
# ASYNCHRONOUS JAVASCRIPT AND XML

Sistemas Distribuídos e Tolerância a Falhas
Universidade da Beira Interior

Paula Freire M3841
Tiago Machado M3863

# Plano da Apresentação

- Apresentação da tecnologia
- Arquitectura e funcionamento
- Ferramentas
- DHTML e Ajax
  - Exemplo
- Exemplos
  - Exemplos existentes
  - Exemplo funcional
- Referências

# Apresentação da tecnologia

- Ajax é o termo dado à utilização de um conjunto de tecnologias.
- É usado para construir páginas Web com conteúdos dinâmicos.
- Serve também para poupar recursos ao servidor.
- Pelo facto de se recorrer a JavaScript, funciona tanto ao nível de servidores Web Apache/Tomcat como do IIS

Ajax não é dependente dos servidores Web, e como tal poderá ser usado com PHP, JSP, ASP e ASP.NET.

# Apresentação da tecnologia (cont.)

**Especificamente:**

- O Ajax permite:
  - De uma forma selectiva, modificar secções de uma página Web, e actualizá-la sem a necessidade de recarregamento de todo o documento)
  - **Exemplo:** A validação de campos de formulário de registo, podem ser processados/validados no momento

Por todo o documento subentende-se: todos os seus componentes já anteriormente carregados: imagens, menus, etc.

# Apresentação da tecnologia (cont.)

**Especificamente:**

- **Poupança de recursos**:
  - O Ajax permite o processamento distribuído por cada computador cliente (através do JavaScript) com dados retirados do servidor.
  - A página Web torna-se mais eficiente dado que é utilizado poder de processamento de vários clientes em vez da utilização apenas do servidor.

JavaScript pelo facto de ser uma linguagem executada do lado do cliente, utiliza apenas recursos locais.


O HTML e CSS são usados para a apresentação das páginas Web. O XMLHttpRequest permite a troca de dados de forma assíncrona entre o servidor e o cliente/browser.

A troca de dados assincrona significa que a resposta do servidor será disponibilizada assim que possível, sem que ocorra a paragem da página e consequentemente a espera dessa mesma resposta.

A DOM define objectos e propriedades de todos os elementos HTML e XML e os métodos (a interface) para aceder aos mesmos. Existe o HTML DOM e XML DOM.

# Arquitectura e seu funcionamento (cont.)

## Como funciona?

- O Ajax utiliza um modelo de programação através de componentes visuais e eventos.
- Estes eventos são acções do utilizador, que chamam funções associadas a elementos da página Web.
- A interactividade é alcançada de uma forma geral através de formulários e botões. DOM permite a ligação dos elementos da página com acções.

## Arquitectura e seu funcionamento (cont.)

De forma a extrair dados do servidor, o XMLHttpRequest fornece dois métodos:

- *open (mode,url,boolean): estabelece a criação da ligação.*
  - **mode**: tipo de pedido, GET ou POST.
  - **url**: localização do ficheiro.
  - **boolean**: modo de operação (síncrono ou assíncrono)
- *send*

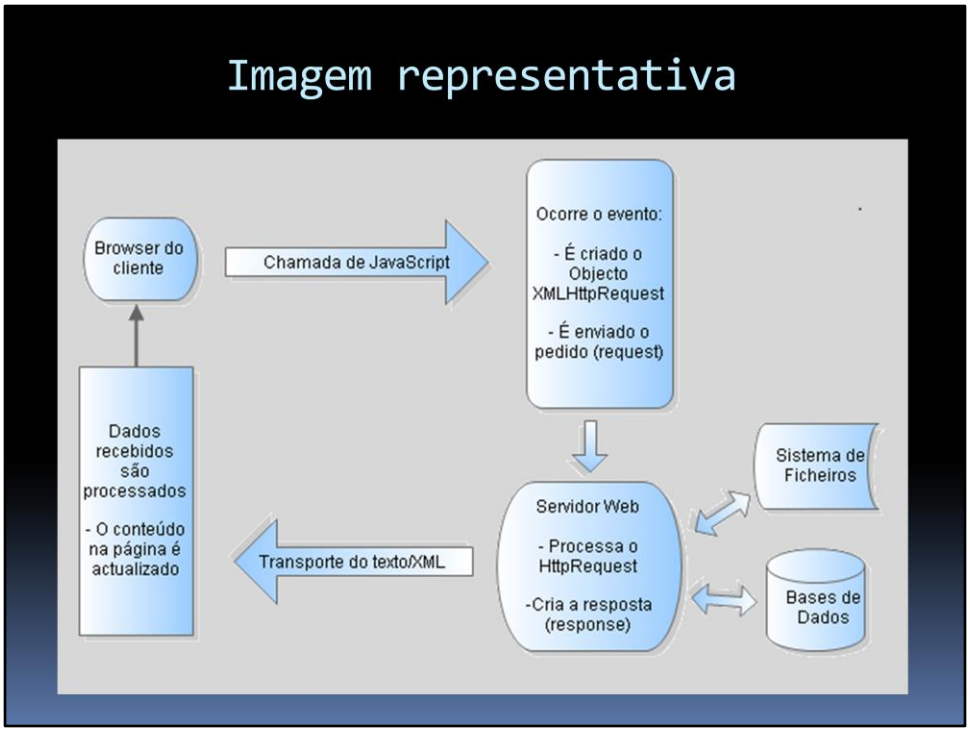
- *Boolean:* true ou false, para decidir se é possível utilizar os elementos já carregados ou se é necessário aguardar
- *send:* envio do pedido com os dados necessários para o servidor.

## Arquitectura e seu funcionamento (cont.)

Os atributos do objecto XMLHttpRequest são os seguintes:

- *readyState;*
- *status;*
- *onreadystatechange;*
- *responseText;*
- *responseXML.*

- Readystate 0 a 4 (ready)
- Status: 200 is ok 404 not found
- Onreadystatechange: a função é chamada cada vez que o atributo readyState se altera
- Response text: resposta em *plain text*
- responseXML: recurso a um ficheiroXML




1. O utilizador interage com um elemento na página Web, que desencadeia uma chamada de JavaScript;
2. Ocorro e evento que cria o objecto XHR, e é enviado o *request* para o servidor;
3. O servidor processa o pedido, acede aos sistemas de ficheiros/bases de dados, e a cria a resposta;
4. É enviado na forma de texto ou sobre a forma de um ficheiro;
5. Os dados são recebidos pelo browser do utilizador, sendo processados no mesmo de forma local e consequentemente o conteúdo pretendido, actualizado.

# Arquitectura e seu funcionamento (cont.)

- HTML/XML DOM

  - Uma interface de programação para o HTML/XML;
  - Independentemente de plataforma e linguagem;
  - Um standard do W3C.

Permite o acesso, a alteração, adição e a remoção de elementos HTML e XML.

De acordo com o DOM, a estrutura de um documento HTML e XML é constituída por nodos.

# Arquitectura e seu funcionamento (cont.)

- HTML DOM – Exemplo:

O atributo de HTML DOM mais utilizado é o objecto de texto.

Cada vez que usamos a tag **<input type="text">** num formulário, é criado um objecto deste tipo.

Por sua vez o acesso à informação introduzida é geralmente realizada através da utilização do elemento **document.getElementById()**

## Arquitectura e seu funcionamento (cont.)

- XML DOM – Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <nota>
    <para>Paula</para>
    <de>Tiago</de>
    <cabeçalho>Compromisso</cabeçalho>
    <body>Apresentação de Sistemas Distribuidos!</body>
  </nota>
```

O standard DOM indica:

- O documento inteiro é um nodo de documento;
- Cada elemento XML é um nodo de elemento;
- O texto nos elementos XML são noso de texto;
- Cada atributo é um nodo de atributo;
- Comentários são nodos de comentário.

## Arquitectura e seu funcionamento (cont.)

Como criar um pedido (request de Ajax) ?

Um utilizador carrega num *link* na página Web:

```
<div id="paula"> <a
href="javascript:carregarAluno(1)"><B>Aluno 1</B></a> </div>
```

**Primeiro é necessário instanciar o objecto XHR**

```
try {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
} catch(e)    //caso contrário
{
        xhr = new XMLHttpRequest();
}
```

O objecto XMLHttpRequest tem de ser criado novamente cada vez que é usado, dado que é automaticante "destruído" após a sua utilização.

A criação do objecto será diferente caso se utilize o Internet Expliorer, recorrendo-se ao ActiveXObject para este navegador.

## Arquitectura e seu funcionamento (cont.)

```
xhr.onreadystatechange = function () {

 if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("id_div").innerHTML =

xhr.responseText;
 }
xhr.open("GET", "info.xml", true);
xhr.send();
```

A informação é retirarada do ficheiro (open), e enviada (send) para que seja apresentada na div anteriormente definida.

- É necessário que o readyState seja 4 (ok) e o estado da página seja 200 para que seja possível emitir uma resposta.
- Neste caso foi emitida uma resposta em texto pleno (*plain text*).

## Arquitectura e seu funcionamento (cont.)

Temos uma div de nome "info", uma caixa de texto, e um botão associado a um evento.

```
<div id="info">Conteudo ainda não foi carregado....</div>

<input type="text" id='texto' />

<input type='button' onclick='alterar()' value='Alterar Conteudo' />
```

Esta é uma situação semelhante – o conteudo é automaticamente carregado do ficheiro em modo assincrono (*true*) , mas com possibilidade de alteração no ficheiro, através de uma caixa de texto para introdução de dados.

## Arquitectura e seu funcionamento (cont.)

Na função alterar(), obtemos o conteúdo introduzido pelo utilizador através de

```
var texto = document.getElementById('texto');
```

E enviamos a informação com um comando POST, com a informação do ficheiro onde tencionamos escrever assim como o seu conteúdo.

```
var data = "file=info.xml&conteudo=" + texto.value;
xhr.open("POST", "index2.aspx", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send(data);
```

É realizado o POST para um outro ficheiro (index2.aspx) de forma a evitar o carregamento da própria página actual (index.aspx), para assim obter uma forma assincrona de comunicação.

## Arquitectura e seu funcionamento (cont.)

No ficheiro index2.aspx.cs fazemos o tratamento e processamento do *request*.

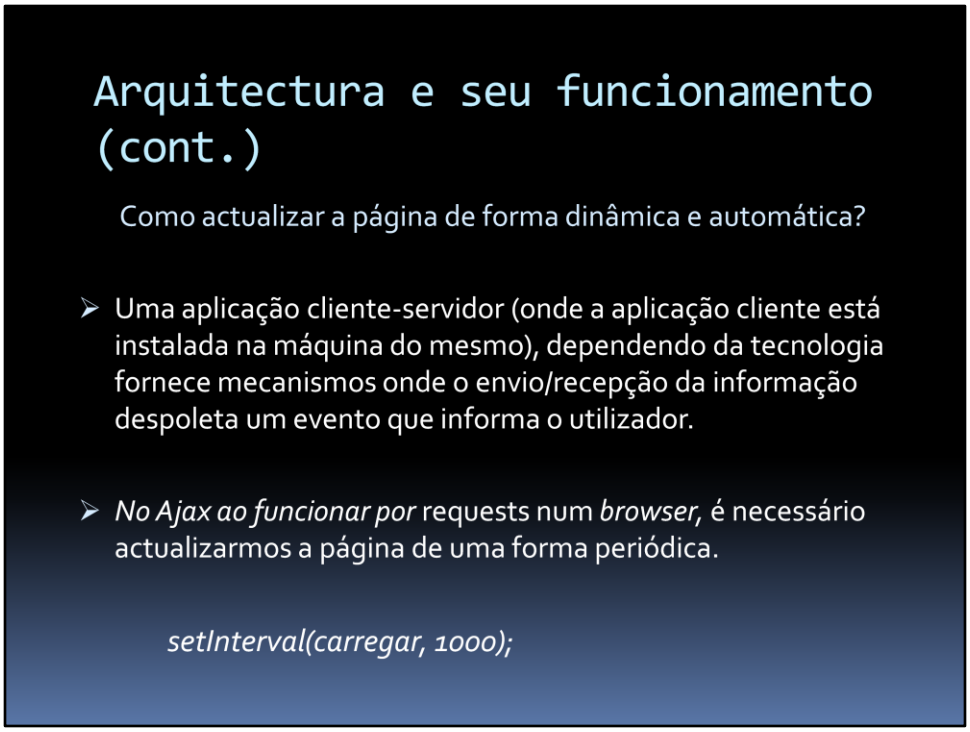
```
string xpto = Request["conteudo"]; //  informação obtida através do input do utilizador na caixa de texto
// caminho do ficheiro
XmlTextWriter w = new XmlTextWriter(Server.MapPath(Request.ApplicationPath) + "/info.xml", null);
w.WriteStartDocument();
w.WriteStartElement("informacao");   // nodo informacao
w.WriteElementString("texto", xpto.ToString()); //atributo texto, onde é guardado o pretendido
w.WriteEndElement();  // fim do nodo
w.WriteEndDocument(); // fim do documento
w.Flush();
w.Close();
```

- *XmlTextWriter w = new XmlTextWriter(Server.MapPath(Request.ApplicationPath)*

Para escrever o ficheiro no mesmo caminho onde o ficheiro correspondente à pagina Web se encontra.

- w.WriteElementString.

Permite guardar string's nos atributos do ficheiro XML.




- carregar() é a função que pretendemos que ocorra periodicamente
- 1000 é o intervalo de tempo em 1000 segundos.

No caso anteriormente demonstrado (e que será mostrado na prática no exemplo funcional), quando um utilizador altera o conteudo, isto significa que o mesmo é actualizado de uma forma completamente dinâmica e automática em todos os restantes utilizadores que tenham a página Web em execução.

## Ferramentas

- Microsoft Visual Studio;
- Eclipse;
- NetBeans IDE;
- Notepad++;

- Firebug: para depuração de Ajax;

- DWR – Direct Web Remoting;
- Prototype;
- Anaa;

1. DWR - framework criada em Ajax que fornece facilidade de comunicação entre o Java existente no servidor e o JavaScript no browser;
2. Prototype - para chamadas remotas com recurso a Ajax;
3. Anaa - uma API para Ajax que fornece várias funções de carregamento e gravação de dados, de e para o servidor.

# DHTML e Ajax

- É possível desenvolver páginas Web com componentes dinâmicos (com recurso a HTML, CSS, JavaScript), recebendo a denominação de Dynamic HTML (DHTML),
- Contudo considera-se apenas que se utiliza Ajax, quando se utiliza o objecto XHR.
- O DHTML não exige processamento ao nível do servidor após o carregamento da página inicial.

## DHTML - Exemplo

```
<table width="150" height="40">
<tr>
 <td onmouseover="mudarFundo('red')"
  onmouseout="mudarFundo('transparent')"
bgcolor="red">
 </td>
 <td onmouseover="mudarFundo('blue')"
onmouseout="mudarFundo('transparent')"
 bgcolor="blue">
</td>
<td onmouseover="mudarFundo('green')"
 onmouseout="mudarFundo('transparent')"
 bgcolor="green">
  </td>
</tr>
</table>
```

Neste exemplo pretendemos demonstrar como é possível utilizar DHTML, sem que se recorra ao servidor.

Consiste na possibilidade do utilizador alterar de forma dinamica a cor do fundo da página. O utilizador dispõe de três opções, vermelho, azul e verde.

Neste caso organizámos estas opções para o utilizador numa tabela DOM, em HTML.

## DHTML - Exemplo

```
function mudarFundo(fundo) {
        document.body.style.background = fundo;
}
```

- O evento associado à mudança de cor de fundo, pelo facto de se utilizar apenas JavaScript para o efeito, ocorre apenas localmente, não utilizando quaisquer recursos do servidor.

De forma a utilizarmos Ajax poderiamos por exemplo enviar para o servidor a cor escolhida, para que na próxima visita à página Web essa informação fosse tida em conta.

## Desvantagens do Ajax

- É necessário que o JavaScript esteja activo no browser.
- Como os dados são carregados dinamicamente, eles não fazem parte da página directamente, pelo que as *keywords* não são detectadas pelos navegadores
- O modo assíncrono pode provocar que a página seja carregada com espaços "vazios"

Por vezes existem situações em que o processamento pode demorar algum tempo após o carregamento por parte do servidor, ocorrendo desta forma temporariamente (ou definitivamente caso exista falhas do servidor) espaços vazios na página ou falhas na actualização do conteúdo.

# Exemplos de utilização

- Google Search (Google Instant e Suggest)
- Google Gmail
- Facebook

## Exemplo funcional

- Os exemplos anteriormente mostrados são baseados no nosso exemplo funcional.

- Na prática...

# Referências


- http://www.adaptivepath.com/ideas/essays/archives/000385.php
- http://www.w3schools.com/dom/default.asp
- http://www.w3schools.com/htmldom/dom_intro.asp
- http://www.w3schools.com/XML/xml_http.asp
- http://www.w3schools.com/dhtml/dhtml_intro.asp
- http://www.xul.fr/ajax-frameworks.html

FIM