



Referências

<https://app.rocketseat.com.br/node/guia-estelar-de-http/group/entendendo/lesson/visualizando-a-comunicação>

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP>


AJAX e jQuery com AJAX

Professor Rafael Escalfoni

Requisições HTTP e JQuery AJAX

1. AJAX
2. JQuery AJAX



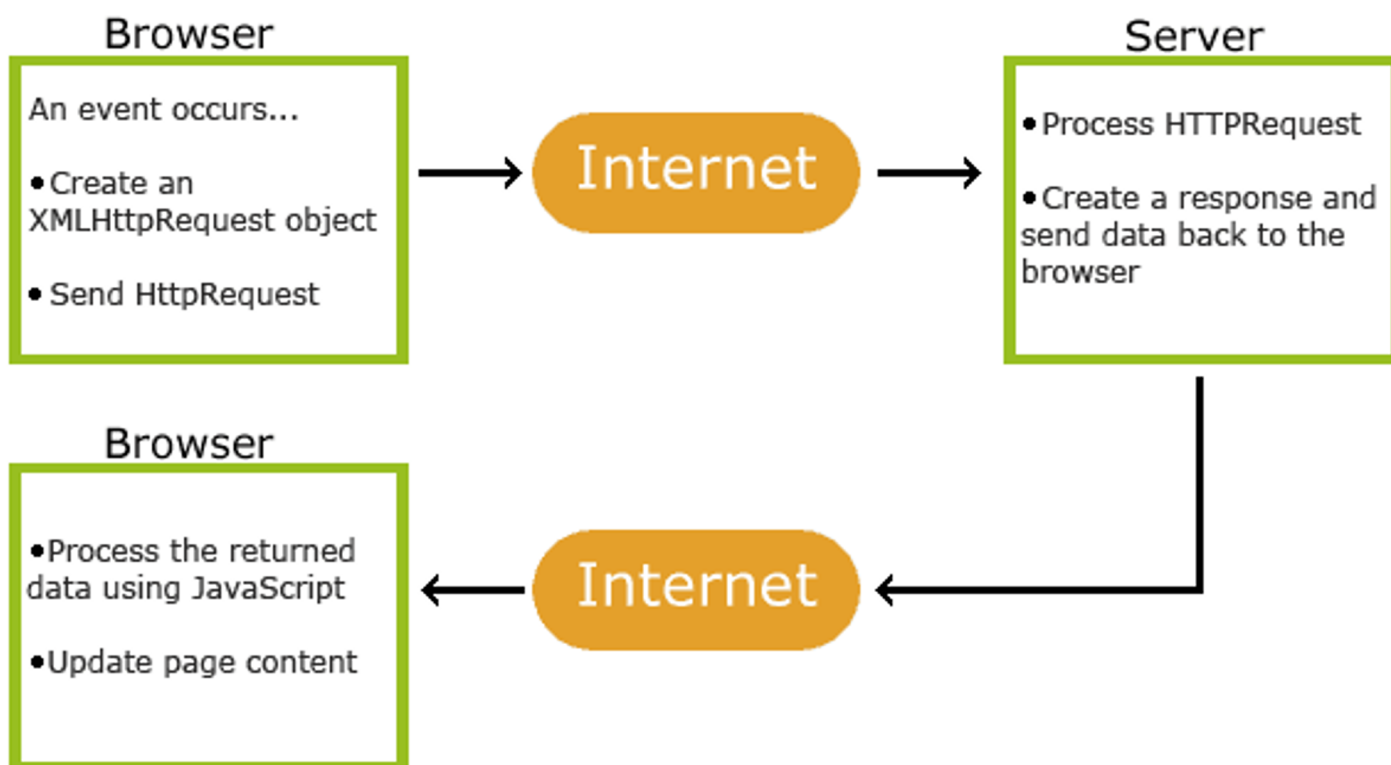


1

AJAX

AJAX

- Como AJAX é possível
 - Atualizar uma página web sem recarregar a página
 - Requisitar dados ao servidor
 - Receber dados do servidor
 - Enviar dados para servidor
- AJAX não é uma linguagem de programação
- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML
- AJAX é uma combinação de
 - Objeto XMLHttpRequest para requisitar dado do servidor
 - JavaScript e HTML DOM para mostrar o dado
- Embora pareça que AJAX só pode transmitir XML, ele também pode transportar texto pleno ou JSON



Como AJAX funciona?

1. Um evento é disparado em uma página (um botão clicado, formulário submetido etc)
2. Um objeto XMLHttpRequest é criado através do JavaScript
3. O objeto XMLHttpRequest envia uma requisição para um servidor web
4. O servidor processar a requisição
5. O servidor envia a resposta de volta à página
6. A resposta é lida pelo JavaScript
7. Uma ação (como atualização da página) é executada pelo JavaScript

Objeto XMLHttpRequest

- Todos os navegadores modernos suportam XMLHttpRequest
- O objeto XMLHttpRequest é usado para trocar dados com o servidor
- Protocolo HTTP
- Criar objeto XMLHttpRequest
 - `var xhttp = new XMLHttpRequest();`
- Fazer requisição
 - `xhttp.open("GET", "ajax_info.txt", true);`
 - `xhttp.send();`

Métodos do objeto XMLHttpRequest

Method	Description
<code>open(method, url, async)</code>	<p>Specifies the type of request</p> <p><i>method</i>: the type of request: GET or POST <i>url</i>: the server (file) location <i>async</i>: true (asynchronous) or false (synchronous)</p>
<code>send()</code>	Sends the request to the server (used for GET)
<code>send(string)</code>	Sends the request to the server (used for POST)

Method	Description
<code>setRequestHeader(header, value)</code>	<p>Adds HTTP headers to the request</p> <p><i>header</i>: specifies the header name <i>value</i>: specifies the header value</p>

Exemplo Requisições

```
// Requisição GET
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- É possível modificar esse código para:
 - fazer uma requisição post
 - xhttp.open("POST", "ajax_test.asp", true);
 - adicionar cabeçalho
 - xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
 - adicionar dados no corpo da mensagem
 - xhttp.send("fname=Henry&lname=Ford");

Resposta do Servidor

- A propriedade **readyState** representa o status do objeto XMLHttpRequest
- A propriedade **onreadystatechange** define a função a ser executada com o **readyState** mudar
- A propriedade **status** e **statusText** contém o código de retorno da resposta e o texto associado ao status do retorno respectivamente

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Resposta do Servidor

Propriedades para recuperar corpo da mensagem

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Métodos para recuperar cabeçalho da resposta

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource



2

AJAX com JQuery

AJAX

- Método **ajax()** para fazer requisições HTTP
 - Recebe um **objeto** como parâmetro
 - Esse objeto contém uma coleção de pares chave/valor especificando os parâmetros da requisição HTTP
 - Sintaxe: `$.ajax({name:value, name:value, ... })`

AJAX

Exemplo GET:

```
$.ajax({  
  type: 'GET',  
  url: 'http://rest.learncode.academy/api/johnbob/friends',  
  success: function(data) {  
    console.log("I have friends!", data); //returns all of johnbob's friends  
  }  
});
```

- **type:** especifica o tipo da requisição
- **url:** especifica a url de destino da requisição
- **success:** recebe um função de callback que é executada quando a requisição termina com sucesso. Essa função recebe como parâmetro o dados contidos no corpo da resposta

AJAX

Exemplo POST:

```
$.ajax({  
  type: 'POST',  
  url: 'http://rest.learncode.academy/api/johnbob/friends',  
  data: {name: 'Billy Bob', age: 27},  
  success: function(data) {  
    console.log("Friend added!", data); //the new item is returned with an ID  
  }  
});
```

- **data:** define os dados enviados

\$.get()

Requisita dados de um servidor

```
$.get(URL, callback);
```

- Exemplo

```
$.get("demo_test.asp", function(data, status) {  
    alert("Data: " + data + "\nStatus: " + status);  
});
```

- `done()` – caso o get tenha sucesso
- `fail()` – caso de falha

```
$.get(servico, parametros)  
    .done(callback)  
    .fail(callback);
```


\$.post

Para submeter formulários

\$.get(URL, data, *callback*);

- URL – obrigatório; especifica a URL para requisição.
- data – opcional; especifica os dados a serem enviados.
- callback – opcional; função a ser executada se a requisição tiver sucesso.

```
$.post("demo_test_post.asp"  
  , {  
    name: "Donald Duck",  
    city: "Duckburg"  
  }  
  , function(data, status) {  
    alert("Data: " + data + "\nStatus: " + status);  
  }  
);
```

Juntando JSON e AJAX

`$.getJSON(URL)` – similar ao `$.get()`, porém retorna json.