DEUTSCHLANDS
FÜHRENDE FERNSCHULE

STUDIENHEFT

WEBD07

Webanwendungen für mobile Geräte

Apps mit Standard-Webtechnologien realisieren

Das Studienheft und seine Teile sind urheberrechtlich geschützt. Jede Nutzung in anderen als den gesetzlich zugelassenen Fällen ist nicht erlaubt und bedarf der vorherigen schriftlichen Zustimmung des Rechteinhabers. Dies gilt insbesondere für das öffentliche Zugänglichmachen via Internet, Vervielfältigungen und Weitergabe. Zulässig ist das Speichern (und Ausdrucken) des Studienheftes für persönliche Zwecke.

WEBD07

Webanwendungen für mobile Geräte

Apps mit Standard-Webtechnologien realisieren

Autor: Ralph Steyer
Fachlektor: Dr. Thomas Wecker

Falls wir in unseren Studienheften auf Seiten im Internet verweisen, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf Inhalt und Gestaltung haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Webanwendungen für mobile Geräte
Apps mit Standard-Webtechnologien realisieren

Inhaltsverzeichnis

sgd
DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

© SGD, 13.09.18, Stojanovic, Milan (919400)

Einleitung

Liebe Fernschülerinnen und Fernschüler,

die Verwendung mobiler Endgeräte boomt. Es scheint fast so, als wäre die ganze Welt elektronisch mobil. Sei es mit Handys, Smartphones oder Tablets, über ebook-Reader bis hin zu Netbooks, Ultrabooks und Notebooks. Und die Grenzen der verschiedenen Gerätetypen verschwimmen. Ebenso kommen immer wieder neue Typen an mobilen Geräten hinzu bzw. mobile Geräte werden internetfähig oder programmierbar. Letztendlich kann die Vielfalt egal sein – Mobilität an sich wird wichtiger und „Always-on" ist dabei mehr oder weniger ein Muss. Und dann bedarf es Anwendungen, die den Besonderheiten mobiler Endgeräte Rechnung tragen und entsprechende Mehrwerte bieten, die man mobil gebrauchen kann. Sind Netbooks, Ultrabooks und Notebooks dabei noch „echte" Computer, die man wie Desktop-Rechner mit Tastatur und Maus bedienen kann, müssen ebook-Reader, Smartphones und Tablets meist ohne solche externen Eingabegeräte auskommen. Je kleiner und leichter diese Geräte sind, desto besser sind sie zu transportieren und damit verzichtet man im Hardwaredesign natürlich auf alles, was nicht notwendig erscheint. Die Eingabe von Anwendern wird dabei bekanntlich mehr und mehr auf wenige Hardwaretasten und vor allen Dingen Touchscreens, Sensoren und Spracheingabe übertragen. Und das erzwingt spezifische Designs der Benutzeroberflächen. Auf der anderen Seite werden gerade Smartphones und Tablets oft in der Hand gehalten und nicht wie klassische Computer fest stehend bzw. liegend auf dem Schreibtisch bedient. Ebenso ändert sich der Standort des Geräts während des Betriebs oft. Besondere Hardwaresensoren – etwa zum Erkennen der Orientierung des Bildschirms oder des Standortes eines Benutzers – unterstützten in der Regel diese spezifische Bedienung. Das Halten des Geräts in der einen Hand erzwingt aber naturgemäß, dass die eigentliche Bedienung mit nur einer Hand erfolgen kann. Zudem bieten mobile Geräte dieser Art viele zusätzliche Möglichkeiten, die bei stationären Geräten eher wenig Nutzen bringen. Etwa besagte Lokalisierung per GPS und andere Sensortechniken.

Nun gibt es allerdings gerade bei Handys, Smartphones und Tablets ein extrem heterogenes Umfeld. Sowohl was die Hardware angeht, aber auch die Bedienkonzepte[1] und insbesondere die Betriebssysteme. Zwar dominieren derzeit zwei Systeme klar den Markt (Android und iOS), aber mit Windows Phone/Mobile, den BlackBerry-Betriebssystemen (RIM), Bada, webOS, Maemo oder Symbian gibt es diverse Alternativen.

Native Programmierung von mobilen Applikationen (Apps) bedeutet nun, dass Sie sich auf ein Betriebssystem oder gar nur eine spezielle Version davon konzentrieren und gegebenenfalls für jedes Zielbetriebssystem eine eigene, neue App erstellen, die funktional identisch ist. Das kann einen immensen Aufwand bedeuten. Wenn Sie neutral von den verschiedenen Zielplattformen bleiben wollen, bietet sich alternativ Web-Technologie als Basis an. Insbesondere HTML5 und CSS3 werden in Verbindung mit JavaScript mittlerweile von allen relevanten Herstellern mobiler Endgeräte unterstützt. Damit können Sie eine App einmal entwickeln und (relativ) problemlos auf allen modernen mobilen Systemen einsetzen. Aber Sie entwickeln dann eben Web-Apps. Mit allen Vorteilen, aber auch Nachteilen. Man muss etwa für die Programmierlogik meist mit JavaScript programmieren. Und da JavaScript ursprünglich nur für einfache Programmieraufgaben und auch nicht für die speziellen Anforderungen mobiler Apps konzipiert wurde, sind mit JavaScript viele Schritte relativ aufwendig beziehungsweise umständlich zu programmieren und man stößt zudem schnell an Grenzen. Aber hier helfen Frameworks wie jQuery Mobile, das als spezielle Erweiterung für mobile Belange auf Basis des Web-Frameworks jQuery das Zusammenspiel von HTML5, JavaScript und

1. Gerade bei älteren Geräten.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

CSS3 mit den speziellen Designs mobiler Oberflächen vereinfacht. Und andere Frameworks wie PhoneGab erlauben aus JavaScript heraus etwa den Zugriff auf die spezielle Hardwarekomponenten mobiler Geräte, was mit purem JavaScript nicht funktioniert.

Ich möchte Ihnen in diesem Studienheft nun den Unterschied zwischen nativer Programmierung von Apps und der Entwicklung von Web-Apps auf der Basis von Standard-Webtechnologien zeigen und dabei die Entwicklung webbasierter Apps in den Fokus stellen. Dazu lassen wir uns von geeigneten Frameworks und Tools für die Entwicklung webbasierter Apps unterstützen und lernen typische Anwendungen mobiler Apps kennen. Vom Aufbau einer geeigneten Oberfläche bis zum Zugriff auf typische Elemente eines modernen Smartphones (Geolocation, Orientierung, Kamera, ...). Darauf aufbauend werden Sie nach dem Durcharbeiten dieses Studienhefts mobile Web-Apps in der Funktionsweise verstehen und grundlegende Funktionalitäten eigenständig realisieren können. Ich hoffe jedenfalls, dass Ihnen der Umgang mit dem Studienheft Spaß macht und Sie davon profitieren.

Ralph Steyer

© SGD, 13.09.18, Stojanovic, Milan (919400)

1 Native Apps versus Web-Apps und mobilen Web-Applikationen

Lernziele

In diesem einleitenden Kapitel stellen wir Web-Apps und native Apps gegenüber und beleuchten die Vor- und Nachteile beider Lösungen. Insbesondere werden wir etwas ausführlicher auf native Apps eingehen, was sich in den anderen Kapiteln dann umdreht – dort werden wir fast ausschließlich Web-Apps betrachten. Ebenso sollen Web-Apps und mobile Web-Applikationen verglichen werden und wir besprechen, was Sie zur Entwicklung und für Tests benötigen. Darüber hinaus werden wir die Bedeutung von einigen Begriffen erläutern, die für die relevanten Themen in dem Studienheft von Bedeutung sind und in der Folge vorausgesetzt werden.

1.1 Native Apps

Der Begriff einer **App** ist traditionellerweise mit einer nativen Applikation[2] verbunden, die auf einem speziellen mobilen Gerät ausgeführt wird. Schon Mitte der 90er-Jahre hatte zum Beispiel Sun dafür spezielle Java-APIs zur Verfügung gestellt, über die sogenannte **Midlets** erstellt werden konnten. Das waren Java-Programme, die sich an die speziellen Gegebenheiten der mobilen Geräte anpassen konnten.

Merksatz:

API steht für Application Programming Interface – Schnittstelle zur Anwendungsprogrammierung.

Im Grunde ist eine native App einer nativen Desktop-Applikation ähnlich. Wie sie ist eine native App beim Anwender bereits in plattformspezifische Anweisungen übersetzt und damit für eine bestimmte Plattform optimiert. Ebenso wird die App auf einem Endgerät im klassischen Sinn **installiert**. Mit allen Vorteilen und Nachteilen.

Die **Vorteile** einer nativen App sind eine gute Performance und vor allen Dingen eine optimale Anpassung an eine spezielle Plattform. Eine solche App kann sich harmonisch in übergeordnete Bedienkonzepte einfügen, wie es Apple mit seinem proprietären Habitat oder auch Microsoft mit seinem neuen Metro-Design ab Windows Phone 7 fordern. Und es können viele vorgefertigte Features aus nativen Schnittstellen und APIs benutzt werden.

Die **Nachteile**, die exakt die Vorteile der Web-Apps andeuten, liegen darin, dass ein Programmierer sich mit anspruchsvollen Programmiertechniken auskennen[3] sowie die dafür notwendigen Entwicklungswerkzeuge[4] beherrschen muss. Weiter kommen oft – insbesondere im Apple-Umfeld – zwingend notwendige Hardwarevoraussetzungen zum Entwickeln und Testen der Apps[5] hinzu und es gibt teils rigide Vorgaben, was man in einer App machen darf und was nicht. Und ein Anwender muss eben die App auch installieren.

2. App ist ja auch explizit eine Kurzform von Application.
3. Etwa Java unter Android oder C# und .NET unter Windows Phone.
4. Beispielsweise Eclipse oder das Visual Studio.
5. Sie können etwa eine native iOS-App nicht auf einem Linux-Rechner entwickeln (zumindest nicht ohne diverse Tricks und Hacks).

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Der entscheidende Punkt ist jedoch, dass Sie bei einer App immer für eine spezielle Zielplattform entwickeln. Angenommen Sie wollen eine App für Android, iOS und Windows Phone entwickeln, dann müssen Sie (mindestens) drei verschiedene Projekte mit unterschiedlichen Programmiertechniken aufsetzen. Das Resultat soll identisch aussehen und sich gleich verhalten (soweit das in den verschiedenen Zielplattformen überhaupt möglich ist), aber von der Programmierung gehen Sie dazu sehr wahrscheinlich drei oder mehr unterschiedliche Wege. Ein immenser Aufwand. Wenn Sie dann zusätzlich beachten, dass Sie auch innerhalb einer Betriebssystemwelt möglicherweise noch für verschiedene Betriebssystemversionen unterschiedliche Lösungen entwickeln müssen, haben Sie schnell viele verschiedene Projekte am Laufen, die alle verwaltet und synchronisiert werden müssen.

Hinweis:

Nicht umsonst werden native Apps oft nur gezielt für eine Plattform bereitgestellt.

1.2 Eine native App – eine beispielhafte Vorgehensweise anhand Android

Ich möchte Ihnen in dem Abschnitt den Weg zu einer nativen App anhand von Android beschreiben, ohne in die Details tief einzusteigen. Der Abschnitt wird aber vor allen Dingen das Verständnis fördern, was Sie bei nativen Apps machen müssten, wie diese grundsätzlich arbeiten und wo Ihnen Web-Apps den Weg erleichtern. Ebenso werden Sie erfahren, was Sie als Entwickler zum Schreiben und Testen von Apps benötigen – speziell für native Android-Apps, aber auch schon allgemein für Web-Apps. Denn wenn wir Web-Apps mithilfe von PhoneGap auf einem mobilen Endgerät zum Laufen bringen wollen, brauchen wir diese Grundlagen.

1.2.1 Android und Java

Das Android-Betriebssystem von Google ist ein Linux-System, das aktuell im Smartphone- und Tablet-Markt dominiert. Unter Linux kann man in verschiedenen Sprachen native Applikationen schreiben, aber native Apps für Android sind in der Regel in **Java** geschrieben. Zumal das Google selbst mit diversen Tools und APIs unterstützt.

1.2.2 Die Programmiersprache Java und die JRE

Die Programmiersprache Java, die es seit 1995 gibt, wird von deren Erfinder Sun, der mittlerweile von Oracle übernommen wurde, als eine einfache, objektorientierte, dezentrale, interpretierte, stabil laufende, sichere, architekturneutrale, portierbare und dynamische Sprache, die Hochgeschwindigkeitsanwendungen und Multithreading unterstützt charakterisiert.

Vom Quellcode zur App – Interpretation und Kompilierung

Java gilt nun als **interpretiert** und **kompiliert** zur gleichen Zeit, was im Grunde einen Widerspruch darstellt. Beide Vorgänge (Interpretation und Kompilierung) beschreiben den Vorgang der Übersetzung von Quelltext in lauffähigen Binärcode, der von einem Computer oder auch einem Smartphone oder Tablet ausgeführt werden kann. Dies kann man auf zwei Arten machen:

© SGD, 13.09.18, Stojanovic, Milan (919400)

1. Entweder wird der Quelltext auf einen Schlag mit einem geeigneten Programm übersetzt und dann dieser daraus resultierende Binärcode auf dem Zielgerät zum Laufen gebracht. Das bedeutet dann, dass der Quelltext **kompiliert** wurde.

oder:

2. Man kann aber auch bei einem Anwender den Quelltext laden, Zeile für Zeile lesen und direkt zur Laufzeit des Programms übersetzen lassen. Das ist dann der Vorgang der **Interpretation**.

Java macht beides. Der eigentliche Quellcode wird in einen binären Zwischencode (sogenannten Bytecode) kompiliert, der ein architekturneutrales und noch nicht vollständiges Objekt-Code-Format ist. Er ist noch nicht lauffähig und muss von einer Laufzeitumgebung interpretiert werden. Dies ist die sogenannte JRE (Java Runtime Environment), deren wesentlichen Bestandteil die **JVM** (Java Virtual Machine – ein virtueller Prozessor) ist.

Da jede Java-Laufzeitumgebung plattformspezifisch ist, arbeitet das endgültige Programm (und damit auch eine entsprechende App) auf dieser virtuellen Plattform. Dort werden alle Elemente hinzugebunden, die für eine spezielle physikalische Plattform notwendig sind.

Hinweis:

Unter Android erleichtert dieses Verhalten die Anpassung von nativen Apps an die verschiedenen Android-Versionen und die davon unterstütze Hardware, ohne das Problem grundsätzlich zu beseitigen.[6] Und jedes Android-Endgerät benötigt so eine JRE, was aber Google sicherstellt, weil eine JRE automatisch mit dem Betriebssystem ausgeliefert wird.

1.2.3 Die Software zum Erstellen und Testen von nativen Android-Java-Apps

Android ist wie gesagt ein Linux-System, aber zur Entwicklung können Sie auch mit anderen Plattformen arbeiten. Windows oder – mit Einschränkungen – auch einem Mac. Und die reine Erstellung von Java-Programmen erfordert außer einem Editor keine weitere Software. Aber Sie müssen ja diese Eingaben im Editor übersetzen, damit daraus lauffähige Apps werden. Und dazu sind gewisse Programme notwendig.

Das JDK/SDK und eine passende IDE

Als Erstes sei das **JDK** (Java Development Kit) genannt. Sie finden beispielsweise das JDK auf den Java-Seiten (Java SE Download) von Oracle unter http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Hinweis:

Statt JDK wird auch gelegentlich vom SDK (Software Development Kit) gesprochen. Beide Bezeichner weisen im Wesentlichen auf das Gleiche.

Wenn Sie ein passenden JDK geladen haben, müssen Sie es installieren. Ein einfacher Installationsassistent führt Sie mit wenigen Schritten zum Erfolg.

Im Umfang des JDK ist nun kein Programm zur Erstellung von Quellcode dabei und alle Tools werden rein auf Befehlszeile bedient. Von daher bietet sich für die Entwicklung eine integrierte Entwicklungsumgebung (IDE) an, die Sie noch zusätzlich benötigen.

6. Zumal Java unter iOS und Windows Mobile keine priorisierte Laufzeitumgebung darstellt.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Merksatz:

IDE ist die Abkürzung für **I**ntegrated **D**evelopment **E**nvironment.

Bei der Entwicklung von Android-Apps ist Eclipse (http://www.eclipse.org) sehr populär. Das ist eine IDE unter der OpenSource-Lizenz. Obwohl Eclipse selbst in Java geschrieben und im Schwerpunkt auf die Entwicklung mit Java ausgerichtet ist, können in Eclipse mithilfe von PlugIns nahezu alle Programmiertechniken abgedeckt werden.

Die Installation von Eclipse ist unproblematisch, wenn auf einem Rechner bereits eine passende JDK installiert ist. Eclipse wird als komprimierte Datei ausgeliefert, die Sie einfach extrahieren müssen. Beim ersten Start sucht sich die IDE die passende Java-Umgebung und richtet das System weitgehend automatisch ein.

Hinweis:

Eclipse eignet sich auch hervorragend zur Erstellung von Webseiten und Web-Apps. Insbesondere sei da das PlugIn **Aptana** (http://aptana.com/) empfohlen, das es auch als eigenständiges Eclipse-Derivat gibt.

Eclipse für die Android-Entwicklung erweitern

Um Apps mit Eclipse erstellen zu können, muss man Eclipse entsprechend erweitern und bestimmte Regeln und Vorgehensweisen bei der Erstellung dieses speziellen Typs einer Java-Applikation einhalten.

- Sie benötigen das **Android SDK** (http://developer.android.com/sdk/index.html) und das **ADT-PlugIn** für Eclipse (http://developer.android.com/sdk/eclipse-adt.html) von Google.
- Mit dem sogenannten **AVD Manager** von Google (Teil des SDK) müssen Sie virtuelle Targets (Zielplattformen) festlegen und einrichten, die bestimmte Android-Versionen und verschiedene Hardwarebedingungen simulieren.

Hinweis:

Diese Installationen sind zwar einfach, können aber zeitaufwendig und mit langwierigen Downloads und mehreren Updates verbunden sein (das gilt insbesondere für die vielen möglichen Targets).

© SGD, 13.09.18, Stojanovic, Milan (919400)

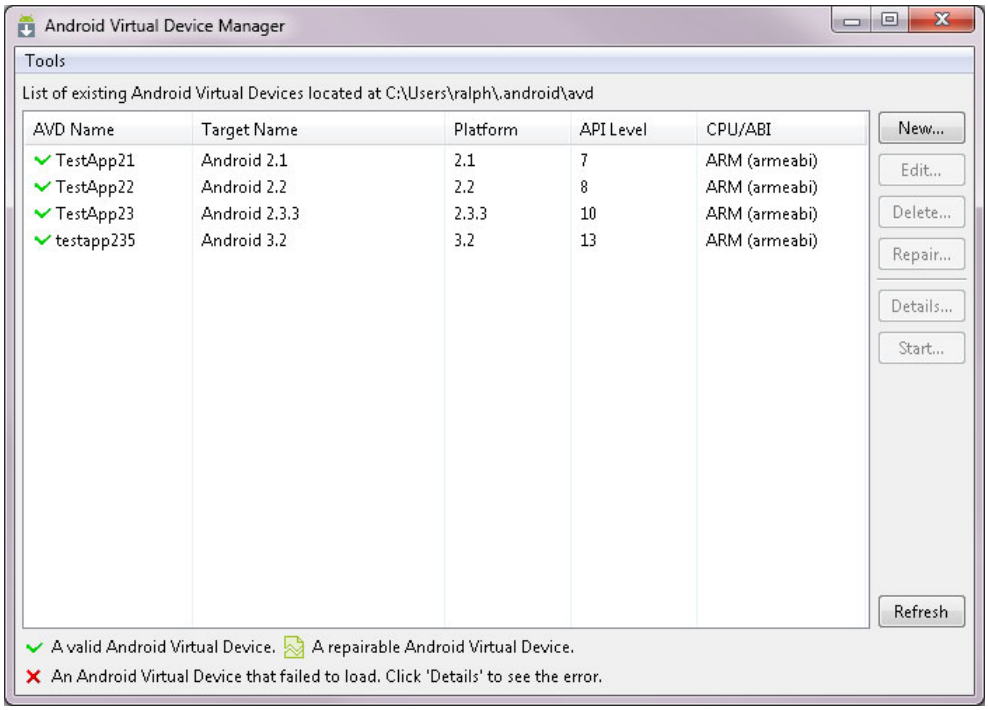

Abb. 1.1: Der AVD Manager im Android SDK

1.2.4 Eine Android-App mit Eclipse und dem ADT-PlugIn erzeugen

Die grundsätzliche Erstellung einer Android-App mit Eclipse und dem ADT-PlugIn läuft in mehreren Schritten ab.

- Zuerst werden Sie in Eclipse ein spezielles ***Android-Projekt*** anlegen.
- Dann schreiben Sie den notwendigen Java-Code samt ergänzender Codestrukturen, die etwa auf XML beruhen. Das setzt natürlich entsprechende Kenntnisse voraus.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

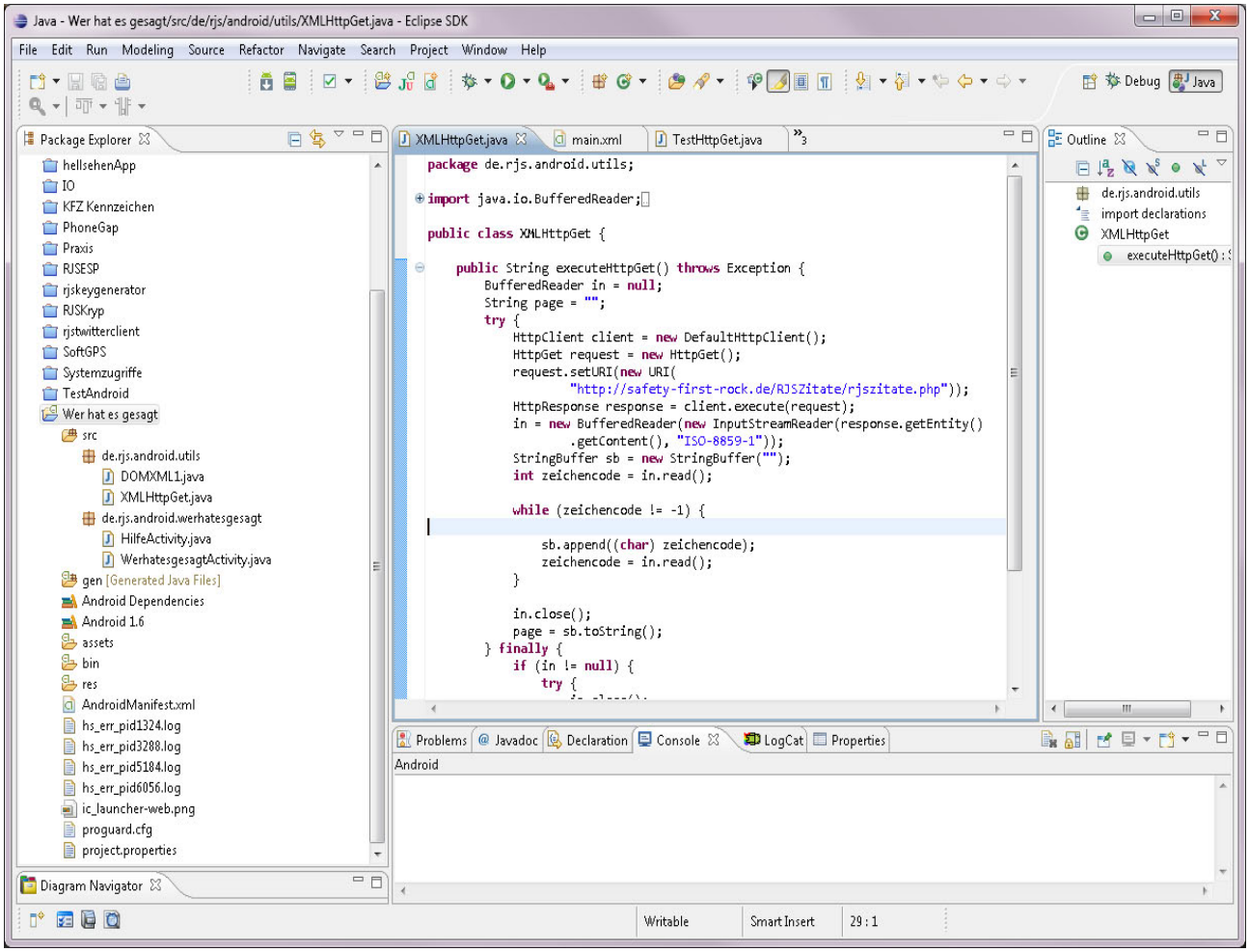

Abb. 1.2: Ein App-Projekt mit dem ADT-PlugIn in Eclipse

- Eine Android-App können Sie aus Eclipse heraus in einem **Emulator** ausführen und testen. Das werden Sie vor einer Fertigstellung immer wieder machen. Dabei ist dieser Emulator allgemeiner zu sehen – Sie können darüber auch hervorragend mobile Web-Apps testen, denn der Emulator enthält – wie jedes reale Android-System – einen Web-Browser, über den Sie Web-Apps bzw. mobile Webseiten laden und testen können.
- Wenn die App fertig ist, werden Sie diese weitergeben wollen. Um eine App weiterzugeben, erzeugen Sie ein spezielles Package. Dieses kann man mit Eclipse und den Android-Erweiterungen von Google erzeugen. Ebenso finden Sie hier Features, um Ihre App im Marktplatz von Google zu veröffentlichen, was aber unseren Rahmen deutlich sprengt.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Übung 1.1:

Eine native Beispiel-App für Android (Verzeichnis auf der beiliegenden CD unter Übungen/Kapitel1/TestAndroid)

Wenn Sie Eclipse samt dem ADT-PlugIn und dem SDK installiert haben und Ihnen passende Zieltargets zur Verfügung stehen, vollziehen Sie die Übung bitte nach. Aber wie schon erwähnt – das Einrichten der passenden Entwicklungsumgebung ist recht aufwendig und Sie wagen sich an ein nicht triviales Thema heran. Wenn Sie da scheitern sollten oder die Einrichtung einer solchen Entwicklungsumgebung nur für eine Übung scheuen, ist das kein Problem. Es ist nur eine Motivation mehr, warum Sie sich auf Web-Apps freuen können. Aber wenn Sie es angehen wollen:

- Legen Sie zuerst ein neues *Android Sample Project* an. Das finden Sie in Eclipse unter dem Menüpunkt **File / New / Other / Android**.

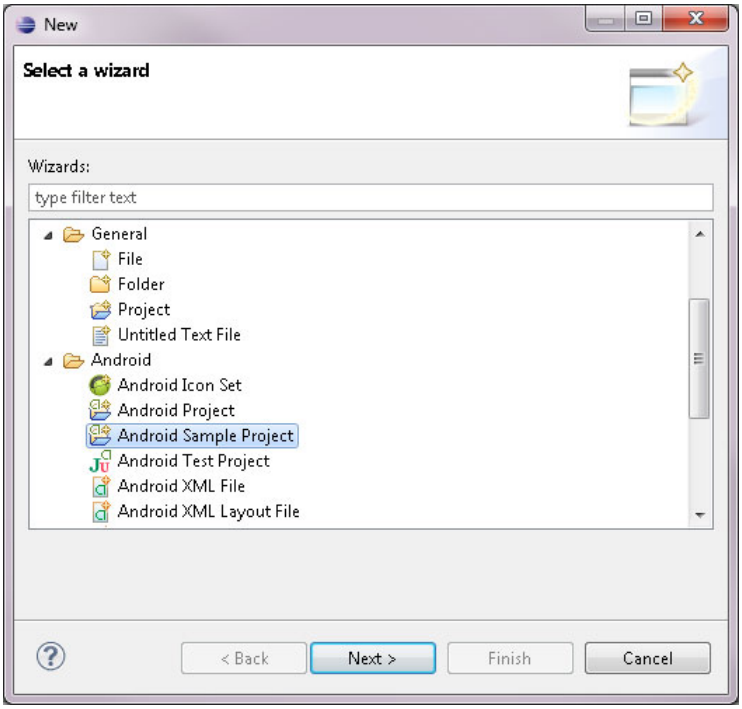

Abb. 1.3: Ein natives Beispielprojekt in Eclipse

- Wählen Sie nun eine Zielplattform aus, auf der die native App laufen soll. Der Wizard sollte Ihnen alle Targets anzeigen, die Sie mit dem AVD-Manager auf Ihrem Rechner eingerichtet haben. Wählen Sie etwa Android 2.1. Dann werden alle Android-Versionen ab 2.1 unterstützt. Gegebenenfalls müssen Sie aber aufpassen, dass eine folgende Beispiel-App eine gewisse Mindestversion voraussetzt.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

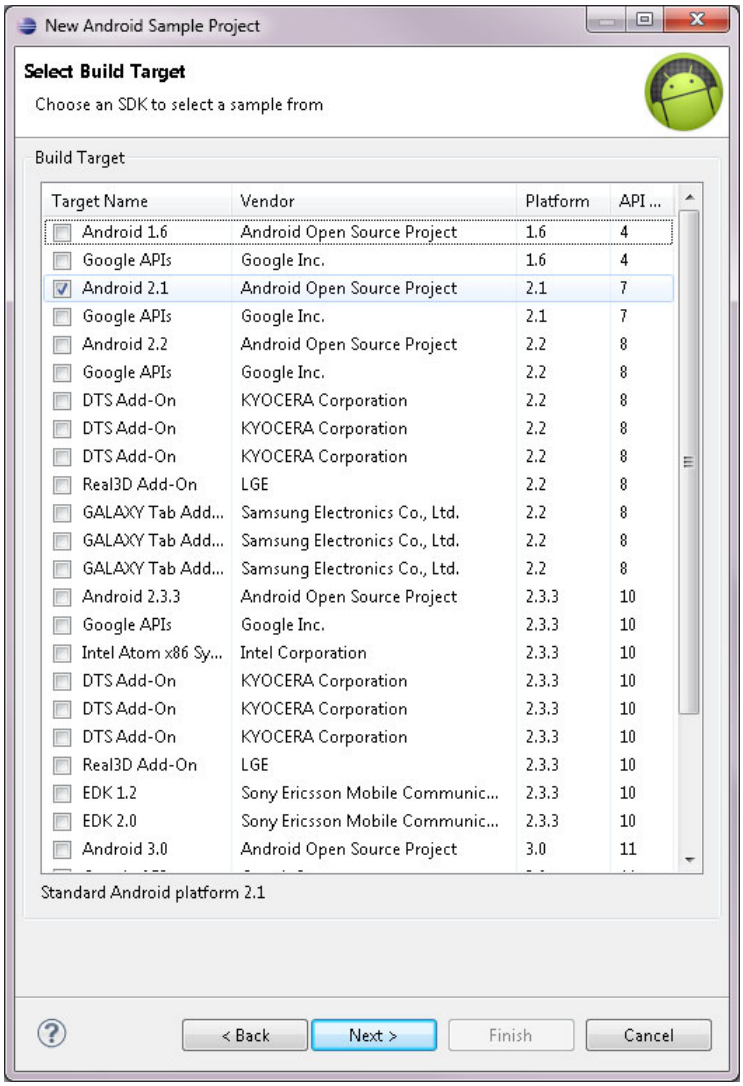

Abb. 1.4: Auswahl der Zielplattform

- Als Projekt können Sie sich aus dem folgenden Dialogfenster einen beliebigen Vorschlag des Wizards auswählen (etwa *Home* oder *ApiDemos*).
- Bestätigen Sie mit *Finish*.
- Nun können Sie die App mit dem Kontextmenü und dort dem Befehl **Run as / Android Application** im Emulator ausführen. Der Start kann eine Weile dauern – also nicht ungeduldig werden.

1.3 Eine native App unter Windows Phone

Ich möchte Ihnen nun noch einen zweiten Weg zu einer nativen App in einer anderen Welt knapp vorstellen. Dieses Mal für Windows Phone. Sie werden gewisse Ähnlichkeiten, aber auch Unterschiede zur Vorgehensweise unter Android erkennen, was das Verständnis von Apps weiter fördern sollte. Und auch hier gilt, dass wir notwendige Grundlagen für Web-Apps mit unter PhoneGap legen.

© SGD, 13.09.18, Stojanovic, Milan (919400)

1.3.1 Windows Phone

Windows Phone bezeichnet den Nachfolger von Windows Mobile, mit dem Microsoft im Zusammenhang mit Windows 8 eine weitere Fusion von mobilen und stationären Anwendungen und Geräten vorantreiben möchte.[7] Insbesondere wird ein vereinheitlichtes Designkonzept namens **Metro** in den Fokus gestellt. Zentraler Aspekt ist der Zwang für Hardwarehersteller, dass ihre Geräte einen Mindeststandard einhalten und eine einheitliche Benutzeroberfläche bereitstellen müssen, damit Windows Phone damit ausgeliefert werden darf. Bei der Benutzerführung geht Microsoft neue Wege und hebt sich damit im Look & Feel und auch der Philosophie deutlich von Android-Geräten ab.

1.3.2 Eine native App für Windows Phone

Um für Windows Phone eine native App zu entwickeln, laden Sie am besten die kostenlosen Windows Phone Developer Tools von Microsoft. Damit können Sie eigene Apps entwickeln und testen – auf dem Windows Phone oder am PC. Die Windows Phone Developer Tools enthalten zur Entwicklung für Windows Phone das Visual Studio Express, Expression Blend for Windows Phone, XNA Game Studio, Silverlight und das .NET-Framework. Insbesondere beinhalten die Tools einen Emulator – den **Windows Phone Emulator**. Diesen können Sie natürlich zum Testen der nativen Windows Phone-Apps, aber auch zum Testen von beliebigen mobilen Webseiten oder Web-Apps unter Realbedingungen verwenden, denn der Emulator bringt den Internet Explorer als integrierten Browser mit. Das Developer Center finden Sie unter http://msdn.microsoft.com/de-de/windowsphone/.

Hinweis:

Auf die Veröffentlichung einer App werden wir auch bei diesem System nicht weiter eingehen. Die wesentlichen Schritte sind aber, dass Sie sich als Entwickler bei Microsoft registrieren müssen und dann Ihre App zur Verifizierung vorlegen können. Wird sie von Microsoft genehmigt, können sie diese dann in einem sogenannten **App Hub** anbieten.

Installation der Windows Phone Developer Tools

Nach dem Download der Windows Phone Developer Tools führen Sie einfach die Installationsdatei aus. Dabei müssen Sie die Lizenzbedingungen bestätigen und etwas warten. Je nach Verbindungsgeschwindigkeit kann die Installation einige Minuten dauern, aber das war alles.

Übung 1.2:

Eine native Beispiel-App für Windows Phone mit Blend (Verzeichnis Übungen/Kapitel1/WindowsPhoneApplication)

Teil der Windows Phone Developer Tools ist das Werkzeug **Blend**, mit dem man sogar ohne große Programmierkenntnisse Apps erzeugen kann. Zumindest solche Apps, die sich auf Standardfunktionalitäten beschränken und weitgehend über optische Effekte bestimmt werden. Aber um schnell einmal eine App zusammenzuklicken, ist das Tool genial.

7. Gerade auch in Verbindung mit Cloud-Diensten.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

- In Ihrem Startmenü sollte ein Menüeintrag **Microsoft Expression / Microsoft Expression Blend** und eine Versionsnummer[8] zu finden sein. Starten Sie darüber das Programm.
- Klicken Sie in Blend nun auf **Datei / Neues Projekt**.

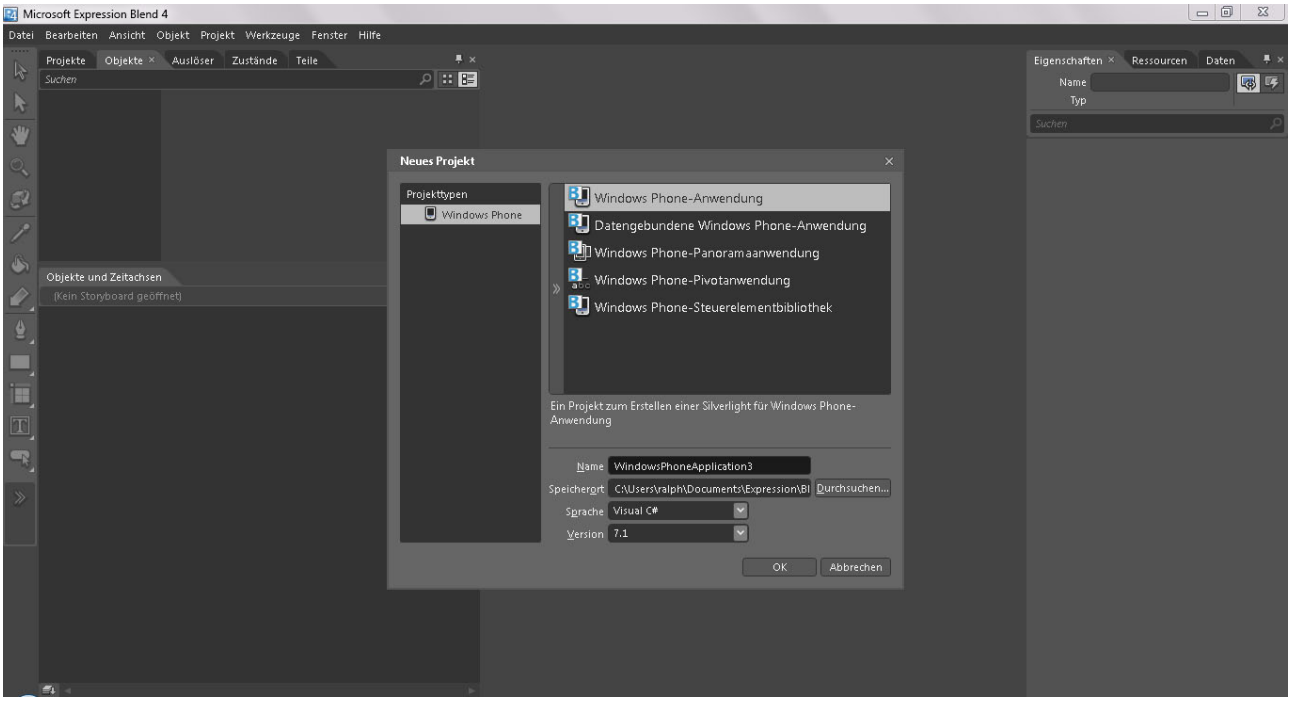

Abb. 1.5: In Blend ein Beispielprojekt anlegen

- Klicken Sie den Beispieltext des Labels doppelt an. Damit wird der Text in dem Label zum Bearbeiten verfügbar. Ersetzen Sie den Text und bestätigen Sie den neuen Text, indem Sie außerhalb des Labels klicken.

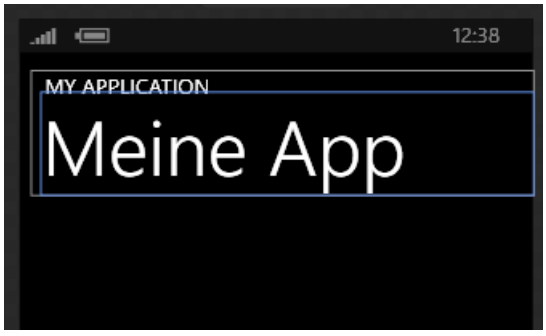

Abb. 1.6: Änderung des Beispieltextes

- In der Standardeinstellung finden Sie in Blend verschiedene Tabs, mit denen Sie die Eigenschaften von selektierten Bestandteilen der Oberfläche ändern sowie Ressourcen und Daten hinzufügen und bearbeiten können. Klicken Sie erneut (aber nur einmal) das Label an. Der Rahmen des Labels sollte selektiert sein. Bei den Eigenschaften finden Sie zahlreiche Möglichkeiten, um das Label zu verändern. Versuchen Sie,

8. Den Unterlagen liegt die Version 4 zugrunde.

© SGD, 13.09.18, Stojanovic, Milan (919400)

zum Beispiel die Textfarbe mit einem bläulichen Farbverlauf zu versehen. Dazu gibt es oberhalb des Farbauswahlfensters verschiedene Schaltflächen und die mittlere ist für Farbverläufe zuständig. Die konkreten Farben wählen Sie natürlich mit dem visuellen und intuitiven Farbeditor aus.

- Auf der linken Seite sollten Sie einen Bereich sehen, in dem Sie Objekte und Zeitachsen auswählen können. Dort sollte der Eintrag *LayoutRoot* zu sehen sein. Selektieren Sie diesen per Klick. Im Eigenschaftenfenster sollte die Hintergrundfarbe zu selektieren sein (*Background*). Wählen Sie auch dafür einen Farbverlauf.
- Suchen Sie nun auf der linken Seite den sogenannten *TextBlock*, selektieren Sie das Symbol und ziehen Sie den Mauszeiger mit gedrückter linker Maustaste über den Anzeigebereich der App. Sie sollten einen transparenten Textblock zeichnen.
- Der Textblock enthält einen Standardtext. Ersetzen Sie diesen durch einen Text Ihrer Wahl.
- Vergrößern Sie die Schrift des neuen Textes.
- Mit F5 können Sie die App nun im Emulator ausführen.

Abb. 1.7: Die App läuft im Emulator der Windows Phone Developer Tools

1.4 Web-Apps

Sie haben gesehen, dass die Erstellung nativer Apps in verschiedener Hinsicht aufwendig und keineswegs trivial ist, denn wir haben mit unseren Übungen ja nur an der Oberfläche gekratzt und für eine voll funktionale native App wäre noch sehr viel zu tun. Und es ist notwendig, dass Sie für jede Zielplattform unterschiedlich vorgehen. Es hat daher eine Menge Charme, klassische RIAs auf mobile Endgeräte zu übertragen bzw. dahingehend anzupassen. Denn die meisten modernen Handys, Tablets und Smartphones sind internetfähig und haben einen Standardbrowser dabei. Und die klassischen Webtechnologien sind in der Regel einfacher zu beherrschen als die mächtigen nativen Techniken wie C# oder Java. Insbesondere ist das Wissen verbreitet, da viele Leute Webseiten erstellen und diese Kenntnisse und Erfahrungen übertragen können.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Nicht zuletzt durch HTML5, aber auch CSS3 werden Webtechnologien auf mobilen Endgeräten konkurrenzfähig. Gerade im mobilen Umfeld wird HTML5 schon sehr weitreichend unterstützt. Und in Verbindung mit JavaScript und einem leistungsfähigen Framework hat man Funktionalitäten, die sich hinter nativen Möglichkeiten oft nicht verstecken brauchen. Man muss vielfach nicht einmal online sein, um solche Web-Apps mobil zu nutzen. Man kann die App einfach im mobilen Endgerät speichern, von dort laden und – wenn sie entsprechend konzipiert sind – offline anwenden.

1.4.1 Entwicklungs- und Testwerkzeuge für Web-Apps

Was Sie für die Programmierung von Web-Apps brauchen werden, ist ein Satz an modernen Webbrowsern, die spezielle Features aus dem mobilen Umfeld unterstützen. Etwa die Widgets und Effekte aus dem Framework jQuery Mobile. Sie wollen ja Ihre Werke zum Testen ansehen. Nur ist das Testen unter realen Bedingungen etwas umständlich.

Verschiedene Wege zum Testen

Sie werden in der Regel Ihre Web-Apps auf einem Desktop-Rechner entwickeln. Allerdings sind die meisten Desktop-Browser nur für einen oberflächlichen Eindruck geeignet, wie eine Web-App in der Realität[9] aussieht und wie sie sich dort verhält. Einen wirklich realistischen Test können Sie nur mit mobilen Geräten erreichen. Das bedeutet, Sie sollten die Web-Apps auch wirklich auf einem Smartphone oder Tablet mit den dort verfügbaren Browsern testen. Die vorher erwähnten **Emulatoren** sind aber eine sinnvolle Alternative, um während der Entwicklung einen realistischen Test auf dem Desktop-Rechner durchzuführen. Unter Umständen ist es dazu aber notwendig, dass Sie die Web-App auf einem Webserver[10] bereitstellen oder erst einmal auf das mobile Gerät kopieren (wenn das möglich ist), damit Sie diese testen können.

Die Erstellung

Zur eigentlichen Entwicklung von Weg-Apps werden Sie als Minimalausstattung bereits mit einem reinen Klartexteditor, wie er bereits bei jedem Betriebssystem mitgeliefert wird, auskommen. Im Gegensatz zur Entwicklung von nativen Apps ist eine IDE und ein passendes SDK nicht unbedingt notwendig. Sie arbeiten ja nur mit HTML, CSS und JavaScript. Einige bessere Editoren bieten einem Programmierer auch bereits einige Unterstützung bei der Codierung von Quellcode an. Notepad++ (http://notepad-plus.sourceforge.net/de/site.htm) ist zum Beispiel ein hervorragender Editor mit so einer Unterstützung. In der Praxis verwenden Sie jedoch meist – ebenso wie bei nativen Apps – mächtigere Programmierwerkzeuge, die Sie bei der Erstellung und Analyse des Quelltextes unterstützen. Ihnen stehen mittlerweile auch in der Webprogrammierung einige richtige IDEs zur Verfügung. Eine kostenlose und dennoch sehr mächtige IDE nennt sich Aptana, die bereits bei nativen Android-Apps im Zusammenhang mit Eclipse erwähnt wurde (http://aptana.com/). Und auch das Visual Studio, das wir im Zusammenhang mit nativen Apps für Windows Phone gesehen haben, beinhaltet mit dem Visual Web Developer ein sehr gutes Tool zum Umgang mit HTML, CSS und JavaScript.

9. Also auf einem mobilen Endgerät wie einem Tablet oder Smartphone.
10. Lokal oder im Internet.

© SGD, 13.09.18, Stojanovic, Milan (919400)

1.5 Mobile Web-Applikationen und mobile Webseiten versus Web-Apps

Nun wollen wir uns noch um die Frage kümmern, was mobile Web-Applikationen bzw. Webseiten von Web-Apps unterscheidet. Es gibt zwar viele Gemeinsamkeiten, aber auch Unterschiede, die sich zum Teil aufgrund der Programmierung, aber insbesondere der Verwendung auf dem mobilen Endgerät resultieren.

Einmal sollte man Webseiten als eher passive Angebote verstehen, die dem Besucher klassische Web-Funktionalitäten bereitstellen. Web-**Applikationen** hingegen sind zuerst einmal aus Sicht des Betrachters viel interaktiver als konventionelle Webseiten. Für unser Umfeld wollen wir diese Details aber nicht weiter unterscheiden. Doch wie passt eine **Web-App** in dieses Umfeld? Ist das eine mobile Webseite, eine mobile Web-Applikation oder was sonst?

Eine Beantwortung der Frage ist schwierig, denn diese Begriffe sind nicht standardisiert. Allgemein versteht man unter einer mobilen Web-Applikation oder einer mobilen Webseite Inhalte, die direkt in einem – bereits auf dem mobilen Endgerät vorhandenen – Browser geladen werden[11] und auch nur dort „leben“. Sie sind also im Grunde gewöhnliche Web-Applikationen oder Webseiten, die nur an den Besonderheiten mobiler Umgebungen orientiert sind. Das sind etwa die spezifischen Eingabemöglichkeiten[12] oder die oft geringen Bildschirmgrößen und -auflösungen. Dadurch, dass mobile Webseiten und Web-Applikationen jedoch immer im Rahmen eines Browsers ausgeführt werden, werden sie auch durch den Browser beschränkt. Das ist ein sinnvolles Sicherheitsfeature, aber eben auch eine Einschränkung gegenüber den Möglichkeiten, die vollständige Programme auf einer Plattform bieten.

Und damit sind die Unterschiede zu einer Web-**App** bereits angedeutet. Diese arbeitet zwar auch mit gewöhnlichen Web-Technologien wie HTML oder JavaScript, wird aber als **eigenständige** App auf dem mobilen Endgerät **installiert** und kann – gegebenenfalls über Schnittstellen wie PhoneGab – Ressourcen des mobilen Endgeräts mehr und besser nutzen als es aus einem bereits vorhandenen Browser heraus möglich ist. Oder anders und ein bisschen leger ausgedrückt – eine Web-App bringt ihren eigenen Browser bereits mit, der auf eine geeignete Weise auf dem mobilen Gerät installiert wird, und der gibt die Web-App als native App aus. Man sollte aber dennoch noch einmal festhalten, dass der Kerncode von mobilen Web-Applikationen/Webseiten und Web-Apps weitgehend identisch ist bzw. sein kann. Wir werden in den Unterlagen überwiegend von Web-Apps oder einfach Webseiten reden und nur dann auf die Unterschiede bzw. Spezifika eingehen, wenn es relevant wird.

11. Meist online über HTTP von einem Webserver.
12. Etwa Bedienung mit dem Finger auf einem Touchscreen statt mit der Maus.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

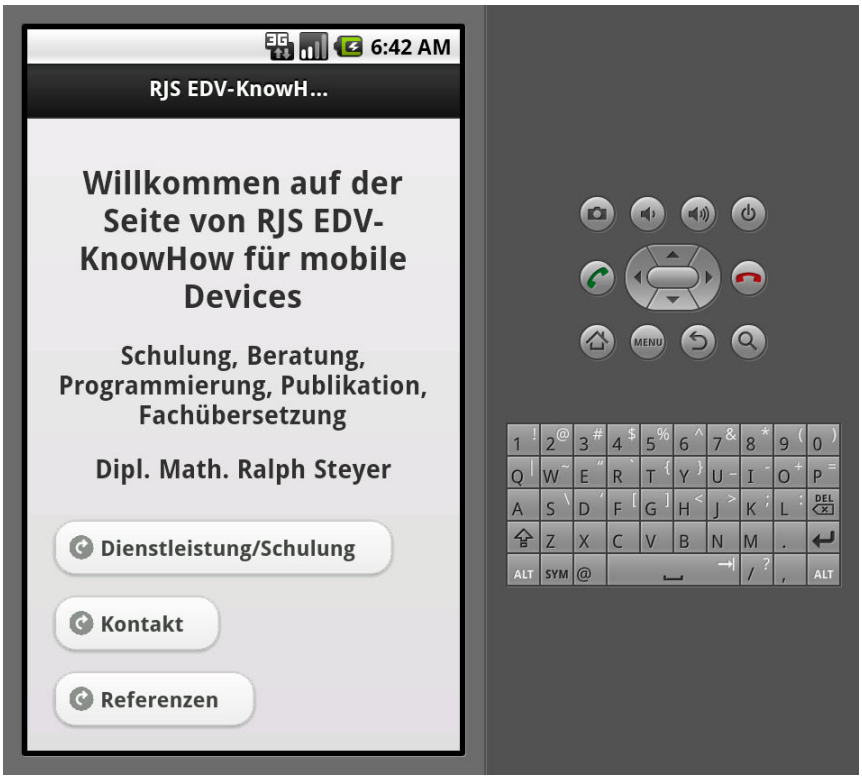

Abb. 1.8: Eine mobile Webseite mit der speziellen Anpassung an kleine Touchscreens – hier im Android-Emulator

Zusammenfassung

Fassen wir am Ende dieses Kapitels zusammen, was Sie an wesentlichen Inhalten gelernt haben. Sie haben einmal in einer Übersicht das weite Gebiet der mobilen Apps kennengelernt. Insbesondere haben wir einen Blick auf native Apps geworfen und wie Sie diese erstellen können. Und Sie wissen, wo die Vor- und Nachteile von nativen Apps und Web-Apps liegen. Ebenso kennen Sie geeignete Wege zum Erstellen und Testen von Web-Apps. Darauf aufbauend werden wir die konkrete Programmierung von Web-Apps behandeln.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Aufgaben zur Selbstüberprüfung

Überprüfen Sie nun bitte Ihre neu erworbenen Kenntnisse. Lösen Sie die nachfolgenden Aufgaben bitte schriftlich und vergleichen Sie Ihre Lösungen mit den Musterlösungen im Anhang. Es ist sehr wichtig und in Ihrem eigenen Interesse die Aufgaben eigenständig zu lösen, ehe Sie im Anhang oder dem Text dieses Kapitels nachschlagen.

1.1 Was ist **kein** wesentliches Kriterium einer Web-App? Kreuzen Sie die richtige Antwort an:

- Sie nutzt als Basis HTML.
- Die Programmierung wird meist mit JavaScript gemacht.
- Web-Apps nutzen oft CSS.
- Web-Apps funktionieren nur auf wenigen mobilen Plattformen.

1.2 Was ist **kein** explizites mobiles Betriebssystem? Kreuzen Sie die richtige Antwort an:

- Android
- iOS
- Windows Mobile
- Bada
- Windows Phone
- BeOS
- webOS

1.3 Welche Bestandteile gehören **nicht** zu den Windows Phone Developer Tools von Microsoft? Kreuzen Sie die richtige Antwort an:

- Blend
- Eclipse
- Visual Studio
- XNA Game Studio
- Windows Phone Emulator
- Silverlight

1.4 Wie kann man mobile Web-Applikationen und Web-Apps differenzieren? Kreuzen Sie die richtige Antwort an:

- Sie verwenden vollkommen verschiedene Techniken und Sprachen für den Quellcode.
- Web-Apps werden auf dem mobilen Endgerät installiert und mobile Web-Applikationen werden dort in einem bereits installierten Standardbrowser ausgeführt.
- Eine mobile Web-Applikation ist onlinefähig und eine Web-App ist nicht onlinefähig.
- Eine mobile Web-Applikation braucht zwingend eine Onlineverbindung und eine Web-App nicht.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

2 jQuery Mobile – die Grundlagen

Lernziele

Die Erstellung von mobilen Webseiten und mobilen Web-Applikationen und Web-Apps erfordert viele Schritte, die Ihnen ein passendes Framework erleichtern kann. Mit jQuery Mobile behandeln wir eines der populärsten und leistungsfähigsten Frameworks für diesen Zweck. jQuery Mobile ist ein berührungsoptimiertes Web Framework auf Basis von grafischen Widgets und HTML5, das eine einheitliche grafische Oberfläche über alle populären Plattformen für mobile Endgeräte bereitstellt.

2.1 Grundlagen

Wie der Name deutlich macht, setzt jQuery Mobile auf dem Basisframework **jQuery** auf. Allerdings ist es darin nicht enthalten, sondern ein eigenständiges Projekt. Die Webseite des mobilen Frameworks finden Sie unter http://jquerymobile.com/. Im Grunde ist die Idee für das mobile Framework aus der jQuery-Welt simpel. Wie bei der Unterstützung von normalen RIAs durch jQuery und jQuery UI wird auf Basis von HTML5, JavaScript und CSS eine einheitliche grafische Oberfläche bzw. Schnittstelle aufgebaut. Und das System soll auf den wichtigsten modernen Tablets, Handys und Smartphones einheitlich funktionieren und sich auch so anfühlen. Dazu muss so ein Framework die große Bandbreite an mobilen Plattformen unter einen Hut bringen, Aussehen und Verhalten vereinheitlichen und für die spezifischen Eingabemöglichkeiten und speziellen mobilen Effekte Unterstützung bieten.

2.1.1 Die Plattformen

Die getestete und garantierte Unterstützung von jQuery Mobile ist weitreichend. Als Plattformen werden im Moment beispielsweise iOS, Android, BlackBerry, Bada, Windows Phone/Mobile, Palm webOS, Symbian, MeeGo und diverse mehr sowie sogar spezielle ebook-Reader-Betriebssysteme wie Kindle unterstützt. Ebenso umfasst die Browserunterstützung alle relevanten Vertreter im mobilen Umfeld.

Hinweis:

Basis der Unterlagen ist die Finalversion 1.1.0 von jQuery Mobile.

A bis C

Bezüglich der Unterstützung verschiedener Plattformen sollte man beachten, dass in der Dokumentation von jQuery Mobile Kategorien angegeben werden, wie weit die Unterstützung einer Plattform und einem Browser von jQuery Mobile garantiert wird. Sogenannte **A-Klasse**-Unterstützung bedeutet eine Garantie für vollen Support. Das bedeutet, dass diese Systeme sich bei den Features des Frameworks genauso verhalten, wie es vorgesehen ist.

Die Einordnung in die B- oder C-Klasse garantiert nicht mehr alle Features.[13] Auf der Seite http://jquerymobile.com/gbs/ finden Sie eine detaillierte Aufstellung, welche Plattformen genau unterstützt werden und in welchem Grad die Features von jQuery Mobile dort bereitstehen. Beachten Sie, dass neue Endgeräte immer besser HTML5 und CSS3 und damit die Widgets des Frameworks unterstützen.

13. B offensichtlich aber noch mehr wie C.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Hinweis:

Sie sollten im Auge behalten, dass viele keineswegs veraltete Handys und Smartphones die geforderten Standards von Frameworks wie jQuery Mobile nicht erfüllen. Und wenn ein mobiles Endgerät die geforderten Standards nicht (vollständig) unterstützt, werden Inhalte nach dem Prinzip der Fehlertoleranz dargestellt. Im „schlimmsten" Fall sehen Anwender aber immer noch eine Webseite ohne Widgets, wie sie eigentlich vom Ersteller vorgesehen sind. In der Regel wird diese Seite dann teilweise funktionieren und wesentliche Inhalte anzeigen, vom Besucher aber viel Scrollen erfordern und nicht gerade gut aussehen.

2.1.2 Die speziellen Features von jQuery Mobile

jQuery Mobile bietet Ihnen nun verschiedene Arten an Unterstützung. An Widgets werden von dem mobilen Framework die bekannten Komponenten aus jQuery UI bereitgestellt, sofern sie auf mobilen Devices sinnvoll sind. Und sie sind auf eine berührungsgesteuerte Benutzerführung über Touchscreens optimiert. Dazu gibt es auch einige neue Events, die nur im mobilen Bereich sinnvoll sind. Ebenso wurde das CSS-Framework von jQuery UI erweitert bzw. an die mobile Zielplattform angepasst. Das Framework arbeitet dazu wie erwähnt intensiv mit CSS3. Und durch neue, spezielle Attribute bei HTML-Tags erfolgt im Hintergrund eine automatische Internationalisierung, was man keinesfalls unterschätzen sollte. Ebenso werden Zugangsfeatures bereitgestellt, um Screenreader zu unterstützen. Auch die Performancebeschränkungen im mobilen Umfeld wurden durch die Bank in der Konzeption des Frameworks berücksichtigt. Was sich nicht zuletzt in der Größe der zu ladenden Dateien niederschlägt. Die geringe Größe des gesamten Frameworks ist beeindruckend.

2.2 Download und Bereitstellung

Das jeweils aktuelle Release von jQuery Mobile erhalten Sie über http://jquerymobile.com/download/. Dort stehen Ihnen verschiedene Dateien zur Verfügung. Da gibt es einmal die JavaScript-Ressourcen (hier sehen Sie die Bezeichner für die Version 1.1.0 von dem Framework):

jquery.mobile-1.1.0.js	Das ist die lesbar aufbereitete Version der JavaScript-Bibliothek, die vor allen Dingen dann von Interesse ist, wenn Sie die JavaScript-Ressourcen des Frameworks selbst analysieren wollen.
jquery.mobile-1.1.0.min.js	Die minimierte Version der JavaScript-Ressource, die für die Auslieferung gedacht ist. Darin wurden überflüssige Zeichen (Kommentare, Leerzeichen, Umbrüche etc.) entfernt. Damit ist der Source nur noch etwa 90 KB groß. Wenn der Webserver und der Browser mit automatischem Zippen und Entzippen umgehen können, kann diese Datei vom Webserver gezipped ausgeliefert und dazu auf etwa 24 KB eingedampft werden.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Zum Framework zählen aber auch CSS-Ressoucen, die es auch in verschiedenen Versionen gibt:

Uncompressed with Default theme: *jquery.mobile-1.1.0.css*	Das ist wieder die lesbar aufbereitete Version der Ressource, die vor allen Dingen dann von Interesse ist, wenn Sie die CSS-Ressourcen des Frameworks selbst analysieren wollen. Diese Version beinhaltet ein sogenanntes Theme oder Thema, was eine bestimmte CSS-Vorgabevorlage für die Standardelemente im Framework darstellt.
Minified and Gzipped with Default theme: *jquery.mobile-1.1.0.min.css*	Die minimierte Version der CSS-Ressource samt CSS-Vorgabethema. Diese Variante ist wieder für die Auslieferung gedacht ist. Der Source ist etwa 60 KB groß. Wenn der Webserver und der Browser mit automatischem Zippen und Entzippen umgehen können, kann auch diese Datei vom Webserver gezipped und auf etwa 7 KB eingedampft werden.
Uncompressed structure without a theme: *jquery.mobile-1.1.0.css*	Die lesbare Version ohne ein Vorgabethema.
Minified and Gzipped structure without a theme: *jquery.mobile-1.1.0.min.css*	Die minimierte Version ohne ein Vorgabethema.

2.2.1 Die Bereitstellung über ein CDN

Sie brauchen nun diese Dateien von jQuery Mobile nicht direkt auf Ihren Rechner zu laden und dann über Ihren eigenen Server selbst zur Verfügung stellen. Sie werden auch über ein **CDN** des Projekts zur Verfügung gestellt. Diese Möglichkeiten können Sie nutzen.

Merksatz:

CDN steht für **C**ontent **D**istribution **N**etwork oder auch Content Delivery Network. Es bezeichnet ein Netz lokal verteilter und über das Internet verbundener Server, mit dem Inhalte möglichst effektiv und für den Anwender intransparent ausgeliefert werden sollen. Dazu werden die Daten im Netz gecached und Anfragen nach Last verteilt.

Über das jQuery-CDN werden sowohl die Ressourcen von jQuery selbst als auch jQuery UI und ebenso jQuery Mobile bereitgestellt. Mit entsprechenden URLs können Sie diese Ressourcen in Ihre Webseiten einbinden. Damit entlasten Sie Ihren Webserver, verlassen sich aber auch auf den Provider des CDN, dass die Ressourcen zuverlässig bereitgestellt werden. Ebenso greifen Sie bei jedem Aufruf Ihrer Seiten/Apps auch auf das CDN zu, was dort protokolliert werden kann. Und Ihre Webseite, Anwendung oder App muss Zugriff auf das Internet haben, auch wenn man es aufgrund der eigentlichen Funktionalität vielleicht gar nicht bräuchte. Es spricht also einiges für die Verwendung von einem CDN, aber auch einiges dagegen. Das Für und Wieder sollten Sie abwägen.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Hinweis:

Da jQuery Mobile explizit auf dem Kernframework jQuery aufsetzt, müssen Sie neben jQuery Mobile auch die JavaScript-Datei von jQuery einbinden. Dabei sollten Sie unbedingt eine passende Version von jQuery verwenden. „Passend" bedeutet, dass sie nicht zu niedrig sein darf. Für jQuery Mobile 1.1.0 sollten Sie etwa mindestens die Version 1.6.4 von jQuery selbst verwenden. Allerdings kann auch eine zu neue Version von jQuery gefährlich werden, denn wenn dort inkompatible Umstrukturierungen vorgenommen wurden, muss man diese möglicherweise auch in jQuery Mobile berücksichtigen. Von daher sollte man immer in der Dokumentation nachsehen, welche Versionen von jQuery als Basis empfohlen werden.

Übung 2.1:

Eine Basisschablone mit allen Referenzen für jQuery Mobile über ein CDN (Übungen/Kapitel2/basis1.html, Übungen/Kapitel2/lib/css/jqm1.css und Übungen/Kapitel2/lib/js/jqm_ready1.js)

Erstellen Sie als Übung auf Basis einer validen HTML5-Syntax eine Basisschablone mit allen notwendigen Referenzen, um mit jQuery Mobile über Ressourcen auf dem Standard-CDN arbeiten zu können. Zudem sollen bereits die notwendigen Ressourcen verknüpft werden, um gegebenenfalls eigene Styles Sheets und Skripte zusätzlich verwenden zu können. Geben Sie zunächst den HTML-Quellcode ein (*basis1.html*):

```
<!DOCTYPE html>
<html lang="de">
      <head>
            <meta charset="utf-8" />
            <title>Eine Basisschablone in HTML5</title>
            <meta name="viewport" content="width=device-width,
            initial-scale=1">
            <link rel="stylesheet" type="text/css"
                  href="http://code.jquery.com/mobile/1.1.0/
                  jquery.mobile-1.1.0.min.css" />
            <link rel="stylesheet" href="lib/css/jqm1.css"
            type="text/css"/>
            <script src="http://code.jquery.com/jquery-
            1.6.4.min.js" type="text/javascript"></script>
            <script src="http://code.jquery.com/mobile/1.1.0/
            jquery.mobile-1.1.0.min.js"
                  type="text/javascript"></script>
            <script type="text/javascript" src= "lib/js/
            jqm_ready1.js"></script>
      </head>
      <body>
      </body>
</html>
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Sie sehen in dem Quellcode zuerst eine Metaangabe, die das Framework zur Skalierung benötigt. Dem folgt eine Referenz auf die CSS-Ressourcen von jQuery Mobile auf dem Standard-CDN über einen absoluten URL[14] und dann einen relativen URL auf unsere eigenen CSS-Ressourcen. Diese eigene CSS-Datei verwalten wir im Unterverzeichnis *lib/css*, was der derzeit allgemein gängigen Projektstruktur entspricht. Danach folgen die Referenzen auf die JavaScript-Dateien von jQuery und jQuery Mobile über absolute URL auf das Standard-CDN und einer relativen URL auf die eigene JavaScript-Datei, die sich im Unterverzeichnis *lib/js* befindet.

Hinweis:

Wenn Sie externe Style Sheets in Ihrem HTML-Code referenzieren, sollten diese **vor** den Skripten eingebunden werden. Das sehen Sie auch in unserer Basisschablone. Andernfalls sind Probleme im Umgang mit jQuery bekannt. Und binden Sie zudem zuerst die CSS-Datei(en) von jQuery und dann Ihre eigenen CSS-Dateien ein. Der Grund ist, dass die später eingebundenen Regeln bei gleicher Priorität vorher definierte Regeln überschreiben. So können Sie sicher sein, dass Ihre Regeln in so einem Fall Vorrang haben. Auch die Reihenfolge der JavaScript-Referenzen ist nicht zufällig. Alle JavaScript-Dateien arbeiten im gleichen Namensraum. Das bedeutet, dass Sie Dinge, die Sie in einer JavaScript-Datei definiert haben, in der anderen unbeschränkt verwenden können. Allerdings nur dann, wenn Sie bereits geladen wurden. Und da jQuery Mobile mit Deklarationen aus jQuery arbeitet und die zu dem Zeitpunkt da sein müssen, werden Sie zuerst auf jQuery referenzieren. Und Sie arbeiten im eigenen JavaScript-Code ja gegebenenfalls mit Elementen aus jQuery und jQuery Mobile und diese müssen zu dem Zeitpunkt definiert sein. Deshalb müssen Sie auch alle Ressourcen von jQuery vor Ihren eigenen Skripten einbinden.

Obwohl wir im Moment nicht viel mit eigenen JavaScripts und CSS machen, erstellen Sie bitte die JavaScript-Datei *jqm_ready1.js*. Sie soll erst einmal nur so aussehen:

```
$(function() { })
```

Erstellen Sie ebenso die CSS-Datei *jqm1.css*. Diese kann im Moment noch leer bleiben.

Laden Sie nun die HTML-Datei in einen Browser[15]. Es sieht noch nicht danach aus, dass viel passiert ist. Aber analysieren Sie die Seite mit dem Firefox-AddOn Firebug oder den Entwicklertools eines anderen Browsers. Sie werden bei der Analyse dem tatsächlich vom Browser verwendeten DOM Ihrer HTML-Struktur sehen. In Firebug können Sie das `head`-Element mit einem Klick auf das Pluszeichen expandieren und dort alle `script`- und `css`-Elemente ebenso expandieren. Sie werden sehen, dass die referenzierten externen Ressourcen in die Webseite eingebunden sind und Ihnen nun mit allen Möglichkeiten zur Verfügung stehen, die Ihnen von dem Framework bereitgestellt werden.

14. Beachten Sie die konkrete Version 1.1.0 – diese werden Sie entsprechend anpassen, wenn neuere Versionen erschienen sind.
15. Das kann ein normaler Desktop-Browser sein.

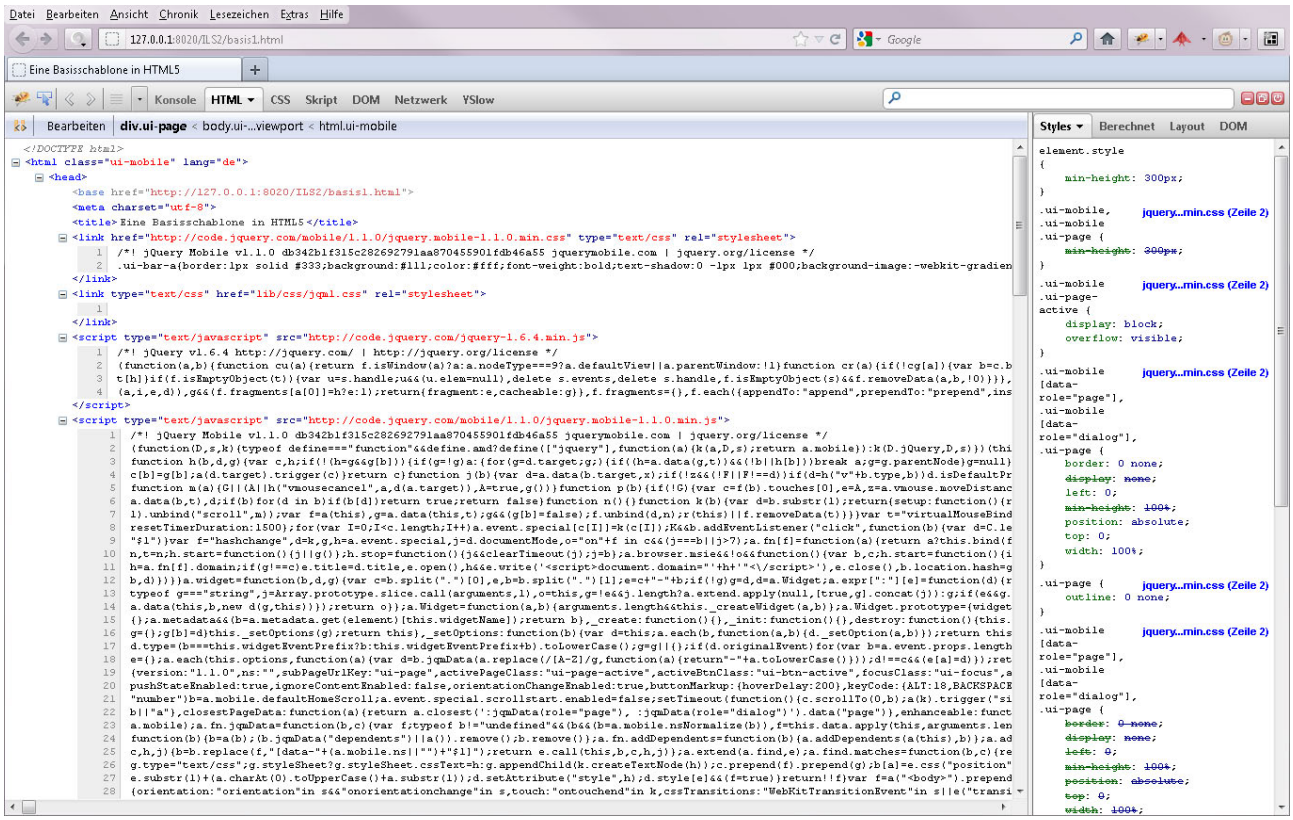

Abb. 2.1: Alle Ressourcen wurden korrekt eingebunden

Hinweis:

Sollten Referenzen nicht die gewünschten Ressourcen anzeigen, müssen Sie die Fehler korrigieren.

Übung 2.2:

Die Basisschablone auf Validität überprüfen

Verwenden Sie den Validierungsservice des W3C unter http://validator.w3.org/ zu einem Test, ob unsere Basisdatei valide nach dem HTML5-Standard ist. Sie können bei diesem Validierungsservice eine lokale Datei hochladen oder in einem Eingabefeld den Code per Zwischenablage zur Überprüfung bereitstellen.

2.2.2 Die Bereitstellung über eigene Ressourcen

Und wenn Sie die notwendigen Dateien für den Umgang mit jQuery Mobile vollständig selbst hosten bzw. bereitstellen wollen, können Sie von der Webseite eine entsprechende ZIP-Datei laden. Diese enthält alle notwendigen Ressourcen sowie diverse Beispiele. Sie müssen dann für eine mobile Anwendung bzw. Web-App die CSS-Datei samt einiger Bilder und die JavaScript-Datei auf dem Webserver oder in einem lokalen Zugriffspfad der Web-App verfügbar machen und entsprechend der üblichen Regeln einbinden.

Übung 2.3:

Eine Basisschablone mit allen Referenzen für jQuery Mobile ohne ein CDN (Übungen/Kapitel2/basis2.html)

Wir passen nun unsere Basisschablone so an, damit Sie ohne ein CDN arbeiten können. Das bedeutet, dass auch alle jQuery-Ressourcen auf Ihrem System bereitgestellt werden müssen und wir die Pfade anzupassen haben. Dazu verwenden wir als Version von jQuery selbst die Version 1.7.2, die auch explizit zu jQuery Mobile 1.1.0 passt. Geben Sie den angepassten HTML-Quellcode ein (*basis2.html*):

```
<!DOCTYPE html>
<html lang="de">
      <head>
            <meta charset="utf-8" />
            <title>Eine Basisschablone in HTML5</title>
            <meta name="viewport" content="width=device-width,
            initial-scale=1">
            <link rel="stylesheet" type="text/css" href="lib/css/
            jquery.mobile-1.1.0.min.css" />
            <link rel="stylesheet" href="lib/css/jqm1.css"
            type="text/css"/>
            <script type="text/javascript" src="lib/js/jquery-
            1.7.2.min.js"></script>
            <script src="lib/js/jquery.mobile-1.1.0.min.js"
            type="text/javascript"></script>
            <script type="text/javascript" src="lib/js/
            jqm_ready1.js"></script>
      </head>
      <body>
      </body>
</html>
```

Die referenzierten lokalen Dateien müssen Sie natürlich auch dort in Ihrem Projektverzeichnis bereitstellen. Ebenso gehört in das Verzeichnis der CSS-Ressourcen ein Unterverzeichnis *images*, das Sie mit der ZIP-Datei geladen haben.

Laden Sie auch diese HTML-Datei in Firefox und analysieren Sie die Seite mit dem Firefox-AddOn Firebug. Kontrollieren Sie, ob alle `script`- und `css`-Ressourcen korrekt eingebunden werden und korrigieren Sie gegebenenfalls die Fehler.

© SGD, 13.09.18, Stojanovic, Milan (919400)

2.3 Das Rollensystem und data-role

Kommen wir zu einem, wenn nicht dem(!), zentralen Aspekt des jQuery Mobile-Frameworks. In modernen Webseiten – insbesondere mit HTML5 – versucht man, sich bezüglich der verwendeten HTML-Tags auf die Struktur einer Seite zu konzentrieren, und überlässt das Design Style Sheets. Man beschränkt sich bei modernen Seiten also möglichst auf wenige Tags, die nur die Struktur beschreiben. Von entscheidender Bedeutung ist neben diversen neuen Elementen in HTML5 zur Abbildung einer Semantik das DIV-Element. Und dieses Gedankengut wird im Framework konsequent umgesetzt.

Das mobile jQuery Framework arbeitet mit einem **Rollensystem** für Elemente einer Seite, um Widgets auszuzeichnen. Über ein Attribut data-role wird jedes Element, das eine bestimmte Aufgabe erfüllen soll, über standardisierte Werte ausgezeichnet. Als Anwender müssen Sie beispielsweise bei der Auszeichnung eines Widgets meist gar nicht mehr direkt programmieren, wie es etwa in jQuery UI der Fall ist. Sie können zum Beispiel ein DIV-Element alleine durch die Auszeichnung mit dem Attribut und einem spezifischen Wert zu einem Button machen. Deshalb brauchen Sie auch hierfür keinen expliziten Skriptbereich und eine entsprechende Anweisung, wie Sie es in jQuery UI machen. Alle Widgets, die Sie im Framework verwenden wollen, können so erzeugt werden. Aber auch Seitenübergänge, Verknüpfungen und Seiten selbst können durch diesen Ansatz gänzlich ohne manuelle Programmierung erstellt werden.

2.4 Der grundsätzliche Aufbau einer mobilen Seite

Der grundsätzliche Aufbau einer Webseite, die vom jQuery Mobile profitieren soll, ist immer ähnlich. Man nutzt wie gesagt über das gesamte Framework insbesondere die `data-role`-Attribute und dokumentiert auch ausdrücklich in einer `DOCTYPE`-Anweisung, dass man HTML als Basis verwendet. Genau genommen sollte man laut offiziellen Empfehlungen eine valide HTML5-Basisstruktur verwenden (wie in unserer Basisseite), aber das Framework funktioniert auch mit älteren HTML-Versionen und oft auch nicht vollständig validen Grundstrukturen. Nur gibt es keinen vernünftigen Grund, warum man auf eine valide HTML5-Basis verzichten sollte.

2.4.1 Eine Seite ist nicht unbedingt eine Seite

Nun kann eine Webseite im Sprachgebrauch des Frameworks zwar eine Seite sein, aber ***Seite*** versteht man in jQuery Mobile eigentlich anders. Es ist ein **abgeschlossenes** Segment **innerhalb** der Webseite. Die typische Seitenstruktur wird entweder eine einzelne Seite (bei wenigen Inhalten) oder intern lokal verlinkte Seiten darstellen.

Im Körper einer einzelnen HTML-Datei wird jede Ansicht oder jedes Element, das als Seite in Hinsicht auf das mobile Display dienen soll (normalerweise ein DIV-Element) mit dem Attribut `data-role="page"` versehen. Innerhalb solch eines Containers können Sie alle gültigen HTML-Anweisungen verwenden (außer den Anweisungen des Grundgerüsts und der spzeillen jQuery-Seite selbst). Typischerweise strukturiert man aber ein solches Seitensegment in einen eigenen Kopf-, Inhalts- und Fußbereich. Diese werden wieder mit spezifischen Werten von data-role gekennzeichnet. Etwa so:

```
<div data-role="page">
      <div data-role="header">...</div>
      <div data-role="content">...</div>
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

```
    <div data-role="footer">...</div>
</div>
```

Sie sollten bei den Werten von `data-role` Ähnlichkeiten zu den neuen semantischen HTML5-Tags erkennen. Was kein Zufall ist!

Übung 2.4:

Eine erste mobile Seite (Übungen/Kapitel2/jqm1.html)

Erstellen Sie nun bitte eine vollständige Webseite, die den Anforderungen von jQuery Mobile genügt, nur eine Seite im Sinn des Frameworks enthält und in unsere übliche Projektstruktur eingebunden ist (*jqm1.html*). Dabei entspricht der Head der Webseite mit den Referenzen der Basisstruktur und wird hier nicht mehr abgedruckt:

```
    <body>
        <div data-role="page">
            <div data-role="header"><h1>Der Titel</h1></
            div>
            <div data-role="content"><p>Der beliebige
            normale Inhalt, der mit
            <span style="background:red">HTML-Tags</span>
            und <i>CSS</i> durchsetzt sein kann.</p>
            </div>
            <div data-role="footer"><h4>Fußzeile</h4></div>
        </div>
    </body>
</html>
```

Durch die Auszeichnungen der DIV-Elemente wird das Framework auf einer passenden Plattform dafür sorgen, dass die Bereiche entsprechend formatiert werden. Zudem wird der page-Block wie erwähnt als eine Seite gesehen. Die enthaltenen DIV-Blöcke füllen die vorgesehenen Rollen als Kopfbereich, Inhaltsbereich und Fußbereich aus.

Wenn Sie die Seite fertig haben, öffnen Sie diese in einem Browser, der das Framework ausreichend unterstützt. Verkleinern Sie ggf. bei einem Test in einem Dekstop-Browser das Browserfenster auf Maße, die Sie üblicherweise im mobilen Umfeld vorfinden. Dann sollten Sie die Seite so sehen, wie sie auf einem mobilen Endgerät auch aussehen wird. Noch besser ist es, wenn Sie die Seite auf Ihr mobiles Endgerät kopieren[16] und dann in einen Browser laden. Alternativ können Sie die Seite auch auf einen Webserver laden und von dort über Ihr mobiles Endgerät anfordern. Und selbstverständlich können Sie einen der vorher besprochenen Emulatoren starten und dann in dem dort verfügbaren Browser die Seite laden (wenn Sie auf einem erreichbaren Webserver veröffentlicht ist).

16. Was leider im Umfeld von Apple und Windows Phone ziemlich umständlich ist. Aber unter Android geht das problemlos.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Abb. 2.2: Die Seite wurde vom Framework formatiert – die Anzeige erfolgt hier im Emulator für Windows Phone

2.5 Verknüpfen von Seiten

Grundsätzlich werden im mobilen Framework alle normalen Hyperlinks aus HTML unterstützt. Beim Verknüpfen von Seiten muss man im Framework jedoch zwischen **externen** und **internen** Links unterscheiden, was sich aus dem abweichenden Verständnis einer Seite ergibt.

2.5.1 Externe Links mit Hijax

jQuery Mobile arbeitet in der Grundeinstellung bei Links auf externe Seiten mit einem AJAX-Request (**Hijax**), sofern der Verweis auf die gleiche Domain geht. Im Unterschied zu einer Desktop-Applikation wird bei mobilen Apps dennoch der Cursor so verändert, dass das Nachladen von Daten angezeigt wird (die drehende Eieruhr – Spinner). Wenn die Anfrage erfolgreich war, werden die neuen Daten in den DOM eingebaut und die mobilen Widgets neu initialisiert. Danach wird die neue Seite animiert angezeigt.

Wenn der Link auf eine andere Domain verweist oder mit `rel="external"` bzw. `data-ajax="false"` ausgezeichnet oder einem `target`-Attribut versehen ist, wird aber stattdessen konventionell eine neue Seite geladen.

Hinweis:

Das Framework unterstützt zudem neben HTTP Protokolle für weitere Linkarten, die sich zum Teil aus dem mobilen Umfeld ergeben. So werden beispielsweise für E-Mail mailto: und Telefon tel:, wtai: sowie dc: unterstützt.

2.5.2 Interne Links und das spezielle Verständnis einer Seite

Unter jQuery Mobile kann ein einzelnes HTML-Dokument – wie angedeutet – mehrere *Seiten* im Sinn des Frameworks enthalten. Dieser Ansatz ist naheliegend, denn klassische Webseiten sind zu groß für kleine Bildschirme auf Handys und Smartphones. Im Fall einer normalen Webseite muss der Besucher mit einem mobilen Endgerät also fast immer scrollen, um die Inhalte der gesamten Webseite zu sehen. Eine Aufteilung der Inhalte einer Webseite durch den Ersteller der Webseite in festgelegte Segmente mit jeweils im logischen Verbund sinnvollen Informationen ist also vorteilhaft. Und das gemeinsame Laden vorab ist im mobilen Bereich nicht zuletzt durch die immer noch meist geringen Bandbreiten naheliegend. Diese bereits geladenen, aber noch nicht angezeigten Segmente, kann man dann per spezieller Links miteinander verbinden und bei einem Wechsel der Seiten fix anzeigen, da die verschiedenen Inhalte ja schon vorhanden sind.

Jeder page-Block benötigt nun für eine sinnvolle interne Verlinkung eine eindeutige ID. Diese wird für die interne Verlinkung über eine klassische Ankernotation verwendet (`href="#meineId"`). Beim Anklicken des Links wird der entsprechend gekennzeichnete Bereich auf dem Bildschirm verschoben.

Hinweis:

Wenn Sie auf eine Seite verlinken, die mit AJAX geladen wurde und mehrere `page`-Segmente beinhaltet, müssen Sie die Attribute `rel="external"` oder `data-ajax="false"` dem Link hinzufügen. Beispiel:

```
<a href="multipage.html" rel="external">Link</a>
```

2.5.3 Zurück in der Historie

Sie kennen sicher aus JavaScript die Funktionalität `history.back()`. Damit gehen Sie in der Historie eine Seite zurück. Mit dem Attribut `data-rel="back"` können Sie genau diese Funktionalität ausnutzen. Alternativ können Sie `data-direction="reverse"` verwenden, wenn Sie die Historie nicht direkt verwenden wollen.

Übung 2.5:

Verknüpfungen von mehreren Seiten (Übungen/Kapitel2/jqm2.html)

Erstellen Sie nun eine Webseite, die sowohl interne als auch externe Verlinkungen enthält und zudem in der Historie zurückgeht (*jqm2.html*). Der Head der gesamten Webseite ist wieder unverändert und er wird hier nicht abgedruckt:

```
<body>
      <div data-role="page" id="p1"><div data-
      role="header"><h1>Seite 1</h1></div>
            <div data-role="content"><h2>Willkommen</h2>
                  <a href="#p2">Zur Seite 2</a><br /><a
                  href="http://jquerymobile.com">jQuery
                  Mobile</a>
            </div>
            <div data-role="footer"><h4>Verlinkung</h4></
            div>
```

© SGD, 13.09.18, Stojanovic, Milan (919400)


```
            </div>
            <div data-role="page" id="p2"><div data-
            role="header"><h1>Seite 2</h1></div>
                  <div data-role="content"><p><a data-rel="back"
                  href="#">Zurück</a></p></div>
                  <div data-role="footer"><h4>Verlinkung</h4></
                  div>
            </div>
      </body>
</html>
```

Sie sehen hier ein Dokument, das zwei Seiten im Sinn von jQuery Mobile enthält. Wenn Sie die Datei laden, wird zuerst nur der erste `page`-Bereich angezeigt, sofern der Browser das Framework korrekt unterstützt.

Der erste Hyperlink im ersten `page`-Segment verweist auf eine ID `p2 (<a href="#p2">)` und führt zum zweiten `page`-Segment, das mit dieser ID ausgezeichnet ist.

Auf der zweiten Seite finden Sie den Link zurück, der die Historie des Browsers verwendet. Beachten Sie, dass der Wert von `href` als # gesetzt wurde. Der Link zurück wird über `data-rel` ausgelöst, aber ein # löst ebenso einen Sprung auf das erste `page`-Segment aus. Das ist sozusagen ein doppelt abgesicherter Rücksprung zur Seite, von der der Besucher kam.

Des Weiteren sehen Sie im ersten `page`-Segment noch einen zweiten Link, der aber auf eine externe Ressource verweist.

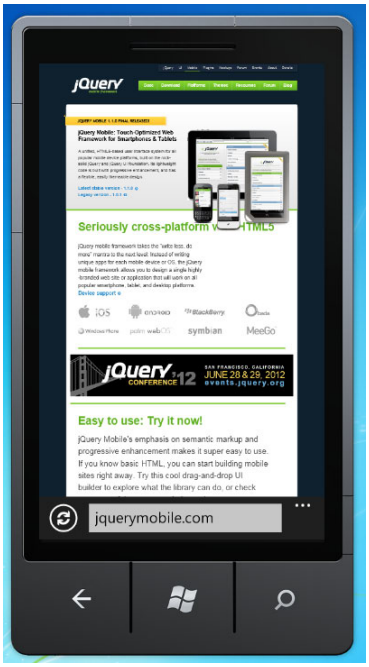

Abb. 2.3: Die externe Seite wurde per Link geladen

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

Zusammenfassung

Sie haben in diesem Kapitel jQuery Mobile in einem ersten Kontakt kennengelernt. Sie kennen die perfekte Grundstruktur als Basiswebseite und können alle notwendigen Ressourcen für eine Webseite bzw. Web-App mit jQuery Mobile einbinden. Insbesondere wurde die Bedeutung des Attributs `data-role` und dem spezielle Verständnis einer Seite unter jQuery Mobile erklärt.

Aufgaben zur Selbstüberprüfung

2.1 Wie werden gewöhnliche DIV-Elemente so gekennzeichnet, dass sie spezielle Rollen in jQuery Mobile erfüllen? Kreuzen Sie die richtige Antwort an:

- Mit dem Attribut data-role.
- Mit dem Attribut target.
- Mit einem enthaltenen Kindelement content.
- Mit einem umgebenden Elternelement content.

2.2 Wofür steht die Abkürzung CDN?

_____

2.3 In welcher Reihenfolge sollten Sie CSS-Referenzen und Skript-Referenzen notieren? Kreuzen Sie die richtige Antwort an:

- Zuerst die Referenzen auf die Skripte und dann die Referenzen auf die CSS-Ressourcen.
- Zuerst die Referenzen auf die CSS-Ressourcen und dann die Referenzen auf die Skripte.
- Es kommt auf die konkrete Situation an.

2.4 Was ist die Funktion des Attributs `data-rel="back"` in jQuery Mobile?

_____

© SGD, 13.09.18, Stojanovic, Milan (919400)

3 jQuery Mobile – erweiterte Features

Lernziele

jQuery Mobile kennt interessante Features, um eine mobile Webseite bzw. Web-App zu optimieren. Wir widmen uns nun diesen Features. Das umfasst Seitenübergänge, die Gestaltung mit CSS-Themen und leistungsfähige Widgets wie Buttons, Dialoge oder Gruppierungen.

3.1 Die Übergänge

Im jQuery Mobile-Framework gibt es mehrere CSS-basierende Übergangseffekte zwischen Seiten oder Objekten, die bei einem Seitenwechselereignis oder dem Anzeigen eines neuen Elements zugewiesen werden können.[17] In der Voreinstellung werden Übergänge von rechts nach links durchgeführt. Wollen Sie das ändern, verwenden Sie einfach das Attribut `data-transition` beim Link, der den Übergang auslöst. Beispiel:

```
<a href="index.html" data-transition="pop">Home</a>
```

Die Übergangsvarianten werden permanent erweitert. Mögliche Werte in der Version 1.1 von jQuery Mobile sind `none`, `turn`, `flow`, `slidefade`, `slide`, `slideup`, `slidedown`, `pop`, `fade` und `flip`. Sie können einen identischens Übergang für den Weg zurück einfach über `data-direction="reverse"` festlegen.

Übung 3.1:

Verschiedene Übergänge (Übungen/Kapitel3/jqm3.html)

Nehmen Sie das letzte Beispiel *jqm2.html* aus Kapitel 2 und testen Sie die verschiedenen Übergänge zwischen den Seiten, indem Sie die möglichen Werte beim Attribut `data-transition` angeben und die Seite in einem kompatiblen Browser, einem Emulator oder Ihrem Smartphone testen (*jqm3.html*). Beachten Sie, dass manche Desktop-Browser die Übergänge nicht anzeigen.

3.2 Dialoge

Unter einem Dialog versteht man einen Bereich der Seite, der wie ein klassisches Dialogfenster aussieht und sich auch so verhält. Auch Dialoge werden im Framework auf einfachste Weise über Attribute generiert. Sie brauchen im Link auf eine neue Seite oder ein neues `page`-Segment nur `data-rel="dialog"` anzugeben. Das Framework kümmert sich automatisch darum, dass der neu anzuzeigende Inhalt in einem formatierten „Dialogfenster“ angezeigt wird. Diese Dialoge werden im Framework grundsätzlich als ***modal*** erstellt und bei der Anzeige wird der Hintergrund abgedunkelt sowie die Applikation angehalten. Beispiel:

```
<a href="dialog.html" data-rel="dialog">Neue Dialog</a>
```

Noch einmal zur Verdeutlichung: Die Tatsache, dass eine Seite ein Dialog ist, wird nicht (!) bei der Seite, sondern in dem öffnenden Link festgelegt.

17. Wenn der entsprechende Browser das auch unterstützt – andernfalls wird der neue Inhalt einfach unmittelbar angezeigt.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Die Übergänge zwischen einer Seite und einem Dialog können wie beim Wechseln zwischen Standardseiten angegeben werden. Geschlossen werden Dialoge vom Framework automatisch, wenn ein Link darin angeklickt wird. Damit können Sie auch ganz einfach einen Schließen-Button erstellen – Sie brauchen bloß einen Link auf die Seite zu setzen, von der aus der Dialog geöffnet wurde. Etwa über `data-rel="back"` im Link. Ein Dialog verfügt auch über einen Schließbutton, über den der Dialog geschlossen werden kann. Alternativ können Sie aber auch den Parameter `close` in der `dialog()`-Methode verwenden, um den Dialog von außen zu schließen. Beispiel:

```
$('#meinDialog').dialog('close')
```

Übung 3.2:

Dialoge verwenden (Übungen/Kapitel3/jqm4.html und Übungen/Kapitel3/lib/js/jqm_ready4.js)

Wir nehmen wieder das Beispiel *jqm2.html* aus Kapitel 2 und modifizieren es so, dass wir Dialoge verwenden. Dazu legen wir auch Übergänge mit dem Attribut `data-transition` fest. Erstellen Sie den folgenden Quellcode(*jqm4.html*). Der Head der Webseite bleibt weitgehend unverändert und wird nicht abgedruckt, nur referenzieren wir auf die JavaScript-Datei *jqm_ready4.js*:

```
<body>
    <div data-role="page" id="p1">
        <div data-role="header"><h1>Seite 1</h1></div>
        <div data-role="content"><h2>Willkommen</h2>
            <a href="#p2" data-transition="flip">Zur
            Seite 2</a><br />
            <a href="#p3" data-transition="flip" data-
            rel="dialog">Zum Dialog 1</a><br />
            <a href="#p4" data-transition="flip" data-
            rel="dialog">Zum Dialog 2</a><br />
            <a href="#p5" data-transition="turn" data-
            rel="dialog">Zum Dialog 3</a>
        </div>
        <div data-role="footer"><h4>Seiten und Dialo-
        ge</h4></div>
    </div>
    <div data-role="page" id="p2">
        <div data-role="header"><h1>Seite 2</h1></div>
        <div data-role="content"><p>
            <a data-rel="back" href="#" data-di-
            rection="reverse">Zurück</a></p></
            div>
        <div data-role="footer"><h4>Eine Seite</h4></
        div>
    </div>
    <div data-role="page" id="p3">
```

© SGD, 13.09.18, Stojanovic, Milan (919400)


```
                <div data-role="header"><h1>Dialog 1</h1></div>
                <div data-role="content"><p>
                        <a data-rel="back" href="#" data-
                        direction="reverse">Zurück</a></p></
                        div>
                <div data-role="footer"><h4>Ein Dialog</h4></
                div>
        </div>
        <div data-role="page" id="p4">
                <div data-role="header"><h1>Dialog 2</h1></div>
                <div data-role="content"><p>
                        <a href="#p1" data-
                        transition="slide">Zurück</a></p></
                        div>
                <div data-role="footer"><h4>Ein Dialog</h4></
                div>
        </div>
        <div data-role="page" id="p5">
                <div data-role="header"><h1>Dialog 3</h1> </
                div>
                <div data-role="content"><p>
                        <a data-transition="slidedown"
                        id="retur">Zurück</a></p></div>
                <div data-role="footer"><h4>Ein Dialog</h4></
                div>
        </div>
    </body>
</html>
```

Sie sehen in diesem Listing, dass wir insgesamt fünf `page`-Segmente mit eindeutigen Ids verwenden – zwei Seiten und drei Dialoge. Aus der Startseite wird auf eine Folgeseite und die drei Dialoge verlinkt. Nur die Kennzeichnung mit `data-rel="dialog"` macht aus dem jeweiligen `page`-Segment einen Dialog. Die Übergänge werden konsistent mit `data-transition` angegeben. Aus der Folgeseite bzw. den Dialogen gehen wir immer wieder zurück auf die Startseite. Dabei verwenden wir alle beschriebenen Techniken. Beachten Sie den dritten Dialog. Dort wird keines der spezifischen Attribute zum Zurückgehen auf die aufrufende Seite verwendet. Aber wir haben ja noch unsere eigene JavaScript-Datei *jqm_ready4.js*:, die wie folgt aussieht und die Sie bitte auch erstellen:

```
$( function() {
        $("#retur").click( function() {
                $('#p5').dialog('close')
        });
})
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Mithilfe der klassischen jQuery-Syntax ordnen wir einem Element mit der Id `#retur` die Reaktion auf einen Klick des Anwenders zu (mit dem Eventhelper `click()`) und dort selektieren wir über die Id den dritten Dialog, dessen `dialog()`-Methode wir – wie oben beschrieben – mit dem Parameter `'close'` aufrufen. Auch darüber kommen wir zur aufrufenden Seite zurück.

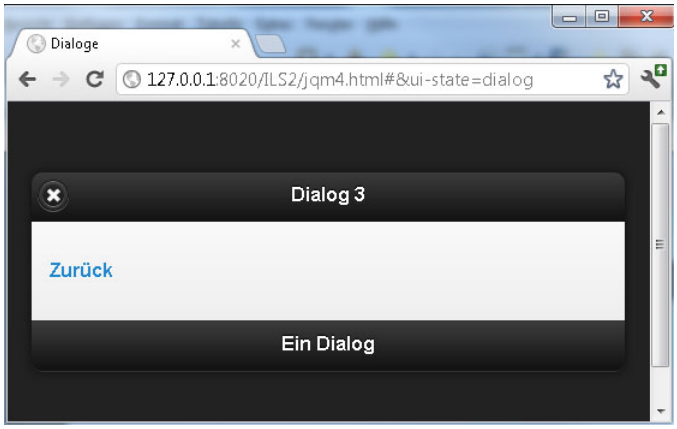

Abb. 3.1: Ein Dialog in jQuery Mobile – hier in einem Desktop-Browser

3.3 Schaltflächen

Während die Bedienung klassischer Web-Applikationen auf dem PC stark am Mauszeiger orientiert ist, orientieren sich mobile Oberflächen an der Bedienung mit dem Finger (Stichwort Touchscreens). Und damit kann man nur ungenau bzw. schwierig den engen sensitiven Bereich eines klassischen Hyperlinks treffen. Von daher müssen sensitive Elemente meist größer werden, um dem Anwender eine bequeme Bedienung zu gewährleisten. Dazu bieten sich Schaltflächen an.

Genauso einfach wie Dialoge werden Schaltflächen erzeugt. Sie müssen nicht mehr tun als in einem Link dem Attribut `data-role` den Wert `button` zuzuweisen. Beispiel:

```
<a href="index.html" data-role="button">Klick</a>
```

Alternativ werden auch Formularelemente vom Typ `input` mit dem `type`-Attributwert `submit`, `reset`, `button` oder `image` vom Framework in einen individuell gestalteten Button konvertiert.

Hinweis:

Beachten Sie den konzeptionellen Unterschied zwischen Dialogen und Schaltflächen. Während bei der Seite, die einen Dialog darstellt, keine explizite Auszeichnung stattfindet, wird im Fall von Schaltflächen der Hyperlink zu einem Button gewandelt (was auch bei den anderen Widgets der Fall ist).

Übung 3.3:

Button einsetzen (Übungen/Kapitel3/jqm5.html)

Speichern Sie das letzte Beispiel *jqm4.html* unter dem Namen *jqm5.html* und fügen Sie in alle Hyperlinks `data-role="button"` hinzu.

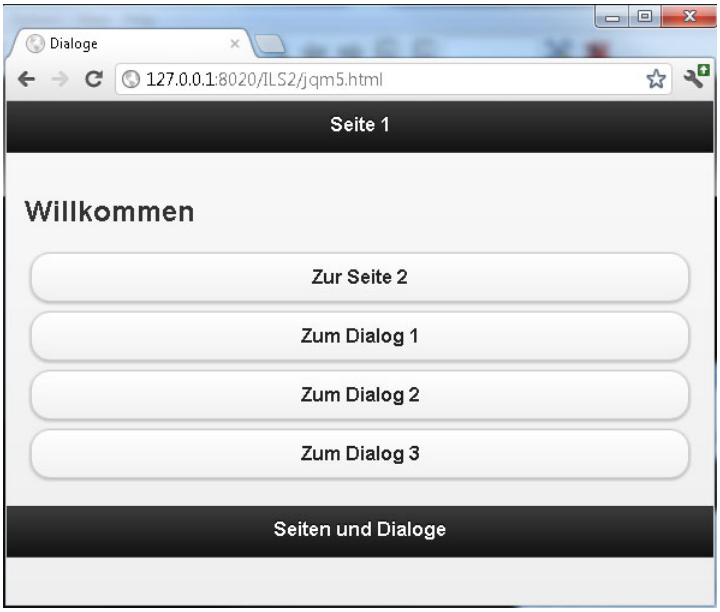

Abb. 3.2: Die Links werden als Button dargestellt

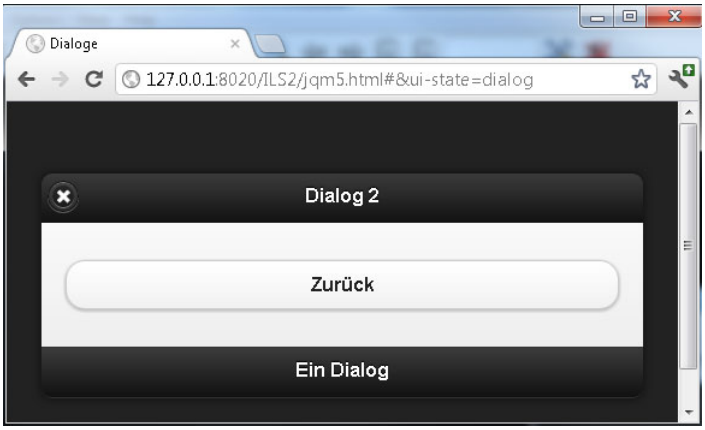

Abb. 3.3: Auch im Dialog kann man diese Schaltflächen verwenden

3.3.1 Schaltflächen mit Icons

Sie können auf einfache Weise grafische Symbole den Buttons hinzufügen. Dazu stellt jQuery Mobile einen Satz an häufig gebrauchten Icons für mobile Apps bereit. Diese sind auf minimale Größe und maximalen Kontrast hin optimiert. Um diese grafischen Symbole zu verwenden, brauchen Sie bloß das `data-icon`-Attribut einem Button hinzuzufügen. Beispiel:

```
<a href="index.html" data-role="button" data-
icon="delete">Löschen</a>
```

Die folgenden Icons stehen etwa zur Verfügung:

Symbol	Beschreibung
`arrow-l, arrow-r, arrow-u , arrow-d`	Pfeile nach links, rechts, oben und unten
`delete`	Löschen
`plus, minus`	Plus- und Minuszeichen
`check`	Harken
`gear`	Getriebesymbol
`refresh`	Aktualisieren
`forward, back`	Vorwärts und Zurück
`grid`	Hash
`star`	Stern
`alert`	Achtungschild
`info`	Infosymbol i
`home`	Home
`search`	Lupe zum Suchen

Die Position der Symbole ist in der Vorgabe links vom Text der Schaltfläche. Zum Überschreiben gibt es aber das Attribut `data-iconpos`. Damit können Sie das Symbol nach rechts (`right`), über (`top`) oder unter (`bottom`) den Text setzen. Mit `data-iconpos="notext"` verzichten Sie ganz auf Text bei der Schaltfläche. Beispiel:

```
<a href="index.html" data-role="button" data-icon="delete" data-
iconpos="right">Löschen</a>
```

© SGD, 13.09.18, Stojanovic, Milan (919400)

Übung 3.4:

Button mit Symbolen versehen (Übungen/Kapitel3/jqm6.html)

Erstellen wir ein neues Beispiel *jqm6.html* und fügen wir in verschiedene Schaltflächen Symbole auf die beschriebene Weise hinzu. Und wir steuern auch die Positionen:

```
<body>
      <div data-role="page" id="p1">
            <div data-role="header"><h1>Seite 1</h1></div>
            <div data-role="content">
                  <a href="http://www.w3c.org" data-
                  transition="flip" data-role="button"
                        data-icon="arrow-r">W3C</a>
                  <a href="http://jquery.com" data-
                  transition="flip" data-rel="dialog"
                        data-role="button" data-
                        icon="info">jQuery</a>
                  <a href="http://jqueryui.com" data-
                  transition="flip" data-rel="dialog"
                        data-role="button" data-icon="info"
                        data-iconpos="right">jQuery UI</a>
                  <a href="http://jquerymobile.com" data-
                  transition="turn" data-rel="dialog"
                        data-role="button" data-icon="info"
                        data-iconpos="notext">jQuery
                        Mobile</a>
            </div>
            <div data-role="footer"><h4>Button mit
            Symbolen</h4></div>
      </div>
</body>
</html>
```

Sie sehen hier einmal zwei verschiedene Symbole in der Standardeinstellung (erster und zweiter Link) und dann das zweite Symbol auf der rechten Seite (dritter Link) und ohne Text (vierter Link).

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

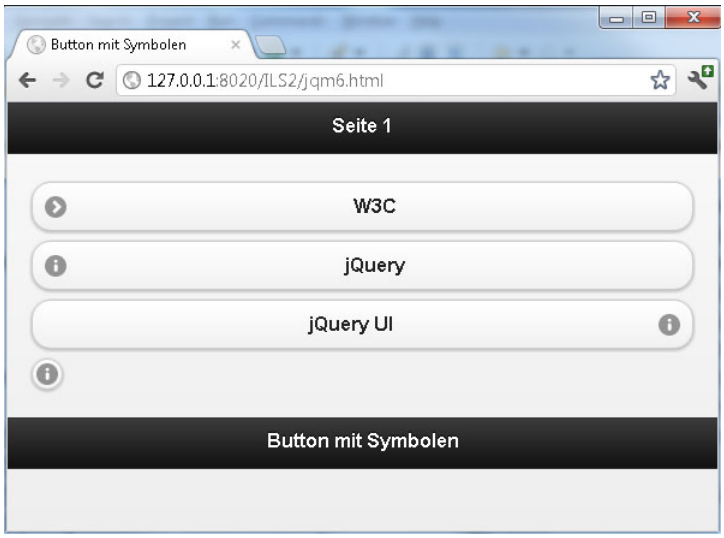

Abb. 3.4: Schaltflächen mit Symbolen

3.3.2 Blockelement oder Inlineelement

In der Grundeinstellung werden Schaltflächen als Blockelemente generiert. Zumindest im zentralen Inhaltsbereich der Seite. Das bedeutet, sie nehmen die gesamte Breite einer Seite ein und erzeugen einen Umbruch. Wenn Sie `data-inline="true"` als Attribut beim Button setzen, erzeugen Sie jedoch ein Inlineelement, das nur den Raum einnimmt, den der Inhalt erzwingt. Außerdem unterbleibt der Zeilenumbruch. Entsprechend wird der Wert `false` ein Blockelement bewirken. In dem Fußbereich und Header werden Schaltfläche automatisch als Inlineelement formatiert und müssten bei Bedarf zu Blockelementen umgewandelt werden.

Übung 3.5:

Button als Inline-Elemente (Übungen/Kapitel3/jqm7.html)

Modifizieren Sie die Seite *jqm6.html* und fügen Sie in der resultierenden Datei *jqm7.html* in allen Schaltflächen `data-inline="true"` als Attribut hinzu. Erstellen Sie zudem im Fußbereich eine Schaltfläche mit den reinen Grundeinstellungen. Sie werden sehen, dass diese Schaltfläche als Inlineelement dargestellt wird.

© SGD, 13.09.18, Stojanovic, Milan (919400)

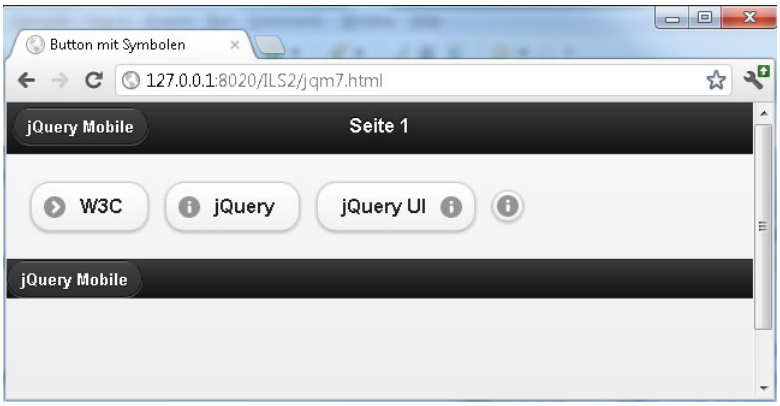

Abb. 3.5: Inline-Schaltflächen

3.3.3 Gruppierung von Schaltflächen

Sie können Schaltflächen visuell zu Gruppen zusammenfassen. Dazu werden sie in einen Container gepackt, der mit dem Attribut `data-role="controlgroup"` versehen ist. In der Grundeinstellung werden die Schaltflächen vertikal zusammengefasst und alle Abstände zwischen den Buttons beseitigt. Auch die Schatteneffekte werden so umgestaltet, dass die Gruppe optisch als gemeinsame Einheit auftritt. Beispiel:

```
<div data-role="controlgroup">
      <a href="eins.html" data-role="button">Ja</a>
      <a href="zwei.html" data-role="button">Nein</a>
      <a href="#" data-role="button">Abbruch</a>
</div>
```

Mit `data-type="horizontal"` können Sie die Anordnung auch horizontal vornehmen. Dann werden die Schaltflächen als Inlineelemente formatiert.

Übung 3.6:

Gruppierte Schaltflächen (Übungen/Kapitel3/jqm8.html)

Erstellen Sie auf Basis unserer Grundschablone die nachfolgende Datei *jqm8.html*, in der zwei Gruppierungen von Schaltflächen vorgenommen werden. Zuerst werden zwei Schaltflächen horizontal angeordnet. Danach folgt die Gruppierung von drei Schaltflächen ohne ausdrückliche Anordnung. Das führt zu einer Anordnung untereinander (vertikal):

```
<body>
      <div data-role="page" id="p1">
            <div data-role="header"><h1>Button</h1></div>
            <div data-role="content">
                  <div data-role="controlgroup" data-
                  type="horizontal">
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

```
                        <a href="http://www.w3c.org" data-
                        role="button">W3C</a>
                        <a href="http://validator.w3.org"
                        data-role="button">Validator</a>
                    </div>
                    <div data-role="controlgroup">
                        <a href="http://jquery.com" data-
                        rel="dialog" data-
                        role="button">jQuery</a>
                        <a href="http://jqueryui.com" data-
                        rel="dialog" data-
                        role="button">jQuery UI</a>
                        <a href="http://jquerymobile.com"
                        data-rel="dialog" data-
                        role="button">jQuery Mobile</a>
                    </div>
                </div>
                <div data-role="footer"><h4>Gruppierung</h4></
                div>
            </div>
        </body>
</html>
```

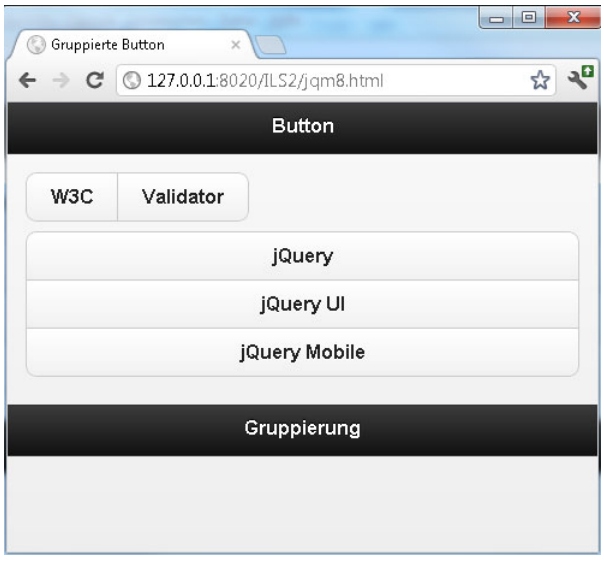

Abb. 3.6: Gruppierte Schaltflächen

© SGD, 13.09.18, Stojanovic, Milan (919400)

3.4 Listen, Toolbars und Navigationsbars

Eine besondere Form an Widgets sind Toolbars bzw. Navigationsbars. Diese werden im Kopf- oder Fußbereich einer Seite verwendet. Im Kopfbereich, der so gut wie immer die Seite beginnt, werden neben der Titelzeile unter Umständen auch ein bis zwei Schaltflächen zu finden sein. Die Fußzeile beschließt die Seite. Es ist nun recht bequem, wenn Sie in diesen Bereichen eine horizontale Navigation oder eine Aufteilung in Tabellenblätter haben. Das Framework beinhalten dazu ein Navigationsbar-Widget (`data-role="navbar"`), um eine unsortierte Liste mit Links in eine horizontale Leiste mit Schaltflächen zu überführen. Um einen der Links als vorselektiert (aktiv) zu setzen, fügen Sie `class="ui-btn-active"` dem Ankerelement hinzu.

Übung 3.7:

Navigationsleisten verwenden (Übungen/Kapitel3/jqm9.html)

Erstellen Sie auf Basis unserer Grundschablone die nachfolgende Datei *jqm9.html*:

```
<body>
      <div data-role="page" id="p1">
            <div data-role="header">
                  <div data-role="navbar"><ul>
                        <li><a href="http://jquery.com"
                        class="ui-btn-active">Home</a></
                        li>
                        <li><a href="http://
                        jqueryui.com">jQuery UI</a></li>
                        <li><a href="http://
                        jquerymobile.com">jQuery Mobile</
                        a></li>
                  </ul>
                  </div>
            </div>
            <div data-role="content">Navigation über die Kopf-
            und Fußzeile</div>
            <div data-role="footer">
                  <div data-role="navbar"><ul>
                        <li><a href="http://www.w3c.org"
                        data-role="button">W3C</a></li>
                        <li><a href="http://
                        validator.w3.org" data-
                        role="button">Validator</a></li>
                  </ul>
                  </div>
            </div>
      </div>
</body>
</html>
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Sie sehen in dem Kopfbereich eine Navigationsbar mit vorselektiertem Register und im Fußbereich eine weitere Navigationsleiste ohne Vorselektion.

Navigationsbars eignen sich auch zum Verschachteln von Elementen, solange das auf dem kleinen Bildschirm sinnvoll ist. Die Positionierung der Kopf- und Fußbereiche erfolgt in der Vorgabe inline. Damit befinden sich diese Elemente im normalen Fluss der Webseite. Der Vorteil ist, dass sie damit auf nahezu alle Devices sichtbar sind, da unabhängig von der Positionierung mit JavaScript und/oder CSS. Die Elemente werden einfach so angeordnet, wie Platz verfügbar ist. Gerade bei den vielen unterschiedlichen Auflösungen im mobilen Umfeld ist das von Vorteil.

Abb. 3.7: Navigationsleisten

3.4.1 Listen

Listen erlauben eine übersichtliche Darstellung von Daten, was bei den kleinen Displays im mobilen Umfeld umso wichtiger ist. Auch die Navigation lässt sich mit Listen gestalten. Das Framework stellt deshalb eine Anzahl an Standardlisten zur Verfügung, die vielfältig formatiert werden können. Das Framework unterstützt verschachtelte Listen, nummerierte Listen, Listen ohne Klickunterstützung (inaktive bzw. read-only-Listen), Listen mit Icons, eingerückte Listen und einige weitere Varianten.

Grundsätzlich brauchen Sie zum Aufbau einer Liste nur die üblichen HTML-Tags verwenden und das Attribut `data-role="listview"` dem Elternelement hinzufügen. Das Framework wird automatisch notwendige Stile und Unterstützung für Ereignisse hinzufügen. Vollkommen im Hintergrund organisiert es bei Bedarf AJAX-Anfragen nach neuen Inhalten sowie den Einbau in den DOM etc. So könnte eine Basisliste aussehen:

```
<ul data-role="listview" data-theme="g">
      <li><a href="http://jquery.com" class="ui-btn-active">Home</
      a></li>
      <li><a href="http://jqueryui.com">jQuery UI</a></li>
      <li><a href="http://jquerymobile.com">jQuery Mobile</a></li>
      </ul>
```

© SGD, 13.09.18, Stojanovic, Milan (919400)

3.5 Formularelemente

Wesentliche Interaktionskomponenten in einer GUI basieren auf Formularelementen. Das Framework jQuery Mobile stellt alle gängigen Formularelemente in einer **optimierten Form** für mobile Devices bereit. Insbesondere ist die Bedienung mit dem Finger explizit vorgesehen. Aber im Grunde basieren alle Formularelemente auf den üblichen nativen HTML-Formularelementen, die vom Framework im Hintergrund initialisiert und dabei angepasst werden.

Hinweis:

Wenn das Framework ein Formularelement nicht anpassen soll, setzen Sie für dieses das Attribut `data-role="none"`.

Die Formularelemente sollten unbedingt in den üblichen `<form>`-Container eingeschlossen werden und eindeutig gekennzeichnet sein. Grundsätzlich werden alle Formularelemente mit einer dynamischen Breite versehen, um sich unterschiedlichen Bildschirmbreiten anzupassen. Auch die Anordnung zwischen verschiedenen Elementen kann sich je nach Bildschirmbreite unterscheiden (übereinander oder nebeneinander)

3.5.1 Feldcontainer

Um die Anpassung von Labels und Formularelementen an breitere Bildschirme zu gewährleisten, empfiehlt die Dokumentation, diese mit einem div- oder fieldset-Element zu ummanteln und diesem das Attribut `data-role="fieldcontain"` hinzuzufügen. Das Framework verwaltet die Elemente dann gemeinsam und gestaltet sie zudem mit Rahmen etc.

Übung 3.8:

Formularelemente mit jQuery Mobile (Übungen/Kapitel3/jqm10.html)

Da Ihnen die verschiedenen Formularelemente bekannt sind, wollen wir einfach ein vollständiges Beispiel ansehen, in dem die wichtigsten verfügbaren Formularelemente notiert werden. Erstellen Sie auf Basis unserer Grundschablone die nachfolgende Datei *jqm10.html*:

```
<body>
      <div data-role="page" id="p1">
            <div data-role="header"><h1>Formulare</h1></div>
            <div data-role="content">
                  <form action="#" method="get">
                        <div data-role="fieldcontain"><label
                        for="name">Einzeilige Texteingabe:</label>
                              <input name="name" id="name" type="text" /></div>
                        <div data-role="fieldcontain"><label
                        for="textarea">Mehrzeilige Texteingabe:</label>
                              <textarea cols="25" rows="4" name="textarea"
                              id="textarea"></textarea></div>
                        <div data-role="fieldcontain"><label
                        for="slider">Schieberegler:</label>
                              <input type="range" name="slider" id="slider"
                              value="50" min="0" max="100" /></div>
```

```
                        <div data-role="fieldcontain"><label
                        for="um">Umschalter</label>
                              <select name="um" id="um" data-role="slider">
                                    <option value="off">An</option><option
                                    value="on">Aus</option></select>
                        </div>
                        <div data-role="fieldcontain">
                              <fieldset data-role="controlgroup">
                                    <legend>Optionsfelder</legend>
                                    <input type="radio" name="radio-choice-1"
                                    id="radio-choice-1" value="choice-1"
                                    checked="checked" />
                                    <label for="radio-choice-1">Ja</label>
                                    <input type="radio" name="radio-choice-1"
                                    id="radio-choice-2" value="choice-2" />
                                    <label for="radio-choice-2">Nein</label>
                              </fieldset>
                        </div>
                        <div data-role="fieldcontain">
                              <fieldset data-role="controlgroup">
                                    <legend>Kontrollkästchen:</legend>
                                    <input type="checkbox" name="checkbox-1"
                                    id="checkbox-1" />
                                    <label for="checkbox-1">An</label>
                              </fieldset>
                        </div>
                        <div data-role="fieldcontain">
                              <label for="select-choice-1"
                              class="select">Auswahlmenü</label>
                              <select name="select-choice-1" id="select-choice-
                              1">
                                    <option value="w">W3C</option><option
                                    value="j">jQuery</option>
                                    <option value="p">Phone Gap</option>
                              </select>
                        </div>
                  </form>
            </div>
            <div data-role="footer">Mobile Version</div>
      </div>
</body>
</html>
```

© SGD, 13.09.18, Stojanovic, Milan (919400)

Sie sehen, dass der gesamte Inhaltsbereich der App aus einem Formular besteht. Darin sehen Sie innerhalb der einzelnen Container mit der Auszeichnung `data-role="fieldcontain"` verschiedene mögliche Formularelemente. Bemerkenswert ist die Auszeichnung des fieldsets mit `data-role="controlgroup"` für die Radiobuttons und `data-role="controlgroup"` für Checkboxen.

Abb. 3.8: Das Formular mit Standardelementen

3.6 Das Themenframework und die allgemeine Gestaltung von Inhalt

Grundsätzlich versucht man in jQuery Mobile, die Gestaltung von Inhalten möglichst weit dem Browser zu überlassen und möglichst wenig Layoutoverhead zu übertragen oder gar ein einheitliches Design zu erzwingen. In der Vorgabeeinstellung verwendet das Framework die Standardstile und -größenangaben von HTML. Im Grunde fügt das Framework auf der Ebene nur ein paar Stile für Tabellen und Fieldsets hinzu, um sie etwas besser handhabbar zu machen. Alles andere wird in schlanke CSS-Klassen verlagert.

Unter jQuery Mobile gibt es dazu ein reichhaltiges Themenframework. Die zentrale Zuordnung von CSS-Regeln erfolgt über das `data-theme`-Attribut. Dieses kann dem Kopf- und Fußbereich zugewiesen werden, um die Darstellung dort anzupassen. Grundsätzlich können Sie dieses Attribut auch dem eigentlichen Inhaltsbereich einer Seite zuweisen, aber in der

Dokumentation des Frameworks wird empfohlen, dass das Attribut stattdessen einzelnen Seiten zugewiesen werden soll. Also allen Containern, die mit `data-role="page"` gekennzeichnet sind. Aber Sie können auch andere Widgets individuell damit gestalten.

Das gesamte Konzept des mobilen CSS-Themenframeworks ist an dem ThemeRoller von jQuery UI orientiert, fügt aber einige Erweiterungen hinzu, die sich aufgrund der speziellen mobilen Bedingungen ergeben. Insbesondere nutzt man intensiv Eigenschaften aus CSS3, um etwa abgerundete Ecken und Gradienten (Farbverläufe) umzusetzen sowie maximal komprimierte Icons. Dazu werden in jedem Thema mehrere Farbkombinationen und verschiedene Designs angeboten.

Grundsätzlich trennt das mobile Themensystem Farbe und Text von Strukturstilen wie Dimensionen und Pufferung. Damit können diese Regeln einmal definiert und beliebig gemischt bzw. kombiniert werden. Die Vorgabethemen in jQuery Mobile sind einfach alphabetisch durchnummeriert (a, b, c, d, e). Dabei verwendet a den maximalen Kontrast und e die meisten Farben.

Übung 3.9:

Themen auswählen (Übungen/Kapitel3/jqm11.html und Übungen/Kapitel3/lib/css/jqm11.css)

Betrachten wir ein kleines Beispiel, in dem wir verschiedene Themen zuweisen werden (*jqm11.html*):

```
<body>
      <div data-role="page" id="p1" data-theme="e">
            <div data-role="header" data-theme="c"><h1>Seite 1</
            h1></div>
                  <div data-role="content">
                  <a href="http://www.w3c.org" data-
                  role="button">W3C</a>
                  <a href="http://jquery.com" data-rel="dialog"
                  data-role="button">jQuery</a>
                  <a href="http://jqueryui.com" data-
                  rel="dialog" data-role="button">jQuery UI</a>
                  <a href="http://jquerymobile.com" data-
                  rel="dialog" data-role="button">jQuery
                  Mobile</a>
            </div>
            <div data-role="footer" data-theme="b"><h4>Themen</
            h4></div>
      </div>
</body>
</html>
```

Selbstverständlich können Sie auch eigene CSS-Regeln in Verbindung mit Themen verwenden. Dazu können Sie in der referenzierten CSS-Datei in dem Beispiel den Hintergrund der Überschrift vom Typ `h1` auf Blau und die Textfarbe auf Weiß setzen.

© SGD, 13.09.18, Stojanovic, Milan (919400)

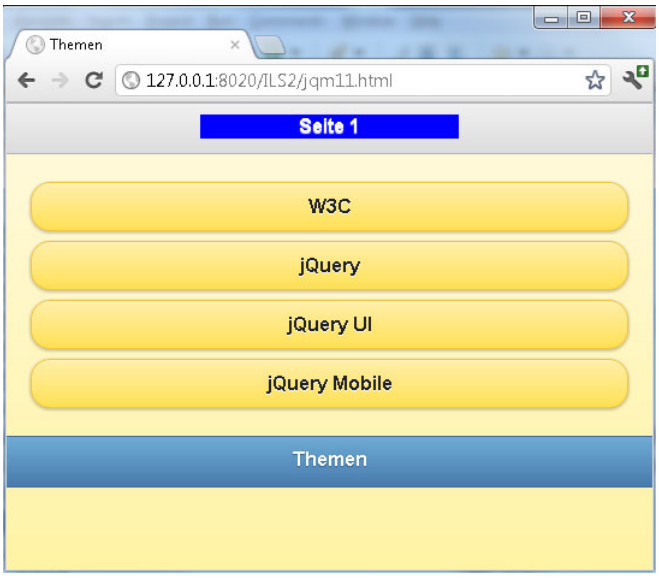

Abb. 3.9: Die Verwendung von verschiedenen Themen

3.6.1 Eigenständige Layouts und der ThemeRoller

Die Themen in jQuery Mobile sind CSS-Bibliotheken, ohne die die Komponenten und Widgets des Frameworks nicht vernünftig dargestellt werden. Diese Themen können Sie selbst verändern, um ein eigenständiges Layout zu schaffen. Aber das ist von Hand mühselig, weil viele Details explizit festgelegt werden müssen. Um diese Erstellung und Anpassung von CSS-Regeln zu vereinfachen, gibt es seit der Finalversion jQuery Mobile 1.0 im Framework ein **ThemeRoller**-Tool (http://jquerymobile.com/themeroller/).

Wenn Sie sich auf der Webseite des ThemeRollers umsehen, sehen Sie auf der linken Seite eine Benutzerschnittstelle zum Designen aller Elemente, die vom Framework verwendet werden. Sie können jedes Standardthema anpassen oder auch ein eigenes, neues Thema hinzufügen. Rechts von der Spalte sehen Sie unmittelbar bei den Beispielen der verschiedenen Komponenten, wie sich Ihre Änderungen auf das aktuelle Design auswirken werden. Wenn Ihr Design fertig ist, müssen Sie es nur noch laden, auf dem Server zur Verfügung stellen und in Ihrer Webseite einbinden. Zum Download sehen Sie rechts oben in der Webseite des ThemeRollers eine entsprechende Schaltfläche.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

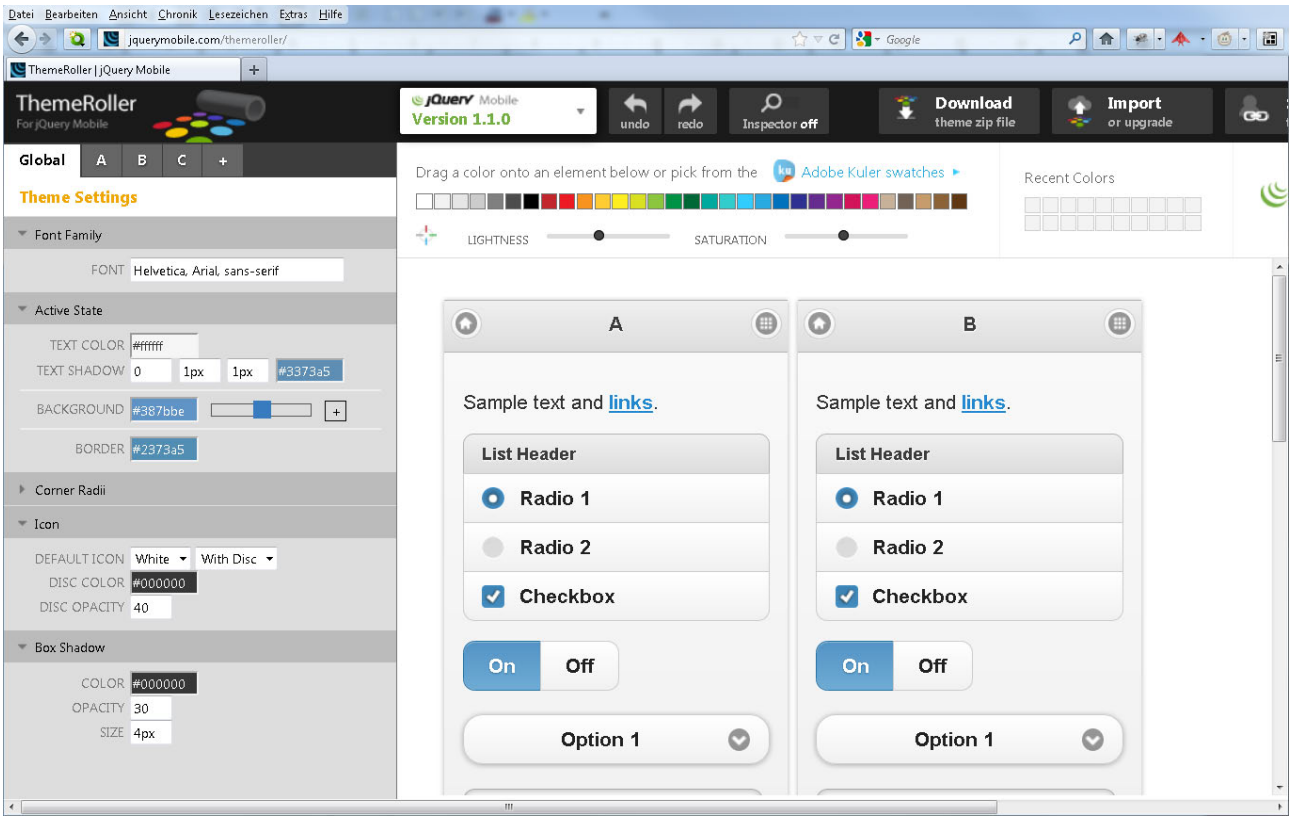

Abb. 3.10: Der ThemeRoller

3.7 Spezielle Ereignisse

Berühren wir nun noch die spezielle Programmierung unter jQuery Mobile. Denn natürlich können Sie – wie unter normalem jQuery – auch bei jQuery Mobile eine eigene Logik programmieren.[18] Und das bedeutet eigentlich immer, dass Sie eine bestimmte Logik aufgrund eines speziellen Ereignisses ausführen wollen. Im Rahmen von mobilen Anwendungen im Allgemeinen gibt es weit mehr Ereignisse als es bei einer Anwendung für einen Desktop üblich und notwendig ist. Das mobile Framework stellt die Unterstützung für eine Reihe spezieller Ereignisse zur Verfügung. Diese werden auf Basis nativer Events erstellt. Die Events lassen sich wie gewohnt über die Ereignisbehandlung in jQuery verwenden – etwa mit `bind()` oder `live()`.

3.7.1 Der Start der Applikation

Was Sie bei einer Programmierung unter jQuery Mobile beachten müssen ist, dass die übliche `$(document).ready()`-Funktion[19] nicht die Bedeutung hat, die sie unter jQuery selbst besitzt[20]. Stattdessen sollten Sie zur Reaktion auf den Start einer mobilen App `$(document).bind ('pageinit')` verwenden. Der Grund liegt darin, dass in jQuery

18. Wir haben das in einem Beispiel angedeutet.
19. Der Handler, der die Fertigstellung des DOM überwacht.
20. Obwohl man sie verwenden kann, wenn man keine Aktionen durchführt, die die nachfolgenden Probleme haben.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Mobile viel Daten mit Ajax geladen werden, eine Seite – wie behandelt – ein Teilsegment der vollständigen Webseite ist und der Handler, der die Fertigstellung des DOM überwacht, nur bei der ersten Seite ausgeführt wird.

Hinweis:

Das erste Ausführen von eigenem Code unter jQuery Mobile ist ein etwas kritisches Thema, bei dem sich über die Entwicklung des Frameworks häufig Veränderungen ergeben haben. Sie sollten bei jeder Version des Frameworks unbedingt in der Dokumentation nachsehen, welches Event in dieser Version priorisiert wird. Sogar noch in der Betaversion von jQuery Mobile 1.1 wurde das Ereignis `pagecreate` empfohlen, was in der Finalversion wieder verworfen wurde.

Übung 3.10:

Auf den Start einer App reagiern (Übungen/Kapitel3/jqm12.html und Übungen/Kapitel3/lib/js/jqm_ready12.js)

Betrachten wir ein kleines Beispiel, in dem wir die spezielle Reaktion auf den Start der App zeigen wollen. Die eigentliche HTML-Basis ist weitgehend uninteressant (*jqm12.html*):

```
      <body>
            <div data-role="page" id="p1" data-theme="e">
                  <div data-role="header" data-
                  theme="c"><h1>Seite 1</h1></div>
                  <div data-role="content">Wir wollten in Bremen
                  kein Gegentor kassieren.
                        Das hat auch bis zum Gegentor ganz gut
                        geklappt.</div>
                  <div data-role="footer" data-
                  theme="b"><h4>Fußzeile</h4></div>
            </div>
      </body>
</html>
```

In der JavaScript-Datei *jqm_ready12.js*, die hier referenziert werden muss, machen wir aber das:

```
$(document).bind('pageinit', function() {
      $("h1").css({
            'background':'#11a', 'font-
            size':'26px', 'color':'#fff'
      });
      $("body").css({
            'color':'#11a', 'font-size':'30px'
      });
})
```


sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Sie sehen, dass wir eine Formatierung von Bestandteilen der HTML-Datei nach dem Initialisieren der Seite vornehmen. Zwar ist das in diesem einfachen Beispiel mit Kanonen nach Spatzen geschossen, weil es nur ein `page`-Segment gibt. Aber die grundsätzliche Programmiertechnik ist hier gut zu sehen.

3.7.2 Spezifische Ereignisse

Hier sind eine Reihe von Ereignissen, die in jQuery Mobile speziell auf mobile Geräte abgestimmt sind und analog mit `bind()` oder ähnlichen Methoden verwendet werden können.

Ereignis	Beschreibung
`orientationchange`	Wenn das Gerät gedreht wird, kann man in der Callback-Funktion über den zweiten Parameter und die Werte „portrait“ oder „landscape“ die Orientierung verwerten.
`pagebeforecreate`	Ein Initialisierungsereignis, das vor der Erstellung der Seite ausgelöst wird.
`pagebeforehide`	Das Ausblenden beginnt. Das beschreibt ein Ereignis, das ausgelöst wird, bevor die Seite wirklich ausgeblendet wird.
`pagebeforeshow`	Die Anzeige beginnt (meist animiert). Das beschreibt ein Ereignis, das ausgelöst wird, bevor die Seite wirklich angezeigt wird.
`pagecreate`	Ein Initialisierungsereignis, das ausgelöst wird, wenn die Erstellung der Seite beendet ist.
`pagehide`	Die Seite ist vollständig ausgeblendet.
`pageshow`	Die Seite wird vollständig angezeigt.
`scrollstart`	Die Verschiebung startet.
`scrollstop`	Die Verschiebung ist beendet
`swipe`	Horizontale Verschiebung von 30 oder mehr Pixel und weniger als 20 Pixel vertikal innerhalb einer Sekunde.
`swipeleft`	Ein swipe-Event nach links.
`swiperight`	Ein swipe-Event nach rechts.
`tap`	Eine schnelle, vollständige Berührung.
`taphold`	Ein gehaltene, vollständige Berührung.

Zusammenfassung

Sie haben in diesem Kapitel gesehen, welche speziellen Effekte und Features über jQuery Mobile in die Welt der mobilen Apps übertragen werden. Das Mobile Framework ist ein extrem vielversprechender Ansatz, um die zahllosen Schwierigkeiten mit den unterschiedlichen Plattformen und Bedingungen im mobilen Umfeld auf einer hohen Ebene zu kapseln. Es ist einfach anzuwenden und – im Rahmen seiner Möglichkeiten – sehr zuverlässig und

© SGD, 13.09.18, Stojanovic, Milan (919400)

breit unterstützt. Und es ist nahezu sicher, dass die Unterstützung auf neuen mobilen Devices zunehmen wird. Damit werden mobile RIAs auf Basis von HTML5, CSS3 und JavaScript eine ernstzunehmende Alternative zu proprietären bzw. nativen Ansätzen.

Aufgaben zur Selbstüberprüfung

3.1 Wie werden in jQuery Mobile Übergänge gekennzeichnet? Kreuzen Sie die richtige Antwort an:

- Mit dem Attribut data-rel.
- Mit dem Attribut data-transition.
- Mit dem Attribut data-role.

3.2 Wofür nutzt man in jQuery Mobile das Attribut data-icon?

3.3 Wie kann man in jQuery Mobile eine Button explizit als Inlineelement festlegen?

3.4 Wozu dient die Angabe data-role="controlgroup"? Kreuzen Sie die richtige Antwort an:

- Sie legen damit Optionsfelder fest.
- Sie können Schaltflächen damit visuell zu Gruppen zusammenfassen.
- Sie legen damit Kontrollfelder fest.
- Sie kennzeichnen damit ein Formular.

3.5 Wie heißt in jQuery Mobile das visuelle Tool zum Erzeugen eines CSS-Themes?

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

4 Native Web-Apps – PhoneGap & Co

Lernziele

Wir wollen in diesem Kapitel besprechen, wie Sie mobile Webseiten und Web-Applikationen, die rein mit HTML5, CSS und JavaScript und eventuell einem Framework wie jQuery Mobile als native Apps auf einem mobilen Endgerät bereitstellen und man bei Bedarf auch auf Hardwarekomponenten wie Kameras, GPS-Empfänger oder Beschleunigungssensoren bei Smartphones zugreifen kann.

4.1 Grundsätzliches zur Bereitstellung auf mobilen Endgeräten

Im ersten Kapitel haben wir bereits ansatzweise besprochen, wie Sie fertige native Apps, Web-Apps oder mobile Webseiten bereitstellen können. Das wollen wir noch etwas ausarbeiten. Für native Apps führt zu einer Verbreitung kaum ein Weg um einen der Marktplätze der bekannten Anbieter herum. Seien es der App Hub von Microsoft (http://create.msdn.com/en-US/), der Android Market von Google (https://play.google.com/store), der App-Store von Apple (http://store.apple.com/de) oder einem der anderen Anbieter. Mit allen oft sehr strengen und einschränkenden Regeln. Mobile Webseiten und Webapplikationen hingegen werden einfach auf ein mobiles Endgerät kopiert und dort über den Browser aufgerufen, wenn das System das Kopieren nicht einschränkt oder Sie stellen die Ressourcen wie normale Web-Angebote auf einem Webserver bereit. Das verwehrt Ihnen aber den Weg in einen der Marktplätze und damit meist die erfolgreiche kommerzielle Verwertung. Aber Sie können auch Web-Apps in native Apps überführen.

4.2 PhoneGap

PhoneGap (http://phonegap.com/) ist ein Framework, das von Adobe unter der Opensource-Lizenz bereitgestellt wird und die Cross-Plattform-Entwicklung mobiler Anwendungen erleichtern soll. Dabei ist es keine Konkurrenz zu einem Framework wie jQuery Mobile, sondern eine Ergänzung. Sie können zum Beispiel jQuery Mobile[21] und PhoneGap gut in Kombination verwenden. Auch sind die Möglichkeiten von HTML5 nutzbar. Nur wozu braucht man PhoneGap, wenn schon ein Framework wie jQuery Mobile umfangreich die Oberfläche und Logik einer App mit HTML, CSS und JavaScript unterstützt?

PhoneGap erzeugt aus diesem reinen Web-Code, der bisher rein über einen Browser auf der mobilen Plattform nutzbar ist, native Applikationen. Über sogenannte **Phonegap-Wrapper** werden native Apps für die gewünschten Zielplattformen erzeugt. Genau genommen läuft eine solche Anwendung unter Phonegap in einem unsichtbaren Browserfenster und das HTML-Grundgerüst samt der verwendeten CSS- und JavaScript-Anweisungen werden nicht umgewandelt.[22] So können PhoneGap-Applikationen etwa einheitlich im App Hub von Microsoft, im Android Market von Google oder in Apples App-Store bereitgestellt werden, sofern die Regeln dieser Marktplätze eingehalten werden.

Und zudem macht es das JavaScript-API von PhoneGap möglich, dass Sie aus JavaScript auf **Hardwarekomponenten** wie z.B. Kameras, GPS-Empfänger oder Beschleunigungssensoren zugreifen. Auch das ist ein wichtiges Feature, das Ihnen etwa jQuery Mobile alleine nicht bietet. Vielleicht sogar ist das der wichtigste Grund, warum man PhoneGap einsetzt.

21. Oder auch iWebKit, jQTouch oder Sencha Touch.
22. Der alternative Ansatz im Titanium-Framework wandelt hingegen das HTML-Grundgerüst in nativen Code um.

© SGD, 13.09.18, Stojanovic, Milan (919400)

4.2.1 Download und Bereitstellung von PhoneGap

Auf der Webseite des Projekts finden Sie einen Button, über den Sie das Framework laden können. Sie erhalten eine Zip-Datei, in der sich im Wesentlichen in einem *lib*-Verzeichnis spezifische PhoneGap-Varianten für die wichtigsten mobilen Betriebssystem befinden. Denn das ist der Preis dessen, dass PhoneGap native Apps bereitstellen will – man muss für jedes gewünschte Betriebssystem eine unterschiedliche Variante vom Framework verwenden. Das betrifft ausdrücklich nicht (!) den eigentlichen Quellcode der Web-Ressoucen selbst, sondern den oben erwähnten PhoneGap-Wrapper.

Auf der Webseite von PhoneGap ist für alle unterstützten Betriebssysteme genau beschrieben, wie Sie PhoneGap einrichten (http://docs.phonegap.com).

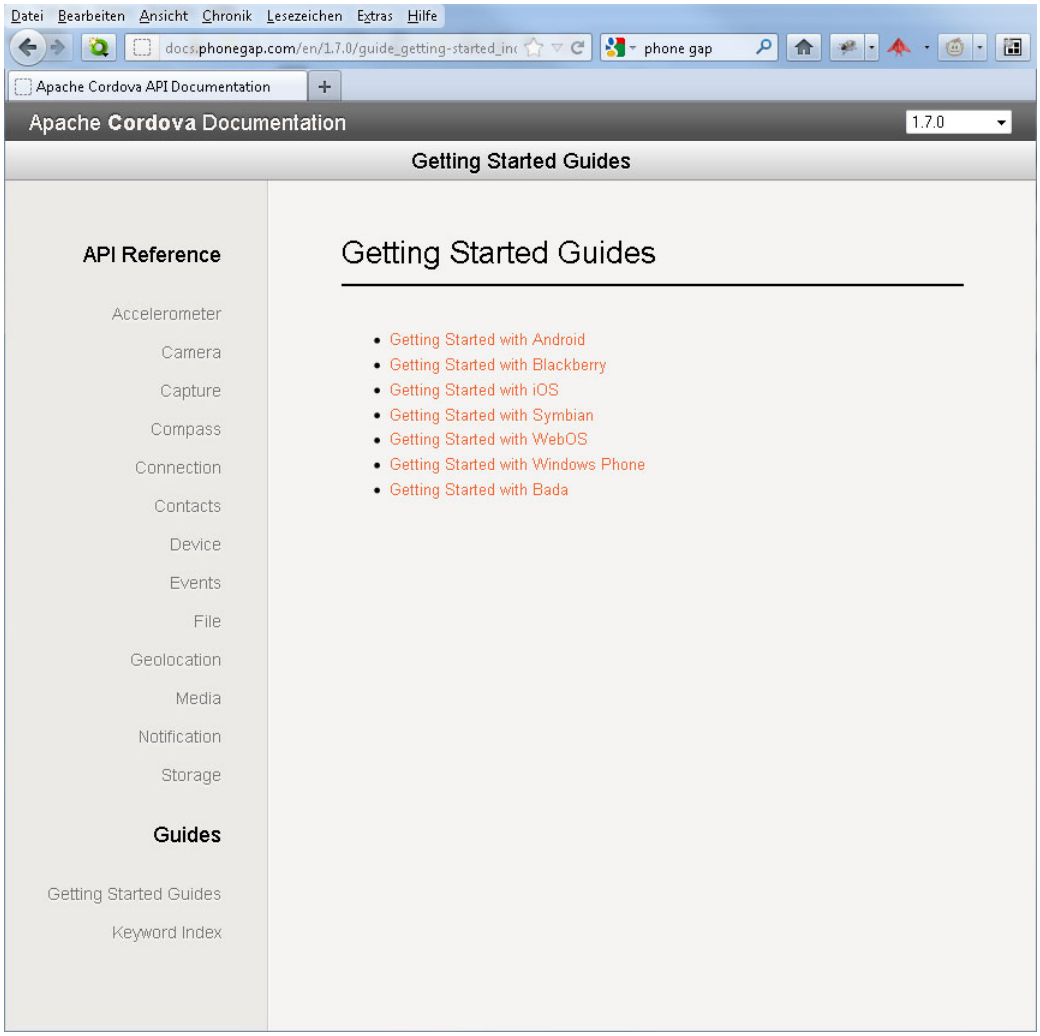

Abb. 4.1: Die offizielle Dokumentation von PhoneGap

Das jeweilige Verfahren ist leider etwas aufwendig und unflexibel, aber im Grunde nicht schwer und in der Dokumentation so genau beschrieben, dass man es eigentlich hinbekommen sollte. Wir werden das Verfahren als Übung für Android als Beispielplattform Schritt-für-Schritt durchspielen und die speziell für Java notwendigen Schritte einfach als gegeben hinnehmen.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Übung 4.1:

PhoneGap für Android einrichten (Verzeichnis Übungen/Kapitel4/PG1)

Wir haben ja in dem Kurs Eclipse samt dem ADT-PlugIn und dem SDK bereits installiert und eingerichtet.[23] Auf der Basis wollen wir das PhoneGap-Framework nun verfügbar machen.

- Legen Sie zuerst ein neues *Android Project* an. Das finden Sie in Eclipse unter dem Menüpunkt **File / New / Other / Android**. Nennen Sie das Projekt *PG1* und wählen Sie als Target die Version 2.3.3 oder höher.
- Als Package-Name müssen Sie im folgenden Dialog eine Angabe machen, die wie ein umgedrehter DNS-Name Ihrer Webseite aussieht (ohne *www*), dem dann ein Namensraum für Ihre PhoneGap-Applikationen folgt. Wählen Sie als Namensraum z. B. *phonegap*.

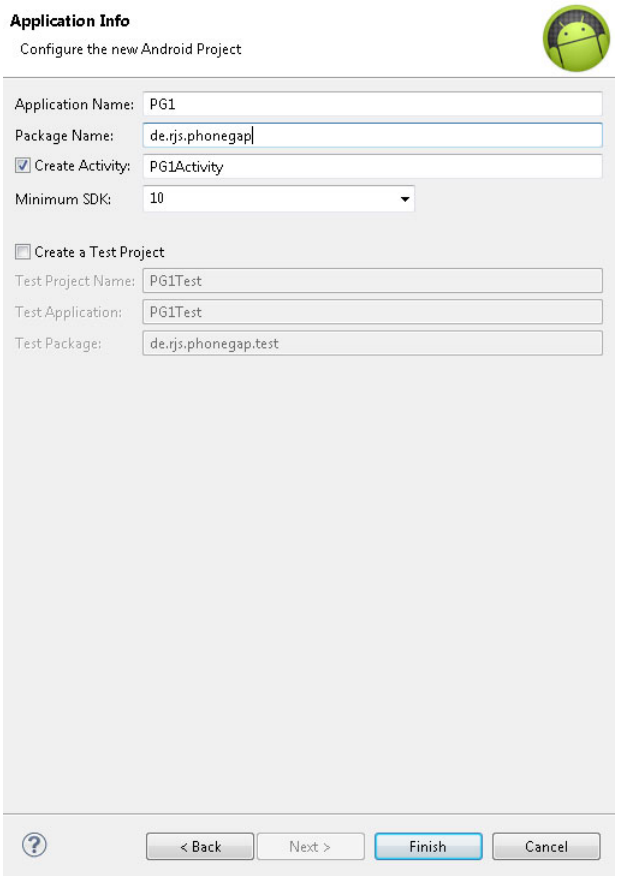

Abb. 4.2: Ein PhoneGap-Projekt in Eclipse anlegen

- Im Wurzelverzeichnis von dem neu angelegten Projekt erstellen Sie die neuen Verzeichnisse */libs* und *assets/www*.

23. Schauen Sie gegebenenfalls noch einmal in Kapitel 1 nach.

© SGD, 13.09.18, Stojanovic, Milan (919400)

- Suchen Sie im *lib*-Verzeichnis der entpackten Downloaddatei von PhoneGap für Android die Datei *cordova-1.7.0.js* . Diese Datei gehört nach den Vorgaben von PhoneGap immer in das Verzeichnis *assets/www* oder ein Unterverzeichnis davon. Wir wollen entsprechend der bisherigen Projektstrukturen ein Unterverzeichnis *lib/js* in *assets/www* verwenden. Erstellen Sie dieses und kopieren Sie diese Datei also dahin.
- Die Datei *cordova-1.7.0.jar*, die Sie ebenso im *lib*-Verzeichnis der entpackten Downloaddatei von PhoneGap finden, kopieren Sie nach */libs*. Das sind die notwendigen Java-Ressourcen von PhoneGap.
- Weiter gibt es im entpackten Downloadverzeichnis von PhoneGap für Android ein Verzeichnis *xml*, das Sie komplett in das Verzeichnis */res* Ihres Projekts kopieren müssen.
- Der nächste Schritt ist Java-spezifisch. Sie müssen unter Umständen sicherstellen, dass die speziellen Java-Ressourcen von PhoneGap von Eclipse gefunden werden. Dazu führen Sie einen Rechtsklick auf den Ordner */libs* aus und wählen Sie dann **Build Paths / Configure Build Path**.... Es kann allerdings auch sein, dass dieser Schritt gar nicht notwendig ist und Eclipse die Ressourcen schon ohne diese Maßnahme findet.

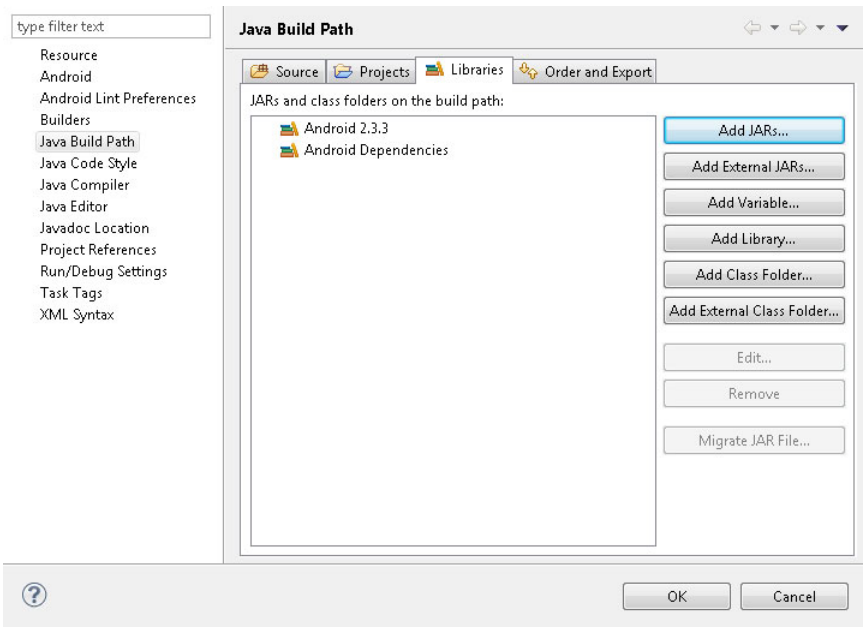

Abb. 4.3: Die Java-Bibliotheken von dem Projekt

- Im *Libraries*-Reiter fügen Sie *cordova-1.7.0.jar* über die Schaltfläche *Add Jars...* hinzu.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

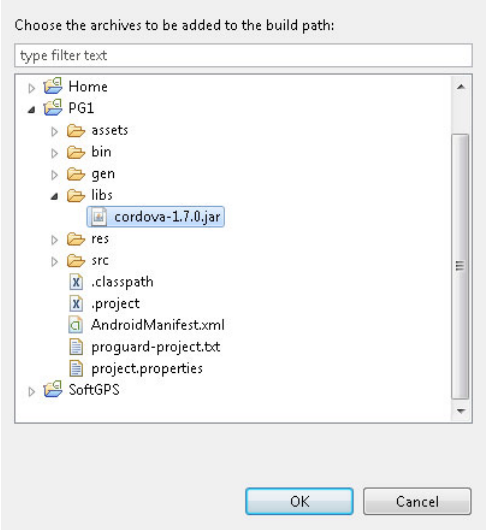

Abb. 4.4: Die Java-Ressourcen von PhoneGap Eclipse bekannt machen

- Nun müssen Sie im Quellcode kurz in die Java-Ebene, aber das wird nicht schwer. Sie müssen bloß eine Zeile Code einfügen, einen Bezeichner ändern und eine Zeile Code ersetzen. Im Verzeichnis *src* Ihres Projekts finden Sie eine Datei, die *PR1Activity.java* heißen sollte, wenn Sie das Projekt *PG1* genannt haben. Andernfalls beginnt die Datei mit Ihrem Projektnamen und dann folgt *Activity.java*. Öffnen Sie die Datei mit einem Doppelklick.
- Fügen Sie hinter der packages-Anweisung die folgende Zeile hinzu: `import org.apache.cordova.*;`
- Ändern Sie hinter dem Schlüsselwort `extend Activity` in `DroidGap`.
- Ersetzen Sie die Zeile mit `setContentView();` durch `super.loadUrl("file:///android_asset/www/index.html");`.
- Speichern Sie die Datei.
- Nun geht es an die XML-Ebene. Führen Sie in Ihrem Projekt einen Rechtsklick auf die Datei *AndroidManifest.xml* aus und selektieren Sie *Open With > XML Editor*.
- Wechseln Sie in das *Source*-Tab und fügen Sie den folgenden Code zwischen die `<uses-sdk.../>`- und `<application.../>`-Tags ein[24]:

```
<supports-screens
            android:largeScreens="true"
            android:normalScreens="true"
            android:smallScreens="true"
            android:resizeable="true"
            android:anyDensity="true" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
```

24. Diese Anweisungen legen bestimmte Verhaltensweisen Ihrer App fest. Etwa, ob sie auf die Kamera oder die GPS-Sensoren des mobilen Geräts zugreifen möchte. Sie können diese XML-Elemente auch weglassen, wenn Sie bestimmte Rechte nicht benötigen. Aber um unsere App möglichst wenig einzuschränken, fordern wir hier universelle Zugriffsrechte (auch wenn wir das nicht benötigen).

© SGD, 13.09.18, Stojanovic, Milan (919400)


```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
<uses-permission android:name="android.permissi-
on.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"
/>
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
```

- Die Unterstützung für das Wechseln der Orientierung von einem mobilen Gerät legt man im `<activity>`-Tag fest. Fügen Sie den folgenden Code innerhalb des `<activity>`-Tags hinzu:
  `android:configChanges="orientation|keyboardHidden"`
- Das sollte es nun an Anpassungen gewesen sein, was für ein universelles PhoneGap-Projekt unter Android einzurichten ist. Wenn Sie nun eine Datei mit Namen *index.html* in dem Verzeichnis *assets/www* anlegen, können darüber beliebigen Web-Code nativ verwenden. Der entscheidende Part ist, dass in der Datei natürlich eine Referenz auf die JavaScript-Datei von PhoneGap notiert werden muss. Etwa so: `<script type="text/javascript" charset="utf-8" src="cordova-1.7.0.js"></script>`. Alle weiteren Web-Ressourcen gehören ebenso einfach in den Ordner *assets/www* eingefügt.

Sie können die App nun im Emulator testen.

4.2.2 Zugriff auf Hardwarekomponenten mit PhoneGap

Neben der möglichen Integration in die verschiedenen Marktplätze über den nativen Wrapper ist eine wesentliche Motivation für den Einsatz von PhoneGap, dass dessen API Zugang zu spezifischen Hardwarekomponenten des mobilen Geräts gestattet. Das API stellt dazu geeignete Objekte zur Verfügung. Diese sind in der Dokumentation von PhoneGap inklusive Beispielen beschrieben. Wir wollen hier zwei Übungen durchführen, die den Einsatz demonstrieren.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

Übung 4.2:

Zugriff auf die Kamera – ohne jQuery Mobile (Verzeichnis Übungen/Kapitel4/PG1)

PhoneGap stellt für den Zugriff auf die Kamera ein neues Unterobjekt `camera` von dem DOM-Objekt `navigator` bereit. Dessen interessante Methode ist `getPicture()`. Damit können Sie offensichtlich Bilder aufnehmen. Die Methode besitzt zwei Callbacks als Parameter. Der erste Parameter wird im Erfolgsfall aufgerufen, der zweite im Fehlerfall. Wir erstellen nun ein Beispiel, dass ohne jQuery Mobile auskommen soll. Aber für den einfacheren Umgang mit JavaScript und dem DOM verwenden wir zumindest jQuery selbst, wobei das – um es ausdrücklich zu betonen – nicht notwendig wäre.

Erweitern Sie das oben angelegte PhoneGap-Projekt *PG1* oder legen Sie es entsprechend der oben beschriebenen Vorgehensweise als ein neues PhoneGap-Projekt an und passen Sie die dabei angelegte HTML-Datei *index.html* aus dem Verzeichnis *assets/www* wie folgt an:

```
<!DOCTYPE html>
<html lang="de">
      <head>
            <meta charset="utf-8" />
            <title>Zugriff auf die Kamera – ohne jQuery Mobile</
            title>
            <meta name="viewport" content="width=device-width,
            initial-scale=1" />
            <link rel="stylesheet" href="lib/css/pg1.css"
            type="text/css"/>
            <script type="text/javascript" charset="utf-8"
            src="lib/js/cordova-1.7.0.js"></script>
            <script type="text/javascript" src="lib/js/jquery-
            1.7.2.min.js"></script>
            <script type="text/javascript" src="lib/js/
            pg1_ready.js"></script>
      </head>
      <body>
            <h4>Zugriff auf die Kamera</h4>
            <button id="aufnahme">Photo aufnehmen</button>
            <button id="loeschen">Photo l&ouml;schen</button>
            <div id="meldung"></div><img id="bild" />
      </body>
</html>
```

Die HTML-Datei enthält die Verweise auf die notwendigen CSS- und JavaScript-Ressourcen sowie zwei Schaltflächen und einen `img`-Tag, in dem das aufgenommene Bild angezeigt werden soll. Dazu gibt es noch einen Div-Bereich, in dem eine Meldung über Erfolg oder Misserfolg der Aufnahme angezeigt werden soll. Die referenzierte CSS-Datei *pg1.css* im Unterverzeichnis *lib/css* soll wie folgt aussehen:

© SGD, 13.09.18, Stojanovic, Milan (919400)


```
#meldung{
      display:block; background:blue; color:white; height:25px;
      font-size: 16px;
      padding:2px; text-align:center;
}
#bild{
      display:none; margin: auto; margin-left: auto; margin-
      right: auto;
}
h4{
      text-align:center;
}
```

Und das wäre der JavaScript-Code der Datei *pg1_ready.js*:

```
var destinationType;
var bildBreite;
$(function() {
      $("#aufnahme").click(function() {
            navigator.camera.getPicture(onPhotoDataSuccess,
            onFail, {
                  quality : 50, destinationType :
                  destinationType.DATA_URL
            });
      });
      $("#loeschen").click(function() {
            $("#meldung").html('');
            $('#bild').css({ 'display' : "none" });
      });
      document.addEventListener('deviceready', function() {
            destinationType =
            navigator.camera.DestinationType; }, false);
});
function onPhotoDataSuccess(imageData) {
      bildBreite = (screen.width * 0.8) + "px";
      $('#bild').attr('src', "" +
      imageData);
      $('#bild').css({ 'display' : "block", 'width' : bildBreite,
      'margin' : 'auto' });
      $("#meldung").html('Aufnahme erfolgreich');
}
function onFail(message) {
      $("#meldung").html('Fehler bei der Aufnahme: ' + message);
```

```
        $('#bild').css({ 'display' : "none" });
}
```

Der entscheidende Part in dem Quellcode ist der Aufruf der Methode `getPicture()`. Diese wird beim Auslösen der entsprechenden Schaltfläche aufgerufen. Der erste Parameter ist der Callback auf die Funktion, die im Erfolgsfall aufgerufen werden soll. Deren Parameter verweist im Konzept von PhoneGap automatisch auf das aufgenommene Bild. In der Funktion wird das Attribut `src` des `img`-Tags der Webseite mit dem Wert belegt, wobei der genaue Typ der Daten mit dem vorangestellten MIME-Type explizit angegeben werden muss. Zudem wird der Anzeigebereich des Bildes in der Größe an die Bildschirmgröße angepasst und eine Erfolgsmeldung ausgegeben.

Der Callback für den Fehlerfall (der zweite Parameter der Methode `getPicture()`) zeigt nur eine Fehlermeldung an und blendet den `img`-Tag aus.

Der dritte Parameter von `getPicture()` legt die genauen Parameter der Aufnahmebedingungen über eine JSON-Notation fest. Von entscheidender Bedeutung ist die Festlegung der Quelle der Aufnahme. Diese wird bei der Initialisierung auf die Kamera des Geräts festgelegt.

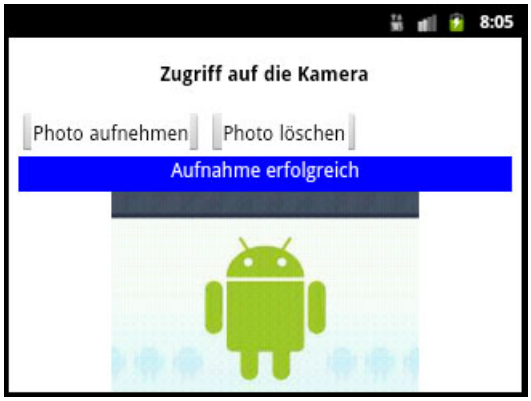

Abb. 4.5: Ein Photo wurde aufgenommen

Übung 4.3:

Musik abspielen – mit jQuery Mobile (Verzeichnis Übungen/Kapitel4/PG2)

In dieser Übung wollen wir Musik über PhoneGap abspielen und zudem jQuery Mobile für die Erstellung der GUI verwenden. Erstellen Sie wieder entsprechend der oben beschriebenen Vorgehensweise ein neues PhoneGap-Projekt *PG2* und passen Sie die dabei angelegte HTML-Datei *index.html* aus dem Verzeichnis *assets/www* wie folgt an:

```
<!DOCTYPE html>
<html lang="de">
      <head>
            <meta charset="utf-8" />
            <title>Musik mit PhoneGap</title>
```

© SGD, 13.09.18, Stojanovic, Milan (919400)


```
            <meta name="viewport" content="width=device-width,
            initial-scale=1" />
            <link rel="stylesheet" type="text/css" href="lib/css/
            jquery.mobile-1.1.0.min.css" />
            <link rel="stylesheet" href="lib/css/pg2.css"
            type="text/css"/>
            <script type="text/javascript" charset="utf-8"
            src="lib/js/cordova-1.7.0.js"></script>
            <script type="text/javascript" src="lib/js/jquery-
            1.7.2.min.js"></script>
            <script src="lib/js/jquery.mobile-1.1.0.min.js"
            type="text/javascript"></script>
            <script type="text/javascript" src="lib/js/
            pg2_ready.js"></script>
      </head>
      <body>
            <div data-role="page">
                  <div data-role="header"><h4>Musik abspielen</
                  h4></div>
                  <div data-role="content">
                        <div id="abspielen" data-
                        role="button"><img src="images/
                        speaker.png"
                              alt="Musik abspielen"/></div>
                  </div>
                  <div data-role="footer" id="meldung"></div>
            </div>
      </body>
</html>
```

Sie sehen hier wieder eine typische Grundstruktur von der HTML-Seite, wie man sie bei jQuery Mobile einsetzt. Die HTML-Datei enthält natürlich die Verweise auf die notwendigen CSS- und JavaScript-Ressourcen. Beachten Sie die Referenz auf jQuery Mobile. Die Schaltfläche zum Abspielen der Tondatei wird mit `data-role` festgelegt. Die referenzierte CSS-Datei *pg2.css* formatiert nur die Ausgabe und die Überschrift wie in pg1.css.

Werfen wir wieder einen genaueren Blick auf den JavaScript-Code der Datei *pg2_ready.js*:

```
var my_media = null;
$(function() {
      $("#abspielen").click(function() {
            playAudio("http://rjs.de/safety_first/
            safetyfirstdemo.mp3");
      });
      $("#abspielen").addClass('ui-disabled');
```

```
        document.addEventListener('deviceready', function() {
              $('#abspielen').removeClass('ui-disabled'); },
              false);
});
function playAudio(src) {
        my_media = new Media(src, onSuccess, onError);
        my_media.play();
        $("#abspielen").addClass('ui-disabled');
        $("#meldung").html("Audiodatei wird wiedergegeben");
}
function onSuccess() {
        $("#meldung").html("Audiodatei fertig abgespielt");
        $("#abspielen").removeClass('ui-disabled');
}
function onError(error) {
        $("#meldung").html('code: ' + error.code + ':' + 'Meldung:
        ' + error.message);
        $("#abspielen").removeClass('ui-disabled');
}
```

Die Stelle, über die das Abspielen der Musikdatei per PhoneGap ausgelöst wird, ist der Aufruf der Methode `play()` innerhalb der Funktion `playAudio()`, welche beim Auslösen der Schaltfläche aufgerufen wird. Diese Methode wird einfach der URL auf eine MP3-Datei auf einem Webserver als Parameter übergeben. Aus diesem URL wird mit einem PhoneGap-Konstruktor `Media()` ein Medienplayerobjekt erzeugt, dem ein Callback für den Erfolgs- und ein Callback für den Fehlerfall als zweiten und dritten Parameter zugeordnet werden.

Beachten Sie, dass wir die Schaltfläche zum Start des Abspielens mit der jQuery-Methode `addClass()` deaktivieren, wenn die Audiodatei wiedergegeben wird. Es wird einfach die jQuery-Standardklasse `ui-disabled` zugewiesen.[25] Und wenn die Wiedergabe beendet wurde oder ein Fehler eingetreten ist, wird die Schaltfläche wieder aktivierbar gemacht. Dazu wird die Klasse zum Deaktivieren einer Komponente in der Oberfläche einfach wieder mit `removeClass()` entfernt.

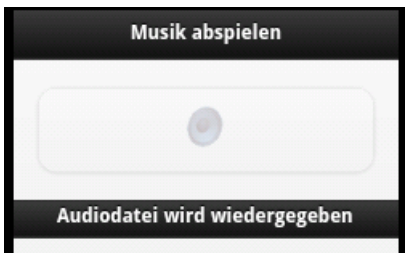

Abb. 4.6: Die Audiodatei wird abgespielt

25. Eine typische jQuery-Technik für so eine Situation.

© SGD, 13.09.18, Stojanovic, Milan (919400)

Zusammenfassung

Sie haben in diesem Kapitel die grundsätzliche Anwendung von PhoneGap gesehen, um Web-Apps auf einem mobilen Endgerät installieren zu können und zudem Zugriff auf wesentlichen Bestandteile der Hardware dieser Geräte zu erhalten. Dem etwas mühsamen Aufwand zum Einrichten eines solchen PhoneGap-Projekts steht der entscheidende Vorteil gegenüber, dass Sie dann mit konventionellen Web-Techniken inklusive Frameworks wie jQuery Mobile arbeiten können.

Aufgaben zur Selbstüberprüfung

4.1 Womit wird der Web-Code bei PhoneGap in die eigentliche App integriert?

_____

4.2 Was ist ein zentrales Argument für den Einsatz von PhoneGap? Kreuzen Sie die richtige Antwort an:

- Apps, die mit PhoneGap und Web-Technologien erstellt werden, sind performanter als native Apps.
- PhoneGap gestattet es aus JavaScript heraus, spezielle Bestandteile der Hardware des mobilen Endgeräts wie Kameras oder Bewegungssensoren zu nutzen.
- PhoneGap-Anwendungen können identisch und ohne Anpassungen auf allen mobilen Plattformen ausgeführt werden.
- Mit PhoneGap schaffen Sie eine einheitliche Benutzerschnittstelle auf allen mobilen Plattformen.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

4.3 Wie können Sie PhoneGap in Ihren mobilen Web-Apps verwenden, nachdem Sie das Framework in Ihr Web-App-Projekt kopiert haben? Kreuzen Sie die richtige Antwort an:

- Sie binden die Referenz auf die enthaltene JavaScript-Bibliothek in der HTML-Datei der Web-App ein und können dann in allen eigenen JavaScripten die Features von PhoneGap verwenden.
- Sie binden die Referenz auf die enthaltene JavaScript-Bibliothek in einer der JavaScript-Dateien ein, die Sie zusätzlich in der Web-App einsetzen.
- Das Kopieren des Frameworks sorgt dafür, dass PhoneGap automatisch gefunden und bereitgestellt wird. Sie können die Features von PhoneGap damit direkt nutzen.

© SGD, 13.09.18, Stojanovic, Milan (919400)

A. Lösungen der Aufgaben zur Selbstüberprüfung

Vergleichen Sie bitte Ihre Lösungen! Bei freier Formulierung kommt es nicht auf eine wörtliche Übereinstimmung an, sondern auf eine inhaltliche. Sind Sie zu keinem Ergebnis gekommen, sollten Sie das entsprechende Kapitel dieses Studienheftes wiederholen und die zugehörige Aufgabe nach einer Pause noch einmal schriftlich lösen. Gegebenenfalls berichtigen Sie Ihr Ergebnis nach einem erneuten Versuch. Lassen Sie kein falsches Ergebnis stehen.

Kapitel 1

1.1 Richtig ist, wenn Sie die Angabe angekreuzt haben, dass Web-Apps nur auf wenigen mobilen Plattformen funktionieren.

1.2 Richtig ist, wenn Sie die Angabe BeOS angekreuzt haben.

1.3 Richtig ist, wenn Sie die Angabe Eclipse angekreuzt haben.

1.4 Richtig ist die Angabe, dass Web-Apps auf dem mobilen Endgerät installiert und mobile Web-Applikationen dort in einem bereits installierten Standardbrowser ausgeführt werden.

Kapitel 2

2.1 Richtig ist, wenn Sie die Angabe data-role angekreuzt haben.

2.2 Content Distribution Network oder auch Content Delivery Network?

2.3 Richtig ist die Angabe, dass zuerst die Referenzen auf die CSS-Ressourcen und dann die Referenzen auf die Skripte zu notieren sind.

2.4 Sie gehen damit bei einem sensitiven Element wie einem Link in der History zurück.

Kapitel 3

3.1 Richtig ist die Angabe data-transition.

3.2 Sie können damit grafische Symbole Elementen wie Buttons hinzufügen.

3.3 Mit der Attributangabe data-inline="true".

3.4 Richtig ist die Angabe, dass Sie damit Schaltflächen visuell zu Gruppen zusammenfassen können.

3.5 ThemeRoller

Kapitel 4

4.1 Mit einem Wrapper.

4.2 Richtig ist die Angabe, dass Sie mit PhoneGap aus JavaScript heraus spezielle Bestandteile der Hardware des mobilen Endgeräts wie Kameras oder Bewegungssensoren nutzen können.

- PhoneGap-Anwendungen können identisch und ohne Anpassungen auf allen mobilen Plattformen ausgeführt werden.
- Mit PhoneGap schaffen Sie eine einheitliche Benutzerschnittstelle auf allen mobilen Plattformen.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

4.3 Richtig ist die Angabe, dass Sie die Referenz auf die enthaltene JavaScript-Bibliothek in der HTML-Datei der Web-App einbinden und dann in allen eigenen JavaScripten die Features von PhoneGap verwenden können.

© SGD, 13.09.18, Stojanovic, Milan (919400)

B. Glossar

API	Application Programming Interface.
Emulator	Ein Programm, das ein anderes in bestimmten Teilaspekten nachbildet.
IDE	Integrated Development Environment.
Java	Eine Programmiersprache, die meist als Basis für native Android-Apps verwendet wird.
JDK	Java Development Kit.
JRE	Java Runtime Environment.
JVM	Java Virtual Machine.
Midlets	Java-Apps.
Prinzip der Fehlertoleranz	Ein Verfahren, damit Browser qualifiziert mit unbekannten oder unklaren Situationen umgehen können.
XHTML	Extensible Hypertext Markup Language – strenges HTML nach XML-Standard.

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

C. Literatur und Quellen

Bücher:

Lunny, A, (2011). *Phonegap Beginner's Guide.* Verlag: Packt Publishing, ISBN-10: 1849515360 ISBN-13: 978-1849515368

Reid, J. (2011). *Jquery Mobile.* Verlag: O'Reilly & Assoc., Inc. ISBN-10: 1449306683 ISBN-13: 978-1449306687

Steyer, R. (2011). *jQuery – Das JavaScript-Framework für interaktives Design.* Verlag: Addison-Wesley, ISBN: 978-3-8273-3072-7 496 Seiten- 1 CD, 1-farbig

Steyer, R. (2011). *Erfolgreich JavaScript lernen.* Verlag: Addison-Wesley, ISBN: 978-3-8273-3103-8 544 Seiten- 1 Website, 1-farbig

Internet:

aptana.com/	Aptana
de.selfhtml.org/	Das Standardwerk zum Nachschlagen von allen Anweisungen rund um HTML, CSS und JavaScript.
developer.android.com/sdk/eclipse-adt.html	Das ADT-PlugIn für Eclipse von Google
developer.android.com/sdk/index.html	Das Android SDK
jquery.com/	Die Webseite von jQuery.
jquerymobile.com/	Die Webseite von jQuery Mobile
jqueryui.com	Die Webseite von jQuery UI.
msdn.microsoft.com/de-de/windowsphone/	Das Developer Center von Microsoft
phonegap.com/	Die Webseite vom PhoneGap-Projekt
validator.w3.org/	Der Validator des W3C
www.eclipse.org	Die Seite vom Eclipse-Projekt
www.oracle.com/technetwork/java/javase	Die Java-Seite von Oracle
www.w3c.org	Das W3C

© SGD, 13.09.18, Stojanovic, Milan (919400)

D. Abbildungsverzeichnis

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

E. Sachwortverzeichnis

© SGD, 13.09.18, Stojanovic, Milan (919400)

sgd DEUTSCHLANDS FÜHRENDE FERNSCHULE

© SGD, 13.09.18, Stojanovic, Milan (919400)

© SGD, 13.09.18, Stojanovic, Milan (919400)

F. Einsendeaufgabe

Name:	Vorname:
Postleitzahl und Ort:	Straße:
Studien- bzw. Vertrags-Nr.:	Lehrgangs-Nr.:

Code: **WEBD07-XX1-K01**
Fernlehrer/in:
Datum:
Note:
Unterschrift Fernlehrer/in:

Bitte reichen Sie Ihre Lösungen per E-Mail über die Online-Lernplattform ein oder schicken Sie sie uns zusammen mit der Aufgabenstellung per Post.

1. **Multiple Choice**

   Beantworten Sie die folgenden 15 Fragen durch Ankreuzen der richtigen Antwort. Zu jeder Frage gibt es nur *eine* richtige Antwort. Jede korrekt gelöste Aufgabe wird mit 4 Punkten bewertet. Sie können also maximal 60 Punkte erreichen.

   1. Was ist **kein** Attribut, mit dem jQuery Mobile bestimmte Standardverhaltensweisen regelt?

      a) ☐ data-role

      b) ☐ data-rel

      c) ☐ data-inline

      d) ☐ data-form

      e) ☐ data-transition

   2. Was ist kein gültiger Wert für die Bezeichnung eines Übergangs in jQuery Mobile (bis Version 1.7)?

      a) ☐ turn

      b) ☐ flow

      c) ☐ slide

      d) ☐ pop

      e) ☐ fade

      f) ☐ flop

      g) ☐ flip

3. Wofür ist API in unserem Zusammenhang die Abkürzung?
   a) ☐ Application Process Interface
   b) ☐ Application Programming Interface
   c) ☐ Analog Programming Interface
   d) ☐ Application Programming Instrument
4. Wie nennt man ein Programm, in dem eine andere Laufzeitumgebung simuliert wird?
   a) ☐ Emulator
   b) ☐ Virtulator
   c) ☐ Analogator
   d) ☐ Application-Tester
   e) ☐ Client-Interface
5. Für was ist IDE in userem Zusammenhang die Abkürzung?
   a) ☐ Integrated Deployment Environment
   b) ☐ Include Development Environment
   c) ☐ Import Development Environment
   d) ☐ Include Deployment Environment
   e) ☐ Integrated Development Environment
6. Auf welchem Betriebssystem basiert Android?
   a) ☐ Linux
   b) ☐ Windows
   c) ☐ MacOS
   d) ☐ BeOS
   e) ☐ Solaris
   f) ☐ DOS
7. Wie nennt Microsoft sein aktuelles Betriebssystem für mobile Geräte?
   a) ☐ Windows Mobile
   b) ☐ Windows Phone
   c) ☐ Windows for Smartphones
   d) ☐ Windows-2-Go

8. Wie wird die Version der jQuery-Mobile-Bibliothek gekennzeichnet, die für die Produktion bzw. Verwendung im eigentlichen Betrieb gedacht ist?

   a) ☐ min

   b) ☐ prod

   c) ☐ production

   d) ☐ run

   e) ☐ final

9. Wie wird in jQuery Mobile eine Seite im Sinn des Frameworks gekennzeichnet?

   a) ☐ Mit data-role="page"

   b) ☐ Mit data-role="side"

   c) ☐ Mit data-rel="page"

   d) ☐ Mit data-icon="page"

10. Wie nennt Microsoft sein aktuelles Oberflächendesign für mobile Geräte und Windows 8?

    a) ☐ Metro

    b) ☐ Glass

    c) ☐ Web-Design

    d) ☐ Unity

11. Mit welcher Sprache werden unter Android native Apps überwiegend geschrieben?

    a) ☐ Basic

    b) ☐ JavaScript

    c) ☐ C++

    d) ☐ Java

    e) ☐ PHP

12. Womit ordnen Sie in jQuery Mobile einem Element ein spezifisches CSS-Thema des Frameworks zu?

    a) ☐ Mit dem Attribut data-theme

    b) ☐ Mit dem Attribut style

    c) ☐ Mit dem Attribut class

    d) ☐ Mit dem Attribut data-rel

13. Wie erzeugen Sie in PhoneGap ein Multimediaplayerobjekt?

    a) ☐ Mit dem Konstruktor der Klasse Media

    b) ☐ Mit new Audio().

    c) ☐ Mit der Funktion getMedia()

    d) ☐ Mit einem Callback auf die Klasse Media

14. Wie stellt PhoneGap den Zugang zur Kamera eines Geräts zur Verfügung?

    a) ☐ Mit new Camera()

    b) ☐ Das DOM-Objekt navigator wird um ein Unterobjekt camera erweitert.

    c) ☐ Das DOM-Objekt window wird um ein Unterobjekt camera erweitert.

    d) ☐ Mit einem Callback auf die Klasse Camera

15. In welchem Verzeichnis befinden sich bei PhoneGap in der Regel die Web-Ressourcen?

    a) ☐ phonegap

    b) ☐ www

    c) ☐ include

    d) ☐ lib

2. **Praxisaufgabe**

   Erstellen Sie die folgende Webseite und schicken Sie die Dateien bitte ein. Führen Sie die folgenden 10 Schritte praktisch aus. Jeder Einzelschritt wird mit maximal 4 Punkten bewertet. Sie können also maximal 40 Punkte erreichen.

   - Erstellen Sie eine Basis-HTML-Seite für jQuery Mobile mit Namen *jqmobile.html*, in der Sie die jQuery-JavaScript-Bibliothek sowie die CSS- und JavaScript-Dateien von jQuery-Mobile referenzieren. Verwenden Sie dazu Ressourcen auf einem CDN.
   - Erstellen Sie in der Webseite zwei Seiten im Sinn von jQuery Mobile. Verwenden Sie bei beiden Seiten das Theme e.
   - Geben Sie der zweiten Seite die Id s2.
   - Fügen Sie dem Kopfbereich der ersten Seite eine charakteristische Überschrift der Ordnung h1 hinzu (<h1>Meine mobile Seite</h1>).
   - Fügen Sie dem Körper der ersten Seite (Inhaltsbereich) ein Bild hinzu. Sie können ein lokales Bild oder einen absoluten URL wie *http://rjs.de/bilder/devil.gif* verwenden.
   - Fügen Sie in einer neuen Zeile im Inhaltsbereich der ersten Seite eine Schaltfläche hinzu, mit der Sie auf die zweite Seite kommen. Öffnen Sie die zweite Seite als Dialog. Legen Sie den Übergang als flip fest.
   - Notieren Sie im Fußbereich der ersten Seite den reinen Textinhalt *Praxisaufgabe*.
   - Fügen Sie dem Kopfbereich der zweiten Seite eine charakteristische Überschrift der Ordnung h1 (<h1>Dialog</h1>) zu.

- Fügen Sie im Inhaltsbereich der zweiten Seite eine Schaltfläche hinzu, mit der Sie auf die erste Seite zurückkommen. Legen Sie den Übergang als slide fest. Als Theme für die Schaltfläche verwenden Sie a.
- Notieren Sie im Fußbereich der zweiten Seite den reinen Textinhalt *Praxisaufgabe*. Als Theme verwenden Sie für den Fußbereich b.

© SGD, 13.09.18, Stojanovic, Milan (919400)