

# Requisições AJAX

Prof. Daniel Di Domenico



# Objetivo da aula

- Implementar requisições AJAX:
  - Comunicação entre o cliente (navegador) e o servidor (servidor Web/PHP) do sistema
    - Utilização da linguagem JavaScript
- Entender as formas de implementar uma requisição AJAX:
  - Execução assíncrona

# AJAX

- **Asynchronous JavaScript And XML**
  - Técnica de desenvolvimento para WEB que permite **enviar requisições HTTP ao servidor em segundo plano**
    - Não é uma linguagem de programação
    - Utiliza JavaScript
- **AJAX permite:**
  - Atualizar o HTML da página sem recarregá-la
  - Comunicação com o servidor após carregar a página
  - Busca e envio de dados para o servidor WEB em segundo plano
    - Requisição assíncrona
- [https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp)

# XMLHttpRequest

- **XMLHttpRequest** é a classe JavaScript que permite a comunicação com um servidor WEB em segundo plano
  - Classe que operacionaliza o AJAX
- Um objeto XMLHttpRequest permite:
  - **1-** Criar uma requisição ao servidor utilizando os métodos HTTP **GET** e **POST** (além dos outros existentes)
  - **2-** Definir uma função de retorno da requisição (callback)
    - Necessário devido a execução assíncrona
  - **3-** Receber a resposta da requisição:
    - Texto (pode ser um JSON)
    - XML

# Exemplo AJAX

- **Requisição assíncrona:**

```
var url = 'servico.php';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url);  
  
xhttp.onload = function() {  
    //Executado após a resposta do servidor  
    var retornoTexto = xhttp.responseText;  
    console.log(retornoTexto);  
}  
  
xhttp.send();
```

Este código só será executado quando for recebida a resposta do servidor

# Exemplo AJAX

- Envio de dados **GET** e **POST**:

```
//GET
```

```
var url = 'servico_get.php?par=1';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url);
```

```
xhttp.onload = function() {  
    //Resposta do servidor  
    var ret = xhttp.responseText;  
    console.log(ret);  
}
```

```
xhttp.send();
```

```
//POST
```

```
var dados = new FormData();  
dados.append("par", 1);
```

```
var url = 'servico_post.php';  
var xhttp = new XMLHttpRequest();  
xhttp.open('POST', url);
```

```
xhttp.onload = function() {  
    //Resposta do servidor  
    var ret = xhttp.responseText;  
    console.log(ret);  
}
```

```
xhttp.send(dados);
```

# JSON

- **JSON:**
  - *JavaScript Object Notation*
  - Formato de troca de dados e informações entre sistemas
    - Descreve **atributos** e **valores**
      - Formato: “chave”: valor
  - Exemplo:

```
{  
  "id":1,  
  "nome":"Fulano de Tal",  
  "endereco":"R. Qualquer"  
}
```

{JSON}

# Exemplo AJAX

- Requisição AJAX **recebendo** dados em JSON:

```
var url = 'pagina.php';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url);  
  
xhttp.onload = function() {  
    //Resposta do servidor  
    var objeto = JSON.parse(xhttp.responseText);  
}  
  
xhttp.send();
```

O retorno JSON  
(texto) foi convertido  
para um objeto  
JavaScript



# Exemplo AJAX

- Requisição AJAX **enviando** dados em JSON:

```
var objeto = {  
    "valor": 1  
}  
  
var url = 'pagina.php';  
var xhttp = new XMLHttpRequest();  
xhttp.open('POST', url);  
xhttp.setRequestHeader('Content-type', 'application/json');  
  
xhttp.onload = function() {  
    //Resposta do servidor  
    var retornoTexto = xhttp.responseText;  
    console.log(retornoTexto);  
}  
  
xhttp.send(JSON.stringify(objeto));
```

Objeto JavaScript  
convertido para  
envio na requisição  
como JSON (texto)

# API Fetch

- **API Fetch** permite a implementação de um acesso a um servidor Web utilizando AJAX
  - Também operacionaliza o AJAX, sendo a substituta da classe XMLHttpRequest
  - [https://www.w3schools.com/jsref/api\\_fetch.asp](https://www.w3schools.com/jsref/api_fetch.asp)
- Características da API Fetch:
  - Os dados resultantes do acesso ao servidor Web são encapsulados em um objeto **Promise**:
    - Valor que estará disponível em algum momento futuro
    - A partir da Promise, pode-se acessar o retorno da requisição ao servidor em um objeto **Response**

# API Fetch

Versão com  
XMLHttpRequest

- Exemplo com a API Fetch (busca):

Funções que executam requisições assíncronas devem ser marcadas como **async**

```
var url = 'servico.php';  
const options = {  
  method: 'GET'  
};
```

```
buscar();  
async function buscar() {  
  var resp = await fetch(url, options);  
  var texto = await resp.text();  
  console.log(texto);  
}
```

```
var url = 'servico.php';  
var xhttp = new XMLHttpRequest();  
xhttp.open('GET', url, true);  
  
xhttp.onload = function() {  
  var texto = xhttp.responseText;  
  console.log(texto);  
}  
  
xhttp.send();
```

resp: objeto **Response**

**await** é requerido pois os métodos **fetch()** e **text()** retornam uma **Promisse**

# API Fetch

Versão com  
XMLHttpRequest  
no slide 9

- Exemplo com a API Fetch (enviar dados):

```
const objetoJson = { "valor": 1 };

var url = 'pagina.php';
const options = {
  method: 'POST',
  headers: {
    'Content-type': 'application/json'
  },
  body: JSON.stringify(objetoJson)
};

enviar();
async function enviar() {
  var resp = await fetch(url, options);
  var json = await resp.text();
  console.log(json);
}
```

Cabeçalhos e corpo da  
requisição setados nas  
opções

Retorno no formato texto

# API Fetch: resposta JSON

- **API Fetch** permite acessar a resposta diretamente como um objeto JavaScript:
  - Não é necessário executar o `JSON.parse()`

```
var url = 'servico.php';
const options = {
  method: 'GET'
};

buscar();
async function buscar() {
  var resp = await fetch(url, options);
  var objeto = await resp.json();
  imprimeObjeto(objeto);
}
```

Retorno como objeto  
JavaScript

Chama uma função para  
utilizar o objeto JS

# API Fetch

- Links úteis sobre a API Fetch do JavaScript:
  - <https://javascript.info/fetch-api>
  - <https://www.devmedia.com.br/javascript-fetch/41206>
  - <https://dmitripavlutin.com/javascript-fetch-async-await>