O que é Ajax?

Ajax significa assíncrono javascript e xml e é um conjunto de tecnologias que manda dados do lado do cliente para o servidor assincronamente. Tudo é feito por trás sem precisar dar um F5 na página, ultimamente JSON tomou o lugar do xml por ser mais simples e mais fácil e também mais rápido.

O que é XHR ou XmlHttpRequest?

XmlHttpRequest é uma API em forma de objeto, os métodos são transferidos por dados entre cliente e servidor e também podem ser usados com outros protocolos sem ser Http e também pode trabalhar com outros tipos de dados sem ser xml (JSON, plain text).

Bibliotecas e outros métodos que podem chamar Ajax

• JQuery;

• Axios;

• Superagent;

• Fetch API;

• Prototype;

• Node Http

Servidor

•Primeiro instale um ambiente de servidor local nos eu computador para podermos testar a situação cliente/servidor. Recomendados XAMPP ou WampServer.

Crie a pasta e coloque ela em htdocs on XAMMP.

•XHR com Text Files

Sempre ligue o tomcat e o apache para executar o arquivo que iremos criar e para acessar ele depois de ligado digite localhost/nomedapasta/nomedoarquivo.html

Primeiro crie um arquivo em HTML e depois crie um botão para pegar o arquivo de texto. Depois crie um botão com o id = “button” e de um texto para o botão. Logo após fazer isso crie um <script>. E crie um event listener para o botão:

//Criar um event listener

document.getElementById('button').addEventListener('click', loadText);

No final do event listener informamos uma função que iremos criar no caso a citada acima foi a “loadText” logo após o event listener crie a função acima:

function loadText(){

}

Agora iremos criar o objeto xhr para receber o XmlHttpRequest isso dentro da função loadText:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

}

Depois iremos criar um arquivo do tipo txt dentro da pasta onde está criando o projeto com o nome simple.txt e depois iremos colocar lorem50 dentro dele.

Depois usaremos a função do xhr Open que informa o tipo de requisição, o arquivo/API e se é assíncrono ou não:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

}

Primeiro iremos usar a forma atual do xhr usando o onload e depois iremos criar uma estrutura de seleção para confirmar se o arquivo do xhr.open funciona:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

/Usando onload forma atual

xhr.onload = function(){

if(this.status == 200){

console.log(this.responseText)

}

}

}

Para ver se está funcionando clicamos no botão e vemos no console se ele mostra o resultado. Usamos o this para referenciar o xhr mas podemos usar o xhr no lugar do this que também funciona. Usamos o status para descobrir se o arquivo também está funcionando usamos o HTTP Status no if(this.status == 200) ,para lembrar os códigos do Http Status:

HTTP Status

200: "OK"

304: "Not Modified"

403: "Forbidden"

404: "Not Found"

E no final usamos o método send() para mandar a solicitação para o servidor.

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

/Usando onload forma atual

xhr.onload = function(){

if(this.status == 200){

console.log(this.responseText)

}

}

//Mandar a solicitação

xhr.send()

}

Depois disso clique no botão e teste se está funcionando. Para olhar o status clicamos em console na ferramenta do desenvolvedor e depois em network atualize a página e clique no botão, vemos que apareceu o arquivo e nele temos o preview que mostra o conteúdo e do lado o headers que mostra o status do arquivo.

E no final do onload criar uma forma de identificar se o xhr tem um erro usando onerror:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

/Usando onload forma atual

xhr.onload = function(){

if(this.status == 200){

console.log(this.responseText)

}

}

xhr.onerror = function(){

console.log('Request Error...')

}

//Mandar a solicitação

xhr.send()

}

Outra forma de usar o xhr mas de uma forma mais antiga é usando o onreadystatechange, para checar os valores usamos essa tabelinha de status:

readyState Values

0: request not inicialized (requisição não iniciada)

1: server connection estabilished (conexão com o servidor estabilizada)

2: request received (requisição recebida)

3: processing request (processando requisição)

4: request finished and response is ready (requisição terminada e a resposta esta pronta)

Para criar um método onreadystatechange fazemos igual o onload porém usando o método onreadystatechange e para testar se o botão está funcionando criamos uma estrutura de seleção e usamos this.readyState == 4 para testar se está pronto:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

/Usando onload forma atual

xhr.onload = function(){

if(this.status == 200){

console.log(this.responseText)

}

}

xhr.onerror = function(){

console.log('Request Error...')

}

//Usando o onreadystatechange forma antiga

//xhr.onreadystatechange = function(){

// console.log('ReadyState' , xhr.readyState)

// if(this.readyState == 4 && this.status == ///200){

//

// //console.log(this.responseText)

// }

//}

//Mandar a solicitação

xhr.send()

}

Criamos um console.log() escrito ReadyState e usamos a propriedade xhr.readyState para nos informar o tipo statusValue.

A diferença entre os dois processos é que o onload só irá funcionar quando ele estiver pronto enquanto o onready irá mostrar todo o status do processo.

Outra forma que podemos usar mas é opcional e usado para loaders é o onprogress:

function loadText(){

//Criar um objeto xhr

const xhr = new XMLHttpRequest();

//OPEN -type, url/file , async

xhr.open('GET','simple.txt', true)

/Usando onload forma atual

xhr.onload = function(){

if(this.status == 200){

console.log(this.responseText)

}

}

xhr.onerror = function(){

console.log('Request Error...')

}

//Usando o onreadystatechange forma antiga

//xhr.onreadystatechange = function(){

// console.log('ReadyState' , xhr.readyState)

// if(this.readyState == 4 && this.status == ///200){

//

// //console.log(this.responseText)

// }

//}

//xhr.onprogress = function(){

//console.log('ReadyState' , xhr.readyState)

//}

//Mandar a solicitação

xhr.send()

}