1. Ajax

AJAX é o acrônimo de Asynchronous Javascript and XML, que em português significa "Javascript e XML assíncronos". Entendendo melhor o significado, seria a chamada de um recurso no servidor a partir de um código Javascript no navegador web, de forma que o resultado **atualize apenas uma parte da página sem precisar fazer uma atualização dela inteira**. Esta chamada é assíncrona, ou seja, o script que a chamou continua sua execução sem esperar pela resposta. Quando o servidor responde, uma função Javascript especificada trata corretamente os dados retornados, fazendo a atualização de parte da tela apenas.

O termo Ajax pode ser referido ao uso conjunto de tecnologias que já existiam há muito tempo, como **Javascript**, DOM, CSS, XML, etc., para a criação de interfaces mais dinâmicas e responsivas. Portanto, quando você usa AJAX, você está utilizando uma abordagem de programação para a web focada em proporcionar uma melhor interação para o usuário. Não é uma ferramenta ou framework que pode ser baixado de algum site, pois os componentes de software que são fundamentais para esta abordagem estão presentes em todos os navegadores web: HTML[1], XHTML[2] e XML[3]; CSS[4]; DOM[5]; Javascript[6]; e, o principal, um componente chamado XMLHttpRequest.

---

[1] HTML - HyperText Markup Language ou Linguagem de Marcação de Hipertexto), que é uma linguagem formada por tags, ou marcadores, que são interpretadas pelos navegadores.
[2] XHTML - **eXtensible Hypertext Markup Language** (abreviado **XHTML**) é uma reformulação da linguagem de marcação HTML, combinada com as regras da linguagem de marcação XML (tags), uma recomendação do W3C de janeiro de 2000, sucessora do HTML 4.01, objetivando a melhoria da exibição das páginas Web em diversos dispositivos (televisão, palm, celular, etc), além da melhoraria da acessibilidade do conteúdo.
[3] XML - sigla para eXtensible Markup Language, é um tipo de linguagem de marcação que define regras para codificar diferentes documentos.
[4] CSS - Cascading Style Sheets é um mecanismo para adicionar estilo a um documento web. O código CSS pode ser aplicado diretamente nas tags ou ficar contido dentro das tags <style>
[5] DOM - Abreviação para Document Object Model (Modelo de Objeto de Documentos), representa uma página web como uma hierarquia ou estrutura de árvore, onde cada elemento da página, tais como imagens, botões, tabelas e textos, são representados como nós e renderizados pelo navegador web.
[6] Javascript - É uma linguagem de programação que permite a você criar conteúdo que se atualiza dinamicamente, controlar múltimídias, imagens animadas etc.

Antes de nos aprofundarmos no Ajax, é importante distinguir claramente o que são scripts síncronos e assíncronos. Na computação, quando **uma tarefa precisa aguardar a tarefa anterior ser concluída para iniciar sua execução, dizemos que estas tarefas são síncronas.** Por outro lado, vamos supor agora que **duas ou mais tarefas precisam ser executadas e que não há nenhuma dependência entre elas**. Ganharíamos tempo se elas fossem executadas ao mesmo tempo, contanto que o instante de conclusão de cada uma não importasse nem influenciasse a outra. Neste caso, **dizemos que as tarefas foram executadas de forma assíncrona.** Tais conceitos são fundamentais para o entendimento de desenvolvimento de aplicações mais dinâmicas e interativas.

1.1. Exemplo básico do uso do Ajax

Nesse tópico será explorado uma aplicação WEB que exibirá na tela um campo de pesquisa, que ao ser digitado um nome, exibirá os dados cadastrais da pessoa pesquisada **sem rescrever toda a tela.** Para implementar o exemplo será construído quatro arquivos, são eles:

- index.html (Quadro 1)
  - Onde será a página de interação com o usuário;
- ajax.js (Quadro 2)
  - Onde terá o script de toda a lógica para as requisições;
- contato.php (Quadro 3)
  - Onde será executada as consultas.
- conn.php (Quadro 4)
  - Onde será realizada a conexão com o Banco de Dados, chamado pelo arquivo contato.php

Um quinto arquivo, por exemplo, estilo.css, para personalização da página pode ser criado, ficando como **opcional**.

Em relação ao banco de dados, deverá ser criado um banco com nome bdajax e criado uma tabela chamada pessoa com a estrutura apresentada no Quadro 5. Lembre-se que esses nomes são apenas sugeridos, ficando a critério do leitor qualquer outro nome. Nos códigos mostrado nos quadros, as referências as chamadas aos arquivos ou funções estão em negrito para ajudar o leitor no encadeamento das chamadas dos arquivos.

O arquivo principal, isto é, o primeiro a ser executado é o index.html. Nesse arquivo, o evento onClick do botão chama a função getDados() que está escrita no arquivo Ajax.js. A função getDados() pega os dados do formulário contido no index.html e cria uma requisição do tipo ajax no servidor chamada de CriaRequest(). Uma vez criada a requisição, está por sua vez é iniciada através do comando **xmlreq.open("GET", "contato.php?txtnome=" + nome, true),** no qual o arquivo contato.php é executado no servidor de forma assíncrona, atualizando assim, apenas parte da página.

Quadro 1

Index.html

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
</head>
<body>
<script type="text/javascript" src="ajax.js"></script>
<div id="Container">
        <h1>Agenda de Contatos utilizando AJAX</h1>
        <hr/>
        <h2>Pesquisar Contato:</h2>
        <div id="Pesquisar">
          Infome o nome: <input type="text" name="txtnome" id="txtnome"/>
          <input type="button" name="btnPesquisar" value="Pesquisar" onclick="getDados();"/>
        </div>
        <hr/>
        <h2>Resultados da pesquisa:</h2>
        <div id="Resultado"></div>
        <hr>
</div>
</body>
</html>
```

Quadro 2

Ajax.js

```
// JavaScript Document
/** * Função para criar um objeto XMLHTTPRequest */
function CriaRequest()
{
	try
	{
		request = new XMLHttpRequest();
	}
	catch (IEAtual)
	{
		try
		{
			request = new ActiveXObject("Msxml2.XMLHTTP");
		}
		catch(IEAntigo)
		{
			try
			{
				request = new ActiveXObject("Microsoft.XMLHTTP");
			}
			catch(falha)
			{
				request = false;
			}
		}
	}
	if (!request)
		alert("Seu Navegador não suporta Ajax!");
	else
		return request;
}
/** * Função para enviar os dados */
function getDados()
{
	// Declaração de Variáveis
	var nome = document.getElementById("txtnome").value;
	var result = document.getElementById("Resultado");
	// recebe um objeto XMLHTTPRequest retornado pela função CriaRequest()
	var xmlreq = CriaRequest();

	// Exibi a imagem de progresso
	//Essa animação será carregada no início da requisição, uma espécie de progresso
	//enquanto usuário aguarda o retorno da pesquisa.

	result.innerHTML = '<img src="Progresso1.gif"/>';

	// Iniciar uma requisição
	xmlreq.open("GET", "contato.php?txtnome=" + nome, true);

	// Atribui uma função para ser executada sempre que houver uma mudança de ado
	xmlreq.onreadystatechange = function()
	{
```

```
                    // Verifica se foi concluído com sucesso e a conexão fechada (readyState=4)
                    if (xmlreq.readyState == 4)
                    {
                            // Verifica se o arquivo foi encontrado com sucesso
                            if (xmlreq.status == 200)
                            {
                                    result.innerHTML = xmlreq.responseText;
                            }
                            else
                            {
                                    result.innerHTML = "Erro: " + xmlreq.statusText;
                            }
                    }
            };
            xmlreq.send(null);
}
```

Para trabalhar com requisições AJAX temos que criar um objeto do tipo XMLHTTPRequest, mas esse objeto não é padrão para todos os navegadores. A função "CriaRequest()" executa uma série de verificações sobre o tipo de navegador e retorna um objeto XMLHTTPRequest. Podem existir navegadores que não suportem AJAX então nesse caso será emitida uma mensagem avisando o usuário desse problema.

Na função getDados() é onde ocorre toda a mágica do AJAX, basicamente essa função trabalha com requisições e respostas, nada muito diferente do conceito do desenvolvimento WEB, a diferença é que nesse caso não será necessária uma nova "carga" na página para carregar as respostas, aquele famoso refresh. Nela são declaradas três variáveis:

- **Nome** : Recebe o conteúdo de textnome
- **Result** : que recebe o local da página onde será carregada a resposta da requisição
- **xmlreq** : Recebe um objeto XMLHTTPRequest retornado pela função CriaRequest()

Uma vez instanciado o objeto XMLHTTPResquest temos que abrir uma requisição chamando o método open(). Esse método requer 3 parâmetros:

- Primeiro argumento define qual método de envio deverá ser utilizado (GET ou POST).

- Segundo argumento especifica a URL do script no servidor.
- Terceiro argumento especifica que a requisição deverá ser assíncrona.

O método send() envia a requisição para o servidor. A requisição pode retornar dois tipos de resposta, responseText e responseXml. Depois fica monitorando a propriedade readyState através do evento onreadystatechange, até o seu valor ser igual a 4 (requisição finalizada) e o status igual a 200 ("OK"), se essas duas condições forem atendidas é só carregar o conteúdo da resposta no local previamente definido na variável result, senão carrega a mensagem de erro.

A função "CriaRequest()" descrita no arquivo ajax.js é quase que padrão, pois ali conseguimos prever a maioria das situações que podem ocorrer em quase todos os browsers atualmente usados.

Uma aplicação Ajax transfere-se muito do processamento do servidor para o cliente, essa mudança tem custos porque estaremos a delegar ao cliente a responsabilidade de realizar determinadas operações para as quais não estaria inicialmente destinado.

Mas no geral sabendo usar o AJAX ele pode trazer muito mais vantagens que desvantagens. A figura 1 demonstra com o funciona uma requisição normal e outra usando o Ajax.

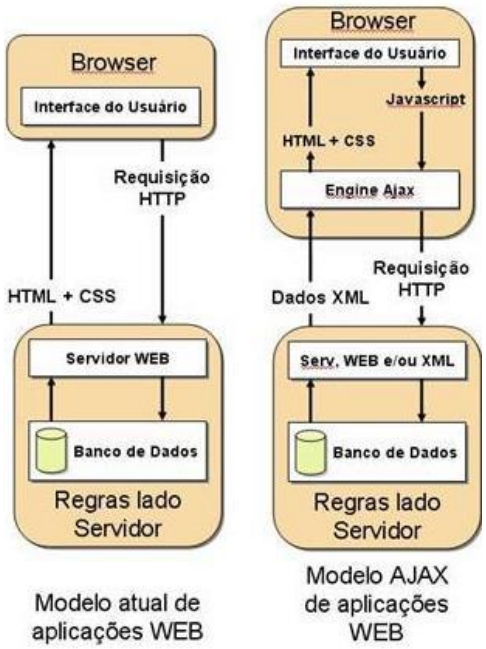

Figura 1 – Uso do Ajax

Quadro 3

contato.php

```
<?php
if (isset($_GET["txtnome"]))   // Verifica se existe a variável txtnome
{
        require_once('conn.php');

        $nome = $_GET["txtnome"];
        // Verifica se a variável está vazia
        if (empty($nome))
                $sql = "SELECT * FROM pessoa";
        else
        {
                $nome .= "%";
                $sql = "SELECT * FROM pessoa WHERE nome like '$nome'";
        }
        sleep(2); // Congela por 2 segundos a execução e exibe a animação (Progresso1.gif)
        $stmt = $pdo->prepare($sql);
        $stmt->execute();
        $pessoa_list = $stmt->fetchAll(PDO::FETCH_ASSOC);

        // Atribui o código HTML para montar uma tabela
        $tabela = "<table border='1'> <thead> <tr> <th>NOME</th> <th>TELEFONE</th>
        <th>CELULAR</th> <th>EMAIL</th> </tr> </thead> <tbody> <tr>";
         $return = "$tabela";
        foreach ($pessoa_list as $row => $pessoa)
         {
          $return.= "<td>" . utf8_encode($pessoa["nome"]) . "</td>";
          $return.= "<td>" . utf8_encode($pessoa["telcel"]) . "</td>";
          $return.= "<td>" . utf8_encode($pessoa["telres"]) . "</td>";
          $return.= "<td>" . utf8_encode($pessoa["email"]) . "</td>";
          $return.= "</tr>";
        }
        echo $return."</tbody></table>";
}
?>
```

Quadro 4

conn.php

```
<?php
$host = 'localhost';
$usuario = 'root';
$senha = '';
$banco = 'bdajax';
$dsn = "mysql:host={$host};port=3306;dbname={$banco}";
try
{
   // Conectando
   $pdo = new PDO($dsn, $usuario, $senha, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET
          NAMES utf8"));
}
catch (PDOException $e)
{
      // Se ocorrer algum erro na conexão
      die($e->getMessage());
}
?>
```

Quadro 5 – Estrutura da tabela pessoa do banco de dados bdajax

```
CREATE TABLE IF NOT EXISTS pessoa (
          nome varchar(50) NOT NULL,
          cpf char(11) NOT NULL,
          end varchar(50) NOT NULL,
          num int(11) NOT NULL,
          telcel char(15) NOT NULL,
          telres char(15) NOT NULL,
          email varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

1.1.1. Interface XMLHttpRequest

A Interface XMLHttpRequest possui um **EventListener** denominado "**onreadystatechange**", que é um manipulador de eventos que é invocado quando o evento readystatechange é acionado. Passamos uma função de tratamento a esse EventListener que verificará os estados da conexão.

A Interface possui uma propriedade denominada "**readyState**" que recebe o valor de uma constante que representa o estado atual da conexão. As constantes que representam os estados são definidos como:

- **UNSENT** = 0; //Nenhuma comunicação feita.
- **OPEN** = 1; //Documento encontrado e sendo carregado

- **SENT** = 2; //Documento carregado (enviado)
- **LOADING** = 3; //Documento XMLHTTPRequest interagindo
- **DONE = 4**; //realizada

Já a propriedade "**status**" da Interface, representa o código de status do protocolo HTTP enviado pelo servidor, como o código "200" para uma requisição com sucesso. Caso o recurso não esteja disponível, é lançado uma exceção INVALID_STATE_ERR pelo Browser. Informações sobre mais códigos procure site https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

As propriedades "**responseText**" e "**responseXML**", devolve um texto puro no caso do responseText ou um xml parseado no caso do responseXML.

A interface XMLHttpRequest também possui os seguintes métodos:

- **Abort:** Cancela a requisição.
- **Open**: Os primeiro parâmetro representa o método HTTP de requisição, que pode ser dos tipos: POST, GET. O segundo parâmetro é a url que receberá a requisição, o terceiro, quarto e quinto (Async, User e Psw) são opcionais.
- **setRequestHeader:** usado antes do método "**send**" e habilita enviar informações de cabeçalho na requisição quando for disparado os recursos ao servidor.
- **Send:** usado para enviar a requisição com o argumento opcional "data" que é do tipo "Text" ou "Document".
- **Get**: Quando o método de requisição escolhido é o **GET**, os argumentos a serem enviados ao servidor é usado no parâmetro "url", como "/pagina.php?id=1&nome=Claudio".
- **Post**: Já no método **POST**, os dados serão enviados como argumento do método "send".