Baseado no material da W3C

Sistemas Web III

Web 2.0 e AJAX

Tiago Cruz de França

tcruz.franca@gmail.com

Web 2.0

Web 2.0

- **A web não impõe fronteiras rígidas**
  - Define apenas um conjunto de práticas e princípios para interligar sites
- **A web 2.0 surgiu como uma nova web com maior interatividade e maiores possibilidades**
  - Interação do ponto de vista da aplicação
    - **Melhor experiência para os usuários**
    - **Novos e melhores serviços (recursos)**
    - **Novas aplicações com maior interação entre humanos**
      - Mídias sociais online (ainda mais participação ativa dos humanos)
      - Colaboração -> inteligência coletiva (sabedoria das multidões)

Web 2.0

- **Do ponto de vista técnico**
  - Software é visto como serviço não mais como produto
    - **Serviço ou conjunto de recursos disponíveis**
  - Modelos de programação
    - **AJAX, SOAP e REST**
      - Sistemas com baixo acoplamento
    - **Representações dos dados: agora semiestruturados XML ou JSON**
      - É possível uso de outros formatos*, inclusive texto simples
  - Serviços/Recursos estão disponíveis
    - **Funcionalidades de um sistema provido por uma entidade pode ser utilizado em outros serviços**

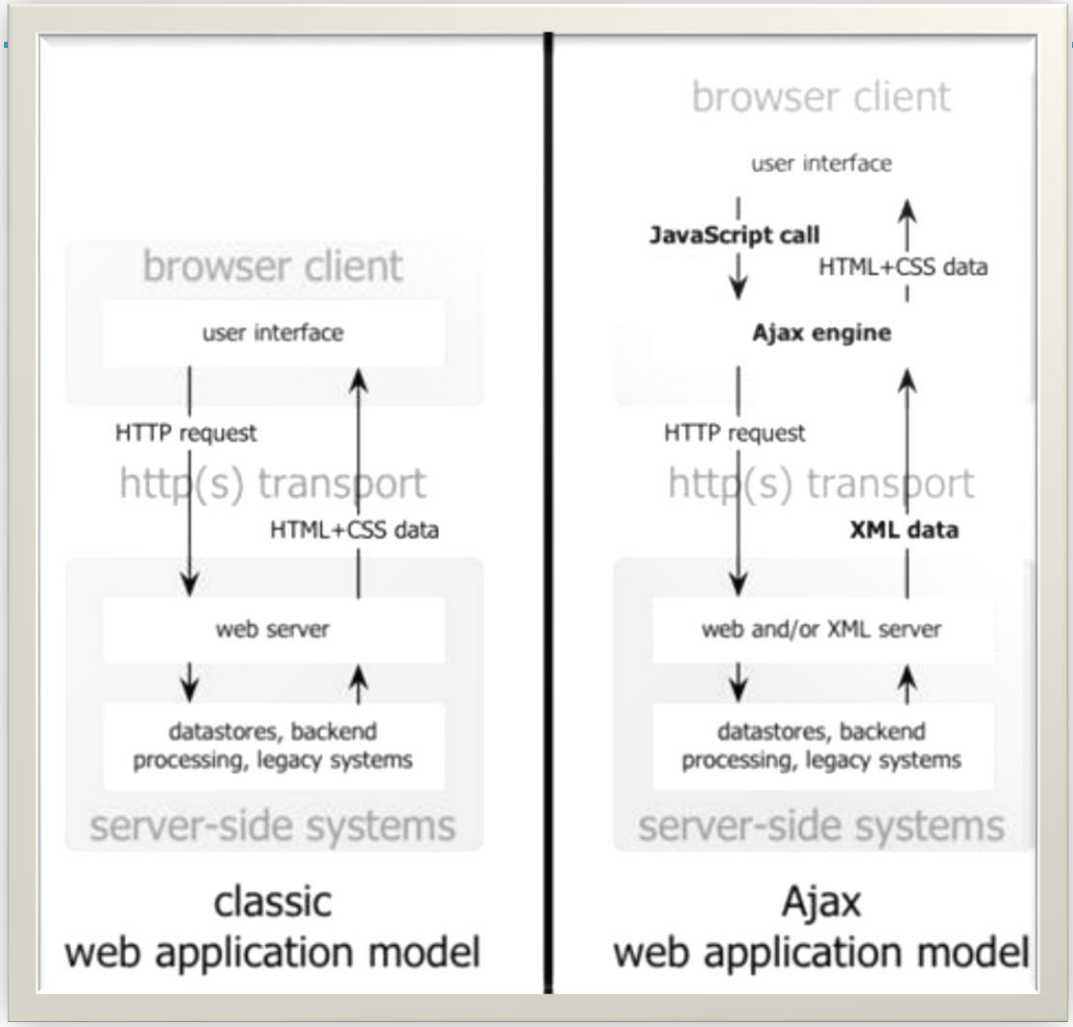
Exemplos de Aplicações da Web 2.0

- **Wikipedia, Facebook, Twitter, LinkedIn, Instagram, buscador Google, Amazon ...**
  - Além de ouras aplicações de blog e outros serviços que permitem a obtenção de dados por RSS/Atom (um XML)

- **Observe que nesses serviços a colaboração é explorada de alguma forma**
  - Permitindo que pessoas interajam
    - **De forma ativa produzindo conteúdo e discutindo com outros**
    - **Não apenas passivos lendo conteúdo de uma minoria que consegue tecnicamente construir e disponibilizar um site**
  - Ou explorando e se tornando indispensável na recuperação de informação (ex: google)

AJAX

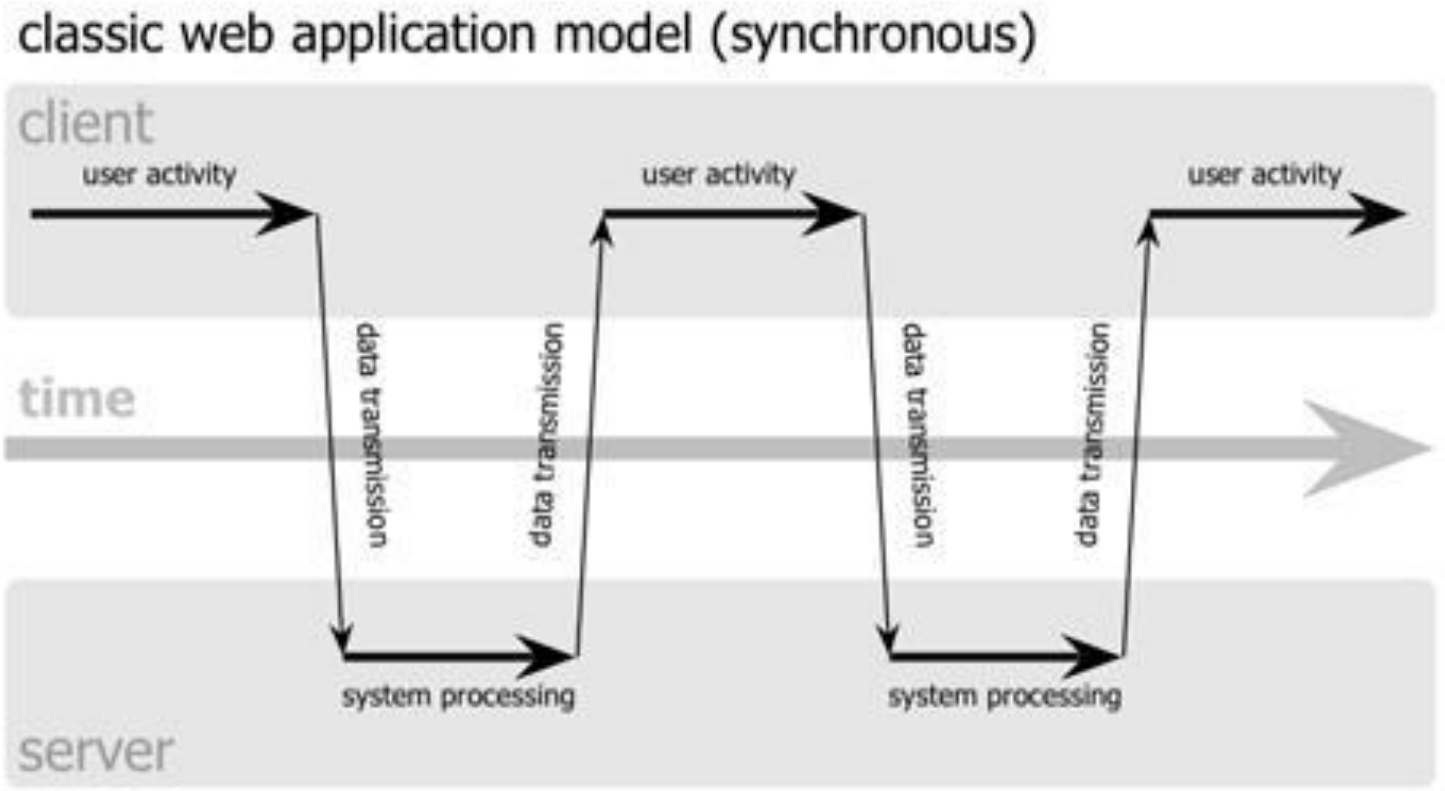
Arquitetura Ajax



AJAX possibilita o uso de comunicação síncrona e assíncrona, diferente da web clássica onde a comunicação é unicamente síncrona.

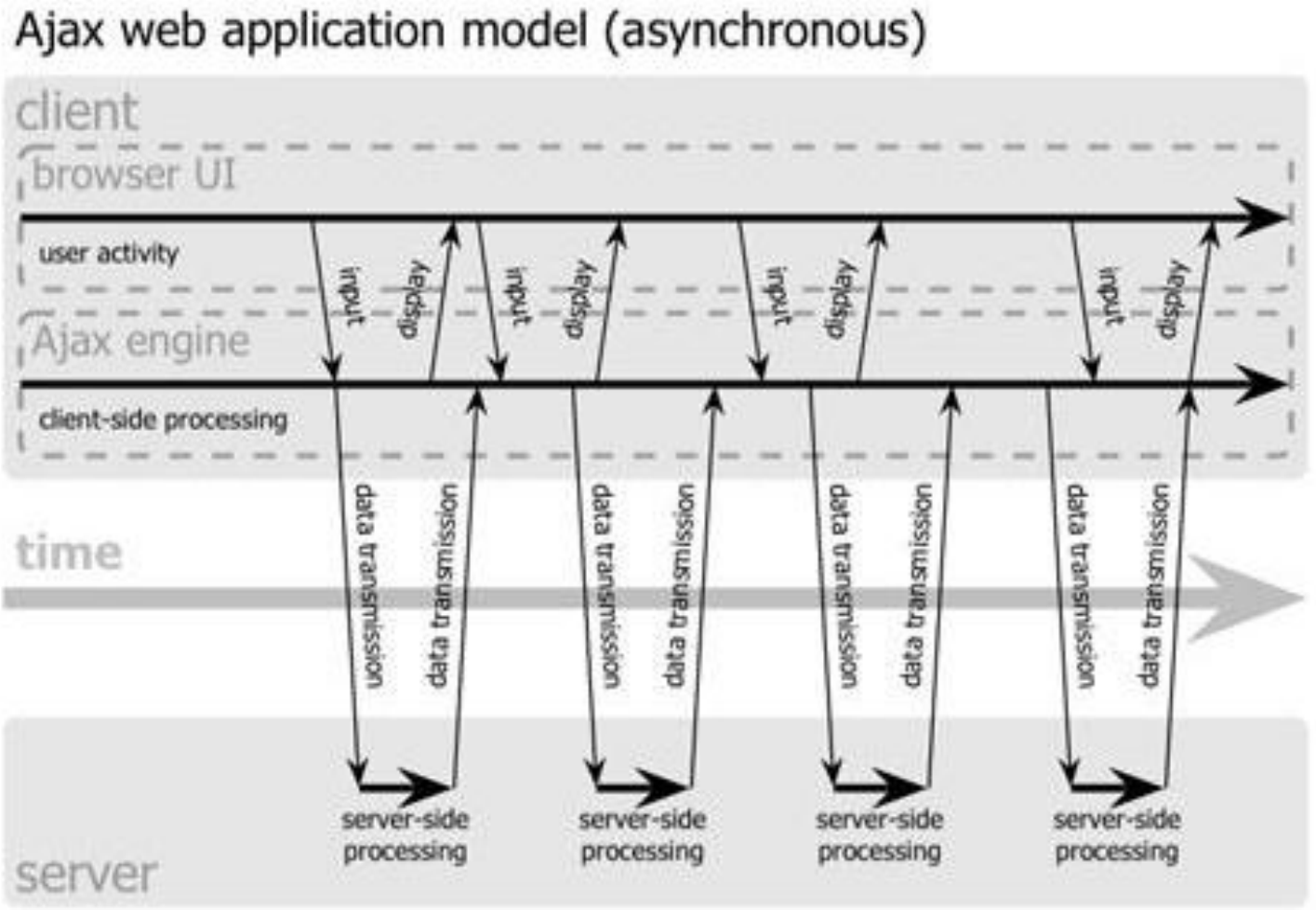
http://www.devmedia.com.br/desmistificando-o-ajax-parte-ii/9499

Ajax – Comunicação síncrona vs assíncrona



http://www.devmedia.com.br/desmistificando-o-ajax-parte-ii/9499

Ajax – Comunicação síncrona vs assíncrona



http://www.devmedia.com.br/desmistificando-o-ajax-parte-ii/9499

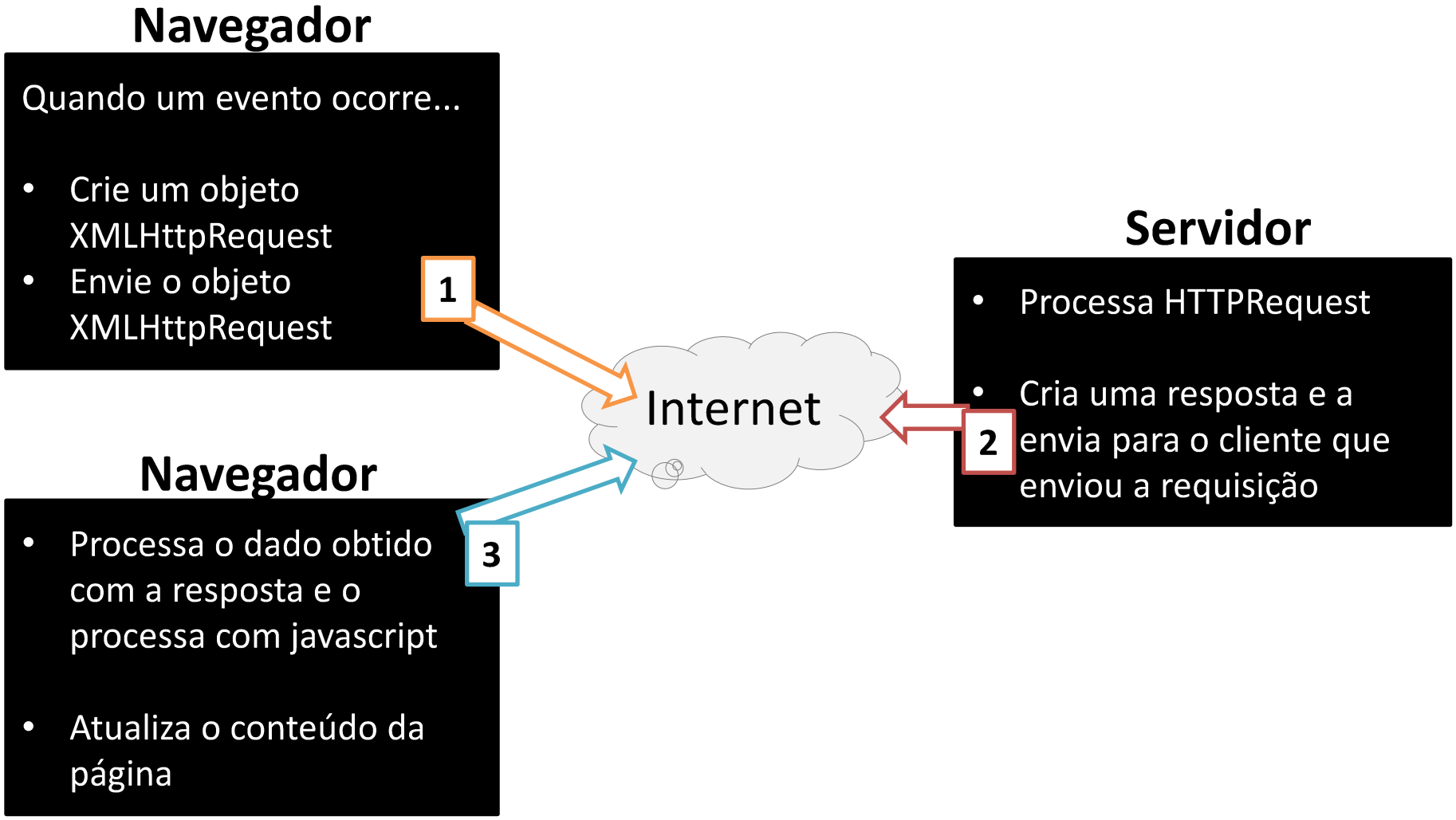
Asynchronous JavaScript and XML

- **Um jeito para usar padrões já existentes**
  - Não é uma nova linguagem de programação
- **Trata de troca de dados entre servidor e cliente**
  - Atualização de partes da página, não da página inteira
    - **Este é o objetivo do AJAX: possibilitar a atualização de partes da página**
- **Crie páginas dinâmicas e rápidas**
  - Atualização assíncrona de páginas web

Exemplos de aplicações que usam AJAX

- **GoogleMaps e GMail**
- **YahooMail**
- **YouTube**
- **Facebook**
- **Etc.**

Funcionamento AJAX



AJAX é Baseado em Padrões da Internet

- **Combina**
  - XMLHttpRequest: troca assíncrona de dados com o servidor
  - JavaScript/DOM: mostrar/interagir com a informação
  - CSS: apresentar o dado com estilo
  - XML: geralmente é a representação usada para os dados transmitidos no nível da aplicação

Exemplo script

```
function loadXMLDoc() {
var xmlhttp;
if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else   {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)  {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
```

Entendendo o Exemplo

- **O código funciona para a página**

```
<!DOCTYPE html>
<html>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change
Content</button>
</body>
</html>
```

- **O código JavaScript pode ser inserido no cabeçalho da página**
  - Funcionalidades dentro da função loadXMLDoc

AJAX

- **Objeto XMLHttpRequest**
  - Principal objeto do AJAX
  - Suportado pelos principais navegadores
    - **IE5 e IE6 usam um objeto equivalente: o ActiveXObject**
  - Usado para troca de dados entre cliente e servidor
    - **Isso pode ocorrer sem a necessidade do envio de requisições que atualizariam toda página HTML**

Criando um objeto XMLHttpRequest

- **Os navegadores (mais modernos) possuem um objeto XMLHttpRequest embutido**

```
//Sintaxe para criação do objeto
xmlhttp = new XMLHttpRequest();
```

- Para versões mais antigas do IE

```
//Sintaxe para criação do objeto para IE5 e IE6
xmlhttp = new ActiveXObject ("Microsoft.XMLHTTP");
```

Enviando uma Requisição para o Servidor

- **Para enviar a requisição, use os métodos *open* e *send***

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

  - *open(method,url,async)*
    - Especifica o método da requisição (ex: GET ou POST), o *path* do recurso desejado e especifica se a requisição deve ser tratada de forma assíncrona ou não (*true* se sim, *false* se não)
  - *send(string)*
    - Envia a requisição para o servidor
    - O parâmetro *string* é usado em requisições POST

Exemplo de Requisições

Usando método GET

```
xmlhttp.open("GET",“aluno.do",true);
xmlhttp.send();
```

- Obs: é possível receber uma resposta em cache

```
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();
```

- Insira um ID único para evitar cache
- Se quiser enviar dados, inclua na URL (*path*, usando URL *encoding* ou replacement)

Exemplo de Requisições

Usando método POST

```
//não será tratada de forma assíncrona
xmlhttp.open("POST",“aluno.do",false);
xmlhttp.send();
```

- Enviando dados
  - **Adicione um cabeçalho HTTP com setRequestHeader()**
    - Assinatura do método: setRequestHeader(header,value)
    - Especifique o cabeçalho e o seu valor
  - **Especifique o dado no método *send***
  - **Veja o exemplo para envio de dados do form**

```
xmlhttp.open("POST",“aluno.do",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Tiago&lname=França");
```

Endereçando um recurso

- **O recurso desejado é passado como parâmetro no método *open***
  - No primeiro exemplo, foi endereçado um arquivo no servidor
    - **Pode apontar para qualquer recurso desejado**
      - O endereço de um servlet ou de uma função PHP, por exemplo
      - Pode ser qualquer recurso (txt, xml, json, php, servlet, etc.)

Comunicação Assíncrona

- **Permite**
  - Que tarefas sejam realizadas no servidor enquanto a página continua provendo outras funcionalidades para um usuário
- **Ferramenta para desenvolvedores**
  - Uma tarefa demorada pode ser iniciada no servidor sem bloquear o uso de uma aplicação
    - **Melhor para o usuário que não precisa ficar parado esperando o servidor**
      - Execute outro procedimento com o JavaScript
    - **Trata da resposta quando ela estiver disponível**

Comunicação Assíncrona

- **Se a comunicação não for assíncrona, parâmetro relativo a comunicação deve ser *false***
  - Não é recomendado, mas pode ser útil em algumas situações

```
xmlhttp.open("GET","ajax_info.txt",false);
xmlhttp.send();
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

  - **Observe: não foi usada a função *onreadystatechange***
    - E não deve ser usada
  - ***O* código foi inserido logo após o *send***

Respostas do Servidor

- **Para capturar a resposta se usa o *responseText* ou o *responseXML***
  - *responseText* – resposta é pega como uma string
    - **Use quando a resposta não for um XML**

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

  - *responseXML* – resposta é pega como um dado XML
    - **É possível parsear a resposta**

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)  {
  txt=txt + x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("myDiv").innerHTML=txt;
```

Eventos

- **Evento *onreadystatechange***
  - Esse evento é disparado assim que o estado do *readyState* muda
    - ***readyState* contém o estado do *XMLHttpRequest* o qual tem 3 propriedades importantes**
      - *onreadystatechange:* armazena uma função ou nome de uma função que é chamada quando *readyState* muda
      - *readyState:* muda de 0 a 4 (requisição não inicializada, conexão com o servidor estabelecida, requisição recebida, processando a requisição e requisição finalizada e resposta disponível, respectivamente)
      - *status*: 200 “OK” ou 400 “*Page not found*”

Função de *Callback*

- **Uma função passada como parâmetro para outra**
  - Se uma página possui mais de uma tarefa AJAX, pode ser útil criar uma função padrão para criar XMLHttpRequest
    - **Crie-a e chame-a em todos os pontos necessários da página**

```
function funcaoPadrao() {
loadXMLDoc(“teste.json",function()  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
    ocument.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  });
}
```

Exercício

1. **Crie uma página usando AJAX para carregar seus dados (nome completo, CPF e matrícula)**
   - Crie um serviço no lado do servidor usando servlet (Web2) para pegar os dados pre-cadastrados em um banco
     - **Ao pegar os dados, faça a thread parar por 30 segundos**
   - Apresente o resultado da consulta para o usuário