i++) {
  const palavra = palavras[i];
  if (contador.has(palavra)) {
    contador.set(palavra, contador.get(palavra) + 1);
  } else {
    contador.set(palavra, 1);
  }
}

console.log(contador.get('palavras')); // 2
```

Neste exemplo, utilizamos o `Map` para contar a ocorrência de cada palavra em um texto. Dividimos o texto em palavras usando o método `split`, e em seguida, percorremos cada palavra e as armazenamos como chave no `Map`. Se a palavra já existe no `Map`, incrementamos o valor associado à chave. Caso contrário, criamos uma nova entrada no `Map` com valor 1.

O `Map` é uma estrutura de dados poderosa em JavaScript, que nos permite organizar, acessar e manipular dados de forma eficiente. Compreender o conceito de `Map` e saber aplicá-lo em diferentes situações é fundamental para desenvolver algoritmos eficientes e lidar com dados de maneira estruturada.

Objetos em JavaScript

Em JavaScript, objetos são estruturas de dados que nos permitem representar entidades complexas com propriedades e comportamentos. Um objeto é composto por um conjunto de pares chave-valor, onde cada chave é uma propriedade e cada valor é o valor associado a essa propriedade.

Exemplo de Criação de Objeto Simples

```
let pessoa = {
  nome: "João",
  idade: 30,
  profissao: "Engenheiro"
};

console.log(pessoa.nome); // João
console.log(pessoa.idade); // 30
console.log(pessoa.profissao); // Engenheiro
```

Neste exemplo, criamos um objeto chamado `pessoa` que possui as propriedades `nome`, `idade` e `profissao`. Podemos acessar o valor de cada propriedade usando a sintaxe `objeto.propriedade`.

Exemplo com Lista de Objetos

```
let carros = [
  { marca: "Toyota", modelo: "Corolla", ano: 2020 },
  { marca: "Honda", modelo: "Civic", ano: 2019 },
  { marca: "Chevrolet", modelo: "Onix", ano: 2021 }
];

for (let i = 0; i < carros.length; i++) {
  const carro = carros[i];
  console.log(carro.marca + " " + carro.modelo + " - " + carro.ano);
}
```

Neste exemplo, temos uma lista de objetos chamada `carros`, onde cada objeto representa um carro com suas respectivas propriedades. Utilizamos o método `forEach` para percorrer cada objeto da lista e exibir as informações do carro.

Laço de repetição com Objeto

```
let pessoa = {
  nome: "João",
  idade: 30,
  profissao: "Engenheiro"
};

for (let propriedade in pessoa) {
  console.log(propriedade + ": " + pessoa[propriedade]);
}
```

Neste exemplo, temos um objeto chamado `pessoa` com propriedades como `nome`, `idade` e `profissao`. Utilizando o laço `for...in`, percorremos cada propriedade do objeto e exibimos o nome da propriedade junto com o valor correspondente.

O laço `for...in` é útil para percorrer as propriedades de um objeto de forma dinâmica, permitindo manipular e exibir os valores associados a cada propriedade.

Exercicios

1. Criando uma lista de contatos

Crie uma lista de contatos, essa lista deve ter o seguintes dados:

- Nome
- Telefone
- Endereço
  - Nome da rua

    - Número da residência
    - Nome do bairro
- Data de nascimento

2. Percorrendo a lista de contato

Crie um laço de repetição que percorra toda a lista de contatos e mostre uma frase contento Nome, Telefone, Endereço e data de nascimento, exemplo:

Bonus: Utilizar template string

```
Bruno Cabral, telefone (11) 99999-9999, mora na Rua Major Prado, no número 200, no bairro Jardim Terezinha, nascido na data de 04/04/19
```

3. Criando estrutura de números

Crie laços de repetição para retornar a seguinte estrutura de numeros:

```
1
12
123
1234
12345
```

Agora tente criar o inverso:

```
12345
1234
123
12
1
```