AJAX (Web)

Origem: Wikipédia, a enciclopédia livre.

AJAX (significa **Asyncronous Javascript And XML**) é o uso sistemático de Javascript e XML (e derivados) para tornar o navegador mais interativo com o usuário, utilizando-se de solicitações assíncronas de informações. AJAX não é somente um novo modelo, é também uma iniciativa na construção de aplicações web mais dinâmicas e criativas. Então, AJAX não é uma tecnologia, são realmente várias tecnologias trabalhando juntas, cada uma fazendo sua parte, oferecendo novos meios poderosos. AJAX incorpora em seu modelo:

- Apresentação baseada em padrões, usando XHTML e CSS;
- Exposição e interação dinâmica usando o DOM;
- Intercâmbio e manipulação de dados usando XML e XSLT;
- Recuperação assíncrona de dados usando o objeto XMLHttpRequest;
- e JavaScript unindo todas elas em conjunto.

O modelo clássico de aplicação web trabalha assim: A maioria das ações do usuário na interface dispara uma solicitação HTTP para o servidor web. O servidor processa algo — recuperando dados, mastigando números, conversando com vários sistemas legados — e então retorna uma página HTML para o cliente. É um modelo adaptado do uso original da Web como um agente de hipertexto, porém o que faz a Web boa para hipertexto não necessariamente faz ela boa para aplicações de software.

Esta aproximação possui muito dos sentidos técnicos, mas não faz tudo que um usuário experiente poderia fazer. Enquanto o servidor está fazendo seu trabalho, o que o usuário estará fazendo? O que é certo, esperando. E a cada etapa em uma tarefa, o usuário aguarda mais uma vez.

Obviamente, se nós estivéssemos projetando a Web do nada para aplicações, não faríamos com que os usuários esperassem em vão. Uma vez que a interface está carregada, por que a interação do usuário deveria parar a cada vez que a aplicação precisasse de algo do servidor? Na realidade, por que o usuário deveria ver a aplicação ir ao servidor toda vez?

A maior vantagem das aplicações **AJAX** é que elas rodam no próprio navegador web. Então, para estar hábil a executar aplicações AJAX, bastar possuir algum dos navegadores modernos, ou seja, lançados após 2001. São eles: Mozilla Firefox, Internet Explorer 5+, Opera, Konqueror e Safari.

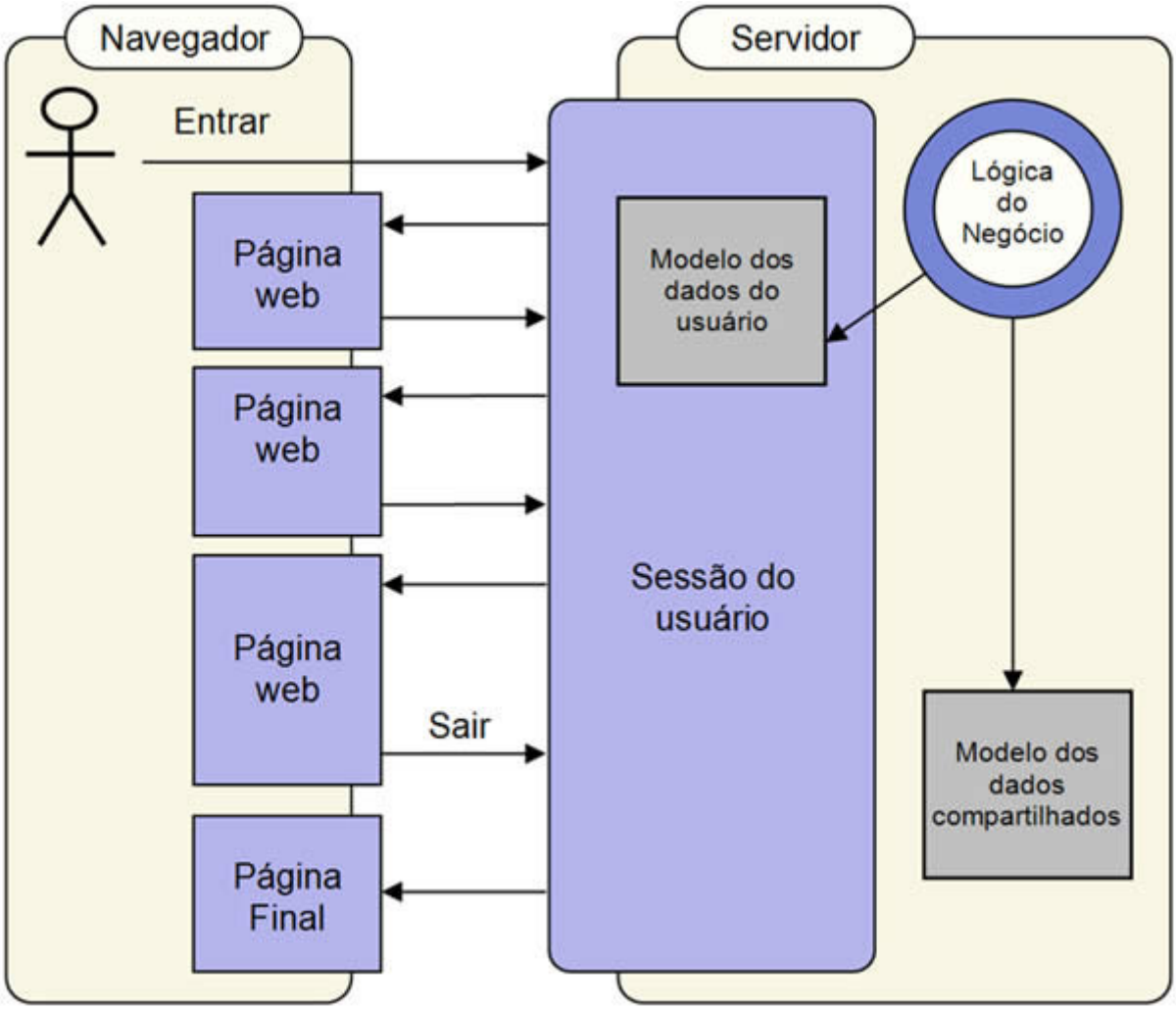
Este texto foi traduzido de Ajax: A New Approach to Web Applications (http://adaptivepath.com/publications/essays/archives/000385.php).

==Os quatro princípios de Ajax== O modelo clássico de aplicação baseado em páginas é amarrado em muitas das estruturas que nós usamos, e também em nossas maneiras de pensar. Vamos fazer uma análise de alguns minutos para descobrir o que são estas suposições essenciais e como necessitamos repensar estas idéias para entendermos Ajax suficientemente.

Tabela de conteúdo

O navegador hospeda uma aplicação, e não conteúdo

Em uma aplicação web clássica baseada em páginas, o navegador é efetivamente um terminal bobo. Ele não sabe nada sobre o que o usuário está realmente realizando em suas ações conseqüentes. Todas essas informações são retidas no servidor web, tipicamente na sessão do usuário. Sessões de usuários no lado servidor são comuns atualmente. Se você está trabalhando em Java ou .NET, a sessão no **lado servidor faz parte da API padrão, assim como controle de solicitações, respostas, e tipos de conteúdo (MIME). A figura abaixo** ilustra o ciclo de vida típico de uma aplicação web clássica.



Quando o usuário entra ou de outra maneira inicia uma sessão, vários objetos são criados no lado servidor, representando, por exemplo, a cesta de compras e as credenciais de cliente, isso em um site de comércio eletrônico. Ao mesmo tempo, a página inicial é servida ao navegador, em um fluxo de marcações HTML que mistura um anúncio de apresentação padrão e dados específicos do usuário juntos com o conteúdo, como por exemplo, uma lista de itens exibidos recentemente.

Toda vez que o usuário interage com o site, um outro documento é enviado para o navegador, contendo a mesma mistura de cabeçalhos e dados. O navegador obedientemente retira o documento anterior e exibe o novo, porque ele é bobo e não sabe que o outro documento produz um resultado muito semelhante.

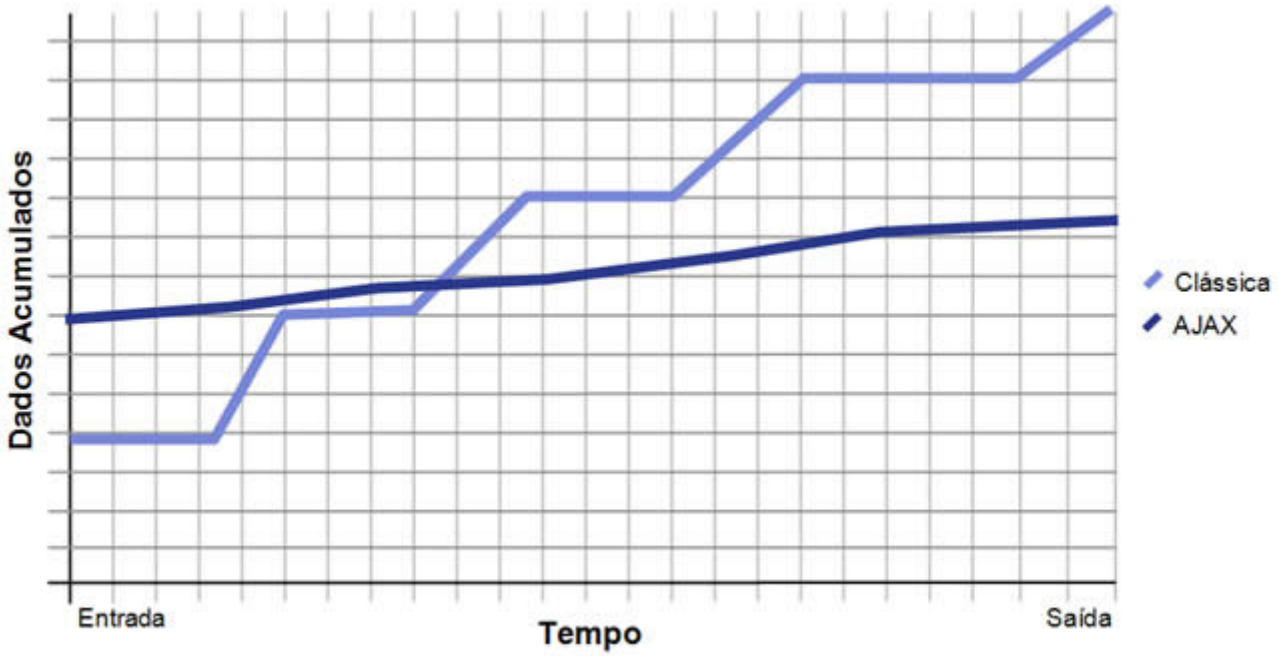
Quando o usuário efetua a saída ou fecha o navegador, a aplicação sai e a sessão é destruída. Qualquer informação que o usuário necessite ver na próxima vez que ele entrar terá que ser passada para a camada de persistência de dados em cada visita. Já em uma aplicação Ajax, parte da lógica da aplicação é movida para o navegador, como a figura abaixo ilustra. Neste novo cenário, quando o usuário entra, um documento mais complexo é entregue ao navegador, uma grande proporção do qual é código JavaScript. Este documento permanecerá com o usuário por toda a sessão, ainda que ele resolva provavelmente alterar sua aparência consideravelmente, enquanto o usuário está interagindo com ele. Ele sabe como responder às informações inseridas pelo usuário e é capaz de decidir se manipula a entrada do usuário ele mesmo ou se passa um solicitação para o servidor web (o qual tem acesso ao banco de dados do sistema e outros recursos), ou ainda, se faz uma combinação de ambos.

Ele também pode armazenar o estado, porque o documento continua persistindo sobre toda a sessão do usuário. Por

exemplo, o conteúdo de uma cesta de compras pode ser armazenado no navegador, em vez de ser armazenado na sessão do servidor.

O servidor fornece dados, e não conteúdo

Como observamos, uma aplicação web clássica oferece a mesma mistura de alegorias, conteúdos e dados em todos os passos. Quando nosso usuário adiciona um item na cesta de compras, tudo que precisamos realmente é responder com o valor atualizado da cesta ou informar se alguma coisa deu errado. Como ilustrado na figura logo abaixo, isto será uma parte muito pequena de todo o documento vivo.



Um carrinho de compras baseado em Ajax pode comportar-se de forma mais esperta, por meio de remessas de solicitações assíncronas ao servidor. O cabeçalho, o histórico de navegação, e outras características do layout da página estão todas carregadas, portanto o servidor necessita enviar de volta somente os dados relevantes.

Uma aplicação Ajax poderia fazer isto de vários modos, como por exemplo, devolver um fragmento de JavaScript, um fluxo de texto simples, ou um pequeno documento XML. Nós mostraremos em detalhes as vantagens e desvantagens de cada um, mais a frente. É suficiente dizer por agora que qualquer um destes formatos será muito menor que a misturada de informações devolvida pela aplicação web clássica.

Em uma aplicação Ajax, o tráfego tem sua maior intensidade no início, com um largo e complexo cliente sendo entregue em uma única explosão, quando o usuário entra. As comunicações subseqüentes com o servidor são muito mais eficientes, de qualquer forma. Para uma aplicação breve, o tráfego cumulativo pode ser menor em uma aplicação de página web convencional. Mas conforme o tamanho médio do tempo de interação aumentar, o custo de largura de banda da aplicação Ajax se torna menor do que sua aplicação clássica equivalente.

A interação do usuário com a aplicação pode ser flexível e contínua dummy

Um navegador web oferece duas maneiras de enviar entradas de dados para um outro computador: com os hyperlinks e formulários HTML.

Os hyperlinks podem sé-carregados com parâmetros CGI (Common Gateway Interface – Interface de Comunicação Comum) apontando para páginas dinâmicas ou servlets. Eles podem estar vinculados com imagens e folhas de estilo (CSS) para oferecer uma pequena melhoria na interface, como por exemplo, definir efeitos quando o mouse estiver sobre eles.

Os controles de formulário oferecem um subconjunto básico de componentes padrões de interface com o usuário: caixas de texto, caixas de checagem e botões de rádio, além de listas de seleção. Entretanto estes controles não são suficientes.

Não existem controles de seleção em árvores, grades para edição, ou caixas de combinação. Os formulários, assim como os hyperlinks, apontam para URLs resididas no servidor.

Alternativamente, os hyperlinks e os controles de formulário podem ser apontados para funções JavaScript. Isto é uma técnica comum em páginas web para prover uma validação de formulário rudimentar em JavaScript, verificando por campos vazios, valores fora de intervalo, e assim por diante, antes de submeter os dados para o servidor. Estas funções JavaScript existem somente enquanto a própria página existe e é substituída quando a página efetuar o seu envio.

Enquanto a página está sendo enviada, o usuário aguarda a sua resposta. A página anterior pode ainda estar visível por algum tempo, e o navegador pode até permitir que o usuário clique em qualquer um dos links visíveis, mas se assim for feito, produzirá resultados imprevisíveis e até entornar em uma confusão com a sessão no servidor. O usuário está normalmente aguardando a página ser atualizada que, frequentemente, possuem quase que as mesmas informações que lhes foram apanhadas instantes atrás. Adicionando um par de calças à cesta de compras não é razoável modificar as categorias em um nível acima por “roupas masculinas”, “roupas femininas”, “infantis” e “acessórios”.

Voltemos ao exemplo do carrinho de compras novamente. Devido ao fato de que nosso carrinho de compras em Ajax pode enviar dados assincronamente, os usuários podem soltar os objetos dentro dele tão rápido quanto eles podem clicar. Se o código de nosso carrinho no lado cliente for robusto, ele tratará esta tarefa facilmente, e os usuários podem continuar com o que eles estão fazendo.

É claro que não existe nenhum carrinho para colocarmos as coisas, somente um objeto em sessão no servidor. Mas os usuários não querem saber sobre objetos de sessão enquanto estão fazendo compras, e a metáfora do carrinho provê uma descrição do mundo real mais confortável do que está acontecendo. Troca de contextos entre a metáfora e o acesso direto ao computador é uma distração para usuários. Aguardar uma página ser atualizada levará o usuário à realidade de estar sentado em um computador por um curto tempo, e nossa implementação em Ajax evita que isto ocorra. Fazer compras é uma atividade transitória, mas se considerarmos um domínio de negócios diferente, por exemplo, um cenário de assistência e atendimento intensivo ou uma tarefa de planejamento complexa, então o custo de interrupção da seqüência de trabalho em alguns poucos segundos, com uma atualização de página, é algo inviável.

A segunda vantagem de Ajax é que podemos associar eventos a um maior número de ações do usuário. Os conceitos mais sofisticados de interface com o usuário, assim como “arrastar e soltar”, se tornam praticáveis, trazendo as experiências dessas interfaces em pé de igualdade com os controles de aplicações desktop. Da perspectiva de usabilidade, esta liberdade não é importante somente porque ela permite exercer nossa imaginação, mas porque ela nos permite combinar a interação do usuário e as solicitações ao servidor de maneira mais completa.

Para comunicar com o servidor em uma aplicação web clássica, necessitamos clicar em um hyperlink ou submeter um formulário, e então aguardar. No entanto, este método interrompe a interação com o usuário. Em contraste, a possibilidade de se comunicar com o servidor em resposta a um movimento ou arraste do mouse, ou até quando digitamos, habilita o servidor a trabalhar juntamente com o usuário. Google Suggest é um exemplo muito simples e efetivo disto: responder às teclas pressionadas enquanto ele digita dentro da caixa de pesquisa, e então, comunicar com o servidor para recuperar e exibir uma lista de possíveis finalizações para as expressões, baseada nas pesquisas feitas por outros usuários do mecanismo de busca em todo o mundo. Aqui demonstraremos uma simples implementação de um serviço similar.

Real codificação requer disciplina

Neste momento, as clássicas aplicações web fazem uso de JavaScript em certas ocasiões, para adicionar características avançadas e exageradas de um programa, as agregando nas páginas. O modelo baseado em páginas previne qualquer uma destas melhorias que consista em um atraso longo demais, o qual limita as utilidades para as quais elas podem ser oferecidas. Isto fez com que JavaScript recebesse injustamente, uma reputação de algo banal – por má sorte da linguagem – e não sendo bem vista pelos desenvolvedores sérios.

Codificar uma aplicação Ajax é algo completamente diferente. O código que você fornece quando os usuários iniciam a aplicação deve executar até que eles encerrem-na, sem interrupção, sem diminuição de velocidade, e sem produção de escapes de memória. Se estivermos mirando no mercado de aplicações poderosas, então temos em vista muitas horas de intenso uso. Para atingirmos este objetivo, devemos escrever códigos de alto-desempenho, e manuteníveis, usando a mesma disciplina e entendimento que é aplicado com sucesso às camadas do servidor.

A base de código será tipicamente mais ampla que qualquer código escrito para uma aplicação web clássica. Boas práticas na construção da base de código se tornam muito importante. O código deve tornar-se, de preferência, responsabilidade de uma equipe do que apenas um indivíduo, criando edições de manutenibilidade, separações de interesses, e estilos e padrões de codificação comum.

Uma aplicação Ajax, portanto, é uma porção de código funcionalmente complexa que comunica eficientemente com o servidor enquanto o usuário continua com seu trabalho. Ela é claramente uma descendência da aplicação clássica baseada em páginas, mas a similaridade não é mais forte do que entre um cavalinho de madeira e uma moderna bicicleta de passeio. Sustentando essas diferenças em mente nos ajudará a criar aplicações web verdadeiramente convincentes.

Prestadores de Seviço no Brasil

Anydesign.net (http://www.anydesign.net/)
Agência digital especialista em planejamento, criação e desenvolvimento de e-business.

- Anydesign.net teaser (http://www.anydesign.net/) Demonstração de um contador de vendas númerico; animado via AJAX.

Demonstrações

O Elcio Ferreira disponibilizou um material em português onde há duas demonstrações de AJAX com PHP.

- Parte 1 (http://www.tableless.com.br/ajaxdemo/) Ajax para quem só ouviu falar.
- Parte 2 (http://www.tableless.com.br/ajaxdemo2/) Ajax (parte 2): encarando o mundo real.

O site CLASSIME (http://www.classime.br22.com/classificados/)usa o ajax em várias partes do site. entre e confira:

- CLASSIME (http://www.classime.br22.com/classificados/)

Quentin Zervaas, um australiano que adora AJAX, escreveu umas classes em PHP que implementam algumas caracteríscas interessantes do uso de AJAX.

- Google Suggest Clone (http://ajax.zervaas.com.au/examples/GoogleSuggestCloneJax)
- Seleção de País/Estado/Cidade (http://ajax.zervaas.com.au/examples/CountryRegionCityJax)

O site Google Maps (http://maps.google.com/) é um bom exemplo de site dinâmico que usa AJAX para carregar somente as partes necessárias dos mapas, sem a necessidade de recarregar toda a página, a medida que o usuário navega pelos mapas.

O autor do site japs.etc.br (http://www.japs.etc.br/) disponibilizou em seu site, ótimos exemplos dos recursos que o ajax pode fornecer:

- Ajax (http://www.japs.etc.br/ajax/) Exemplos de AJAX

O Framework ATLAS.NET da Microsoft é uma boa ajuda para quem trabalha com AJAX.

- ATLAS.NET (http://atlas.asp.net/)

Artigos

- Reunião de Artigos sobre Ajax... (http://www.robsonjunior.com.br/post/juntao-ajax/)
- Cloning Google Suggest with AjaxAC (http://www.phpriot.com/d/articles/php/application-design/google-suggest-ajaxac/index.html) demonstra a criação de um clone para a pesquisa Google com sugestões.
- Porque AJAX com IFrame é melhor do que usando XMLHttpRequest (http://www.phpclasses.org/blog/post/51-PHPClasses-20-Beta-AJAX-XMLHttpRequest-x-IFrame.html)

Portais

- Ajax Impact (http://ajaximpact.com/)
- AjaxAC (http://ajax.zervaas.com.au/) cria algumas classes em PHP
- Ajax Goals (http://www.ajaxgoals.com/)
- Ajax Matters (http://www.ajaxmatters.com/) possui informações interessantes sobre AJAX
- Ajax Patterns (http://ajaxpatterns.org/wiki/index.php)
- Ajaxian (http://ajaxian.com/)
- AFLAX (http://en.wikipedia.org/wiki/AFLAX) AJAX combinado com Flash

Listas de Discussão

- AJAX World Group (http://groups.google.com/group/ajax-world), Profissionais do mundo inteiro discutem sobre AJAX.

Ferramentas

- Atlas (http://atlas.asp.net/), Microsoft's AJAX toolkit.
- Dojo Toolkit (http://dojotoolkit.org/), AJAX/DHTML toolkit.
- Prototype (http://prototype.conio.net/), open-source framework.
- Scriptaculous (http://script.aculo.us/) são várias bibliotecas em Javascript utilizando prototype OO.
- Rico, open-source.
- Sajax (http://www.modernmethod.com/sajax/), Simple AJAX toolkit
- Rialto (http://rialto.application-servers.com/), Rich Internet AppLication TOolkit.
- DWR (http://getahead.ltd.uk/dwr/), Direct Web Remoting - framework AJAX para Java Servlets
- [1] (http://www.orkut.com/Community.aspx?cmm=3214697), Comunidade Framework Ajax, no Orkut.
- [2] (http://www.indiankey.com/cfajax/), CFAJAX ajax for ColdFusion.
- [3] (http://www.phpclasses.org/browse/package/2520.html), PAJAX - ajax for php made a brazilian.
- ZK (http://zk1.sourceforge.net/), Ajax mas nenhum Javascript. Demo (http://www.potix.com/zkdemo/userguide)

Retirado de "http://pt.wikipedia.org/wiki/AJAX_%28Web%29"

Categorias de páginas: !Esboços sobre informática | WWW | Programas de computador | XML