**AJAX**

AJAX é o sonho de um desenvolvedor, porque você pode:

* Atualizar uma página da Web sem recarregar a página
* Solicitar dados de um servidor - depois que a página foi carregada
* Receber dados de um servidor - depois que a página foi carregada
* Enviar dados para um servidor - em segundo plano

O que é AJAX?

AJAX = JavaScript assíncrono e XML. AJAX não é uma linguagem de programação. AJAX apenas usa uma combinação de:

- Um objeto built-in do XMLHttpRequest do browser (para pedir dados de um web server) - JavaScript e HTML DOM (para exibir ou usar os dados) - AJAX é um nome enganoso. Aplicativos AJAX podem usar XML para transportar dados, mas é igualmente comum para transportar dados como texto sem formatação ou texto JSON.

O AJAX permite que as páginas da Web sejam atualizadas de forma assíncrona, trocando dados com um servidor web nos bastidores. Isso significa que é possível atualizar partes de uma página da Web, sem recarregar a página inteira.

1. Um evento ocorre em uma página da Web (a página é carregada, clique em botão)

2. Um objeto XMLHttpRequest é criado pelo JavaScript

3. O objeto XMLHttpRequest envia uma solicitação para um servidor web

4. O servidor processa o pedido

5. O servidor envia uma resposta para a página da Web

6. A resposta é lida por JavaScript

7. A ação correta (como a atualização da página) é realizada pelo JavaScript

A chave do AJAX é o objeto XMLHttpRequest. O objeto XMLHttpRequest é usado para trocar dados com um servidor nos bastidores.

Criar um objeto XMLHttpRequest

Todos os navegadores modernos (Chrome, IE7 +, Firefox, Safari e Opera) possuem um objeto XMLHttpRequest embutido. Sintaxe para criar um objeto XMLHttpRequest:

Variable = new XMLHttpRequest ();

Versões antigas do Internet Explorer (IE5 e IE6) utilizam um objecto ActiveX:

Variable = new ActiveXObject ("Microsoft.XMLHTTP");

Para lidar com todos os navegadores, incluindo IE5 e IE6, verifique se o navegador suporta o objeto XMLHttpRequest. Se isso acontecer, crie um objeto XMLHttpRequest, se não, crie um ActiveXObject: var xhttp;  
 if (window.XMLHttpRequest) {  
    xhttp = new XMLHttpRequest();  
    } else {  
     // code for IE6, IE5  
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");  
}

Acesso em domínios

Por razões de segurança, os navegadores modernos não permitem o acesso entre domínios. Isso significa que tanto a página da Web como o arquivo XML que ele tenta carregar, devem estar localizados no mesmo servidor.

Se você quiser usar em uma de suas próprias páginas da web, os arquivos XML que você carrega devem estar localizados em seu próprio servidor.

**Métodos de objeto XMLHttpRequest**

|  |  |
| --- | --- |
| Método | Descrição |
| new XMLHttpRequest() | Cria um novo objeto XMLHttpRequest |
| abort() | Cancela o pedido atual |
| getAllResponseHeaders() | Retorna informações de cabeçalho |
| getResponseHeader() | Retorna informações de cabeçalho específicas |
| open(method, url, async, user, psw) | Especifica a solicitação  Método: o tipo de solicitação GET ou POST  Url: o local do arquivo  Async: true (assíncrono) ou false (síncrono)  Usuário: nome de usuário opcional  Psw: senha opcional |
| send() | Envia o pedido para o servidor  Usado para solicitações GET |
| send(string) | Envia a solicitação para o servidor.  Usado para pedidos POST |
| setRequestHeader() | Adiciona um par label / value ao cabeçalho a ser enviado |

**Propriedades do objeto XMLHttpRequest**

|  |  |
| --- | --- |
| Propriedade | Descrição |
| onreadystatechange | Define uma função a ser chamada quando a propriedade readyState for alterada |
| readyState | Mantém o status do XMLHttpRequest.  0: pedido não inicializado  1: conexão do servidor estabelecida  2: pedido recebido  3: solicitação de processamento  4: pedido terminado e resposta está pronta |
| responseText | Retorna os dados de resposta como uma string |
| responseXML | Retorna os dados de resposta como dados XML |
| status | Retorna o status-número de um pedido  200: "OK"  403: "Forbidden"  404: "Page not found" |
| statusText | Retorna o status do texto (por exemplo, "OK" ou "Não encontrado") |

**Enviar uma solicitação para um servidor**

Para enviar uma solicitação a um servidor, usamos os métodos open () e send () do objeto XMLHttpRequest: xhttp.open("GET", "ajax\_info.txt", true);  
 xhttp.send();

|  |  |
| --- | --- |
| Método | Descrição |
| open(*method, url, async*) | Especifica o tipo de solicitação  Método: o tipo de pedido: GET ou POST  Url: o local do servidor (arquivo)  Async: true (assíncrono) ou false (síncrono) |
| send() | Envia o pedido para o servidor (usado para GET) |
| send(*string*) | Envia o pedido para o servidor (usado para POST) |

GET ou POST?

GET é mais simples e mais rápido do que POST, e pode ser usado na maioria dos casos. No entanto, sempre use solicitações POST quando:

* Um arquivo em cache não é uma opção (atualizar um arquivo ou banco de dados no servidor).
* Enviando uma grande quantidade de dados para o servidor (POST não tem limitações de tamanho).
* Enviando entrada do usuário (que pode conter caracteres desconhecidos), POST é mais robusto e seguro do que GET.

Uma simples solicitação GET: xhttp.open("GET", "demo\_get.asp", true);  
 xhttp.send();

Uma simples solicitação POST: xhttp.open("POST", "demo\_post.asp", true);  
 xhttp.send();

Se você deseja enviar informações com o método GET, adicione as informações ao URL: xhttp.open("GET", "demo\_get2.asp?fname=Henry&lname=Ford", true);  
xhttp.send();

Para dados POST como um formulário HTML, adicione um cabeçalho HTTP com setRequestHeader (). Especifique os dados que você deseja enviar no método send(): xhttp.open("POST", "ajax\_test.asp", true);  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");

|  |  |
| --- | --- |
| Método | Descrição |
| setRequestHeader(*header, value*) | Adiciona cabeçalhos HTTP ao pedido  Header: especifica o nome do cabeçalho  Valor: especifica o valor do cabeçalho |

O URL - um arquivo em um servidor

O parâmetro url do método open() é um endereço para um arquivo em um servidor: xhttp.open("GET", "ajax\_test.asp", true);

O arquivo pode ser qualquer tipo de arquivo, como .txt e .xml, ou arquivos de scripts de servidor como .asp e .php (que podem executar ações no servidor antes de enviar a resposta de volta).

Assíncrono - Verdadeiro ou Falso?

Para enviar a solicitação de forma assíncrona, o parâmetro async do método open() deve ser definido como true: xhttp.open("GET", "ajax\_test.asp", true);

O envio de pedidos assíncronos é uma grande melhoria para os desenvolvedores web. Muitas das tarefas executadas no servidor são muito demoradas. Antes de AJAX, esta operação poderia causar o aplicativo para travar ou parar.

Ao enviar de forma assíncrona, o JavaScript não precisa aguardar a resposta do servidor, mas sim: Executar outros scripts enquanto aguarda a resposta do servidor . Lidar com a resposta quando a resposta estiver pronta

Async = true

Ao usar async = true, especifique uma função a ser executada quando a resposta estiver pronta no evento onreadystatechange: xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML = this.responseText;  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

Async = false

Para usar async = false, altere o terceiro parâmetro no método open () para false: xhttp.open("GET", "ajax\_info.txt", false);

Usando async = false não é recomendado, mas para algumas pequenas solicitações isso pode ser ok. Lembre-se de que o JavaScript NÃO continuará a ser executado, até que a resposta do servidor esteja pronta. Se o servidor estiver ocupado ou lento, o aplicativo travará ou parará. Nota: Quando você usa async = false, NÃO escreva uma função onreadystatechange - basta colocar o código após a instrução send() : xhttp.open("GET", "ajax\_info.txt", false);  
xhttp.send();  
document.getElementById("demo").innerHTML = xhttp.responseText;

**AJAX - Resposta do Servidor**

A propriedade onreadystatechange

A propriedade readyState contém o status do XMLHttpRequest. A propriedade onreadystatechange define uma função a ser executada quando o readyState é alterado. A propriedade status ea propriedade statusText contém o status do objeto XMLHttpRequest.

|  |  |
| --- | --- |
| Método | Descrição |
| onreadystatechange | Define uma função a ser chamada quando a propriedade readyState for alterada |
| readyState | Mantém o status do XMLHttpRequest.  0: pedido não inicializado  1: conexão do servidor estabelecida  2: pedido recebido  3: solicitação de processamento  4: pedido terminado e resposta está pronta |
| Status | 200: "OK"  403: "Forbidden"  404: "Page not found" |
| statusText | Retorna o status do texto (por exemplo, "OK" ou "Não encontrado") |

A função onreadystatechange é chamada toda vez que o readyState é alterado. Quando readyState é 4 e o status é 200, a resposta está pronta.

Usando uma função de retorno de chamada

Uma função callback é uma função passada como um parâmetro para outra função. Se você tiver mais de uma tarefa AJAX em um site, você deve criar uma função para executar o objeto XMLHttpRequest e uma função callback para cada tarefa AJAX. A chamada de função deve conter o URL ea função a ser chamada quando a resposta estiver pronta.

**Propriedades de Resposta do Servidor**

|  |  |
| --- | --- |
| Método | Descrição |
| responseText | Obter os dados de resposta como uma string |
| responseXML | Obter os dados de resposta como dados XML |

**Métodos de resposta do servidor**

|  |  |
| --- | --- |
| Método | Descrição |
| getResponseHeader() | Retorna informações de cabeçalho específicas do recurso de servidor |
| getAllResponseHeaders() | Retorna todas as informações de cabeçalho do recurso de servidor |

A propriedade responseText

A propriedade responseText retorna a resposta do servidor como uma seqüência de caracteres JavaScript e você pode usá-la de acordo: document.getElementById("demo").innerHTML = xhttp.responseText;

A propriedade responseXML

O objeto XML HttpRequest tem um analisador XML embutido. A propriedade responseXML retorna a resposta do servidor como um objeto XML DOM. Usando esta propriedade, você pode analisar a resposta como um objeto XML DOM.

O método getAllResponseHeaders()

O método getAllResponseHeaders() retorna todas as informações de cabeçalho da resposta do servidor.

O método getResponseHeader()

O método getResponseHeader() retorna informações de cabeçalho específicas da resposta do servidor.