



Introdução ao Ajax

Helder da Rocha

(helder.darocha@gmail.com)

Programa (1)

- O que é Ajax?
 - Por que usar?
 - Quando usar?
 - Quem usa?
 - Alternativas
- Fundamentos tecnológicos
 - Arquitetura Web tradicional e Web 2.0
 - XML
 - JavaScript
 - CSS
 - DOM

Programa (2)

- Requisição e resposta (sem frameworks)
 - Obtendo o XMLHttpRequest
 - Criando um request
 - Obtendo a resposta
 - Processando a resposta
- Processamento passo-a-passo
- Processando resposta em XML
- Frameworks para uso com Java
 - Passo-a-passo com DWR
 - Overview de JSF

O que é Ajax?

- O que é Ajax? O que não é Ajax?
- Por que usar?
- Quando usar?
- Quem usa?
- Alternativas

Ajax

- Ajax é uma solução **lado-cliente** baseada em HTML, JavaScript e DOM que permite que a comunicação entre o browser e o servidor Web ocorra de forma assíncrona
- Ajax não é uma linguagem nova, nem mesmo uma tecnologia nova
- Ajax não é uma solução lado-servidor

Por que usar?

- A comunicação HTTP é ineficiente
 - Para cada requisição há uma resposta
 - Cada resposta devolve uma página inteira
 - É preciso esperar toda a página carregar antes de usar uma aplicação Web
- Ajax permite comunicação assíncrona
 - Pequenos trechos de dados podem ser transferidos assíncronamente
 - Permite que aplicação funcione enquanto dados são transferidos

Quando usar?

- Use em aplicações Web interativas que sofrem com o modelo requisição-resposta
 - Aplicações com menus, muitas opções, que requerem interatividade em tempo real
 - Aplicações que modelam aplicações gráficas de desktop
- Não use em aplicações que realmente precisam carregar uma página inteira
 - Ex: alguns tipos de sistemas de informação

Quem usa? Exemplos

- Aplicações mais populares
 - Google Maps
 - Google GMail
 - Yahoo Flickr
 - ...

Alternativas

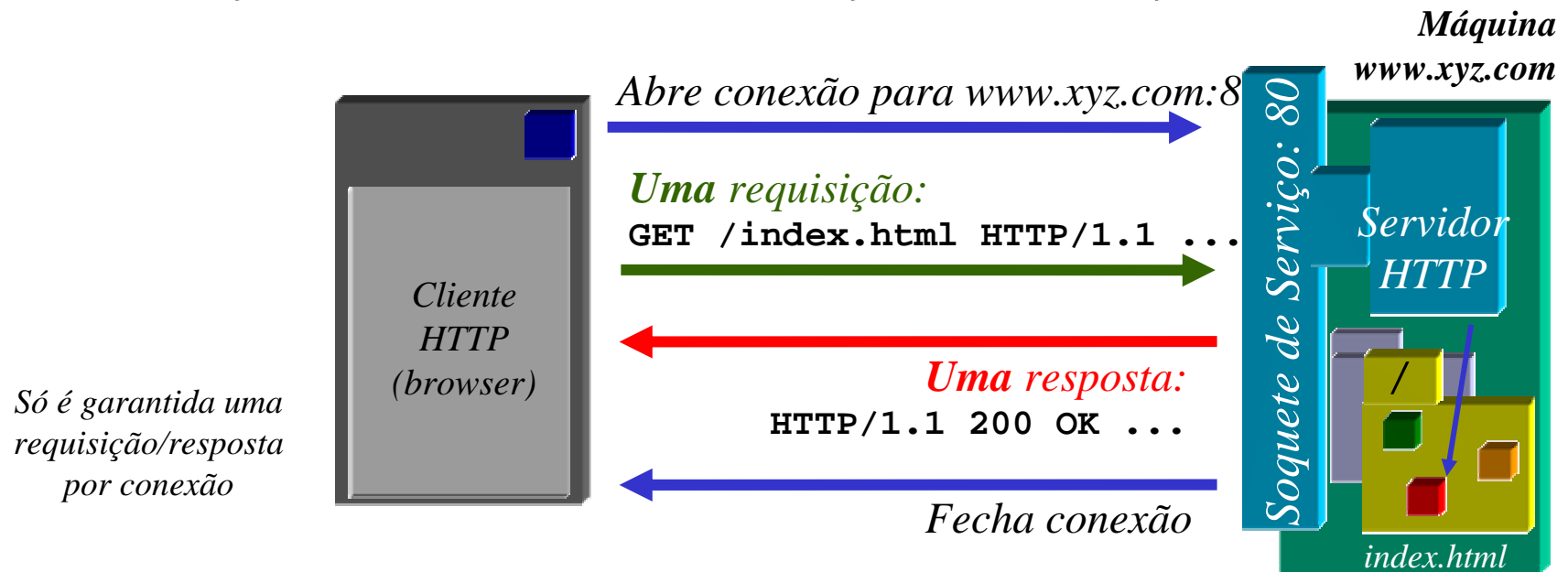
- Flash
- SVG
- Java Web Start

Fundamentos tecnológicos

- Arquitetura Web e Web 2.0
- XML
- JavaScript
- CSS
- DOM

Arquitetura Web

- Baseada em cliente, protocolo HTTP e servidor
- Principais características
 - Protocolo de transferência de arquivos (HTTP: RFC 2068) não mantém estado da sessão do cliente
 - Servidor representa sistema de arquivos virtual e responde a comandos que contém URLs
 - Cabeçalhos contém meta-informação de requisição e resposta



Exemplo de requisição/resposta HTTP

- 1. Página HTML

```

```

- 2. Requisição: browser solicita imagem

```
GET /tomcat.gif HTTP/1.1  
User-Agent: Mozilla 6.0 [en] (Windows 95; I)  
Cookies: querty=uiop; SessionID=D236S11943245
```

*Linha em
branco
termina
cabeçalhos*

- 3. Resposta: servidor devolve cabeçalho + stream

```
HTTP 1.1 200 OK  
Server: Apache 1.32  
Date: Friday, August 13, 2003 03:12:56 GMT-03  
Content-type: image/gif  
Content-length: 23779
```

```
!#GIF89~¾ 7  
.55.a 6αÜ4 ...
```

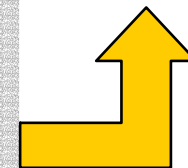
*Interpreta
HTML*



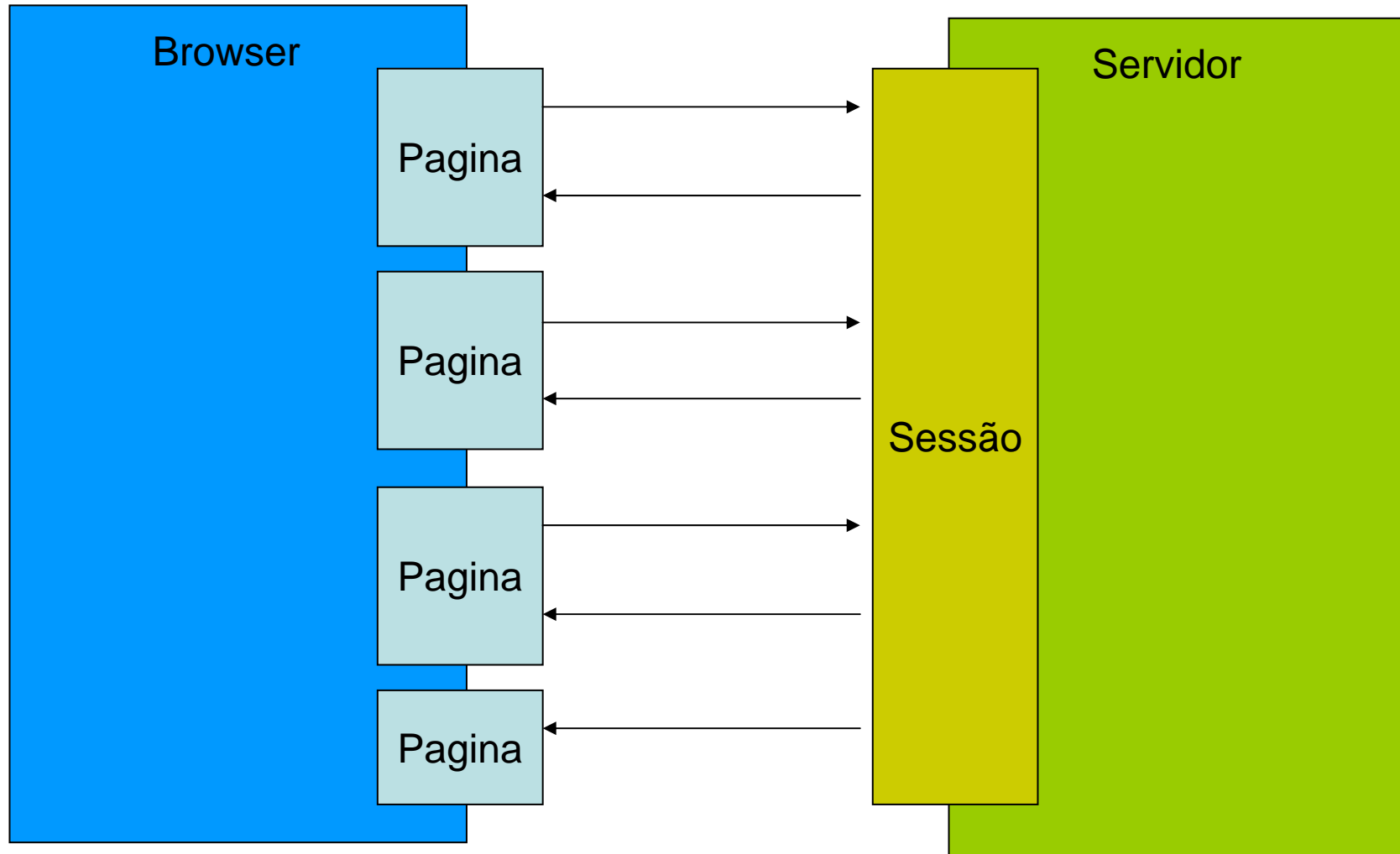
*Gera
requisição
GET*



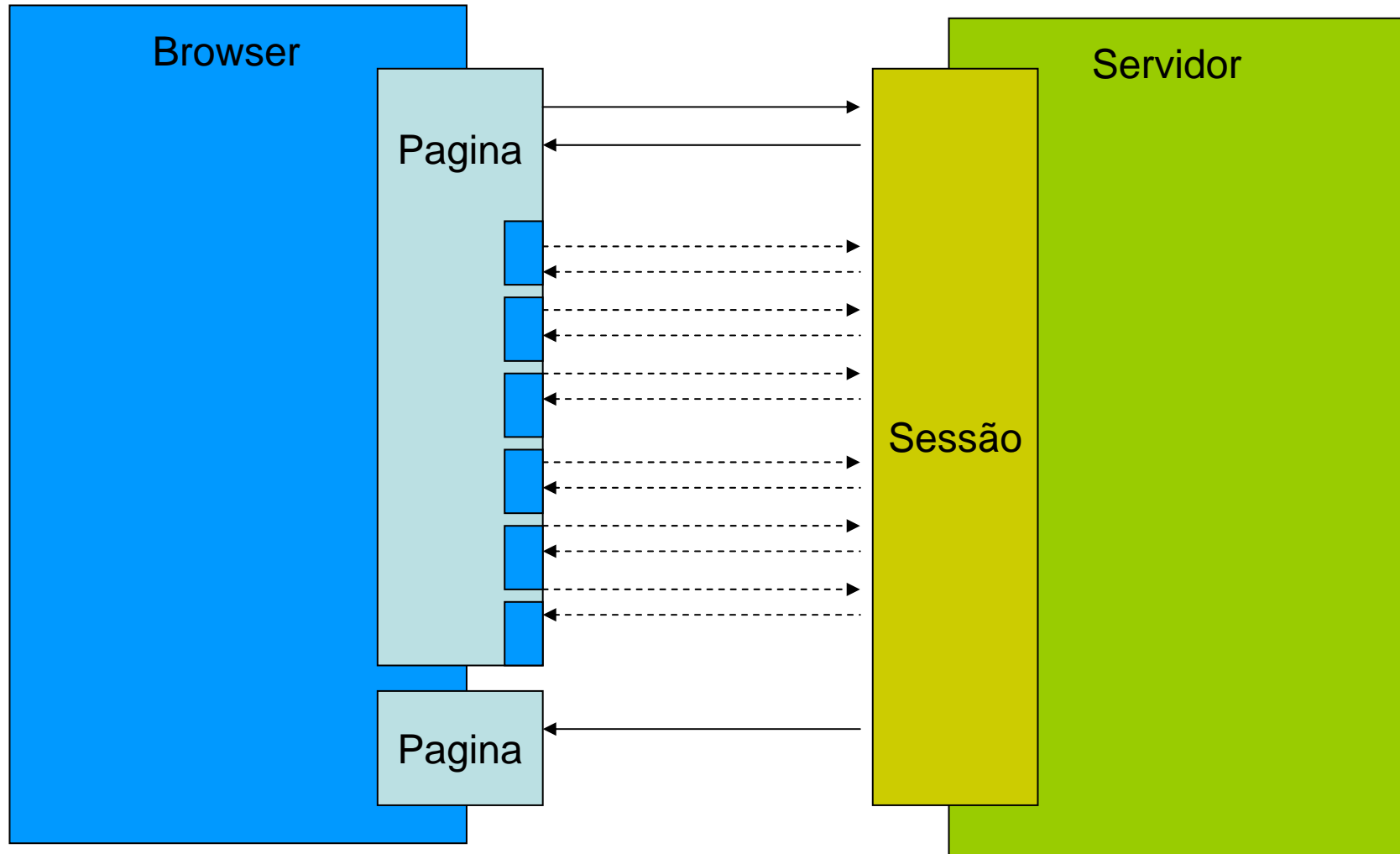
tomcat.gif



Ciclo de vida de aplicação Web



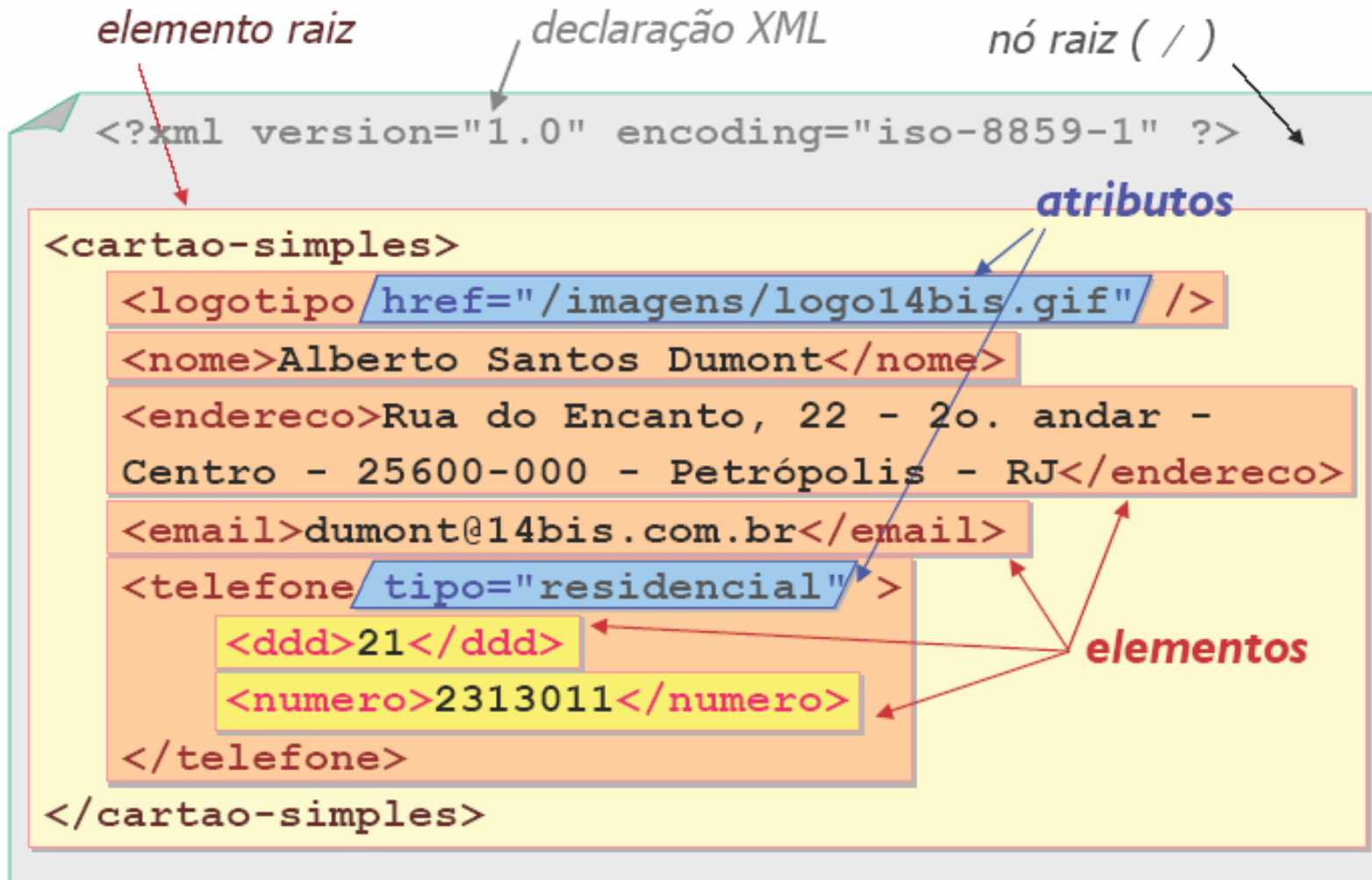
Ciclo de vida de aplicação Ajax (Web 2.0)



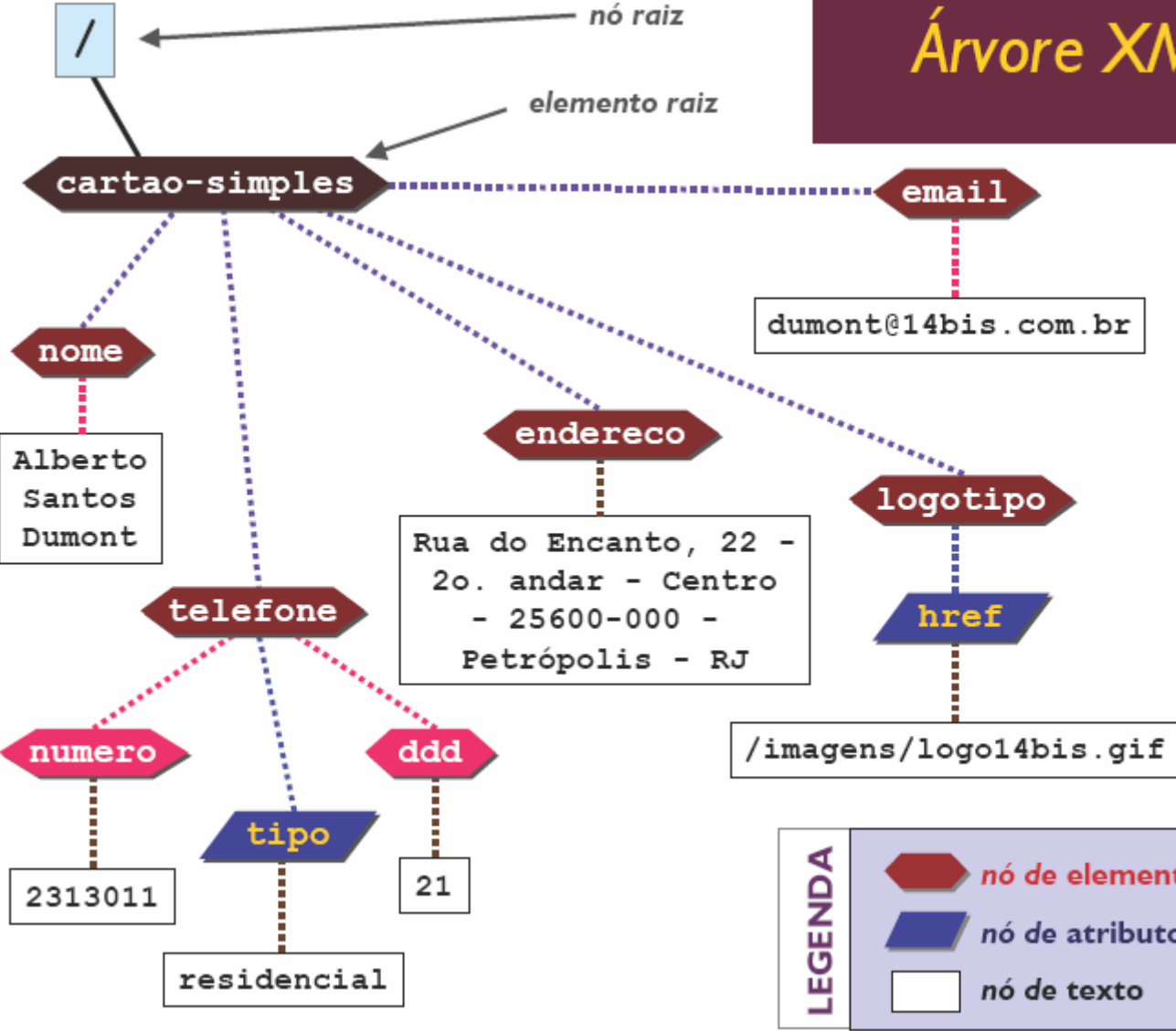
XML

- Fornece um meio simples de transmitir informações estruturadas entre o cliente e o servidor
 - Pode-se transferir toda a informação de objetos independente de linguagem
 - Validação XML Schema
 - Manipulação via DOM, SAX, mapeamento objeto-XML, JAXB, Web Services
- Forma mais comum para devolver dados ao cliente (garante mais controle)

Documento XML



Árvore XML



JavaScript

- Linguagem de propósito geral projetada para ser embutida em aplicações
- Permite interação com o modelo de objetos do browser e com o DOM
- Aplicações Ajax são escritas em JavaScript
- Incluído na página de três formas
 - `<script src="url_da_api.js" />`
 - `<script> ... código estático </script>`
 - Dentro de atributos especiais (onload, onXXX) ou usando prefixo javascript: em atributos comuns

Exemplo de JavaScript

```
function soma(a, b) {  
    return a + b;  
}
```

biblio.js

pagina.html

```
<script LANGUAGE=JavaScript SRC="biblio.js"></script>  
(...)  
<script>  
    resultado = soma(5, 6);  
    document.write("<P>A soma de 5 e 6 é " + resultado);  
</script>
```

JavaScript

- Tem palavras chave parecidas com as de Java (mas têm outras)
- Não é strongly-typed como Java
 - Declaração de variáveis globais é opcional
 - Declaração de variáveis locais com “var”
- Integra-se com Java
- É baseada em objetos (pode-se criar objetos a partir de protótipos mas não de herança)
 - Não suporta sobrecarga ou sobreposição
 - Funções (function) são objetos

Tipos e objetos nativos ECMAScript

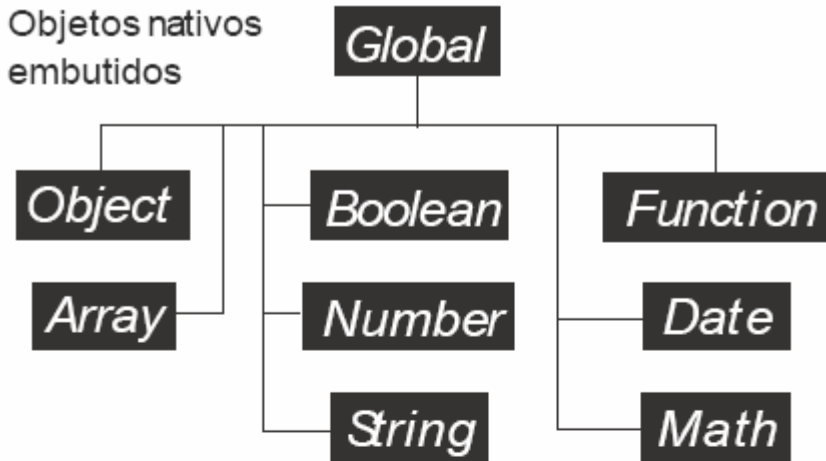
function

Tipo de objeto que representa funções, métodos e construtores

object

Tipo de dados nativo que representa coleções de propriedades contendo valores de tipos primitivos, function ou object

Objetos nativos embutidos



Tipos de dados primitivos (que representam valores)

undefined

representa valores `undefined` ainda não definidos

null

representa o valor `null` nulo

boolean

representa valores booleanos `true` `false`

number

representa números de ponto-flutuante IEEE 754 com precisão de 15 casas decimais (64 bits)

Min: $\pm 4.94065 \text{ e-}324$
 Max: $\pm 1.79769 \text{ e+}308$
 NaN
 Infinity
 -Infinity

string

representa cadeias ordenadas (e indexáveis) de caracteres Unicode.

"\u0000 - \uFFFF"
 '\u0000 - \uFFFF'
 ''
 ""
 "abcde012+\$_@..."

JavaScript no browser

- Os objetos, variáveis, etc. estão disponíveis a partir do objeto raiz, que no browser é document
- Todos os elementos da página estão em uma árvore a partir de document
- Pode-se criar novos elementos e anexá-los a document, fazendo-os aparecer dinamicamente na página (usando o DOM)
- Exemplo
 - `document.forms[0]` – primeiro formulário da página
 - `document.getElementById("botao_2")` – acessa um elemento HTML que tenha ID botao_2

CSS

- Permite definir estilos reutilizáveis para elementos de página
- Estilos podem ser atribuídos estaticamente a uma página e alterados dinamicamente via JavaScript e DOM
- Incluídos numa página de 3 formas:
 - Via `<link rel=stylesheet href="folha.css" />`
 - `<style> ... CSS ... </style>`
 - Em atributos `...`
- Veja exemplos

CSS Sintaxe básica

- Uma folha de estilos é uma coleção de regras
- Cada regra tem o formato:
 - seletores { propriedade: valor, ... }
- Seletores são elementos, ids, classes:
 - h1, h2 {...}
 - h1.principal, .outros {...}
 - #id35 {...}
- Classes e IDs são definidos em elementos
 - <h1 class="principal titulo" id="id99">
- Propriedades definem estilo:
 - h1 {color: red}

CSS para layout

- Há várias propriedades que definem layout e visibilidade
 - position, absolute, relative, static
 - visibility
 - display
- Várias podem ser alteradas via JavaScript para realizar mudanças dinâmicas de posicionamento e visibilidade.

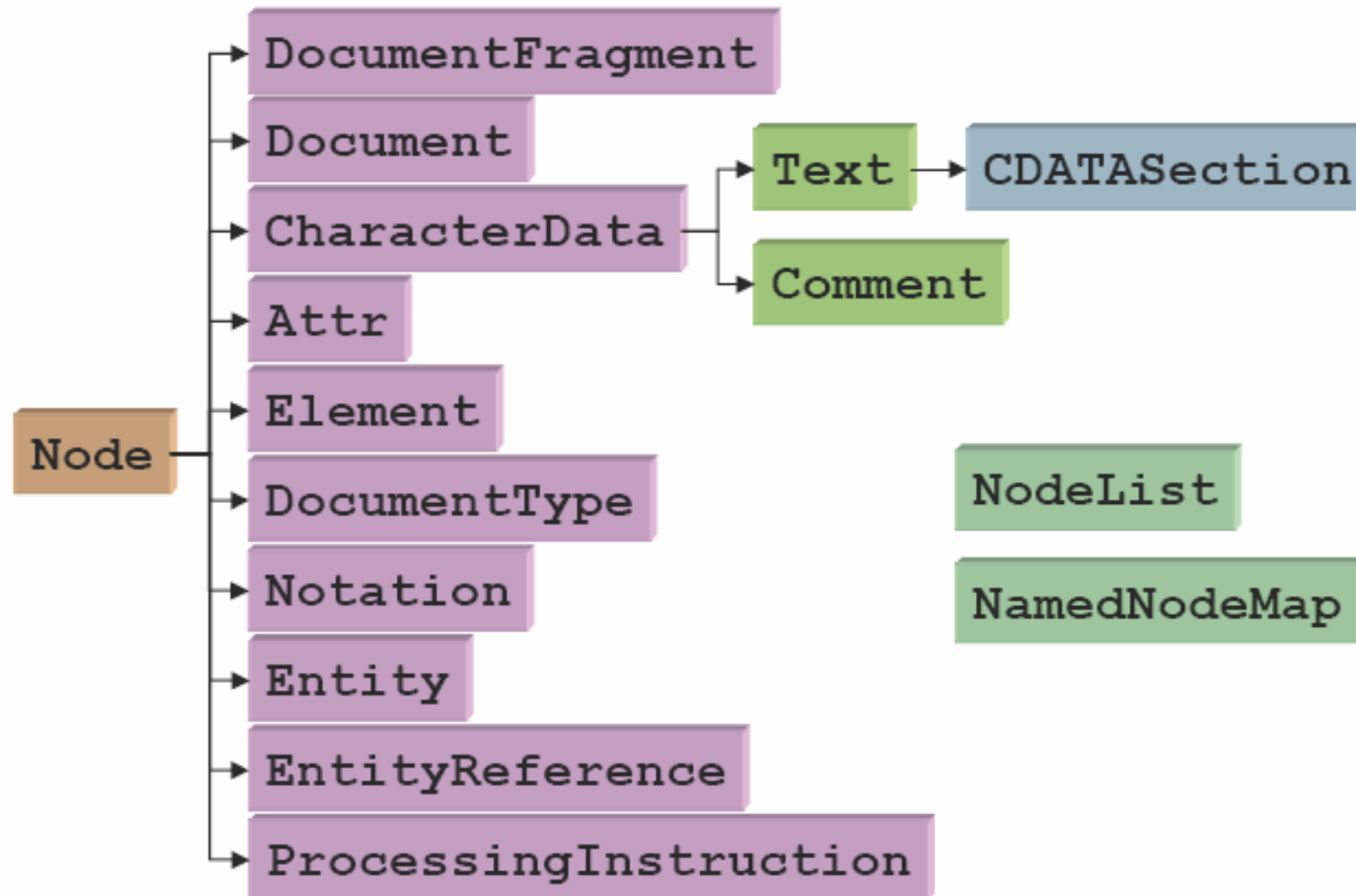
Alterando CSS via JavaScript

- Localize o elemento
 - `var e = document.getElementById("id02");`
- Altere seu estilo
 - `e.className = 'outros';`
 - `e.style.border="solid red 1px";`
 - `e.style.display="block";`

DOM

- É impossível usar JavaScript no browser sem usar document, que é a raiz do DOM
- O DOM é um modelo de objetos padrão, independente de linguagem, usado para representar elementos, atributos, nós de texto, etc.
- Tem uma API padrão independente de linguagem

Hierarquia do DOM



Como criar nós, atributos

```
var doc = document;
```

`<carta>`

Element

```
carta = doc.createElement("carta")
```

`<mensagem>`

Element

```
mens = doc.createElement("mensagem")
```

Bom dia!

String

```
texto = doc.createTextNode("Bom dia!")
```

`<mensagem id="1">`

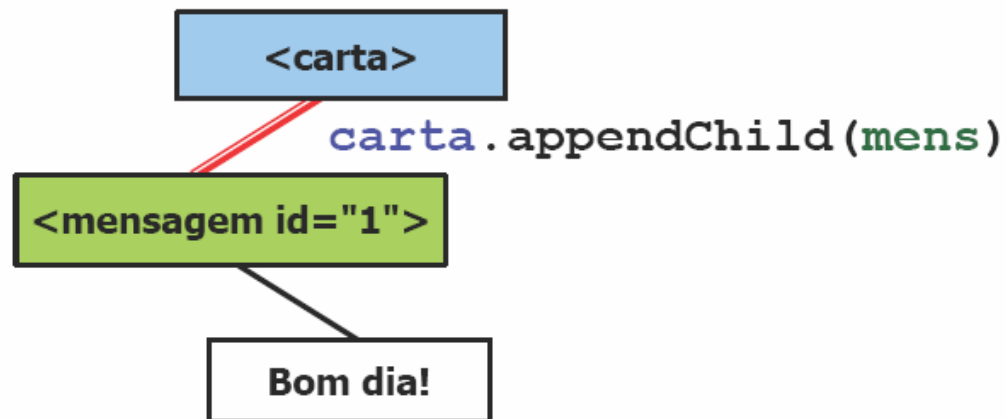
```
mens.setAttribute("id", "1")
```

Como montar uma árvore

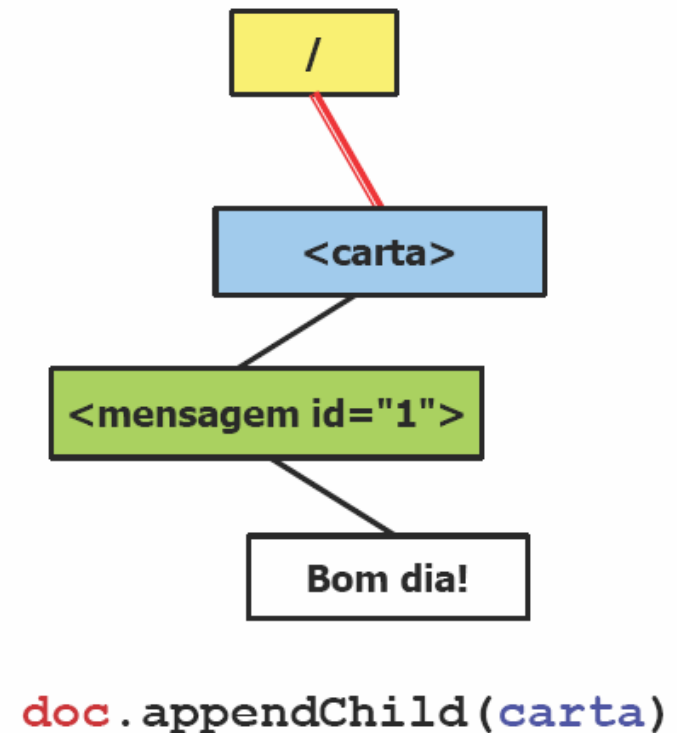
1. Sub-árvore `<mensagem>`



2. Sub-árvore `<carta>`



3. Árvore completa



innerHTML

- Método menos prolixo para gerar XML/HTML (evita o uso de createElement e appendChild)
- pai.innerHTML = "<filho>...</filho>";
- Mais prático para alterações dinâmicas (por exemplo, quando um fragmento é recebido assincronamente)

Exemplos interativos

- Alerta em JavaScript
- Alteração de formulário em JavaScript
- Ocultação com JavaScript e CSS
- Alteração de estilo com CSS
- Localização de elemento com DOM
- Criação de árvore DOM dinâmica

Conclusões

- Aplicações Ajax são escritas em JavaScript
- CSS e DOM permitem grande parte do comportamento dinâmico (DHTML) usado em aplicações Ajax através do uso de JavaScript
- É importante conhecer essas tecnologias para utilizar melhor os recursos do Ajax

Ajax passo-a-passo

- Obtendo o XMLHttpRequest
 - independente do browser
- Criando um request
 - Métodos do objeto XMLHttpRequest
- Obtendo a resposta
 - Estados (ready states)
- Processando a resposta
 - Obtendo e usando os dados

Obtendo o XMLHttpRequest

- O XMLHttpRequest é o objeto do DOM que irá realizar a comunicação assíncrona
 - Ou seja, é o coração do Ajax
 - Nos browsers modernos (IE7, FireFox, etc.) é obtido da seguinte forma:
 - `http_request = new XMLHttpRequest();`
 - Nos browsers Microsoft antigos, há duas formas
 - `http_request = new ActiveXObject("Msxml2.XMLHTTP");`
 - `http_request = new ActiveXObject("Microsoft.XMLHTTP");`
- dependendo da versão

XMLHttpRequest cross-browser

- A solução é lidar com os diferentes browsers

```
var http_request = false;
if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
    try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            http_request = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
    }
}

if (!http_request) {
    alert('Giving up :( Cannot create an XMLHTTP instance');
    return false;
}
// agora pode usar o http_request
```

Criando um Request

1. Pegue o que for necessário do formulário Web
`dado = document.getElementById("campo1").value;`
2. Construa a URL de conexão
`url = "/scripts/dados.php?dado=escape(dado)";`
3. Abra conexão ao servidor
`http_request.open("GET", url, true);`
4. Defina uma função para executar quando terminar
`http_request.onreadystatechange = updatePage;`
5. Envie a requisição
`http_request.send(null);`

Propriedades importantes

- `onreadystatechange`
 - deve receber o nome de uma função que será executada quando a requisição terminar
- `readyState`
 - deve ser lida para se saber em que estado está a resposta; o estado útil é 4
- `status`
 - contém o status HTTP (200, 404, etc.)
- `responseText` e `responseXML`
 - contém dados da resposta

Obtendo uma resposta

1. Não faça nada até que o valor de readyState seja 4
 2. Leia o que está em responseText ou responseXML
- Exemplo (se onreadystatechange apontar para a função abaixo)

```
function updatePage() {  
    if (http_request.readyState == 4) {  
        var response = http_request.responseText;  
        document.getElementById("resposta").value  
            = response;  
    }  
}
```

Pode-se chamar o método de conexão usando o evento onChange dos campos do form HTML (veja exemplo)

Processando a resposta

- A resposta pode retornar como texto comum ou como XML
- Se for texto comum (`responseText`) pode ser usada como está ou processada (usando expressões regulares, etc.)
- Se for texto XML (`responseXml`), pode ser manipulada usando DOM para extrair campos significativos

Um exemplo passo-a-passo

- Fonte: Make asynchronous requests with JavaScript and Ajax (Brett McLaughlin)
 - <http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro2/index.html>

```
<script language="javascript" type="text/javascript">
  var request = false;
  try {
    request = new XMLHttpRequest();
  } catch (trymicrosoft) {
    try {
      request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (othermicrosoft) {
      try {
        request = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (failed) {
        request = false;
      }
    }
  }

  if (!request)
    alert("Error initializing XMLHttpRequest!");

  function getCustomerInfo() {

  }
</script>
```

```
<body>
  <p></p>
  <form action="POST">
    <p>Enter your phone number:
      <input type="text" size="14" name="phone" id="phone"
        onChange="getCustomerInfo();" />
    </p>
    <p>Your order will be delivered to:</p>
    <div id="address"></div>
    <p>Type your order in here:</p>
    <p><textarea name="order" rows="6" cols="50"
id="order"></textarea></p>
    <p><input type="submit" value="Order Pizza" id="submit" /></p>
  </form>
</body>
```

```
<script>
function getCustomerInfo() {
    var phone = document.getElementById("phone").value;
    var url = "/cgi-local/lookupCustomer.php?phone=" + escape(phone);
    request.open("GET", url, true);
    request.onreadystatechange = updatePage;
    request.send(null);
}

function updatePage() {
    if (request.readyState == 4) {
        if (request.status == 200) {
            var response = request.responseText.split("|");
            document.getElementById("order").value = response[0];
            document.getElementById("address").innerHTML =
                response[1].replace(/\n/g, "
");
        } else
            alert("status is " + request.status);
        }
    }
}
</script>
```

Ready States

- Pode haver 5 estados durante a requisição e resposta assíncrona
- Eles são lidos através da propriedade `readyState`
 - 0 – não inicializado
 - 1 – não enviado
 - 2 – sendo processado (cabeçalhos)
 - 3 – sendo processado (parte dos dados)
 - 4 -concluído
- São dependentes de browser (o único realmente confiável é 4)

Resposta em XML

- Facilita apresentação dos dados
- Exemplo:
- Processamento

```
<ratings>
  <show>
    <title>Alias</title>
    <rating>6.5</rating>
  </show>
  <show>
    <title>Lost</title>
    <rating>14.2</rating>
  </show>
  <show>
    <title>Six Degrees</title>
    <rating>9.1</rating>
  </show>
</ratings>
```

```
var xmlDoc = request.responseXML;
var showElements = xmlDoc.getElementsByTagName("show");
for (var x=0; x<showElements.length; x++) {
    var title = showElements[x].childNodes[0].value;
    var rating = showElements[x].childNodes[1].value;
}
```

Transferência de objetos

- O estado de objetos pode ser passado do servidor para o JavaScript no cliente através da serialização em XML
 - Requer que servidor converta objeto para XML (ex: mapeamento) e que cliente processe XML (usando DOM)
 - Soluções de baixo nível incluem soluções próprias, JAXB, templates e a API JSON (JavaScript Object Notation)
 - Solução mais fácil é usar frameworks

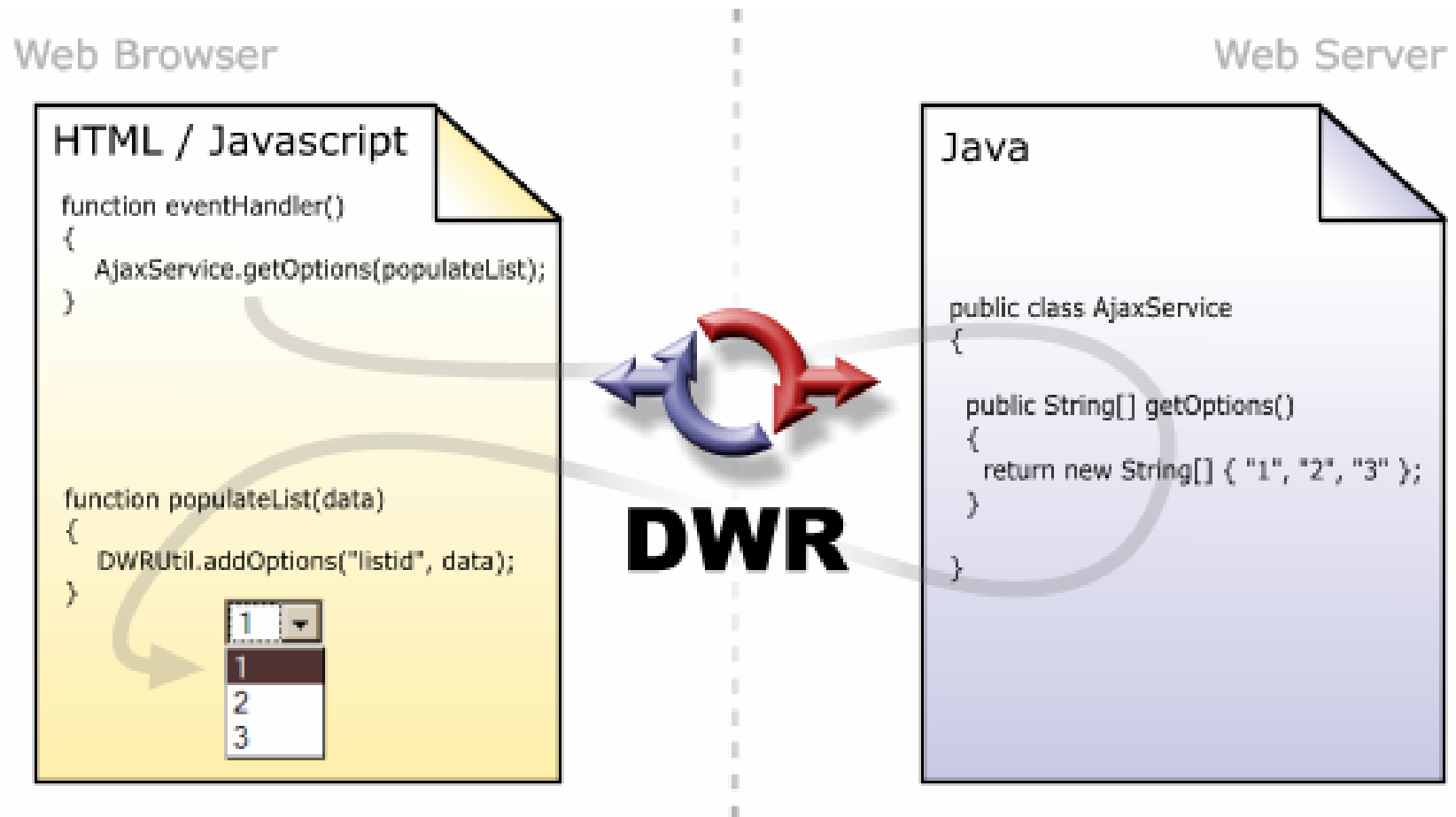
Frameworks

- Para qualquer aplicação é essencial entender os fundamentos da programação com Ajax
- Para trabalhar com aplicações mais complexas, é importante poder trabalhar em um nível de abstração maior
- Solução: usar frameworks
- Principais frameworks para Java
 - DOJO (usado em componentes JSF)
 - DWR (Direct Web Remoting)

DWR

- Solução simples da Apache
 - <http://getahead.ltd.uk/dwr/>
- Permite que código em um browser use funções Java como se estivesse no browser (gera JS a partir de classes Java)
- Consiste de duas partes
 - Um servlet
 - Uma API JavaScript

Arquitetura



Fonte: DWR

Como usar DWR (1)

1. Baixe o pacote em <http://getahead.ltd.uk/dwr/>
2. Instale o arquivo `dwr.jar` no seu `WEB-INF/lib`
3. Configure seu `web.xml`

```
<servlet>
  <servlet-name>dwr-invoker</servlet-name>
  <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>dwr-invoker</servlet-name>
  <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

Como usar DWR (2)

4. Crie um arquivo dwr.xml e guarde no WEB-INF

```
<!DOCTYPE dwr PUBLIC  
  "-//GetAhead Limited//DTD Direct Web Remoting 1.0//EN"  
  "http://www.getahead.ltd.uk/dwr/dwr10.dtd">
```

```
<dwr>  
  <allow>  
    <create creator="new" javascript="JDate">  
      <param name="class" value="java.util.Date"/>  
    </create>  
    <create creator="new" javascript="Hello">  
      <param name="class" value="hello.HelloWorld"/>  
    </create>  
  </allow>
```

```
</dwr>
```

Define que classes o DWR pode criar e disponibilizar para uso pelo JavaScript (classes devem ter construtor default)

Como usar DWR (3)

5. Abra a URL <http://localhost:8080/contexto/dwr/>
 - Lista de classes disponiveis
 - Lista de métodos que podem ser chamados

Como usar DWR (4)

- 6. Crie aplicações que usem os objetos
 - Veja o código fonte da classe em <http://localhost:8080/contexto/dwr/>
 - Ache a linha que executa o método que você quer usar
 - Cole o texto em uma página HTML ou JSP
 - Inclua os arquivos JavaScript necessários

```
<script src='dwr/interface/HelloWorld.js'></script>
```

```
<script src='dwr/engine.js'></script>
```

Exemplo

- Suponha que tenhamos o seguinte método Java

```
public class HelloWorld {  
    public String getMessage(String n) {...}  
}
```

- Pode-se usar o JavaScript das seguintes formas:

```
<script type="text/javascript" src="dwr/interface/HelloWorld.js"></script>
```

...

```
function handleGetData(str) { alert(str); }  
HelloWorld.getMessage("Hello", handleGetData);
```

OU

```
HelloWorld.getMessage("Hello", function(str) { alert(str); });
```

OU

```
HelloWorld.getMessage("Hello", {  
    callback:function(str) { alert(str); }  
});
```

Como passar objetos

- Suponha que haja as seguintes classes

```
public class PedidoDAO {  
    public void addPedido(Pedido p) {...}  
}
```

```
public class Pedido {  
    private String nome;  
    private double preco;  
    private int[] categorias;  
}
```

- Um Pedido pode ser adicionado em JavaScript usando:

```
var pedido = {  
    name: "Computador", preco: 345.99,  
    categorias:[456, 999]  
};  
PedidoDAO.addPedido(pedido);
```

dwr.xml essencial

- Veja mais em <http://getahead.ltd.uk/dwr/server/dwrxml>
- O tag mais importante dentro de <dwr> é <allow>. Dentro de <allow> há
 - Creators <create> e Converters <convert>
- Creators são necessários para cada classe em que se executa métodos
 - use com atributo creator="new" por enquanto ou veja mais em getahead.ltd.uk/dwr/server/dwrxml/creators)
 - javascript="NomeDoObj"
 - scope="request|session|..."
 - <param name="class" value="nome.da.Classe" />
- Converters servem para converter tipos

Exemplo

```
<!DOCTYPE dwr PUBLIC
  "-//GetAhead Limited//DTD Direct Web Remoting 1.0//EN"
  "http://www.getahead.ltd.uk/dwr/dwr10.dtd">
<dwr>
  <allow>
    <create creator="new" javascript="dao">
      <param name="class"
        value="exemplo.ProdutoDAO"/>
      <include method="getProduto"/>
      <include method="addProduto"/>
    </create>
    <convert converter="bean"
      match="exemplo.Produto">
      <param name="include" value="id, nome, preco"/>
    </convert>
  </allow>
</dwr>
```

Utilitários

- Inclua **util.js**
- `$()` lê elementos pelo ID:
 - `$` equivale a `document.getElementById`
 - Ou seja `$(form1)` recupera o elemento `form1`
- Outras funções (veja documentação em <http://getahead.ltd.uk/dwr/browser/util/gettext>)
 - `getValue(id) / setValue(id, value)`
 - `addRows(...)` e `removeAllRows(id)` – facilita a manipulação de tabelas (veja exemplos)

Como começar

- Instale e teste os exemplos
- Analise o código e tente incluir as funcionalidades em sua aplicação. Lembre-se de
 - Verificar o dwr.xml
 - Verificar se objetos estão no WEB-INF/classes
 - Verificar se arquivos JS estão sendo carregados

Exercício

- Monte a aplicação abaixo, que permite listar produtos em uma tabela (nome e preço), acrescentar e remover, usando DWR
- Dicas:
 - Utilize código HTML disponível
 - Siga o passo-a-passo para instalar o DWR
 - Veja exemplo
<http://getahead.ltd.uk/dwr/examples/table>

Exercício avançado

- Adapte a interface da aplicação abaixo (que lista produtos gravados em um banco) para que funcione com Ajax

Outras alternativas: JSF / DOJO

- Vários componentes JSF suportam Ajax. Para usá-los em uma página é preciso importar as bibliotecas padrão

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
```

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

```
<%@ taglib prefix="dl" uri="http://java.sun.com/blueprints/dl" %>
```

- E depois utilizar o tag abaixo para mapear os elementos desejados

```
<dl:dlabel valueBinding="#{Classe.prop}" />
```

- Veja um tutorial completo sobre essa técnica em <http://java.sun.com/javaee/javaserverfaces/ajax/tutorial.jsp>

Conclusões

- Ajax é uma técnica para aplicações Web com programação lado-cliente que permite maior eficiência e resposta
- Ajax depende de programação em JavaScript e utiliza várias tecnologias já existentes e consagradas como CSS, JavaScript, DHTML e DOM
- Integração com Java através de frameworks permite que o modelo de objetos seja compartilhado entre cliente e servidor
- As mais populares soluções Java hoje são o DWR e o DOJO (nativo JSF)

Obrigado



helder.darocho@gmail.com