Linguagens de Marcação e Scripts

—

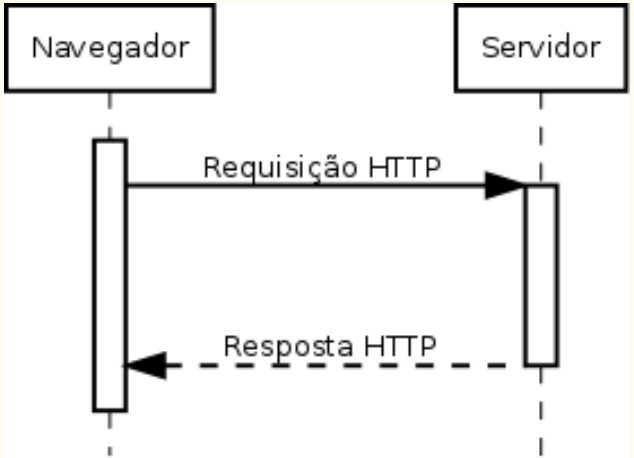
Prof. Aníbal Cavalcante de Oliveira
UFC - QXD0164 - 2019.1

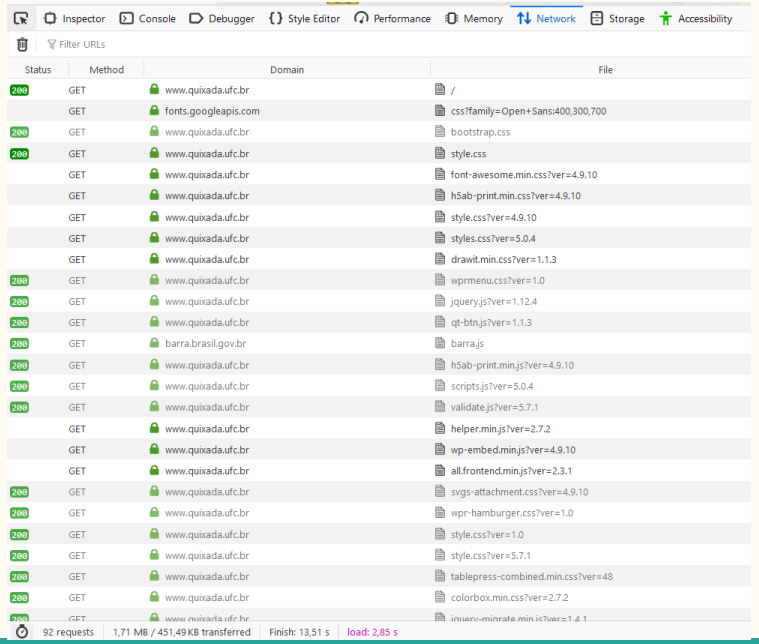
Agenda - Aula 19

- Introdução ao AJAX (**A**synchronous **J**avaScript And **X**ML)

O problema da Web Tradicional

Como foi visto aula passada, para carregar uma página completa, o **cliente** faz um série de requisições para o **servidor** utilizando métodos do protocolo **HTTP**, como o **método GET** e o método **POST**. Para cada requisição o servidor devolve uma resposta com o recurso solicitado, por exemplo: vários arquivos em html, css, javascript, imagens e etc...



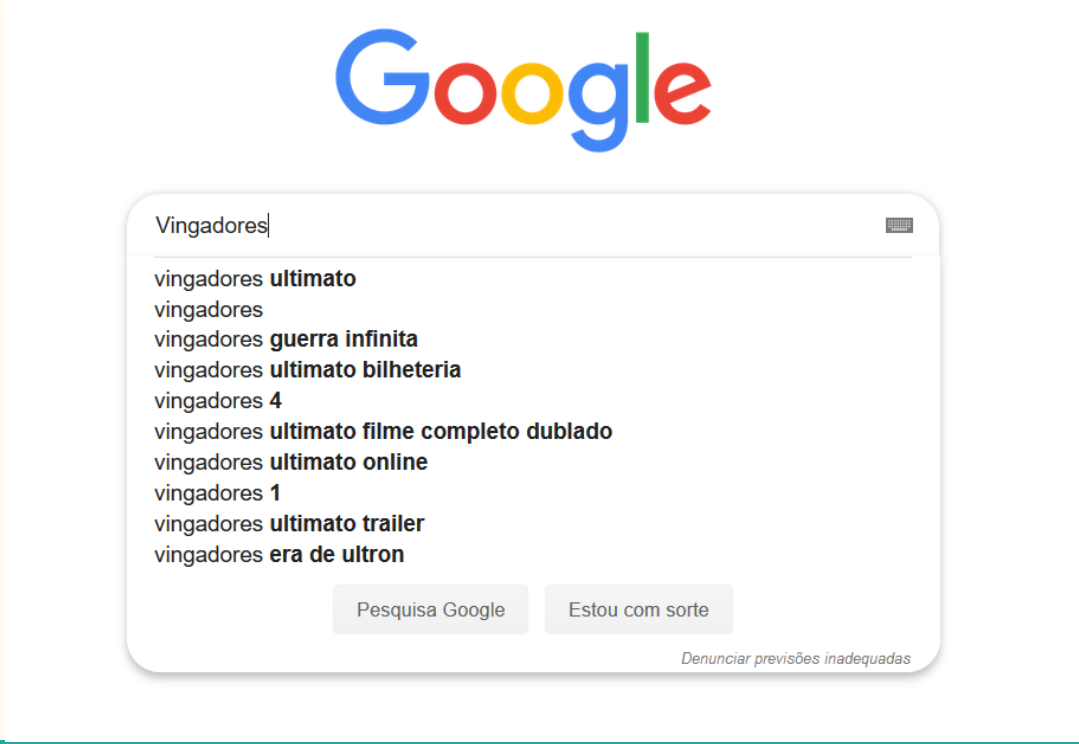


O problema da Web Tradicional

E se o usuário não precisa carregar todo o conteúdo da página, mas apenas uma parte dela, a parte com a qual ele está interagindo? Seria eficiente recarregar todo o conteúdo da página só para alterar algumas poucas informações dela?

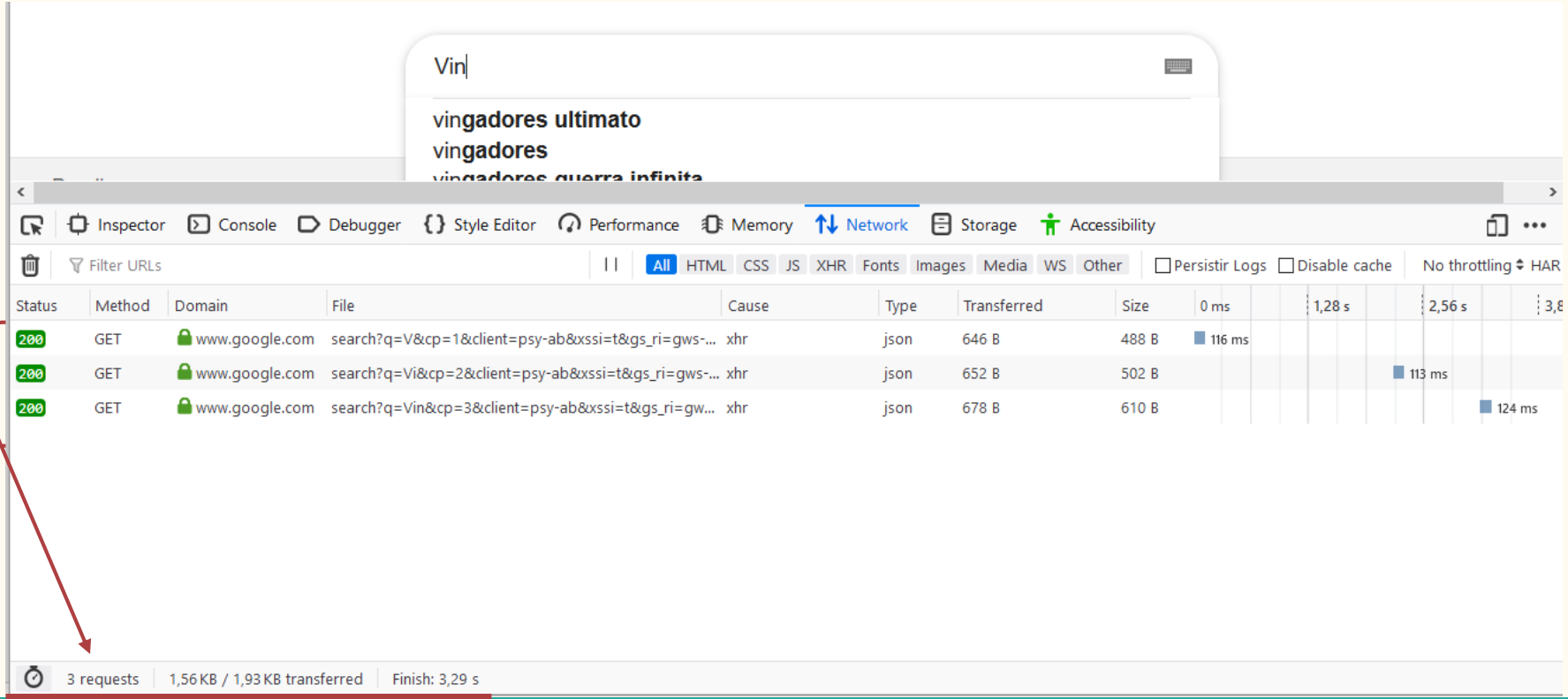
Vou te contar um segredo: “Você já usa AJAX mesmo sem saber....”

Um exemplo clássico acontece quando estamos digitando algum termo de busca no **Google.** A página inteira é carregada apenas uma vez, porém a medida que vamos digitando novos caracteres, o serviço do Google nos sugere novos termos de busca....



Vamos fazer um teste

Abra o navegador na página do Google, aperte a tecla F12, e selecione a opção **Network.** Veja o que acontece quando você digita um letra no teclado....



O que é AJAX

A palavra AJAX é o acrônimo de **A**synchronous **J**avaScript and **X**ML

AJAX é um conjunto de tecnologias que basicamente nos permitem:

1 – Carregar novos dados do Servidor Web, mesmo que a página já tenha sido totalmente carregada.

2 – Atualizar apenas a parte da página (User Interface) que o usuário está interagindo.

3 – Realizar requisições e receber respostas do Servidor Web em background, nos bastidores. Isto significa que é possível atualizar partes de uma página web, sem recarregar a página inteira.

Como o AJAX funciona (Fluxo Convencional)

Modelo convencional

1. Uma requisição HTTP é enviada do navegador para o servidor.
2. O servidor recebe a requisição e busca os dados.
3. O servidor envia os dados requisitados para o navegador.
4. O navegador recebe os dados e recarrega a página para fazer com que apareçam.

Durante esse processo, os usuários não conseguem fazer nada além de esperar que seja concluído. Além de ser uma perda de tempo para o usuário, também demanda muitos recursos do servidor.

Como o AJAX funciona

Modelo AJAX

1. O navegador gera uma chamada do JavaScript que então ativa o objeto JAVASCRIPT responsável por realizar uma requisição AJAX.
2. Em segundo plano o navegador cria uma requisição HTTP para o servidor.
3. O servidor recebe a requisição, busca os dados e envia para o navegador.
4. O navegador recebe os dados requisitados que irão aparecer imediatamente na página. Não é necessário recarregar.

O que é AJAX

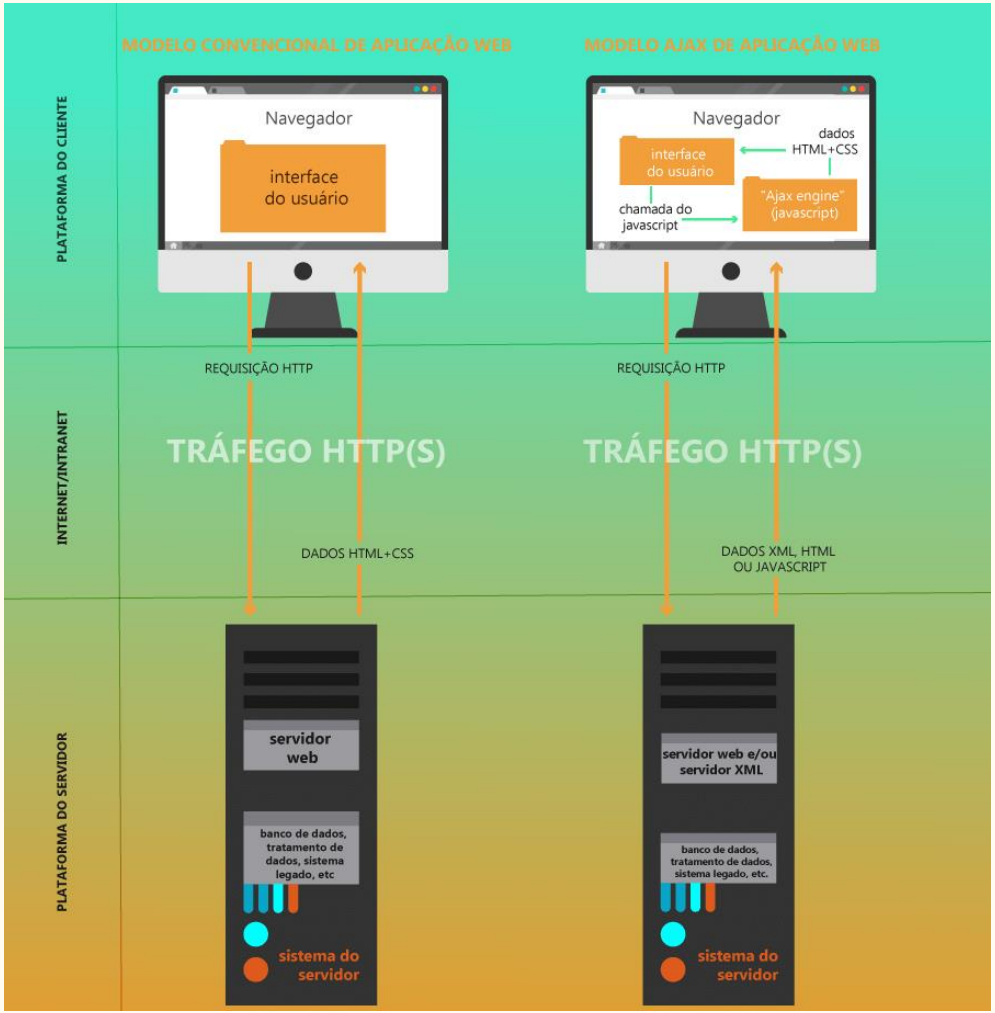
Antes de explicar o que é AJAX, deve-se explicar o que ele não é.

1- AJAX não é um framework, uma API nem uma tecnologia em si.

2- AJAX é uma funcionalidade implementada por um conjunto de objetos de **JavaScript**, sendo o mais importante chamado de: **XMLHttpRequest**.

Este objeto, o **XMLHttpRequest**, trata uma requisição ou resposta de servidor com um documento DOM, e contém uma série de funções que possibilita que o browser possa realizar requisições e receber respostas do servidor sem que este tenha que atualizar toda a tela (o chamado refresh).

O que é AJAX


MODELO CONVENCIONAL DE APLICAÇÃO WEB
MODELO AJAX DE APLICAÇÃO WEB
PLATAFORMA DO CLIENTE
INTERNET/INTRANET
PLATAFORMA DO SERVIDOR
Navegador
interface do usuário
Navegador
interface do usuário
dados HTML+CSS
chamada do javascript
"Ajax engine" (javascript)
REQUISIÇÃO HTTP
REQUISIÇÃO HTTP
TRÁFEGO HTTP(S)
TRÁFEGO HTTP(S)
DADOS HTML+CSS
DADOS XML, HTML OU JAVASCRIPT
servidor web
banco de dados, tratamento de dados, sistema legado, etc
sistema do servidor
servidor web e/ou servidor XML
banco de dados, tratamento de dados, sistema legado, etc.
sistema do servidor

AJAX – JSON ou XML

Com o AJAX, o formato dos dados que são trocados nas mensagens entre o **cliente** e **servidor** pode ser dois:

1 – Arquivos no formato XML (forma mais antiga, arquivos com extensão **.xml**);

2- Arquivos no formato em JSON, ou objeto JavaScript (forma mais utilizada, pela melhor legibilidade do arquivo, de extensão **.json**);

Hoje o mercado adotou o formato JSON, veremos a seguir a vantagens de utilizarmos arquivos em JSON.

AJAX – JSON ou XML

Exemplo de **dado no formato XML** para representar um catálogo de discos:

```xml
<CATALOG>
        <CD>
                <TITLE>Empire Burlesque</TITLE>
                <ARTIST>Bob Dylan</ARTIST>
                <COUNTRY>USA</COUNTRY>
                <COMPANY>Columbia</COMPANY>
                <PRICE>10.90</PRICE>
                <YEAR>1985</YEAR>
        </CD>
        <CD>
                <TITLE>Hide your heart</TITLE>
                <ARTIST>Bonnie Tyler</ARTIST>
                <COUNTRY>UK</COUNTRY>
                <COMPANY>CBS Records</COMPANY>
                <PRICE>9.90</PRICE>
                <YEAR>1988</YEAR>
        </CD>
</CATALOG>
```

AJAX – JSON ou XML

Exemplo de **dado no formato JSON** para representar um catálogo de discos:

```
{
"CATALOG": {
            "CD": [
                        {
                                    "TITLE": "Empire Burlesque",
                                    "ARTIST": "Bob Dylan",
                                    "COUNTRY": "USA",
                                    "COMPANY": "Columbia",
                                    "PRICE": "10.90",
                                    "YEAR": "1985“
                        },
                        {
                                    "TITLE": "Hide your heart",
                                    "ARTIST": "Bonnie Tyler",
                                    "COUNTRY": "UK",
                                    "COMPANY": "CBS Records",
                                    "PRICE": "9.90",
                                    "YEAR": "1988“
                        }
                ]
        }
}
```

JSON – Java **S**cript **O**bject **N**otation

JSON é um **texto**, escrito com notação de objeto **JavaScript**.

Objetos em JSON sempre estão no formato **atributo-valor**.

Representa as informações de forma mais **compacta**;

É **simples**, de **fácil leitura e compreensão**. A maioria das linguagens de programação possuem **funções para tratar** objetos em JSON.

A simplicidade de JSON ajudou a popularizar seu uso, especialmente como uma alternativa para XML em **AJAX**.

JSON

```
{ "name":"John" }
```

Tipos de dados de um objeto escrito em JSON

1 - **Number**: um número que pode ter sinal, uma parte fracionária separada por um ponto (.) e eventualmente usar a notação E exponencial.

2 - **String:** uma cadeia de zero ou mais caracteres Unicode. Strings são delimitados por aspas duplas (") e suportam a barra inversa (\) como caractere de escape. Por exemplo: "Fácil como fazer um \"Alô mundo\" em JS."

3 - **Array**: uma lista ordenada de zero ou mais valores, cada um podendo ser de qualquer tipo. Arrays são delimitados por colchetes ([]), dentro dos quais ficam os valores, também conhecidos como elementos, separados por vírgulas. O primeiro elemento é o de índice 0.

4 – **Booleano**: Valores booleanos com **true** ou **false** podem ser passados sem aspas duplas. Por exemplo: { "vendido":true }

5 - **null**: Valor vazio ou nulo, para representar que a informação não existe.

JSON – Java Script Object Notation

Podemos ter encadeamento de objetos através de Arrays, como mostrado abaixo:

Exemplo de uma lista de alunos com suas notas

```
1 {"Alunos":[
2       { "nome": "Edson Sales Arantes", "notas": [ 8, 9, 5 ]  },
3       { "nome": "Luiz Livelli ", "notas": [ 8, 10, 7 ] },
4       { "nome": "Caique Caicedo De Plata", "notas": [ 10, 10, 9 ] }
5 ]}
```

JSON – Java Script Object Notation

Exemplo de uma representação de um menu

```
{"menu":{
    "id": "file",
    "value": "File",
    "popup":{
       "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
      ]
    }
}}
```

Usando JSON no JavaScript

Atualmente todos os principais navegadores suportam no mínimo a quinta edição do ECMAScript que provê uma forma segura e rápida de codificar e decodificar JSON:

A função **JSON.parse()** permite transformar texto JSON em objetos JavaScript.

```
<html>
<body>
        <h2>Criando Objeto de um JSON</h2>
        <p id="demo"></p>
<script>
        var txt = '{"name":"John", "age":30, "city":"New York"}';

        var obj = JSON.parse(txt);
        document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>
</body>
</html>
```

Usando JSON no JavaScript

Podemos transformar um objeto JavaScript em um texto no formato JSON.

Usamos a função JavaScript **JSON.stringify()** para convertê-lo em uma String JSON.

```
<html>
<body>
        <h2>Criando uma String em JSON de um objeto em JavaScript</h2>
        <p id="demo"></p>

<script>
        var obj = { name: "John", age: 30, city: "New York" };

        var myJSON = JSON.stringify(obj);
        document.getElementById("demo").innerHTML = myJSON;
</script>
</body>
</html>
```

AJAX - O objeto XMLHttpRequest

A pedra angular da AJAX é o objeto **XMLHttpRequest**.

Todos os navegadores modernos suportam o objeto **XMLHttpRequest**.

O objeto **XMLHttpRequest** pode ser usado para trocar dados com um servidor web nos bastidores. Isto significa que é possível atualizar partes de uma página web, sem recarregar a página inteira.

Exemplo

```
var xhttp = new XMLHttpRequest();
```

Métodos objeto XMLHttpRequest

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Propriedades do objeto XMLHttpRequest

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
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Primeira requisição AJAX

Vamos fazer nossa primeira requisição AJAX e obter um texto de resposta.

```
  <h2>O Objeto XMLHttpRequest</h2>
  <button type="button" onclick="loadDoc()">Request data</button>
  <p id="demo"></p>
  <script>
          function loadDoc() {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
              if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML = this.responseText;
              }
            };
            xhttp.open("GET", "https://lmsqxd.herokuapp.com/usuarios");
            xhttp.send(null);
          }
</script>
```

Primeira requisição AJAX

Agora com a função JSON.parse (a resposta é um Array de objetos)

```
  <h2>O Objeto XMLHttpRequest</h2>
  <button type="button" onclick="loadDoc()">Request data</button>
  <p id="demo"></p>
  <script>
          function loadDoc() {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
               if (this.readyState == 4 && this.status == 200) {
                  var pessoa = JSON.parse(this.responseText);
                  var p = document.getElementById("demo");
                  p.innerHTML = pessoa[0].nome + ", " +
                                pessoa[0].email + ", " +
                                pessoa[0].nascimento;
               }
            };
            xhttp.open("GET", "https://lmsqxd.herokuapp.com/usuarios");
            xhttp.send(null);
          }
</script>
```