

JS para Desenvolvimento WEB



JS



JS

Aula 5

Integração com APIs

APIs WEB

(Application programming interface)

- Uma interface que define uma série de regras para troca de informação entre dois sistemas via internet.
- Define como a informação dever ser pedida e como ela será entregue.
- Utiliza uma linguagem de marcação como Json ou XML para transmissão de dados, o que torna a comunicação independente das tecnologias utilizadas de cada lado.
- Por exemplo, um backend PHP pode trocar dados com um frontend em JS.

APIs REST

- Um sistema disponibiliza dados através de um endpoint identificado por uma URL (Uniform Resource Locator).

URL = <protocolo>://<endereço>/<parâmetros>?<query>

- A requisição é feita via protocolo HTTP (ou HTTPS).
- Por exemplo:

GET <https://meu.dominio/api/v2>

HTTPS

- **GET:** Tipo de requisição para **pedir** informações. Pode-se filtrar as informações via *query string*.

GET https://url.da.api/v2/produtos?categoria=tecnologia

- **POST:** Tipo de requisição para **enviar** dados. Os dados devem ser enviados no *body*.

POST https://url.da.api/v2/usuario

body = {nome : "mauro", senha : "batata-doce"}

HTTPS

- **PUT**: Tipo de requisição para **atualizar** informações.

PUT https://url.da.api/v2/usuario

body = {id : 1234, endereco : "novo endereco"}

- **DELETE**: Tipo de requisição para **excluir** informações.

DELETE https://url.da.api/v2/usuario

body = {id : 1234}

APIs REST

- A resposta carrega um código de status:

2xx = Sucesso

- 200 = Aceito com resposta
- 202 = Aceito (para processos assíncronos)
- 204 = Aceito sem resposta

3xx = Redireção

- 301 = Movido permanentemente
- 302 = Movido temporariamente

5xx = Erro no servidor

- 500 = Erro interno no servidor

APIs REST

4xx = Erro do cliente

400 = Bad request

401 = Não autorizado (Cliente não autenticado)

403 = Não autorizado (Cliente autenticado)

404 = Não encontrado

<https://restfulapi.net/http-status-codes/>

JSON

(JavaScript Object Notation)

- Uma formatação leve para troca de dados, humanamente legível e fácil de interpretar do ponto de vista computacional.
- É um subconjunto da linguagem Javascript.
 - Muito similar a objetos;
 - **Chaves** necessitam de aspas duplas;
 - Funções, undefined ou NaN não são aceitos como **valor**;
- Utiliza convenções compatíveis com várias linguagens de programação.

JSON

 exemplo.json X

Users > mauro > Downloads >  exemplo.json > ...

```
1  {
2      "numero"    : 1234,
3      "string"    : "string",
4      "array"     : ["array", 1, true],
5      "objeto"    : { },
6      "booleano"  : true,
7      "null"      : null
8  }
9
```

Fetch

```
fetch(url, options);
```

url: uma **string** que representa o endpoint alvo da requisição.

options: (opcional) um **objeto** que identifica algumas configurações.

```
options = {  
  method? : 'GET' | 'POST' | 'PUT' | 'DELETE'  
  headers? : {}  
  body? : "  
  ...  
}
```

Fetch

headers: um **objeto** que fornece informações aos servidor.

```
headers = {  
  //indica que o conteúdo do body é um JSON  
  'Content-Type' : 'application/json'  
  //token de sessão  
  'Authorization' : 'Bearer <token>'  
}
```

Fetch

body: uma **string** que carrega as informações que deseja-se passar ao servidor.

```
data = { nome : 'Mauro', senha : 'batata-doce' };
```

```
body = JSON.stringify( data );
```

Fetch

```
fetch('https://url.da.api/v2')  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Erro:', error));
```

Fetch

```
async function fetchData() {  
  try {  
    const response = await fetch('https://url.da.api/v2');  
    const data = await response.json();  
    console.log(data);  
  } catch (error) {  
    console.error('Error: ', error);  
  }  
}  
  
fetchData();
```

Exemplo

<https://github.com/DeivisFelipe/Minicurso-JS>

O que achou?



<https://forms.gle/cbtkTasZdtREyuxa6>