Studienarbeit im Fach Multimedia und Webtechnologien

bei Prof. Dr.-Ing. Stefan Gössner

im WS 05/06

Thema:

AJAX- basierte Webanwendungen
Asynchronous JavaScript and XML

Daniel Adolf, Marc Fliege, Christian Laun

Inhaltsverzeichnis

Was heißt AJAX und was bedeutet es?

Der Begriff „*AJAX*“ ist ein Kunstwort, welches sich aus der Wortfolge „**A**synchronous **J**avascript **a**nd **X**ML“ zusammensetzt.

Asynchron: Bei einer asynchronen Arbeitsweise wird der Datenfluss nicht von einer zentralen Takteinheit (Taktgeber) gesteuert. Dadurch sind die zeitlichen Abstände zwischen den Schritten eines Ablaufs unterschiedlich.

Javascript: JavaScript ist eine objektbasierte Skriptsprache um statische HTML-Seiten dynamisch zu gestalten.

XML: Die Extensible Markup Language, abgekürzt XML, ist ein Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur. XML definiert dabei die Regeln für den Aufbau solcher Dokumente.

Dabei handelt es sich um ein Datenübermittelungskonzept zwischen Server und Browser. Das Novum an diesem Verfahren ist die Dynamik beim Hochladen der Internetseiten. Dabei muss nicht der gesamte Inhalt neu aufgebaut werden, sondern nur die sich veränderten Teile werden „nachgeladen“.
Vorteile für die Nutzer ergeben sich durch schnellere Ladezeiten der Web-Anwendung und die ständige Aktualität der Daten.

Komponenten von AJAX

Der Grundaufbau von AJAX basiert auf verschiedensten Web-Technologien, welches ein interaktives Arbeiten im Web ermöglicht. Folgende Technologien kommen zur Umsetzung dynamischer Internetseiten zum Einsatz:
CSS (Cascading Style Sheets; Auszeichnungssprache zur Definition des Layouts von Dokumenten), HTML (Hyper-Text-Markup-Language) definieren das Layout und den Aufbau von Internetseiten. XML (Extensible Markup Language) wird zum Datenaustausch zwischen Server und Client genutzt. DOM (Document Object Model; Repräsentation strukturierter Dokumente als objekt – orientiertes Modell) wiederum stellt die Inhalte der Seite dar und ermöglicht eine aktive Manipulation des Inhaltes über JavaScript. Das XMLHttpRequest-Objekt wird genutzt, um Daten auf asynchroner Basis mit dem Webserver austauschen zu können.

Prinzip und Funktionsweise:

Klassisch

Anwendungen, welche auf AJAX basieren, lassen den Eindruck erwecken, die Anwendung würde vollständig auf dem Rechner des Nutzers ablaufen, da es kaum zu Unterbrechungen während des Nutzens der Dokumente kommt.

Bei herkömmlichen Webanwendungen wird die Ladezeit der Daten durch eine Anfrage-Antwort-Transaktion zwischen Client und Server bestimmt. Dadurch können doch erhebliche Wartezeiten durch Übertragungsprobleme entstehen. Bei diesen Wartezeiten kann der User nicht weiter eingreifen und muss warten, bis die gewünschte Antwort des Servers auf dem Clientrechner dargestellt wird.

Die herkömmlichen Verfahren transferieren die vom User bearbeiteten Formulare und Scripte an den jeweilig zuständigen Server, welcher dann erst die Bearbeitung der Daten übernimmt. Nach Bearbeitung der benutzerspezifischen Daten generiert der Webserver die angeforderte Seite und gibt sie dem Client zurück. Dieser beginnt dann mit dem Neuaufbau der Seite. Aufgrund der Tatsache, dass der Webserver bei jeder Anfrage des Nutzers eine neue Webseite erstellt und diese übermitteln muss, erscheint dem Benutzer die Anwendung als träge und unflexibel.

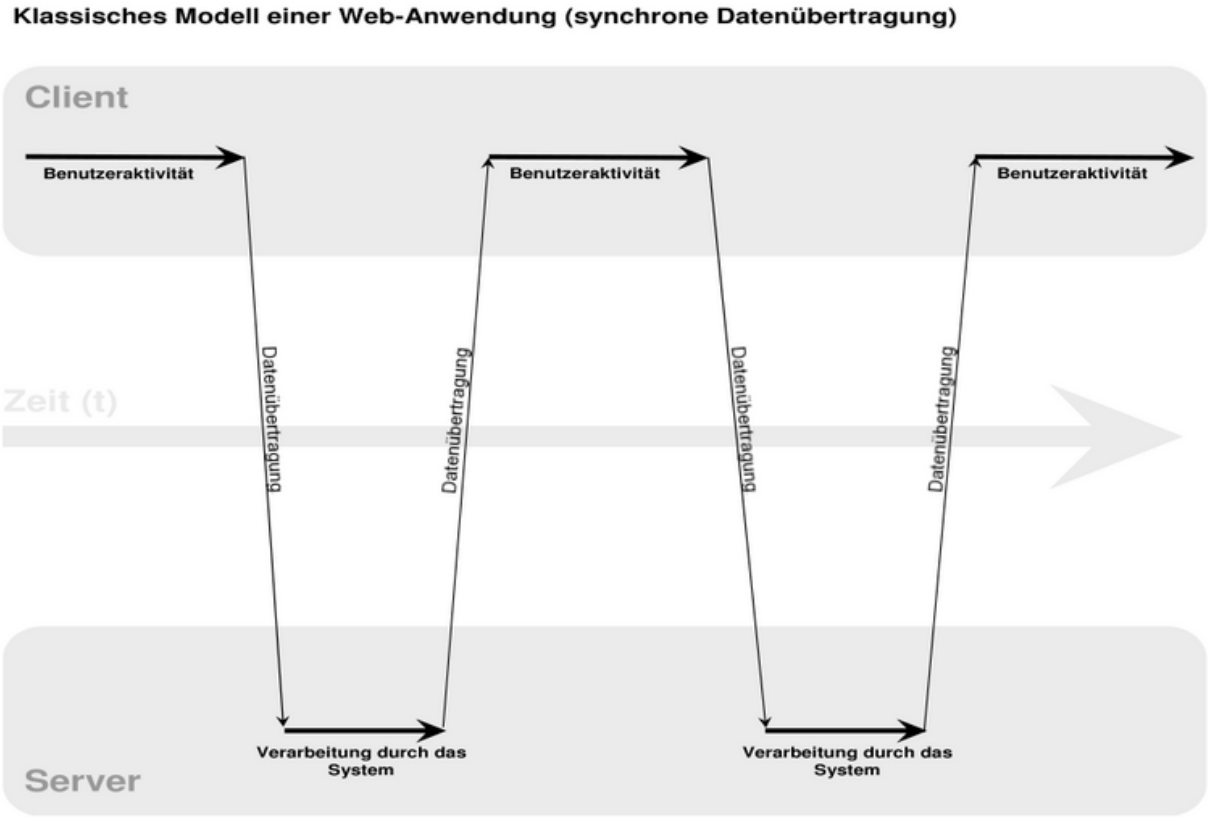

Abb. 1 Quelle: www.wikipedia.de 2005-12-17

AJAX

Bei diesem Verfahren ist es nicht mehr nötig, die komplette Seite neu aufzubauen. Durch das Zusammenspiel, der oben schon erwähnten Webanwendungen, ist es möglich gezielte Anfragen an den Server zu stellen. Das aktuelle Dokument wird, ungeachtet der Bearbeitung und der Transaktion im Hintergrund, weiter genutzt. Wenn der Server den „Request“ stellt, wird die aktuelle Seite ohne Einschränkung der Funktionen aktualisiert. Dies wird mit Hilfe eines XML -basierten Web- Service realisiert. Der Request wird nicht direkt vom Browser verarbeitet, sondern in das Javascript-Dokument eingebunden. Durch diesen wesentlich geringeren Datentransfer zwischen Server und Client, ist es möglich die Benutzeranfragen wesentlich schneller zu bearbeiten.

Ein weiterer Vorteil ist der direkte Einsatz des Clients bei wesentlichen Verarbeitungsschritten der Anfrage. Dadurch verringert sich die Auslastung des Servers und somit müssen auch Daten, welche der Client im Stande ist zu verarbeiten, nicht unnötig zum Server gesendet werden.

Durch das Einbinden in die Java-Script-Routinen ist es AJAX möglich, nur die Teile der Webseite zu aktualisieren, welche definitiv verändert wurden. Die Aktualisierung erfolgt über Dynamisches HTML.

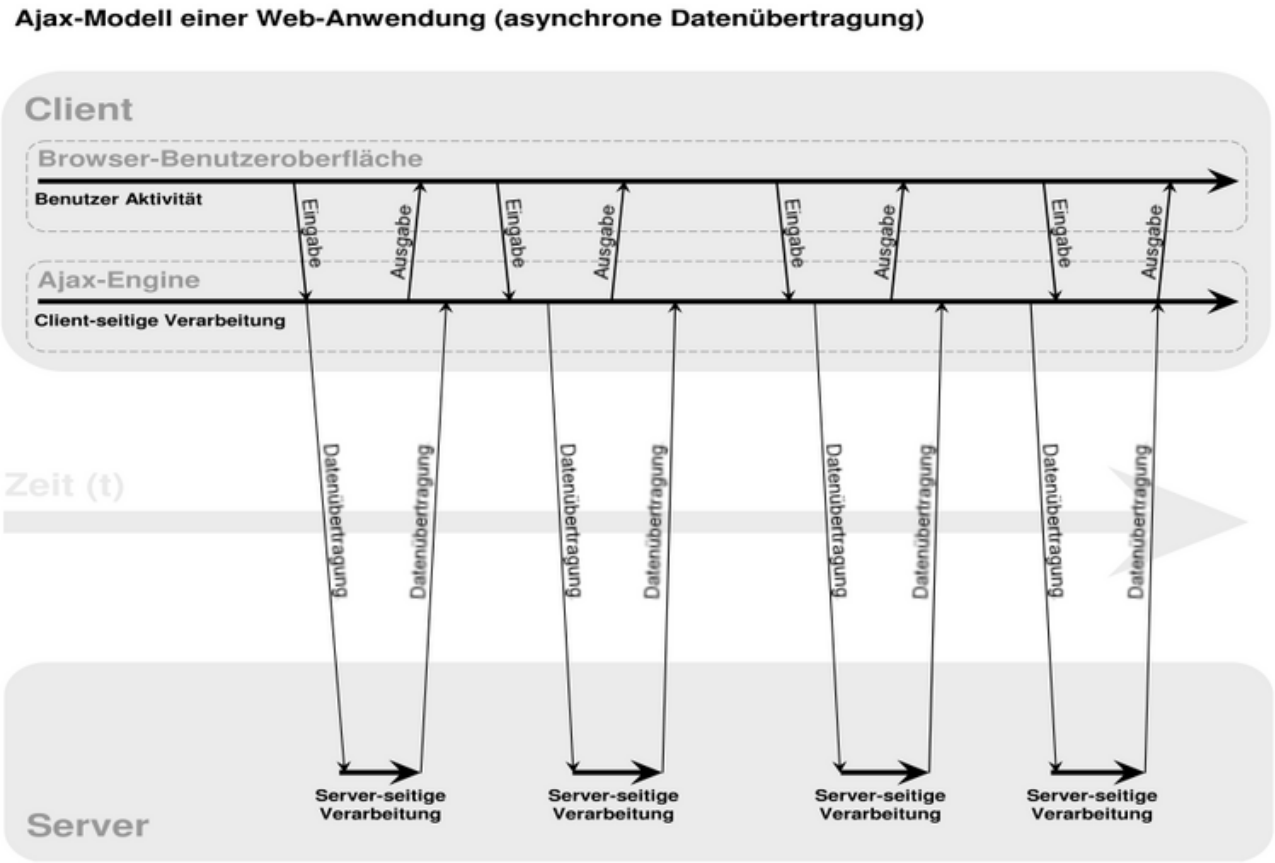

Abb. 2 Quelle: www.wikipedia.de 2005-12-17

Die AJAX- Anwendungen(Client-Plattformen) können mit allen Browsern ausgeführt werden, welche die Verfahren CSS, (X)HTML, XML, XSL, Javascript und DOM implementiert haben.
Ein weiteres entscheidendes Element ist, dass der eigentliche Prozess des Datenaustausches auf dem Server hinterlegt ist.
Die Hinterlegung erfolgt z.B. mit EJBs (**Enterprise Java Beans** sind standardisierte Komponenten innerhalb eines J2EE-Servers [Java 2 Enterprise Edition]. Sie vereinfachen die Entwicklung komplexer mehrschichtiger verteilter Softwaresysteme mittels Java). Das Ajax-Konzept selbst erfordert keine spezifische Technik, mittels derer die Serverseitige Programmlogik umgesetzt werden muss. Sowohl der Server als auch die Anwendungslogik werden im Ajax-Kontext als Server-Plattform bezeichnet.
Der größte Vorteil von AJAX besteht in der Veränderung von Daten, ohne die Webseite vollständig neu aufzubauen.

AJAX: asynchrone Kommunikation

Zur Veranschaulichung der Asynchronität siehe Abbildung 3. Die Kommunikation zwischen Browser und Server, die der Benutzer wahrnimmt, ist die, die seine Aktionen auslösen: das Anklicken eines Links oder das Abschicken eines Formulars. Den Takt der Kommunikation gibt der Benutzer vor (im Bild als diskrete Zeitpunkte *t=1,2,3* dargestellt); zwischen den Takten redet der Browser normalerweise nicht mit dem Server. Bei Ajax ist das anders.

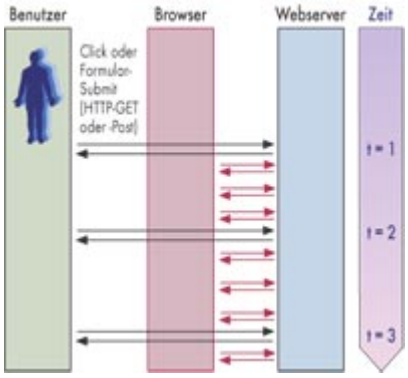

Abb. 3 Quelle:www.heise.de 2005-12-16

Hier kann ein Script eine Kommunikation außerhalb des Benutzertaktes auslösen (im Bild als rote Pfeile dargestellt). Der Server antwortet mit einer XML-Nachricht, die Javascript als DOM-Objekt zur Verfügung steht.

AJAX: Vorteile und Probleme

Mit AJAX bieten sich neue, erweiterte Möglichkeiten bei der Gestaltung fortschrittlicher Web-Oberflächen. Neben all den bedeutenden Vorteilen kann die Verwendung von AJAX aber auch zu Problemen führen, die immer bedacht werden sollten. Einige Vor- und Nachteile von AJAX sollen nachfolgend kurz genannt werden.

Da derzeit fast nur über die Vorteile und Möglichkeiten von AJAX gesprochen wird, wird hier nachfolgend verstärkt auf die Nachteile und Probleme von AJAX eingegangen. Zwar bietet AJAX ein enormes Potential für fortschrittliche Web-Oberflächen, aber besteht auch die Gefahr, dass der Trend um AJAX dazu führt, dass immer mehr Web-Seiten AJAX in Bereichen einsetzen, wo es absolut nachteilig ist, ähnlich, wie es ursprünglich mit DHTML gewesen ist.

Vorteile

- Seiteninhalte können durch neue Daten vom Server geändert werden, ohne dass die Seite komplett neu geladen werden muss. Damit wird die Serverlast verringert. Änderungen werden schneller sichtbar, die User bekommen schneller Feedback. Das erleichtert und verbessert die Nutzung. Zudem bleibt beim Nachladen der Daten der aktuelle Zustand erhalten (Position des Cursors, von Elementen, usw.). Das vereinfacht die Programmierung.
- Es können Oberflächen geschaffen werden, die Desktop-Applikationen ähneln. Damit können zunehmend mehr Aufgaben über Web-Applikationen erledigt werden. Das kommt der zunehmenden mobilen Nutzung zugute. Die Stärke des Webs, immer und überall verfügbar zu sein, wird damit besser ausgenutzt

Nachteile / Probleme / Grenzen

- AJAX setzt voraus, dass JavaScript aktiviert ist. Damit scheidet die Verwendung von AJAX für normale Web-Seiten, die auch ohne Javascript benutzbar sein müssen, weitgehend aus. Vorrangiges Einsatzgebiet für AJAX sind Web-Seiten mit Applikationscharakter, z.B. in Intranets. Auf normalen Web-Seiten sollte AJAX derzeit nur als optionales Tool verwendet werden.
- AJAX setzt beim Internet Explorer voraus, dass die Ausführung von ActiveX-Objekten erlaubt ist. Aus Sicherheitsgründen haben manche Benutzer/innen dieses abgeschaltet.
- Die Verwendung von AJAX kann zu einem deutlich höheren Aufwand für die Client- und Serverseitige Programmierung und damit auch einen finanziellen Mehraufwand führen. Dort, wo AJAX zum Einsatz kommt, wird häufig zumindest für eine Übergangszeit auch noch eine Alternativlösung implementiert werden müssen, damit wichtige Teilbereiche einer Seite, z.B. Formulare, auch ohne AJAX verwendet werden können. Das wird zu einem deutlichen Mehraufwand führen.
- Die Trennung von Struktur (XHTML) und Layout (CSS) wird möglicherweise wieder aufgehoben. Die Entwicklung im Bereich Web-Frontends geht hin zu einem semantischen Web. D.h., nicht das Layout, sondern die strukturierte Darstellung der Daten in XML-Form ist vorrangig relevant. Das Layout wird weitgehend getrennt davon, je nach

Ausgabemedium über CSS oder XSLT erzeugt. DHTML bedeutet das Zusammenspiel von Javascript und CSS. Die Gefahr ist also, dass hier diese sinnvolle Trennung nicht beibehalten wird.

- Schnittstellen werden möglicherweise unbedienbar, da sie neue Elemente enthalten, die die Nutzer nicht erwarten, oder Bedienelemente nicht wie gewünscht funktionieren (z.B. kein Submit-Button, Back-Button funktioniert nicht). Mittlerweile haben sich bestimmte Abläufe und Quasi-Standards bei der Bedienung von Web-Seiten durchgesetzt. Formulare werden über den Submit-Button abgeschickt, Navigationen befinden sich innerhalb bestimmter Bereiche der Web-Seite. Wer heute etwas über einen Online-Shop bestellt wird feststellen, dass die meisten Shops relativ ähnlich aufgebaut und deshalb einfach zu benutzen sind. Wenn jetzt bestimmte Elemente, z.B. Formulare, nicht mehr so wie gewohnt funktionieren, kann das mehr zur Verwirrung beitragen, als das es den Anwendern nutzt.
- Seiten, die mittels AJAX verändert worden sind, können nicht einfach ausgedruckt werden, da der Browser derzeit die Seite so ausdruckt, wie sie ursprünglich vom Server geladen worden ist.
- AJAX-Anwendungen unterliegen den Beschränkungen des zustandlosen HTTP-Protokolls. Eine dauerhafte Verbindung zum Server ist nicht möglich, d.h. das AJAX-Clients sich nicht beim Server registrieren um von ihm über bestimmte Ereignisse informiert zu werden, auf die der AJAX-Client dann reagieren kann. Stattdessen muss er regelmäßig Anfragen an den Server schicken, um zu klären, ob ein neues Ereignis eingetreten ist. Das kann den Vorteil von AJAX, die Serverlast zu verringern, in das Gegenteil umkehren.
- DHTML / AJAX ist nicht vereinbar mit Barrierefreiheit. Seiten, die barrierefrei sein sollen, müssen immer auch ohne Javascript bedienbar sein. Da AJAX aber Javascript erfordert, ist hier die Barrierefreiheit nicht gegeben.
- Der sozusagen hinter dem Rücken der User durchgeführte Datenaustausch kann zu Verunsicherungen führen, da die User/innen nicht mehr die Kontrolle über ihre Daten haben (insbesondere bei Formularen, wo die Eingaben normalerweise erst nach Betätigen des 'Submit'-Buttons gesendet werden).Es entsteht eine Transparenz und man entwickelt sich hin zum „gläsernen User“.
- Nach wie vor sind zahlreiche Methoden und Verhaltensweisen in unterschiedlichen Browsern verschieden oder auch gar nicht implementiert. Bestimmte Methoden funktionieren nicht oder zumindest nicht wie gefordert. Das bedeutet, dass umfangreiche Tests in verschiedenen Browsern nötig sind. Nicht selten nimmt der Aufwand dafür, bestimmte Verhaltensweisen, die nicht oder fehlerhaft im Browser implementiert sind,

mittels Javascript nachzubilden, einen wesentlichen Teil des Gesamtaufwands bei der Programmierung ein.

Eine ganze Reihe von weiteren Problemfällen mit AJAX kann man in *http://sourcelabs.com/ajb/archives/2005/05/ajax_mistakes.html* nachlesen.

Herausforderung

Die Aufgabe in der Zukunft wird es sein, dafür zu sorgen kreative und konzeptionelle Lösungen zur Behebung der genannten Probleme zu Lösen. Im Wesentlichen besteht die Aufgabe auch darin, deutlich zu machen, was sich geändert hat und wodurch diese Aktion erfolgte, bzw. auch wie die Interaktion jederzeit wieder aufrufbar oder revidierbar ist.
In einem sehr guten Artikel zum Thema Web 2.0 von Tim O'Reilly schreibt er, daß die zukünftigen Anforderungen an ein Internet-Unternehmen sich wandeln werden. Die Stärke liegt nicht mehr darin eine Software zu entwickeln und diese zu verkaufen. Vielmehr ist die Kontrolle und das Sammeln von Daten und deren ständige Verarbeitung, Aufbereitung und Bereitstellung eine der wichtigsten Komponenten.

Beispiele

Mittlerweile halten immer mehr Ajax-Anwendungen Einzug in das tägliche Surfen und manchmal bemerkt man erst auf den zweiten Blick, dass das Surfen gerade einen Tick einfacher war. Wenn z.B. nach dem Ausfüllen eines Textfeldes auf den Senden-Button gedrückt wird und das eben eingegebene sofort auf der Seite lesbar ist, ohne dass sich die Webseite neu aufbaut.

Ein sehr schön anschauliches Beispiel ist der XMLHttpRequest Business Card Creator von Thomas Baekdal. Bei Eingabe einer ungültigen E-Mail-Adresse wird der Benutzer sofort auf den Fehler hingewiesen. Ganz ohne Absenden und Laden *(http://www.baekdal.com/x/xmlhttprequest).*

Wie es bei allen neuen Ansätzen so ist, verbreitet auch Ajax sich rasend schnell und nicht immer zum Guten. Während der beschriebene Ansatz, die Bedienung von Webapplikationen eindeutig erleichtern und ihre Zugänglichkeit deutlich erhöhen kann, gibt es auch fragwürdige Einsatzgebiete

Ein weiteres Beispiel ist Google Suggest *(www.google.com/webhp?comlete=1&hl=en).* Hier werden bereits nach dem Eingeben erster Buchstaben relevante Ergebnisse unterhalb des Eingabefeldes angezeigt. Inklusive der Anzahl erzielter Treffer.

Ein anderes schönes Beispiel für eine auf Ajax beruhende Site ist zum Beispiel das neue Yahoo Mindset *(http://mindset.research.yahoo.com).* Mit einem Schieberegler lassen sich die Suchergebnisse zwischen *shopping* und *research* variieren. Verschiebt man den Regler weiter nach rechts, werden zunehmend Ergebnisse angezeigt, die über die Historie oder Fakten zum Suchergebnis berichten. Verschiebt man den Regler nach rechts, werden die Ergebnisse von Shopping Seiten und Preisvergleichern bestimmt.
Google Mail *(http://gmail.google.com)* und Google Maps *(http://maps.google.com)* waren die ersten breiten Demonstrationen von Web Anwendungen.

Historie zu AJAX

Im Laufe der letzten zehn Jahre ist die Anzahl der Techniken, die das Web dynamischer gestalten, stetig gestiegen. Unter dem Namen **Ajax** erregt seit Kurzem eine Idee Aufsehen, die nicht wie DHTML und Flash die Darstellung von Webseiten beleben soll, sondern bei der Kommunikation zwischen Browser und Webserver ansetzt und asynchron XML-Nachrichten austauscht.

1995

Als Ende 1995 Netscape seinen Browser mit einer eigenen Programmiersprache ausstattete, dienten die ersten Listing-Beispiele dazu, HTML-Formulare „smart“ zu machen. Das Formular war etwa in der Lage, gewisse Eingabefehler zu erkennen und den Benutzer darauf hinzuweisen. Die Verbesserung bestand vor allem darin, dass der Browser den Fehler erkennt und nicht erst die Formulardaten an den Server überträgt, der mit der Fehlermeldung antwortet. 1995, als die Leitungen noch etwas schmaler waren, stellte dieser Ansatz einen wichtigen Schritt dar, um Webanwendungen schneller und damit komfortabler zu gestalten. Bei der Verarbeitung steckt die gesamte Logik im Client, genauer im Javascript-Programm, das er ausführt. Das ist ein guter Ansatz, wenn die Logik nicht auf große Datenmengen zugreifen muss, um ihre Arbeit zu erledigen. Zur Illustration folgendes Szenario: Ein HTML-Formular erlaubt die Eingabe einer Adresse mit Postleitzahl, Ort und Straße. Nun heißt es, die Plausibilität der Eingabe zu prüfen. Es geht um die Beantwortung der Frage, ob in der genannten Stadt das genannte Postleitzahlgebiet und die genannte Straße liegen. Der klassische Ansatz besteht in der Überprüfung der Daten auf der

Serverseite nach einem „Submit“ des Formulars. Andererseits wäre es bei Verwendung von Javascript im Client notwendig, die Information über alle Straßen in Deutschland zur Überprüfung an den Browser zu senden. Beide Ansätze sind nicht optimal und genau hier setzt **Ajax** an. Bei dieser Idee schickt der Browser die eingegebenen Daten ohne Zutun und ohne dass der Benutzer es merkt an den Server. Der überprüft die Daten und schickt eine Nachricht, die das Ergebnis der aufwendigen Prüfung enthält, an den Browser zurück. Dieser reagiert auf das zurückgelieferte Ergebnis, etwa mit einer Fehlermeldung, dass die eingegebene Straße nicht im eingegebenen Postleitzahlengebiet liegt.

1998

Browserkrieg zwischen Netscape und Microsoft. Die konkurrierenden Unternehmen verteilten ihre Browser kostenlos. Immer neue Browser-Technologien wurden entwickelt um sich gegenüber der Konkurrenz abzusetzen. Es gab im Bereich Web-Entwicklung einen extremen Schub, immer mehr Unternehmen haben den Schritt ins Internet gewagt. Zu dieser Zeit entstanden sehr aufwendige Web-Seiten die nicht sehr bedienerfreundlich waren, sondern unübersichtlich. Weiterhin entstanden Probleme die Web-Seiten auf den verschiedenen Browsern richtig aufbauen zu können.

2001

Viele Firmen zogen sich aus dem Internet-Geschäft zurück. Der Nutzen der Web-Seite stand nicht zum Verhältnis zu den Kosten! Weg vom Layout zu mehr Inhalt war der Tenor. Es wurde mehr darauf geachtet die Aktualisierung der Web-Seiten mit Hilfe von Content-Management-Systeme einfach und schnell zu gestalten, um die Kosten zu senken. Die Notwendigkeit verbindliche Standards in der Web-Technologie einzuführen wurde immer größer. Firma Microsoft führt das XMLHTTP - Request ein. Warum hat sich das XMLHTTP - Request nicht durchgesetzt? Das XMLHTTP - Request war ein Erweiterungsobjekt von Microsoft, was von keinen anderen Browser unterstützt wurde. Es gab gravierende Sicherheitslücken. Nicht nur die gewünschten Remote - Dateien der Webseiten konnten mittels XMLHTTP - Request ausgelesen werden, sondern auch Dateien von der lokalen Festplatte. Die Anwendung von DHTML (Dynamic HTML), sprich die dynamische Änderung des Layouts und Inhaltes mittels Javascript war nur sehr beschränkt möglich: Sinnvolle Anwendung mit Netscape-4 Browser nicht möglich. Netscape und Microsoft Browser stürzten häufig bei DHTML Anwendungen ab. Der Grund dafür war, dass Javascript relativ langsam ist und dadurch komplexe Anwendungen mit der damals vorhandenen Hardware kaum möglich waren. Relativ aufwendig zu programmieren.

2002

DHTML beschränkte sich im Wesentlichen auf Animationen und wurde nach und nach von Flash-Animationen ersetzt. Flash-Animationen liefen auf beiden Browsern viel problemloser und brauchten nicht unterschiedlich entwickelt werden.

2004/2005

Die Konsolidierung ist weitestgehend abgeschlossen und Standards wie festgelegte Schnittstellen haben sich weitestgehend auf XML Basis durchgesetzt. Es gab einen extremen Leistungszuwachs der Hardware. Weiterhin gibt es einen starken Anstieg der Datenmenge die im World Wide Web verarbeitet und zur Verfügung gestellt wird. Immer mehr Dienste werden über das Internet angeboten. Durch die rasante Entwicklung der Hardwarekomponenten ist die dynamische Anwendung von Internetseiten mittels Javascript wieder interessant geworden. Die Anwendung von Javascript ist mittlerweile in allen Browsern weitgehend identisch. XMLHTTP - Request bekommt einen neuen Namen: AJAX (April 2005)! Mittels AJAX wird die Lücke, zusätzliche Daten vom Server anzufordern um HTML-Dokumente aktualisieren zu können, geschlossen. XMLHTTP - Request ist in allen modernen Browsern implementiert. AJAX spielt zurzeit eine immer größere Rolle in der Web-Technologie, Programmierer kommen nicht mehr um die Technologie herum, wenn sie nicht ins Hintertreffen geraten wollen.

Programmierung

Allgemein muss im JavaScript das ”request” für die unterschiedlichen Browser definiert werden. Ohne diese Vorarbeit, kann nicht jeder Browser das http-Request anwenden. Zum Beispiel hat Microsoft die Eigenart, das http-Request über das ActiveXObjekt zu steuern. Der Grundtyp, womit alle XMLHTTP-Request Scripte beginnen sollten, ist in dem folgenden Quellcode angegeben:

```
var http_request = false;

    function makeRequest(url) {

        http_request = false;

        if (window.XMLHttpRequest) {                    Mozilla, Safari …
        http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
        http_request.overrideMimeType('text/xml');
        }


        } else if (window.ActiveXObject) { // IE
        try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
        try {
        http_request = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
        }
        }
```

AJAX und die Alternativen

Neben der Möglichkeit mit Javascript und dem XMLHttpRequest dynamische Aktionen durchzuführen gibt es auch andere Ansätze die ähnliches leisten. Zum Beispiel die von Mozilla entwickelte Lösung **XUL (XML User Interface Language)**, die von Microsoft entwickelte **XAML (eXtensible Application Markup Language)** und die auf Flash gestützte Markup-Sprache **OpenLaszlo**. Auch **Greasemonkey** bietet eine gute Möglichkeit interaktive dynamische Elemente auf Benutzerebene zu einer Webseite hinzuzufügen.

AJAX Frameworks

Seit Beginn dieses Jahres ist eine Vielfalt von Frameworks entstanden, um die Ajax Funktionalität einfach zu implementieren. Im Folgenden werden exemplarisch einige Frameworks aufgezählt:

• Rico (JavaScript - http://openrico.org)
• Prototype (JavaScript - http://prototype.conio.net)
• Sajax (PHP - http://www.modernmethod.com/sajax)
• Backbase (JAVA, .NET - http://www.backbase.com)
• ADF Faces (JAVA, JSF - http://www.oracle.com)

Fazit

Die aktuellen Entwicklungen im Umfeld von AJAX Werkzeugen zeigen viele interessante Ansätze wie z.B. im Bereich der Echtzeit-Datenverarbeitung und –Benachrichtigung, in der Ajax-Komponenten für publish/subscribe Frameworks entwickelt werden, um die Server Last insbesondere für Seiten mit hohen Zugriffszahlen zu reduzieren.

Quellenverzeichnis

Internet

www.exanto.de	18.12.05
www.ajaxtalk.de	18.12.05
http://www.pdv-tas.de	18.12.05
www.google.com/webhp?comlete=1&hl=en	17.12.05
http://maps.google.com	17.12.05
http://gmail.google.com	17.12.05
www.get-the-code.de	15.12.05
www.wikipedia.de	16.11.05
www.heise.de	16.11.05
www.drweb.de	16.11.05
www.pixelgrahix.de	16.11.05
www.agenturblog.de	16.11.05

Bücher

Flanagan,D. JavaScript kurz & gut : Übersetzt durch G.Selke, Auflage 2, O`Reilly Verlag Köln, 2003

Anhang:

Programmierungsbeispiel zum Selberprogrammieren:

Schritt 1: Die Hauptformularseite in HTML erstellen (siehe Listing A). Dieses Beispiel beschränkt sich auf das nötige Minimum, aber natürlich können zusätzliche Datenelemente, wie Auswahlfelder oder Textbereiche, eingefügt werden. Die Datei wird **step1.htm** genannt.

Listing A

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM METHOD=POST ACTION=”saveData.jsp”>
<INPUT TYPE=TEXT NAME=DATA ID=DATA>
<INPUT TYPE=SUBMIT VALUE=Submit>
</FORM>
</BODY>
```

Schritt 2: Eine Seite auf dem Server schreiben, welche die eingereichten Daten anzeigt. Da AJAX rein clientseitig arbeitet, kann man diesen Code in jeder beliebigen Sprache erstellen. Für dieses Beispiel wurde Java gewählt. Man könnte aber auch jegliche andere serverseitige Sprache benutzen. Die Datei wird **saveData.jsp** genannt. Das müsste dann wie folgt aussehen:

```
<%=”Data Saved: “ + request.getParameter(“DATA”)%>
```

Test: step1.htm in den verwendeten Browser laden, einige Daten eingeben und auf Einreichen(submit) klicken. Die Daten müssten in dem Browser angezeigt werden, andernfalls ist ein Debugging des serverseitigen Codes durchzuführen. Dann die Seite neu laden. Der Browser zeigt nun eine Warnmeldung an, dass ein erneutes Laden der Seite die Formulardaten ein zweites Mal einreicht. Dies ist für die Benutzer störend und für die Webentwickler ein Problem.

Schritt 3 (AJAX Implementierung)

Hier wird die Datei step1.htm so geändert, dass eine asynchrone Formulareingabe über das XMLHttpRequest-Objekt mit AJAX unterstützt wird. Dazu die Datei step1.htm in step3.htm kopieren und die folgenden Änderungen vornehmen:

Listing B

```
<HTML>
<HEAD>
<SCRIPT>
```

```
function submitForm()
{
  var http = null;
  if(window.XMLHttpRequest)
    http = new XMLHttpRequest();
  else if (window.ActiveXObject)
    http = new ActiveXObject("Microsoft.XMLHTTP");
  http.onreadystatechange = function()
  {
    if(http.readyState == 4)
      alert("Server Response Was: " +
        http.responseText);
  };
  http.open('POST', 'saveData.jsp', true);
  http.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
  http.send("DATA=" + document.getElementById('DATA').value);
}
</SCRIPT>
</HEAD>
<BODY>
  <FORM METHOD=POST ACTION="saveData.jsp">
    <INPUT TYPE=TEXT NAME=DATA ID=DATA>
    <INPUT TYPE=BUTTON VALUE=Submit ONCLICK="submitForm()">
  </FORM>
</BODY> </HTML>
```

Die Schaltfläche zum Einreichen in einen Schaltflächentyp umwandeln.

Einen durch Klicken ausgelösten Event-Handler für die Schaltfläche hinzufügen.

Die JavaScript-Event-Handler-Funktion hinzufügen.

Die XMLHttpRequest-Call-Back-Routine hinzufügen.

Die komplette Liste der Werte sieht wie folgt aus:

0 (nicht initialisiert)

1 (lade)

2 (geladen)

3 (interaktiv)

4 (vollständig)

Nun soll die Bedeutung des in Listing C enthaltenen Codes näher erläutert werden.

Listing C

```
<if(window.XMLHttpRequest)
http = new XMLHttpRequest();
else if (window.ActiveXObject)
```

Hier ist zunächst einmal die Code-Zeile zu betrachten, welche das XMLHttpRequest-Objekt erzeugt. Dieser Code verwendet je nach Art des Browsers verschiedene Instantiierungstypen. Mozilla, Firefox und Safari implementieren das XMLHttpRequest als ein spezifisches Objekt, wogegen der Internet Explorer es als Active X-Control implementiert. Nun folgt der Code, der die Server-Antwort verarbeitet (siehe Listing D).

Listing D

```
<http.onreadystatechange = function()
{
  if(http.readyState == 4)
    alert(“Server Response Was: “ +
      http.responseText);
};>
```

Hierfür wird eine anonyme Funktion verwendet, die dem Ereignis ReadyStateChange des XMLHttpRequest-Objekt zugewiesen ist. Das ReadyStateChange-Ereignis wird ausgelöst, wenn das XMLHttpRequest-Objekt seinen Zustand ändert oder wenn verschiedene Netzwerk-Ereignisse eintreten. Eines davon ist der Eingang von Daten vom entfernten Host beim Client. Bei der Ankunft der Daten vom Server beim Client, zeigt dieser Handler dem Benutzer die Server-Anwort an. Dieser Event-Handler ist jedoch nur für synchrone Anfragen erforderlich.

Damit kommt der geschäftliche Aspekt dieser Vorführung in Sicht (siehe Listing E).

Listing E

```
<Hhttp.open(‘POST’, ‘saveData.jsp’, true);
  http.setRequestHeader(‘Content-Type’,
    ‘application/x-www-form-urlencoded’);
      http.send(“DATA=” + document.getElementById(‘DATA’).value); >
```

Hier findet das eigentliche Versenden von Daten an den Server statt. Die einzelnen Code-Zeilen führen die folgenden Schritte aus:

Öffnen einer HTTP-Verbindung zum Server. Die Parameter sind der HTTP-Aufruftyp, die URL sowie wahr für asynchrone und falsch für synchrone Verbindungen.

Den Content-Typ für die POSTed-Formulardaten festlegen (auslassen für GET, HEAD und so weiter).

Den Datenstream über HTTP an den Host senden.

Damit ist dieses Beispiel zur Anwendungsentwicklung mit AJAX abgeschlossen.