jQuery & Co
JavaScript-Frameworks im Vergleich

Tobias Maier[1]

[1]DHBW Stuttgart Campus Horb, 2011

Zusammenfassung. Diese Arbeit erklärt die Idee hinter JavaScript, stellt die geläufigsten JavaScript-Frameworks vor und vergleicht jQuery, MooTools, Prototype und YUI anhand mehrerer praktischer Beispiele, wie etwa dem dynamischen Nachladen von Seiteninhalten oder verschiedenen Animationen.

1 Einleitung

Der Begriff Web 2.0 prägt seit einigen Jahren den Sprachjargon rund um das World Wide Web. Entstanden ist er irgendwann um 2003. Das Web 2.0 steht für eine neue Ära im Umgang mit dem Internet. Der Gebrauch der Versionsnummer soll eine Abgrenzung zum bisherigen WWW, sozusagen der Version 1.0 darstellen.

Bestanden Websites bis dahin meist aus statischen HTML-Seiten, die von einem festen Autoren-Kreis bearbeitet wurden, rückte nun die dynamische Bearbeitung von Inhalten immer mehr in den Vordergrund. Benutzer einer Website konnten und sollten selbst Inhalte einstellen. Überall entstanden Gästebücher, Foren, Weblogs und Wikis. Das Content-Management wurde zu einem zentralen Begriff: Bearbeiter einer Website brauchten keine umfassenden Programmierkenntnisse mehr, sondern konnten mit webbasierten Oberflächen ihre Seiten bearbeiten und die Änderungen sofort sehen bzw. online stellen.

Heute werden die meisten Seiten beim Aufrufen dynamisch aus einer Datenbank zusammengebaut. Die Aktualität der Inhalte ist dadurch deutlich gestiegen, ebenso die Unabhängigkeit von Inhalt und Design. Selbst für die größten Internetauftritte ist es nur noch ein geringer Aufwand, das Layout komplett umzustellen. Hätte man dafür früher alle HTML-Seiten einzeln anpacken müssen, genügen in Zeiten des Web 2.0 meist wenige Änderungen in einigen zentralen Dateien.

Die grundlegende Einstellung gegenüber dem Internet hat sich stark verändert. Das World Wide Web wird heute als Plattform gesehen, auf der praktisch jeder Nutzer seine Daten ablegen kann, anstatt sie wie früher nur lokal zu speichern. Inhalte werden heute mit anderen Menschen geteilt, indem man sie „online stellt“. Emails mit Anhängen zu versenden, wie es zu Zeiten des statischen Internets nötig war, wird immer unattraktiver. Diese Entwicklung hält in Form des Cloud Computing mittlerweile auch vermehrt Einzug in der Geschäftswelt.

Eine wichtige Begleiterscheinung des Web 2.0 ist die zunehmende Interaktivität. Wie bereits erwähnt wollen Benutzer selbst Inhalte einstellen und verändern. Dabei wollen sie auf gewohnte Oberflächen zugreifen und sofort eine Antwort vom System erhalten, wie sie es von ihren lokalen Betriebssystemen gewohnt sind. Das Absenden

eines klassischen HTML-Formulars und das Warten auf den Ladevorgang der anschließenden Seite passen überhaupt nicht in das Web 2.0-Konzept. Deshalb haben sich parallel zu der „Idee“ Web 2.0 auch einige Technologien entwickelt und weiterentwickelt, die das Arbeiten und Leben im World Wide Web vereinfachen sollen.

Die klassische Server-Client-Aufteilung im WWW ist denkbar einfach: Der Server liefert Webseiten als HTML-Code aus, der Client, in Form eines Browsers, hat lediglich die Aufgabe, sie dem Benutzer entsprechend darzustellen. Interaktion des Benutzers mit der Website umfasst immer ein Senden der Daten an den Server, die dortige Verarbeitung und anschließende Auslieferung des Ergebnisses mittels einer neuen HTML-Seite in Richtung Browser. So war es zu Zeiten des Web 1.0.

Inzwischen hat sich die Aufgabenverteilung zwischen Server und Client stark verändert. Der Server liefert weiterhin Seiten aus, die er jedoch zumeist zum Aufrufzeitpunkt erst dynamisch generiert. Die Inhalte dazu bezieht er aus einer Datenbank, die als dritte Komponente hinzugekommen ist. Der Client (Browser) stellt die Inhalte, die er bekommt, nicht nur dar, sondern ist darüber hinaus auch für Animationen, Interaktionsmöglichkeiten und dynamisches Nachladen einzelner Seitenteile verantwortlich.

JavaScript ist die Wundersprache, mit der das alles möglich ist. Sie existiert schon deutlich länger als das Web 2.0, war lange Zeit allerdings als „böse“ Script-Sprache verpönt. Heute noch gibt es etliche Benutzer, die die Ausführung von JavaScript im Browser mittels Deaktivierung unterbinden, weil sie Angst vor Hackern und Viren haben.

Tatsächlich lassen sich mit JavaScript jedoch all die schönen Features auf einer Website verwirklichen, auf die der heutige Internet-Benutzer nicht mehr verzichten möchte: Formulare werden im Hintergrund an den Server gesendet, der Benutzer kann

währenddessen auf der Seite weiterarbeiten. Ebenso wird die Antwort des Servers dynamisch in die Seite eingepflegt, ohne dass der Besucher auf das Neuladen warten muss. Beim Aufbau von großen Seiten mit vielen Bildern ist es üblich, zunächst nur das Basisgerüst der Seite an den Browser zu schicken und diesen dann zu veranlassen, die restlichen Inhalte wie Bilder oder Videos nachzuladen.

Um den Einsatz dieser Features allen Webentwicklern zu ermöglichen, sind im Lauf der letzten Jahre mächtige Frameworks entstanden, die in unterschiedlichem Umfang Funktionen mit sich bringen. Sie können meist mit wenig Aufwand eingebunden und genutzt werden. Welche JavaScript-Frameworks aktuell gebräuchlich sind, worin sie sich unterscheiden und wie sie im Vergleich abschneiden, wird in den nachfolgenden Abschnitten des Artikels beschrieben.

2 Von Prototypen, Sandkästen und Fußball-Mannschaften

An dieser Stelle sei zunächst auf die Programmiersprache JavaScript eingegangen. JavaScript wurde 1995 von Netscape veröffentlicht, zunächst unter dem Namen LiveScript. Ein Jahr später reagierte Microsoft mit seiner eigenen Script-Sprache JScript darauf. Auf den damals vorherrschenden „Browser-Krieg“ sei nur am Rande verwiesen.

Die European Computer Manufacturers Association beendete diesen Streit im Jahr 1997 mit der Veröffentlichung von ECMA-262 (ECMAScript), welcher als Standard für JavaScript anerkannt wurde und bis heute Gültigkeit hat (mittlerweile sogar in Form einer ISO-Norm). Aktuell befindet sich JavaScript in seiner Version 1.8 (seit 2008).

Die Sprache selbst wird als objektorientiert, aber klassenlos bezeichnet. Sie arbeitet mit sogenannten Prototypen. Das sind Objekte, die ihre Eigenschaften und Methoden an andere Objekte vererben können. Das Konstrukt von Klassen und Instanzen, das den Kern der meisten objektorientierten Sprachen (vgl. Java oder C#) bildet, kennt JavaScript nicht. Gleichzeitig kann JavaScript jedoch auch vollständig prozedural oder funktional programmiert werden. Ebenso ist eine Vermischung der verschiedenen Paradigmen möglich.

Die Ausführung von JavaScript unterliegt dem Sandbox-Prinzip. Das bedeutet in diesem Fall, es wird im Browser ausgeführt und kann lediglich auf Objekte des Browsers und der aktuellen Website zugreifen. Ein Verlassen dieses Kontextes ist nicht vorgesehen. So kann mit JavaScript beispielsweise nicht auf das Dateisystem des Rechners zugegriffen werden. Jede einzelne Browser-Seite wird außerdem isoliert behandelt. Das Script der einen Seite kann nicht auf andere zugreifen. Ebenso typisch für das Sandbox-Prinzip ist, dass JavaScript jederzeit im Browser deaktiviert werden kann.

Damit wären die Prototypen und Sandkästen erklärt, aber was hat es mit der Fußball-Mannschaft auf sich? Gemeint damit ist natürlich AJAX. Allerdings interessiert im Zusammenhang mit JavaScript nicht der niederländische Rekordmeister Ajax Amsterdam, sondern das Akronym für „Asynchronous JavaScript and XML“.

Dahinter verbirgt sich das Konzept der asynchronen Kommunikation zwischen Server und Browser, auf das in der Einleitung bereits eingegangen wurde (dynami-

sches Nachladen von Seiten, Verarbeiten von Formularen im Hintergrund usw.). Es kann mit Sicherheit als größte technologische Entwicklung im Rahmen von Web 2.0 bezeichnet werden. Näher eingegangen wird darauf in einem späteren Abschnitt, wenn es um den Vergleich der AJAX-Fähigkeiten der ausgewählten Frameworks geht.

Eine weitere Eigenschaft von JavaScript, die an dieser Stelle nicht unerwähnt bleiben sollte, ist die teilweise immer noch sehr unterschiedliche Implementierung in den einzelnen Browsern. Der Sprachkern von JavaScript ist zwar standardisiert, dennoch entwickelt jeder Browser-Hersteller die Sprache individuell weiter, ergänzt sie um eigene Befehle und Möglichkeiten, die dann wiederum von anderen Herstellern in meist leicht abgeänderter Form nachimplementiert oder durch komplett eigene Funktionen ersetzt werden.

Besonders in der Vergangenheit wurde es Webentwicklern dadurch nicht gerade leicht gemacht, JavaScript browserunabhängig zu entwickeln. Genau das ist aber auch ein weiterer Grund für die Notwendigkeit und den breiten Einsatz von Frameworks. Denn diese sind in den allermeisten Fällen so geschrieben, dass ihre mitgelieferten Funktionalitäten komplett browserunabhängig funktionieren.

3 Überblick

Die Masse an JavaScript-Frameworks ist nahezu unüberschaubar. Einige der geläufigsten sollen nachfolgend kurz vorgestellt werden.

3.1 jQuery

jQuery ist das wohl bekannteste und meist eingesetzte JavaScript-Framework. So nutzen beispielsweise 40 Prozent aller Seiten, die JavaScript einsetzen, jQuery (Stand April 2010). jQuery enthält umfassende Methoden zur DOM-Manipultion sowie ein eigenes Event-System, das es dem Programmierer ermöglicht, Benutzereingaben bequem abzufangen. Darüber hinaus erweitert jQuery durch viele Hilfsfunktionen das klassische JavaScript. Animationen und AJAX-Funktionalität sind standardmäßig implementiert und werden durch etliche jQuery-PlugIns noch erweitert.

Die eigentliche jQuery-Bibliothek umfasst nur eine einzige JavaScript-Datei, auf deren Funktionen der Entwickler durch einfaches Einbinden der Datei zugreifen kann. Mittlerweile ist jQuery auch in verschiedenen Entwicklungsumgebungen integriert, so etwa in Microsofts Visual Studio.

Veröffentlicht wurde jQuery erstmals 2006 von John Resig. Es kann wahlweise unter MIT- oder GPL-Lizenz genutzt werden.

3.2 MooTools

MooTools steht für My Object Oriented Tools. Die Bibliothek besteht aus mehreren Komponenten, die je nach Einsatzgebiet vom Entwickler individuell zusammengestellt werden können. Es umfasst ähnlich wie jQuery Funktionen für Drag and Drop, verschiedene weitere Effekte, ein Event-System und AJAX.

Eine Besonderheit dieses Frameworks ist, dass JavaScript damit im klassischen Sinne objektorientiert geschrieben werden kann, mit Klassen und Instanzen wie der

moderne Entwickler sie aus Java oder C# kennt. (Weiter-)Entwickelt wird MooTools von einer Entwickler-Community. Das Framework steht unter MIT-Lizenz.

3.3 Prototype

Prototype ist ein Basis-Framework, das lediglich grundlegende Funktionalitäten zur vereinfachten JavaScript-Programmierung (sogenannte Kurzbefehle oder Shortcuts) sowie Funktionen für AJAX und rudimentäre DOM-Manipulation mit sich bringt. Auf aufwändige Effekte wie Drag and Drop oder animierte Elemente wird dabei verzichtet. Prototype ist somit ein sehr schlankes, schnelles Framework.

Es wurde 2005 von Sam Stephenson entwickelt und steht wie die meisten anderen JavaScript-Frameworks unter der MIT-Lizenz.

Besonders anzumerken ist, dass auf Basis von Prototype weitere Frameworks entwickelt und veröffentlich wurden. Ein bekanntes Beispiel dafür ist script.aculo.us. Es erweitert Prototype um die fehlenden Effekte und Animationen. Ein weiteres Beispiel ist Ruby on Rails (ein Framework der Programmiersprache Ruby). Es implementiert Prototype in erster Linie wegen den AJAX-Funktionen, zusätzlich script.aculo.us für die GUI-Effekte.

3.4 YUI Library

Die Yahoo User Interface Library ist das einzige hier aufgeführte JavaScript-Framework, hinter dem ein großer Konzern und kein einzelner Entwickler oder eine Community steht. Entwickelt wurde es 2005 und steht mittlerweile unter BSD-Lizenz.

Im Vergleich zu den anderen Frameworks ist es ein absolutes Schwergewicht: Kommen die meisten JavaScript-Frameworks mit nur einer Quellcode-Datei aus (jQuery 92KB, MooTools 152KB und Prototype 160KB), so bringt die aktuelle YUI3 immerhin 1175 Quellcode-Dateien (insgesamt 11,3MB) mit sich.

Nicht nur der reine Code-Umfang, auch das Repertoire der YUI ist beeindruckend. Es gibt quasi nichts, was ein anderes Framework kann, was die YUI nicht ebenso ermöglicht und noch um etliche Variationen ergänzt. Das Framework verfügt auch über die ausführlichste Online-Demo.

Aus YUI hervorgegangen ist ein weiteres Framework namens Ext JS. Es war ursprünglich eine Erweiterung der YUI, wurde dann allerdings ausgegliedert und eigenständig weiterentwickelt. Es eignet sich besonders für Formular-Darstellungen sowie Menüoberflächen.

3.5 Dojo Toolkit

Das Dojo Toolkit ist ein umfassendes Framework mit allen relevanten Basisfunktionen. Es zeichnet sich durch ein Paket für die Darstellung von Tabellen und Schaubildern aus, wie sie beispielsweise aus Microsoft Excel oder vergleichbaren Tabellenkalkulationsprogrammen bekannt ist.

Entwickelt und verbreitet wird das Dojo Toolkit seit 2004 von der Dojo Foundation unter BSD- oder Academic Free License.

3.6 QooXDoo

Das FrameWork QooXDoo wurde 2009 von Mitarbeitern von 1&1 und GMX entwickelt und wird in deren Webmailern eingesetzt. Darüber hinaus steht es Entwicklern unter der LGPL zur Verfügung. Die Stärken von QooXDoo sind Oberflächen und Formulare. Es unterstützt außerdem explizit die Entwicklung für Smartphones und stellt dafür eigene Funktionen zur Verfügung.

4 Vergleich

Nach diesem kurzen Überblick sollen sich die einzelnen Frameworks nun im Vergleich beweisen. Für den praktischen Test antreten sollen

- jQuery in der Version 1.7, weil es das bekannteste Framework ist,
- Prototype in der Version 1.7, um herauszufinden, ob es dank Schlankheit in seinen Disziplinen wirklich schneller ist,
- YUI in der Version 3.4.1, um einen allseits würdigen Gegner für jQuery zu haben,
- MooTools in der Version 1.4.1

Es werden mehrere kleine Beispielanwendungen beschrieben, die jeweils mit allen vier Frameworks realisiert werden. In die spätere Bewertung soll einfließen, wie aufwändig die Programmierung war, wie zufriedenstellend das Ergebnis aus Entwickler- und Benutzersicht ist und wie die Reaktionszeiten der Webanwendung sind.

4.1 Vergleichsaufbau

Um vergleichbare Ergebnisse zu erhalten, finden die Tests alle auf demselben Betriebssystem (Windows 7, 64-bit) und im selben Browser (Mozilla Firefox 8) statt. Die Quelldateien der Frameworks liegen jeweils auf derselben Ebene im Dateisystem. Für jeden Test mit jedem Framework wird eine eigene HTML-Datei angelegt, die folgendem Aufbau entspricht:

```
<html>
    <head>
        <title>Test</title>
        <script type="text/javascript"
                src="../framework/framework.js">
        </script>

        <script type="text/javascript">
            function run() {
                // Individueller JavaScript-Code
            }
        </script>
    </head>
    <body onload="run()">
        <div id="main"></div> // HTML-Seiteninhalt
    </body>
</html>
                                                (index.html)
```

Die JavaScript-Funktion run() wird in den einzelnen Testfällen mit dem entsprechenden Code befüllt. Sie wird nach dem Laden der Seite durch das onload-Attribut des body-Tags aufgerufen. Es gibt zwar (in jQuery und YUI) Konstrukte, mit denen dieser Aufruf eleganter gelöst werden könnte, doch um eine optimale Vergleichbarkeit der Frameworks zu erreichen, sollte der komplette Aufbau identisch sein.

Für die Untersuchung der Ergebnisse und die Zeitmessungen wird das Firefox-Addon Firebug in der Version 1.8.4 eingesetzt.

4.2 Dynamisches Nachladen von Inhalten

Die erste Disziplin lautet AJAX. Dafür wird im Testverzeichnis eine Datei ajax.html angelegt, die einen beliebigen HTML-Code enthält, in diesem Fall einen zufällig generierten Blindtext.

Das Ziel ist es nun, den Inhalt dieser Datei dynamisch, d.h. nach dem eigentlichen Laden der Seite nachzuladen und den zusätzlichen Inhalt in das div-Tag (main) zu schreiben. Das Beispiel ist einfach, aber realistisch, da darin eine der grundlegendsten Anforderungen an ein JavaScript-Framework besteht.

Die Implementierung mittels jQuery, MooTools und Prototype funktioniert problemlos mittels Einzeiler. Auffällig ist direkt die ähnliche Syntax zwischen MooTools und jQuery.

```
function run() {
    $('#main').load('ajax.html');
}
```
(1_jquery.html)

```
function run() {
    $('main').load('ajax.html');
}
```
(1_mootools.html)

```
function run() {
    new Ajax.Updater('main', 'ajax.html');
}
```
(1_prototype.html)

```
function run() {
    YUI().use("io-base", "node", function(Y) {
        Y.io('ajax.html', {
            on : {
                success: function (id, result) {
                    Y.one('#main').set("innerHTML",
                                       result.responseText);
                }
            }
        });
    });
}
```
(1_yui.html)

YUI hingegen wirkt bei diesem minimalen Beispiel im Vergleich sehr schwerfällig. Bereits bei der Suche in den API Docs nach den richtigen Befehlen wird schnell klar, dass das Framework tatsächlich sehr mächtig aber damit leider auch unübersichtlich ist.

Dieser erste Eindruck wird durch die Zeitmessung jedoch nicht bestätigt, wie die nachfolgende Tabelle 1 zeigt:

Framework	**Messung 1**	**Messung 2**	**Messung 3**	**Durchschnitt**
jQuery	42 ms	49 ms	47 ms	46,0 ms
MooTools	48 ms	49 ms	49 ms	48,7 ms
Prototype	57 ms	69 ms	47 ms	57,7 ms
YUI	3 ms	6 ms	6 ms	5,0 ms

Tabelle 1: Zeitmessung AJAX-Vergleich

Wenig überraschend ist, dass jQuery und MooTools eng beieinander liegen, nachdem ihre Syntax bereits ähnlich ist. Es bleibt hier abzuwarten, wie sie sich in den weiteren Versuchen unterscheiden. Etwas enttäuschend ist, dass Prototype sehr schwach abschneidet, obwohl es als reines Basis-Framework gerade im AJAX-Bereich überzeugen sollte. Am interessantesten ist jedoch sicherlich das Ergebnis von YUI, das offensichtlich trotz gigantischem Funktionsumfang sehr performant ist. Bei einem Zeitunterschied um nahezu Faktor 10 geht es trotz aufwändigerer Programmierung als klarer Sieger aus diesem ersten Vergleich hervor.

4.3 Formularvalidierung

Der zweite Vergleich bezieht sich auf die Validierung von Formularen. Im Web2.0 ist es üblich, dass Benutzereingaben direkt vom Browser per JavaScript hinsichtlich verschiedener Kriterien überprüft werden und der Benutzer bei Falscheingabe eine unmittelbare Rückmeldung erhält, ohne lange auf die serverseitige Verarbeitung des Formulars warten zu müssen.

In der nachfolgenden Beispielanwendung soll ein Formular für einen Eintrag in ein beliebiges Gästebuch validiert werden. Mindestanforderung an die Validierung sollte sein, dass Pflichtfelder als solche erkannt und überprüft werden sowie Email-Adressen auf ihre syntaktische Korrektheit hin kontrolliert werden.

Bei jQuery ist schnell klar, dass es zwar Hilfsfunktionen gibt, um bequem auf einzelne Formularelemente zuzugreifen, der Entwickler jedoch immer noch einen großen Teil selbst implementieren muss. Die Suche nach einer geeigneten Erweiterung, einem sogenannten jQuery-Plugin, ergibt jedoch einige Treffer. Die Wahl fällt auf das Validation-Plugin von Jörn Zaefferer (www.bassistance.de/jquery-plugins). Er liefert auch den nachfolgenden Beispielcode.

```
function run() {
    $("#testform").validate();
}

…

<form class="cmxform" id="testform">
    <fieldset>
        <legend>Gästebucheintrag</legend>
        <p>
            <label for="cname">Name</label>
            <em>*</em>
            <input id="cname" name="name" size="25"
                   class="required" minlength="2" />
        </p>
        <p>
            <label for="cemail">E-Mail</label>
            <em>*</em>
            <input id="cemail" name="email" size="25"
                   class="required email" />
        </p>
        <p>
            <label for="curl">URL</label>
            <em>  </em>
            <input id="curl" name="url" size="25"
                   class="url" value="" />
        </p>
        <p>
            <label for="ccomment">Your comment</label>
            <em>*</em>
            <textarea id="ccomment" name="comment"
                      cols="22" class="required">
            </textarea>
        </p>
        <p>
            <input class="submit" type="submit"
                   value="Submit"/>
        </p>
    </fieldset>
</form>
                                                (2_jquery.html)
```

Das Einbinden des Validators ist lediglich eine Zeile JavaScript-Code. Welche Formularfelder wie zu validieren sind, wird über das class-Attribut der einzelnen input-Tags im HTML-Code festgelegt. So steht „required“ beispielsweise für ein Pflichtfeld und „email“ für ein Email-Feld. Diese Anweisungen lassen sich beliebig kombinieren. Damit erfüllt jQuery erst einmal alle Anforderungen an diesen Versuch.

Wiederum nahezu identisch verläuft der Test mit MooTools. Auch hier ist ein weiteres Plugin nötig. MooTools bietet auf seiner Website den sogenannten MoreBuilder an, mit dem man sich alle gewünschten Funktionalitäten in eigene Plugins zusammen-

packen kann. Der Quellcode, den der Entwickler in seine Website einbauen muss, unterscheidet sich von jQuery ebenso nur minimal.

```
function run() {
    new Form.Validator.Inline(document.id('testform'));
}

…

        <p>
            <label for="cemail">E-Mail</label>
            <em>*</em>
            <input id="cemail" name="email" size="25"
                   class="required validate-email" />
        </p>
                                          (2_mootools.html)
```

Die HTML-Angaben für das Formular sind bis auf eine Kleinigkeit identisch: in MooTools-Syntax lautet die Definition für ein Email-Feld „validate-email“ anstatt „email“.

Prototype versagt bei der Formularvalidierung jedoch seine Dienste. Wie bereits bei jQuery angedeutet, vereinfacht es dem erfahrenen JavaScript-Entwickler durch einige Hilfsfunktionen zwar die Arbeit an der einen oder anderen Stelle. Von einer Out-of-the-box-Lösung kann jedoch nicht die Rede sein.

Ebenso sieht es auch mit YUI aus. Eine Formularvalidierung ist nur mit einigem Aufwand realisierbar und die Dokumentation in diesem Bereich ist alles andere als ausreichend.

Fazit: In der Kategorie Formularvalidierung sind jQuery und MooTools ähnlich brauchbar. Auch was die Reaktionszeit angeht, unterscheiden sie sich kaum. Einzig die Tatsache, dass MooTools die Funktionalitäten als eigene Erweiterung bereitstellt, während bei jQuery das Plugin eines Drittanbieters genutzt wird, ist als Pluspunkt für MooTools zu werten. Prototype und YUI scheiden in diesem Vergleich aus.

4.4 Animation

Eine Animation, die man mittlerweile auf vielen Webseiten findet, ist der sogenannte Accordion-Effekt. Dahinter verbirgt sich das fließende Auf- oder Zuklappen eines Bereichs, meistens unter einer entsprechend formatierten Überschrift. Der Effekt ist nicht groß, allerdings ohne Framework nicht trivial zu implementieren, vor allem nicht performant.

jQuery bringt hier erneut keine fertige Lösung mit sich. Dafür wird die Erweiterung jQuery UI benötigt. Ähnlich wie die erweiterte Bibliothek der MooTools lässt sie sich auf der jQuery-Website konfigurieren und mit individuellem Funktionsumfang herunterladen. Ist die Erweiterung eingebunden, geht es gewohnt einfach weiter.

```
function run() {
    $("#accordion").accordion();
}
```

```
…

<div id="accordion">
    <h3><a href="#">Titel 1</a></h3>
    <div>
        <p>Dieser Text wird bei Klick auf Titel 1
            eingeblendet und bei Klick auf einen anderen
            Titel wieder ausgeblendet.</p>
    </div>
    <h3><a href="#">Titel 2</a></h3>
    <div>
       <p>Gleiches Verhalten wie oben.</p>
    </div>
</div>
                                                (3_jquery.html)
```

Es ist also lediglich auf die Verwendung der richtigen HTML-Tags zu achten und schon funktioniert der Accordion-Effekt. Nahezu simultan verhält sich auch hier die MooTools-Syntax.

```
function run() {
    new Fx.Accordion($('accordion'),
                     '#accordion h2',
                     '#accordion .content');
}

…

<div id="accordion">
    <h2>Titel 1</h2>
    <div class="content">
        <p>Dieser Text wird bei Klick auf Titel 1
            eingeblendet und bei Klick auf einen anderen
            Titel wieder ausgeblendet.</p>
    </div>
    <h2>Titel 2</h2>
    <div class="content">
       <p>Gleiches Verhalten wie oben.</p>
    </div>
</div>
                                              (3_mootools.html)
```

Das Ergebnis ist bei beiden Frameworks kaum zu unterscheiden. jQuery wirkt bei der Ausführung minimal flüssiger. Messen lässt sich diese Empfindung jedoch nicht.

YUI bietet für dieses Beispiel eine Unmenge von Möglichkeiten. Das folgende Beispiel entspricht im Ergebnis nicht ganz den anderen beiden, erfüllt jedoch dieselbe Funktion.

```
function run() {
    YUI().use('anim', function(Y) {
        var module = Y.one('#accordion');
        var content =
         module.one('.content').plug(Y.Plugin.NodeFX, {
            from: { height: 1 },
            to: {
                height: function(node) {
                    return node.get('scrollHeight');
                }
            },
            easing: Y.Easing.easeOut,
            duration: 0.5
        });
        var onClick = function(e) {
            module.toggleClass('hidden');
            content.fx.set('reverse',
                           !content.fx.get('reverse'));
            content.fx.run();
        };
        var control = Y.Node.create(
            '<a title="show/hide content"
                class="yui3-toggle">
                <em>toggle</em>
             </a>'
        );
        module.one('.header').appendChild(control);
        control.on('click', onClick);
    });
}

…

<div id="accordion">
    <div class="header">
        <h2>Titel</h2>
    </div>
    <div class="content">
        <p>Dieser Text wird bei Klick auf toggle
           ein- oder ausgeblendet.</p>
    </div>
</div>
                                                (3_yui.html)
```

Die Effekte von jQuery und MooTools lassen sich selbstverständlich auch mit den verschiedensten Optionen (z.B. duration) modifizieren. An den riesigen Umfang der Einstellungsmöglichkeiten von YUI kommen sie jedoch nicht heran. Dafür lässt sich aber vor allem mit jQuery eine schnelle, schlanke Implementierung realisieren, was mit YUI nicht möglich ist.

Je nach dem, auf was der Entwickler mehr Wert legt (einfache Implementierung oder umfangreiche Anpassbarkeit) gewinnt in diesem Vergleich jQuery / MooTools oder YUI. Klarer Verlierer ist Prototype, da dieses Framework überhaupt keine vergleichbaren Funktionen bereitstellt.

4.5 Popups mit Drag & Drop

Der letzte Vergleich ist gleichzeitig auch der scheinbar aufwändigste. Es soll aus einer Webseite heraus ein Popup mit beliebigem Inhalt geöffnet werden. Dieses Popup soll kein neues Browser-Fenster sein, sondern ein mittels CSS entsprechend angepasster div-Container, der sich im Vordergrund befindet und alle gewöhnlichen Eigenschaften eines eigenen Fensters aufweist (schließbar mittels Button und verschiebbar per Drag and Drop).

Prototype wird in dieser Kategorie erneut nicht antreten, dafür sollten die anderen drei Frameworks gute Ergebnisse liefern. Das Beispiel ist für Web2.0-Anwendungen unverzichtbar, entsprechend hoch sind die Erwartungen.

Die Lösung in jQuery für diese Aufgabe heißt dialog und ist erneut Bestandteil von jQuery UI. Die Implementierung ist trotz der umfangreichen Anforderungen einfach.

```
function run() {
    $( "#dialog" ).dialog();
}

…

<div id="dialog" title="Neues Fenster">
   <p>Dies ist der Inhalt des Popups.</p>
</div>
```
(4_jquery.html)

Der JavaScript-Aufruf ist so simpel wie bei allen anderen jQuery-Beispielen. Dabei könnte der Methode dialog() noch eine ganze Liste an Parametern mitgegeben werden, womit die Eigenschaften des Popups definierbar sind. Was beim Standardaufruf ohne Parameter herauskommt, ist jedoch genau das, was die Aufgabe verlangt: ein Popup mit Titelleiste und Inhalt, Schließen-Button, Drag and Drop per Klick auf die Titelleiste sowie darüber hinaus die Möglichkeit, die Fenstergröße per Ziehen mit der Maus zu verändern.

Nach drei nahezu identischen Tests wäre an dieser Stelle zu erwarten gewesen, dass MooTools zumindest eine ähnliche Funktionalität mit sich bringt. Dem ist allerdings nicht so. Weder die Kernfunktionalitäten noch die zusätzlichen Module enthalten etwas, das einem Popup oder einer Dialogbox ähnlich kommt. Die Grundfunktionalitäten wie Drag and Drop sind zwar verfügbar, doch das eigentliche Popup müsste sich der Webentwickler nun selbst zusammenbauen.

Eine ausführliche Suche nach Plugins von Drittanbietern brachte zwar einige Ergebnisse, die jedoch in keinster Weise den gestellten Anforderungen entsprechen. Sie basieren fast alle auf dem Konzept der sogenannten Lightbox, die für die Anzeige von

Bildern im Vordergrund verwendet wird und nicht verschiebbar ist. Sie verdunkelt außerdem den Hintergrund, was bei einem normalen Popup nicht gewünscht ist. Damit scheidet MooTools in diesem Vergleich aus.

YUI hingegen bietet alle nötigen Komponenten. Mit etwas Aufwand lassen sie sich zu dem gewünschten Ergebnis zusammenbasteln.

```
function run() {
    YUI().use('transition', 'panel', 'dd-plugin',
    function (Y) {
        var panel, bb;

        function showPanel() {
            panel.show();
            bb.transition({
                duration: 0.5,
                top     : '80px'
            });
        }

        function hidePanel() {
            bb.transition({
                duration: 0.5,
                top     : '-300px'
            }, function () {
                panel.hide();
            });
        }

        panel = new Y.Panel({
            srcNode: '#panel',
            width  : 330,
            xy     : [300, -300],
            zIndex : 5,
            modal  : true,
            visible: false,
            render : true,
            buttons: [
                {
                    value  : 'Close',
                    section: 'footer',
                    action : function (e) {
                        e.preventDefault();
                        hidePanel();
                    }
                }
            ]
        });
```

```
        panel.plug(Y.Plugin.Drag,
                   {handles:['.header']});
        bb = panel.get('boundingBox');
        showPanel();
    });
}

…

<div id="panel">
    <div class="header">
        Neues Fenster
    </div>
    <div class="content">
        <p>Hier steht der Inhalt des Popups.</p>
    </div>
</div>
```

(4_yui.html)

Wie das vorangegangene Beispiel zeigt auch dieser Vergleich wieder, dass jQuery und YUI zu einem ähnlichen Ergebnis führen, wobei jQuery einfach einzubinden ist und nur bei Bedarf durch zusätzliche Optionen angepasst werden muss, während YUI von vornherein nur Bausteine zur Verfügung stellt, die in jedem Fall mit mehr Aufwand kombiniert werden müssen.

5 Fazit

Diese Vergleiche ließen sich endlos fortsetzen. Mit vier grundlegenden Beispielen kann man sicher nicht abschließend sagen, welches Framework nun das beste, und welches das schlechteste ist. Sicherlich kommt das auch komplett auf den Einsatzzweck und persönliche Vorlieben des Entwicklers an. Dennoch zeichnen sich deutliche Tendenzen ab und um es nicht endlos in die Länge zu ziehen, seien an dieser Stelle noch einmal kurz die bisherigen Ergebnisse zusammengefasst und als Anregung zu weiteren Nachforschung an die Leserschaft gerichtet.

Das Framework Prototype wurde in die Vergleiche mit einbezogen, da es sich als Basis-Framework nur auf Kernfunktionalitäten, vor allem AJAX, beschränkt. Dass es sich in den Oberflächen-bezogenen Vergleichen wie Accordion-Effekt und Popups nicht gut bzw. überhaupt nicht schlagen würde, war von Anfang an klar. Im ersten Versuch, dem dynamischen Nachladen von Inhalten, hat es mit der längsten gemessenen Ausführungsdauer jedoch genauso schlecht abgeschnitten. Damit lässt sich wohl feststellen, dass Prototype nicht das effektivste aller Frameworks ist.

YUI war zunächst als umfangreichstes Framework angetreten und hat die Gegner beim AJAX-Vergleich fast um Faktor zehn geschlagen. Bei der Formularvalidierung konnte mit YUI mit angemessenem Aufwand leider kein Ergebnis erreicht werden, dafür bei den restlichen Vergleichen. Anzumerken ist hier, dass der Aufwand für den Webentwickler bei allen Beispielanwendungen für den YUI-Code am größten ist und

die Einarbeitung aufgrund des großen Funktionsumfangs aufwändiger ist als bei den anderen Frameworks.

jQuery und MooTools haben in den ersten drei Vergleichen nahezu identisch abgeschnitten. Ebenso war der Aufwand für den Entwickler relativ gleich und es ist aufgefallen, dass große Teile der Syntax ähnlich sind. Beim letzten Beispiel, dem Popup mit Drag and Drop, hat MooTools deutlich gegen jQuery verloren, ebenso gegen YUI.

Faktisch hat jQuery das Rennen gewonnen. Mit Hilfe des Frameworks konnten alle Aufgaben mit minimalem Aufwand und guten Ergebnissen gelöst werden. Alle anderen Frameworks konnten mindestens eine Aufgabenstellung nicht ausreichend erfüllen. Jedoch konnte YUI die AJAX-Aufgabe mit Abstand am besten lösen, deshalb fällt das Ergebnis bei weitem nicht so eindeutig aus.

Abschließend bleibt festzustellen, dass die JavaScript-Frameworks-Welt nicht schwarz und weiß ist. Jeder Entwickler muss weiterhin selbst herausfinden, für welche Anwendungsfälle er ein Framework einsetzen will und dann entscheiden, welches für ihn das Beste ist. Wichtig ist lediglich, dass er sich entscheidet. Mehrere Frameworks zu mischen, ist auf keinen Fall eine Lösung. Doch darüber wurde an derer Stelle schon genug geschrieben.

6 Quellen

- http://webstandard.kulando.de/post/2008/08/21/top-10-aller-javascript-frameworks
- http://jquery.com
- http://mootools.net
- http://prototypejs.org
- http://dojotoolkit.org
- http://www.yuilibrary.com
- http://www.sencha.com/products/extjs
- http://qooxdoo.org
- http://bassistance.de/jquery-plugins/jquery-plugin-validation