

# Instant Messaging per AJAX, Analyse und Vergleich

Daniel Leese

## Zusammenfassung

Ziel dieser Seminararbeit, die im Rahmen der Veranstaltung SOA & Web2.0 bei Prof. Dr. Reich erstellt wurde, ist die Auseinandersetzung mit modernen Internet-technologien, insbesondere dem sogenannten Web2.0. Zu diesem Zweck wurden browserbasierte Instant Messaging Dienste, welche die AJAX Technologie verwenden, getestet und untersucht. Zum veranschaulichen dieser Technologien wurde auch eine rudimentäre Instant Messaging Applikation entwickelt und analysiert.

Die Arbeit soll hierbei die Grundlagen von AJAX für Web2.0 Einsteiger, wie zum Beispiel Studenten der Informatik und verwandter Disziplinen, die zwar über fundierte Computerkenntnisse verfügen, sich mit aktuellen Entwicklungen im Internet jedoch noch nicht befasst haben, begreifbar machen und in verständlicher Form darstellen.

Kein Bestandteil der Seminararbeit sind die, sicherlich äußerst wichtigen, Designaspekte des Web2.0. (In-)usability, Farbverläufe, "Tag Clouds" und Animationen werden anderen Autoren überlassen. Es wird auch nicht zu tief in die Materie eingestiegen da dies eher hinderlich für die Verständlichkeit wäre und auch den Rahmen des Seminars sprengen würde.

## Index Terms

Web2.0, AJAX, JavaScript, PHP, DOM, ICQ.



## 1 EINFÜHRUNG

**T**EXTE die sich mit Web2.0 oder AJAX befassen gibt es, da AJAX einer der momentanen Trends in der Webentwicklung ist, wie Sand am Meer<sup>1</sup>. Jedoch sind viele dieser Werke sehr komplex und für den Einstieg in das Thema eher abschreckend. Andere Werke sind eher wie Marketingpräsentationen geschrieben und schwärem von AJAX in den höchsten Tönen und preisen es als immer und überall einzusetzendes Heilmittel für alle Probleme des World Wide Web.

- 
- Daniel Leese, Student der Hochschule Furtwangen, Studiengang Computer Networking, 3. Semester  
E-mail: leese@hs-furtwangen.de

1. Vgl. <http://www.amazon.com/gp/bestellers/books/379359011>

Als nun in der Vorlesung SOA & Web2.0 bekannt gegeben wurde, das im Rahmen dieser Veranstaltung eine Arbeit aus dem oben genannten Themenkreis verfasst werden kann wollte der Autor diese Gelegenheit nutzen über ein praktisches Beispiel für den sinnvollen Einsatz von AJAX zu schreiben. Instant Messaging erschien für diesen Zweck ideal da es nicht deterministische Ereignisse, wie z.B. einen Nachrichteneingang, beinhaltet und Kommunikation in beide Richtungen, d.h. vom Client zum Server und umgekehrt möglich sein muss. Nach Meinung des Autors ist dieses Beispiel auch einer der wenigen Fälle bei denen AJAX bzw. ereignisbasierte Kommunikation zwingend notwendig ist.

Somit sollte sich diese Arbeit von den vielen anderen, bei denen AJAX nur zum Selbstzweck, nämlich modern zu sein und einfach "AJAX zu haben", eingesetzt wird, abheben.

## **1.1 Die Entwicklung des Web**

Um zu sehen wie sich AJAX von bisherigen Webtechnologien unterscheidet folgt eine Abgrenzung der einzelnen Entwicklungsschritte und Konzepte. Hierfür eignet sich gut die geschichtliche Entwicklung des Web.

### *1.1.1 Statisches Web*

Als statisch bezeichnet man im Bezug auf Internetseiten solche Seiten die einmal mit einem Texteditor unter Verwendung der HTML Seitenbeschreibungssprache erstellt werden und so auf einem Webserver abgelegt werden. Eine Manipulation oder Ergänzung der Seiten durch User ist nicht möglich, und auch der Autor der Webseite muss die Seite erneut mit dem Texteditor bearbeiten um Änderungen vorzunehmen.

Dies hat zur Folge das solche Seite nur umständlich zu warten oder aktualisieren sind und somit häufig schnell veralten, bzw. nur zur Darstellung von Informationen geeignet sind. Dynamische Prozesse, Kollaboration oder Interaktive Anwendungen sind mit dieser Technologie nicht möglich.

### *1.1.2 Datenbanken und Serverskripte*

Der nächste große Schritt in der Entwicklung des Webs waren Seiten mit Serverscripten (z.B.: PHP, JSP, ...) und Datenbankanbindung (z.B.: zu einer MySQL Datenbank). Solche Seiten besitzen schon eine gewisse Interaktivität, so können sie im Webbrowser "editiert" werden in dem z.B. ein neuer Artikel hinzugefügt wird.

Dies geschieht in der Regel über sogenannte Formulare. Diese Formulare sind jedoch nicht interaktiv benutzbar und werden, wenn sie ausgefüllt sind komplett dem Webserver zur Verarbeitung übergeben. Dieser Verarbeitet die eingegebenen Daten dann und speichert sie gegebenenfalls in der Datenbank ab. Nun steht bei erneutem Zugriff die jedes mal neu per Script aus der Datenbank generierte Seite zu Verfügung.

### 1.1.3 Interaktive, dynamische Webanwendungen

Die aktuelle "Ausbaustufe" des Web wird, ob ernsthaft oder manchmal eher lakonisch, Web2.0 genannt. Hierbei werden auch die bekannten Technologien wie Datenbanken oder Serverscripte verwendet, jedoch ergänzt um neue Konzepte wie z.B. AJAX.

Solche Seiten zeichnen sich durch dynamisches Reagieren auf Benutzereingaben aus. So könnte z.B. ein Benutzer der ein Webformular editiert nach jedem ausgefüllten Feld sofort angezeigt bekommen ob die Eingabe in dieses Feld korrekt war, und das ohne das die Webseite neu geladen werden muss.

## 1.2 AJAX Grundlagen

Um die im vorherigen Kapitel angesprochene Interaktion mit dem Benutzer zu erreichen wird nahezu ausschliesslich AJAX (Asynchronous JavaScript and XML) verwendet. Bei AJAX handelt es sich um ein neues Konzept zur Verwendung altbekannter Technologien wie dem zuvor oftmals als eher nutzlose Spielerei abgetanen JavaScript und nicht um eine neue Programmiersprache oder ähnliches.

Folgende Technologien kommen bei AJAX zum Einsatz<sup>2</sup>:

- HTML/CSS
- DOM
- JavaScript
- XMLHttpRequest-Objekt

Hierbei wird HTML/CSS wie bisher zur Seitenbeschreibung verwendet, also für Textformatierung, Bereiche und Hyperlinks. Diese mit HTML formatierten Elemente der Webseite werden mittels JavaScript über das DOM (Document Object Model) adressiert und manipuliert. Solche Manipulationen werden vom Webbrowser dynamisch vorgenommen, die Seite ändert sich somit direkt vor den Augen des Benutzers und nicht wie bisher erst nach einem erneuten Anfordern und Laden der Seite.

Für die Kommunikation mit dem Webserver ist das XMLHttpRequest Objekt zuständig. Dieses kann aus JavaScript aufgerufen werden und Daten, z.B. mit XML formatiert, an den Webserver senden. XMLHttpRequest Anfragen werden asynchron verarbeitet, daher muss ein Skript nicht warten, bis die Anfrage beantwortet ist, sondern kann solange andere Aufgaben abarbeiten.

Das XMLHttpRequest Objekt wurde von Microsoft entwickelt und das erste mal in den, 1999 erschienenen, Internet Explorer 5 integriert. Zunächst fand es bei den Webentwicklern keinerlei Beachtung, vermutlich da es zu Beginn nur in Microsoft Browsern zu Verfügung stand. Jedoch wurde es

2. vgl. [http://www.openajax.org/member/wiki/Whitepaper\\_20060730](http://www.openajax.org/member/wiki/Whitepaper_20060730)

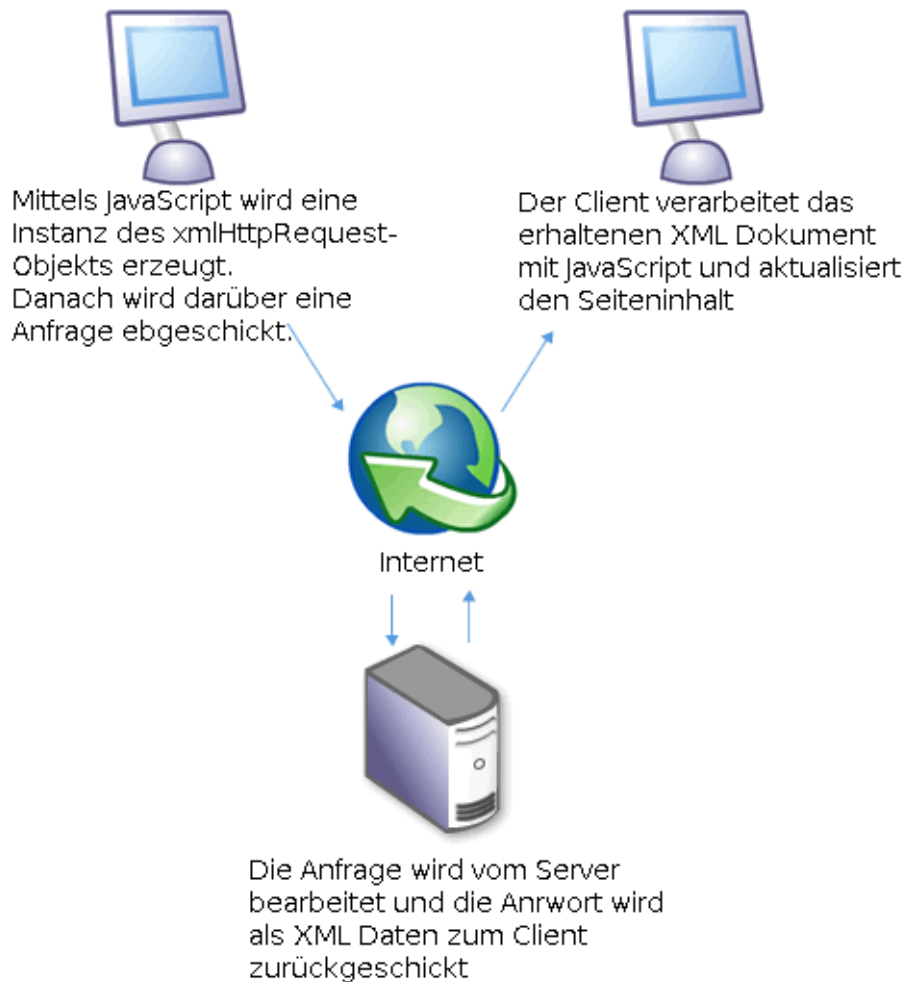


Abbildung 1. Vereinfachte Funktionsweise von AJAX

nach und nach in allen großen Browsern implementiert und z.B. bei Googels Webmail verwendet.

Heutzutage wird das XMLHttpRequest Objekt in der Regel nicht mehr direkt verwendet. Diverse JavaScript Bibliotheken wie z.B.: Prototype, jQuery oder MooTools stellen es gekapselt zu Verfügung. Diese Bibliotheken vereinfachen die "AJAX-Programmierung" sehr und erfreuen sich daher großer Beliebtheit. Daher wurde auch für das Beispiel zu diesem Text die Prototype Bibliothek verwendet.

Richtig populär wurde AJAX jedoch durch den Artikel "Ajax: A New Approach to Web Applications"<sup>3</sup> von Jesse James Garrett aus dem Jahr 2005. Auch war es Garrett der mit diesem Artikel den Begriff AJAX prägte.

3. Originalartikel: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>

## 2 INSTANT MESSAGING

Unter klassischem Instant Messaging versteht man das Versenden von kurzen formlosen Nachrichten unter Bekannten/Freunden mit **1 zu 1** oder **1 zu n** Teilnehmern. Auch wird ständig der Status der einzelnen Benutzer wie z.B. "Verfügbar", "Beschäftigt" oder "Offline" vorgehalten. Somit kann jeder Benutzer eines Instant Messaging Netzwerks sehen welche seiner Kontakte online und ansprechbar sind. Dies ermöglicht es z.B.: wenn ein Teilnehmer spontan etwas unternehmen will auch nur die Kontakte zu adressieren die ebenfalls gerade anwesend und auch bereit für spontane Aktionen sind.

Einige bekannten Instant Messaging Protokolle:

- ICQ/AIM (Sog. Oscar Protokoll)
- MSN
- Jabber (Open Source)
- Google Talk

Im Gegensatz dazu wird eine Kommunikation von n zu n Teilnehmern im Internet als Chat bezeichnet. Jedoch sind hier die Übergänge zum Instant Messaging nicht klar abgegrenzt, so unterstützen manche Instant Messaging Protokolle auch **n zu n** Kommunikation und den fließenden Übergang zwischen den verschiedenen Kommunikationsarten.

### 2.1 Anbieter von browserbasiertem Instant Messaging

#### 2.1.1 Anbieterspezifisch

- Yahoo! Messenger for the Web
- MSN Web Messenger: DHTML
- AIM Express

#### 2.1.2 Multi Messenger

- Meebo  
Bei diesem Anbieter basiert die Implementierung der einzelnen Instant Messaging Protokolle basiert auf der freien libpurple Bibliothek, einer in C geschriebenen Instant Messaging Funktionsammlung für verschiedene Protokolle wie z.B. ICQ/AIM, MSN, Yahoo! Messenger, uvm. Libpurple wird von dem Pdgin Projekt (Ein beliebtes IM Programm auf freien Betriebssystemen, ehemals Gaim) entwickelt und gepflegt. Auch setzt Meebo auf ein sehr verspieltes und komplett durchgestyltes Userinterface. Manche Funktionen, wie z.B. das verschieben der einzelnen Chatfenster, kommen einem Informatiker vielleicht sinnlos vor, doch es hat sich gezeigt das solche Dinge bei vielen Usern einen sehr hohen

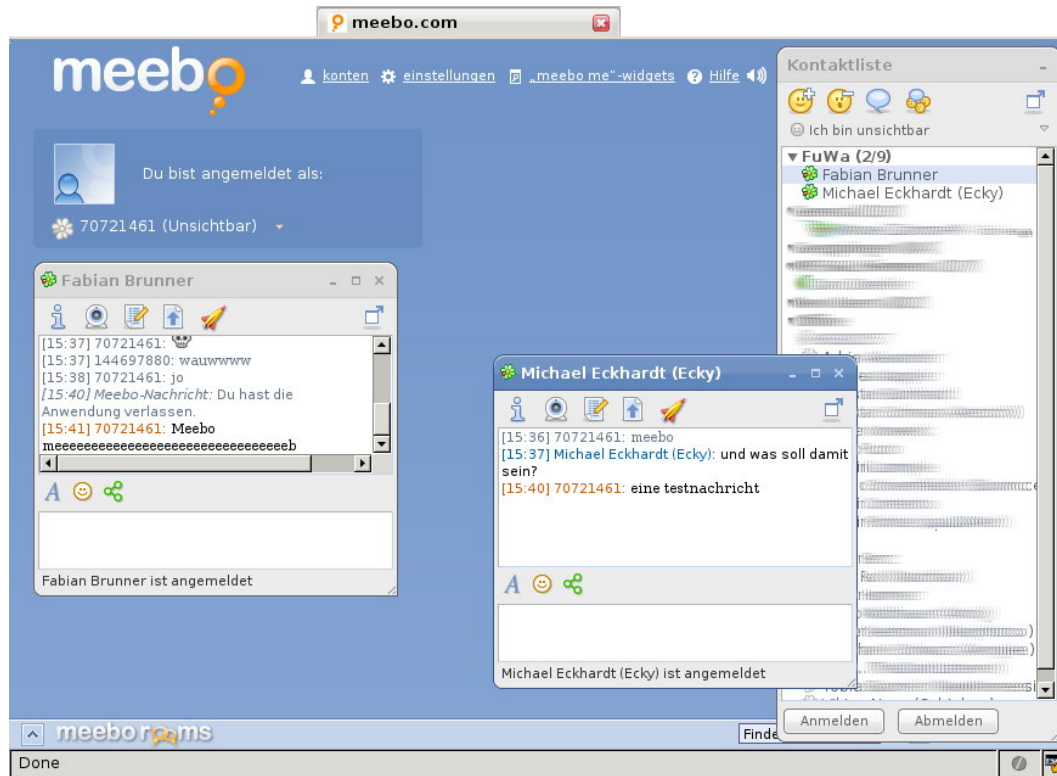


Abbildung 2. Meebo Oberfläche im Webbrowser

Stellenwert einnehmen und oftmals sogar als das entscheidende Alleinstellungsmerkmal wahrgenommen werden. Meebo ist dadurch zumindest ein hervorragendes Beispiel für die desktopartige Funktionalität auf Webseiten, die durch AJAX ermöglicht wird (Siehe Abbildung 2).

- KOOL IM  
Bietet genau die selben IM Netzwerke und GUI-Funktionen wie Meebo, hat aber zusätzlich noch ein sympathisches Logo.

Auch die drei weiteren getesteten Dienste ILoveIM, communicationtube und imo.im unterscheiden sich nur marginal, so daß ich neben der Namensnennung nicht weiter auf sie eingehen werde.

### 3 PRAKTISCHES BEISPIEL

Zum besseren Verständnis der der AJAX Konzepte wurde eine kleine Beispielanwendung geschrieben. Neben den im Kapitel 1.2 erwähnten AJAX Technologien wurde serverseitig PHP auf einem Apache unter Arch Linux

verwendet. Die Instant Messaging Funktionalität stellte die bereits erwähnte libpurple<sup>4</sup> mittels PHP Bindungs<sup>5</sup> bereit.

Die Beispielanwendung ermöglicht nur das Empfangen von ICQ Nachrichten eines Accounts (also kein Multi-Messenger). Dies ist nach Meinung des Autors jedoch völlig ausreichend die zu Grunde liegenden Konzepte zu verstehen.

### 3.1 Aufbau der Beispielanwendung

Die Beispielanwendung besteht aus 5 Dateien (Siehe Abbildung 3)

- In der **index.htm** befindet sich das HTML Grundgerüst der Seite, im wesentlichen ein div-Bereich und ein Button zum Aktualisieren, d.h. um nachzuschauen ob in der Zwischenzeit weitere Nachrichten auf dem Server eingetroffen sind. Hierbei zeigt sich auch eine erste Grenze von AJAX. Es kann zwar der Client mittels XMLHttpRequest Objekt jederzeit Nachrichten an den Server senden, jedoch kann der Server von sich aus keine Kommunikation mit dem Client aufbauen. Dieses Manko wäre z.B. durch Polling lösbar, d.h. der Client müsste von sich aus z.B. alle 3 Sekunden (das messaging soll ja noch "instant" sein) den Server kontaktieren und nachfragen. Dies würde jedoch den Rahmen dieser Seminararbeit sprengen und das Beispiel auch nur unnötig verkomplizieren, daher genügt hier der Aktualisieren-Button.

Die Datei beinhaltet auch noch zwei JavaScript Funktionen zum Abschicken der Anfrage (**hol\_datei()**) nach neuen Nachrichten und zum Darstellen der Antwort (**zeige\_datei(originalRequest)**) in dem o.g. div-Bereich mittels DOM.

- Das **phurple.php** Script wird permanent auf dem Server ausgeführt und implementiert die libpurple Bindings. Mit Hilfe der von libpurple bereitgestellten Funktionen wird ein Testuser angemeldet und eintreffende Nachrichten werden entgegengenommen. Darauf werden sie in eine Textdatei geschrieben.
- Bei **icq.tmp** handelt es sich um die gerade erwähnte Textdatei. Die Nachrichten werden Zeilenweise (eine Zeile pro Nachricht) im Format:  
**(NummerDesSenders) (Uhrzeit) AliasDesSenders: Nachrichtentext**  
 abgelegt. Somit ist gewährleistet das auch bei Programmabstürzen keine Nachrichten verlorengehen und auch komplizierte Interprozesskommunikation entfällt.

4. <http://developer.pidgin.im/wiki/WhatIsLibpurple>

5. <http://sourceforge.net/projects/phpurple>

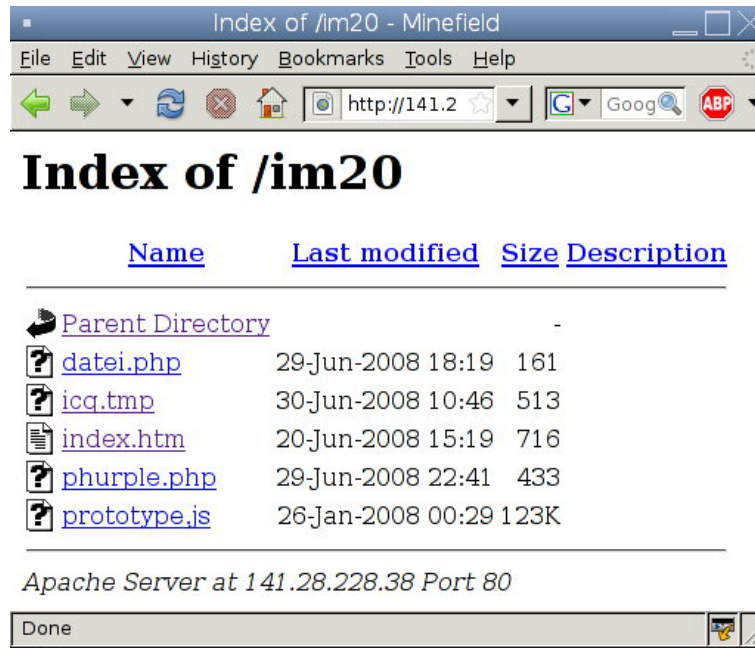


Abbildung 3. Dateien der Beispielanwendung

- Die **datei.php** wird nach einem Klick auf den Aktualisieren Button von der JavaScript Funktion **hol\_datei** über **Ajax.Request** aufgerufen und gibt den Inhalt der **icq.tmp** zurück.
- **prototype.js** ist eine weitverbreitete JavaScript Bibliothek die viele, gerade in Verbindung mit AJAX, oft gebrauchten Funktionen von JavaScript kapselt. Sie ist in der **index.htm** inkludiert und stellt auch das verwendete Ajax.Request zu Verfügung.

### 3.2 Ablauf einer Anfrage

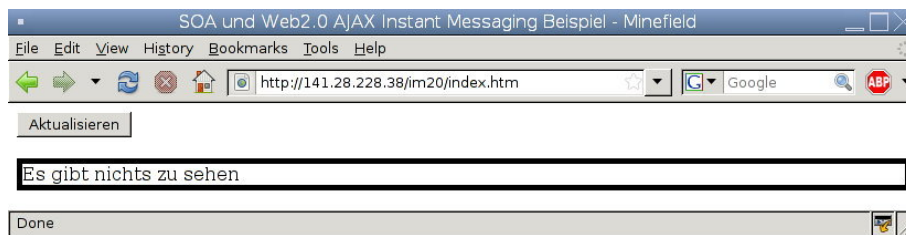


Abbildung 4. Seite im Urzustand

Nach dem aufrufen der **index.htm** präsentiert sich die Seite dem User recht nüchtern (Siehe Abbildung 4) Schwarz umrandet ist der **div**-Bereich

zu erkennen der nach dem Druck auf den Aktualisieren Button die aktuellen Nachrichten aufnimmt.

Wird nur Aktualisieren gedrückt tritt das onClick Ereignis auf und die `hol_datei()` JavaScript Funktion wird aufgerufen. Diese sendet den von `prototype.js` bereitgestellten `Ajax.Request` an den Server der die Ausgabe von `datei.php` zurückgibt. Ist dies geschehen tritt das `onComplete` Ereignis auf und die `zeige_datei()` Funktion wird aufgerufen. Diese Adressiert den `div`-Bereich mittels `Document.getElementById` (über das DOM) und gibt die zurückgegebenen Nachrichten aus (vgl. Abbildung 5).

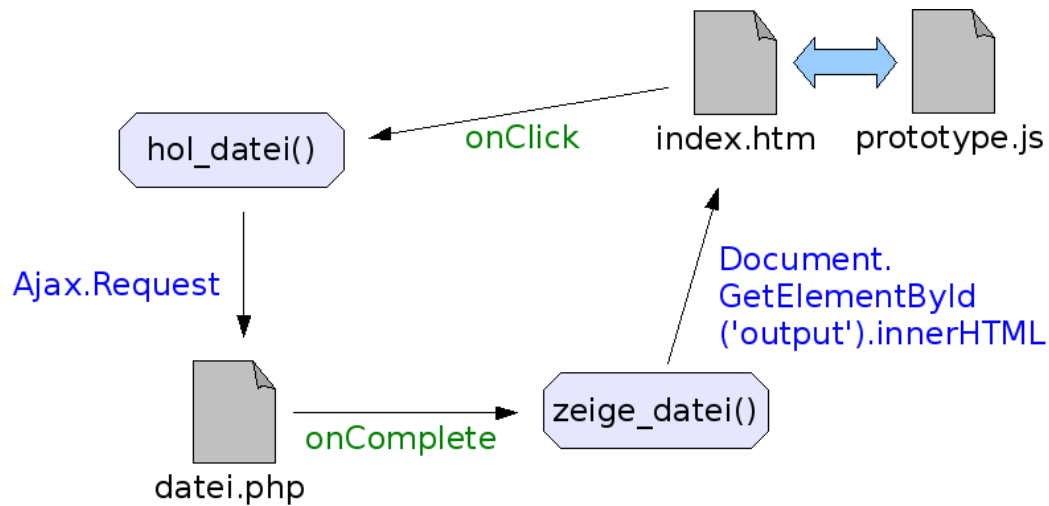


Abbildung 5. Ablauf einer Anfrage

Nun ist der aktuelle Nachrichtenverlauf in dem o.g. `div`-Bereich zu sehen (Abbildung 6).

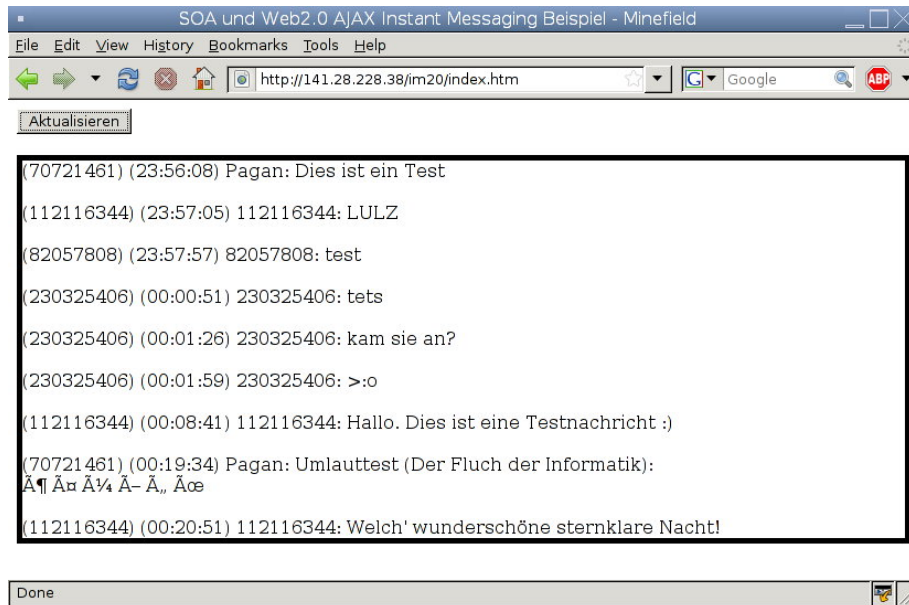


Abbildung 6. Ablauf einer Anfrage

## 4 ZUSAMMENFASSUNG

Das bereitstellen eines webbasierten Instant Messaging Dienstes ist heutzutage, Dank der vielen Bibliotheken wie libpurple, phpurple, prototype und starker Scriptsprachen wie PHP kein großes Problem mehr. Dies zeigt sich auch an den vielen Anbietern mit größtenteils nur marginalen Unterschieden. Dies ist wohl auch der Grund für die, nach Meinung des Autors, völlig übertriebenen Designs und Usability-Features der Anbieter, da diese so wohl versuchen Akzente zu setzen und sich hervorzuheben.

Auch ist es interessant das AJAX streng genommen nur eine Richtung des Instant Messaging, nämlich das Versenden (also von Client zu Server) direkt unterstützt und das Empfangen gerade eingetreffener nachrichten "Just-in-time" nicht ohne weiteres möglich ist (vgl. Kapitel 3.1).

Abschliessend ist festzuhalten das AJAX eine faszinierende, und für Informatiker einfach zu erlernende, Technologie ist die Internetseiten entscheidend bereichern kann bzw. manche Dienste erst möglich macht.

## ANHANG A QUELLTEXTE

### A.1 index.html

```
<html>
  <head>
    <script src="prototype.js" type="text/javascript"></script>
    <script type="text/javascript" language="JavaScript">
      function hol_datei() {
        var myAjax = new Ajax.Request(
          "datei.php",
          { method: 'get', onComplete: zeige_datei }
        );
      }

      function zeige_datei( originalRequest ) {
        document.getElementById('output').innerHTML = originalRequest.responseText;
      }
    </script>

    <title>SOA und Web2.0 AJAX Instant Messaging Beispiel</title>
  </head>

  <body>
    <input type="button" value="Aktualisieren" onClick="hol_datei()" />
    <br /> &nbsp; <br />
    <div style="width: 800px; border: 1px solid black; border-color: black; border-style: solid; border-width: 1px; padding: 5px; border-radius: 5px;" id="output">
      Es gibt nichts zu sehen
    </div>
  </body>
</html>
```

### A.2 datei.php

```
<?php
$data = file("icq.tmp");
#$data = file("/var/log/auth.log");

$i=0;
while($i < count($data)){
  echo $data[$i] . "<br/>";
  $i++;
}
?>
```

### A.3 phurple.php

```
#!/usr/bin/php
<?php

$uiid = "php";
init_libpurple($uiid);

$prpls = purple_plugins_get_protocols();

foreach ($prpls as $i=>$int)
  $prpls[$i] = purpleplugin_from_int($int);

$prpl = $prpls[4]; //4=ICQ
$name = "447163586";
$password = "123qwe";

$account = purple_account_new($name, purple_plugin_get_id($prpl));
purple_account_set_password($account, $password);
purple_account_set_enabled($account, $uiid, TRUE);

mainloop();
?>
```

## DANKSAGUNGEN

Der Autor dankt hiermit den folgenden Personen:

- Prof. Dr. Christoph Reich, für das halten der SOA & Web2.0 Vorlesung

- Philip Waldhauer, für die Hilfe beim installieren der php-libpurple Bindings und das Testen der Beispielanwendung
- Marco Schönfelder und Michael Eckhardt für das Testen der Beispielanwendung