AJAX - Konzept und Anwendung

Stefan Barthel, Rami Eid-Sabbagh, Stephan Müller, Christian Tinnefeld

Konzepte und Methoden der Web-Programmierung 2005
bei Prof. Dr. sc. nat. Meinel
Hasso-Plattner-Institut für Softwaresystemtechnik

{stefan.barthel,rami.eidsabbagh.stephan.mueller,christian.tinnefeld}@hpi.uni-potsdam.de

Ajax steht für Asynchronous Javascript and XML und bezeichnet ein Konzept für die Erstellung von Web Anwendungen. Dieser Bericht erläutert das Konzept, stellt die eingesetzten Technologien vor und demonstriert die Verwendung anhand eines konkreten Beispiels. Abschließend werden die Vor- und Nachteile von Ajax aufgezeigt, sowie der Einsatz von Ajax im Semesterprojekt evaluiert.

1 Einleitung

Mit Ajax bezeichnet man ein Konzept, das die asynchrone Kommunikation zwischen Webclient und Webserver ermöglicht. Ajax selbst ist keine eigenständige Technologie, sondern wird realisiert durch die Verwendung mehrerer existierender Technologien. In Kapitel 1.1 werden die Nachteile von klassischen Webanwendungen erläutert, die dann in der in Kapitel 1.2 erläuterten Motivation für den Einsatz von Ajax resultieren.

In Kapitel 2 wird das eigentliche Konzept von Ajax vorgestellt, sowie die eingesetzten Technologien und deren Zusammenwirken beschrieben. Kapitel 3 wird dem Anspruch der Anwendung im Dokumententitel gerecht und beschreibt ausführlich den Aufbau und den Ablauf einer exemplarischen Ajax Applikation. Im vierten Kapitel werden die Vor- und Nachteile, sowie der Einsatz im Semesterprojekt evaluiert, Kapitel 5 bietet eine Zusammenfassung der wichtigsten Fakten.

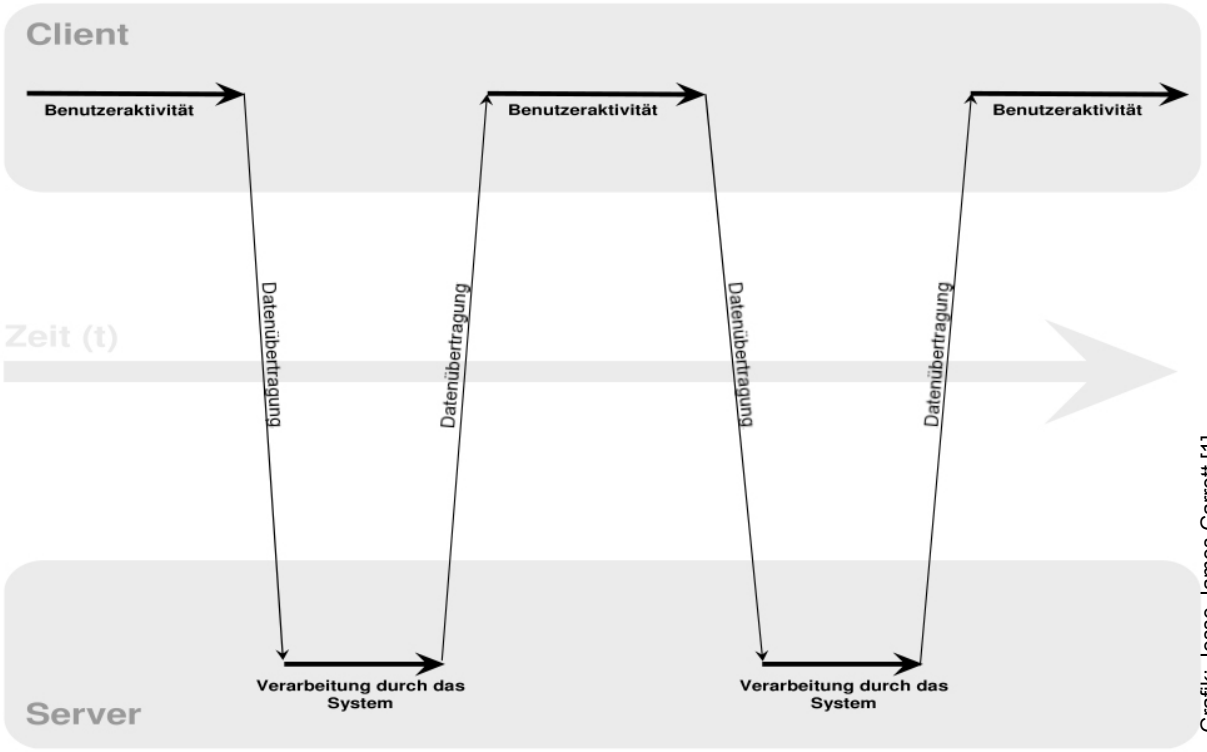
1.1 Nachteile von klassischen Webanwendungen

In Abbildung 1 werden die Aktivitäten auf Client- und Serverseite beim klassischen Modell einer Web-Anwendung dargestellt. Zunächst liegt eine Aktivität auf Clientseite vor: dies kann z.B. das Betrachten einer Webseite sein, gefolgt von einem Aufruf eines Hyperlinks, der in der Webseite angezeigt wird. Dies resultiert in einer Datenübertragung zum Server, durch die dem Server mitgeteilt wird, welche Seite zum Client übertragen werden soll. Der Server wertet die Anfrage aus und übermittelt die angefragte Webseite zum Client.

Dieses Modell beinhaltet mehrere Nachteile: zum einen enthält es den charakteristischen Nachteil synchroner Kommunikation: die Benutzeraktivität ist

unterbrochen, solange der Client auf die Antwort vom Server wartet (siehe unterbrochene Benutzeraktivitätslinie).

Des Weiteren werden bei der Datenübertragung unnötige Daten übermittelt: möchte der Nutzer z.B. auf einer Webseite der Telefonauskunft die Nummer von einem bestimmten Teilnehmer herausfinden, wird neben dieser Information, auch HTML-Code übermittelt, der den Aufbau der Seite beschreibt, die die Information enthält. Die eigentlich benötigte Information ist aber lediglich die Nummer an sich.



Grafik: Jesse James Garrett [1]

Abbildung 1: Klassisches Modell einer Web-Anwendung (synchrone Übertragung)

Auch die Art der Interaktion mit der Web-Anwendung ist für den Benutzer konträr zu einer klassischen Desktop-Anwendung. Bei einer Desktop-Anwendung wie z.B. Microsoft Word kann es zwar zu kurzen Ladezeiten kommen, der Benutzer ist aber prinzipiell die ganze Zeit in der Lage mit der Anwendung zu interagieren.

1.2 Motivation für den Einsatz von Ajax

Aus den vorher beschriebenen Nachteilen von klassischen Webanwendungen resultiert folgende Motivation für den Einsatz von Ajax.

Eine Web-Anwendung soll durch die Interaktion vom Benutzer nicht unterbrochen werden. Auch die intuitive Bedienung einer Webseite vergleichbar mit der einer Desktop-Anwendung ist wünschenswert. Des Weiteren sollen nur aktuell benötigte Daten sukzessive übertragen werden.

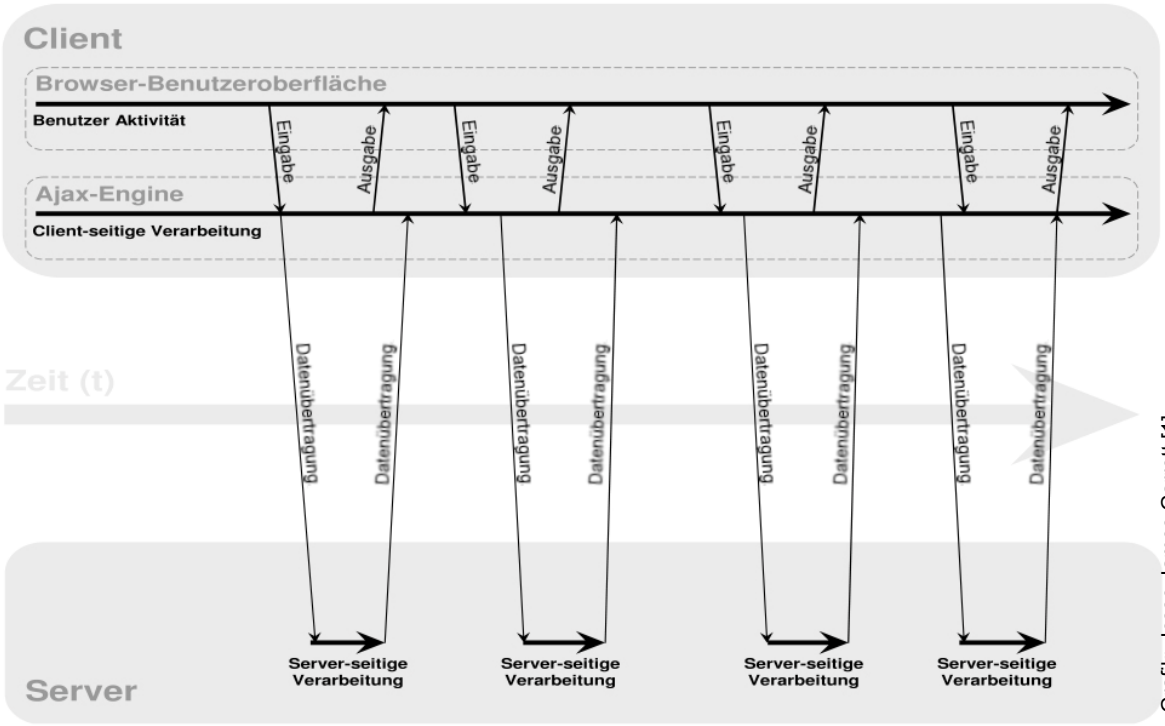
Diese Anforderungen können mit Techniken wie Macromedia Flash oder Director erfüllt werden, dennoch haben sie einen Nachteil: sie benötigen ein proprietäres

Plugin. Nur wenn das Plugin für die jeweils genutzte Plattform vorhanden und installiert ist, kann die Web-Anwendung genutzt werden.

2 Ajax – Das Konzept

AJAX steht für „Asynchronous Javascript and XML“ und bezeichnet ein Konzept, das durch die Verwendung mehrerer, existierender Technologien realisiert werden kann. Die eingesetzten Technologien und deren Zusammenwirken werden im letzten Teil dieses Kapitels genauer beschrieben.

Zunächst wird die prinzipielle Funktionsweise einer AJAX Web-Anwendung anhand der Aktivitäten auf Client- und Serverseite beschrieben. Diese werden in Abbildung 2 veranschaulicht.



Grafik: Jesse James Garrett [1]

Abbildung 2: Ajax Web-Anwendung (asynchrone Übertragung)

Der Client setzt sich hier aus Browserbenutzeroberfläche und Ajax-Engine zusammen. Die Interaktion des Benutzers mit der Benutzeroberfläche des Browsers resultiert zunächst einmal in einer Kommunikation mit der Ajax-Engine. Die Ajax-Engine kommuniziert daraufhin mit dem Webserver und fordert sukzessiv die aktuell benötigten Daten an und gibt sie nach Erhalt an den Browser weiter. Doch schon vor dem Empfang einer Antwort vom Server kann die Ajax-Engine dem Benutzer eine Rückmeldung auf seine Anfrage ausgeben. Dadurch wird dem Benutzer das Gefühl vermittelt er würde mit einer Desktop-Anwendung arbeiten, anstatt mit einer klassischen Web-Anwendung.

Wie sich dieses Konzept für den Benutzer darstellt wird anhand eines kurzen Beispiels erläutert: der Outlook-Webmailer von Microsoft basiert auf dem Ajax-Konzept. Öffnet zum Beispiel ein Benutzer eine Email in seinem Posteingang geschieht folgendes: die Ajax Engine fordert beim Webserver den Inhalt der entsprechenden Email an. Doch noch bevor die Antwort auf diese Anfrage eingetroffen ist, öffnet die Ajax-Engine das Fenster für die Detailansicht einer einzelnen Email. Sobald dann die Antwort mit dem Email-Body vom Webserver eingetroffen ist, wird der Inhalt der Email im zuvor geöffneten Fenster angezeigt. Dadurch entsteht für den Benutzer der Eindruck einer “durchgängigen“ Anwendung.

Dieses Konzept wird durch den prinzipiellen Vorteil der asynchronen Kommunikation ermöglicht: im Gegensatz zur synchronen Kommunikation, kann bei der asynchronen Kommunikation die Entität, die eine Anfrage gestellt hat, weiterarbeiten, bis die Antwort auf die Anfrage eingetroffen ist. Dies lässt sich auch in Abbildung 2 erkennen. Noch bevor die Ajax-Engine die Antwort auf ihre Anfrage zum Server erhalten hat, gibt sie schon eine Ausgabe an den Browser.

2.1 Verwendete Technologien

Wie bereits am Anfang des Kapitels erwähnt werden bei Ajax mehrere existierende Technologien eingesetzt. Diese werden zunächst im Einzelnen erläutert, danach wird ihr Zusammenspiel im Kontext von Ajax beschrieben.

2.1.1 HTML + CSS

HTML und CSS bestimmen bei Ajax das letztendliche Erscheinungsbild der Webseite. Cascading Style Sheets (CSS) werden benutzt um den Inhalt und den Stil von Webseiten zu separieren. Durch den Einsatz von CSS müssen Informationen wie z.B. die Hintergrundfarbe von Webseiten lediglich zentral in der CSS Datei geändert werden und können dann auf alle Webseiten, deren Stil durch die bearbeitete CSS Datei beschrieben wird, angewandt werden.

2.1.2 DOM

DOM steht für Document Object Model und beschreibt die Gliederung von strukturierten Dokumenten wie HTML oder XML. Dabei wird die Struktur speicherintern als Baumstruktur (mit Blättern und Knoten) abgebildet. Des Weiteren kann man eine bestehende DOM-Struktur mittels Javascript manipulieren und dadurch z.B. neue Knoten einfügen oder den Inhalt bestehender Knoten ändern. Die derartige Änderung von z.B. gerade angezeigten HTML-Dokumenten erfolgt transparent für den Nutzer und wird unmittelbar dargestellt.

2.1.3 XMLHttpRequest

Unter XMLHttpRequest versteht man eine Anwendungsschnittstelle (API) zum Datenaustausch zwischen Webbrowser und Webserver. Sie kann als Objekt im Browser durch Verwendung von Javascript exemplarisiert werden und kommuniziert asynchron mit dem Webserver. Dadurch ist der Browser, während er auf eine

Antwort vom Webserver wartet, nicht blockiert. Die Antwort vom Webserver auf eine XMLHttpRequest Anfrage kann, wie der Name schon suggeriert, als eine XML-Datei strukturiert werden. Ist dies der Fall, können in dieser Datei strukturierte DOM-Elemente abgelegt werden.

2.1.4 Javascript

Wie schon bei vorher erläuterten Technologien erwähnt, kommt Javascript zum Einsatz. Zum einen wird Javascript benutzt um das XMLHttpRequest Objekt zu exemplarisieren, zum anderen wird es zum Auswerten der XMLHttpRequest Rückgabe benutzt und der ggf. daraus resultierenden Manipulation der DOM Struktur.

Javascript ist eine so genannte “cross-platform scripting language“. Dies bedeutet, dass Javascript auf den verschiedensten Plattformen eingesetzt werden kann und prozessor- und betriebssystemunabhängig ist.

2.1.5 Serverkomponenten

Die bisher beschriebenen Technologien konzentrieren sich alle auf die Clientseite. Auf der Serverseite muss allerdings eine Komponente vorhanden sein, die die Anfrage entgegen nimmt, bearbeitet und die Antwort übermittelt.

Bei Ajax gibt es keine konkrete Vorgabe, in welcher Sprache oder mit welchem Framework diese Komponente erstellt wurde. Daher ist es möglich diese Komponente u.a. als Enterprise Java Bean, Mircosoft .NET Komponente, Ruby Script Komponente oder unter Benutzung von PHP zu erstellen.

2.2 Aufbau einer Ajax-Anwendung

In Abbildung 3 ist der Aufbau einer Ajax Web-Anwendung dargestellt. Bei der Benutzung dieser Anwendung kommen die zuvor vorgestellten Technologien folgendermaßen zum Einsatz. Nehmen wir an, der Nutzer navigiert über eine Webseite und drückt einen Button, was in einer Serveranfrage resultiert.

Die bereits vorhandene Webseite ist im Speicher des Webbrowsers als DOM-Struktur abgebildet. Die Interaktion mit der Benutzeroberfläche (das Drücken des Buttons) erzeugt einen Javascript Aufruf, der an die Ajax-Engine weitergeleitet wird. Unter der Ajax-Engine kann man sich einige Javascript Methoden vorstellen, die die Anfragen von der Benutzeroberfläche entgegennehmen und daraufhin das XMLHttpRequest Objekt mit den entsprechenden Parametern erzeugen. Diese Parameter spezifizieren u.a. welche Komponente auf Serverseite die Anfrage bearbeiten soll. Gleichzeitig wird ein Callbackhandler erzeugt, der später die Antwort vom Webserver entgegen nimmt.

Danach wird die Anfrage über das Http Protokoll (z.B. in Form von strukturiertem Text) zum Server übermittelt. Dort bearbeitet die vorher spezifizierte Komponente die Anfrage. Zur Bearbeitung der Anfrage sind ggf. Zugriffe auf den Speicher oder andere Komponenten notwendig. Die Antwort auf die Anfrage wird als XML Datei strukturiert und zurück an den Client übermittelt.

Dort nimmt der Callbackhandler die Antwort entgegen und wertet sie aus. Kommt es zu einer Änderung der HTML-Struktur, geschieht dies über die Javascript API der DOM-Struktur.

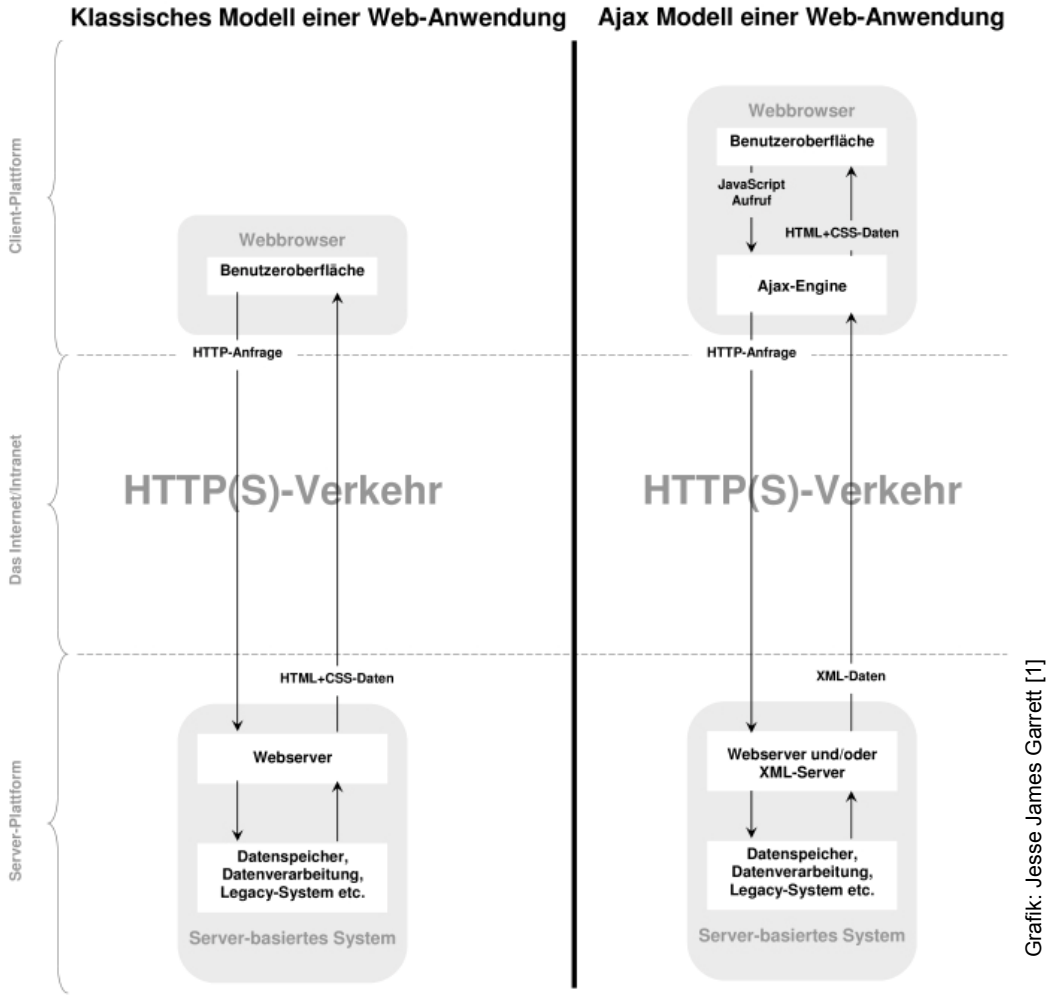


Abbildung 3: Aufbau von Web-Anwendungen

Im Vergleich zum Aufbau der klassischen Web-Anwendung in Abbildung 3 fällt auf, dass bei Ajax nur die relevanten Informationen übertragen werden und daraufhin die bestehende HTML-Struktur dynamisch geändert wird. Bei der klassischen Web-Anwendung resultiert jede Interaktion in einer kompletten Neuübermittlung des HTML-Codes.

3 Verwendung von Ajax

Neben dem Konzept wird in diesem Bericht auch die Verwendung von Ajax erläutert. Dies geschieht anhand eines konkreten Beispiels: im Kontext zur Lehrveranstaltung, in der dieser Bericht entstanden ist, wird nun demonstriert, wie

man mit unter Zuhilfenahme von Ajax beim Abspielen eines Videostreams Zusatzinformationen passend zur aktuellen Videoposition einblenden kann. Der komplette Quellcode zu diesem Beispiel findet sich unter [9].

3.1 Aufbau des Demonstrations-System

In Abbildung 4 wird der Aufbau des Systems erläutert. Auf der Serverseite sind vier Objekte vorhanden: eine HTML Datei (Ajax_Demo.html), eine Javascript Datei (ajax.js), ein Realmedia Videostream (Video.rm) und eine PHP Komponente (text_query.php).

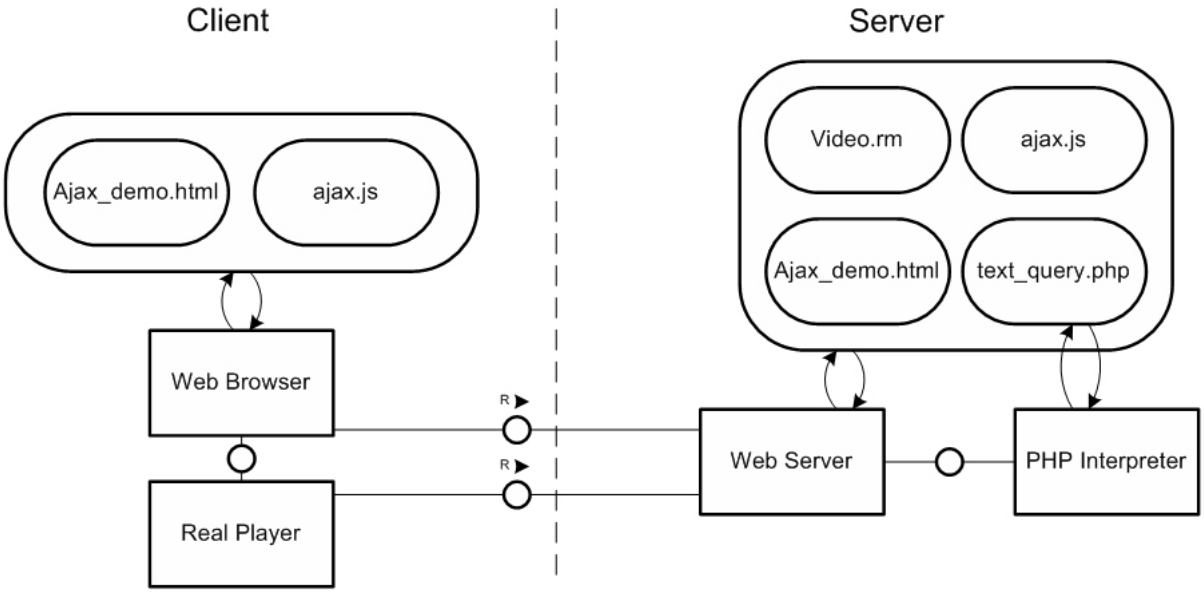


Abbildung 4: Aufbau des Demonstrations-Systems

Ruft der Client die HTML Seite auf, wird automatisch die Javascript Datei mit übermittelt. Über die Javascript Methoden wird die Wiedergabe des Videostreams kontrolliert.

Die PHP Komponente übernimmt hier die Rolle der Serverkomponente, die die Anfragen auswertet.

3.2 Ablauf des Demonstrations-System

In Abbildung 5 wird der Ablauf des Systems gezeigt. Die Ajax_Demo.html Datei sowie die Ajax.js Datei wurden zuvor vom Server auf den Client übertragen und werden dort ausgeführt.

Drückt der User auf der HTML Seite einen Button während der Videostream Wiedergabe, resultiert dies im Aufruf der talktoServer Methode. Wie im

nachfolgenden Listing 1 zu erkennen, wird zunächst ein neues XMLHttpRequest Objekt erzeugt.

Dies geschieht durch den Aufruf der Methode newXMLHttpRequest, siehe Listing 2. In dieser Methode wird überprüft, welcher Browser der Client verwendet. Wird ein Mozilla oder Firebird Browser verwendet, wird das XMLHttpRequest Objekt direkt exemplarisiert. Wird der Internet Explorer von Mircosoft benutzt, erfolgt die Exemplarisierung über ActiveX.

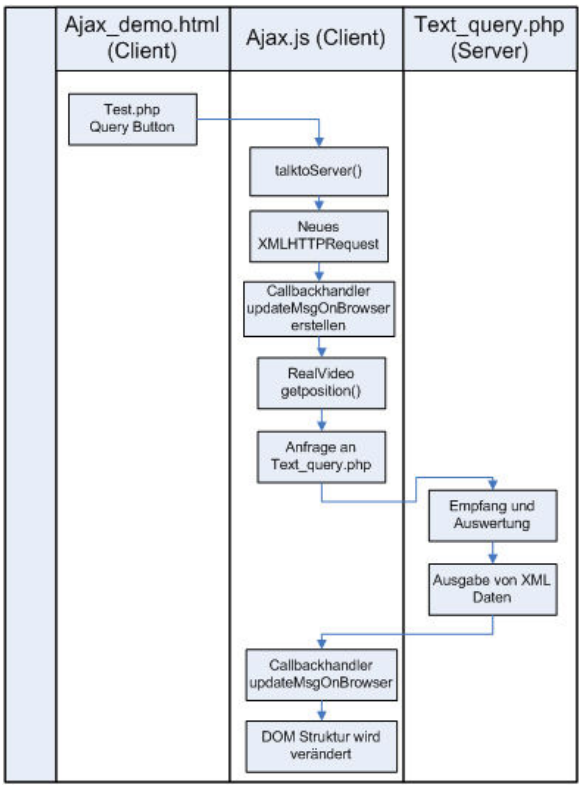


Abbildung 5: Ablauf des Demonstrations-Systems

Nachdem das XMLHttpRequest Objekt erzeugt worden ist, wird ihm in der talktoServer Methode ein Callbackhandler zugewiesen. Dieser ist in diesem Fall die Methode updateMsgOnBrowser, die aufgerufen wird mit der Antwort vom Webserver als Übergabeparameter. Danach wird das XMLHttpRequest geöffnet und die Komponente auf der Serverseite, die die Anfrage bearbeiten soll wird bestimmt (in diesem Fall text_query.php). Nun wird die aktuelle Position vom Videostream ermittelt (document.javademo.GetPosition) und zum Webserver übermittelt.

Die Text_query PHP-Komponente wertet nun die Anfrage aus und schreibt einen zur übergebenen Textmarke vorher definierten String in eine XML-Datei. Diese wird zurück zum Client übermittelt.

Auf der Clientseite nimmt nun die updateMsgOnBrowser Methode die XML-Datei entgegen und parst den String in den DOM-Baum, der die dargestellte Webseite enthält.

```
function talktoServer(){
 var req = newXMLHttpRequest();
 var callbackHandler = getReadyStateHandler(req,updateMsgOnBrowser);
   req.open("POST", "text_query.php", true);
   req.send("msg="+document.javademo.GetPosition());
}
```

Listing 1: Ajax.js : talktoServer Methode

```
function newXMLHttpRequest() {
 var xmlreq = false;
 if (window.XMLHttpRequest) {
   xmlreq = new XMLHttpRequest();
 } else if (window.ActiveXObject) {
       xmlreq = new ActiveXObject("Msxml2.XMLHTTP");
   }  return xmlreq;
 }
```

Listing 2: Ajax.js : newXMLHttpRequest Methode

4 Vor- und Nachteile von Ajax

Die Vorteile von Ajax wurden bereits in Kapitel 1 mit der Motivation für den Einsatz von Ajax vorgestellt und liegen klar auf der Hand: Ajax ermöglicht im Gegensatz zu klassischen Web-Anwendungen eine wesentlich effizientere Datenübertragung, da nur aktuell benötigte Daten sukzessiv übertragen werden. Dies resultiert in einer besseren Performance auf der Clientseite, sowie einer “unterbrechungsfreien“ Handhabung. Darüber hinaus basiert Ajax auf mehreren existierenden Technologien, wodurch Ajax Web-Anwendungen mit modernen Browsern, ohne die Installation von zusätzlichen Plugins, genutzt werden können.

Die Tatsache, dass Ajax auf mehreren existierenden Technologien basiert, kann aber auch als Nachteil verstanden werden: möchte man eine professionelle Ajax Anwendung erstellen, braucht man viele Domain Experten. Die bessere Performance auf der Clientseite geht einher mit einer schlechteren Performance auf der Serverseite. Bei Ajax werden viele kurze Verbindungen aufgebaut: dies hat zur Folge, wenn sehr viele User eine Ajax Anwendung nutzen, dass der Server wesentlich mehr Verbindungsanfragen bearbeiten muss, als bei gleicher Nutzeranzahl und einer klassischen Webanwendung. Der nächste Nachteil von Ajax besteht darin, dass Suchmaschinen die Informationen, die über Ajax bereitgestellt werden, nicht automatisch indizieren können. Da hier die Informationen in Datenbanken oder XML-Dateien gespeichert werden und erst auf spezielle Anfrage oder nach mehreren Benutzerschritten auf einer Webseite angezeigt werden, können die Roboter von Suchmaschinen sie nicht so einfach auslesen wie statische

HTML-Seiten. Auch wenn das veränderte Bedienkonzept von Ajax durchaus Vorteile besitzt, hat es auch einen gravierenden Nachteil: es widerspricht dem klassischen Bedienkonzept von Webanwendungen. Da es sich hier um zustandslose Web-Anwendungen handelt, können nicht ohne weiteres Bedienelemente wie die Vor- und Zurückbuttons genutzt werden, das Erstellen von Lesezeichen ist nicht möglich. Zwar gibt es mittlerweile Workarounds, die künstlich den Zustand einer Ajax-Webandwendung speichern, indem sie so genannte Waypoints erstellen: dennoch erfordern solche Mechanismen die Erstellung individueller Bedienelemente und können nicht mit dem gewohnten Buttons bedient werden.

Auch wenn ein Blick auf die eingesetzten Technologien suggeriert, dass Ajax absolut plattformunabhängig ist, hat sich bei der praktischen Verwendung in Kapitel 3 folgendes gezeigt: es muss explizit abgefragt werden, wie das XMLHttpRequest Objekt erstellt wird bzw. welcher Browser verwendet wird. Dies bedeutet, dass die Web-Anwendung nur mit den Browsern genutzt werden kann, die explizit abgefragt bzw. unterstützt werden.

4.1 Verwendung von Ajax im Semesterprojekt

Prinzipiell ist Ajax prädestiniert für Web-Anwendungen, die ein hohes Maß an Userinteraktion erfordern. Dies ist bei der Wiedergabe von Videostreams i.d.R. nicht der Fall, da es sich hier eher um eine passive Applikation handelt.

Allerdings wurde durch die Demonstration in Kapitel 3 bewiesen, dass es durchaus Features in diesem Projekt gibt, bei denen der Einsatz von Ajax sinnvoll ist. Der zentrale Vorteil von Ajax in diesem Kontext liegt darin, dass Seiteninhalte geändert werden können, ohne dass der Videostream unterbrochen werden muss. Dieser Vorteil kann bei der Implementierung folgender Funktionen ausgenutzt werden:

Wie schon bereits in Kapitel 3 gezeigt, können Zusatzinformationen in Relation zur aktuellen Videoposition angezeigt werden. Diese Funktion könnte man dahingehend erweiterten, dass man die Datenübermittlung per XMLHttpRequest zum Server dafür nutzt, dass auch Informationen oder Kommentare vom Nutzer beim Betrachten des Videos eingegeben werden. Diese Notizen könnten dann wahlweise nur dem Nutzer selber oder allen weiteren Betrachtern zur Verfügung gestellt werden.

Auch könnte Ajax dazu genutzt werden, dass während der Vorlesung Übungsaufgaben gelöst werden. Dabei könnten die Aufgaben und die Lösungen vom Nutzer per Ajax zum Server übermittelt und dort ausgewertet werden. Hat der Nutzer eine Aufgabe nicht korrekt gelöst, wird automatisch der Teil der Vorlesung wiederholt, indem das relevante Konzept erläutert wird.

Doch auch Anwendungsmöglichkeiten in Bezug auf das Erscheinungsbild sind möglich. Kontrolliert man das Frame, indem das Realvideo abgespielt wird, mit Ajax wären durch den Aufruf entsprechender Javascriptmethoden Größen- und Positionsänderungen des Videos möglich.

5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass Ajax ein Konzept mit einem hohen Potential ist. Es verbessert die Usability von Web-Anwendungen und sorgt für eine effizientere Datenübertragung, da lediglich benötige Daten sukzessiv übertragen werden. Ajax Anwendungen lassen sich von den allermeisten Usern benutzen, da lediglich ein aktueller Browser benötigt wird, der die eingesetzten Technologien unterstützt. Der Hauptaspekt von Ajax ist die asynchrone Kommunikation, die über die XMLHttpRequest API realisiert wird.

Momentan finden sich in vielen Fachzeitschriften Artikel über Ajax und Ajax ist eines der aktuellen Buzzwords im Bereich Webanwendungen. Unter Berücksichtigung der in Kapitel 4 vorgestellten Vor- und Nachteile bleibt abzuwarten, ob Ajax eine große Akzeptanz und Verbreitung erreicht oder ob es sich im Nachhinein als Hype entpuppt.

6 Referenzen

[1] Jesse James Garett, Ajax: A New Approach to Web Applications, adaptive path publications, February 2005
[2] Linda Dailey Paulson, Building Rich Web Applications, IEEE Computer Society, October 2005
[3] Dion Hinchcliff, 5 Earth-Shattering Things You Should Know About Ajax, SOAWebServices Jounral, October 2005
[4] Jeffery Veen, The Buisness Value of Web Standards, adaptive path publications, September 2003
[5] AJAX Hello World, http://www.hackorama.com/ajax
[6] Andrew S. Tanenbaum, Computer Networks. Prentice Hall, second edition, 1988
[7] Microsoft Outlook Web Access, http://www.microsoft.com/exchange
[8] Stefan Mintert, Ajax: die nächste Generation der Web-Andwendungen, iX – Magazin für professionelle Informationstechnik, November 2005
[9] Quellcode unter http://myhpi.de/~chrtinne/php/php.zip