KaffeeKlatsch
Das Magazin rund um Software-Entwicklung

ISSN 1865-682X
08/2012

Web2touch

Web-Entwicklung für mobile Endgeräte mit jQuery Mobile

von Werner Eberling

Web2touch

Web-Entwicklung für mobile Endgeräte mit jQuery Mobile

von Werner Eberling

Bei der Entwicklung einer Web-Anwendung kommt man in vielen Bereichen heute nicht mehr darum herum, sein Angebot auch für mobile Endgeräte zu optimieren. *JQuery Mobile* bietet hierfür eine interessante und leicht zu erlernende Plattform an, die in diesem Artikel vorgestellt werden soll.

Ob On- oder Offline, für das Aussehen und die Bedienung von Applikationen auf mobilen Endgeräten (Smartphones oder Tablets) gelten eigene Regeln. Bedienelemente, die mit einem Mauszeiger noch einfach zu treffen waren, sind für die dagegen im ersten Moment doch etwas plumper anmutende *Touch*-Bedienung unbrauchbar. Hier sollte ein *Button* schon eine gewisse Grösse besitzen, um ihn auch sinnvoll treffen zu können.

Diese besonderen Anforderungen an die Benutzerschnittstelle stellen eine besondere Herausforderung für eine Klasse von Anwendungen dar, die im ersten Moment auf mobilen Geräten genauso funktionieren sollten, wie auf dem klassischen Laptop oder Desktop PC: Web-Anwendungen. Obwohl heute jedes Smartphone oder Tablet mit einem mobilen Browser ausgestattet ist, bedeutet dies noch lange nicht, dass mit diesen Geräten auch jede Web-Seite zu bedienen ist. Genau hier kommt jQuery Mobile [1] ins Spiel. Seines Zeichens ein Web-Framework für die Erstellung von Anwendungen für *Touchscreen*-basierte Endgeräte – also Tablets oder Smartphones.

Progressiv und unaufdringlich

Oh je, ein *UI-Framework* basierend auf *JavaScript!* Bedeutet das nicht Unmengen an unlesbarem *Script-Code*, der auf magische Art und Weise die buntesten Effekte auf den Browser zaubert und im entscheidenden Moment eine weiße Seite darstellt, weil irgendwo in den Tiefen des Codes ein Semikolon fehlt?

Nicht ganz, obwohl jQuery drauf steht (und natürlich auch jQuery drin ist), wird hier niemand gezwungen JavaScript-Code zu schreiben. Ganz im Gegenteil. Als HTML5-basiertes Framework verfolgt jQuery Mobile die Konzepte des *unobtrusive JavaScript* [2]. Diese umfassen u. a. die strikte Trennung zwischen Struktur, Layout und Verhalten einer Web-Seite. Für jede dieser Aufgaben kommt dabei jeweils die Technologie zum Einsatz, die dafür (im Web-Umfeld) am besten geeignet ist:

- *HTML* – zur Beschreibung der Seitenstruktur
- *CSS* – zur Definition des Layouts
- *JavaScript* – zur Implementierung des Verhaltens

Darüber hinaus wird sehr viel Wert auf die Ideen des sogenannten *Progressive Enhancements* gelegt, was nicht weniger bedeutet, als dass eine mit jQuery Mobile erstellte Web-Seite auch dann noch bedienbar ist, wenn der Browser die benötigte HTML/CSS-Version oder vielleicht überhaupt kein JavaScript unterstützt. Die JavaScript-Erweiterungen dürfen damit nicht zu Voraussetzungen für die korrekte Funktionsweise der Seite werden.

Offiziell unterstützte Plattformen (jQuery Mobile 1.1.1)
Android
Blackberry
iOS
palm webOS
Windows Phone
Samsung Bada
Symbian
MeeGo

Los geht's

Aber was bedeutet das jetzt im Detail? Grundsätzlich ist eine mit jQuery Mobile erstellte Web-Seite erst einmal nichts anderes als eine HTML-Seite nach dem HTML5-Standard, was sich durch den *DOCTYPE* am Anfang der Seite ausdrückt:

```
<!DOCTYPE html>
```

Nachdem es ja nicht bei „einfachem" HTML bleiben soll, muss im HTML-Kopf der Seite die *jQuery Mobile Library* und das dazugehörende *Stylesheet* eingebunden werden. Hierbei ist darauf zu achten, dass jQuery Mobile seinerseits wieder jQuery benötigt (aktuell in Version 1.7.1), das ebenfalls eingebunden werden möchte.[1]

[1] Bei der Auswahl der jQuery-Version ist darauf zu achten, dass immer genau die Version verwendet wird, die bei der jQuery Mobile-Version der Wahl angegeben ist. Hier gilt nicht das Konzept neuer ist besser, so dass die Verwendung einer aktuelleren jQuery-Version zu ungewollten Effekten führen kann.

```
<link rel="stylesheet"
  href="css/jquery.mobile-1.1.1.min.css" />
<script src="js/jquery-1.7.1.min.js" />
<script src="js/jquery.mobile-1.1.1.min.js" />
```

Falls die benötigten Bibliotheken nicht innerhalb der Anwendung mit ausgeliefert werden sollen, ist auch die Verwendung des jQuery eigenen *Content Distribution Networks* (CDN) möglich: http://code.jquery.com/mobile/1.1.1/jquery.mobile-1.1.1.min.css

Um Probleme mit der Darstellung der Seite zu umgehen, sollten zusätzlich noch die gewählte Zeichencodierung und der initial zu verwendende Zoom-Faktor angegeben werden:

```
<meta charset="UTF-8" />
<meta name="viewport"
  content="width=device-width, initial-scale=1"/>
```

Speziell das Fehlen der *Viewport*-Angabe kann zu ungewünschten Darstellungen auf den Zielgeräten führen und den Benutzer zu unnötigen Zoom-Aktionen zwingen.

Die erste Seite

Nach der geleisteten Vorarbeit wird es jetzt Zeit sich dem Inhalt der Seite, also dem *HTML-Body*, zu widmen. Da HTML lediglich für die Struktur verwendet wird, kommen hier vor allem *DIV*-Blöcke zum Einsatz. Eine jQuery Mobile HTML-Seite besteht dabei aus mindestens einer logischen Seite, die durch ein *DIV* mit der Rolle *page* definiert wird. Diese Seite ist wiederum in die folgenden drei Bereiche aufgeteilt.

- *header* – für den oberen Navigationsbereich
- *content* – für den eigentlichen Inhalt der Seite
- *footer* – für den unteren Navigationsbereich

```
<div data-role="page">
  <div data-role="header">
    <h1>Meine JQM App</h1>
  </div>

  <div data-role="content">
    <p>Lorem ipsum...</p>
  </div>

  <div data-role="footer">
    <h4>Kaffeklatsch 08/2012</h4>
  </div>
</div>
```

Das Ergebnis sieht, dank der fleissigen Hintergrundarbeit von jQuery Mobile und den darin enthaltenen Styles, schon überraschend ansehnlich aus:

Abbildung 1

Auch der Effekt, dass keine *Locationbar* mehr im Browser zu sehen ist, ist jQuery Mobile zu verdanken, das einfach dafür sorgt, dass diese aus dem Bild geschoben wird, um den Eindruck einer nativen Anwendung zu vermitteln.

Aus eins mach zwei

Mit einer Seite ist natürlich noch kein Staat zu machen. Daher kann eine jQuery Mobile-Seite mehrere logische Seiten enthalten, zwischen denen über die Verwendung einfacher HTML-Anker gewechselt werden kann. Allerdings muss hierzu jeder Seite eine eindeutige *id* zugewiesen werden. Soll der Übergang dann noch mit einem speziellen Effekt erfolgen[2], so kann dieser innerhalb des Anker-Elements angegeben werden:

```
<div data-role="page" id="p1">
  …
    <div data-role="content">
      <a href="#p2"
        data-transition="slide">weiter</a>
    </div>
  …
</div>
<div data-role="page" id="p2">
  …
    <div data-role="content">
      <a href="#p1"
        data-transition="slide"
        data-direction="reverse">zurück</a>
    </div>
  …
</div>
```

Neben dem hier verwendeten *Slide*-Effekt gibt es noch eine Reihe weiterer Übergänge, die einfach durch Anga-

[2] Leider ist der Übergangseffekt in gedruckter Form nicht wirklich darstellbar, daher sei dem interessierten Leser hier der Selbstversuch empfohlen.

be des entsprechenden Namens verwendet werden können. Bei Interesse sei hier auf die jQuery Mobile Website [1] oder direkt auf die Seite von *jQtouch* [3] verwiesen, das sich für die Implementierung dieser Effekte verantwortlich zeichnet.

Das obligatorische „Aber“

Was hier gerade so einfach aussieht, kann je nach Zielgerät leider ziemliche Probleme mit sich bringen. Das für die Übergangseffekte verwendete *Plugin* hat leider Probleme auf bestimmten Android-Versionen. So machen die Übergänge auf einem *HTC Incredible S* mit *Android 2.3* – oder wahlweise auch *Android 4* – leider nur im mobilen Firefox wirklich Freude. Der Android eigene Browser kommt hier leider ziemlich ins schwitzen oder besser gesagt ins ruckeln. Auf *iOS*-Geräten hingegen laufen die Animationen ohne größere Probleme.

Vom Link zum Button

Die gerade verwendeten Links sind natürlich nicht das, was man sich in einer mobilen Anwendung als Auswahlelemente vorstellt. Hier sollten es dann schon die klassischen Buttons sein, die dank ihrer Größe dem evtl. nicht ganz so treffsicheren Benutzer deutlich entgegen kommen. Die hierfür notwendige Änderung ist, dem allgemeinen Schema folgend, wieder überraschend einfach. So muss dem verwendeten Anker lediglich mitgeteilt werden, dass er die Rolle *button* einnehmen soll:

```
<a href="#p2" data-role="button">weiter</a>
```

Das konkrete visuelle Ergebnis dieser Änderung hängt davon ab, in welchem Bereich der Seite der Anker verwendet wurde. Wird er im *Content*-Bereich gesetzt, ist das Ergebnis ein über die gesamte Bildschirmseite reichender Knopf, der den Übergang zur nächsten Seite ermöglicht. Im *Header* oder *Footer* verwendet, wird er als kleiner Navigationsknopf sichtbar. Im Header ist dabei zu beachten, dass der erste Knopf links und der zweite angegebene Knopf rechts vom Titel erscheint. Im Footer hingegen werden die Knöpfe einfach von links nach rechts angeordnet.

Neben Text ist es auch möglich *Icons* in Buttons zu verwenden. Dafür stellt jQuery Mobile eine Auswahl an vordefinierten Icons bereit, die aber jederzeit erweitert werden können. Die Position relativ zum Text kann über das Attribut *data-iconpos* angegeben werden:

```
<a href="#p2" data-transition="slide"
   data-role="button" data-icon="arrow-r"
   data-iconpos="right">weiter</a>
```

Abbildung 2

Im Dialog

Nicht immer möchte man dem Benutzer den Inhalt in Form einer kompletten Seite präsentieren. Bei bestimmten Aktionen, wie z. B. dem Löschen von Datensätzen, soll lediglich eine Meldung „vor“ der aktuell angezeigten Seite erscheinen.[3] Dies ist mit dem Einsatz von Dialogen möglich.

Ob ein Inhalt als Dialog dargestellt wird, wird entweder bei der Definition des Anker-Elements entschieden, das auf den Inhalt verweist,

```
<a href="#p2" data-rel="dialog">weiter</a>
```

oder ein bestimmter Inhalt wird direkt in Form eines Dialogs definiert – analog zur Definition einer Seite:

```
<div data-role="dialog" id="d1">
  <div data-role="header">
    <h1>JQM Dialog</h1>
  </div>
  <div data-role="content">
    <p>Sind Sie sicher, daß Sie das tun wollen?</p>
    <a href="#p2" data-role="button">Ja</a>
    <a href="#" data-role="button"
      data-rel="back">Nein</a>
  </div>
  <div data-role="footer">
    <h4>Kaffeklatsch 08/2012</h4>
  </div>
</div>
```

Entdecke die Möglichkeiten

Die Navigation zwischen verschiedenen Seiten und der Anzeige von Dialogen ist natürlich noch lange nicht das Ende der Fahnenstange. JQuery Mobile bietet ei-

[3] Auch wenn die „Hintergrundseite“ nicht mehr wirklich sichtbar ist, kann über den standardmässig verwendeten Übergangseffekt *pop* der Anschein erweckt werden, der Dialog würde sich vor der zuletzt gesehenen Seite öffnen.

nen bunten Strauß an UI-Komponenten, von Formular-Elementen über Listen und Tabellen, bis hin zu *Tool-* und *Navigation-Bars*. All diese Elemente hier im einzelnen vorzustellen würde den Rahmen dieses Artikels sprengen, daher sei der geneigte Leser hier auf die einschlägige Literatur [4] oder die *Demo-Website* von jQuery Mobile [5] verwiesen. Als Beispiel soll hier stellvertretend das Akkordion dienen, also die Darstellung einer Liste mit ausklappbarem Inhalt, bei der sich jeweils das ausgeklappte Element schließt, wenn ein Neues geöffnet wird.

Abbildung 3

Klapp ein, klapp aus

Wie gewohnt werden die einzelnen Elemente wieder als *DIV*-Blöcke definiert. Das gesamte Akkordeon wird durch einen *DIV*-Block mit der Rolle *collapsible-set* beschrieben, der die eigentlichen ausklappbaren Elemente enthält. Jedes dieser ausklappbaren Elemente wird dabei wieder als *DIV*, diesmal mit der Rolle *collapsible*, beschrieben.

Standardmäßig sind alle Elemente geschlossen. Soll der Inhalt eines der Elemente direkt beim Start sichtbar sein, muss für dieses im entsprechenden *DIV* das Attribut *data-collapsed* auf *false* gesetzt werden.

Der eigentliche Inhalt des aufklappbaren Elements wird innerhalb des *DIVs* angegeben. Er besteht aus einem *Heading*-Element und beliebigem weiteren Inhalt, der in der Regel von einem Paragraphelement umgeben ist. Die Überschrift wird dabei als Titel genommen und ist auch im zugeklappten Zustand sichtbar. Der restliche Inhalt wird nur angezeigt, wenn das Element aufgeklappt ist:

```
<div data-role="content">
  <div data-role="collapsible-set">
    <div data-role="collapsible"
      data-collapsed="false">
      <h1>Collapsible #1</h1>
      <p>Lorem ipsum...</p>
    </div>
    <div data-role="collapsible">
      <h1>Collapsible #2</h1>
      <p>Lorem ipsum...</p>
    </div>
    ...
  </div>
</div>
```

Wird diese Seite in einem Browser geöffnet, wird deutlich, dass sich jQuery Mobile wieder um alle notwendigen Details bzgl. der Darstellung und inkl. der Anzeige entsprechender Icons sowie um die Logik bzgl. Öffnung und Schließung einzelner Elemente kümmert.

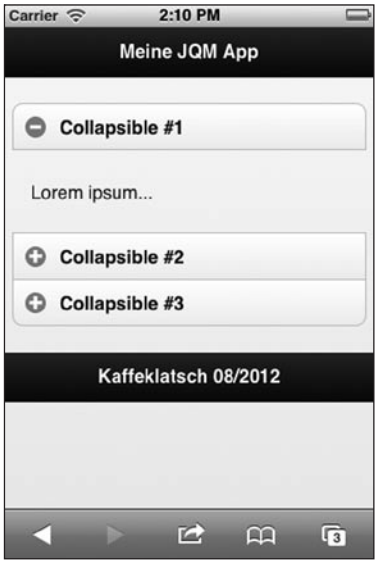

Abbildung 4

Endlich JavaScript

Nach fast vier Seiten eines Artikel über ein JavaScript-basiertes *UI-Framework* wird es nun langsam Zeit auch einmal wirkliches JavaScript zu sehen. Obwohl es schon als sehr erfreulich anzumerken ist, wie weit man bei jQuery Mobile ohne eine einzige Zeile JavaScript kommen kann.

JQuery Mobile präsentiert sich mit einem für die ersten Schritte sehr guten Standardverhalten. Sobald es in die Entwicklung komplexerer Seiten geht, sieht man sich aber doch hin und wieder der Aufgabe gegenüber, dieses zu ändern bzw. dynamisch die Inhalte von Seiten anzupassen. Hierfür bietet jQuery Mobile eine auf jQue-

ry aufbauende JavaScript-API an, die es ermöglicht mit programmatischen Mitteln Einfluss auf das Verhalten der Anwendung zu nehmen.

Um beispielsweise vorhandene *Defaults* zu ändern, kann das *mobileinit-Event* abgefangen werden. Dieses wird gefeuert, bevor das Framework seine Arbeit beginnt. Um nun Werte zu verändern, wird das Objekt *$.mobile* vom Framework zur Verfügung gestellt.

```
$(document).bind("mobileinit", function(){
  $.mobile.allowCrossDomainPages = true;
  $.mobile.defaultPageTransition = 'slide';
});
```

Da *mobileinit* kurz nach dem Laden der entsprechenden jQuery Mobile-Bibliothek gefeuert wird, muss der eigene Code, der zur Änderung der *Defaults* verwendet wird, vor dieser eingebunden werden:

```
<script src="js/kaffeeklatsch-08-2012.js" />
<script src="js/jquery.mobile-1.1.1.min.js" />
```

JavaScript und UI-Komponenten

Grundsätzlich können auch grafische Komponenten direkt aus JavaScript heraus erzeugt und in der Seite platziert werden. Hier ist allerdings, sofern möglich, dem HTML-basierten Ansatz der Vorzug zu geben, da so am einfachsten dem Konzept des *Progressive Enhancements* gefolgt werden kann. Interessant ist allerdings die Möglichkeit, die Aufbereitung der Seite durch jQuery Mobile programmatisch noch einmal anzustoßen, nachdem die darzustellenden Daten geändert wurden:

```
$('#p1').trigger('create');
```

Themenfrage

Ein letztes zentrales Feature von jQuery Mobile, das im Rahmen dieses Artikels angesprochen werden soll, ist das sogenannte *Theming*, also die umschaltbare grafische Darstellung der einzelnen Komponenten. Standardmäßig werden fünf verschiedene Farb- und Darstellungsschemen unterstützt, die über das Attribut *data-theme* ausgewählt werden können. Die einzelnen Themen werden dabei durch die Wahl eines der Buchstaben von *a* bis *e* gewählt:

```
<div data-role="header" data-theme="b">
  …
</div>
```

Da für die Darstellung CSS-Klassen verwendet werden, kann das gewünschte Schema auch über das Setzen der entsprechenden Style-Klasse gewählt werden:

```
<div data-role="content" class="ui-body-d">
  …
</div>
```

Eigene Farbschemen können durch die Definition eigener Style-Werte erzeugt werden. Dabei können entweder bestehende Themen (*a – e*) geändert oder zusätzliche Themen (*f – z*) eingeführt werden. Durch die Veränderung bestehender Themen kann die Tatsache ausgenutzt werden, dass standardmäßig die Themen *a* und *c* für die Darstellung der Seite und des Inhaltsbereiches verwendet werden, also keine zusätzliche Themenangabe notwendig ist.

Um bei der Themenerstellung nicht ständig zwischen CSS-Definition und Browser wechseln zu müssen, stellt jQuery Mobile das web-basierte Tool *Theme Roller* [6] zur Verfügung. Mit diesem Tool können Themen interaktiv erstellt und verändert werden. Erzeugte Themen können am Ende in Form einer ZIP-Datei, die das *Stylesheet* und die benötigten Bilder enthält, heruntergeladen werden.

Spiel oder Ernst?

Natürlich stellt sich bei jedem Framework die Frage, ob man es nun mit einem netten technischen Spielzeug oder einer ernsthaften Plattform zur Web-Entwicklung zu tun hat. Im Falle von jQuery Mobile lässt sich diese u. a. durch einen Blick auf die Seite *JQM Gallery* [7] beantworten. Hier findet sich eine immer weiter wachsende Reihe von Referenzprojekten, wie z. B. die mobilen Web-Seiten von Ikea [8] oder Disney World [9]. Und sogar im Wahlkampf um das weiße Haus wird inzwischen auf jQuery Mobile gesetzt [10].

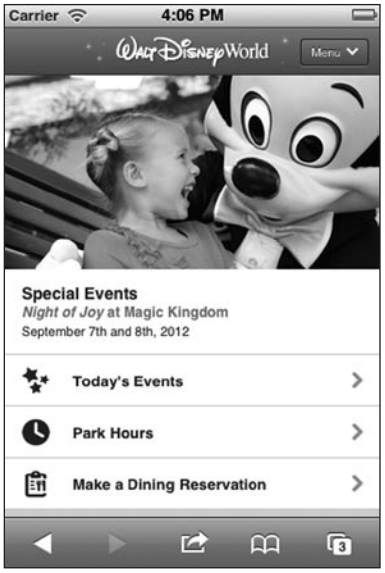

Abbildung 5

Besonders interessant wird der Einsatz von jQuery Mobile im Zusammenspiel mit Applikationsplattformen wie *PhoneGap* [11][12], die es ermöglichen, HTML5-basierte Web-Applikationen offline auf einem Smartphone oder Tablet zu betreiben und diesen Anwendungen Zugriff auf die Geräte-Hardware zu geben. Aber auch wenn es „nur" darum geht die eigene Web-Seite „Smartphone-freundlicher" zu gestalten, ist jQuery Mobile sicherlich einen gute Wahl.

Tools zum Abschluss

Wer seine Web-Seite nun mit jQuery Mobile, aber nicht „von Hand" zusammenbauen möchte, dem sei die auf der Website [1] aufgeführte Liste von Entwicklungs-Tools bzw. Plugins zur genaueren Lektüre empfohlen.

Referenzen

[1] jQuery *Mobile*
http://jquerymobile.com

[2] Wikipedia *Unobtrusive JavaScript*
http://de.wikipedia.org/wiki/Unobtrusive_JavaScript

[3] jQTouch *Zepto/jQuery plugin for mobile web development*
http://www.jqtouch.com

[4] Firtman, Maximiliano *jQuery Mobile: Up and Running Using HTML5 to Design Web Apps for Tablets and Smartphones* , O'Reilly Media, 2012

[5] jQuery *Demos and Documentation*
http://jquerymobile.com/demos/1.1.1

[6] ThemeRoller *Theme Settings*
http://jquerymobile.com/themeroller

[7] jQuery Mobile Gallery
http://www.jqmgallery.com

[8] Ikea Schweden
http://m.ikea.com/se

[9] Disney World Orlando
http://m.disneyworld.disney.go.com

[10] Obama Biden
http://www.barackobama.com/m

[11] PhoneGap Homepage
http://phonegap.com

[12] Eberling, Werner *Mind the gap! – Plattformübergreifende mobile Entwicklung mit PhoneGap*, KaffeeKlatsch 01/2012, Bookware, Erlangen 2012

Kurzbiographie

Werner Eberling ist Principal Consultant und Technical Lead bei der MATHEMA Software GmbH in Erlangen. Seit 1999 beschäftigt er sich mit verteilten Systemen, sein Fokus liegt dabei auf CORBA und der JEE. Neben seiner Projekttätigkeit hält er Technologie-Trainings und ist als Sprecher auf nationalen und internationalen Konferenzen anzutreffen. Sein aktuelles Steckenpferd ist die Anwendungsentwicklung für mobile Endgeräte bzw. deren Anbindung an Enterprise Systeme. Werner Eberling ist u. a. Autor des Buches *Enterprise JavaBeans 3.1 / Das EJB-Praxisbuch für Ein- und Umsteiger*, erschienen im Hanser Verlag.