JQUERY – AJAX

DIOGO CEZAR TEIXEIRA BATISTA
http://inf.cp.utfpr.edu.br/diogo
diogo@diogocezar.com

O QUE É 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 assincronamente;
  - permite que aplicação funcione enquanto dados são transferidos.

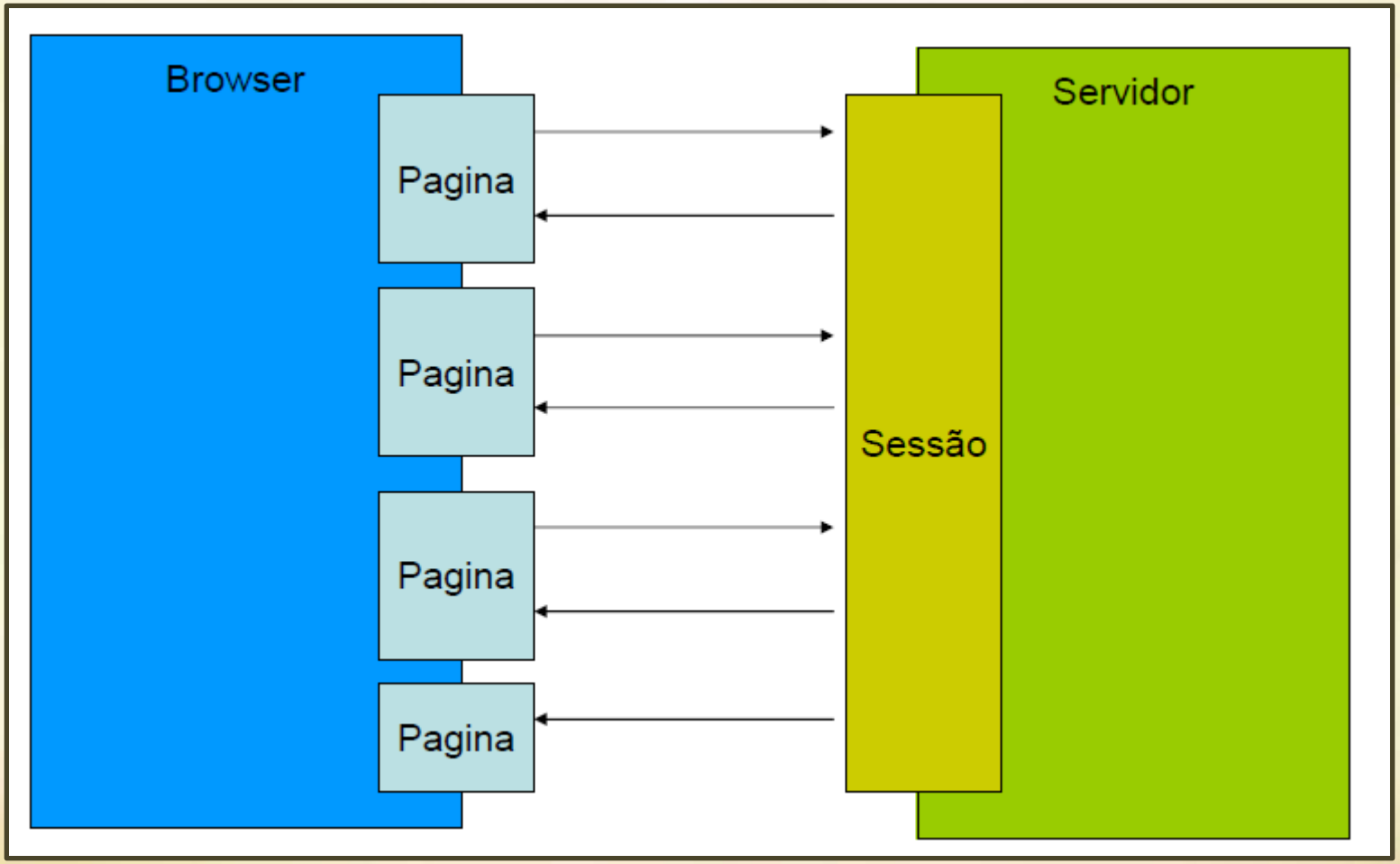
QUANDO USAR?

- 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.

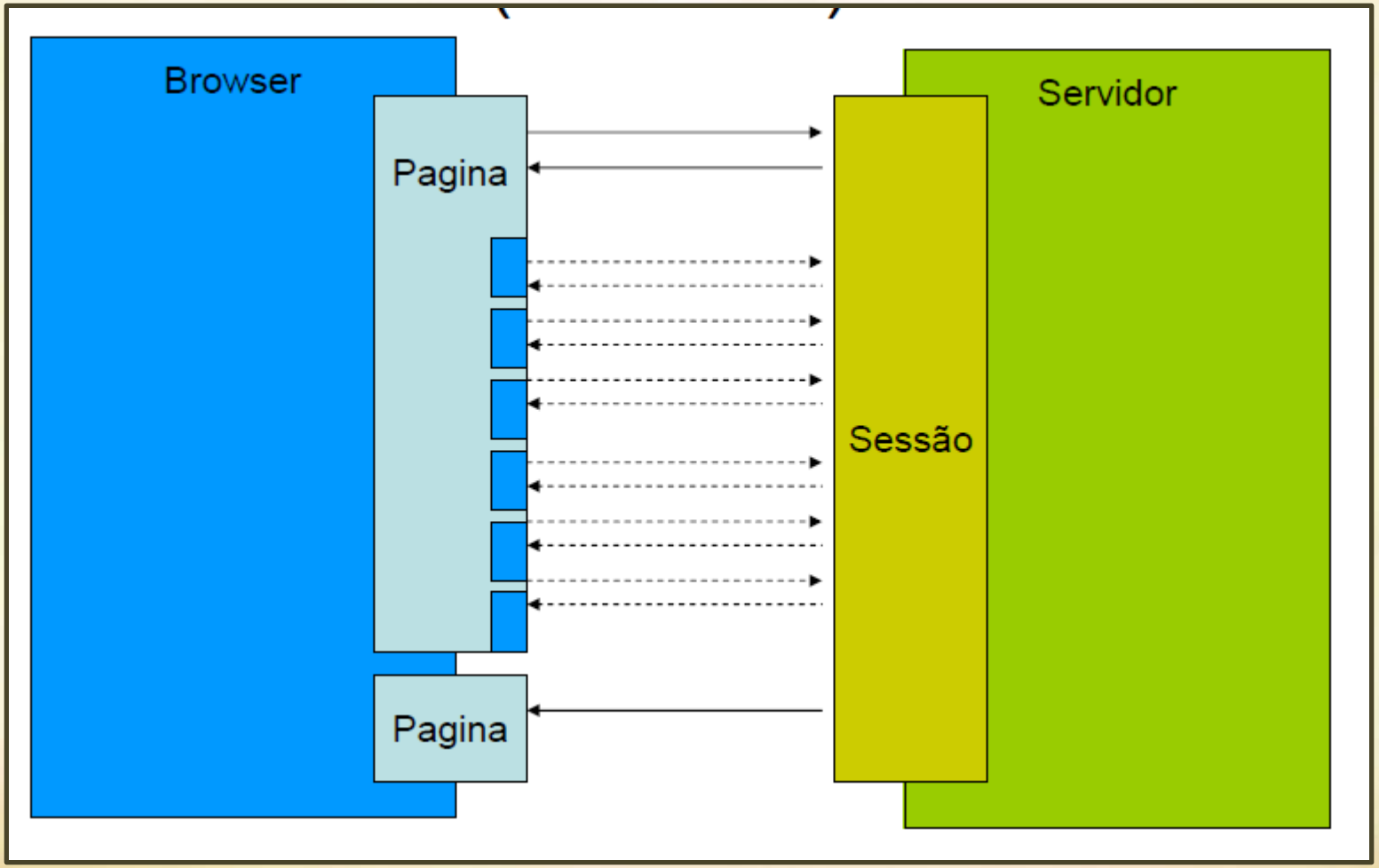
QUEM USA? EXEMPLOS

- Google;
- Yahoo;
- Microsoft;

CICLO DE VIDA DE APLICAÇÃO WEB



CICLO DE VIDA DE APLICAÇÃO AJAX (WEB 2.0)



MARAVILHAS DO JQUERY

- uma das principais funcionalidades do JQuery é o gerenciamento de AJAX;
- funções próprias para controle e gerenciamento da transmissão de dados;
- em geral: CONSEGUE-SE ACESSAR UM CONTEÚDO DE UMA PÁGINA PHP DENTRO DO JAVASCRIPT!
- existem diferentes formas para se trabalhar com AJAX em JQUERY, basicamente todas seguem a mesma lógica, mas a sintaxe pode ser diferenciada;

COMO FAZER? – LOAD HTML

- Sintaxe:
- load(url,parâmetros,callback)
  - url: A URL que é solicitada a requisição.
  - parâmetros: Um objeto cuja propriedades são serializadas em uma série de parâmetros codificados corretamente e que se passam a requisição. Se utilizado, deve-se especificar se a requisição utiliza o método POST, caso contrário, se omitido, a requisição utiliza o método GET.
  - callback: Uma função chamada após a solicitação já ter sido processada.
- Esse comando serve apenas para ler HTML's

EXEMPLO 1

- carrega o conteúdo de arquivo.html na div com id conteúdo;

```
$("#conteudo").load("arquivo.html");
```

EXEMPLO 2

- carrega o conteúdo de arquivo.html na div com id conteúdo e dispara a função aviso() quando o conteúdo for lido;

```
<script type="text/javascript">
$(document).ready(function() {
    $("input[type=button]").click(function(event) {
        $("#conteudo").load('arquivo.html',aviso());
    });
});
function aviso(){
    alert('O conteúdo será carregado agora!');
}
</script>
```

FILTRANDO O RESULTADO

- nem sempre é interessante manipular tudo o que o Script PHP escrever em tela;
- por isso temos a opção de filtrar o resultado obtido, com os seletores disponíveis pelo próprio JQuery;

```
$('#box').load('meu_arquivo.html #conteudo')
```

- o exemplo acima só irá retornar o div com id conteudo do arquivo meu_arquivo.html
- esse recurso é exclusivo do método load();

COMO FAZER? - $.POST / $.GET

- $.post(url, parametros, funcao_retorno(data));
  - url: A URL que é solicitada a requisição.
  - parâmetros: Um objeto cuja propriedades são serializadas em uma série de parâmetros codificados corretamente e que se passam a requisição.
  - função_retorno(data) : Uma função chamada após a solicitação já ter sido processada.
  - data é o resultado retornado
- utilizado para resgatar recursos das linguagens de programação, em nosso caso, páginas PHP.

USANDO GET E POST

- até agora não passamos parâmetros para serem tratados nas páginas de destino;
- ao utilizar $.post os atributos serão enviados por post;
- ao utilizar $.get os atributos serão enviados por get
- para passar parâmetros deve-se utilizar a seguinte estrutura:

```
{variavel1: valor1,
 variaval2: valor2,
 ...
 variaveln: valorn}
```

EXEMPLO

```
<script type="text/javascript">
$(document).ready(function() {
    $("input[type=button]").click(function(event) {
        var acao = $(this).attr("value");
        $.post('requisicao.php',{acc:acao},
           function(data){
               $("#resultado").empty().html(data);
            }
        );
    });
});
</script>
```

O retorno para a div será o que for impresso pelo PHP

```
<?php
$acao = $_POST['acao'];
if(empty($acao)){
   echo "Retorno";
}
?>
```

AÇÕES ANTES E DEPOIS DO AJAX

- é possível estabelecer ações para outras divs enquanto o AJAX é executado, ideal para criar uma div "Loading"
- para isso, utiliza-se os comandos:
  - $('#loading').ajaxStart('faça alguma coisa');
  - $('#loading'). ajaxStop('faça alguma coisa');
- existem ainda outros comandos de controle para o Ajax:
  - ajaxComplete(callback)
  - ajaxError(callback)
  - ajaxSend(callback)
- http://docs.jquery.com/Ajax

$.AJAX

- existe ainda uma terceira forma de se trabalhar com Ajax usando o objeto ajax;
- é uma implementação "baixo-nível" dos métodos vistos anteriormente $.POST e $.GET;
- maiores detalhes podem ser encontrados:
  - http://docs.jquery.com/Ajax/jQuery.ajax#options

ATIVIDADES

- A partir do exemplo que consulta nomes ao pressionar uma tecla no input, desenvolva:
- Um gerenciador genérico:
  - Deverá receber como parâmetro:
    - Qual tabela vai consultar?
    - Quais campos deverá comparar?
    - O que foi digitado?
- Dessa forma você terá um gerenciador genérico para todas as páginas de seu projeto integrador.