

# Ajax and search engines

Markus Korzendorfer, Matrikelnummer: 228523

## Zusammenfassung

Eine neue Technik der Web-Entwicklung erlebt derzeit ihren Durchbruch: Ajax (Asynchronous JavaScript and XML) gibt dem Web neue Interaktionsmuster, die an lokale Anwendungsprogramme erinnern: Nebenläufigkeit, asynchrone Aktualisierung von Seitenbestandteilen und ansprechende visuelle Interaktion machen das Web dynamischer. Zwar sind die eingesetzten Technologien - JavaScript, DOM, CSS und manchmal XML - eigentlich nicht neu, aber sie führen zu einer neuen Art von webbasierten Anwendungen. Ajax bringt Leichtigkeit in die sonst starren Webseiten und ist variabel einsetzbar. Einer der ersten produktiven Anwendungen war „Google Suggest“- eine Suchmaske, die während des Tippens zeitsparende Tipps gibt.

Dieses Dokument bietet einen Überblick über die Ajax-Technologie. Nach einer Einführung zum Thema werden zunächst die Grundlagen und Basiswissen von Ajax erläutert. Anschliessend wird in Kapitel 3 [Praktisches Beispiel] der Suchdienst Google Suggest, vereinfacht, nachgebaut und erklärt.

## Index Terms

Ajax, Javascript, XML, PHP, Web 2.0, Ajax Suggest.



## 1 EINLEITUNG

Die Professionalisierung des Web-Designs und die verstärkte kommerzielle Nutzung des Mediums haben dazu geführt, dass sich bestimmte Technologien im WWW als Standard etabliert haben und von den meisten modernen Browsern hinreichend unterstützt werden. Auf dieser Basis sind in der jüngsten Vergangenheit interaktive webbasierte Anwendungen entstanden, mit denen die Lücke zwischen Desktop und WWW effektiv überbrückt werden kann.

Die kommerzielle Nutzung des World Wide Web hat in den letzten Jahren mit der Etablierung von XML und verwandten Standards dazu geführt, dass es sich von einem globalen Informationssystem für technisch versierte Nutzer zu einem strategisch wichtigen Kommunikationskanal für fast jedes Unternehmen gewandelt hat.

Die Weiterentwicklung der technischen Infrastruktur sowie ein gestiegenes Qualitätsbewusstsein bei der Gestaltung von Benutzungsschnittstellen und im Software-Entwicklungsprozess haben zur Ausprägung bestimmter Entwurfsmuster geführt, die heute oft mit dem Schlagwort Ajax beschrieben

werden. Bei Ajax handelt es sich nicht um eine neuartige Technologie, sondern um einen Sammelbegriff für bestimmte Techniken zur systematischen und effizienten Ausnutzung bestehender Funktionalität, die das World Wide Web heute bereitstellt.

Was ist also neu an Ajax? Die Browser hatten alle Voraussetzungen dafür bereits mindestens seit 2001. Den Mainstream erreicht Ajax allerdings erst im Februar 2005 durch einen Artikel von Jesse James Garrett *Ajax: A New Approach to Web Applications*<sup>1</sup>, in dem diese Technologie beschrieben und die Abkürzung Ajax erstmals in der Öffentlichkeit definiert wurde. Die Bewegung zum „Web 2.0“ war bereits in vollem Gange, und verschiedene viel-diskutierte Projekte wie Google Maps und Gmail popularisierten die Ajax-Funktionen schnell.

Markus Korzendorfer  
12. Februar 2009

## 2 AJAX GRUNDLAGEN

Dieses Kapitel befasst sich mit den Grundlagen von Ajax und gibt einen kurzen Einblick in die Arbeitsweise. Moderne Browser bieten mit dem Objekt *XMLHttpRequest* eine mächtige Schnittstelle zur Kontrolle von HTTP-Transaktionen. Für die asynchrone Kommunikation zwischen Browser und Webserver erlaubt das Objekt zum einen die Registrierung von Callback-Funktionen, die bei jeder Änderung des Transaktionszustands ausgewertet werden, zum anderen kann auf alle Felder von Anfrage und Antwort zugegriffen werden.

### 2.1 Ajax Einführung

#### 2.1.1 Was ist Ajax

Ajax ist die Abkürzung für *Asynchronous Javascript and XML*. Diese Technik dient dazu, dass Webseiten mit einem Server kommunizieren können, ohne dass die entsprechende Webseite dabei komplett neu geladen werden muss und der Anwender durch Neuladen beim Surfen auf der Webseite unterbrochen wird. Ajax ermöglicht es also innerhalb einer HTML-Seite Anfragen an den entsprechenden Server zu stellen, ohne die Webseite komplett neu zu laden. Es werden dabei immer nur die Teile neu- bzw. nachgeladen, die sich geändert haben oder von der Webseite nun benötigt werden.

Ein einfaches Beispiel dafür ist ein Warenkorbsystem ohne Ajax. Bestellen Sie beispielsweise ein Buch, lädt Ihr Browser nach dem Anklicken des *Hinzufügen-Buttons* die Warenkorb-Übersichtsseite, auf welcher nun Ihr bestelltes Buch aufgelistet wird. Möchte Sie ein weiteres Buch bestellen müssen Sie nun wieder aus Ihrem Warenkorb in die Bücherseite wechseln.

1. siehe Quelle 11

Mit einem Ajax-Warenkorb könnten Sie Produkte in Ihren Warenkorb legen, ohne jedes Mal auf der Warenkorb-Übersichtsseite zu landen. Diese Seite wird also nicht nach dem Anklicken des *Hinzufügen-Buttons* geöffnet.

Ajax-Anwendungen sind dadurch deutlich Zeit - und Ressourcensparender, weil immer nur die Daten nachgeladen werden, die gerade benötigt oder geändert wurden und nicht immer eine ganze Seite.

### 2.1.2 Vor- und Nachteile von Ajax

- Vorteile
  - Nicht die komplette Webseite muss neu geladen werden (Inhalte werden nur teilweise nachgeladen)
  - Reduzierung des Traffics
  - Schnellere Reaktion auf Benutzereingaben
  - kostenlos
  - kein Browser-Plugin nötig
  - Usability wird verbessert
- Nachteile
  - Javascript muss beim Besucher aktiviert sein<sup>1</sup>
  - SEO leidet (Suchmaschinen-Robots können kein Javascript)
  - Usability Verschlechterung:
    - \* Probleme mit Browsernavigation (Vor- und Zurückbutton funktioniert nicht mehr)
    - \* Lesezeichen erstellen funktioniert nicht (es lassen sich nicht die einzelnen Zustände einer Seite abspeichern)
    - \* Eventuell für den Benutzer unerwartetes Verhalten

## 2.2 Aufbau und Ablauf

### 2.2.1 Vergleich mit herkömmlichen Webanwendungen

Klassische Web-Anwendungen benötigen für jede Änderung oder Aktion des Benutzers ein komplettes Neuladen der Webseite, es muss also eine HTTP-Verbindung zum Server aufgebaut werden, welcher dann die neuen, aktualisierten Daten zurück an den Browser schickt. Die Grafik unterhalb verdeutlicht dieses Verhalten:

1. siehe Quelle 08



Abbildung 1: Normale Http-Anfrage - Quelle [04]

Setzt man für die gleiche Anwendung Ajax ein, wird nur noch ein kleiner Teil der Webseite neu geladen. Mit dem Anklicken des *Hinzufügen-Buttons* aus unserem Beispiel, baut auch die Ajax-Version eine Verbindung mittels JavaScript zum Webserver auf. Auch die Antwort des Webserver kann dann im Javascript ausgewertet werden und somit nur ein Teil der Webseite aktualisiert werden. Deswegen muss die Webseite nicht mehr komplett neu geladen werden.

Die Ajax-Anfrage ist hierbei ein normaler HTTP-Request, der grosse Vorteil gegenüber dem klassischen Neuladen der Seite liegt aber darin, dass die übertragenen Daten geringer sind, da nur Änderungen übertragen werden müssen.

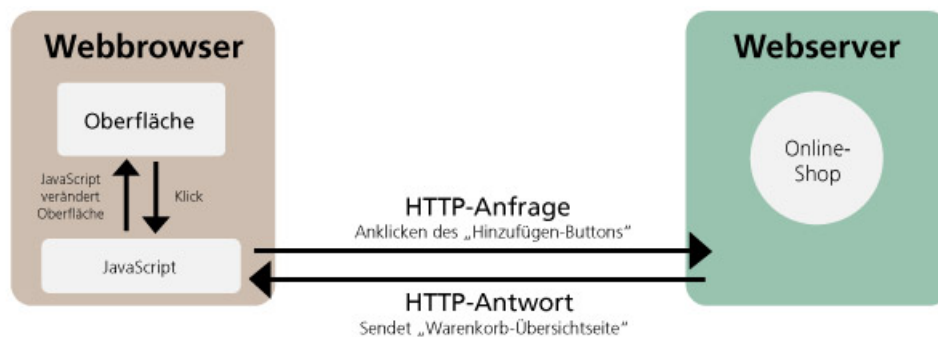


Abbildung 2: Http-Anfrage mit Ajax - Quelle [04]

Durch die Möglichkeit von Ajax immer nur bestimmte Teile einer Webanwendung oder Webseite nachzuladen, werden Anwendungen deutlich dynamischer und lassen sich schneller handhaben. Daher empfinden Besucher Webseiten die Ajax-Technologien einsetzen oft als schneller und dynamischer. Zusätzlich vermitteln sie oft das Gefühl, als ob die Anwendung vollständig auf dem Rechner des Anwenders laufen würden.

### 2.2.2 Ajax Webtechnologien

Ajax selbst basiert auf vielen bestehenden Webtechnologien.<sup>1</sup>

- (X)HTML: Auszeichnungssprache für Webseite
- CSS: Zur Formatierung vom HTML
- JavaScript: Grundlage der Interaktion, Darstellung von Daten und zur Arbeit mit dem DOM
- DOM (Document Object Model): Objektorientierter Zugriff auf Struktur der Webseite
- XMLHttpRequest-Objekt: Zentrales Objekt für Ajax, benötigt um Anfragen an den Webserver zu stellen
- XML: Standardisierte Datenstruktur

Ausserdem lässt sich Ajax in Verbindung mit vielen bestehenden Technologien einsetzen.

- Übliche Scriptsprachen im Web, wie:
  - PHP
  - Python
  - Perl
  - Java
  - etc.
- XSLT - Zur Fomatierung von XML-Antworten
- SOAP/XML-RPC

Neben den grundlegenden Techniken sind auch neue Technologien hinzukommen, die wichtigsten dabei sind:

- JSON: Alternative Struktur/Schreibweise für Datenübertragung vom/zum Webserver

### 2.2.3 JSON als XML-Alternative

XML-Daten sind in vielen Fällen oft nicht der ideale Weg um Daten zu strukturieren. Durch das Tag-System von XML werden oft kleine Datenbestände sehr aufgebläht und unübersichtlich. Zusätzlich ist das Ansprechen einzelner Knoten (XML-Nodes) in einer XML-Datei nicht immer leicht.<sup>2</sup>

Eine Alternative dazu bietet JSON, kurz für *JavaScript Object Notation*, einer Vorgehensweise um Daten auf einfache Weise zu strukturieren. JSON ist sehr leicht zu erlernen, zu handhaben und hat sich inzwischen für viele weitere Programmiersprachen durchgesetzt.

Der groe Vorteil von JSON im Vergleich zu XML liegt in der einfachen Handhabung. Da JSON selbst gültiges Javascript darstellt kann es direkt ausgeführt und somit in ein Javascript-Objekt überführt werden. Der Zugriff auf die einzelnen Eigenschaften kann dann über normalen Attributzugriff geschehen.

1. siehe Quelle 01

2. siehe Quelle 05 und 06

## 2.3 XMLHttpRequest-Objekt

Das XMLHttpRequest-Objekt ist eine sogenannte API, die den Transport von Daten über das Webprotokoll HTTP ermöglicht. Es bildet die Grundlage der Ajax-Technik und wird von JavaScript aus verwendet. Mit Hilfe des XMLHttpRequest-Objekts wird es möglich in JavaScript HTTP-Anfragen an einen Server zu stellen und die entsprechenden Antworten des Server zu verarbeiten. Es stellt also eine Schnittstelle zur Kommunikation mit dem Server dar.<sup>1</sup>

### 2.3.1 XMLHttpRequest-Objekt erzeugen

Ein XMLHttpRequest-Objekte erzeugt man wie bereits besprochen mit JavaScript. Mit dem folgenden Code erzeugt man auf einfachste Weise ein XMLHttpRequest-Objekt<sup>2</sup>.

```
1 var xmlhttpObject = new XMLHttpRequest();
```

### 2.3.2 XMLHttpRequest-Objekt nutzen

Um das XMLHttpRequest-Objekt benutzen zu können ist es wichtig seine Funktion und damit seine API zu verstehen. Diese wird durch Methoden und Attribute abgebildet.

Methoden vom XMLHttpRequest-Objekt:<sup>3</sup>

- `abort()`:
  - Bricht einen laufenden HTTP-Request ab
- `getResponseHeader(header)`:
  - Gibt einen bestimmten HTTP-Header der Antwort zurück
- `getAllResponseHeaders()`:
  - Gibt alle HTTP-Header der Antwort zurück
- `open(method, url)`:
  - Bereitet eine HTTP-Anfrage vor
  - `method` kann POST, GET, oder PUT sein
  - `url` ist die Adresse, die aufgerufen werden soll (kann GET-Parameter enthalten)
- `send(body)`:
  - Sendet die HTTP-Anfrage

1. siehe Quelle 07

2. Bezieht sich auf Mozilla Firefox

3. siehe Quelle 02

- Der Parameter `body` kann POST-Daten enthalten, diese werden wie GET-Parameter in der URL formatiert. Sollen keine POST-Daten gesendet werden kann null angegeben werden.
- `setRequestHeader(header, value):`
  - Setzt einen HTTP-Header für die HTTP-Anfrage auf einen bestimmten Wert `value`

#### Attribute des XMLHttpRequest-Objekts<sup>1</sup>

- `readyState`
  - Aktueller Status des Objekts:
    - \* 0: Nicht initialisiert
    - \* 1: Objekt ist bereit, keine Daten gesendet
    - \* 2: Anfrage wurde gesendet
    - \* 3: Daten werden empfangen (`onreadystatechange` wird evtl. mehrmals aufgerufen)
    - \* 4: Alle Daten wurden empfangen
- `onreadystatechange`
  - Referenz auf eine Funktion, die aufgerufen wird, wenn sich etwas bei der HTTP-Anfrage ändert (Erfolg/Fehler/...), siehe `readyState`
- `responseText`
  - HTTP-Antwort als Text
- `responseXML`
  - HTTP-Antwort als XML, Zugriff per DOM möglich
- `status`
  - HTTP-Statuscode der Antwort (z.B. 200 für Ok, 404 für Not Found, ...)
- `statusText`
  - HTTP-Statustext der Antwort (z.B. Ok, Not Found, ...)

### 3 PRAKTISCHES BEISPIEL

Auf den folgenden Seiten wird nun das Prinzip von Google-Suggest nachempfunden und erklärt. Im Anhang dieses Textes findet man den entsprechenden Quellcode. Herkömmliche Suchsysteme sind langsam. Nach dem Eintippen muss entweder ein Button oder eine Taste gedrückt werden, die Webseite lädt neu und erst jetzt werden die Ergebnisse präsentiert. Deutlich einfacher geht es mit einer Suche, die automatisch unter dem Suchfeld mögliche Suchbegriffe präsentiert. Abbildung 3 verdeutlicht den Sachverhalt.

1. siehe Quelle 03

**Adressbuch**

Bitte Name eingeben

m

Michael

Mona

Manuel

Marc

Marcel

Abbildung 3: Ajax-Suggest

### 3.1 Formular bauen

Aller Anfang bildet ein herkömmliches HTML-Formular, das auch bei einer normalen Suchfunktion eingesetzt wird. Jedoch soll nach jedem Tastendruck eine Suche ausgelöst werden. Dies wird durch den Einbau der Javascript-Funktion *suggest* erreicht, die nach Tastendruck *onkeyup* den aktuellen Textinhalt *this.value* verarbeitet.

```
1 <input type="text" id="eingabe" name="q" onkeyup="suggest(this.value)"/>
```

### 3.2 Ajax-Funktionen erstellen

Fast alle Browser haben keine Probleme mit Ajax und der verknüpften Verarbeitung von XML-Objekten. Mittels Überprüfung, ob ein Ajax-Objekt erzeugt werden kann, bekommt man eine Fehlermeldung.

```
1 xmlhttp=httpXMLobjects();
2 if (xmlhttp==null) {
3   alert ("Browser does not support AJAX");
4   return;
5 }
```

Ist Ajax verfügbar, wird nach jedem Tastendruck der Inhalt des Textfelds an die Funktion *suggest* weitergegeben. Dabei gibt es viele Möglichkeiten, welcher Inhalt im Textfeld stecken kann. Ist das Textfeld ganz leer, sollen natürlich auch keine Suchhilfen angezeigt werden. Dafür wird die Länge des übergebenen Strings *suchbegriff* überprüft. Ist der String leer, wird das Ausgabefeld ausgabe geleert.

```
1 if (suchbegriff.length==0) {
2   document.getElementById("ausgabe").innerHTML="";
3   return;
4 }
```

In den meisten Fällen ist jedoch ein Suchbegriff übergeben und dieser muss an den Server zur Suche geschickt werden. PHP wurde in diesem Fall als geeignetes Backend gewählt. Für den Versand des Suchbegriffs wird eine URL gebastelt, die an die Suchdatei *processor.php* den String *suchbegriff* und eine Zufallszahl *Math.random()* hängt. Die Zufallszahl unterbindet das Cachen der meisten Browser und sorgt somit für stets aktuelle Suchergebnisse. In den folgenden drei *xmlHttpRequest*-Funktionen wird die oben definierte URL aufgerufen und auf Antwort des Servers gewartet.

```

1 var aufruf="processor.php"+"?q="+suchbegriff+"&sid="+Math.random();
2   xmlhttp.onreadystatechange=stateChanged;
3   xmlhttp.open("GET",aufruf,true);
4   xmlhttp.send(null);

```

### 3.3 Suche in PHP

Ist der Suchbegriff erst einmal im PHP-Skript<sup>1</sup> angekommen, ist die weitere Verarbeitung einfach. In diesem Fall wird ein Array durchsucht, kompliziertere Suchsysteme können jedoch ohne Probleme angeschlossen werden - nur die Ausgabe muss wieder stimmen. Zuerst wird jeder Suchbegriff mit in einer Schleife mit den gegebenen Begriffen verglichen. Die Funktion *strtolower* glättet zunächst Suchbegriff und gegebene Terme auf Kleinschrift, um immer gleiche Ausgangssituationen zu erreichen.

Anschliessend wird überprüft, ob der Suchbegriff *\$q* mit der gleichen Anzahl Zeichen des gegebenen Begriffs übereinstimmt. Hierzu wird die Funktion *substr()* genutzt, die ausgehend von der Länge des Suchbegriffs *strlen(\$q)* Zeichen ausgibt. Beispiel: Der Suchbegriff *ma* wird eingegeben. Mittels *strlen* wird eine Länge des String von 2 Zeichen ermittelt. Der aktuell zu überprüfende Begriff ist Manuel. Die Funktion *substr* liefert mit diesem Begriff und der Länge des Suchbegriffs von 2 Zeichen die Ausgabe *ma*. Dies stimmt also mit dem Suchbegriff überein und kann den Vorschlägen eingefügt werden.

```

1 if (strtolower($q)==strtolower(substr($keyword_aktuell,0,strlen($q))) {
2     $hint=$hint.'<div class="ergebnis"><a href="detail.php?q='.$keyword_aktuell.'">'.$keyword_aktuell.'</a></div >';
3 }

```

### 3.4 Vorschläge anzeigen

Mit der Suchantwort aus PHP in den Händen ist die Suggest-Suche fast schon geschafft. Die Ausgabe des PHP-Skripts muss nur noch unter dem Suchfeld platziert werden. Hierzu wird ein Platzhalter unter dem Input-Feld angebracht, in den gleich die Vorschläge kommen.

1. siehe Quelle 10

```
1 <div id="ausgabe"></div>
```

Zum Einfügen der Vorschläge wartet Ajax auf den fertigen Aufruf. Je nach Engine wird der komplettierte Aufruf von der Funktion *xmlHttpRequest.readyState* als *4* oder *complete* übergeben. War somit alles erfolgreich, kann die PHP-Antwort als *xmlHttpRequest.responseText* in das HTML-Element *ausgabe* eingefügt werden.

```
1 if (xmlHttpRequest.readyState==4 || xmlHttpRequest.readyState=="complete") {  
2     document.getElementById("ausgabe").innerHTML=xmlHttpRequest.responseText;  
3 }
```

## 4 FAZIT UND AUSBLICK

In diesem Dokument hat man Ajax kennengelernt, eine Technologie, mit der sich webbasierte Anwendungen interaktiver und damit flüssiger gestalten lassen als im klassischen WWW. Obwohl die technischen Grundlagen von Ajax schon seit einigen Jahren verfügbar sind, trat die Kombination von skriptbasierten DOM-Manipulationen und asynchroner Kommunikation zwischen Browser und Server erst im Jahr 2005 mit zukunftsweisenden Anwendungen wie Google Maps oder Backpack verstärkt in das öffentliche Bewusstsein.

Diese späte Euphorie über Technologien, die eigentlich schon lange bekannt sind, verdankt Ajax vor allem der fortschreitenden Konsolidierung der W3C-Standards, die sich als Vereinheitlichung der Browserplattformen äussert. Dadurch können portable Anwendungen auf der Basis zukunftsweisender Standards entwickelt werden, die die Leistungsfähigkeit klassischer Desktop-Anwendungen mit der Informationsfülle eines weltweiten Informationssystems verknüpfen. Neuartige Benutzungsschnittstellen und Interaktionsmuster sind im Entstehen und regen die Phantasie von Nutzern und Entwicklern gleichermassen zur Erschaffung neuartiger Anwendungen an.

Damit zeigt sich, dass die Stärken von Ajax vor allem im Zusammenspiel unterschiedlicher Technologien liegen, deren Kombination neue Anwendungsmöglichkeiten erschliesst. Dabei ergeben sich zwangsläufig einige Änderungen im Umgang mit webbasierten Anwendungen, an die sich sowohl Nutzer als auch Entwickler werden gewöhnen müssen: neuartige Benutzungsschnittstellen, erweiterte Formen der Interaktion, detailliertere Profilbildung. Bei genauerem Hinsehen zeigt sich allerdings auch, dass hier ein von Ajax weitgehend unabhängiger Paradigmenwechsel stattfindet: Die Basis-Technologien sind es, die heute diesen neuartigen Umgang mit den bestehenden Systemen ermöglichen. Ajax stellt nur den Begriff, um die Entwicklung leichter in Worte fassen zu können.

## LITERATUR

- [01] Wikipedia - Ajax  
[http://de.wikipedia.org/wiki/Ajax\\_\(Programmierung\)](http://de.wikipedia.org/wiki/Ajax_(Programmierung))  
(Zugriff am 01..02.2009)
- [02] Dynamic ajax - Methoden  
[http://www.dynamicajax.com/fr/XMLHttpRequest\\_Methods-271\\_272\\_301.html](http://www.dynamicajax.com/fr/XMLHttpRequest_Methods-271_272_301.html)  
(Zugriff am 12.02.2009)
- [03] Dynamic ajax - Properties  
[http://www.dynamicajax.com/fr/XMLHttpRequest\\_Properties-271\\_272\\_302.html](http://www.dynamicajax.com/fr/XMLHttpRequest_Properties-271_272_302.html)  
(Zugriff am 12.02.2009)
- [04] Freie Universität Berlin - AjaxParts in WebParts  
<https://www.inf.fu-berlin.de/w/VNBI/AjaxParts>  
(Zugriff am 12.02.2009)
- [05] Wikipedia - JSON  
<http://de.wikipedia.org/wiki/JSON>  
(Zugriff am 12.02.2009)
- [06] JSON  
<http://json.org/json-de.html>  
(Zugriff am 07.02.2009)
- [07] AJAX - Frische Ansätze für das Web-Design  
<http://www.teialehrbuch.de/Kostenlose-Kurse/AJAX/31946-AJAX-Grundlagen.html>  
(Zugriff am 03.02.2009)
- [08] SELFHTML - JavaScript  
<http://de.selfhtml.org/javascript/intro.htm>  
(Zugriff am 02.02.2009)
- [09] SELFHTML - XML  
<http://de.selfhtml.org/xml/>  
(Zugriff am 01.02.2009)
- [10] SkyHome - PHP  
<http://www.skyhome.de/php/>  
(Zugriff am 11.02.2009)
- [11] adaptive path  
<http://www.adaptivepath.com/ideas/essays/archives/000385.php>  
(Zugriff am 05.02.2009)

## ANHANG A BEISPIEL AJAX SUGGEST

### A.1 ajax.html

```
1 <html>
2 <head>
3 <title>Ajax_Suggest</title>
4 <script src="search.js"></script>
5 <link rel="stylesheet" type="text/css" media="screen" href="style.css" />
6 </head>
7 <body>
8 <div id="main">
9 <h1>Ajax Suggest</h1>
10 <form method="GET" action="suche.php">
11 Bitte Name eingeben
12 <input type="text" id="eingabe" name="q" onkeyup="suggest(this.value)" />
13 </form>
14 <div id="ausgabe"></div>
15 </div>
16 </body>
17 </html>
```

## A.2 search.js

```

1 var xmlhttp;
2
3 function suggest(suchbegriff) {
4 xmlhttp=httpXMLObjects();
5 if (xmlhttp==null) {
6 alert ("Browser does not support AJAX");
7 return;
8 }
9 if (suchbegriff.length==0) {
10 document.getElementById("ausgabe").innerHTML="";
11 return;
12 }
13 else {
14 var aufruf="processor.php"?q="+suchbegriff+"&sid="+Math.random();
15 xmlhttp.onreadystatechange=stateChanged;
16 xmlhttp.open("GET",aufruf,true);
17 xmlhttp.send(null);
18 }
19 }
20 function stateChanged() {
21 if (xmlhttp.readyState==4 || xmlhttp.readyState=="complete") {
22 document.getElementById("ausgabe").innerHTML=xmlhttp.responseText;
23 }
24 }
25
26 //AJAX-Standards
27 //Weniger interessant
28
29 function httpXMLObjects() {
30 var xmlhttp=null;
31 try {
32 // Fuer Firefox, Opera und Safari
33 xmlhttp=new XMLHttpRequest();
34 }
35 catch (e) {
36 // Der Internet Explorer wills wieder anders
37 try {
38 xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
39 }
40 catch (e) {
41 xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
42 }
43 }
44 return xmlhttp;
45 }

```

## A.3 processor.php

```

1 <?php
2 //Array mit Daten beladen
3 $keywords[]="Anna";
4 $keywords[]="Adrian";
5 $keywords[]="Aaron";
6 $keywords[]="August";
7 $keywords[]="Eva";
8 $keywords[]="Sebastian";
9 $keywords[]="Sven";
10 $keywords[]="Horst";
11 $keywords[]="Michael";
12 $keywords[]="Lena";
13 $keywords[]="Lisa";
14 // .... usw.
15
16 //Suchbegriff aus der URL per GET filtern
17 $q=$_GET["q"];
18
19 //String-Laenge ermitteln und falls groesser 0 -> weiter
20 if (strlen($q) > 0){
21 $hint="";
22 foreach($keywords as $keyword_aktuell) {
23 if (strtolower($q)==strtolower(substr($keyword_aktuell,0,strlen($q)))) {
24 $hint=$hint."<div class='ergebnis'><a href='detail.php?q='.$keyword_aktuell.'>".$keyword_aktuell."</a></div>";
25 }
26 }
27 }
28 }
29

```

```
30 //Checken, ob Eintraege gefunden wurden. Wenn nicht, Fehler
31 if (empty($hint)) {
32     $response="Leider keine Eintr&auml;ge";
33 }
34 else{
35     $response=$hint;
36 }
37
38 //Und nur noch das Ganze ausgeben
39 echo $response;
40 ?>
```

## A.4 style.css

```
1 body {
2     background:# f5f5f5;
3     margin:10px;
4     font-size:12px;
5     font-family: Arial;
6 }
7 #main {
8     width:300px;
9     border:1px solid grey;
10    background: white;
11    margin-left: auto;
12    margin-right: auto;
13 }
14 #main input {
15     width:300px;
16     font-size:20px;
17     background:lightgrey;
18     border:1px solid white;
19 }
20 #main h1 {
21     text-align:center;
22 }
23 .ergebnis {
24     border-bottom:1px solid grey;
25     font-size:16px;
26     padding:4px;
27 }
28 .ergebnis a {
29     color:black;
30     text-decoration: none;
31 }
```