

Universidade Federal do ABC

MC0037 – Programação para Web

Aula 07: AJAX e jQuery



Introdução

➤ Aulas passadas

- POO
- Banco de Dados
- Hibernate
- MVC – SpringMVC
- Automatização e configuração – SpringBoot
- Front-end
 - HTML (estrutura das páginas)
 - JSP (algumas interações com back-end)
 - CSS (layout)



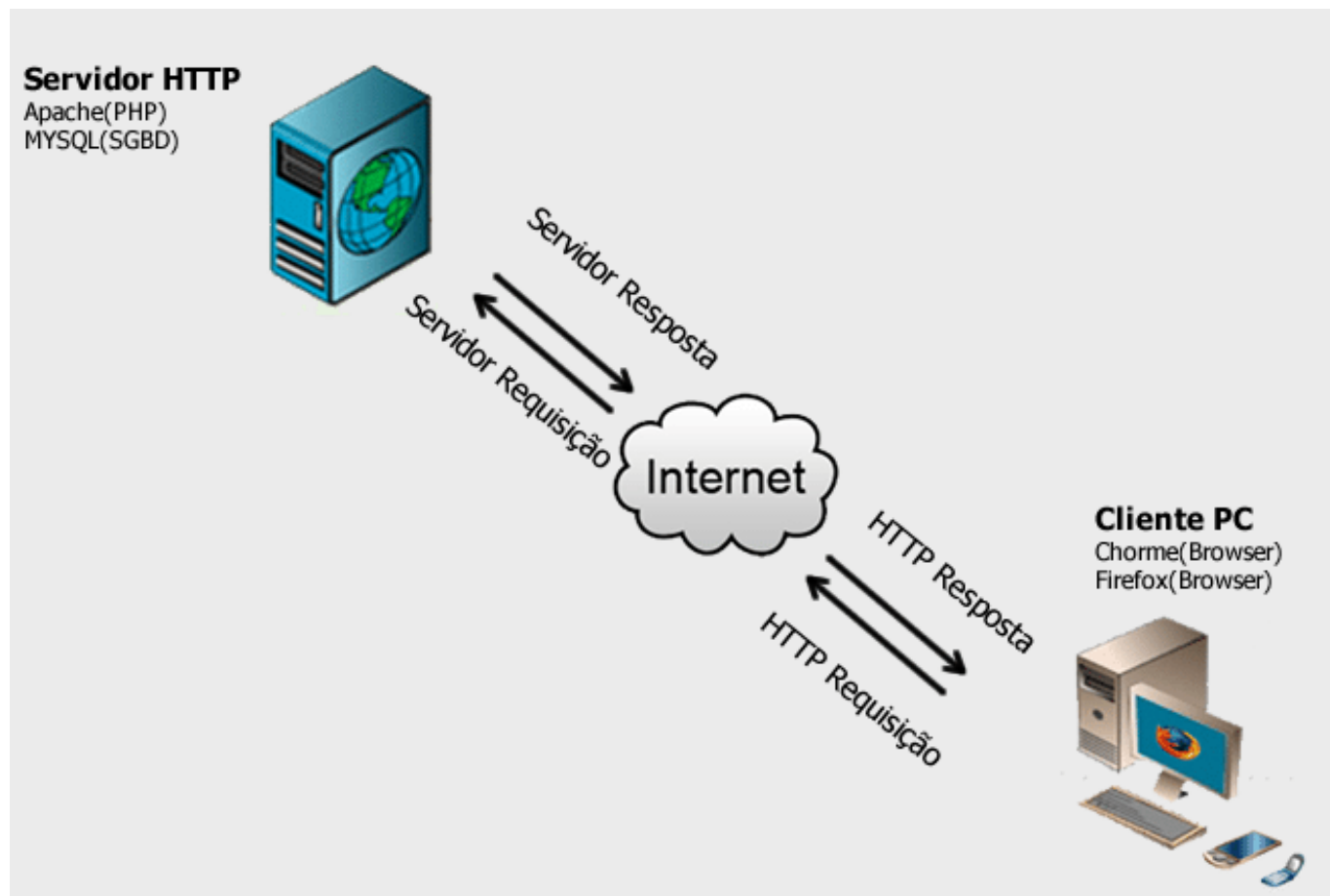
Aula de hoje: Roteiro

- AJAX
- jQuery



O que é AJAX?

➤ Introdução






O que é AJAX?

➤ Introdução



san f 


- san francisco weather
- san francisco
- san francisco giants
- san fernando valley
- san francisco state university
- san francisco hotels
- san francisco 49ers
- san fernando
- san fernando mission
- san francisco zip code

[Google Search](#) [I'm Feeling Lucky](#)



O que é AJAX?

➤ Introdução

 Cartão: 1 Produto ✕

Shopping cart summary

1. SUMMARY


2. LOGIN

3. ADDRESS

4. SHIPPING

5. PAYMENT

Your shopping cart contains: 1 product

PRODUCT	DESCRIPTION	REF.	UNIT PRICE	QTY	TOTAL
	iPod shuffle Color : Green	--	R\$ 82,57	<div>+ -</div> 1	R\$ 82,57 DELETE
Total products (tax incl.):					R\$ 82,57
Shipping:					Free Shipping!
Total (tax excl.):					R\$ 66,05
Total tax:					R\$ 16,52
Vouchers <input type="text"/> OK					<div>TOTAL:</div> <div>R\$ 82,57</div>

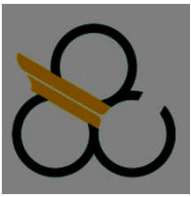
« Continue shopping Next »



O que é AJAX?

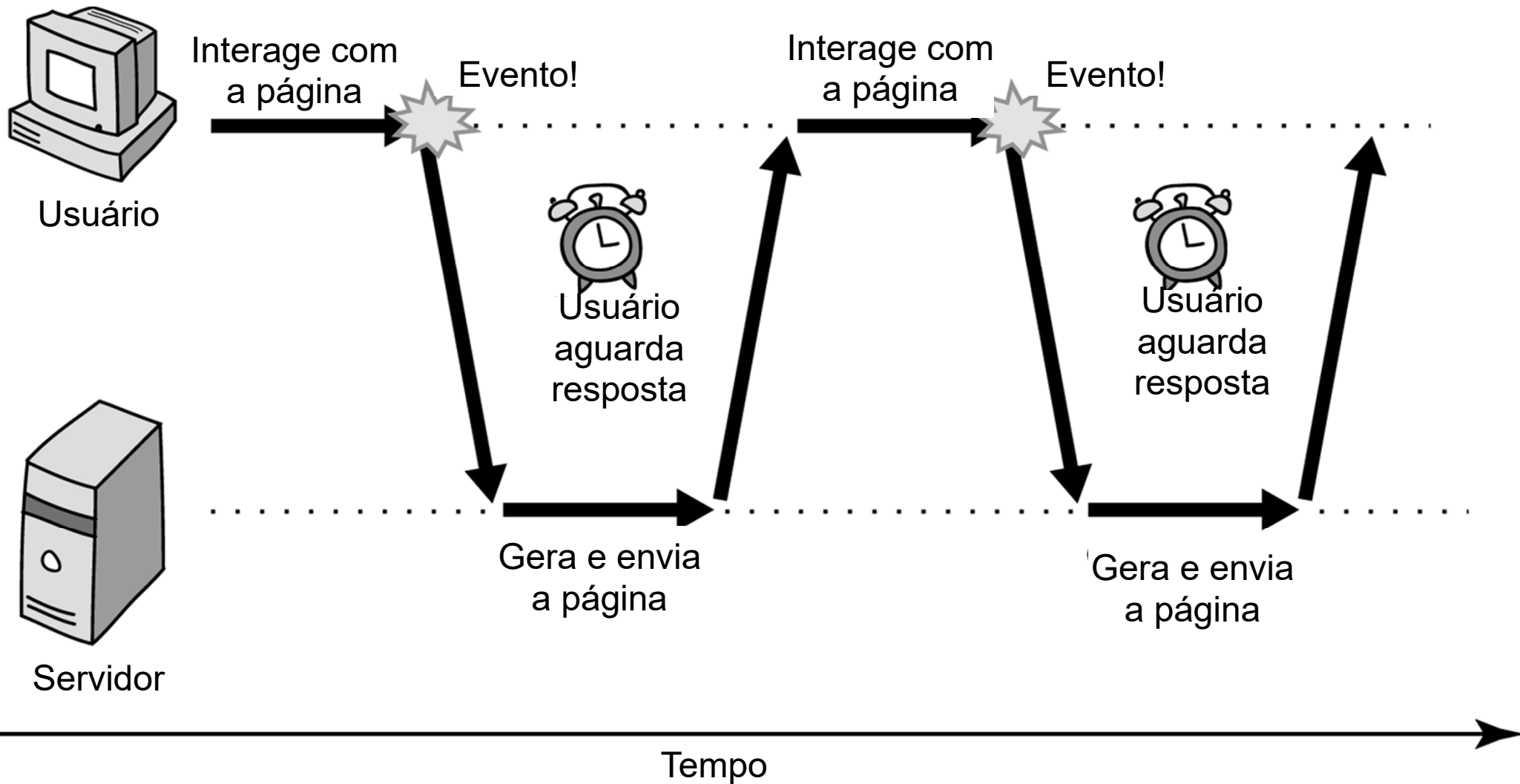


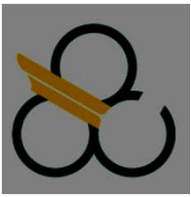
- **AJAX:** Asynchronous Javascript + XML
- É uma técnica que permite o envio de requisições assíncronas
- Uma forma para carregar dados dinamicamente de um servidor sem recarregar a página
- Permite apresentar dados ou atualizar a página dinamicamente, de maneira suave e contínua
- Auxilia na criação de sites web mais reativos, mais amigáveis ao usuário
- Exemplos: Gmail, Google Maps, Google Docs / Spreadsheets, Google Suggest, Facebook, etc.



Forma síncrona de comunicação

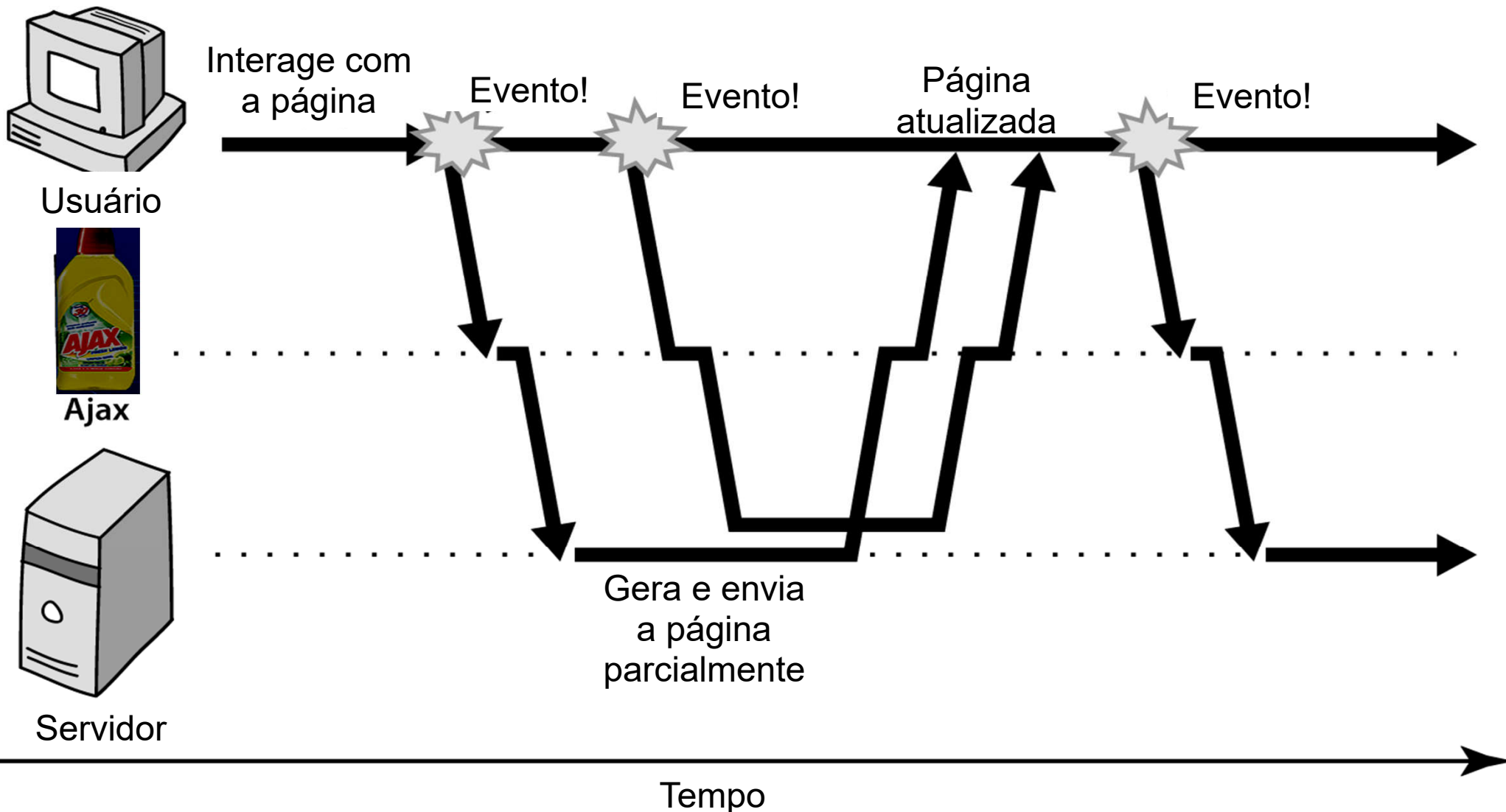
➤ Modelo de aplicação web clássica (comunicação **síncrona**): o usuário deve esperar o carregamento da página (**padrão clica-espera-atualiza**)

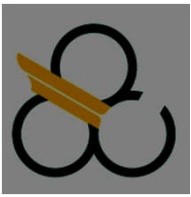




Forma assíncrona de comunicação

➤ Forma **assíncrona**: o usuário pode continuar interagindo com a página enquanto é carregada





Ajax

➤ Exemplos de usos do AJAX:

- Validação de dados em tempo real: dados de formulários como usuário/senha, telefones, CPF, etc. que requerem uma validação do lado do servidor
- Preenchimento automático de campos (*auto-completion*)
- Permite criar interfaces mais sofisticadas
- Permite manter dados atualizados, como por exemplo: cotação da bolsa, previsão do tempo, pontos em jogos esportivos, etc.

➤ Ajax permite a busca e inserção somente dos dados atualizados sem recarregar toda a página

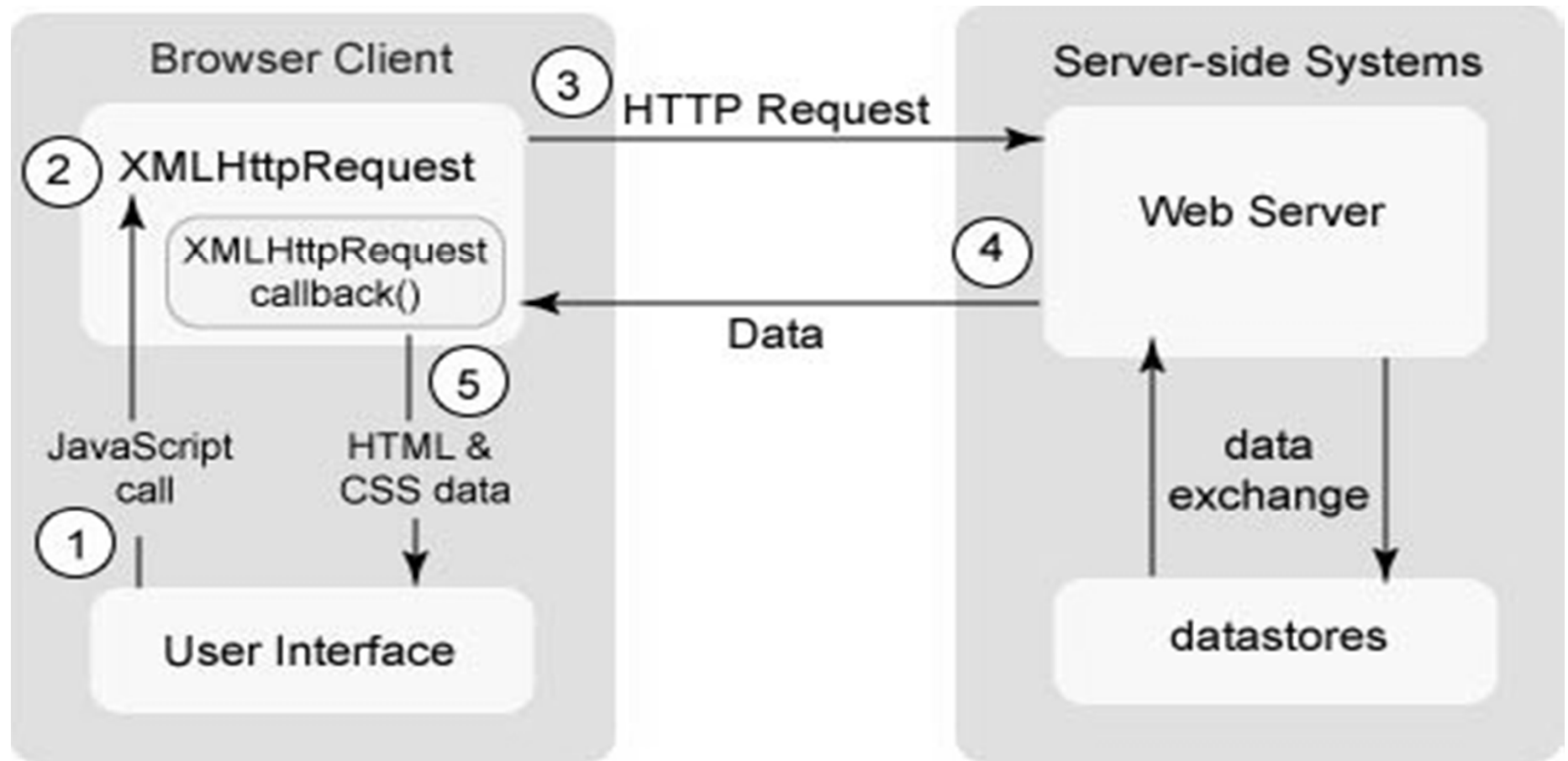


Utilizando o AJAX

- O AJAX utiliza um conjunto de conceitos e tecnologias (existentes) para requisitar um conteúdo do servidor web e utilizá-lo para atualizar as páginas web
- O elemento central do Ajax é um **objeto JavaScript** chamado **XMLHttpRequest** que é utilizado para possibilitar a comunicação assíncrona com o servidor
- **Idéia básica do AJAX:**
 - ❑ O navegador cria um objeto XMLHttpRequest
 - ❑ Através deste objeto, são requisitados dados do servidor web
 - ❑ Os dados são enviados de volta pelo servidor no formato XML
 - ❑ Usando JavaScript os dados XML são lidos e inseridos na página web



Requisição AJAX





Utilizando o objeto XMLHttpRequest

➤ Exemplo: Requisição síncrona

```
function testeAjax() {  
    var ajax = new XMLHttpRequest();  
    ajax.open("GET", "URL", false);  
    ajax.send(null);  
    // neste ponto a requisição foi completada  
    // fazer alguma coisa com a resposta: ajax.responseText  
}
```

- *open(metodo, url, async)*: inicia uma requisição web com os seguintes parâmetros:
 - **metodo**: pode ser “GET” ou “POST”
 - **url**: URL a ser carregada (recurso em um servidor)
 - **async**: indica se a requisição é síncrona (true) ou assíncrona (false)
- *send(data)*: chamado depois do **open** para completar a requisição (usado no POST, com GET passar null ou vazio)



Exemplo: Requisição síncrona

```
<script type="text/javascript">  
function testeAjax1() {  
    var ajax = new XMLHttpRequest();  
    ajax.open("GET", "testeAjax.txt", false);  
    ajax.send(null);  
    document.getElementById("caixaTexto").value = ajax.responseText;  
}  
</script>
```

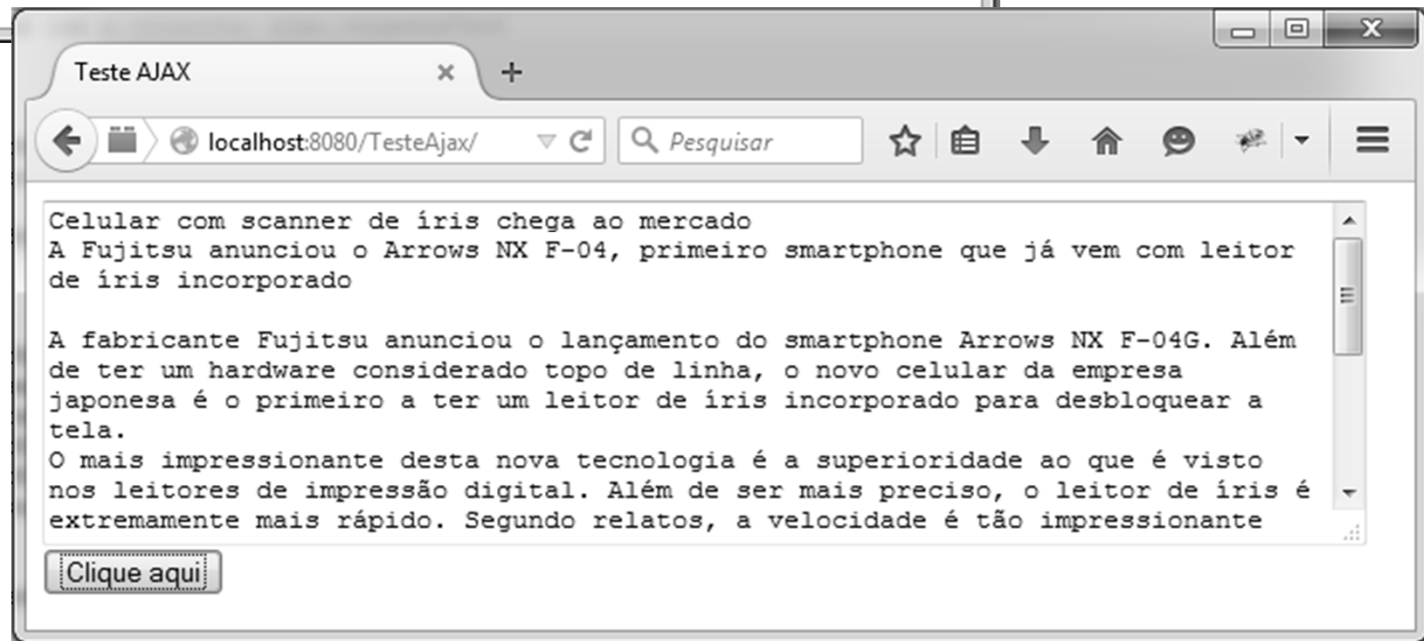
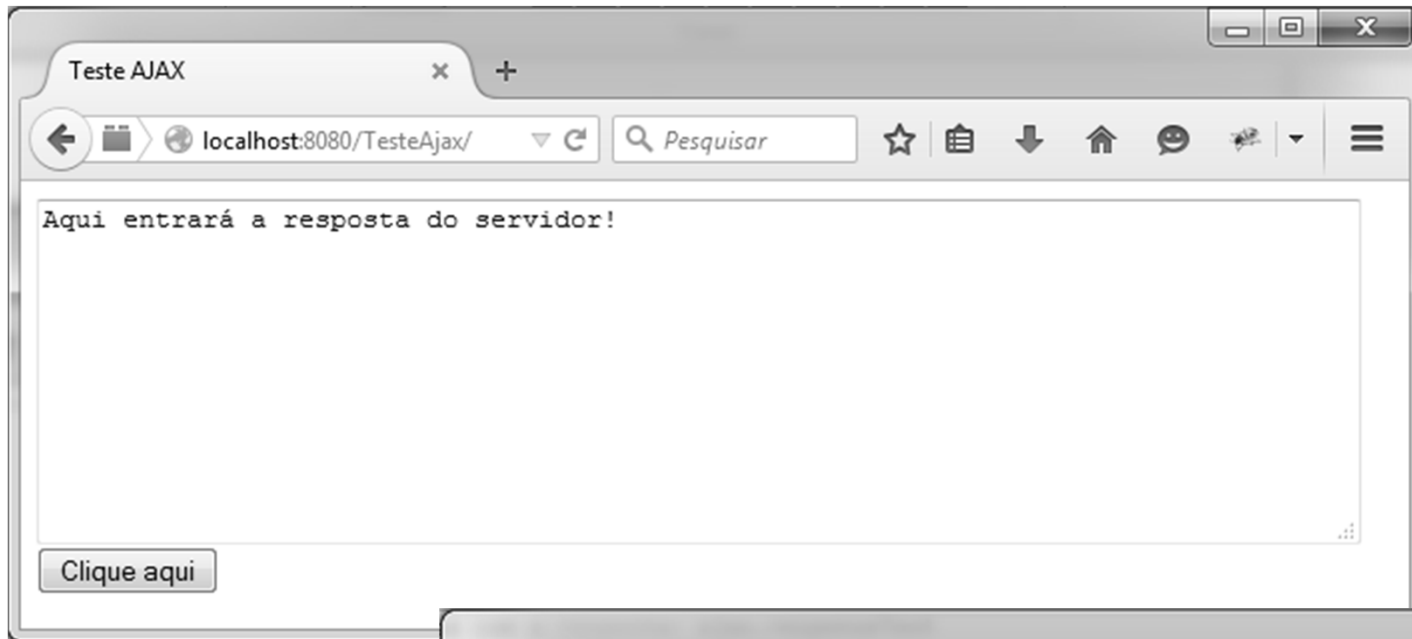
Solicita um arquivo
texto ao servidor

Insere os dados
do arquivo texto
na área de texto

```
<body>  
    <textarea id="caixaTexto" rows="10" cols="80">Aqui entrará a  
    resposta do servidor!</textarea>  
    <br>  
    <input type="button" id="botao" value="Clique aqui"  
        onclick="testeAjax1();" />  
</body>
```



Exemplo: Requisição síncrona





Tratando erros

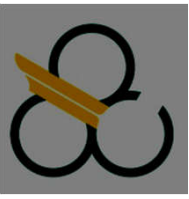
➤ Em uma requisição web podem ocorrer diversos tipos de problemas:

- ❑ URL informada está incorreta ou a página não existe
- ❑ Perda de conexão com a rede
- ❑ Servidor pode estar fora do ar ou sobrecarregado

➤ O objeto XMLHttpRequest possui propriedades *status* e *statusText* que indicam um código de erro e uma mensagem relacionada ao erro

```
function testeAjax1() {  
    var ajax = new XMLHttpRequest();  
    ajax.open("GET", "testeAjax.txt", false);  
    ajax.send(null);  
    if (ajax.status == 200) {  
        document.getElementById("caixaTexto").value =  
            ajax.responseText;  
    } else {  
        alert("Erro: " + ajax.status + " , mensagem " +  
            ajax.statusText);  
    }  
}
```

200 é o código que indica
que a requisição foi
processada com sucesso



Requisição assíncrona

➤ Exemplo: Requisição assíncrona

```
function testeAjax2() {  
    var ajax = new XMLHttpRequest();  
    ajax.onreadystatechange = function() {  
        if (ajax.readyState == 4 && ajax.status == 200) {  
            // fazer alguma coisa com a resposta: ajax.responseText  
        }  
    }  
    ajax.open("GET", "URL", true);  
    ajax.send();  
}
```

- A requisição é enviada para o servidor e é especificada uma função que será chamada quando a resposta for recebida
- *onreadystatechange*: evento que ocorre quando o estado da requisição muda (registrar uma função para tratar este evento)
→ *função callback*



Exemplo: Requisição assíncrona

```
function testeAjax2() {  
    var ajax = new XMLHttpRequest();  
    ajax.onreadystatechange = function() {  
        if (ajax.readyState == 4 && ajax.status == 200) {  
            // fazer alguma coisa com a resposta: ajax.responseText  
        }  
    }  
    ajax.open("GET", "URL", true);  
    ajax.send();  
}
```

- *readyState*: estado corrente da requisição, que pode ser:
- 0: não inicializado
 - 1: não enviado
 - 2: enviado
 - 3: em andamento
 - 4: completo



Exemplo: Requisição assíncrona

```
<script type="text/javascript">
    function testeAjax2() {
        var ajax = new XMLHttpRequest();
        ajax.onreadystatechange = function() {
            if (ajax.readyState == 4 && ajax.status == 200) {
                document.getElementById("caixaTexto").
                    value = ajax.responseText;
            }
        }
        ajax.open("GET", "testeAjax.txt", true);
        ajax.send();
    }
</script>
```

```
<body>
    <textarea id="caixaTexto" rows="10" cols="80">Aqui entrará a
        resposta do servidor!</textarea><br>
    <input type="button" id="botao" value="Clique aqui"
        onclick="testeAjax2();" />
</body>
```



Processando dados XML

- Servidores web comumente utilizam o formato XML para montar as respostas às requisições
- Exemplo de um arquivo XML:

```
<?xml version = "1.0" encoding="ISO-8859-1"?>
  <deitel:books xmlns:deitel = "http://www.deitel.com/booklist">
    <book>
      <title>Visual C#, 2a ed.</title>
    </book>
    <book>
      <title>Java, 7a ed.</title>
    </book>
    <book>
      <title>C++, 6a ed.</title>
    </book>
    <book>
      <title>Internet and WWW, 4a ed.</title>
    </book>
  </deitel:books>
```



XML e Ajax

- Usando Ajax, os dados (arquivo XML) são carregados, processados e apresentados
- Como examinar e obter os dados XML?
 - ❑ Seria trabalhoso acessar um grande e complexo arquivo como se fosse uma string
 - ❑ Felizmente, os dados de um arquivo XML são analisados e convertidos (parsed) em um conjunto de objetos
 - ❑ Ou seja, existe um XML DOM muito similar ao HTML DOM
 - ❑ As tags do XML formam uma estrutura de árvore



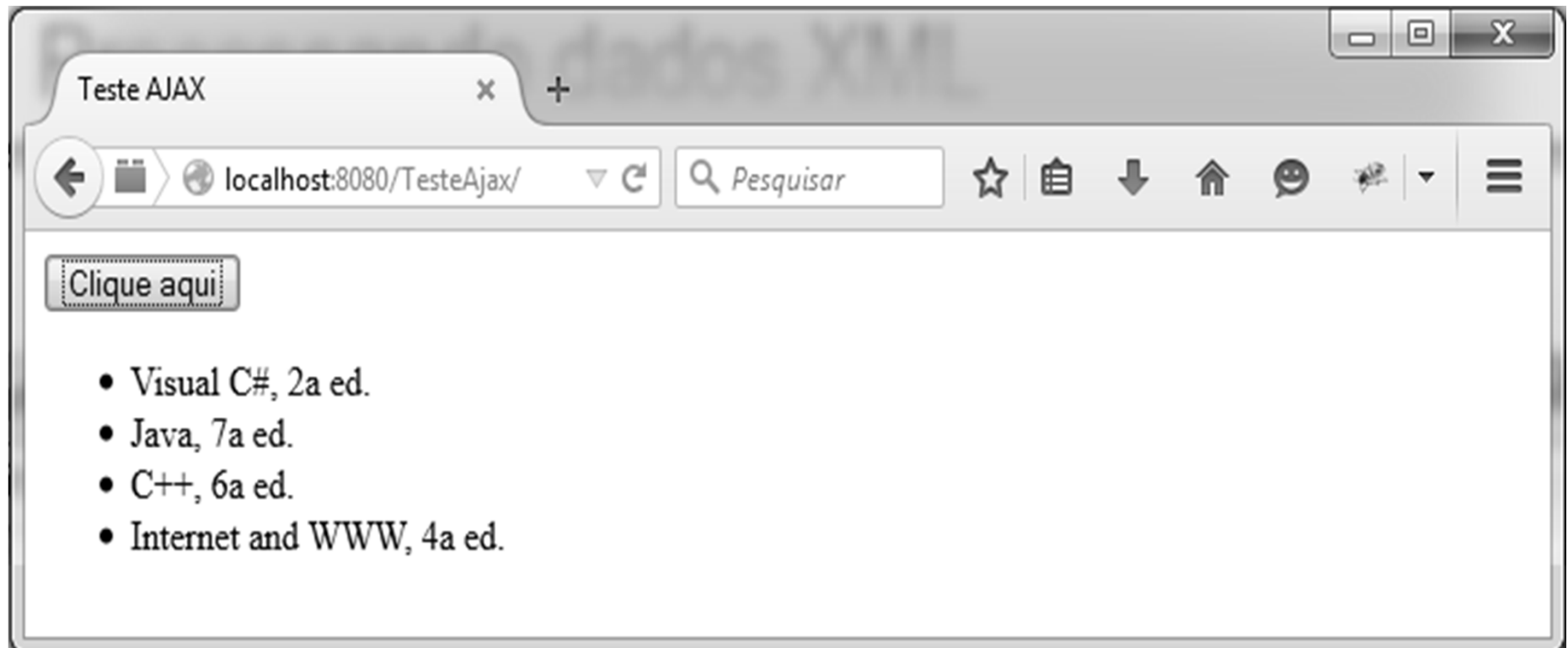
Processando dados XML

➤ Exemplo: Esta função apresenta os dados (livros) do arquivo XML apresentado antes em uma lista não ordenada

```
➤ function testeAjax4() {  
    var ajax = new XMLHttpRequest();  
    ajax.onreadystatechange = function() {  
        if (ajax.readyState == 4 && ajax.status == 200) {  
            var livros = ajax.responseXML.getElementsByTagName("book");  
            var ul = document.createElement("ul");  
            for (var i = 0; i < livros.length; i++) {  
                var no = livros[i].getElementsByTagName("title")[0];  
                var titulo = no.firstChild.nodeValue;  
                var li = document.createElement("li");  
                li.innerHTML = titulo;  
                ul.appendChild(li);  
            }  
            document.getElementById("conteudo").appendChild(ul);  
        }  
    }  
    ajax.open("GET", "livros.xml", true);  
    ajax.send();  
}
```



Processando dados XML



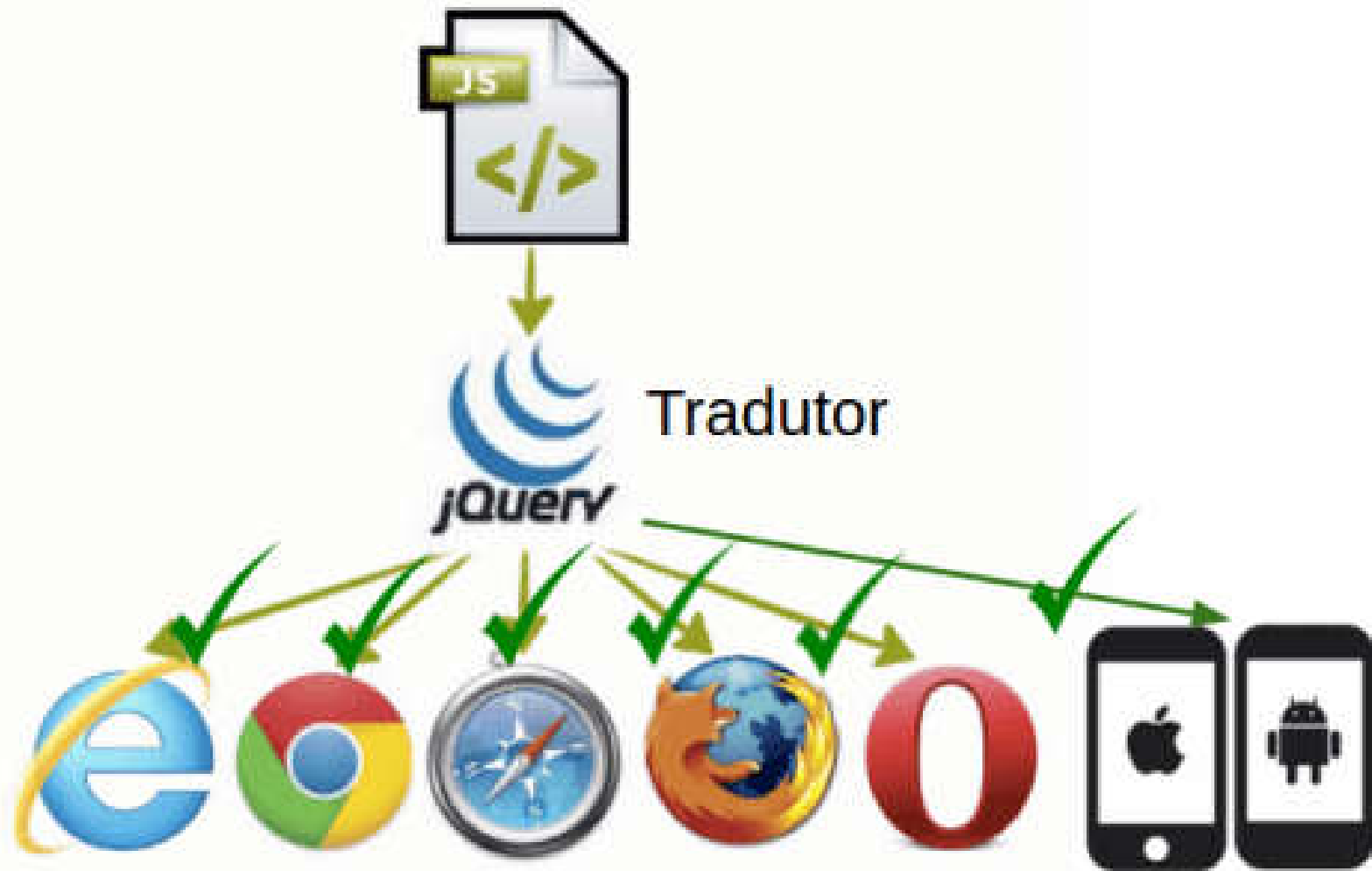


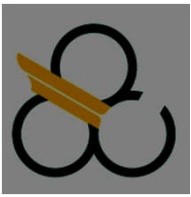
jQuery

- jQuery é uma biblioteca JavaScript, desenvolvida em 2006 por John Resig
- Simplifica e agiliza o desenvolvimento de várias tarefas baseadas em Javascript (*“write less, do more”*)
 - ❑ Selecionar elementos HTML e modificar seu conteúdo
 - ❑ Tratamento de eventos
 - ❑ Suporte a Ajax
 - ❑ Animações
 - ❑ Suporte a diferentes navegadores
- É um dos mais populares frameworks Javascript, é utilizado por grandes empresas como Google, Microsoft, IBM, Netflix



jQuery





jQuery

```
var paragrafos = document.querySelectorAll('p');  
for( var i = 0; i < paragrafos.length; i++){  
    paragrafos[i].textContent = "novo texto";  
}
```

JS

```
$('p').text('novo texto');
```

jQuery



jQuery

- Baixar a versão do jQuery no site e acrescentar nas páginas HTML:
- Site: <https://jquery.com/download/>
- Código: `<script src="resources/js/jquery.js"></script>`



jQuery

- Uma das facilidades do jQuery é selecionar elementos HTML e executar alguma ação sobre o elemento
- Por exemplo, o código:

```
<script type="text/javascript">
    $(document).ready(function(){
        $("button").click(function(){
            alert("Olá!")
        });
    });
</script>
```

- `$(document).ready`: garante que o código execute depois que o navegador carregar o documento (ou seja, quando o documento estiver pronto para ser manipulado)
- `$("button").click`: trata um evento no botão (clique no botão) → neste exemplo, mostra uma mensagem ao usuário

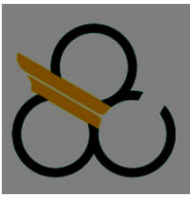


jQuery

- A sintaxe básica do jQuery é: **`$(elemento).evento()`**
- Onde **elemento** é um elemento HTML, e **evento** é o evento a ser tratado (por exemplo, um clique no botão) e uma ação é associada ao evento (uma função tratadora do evento)

```
<script type="text/javascript">
    $(document).ready(function(){
        $("button").click(function(){
            $("p").hide();
        });
    });
</script>
```

- Neste exemplo, quando o usuário clicar em um botão, todos os elementos marcados com **<p>** (parágrafos) serão escondidos, ou seja, desaparecerão da página
- Pode-se também selecionar somente um elemento HTML, utilizando o atributo **id** do elemento (com o símbolo # antes)
- Ex.: `$("#ok").hide();` // esconde o botão cujo id="ok"



jQuery

- Um evento é gerado (disparado) no momento em que o usuário realiza alguma ação
- O jQuery possui vários métodos para tratar eventos em elementos HTML, por exemplo: `click()`, `dblclick()`, `mousedown()`, `mouseup()`, etc.
- Uma função para tratar um evento é declarada como parâmetro do método do evento (ou seja, é a função que deve ser tratada quando o evento ocorrer)
- * Obs: Veja exemplos de eventos e efeitos (animações) do jQuery no site:
 - <http://www.w3schools.com/jquery>



Utilizando o AJAX com jQuery

- O jQuery oferece alguns métodos que dão suporte às funcionalidades do Ajax
- Os métodos `get()` e `post()` podem ser utilizados para enviar requisições do tipo GET ou POST para o servidor
- Ambos métodos permitem trazer dados do servidor, porém, o `post()` permite enviar dados na requisição que podem ser processados para retornar uma resposta



Utilizando o AJAX com jQuery

```
$.ajax({  
    url : 'http://localhost:8080/Banco/findAll',  
    type : 'POST',  
    dataType : 'json',  
    data : {  
        searchText : request.term,  
    },  
    success : function(data) {  
        response($.map(data, function(item) {  
            console.log(item['descricao'])  
            return {  
                label : item['descricao'],  
                value : item['descricao']  
            }  
        }  
    )));  
});
```




Utilizando o AJAX com jQuery

```
<script type="text/javascript">
    $('#campo-busca').autocomplete({
        source : function(request, response) {
            $.ajax({
                url : 'http://localhost:8080/Banco/findAll',
                type : 'POST',
                dataType : 'json',
                data : {
                    searchText : request.term,
                },
                success : function(data) {
                    response($.map(data, function(item) {
                        console.log(item['descricao'])
                        return {
                            label : item['descricao'],
                            value : item['descricao']
                        }
                    }
                ));
            }
        });
    });
</script>
```



Utilizando o AJAX com jQuery

➤ Incluir os seguintes arquivos:

- `<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">`
- `<script src="https://code.jquery.com/jquery-1.10.2.js"></script>`
- `<script src="https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>`

➤ Acrescentar o seguinte campo:

- `<label>Banco</label>`
- `<form:input id="campo-busca" path="banco" />`



Utilizando o AJAX com jQuery

Banco

banc|

Banco do Brasil

Banco do Nordeste



Utilizando o AJAX com jQuery

- Função para enviar uma requisição POST ao servidor para modificar o status de um aluno (de não formado para formado)

```
<script type="text/javascript">  
    function mudaStatus(id){  
        $.post("mudaStatus",{ 'id' : id}, function(){  
            $("#td1aluno_"+id).html("Sim");  
            $("#td2aluno_"+id).html("");  
        })  
    }  
</script>
```

- No servidor, a requisição é tratada pelo AlunoController e é modificado o valor no banco de dados (através de AlunoDAO)
- Quando chega a resposta do servidor, os campos da tabela são modificados (conforme a figura anterior)



Utilizando o AJAX com jQuery

```
<table border=1>
  <tr>
    <td><b>Id</b></td>
    <td><b>Nome</b></td>
    <td><b>E-mail</b></td>
    <td><b>Endereço</b></td>
    <td><b>Formado?</b></td>
    <td><b>Alterar status?</b></td>
    <td><b>Alterar?</b></td>
    <td><b>Remover?</b></td>
  </tr>
  <c:forEach items="${alunos}" var="aluno">
    <tr>
      <td>${aluno.id}</td>
      <td>${aluno.nome}</td>
      <td>${aluno.email}</td>
      <td>${aluno.endereco}</td>
      <c:if test="${aluno.formado eq false}">
        <td id="td1aluno_${aluno.id}">Não</td>
      </c:if>
      <c:if test="${aluno.formado eq true}">
        <td>Sim</td>
      </c:if>
      <c:if test="${aluno.formado eq false}">
        <td id="td2aluno_${aluno.id}"><a href="#"
          onClick="mudaStatus(${aluno.id})">Mudar</a></td>
      </c:if>
      <c:if test="${aluno.formado eq true}">
        <td></td>
      </c:if>
      <td><a href="exibeAluno?id=${aluno.id}">Alterar</a></td>
      <td><a href="removeAluno?id=${aluno.id}">Remover</a></td>
    </tr>
  </c:forEach>
</table>
```



Referências

- P.J. Deitel, H.M. Deitel. Ajax, Rich Internet Applications e desenvolvimento Web, ed. Pearson, 2008.

- Ajax Tutorial
- <http://www.w3schools.com/ajax/default.asp>

- jQuery
- <http://jquery.com/>

- jQuery Fundamentals
- <http://jqfundamentals.com/legacy/>