Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCA UFES
Departamento de Computação

Ajax

Desenvolvimento de Sistemas para WEB
Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

O que é Ajax?

Asynchronous **J**avaScript **a**nd **X**ML

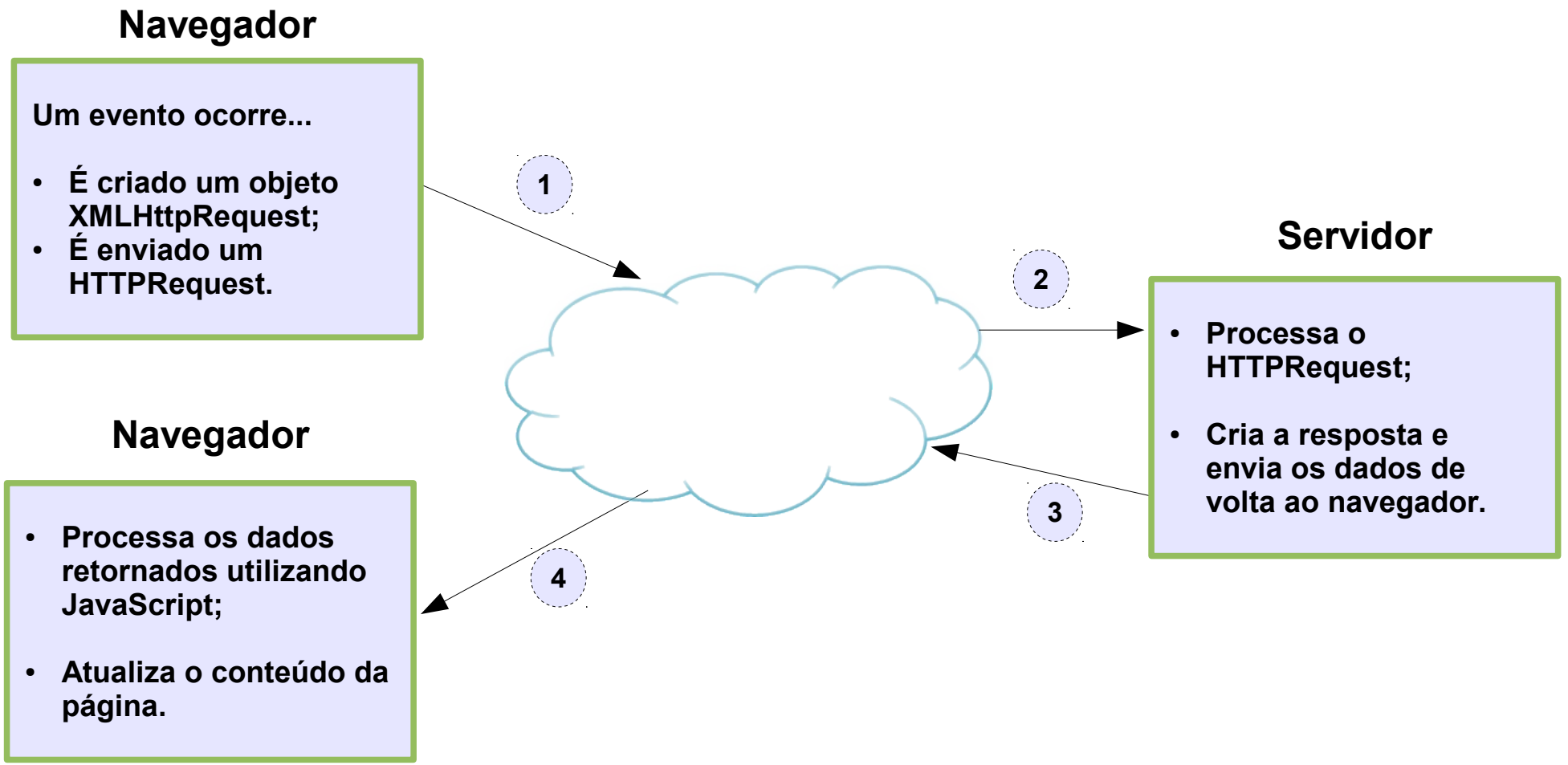
(JavaScript Assíncrono e XML)

- É uma técnica para criar páginas web fáceis e dinâmicas;
- Permite que páginas web sejam atualizadas de forma assíncrona através da troca de pequenas porções de dados com o servidor.
- Permite atualizar partes da página Web sem ser necessário recarregar toda a página.
- Páginas clássicas da Web (sem Ajax) devem recarregar toda a página caso o conteúdo deva ser modificado.
- Exemplos de aplicações Web com Ajax: Google Maps, Gmail, Youtube, and Facebook tabs.

Ajax

- Antes, para atualizar algo da página, por menor que fosse, era necessário atualizar toda a página.
- Assim, cada clique do usuário fazia o navegador carregar uma nova página inteira, mesmo o que já tinha na página antiga.
- Mas:
  - O site não responde muito rápido;
  - A conexão as vezes é lenta;
- O Ajax permite que seu documento HTML acesse outras páginas ou arquivos por “baixo dos panos”, trazendo informações atualizadas sem a necessidade de recarregar toda uma página.
- Com a manipulação de DOM, que permite que porções do documento sejam atualizados, é possível atualizar a página utilizando o Ajax.

Como o Ajax funciona



Ajax e os padrões da Internet

- Ajax baseia-se nos padrões da Internet, utilizando a combinação de:
  - Objeto XMLHttpRequest (para trocar os dados de forma assíncrona com o servidor)
  - JavaScript/DOM (para exibir e interagir com a informação)
  - CSS (para definir um estilo aos dados)
  - XML (geralmente utilizado como formato para transferir os dados)

- Aplicações em Ajax são independentes de navegadores e plataformas.

Primeiro Teste!

```
<html>
 <head>
   <script>
      function getText() {
         aux = new XMLHttpRequest();
         aux.open("GET","texto.txt",false);
         aux.send();
         texto.innerHTML = aux.responseText;
      }
   </script>
 </head>
 <body>
    <div id="texto"></div>
    <input type="button" onclick="getText()" value="Ajax" />
 </body>
</html>
```

Entendendo o XMLHttpRequest

- “Requisição de XML por HTTP”:
  - Fornece um meio de solicitar dados ao servidor via HTTP;
  - As requisições podem ser assíncronas, não interferindo na usabilidade do site;
  - Ele permite que o Ajax funcione, por isso é considerado o “Coração do Ajax”;
  - É uma API (Application Programming Interface) que pode ser utilizada por linguagens de script;
  - Permite então uma conexão independente entre o servidor e a página cliente (server-side e client-side);
  - Na maioria das vezes, utiliza-se o retorno XML, mas também pode retornar um texto puro, sem formatação.

XMLHttpRequest

- Inicialmente desenvolvido pela Microsoft, como parte do Outlook Access 2000.
  - Batizada como XMLHTTP, foi inserida no Internet Explorer 5.0, como um objeto ActiveX.
- Em 2002, o Mozilla 1.0 incorporou a primeira implementação compatível: XMLHttpRequest.
- Depois, a tecnologia foi implementada no Opera, Safari, Konqueror e outros navegadores;
- Em abril de 2006, o W3C publicou o primeiro "rascunho" da especificação do objeto XMLHttpRequest;

Utilizando o XMLHttpRequest

- O modo de utilizar o XMLHttpRequest depende do navegador que o usuário estiver utilizando:
  - Internet Explorer 6 e 5:
    - **`ajax = new ActiveXObject(“Microsoft.XMLHTTP”);`**
  - IE7+, Firefox, Chrome, Opera, Safari:
    - **`ajax = new XMLHttpRequest;`**

- Exemplo de código:

```
var ajax;
if (window.XMLHttpRequest) ajax = new XMLHttpRequest();
else ajax = new ActiveXObject("Microsoft.XMLHTTP");
```

Principais métodos e propriedades

- Métodos:
  - abort;
  - getAllResponseHeaders;
  - getResponseHeader;
  - open;
  - send;
  - setRequestHeader.
- Propriedades:
  - readyState;
  - status;
  - statusText;
  - responseText;
  - responseXML;
  - onreadystatechange;

Métodos

- abort:
  - Cancela a requisição corrente: ajax.abort()
- getAllResponseHeaders:
  - Retorna todos os cabeçalhos HTTP como string: ajax.getAllResponseHeaders()
- getResponseHeader:
  - Retorna o valor de um cabeçalho especificado: ajax.getResponseHeader('Content-Type')

Métodos

- open:
  - Define a URL a ser aberta e suas opções:

    open(método, URL, assinc, usuário, senha)
    - Método: GET, POST, PUT, DELETE ou HEAD;
    - URL: endereço a ser acessado, completo ou relativo;
    - Assinc: define se a requisição é assíncrona ou não;
    - Usuário: nome de usuário, se a página é restrita;
    - Senha: senha de acesso, se a página é restrita.
  - Exemplo:

    ajax.open('GET', 'programa.php', true);

Métodos

- send:
  - Envia a requisição ao servidor Web:

    ajax.open('GET', 'programa.php', true);

    ajax.send();
- setRequestHeader:
  - Altera o valor de um cabeçalho HTTP:

    ajax.setRequestHeader(

    'Content-type' , 'application/x-www-form-urlencoded')

Enviando informações com POST

- Para enviar dados com o método POST, deve-se adicionar o seguinte cabeçalho HTTP:

```
ajax.open( "POST" , "getPost.php", true );
ajax.setRequestHeader(
   "Content-type",
   "application/x-www-form-urlencoded" );
```

- E especificar os dados a serem enviados com o método send:

```
ajax.send( "nome=Jacson&idade=27" );
```

Propriedades

- readyState:
  - Estado do objeto XMLHttpRequest:

```
if (ajax.readyState == 4)
    alert( "Ajax carregado" );
```

  - Estados possíveis:
    - 0 → não inicializado;
    - 1 → aberto;
    - 2 → enviado;
    - 3 → recebendo;
    - 4 → carregado.

Propriedades

- status:
  - Código de resposta HTTP:
  -
```
if (ajax.readyState == 4)
  if (ajax.status = 200)
    alert('Documento recebido com sucesso!');
```

  - Códigos mais comuns:

    200 → OK; 202 → Accepted;

    400 → Bad Request; 403 → Forbidden;

    404 → Not Found.

Propriedades

- statusText:
  - Retorna o texto referente ao estado da solicitação:

    alert (“Status: ” + ajax.statusText);

  - Poderá apresentar uma das mensagens como status:

    “OK”, “Accepted”, “Bad Request”, “Forbidden”, “Not Found”.

Propriedades

- responseText:
  - Retorna a resposta do servidor como uma string:

    alert(ajax.responseText)

- responseXML:
  - Retorna a resposta do servidor como um objeto XML:

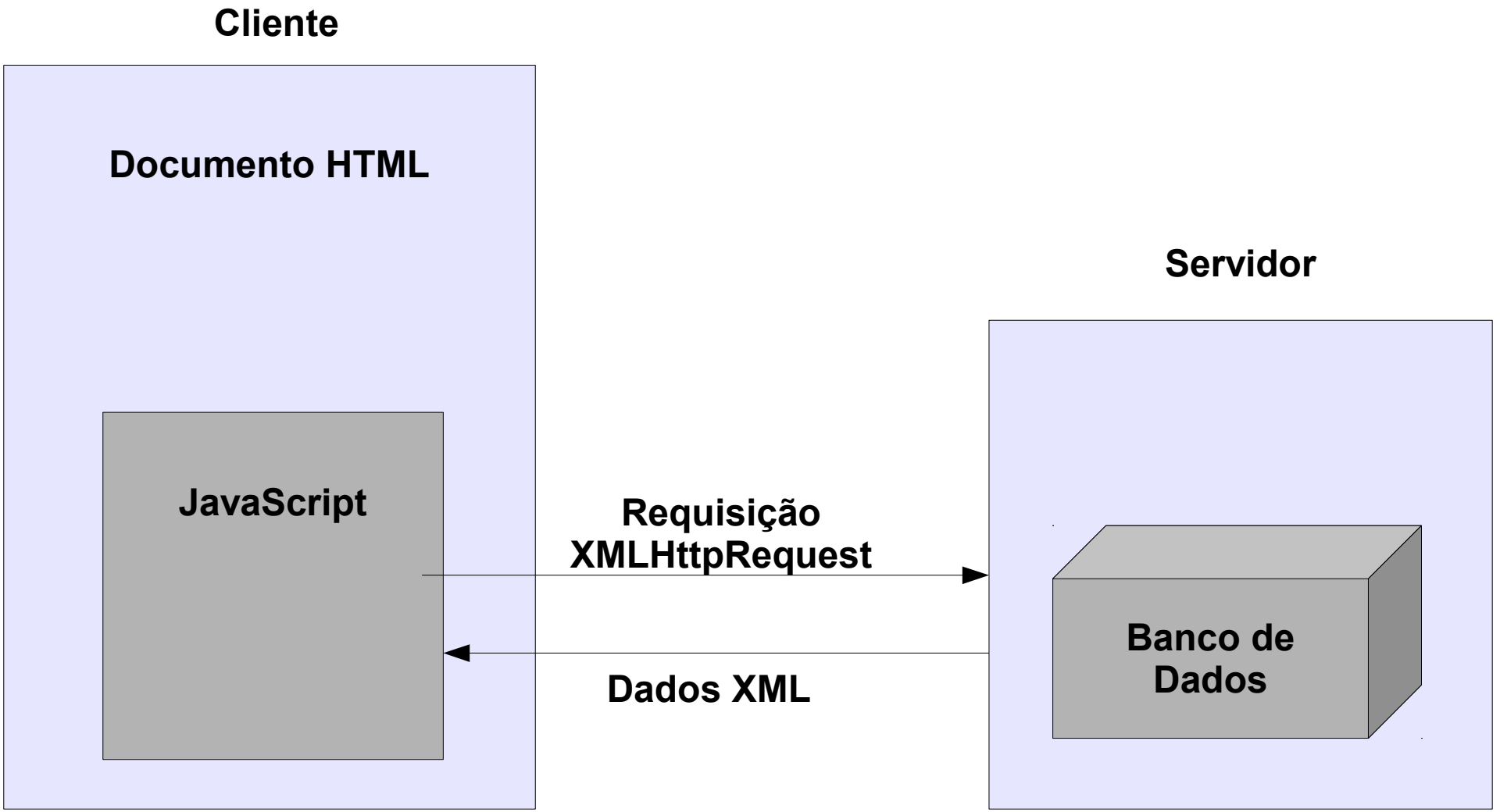
    var docXML = ajax.responseXML;
  - Então o mesmo pode ser processado pelo JavaScript para ser utilizado na página.

Propriedades

- onreadystatechange:
  - Define a função que irá manipular os eventos, sendo a mesma chamada a cada mudança de estado do Ajax:

```
ajax.open('GET' , 'produto.php?id='+produto, true);
ajax.onreadystatechange = function() {
 if (ajax.state == 4)                    // foi carregado
  if (ajax.status == 200)                // o status é “OK”
    alert( ajax.responseText );// exibe resposta
}
ajax.send();
```

Criando nosso Ajax



1º Passo: Iniciar um XMLHttpRequest

```
var ajax;

function iniciaAjax() {
    if (ajax) return;
    if (window.ActiveXObject)
        ajax = new ActiveXObject("Microsoft.XMLHTTP")
    else if (window.XMLHttpRequest)
        ajax = new XMLHttpRequest();
   else
        alert('Seu navegador não possui suporte a Ajax');
}
```

2º Passo: Criando o HTML para obter dados

```
<fieldset>
  <legend>Formulario Ajax</legend>
  <form>
    Nome: <input name='nome'  id='nome' type='text'  /> <br/>
    Valor:<input name='valor' id='valor' type='text' /> <br/>
    <input type='button' onclick='Processar()'
                          value='Processar...' /> <br/>
  </form>
</fieldset>
<fieldset>
   <legend>Resposta do Ajax:</legend>
   <div id='resposta' name='resposta'>Sem resposta.</div>
</fieldset>
```

3º Passo: Efetuando o pedido ao servidor

```
iniciaAjax(); //iniciando o Ajax
ajax.onreadystatechange = function() { //tratando a futura resposta
   if (ajax.readyState == 4 && ajax.status == 200)
       document.getElementById("resposta").innerHTML =
                                          ajax.responseText;
}
nome  = document.getElementById("nome").value;  // lendo nome
valor = document.getElementById("valor").value; // lendo valor
dados = 'nome='+ nome + '&valor=' + valor;  // formatando dados
ajax.open('POST', 'processar.php', true);   // iniciando envio
ajax.setRequestHeader(     //adicionando header de dados POST
       'Content-type','application/x-www-form-urlencoded' );
ajax.send( dados );
```

4º Passo: Respondendo ao cliente

Código-fonte do “processar.php”:

```
<?php
   $nome  = $_POST['nome'];
   $valor = $_POST['valor'];
   if ($nome == '' || $valor == '') {
      echo 'Dados Invalidos!';
      exit;
   }
   $quadrado = $valor * $valor;
   echo "$nome, ".
        "o servidor informou que o ".
        "quadrado de $valor é $quadrado";
?>
```

Ajax com jQuery

- O jQuery traz consigo funções para a utilização dos recursos do Ajax.
- Permite carregar e enviar informações.

Método Load

```
<h2>jQuery e AJAX são legais!!!</h2>
<p id="p1">Texto de um parágrafo.</p>
```

teste.txt

- Syntaxe:

  $(*seletor*).load(URL, dados, função de callback);

- Ele carrega os dados do servidor e os insere no seletor escolhido.
- Exemplos:

  ```
  $("#div1").load("teste.txt");
  ```

  Carrega o conteúdo de teste.txt e o insere em “div1”:

  ```
  <div id="div1"></div>
  ```

  ```
  $("#div1").load("teste.txt #p1");
  ```

  Carrega o conteúdo de teste.txt, mas filtra pelo “id=p1” e insere somente este texto no “div1”.

Método Load com Callback

- O callback do método load possui três parâmetros em sua função de callback:
  - responseText: conteúdo obtido do servidor;
  - statusTxt: status da chamada ao servidor;
  - xhr: contém o objeto XMLHttpRequest.
- Exemplo:

```
$("#div1").load("teste.txt",
    function(responseTxt, statusTxt, xhr){
       if( statusTxt=="success" )
         alert("O conteúdo foi carregado com sucesso!");
       if(statusTxt=="error")
         alert("Erro: "+xhr.status+": "+xhr.statusText);
    }
);
```

Método $.get e $.post

- Esses métodos são utilizados para requisitar informações ao servidor utilizando o GET e o POST do HTTP.
- Com esses métodos, não é necessário vincular a resposta a um elemento da página, como o load.
- Funcionalidades do GET e do POST:
  - GET: solicita dados de um recurso especificado;
  - POST: submete dados para serem processados por um recurso especificado.

$.get

- Syntaxe:

```
$.get(URL, [dados], função de callback);
```

- Exemplo:

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

$.post

- Syntaxe:

```
$.post(URL,dados do post,função do callback);
```

- Exemplo:

```
$.post( "teste.php",
    {
        nome:"Jeiks",
        cidade:"Alegre"
    },
    function( data, status ){
        alert("Dados: "+data+"\nStatus: "+status);
    }
);
```

Ajax e XML

- Uma forma de obter informações do servidor é através de arquivos XML.
- Com eles, torna-se possível criar uma estrutura para enviar os dados.
- Tanto o PHP quanto o JavaScript possuem ferramentas para trabalhar com XML.

Para aprender mais

- Algumas referências:

  <http://www.w3schools.com/jquery/jquery_ajax_intro.asp>

  <https://api.jquery.com/jQuery.ajax>

  <http://www.w3schools.com/jquery/jquery_ref_ajax.asp>

  <http://www.w3schools.com/jquery/ajax_get.asp>

  <http://www.w3schools.com/jquery/ajax_post.asp>