

Sven Wächli und Mario Rimann

Dezentes Ajax

TYPO3 und Ajax geschickt kombinieren

Mit Hilfe von Ajax ist es problemlos möglich, Teilbereiche von Webseiten nachzuladen, ohne dabei auf Frames angewiesen zu sein. Dabei ergeben sich allerdings zahlreiche Fragen: Ist diese Lösung benutzerfreundlich, zugänglich und abwärtskompatibel? Können die Seiten mit vertretbarem Aufwand unterhalten werden? Was geschieht bei deaktiviertem JavaScript? Und vor allem: Werden die Seiten von Suchmaschinen noch gefunden? Eine neue TYPO3-Extension verspricht Antworten und zeigt einen möglichen Weg, die Klippen zu umschiffen.

Der als AJAX bezeichnete Technologiemix aus „Asynchronous JavaScript and XML“ ermöglicht schlanke Web-Applikationen, die sich anfühlen wie gewohnte Desktop-Programme. JavaScript ist für die meisten Screenreader und auch für Suchmaschinen jedoch oft eine unüberwindbare Hürde. Damit ist der Einsatzbereich dieser Technologie beschränkt. Außerdem können Anwender JavaScript in ihrem Browser deaktivieren, was in einigen Computerzeitschriften aus Sicherheitsgründen sogar empfohlen wird. Für die meisten Websites wird somit ein anderes Fundament benötigt. Dies ist jedoch kein Grund, um auf den Einsatz von Ajax zu verzichten.

Für die bessere Verwaltung sowie für eine suchmaschinen-taugliche und benutzerfreundliche Aufbereitung von Web-Content liegt es nahe, ein Content Management System wie zum Beispiel TYPO3 einzusetzen. Darüber hinaus kann es sinnvoll sein, den Inhaltsbereich von Seiten zu ändern, während andere Elemente wie etwa Animationen oder Videosequenzen kontinuierlich weiterlaufen. Dies kann mit Frames, mit iframes oder auch mit Flash bewerkstelligt werden – die Nachteile sind jedoch nicht unerheblich [1]. Um diese Nachteile beim Einsatz von Frames, iframes oder Flash zu vermeiden und die Vorteile von zeitgemäßem Code zu nutzen – etwa ein mit der Schriftgröße skalierendes Layout – wurde eine Ajax-basierte Lösung gesucht. Das von TYPO3 bereitgestellte Fundament sollte nicht beeinträchtigt und die Arbeit im Backend nicht erschwert werden. TYPO3 und Ajax sollten dabei konsequent getrennt werden. Zum einen sollte eine Website auch ohne Ajax funktionieren, zum anderen sollen sich Administratoren und Editoren nicht mit Ajax-spezifischen Anforderungen auseinandersetzen müssen.

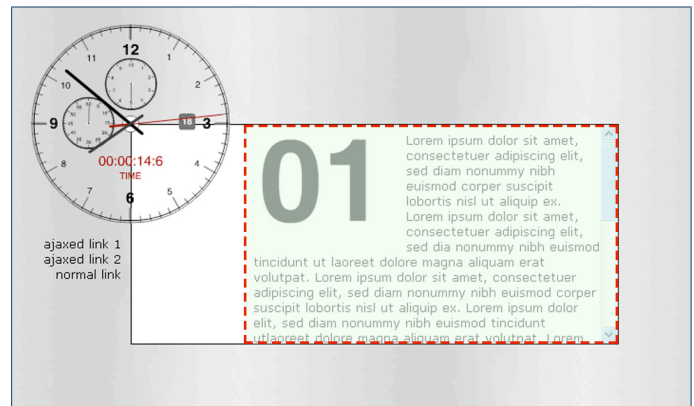
Das Ergebnis ist eine Lösung, mit der Entwickler ihre TYPO3-Websites mit wenig Aufwand um Ajax-Funktionen erweitern können. Mit der Unterstützung von Mario Rimann entstand in den letzten Monaten unter dem Namen tut_unobtrusiveajax [2] eine sogenannte Tutorial-Extension, die dazu dient, die Ergänzung einer Website um Ajax-Features nachvollziehen zu können.

Schichten trennen

Eine bestehende TYPO3-Website kann mit wenig Aufwand um AJAX-Funktionalität erweitert werden. Dazu müssen in TYPO3 die betroffenen Links und Content-Bereiche markiert, die JavaScripts zum „ajaxifizieren“ der Links eingebunden und ein PHP-Skript zur Verarbeitung der markierten Content-Bereiche erstellt und auf dem Server abgelegt werden. Benötigt werden:

- Anpassungen der Links, die Content dynamisch laden sollen (class="ajaxLink")
- Spezielle Kommentare (Hooks), die den dynamischen Inhaltsbereich in allen Seiten umschließen
- JavaScript, das in jede TYPO3-Seite eingebunden wird

- Ein einfaches PHP-Skript, das die Teilbereiche zwischen den Hooks auslesen kann



Eine Demo-Seite verdeutlicht den Nutzen von AJAX für TYPO3.

Der Schlüssel für den unaufdringlichen Einsatz von Ajax liegt im Laden einer voll funktionsfähigen Seite und dem nachträglichen „ajaxifizieren“ der Dokumentstruktur per JavaScript. Listing 1 zeigt einen Link vor der DOM-Manipulation, Listing 2 einen Link mit Event-Handler nach der Veränderung des DOM.

HTML

```
<li class="ajaxLink"><a href="index.php?id=56">ajaxed link 1</a>
```

Listing 1

HTML

```
<a href="javascript:void(0);onclick="javascript:sndReq('index.php?id=56');">ajaxed link 1</a>
```

Listing 2

Die DOM-Manipulation, die Ajax aktiviert, wird beim Laden der Seite ausgelöst. Im Sinne von „unobtrusive JavaScript“ [3] wird einerseits Rücksicht auf bereits vorhandenes JavaScript genommen und andererseits die Manipulation ausschließlich bei Ajax-fähigen Clients ausgeführt [4].

Suchmaschinen, Screenreader und Besucher mit deaktiviertem JavaScript oder mit generell nicht kompatiblen Clients navigieren die Website ohne Ajax. Die Website ist somit zugänglich und abwärtskompatibel. Suchmaschinen können die einzelnen Seiten zudem erfassen.

JAVASCRIPT

```
if(window.onload){
    var existingOnload = window.onload;
}else{
```

```
var existingOnload = function(){}; //do nothing
}
function secondOnload() {
    existingOnload();
    getElementsByClassNameAndReplace("ajaxLink");
}
var W3CDOM=(document.createElement && document.getElementsByTagName);
//zusätzliche Abfragen wie z. B. Browser/Versionen könnten hier eingesetzt werden
if(W3CDOM){
    window.onload = secondOnload;
}
```

Listing 3

Sofern die DOM-Manipulation der Seite durchgeführt wurde, verweisen die Navigationslinks bzw. die mit class="ajaxLink" ausgezeichneten Links nun nicht mehr direkt auf eine Seite. Ajax übergibt statt dessen die Linkadresse als Parameter an den Server, wo ein PHP-Skript die angesteuerte Seite aufruft, den Teilbereich zwischen den gesetzten Hooks ausliest und an den Client zurückliefert. Mit Hilfe von JavaScript wird das erhaltene Fragment an der richtigen Stelle ersetzt, ohne dass die Seite neu geladen wird.

TYPO3 HTML-TEMPLATE

```
<div id="stuffToLoad">
  <!--uniqueContentSTART-->
  ###CONTENT###
  <!--uniqueContentEND-->
</div>
```

Listing 4

PHP

```
function getPageContent($sUrl) {
    return implode(file($sUrl), "");
}
function parseContentByTagName($sContent) {
    preg_match_all(
        '/<!--uniqueContentStart-->([\n\r\w\W.]*?)<!--uniqueContentEnd-->/i',
        $sContent,
        $aContent
    );
    return implode($aContent[0], "");
}
```

Listing 5

Je nach Fall kann es sinnvoll sein, im gleichen Arbeitsgang weitere Elemente auszuwechseln, etwa den Seitentitel.

Fazit

Der Teufel steckt wie immer im Detail. Um die History und die Bookmark-Funktionalität des Browsers zu erhalten, wurde das JavaScript „dhtmlHistory.js“ eingesetzt [5]. Zusätzliche Workarounds waren nötig, um „simulateStaticDocuments“ und „realUrl“ zu unterstützen. Damit funktioniert die mit der Extension beschriebene Lösung für einfache Websites. Im konkreten Anwendungsfall können weitere Anpassungen nötig werden, etwa für die Unterstützung von Mehrsprachigkeit oder bei speziellen Anforderungen an das Seiten-Layout. Diese Spezialfälle werden in der Tutorial-Extension nicht abgehandelt. JavaScript-Kenntnisse vorausgesetzt, sollte man die Tutorial-Scripts jedoch schnell an die eigenen Bedürfnisse anpassen können.

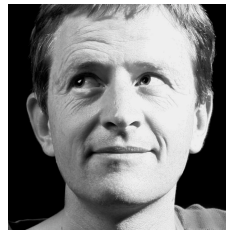
Links und Literatur

Softlink 1518

- [1] Frames: <http://www.mediacollege.com/internet/html/frames/pros-cons.html>

- [2] tut_unobtrusiveajax-Extension im TER: http://typo3.org/extensions/repository/view/tut_unobtrusiveajax/0.2.0/
- [3] unobtrusive JavaScript: <http://alistapart.com/articles/behavioralseparation>
- [4] Peter-Paul Koch on JavaScript: ISBN 0-321-42330-5: <http://www.quirksmode.org/>
- [5] ajax history libraries: <http://codinginparadise.org/weblog/2005/09/ajax-history-libraries.html>

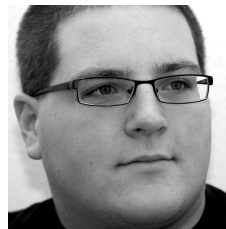
DER AUTOR



Sven Wächli ist Partner der Arbeitsgemeinschaft Screenteam in Zürich (www.screenteam.ch). Als Diplomierter Multimedia Producer SAE, Programmierer und Web-Enthusiast beschäftigt er sich mit der Erstellung und Verbesserung von Websites und Webapplikationen und ist tätig als Prüfungsautor und Prüfungsexperte bei I-CH Informatik Berufsbildung

Schweiz.

DER AUTOR



Mario Rimann ist eine der treibenden Kräfte bei der Lancierung der TYPO3 Developer Days. Unterdessen hat er eine kleine aber feine Firma gegründet, die "engineering and event management" anbietet (www.rimann.org).