

AJAX

Programação Client-Side

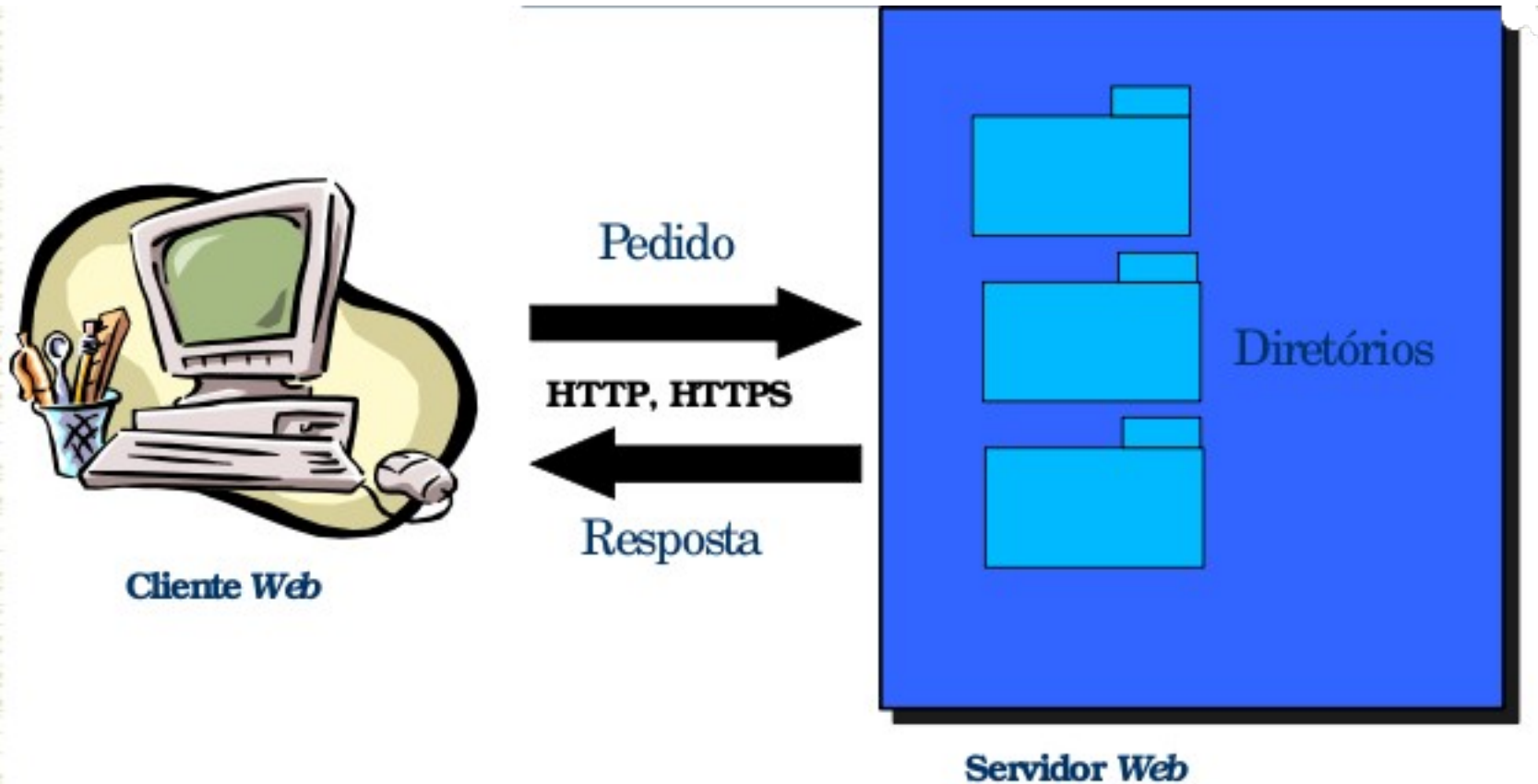
Javascript

Divisão do Curso

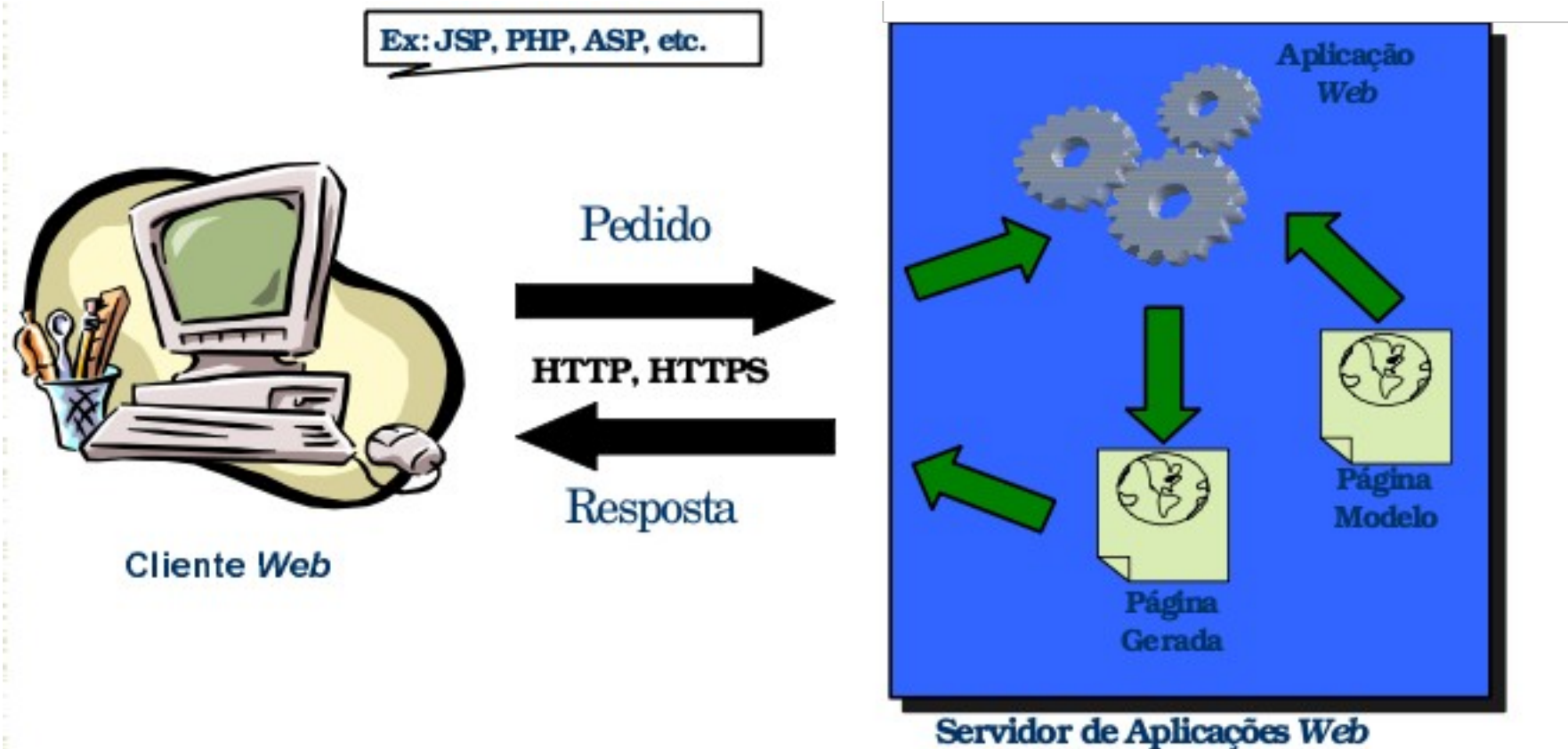
- Introdução
- Sintaxe do Javascript
- Objetos do Javascript
- Manipulação de Eventos + DOM
- OO + CSS
- Requisição Assíncrona + XML

Introdução

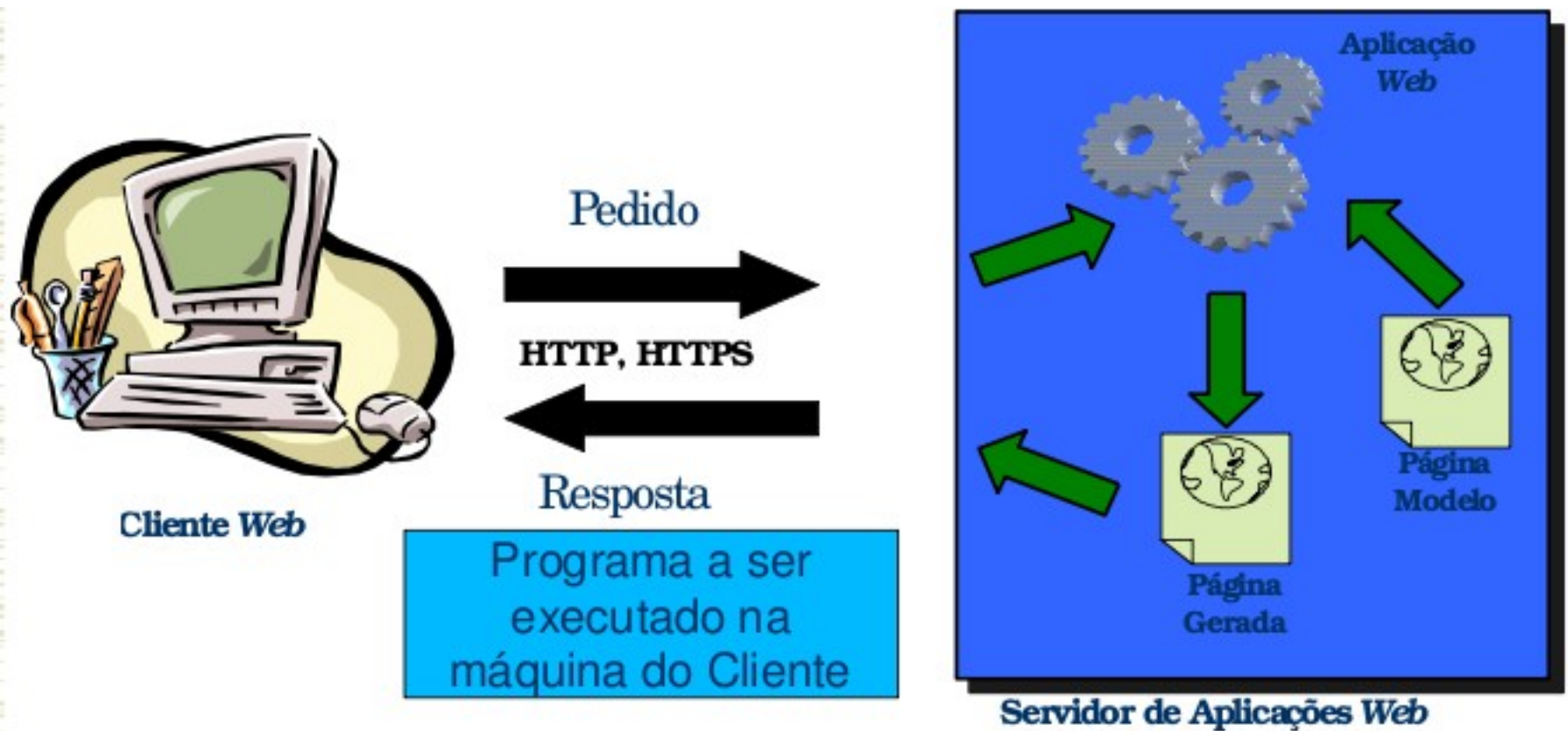
Programação Web



Programação Web



Programação Web



Programação Web

- Alternativas para client-side
 - Javascript
 - Java Applets
 - Flash + Action Script
 - AJAX

JavaScript X Java

- Semelhança do nome
- Semelhanças na sintaxe
- Tudo mais é diferente
- JavaScript é uma linguagem interpretada – Linguagem de Script.
- Programas em Java são compilados para um código de máquina
- Linguagens de programação distintas

JavaScript?

- HTML não é linguagem de programação
- Javascript é uma extensão do HTML

HTML + CSS + Javascript = DHTML

- Faz parte da página e pode interagir com todos os seus componentes
- Suportada pelas principais versões de browser
- Linguagem de programação baseada em objetos
- Criação de páginas interativas e dinâmicas

Para que JavaScript?

- Realizar operações matemáticas
- Gerar documentos com aparência definida na hora da visualização
- Abrir janelas do browser, trocar informações entre janelas, menus pop-up, help, troca de imagens,...
- Manipular propriedades do browser como o histórico, barra de estado, documento,...
- Interagir com o conteúdo do documento
- Tratar a página como uma estrutura de objetos
- Interagir com o usuário através do tratamento de eventos
- Checagem de formulários antes do seu envio

Exemplo



Faça seu primeiro helloWorld!

Sintaxe Javascript

Sintaxe Javascript

- Similar a linguagem C
- Similar a linguagem C++
- Similar a linguagem PHP
- Similar a linguagem Java



Sintaxe Javascript

- Operadores matemáticos
 - + adição de valor e concatenação de strings
 - subtração de valores
 - * multiplicação de valores
 - / divisão de valores
 - % *obtem o resto de uma divisão*

Quão complicado é fazer uma calculadora em Ajax?

Sintaxe Javascript

- Operadores lógicos

== Igual

!= Diferente

> Maior

>= Maior ou Igual

< Menor

<= Menor ou Igual

&& E

|| Ou

= é utilizado para atribuições
`var nome = "Meu nome";`

Sintaxe Javascript

- Controles especiais

`\b` - backspace

`\f` - form feed

`\n` - new line characters

`\r` - carriage return

`\t` - tab characters

`//` - Linha de comentário

`/*...*/` - Delimitadores para inserir um texto com mais de uma linha como comentário.

Os delimitadores naturais para uma string são `"` ou `'`.
Caso seja necessário a utilização destes caracteres como parte da string, utilize `\` precedendo `"` ou `'`.

Sintaxe Javascript

- Estruturas de controle

```
if (condição) {  
    // instruções caso condição == true  
} else if (condição 2) {  
    // instruções caso condição 2 == true  
} else {  
    // instruções caso ambas as condições  
    sejam false  
}
```

Sintaxe Javascript

- Loops

Faça seu primeiro segundo HelloWorld x 10

```
for(inicialização; condição; incremento) {  
    // instruções a serem realizadas enquanto  
    condição for true  
}
```

```
inicialização;  
while(condição) {  
    // instruções a serem realizadas enquanto  
    condição for true  
    incremento;  
}
```

Sintaxe Javascript

- Variáveis

- A variável é criada automaticamente, pela simples associação de valores a mesma.
- Ex. `novaVariavel = "Jose";`
- Ex.2: `var novaVariavel2 = true;`

Global - Declaradas/criadas fora de uma função. As variáveis globais podem ser acessadas em qualquer parte do programa / página.

Local - Declaradas/criadas dentro de uma função. Só podem ser utilizadas dentro da função onde foram criadas e precisam ser definidas com a instrução Var.

Sintaxe Javascript

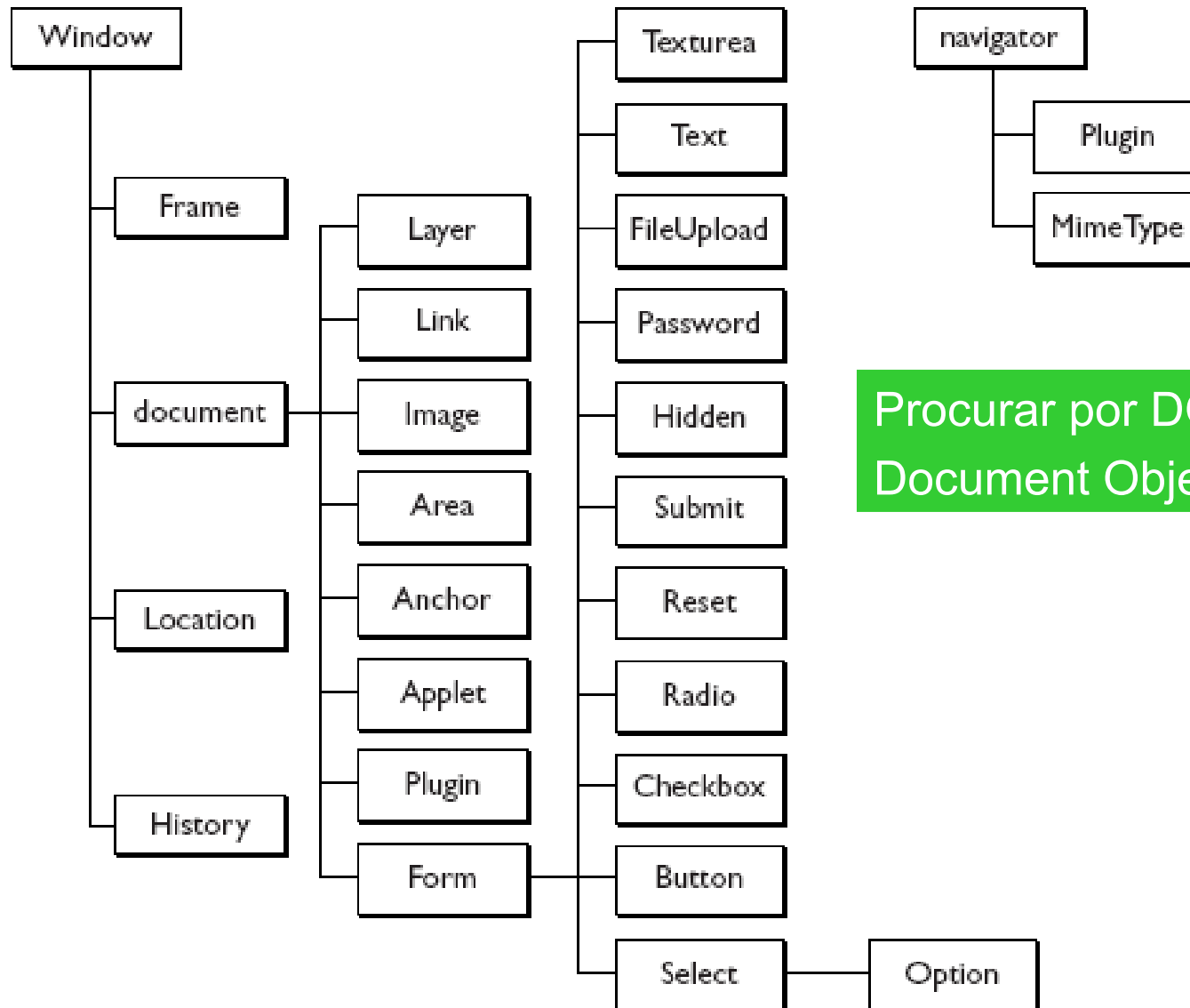
- Funções

- Um conjunto de instruções, agrupadas para executar uma determinada tarefa.
- Para as funções podem ser passadas informações, as quais são chamadas de parâmetros.
- As funções podem ou não retornar alguma informação, o que é feito com o comando Return.

```
function NomeDaFuncao ( [parametros] ) {  
  
    ...  
  
    [return (ValorDeRetorno) ]  
}
```

Objetos do Javascript

Objetos Javascript



Procurar por DOM:
Document Object Model

Objetos Javascript

Objeto Window

Coloque uma informação sobre o site na barra de Status

- Propriedades:
- **defaultStatus** - Conteúdo padrão da barra de status do browser
- **frames** - Vetor que armazena as referências para as frames da janela atual.
 - **parent** - Refere-se a janela pai da frame atual.
 - **self** - Refere-se a janela atual.
- **status** - Define uma mensagem que irá aparecer no rodapé do browser, em substituição por exemplo, a URL de um link, quando estivermos com o mouse sobre o link.
- **top** - Refere-se a janela de nível mais alto do esquema hierárquico do JavaScript.
- **window** - Refere-se a janela atual. Funciona de modo análogo a self.
Ex: `window.status='Qualquer coisa'`

Objetos Javascript

Objeto Window

Peça o nome do usuário e escreva-o na página HTML (prompt.html)

- Métodos:
- **alert** - Mostra uma caixa de alerta, seguido de um sinal sonoro e o botão de OK.
Ex: `alert('Esta é uma janela de alerta!')`
- **close** - Termina a sessão atual do browser.
Ex: `top.close()`
- **confirm** - Mostra uma caixa de diálogo, seguida de um sinal sonoro e os botões de OK e Cancel. Retorna um valor verdadeiro se o usuário escolher OK.
Ex: `retorno=confirm('Deseja prosseguir?')`
- **open** - Abre uma nova sessão do browser, como se o usuário pressionasse <CTRL>+N
- **prompt** – Similar ao alert, permite entrada de texto.

Objetos Javascript

`window.open("URL", "Nome", ["características"]);`

- URL - endereço para abertura da nova janela.
- Nome - nome da nova janela, definido pelo programador.
- Características - série de opções de configuração da nova janela, se especificados devem estar na mesma string, separados por vírgulas e sem conter espaços.
 - toolbar=0 ou 1
 - location=0 ou 1
 - directories=0 ou 1
 - status=0 ou 1
 - menubar=0 ou 1
 - scrollbars=0 ou 1
 - resizable=0 ou 1
 - width=valor inteiro positivo
 - height=valor inteiro positivo

```
window.open("http://  
www.google.com", "Nova", "toolbar=1,location=1,directori  
es=0,status=1,menubar=1,scrollbars=1,resizable=0,width  
=320,height=240");
```

Objetos Javascript

- Mais um exemplo

```
<html>  
  <body>  
    <script>  
      window.location= "http://alice.ufsj.edu.br";  
    </script>  
  </body>  
</html>
```

Exemplo: redirecionar.html

Objetos Javascript

- Outro um exemplo

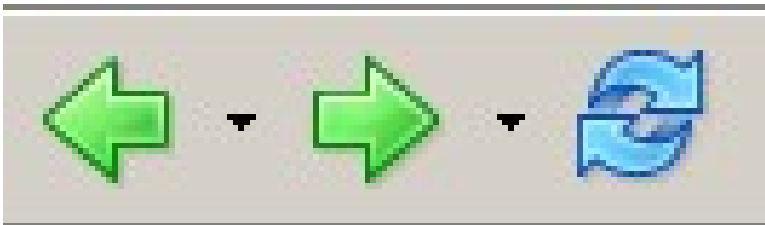
```
<html>
<body>
<script>
document.write(window.location.hostname + "<br>");
document.write(window.location.port + "<br>");
document.write(window.location.protocol + "<br>");
</script>
</body>
</html>
```

Exemplo: location.html

Objetos Javascript

Objeto History

- Métodos
- **back()** - Retorna à página anterior. Equivale a clicar o botão *back* do browser.
- **forward()** - Passa para a próxima página. Equivale a clicar o botão *forward* do browser.
- **go()** - Permite que qualquer URL que esteja presente na relação de páginas visitadas do objeto history, seja carregada. Ex: `history.go(0)` = reload



Ver exemplo: [history.html](#)

Objetos Javascript

- Hierarquia Top – Down

```
<html>
```

```
  <body>
```

```
    <form NAME="clientes">
```

```
      <input TYPE="text" NAME="nome" value="">
```

```
    </form>
```

```
  </body>
```

```
</html>
```

Não é necessário chamar
window.document.forms...

```
document.forms[0].elements[0].value;
```

```
document.clientes.nome.value;
```

Objetos Javascript

Objeto Input TEXT

Ver exemplos:

[validarFormulario.html](#)

[SelecionarCheck.html](#)

- Algumas propriedades:
 - type, size, maxlength, name, value, style
- type=text : entrada de dados texto
- size : Especifica o tamanho do campo na tela.
- maxlength : Especifica a quantidade máxima de caracteres permitidos.
- name : Especifica o nome do objeto
- value : Armazena o conteúdo do campo.
- Style: Propriedades CSS

Utilizando Javascript

Utilizando Javascript

3 maneiras de definir Javascript:

- Inline
 - descritores HTML sensíveis a eventos
- No documento
- Arquivo Externo



Utilizando Javascript

Inline - Exemplos

```
<a href="javascript:alert('Tem  
Certeza?') "> link </a>
```

```
<body  
  onLoad='window.open(this.location);'>
```

```
<form onSubmit="return  
  validarCampos();">
```

Ver exemplo: eventos.html

Utilizando Javascript

Código JavaScript pode ser acionado através de eventos nativos do HTML, como **links** e **botões de submissão** de formulários usando uma URL "javascript":

```
<a href="javascript:alert('Tem  
Certeza?')"> link </a>
```

Utilizando Javascript

Atributo HTML	Quando é executado	Descritores HTML onde é suportado
onclick	Quando um objeto é clicado pelo mouse	<a>, <input>
onselect	Quando um objeto é selecionado	<input type=text>, <textarea>
onchange	Quando o conteúdo é modificado	<input type=text>, <textarea>, <select>
onfocus	Quando um componente de formulário ou janela se torna ativa	<textarea>, <body>, <form>, <input>, <select>, <option>
onblur	Quando um componente de formulário ou janela se torna inativa	<textarea>, <body>, <form>, <input>, <select>, <option>
onmouseover	Quando o mouse está sobre um link	<a>, <area>
onmouseout	Quando o mouse deixa um link	<a>, <area>
onsubmit	Antes de enviar um formulário	<form>
onreset	Antes de limpar um formulário	<form>
onload	Após carregar uma página, janela ou frame	<body>
onunload	Ao deixar uma página, janela ou frame	<body>
onerror	Quando um erro ocorre durante a carga	, <body>
onabort	Quando a carga de uma imagem é abortada	

Utilizando Javascript

Forma 2: No documento

- Bloco delimitado pelos descritores HTML `<SCRIPT>` e `</SCRIPT>`.
- Pode haver vários blocos `<SCRIPT>` em qualquer lugar da página.

```
<script>  
    Alert("Ola mundo");  
</script>
```

Ver exemplo: [javascriptCssCor.html](#)

Utilizando Javascript

Forma 2: No documento

- Possibilidade de definir funções.

```
<script>  
    function ola(mensagem) {  
        alert(mensagem) ;  
    }
```

```
</script>
```

```
(...)
```

Ver exemplo: [mudarCor.html](#)

```
<body onLoad="ola( 'Ola mundo!' ) ;">
```

Utilizando Javascript

Forma 3: Arquivo externo

- Funções a serem utilizadas em vários documentos

```
<head>
```

```
  <script LANGUAGE=JavaScript  
  SRC="scripts.js"></script>
```

```
</head>
```

```
<body onLoad="ola ( 'Ola mundo!' ) ;">
```

Lembrar de Endereços absolutos e relativos!

Neste exemplo o scripts.js está na mesma pasta da página.

Trabalhando com textos

- Tamanho da string (atributo length)
- Parte da string (substring())
- Como saber se um e-mail contém uma arroba?
- Como saber se o CPF digitado está correto?
- Como concatenar nome e sobrenome?

Ver exemplo:
[validarFormulario.html](#)

Temporizando eventos

```
var temp =  
    setTimeout("alert('oi')", 5000);  
clearTimeout(temp);
```

Será possível fazermos animações com Javascript?

Ver exemplo:
[temporizacao.html](#)
[temporizacao-2.html](#)
[temporizacao-3.html](#)

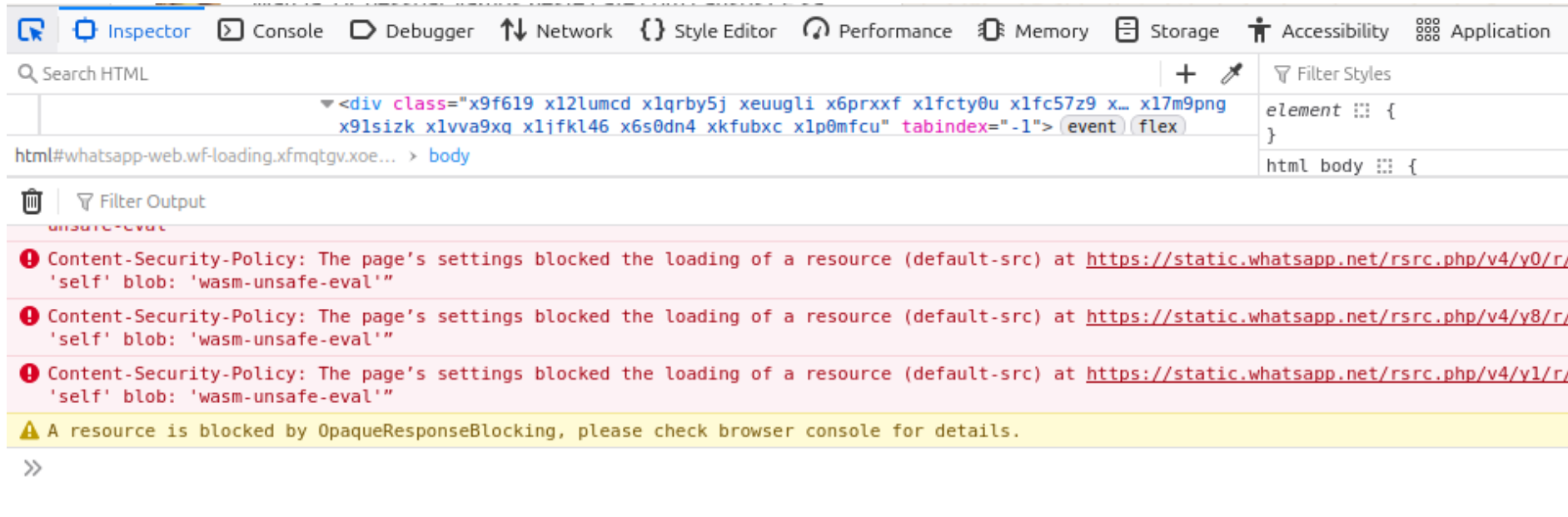
Trabalhando com arrays

- Estruturas de dados simples (inteiro, char, bool)
- Estruturas de dados compostas (Strings, Datas, arrays, ...)
- Toda String é um array
- Todo array possui tamanho
- Podemos acessar uma posição do array usando sua posição.
- Array em Javascript aceita tipos distintos
- Possibilidade de acessar por nome
- Arrays podem implementar árvores

Ver exemplo:
[tabelaDeCores.html](#)

Ver exemplo:
[einstein.html](#)

Ferramentas



Ctrl + Shift + C

XML DOM + Manipulação de Eventos

XML

- Estrutura de dados
- Permite uma estrutura de árvore
- Dados em texto
- Tags, atributos e valores
- Independe de linguagem de programação
- Permite integração de sistemas distintos

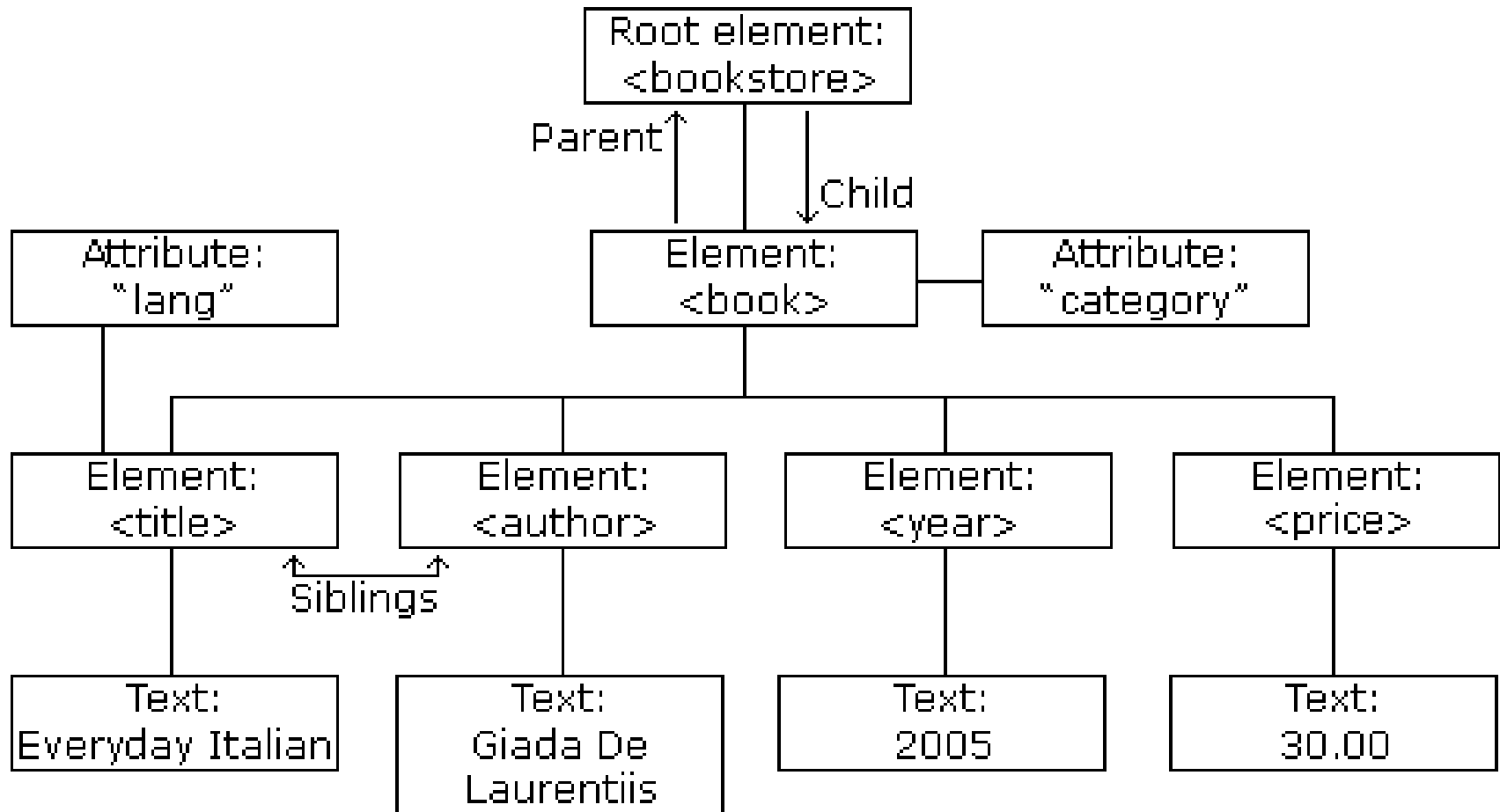
XML

```
<bolo>
<ingredientes qtd=3>
    <ingrediente nome=farinha qtd=2 medida=copo />
    <ingrediente nome=ovo qtd=3 medida=unidade />
    ...
</ingredientes>
<preparo>
    <passo numero=1 desc="Quebre os ovos" />
    ...
</preparo>
</bolo>
```

DOM

- Tecnologia XML (Há também o SAX)
- Document Object Model
- Disponível em várias linguagens
- Maneira de ler um XML como árvore
- Nós pais, filhos e vizinhos

XML DOM - Árvores



HTML DOM

- **Acessando Objetos HTML**
- **`document.getElementById('id')`**
 - Id único na página. Retorna apenas um elemento
- **`getElementsByTagName('tag')`**
 - Retorna todos os elementos definidos por esta tag

`elementoPorId.html`

`elementoPorNomeTag.html`

HTML DOM

- Acessando atributos de uma tag diretamente

```
<div id='help' class='help'  
    style='background-color:yellow'>Texto</t  
exto>
```

```
<script>
```

```
document.getElementById("help").className=  
    "nova";
```

```
</script>
```

tabelaColorida1.html

HTML DOM

- Acessando atributos de uma tag por DOM

```
<div id='help' class='help'
    style='background-color:yellow'>Texto</t
exto>
<script>
document.getElementById("help").setAttribu
te("className","nova");
</script>
```

TabelaColorida2.html

TabelaColorida3.html

HTML DOM

- Todo Atributo pode ser acessado diretamente ou pelos métodos
- `getAttribute("nome")`
- `setAttribute("nome", "valor")`
- O conteúdo da TAG é o `innerHTML`.

InnerHTML.html

HTML DOM

- Prática!
 - Criar uma tabela colorida utilizando CSS para definir cores pares e ímpares;
 - Possível mudar a cor do texto?
 - É possível adicionar numeração nas linhas da tabela?

Javascript DOM - propriedades

childNodes – Array de nós filhos

firstChild – Primeiro nó filho

lastChild – Último nó filho

nodeName – Nome do nó

nodeType – Tipo do nó

nodeValue – Valor contido no nó

nextSibling – Próximo nó com o mesmo pai (irmão)

previousSibling – Nó anterior com o mesmo pai

parentNode – Nó pai

Javascript DOM - métodos

AppendChild – adiciona um novo nó filho

HasChildNodes – booleano se o nó tem filhos

RemoveChild – remove um nó filho

CreateAttribute – Cria um novo atributo para o elemento

CreateElement – Cria um novo elemento documento

CreateTextNode – Cria um item texto

Javascript DOM - métodos

Prática!

- Adicionar contador de caracteres

[adicionandoElemento.html](#)

Manipulação de Eventos

- A partir de um evento, alterar um atributo do HTML
- Nem todo atributo é acessado da mesma forma

[adicionandoEventos.html](#)

Manipulação de Eventos

```
function digaQueClicou() {  
    alert('Clicou no div');  
}  
  
var elementos =  
    document.getElementsByTagName('DIV');  
for (var i = 0 ; i < elementos.length ; i++){  
    el = elementos[i];  
    el.style.cursor = 'pointer';  
    if (el.addEventListener){ // ← MOZILLA!  
        el.addEventListener('click', digaQueClicou,  
false);  
    } else if (el.attachEvent){ // ← IE!  
        el.attachEvent('onclick', digaQueClicou);  
    }  
}
```

HTML DOM

- Prática!
 - Adicionar evento para alterar cor com mouseover e mouseout dinamicamente.
 - Criar formulário com máscara.

[mascara.html](#)

XHTML

HTML

- Linguagem de marcação de Hipertexto
- Conjunto de Tags definidas
 - Variam com navegador
- Conjunto de atributos definidos
- Tags com funções específicas

XHTML

- X = eXtensible
- HTML extendido
 - Criação de novas tags
 - Criação de novos atributos
- O navegador irá ignorar tags e atributos desconhecidos
- O Javascript poderá fazer uso delas

Criando novos atributos

- A criação de novos atributos HTML possibilita a definição de novos elementos criados a partir de elementos comuns e adição de eventos a estes.

quadro.html + quadro.js

Como fazer um quadro abre/fecha em HTML?

Validação de formulário

- E se os campos de um formulário tivessem a informação se o mesmo é obrigatório ou não?
- E se eles possuíssem a informação sobre seus tipos?
- Isto é possível?

`validandoFormulario.html`

Possibilidade de definir tipos para um formulário

Utilizando componentes - 1

- Há vários componentes disponíveis no mercado que possibilitam a extensão do HTML.
- Muitos destes componentes são gratuitos.
- Exemplos:
- www.dhtmlgoodies.com

`multipleSelect.html +
multipleSelect.js`

Como fazer um select como
Desktop?

Utilizando componentes - 2

- Tabela ordenável
- Definição de tipos no cabeçalho
- Quais tipos este componente suporta?
- Qual o método de ordenação utilizado?
- Esta tabela funcionaria para 1000 linhas?

Exemplo na web:

http://mochikit.com/examples/sortable_tables/index.html

Orientação a Objetos

Orientação a Objetos

- Programação Estruturada
 - Estrutura de Dados (struct, record, ...)
 - Funcionalidades (funções e procedimentos)
- Orientação a Objetos
 - Estrutura de dados + funcionalidades
 - Estrutura de dados = atributos
 - Funcionalidades = métodos

Orientação a Objetos

- Herança
- Polimorfismo
- Agregação
- Composição
- Encapsulamento

Escopo de Variáveis

```
<script>  
  var i = 0;  
  function a(){  
    var j = 0;  
  }  
  alert(i);  
  alert(j);  
</script>
```

exemplo01.html

Escopo delimitado por chaves

Escopo de Variáveis

```
<script>  
  var i = 0;  
  function a(){  
    var j = 0;  
  }  
  alert(i);  
  alert(a().j);  
</script>
```

exemplo02.html

Variáveis não possuem acesso público

Escopo de Variáveis

```
<script>  
  var i = 0;  
  function a(){  
    var j = 0;  
  }  
  alert(i);  
  alert(a.j);  
</script>
```

exemplo03.html

Funções são usadas como classes

Escopo de Variáveis

```
<script>
  var i = 0;
  function a(){
    this.j = 0;
  }
  alert(i);
  alert(a.j);
</script>
```

exemplo04.html

Palavra-reservada “this” torna o acesso interno possível

Para acesso é necessário instanciar o objeto

Escopo de Variáveis

```
<script>  
  var i = 0;  
  function a(){  
    this.j = 0;  
  }  
  alert(i);  
  var instancia = new a();  
  alert(instancia.j);  
</script>
```

exemplo05.html

Classes são funções

Instanciando-as acessaremos
seus atributos

Escopo de Variáveis

```
<script>  
  function Pessoa(){  
    this.nome = "Nando";  
    var idade = "33";  
  }  
  var instancia = new Pessoa();  
  alert(instancia.nome);  
  alert(instancia.idade);  
</script>
```

exemplo06.html
Qual será mostrado?

Encapsulamento

```
<script>
```

```
function Pessoa(){  
    this.nome = "Paulo";  
    var idade = "21";  
    function getIdade(){  
        return idade;  
    }  
}  
  
var instancia = new Pessoa();  
alert(instancia.nome);  
alert(instancia.getIdade());
```

```
</script>
```

exemplo07.html

Tentativa de Encapsulamento

Encapsulamento

```
<script>
  function Pessoa(){
    this.nome = "Alirio";
    var idade = "24";
    this.getIdade = function (){
      return idade;
    }
  }
  var instancia = new Pessoa();
  alert(instancia.nome);
  alert(instancia.getIdade());
</script>
```

exemplo08.html

Encapsulamento

O acesso ao atributo só pode ser feito por métodos

Encapsulamento

```
<script>
```

```
function Pessoa(){  
    this.nome = "Alirio";  
    var idade = "24";  
    this.getIdade = function (){  
        return idade;  
    }  
    this.setIdade = function (idadeV){  
        idade = idadeV;  
    }  
}  
var instancia = new Pessoa();  
alert(instancia.nome);  
instancia.setIdade("25");  
alert(instancia.getIdade());
```

```
</script>
```

exemplo09.html

Encapsulamento

O acesso ao atributo só pode ser feito por métodos

Encapsulamento

```
<script>
  function Pessoa(){
    this.nome = "Lauro";
    var idade = "43";

    Pessoa.prototype.getIdade =
    function getIdade(){
      return idade;
    }
    Pessoa.prototype.getNascimento =
    function getNascimento(){
      return idade;
    }
  }
  var p = new Pessoa();
  alert(p.nome + " " + p.getIdade() + " " + p.getNascimento());
</script>
```

exemplo10.html

Utilização do Prototype

Herança

```
function Eletrodomestico() {  
    this.ligado = false;  
    this.ligar = function() {  
        this.ligado = true;  
    }  
    this.desligar = function() {  
        this.ligado = false;  
    }  
}
```

Heranca.html

Definição da classe mãe

Herança

```
function Ventilador(velMax) {  
  
    var maximaPermitida = 5; // Uso de encapsulamento  
    var velocidadePadrao = 3; // Variáveis privadas  
    if (velMax > 0 && velMax <= maximaPermitida) {  
        this.velocidadeMaxima = velMax;  
    } else {  
        this.velocidadeMaxima = velocidadePadrao;  
    }  
}
```

Heranca.html

Definição da classe filha

Herança

```
Ventilador.prototype = new Eletrodomestico();
```

```
// Define o objeto protótipo
```

```
ventilador = new Ventilador(4);
```

```
alert(ventilador.ligado); // Retorna false
```

```
ventilador.ligar();
```

```
alert(ventilador.ligado); // Retorna true
```

```
</script>
```

Heranca.html

Criando a Herança com prototype

Composição

```
function Livro(titulo, autor) {  
    this.titulo = titulo;  
    this.autor = autor;  
}
```

```
function Capitulo(titulo, paginas) {  
    this.titulo = titulo;  
    this.paginas = paginas;  
}
```

Composicao.html

Um Objeto composto de outro
objeto

Composição

```
var livro = new Livro("Os Três Mosqueteiros", "Alexandre Dumas");  
var capitulo1 = new Capitulo("Os três presentes do sr. D'Artagnan pai", 11);  
var capitulo2 = new Capitulo("A antecâmara do sr. Tréville", 8);
```

```
// Os objetos do tipo Capitulo fazem parte da composição do objeto livro  
livro.capitulo1 = capitulo1;  
livro.capitulo2 = capitulo2;
```

```
// Acessando as propriedades:  
alert(livro.titulo + " - " + livro.autor + "\\n\\t" +  
      livro.capitulo1.titulo + " - " +  
      livro.capitulo1.paginas + " páginas\\n\\t" +  
      livro.capitulo2.titulo + " - " +  
      livro.capitulo2.paginas + " páginas");
```

Modelo simples: Um livro é
composto de capítulos

Requisição Assíncrona

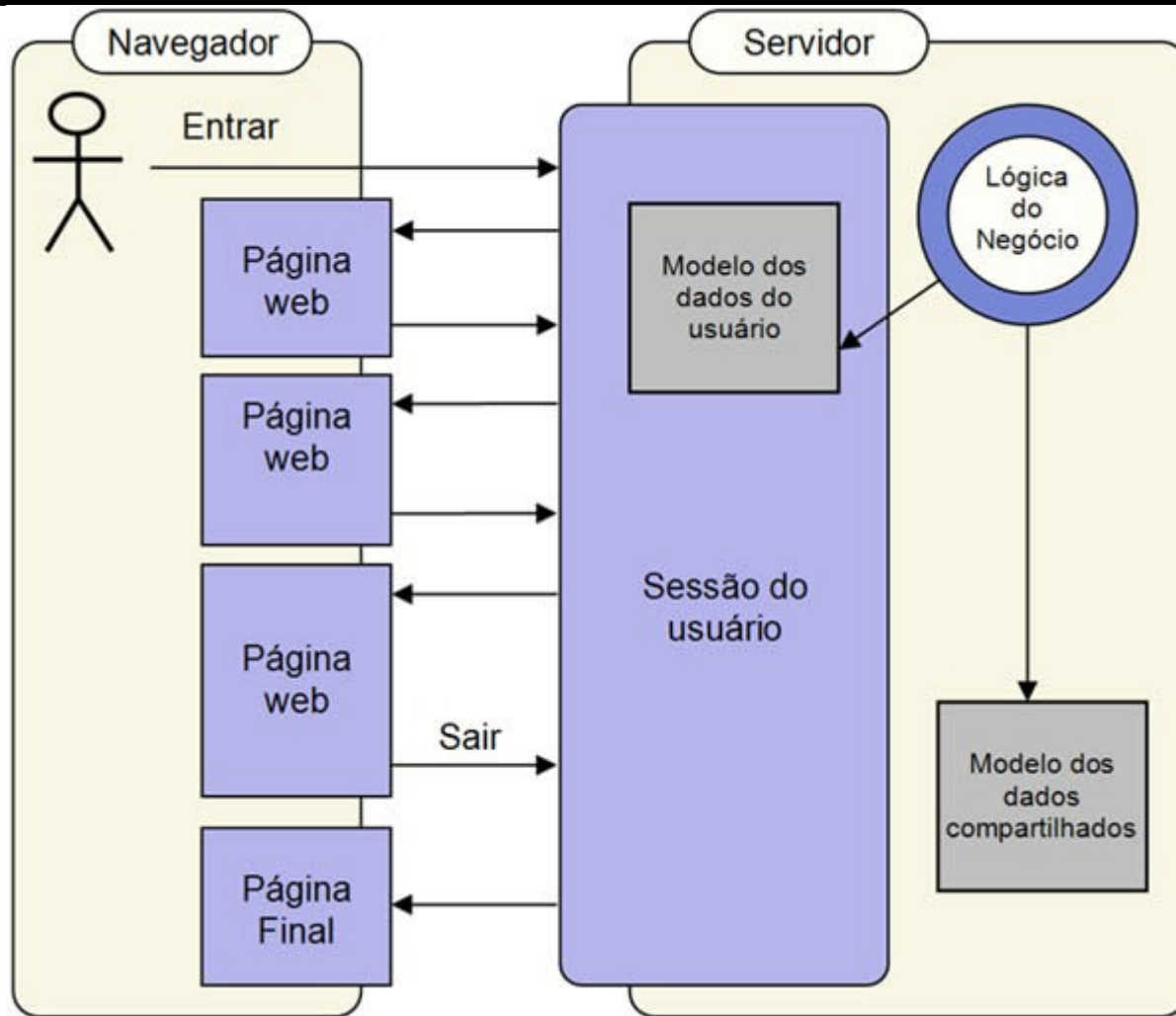
Porque comunicação Assíncrona?

- Com Javascript é possível:
 - Arrastar e soltar (Drag'n'Drop)
 - Menus pop-up
 - Ordenar listas
 - Efetuar cálculos
 - Validar formulários
- De que adianta mostrar ao usuário um resultado que não corresponde aos dados do servidor?

Porque comunicação Assíncrona?

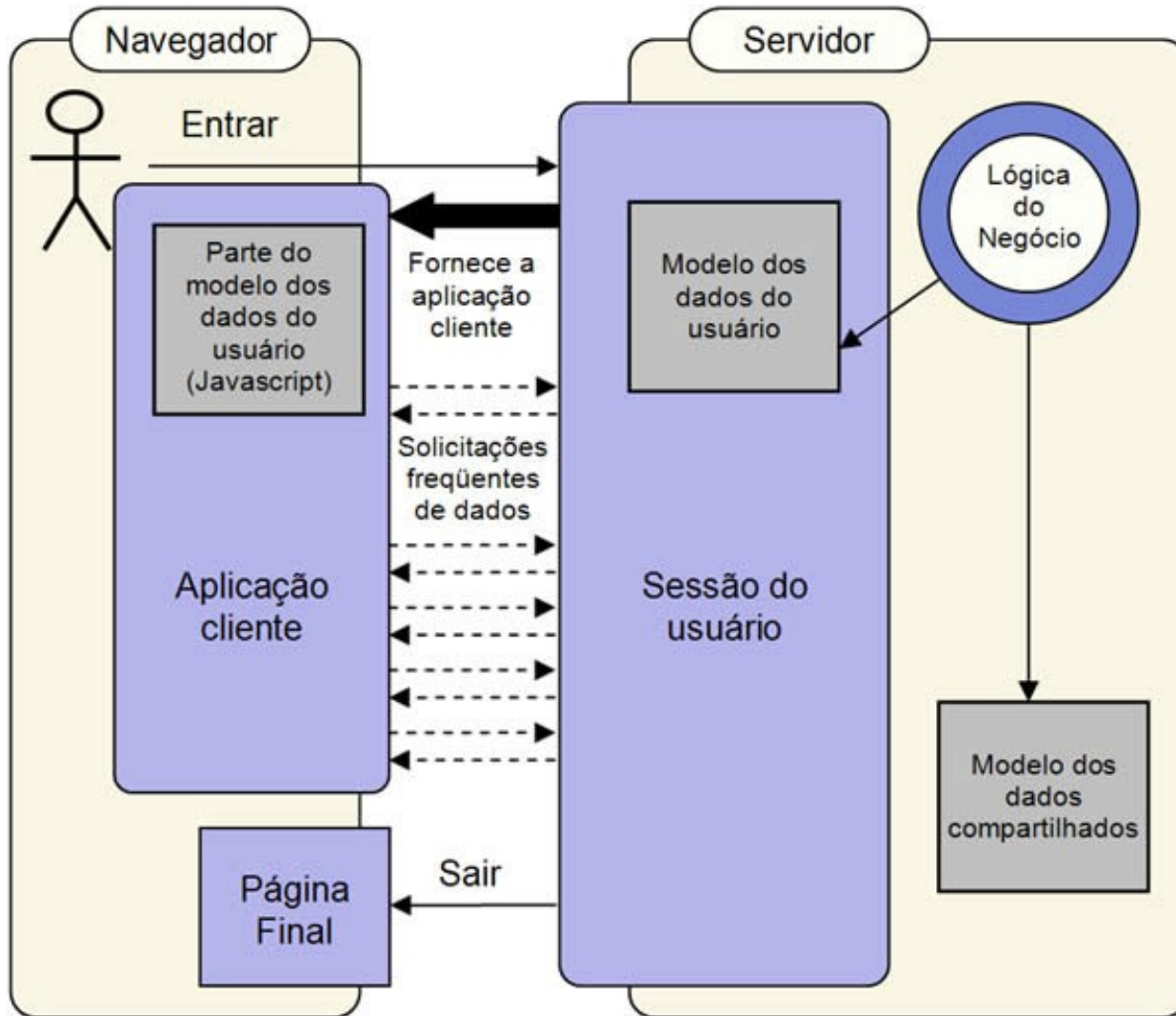
- Atualização de trechos da página
- Melhor aproveitamento de banda
- Divisão de carga: Parte do processamento é feito no cliente
- Processar dados no servidor enquanto o cliente executa outras tarefas.

Modelo tradicional



[http://pt.wikipedia.org/wiki/AJAX_\(programa%C3%A7%C3%A3o\)](http://pt.wikipedia.org/wiki/AJAX_(programa%C3%A7%C3%A3o))

Com Ajax



Exemplo: webmail

- Arrastar e soltar emails lidos para a lixeira
- Buscar lista de endereços para enviar novos emails
- Adicionar usuários a estas listas
- Teclas de atalho (Ctrl + D = Remover)
- Se os emails arrastados para a lixeira não forem apagados?
- Como avisar o servidor de uma ação Javascript?

XMLHttpRequest!

XMLHttpRequest

Criando a requisição

```
var req = null;
if (window.XMLHttpRequest) { // ← Mozilla!
    req = new XMLHttpRequest();
} else if (window.ActiveXObject) { ← IE!
    try {
        req = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            req = new
ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
    }
}
```

Recebendo resposta

```
req.onreadystatechange = function() {  
    if(req.readyState == 4) {  
        if(req.status == 200) {  
            //Caso ok req.responseText ou response.XML  
        } else {  
            // Caso erro req.statusText  
        }  
    }  
};  
req.open("GET", pagina, true);  
req.send(null);
```

exemploAjax.html

Propriedades do objeto

- **onreadystatechange** - Associa um evento que será chamado a cada alteração do objeto
- **readyState** – Contém o estado do objeto
- **responseText** – resposta no formato String
- **responseXML** – resposta no formato XML
- **status** – estado da requisição no formato numérico
- **statusText** – estado da requisição no formato texto

Estado do objeto: readyState

- 0 – O objeto não foi inicializado com dados
- 1 – O objeto está carregando seus dados
- 2 – O objeto terminou de carregar seus dados
- 3 – O dados do objeto não estão totalmente carregados (usuário já pode interagir)
- 4 – Os dados do objeto estão completamente carregados

Estado da resposta

200 – OK

204 – Documento vazio

301 – Documento migrado para outro endereço

401 – Não autorizado

403 - Proibido

404 – Não encontrado

408 - Timeout

500 – Erro no servidor

Estado da resposta

Lembrando que sem servidor, não haverá estado da resposta e o mesmo será ZERO.

`escrevendoNoDocumento.html`

Métodos

- **abort()** – Cancela a requisição
- **getAllResponseHeaders()** - retorna todos os cabeçalho HTTP no formato String
- **getResponseHeader()** – retorna o cabeçalho HTTP invocado pelo método
- **open()** – especifica os atributos necessários para fazer a conexão com o servidor
- **setRequestHeader()** – Adiciona um conjunto chave / valor para o cabeçalho quando o mesmo for enviado

Método open()

```
open("método","url",true/false)
```

- O primeiro parâmetro é o método pedido HTML – GET, POST, HEAD ou outro método qualquer. **Mantenha o nome do método em maiúsculas.**
- O segundo parâmetro é a URL da página que está a pedir.
- O terceiro parâmetro define se o pedido é assíncrono. Se TRUE, a execução da função JavaScript irá continuar enquanto que a resposta do servidor ainda não foi recebida. Isto é o A de AJAX.

Métodos HTTP

- GET - método mais comum, requisição de uma página
- HEAD – Apenas o cabeçalho do GET
- POST – Similar ao GET. Maior e permite enviar dados
- PUT – Envia uma nova versão do arquivo requisitado
- DELETE – Remove o arquivo requisitado
- TRACE – Envia de volta uma cópia da requisição para monitorar seu progresso
- OPTIONS – Retorna uma lista dos métodos disponíveis
- CONNECT – requisição baseada em proxy para tunel SSL

Atenção!

O método `open("método","url",false)` só é capaz de acessar "url" dentro do próprio domínio não sendo capaz de acessar endereços absolutos, ou seja, endereços que começam com `http://www.....`

Método send()

O parâmetro do método send() pode ser constituído por quaisquer dados que pretenda enviar ao servidor ao enviar (POST) o pedido.

Valor comum: **null**!

Os dados devem estar sob a forma de uma linha de texto de pergunta, tipo:
nome=valor&outroNome=outroValor

Note-se que se pretende enviar (**POST**) dados, você deve alterar o tipo MIME do pedido usando a seguinte linha:

```
req.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

De outra forma o servidor irá ignorar os dados.

onreadystatechange

O que fazer quando o estado for alterado?

```
req.onreadystatechange = alertContents;
```

```
function alertContents() {  
    // Alguma coisa  
}
```

```
req.onreadystatechange = function() {  
    // Alguma coisa  
};
```

Esta função será chamada toda vez que o estado o objeto for alterado, ou seja, 1, 2, 3 e 4

Exemplo

- É possível carregar apenas a região de conteúdo de um site?
- Quais as vantagens disto?

siteEmAjax/index.html

Exercício

- Criar um formulário que a partir do país seleciona o estado, a partir do estado seleciona a cidade e a partir da cidade.
- Integrar com PHP e banco de dados
- Alterar o metodo send() para enviar parâmetros

formularioDinamico.html

Cadê o X do Ajax?
XML

Porque XML?

- Separação do conteúdo da formatação
- Simplicidade e Legibilidade, tanto para humanos quanto para computadores
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura (Chamados DTDs)
- Interligação de sistemas distintos
- Concentração na estrutura da informação, e não na sua aparência
- API padrão em várias linguagens (SAX/DOM)

Exemplo de XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
<título>Pão simples</título>
<ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
<ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
<ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
<ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
<Instruções>
<passo>Misture todos os ingredientes, e dissolva bem.</passo>
<passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
<passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
</Instruções>
</Receita>
```

Porque XML?

- Como preencher vários campos ao mesmo tempo?
- Como separar o conteúdo da resposta?
- Como enviar do servidor para o cliente dados estruturados?

XML = Dado estruturado e navegável

Requisição para XML

`req.responseText` → `req.responseXML`

E o dado recebido? Como saber que ele é XML?

Dados em XML

```
var doc = null;
var text = ajax.responseXML;
if (window.ActiveXObject) {
try{
    doc=new ActiveXObject("Microsoft.XMLDOM");
    doc.async="true";
    doc.loadXML(text);
    }catch (e){
alert(e);
    }
} //Continua...
```

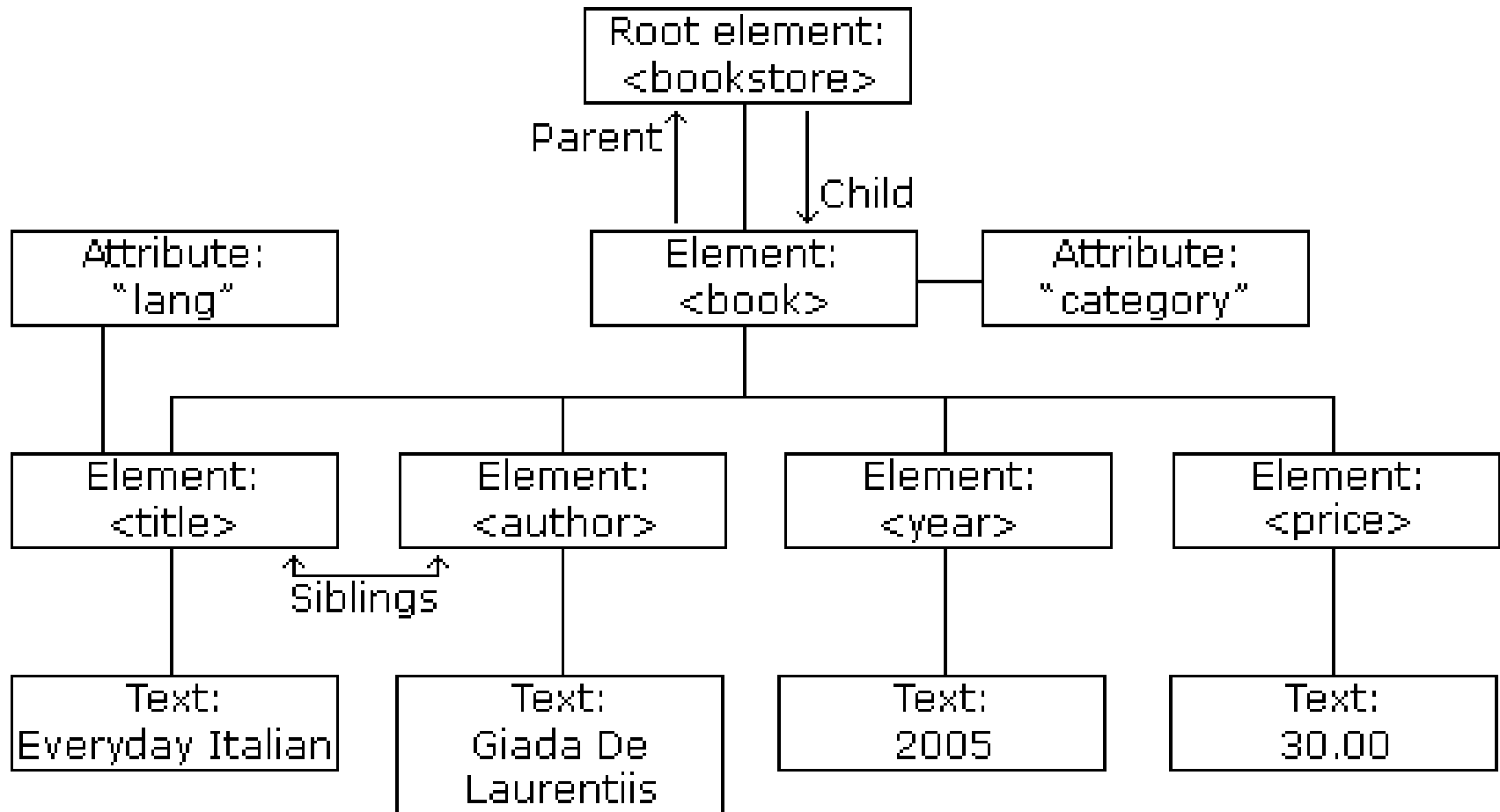
Dados em XML

```
// code for Mozilla, Firefox, Opera, etc.  
else {  
var parser=new DOMParser();  
    doc=parser.parseFromString(text,"text/xml");  
}
```

Navegando em XML

Como buscar um trecho do documento?

XML DOM - Árvores



Javascript DOM - propriedades

childNodes – Array de nós filhos

firstChild – Primeiro nó filho

lastChild – Último nó filho

nodeName – Nome do nó

nodeType – Tipo do nó

nodeValue – Valor contido no nó

nextSibling – Próximo nó com o mesmo pai (irmão)

previousSibling – Nó anterior com o mesmo pai

parentNode – Nó pai

Javascript DOM - métodos

AppendChild – adiciona um novo nó filho

HasChildNodes – booleano se o nó tem filhos

RemoveChild – remove um nó filho

CreateAttribute – Cria um novo atributo para o elemento

CreateElement – Cria um novo elemento documento

CreateTextNode – Cria um item texto

Javascript DOM - métodos

```
for(var i = 0 ; i < cep.childNodes.length ; i++){  
    if(cep.childNodes[i].nodeType != 1)  
        continue;  
    var valor = cep.childNodes[i].firstChild.data;  
    var elemento = cep.childNodes[i].nodeName;  
    switch(elemento){  
        case 'cidade':  
            document.getElementById('cidade').value = valor;  
            break;  
        (...)  
    }
```

Ajax e Webservice

- Possibilidade de comunicação assíncrona com servidor
 - Troca de grande quantidade de dados
 - Integração
-
- http://republicavirtual.com.br/web_cep.php?cep=87013230
 - http://cotacao.republicavirtual.com.br/web_cotacao.php?formato=xml

Exemplo: cep.html

Referências

- <http://www.w3schools.com/js/>
- <http://www.imasters.com.br>
- <http://www.scriptbrasil.com.br/>
- <http://www.scriptfacil.com.br/>
- <http://www.w3.org/TR/html4/interact/scripts.html>
- <http://www.w3.org/DOM/>
- <http://www.criarweb.com/javascript/>
- <http://javascript.internet.com/>
- <http://www.javascript.com/>
- <http://www.javascriptkit.com/>
- <http://www.mozilla.org/js/>
- <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- <http://www.codebeach.com/>
- <http://www.microsoft.com/scripting/>
- http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide
- http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Class-Based_vs._Prototype-Based_Languages
- <http://www.javascript-tutorial.com.br/>
- Tutorial online - <http://unix.wmonline.com.br/js/tutoriais/cursojs/index.htm>

Referências

- <http://www.ajaxprojects.com/>
- <http://www.imasters.com.br>
- <http://www.w3.org/DOM/>
- http://republicavirtual.com.br/web_cep.php?cep=87013230
- http://cotacao.republicavirtual.com.br/web_cotacao.php?formato=xml
- <http://wiki.script.aculo.us/scriptaculous/show/Demos>
- <http://openrico.org/>
- <http://getahead.ltd.uk/dwr>
- <http://code.google.com/>
- <http://developer.yahoo.com/yui>
- <http://www.gmail.com>
- <http://www.flickr.com>
- <http://labs.google.com/suggest>
- <http://www.backbase.com/>
- <http://www.start.com/3/>
- <http://developer.mozilla.org/pt/docs/AJAX>