

Programação Web

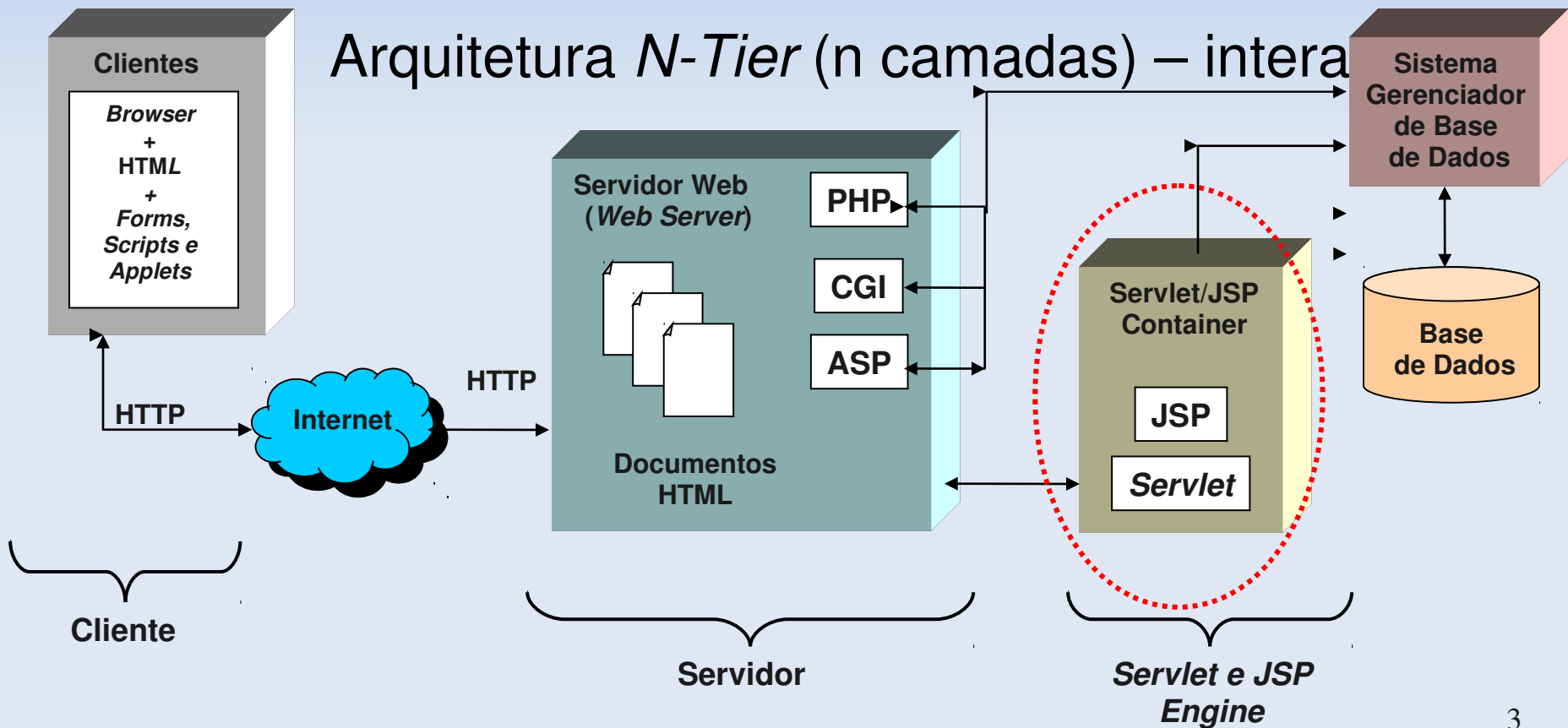
Slides inicialmente elaborados pelo prof. Rudinei Goularte e profa. M. Graça Pimentel

- Renata Pontin

Agenda

- Arquitetura Web
- Manipulação de Documentos Estruturados
- JavaScript
- AJAX

Arquitetura Web



Programação no Cliente

- DOM & SAX
- Javascript
- Ajax

Manipulação de Documentos Estruturados

XML

XML

- *Extensible Markup Language* (XML).
- Linguagem de marcação.
- “arquivos textos de forma padronizadas e com informações separadas”.
- Contém *tags* e metadados.

Exemplo de XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited with XML Spy v2007 (http://www.altova.com) -->
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  .
  .
  .
</bookstore>
```

Aplicações

- Aplicações com XML envolvem manipulação dos arquivos.
- Manipular ou processar arquivos
 - Localizar informações.
 - Inserir novos dados.
 - Remover dados.
 - Modificar dados.

Exemplo de aplicações

- Troca de informações entre processos.
- Troca de informações entre serviços.
- Gravação e recuperação de dados históricos de alguma aplicação.
- Cadastro de clientes.
- Instituições de Ensino.
- Indústrias.
- Comércio.
- Etc.

Técnicas de Manipulação de arquivos XML

- Criação de analisadores léxicos, onde letra por letra é examinada obtendo assim as informações desejadas.
 - Muito trabalhoso.
- *Simple API for XML (SAX).*
- *Document Object Model (DOM).*
- Outras?

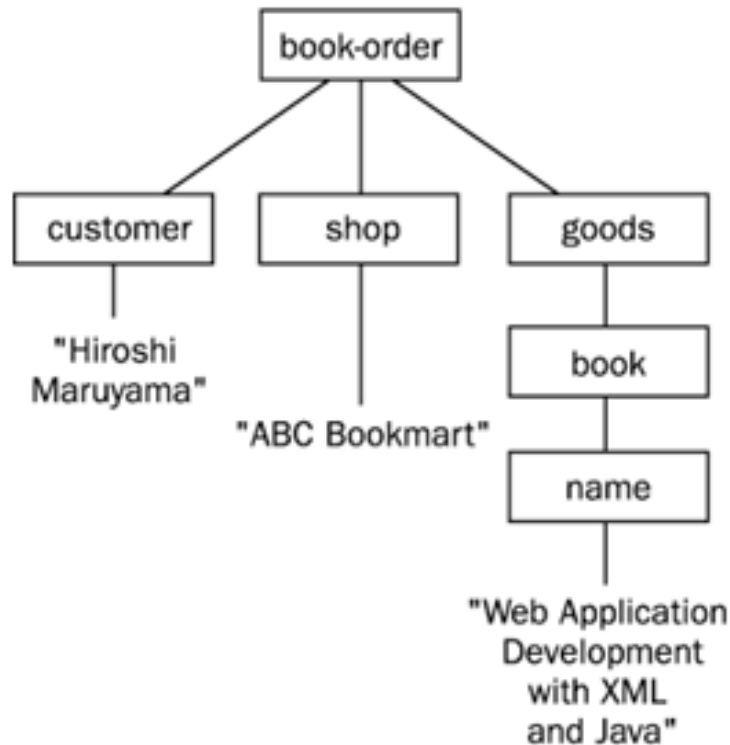
```

<?xml version="1.0" encoding="utf-8"?>
<book-order>
  <customer>Hiroshi Maruyama</customer>
  <shop>ABC Bookmart</shop>
  <goods>
    <book>
      <name>Web Application Development with
XML and Java</name>
    </book>
  </goods>
</book-order>

```



Parsing



```

startElement: book-order
startElement: customer
characters: Hiroshi Maruyama
endElement: customer
startElement: shop
characters: ABC Bookmart
endElement: shop
startElement: goods
startElement: book
startElement: name
characters: Web Application
Development with XML and Java
endElement: name
endElement: book
endElement: goods
endElement: book-order

```

Simple API for XML (SAX)

- Desvantagem
 - Não possui muitas operações de manipulação dos dados.
- Vantagem
 - Não necessita armazenar ou carregar todo o conteúdo do arquivo em memória.
 - É recomendado para arquivos grandes.

Document Object Model **(DOM)**

- Vantagem
 - Possui API robusta para a manipulação dos dados.
- Desvantagem
 - Consumo de memória, pois armazena todo o documento em memória em forma de árvore.
 - Não recomendado para arquivos grandes.

DOM

- Encontra-se na terceira versão.
- Pode ser utilizado para HTML também.
- É um padrão W3C.
- XML DOM é um modelo de objeto de documento para XML.
- Independente de linguagem e plataforma.
- Define um padrão para a manipulação de documentos XML.
- Define um padrão para acessar documentos XML, objetos.
- “Parser de arquivos estruturados”.

XML DOM e Árvore

- DOM é utilizado para manipulação de arquivos.
- Carrega os arquivos em forma de árvore na memória.
- Todos os elementos são manipulados como nós da árvore.
- Relação hierárquica entre os nós
 - Pais, filhos, vizinhos, raiz e nós folhas
- Manipular árvores e nós já é de conhecimento dos programadores -> Estrutura de Dados.

Árvore XML DOM - Definição

- O primeiro nó é chamado de nó raiz.
- Todo nó, exceto o nó raiz possui um nó pai.
- Um nó pode ter qualquer número de filhos.
- Um nó folha é um nó sem filhos.
- Irmãos são os nós com o mesmo pai.

Árvore DOM

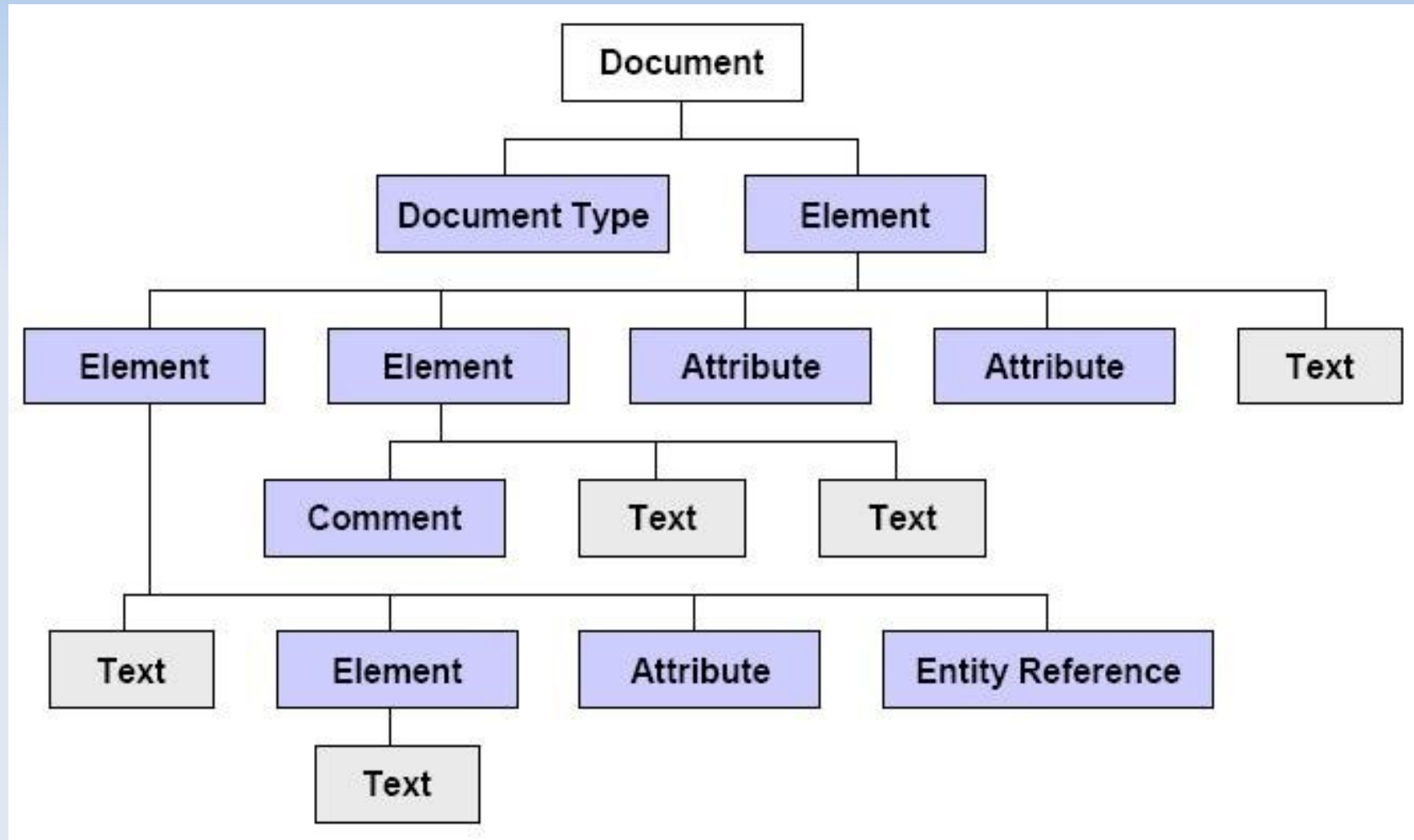


Figura 1, Árvore DOM, adaptada de (Hall M., 2007)

Árvore DOM

- Objetivo
 - Determinar e mostrar como os dados serão representados e manipulados em memória.

Árvore XML DOM

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cadastro>
  <peessoa tipo="FISICA">
    <nome>Matheus Q. Barbosa</nome>
    <nascimento>30_05_1981</nascimento>
    <email>barbosamq@gmail.com</email>
  </peessoa>
  .
  .
  .
  <peessoa>
    . . .
  </peessoa>
</cadastro>
```

Figura 2: arquivo exemplo XML.

Exemplo de árvore XML DOM

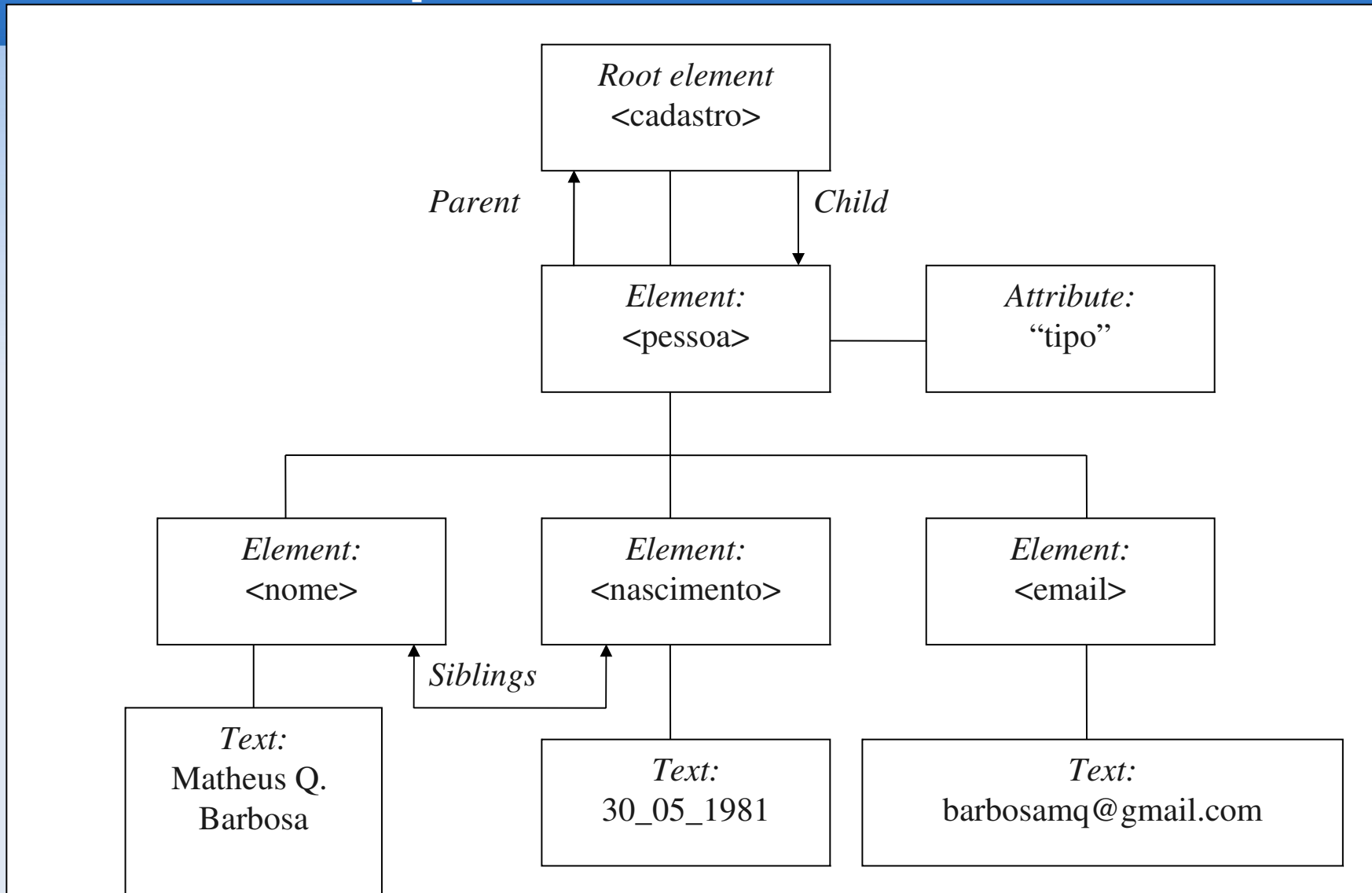


Figura 3. Árvore XML DOM na memória, representando o arquivo XML exemplo²⁰

Árvore XML DOM: Grau de Parentesco e navegação

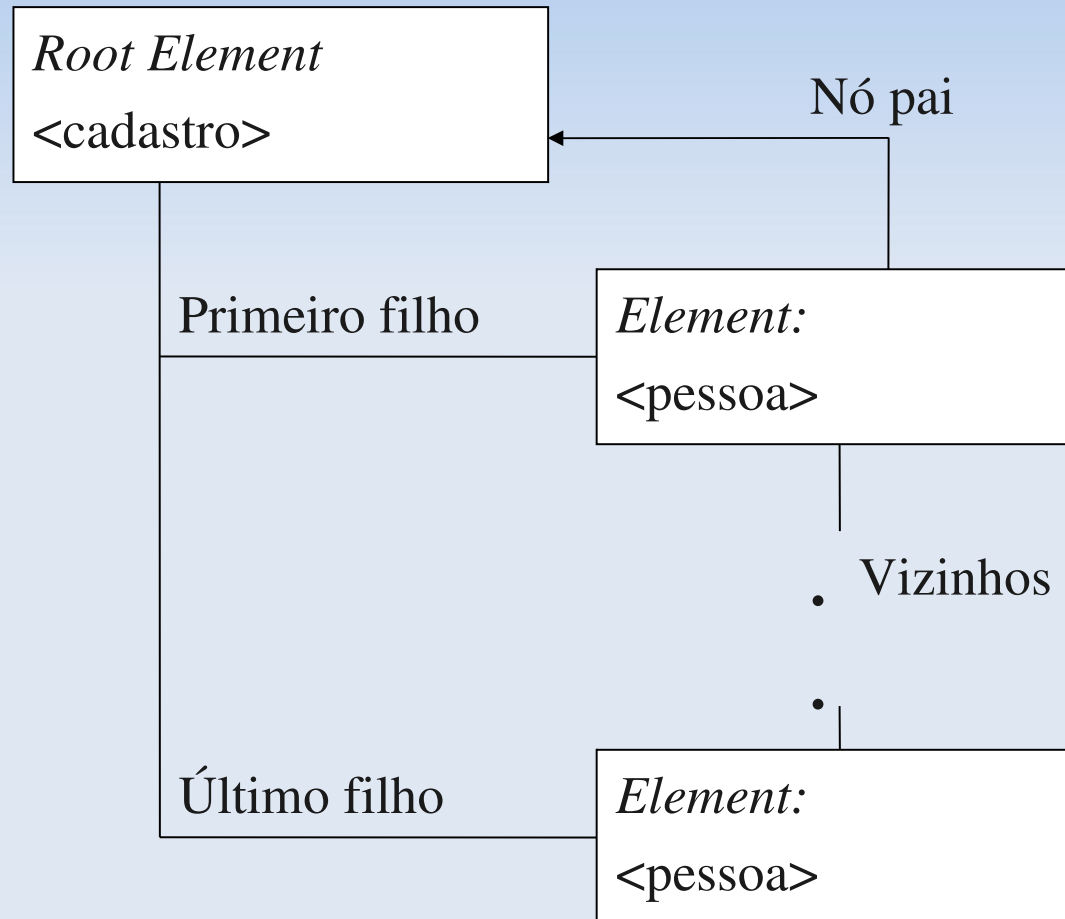


Figura 4, Árvore DOM carregada em memória referente ao arquivo exemplo.xml²¹

Tipos de nós das árvores XML DOM

Tipo do nó	Descrição	Filhos
<i>Document</i>	Representa todo o documento. É o nó raiz da árvore DOM.	<i>Element (max. one), ProcessingInstruction, Comment, DocumentType</i>
<i>DocumentFragment</i>	Representa um tipo de objeto documento, o qual pode manter uma porção de documentos.	<i>Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference</i>
<i>DocumentType</i>	Provê uma interface para as entidades definidas por um documento.	Nulo

Tipos de nós das árvores XML DOM

<i>Processing Instruction</i>	Representa uma instrução de processamento.	Nulo
<i>EntityReference</i>	Representa uma entidade de referência.	<i>Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference</i>
<i>Element</i>	Representa um elemento.	<i>Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference</i>
<i>Attr</i>	Representa um atributo.	<i>Text, EntityReference</i>

Tipos de nós das árvores XML DOM

<i>Text</i>	Representa o conteúdo textual em um elemento ou atributo.	Nulo
<i>CDATA Section</i>	Representa um seção CDATA num documento.	Nulo
<i>Comment</i>	Representa um comentário.	Nulo
<i>Entity</i>	Representa uma entidade.	<i>Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference</i>
<i>Notation</i>	Representa uma notação declarada em DTD.	Nulo

Utilização de XML DOM

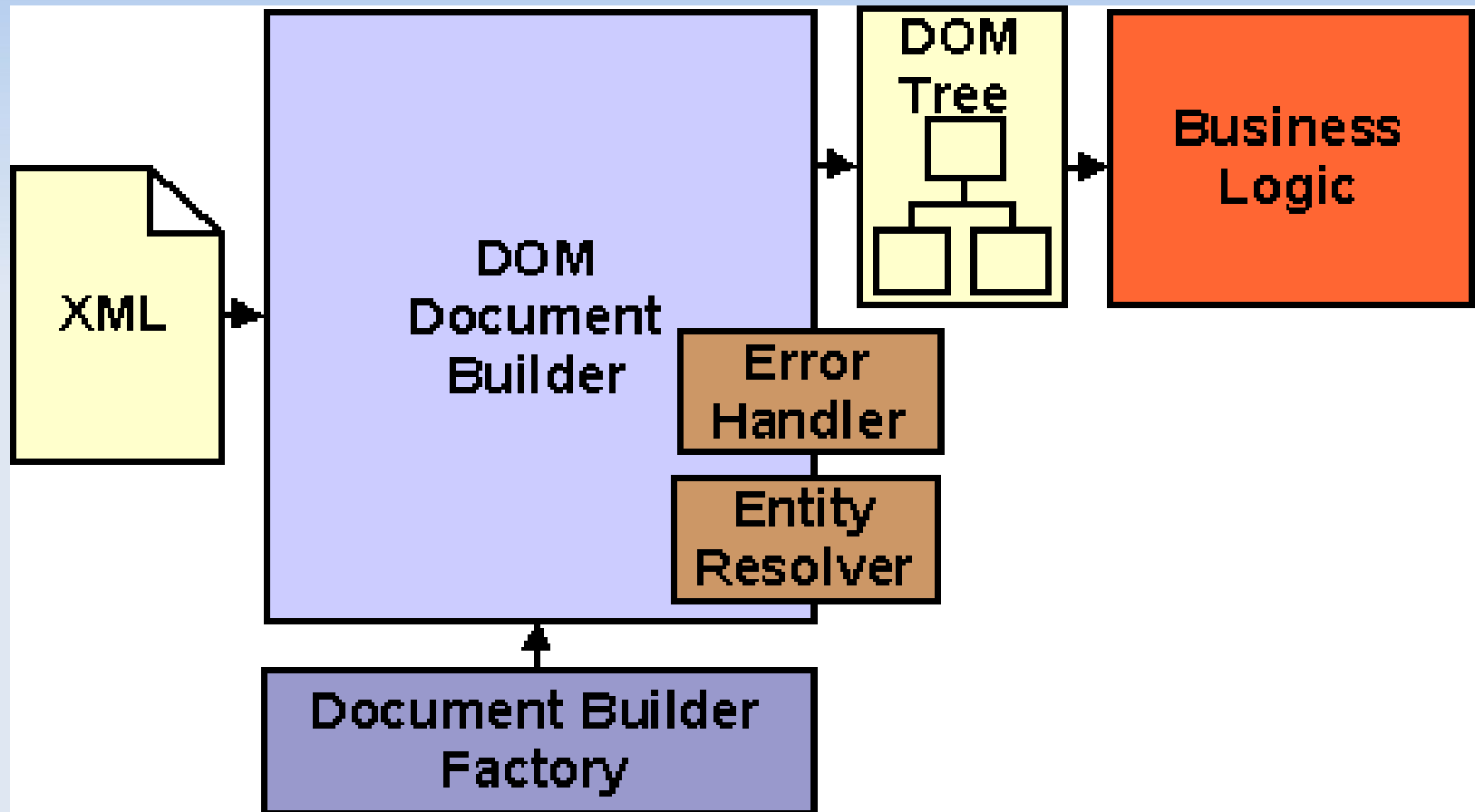


Figura 5, Representação da linguagem DOM, adaptada de (Violleau T., 2001). ²⁵

API JAVA

- **Package org.w3c.dom**
- API que provê interfaces para DOM
 - Java API for XML Processing (JAXP)
- Descrição
 - O DOM nível 2 permite programas acessarem e atualizarem conteúdos e estrutura de documentos dinamicamente.

Carregando um arquivo em memória

Para carregar o arquivo em memória é necessário utilizar um documento e um *parser*.

As linhas de código abaixo fazem isso, retornando o nó raiz da árvore DOM. A navegação será iniciada a partir do nó raiz.

```
DocumentBuilderFactory factory =  
DocumentBuilderFactory.newInstance();  
DocumentBuilder builder = factory.newDocumentBuilder();  
Document document = builder.parse(new File("introduction.xml"));  
Node root = document.getDocumentElement();
```

Localizando um nó na árvore

Uma maneira de localizar um nó é através de uma lista de nós ou utilizando métodos de busca na árvore através do nome da *tag*, no caso a *tag message*.

Existem outras maneiras de realizar a busca.

```
Element mymessageNode = ( Element ) root;  
NodeList messageNodes =  
    mymessageNode.getElementsByTagName( "message" );  
Node message = messageNodes.item( 0 );
```

Criando um nó na árvore

- Uma das maneiras de se criar um nó é fazer o clone de um nó já existente. Após fazer o clone de um nó já existente é possível modificar todas as propriedades do nó e o seu conteúdo.

Node message2 = message.cloneNode(true);

Modificando um nó na árvore

- Uma maneira de trocar a informação do objeto é através do método **replaceChild**, porém as versões mais novas da linguagem possuem novos métodos.
- Pode-se não só trocar as informações da *tag*, mas também trocar atributos.

```
Text nt = document.createTextNode("Informação Nova");  
Text ot = ( Text ) message2.getChildNodes().item( 0 );  
message2.replaceChild( nt, ot );
```

Adicionando um nó na árvore

- Uma maneira de adicionar objetos é através do método **appendChild()**, esse método permite adicionar um nó filho a qualquer outro nó da árvore.

message.getParentNode().appendChild(message2);

Removendo um nó na árvore

- O método que permite a retirada de algum nó da árvore é o **removeChild()**, neste método é necessário informar o nó antigo que será removido.

`message.getParentNode().removeChild(oldText);`

Exemplo 1: DOM em JAVA

- Um exemplo simples apenas para demonstrar como DOM funciona.
- Dado o seguinte arquivo **introduction.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>  
<mymessage>  
    <message>teste</message>  
</mymessage>
```

Exemplo 1: DOM em JAVA

```
//imports Necessários
import java.io.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;
import org.xml.sax.*;
import org.w3c.dom.*;

//Classe ReplaceText
public class ReplaceText {
    private Document document;
    //construtor
    public ReplaceText( String fileName ){...}
    //método main
    public static void main( String args[] ){...}
}
```

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
//carrega o documento na memória construindo a árvore
Document document = builder.parse(new File("introduction.xml"));
Node root = document.getDocumentElement(); //obtem o nó raiz
if ( root.getNodeType() == Node.ELEMENT_NODE ) {
    Element mymessageNode = ( Element ) root;
    //localiza o elemento pelo nome da tag
    //no caso busca o elemento message
    NodeList messageNodes=mymessageNode.getElementsByTagName("message");
    //verifica se tem elemento
    if ( messageNodes.getLength() != 0 ) {
        Node message = messageNodes.item( 0 );
        // cria o nó de texto
        Text newText = document.createTextNode( "Novo texto" );
        // obtem o nó de texto antigo
        Text oldText =( Text ) message.getChildNodes().item( 0 );
        // troca o velho pelo novo
        message.replaceChild( newText, oldText );
    } ...
}
```

Exemplo1: Restante do arquivo

- Constituído dos objetos que vão executar a gravação da árvore em memória no arquivo XML.
- Constituído de tratadores de erros: *catch's*
- Após a execução do programa:
 - Output written to: `introduction.xml`

Exemplo1: Resultado

- **Arquivo Introduction.xml antigo**

```
<?xml version="1.0" encoding="UTF-8"?>  
<mymessage>  
  <message>teste</message>  
</mymessage>
```

- **Arquivo Introduction.xml novo**

```
<?xml version="1.0" encoding="UTF-8"?>  
<mymessage>  
  <message>Novo texto</message>  
</mymessage>
```

Exemplo 2: DOM e JAVA

- Um exemplo de como mostrar todas as tags de um arquivo XML.
- Dado o **introduction.xml** visto anteriormente
- O seguinte programa:

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
public class DOMElements{
    static public void main(String[] arg)...
}
```

Exemplo2: Programa

```
try {
    BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Enter XML File name: ");
    String xmlFile = bf.readLine();
    File file = new File(xmlFile);
    //arquivo carregado na memória
    if(file.exists()){
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(xmlFile);
        // Obtém a lista com todos os elementos do documento
        NodeList list = doc.getElementsByTagName("*");
        System.out.println("XML Elements: ");
        for (int i=0; i<list.getLength(); i++) {
            // Obtém o elemento
            Element element = (Element)list.item(i);
            // Imprime o nome do elemento.
            System.out.println(element.getNodeName());
        }
    } else {...}}catch(Exception e){...}
```

Exemplo2: Resultado

- Dado o arquivo introduction.xml
`<?xml version="1.0" encoding="UTF-8"?>`
`<mymessage>`
 `<message>Novo texto</message>`
`</mymessage>`
- Execução do programa:
Enter XML File name: `introduction.xml`
- **Resultado da execução**
XML Elements:
 mymessage
 message

Conclusão

- **DOM** permite manipulação de arquivos XML em memória.
- Utiliza o conceito de **árvore** para a manipulação dos arquivos.
- Possui vários métodos para manipular nós.
- Independência de plataforma ou linguagem, várias API's para diferentes linguagens de programação.
- É um **padrão W3C**.

SAX

SAX: Event-Driven API

- Um documento XML é processado (*parsing*) em um único passo e a seqüência de eventos é notificada para uma aplicação.
- Exemplo de eventos: leitura de uma tag de início; ocorrência de um atributo; existência de texto.
- Uma aplicação pode registrar *handlers de eventos*.
- Um processador XML baseado em SAX notifica os *handlers de eventos*.

SAX

Interface SAX

- A especificação SAX define uma interface em termos de um conjunto de *handlers* de eventos.
- Uma aplicação deve implementar os métodos da interface associados aos eventos que devem ser tratados.
- Projetada como uma API **leve** que não gera uma estrutura em árvore de um documento XML.

SAX

- Uma interface Java simples
 - Um processador SAX é uma classe que implementa a interface *org.xml.sax.Parser*.
 - Ao percorrer a árvore de nós do documento XML, o parser realiza chamadas para os métodos implementados pela aplicação.

XML

SAX

□ Funcionamento

- Disponibiliza a informação contida em um documento XML através de uma seqüência de eventos.
- Para utilizar SAX é necessário:
 - Definir um modelo de objetos próprio.
 - Criar uma classe que “escuta” eventos SAX e gera o modelo de objetos definido.
- SAX => realmente simples!
 - O parser apenas gera eventos.
 - Cabe à aplicação interpretar esses eventos.

XML

SAX

□ Vantagens

○ Velocidade

- O processo de parsing é bem mais rápido graças à simplicidade da API.
- O documento não necessita estar totalmente na memória para que seu processamento seja iniciado.

○ Não impõe um modelo de objetos.

□ Desvantagem

○ Oferece poucas facilidades às aplicações.

Nome do método	Descrição
<i>startDocument()</i>	Recebe a notificação de início do documento.
<i>endDocument()</i>	Recebe a notificação de final do documento.
<i>startElement(String uri, String localpart, String name, Attributes amap)</i>	Recebe a notificação de início de um elemento.
<i>endElement(String name)</i>	Recebe a notificação de final de um elemento.
<i>characters(char ch[], int start, int length)</i>	Recebe a notificação de dados (texto).
<i>ignorableWhitespace(char ch[],int start, int length)</i>	Recebe a notificação de espaço em branco no conteúdo do elemento.
<i>processingInstruction(String target, String data)</i>	Recebe a notificação de uma instrução de processamento.
<i>setDocumentLocator(Locator locator)</i>	Recebe um objeto para localizar a origem dos eventos do documento. O objeto <i>Locator</i> fornece informação sobre a localização do evento, por exemplo, o número da linha e a

DOM versus SAX

DOM é indicado quando:

- A estrutura de um documento XML deve ser modificada.
- O documento XML deve ser compartilhado na memória com outras aplicações.
- O tamanho dos documentos XML não é MUITO grande.
- As aplicações devem iniciar o processamento depois de concluir a validação.

DOM versus SAX

SAX é indicado quando:

- ❑ A aplicação precisa de memória e desempenho
- ❑ A aplicação não precisa reconhecer a estrutura (complexa) de um documento XML

Referências

- (Armstrong, 2001) Eric Armstrong 2001, Working with XML The Java API for Xml Processing (JAXP) Tutorial, disponível em <http://java.sun.com/webservices/jaxp/dist/1.1/docs/tutorial/index.html>, acessado em 26 de Junho de 2007 as 12hs.
- (Deitel & Deitel, 2002) Harvey M. Deitel & Paul J. Deitel 2002, Java Como Programar, 4.^o Edição, Editora: BOOKMAN, Ano de Edição: 2002, Capítulo 4, Document Object Model (DOM), disponível em: <http://www.inf.ed.ac.uk/teaching/courses/ec/miniatures/dom-up.pdf>, acessado em 26 de Junho de 2007 as 12hs.
- (Hall M., 2007) Marty Hall 2007, Creating and Parsing XML Files with DOM, disponível em <http://courses.coreservlets.com/Course-Materials/pdf/java5/21-XML-and-DOM.pdf>, acessado em 26 de Junho de 2007 as 12hs.
- (Harold, 2002) Elliotte Rusty Harold, 2002, Processing XML with Java, disponível em <http://www.cafeconleche.org/books/xmljava/chapters/index.html>, acessado em 26 de Junho de 2007 as 12hs.
- (Java, 2007) Sun Microsystems, Inc.2007, Java API for XML Code Samples, disponível em <http://java.sun.com/developer/codesamples/xml.html#dom>, acessado em 26 de Junho de 2007 as 12hs.
- (Violleau T., 2001) Thierry Violleau November 2001, Technology and XML Part 1 -- An Introduction to APIs for XML Processing, disponível em <http://java.sun.com/developer/technicalArticles/xml/JavaTechandXML/#code14>, acessado em 26 de Junho de 2007 as 12hs.
- (W3Schools, 2007) Refsnes Data 2007, XML DOM Tutorial, disponível em <http://www.w3schools.com/dom/default.as>, acessado em 26 de Junho de 2007 as 12hs.

JavaScript

A Linguagem JavaScript


- É a linguagem de *scripting* da Web.
 - A mais popular.
 - Adiciona interatividade ao HTML.
 - Normalmente embutida nas páginas HTML.
- Utilizada para *design*, validar formulários, tratar eventos, detectar navegador, criar *cookies*, etc.
- É uma linguagem fácil de se aprender.

A linguagem JavaScript

- Linguagem interpretada.
- Não é a mesma coisa que Java.
- Foi criada em 1995 por Breidan Eich da Netscape.
- Atualmente funciona em todos os mais populares navegadores:
 - Internet Explorer, Mozilla, Firefox, Netscape e Opera.

Onde colocar o código JavaScript?

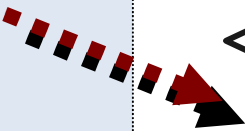
- Colocar o JavaScript em **<head>** e **<body>** são convenções, a maioria dos navegadores aceitam o elemento.



```
<html>  
  <head>  
    <script type="text/javascript">  
      ...  
    </script>  
  </head>  
  ...  
</html>
```

Onde colocar o código JavaScript?

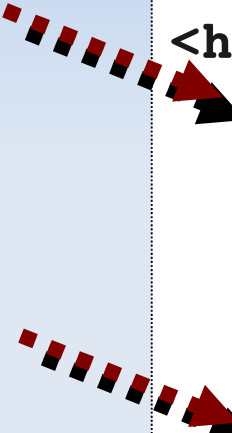
- Dentro do elemento `<body>` de HTML



```
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">
      ...
    </script>
  </body>
  ...
</html>
```

Onde colocar o código JavaScript?

- Colocar uma parte em <head> e outra em <body>



```
<html>
  <head>
    <script type="text/javascript">
      ...
    </script>
  </head>
  <body>
    <script type="text/javascript">
      ...
    </script>
  </body>
  ...
</html>
```


Onde colocar o código JavaScript?

- Pode-se utilizar um documento externo com seu código JavaScript.

```
<html>
  <head>
    <script src="myFunctions.js"></script>
  </head>
  <body>
  </body>
</html>
```

Operadores Aritméticos

Operador	Operação	Exemplo
+	Adição	$x+y$
-	Subtração	$x-y$
*	Multiplicação	$x*y$
/	Divisão	x/y
%	Módulo (resto da divisão inteira)	$x\%y$
-	Inversão de sinal	$-x$
++	Incremento	$x++$ ou $++x$
--	Decremento	$x--$ ou $--x$

Operadores de Comparação

Operador	Função	Exemplo
<code>==</code>	Igual a	<code>(x == y)</code>
<code>!=</code>	Diferente de	<code>(x != y)</code>
<code>===</code>	Idêntico a (igual e do mesmo tipo)	<code>(x === y)</code>
<code>!==</code>	Não Idêntico a	<code>(x !== y)</code>
<code>></code>	Maior que	<code>(x > y)</code>
<code>>=</code>	Maior ou igual a	<code>(x >= y)</code>
<code><</code>	Menor que	<code>(x < y)</code>
<code><=</code>	Menor ou igual a	<code>(x <= y)</code>

Operadores Bit a bit

Operador	Operação	Exemplo
&	E (AND)	(x & y)
	OU (OR)	(x y)
^	Ou Exclusivo (XOR)	(x ^ y)
~	Negação (NOT)	~x
>>	Deslocamento à direita (com propagação de sinal)	(x >> 2)
<<	Deslocamento à esquerda (preenchimento com zero)	(x << 1)
>>>	Deslocamento à direita (preenchimento com zero)	(x >>> 3)

Operadores de Atribuição

Operador	Exemplo	Equivalente
=	x = 2	Não possui
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
&=	x &= y	x = x & y
=	x = y	x = x y
^=	x ^= y	x = x ^ y
>>=	x >>= y	x = x >>= y
<<=	x <<= y	x = x <<= y
>>>=	x >>>= y	x = x >>>= y

Operadores Lógicos

Operador	Função	Exemplo
&&	E Lógico	(x && y)
 	OU Lógico	(x y)
!	Negação Lógica	!x

Variáveis em JavaScript

```
var strname = algumValor;  
//alguns navegadores aceitam sem ";"  
//melhor sempre utilizar o ";"  
  
strname = algumValor;  
// o "var" não é obrigatório  
  
var strname = "Hege";  
// pode-se atribuir valores desta forma  
  
strname = 5;  
// não possui tipificação forte
```

Criando/utilizando funções

```
<html>
  <head>
    <script type="text/javascript">
      function displaymessage(message) {
        window.alert(message);
      }
    </script>
  </head>

  <body>
    <form>
      <input type="button" value="Click me!"
        onclick="displaymessage('Hello World!');" >
    </form>
  </body>
</html>
```


Criando/utilizando funções

```
<html>
  <head>
    <script type="text/javascript">
      function prod(a,b){
        x=a*b;
        //pode ter um return
        return x;
      }

      product=prod(2,3);

    </script>
  </head>
  ...
</html>
```

Outros comandos

- Break Loops
- For...In
- Try...Catch
- Throw
- ...

Programação Orientada a Objetos

- JavaScript é uma linguagem de programação orientada a objetos
 - Objetos podem ser definidos.
 - Tipos de variáveis podem ser criados.
- Propriedades
 - Valores associados a objetos.
- Métodos
 - Ações que podem ser executadas em objetos.

Programação Orientada a Objetos

```
<script type="text/javascript">  
  
var str="Hello world!";  
document.write(str.length);  
document.write(str.toUpperCase());  
  
</script>
```

Criando objetos

```
//cria o objeto personObj

personObj=new Object();
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=50;
personObj.eyecolor="blue";

//pode-se também adicionar métodos

//define o método
function eat(){
    //codigo do método
}

//adiciona o método
personObj.eat=eat;
```

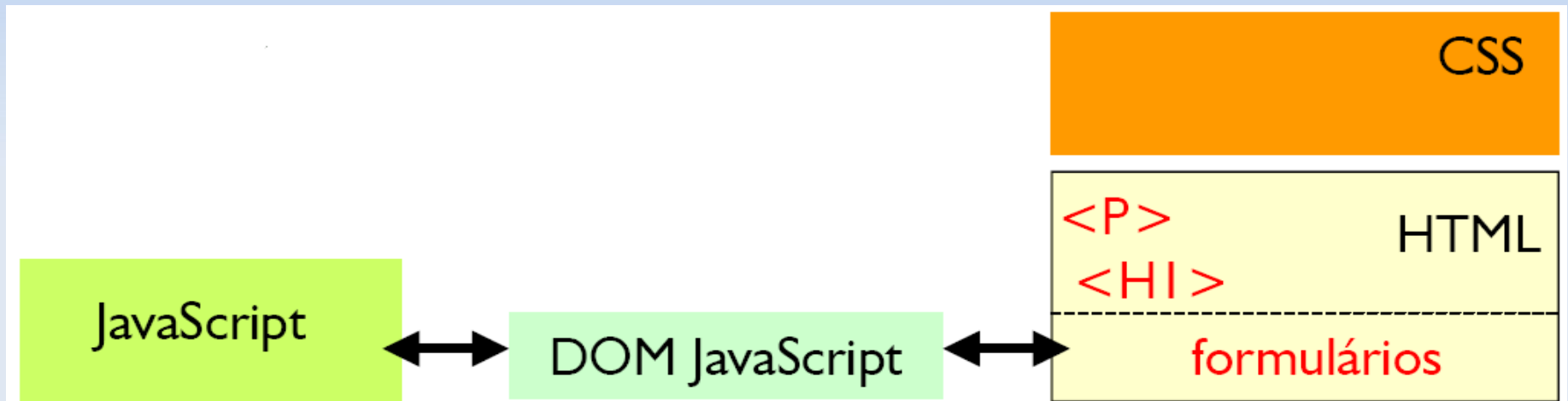
Alguns Objetos de JavaScript

- String
 - http://www.w3schools.com/jsref/jsref_obj_string.asp
- Date
 - http://www.w3schools.com/jsref/jsref_obj_date.asp
- Array
 - http://www.w3schools.com/jsref/jsref_obj_array.asp
- Math
 - http://www.w3schools.com/jsref/jsref_obj_math.asp

JavaScript e DOM

- **Document Object Model** do JavaScript mapeia algumas estruturas do HTML a objetos (variáveis) da linguagem:
 - Propriedades dos objetos (e conseqüentemente dos elementos da página) poderão ser alteradas em tempo de execução.
 - Mapeamento restringe-se a elementos de formulário, vínculos, imagens e atributos da janela do *browser*.
 - Permite validação de campos dos formulários, cálculos locais, imagens dinâmicas, abertura de novas janelas, controle de frames, etc.
 - Não é completa. Não mapeia parágrafos, títulos ou folhas de estilo.

JavaScript e DOM



DOM do W3C

- **Document Object Model** do W3C
 - Mapeia todos os elementos do HTML e folha de estilos, tornando-os acessíveis como objetos JavaScript.
- Desvantagem: compatibilidade
 - A Microsoft utiliza DOM diferente.
 - A W3C tenta padronizar outro.
 - Na prática W3C DOM funciona bem com XML, mas é problemático com HTML.

Referências

- <http://www.w3schools.com/js/>
- Grillo, Filipe del Nero. Aprendendo JavaScript. Notas Didáticas do ICMC-USP, 72, 2008.
- Flanagan, David. JavaScript : the definitive guide. O'Reilly & Associates, c1998.

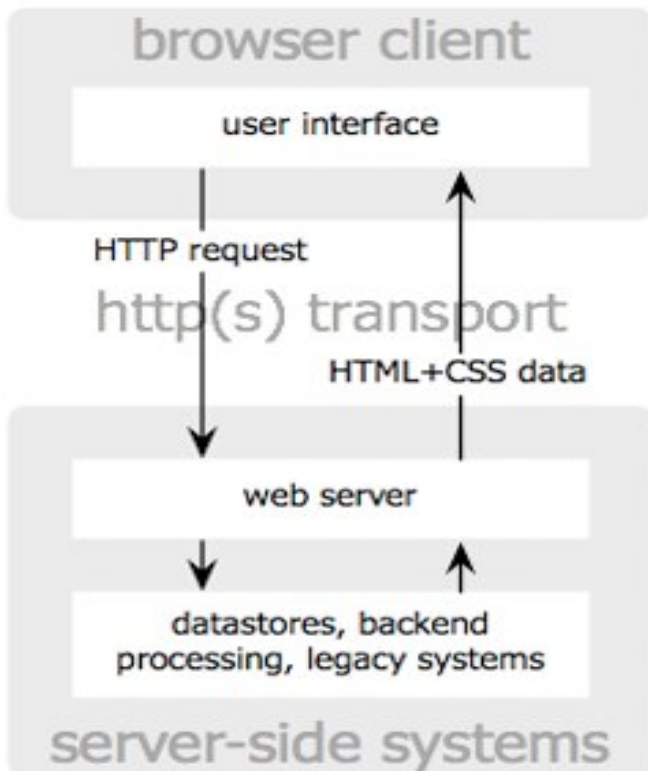
AJAX

O que é?

- Asynchronous JavaScript And XML.
- Não é uma linguagem de programação!
- Técnica baseada em *Web Standards*.
 - **JavaScript**
 - **XML**
 - **HTML**
 - **CSS**

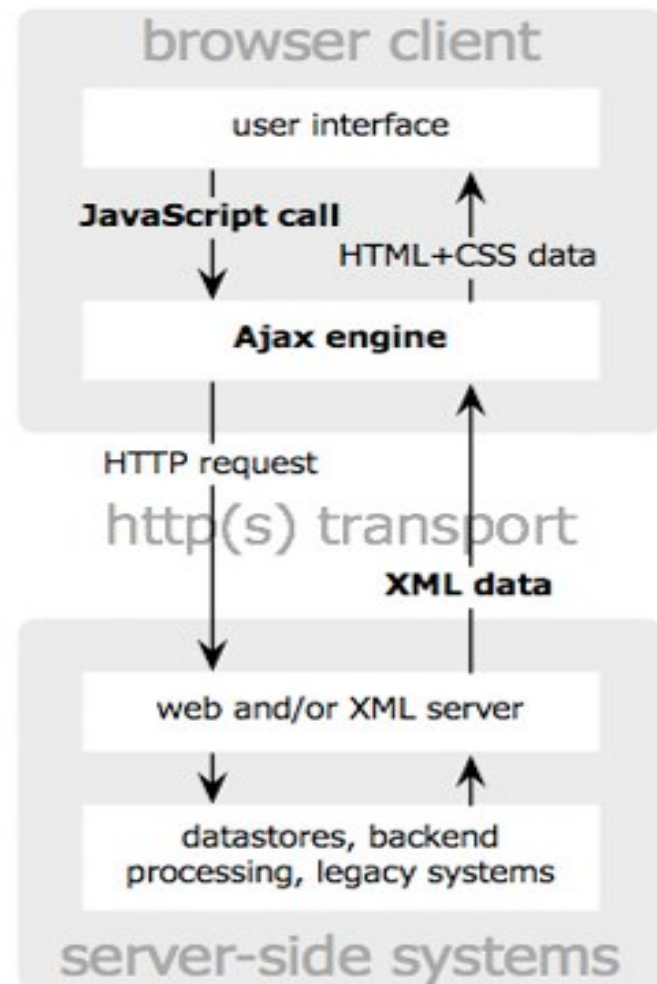
Por quê?

- Gap entre “Web Experience” e “Desktop Experience”.
- Para desenvolver aplicações Web melhores, mais rápidas e mais interativas.



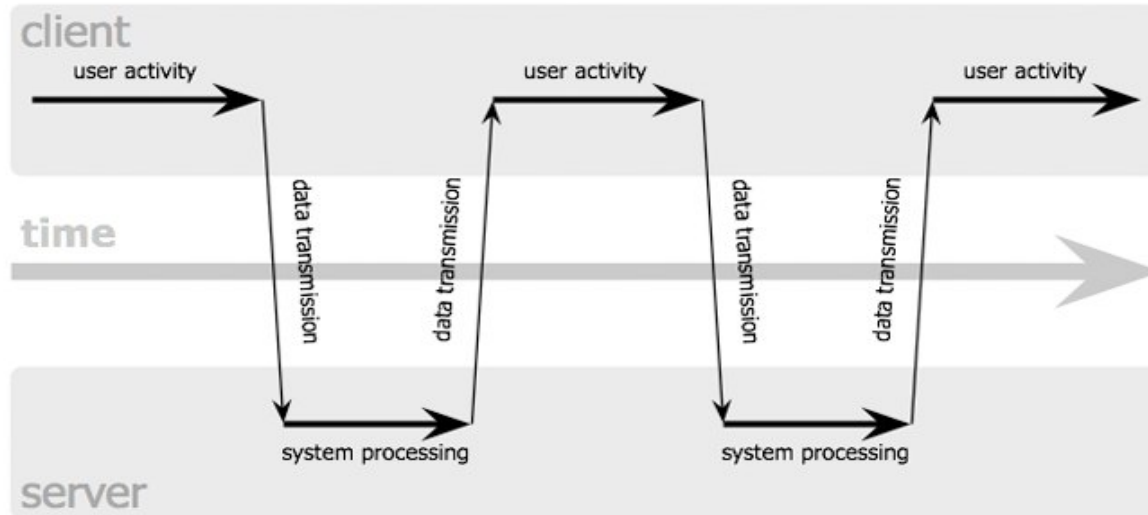
**classic
web application model**

Jesse James Garrett / adaptivepath.com

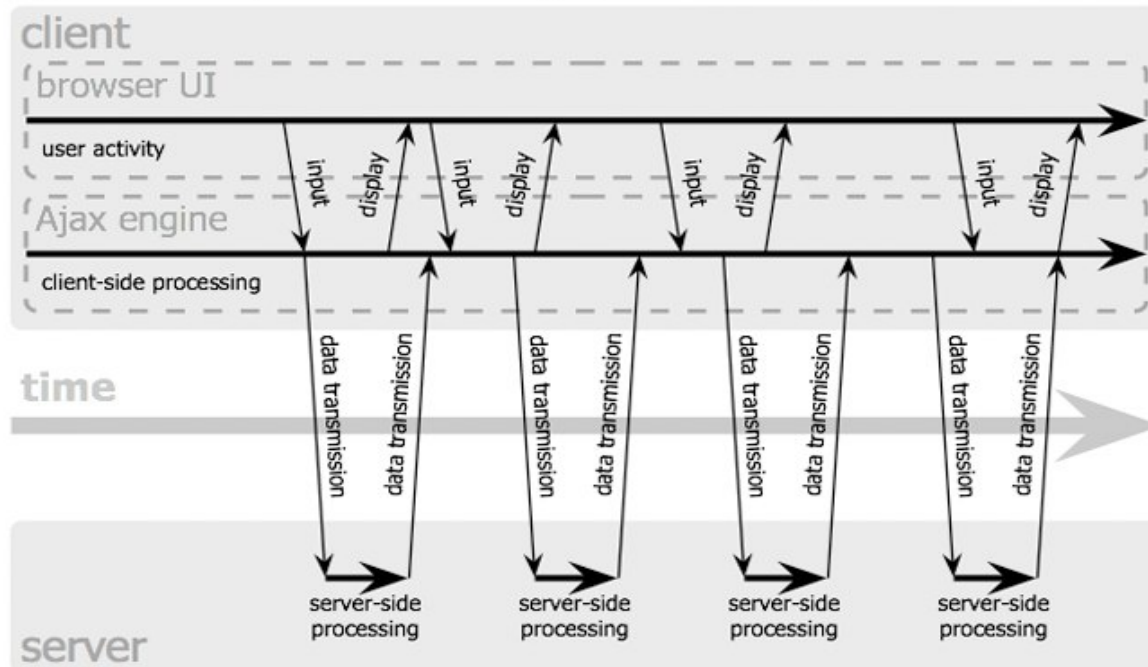


**Ajax
web application model**

classic web application model (synchronous)

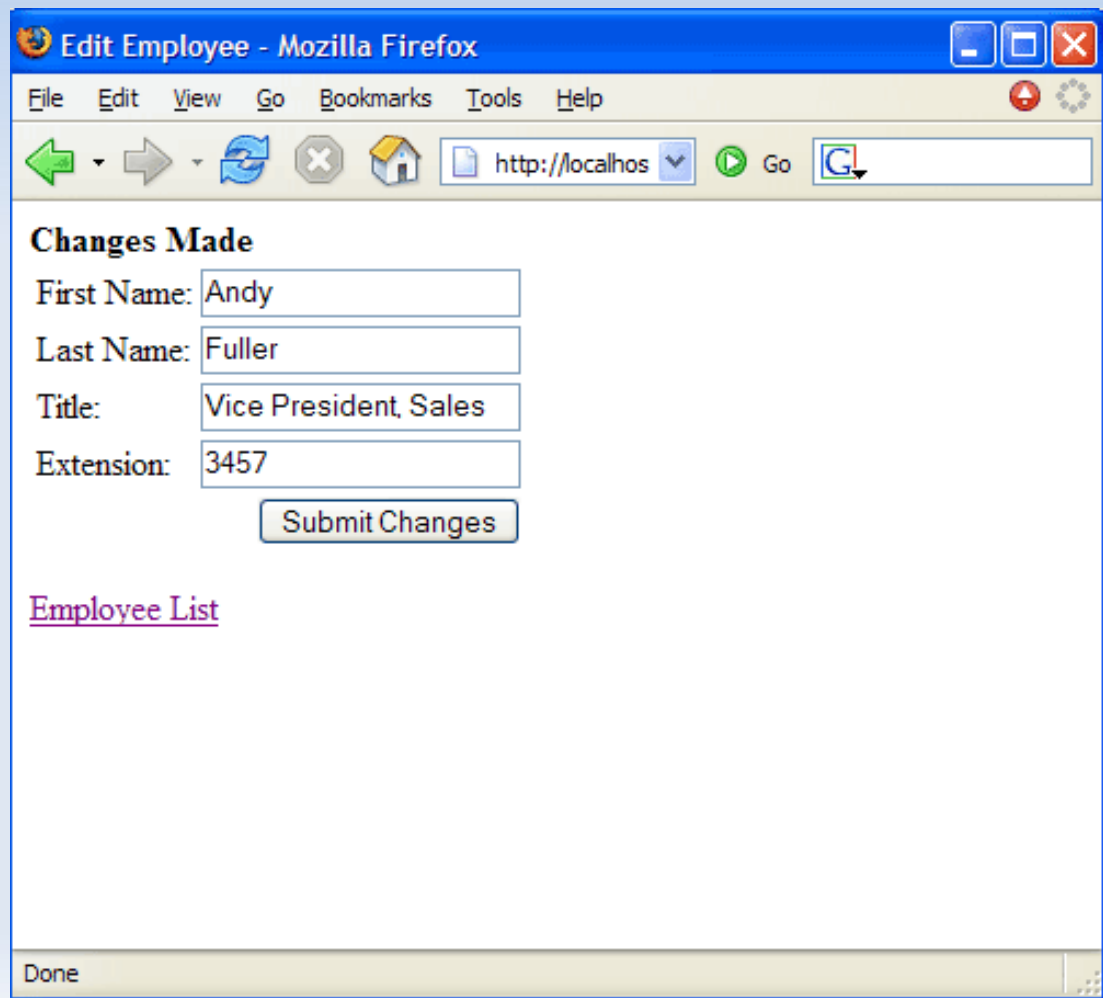


Ajax web application model (asynchronous)

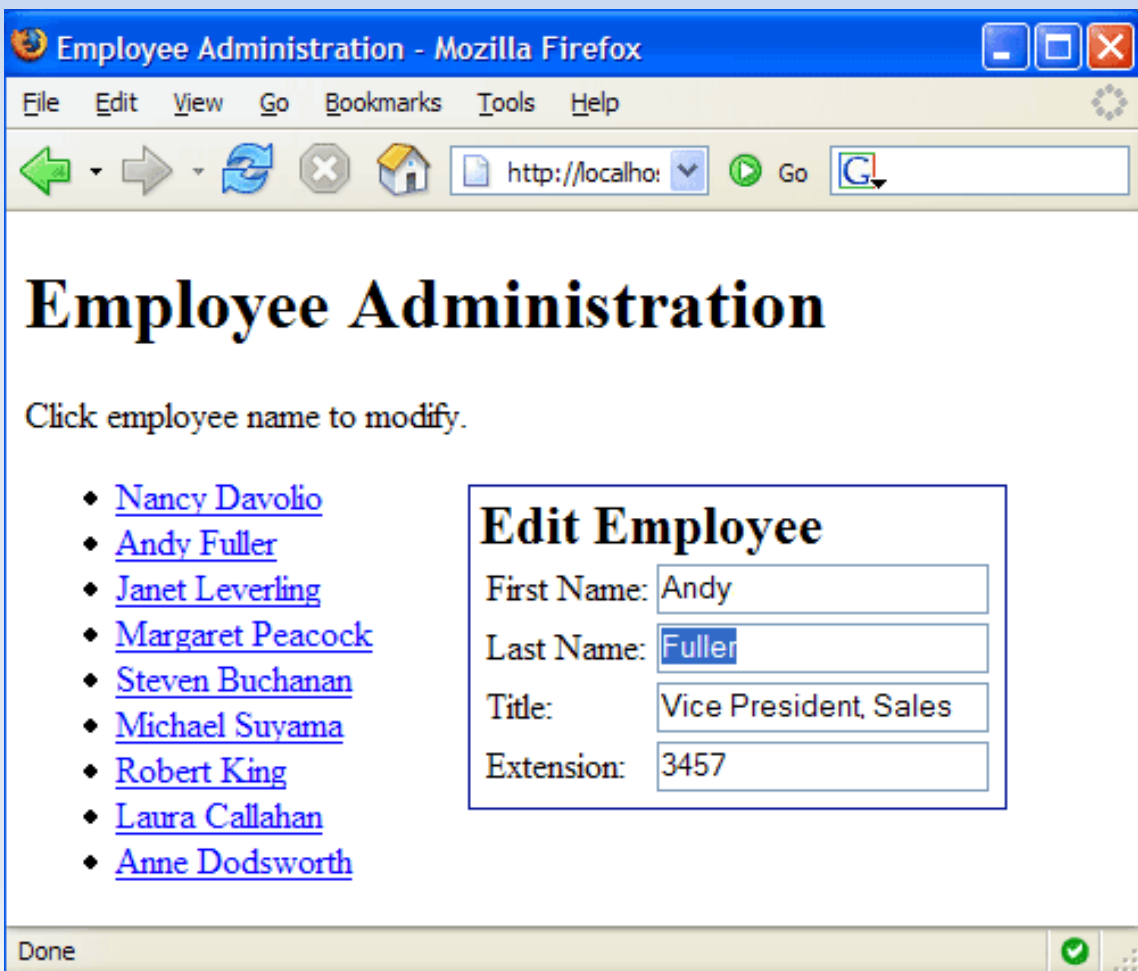




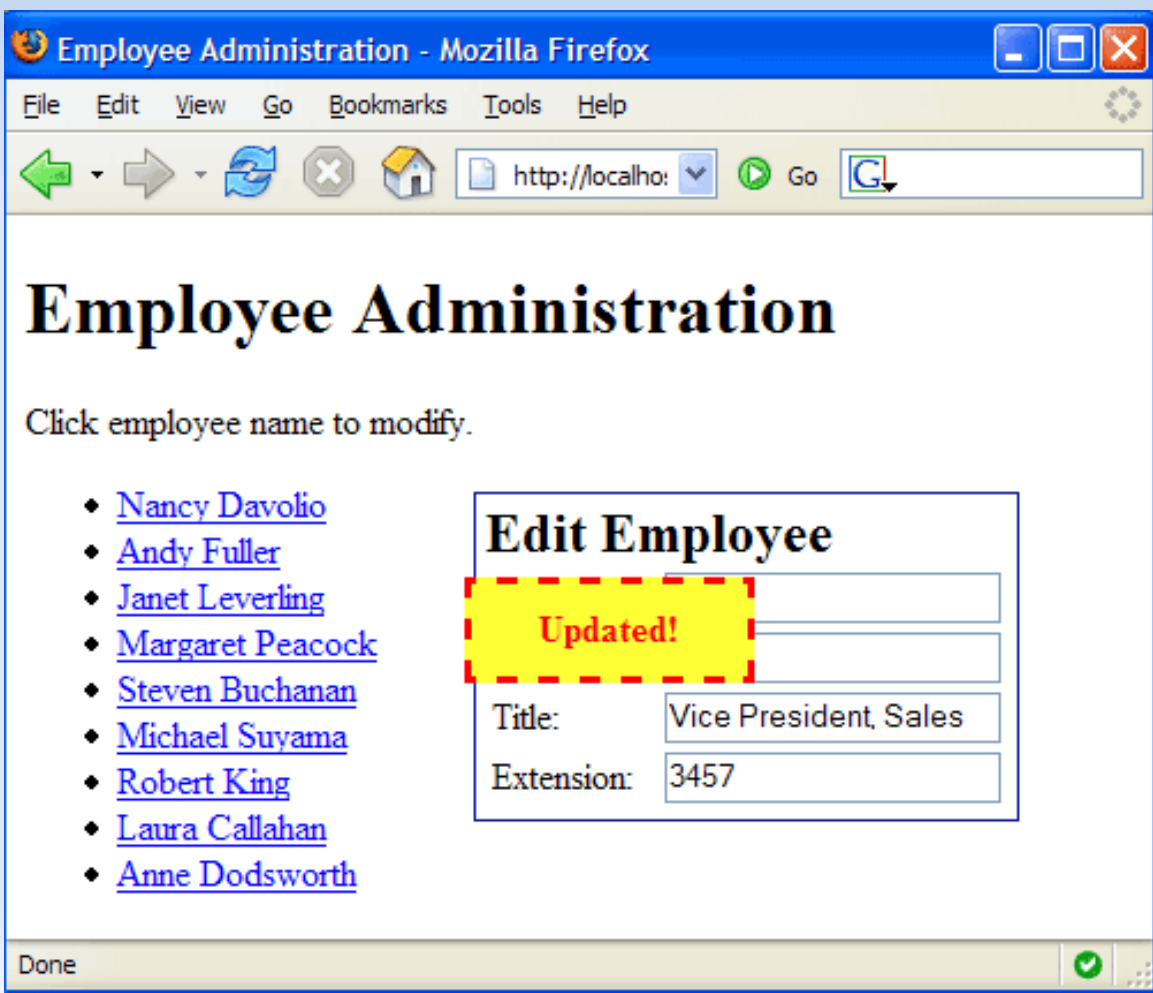
Aplicação Web Tradicional



Aplicação Web Tradicional



Aplicação Web **AJAX**



Aplicação Web AJAX

AJAX

- outros exemplos:
 - Google suggests
 - Google Maps
 - Gmail

Como?

- Transferência de dados assíncrona via HTTP requests.
- Browsers fornecem implementação de AJAX engines.
 - No início da sessão, Browser carrega um JavaScript, em vez de uma página.
 - Script faz comunicação.
 - Objeto AJAX (**XMLHttpRequest**).

```
<html><body>
  <script type="text/javascript">
    function ajaxFunction() {
      var xmlHttp;
      try { // Firefox, Opera 8.0+, Safari
        xmlHttp=new XMLHttpRequest();
      } catch (e) {
        // Internet Explorer
        try {
          xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
          try {
            xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
          } catch (e) {
            alert("Your browser does not support AJAX!");
            return false;
          }
        }
      }
    }
  </script>
  <form name="myForm"> Name: <input type="text" name="username" />
                                Time: <input type="text" name="time" />
  </form>
</body> </html>
```

Como?

.....

```
xmlHttp.onreadystatechange=function() {  
    if(xmlHttp.readyState==4) {  
        document.myForm.time.value=xmlHttp.responseText;  
    }  
}  
xmlHttp.open("GET","time.asp",true);  
xmlHttp.send(null);  
}  
</script>  
<form name="myForm">  
    Name: <input type="text" onkeyup="ajaxFunction();" name="username" /> Time: <input type="text" name="time" />  
</form>
```

Referências

- <http://www.w3schools.com/Ajax/default.asp>
- Crane, Dave et al. *Ajax in Action*. Greenwich, Conn : Manning, 2006.