Fundamentos do Ajax – o que é Ajax?

- AJAX é meramente um termo utilizado para descrever o processo de utilização do objeto **XMLHttpRequest** do JavaScript, para recuperar ou enviar informações ao servidor de forma **assíncrona**.

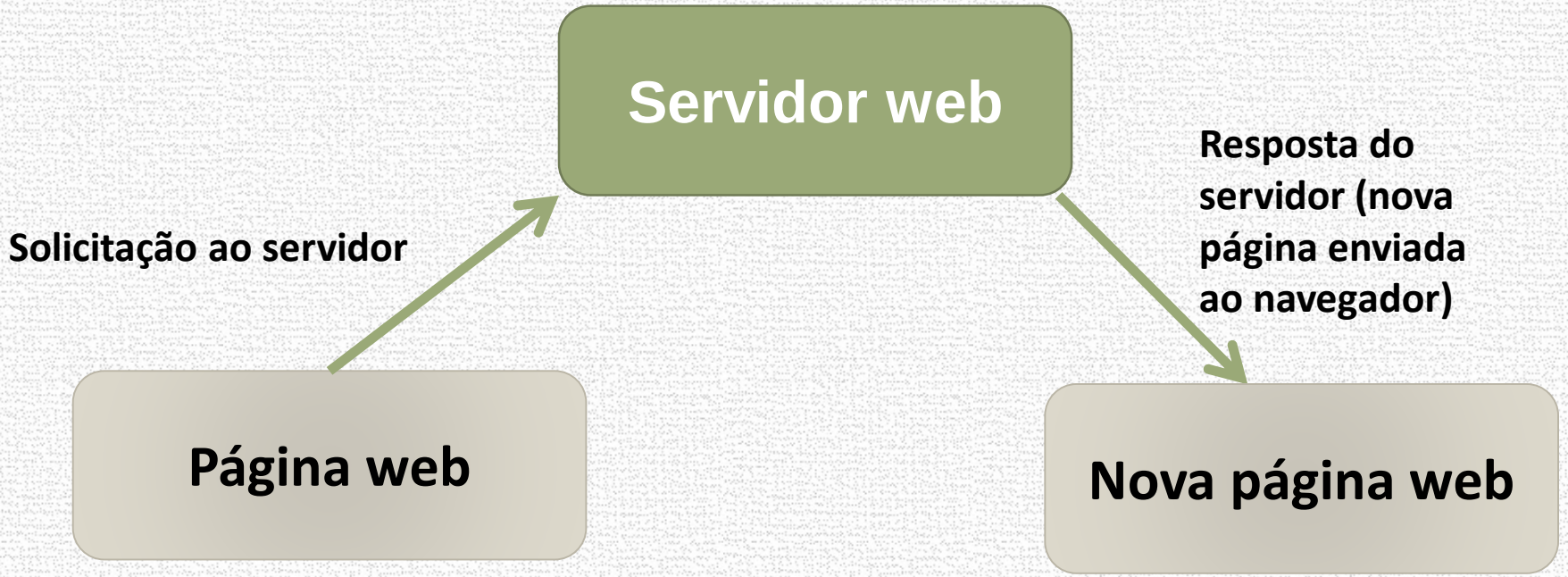
- AJAX é o acrônimo de **Asynchronous JavaScript and XML**, ou seja, JavaScript e XML assíncronos;

- AJAX é uma técnica que integra a utilização das linguagens JavaScript e HTML, juntamente com Folha de Estilos em Cascata (CSS) e uma linguagem de servidor – no nosso caso, o PHP. Mas pode ser qualquer outra linguagem de servidor.

O que o AJAX não é...

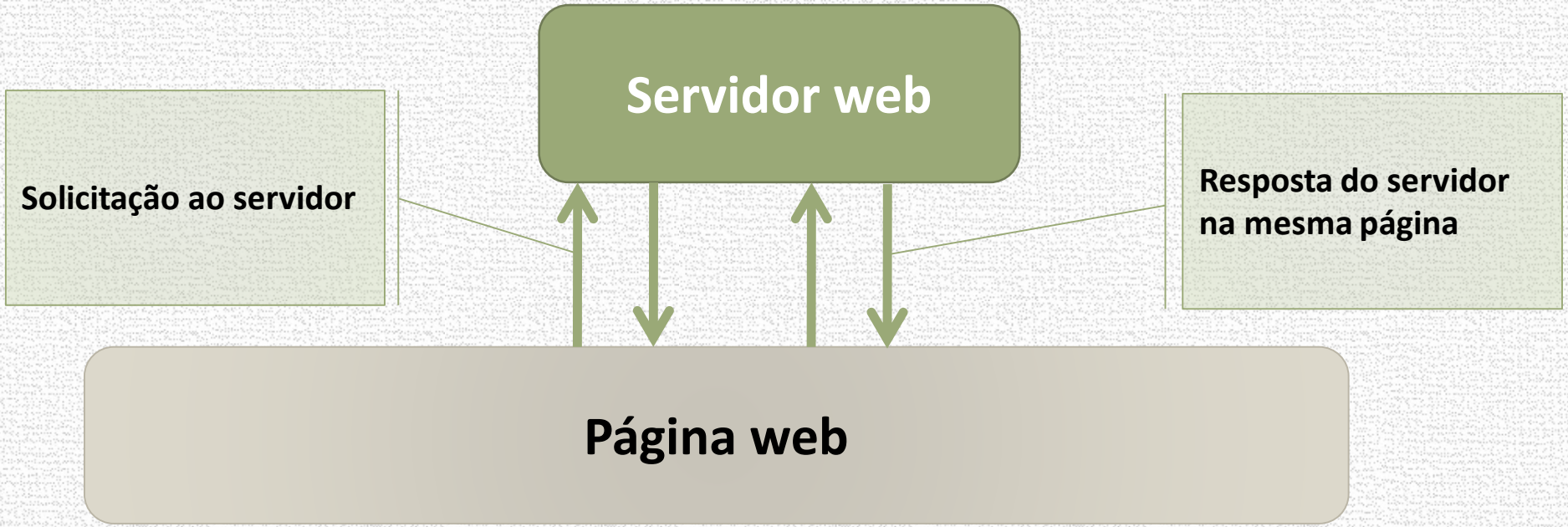
- AJAX não é uma nova linguagem de programação;
- AJAX não é um novo ambiente de desenvolvimento;
- AJAX não é um novo conjunto de bibliotecas ou um recente framework para JavaScript (como jQuery, Prototype, MooTools, etc...);
- AJAX não é uma forma de representação de dados (como o JSON).

Solicitação/resposta sem AJAX (clica e espera)



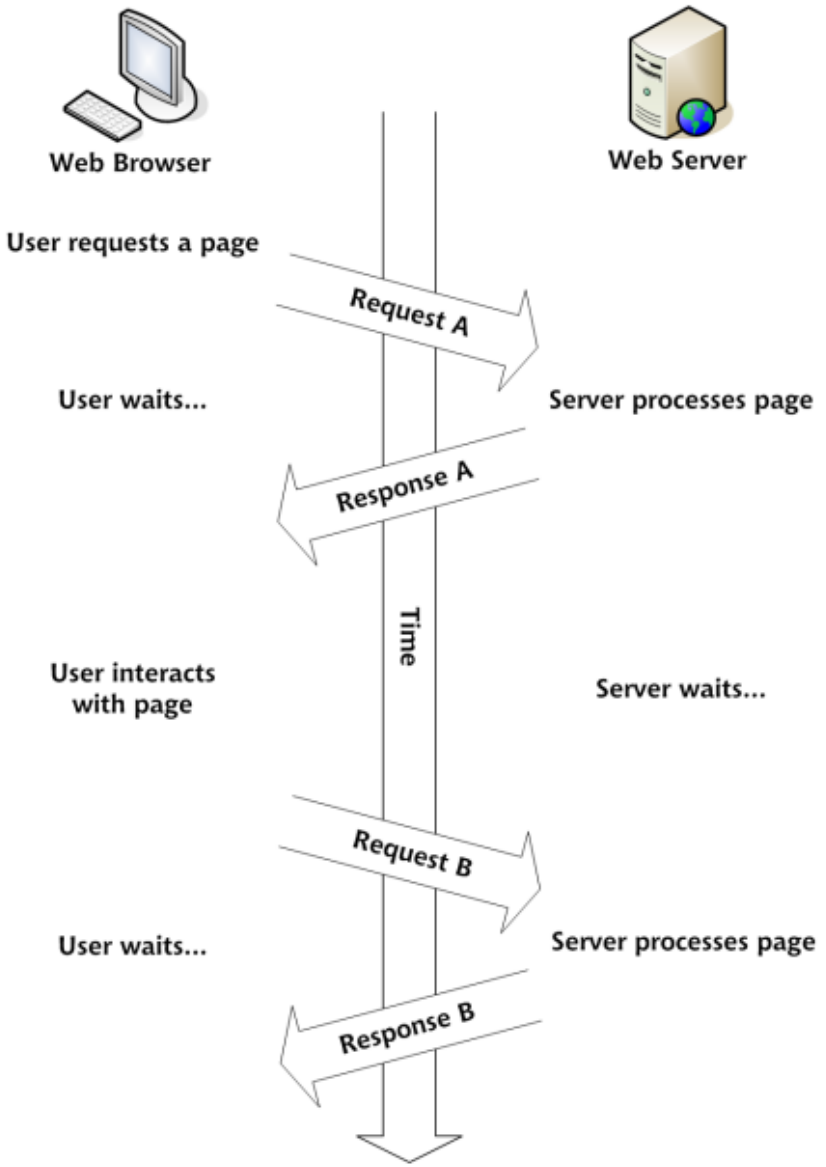
A execução do código no cliente fica congelada até que o servidor envie a resposta da solicitação, gerando uma nova página.

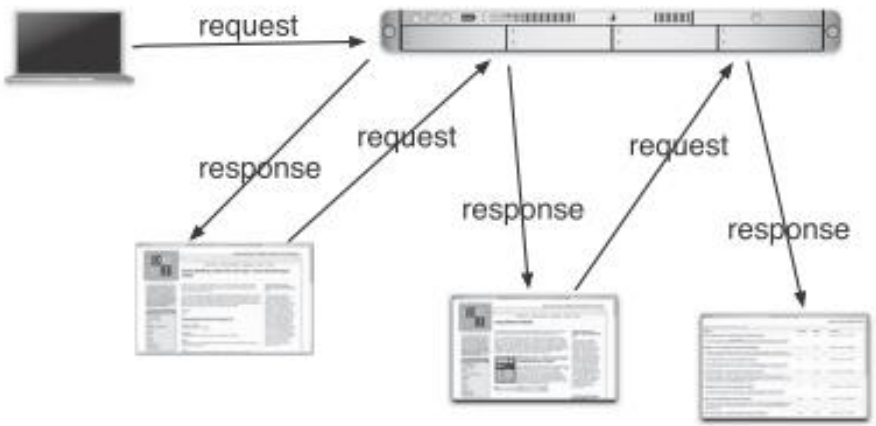
Solicitação/resposta com AJAX (clica e continua)



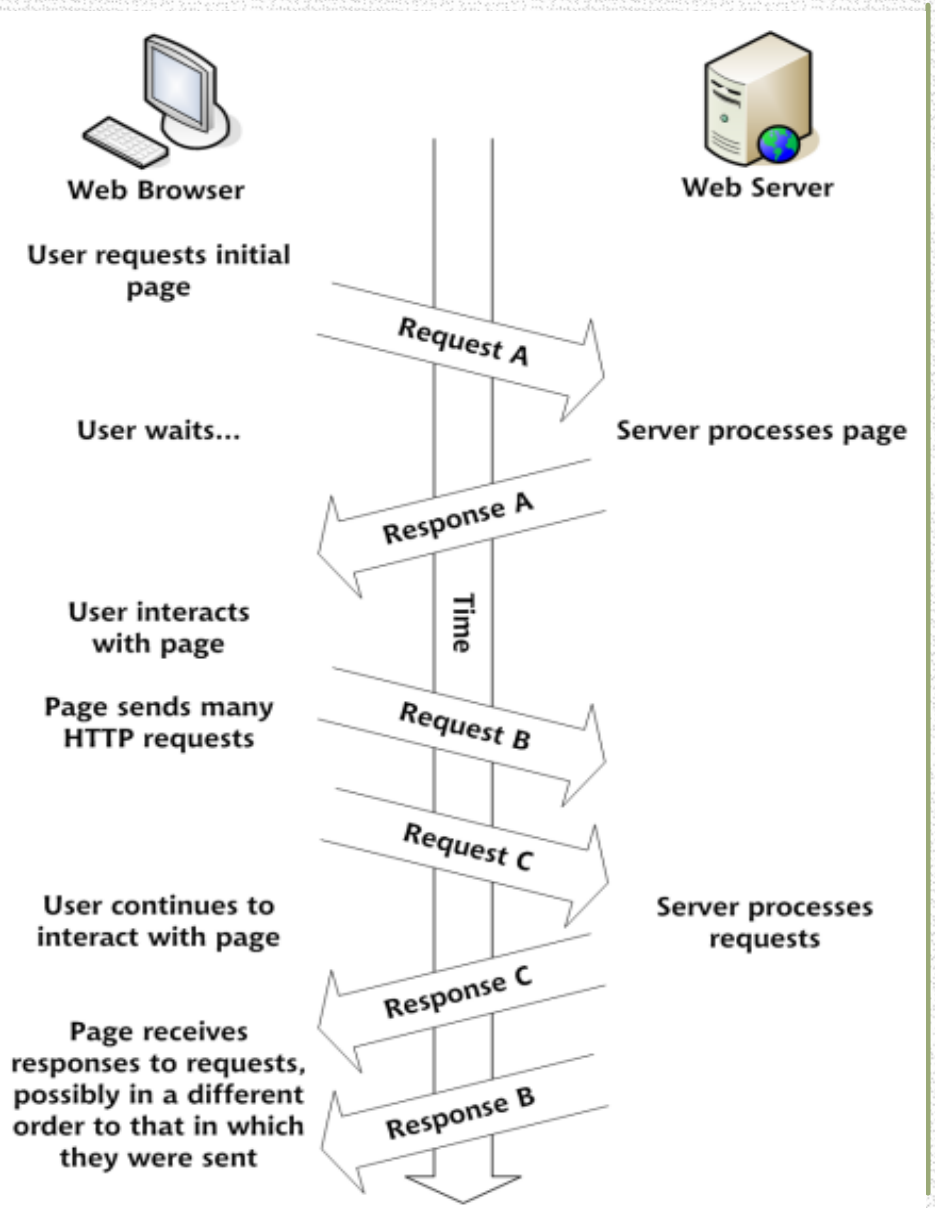
- ✓ O termo “assíncrono”, agora, se refere ao fato de que, enquanto a resposta de uma solicitação ainda não chegou ao cliente, o navegador pode ir processando o restante da página normalmente, sem precisar parar e aguardar pelo retorno do resultado que está enviado pelo servidor.

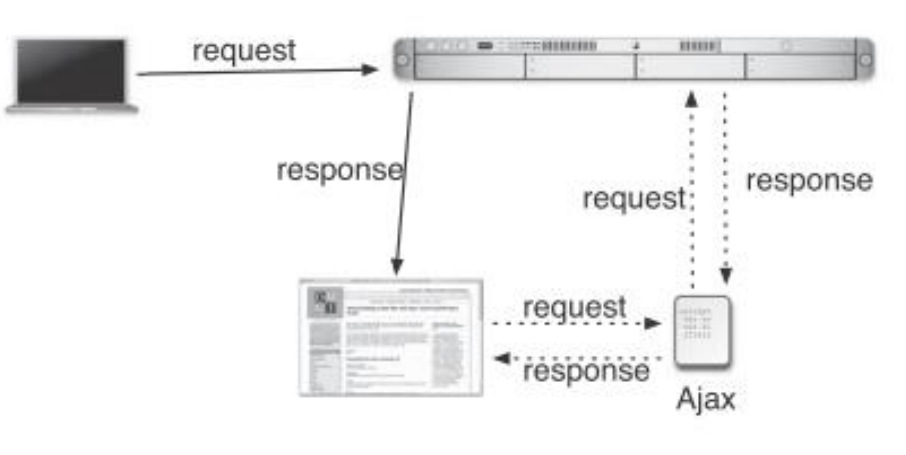
Comunicação síncrona – sem AJAX





Comunicação assíncrona – com AJAX





Quando não usar AJAX?

- Páginas em que a resposta ou atualização gera um conteúdo muito extenso;
- Manipulação dinâmica da interface gráfica da página web. Exemplo: criação dinâmica de menus, formatação de elementos, estilização de objetos, etc... Deixe isso a cargo do CSS ou JavaScript;
- Aplicações web que geram páginas totalmente diferentes umas das outras quando o usuário interagem com elas (o que justificaria uma nova página gerada no formato de comunicação tradicional).
- ***Observação:*** **deve-se escolher com cuidado o local da página onde o resultado da resposta AJAX será mostrado.**

Tipos de dados retornados

- Uma resposta em AJAX retorna dados ao cliente em diversos formatos. Os dois mais comuns são:
  - **XML** – a informação retorna como um arquivo no formato XML;
  - **Texto puro** – a informação retorna como uma string única. Ideal quando o servidor devolve código em HTML;
  - **JSON** – (JavaScript Object Notation) Notação do JavaScript para objetos. A informação é recebida como uma string representando um objeto no formato JSON.

Criação do objeto XMLHttpRequest

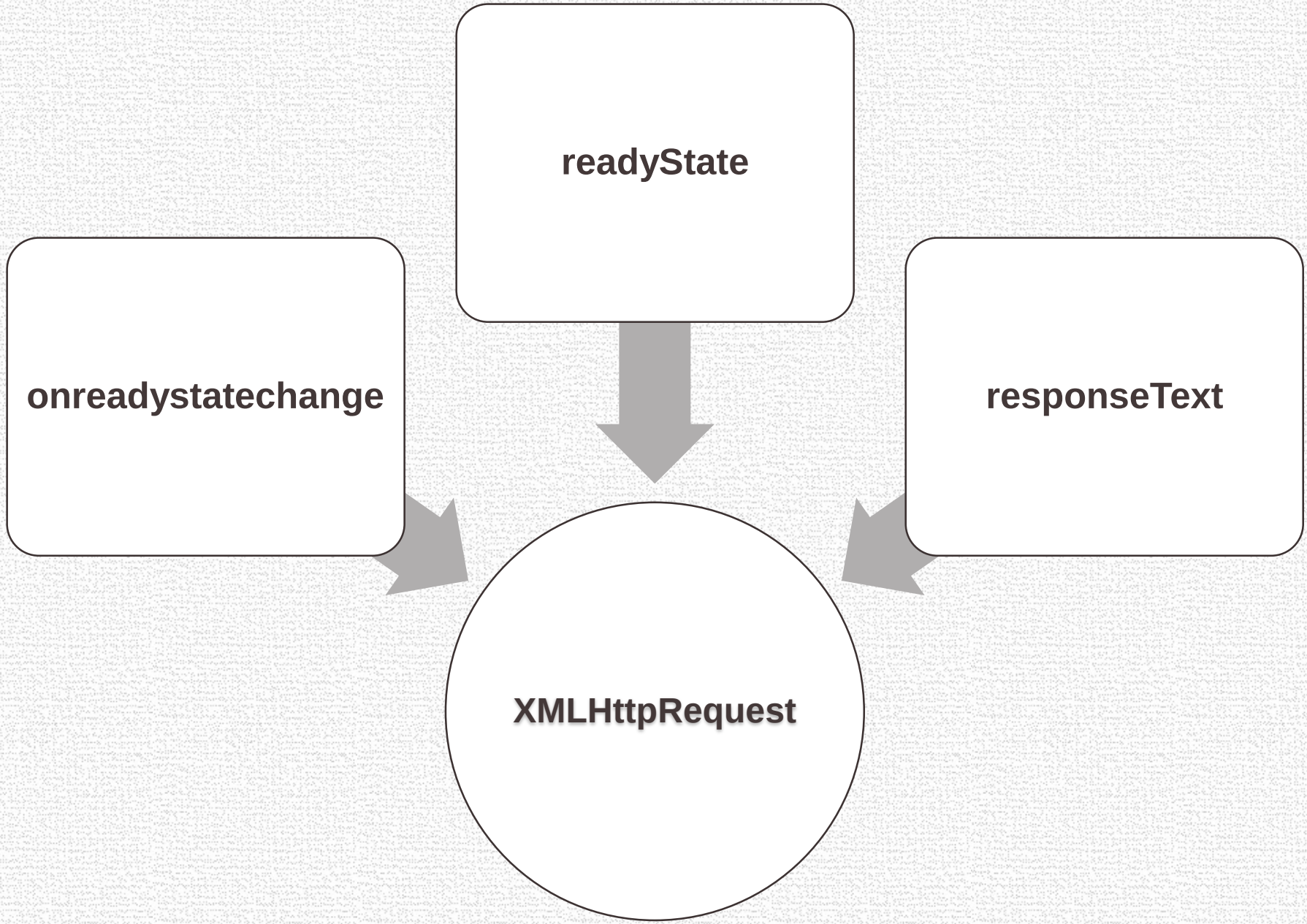

```
function criaAjax()
 {
 var ajax = false;
 if(window.XMLHttpRequest) //testa se o objeto ajax
está disponível no Javascript do navegador que está
executando a aplicação, na forma de uma propriedade
do objeto window
  ajax = new XMLHttpRequest(); //cria o objeto AJAX
em navegadores modernos
 else
  ajax = new ActiveXObject("Microsoft.XMLHTTP");
//cria o objeto AJAX nos navegadores IE5 e IE6

 return ajax;     //retorna o objeto criado ou  false
 }
```

O objeto XMLHttpRequest

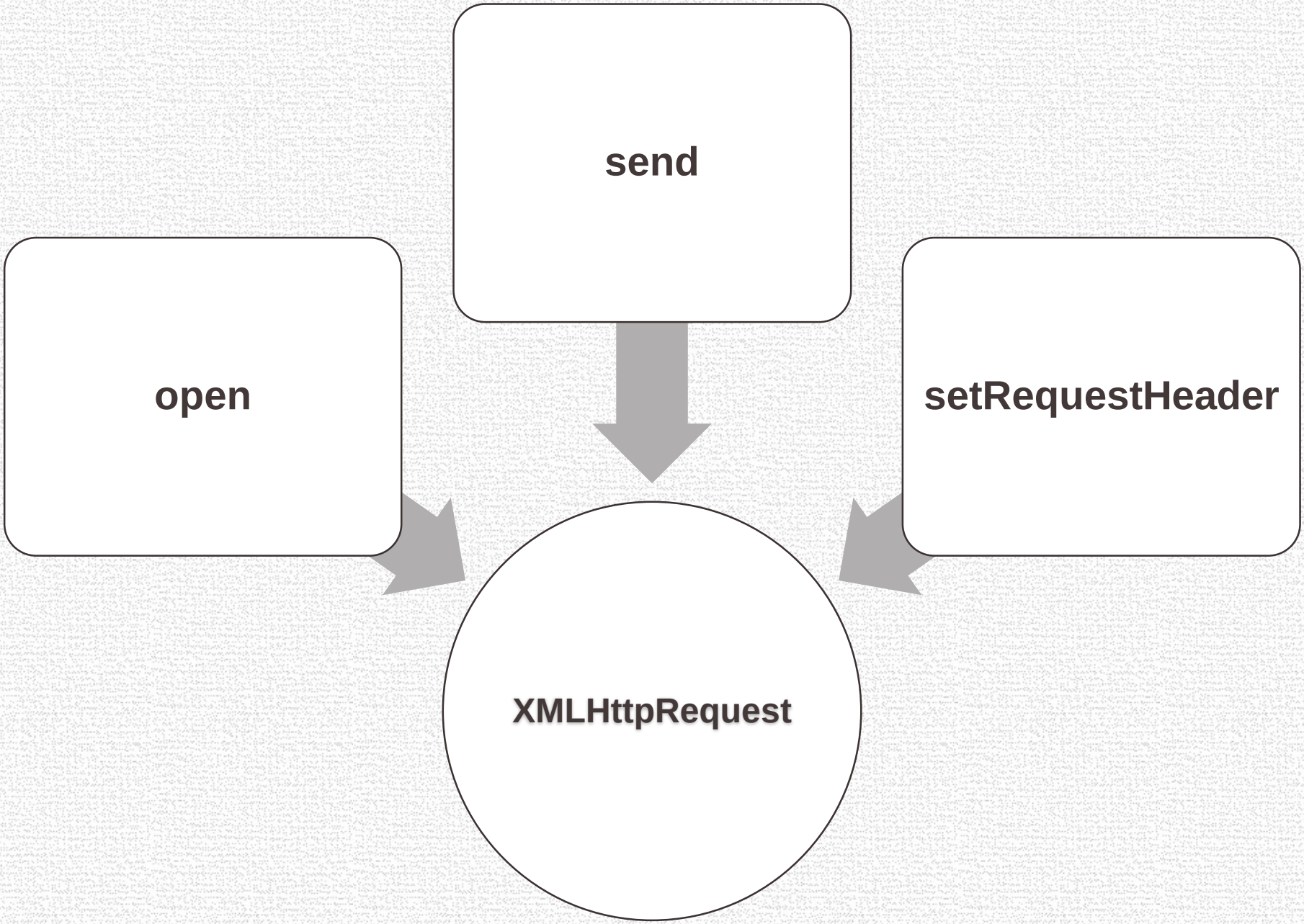
- Métodos e propriedades

Propriedades	Métodos
onreadystatechange	abort
readyState	getAllResponseHeaders
responseText	getResponseHeader
responseXML	open
status	send
statusText	setRequestHeader

XMLHttpRequest – propriedades mais comuns



XMLHttpRequest – métodos mais usados



Métodos mais importantes

Método	Descrição
abort( )	Cancela a solicitação Ajax em andamento
open( )	Estabelece uma conexão entre o JavaScript (máquina-cliente) e o servidor web (máquina-servidora)
setRequestHeader( )	Altera o valor de um cabeçalho HTTP, enviando ao servidor diversos tipos de informação, tais como: comprimento dos dados, tipo de documento, método de envio, etc...
send( )	Envia ao servidor a solicitação Ajax

Propriedades mais importantes

Propriedade	Descrição
onreadystatechange	Define a função JavaScript que irá manipular os dados retornados pelo servidor em resposta a uma solicitação Ajax (XML, texto ou JSON)
readyState	Retorna o estado do objeto XMLHttpRequest após a solicitação ao servidor
responseText	Retorna a resposta do servidor como texto puro
status	Retorna o estado da solicitação Ajax

Método abort( ) do objeto ajax

- Encerra sumariamente a conexão Ajax com o servidor. Pode ser usado nos casos em que o servidor está demorando muito para devolver a resposta.

- Exemplo: `ajax.abort();`

Método setRequestHeader( )

- Altera o valor de uma cabeçalho HTTP. **Sintaxe:**

```
ajax.setRequestHeader("cabeçalho", "valor");
```

Uso	Significado
ajax.setRequestHeader("Content-Type","text/xml");	Tipo de dados sendo enviado
ajax.setRequestHeader("encoding","ISO-8859-1");	Conjunto de caracteres sendo enviado
ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");	Método POST para envio de formulários
ajax.setRequestHeader("Content-length", strData.length);	Tamanho dos dados sendo enviados

Método open( )

- Abre uma conexão Ajax com o servidor. **Sintaxe:** `ajax.open("método", "URL", "assinc", "usuário", "senha");`

Parâmetro	Significado
Método (exigido)	Método de envio dos dados. Geralmente, GET ou POST. Escreva em maiúsculas para evitar problemas em alguns navegadores
URL (exigido)	Nome do script no servidor que receberá a solicitação Ajax
Assinc (opcional)	True ou False. Se não especificado, o padrão é True. True indica que a solicitação é assíncrona e o navegador continuará executando o restante do script, sem esperar uma resposta do servidor. Se for false, a resposta deve ser recebida antes de a execução do script continuar.
usuário/senha (opcional)	Nome e senha de usuário, se o servidor que irá atender a solicitação exigir autenticação

Método open( ) - exemplos


```
ajax.open("GET", "trataForm.php",
true);

ajax.open("POST", "validaForm.jsp",
false, "root", "aluno");

ajax.open("POST", "olaMundo.php");

ajax.open("GET",
"script.php?idade=30&conta=008642");
//URLencode
```

Método send( )

- Envia a requisição ao servidor. **Sintaxe:** `ajax.send("valor");`

Valor	Significado
null	Neste caso, o objeto Ajax não está enviando nenhum dado ao servidor. Apenas, eventualmente, recebendo
dados	Envia os dados especificados ao servidor. Estes dados podem estar em diversos formatos, como texto ou **URLencode**. Geralmente utilizado com formulários.

Exemplos:

```
ajax.send(null); ou
ajax.send("servidor.php?CEP=88010-001&fone=30124056");
```

Propriedade onreadystatechange

- Dispara uma função JavaScript de acordo com o estado do objeto Ajax. Esta função irá tratar os dados recebidos do servidor, após a solicitação retornar ao cliente. Esta função pode ser anônima ou uma literal de função. Forma geral:

```
ajax.onreadystatechange = function() {comandos};
```

EXEMPLO

```
ajax.onreadystatechange = trataResposta; //esta
função pode estar implementada em qualquer lugar do script

function trataResposta()

{

alert('Dentro desta função, receberemos os dados
ajax vindos do servidor e faremos todo o
processamento para incluí-los na página que
enviou a solicitação');

}
```

Propriedade readyState

- Retorna o estado do objeto Ajax (o que está acontecendo com a solicitação). **Sintaxe:** `var estado = ajax.readyState;`

Valor retornado	Significado
0	Objeto Ajax não inicializado – nada ainda começou a ser enviado para o servidor – open() ainda não foi chamado
1	Objeto Ajax aberto – open() já foi chamado, mas send() ainda não
2	O método send() foi chamado e a requisição foi enviada
3	O Ajax está recebendo dados do servidor
4	Todos os dados foram recebidos e a conexão com o servidor foi fechada

Propriedade status

- Retorna o resultado da solicitação ajax enviada pelo servidor, depois que a conexão XMLHttpRequest já foi encerrada. **Sintaxe:**

```
var estado = ajax.status;
```

Valor retornado	Significado
200	Arquivo no servidor foi encontrado e a solicitação atendida corretamente
403	O objeto Ajax não tem as permissões necessárias para acessar o arquivo solicitado no servidor
404	O objeto Ajax não conseguiu localizar o arquivo que trataria a solicitação no servidor
500	Erro no servidor
503	Servidor está sobrecarregado e não pode atender a solicitação do objeto Ajax

Propriedade responseText

- Retorna os dados do servidor no formato de texto puro. **Sintaxe:**

```
var retorno = ajax.responseText;
```

Um exemplo completo

```
ajax.open("GET", "produto.php", true);
ajax.onreadystatechange = function() //trata a solicitação
 {
 if(ajax.readyState == 4) //objeto terminou de carregar e
conexão já se encerrou
   {
   if(ajax.status == 200) //solicitação atendida
     {
     alert(ajax.responseText);//exibe o texto de resposta
     }
   }
 }
ajax.send(null);
```

Passos para a execução de uma solicitação AJAX

- **Passo1)** criar o objeto XMLHttpRequest
- **Passo2)** criar a função de tratamento dos dados de retorno associando-a a onreadystatechange
- **Passo3)** testar as propriedades readyState e status, dentro da função definida no passo anterior
- **Passo4)** coletar os dados a serem enviados ao servidor (no caso de uma requisição POST ou GET com formulário)
- **Passo5)** estabelecer uma conexão com o servidor por meio do método open(`argumentos`)
- **Passo6)** usar o método send(`var dados`) para enviar dados de formulário ao servidor (com GET ou POST), ou usar send(`null`) quando nada precisa ser enviado ao servidor

O DOM (Document Object Model)

- O DOM (modelo de objetos de documento) é, simplificadamente, uma representação, que o JavaScript faz, de todos os elementos utilizados em uma página web;
- Esta representação é feita de forma hierárquica, utilizando-se uma estrutura em forma de árvore. Cada elemento neste modelo recebe o nome de **nó** ou **nodo**;
- Todos os elementos de uma página podem ser acessados e alterados através do DOM. Novos elementos podem ser dinamicamente criados por meio do DOM;
- A linguagem JavaScript possui objetos, métodos e atributos específicos para tratar os elementos de uma página através do DOM, manipulando adequadamente documentos HTML e XML.

Acessando elementos pelo DOM

- Existem várias formas de se alcançar um elemento de uma página HTML, de modificá-lo, de destruí-lo ou, mesmo até, de criar um novo elemento;
- Adiante, veremos os meios mais comuns de se fazer isso.

document.getElementById("id")

- Forma muito usual, já vista em muitos de nossos exemplos.

```
<script>
   var elemento = document.getElementById("caixa");
   if(elemento)
   alert(elemento.value);
</script>
```

No formulário:

```
<input type="text" id="caixa" name="caixa" value="Olá!">
```

getElementsByTagName("nome-da-tag")

- Este método armazena em um vetor JavaScript todos os elementos de uma página que coincidem com o nome da tag passada como parâmetro, na ordem em que eles aparecem no documento. Este método pode ser atribuído a um elemento específico da página, e não somente ao objeto *document*.
- Exemplo: mudar dinamicamente a cor da fonte dos links:

```
<a href="#"> link 1 </a>

<a href="#"> link 2 </a>
```

- No script:

```
<script>

  var links = document.getElementsByTagName("a"); //vetor

  for(var i = 0; i < links.length; i++)

        links[i].style.color = "#0000176";

</script>
```

getElementsByClassName("nome-da-classe")

- Este método permite alcançar todos os elementos de uma página que contenham o mesmo nome de classe. Estes são armazenados em um vetor JavaScript , na ordem em que aparecem no documento.
- Em uma página:

```
<a href="#" class="menu"> link 1 </a>

<a href="#" class="menu"> link 2 </a>
```

- No script:

```
<script>

   var links = document.getElementsByClassname("menu"); //vetor

   for(var i = 0; i < links.length; i++)

      links[i].style.color = "#0000176";

</script>
```

Caso particular – acesso a elementos de formulários

- Em um formulário, todos os elementos podem estar disponíveis para o JavaScript por meio de seus atributos ***name***. O name de um elemento dentro do formulário torna-se uma propriedade do objeto formulário.
- Exemplo:

```
<script>
   var caixa = document.form1.login;
   var rotuloDobotao = document.form1.enviar.value;
</script>
```

No formulário:

```
<form name="form1">
 <input type="text" name="login" value="Olá!" />
 <input type="submit" name="enviar" value="Conecte-se" />
</form>
```

Manipulando o DOM com AJAX e JavaScript

- Pode-se dizer que este processo tem o mesmo objetivo de mostrar ou ocultar elementos em uma página usando puro JavaScript, só que de uma forma mais avançada;

- Permite que sua página apresente objetos apenas no instante em que eles são necessários, não sendo preciso criá-los desde o momento em que a página é carregada. Além do mais, estes elementos podem ter seu conteúdo atualizado pelo servidor, tornando a aplicação verdadeiramente dinâmica.

Criar elementos – createElement("nome-do-elemento")

- **Passo 1)** criar o elemento com **createElement('tag');**
- **Passo2)** manipular as propriedades do elementos como quiser;
- **Passo3)** incorporar o elemento criado ao seu elemento-pai com **appendChild();**

Excluir elementos – removeChild("nome-do-elemento")

- **Passo 1)** acessar o elemento-pai do objeto a ser excluído por meio do método **parentNode()**;

- **Passo2)**excluir o elemento em questão por meio do método **removeChild()**.

Ajax, PHP, JavaScript e a linguagem XML

- A linguagem XML destina-se a fornecer um formato padronizado que facilita o intercâmbio e armazenamento de qualquer tipo de informação. Grosso modo, podemos dizer que o XML permite que criemos nossas próprias tags;
- O Ajax relaciona-se muito intimamente com a linguagem XML, a tal ponto de utilizá-la como uma das formas de recuperar dados do servidor em uma determinada requisição;
- Um documento XML, assim como o próprio HTML, também possibilita que seus elementos sejam acessados através de uma estrutura hierárquica: a árvore de elementos DOM;
- Tanto a linguagem JavaScript quanto a linguagem PHP podem manipular os elementos (nós) de uma documento XML e utilizar estes elementos para trocar dados entre as duas linguagens.

Estrutura de um documento XML


```
<?xml version="1.0" encoding="iso-8859-1"?>
<funcionarios>
   <funcionario ID="2107">
      <nome>Alessandra</nome>
      <salario>1000</salario>
   </funcionario>

   <funcionario ID="0200">
      <nome>Cid Onir</nome>
      <salario>700</salario>
   </funcionario>

   <funcionario ID="1020">
      <nome>Bianca</nome>
      <salario>980</salario>
   </funcionario>

   <funcionario ID="0131">
      <nome>Hermógenes</nome>
      <salario>688</salario>
   </funcionario>

   <funcionario ID="0011">
      <nome>Romário</nome>
      <salario>269</salario>
   </funcionario>
</funcionarios>
```

Escrita básica de um documento XML

Elementos	Exemplo
declaração	`<?xml version="1.0" encoding="iso-8859-1"?>`
elemento-raiz	`<funcionarios>` (equivale ao <html>)

elementos-filhos	`<funcionario>`
	`<nome>`
	`<salario>`

fechamento	`<salario> </salario>`
comentário	`<!-- como no HTML -->`

Observação: tags XML, como no XHTML, também são sensíveis ao caso