Aula: Usando APIs com JavaScript

Prof. Karan Luciano

Desenvolvimento Web - Ensino Médio

25 de outubro de 2024

Sumário

O que é uma API?

- API significa Interface de Programação de Aplicações.
- É como uma ponte que permite que diferentes programas conversem entre si.
- Por exemplo, quando você usa um app de clima, ele usa uma API para pegar as informações do tempo.

Por que usar APIs?

- Acessar informações e serviços que já existem.
- Facilita a criação de aplicativos com mais recursos.
- Permite integrar com outros serviços, como mapas ou redes sociais.

Formatos de Dados em APIs

- APIs trocam dados em formatos que ambos os programas entendem.
- Os formatos mais comuns são JSON e XML.
- Hoje, o JSON é mais usado porque é mais simples e fácil de trabalhar com JavaScript.

O que é JSON?

- JSON significa JavaScript Object Notation.
- É uma forma leve de organizar e trocar dados.
- Fácil de ler para pessoas e para computadores.

Estrutura de um Objeto JSON

- Os dados são organizados em pares chave-valor.
- As chaves são palavras que identificam os dados e ficam entre aspas.
- Os valores podem ser textos, números, listas ou outros objetos.

Exemplo de Objeto JSON

```json
{
    "nome": "Maria",
    "idade": 30,
    "profissao": "Engenheira",
    "habilidades": ["JavaScript", "Python", "C++"],
    "endereco": {
        "cidade": "São Paulo",
        "pais": "Brasil"
    }
}
```

Diferenças entre JSON e Objetos JavaScript

- JSON é uma string formatada de um jeito específico.
- Objetos JavaScript são usados diretamente no código.
- No JSON, as chaves sempre usam aspas duplas. Nos objetos JavaScript, podem usar aspas simples, duplas ou sem aspas.

Convertendo entre JSON e Objetos

- `JSON.parse()` transforma uma string JSON em um objeto JavaScript.
- `JSON.stringify()` transforma um objeto JavaScript em uma string JSON.

Exemplos de Conversão

```
// String JSON
const jsonString = '{"nome": "Carlos", "idade": 25}';

// Convertendo para objeto JavaScript
const obj = JSON.parse(jsonString);
console.log(obj.nome); // Saída: 'Carlos'

// Objeto JavaScript
const pessoa = {
    nome: 'Ana',
    idade: 28
};

// Convertendo para JSON
const json = JSON.stringify(pessoa);
console.log(json); // Saída: '{"nome":"Ana","idade":28}'
```

O que é a fetch API?

- A fetch API é uma forma moderna de fazer pedidos para a internet usando JavaScript.
- Ela ajuda o seu site ou app a pegar informações de outros lugares, como dados de clima ou informações de usuários.
- É mais fácil de usar e entender do que as formas antigas.

Fazendo um Pedido GET

```
fetch('https://api.exemplo.com/dados')
    .then(response => response.json())
    .then(data => {
        console.log(data);
    })
    .catch(error => {
        console.error('Erro:', error);
    });
```

- `fetch()` faz o pedido para a URL indicada.
- Depois, transforma a resposta em JSON.
- Mostra os dados no console.
- Se der erro, mostra uma mensagem de erro.

Usando async/await

```
async function obterDados() {
    try {
        const response = await fetch('https://api.exemplo.com/
    dados');
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Erro:', error);
    }
}

obterDados();
```

- async/await ajuda a escrever código assíncrono de forma mais fácil.
- O código fica mais parecido com comandos normais.
- Usa try/catch para tratar erros.

Enviando Dados com um Pedido POST

```
const dados = {
    nome: 'João',
    idade: 35
};

fetch('https://api.exemplo.com/usuarios', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(dados)
})
.then(response => response.json())
.then(data => {
    console.log('Sucesso:', data);
});
```

- Dizemos que o método do pedido é `POST`.
- Dizemos que estamos enviando dados em JSON.
- Convertimos os dados para uma string JSON antes de enviar.

Como Usar os Dados Recebidos

- Os dados que vêm da API geralmente estão em JSON.
- Depois de transformar em objeto JavaScript, você pode usar esses dados no seu site ou app.
- Pode mostrar informações na tela, atualizar imagens, criar listas, etc.

Exemplo: Listando Usuários

```
async function listarUsuarios() {
    try {
        const response = await fetch('https://api.exemplo.com/usuarios');
        const usuarios = await response.json();

        usuarios.forEach(usuario => {
            console.log(`Nome: ${usuario.nome}, Idade: ${usuario.idade}`);
        });
    } catch (error) {
        console.error('Erro:', error);
    }
}

listarUsuarios();
```

Exemplo: Atualizando a Página

```
async function mostrarPosts() {
    try {
        const response = await fetch('https://api.exemplo.com/
    posts');
        const posts = await response.json();

        const lista = document.getElementById('lista-posts');
        posts.forEach(post => {
            const item = document.createElement('li');
            item.textContent = post.titulo;
            lista.appendChild(item);
        });
    } catch (error) {
        console.error('Erro:', error);
    }
}

mostrarPosts();
```

- Cria itens na lista do HTML com os dados recebidos.
- Mostra os títulos dos posts na página.

Tratando Erros nos Pedidos

- Sempre pode dar erro quando você pede dados de uma API.
- Pode ser problema na internet, no servidor, ou nos dados.
- É importante tratar esses erros para que o usuário saiba o que aconteceu.

Exemplo: Tratando Erros - Parte 1

```
async function buscarDados() {
    try {
        const response = await fetch('https://api.exemplo.com/dados');

        if (!response.ok) {
            throw new Error('Erro na requisição: ' + response.status);
        }
```

- Verifica se a resposta foi bem-sucedida.
- Se não, lança um erro com a mensagem apropriada.

Exemplo: Tratando Erros - Parte 2

```
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Erro:', error);
    }
}

buscarDados();
```

- Mostra o erro no console para ajudar a identificar o problema.

Resumo da Aula

- Aprendemos o que são APIs e para que servem.
- Entendemos o formato JSON e como usá-lo com JavaScript.
- Vimos como fazer pedidos para pegar ou enviar dados usando a fetch API.
- Aprendemos a mostrar os dados recebidos na página.
- Sabemos como tratar erros quando algo dá errado.

Próximos Passos

- Praticar fazendo pedidos para diferentes APIs.
- Criar pequenos projetos que usam dados de APIs.
- Explorar mais sobre como melhorar a interação do usuário com os dados.

Obrigado!

Alguma pergunta?