



Manipulação de JSON

Resumo

JSON (JavaScript Object Notation) é um formato leve e independente de linguagem para troca de dados. Ele é baseado na sintaxe do JavaScript, mas pode ser usado com qualquer linguagem de programação. O JSON é amplamente utilizado para transmitir e armazenar dados estruturados.

O formato JSON é composto por pares de chave-valor, onde as chaves são strings e os valores podem ser de diferentes tipos de dados, como strings, números, booleanos, arrays, objetos aninhados e valores nulos. Os pares de chave-valor são separados por vírgulas e são colocados entre chaves {}.

Aqui está um exemplo simples de um objeto JSON:

```
```json
{
 "nome": "John",
 "idade": 30,
 "cidade": "Nova York",
 "hobbies": ["leitura", "correr", "cozinhar"],
 "casado": false
}
```
```

Neste exemplo, temos um objeto JSON com várias propriedades. A propriedade “nome” tem o valor “John”, a propriedade “idade” tem o valor 30, a propriedade “cidade” tem o valor “Nova York” e assim por diante.

Uma das principais vantagens do JSON é a sua simplicidade e legibilidade para humanos. Além disso, é amplamente suportado por diferentes plataformas e linguagens de programação, tornando-o um formato de escolha para a comunicação e o armazenamento de dados estruturados.

O JavaScript fornece métodos embutidos, como `JSON.stringify()` e `JSON.parse()`, para serializar e desserializar objetos JavaScript em strings JSON e vice-versa, permitindo que os desenvolvedores trabalhem facilmente com dados JSON em suas aplicações.

Em resumo, o JSON é um formato de dados estruturados baseado em texto que é amplamente usado para trocar informações entre diferentes sistemas e linguagens de programação.

Melhores Práticas e Técnicas Avançadas de Manipulação de JSON^[1]

Embora o JSON seja um formato de dados simples e fácil de usar, existem algumas melhores práticas que podem ajudar a otimizar o uso e a manipulação de dados JSON.

1. Mantenha o JSON o mais plano possível: Embora o JSON permita a criação de estruturas de dados complexas e aninhadas, é geralmente mais eficiente manter a estrutura de dados o mais plana possível. Isso facilita a leitura e a manipulação dos dados.

2. Use nomes de chaves significativos: Os nomes das chaves devem ser descritivos e significativos. Isso torna o JSON mais legível e fácil de entender.

3. Evite valores nulos: Sempre que possível, evite usar valores nulos em seu JSON. Valores nulos podem ser confusos e podem levar a

erros de interpretação.



4. Use arrays para dados repetitivos: Se você tem um conjunto de dados que se repete, como uma lista de itens, use um array para armazenar esses dados.

5. Valide o JSON: Sempre valide o JSON antes de usá-lo. Existem muitas ferramentas online gratuitas que podem ajudar a validar o JSON e garantir que ele esteja no formato correto.

Além dessas melhores práticas, existem algumas técnicas avançadas que podem ser úteis ao trabalhar com JSON.

1. Transformação de JSON: Existem muitas bibliotecas disponíveis que permitem transformar JSON de uma forma para outra. Isso pode ser útil quando você está trabalhando com APIs que retornam JSON em um formato que não é ideal para suas necessidades.

2. Consulta de JSON: Existem várias linguagens de consulta JSON, como JSONPath e JSON Query, que permitem consultar e manipular dados JSON de maneira semelhante ao SQL para bancos de dados.

3. JSON Schema: JSON Schema é uma maneira de descrever a estrutura do seu JSON e pode ser usado para validação, documentação e geração de interface do usuário.

4. Streaming de JSON: Para grandes conjuntos de dados, pode ser mais eficiente processar o JSON em um fluxo, em vez de carregar todo o conjunto de dados na memória de uma vez.

Em resumo, o JSON é uma ferramenta poderosa para a troca de dados, mas, como qualquer ferramenta, é importante saber como usá-

la efetivamente. Com as melhores práticas e técnicas avançadas, você pode otimizar o uso do JSON em suas aplicações.



Conteúdo Bônus

No JavaScript, trabalhar com formatos JSON é bastante comum e simples. O JSON (JavaScript Object Notation) é uma forma de representar dados estruturados usando uma sintaxe baseada em JavaScript. É amplamente utilizado para transmitir e armazenar dados.

Existem duas operações principais que você pode realizar ao trabalhar com JSON no JavaScript:

1. Converter um objeto JavaScript em uma string JSON (serialização):

Isso envolve converter um objeto JavaScript em uma string JSON para que possa ser transmitido ou armazenado. Você pode usar o método `JSON.stringify()` para fazer isso. Aqui está um exemplo:

```
```javascript
const objeto = { nome: "John", idade: 30, cidade: "Nova York" };
const jsonString = JSON.stringify(objeto);
console.log(jsonString);
```
```

A saída será:

```
...
{"nome":"John","idade":30,"cidade":"Nova York"}
...
```

2. Converter uma string JSON em um objeto JavaScript (desserialização):

Isso envolve converter uma string JSON de volta em um objeto JavaScript para que você possa trabalhar com os dados contidos nela. Você pode usar o método `JSON.parse()` para fazer isso. Aqui está um exemplo:

```
```javascript
const jsonString = '{"nome":"John","idade":30,"cidade":"Nova York"}';
const objeto = JSON.parse(jsonString);
console.log(objeto.nome);
console.log(objeto.idade);
console.log(objeto.cidade);
```

A saída será:

```
...
John
30
Nova York
...
```

Esses são os conceitos básicos para trabalhar com JSON no JavaScript. É importante lembrar que o JSON suporta uma variedade de tipos de dados, incluindo strings, números, booleanos, arrays e objetos aninhados.

Você também pode fazer manipulações mais avançadas com JSON, como filtrar e percorrer arrays JSON ou modificar propriedades de objetos JSON. Essas operações são realizadas usando as estruturas e métodos padrão do JavaScript, como loops, condicionais e métodos de array.

Espero que isso tenha sido útil!

## Referência Bibliográfica



FLANAGAN, David. JavaScript: **O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!: programação JavaScript**. 1ª Ed. São Paulo: Alta Books, 2016.

## ATIVIDADE PRÁTICA

**Título da Prática:** Como podemos manipular dados JSON?

**Objetivos:** Implementar uso do fetch


**Materiais, Métodos e Ferramentas:** Para realizar esta prática vamos utilizar o Visual Studio Code

### Prática

Neste desafio, você deve criar um código em JavaScript para consumir uma API fake e exibir os dados retornados em uma página web. Utilizaremos a função 'fetch', que permite fazer requisições HTTP assíncronas, para buscar os dados da API.

A API fake fornece informações em formato JSON, incluindo imagens e descrições. Seu objetivo é listar esses dados em um '<div>' no HTML, exibindo as imagens e as descrições correspondentes.

Ao carregar a página, o código JavaScript deve fazer uma requisição GET à URL da API fake. Se a requisição for bem-sucedida, os dados JSON devem ser convertidos em um objeto JavaScript utilizando o método 'response.json()'. Em seguida, os dados devem ser percorridos e para cada

item, um elemento '`<img>`' deve ser criado. A propriedade '`src`' do elemento '`<img>`' deve ser definida com a URL da imagem correspondente e  propriedade '`alt`' deve ser definida com a descrição correspondente.

Os elementos '`<img>`' devem ser adicionados como filhos de um '`<div>`' com um determinado id, que será usado para exibir os dados na página web.

Certifique-se de substituir a URL da API fake pela URL correta que você irá utilizar. Além disso, adapte o código para corresponder à estrutura dos dados retornados pela API fake que você está consumindo.

Este desafio visa testar suas habilidades em consumir APIs, manipular dados JSON, criar elementos HTML dinamicamente e manipular o DOM de uma página web. Boa sorte!

Boa sorte!

## **Resolução**

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Exemplo de consumo de API com fetch</title>
```

```
</head>
```

```
<body>
```

```
<div id="dados"></div>
```

<script>



// Função para carregar os dados da API e exibir no HTML

```
async function carregarDados() {
```

```
 try {
```

```
 const response = await fetch('https://api.fake.com/dados'); // Substitua
pela URL da API fake
```

```
 if (!response.ok) {
```

```
 throw new Error('Erro ao carregar os dados da API!');
```

```
 }
```

```
 const dados = await response.json();
```

```
 const divDados = document.getElementById('dados');
```

```
 dados.forEach((item) => {
```

```
 const img = document.createElement('img');
```

```
 img.src = item.imagem; // Substitua pela propriedade correta da
imagem na API
```

```
 img.alt = item.descricao; // Substitua pela propriedade correta da
descrição na API
```

```
 divDados.appendChild(img);
```

```
 });
```

```
 } catch (error) {
```



```
console.error(error);
```



```
}
```

```
}
```

// Chamar a função para carregar os dados quando a página for carregada

```
document.addEventListener('DOMContentLoaded', carregarDados);
```

```
</script>
```

```
</body>
```

```
</html>
```

**[Ir para exercício](#)**