

Zürcher Hochschule für angewandte Wissenschaften

Diplomstudium Informatik



Seminararbeit 5. Semester

Entwickeln von Anwendungen für Handheld B

Evaluation von 10 Frameworks für Touch Devices

Autoren

Oliver Aeschbacher

René Kamer

Dozent

Christian Vils

Projektstart

4. Oktober 2012

Projektpräsentation

16. Januar 2013

Inhaltsverzeichnis

1. Aufgabenstellung.....	6
1.1. Ausgangslage	6
1.2. Ziele der Arbeit	6
1.3. Erwartetes Resultat	7
1.4. Motivation	7
2. Planung und Termine	8
2.1. Projektplan	8
2.2. Termine	8
2.3. Soll-/Ist-Vergleich	9
2.4. Fazit der Zeitplanung	10
3. Einleitung.....	11
3.1. Vorgehen	11
3.1.1. Vorgehen: Evaluation der Frameworks	11
3.2. Aufteilen der Aufgaben	14
3.2.1. Aufteilung der Evaluation.....	14
3.3. Umgebung der Evaluation.....	14
4. Evaluation der Frameworks.....	15
4.1. Appcelerator Titanium	15
4.1.1. Einleitung.....	15
4.1.2. Wie funktioniert das Framework	15
4.1.3. Evaluation	16
4.1.4. Finale Bewertung.....	18
4.2. Sproutcore Touch	19
4.2.1. Einleitung.....	19
4.2.2. Wie funktioniert das Framework	19
4.2.3. Evaluation	20
4.2.4. Finale Bewertung.....	22
4.3. PhoneGap	23
4.3.1. Einleitung.....	23
4.3.2. Wie funktioniert das Framework	23
4.3.3. Evaluation	23
4.3.4. Finale Bewertung.....	26
4.4. Motorola Rhodes.....	26
4.4.1. Einleitung.....	26

4.4.2.	Wie funktioniert das Framework	26
4.4.3.	Evaluation	27
4.4.4.	Finale Bewertung.....	29
4.5.	iUI	30
4.5.1.	Einleitung.....	30
4.5.2.	Wie funktioniert das Framework	30
4.5.3.	Evaluation	30
4.5.4.	Finale Bewertung.....	33
4.6.	Sencha Touch	33
4.6.1.	Einleitung.....	33
4.6.2.	Wie funktioniert das Framework	33
4.6.3.	Evaluation	34
4.6.4.	Finale Bewertung.....	36
4.7.	XUI	36
4.7.1.	Einleitung.....	36
4.7.2.	Wie funktioniert das Framework	37
4.7.3.	Evaluation	37
4.7.4.	Finale Bewertung.....	39
4.8.	iWebKit.....	39
4.8.1.	Einleitung.....	39
4.8.2.	Wie Funktioniert das Framework.....	39
4.8.3.	Evaluation	40
4.8.4.	Finale Bewertung.....	42
4.9.	jQuery Mobile.....	42
4.9.1.	Einleitung.....	42
4.9.2.	Wie funktioniert das Framework	43
4.9.3.	Evaluation	43
4.9.4.	Finale Bewertung.....	45
4.10.	jQPad	46
4.10.1.	Einleitung.....	46
4.10.2.	Wie funktioniert das Framework	46
4.10.3.	Evaluation	46
4.10.4.	Finale Bewertung.....	48
4.11.	Gegenüberstellung der Frameworks.....	49
4.12.	Erklärungen zu Tutorials und Verbreitung	49

4.13.	Fazit der Evaluation	50
5.	Projekt Lupen-App.....	51
5.1.	Umgebung des Projekts.....	51
5.1.1.	Wahl des Frameworks	51
5.1.2.	Installation der Frameworks.....	51
5.1.3.	Tools	52
5.2.	Entwicklung	52
5.2.1.	Einarbeitung	52
5.2.2.	Programmierung	53
5.2.2.1.	Layout	53
5.2.2.2.	Kamera-Ansteuerung	54
5.2.2.3.	Verwendung von loupe.js.....	55
5.2.2.4.	Verbesserungsmöglichkeiten	56
5.2.3.	App Debugging / Testing	56
5.2.4.	App Deployment.....	58
5.2.5.	Testgeräte.....	59
6.	Schlusswort	60
7.	Github Repository.....	60
8.	Quellen	61
9.	Abbildungsverzeichnis.....	63

1. Aufgabenstellung

1.1. Ausgangslage

Bereits die ersten Mobiltelefone enthielten kleine Anwendungen, wie Kalender, Taschenrechner oder Spiele. Diese waren jedoch vom Hersteller exakt für die jeweiligen Geräte konzipiert und fix installiert.

Das Erscheinen moderner Devices wie iPhone oder Android-Geräten erlaubt nun, dass man den Funktionsumfang der Geräte beinahe beliebig erweitern kann mit selbstgeschriebenen und/oder fremden Apps.

Aus der heutigen Zeit sind Touch-Devices nicht mehr wegzudenken. Handys wie das iPhone oder das Samsung Galaxy beziehungsweise Tablets wie das iPad oder das Asus Eee Pad bieten eine schier unendliche Menge an Applikationen. Um diese zu entwickeln bedarf es der entsprechenden Frameworks.

1.2. Ziele der Arbeit

Das Ziel dieses Projekts ist, auf der Basis von <http://woorkup.com/2010/08/25/10-useful-frameworks-to-develop-html-based-webapps-for-touch-devices/> eine Evaluation der 10 dort vorgestellten Frameworks zu vollziehen auf theoretischer Basis:

- Einfachheit
- Entwicklungsumgebung
- Support
- Tutorials
- Building
- Testing
- Verbreitung

und diese Erkenntnisse zur Verfügung zu stellen.

Auf der Basis dieser Evaluation werden 1 - 2 dieser Frameworks herangezogen, um eine kleine Lupen-Anwendung zu schreiben für Android-Telefone und darin die Erkenntnisse der Evaluation einfließen zu lassen.

Die Lupen-Anwendung soll eine Hilfe sein für Leute, welche nicht mehr so gut sehen können und mittels der Lupen-App kleine Schriften oder ähnliches wieder lesen können, welche zuvor von der Device-Kamera abfotografiert wurden (im Idealfall sogar eine Lupe in Realtime).

1.3. Erwartetes Resultat

Ein Dokument, welches die 10 Frameworks gegenüberstellt in den unter Aufgabenstellung genannten Punkten.

Des Weiteren eine App, welche unter den 1 - 2 verheissungsvollsten Frameworks geschrieben wurden. Dazu wird ein Teaser erstellt und der ganzen Klasse zur Verfügung gestellt. Ebenfalls wird eine Präsentation von ca. 40 - 60 Minuten gehalten vor der Klasse (Die Zeitdauer bestimmt sich dadurch, dass an diesem Projekt 2 Personen arbeiten)

1.4. Motivation

Schon länger beschäftigt uns das Thema Handhelds und die Programmierung dazu. Dieses Seminar gibt uns nun die Möglichkeit, dieses Wissen aufzubauen und zu festigen.

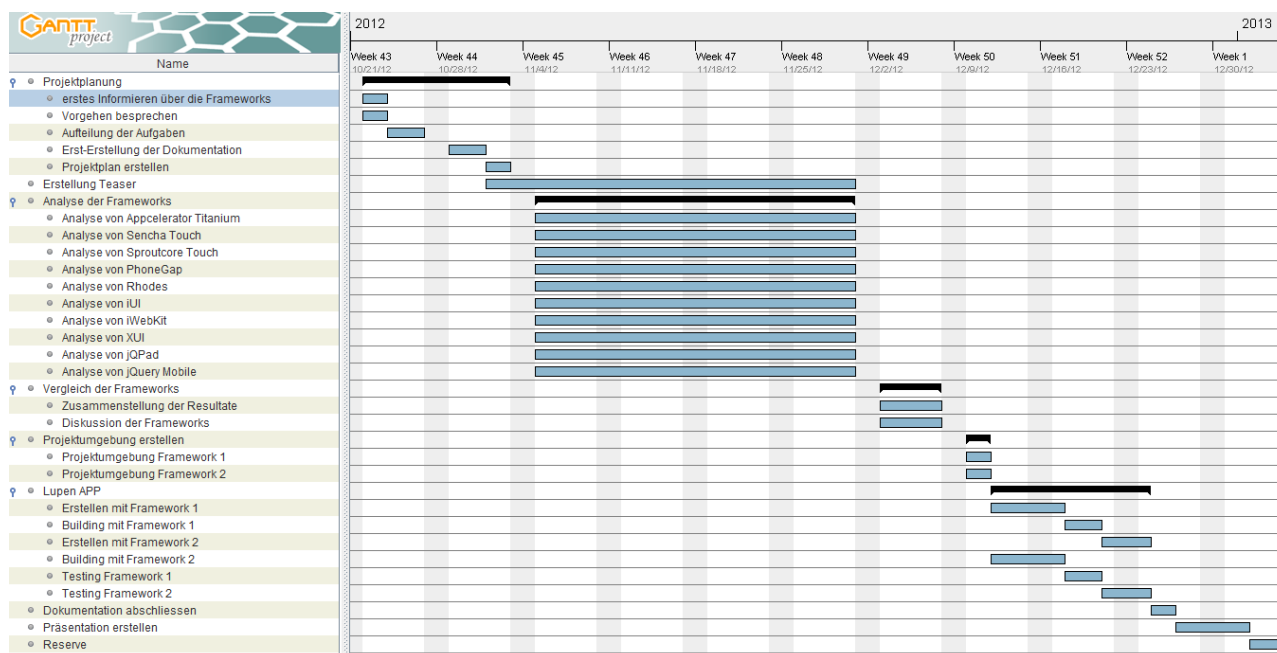
Im Speziellen interessieren uns Frameworks (nicht nur in Bezug auf Handhelds) und deren Möglichkeiten, unsere Arbeit zu vereinfachen und zu vereinheitlichen. Nicht zuletzt ist auch die Möglichkeit spannend, die unter den Frameworks geschriebenen Programme auf verschiedenen Plattformen laufen zu lassen, quasi 7 Fliegen mit einer Klatsche (iOS, Android, Blackberry, Windows Phone, Palm, WebOS, Bada, Symbian).

Daher haben wir uns entschieden, 10 dieser Frameworks einem Evaluationsprozess zu unterziehen und unter ausgewählten Frameworks eine kleine App zu realisieren.

Da leider unsere Kriegskasse ein wenig eingeschränkt ist, wird diese Realisierung ausschliesslich auf Android stattfinden.

2. Planung und Termine

2.1. Projektplan



Der Projektplan wurde mit GanttProject entwickelt. Leider hat die aktuelle Version einen Bug und somit können die Vorgänger sowie Nachfolger eines Arbeitszeitraumes nicht visuell dargestellt werden.

Der Projektplan in dieser Form zeigt an, in welchem Zeitraum der jeweilige Task zu erledigen ist, für die exakte Stunden-Aufteilung wird auf den Soll-/Ist-Vergleich verwiesen.

Da der Plan aufgrund seiner Grösse in diesem Dokument etwas schlecht lesbar ist, ist dieser noch als JPG im „Documents“ Ordner des Projekts abgelegt sowie auch das Gantt-File, welches mit GanttProject geöffnet werden kann.

2.2. Termine

Kick-Off: 19.09.2012

Projektfreigabe: 04.10.2012

Zwischenbesprechung: 12.12.2012

Präsentation: 16.01.2012

2.3. Soll-/Ist-Vergleich

Generell werden die Projekt-Mitarbeiter mit Kürzeln versehen, René Kamer mit RKA, Oliver Aeschbacher mit OLA.

Name	Zuständig	Soll	Ist
Projektplanung	-	11	10
Erstes Informieren über die Frameworks	RKA + OLA	2	2
Vorgehen besprechen	RKA + OLA	1	2
Aufteilung der Aufgaben	RKA + OLA	2	1
Erst-Erstellung der Dokumentation	RKA	3	2
Projektplan erstellen	RKA	3	3
Erstellung des Teasers	OLA	3	3
Analyse der Frameworks	-	70	82
Analyse von Appcelerator Titanium	RKA	7	9
Analyse von Sencha Touch	OLA	7	10
Analyse von Sproutcore Touch	RKA	7	9
Analyse von PhoneGap	RKA	7	8
Analyse von Rhodes	RKA	7	8
Analyse von iUi	RKA	7	6
Analyse von iWebKit	OLA	7	8
Analyse von XUI	OLA	7	9
Analyse von jQPad	OLA	7	6
Analyse von jQuery Mobile	OLA	7	9
Vergleich der Frameworks	-	6	7
Zusammenstellung der Resultate	RKA	2	2
Diskussion der Frameworks (inkl Doku)	RKA + OLA	4	5
Projektumgebung erstellen	-	8	3
Projektumgebung Framework 1	RKA + OLA	4	3
Projektumgebung Framework 2	-	4	1
Lupen APP	-	30	20
Erstellen mit Framework 1	RKA + OLA	8	11
Building mit Framework 1	RKA + OLA	3	4
Testing mit Framework 1	RKA + OLA	4	6
Erstellen mit Framework 2	-	8	1
Building mit Framework 2	-	3	1
Testing mit Framework 2	-	4	1
Dokumentation abschliessen	RKA + OLA	4	7
Präsentation erstellen	RKA + OLA	10	10
Reserve		8	1
Total		150	143

Alle Einheiten sind in Stunden angegeben.

Anmerkung: In Reserve enthalten „Erstellen mit Framework1“: 1h

¹ Weshalb wir dies nicht durchgeführt haben, entnehmen Sie bitte Kapitel [Wahl des Frameworks].

2.4. Fazit der Zeitplanung

Im Grossen und Ganzen ist die Zeitplanung nicht schlecht ausgefallen, wie wir nach der Durchführung festgestellt haben. Wir haben den Aufwand für die Evaluation der Frameworks etwas unterschätzt.

Kleinere Frameworks haben noch in die vorgegebene Zeit gepasst, sobald es ein etwas grösseres Framework war hat man die vorgegebenen 7 Stunden schnell überschritten.

Dies haben wir aber im Verlauf des Lupen-Projekts wieder wettgemacht, dies auch dank der Planänderung (weiter unten beschrieben) welche vorgenommen wurde.

Die Reserve wurde leicht angekratzt mit einer Stunde, ansonsten sind wir mit 143 ausgeführten Stunden knapp unter den 150 Stunden, welche für das gesamte Projekt geplant waren.

3. Einleitung

Dieser Teil beschreibt, wie wir vorgegangen sind in der Projekt-Anfangsphase, um die detaillierte Vorgehensweise der Evaluation der Frameworks zu beschreiben. Hierbei beschreiben wir die Kriterien und wie sie auf die Evaluation angewendet werden sowie die Gewichtung derselbigen. Daraus resultierend werden die Aufgaben auf die jeweiligen Projektmitarbeiter aufgeteilt, was im zweiten Abschnitt dieses Kapitels dargestellt ist.

3.1. Vorgehen

3.1.1. Vorgehen: Evaluation der Frameworks

Die Frameworks werden jeweils durch dieselben 8 Kriterien bewertet: Einfachheit, Entwicklungsumgebung, Support, Dokumentation, Tutorials, Building, Testing und Verbreitung. Zum Schluss werden alle Frameworks einander gegenübergestellt in einer Entscheidungsmatrix, in welcher die jeweiligen Kriterien mit Zahlen von 1 - 10 bewertet werden und danach die Summe gebildet, um die 1 - 2 geeignetsten Frameworks zu ermitteln. Die Frameworks mit der höchsten Gesamtpunktzahl werden für die Lupen-App berücksichtigt.

Die Kriterien werden nachfolgend einzeln beschrieben betreffend der Bewertung:

- Einfachheit
 - Subjektives Empfinden: Die Einfachheit des eingesetzten Frameworks ist schwerlich objektiv zu beschreiben, da unterschiedliches Vorwissen oder generelle Präferenzen stark in die Bewertung mit einfließen können. Deshalb wird hier ausnahmsweise auch das subjektive Empfinden als Kriterium miteinbezogen. Dies ist aber das schwächere Kriterium als die Anzahl benötigter Technologien, um die nicht-technische Beschreibung des subjektiven Empfindens ein wenig abzuschwächen.
 - Anzahl benötigter Technologien: Man kann verschiedene Technologien einsetzen. Alle Frameworks basieren auf HTML, CSS und JavaScript, jedoch in unterschiedlicher Ausprägung. Weiters gibt es noch verschiedene zusätzliche Möglichkeiten wie z.B. XML im W3C Widget Specification Format oder ähnliches. Als ideal werden 2-3 Technologien angesehen, welche zum Schreiben einer App ausreichen. Zu wenige Möglichkeiten in der Basis sind schlecht sowie auch zu viele Technologien, welche in einer Basisanwendung eingesetzt werden müssen. Dem wird in diesem Kriterium Rechnung getragen.
 - Intuitive Benutzbarkeit: Werden die eingesetzten Technologien so eingesetzt, wie man sie aus dem Alltag kennt oder werden diese anders benutzt, umgeordnet, eventuell sogar eingeschränkt in der Benutzung? Ein hohes Mass an Normalgebrauch gibt eine schnelle Einarbeitungszeit, weshalb dies hoch zu gewichten ist.
- Entwicklungsumgebung

- Aufzählung: Eine Aufzählung der Entwicklungsumgebungen. Mehr Möglichkeiten bieten Entwicklergruppen grössere Flexibilität: Der Web-Programmierer schreibt lieber mit einem Web-Programmier Tool, der alteingesessene Programmierer eher in einem Plaintext-Editor.
- Alternativen/Erweiterungen: Gibt es Alternativen (evtl. auch etwas exotische) welche unter dem Punkt „Aufzählung“ beschriebenen Tools ergänzen/verbessern?
- Hilfestellungen durch Framework: Wie weit geben die Frameworks Hilfestellung, bieten z.B. Text-Completion, auf das Framework zugeschnittene Fertigbausteine oder Hinweise an? Je mehr Hilfestellungen existieren, desto besser wird dieser Punkt bewertet.
- Support
 - Supporthotline: Existiert eine Supporthotline? Wie schätzt man deren Qualität ein, ist diese in Landessprache verfügbar?
 - Online-Möglichkeiten: Gibt es weitere Möglichkeiten den Support zu erreichen? Z.B. Mail, Chat, Live-Support oder ähnliches? Gibt es ein Entwicklerforum oder eine Online-Gruppe, in welchen man sich austauschen kann? Je mehr Möglichkeiten existieren, umso besser wird hier bewertet.
 - Preis: Was sind die Kosten für den Support? Wie wird dies auf die Projektgrösse berechnet, auf die Anzahl Entwickler? Was ist das Preismodell? Das Preis/Leistungsverhältnis wird bewertet.
 - Anleitungen im Netz: Gibt es anständige Anleitungen, welche im Netz verfügbar sind? Sind sie vom Anbieter erstellt oder muss auf Dritt-Anleitungen zurückgegriffen werden? Je mehr Anleitungen von guter Qualität vorhanden sind, umso besser wird dieses Kriterium bewertet.
- Dokumentation
 - Verfügbarkeit: Wie ist die Verfügbarkeit der Dokumentation? Gibt es sie nur als Buch oder sind sie online verfügbar? Je weniger Aufwand betrieben werden muss, umso besser wird dieses Kriterium bewertet.
 - Struktur: Hat die Dokumentation eine Struktur oder ist es ein simples Textfile ohne Möglichkeiten zur Suche? Ist es einfach, etwas zu finden in der Dokumentation? Je einfacher etwas zu finden ist, umso besser wird hier bewertet.
 - Ausführlichkeit: Wie ausführlich ist die Dokumentation? Sobald man eine Ausführlichkeit erreicht, welche das Verständnis erleichtert, wird hier gut bewertet. Eine zu hohe Ausführlichkeit kann jedoch kontraproduktiv sein, was ebenfalls in diesen Punkt einfließt.
- Tutorials
 - Google Tutorial Suche: Hier wird eine simple Google-Suche gemacht der Form: „<Framework-Name>“ + „Tutorial“. Je mehr Treffer Google zurückgibt, umso mehr Punkte werden vergeben. 10 Punkte für die meisten Treffer, 1 Punkt für die wenigsten Treffer.
 - Qualität der ersten 5 Google-Hits: Es wird die Ausführlichkeit und Verständlichkeit der ersten 5 Google Treffer bewertet. Tutorials sollten einfach aufgebaut und simpel sein, jedoch trotzdem ein gewisses Mass an Komplexität aufweisen. Das richtige Mittelmaß sollte angestrebt werden und wird entsprechend bewertet.
- Building

- Wie einfach /aufwändig ist der Prozess: Ist das Building einfach zu machen? Muss man hohe Aufwände betreiben, um den Build zu erstellen? Je einfacher der Prozess umso besser wird bewertet.
- Verschiedene Möglichkeiten: Existieren verschiedene Möglichkeiten, den Build zu erstellen? Z.B. Lokal, Online, per Post?
- Build-Tool: Stellt der Framework-Anbieter Build-Tools zur Verfügung oder muss auf ein IDE oder ähnliches zurückgegriffen werden (Cross-Platform). Gibt es ein Build-Tool wird dies positiv bewertet.
- Testing
 - Wie kann getestet werden: Sind Testing-Tools oder Frameworks vorhanden? Kann anständig Debugging betrieben werden? Muss man auf Methoden wie Print-Outs zurückgreifen? Je mehr Möglichkeiten das Framework von sich aus bietet, desto besser wird hier bewertet.
 - Methodik: Sind die einsetzbaren Test-Möglichkeiten methodisch oder strukturiert? Konformität mit der gängigen Lehrmeinung wird hier hoch bewertet.
- Verbreitung
 - Google Suche: Hier wird eine simple Google-Suche gemacht mit dem Namen des Frameworks. Je mehr Treffer Google zurückgibt, umso mehr Punkte werden vergeben. 10 Punkte für die meisten Treffer, 1 Punkt für die wenigsten Treffer.
 - Anzahl Apps: Wie viele mit diesem Framework geschriebene Apps sind verfügbar. Je mehr Apps damit erstellt wurden, umso besser fällt die Punktzahl aus.
 - Persönliche Erfahrung: Hat man schon von diesem Framework gehört? Gibt es gute oder schlechte Kritiken, welche man schon gelesen hat? Eine Zusammenfassung der Eindrücke spiegelt sich in der vergebenen Punktzahl wieder.
- Spezielles: Gibt es noch irgendwelche speziell nennenswerte Features oder Vorteile, welche dieses Framework auszeichnen? Hiermit können maximal 2 Zusatzpunkte vergeben werden.

3.2. Aufteilen der Aufgaben

3.2.1. Aufteilung der Evaluation

Die Aufteilung erfolgt der Logik nach mit 50/50 aufgeteilt auf die Projektmitarbeiter. Dabei wurde der Grösse der jeweiligen Frameworks einigermaßen Rechnung getragen, dass die Arbeitslast in etwa gleich verteilt ist, wie in der Planung bereits berücksichtigt.

Analyse von Appcelerator Titanium	RKA
Analyse von Sproutcore Touch	RKA
Analyse von PhoneGap	RKA
Analyse von Rhodes	RKA
Analyse von iUi	RKA
Analyse von iWebKit	OLA
Analyse von Sencha Touch	OLA
Analyse von XUI	OLA
Analyse von jQPad	OLA
Analyse von jQuery Mobile	OLA

3.3. Umgebung der Evaluation

Für die Evaluation wurden jeweils die Notebooks von RKA und OLA genutzt. Dies sind handelsübliche Geräte, wie sie jedermann zu Hause stehen hat.

Für jedes Framework wurden jeweils die Umgebungen geschaffen, in denen sie benutzt werden. D.h. es wurde jeweils das entsprechende IDE installiert, gewisse Libraries heruntergeladen, eingesetzt und die jeweilige Umgebung auch kurz ausprobiert, um ein ungefähres Feeling für das Framework zu bekommen. Dies hatte den Vorteil, dass auch Code-Ausschnitte ausprobiert werden konnten und die Tutorials ebenfalls kurz getestet werden konnten.

Die Informationen wurden über handelsübliche Browser zusammengesucht (Firefox, Internet Explorer, Opera) und entsprechend dokumentiert mit Office und LibreOffice.

4. Evaluation der Frameworks

4.1. Appcelerator Titanium

4.1.1. Einleitung



Appcelerator Titanium ist eine Plattform, um mobile Apps, Tablet-Apps sowie Desktop Apps zu entwickeln. Entwickelt wurde es von Appcelerator und der Öffentlichkeit wurde es im Jahre 2008 präsentiert. Es unterstützt verschiedene Systeme: iPhone, iPad, Android Devices und neu werden dank der strategischen Partnerschaft mit RIM auch Blackberry Devices unterstützt. Es werden im Wesentlichen zwei Technologien eingesetzt: JavaScript sowie Titanium API. Oftmals wird Appcelerator Titanium als Cross-Compiler bezeichnet, dies stimmt jedoch nur begrenzt. Vielmehr wird aus dem JavaScript mit der Titanium API auf dem Gerät der gesamte Code interpretiert, daher wäre eine Interpretiersprache wohl die bessere Bezeichnung. Mittlerweile haben sich sehr viele Entwickler mit Appcelerator Titanium beschäftigt, soeben hat das Unternehmen die Grenze von 300'000 registrierten Entwicklern durchbrochen.

4.1.2. Wie funktioniert das Framework

Die Entwicklung innerhalb dieses Frameworks basiert hauptsächlich auf JavaScript, hinzu kommt noch die Titanium-Schnittstelle, über welche man Zugriff auf über 5'000 Schnittstellen der jeweiligen Hersteller haben soll. Es wird NICHT nativ entwickelt mit z.B. Objective C oder ähnlichem, sondern ausschliesslich in der genannten Umgebung JavaScript und Titanium API. Eine Kompilierung der jeweiligen Codeteile ist nicht notwendig. Die Files können, so wie sie sind, direkt auf das Device geladen werden und werden erst auf dem Gerät interpretiert. Dies gibt eine kleine Verzögerung beim Starten der Apps, welche sich jedoch Testberichten zufolge nicht all zu stark auswirkt.



4.1.3. Evaluation

Einfachheit: Hier liegt wohl der grösste Knackpunkt von Appcelerator Titanium: Es ist ziemlich undurchsichtig auf den ersten sowie auf den zweiten Blick. Der Einsteiger findet zwar viele Code-Snippets, doch die eingesetzten Beispiele setzen sofort auf einem sehr hohen Level an und zeigen eher Features auf als dass sie beim Einstieg helfen. Zwar beschränken die zwei eingesetzten Technologien JavaScript und Titanium API den Lernaufwand, doch die sehr komplexen Möglichkeiten überfordern ziemlich schnell, eine gründliche Einarbeitung ins Framework ist zwingend notwendig, um diesen Berg an Informationen einigermaßen überblicken zu können.

Die gänzliche Absenz von strukturierten Elementen wie z.B. bei HTML verwirren zu Anfang sehr stark, erinnern jedoch in ihrer Art der GUI-Programmierung mit Java. Erfahrene JavaScript Programmierer haben hier bestimmt einen klaren Vorteil, doch auch sie müssen sich noch mit der Titanium API beschäftigen. Jedoch ist positiv zu bewerten, dass mit relativ wenigen Technologien gearbeitet wird, man also nicht allzu stark links und rechts schauen muss, um eine Implementation zu erreichen. Weiters ist die Homepage sehr business-lastig aufgebaut, auf den ersten Blick ist nicht ersichtlich, wie man als Entwickler an die benötigte Information kommt. Es wird eher der Fokus auf Verkauf, Marketing, Möglichkeiten und Zahlen gelegt, hier Punkten andere Frameworks mit einer wesentlich entwickler-orientierteren Präsentation der Fakten.

Final lässt sich sagen, dass eine steile Lernkurve vonnöten ist, um sich schnell im Framework zurechtzufinden, dies schlägt sich auch in der Punktebewertung nieder, die eher mässig ist.

Entwicklungsumgebung: Die Entwicklungsumgebung ist fest vorgegeben, sie nennt sich Titanium Studio. Obwohl noch ein SDK herunterladbar ist, wird von diversen Seiten abgeraten mit normalen Editoren zu entwickeln, weil damit viele Möglichkeiten verloren gehen, unter anderem das direkte Ausführen in einem Emulator, welcher im Titanium Studio eingebettet ist.

Obwohl es aussieht wie ein Eclipse, auch in seiner Funktionalität stark daran angelehnt ist, kommt das IDE nicht von der Eclipse Foundation. Entwickelt wurde es aus dem Aptana Studio, welches von Appcelerator im Januar 2011 aufgekauft worden ist.

Titanium Studio bietet viele Annehmlichkeiten wie Code-Completion, Dive-In Code, Compiler Correction Marks, automatische Prüfung von Plausibilität. Manchmal sogar etwas zu viel, denn viele Funktionen werden mittels des Titanium API ein wenig versteckt, man möchte öfters viel granularer auf die jeweiligen Funktionen zugreifen.

Viele Entwickler raten von Titanium Studio ab, es wird oft verschrien als Prototyping-Sprache, die meisten der Apps würden nach dem Prototyping-Stadium nochmals in Native-Programmiersprachen geschrieben. Dies konnte aus Zeitmangel in der Evaluationsphase nicht ausprobiert und beurteilt werden.

Im Grossen und Ganzen überzeugen die Möglichkeiten der Entwicklungsumgebung, jedoch müssen kleine Abstriche gemacht werden wegen des Verdeckens der Möglichkeiten, der begrenzten Auswahl von Alternativen sowie dem Hinweis auf eine Prototyping Sprache.

Support: Wie viele andere hat auch Appcelerator einen Slogan für seinen Support: „Get Access to mobile Experts“. Es werden einige Kontaktmöglichkeiten vorgestellt: Chat, E-Mail und Telefon. Der Service umfasst auch mehrere gut klingende Möglichkeiten wie: Hilfe um Titanium schnell zum Laufen zu bringen, Fixes für die Applikationen zu entwickeln oder beim Launch der App zu helfen. Ziemlich schnell wird jedoch klar, dass der Support mehr Business-Orientiert angelegt ist als bei anderen Anbietern.

Es wird kein einziger Preis genannt bei den Supportangeboten, alles läuft über Offerten, somit kann man sich als Kunde nur ein Bild verschaffen, wenn man solche Offerten einholt, was ein klarer Minuspunkt ist.

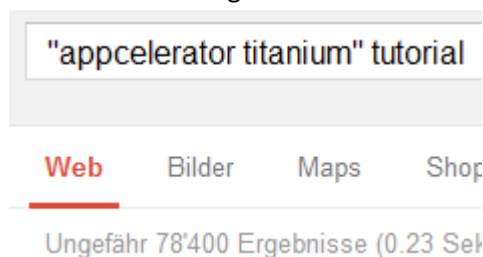
Eine positiv zu bewertende Funktion ist die FAQ-Sektion von Appcelerator, welche als offene Plattform angelegt ist und innert kurzer Zeit Antworten liefert. Der Rest des Supports kann nur schwer bewertet werden.

Dokumentation: Hier liegt die Stärke von Appcelerator Titanium, die Dokumentation ist sehr reichhaltig und äusserst umfangreich. Auf der Homepage von Appcelerator finden sich sehr viele Entwickler-Dokumentationen, an gewissen Stellen sogar übertrieben viele, dies ist aber der Komplexität des Frameworks zuzuschreiben. Ebenfalls ist die Dokumentation sehr gut in die Entwicklungsumgebung eingebettet und man kann sich auch einige Bücher zum Framework bestellen. Ebenfalls ist ein Wiki verfügbar, welches sich aber noch im Aufbau befindet.

Die Struktur der Dokumentation ist dabei sehr einfach, aber äusserst effizient gehalten. Man findet schnell, was man benötigt, hat aber an einigen Stellen einen gewissen Overhead, da zu viele Informationen in einen Menüpunkt integriert sind.

Abschliessend lässt sich sagen, dass die Dokumentation sehr gut gehalten ist, für Einsteiger jedoch zu massig und überladen, daher wird es hier leichte Abstriche in der Bewertung geben.

Tutorials: Die Google Suche fördert 78'400 Ergebnisse zu Tage:



Eine erschreckend niedrige Zahl für ein so weit verbreitetes Framework. Auch die Qualität der gefundenen Tutorials ist eher mager, die meisten beschränken sich darauf, Features von Appcelerator Titanium zu zeigen, indem man vorgefertigte Code-Stücke herunterlädt und ausprobiert.

Unter den ersten 5 Links, welche aus der Suche resultiert sind, findet sich als erstes das Appcelerator-Wiki, welches in etwa dasselbe abdeckt wie die Appcelerator Homepage selbst, das Wiki scheint sich noch im Aufbau zu befinden. Die restlichen Links sind eher Feature-Präsentationen von Appcelerator als Tutorials. Es werden oft Codestücke zum Download angeboten, welche man ausprobieren kann. Der Lerneffekt ist eher gering, wenn man sich jedoch schon ein wenig auskennt kann das durchaus nützlich sein. Die Absenz von guten Tutorials und deren geringe Zahl schlagen sich entsprechend in der Bewertung nieder.

Building: Building gibt es bei Appcelerator eigentlich nicht. Die Codestücke, welche mit Titanium Studio programmiert wurden, können direkt auf das Gerät geladen werden und werden dort interpretiert. Somit entfällt das Build / Compile des Codes, der in Titanium Studio eingebaute Live-Emulator gibt schon im Vorfeld einen guten Einblick, wie das Programm aussehen wird.

Dieses Vorgehen erzeugt aber auch relativ starke Probleme. Diverse Berichte zeigen, dass auf dem eingebetteten Emulator alles Problemlos läuft, sich aber auf dem Zielgerät Fehler zeigen, welche bei einem vorrangigen Build ausgeräumt würden.

Eine Bewertung ist hier schwierig, da es ja eigentlich kein Building gibt. Jedoch wird die Möglichkeit, die Scripts direkt auf das Mobile Gerät zu spielen, als positiv bewertet. Jedoch sind die Probleme, die dabei entstehen wiederum als negativ zu bewerten. Ein guter Mittelweg ist in der Bewertung das fairste.

Testing: Hier liegt wieder eine der Stärken von Appcelerator Titanium. Da eine Registrierung als Technology Partner möglich ist, haben diverse Firmen diese Möglichkeit wahrgenommen. Technology Partners können über den Appcelerator Marketplace Erweiterungen für Appcelerator Titanium programmieren. Dies schlägt sich in diversen Erweiterungen nieder, welche explizit für das Testing geschrieben wurden, es werden diverse Möglichkeiten abgedeckt: Debugging-Tools, Trace-Tools, Beta Testing Apps und vieles mehr.

Wie in diversen Marktplätzen tummeln sich hier gute wie weniger gute Tools, nach ein wenig Suchen hat man jedoch gute Tools zur Hand, welche Methodisch gut aufgebaut sind und Testing nach Schulbuch oder Industriestandard ermöglichen. Die Preise sind teilweise jedoch eher hoch. Alles in allem zeichnet sich jedoch für das Testing ein sehr gutes Bild.

Verbreitung: Die Google Suche nach „Appcelerator Titanium“ zeigt eine Anzahl von 621'000 Ergebnissen an, was doch ziemlich erstaunt. Denn die Hersteller-Homepage gibt eine Anzahl von 35'000 geschriebenen Apps an. Diese Zahl lässt sich jedoch schwer überprüfen. Wenn man jedoch einige Apps anschaut wird schnell klar, dass es einige Global Players hat, welche ihre Apps mit Appcelerator Titanium geschrieben haben, so zum Beispiel: Mitsubishi, ebay, Merck oder PayPal. Daher kann man von einer grossen Verbreitung ausgehen.

Auf Nachfrage hat der eine oder andere Kollege schon vom Framework gehört, jedoch nach kurzer Zeit wieder die Finger davon gelassen. Dies ist wohl hauptsächlich auf die unter „Einfachheit“ erwähnten Punkte sowie durch das unglückliche Design der Homepage zu erklären.

Spezielles: Einige Spezialfunktionen bzw. Services sind löblich zu erwähnen. So bietet Appcelerator zum Beispiel eigene Clouds in verschiedenen Ausbaustufen an. Diese werden z.B. benötigt für Push-Nachrichten oder Mobile Backend as a Service (MBaaS).

Der eigene Marktplatz für Entwickler ist ebenfalls löblich, vor allem, weil dort viele Tools gefunden werden können, welche das Entwickeln vereinfachen, z.B. Testing Tools, Analyse Tools (Verbreitung, Benutzung) und vieles mehr.

4.1.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	4
Entwicklungsumgebung	6
Support	4
Dokumentation	7
Tutorials	4
Building	5
Testing	9
Verbreitung	7
Spezielles	2
Total	48

4.2. Sproutcore Touch

4.2.1. Einleitung



Sproutcore Touch ist ein Open-Source Java-Script Framework dessen Ziel es ist, Web-Applikationen zu entwickeln mit erweiterten Möglichkeiten der Hardwarebeschleunigung von Touch Devices. Wenn mit Sproutcore entwickelt wird, ist fast jeglicher Code in JavaScript gehalten. Die Unterstützung von Mobile Devices beschränkt sich hierbei grösstenteils auf Apple-Produkte. Dies auch darum, weil der Gründer von Sproutcore noch bis 2010 bei Apple gearbeitet hat (Gründung von Sproutcore im Jahr 2007) und diverse Apple-Entwickler im Rahmen der Web 2.0 Initiative stark bei Sproutcore mitgearbeitet haben. Apples Service MobileMe wurde sogar komplett mit Sproutcore entwickelt.

So erklärt sich auch das Look-and-Feel des Frameworks, welches sowohl auf Desktop-Browsern wie auch auf Mobile-Browsern stark an Apple-Produkte erinnert.

4.2.2. Wie funktioniert das Framework

Die Entwicklung in Sproutcore ist primär in JavaScript gehalten, es lassen sich aber prinzipiell sehr viele verschiedene Technologien mit einbinden. Die Entwicklung von Sproutcore kommt historisch von Ruby, dies merkt man auch beim Entwickeln mit Sproutcore stark.



Zu Beginn wird mittels wenigen Befehlen die Struktur und erste Dateien erzeugt, auf welchen man aufbaut. Diese werden dann immer stärker verknüpft und erweitert. Die Verknüpfung basiert dabei auf dem Model-View-Controller (MVC) Schema, was für ein Touch-Framework eher erstaunlich ist. Zum guten Schluss kann mittels einem lokalen Build eine schlanke, schnelle Applikation erstellt werden.



4.2.3. Evaluation

Einfachheit: Das Framework ist im Aufbau ziemlich simpel, der Schwierigkeitsgrad steigt jedoch ziemlich schnell. Es lässt sich aber, wenn man sich ans MVC-Paradigma gewohnt ist, relativ schnell erlernen. Die ersten einfachen Programme lassen sich einfach verstehen (z.B. mit den Beispielen auf der SproutCore Homepage) und kommen auch schnell zum Laufen. Dies auch dank der einfachen Schritt-für-Schritt Anweisungen auf der Sproutcore Homepage.

Die eingesetzten Technologien beschränken sich, zumindest in den ersten Beispielen, auf HTML, CSS und JavaScript. Damit lassen sich schon ziemlich gute Resultate erzielen. Will man jedoch für andere Produkte entwickeln als iPad und iPhone im Mobilbereich, dann muss man auch andere Technologien einsetzen (z.B. Objective-J, JSS), es bestehen ebenfalls Schnittstellen zu den bedeutendsten anderen Frameworks (Sencha Touch, PhoneGap, etc.).

Da die Entwicklung mittels MVC-Paradigma passiert, könnte man sagen, die Bedienung ist nicht intuitiv. Da sich aber dieses Paradigma immer grösserer Beliebtheit erfreut, wird es viele Leute geben, welche es durchaus als intuitiv bezeichnen würden. Jedoch ist die Entwicklung von HTML auf Basis von JavaScript doch schon etwas speziell, daher bezeichnen wir es als leicht unintuitiv.

Entwicklungsumgebung: Ein fest vorgegebenes IDE gibt es in diesem Sinne nicht bei Sproutcore, auch keine offizielle Empfehlung. Es gibt mittlerweile Greenhouse, ein IDE nur für Sproutcore, jedoch befindet sich dies noch im Alpha-Stadium, weshalb noch nicht sehr viele Entwickler damit arbeiten, obwohl es öffentlich ist.

Prinzipiell lassen sich alle Texteditoren verwenden, idealerweise natürlich ein Texteditor, welcher JavaScript unterstützt. Da es auch einige Rails-Komponenten in Sproutcore hat, kann es auch von Vorteil sein, ein IDE zu wählen, welches Rails unterstützt.

Populär sind NetBeans (es kann hier praktisch alles eingebunden werden), Rubymine (Eigentlich eher Rails/Ruby zentriert, versteht mittlerweile jedoch auch JavaScript) sowie WebStorm (JavaScript-Zentriert).

Da nichts vorgegeben ist, hat man hier ein freies Gefäss und kann sich sein IDE so zusammenstellen, wie man es möchte. Bei der Bewertung wird hier daher ein Mittelweg gewählt, jedoch ein wenig höher gewichtet wegen der guten Möglichkeiten der Auswahl.

Support: Bei Sproutcore findet man keinen klassischen Support. Weder per Telefon, noch per Mail. Es gibt jedoch Möglichkeiten, mit dem Sproutcore Team in Verbindung zu treten, dies sind: Eine Mailing List, ein Blog (mit Möglichkeit RSS), Twitter und Facebook.

Sproutcore betreibt auch User-Groups. Dies sind Gruppierungen an verschiedenen Orten, welche sich mehr oder weniger regelmässig treffen, um sich auszutauschen und ihre Erfahrungen abzugleichen. Sproutcore veranstaltet auch in unregelmässigen Abständen Events, bei welchen sie informieren, aber auch meist für Diskussionen bereit stehen. Speziell erwähnt sei das zweijährlich stattfindende „SproutCore WWDC Bash“, eine Veranstaltung, die einen ernsten Kern aber auch viele Spass-Events hat.

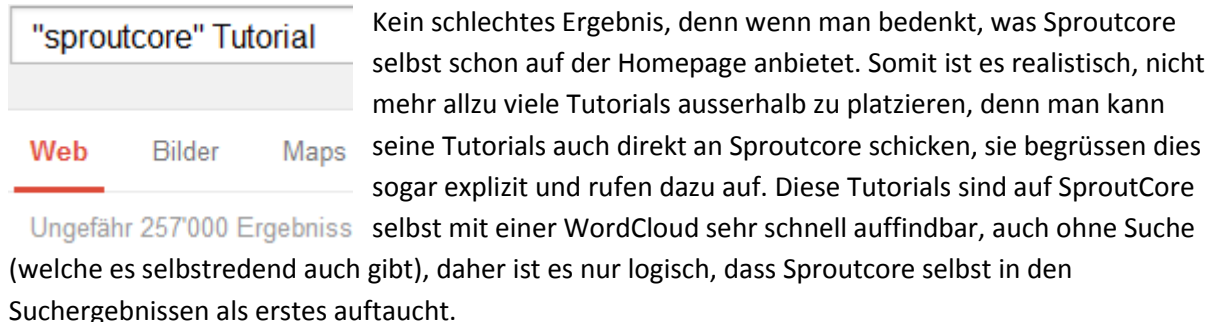
Alles in allem ist der Support wohl einer der Schwachpunkte von Sproutcore, was durch seine breite Userbasis mit all ihren Homepages ein wenig ausgeglichen wird. Dennoch können hier nur wenige Punkte vergeben werden.

Dokumentation: Die Dokumentation gehört zu den Stärken von Sproutcore. An mehreren Stellen online verfügbar bietet diese einen sehr sauber ausgeführten Service, um schnell an Informationen zu kommen. Dies zeigt sich in den Guides (Step-By-Step für Anfänger, weiterführende Tutorials für

Leute die schon weiter sind und grob verschachtelte Beispiele für Profis), bei der eigentlichen Funktionsdokumentation sowie bei der Hilfe, die beim Erzeugen des Framework-Stammes verfügbar ist. Der Aufbau ist immer sehr fein gegliedert und bietet für jedermann das richtige Niveau an, ohne einem jemals zu viel zuzumuten.

Durch die Summe der Möglichkeiten erreicht hier Sproutcore eine sehr gute Bewertung, ein wenig Abzug muss gemacht werden, weil man den Desktopteil von Sproutcore immer mitschleppt. Eine separate Dokumentation für die Touch-Funktionen wäre wünschenswert.

Tutorials: Die Google Suche fördert 257'000 Ergebnisse zu Tage:



Kein schlechtes Ergebnis, denn wenn man bedenkt, was Sproutcore selbst schon auf der Homepage anbietet. Somit ist es realistisch, nicht mehr allzu viele Tutorials ausserhalb zu platzieren, denn man kann seine Tutorials auch direkt an Sproutcore schicken, sie begrüßen dies sogar explizit und rufen dazu auf. Diese Tutorials sind auf SproutCore selbst mit einer WordCloud sehr schnell auffindbar, auch ohne Suche (welche es selbstredend auch gibt), daher ist es nur logisch, dass Sproutcore selbst in den Suchergebnissen als erstes auftaucht.

Der zweite Link verweist auf das Wiki von Sproutcore, welches den gewohnten Standard an Dokumentation und Tutorials bietet, jedoch in einer Wiki-Struktur eingebettet. Somit kann entschieden werden, welches einem besser liegt: Wiki oder normale Web-Struktur.

Der dritte Link behandelt ein vieldiskutiertes Thema in einem Tutorial, den Umgang mit SC.tableView. Dieses Tutorial ist äusserst ausführlich gehalten, was ein grosses Plus ist. Ebenfalls sind alle Code-Snippets einzeln herunterladbar. Äusserst gelungen.

Der vierte Link gibt in mehreren Video-Tutorials verschiedene Funktionalitäten zu sehen, dies ebenfalls im gewohnt aufgeräumten Look und Code.

Der fünfte Link stammt von BroadcastingAdam, wer sich ein wenig in der Web-Entwickler-Szene auskennt, kennt auch diese Page und weiss um ihre grossen Bemühungen, gute Tutorials zu bieten. Alles in allem sind alle betrachteten Tutorials-Pages von exzellenter Qualität, nicht zu viel, nicht zu wenig, (meist) mit Abstufungen des Schwierigkeitsgrades und guten Beispielen.

Building: Da es sich bei Sproutcore nicht um ein Building für eine App handelt, sondern ein Building für eine Web-Page, muss hier ein Abzug in der Bewertung gemacht werden. Nichtsdestotrotz ist der Building-Prozess sehr eindrücklich: Es wird vom vorgefertigten Code ein Build erzeugt mit einem einfachen Befehl. Dieser löst eine ganze Reihe von Aktionen aus: Der gesamte JavaScript-Code wird in ein einziges File geschrieben (dies erhöht die Geschwindigkeit der Page enorm), ebenso werden andere Sources sauber zusammengefasst. Der Clou ist jedoch, dass Bilder automatisch auf die richtige Grösse zugeschnitten werden und auch für die Verwendung im Web optimiert werden (Auflösung, etc.). Der Build, der dabei erzeugt wird, unterstützt praktisch alle nativen Funktionen eines iPads oder iPhones.

Im Grossen eine tolle Methode, einen Build zu erzeugen, jedoch schade, dass dies nicht in eine fertige App passiert, welche auf verschiedenen Systemen läuft und über einen Store bezogen werden kann. Dies schlägt sich entsprechend in der Bewertung nieder.

Testing: Sproutcore ist eines der wenigen Frameworks, welches starken Wert auf Testing legt. Es steht ein komplettes Unit-Testing Framework zur Verfügung, um den Code auf Herz und Nieren zu

überprüfen. Die gelungenen Anleitungen dazu sind auch für Anfänger sehr gut zu verstehen und könnten, ohne Veränderungen, im Schulunterricht eingesetzt werden. Dies liegt einerseits an ihrer Verständlichkeit und auch an deren Praxisorientierung, jeder kann sofort umsetzen, was in den Anleitungen steht. Ebenso legen sie starken Wert auf Test-Driven-Development, dieser Methode wird ebenfalls eine komplette Anleitung gewidmet.

Allgemein ist das Testframework sehr aufgeräumt und stellt die Ergebnisse der Tests sauber dar in verschiedenen Tiefen und Ausprägungen. Die Methodik verläuft dabei fast schulbuchmässig und die Struktur ist jederzeit gut eingehalten und gut verständlich.

Dieser Punkt wird sehr gut bewertet, da die Ausführlichkeit, Handhabung und Dokumentation exzellent gehalten sind.

Verbreitung: Die Google-Suche nach „Sproutcore“ zeigt eine Anzahl von 481'000 Ergebnissen an, dies wohl ebenfalls, da Sproutcore schon sehr vieles unter ihrem eigenen Dach vereint.

Eine Anzahl der Apps, die damit geschrieben wurden liess sich leider nicht in Erfahrung bringen, denn diese sind im Web nicht unbedingt als solche zu erkennen und auch Sproutcore selbst stellt hier keine Zahlen zur Verfügung

Jedoch stolpert man immer wieder über dieses Framework, sei es, wenn man im Internet zu Web-Entwicklung googelt, sich mit Freunden oder Arbeitskollegen unterhält oder in einschlägigen Publikationen. Dies kann auch damit zu tun haben, dass dieses Framework auch schon eher zu den älteren Hasen in diesem Business zählen.

Spezielles: Ein spezieller Build für schlanke Page-Strukturen (näher beschrieben unter „Building“) wird angeboten. Da dies jedoch nicht in ein App gewandelt werden kann, werden hier keine Punkte vergeben.

4.2.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	7
Entwicklungsumgebung	6
Support	4
Dokumentation	9
Tutorials	6
Building	5
Testing	9
Verbreitung	4
Spezielles	0
Total	50

4.3. PhoneGap

4.3.1. Einleitung



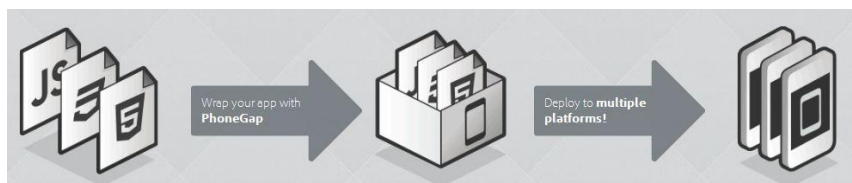
PhoneGap ist ein Open Source Framework, um in kurzer Zeit Cross-Plattform Mobile Apps zu erstellen mit HTML5, Javascript und CSS. Es können damit Apps für iPhone, Android, Windows Mobile, Blackberry, WebOS, Symbian, Tizen und Bada erstellt werden.

Es werden allgemein Standard-Web-Technologien eingesetzt um die Grenze zwischen Web-Basierter Technologie und Mobile Devices immer stärker zu verwischen. Das Framework wurde schon über eine Million Mal heruntergeladen und zählt mittlerweile eine Community von 400'000 Entwicklern. Das Framework benutzt die Technologie Apache Cordova, welche unter der Ägide der Apache Software Foundation mitentwickelt wurde. Dadurch kann gesagt werden, dass PhoneGap auch in Zukunft Open Source bleiben wird. Mittlerweile wurde PhoneGap von Adobe aufgekauft und weiterentwickelt

4.3.2. Wie funktioniert das Framework

Die Entwicklung von PhoneGap Apps folgt dem klassischen Web-Entwicklungs-Ansatz. Es wird ein HTML erstellt, welches mittels CSS formatiert wird, danach werden spezifische JavaScript Tags benutzt um die Gerätefunktionen anzusteuern. Es wird NICHT nativ entwickelt mit z.B. Objective C für iOS, es können auch keine Zusatzfunktionen damit erreicht werden.

Diese Skripts werden danach kompiliert, hier gibt es zwei Möglichkeiten: Ein lokales kompilieren auf dem eigenen Rechner (jeweils für 1 Framework) oder das Kompilieren auf build.PhoneGap.com (Kompilieren auf mehrere Frameworks gleichzeitig). Was herauskommt sind binary Files, welche mit wenigen Abwandlungen direkt in den jeweiligen Store (iTunes Store, etc.) geladen werden können.



4.3.3. Evaluation

Einfachheit: Die sehr guten und ausführlichen Beispiele geben schnell einen Eindruck, wie das Framework funktioniert, sehen einfach aus in der Anwendung und es ist schnell ersichtlich, dass mit wenigen Zeilen JavaScript eine hohe Funktionsdichte geboten wird.

Die Technologie wird eingesetzt wie man es aus HTML, CSS und JavaScript kennt, jeder Programmierer, welcher jemals eine Homepage selbst geschrieben hat, wird sich innert kürzester Zeit zurechtfinden. Die Schichtung der drei Haupttechnologien ist genauso, wie man sie tagtäglich einsetzt beim Programmieren einer Homepage, jedoch ist noch ein zusätzliches JavaScript

einzubinden, um die Funktionalität der Cordova Schnittstelle vollständig ausschöpfen zu können. Die Basis der Entwicklung sind HTML, CSS und JavaScript. Mehr braucht es nicht, um eine App erstellen zu können. Es bieten sich für Fortgeschrittene jedoch noch mehr Möglichkeiten wie z.B. das Anlegen eines config.xml zur Angabe von Meta-Daten, einbinden eigener JavaScripts sowie WebView Embedding für ausgewählte Touch-Devices. In der Summe ist hier von einem einfachen und gut verständlichen Framework zu sprechen.

Entwicklungsumgebung: Als Entwicklungsumgebung wird von PhoneGap Eclipse angegeben, welches mit dem jeweiligen SDK (für die jeweiligen Betriebssystemumgebungen wie Android oder iOS) und Cordova erweitert wird. Es sind aber auch schon Plug-Ins für Dreamweaver entwickelt worden. Prinzipiell reicht ein Text-Editor, um die Anwendungen zu entwickeln, da ein Online-Build-Tool verwendet werden kann für die Kompilierung. Es wird den Entwicklern also viel Freiheit gelassen, wie eine App entwickelt werden kann.

Das empfohlene IDE, Eclipse, kennt fast jeder Entwickler und kennt seine Stärken wie: Text-Completion, Korrekturvorschläge, Fertigbausteine, Code-Highlighting, Containering, Variable-Lookup, Variablen-Unterstützung und vieles mehr. Die extrem einfache Erweiterbarkeit tut ihren Rest und daher eignet sich dieses Tool hervorragend, um zu entwickeln.

Erweiterungen sind, wie oben Erwähnt, für Dreamweaver vorhanden. Mit dieser Erweiterung lassen sich Cordova-Funktionen auch in Dreamweaver ansteuern und verwenden. Eine weitere Erweiterung bzw. abgeleitetes Framework stellt appMobi PhoneGap XDK dar, welches auf PhoneGap aufbaut, aber ein noch weiter gefasstes Framework mit diversen Erweiterungen und eigenem IDE darstellt. Die verschiedenen Möglichkeiten der Entwicklungsumgebung tragen zu einer guten Bewertung bei, einziger Wehrmutstropfen ist, dass es etwas kompliziert ist, bis man alles bereit hat bei der Installation der Umgebung.

Support: Unter dem Slogan „Get all the tools, help and training you need to build great PhoneGap apps“ stehen diverse Supportmöglichkeiten zur Verfügung, wie es unter dem Dach von Adobe zu vermuten war. Es stehen diverse Support-Packages zur Verfügung, vom einfachen Ein-Personen Entwickler (1 Person, Best Effort, 24.95\$) bis zum Enterprise-Kunden (X Personen, 24x7, 4h Reaktionszeit, ab 2000\$) können diese Support-Packages diverse Bedürfnisse abdecken.

Es stehen dabei diverse Möglichkeiten wie Bug-Fix-Patching, Knowledge-Base, Chats, privates Forum und noch vieles mehr bereit. Dies zeigt, dass für wenige Dollars ein immenses Wissen zur Verfügung gestellt wird, daher wird das Preis Leistungsverhältnis mit sehr gut bewertet.

Es gibt aber auch genügend Möglichkeiten, kostenlos mit den Entwicklern in Verbindung zu treten, so z.B. Google Groups, ein PhoneGap Forum und diverse andere Foren, worin sich die Entwickler von PhoneGap bewegen. Ein kurzer Test zeigt, dass man z.B. in den Google Groups meist innert 2 Stunden eine Antwort erhält. Auch dies zeigt eine vorbildliche Haltung gegenüber den Usern. Es werden eine Vielzahl von sauber ausgearbeiteten Anleitungen angeboten und auch gepflegt. Die Übersicht ist dabei stets gewährleistet und bietet grossen Komfort.

Dokumentation: Wie bereits unter dem Punkt „Support“ erwähnt steht eine perfekt gepflegte Dokumentation im Netz bereit, welche einfach navigierbar und gut durchdacht ist. Die Struktur ist nicht eindimensional gehalten, über viele Stellen gelangt man an denselben Ort. Die Dokumentationen sind allesamt Online oder Offline verfügbar und kosten nichts, auch hier wird der Open-Source Charakter des gesamten Frameworks hervorgehoben.

Die Dokumentationen enthalten viele Informationen, doch allesamt erscheinen nützlich und helfen

weiter. Falls man in die Tiefe gehen will, ist mit wenig ausprobieren und ein wenig Suche im Netz schnell das Richtige gefunden.

Tutorials: Die Google Suche fördert 1.09 Mio. Ergebnisse zu Tage:

"Phonegap" Tutorial

Web

Bilder

Maps

Ungefähr 1'090'000 Ergebnisse

Dies spricht für eine grosse Unterstützung durch Tutorials. Als erster Link taucht gleich das Developer-Portal von PhoneGap auf, welches wie oben erwähnt einige gute Tutorials bietet.

Auch die restlichen vier Links, welche ausprobiert wurden, bieten sehr gut ausgearbeitete Tutorials, der 5. Link ist sogar ein 10 Minütiges Video-Tutorial für Android-Entwicklung und wie man die Entwicklungsumgebung aufsetzt. Alles in allem ist die Qualität absolut erstaunlich. Die meisten führen von einfachen Aktionen auf kompliziertere Zusammenhänge ohne jemals den Schwierigkeitsgrad zu stark zu erhöhen.

Building: Die Builds sind mehr oder weniger einfach zu erstellen. Wählt man den lokalen Weg, kommt man nicht umhin, für jedes unterstützte Touch-Betriebssystem den Build im entsprechenden IDE zu machen. Wählt man jedoch den Weg über das Building Tool build.phonegap.com, ist dies sehr einfach und die Build-Engine erstellt für jedes Touch-Betriebssystem in einer guten Geschwindigkeit die entsprechenden Binary Files.

Vor allem die Möglichkeit, über das Online-Build Tool zu arbeiten, bietet extrem komfortable Möglichkeiten, es können nicht nur ZIP Files mit den entsprechenden HTML-Dateien hochgeladen werden, es bietet auch die Möglichkeit, ein Git-Repo direkt anzugeben. Das direkte Ausbringen der entsprechenden Files auf ein Testgerät rundet das üppige Angebot ab.

Testing: Auf den ersten Blick ist Testing eher schwierig. Auf den zweiten Blick jedoch sieht die Situation anders aus. PhoneGap stellt zwar von Hause aus keine Testing-Tools zur Verfügung, jedoch ist die Community hier schon wesentlich weiter. Diverse Entwickler haben Test-Tools entwickelt, welche sich einerseits direkt während der Entwicklung benutzen lassen (im IDE) oder auf dem Touch-Device selbst.

Das beste gefundene Tool, welches ein aussenstehender Entwickler erstellt hat, ist nun sogar von PhoneGap in ihr Portfolio übernommen worden. Erreichbar über <http://debug.phonegap.com/> bietet dieses eine gute Möglichkeit, über eingebettete Skriptteile die komplette App zu debuggen.

Verbreitung: Die Google Suche nach „PhoneGap“ zeigt eine Anzahl von 4.4 Mio Ergebnissen an. Die Anzahl der damit geschriebenen Apps vergrössert sich natürlich täglich, der aktuelle Stand weist eine Zahl von 1181 Apps auf der Feature-List von PhoneGap an, inoffiziell ist von über 30'000 Apps die Rede. Dabei sind auch sehr respektable Apps zu finden, wie z.B. der Logitech Squeezebox™ Controller, welcher eine komplette Steuerung einer Multi-Room-Media Lösung ermöglicht. Auch bei Nachfragen im Kollegenkreis kennt man PhoneGap, es wurde auch schon öfters bei Mobile-Konferenzen erwähnt, und dies nur lobend.

Spezielles: In diesem Framework ist so vieles gut gemacht, man sehe sich nur mal die Building-Homepage an, die Unterstützung durch die Entwicklercommunity (welche selbst noch Tools beisteuert), die Umsetzung des Konzepts, usw.

4.3.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	8
Entwicklungsumgebung	7
Support	9
Dokumentation	8
Tutorials	9
Building	10
Testing	6
Verbreitung	9
Spezielles	2
Total	68

4.4. Motorola Rhodes

4.4.1. Einleitung



Rhodes ist ein Open Source Framework, welches auf Ruby basiert, um schnell Apps für alle Smartphone-Betriebssysteme zu erstellen. Unterstützt werden iPhone, Android, RIM, Windows Mobile und Windows Phone 7. Mit Rhodes werden native Anwendungen erstellt, welche mit lokalen Daten sowie mit synchronisierten Daten gearbeitet werden kann. Es können direkt verschiedenste Funktionen der Geräte angesteuert werden, so z.B. GPS, PIM Kontakte, Kalender, Kamera, Push, Barcode, Bluetooth, NFC (Near Field Communication) und einiges mehr.

Es ist primär angedacht, um Enterprise-Anwendungen zu schreiben, hat sich aber mittlerweile auch im „Fun-Sektor“ durchgesetzt, wie einige Beispiele zeigen. Dies schlägt sich auch in der brutal grossen Dokumentation nieder, welche leider nicht besonders gut strukturiert ist.

4.4.2. Wie funktioniert das Framework

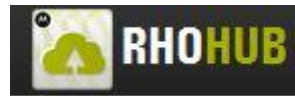
Die Entwicklung unter Rhodes funktioniert wie eine Ruby on Rails Applikation, jedoch mit sehr vielen unterstützenden Elementen. Es wird zuerst ein sogenanntes „Scaffold“ erzeugt, dies heisst, dass schon viele Files mit einem einzigen Befehl erzeugt werden und dann darauf aufgebaut wird (natürlich kann man den Befehl auch mehrmals benutzen). Die zu programmierenden Techniken umfassen HTML, Ruby und Javascript, es kann jedoch mit diversen zusätzlichen Techniken gearbeitet werden, so auch mit Native-C oder Java, die entsprechenden Schnittstellen sind vorhanden.



Rhodes folgt dem MVC-Paradigma (Model-View-Controller) und liegt damit stark im Trend, MVC-Programmiersprachen erfreuen sich wachsender Beliebtheit.

Die Builds können mit Rhodes lokal gemacht werden, jedoch gibt es auch den Service RhoHub, welcher stark an Github erinnert und auch genau gleich funktioniert. Jedoch gibt es einen grossen

Unterschied zu Github: Es können auf RhoHub direkt die entsprechenden Sources kompiliert und danach heruntergeladen werden, was die Cross-Plattform Entwicklung wesentlich vereinfacht.



4.4.3. Evaluation

Einfachheit: Das Framework hat eine sehr steile Lernkurve, schon in den ersten Tutorials und Beispielen wird ein Grundwissen, mindestens in Ruby, vorausgesetzt. Ebenfalls kann das MVC-Paradigma als vorausgesetzt gelten, denn eine Anleitung, wie so etwas benutzt wird, gibt es nirgends. Die Komplexität des Frameworks schlägt sich 1:1 in der eher schwierig zu erlernenden Struktur nieder. Hinzu kommt, dass die Homepage keine zentrale Anlaufstelle bietet, man findet das, was man sucht, eher schlecht. Die Struktur der Rhodes Homepage ist in die Struktur der Motorola Homepage eingebettet, was ein Grund für die schlechte Auffindbarkeit der Informationen ist.

Primär lassen sich mit HTML, CSS, JavaScript und Ruby schon ansehnliche Applikationen bauen, jedoch behelfen sich viele mit zusätzlichen Technologien wie Native-C, um richtig saubere Apps schreiben zu können. Es muss dabei jedoch nach einem genau vorgegebenen Schema vorgegangen werden, ansonsten kann der Build nicht erstellt werden und bricht mit relativ undifferenzierten Fehlermeldungen ab.

Die Benutzbarkeit folgt exakt derjenigen von Ruby (MVC), daher werden Programmierer, welche damit arbeiten, wenig Mühe haben, sich darin zurechtzufinden. Für Anfänger stellt dies jedoch eine sehr hohe Hürde dar, da die Benutzung von MVC nicht unbedingt sehr intuitiv ist. Nach einiger Einarbeitung sollte dies jedoch gut gehen.

Durch den Einsatz verschiedenster Technologien und dem Aufbau auf MVC wird hier eine gute Bewertungsnote erzielt.

Entwicklungsumgebung: Die von Motorola Rhodes zur Verfügung gestellte Software „RhoMobile Suite“ bietet ein umfangreiches Set von Möglichkeiten, welche man sich auch schon von Eclipse oder ähnlichen Umgebungen gewohnt ist: Text-Completion, on the run pre-compiling, Emulator und vieles mehr. Es enthält die aktuellen Komponenten von RhoElements (HTML5 Framework), RhoStudio sowie RhoConnect (Backend Server Services). Die Ähnlichkeiten zu Eclipse sind frappant, es wird sogar ein Plugin angeboten, welches in Eclipse eingebunden werden kann, damit Eclipse die gleichen Möglichkeiten bietet wie die RhoMobile Suite.

Es ist jedoch nicht zwingend mit diesen beiden IDE's zu arbeiten. Im Grunde kann jeder Texteditor dazu verwendet werden. Von Vorteil werden jedoch solche verwendet, welche HTML, Ruby und JavaScript beherrschen, um sich seine Arbeit zu erleichtern.

Durch die vielen Möglichkeiten, welche die Suite bietet, wird hier ein sehr gutes Resultat in der Bewertung erzielt.

Support: Durch die Orientierung des Frameworks auf Enterprise-Entwickler ist auch der Support äusserst professionell gehalten und befindet sich unter dem gleichen Dach wie der sonstige Motorola-Support. Es gibt die Möglichkeit, den Support per Telefon, Kontaktformular sowie per E-Mail zu erreichen. Leider ist für diese Dienstleistungen jeweils kein Preis angegeben, jedoch wird darauf verwiesen, dass der Support nur in Anspruch genommen werden kann, wenn man ein Service-Agreement oder Garantie hat. Dafür ist der Support in mehreren Sprachen verfügbar, auch auf

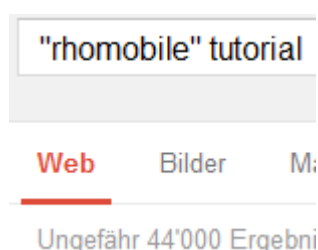
Deutsch.

Gratis hingegen ist die Google-Gruppe, wo man einerseits Kontakt zu den Entwicklern hat sowie auch zur restlichen Rhodes-Community. Leider ist diese nicht besonders gut besucht, was daher eher in einer schlechten Bewertung resultiert.

Es gibt auch zahlreiche Anleitungen im Netz, welche leider wegen des Namens (derselbe wie die berühmte Pianomarkte Fender Rhodes) teilweise eher schlecht zu finden sind. Dafür sind die gefundenen Quellen meist von guter Qualität, auch viele vom Anbieter selbst lassen sich finden. Über alles gesehen bietet der Support viele Möglichkeiten, jedoch muss wegen der fehlenden Angabe von Preisen sowie die etwas sperrige Bedienung des Supports ein Abzug in der Bewertung geltend gemacht werden.

Dokumentation: Die Dokumentation ist sehr gross, beinahe zu gross gehalten. Es gibt sehr viel zu entdecken, hauptsächlich auch daher, dass die Ressourcen wenig übersichtlich dargestellt sind. Die Dokumentationen sind äusserst ausführlich gehalten und decken jegliches Problem ab, das man haben könnte während des Programmierens. Hier ist auch die grosse Schwachstelle, denn man kämpft sich durch sehr viel Material, bevor man gefunden hat, was man sucht. Die Dokumentation ist online verfügbar, jedoch gibt es diese auch als Buch und, was ein kleines Highlight darstellt: Praktisch jedes Tutorial ist auch als Video verfügbar über Vimeo.com. In die Bewertung fliesst einerseits die ausführliche Dokumentation ein, jedoch auch die Unübersichtlichkeit sowie die Tatsache, dass alles überladen ist. Daher ergibt sich eine Mittelmässige Bewertung.

Tutorials: Die Google Suche fördert 44'000 Ergebnisse zu Tage:



Bemerkung: Da das Framework denselben Namen trägt wie das Fender Rhodes, ein weltbekanntes Elektropiano, wurde hier nach dem alten Namen „rhomobile“ gegoogelt, andernfalls wäre das Ergebnis stark verzerrt ausgefallen.

Bezeichnend für eine Enterprise-Lösung findet sich in Google relativ wenig zu Tutorials. Diese sind hauptsächlich auf der Rhodes-Homepage zu finden, was sich auch in den Suchergebnissen zeigt.

Gleich die ersten 4 Links zeigen auf die Rhodes bzw. RhoMobile Homepage, wo sich ein Füllhorn an Tutorials finden lässt (nach längerem Suchen, siehe oben)

Erst der fünfte Link zeigt auf Stackoverflow.com, wo eine Diskussion um rhomobile und deren Tutorials im Gange war.

Building: Bei Rhodes gibt es zwei Möglichkeiten, einen Build zu erstellen: Lokal sowie online.

Die lokalen Builds werden unkompliziert entweder per Kommandozeile gemacht oder direkt im RhoStudio kompiliert. Dies ist deswegen so unkompliziert, weil man keine Dependencies angeben muss, keine speziellen Flags oder ähnliches. Somit hat man in kurzer Zeit eine auf dem Device lauffähige Version.

Der online Build ist ebenfalls ziemlich simpel und wird über RhoHub abgewickelt. RhoHub stellt dieselben Möglichkeiten wie Github zur Verfügung und funktioniert genau gleich. Jedoch gibt es einen speziellen Button, der neben dem Projekt erscheint. Sobald man diesen Anklickt kann man die Plattform wählen und den Build beginnen. Danach kann man den kompilierten Code herunterladen. Kleiner Wehrmutstropfen: Man kann immer nur eine Plattform pro Vorgang angeben (Bei PhoneGap wird mit einem Klick jedes zuvor ausgewählte Framework direkt kompiliert).

Bei beiden Build-Möglichkeiten muss man sich zwingend an die vorgegebenen Strukturen halten, ansonsten wird der Building-Prozess fehlschlagen und relativ undifferenzierte Fehlermeldungen an den Tag legen.

Durch die verschiedenen Möglichkeiten und die Einfachheit des Prozesses erreicht Rhodes hier sehr gute Noten in der Bewertung, jedoch einen kleinen Abzug fürs einzelne Kompilieren.

Testing: Das Testing verhält sich wie bei Ruby: Rudimentär, aber gut. Es wird einerseits Unit-Testing angeboten, wie man es aus diversen Sprachen wie z.B. Java kennt und über weite Strecken äusserst strukturiert und methodisch funktioniert. Die relativ schwache Automatisierung des Testings sowie die manuelle Testerzeugung überzeugen jedoch nicht restlos.

Die sehr gute Dokumentation des Testings (hier für einmal nicht überladen) hilft jedoch weiter. Hier sind einerseits die Unit-Tests beschrieben, jedoch wird auch noch stark auf Logging, Debugging sowie Profiling (Performance-Testing) eingegangen. Es wird gut erklärt, wie Counters, Log-Abschnitte sowie Servermeldungen interpretiert werden können und daraus Schlüsse für die App gezogen werden können.

Die schwache Automatisierung wird durch die starke Dokumentation beinahe kompensiert, alles in allem gereicht es für eine gute bis sehr gute Bewertung.

Verbreitung: Die Google-Suche nach RhoMobile (es wird nicht nach Rhodes gesucht, siehe Abschnitt „Tutorials“) ergibt 268'000 Ergebnisse.

Es konnte leider keine Anzahl von Apps, welche mit dem Framework erstellt wurden, in Erfahrung gebracht werden. Dies liegt wohl auch daran, dass viele mit Rhodes programmierte Apps hauptsächlich intern in Firmen verwendet werden und ihren Weg in die jeweiligen App-Stores nicht gefunden haben und auch nie finden werden.

Von Rhodes hört man auch nichts, es ist uns erst im Verlaufe dieses Projektes bekannt geworden. Jedoch hinterlässt das Framework einen guten Eindruck. Deshalb muss man sich schon fragen, wieso man noch nie davon gehört hat.

Spezielles: Die angebotenen Tutorials(siehe Abschnitt „Tutorials“) im Video-Format sind äusserst hilfreich, weshalb hier ein Sonderpunkt vergeben wird.

4.4.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	6
Entwicklungsumgebung	7
Support	6
Dokumentation	6
Tutorials	3
Building	9
Testing	7
Verbreitung	3
Spezielles	1
Total	48

4.5. iUI

4.5.1. Einleitung



iUI ist ein Framework, welches JavaScript Libraries, CSS und Bilder bereitstellt um mobile Webapps zu erstellen. Ursprünglich wurde es nur für iPhone erstellt, mittlerweile unterstützt es (nach eigenen Angaben) die meisten Smartphones und Tablets (jedoch ohne exakte Angabe, welche dies sind). Es lassen sich damit Navigations-Menüs und Interfaces im iPhone-Style erstellen aus Standard-HTML. Des Weiteren unterstützt es Orientierungswechsel des Smartphones.

Das Framework ist extrem leichtewichtig (1.2 MB), lässt sich jedoch mit Extensions erweitern, welche jedoch noch nicht allzu zahlreich vorhanden sind.

4.5.2. Wie funktioniert das Framework

Die Entwicklung unter iUI passiert wie die Entwicklung einer normalen HTML Page und benötigt auch keine anderen Kenntnisse als HTML, CSS und JavaScript. Es wird kein natives App erzeugt, sondern es wird eine Webpage erstellt, welche dem Look-And-Feel des iPhones nachempfunden ist. D.h. die Page verhält sich so, als wären wir im Betriebssystem des iPhones und navigieren darin.

Dies wird erreicht durch verschiedene Thema-CSS, einige Zeilen JavaScript, welche eingebunden werden müssen und die Einbettung der beiden Technologien in HTML.

Das Framework ist äusserst puristisch gehalten, die Entwickler wollen jedem die Chance geben, das Framework zu verstehen. Die Möglichkeit Extensions einzusetzen erweitern die Möglichkeiten ein wenig, jedoch sind diese Extensions auch (noch) nicht in grosser Zahl vorhanden, aktuell sind es deren 9, unter anderem „Google Analytics“ Code, „Reachability Test“ sowie „Theme Switcher“.



Unterstützte Browser/Betriebssysteme

4.5.3. Evaluation

Einfachheit: Durch die begrenzten Möglichkeiten, welche das Framework bietet, ist auch dessen Einfachheit ziemlich offensichtlich. Die gesamte Dokumentation lässt sich in einer Stunde durchlesen und auch gut verstehen (ein wenig HTML und JavaScript Wissen vorausgesetzt).

Die eingesetzten Technologien HTML, CSS und JavaScript sind dabei die Hauptkomponenten, Schnittstellen für andere Sprachen sind nicht vorhanden, daher müssen andere Technologien auf eigene Gefahr eingesetzt werden. Jedoch, so sagt die Homepage: Solange es Core-Technologien sind

(z.B. XML) sollten diese mit iUI funktionieren.

Das Framework wird genauso genutzt, wie man es von der normalen HTML-Programmierung kennt: Man schreibt sein HTML, ergänzt die CSS-Definitionen um den Look ein wenig anzupassen und versieht die Page mit eventuellen JavaScript-Funktionalitäten (enthalten sein müssen natürlich die iUI-JavaScript-Funktionen).

Die Bewertung in diesem Punkt ist sehr gut, denn es ist simpel und relativ einfach zu verstehen.

Entwicklungsumgebung: Durch den rudimentären Einsatz von neuen Technologien benötigt iUI keine spezielle Entwicklungsumgebung, es wird jedoch darauf hingewiesen, dass von Vorteil ein WYSIWYG-Editor (What you see is what you get) eingesetzt wird, z.B. Dreamweaver. Dadurch wird einem die Programmierung ein wenig einfacher gemacht und man kann sofort sehen, ob das, was man programmiert, auch richtig angezeigt wird.

Da hier ein offenes Feld an Entwicklungsumgebungen genutzt werden kann, kann hier nicht effektiv bewertet werden, daher liegt die Bewertung im Mittelfeld mit einem kleinen Plus, da man sehr grosse Freiheiten geniesst.

Support: Da das Framework komplett Open-Source gehalten ist und das Entwicklerteam auch keinerlei Intentionen hat, damit Geld zu verdienen, gibt es keinen Support im eigentlichen Sinn. Es gibt jedoch einige Möglichkeiten, mit den Leuten von iUI und deren Community in Kontakt zu treten. Einerseits sei hier die Google-Gruppe iPhoneWebDev erwähnt, wo es einen Abschnitt für iUI gibt. Andererseits gibt es auch die Google-Gruppe iui-developers, wo sich auch der Erfinder von iUI des Öfteren zu einem Post hinreissen lässt und Hilfestellung bietet. Auch ein direktes Mail an die Entwickler kann man versenden. Die Mittel der Kommunikation sind dabei immer so gehalten, dass für die Entwickler sowie für den Anfrager keine Kosten entstehen.

Die Anleitungen sind gut verfügbar, aber auf Grund der Grösse des Frameworks eher klein, dafür jedoch gut aufgebaut und sauber.

Aufgrund der fehlenden Supportmöglichkeiten muss hier ein starker Abzug gemacht werden, die Punktzahl entspricht einem ungenügend.

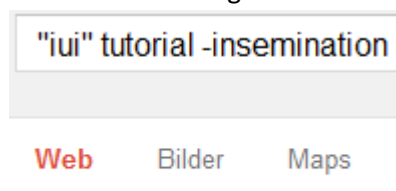
Dokumentation: Die Dokumentation von iUI ist hauptsächlich online verfügbar, es existiert jedoch ein Buch, in welchem ebenfalls in einem Kapitel auf iUI eingegangen wird. Jedoch ist die Angabe zu diesem Buch von der Homepage verschwunden, leider ohne Angabe von Gründen.

Die Dokumentation ist sauber aufgebaut in Struktur und Ordnung, dies durch einfache Online-Verlinkung und Sub-Gruppen. Was man sucht, findet man ziemlich schnell, wenn es denn existiert. Denn man erwartet meist etwas mehr vom Framework und ist dann enttäuscht, wenn die gesuchte Funktion nicht existiert.

Die Dokumentation ist in ihrer Ausführung sehr handlich gehalten, dies hängt auch mit der Grösse des Frameworks zusammen. Das Verständnis ist sehr gut, denn es werden einfache Konstrukte eingesetzt, dies ganz im Sinne der Erfinder, die das Framework auch einfach halten wollen.

Allgemein kann man hier eine gute Bewertung geben, einzig der kleine Funktionsumfang trübt die Bewertung.

Tutorials: Die Google Suche fördert 400'000 Ergebnisse zu Tage:



Bemerkung: Da iUI eine Abkürzung für Intrauterine Insemination ist (In-Vitro Befruchtung) wurden in den Google Ergebnissen diese Resultate ausgefiltert.

Die grosse Anzahl an Ergebnissen mag auf den ersten Blick

erstaunen. Jedoch ist hier das User-Modell stark auf die Community ausgerichtet. Es werden wenige Sachen auf der Homepage erklärt (dort jedoch gut und ausführlich), der Rest wird der Community überlassen. Somit lassen sich auch entsprechend viele User-Tutorials finden, was sich im Resultat der Google Suche widerspiegelt.

Der erste Link verweist simpel auf ein Frage-Antworte Forum, worin gefragt wird, wo man gute Tutorials für iUI finden kann und der entsprechenden Beantwortung (es wird auf die Programmierer-Seiten von Aaron Lerch sowie Scott Hanselman verwiesen).

Der zweite und dritte Link verweisen jeweils einmal auf die Dokumentationsseiten von iUI sowie deren Google-Gruppe.

Der vierte Link zeigt auf wholemap.com, wo sich ein relativ schlankes Tutorial zu einem „Hello World“-Beispiel finden lässt, welches gut und sauber funktioniert.

Der letzte Link ist abermals eine kurze Einführung zu einem „Hello World“-Beispiel, welches aber schlussendlich ebenfalls zu wholemap.com verweist, wo der Rest der Anleitung zu finden ist.

Building: Einen Building Prozess gibt es bei iUI nicht, der Browser interpretiert genauso, wie man die Pages geschrieben hat.

Da hier demnach keine objektive Bewertung gemacht werden kann, wird die Punktezahl 5 vergeben, dies als Mittelweg.

Testing: Einen Testing-Prozess gibt es für iUI nicht. Jedoch wird auf das sehr gute JavaScript Framework QUnit verwiesen, welches sehr gute Testresultate liefert. Dies demonstrieren die Entwickler auch auf ihrer Homepage, die Test-Demos sind unter http://iui-js.appspot.com/mobile/demos.html#_tests zu finden. Dieses externe Framework untersucht die Verhaltensweise der eingebetteten Java-Scripts und liefert strukturierte, methodisch ausgewertete Ergebnisse.

Speziell erwähnt sei auch die auf der iUI Homepage angebotene Extension „Reachability“, welches Tests für die Erreichbarkeit von hybriden Web-Apps zur Verfügung stellt. Eigentlich nichts schweres, jedoch kann dieser Test viele kleine Fehler finden.

Da das Framework keine eigenen Test-Methoden zur Verfügung stellt, wird hier keine gute Note verliehen, jedoch gibt es einen kleinen Punkte-Bonus, da auf externe Mittel verwiesen wird und diese auch erklärt werden.

Verbreitung: Die Google-Suche nach iUI bringt 18.1 Mio Ergebnisse hervor (auch nach Abzug der Ergebnisse von Intrauterine Insemination). Nach kurzer Durchsicht zeigt sich jedoch, dass hier auch diverse andere Sachen mit iui abgekürzt werden, unter anderem „intelligent user interface“, eine Abteilung namens iui an der Hochschule Osnabrück und einiges mehr. Deshalb kann hier kein Vergleich angestellt werden. Jegliche Versuche, die Ergebnisse zu filtern, ergaben immer noch sehr durchwachsene Ergebnisse. Daher lässt sich hier ein gescheiter Vergleich nicht anstellen. Auch die Anzahl der damit programmierten Web-Apps ist nirgends angegeben und liess sich auch mit einigem Aufwand nicht in Erfahrung bringen. Jedoch ist, auch aufgrund der Einfachheit des Frameworks, mit einer relativ grossen Verbreitung zu rechnen. Vom Framework selbst wurde vorher noch nie etwas gehört innerhalb der Autorengruppe und es wurde erst im Rahmen dieser Seminararbeit bekannt.

Spezielles: -

4.5.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	8
Entwicklungsumgebung	3
Support	4
Dokumentation	8
Tutorials	8
Building	5
Testing	6
Verbreitung	6
Spezielles	0
Total	48

4.6. Sencha Touch

4.6.1. Einleitung



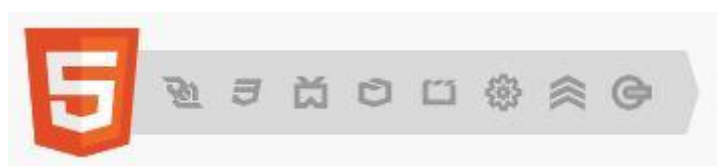
Sencha Touch ist ein HTML5-Framework, welches unter verschiedenen Lizenzen erworben werden kann. Zum Einen stehen Gratisversionen unter einer

kommerziellen und einer OpenSource-Lizenz zur Verfügung und zum Anderen eine kostenpflichtige OEM-Version. Unterstützt werden die Systeme iOS, Android, BlackBerry, Kindle Fire und weitere.

Aktuell liegt die Version 2 vor, allerdings wird die Version 1.x immer noch angeboten und supportet. Die Version 2 bietet allerdings zusätzliche Features an wie Native-APIs, mit welchem Hardware-Komponenten wie beispielsweise die Kamera angesprochen werden kann.

4.6.2. Wie funktioniert das Framework

Das Framework basiert auf HTML5 und bietet mittels JavaScript ein Menge weiterer Funktionen, unter anderem verschiedene Komponenten, Layouts, Zeichnungs- und Animations-Tools, Diagramme sowie Ansteuerung von Geräteschnittstellen. Speziell ist, dass das gesamte UI in JavaScript geschrieben wird. Dies wird mit einer JSON-ähnlichen Notation erreicht.



4.6.3. Evaluation

Einfachheit: Die Dokumentation und die How-To's auf der Sencha-Homepage geben einen guten Einblick in die Funktionsweise des Frameworks. Auffällig ist, dass nur JavaScript-Code für die Programmierung verwendet wird. Der HTML-Inhalt wird dann mittels HTML-Attributen in JSON angegeben. Für Kenner von anderen Sencha-Frameworks wie beispielsweise Ext.Js wird dies kein Problem sein. Wenn man sich jedoch gewohnt ist, JavaScript ergänzend zu HTML zu schreiben, erscheint die Sencha-Art eher fremd. Dies gilt allerdings nur für die Verwendung der von Sencha Touch bereitgestellten Komponenten. Wenn die Komponenten selbst entwickelt werden kann dies auch gewohnt mit einem HTML-Layout und eingebettetem JavaScript entwickelt werden. Allerdings werden wir einen kleinen Abzug für die eher ungewohnte Notation notieren.

Entwicklungsumgebung: Die Dokumentation lässt keine Schlüsse offen für eine spezifische IDE. Im Prinzip kann ein normaler Texteditor verwendet werden. Die Internet-Recherche ergab, dass Entwicklungsumgebungen wie Netbeans, Eclipse / Aptana, WebStorm, ... von der Community verwendet werden.

Nach der Installation des Sencha SDKs und den dazugehörigen SDK Tools kann es losgehen. Mit dem durch die SDK Tools installierten „Sencha“-Tool kann eine komplette Applikationsstruktur generiert werden. Dieser Ansatz ist bereits durch andere Frameworks (wie beispielsweise Rails) bekannt. Die Applikation kann anschliessend bereits über einen Browser aufgerufen werden und eine schöne Begrüssung wird angezeigt. Darauf ist auch auf das Main.js verwiesen, welches für das Rendering der Startseite verantwortlich ist. Dies bietet einen sehr komfortablen und schnellen Einstieg und wird entsprechend gut bewertet.

Support: Es stehen grundsätzlich drei Support-Lizenzen zur Verfügung, welche in eine Standard und zwei Premium Lizenzen unterteilt werden. Die Lizenzen sind jeweils Entwicklerbasiert und der Preis bewegt sich zwischen 299\$ und 4995\$. Dafür erhält man Updates, Upgrades und Zugang zu Standard und Premium Foren. Bei einer Premium-Lizenz stehen zusätzlich Telefonsupport und Emergency-Bugfixes zur Verfügung.

Kostenlos kann natürlich über das Sencha-Forum der Community ebenfalls um Ratschlag gebeten werden.

Alles in Allem stehen diverse Support-Möglichkeiten zur Verfügung und wird entsprechend gut bewertet.

Dokumentation: Sencha bietet auf ihrer Homepage eine sehr ausführliche, teilweise extrem technische Dokumentation. Trotzdem ist diese immer zweckgemäss gehalten und bietet viele In-Browser-Beispiele, welche die fertige App demonstrieren. Die Gliederung ähnelt einer API, bei welcher die JavaScript-Prototypen und –Methoden aufgelistet werden. Dies macht den Einstieg für einen Anfänger eher schwierig, allerdings ist dies bei allen Sencha-Frameworks so gehalten, sodass sich Sencha-Fans sicherlich darin wohlfühlen.

Tutorials: Diese sind in den meisten Fällen direkt in die Dokumentation eingebettet. So wird die technische Dokumentation gelungen abgerundet.

Direkt auf der Website ist auch ein Forum verfügbar, welches eine gute Gliederung nach Framework, Version, Kundenmodell, etc. präsentiert. Unter anderem bietet das Forum sogenannte Showcases, welche Beispielimplementierungen, Tutorials, Repositories zu eigens geschriebenen Komponenten,

etc. beinhalten. Diese Einträge machen einen sehr guten Eindruck, da bei den meisten Posts ebenfalls Feedback und Anregungen vorhanden sind.

Die Suche nach "Sencha Touch Tutorial" ergab 387'000 Treffer. Der erste Treffer verweist auf die Sencha-Homepage. Dort befindet sich eine nette Sammlung an Tutorials verschiedener Themenbereiche. Sehr angenehm wirkt auf den ersten Blick auch die Angabe des Schwierigkeitsgrads, welcher über „Easy“, „Medium“ und „Hard“ angegeben wird. Ebenfalls sehr schön, dass je nach Tutorial ein Video oder reine Dokumentation mit Source-Code zur Verfügung steht.

Der zweite Link verweist ebenfalls auf die Sencha-Homepage und beschreibt ausführlich ein „Hello World“-Beispiel. Dies ist sehr einfach gehalten und abgesehen von den HTML-Tags wird jede Zeile einzeln erklärt.

Der dritte Link verweist auf den Sencha-Blog, welcher ebenfalls wertvolle Beiträge beinhaltet wie zum Beispiel „Wie schreibe ich eigene Komponenten“. Hier fällt allerdings auf, dass nicht nur Sencha Touch sondern ebenfalls andere Sencha-Frameworks beschrieben sind. Also ist bei den Artikeln darauf zu achten, dass über das richtige Framework gesprochen wird.

Der vierte und fünfte Link verweisen auf die Seite miamicoder.com von Jorge Ramon. Dort werden Schritt-Für-Schritt-Anleitungen zur Entwicklung von eigenen Sencha-Touch-Apps angeboten und sind sehr einfach aufgebaut.

Gesamthaft bewerten wir die Tutorials sehr gut.

Building: Das Building erfolgt lokal über die Sencha SDK Tools. Dabei wird die gebildete Sencha-Library optimiert, d.h. es werden nur die benötigten Komponenten in den Build kopiert. Diese Einsparung macht sich dann beim Download und der Laufzeit bemerkbar.

Um ein natives Package zu erstellen können ebenfalls die Sencha SDK Tools eingesetzt werden, allerdings muss die App mit zusätzlichen herstellerspezifischen Tools signiert werden. Wenn die App beispielsweise über den Android Market angeboten werden soll, muss die App vor dem Build über das Android SDK zertifiziert werden. Dies gilt natürlich auch für den Apple Store, hier ist der Aufwand jedoch noch etwas höher, da Apple ja bekanntermassen hohe Qualitätsanforderungen und – Prüfungen durchführt.

Das bedeutet letztendlich, dass ein Native-Build geräteabhängig durchgeführt werden muss. Allerdings finden sich viele Verweise auf den PhoneGap-Online-Build. Dies gelingt allerdings nicht mit der reinen Sencha-App.

Positiv: Der Build lässt sich mit dem Tool Ant automatisieren. Dies dürfte vor allem Java-Entwicklern bestens bekannt sein und die Einarbeitung dürfte sich entsprechend mühelos gestalten.

Alles in allem etwas enttäuschend, dass die App für jede Art Device eigens gebildet werden muss, allerdings erfreulich, dass ein externes Build-Tool eingesetzt werden kann. Deshalb bewerten wir diesen Punkt eher durchschnittlich.

Testing: Das erste Testwerkzeug ist laut diversen Foreneinträgen und Blogs der Simulator, also manuelles Testing. Dies ist im Grunde ein allgemeines Problem von RIAs (Rich Internet Applications), da sich die Struktur des Dokuments zur Laufzeit ändert.

Entsprechend findet sich leider (noch) kein Tutorial in der Sencha-Dokumentation in Punkto „automatisierte Tests“. Allerdings gibt es einen sehr schönen und ausführlichen Blog-Eintrag unter <http://www.sencha.com/blog/automating-unit-tests>. Dort wird auf Syntax Checks und Unit-Tests eingegangen. Für Web-Entwickler dürfte der Einstieg relativ einfach sein, da insbesondere für Unit-Tests das Jasmine-Framework eingesetzt wird, mit welchem JavaScript getestet werden kann.

Hier bewerten werden wir in der Bewertung entsprechend Abzug geben, da sich automatisierte Tests eher umständlich gestalten.

Verbreitung: Die Suche mit „Sencha Touch“ ergibt ein Ergebnis von 1,49 Mio Einträgen.

Lustigerweise ergibt die Suche mit „Sencha Touch 2“ ein Ergebnis von 4,26 Mio Einträgen. Dieses Ergebnis ist aus unserer Sicht relevanter, da die neue Version mehr Features bietet und auch in der Dokumentation mehr zu finden ist. Das Ergebnis ist schon recht beeindruckend und deutet auf eine grosse Verbreitung hin.

Auf der Sencha-Homepage finden sich unter der Rubrik „Who’s using it“ interessante Informationen. Seit 2010 haben mehr als 500’000 Entwickler das Framework runtergeladen und damit Zehntausende von Apps geschrieben. Einige davon werden in der App Gallery vorgestellt, welche wöchentlich aktualisiert wird und über 200 Apps beinhaltet.

Spezielles: Speziell zu erwähnen sind die Live-Demos in den Dokumentationen. Mit einem WebKit-Fähigen Browser wie Chrome oder Safari können die Demos direkt im Browser ausprobiert werden. Deshalb vergeben wir hier die möglichen zwei Extrapunkte.

4.6.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	6
Entwicklungsumgebung	8
Support	7
Dokumentation	8
Tutorials	7
Building	6
Testing	6
Verbreitung	8
Spezielles	2
Total	58

4.7. XUI

4.7.1. Einleitung



XUI ist ein Micro-JavaScript-Framework für HTML5 basierte, mobile Applikationen. Es wird seit dem Jahr 2008 entwickelt und ist mit einer Basisgrösse von 10.4Kb extrem klein. Dies kommt daher, dass das Framework einzig DOM-Zugriff und -Manipulation bereitstellt und weder Komponenten noch Native-APIs zur Verfügung stellt. Ausserdem wurde es von PhoneGap-Entwicklern ins Leben gerufen mit der Vision, ein JavaScript-Framework zu erstellen, welches

performanter ist als die Kontrahenten JQuery, Prototype, MooTools, usw. XUI steht unter der MIT-Lizenz und kann entsprechend frei verwendet werden.

4.7.2. Wie funktioniert das Framework

Das Framework funktioniert ähnlich wie JQuery. Die DOM-Einstiegsfunktion heisst `$x(...)` und bietet anschliessend verschiedene Manipulations-, Zugriffs-, und Event-Funktionen an. Ausserdem werden Ajax-Aufrufe angeboten. Im Gegensatz zum klassischen JQuery-Framework werden hier auch Touch-Gesten unterstützt. Allerdings stehen zurzeit leider lediglich vier Plugins zur Verfügung.

Im Vorfeld soll hier noch erwähnt werden, dass das Framework nicht als eigenständiges Framework entwickelt wurde, sondern ergänzend zu beispielsweise PhoneGap eingesetzt werden kann, was die Evaluation eher schwierig gestaltete.

```
// a trivial example
x$('#btn').click( function (e) {
    x$('#msg').html('Thanks for your submission!');
})
```

4.7.3. Evaluation

Einfachheit: Das Framework basiert ausschliesslich auf JavaScript, welches dann in das eigene HTML-Layout eingebettet und mit eigenen CSS-Styles formatiert werden kann. Als Web-Entwickler fällt der Einstieg sehr leicht. Die Dokumentation ist in ca. 20 Minuten gelesen und die Syntax ist der von JQuery, Prototype oder MooTools sehr ähnlich. Für unsere Arbeit fehlt uns jedoch die Ansteuerung von Gerätespezifischen Schnittstellen. Dieses Produkt ist allerdings nicht darauf ausgelegt, sondern sollte in Kombination mit einem grösseren Framework wie PhoneGap benutzt werden. Trotzdem bewerten wir hier mit sehr gut.

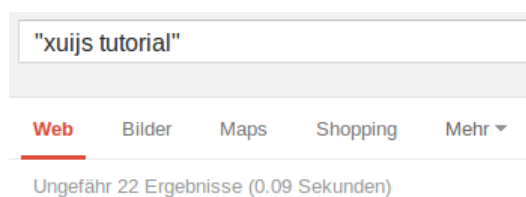
Entwicklungsumgebung: Hier sind alle Möglichkeiten offen. Da das Framework sehr einfach gehalten ist, würde ein normaler Editor mit eingebautem Wörterbuch schon genügen. Es sind bekannterweise aber auch IDEs verfügbar wie Eclipse oder NetBeans, welche JavaScript unterstützen. Hier kommt es dann stark darauf an, welche zusätzlichen Frameworks eingesetzt werden. Demzufolge ist die unendliche Flexibilität sehr angemessen und wird hier sehr gut bewertet.

Support: Direkten Support gibt es bei diesem sehr schlicht gehaltenen Framework nicht, man ist also auf die Community angewiesen. Auf der XUI-Homepage wird auf einen PhoneGap-IRC-Chat und die PhoneGap-Google-Groups-Seite verwiesen. Mit etwas Glück findet sich auch was im Stackoverflow-Forum. Wer sich nun denkt, er könne sich im schlimmsten Fall auf Google verlassen, ist definitiv zu optimistisch. Zum einen existieren andere Frameworks mit gleichem Namen und andererseits wird das Framework zwar auf vielen Seiten erwähnt, jedoch praktisch immer als Beigeschmack von PhoneGap. Dies deutet darauf hin, dass die zuverlässigsten Quellen bei Problemen tatsächlich die Google-Group sowie der IRC-Chat sind.

Diesen Punkt bewerten wir entsprechend mit genügend.

Dokumentation: Diese gestaltet sich sehr rudimentär und technisch, ist dafür sehr übersichtlich und in wenigen Minuten gelesen. Allerdings wird dies gerade bei Neueinsteigern viele Probleme und Missverständnisse nach sich ziehen. Allem Anschein nach ist dies aber auch nicht die Zielgruppe, sondern eher eingeseessene Web-Entwickler, welche mehr Dynamik aus einer Seite herausholen möchten. Die komplette Dokumentation ist über die XUI-Homepage in HTML-Form einsehbar und weist eine einfache und klare Strukturierung auf, welche nach Themenbereich gegliedert ist. Ansonsten sieht es mit Quellen von Drittanbietern ähnlich mager aus wie mit dem Support. Wir bewerten hier ebenfalls mit dem Durchschnitt, da die hausgemachte Dokumentation zwar knapp aber dennoch angemessen ausfällt, jedoch fremde Quellen gesamtheitlich fehlen.

Tutorials: Hier müssen wir leider auf die Dokumentation verweisen. Es existieren praktisch keine



Tutorials von Drittpersonen. Auch die Google-Suche lieferte bei der Eingabe „xuijs tutorial“ gerade mal 22 Treffer, wobei hier explizit mit „xuijs“ gesucht wurde, um das Ergebnis nicht mit Treffern von anderen Frameworks zu verfälschen.

Building: Wie bereits erwähnt ist dieses Framework nicht als Standalone-Framework gedacht. Entsprechend gibt es hier keinen spezifischen Build-Vorgang. Einzig ist beim Einsatz darauf zu achten, dass die richtige Version für die jeweiligen Browser heruntergeladen wird. Hier stehen die Versionen für die Browser WebKit, FireFox und Opera, eine Version für Blackberry Mobile und wie so oft eine für IE.

Wir bewerten diesen Punkt mit genügend, da dies eigentlich gar nicht zu diesem Framework passt.

Testing: Hier werden keine speziellen Hilfsmittel angeboten. Allerdings könnte hier - wie bei vielen JavaScript-Implementierungen - auf Jasmine zurückgegriffen werden. Sehr interessant ist allerdings der Menüpunkt "Tests" der XUI-Homepage. Hier können Testläufe für das Framework selbst gestartet werden. Leider sind diese während unserer Evaluation gänzlich fehlgeschlagen. Dies scheint an Pfad-Anpassungen innerhalb der Homepage-Struktur zu liegen, wie wir dem Code entnehmen konnten. Hoffentlich wird dieser Fehler bald wieder behoben.

Für die Bewertung müssen wir hier leider ebenfalls massive Abzüge geben, weil zwar Test-Szenarien für das Framework vorhanden sind, diese jedoch nicht funktionieren. Des Weiteren fehlen hier ebenfalls Referenzen und Tutorials.

Verbreitung: Trotz all diesen fehlenden Informationen, scheint das Framework relativ weit verbreitet zu sein. GitHub-Forks existieren zur Zeit 132. PhoneGap hat im Vergleich dazu 623, obwohl dieses Framework massiv viel mächtiger ist als XUI. Wir vermuten hier, dass die fehlenden Informationen daraus resultieren, dass sich XUI sehr stark an den Kontrahenten jQuery, Prototype und MooTools orientiert, was zu einer leichten Anwendung führen kann. Die Google-Suche bringt hier dennoch keine brauchbaren Resultate.

Diesen Punkt bewerten wir aus Fairness-Gründen als genügend.

Spezielles: Sehr schön ist der Framework-Test auf der Homepage. Da dieser jedoch nicht funktioniert, geben wir hier keine Punkte.

4.7.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	9
Entwicklungsumgebung	9
Support	5
Dokumentation	5
Tutorials	0
Building	5
Testing	4
Verbreitung	5
Spezielles	0
Total	42

4.8. iWebKit

4.8.1. Einleitung



iWebKit ist ein Webseiten-Framework für iPhone, iPad und iPod Touch und unterstützt grundsätzlich alle gängigen Web-Technologien wie HTML, CSS, JavaScript und PHP.

Es ist unter der LGPL-Lizenz verfügbar, das heisst, es ist für nicht-kommerzielle Nutzung kostenlos. Andernfalls kann jedoch eine kommerzielle Lizenz für rund 20 Euro erworben werden.

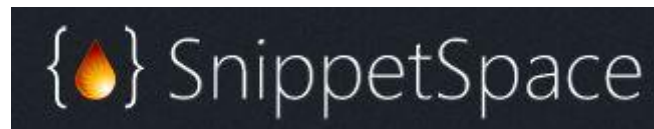
Entwickelt wird es vom 20-jährigen Studenten Christopher Plieger. Er startete die Entwicklung im September 2008 und hat es seither auf Grund der zunehmenden Verbreitung laufend weiterentwickelt. Aktuell liegt die Version 5.04 vor, Version 6 befindet sich zur Zeit im Entwicklungsstadium.

4.8.2. Wie Funktioniert das Framework

Das Framework besteht grundsätzlich aus einem Template, welches die Technologien HTML, CSS und JavaScript beinhaltet, wobei JavaScript lediglich für das Ermitteln der Bildschirmauflösung verwendet wird. Für RSS-Feeds ist bereits eine PHP-Vorlage vorhanden, welche entsprechend angepasst und auf der eigenen Seite eingebunden werden kann.

Speziell ist, dass es sich nicht um ein App-Framework im eigentlichen Sinne handelt, sondern um ein Webseiten-Template, welches auf Apple-Produkte optimiert ist. Über die Homepage sowie über die mitgelieferte Demo kann schnell ein Überblick über die verschiedenen Funktionen verschafft werden. Es werden verschiedene Form-Elemente unterstützt und iPhone-Apps wie Mail, SMS,

iTunes, Appstore, Telefon / Kontakte, YouTube und Google-Maps können direkt angesprochen werden.



4.8.3. Evaluation

Einfachheit: iWebKit ist extrem einfach gehalten. Nach dem Download wird mit einem 13-seitigen User-Guide eine sehr einfache Einführung gegeben. Danach kann es auch schon losgehen. Einstiegspunkt ist wie üblich die index.html-Datei. Hier kann dann der gewünschte Inhalt sowie verschiedene Komponenten wie BreadCrumb, Navigation, Listen, Forms usw. eingebunden werden. Dafür muss nicht einmal zwingend HTML-Knowhow vorhanden sein. Wenn man dem User-Guide folgt, können die einzelnen Code-Snippets direkt kopiert und eingefügt werden. Nur der Inhalt muss selbst angepasst werden.

Für Webseiten, welche ausschliesslich auf Apple-Produkte zugeschnitten sind, reicht dies vollkommen aus. Wenn aber Wert auf Browser-Kompatibilität gelegt wird, muss doch einiges an Handarbeit investiert werden, da nur schon die Demo auf dem Firefox nicht anständig angezeigt wird.

Dass sogar totale Web-Anfänger eine eigene Webseite erstellen können, bewerten wir hier mit sehr gut. Für die fehlende Browserkompatibilität werden wir jedoch einen Abzug geltend machen.

Entwicklungsumgebung: Hier genügt grundsätzlich ein normaler Texteditor, es kann aber eine Entwicklungsumgebung eigener Wahl eingesetzt werden, Hauptsache es werden die genannten Technologien unterstützt. Durch unsere Internetrecherche konnten wir bei diesem Framework jedoch keine Präferenzen entdecken.

Ebenfalls sehr positiv ist, dass Plugins existieren für andere Frameworks wie das Grails-Framework oder Drupal. So können Applikation, welche auf diesen Frameworks basieren, auf einfache Art und Weise mit einem iPhone Look & Feel ausgestattet werden.

Insgesamt sehr positiv auf Grund der grossen Flexibilität und dafür, dass für andere Frameworks Plugins zur Verfügung stehen. Dies wird entsprechend gut bewertet.

Support: Da iWebKit von einem einzigen Entwickler ins Leben gerufen wurde und voran getrieben wird, ist der direkte Support auch über die Homepage zu suchen. Hier ist vor allem das Forum der Dreh- und Angelpunkt. Beiträge werden häufig erfasst und beantwortet, häufig sogar durch Christopher Plieger selbst. Das Forum weist ebenfalls eine gute Strukturierung nach verschiedenen Themen auf, so können entsprechende Beiträge schnell gefunden werden. Daneben gibt es noch einen Blog, welcher aber im Gegensatz zum Forum sehr spärlich ausfällt.

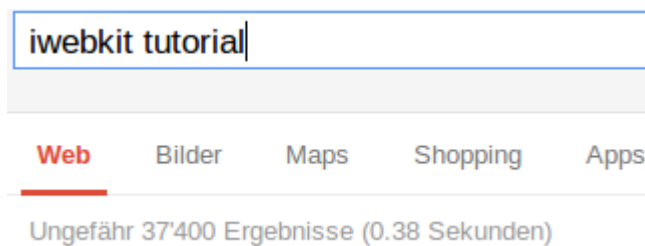
Die Community ist aber auch aktiv und so findet man auch schnell Einträge in bekannten Foren wie Stackoverflow.

In Anbetracht dessen, dass dieses Framework von einer Person entwickelt wird, fällt dies sehr positiv aus und wird entsprechend gut bewertet.

Dokumentation: Diese wird direkt mit dem Framework als PDF-Datei heruntergeladen und ist sehr einfach und strukturiert gehalten. Am Anfang wird ein sehr kurzer Abriss über HTML gehalten, doch es wird gleichzeitig betont, dass grundsätzlich keine Vorkenntnisse vorhanden sein müssen. Im Gegenteil, die Beispiele sind so gehalten, dass diese per Copy & Paste übernommen werden können und nur noch der textliche Inhalt angepasst werden muss. Gelesen sind die 13 Seiten relativ schnell und man ist danach in der Lage, eine einfache Webseite zu erstellen.

Des Weiteren finden sich auf YouTube viele anschauliche Videos, wie eine App erstellt werden kann. Grundsätzlich sehr gut gelungen, da sie vor allem sehr einsteigerfreundlich aufgebaut ist und viel Komplexität erspart bleibt. Dies kann jedoch für Fortgeschrittene eher Mühsam sein, da es in diesem Dokument eher schwierig ist, direkt ein relevantes Kapitel zu lesen, ohne den Rest überfliegen zu müssen. Wir bewerten dies mit gut.

Tutorials: Die Suche nach iWebKit Tutorial ergibt 37'400 Ergebnisse, was auf den ersten Blick nicht



wirklich überragend ist. Auf den zweiten Blick sind diese jedoch sehr einfach gehalten und visualisieren in jeder Etappe das erwartete Resultat.

Die erste Seite verweist auf ein YouTube-Video, welches vom Download bis zur eigenen kleinen Seite alles zeigt, was man als Anfänger

wissen muss.

Der zweite und dritte Link verweisen auf einen Blog, welcher in neun Schritten das Erstellen einer eigenen iPhone App anhand iWebKit erklärt. Dies gestaltet sich sehr einfach und intuitiv.

Der vierte Link verweist auf einen Blog, welcher das Aufsetzen und Erstellen einer Tapestry-Applikation zusammen mit iWebKit erklärt. Tapestry ist ein Framework um Web-Applikationen mit Java zu erstellen. Hier wird wieder mehr Know-How vorausgesetzt, allerdings eher auf Grund des Tapestry-Frameworks.

Der fünfte Link zeigt schlussendlich auf die Seite HTMLGoodies.com. Dieses enthält allerdings fast keine Erklärungen, sondern fertiger Code, welcher dann kopiert werden kann, um das dargestellte Resultat zu erzielen.

Building: Da mit iWebKit lediglich eine Webseite erstellt wird, entfällt das Building komplett. Falls es mit anderen Frameworks wie Grails oder PhoneGap kombiniert wird, muss dies natürlich mit dem jeweiligen Framework-Build durchgeführt werden. An dieser Stelle kann aber nicht darauf eingegangen werden.

Wir bewerten hier der Fairness halber mit genügend.

Testing: Das Framework selbst bietet - wie bereits angesprochen - im Grunde nur ein auf iPhones und Co. zugeschnittenes Layout. Dieses ist grundsätzlich statisch und kann nur manuell getestet werden. Hier kommt es wieder auf die zusätzlichen Technologien an, wie die Seite schlussendlich getestet werden kann. Aus diesem Grund kann dieser Punkt kaum bewertet werden, weshalb wir die Note genügend geben.

Verbreitung: Die Google-Suche nach iWebKit ergibt insgesamt 162'000 Treffer. Für ein so kleines Framework, ist dies doch recht erstaunlich. Dies deutet darauf hin, dass es sich auf Grund der Einfachheit schnell beliebt machen konnte. Auch für komplexere Web-Applikationen wird hier das Apple-Look & Feel sehr komfortabel bereitgestellt und kann meist ohne grössere Anpassungen

verwendet werden, sofern Apple-User der Zielgruppe genügen.

Leider konnten wir nicht genau ermitteln, wie viele Apps tatsächlich damit schon entwickelt wurden, doch es dürfte für sich sprechen, dass es sogar auf der Apple-Homepage unter der Rubrik "Productivity" beschrieben wird.

4.8.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	7
Entwicklungsumgebung	8
Support	7
Dokumentation	7
Tutorials	3
Building	5
Testing	5
Verbreitung	2
Spezielles	0
Total	44

4.9. jQuery Mobile

4.9.1. Einleitung



jQuery Mobile stammt von der jQuery Foundation und basiert auf jQuery und jQuery UI. Es wurde mit der Vision entwickelt, kompakt, schnell, einfach anpassbar und auf möglichst allen Systemen lauffähig zu sein. Unterstützt werden Plattformen wie iOS, Android, Blackberry, Bada, Windows Phone, Palm WebOS, Symbian und MeeGo. Die aktuellste Version ist 1.2.0 und steht unter der MIT-Lizenz. Somit ist dieses Framework für jeden Verwendungszweck frei verfügbar.

4.9.2. Wie funktioniert das Framework

Dieses Web-App-Framework ist komplett in JavaScript geschrieben und kann ganz gewohnt in das eigene HTML-Layout eingebettet werden. Die Anwendung gestaltet sich dann ähnlich wie es bei den Geschwistern jQuery und jQuery UI bereits bekannt ist. Allerdings werden hier die HTML5-Attribute "data-..." bereits ganzheitlich interpretiert. So können ganz einfach Web-Applikationen mit einem klassischen App-Look erstellt werden.



4.9.3. Evaluation

Einfachheit: Eingesetzt werden hier die bekannten Technologien HTML, CSS und JavaScript, welche für einen Web-Entwickler ganz gewohnt angewendet werden können. Wer vorher schon mit jQuery gearbeitet hat, dürfte keine Schwierigkeiten bei der Einarbeitung haben. Sehr schön ist, dass viele Grundfunktionalitäten ausschliesslich über entsprechende HTML-Tags und -Attribute erreicht werden können. Für Design-Anpassungen gibt es auch noch einen schönen Drag & Drop-Editor, welcher anschliessend das fertige CSS bereitstellt.

Das Framework selbst hat man mit rund zwei Klicks heruntergeladen, es kann aber auch über den entsprechenden Online-Link in das Layout integriert werden. So ist es auch in den Beispielen der Dokumentation gehalten, was das Kopieren der Beispiele zum Kinderspiel macht.

Sehr schön und einfach gehalten, sowohl für Anfänger als auch für eingesessene Web-Entwickler. Wir bewerten diesen Punkt sehr gut.

Entwicklungsumgebung: Diese Entscheidung wird jedem offen gelassen, wie es bei vielen JavaScript-Frameworks der Fall ist. Laut Community soll hier Aptana bereits ein jQuery-Bundle zur Verfügung stellen, welches bekannte Features wie Code-Completion zur Verfügung stellen soll. Dreamweaver wird auch oft eingesetzt, diese IDE wird bereits mit jQuery und jQuery Mobile ausgeliefert. Doch für welche man sich entscheidet, ist jedoch schlussendlich Geschmackssache.

Alles in Allem wird hier eine sehr grosse Entscheidungsfreiheit geboten, was wir entsprechend gut bewerten.

Support: Leider gibt es hier keinen Telefon- oder E-Mail-Support, was aber auf Grund der freien Verfügbarkeit nicht weiter wundert. Viele gute Inputs und Hilfestellungen können jedoch über das Forum bezogen werden. Dieses weist eine sehr intensive Nutzung auf, und es werden viele Themenbereiche abgedeckt. Für anschaulich Beispiele und Einstiegshilfen kann auch der Blog konsultiert werden. Dieser kommt allerdings etwas überladen daher und es fehlt eine hilfreiche Suchfunktion.

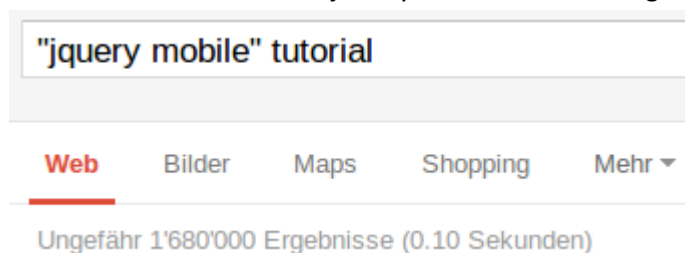
Hilfreich zeigte sich auch die entsprechende Google-Group. Dort werden sehr viele Probleme und Themenbereiche abgedeckt und häufig Posts erfasst. Erfreulich sind auch die vielen Verweise auf dazugehörige GitHub-Repositories, wo dann der entsprechende Code konsultiert werden kann.

Dokumentation: Die Dokumentation ist online über die Homepage verfügbar, welche man bereits über den ersten Menüpunkt erreicht. Durch das übersichtliche Inhaltsverzeichnis wird man ganz behutsam an die Materie herangeführt. Zuerst werden die absoluten Basics erklärt, welche sich ganzheitlich mit Seitenaufbau und Layout beschäftigen und erst viel später kommt man mit JavaScript in Kontakt. Dennoch können die entsprechenden Kapitel auch von Profis sehr einfach und schnell gefunden werden, wobei die ganze Sache etwas mehr Gehirnschmalz fordert. Besonders gut ist hier die hierarchische Struktur gelungen, welche je nach gewähltem Kapitel in die entsprechenden Unterbereiche führen ohne dass das Gefühl entsteht, man wäre hier an einem falschen Ort gelandet.

Des Weiteren ist die Dokumentation die beste Werbung für das Framework, da diese sowohl von Desktop-Rechner wie auch von Touch-Devices sehr angenehm gelesen werden kann.

Diesen Punkt bewerten wir sehr gut.

Tutorials: Die Suche nach "jQuery Mobile" Tutorial ergibt 1'680'000 Resultate. Hier wurden die



Anführungszeichen explizit verwendet, um Treffer für das jQuery-Framework auszuschliessen. Hier wird vermutlich die Beliebtheit von jQuery deutlich, was eine Entscheidung für dieses Framework nahezu selbstverständlich macht.

Der erste Treffer verweist auf die Resources-Rubrik der jQuery Mobile Homepage. Dort wird auf verschiedene Quellen wie Bücher, Plugins, Erweiterungen und mit jQuery Mobile geschriebene Apps verwiesen.

Der zweite Link verweist auf eine Sammlung von nützlichen Tutorials. Hier werden allgemeine Informationen, How-To's, Integration in weitere Frameworks und vieles mehr angeboten.

Über den dritten Link gelangt man auf eine Seite, welche 11 Tutorials für dieses Framework evaluiert hat und auf diese verweist. Diese sind alle einfach gehalten und sehr eindrücklich illustriert, ohne die technischen Details auszulassen.

Der vierte Link führt auf die Seite spyrestudios.com, welche eine Einführung für Anfänger enthält. Hier werden die wichtigsten vier Punkte aufgegriffen und erklärt.

Letztlich findet man beim fünften Link ein Tutorial, welches die gängigsten Komponenten erklärt. Dabei wird Schritt für Schritt vom Basis-Seitenaufbau bis zum Popup die Funktionsweise erklärt. Der Inhalt sowie die Komplexität sind bei diesen Tutorials sehr angemessen. Meistens wird das zu erwartende Ergebnis zu Beginn angepriesen und die einzelnen Schritte mit Bildern illustriert.

Building: Da es sich hier um ein webbasiertes Framework handelt, fällt das Building weg. Allerdings wird in der offiziellen Dokumentation auf PhoneGap verwiesen, um eine native App builden zu können. An dieser Stelle wird auch auf bekannte Probleme eingegangen und wie man diese mit der richtigen jQuery Mobile Konfiguration lösen kann. Laut Community werden PhoneGap und jQuery Mobile häufig miteinander kombiniert, da sich letzteres sehr schön für das Layouting der Seite eignet.

Dieser Punkt kann so nicht direkt bewertet werden, somit geben wir hier eine durchschnittliche Note.

Testing: Dieser Punkt wird über die offizielle Homepage leider nicht abgedeckt. Allerdings liess sich ein Framework finden, welches die jQuery Foundation selbst einsetzt, um ihre Frameworks zu testen: QUnit. Diesbezüglich findet man auch schnell Tutorials und Hilfestellungen über Google und das jQuery-Forum. Es können aber auch andere Frameworks wie Jasmine oder Selenium eingesetzt werden.

Schön, dass es Möglichkeiten gibt, allerdings schade, dass dies nicht offiziell über die Dokumentation abgehandelt wird. Wir bewerten hier mit dem Durchschnitt.

Verbreitung: Die Google-Suche bringt hier 4'520'000 Treffer, was eine grosse Verbreitung vermuten lässt. Dies bestätigt auch die JQM-Gallery (jQuery-Mobile-Gallery), welche rund 170 Apps zählt. Hier werden aber anscheinend nur die reinen jQuery Mobile Apps aufgelistet. Die Anzahl von nativen Apps, welche beispielsweise mit Hilfe von PhoneGap entwickelt wurde, sei hier noch nicht erwähnt. Die Bekanntheit von jQuery und Co. ist zweifelsohne sehr beeindruckend. Nahezu jeder, der schon mal eine Web-Seite entwickelt hat, ist mit einem dieser Frameworks in Berührung gekommen. jQuery selbst wurde sogar von unseren Webentwicklungs-Dozenten empfohlen.

Spezielles: Hier ist uns aufgefallen, dass sehr grossen Wert auf Erreichbarkeit und Konformität gelegt wird. Diesbezüglich wird ebenfalls auf Sehbehinderte Rücksicht genommen, welche die Homepage nicht ohne zusätzliche Hilfsmittel lesen können. Hier werden diese Möglichkeiten explizit unterstützt und gefördert.

Ebenfalls sehr schön ist der anfangs erwähnte Theme-Editor, mit welchem auf ganz einfache Art und Weise ein eigenes Design erstellt werden kann. Nach Abschluss der Arbeit kann das fertige Design heruntergeladen werden, welches ein CSS, Bilder und eine Beispielseite enthält.

Wir bewerten hier mit dem Maximum von zwei Punkten.

4.9.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	9
Entwicklungsumgebung	8
Support	7
Dokumentation	8
Tutorials	10
Building	5
Testing	5
Verbreitung	10
Spezielles	2
Total	64

4.10. jQPad

4.10.1. Einleitung



jQPad ist ein iPad-Web-Framework, eine Mischung zwischen Template-Engine und JavaScript-Framework. Damit lassen sich auf einfache Weise Web-Seiten erstellen, welche mit iPad kompatibel sind. Es setzt auf jQuery und iScroll auf und lässt sehr viel Freiraum für eigene Designs. jQPad befindet sich praktisch seit Anfang 2011 im Beta-Stadium, da es lediglich von einer Person - Adrian Conney - entwickelt wird und steht unter der General Public License.

4.10.2. Wie funktioniert das Framework

Wie es für eine iPad App so typisch ist, bietet jQPad zwei Hauptbereiche. Eine Navigationsleiste auf der linken Seite und eine Inhaltsleiste auf der rechten Seite. Der Inhalt wird komplett mit AJAX geladen, sodass die Navigation bei jedem Seitenwechsel beibehalten werden kann. Geschrieben ist der Grossteil des Frameworks in JavaScript, im Idealfall muss aber nur das Template angepasst und allenfalls durch eigene Design-Aspekte ergänzt werden.



4.10.3. Evaluation

Einfachheit: Hier werden die Technologien HTML5, CSS und JavaScript verwendet, wobei die einzelnen Seitenaufrufe mittels AJAX geschehen. Die Anwendung der Technologien gestaltet sich wie bei einer herkömmlichen Web-Seite: Zuerst wird das Standard-HTML-Dokument auf den gewünschten Inhalt angepasst und allenfalls mit eigenen Styles via CSS ergänzt. Der Einsatz von eigenen JavaScript-Blöcken ist optional.

Um das Framework nun effizient einzusetzen, muss nach einer vorgeschriebenen Notation vorgegangen werden. Diese wird über das im Download-Paket mitgelieferte Read-Me sehr anschaulich und einfach erklärt. Schade ist allerdings, dass die Beispiel-Seite, welche auf die eigenen Bedürfnisse angepasst werden soll, bei uns nicht funktioniert hat. Diese wird zwar korrekt geladen, allerdings reagieren die Links nicht, was die Anwendung nicht gerade verständlich gestaltet.

Alles in Allem einfach gehalten, allerdings fehlt die Demo-Funktionalität auf herkömmlichen Computern. Dies bewerten wir mit genügend bis gut.

Entwicklungsumgebung: Da hier ein Web-Framework vorliegt, gibt es diesbezüglich keine Einschränkungen. Unsere Internetrecherche bezüglich Präferenzen haben keine konkreten Ergebnisse geliefert. Dies liegt daran, dass es noch keine grosse Verbreitung erreicht hat, was vermutlich auch dem Beta-Stadium zugeschrieben werden kann.

Da es jedoch auf jQuery basiert, wäre die Nutzung von Aptana oder Dreamweaver naheliegend, da diese IDEs bereits jQuery-Funktionalitäten anbieten.

Sehr grosse Flexibilität, was wir hier natürlich sehr gut bewerten.

Support: Zur Zeit gibt es lediglich die Möglichkeiten entweder per E-Mail direkt an Adrian Cooney oder über die Google-Groups zu gehen. Allerdings existieren bei letzterem lediglich zwei Posts, weshalb hier keine genaueren Angaben in Sachen Zuverlässigkeit gemacht werden können. Laut Angaben auf der Google-Code-Seite sollen nach erstem finalen Build mehr Leute an dem Projekt mitarbeiten, was die Support-Möglichkeiten verbessern soll.

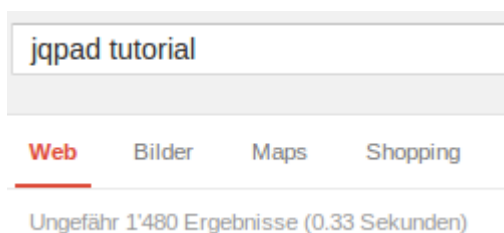
Inwiefern der Support via E-Mail funktioniert, können wir an dieser Stelle leider nicht beurteilen. Sehr schade, dass die Möglichkeiten so eingeschränkt sind. Eine mögliche Erklärung hierfür wäre die geringe Verbreitung. Hoffentlich wird sich dies in Zukunft verbessern, zum jetzigen Zeitpunkt müssen wir diesen Punkt als ungenügend bewerten.

Dokumentation: Diese ist extrem klein und rudimentär gehalten. Das heruntergeladene Packet enthält eine Readme-Datei im HTML-Format, welche ungefähr vier A4-Seiten entspricht. Zwar werden hier die einzelnen Bereiche mit Beispielen erklärt, allerdings können über die Demo nur etwa die Hälfte davon nachvollzogen werden.

Zurzeit ist die Dokumentation leider nur in dieser Form verfügbar, dies soll sich allerdings in Zukunft noch ändern.

Zusammenfassend können wir sagen, dass zwar die grundlegenden Funktionen erläutert werden, diese allerdings selbst durch try and error erprobt werden müssen. Dazu kommt die eingeschränkte Verfügbarkeit, was wir entsprechend mit ungenügend bewerten.

Tutorials: Google liefert hier 1'480 Ergebnisse, wobei weder die ersten fünf Links noch alle weiteren ein Tutorial beschreiben. An dieser Stelle muss mit der mitgelieferten Dokumentation vorliebgenommen werden.



Building: Wie bei anderen von uns evaluierten Web-Frameworks existiert auch hier kein nativer Build-Vorgang. Ob dieses Framework bereits mit anderen wie PhoneGap kombiniert wurde, konnten wir leider nicht eruieren.

Entsprechend müssen wir hier die Note genügend vergeben, da dieser Punkt hier nicht bewertet werden kann.

Testing: Sowohl die Dokumentation als auch das Git-Repository geben keinen Aufschluss auf irgendeine Form von automatisierten Tests. Hier kommen wieder die üblichen Kandidaten wie QUnit, Jasmine und Selenium ins Spiel. Ansonsten sind manuelle Tests unverzichtbar.

Fazit: Bei jQPad können die gängigen Testlibraries für Web-Applikationen verwendet werden, allerdings wäre ein Hinweis in der Dokumentation wünschenswert. Die Bewertung fällt hier durchschnittlich aus.

Verbreitung: Google liefert 3'030 Resultate, was extrem wenig sind, wenn man den Bedarf an iPad-Applikationen betrachtet. Wir können uns dies nur damit erklären, dass selbst nach zwei Jahren Entwicklung immer noch eine Beta-Version vorliegt. Bekanntermassen werden diese sehr selten für produktive Systeme eingesetzt, vor allem, wenn kein offizieller und schneller Support verfügbar ist.

4.10.4. Finale Bewertung

Kriterium	Punkte
Einfachheit	8
Entwicklungsumgebung	8
Support	3
Dokumentation	3
Tutorials	1
Building	5
Testing	5
Verbreitung	1
Spezielles	0
Total	34

4.11. Gegenüberstellung der Frameworks

Name	Appcelerator Titanium	Sproutcore Touch	PhoneGap	Rhodes	iUI
Einfachheit	4	7	8	6	8
Entwicklungsumgebung	6	6	7	7	3
Support	4	4	9	6	4
Dokumentation	7	9	8	6	8
Tutorials	4	6	9	3	8
Building	5	5	10	9	5
Testing	9	9	6	7	6
Verbreitung	7	4	9	3	6
Spezielles	2	0	2	1	0
Total	48	50	68	48	48

Name	iWebKit	Sencha Touch	XUI	jQPad	jQuery Mobile
Einfachheit	7	6	9	8	9
Entwicklungsumgebung	8	8	9	8	8
Support	7	7	5	3	7
Dokumentation	7	8	5	3	8
Tutorials	3	7	0	1	10
Building	5	6	5	5	5
Testing	5	6	4	5	5
Verbreitung	2	8	5	1	10
Spezielles	0	2	0	0	2
Total	44	58	42	34	64

4.12. Erklärungen zu Tutorials und Verbreitung

Wie vor allem die Google-Suche zum Bewertungskriterium „Verbreitung“ gezeigt hat, ist es bei einigen Frameworks schwierig, aufgrund der Namensgebung vernünftige Google-Resultate zu erzielen. Dies vor allem bei den Frameworks iUI und XUI, welche jeweils Abkürzungen für diverse Dinge darstellen. Da ein so gründliches Ausfiltern nicht möglich ist in den Suchresultaten, haben wir uns entschlossen, für diese beiden Frameworks eine Einschätzung vorzunehmen. Da wir jeweils für undifferenzierte Resultate eine 5 vergeben haben, geben wir an XUI 5 Punkte sowie an iUI 6 Punkte in der Bewertung. Dies entspricht wohl auch der realen Verbreitung der Frameworks nicht schlecht.

Im Bewertungspunkt Tutorial vergeben wir Punkte nach Google-Treffern, jedoch wird noch ein Punkt abgezogen, falls die Qualität der Tutorials in den ersten fünf Treffern nicht genügend ist. Die Erklärung wird in folgender Tabelle dargestellt:

Framework	Bewertung	Abzug	Total	Begründung
Appcelerator Titanium	78'400 = 5P	-1P	4P	Qualität der Tutorials ist dürftig, vieles doppelt
Sproutcore Touch	257'000 = 6P	0P	6P	Qualität in Ordnung
PhoneGap	1'090'000 = 9P	0P	9P	Sehr gute Qualität
Rhodes	44'000 = 4P	-1P	3P	Begrenzte Anzahl, sehr komplex
iUI	400'000 = 8P	0P	8P	Gute Qualität, viel brauchbares
iWebKit	37'400 = 3P	0P	3P	Gute Qualität, gut ausgearbeitet, Videos
Sencha Touch	387'000 = 7P	0P	7P	Sehr gute Qualität
XUI	22 = 1P	-1P	0P	Praktisch nicht vorhanden
jQPad	1'480 = 2P	-1P	1P	Nur Dokumentation vorhanden
jQuery Mobile	1'680'000 = 10P	0P	10P	Gute Qualität, angemessen

4.13. Fazit der Evaluation

Die Evaluation hat einen klaren Sieger hervorgebracht: PhoneGap mit 68 Punkten von maximal 82 möglichen. Die intuitive Handhabung sowie das direkte Building seien hier nur im Speziellen erwähnt, denn durchwegs zeigt PhoneGap in allen Bereichen seine Stärke.

Auf dem zweiten Platz befindet sich, mit einem knappen Abstand von 4 Punkten, jQuery Mobile. Ausschlaggebend bei diesem Framework sind sicher die langjährige Erfahrung, welche das Team von jQuery mit sich bringt, sowie auch die Community, welche dieses System mitträgt und laufend mitarbeitet, jQuery Mobile besser zu machen.

Im Verlauf der Evaluation hat sich vor allem eines gezeigt: Es gibt eine Fülle an Möglichkeiten, für Mobile-Devices zu programmieren. Von einfachen HTML-Seiten, welche im Browser angezeigt werden, bis zu nativen Apps, welche jeweils vorkompiliert für das jeweilige Touch-Betriebssystem entstehen. Die Bandbreite der Technologien ist dabei so gross, dass für praktisch Jedermann ein Framework existiert, wo er ansetzen und sich grundsätzlich in die Materie einarbeiten kann. Egal ob aus der Java-, der Web- oder der C-Welt kommend.

Die Möglichkeiten in der Ausführung werden ebenfalls immer Interessanter: Benutzung der Kamera, Bewegungssensoren und Touch-Gesten, um nur einige Beispiele zu nennen. Diese neuen Möglichkeiten plattformübergreifend nutzen zu können, wird sich für die Frameworks zu einer zentralen Frage entwickeln.

Obwohl es für einige Leute trocken anhört, so hat uns das Einarbeiten in diese 10 Frameworks doch auch einige schöne Momente beschert, sei es ein Lachen oder ein Staunen. Die Arbeit war intensiv und wurde von uns teilweise unterschätzt in Umfang und Arbeit. Dies zeigt sich auch im Stundenraster wo man sieht, dass die geplanten Soll-Zeiten doch einige Male überschritten wurden. Jedoch ist dabei eine grobe Zusammenfassung der jetzigen Situation in der App-Entwicklungsszene entstanden, welche vielleicht dem einen oder anderen eine Entscheidung abnehmen kann, ohne sich stundenlang durch das Internet zu wühlen.

5. Projekt Lupen-App

5.1. Umgebung des Projekts

5.1.1. Wahl des Frameworks

Wir haben uns entschieden, entgegen der ursprünglichen Planung, nicht zweimal eine Lupen-App zu schreiben, sondern mit 2 Frameworks einmal eine App zu schreiben. Dies hat hauptsächlich den Grund, dass das zweitplatzierte jQuery Mobile kein natives Entwickeln ermöglicht. Deshalb wird jQuery Mobile als Ergänzung zum Evaluationssieger PhoneGap verwendet, um eine gute Darstellung zu erreichen.

Diese zwei Frameworks ergänzen sich jedoch gut in Funktion und Umgang, weshalb wir keine Probleme erwarten in der Umsetzung. Es wird auch in diversen anderen Projekten, welche im Netz zu finden sind, PhoneGap mit jQuery ergänzt. Die grosse Verbreitung von jQuery Mobile hat sich auch darin niedergeschlagen, dass PhoneGap dieses voll unterstützt im Building-Prozess.

5.1.2. Installation der Frameworks

Im Voraus möchten wir hier anmerken, dass wir die Installation für unsere Geräte optimiert haben. Wie unter dem Punkt Testgeräteerwähnt, besitzen wir beide Android-Devices, somit haben wir darauf verzichtet, Umgebungen für andere Zielgeräte einzurichten. Dies erübrigt sich wegen des sehr guten Online-Builds von PhoneGap sowieso.

Um PhoneGap installieren zu können, benötigt man eine Entwicklungsumgebung, das Android SDK und natürlich PhoneGap (aka Cordova). Wir haben uns hier für das „ADT-Bundle“ entschieden, dieses enthält eine modifizierte Eclipse-Version inklusive dem Android SDK. Somit muss die gewünschte Entwicklungsumgebung nicht noch manuell durch die Installation von benötigten Plug-Ins erweitert werden. Dies gestaltete sich sehr einfach: Herunterladen, entpacken, starten. Ab diesem Zeitpunkt wären wir bereits in der Lage gewesen, eine native Android-App zu entwickeln.

Als letztes kommt nun die PhoneGap-Library ins Spiel. Damit lässt sich mit einem Scaffolding-Verfahren das Grundgerüst für eine PhoneGap-App erstellen. Dies geschieht mit dem „Create“-Befehl:

```
oliver@mtwords ~/git-repos/hand-held $ ./create /home/oliver/git-repos/hand-held/project ch.zhaw.loupe loupe
```

Als Parameter werden Projekt-Ordner, Package-Name und Projekt-Name angegeben.

Nun wird die gesamte Projekt-Grundstruktur automatisch erstellt. Hier sind wir auf die ersten Probleme gestossen: Der Befehl schlägt fehl, wenn nicht alle Grundvoraussetzungen erfüllt sind. Dies können fehlende Einträge in der Path-Variable sein oder auch wenn das Build-Tool „Ant“ nicht installiert ist. Leider wurden wir hier lediglich mit der Fehlermeldung „An Error occurred: Project could not be created.“ darauf aufmerksam gemacht. Hier hat sich jedoch sehr schön gezeigt, dass der Online-Support der Community tatsächlich von guter Qualität ist. Wir mussten nicht lange googeln, und schon hatten wir einen entsprechenden Forumseintrag gefunden, welcher beschreibt, was alles als Voraussetzung verlangt wird.

Nachdem wir dann die Path-Variable um den Ort des Android-SDKs ergänzt und Ant installiert hatten, konnten wir das Projekt erfolgreich erstellen.

Anhand der erstellten Projekt-Struktur sieht man schnell, dass PhoneGap tatsächlich ein natives Grundgerüst für die App bereitstellt. In unserem Fall ist dies eine Java-Klasse, welche dann die „index.html“ als URL lädt. Dies ist dann auch der Einstiegspunkt für unsere Lupen-App. Auf jeden Fall lässt sich dies bereits über den Emulator starten.

jQuery Mobile mussten wir nicht im Sinne einer Installation hinzufügen, sondern lediglich die JavaScript-Dateien und eine CSS-Datei herunterladen und in unserem index.html einbinden. Dies geschieht wie gewohnt mittels den „script“- und „link“-Tags.

Obwohl hier nicht alles reibungslos ablief, war dies dennoch ohne grosse Mühe und mit wenig Zeitaufwand möglich, eine Entwicklungsumgebung inklusive „Hello World“-App zu erstellen, auf welcher wir anschliessend unsere Lupen-App aufgebaut haben. In diesem Sinne hat sich unsere Evaluation vollumfänglich bestätigt.

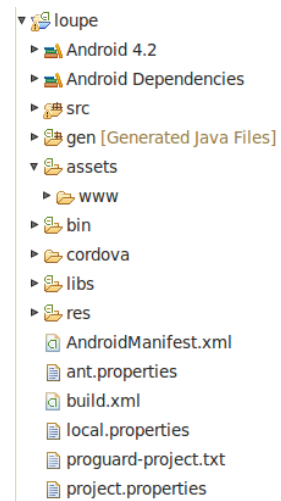


Abbildung 1 PhoneGap-Projektstruktur

5.1.3. Tools

Hier werden alle weiteren verwendeten Tools aufgeführt, welche wir im Rahmen dieses Projekts verwendet haben.

Als Entwicklungsumgebung haben wir, wie erwähnt, ein modifiziertes Eclipse verwendet, welches mit dem ADT-Bundle mitgeliefert wird und ebenfalls das Android-SDK enthält.

Für den Lupen-Effekt unserer App haben wir loupe.js (<http://www.netzgesta.de/loupe/>) eingesetzt. Dies machte bei uns einen sehr schönen visuellen Eindruck und liess sich auch relativ einfach in unsere App integrieren. Alternativ wären auch Plugins für jQuery möglich gewesen, allerdings hat sich hier schlussendlich das Design von loupe.js durchsetzen können.

Wie bereits gesagt, setzen wir jQuery Mobile für das Layout ein. Der Grund dafür ist, dass damit auf einfache und schnelle Weise ein App-Look erstellt werden kann. Hier hat sich unsere Evaluation ebenfalls in der Einfachheit und der Dokumentation bestätigt.

5.2. Entwicklung

5.2.1. Einarbeitung

Als wir mit der Entwicklung starteten, haben wir uns zuerst einen Überblick über die benötigten Komponenten verschafft und diese anhand von Beispielen angewendet. Für unsere Lupen-App benötigten wir folgende Kenntnisse:

- Verwendung von loupe.js
- Verwendung der Kamera via PhoneGap
- Verwendung von jQuery Mobile

Für die Einarbeitung dieser einzelnen Komponenten haben wir uns etwas mehr Zeit genommen, als dies ursprünglich geplant war. Allerdings hat sich dies anschliessend beim Zusammenführen ausgezahlt, da das Grundgerüst innerhalb kürzester Zeit fertiggestellt werden konnte und somit am Schluss noch etwas Zeit für Layouting und Korrekturen übrig blieb.

5.2.2. Programmierung

Der Programmcode, den wir für unsere App selbst beigesteuert haben, ist im Grunde ziemlich überschaubar, wenn man das Resultat anschaut. Jedoch lag – wie gesagt – der grosse Aufwand in der Einarbeitung. Wir möchten im Folgenden auf die Kernfunktionalitäten eingehen, welche die verschiedenen Komponenten betreffen. Für den Rest wie CSS-Definitionen usw. möchten wir auf den Code verweisen. Dieser befindet sich ausschliesslich im Ordner `project/assets/www`.

5.2.2.1. Layout

Das Layout haben wir mit jQuery Mobile realisiert. Dies gestaltete sich – nicht zuletzt auf Grund der sehr guten Dokumentation – relativ einfach:

```
<div id="page" data-role="page" data-theme="b" data-content-theme="a">
  <div data-role="header" data-theme="b">
    ...
  </div>

  <div id="photoWrapper" data-role="content">
    ...
  </div>
</div>
```

Die Magie von jQuery Mobile kommt hier bei den „data-...“-Attributen ins Spiel. Diese werden interpretiert und entsprechend designed. In diesem Code-Snipped wird durch das Attribut `data-role="header"` (im Folgenden als Header bezeichnet) ein dominanter Balken dargestellt. Das Attribut `data-role="content"` (im Folgenden als Content bezeichnet) bewirkt, dass unterhalb des Headers Platz für den eigentlichen Inhalt reserviert wird. Über die Attribute `data-theme="..."` können die von jQuery Mobile vordefinierten Themes verwendet werden, welche selbstverständlich für die verschiedenen Layout-Bereiche unterschiedlich gewählt werden können.

5.2.2.2. Kamera-Ansteuerung

Die Kamera-Ansteuerung von PhoneGap gelingt mit wenigen Zeilen JavaScript. Zuerst brauchen wir eine Funktion für das Aufnehmen des Fotos:

```
function capturePhoto() {
  // Take picture using device camera and retrieve image as base64-encoded string
  navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 50,
    destinationType: destinationType.DATA_URL, correctOrientation: true });
}
```

Auf den ersten Blick sehr einfach gehalten, wir brauchen allerdings noch zwei weitere Funktionen: Für das Auslesen des Fotos und für den Fehlerfall. Diese müssen der Funktion `navigator.camera.getPicture(...)` als Parameter mitgegeben werden.

Auslesen des Fotos:

```
function onPhotoDataSuccess(imageData) {
  // Get image handle
  var smallImage = document.getElementById('img');
  // Unhide image elements
  smallImage.style.display = 'block';
  // Show the captured photo
  // The inline CSS rules are used to resize the image
  smallImage.src = "data:image/jpeg;base64," + imageData;
}
```

Hier wird das frisch gemachte Foto als Parameter entgegengenommen und anschliessend einem HTML-IMG als Source gesetzt. Somit wird nach der Aufnahme das Bild schon in die App eingebettet.

Fehlerfall:

```
function onFail(message) {
  alert('Failed because: ' + message);
}
```

Dies haben wir sehr einfach gehalten. Es wird eine Meldung mit der entsprechenden Fehlermeldung angezeigt, welche wir von PhoneGap erhalten.

Nun müssen wir die Funktion noch aufrufen. Dies haben wir über einen Button realisiert, welcher sich im Header befindet:

```
<button id="photoCapturer" data-theme="b" onclick="capturePhoto();" >
  Capture Photo
</button>
```

Wie man sieht wird die Hauptfunktion `capturePhoto()` aufgerufen, welche die übrigen Funktionen dann als Parameter an PhoneGap weiterreicht.

5.2.2.3. Verwendung von loupe.js

Mit den bisherigen Werkzeugen können wir bereits ein App-Layout erstellen und ein Foto über die Kamera aufnehmen. Was wir jetzt noch brauchen ist die Lupe, welche wir anschliessend frei über das Foto bewegen können, um schwererkennbare Bereiche genau betrachten zu können.

Um dies zu erreichen, muss ein HTML-IMG-Element in einem HTML-DIV-Element eingebettet werden. Dies wird von der Library so vorgeschrieben, was unseren Ansprüchen aber völlig genügt. Wir haben dies direkt im Content eingebettet:

```
<div id="photoWrapper" data-role="content">
  <img id="img" onload="loupe.add(this);" src="" />
</div>
```

Zentral ist hier das onload-Attribut, welches mittels dem Aufruf `loupe.add(this)` die Lupe auf dem HTML-IMG registriert.

Nun hat man bereits eine voll funktionsfähige Lupen-App, welche allerdings noch durch ein paar CSS-Eigenschaften ergänzt werden sollte, um das Ganze noch abzurunden. Bei uns sieht das Endresultat so aus:

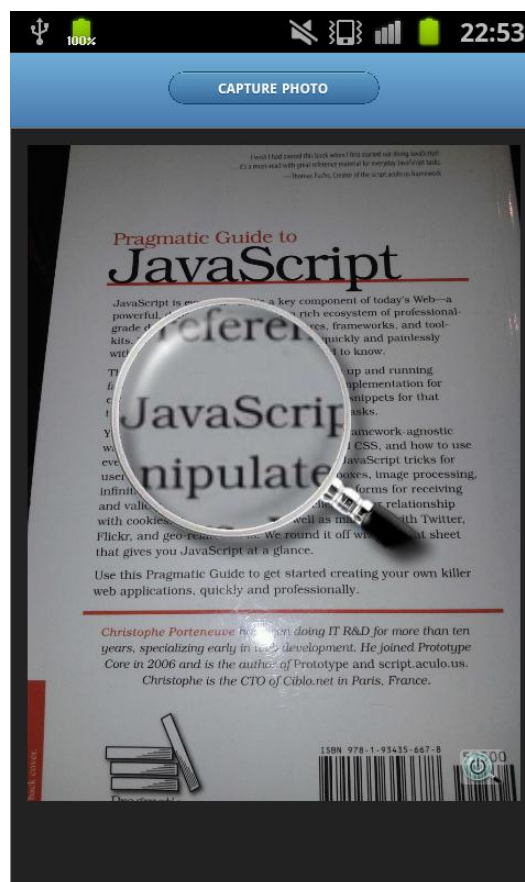


Abbildung 2 Die Lupen-App

5.2.2.4. Verbesserungsmöglichkeiten

Wir haben hier mit wenig Programmieraufwand ein sehr schönes Resultat erzielen können. Das Ziel haben wir sicherlich erreicht, allerdings hätten wir gerne noch ein paar Verbesserungen eingebracht, welche wir leider aus Zeitgründen nicht mehr machen konnten:

1. Wenn ein Foto im Hochformat aufgenommen wurde und anschliessend in der App das Device auf Querformat gedreht wird, arbeitet die Lupe nicht korrekt. Es wird intern immer noch mit dem Hochformat gearbeitet, was dazu führt, dass die Vergrößerung am falschen Ort angezeigt wird und demzufolge auch nicht über das ganze Foto genutzt werden kann. Dies liegt aber hauptsächlich an der loupe.js-Library, welche offenbar nicht auf ein dynamisches Layout ausgelegt wurde.
2. Es wäre schön, wenn man bereits aufgezeichnete Fotos über die Galerie laden könnte. Dies würde nochmals etwas Programmieraufwand bedeuten, wäre aber ein sehr nettes Feature, was wir in Zukunft noch einbauen könnten.

5.2.3. App Debugging / Testing

Wir haben uns bei der Entwicklung hauptsächlich auf das Debugging konzentriert und das automatisierte Testing beiseite gelassen. Dies kommt daher, dass wir in diesen Technologien zu wenig Erfahrung haben und uns daher nicht durch diesen Punkt aufhalten lassen wollten.

Aber PhoneGap bietet in Punkto Debugging ein sehr angenehmes Werkzeug, was wir auch schon bei der Evaluation erwähnt hatten: debug.PhoneGap.com. Dieses kann entweder während der Entwicklung oder sogar über den Online-Build eingeschaltet werden.

Entwicklung: Während der Entwicklungsphase, kann ein JavaScript-File über das index.html eingebunden werden. Danach kann über einen WebKit-Browser wie beispielsweise Chrome oder Safari live auf die App zugegriffen werden.

Online-Build: Wenn der Online-Build genutzt wird, kann vor dem Builden der Debug-Modus eingeschaltet werden. Hier wird das Javascript-File automatisch während dem Build-Prozess eingebunden. Nach dem erfolgreichen Build kann dann direkt auf der Build-Seite die Debug-Seite geöffnet werden.

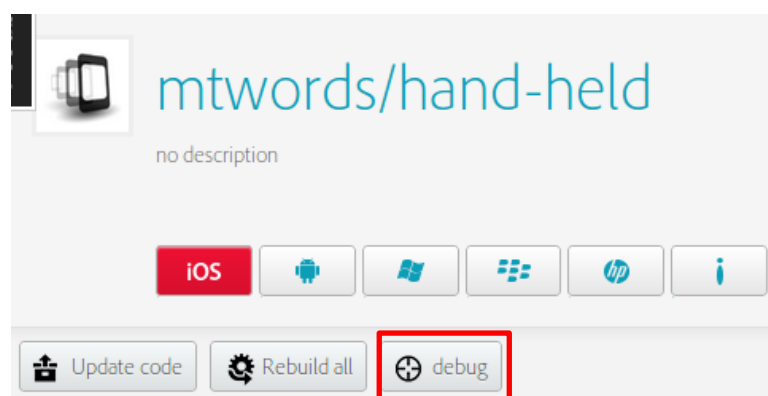


Abbildung 3 Debug über Build-Seite starten

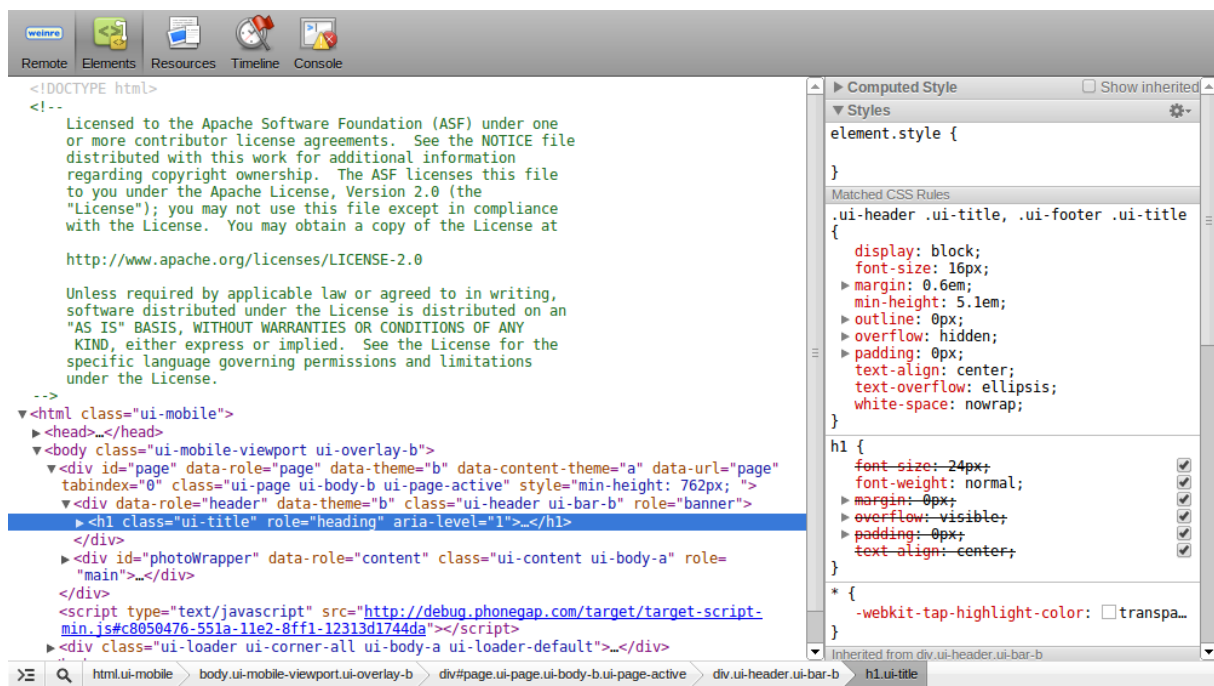


Abbildung 4 Elements-Ansicht der Debug-Seite

Oben werden fünf Register angezeigt: Remote, Elements, Resources, Timeline und Console. Über das Register „Remote“ werden alle Geräte angezeigt, welche zurzeit die App geöffnet haben. Wir hatten während der Entwicklungsphase jeweils nur eines aktiv. Über das Register „Elements“ wird der Code der aktuell geöffneten App-Seite angezeigt. Diese ist sehr stark an Browser-Entwicklungstools wie beispielsweise FireBug angelehnt und man kann direkt die Elemente inspizieren und editieren. Wenn man beispielsweise den App-Header betrachtet, sieht dies folgendermassen aus:

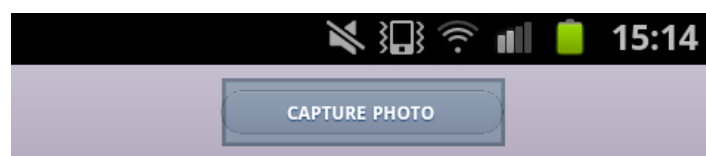


Abbildung 5 Anzeige bei Inspiziertem Header

Man sieht, dass der Button für das Schiessen eines Photos umrandet wurde. Dies erfolgt automatisch auf dem Gerät, wenn mit der Maus über ein Element auf der Debug-Seite gefahren wird. Man kann aber wie gesagt die Elemente direkt editieren. Im Folgenden haben wir die CSS-Eigenschaft „min-height“ des gesamten Headers verändert:

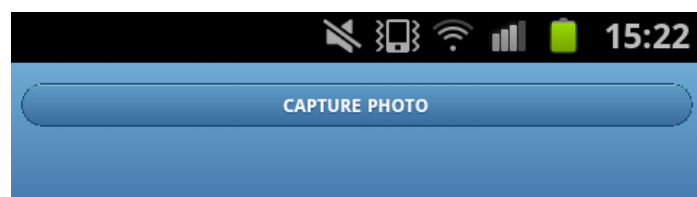


Abbildung 6 Anzeige nach angepasster Header-Grösse

Auf dem Gerät wurde die Änderung direkt angewendet um man sieht, dass durch die Anpassung der Höhe der Button „Capture Photo“ ebenfalls vergrößert wurde. Dieses Verhalten ist nicht zwingend erwünscht, also kann man weitere Änderungen vornehmen, bis alles so aussieht, wie man dies will.

Im Rahmen unserer Entwicklung haben wir vor allem mit dem Register „Elements“ gearbeitet. Die übrigen Register haben bei unserer App keinen Output geliefert, wären aber vermutlich bei grösseren Applikationen sehr nützlich, bei denen viele PhoneGap-Funktionen angesteuert werden.

Dieses Tool hat uns insbesondere während der Entwicklung sehr geholfen, der einzige Nachteil ist, dass es relativ langsam ist. Dies verwundert aber nicht, da die gesamten Informationen live über das Internet ausgetauscht werden müssen.

5.2.4. App Deployment

Das Builden sowie das Deployment der App gestaltet sich sehr einfach. Einzige Voraussetzung ist, dass das Device über das System erreichbar ist. Bei Windows geschieht dies über entsprechende Treiber, bei Linux muss die entsprechende Android Rule hinzugefügt werden und bei Mac OSX funktioniert dies sogar ohne jegliche Eingriffe. Genaue Installationsanleitungen können über Android.com bezogen werden.

Bei PhoneGap existieren grundsätzlich zwei Möglichkeiten:

Lokal: Der Build und das Deployment können direkt aus Eclipse gemacht werden, indem eine „Run Configuration“ eingerichtet und gestartet wird. Dort kann bequem das angeschlossene Device ausgewählt und auf „OK“ geklickt werden. Voilà! Das Android-Paket wird gebuildet und direkt auf dem Device installiert und gestartet.

Online: Über den Link build.PhoneGap.com kann dies mühelos durchgeführt werden. Bei *einer* App kann dies gratis durchgeführt werden, ansonsten muss eine Lizenz erworben werden, mit welcher bis zu 25 Apps permanent gebuildet werden können.

Um den Build durchführen zu können hat man die Möglichkeit ein Git-Repository oder ein Zip-File anzugeben. Wir haben hier mit unserem Repository gearbeitet. Danach kann es auch schon losgehen: PhoneGap liest das Repository ein und erkennt selbstständig, welche Files für den Build miteingeschlossen werden müssen. Dies kann aber auch gefährlich sein, wenn mehrere Web-Anwendungen im Repository vorhanden sind, wird automatisch diejenige verwendet, welche zuerst gefunden wurde! Da man jedoch normalerweise pro Repository nur eine Web-Anwendung hat, fällt dies kaum ins Gewicht.

Der Build-Vorgang dauerte bei uns etwa eine Minute zeigte anschliessend folgendes Bild:

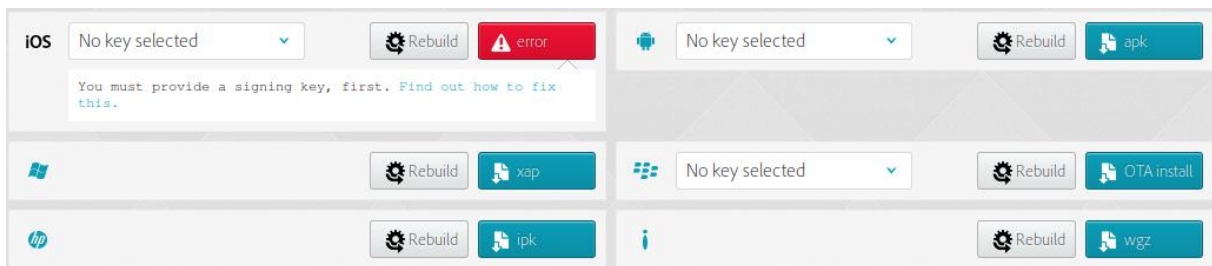


Abbildung 7 Übersicht des Online-Builds

Wie man sieht sind die Pakete für Android, Windows Mobile, Blackberry, Symbian und WebOS erfolgreich verlaufen. Einzig für iOS ist der Build fehlgeschlagen, da wir keine Signierung in unserem Projekt verfügbar hatten. Diesem Problem haben wir uns aber nicht näher gewidmet, da wir sowieso nur Android-Devices zur Verfügung haben.

Das entsprechende Paket kann nun heruntergeladen und auf dem Gerät installiert werden. Hierzu muss eine entsprechende Software vorhanden sein, welche diesen Vorgang ermöglicht. Nach der Installation findet man die App ganz gewohnt über die Anwendungsübersicht und kann diese starten. Bei uns war dies auf Anhieb erfolgreich, was wiederum unsere Erkenntnisse aus der Evaluationsphase bestätigt. Einziger Wermutstropfen: Die App enthält das PhoneGap-Logo als App-Icon. Dies könnte aber manuell oder über ein Build-Tool wie Ant übersteuert werden.

5.2.5. Testgeräte

Zum Testen der programmierten App stehen 3 reale Geräte bereit sowie virtuelle Geräte:

- 1 x Samsung Galaxy Tab II
- 1 x Sony Experia T
- 1 x Samsung Galaxy S Advance
- Virtuelle Geräte des Android SDK, mit denen verschiedene Android-Modelle simuliert werden können

Hauptsächlich wird im emulierten Gerät des Android SDK getestet. Dieses hat aber eine eher schlechte Laufzeit (langsam, mühsam) weshalb freigestellt wird, ob auch auf den physischen Geräten getestet wird in der Anfangsphase (normalerweise eher gegen Ende eines Projekts vorgesehen).

6. Schlusswort

Wir durften uns bei dieser Arbeit dem spannenden Thema der Entwicklung von mobilen Apps widmen, was uns sehr grossen Spass gemacht hat. Das interessante Thema von plattformübergreifenden Methoden und Werkzeugen, hat uns beide schon immer interessiert und wir konnten uns hier mit sehr vielen Inputs und Möglichkeiten beschäftigen, was uns schlussendlich auch noch zu einem sehr zufriedenstellenden Resultat – der Lupen-App – geführt hat.

Des Weiteren haben wir verschiedene Aspekte einer wissenschaftlichen Arbeit aufgegriffen: Planung, Evaluation, Entscheidung, Umsetzung und Dokumentation. In all diesen Bereichen haben wir eine Menge gelernt, was wir auf dem weiteren Weg mitnehmen können. Eine Planung aufzustellen ist nicht einfach, vor allem der Evaluationsprozess, welcher in unserem Projekt sehr umfangreich ausgefallen ist, gestaltete sich sehr schwierig. Dies hat auch dazu geführt, dass wir länger dafür gebraucht haben, als wir eigentlich anfangs eingeplant hatten. Für uns ist hier wichtig, dass wir gelernt haben, Dinge nicht zu unterschätzen. Eventuell wäre bei einer so grossen Auswahl an Frameworks auch Sinnvoll gewesen, eine Vorauswahl zu machen oder den Kriterienkatalog etwas schlanker zu halten.

Abschliessend können wir sagen, dass sich jeder, der eine eigene App schreiben möchte, sich unbedingt mit PhoneGap und jQuery Mobile auseinander setzen sollte. Insbesondere dann, wenn die App auf unterschiedlichen Plattformen funktionieren muss, ist dies unserer Meinung nach die beste Wahl.

7. Github Repository

Da wir zum Zwecke der Zusammenarbeit über grosse Distanzen ein Github Repository betrieben haben, verzichteten wir auf die Angabe des gesamten Codes in der Dokumentation und stellen stattdessen den Code direkt über das GitHub Repository bereit. Dieses ist zu finden unter:

<https://github.com/mtwords/hand-held>

Dort finden sich dieses Dokument, die Präsentation sowie alle relevanten Codeteile und Materialien. Erklärung zur Ordner-Struktur:

- Build: Hier befindet sich der Android-Build, der heruntergeladen und installiert werden kann. Des Weiteren werden noch die Builds für Blackberry, Symbian, WebOS sowie Windows Phone bereitgestellt. Dies der Vollständigkeit halber, getestet und optimiert wurde ausschliesslich für Android.
- Documents: Enthält dieses Dokument, die Präsentation, den Teaser sowie den Projektplan
- Media: Mediendateien wie Bilder oder ähnliches, welche im Verlaufe des Projekts benötigt wurden.
- Project: Der Code des Lupen-Apps befindet sich hier. Der Hauptteil findet sich im Ordner `..\assets\www\`
- Tools: Hier befindet sich die Grundlage des Projekts, das Loupe.js

8. Quellen

Appcelerator:

<http://www.appcelerator.com/>

<https://github.com/appcelerator/KitchenSink/tree/master/Resources>

<http://www.d-mueller.de/blog/appentwicklung-webtechniken-mit-appcelerator-titanium/>

<http://www.universalmind.com/mindshare/entry/mobile-html5-PhoneGap-vs-appcelerator-titanium>

<http://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerator-titanium/>

<https://github.com/appcelerator/KitchenSink/tree/master/Resources>

Sencha:

<http://www.sencha.com/products/touch/>

<http://miamicoder.com/>

Sproutcore:

<http://www.sproutcore.com/>

<http://opensource.org/licenses/MIT>

<http://ajaxian.com/archives/sproutcore-2010>

http://broadcastingadam.com/2011/04/sproutcore_login_tutorial/

<http://www.veebsbraindump.com/2010/10/sproutcore-crud-tutorial-using-sc-tableview/>

<https://groups.google.com/forum/?fromgroups=#!topic/sproutcore/EFoUcd5fC2w>

<http://www.appnovation.com/how-access-functions-user-created-android-sproutcore-without-PhoneGap-plugins>

PhoneGap:

<http://www.PhoneGap.com/>

<http://dev.appmobi.com/?q=node/153>

<http://www.adobe.com>

<http://build.PhoneGap.com/>

<http://debug.PhoneGap.com/>

<http://mobile.tutsplus.com/tutorials/PhoneGap/creating-an-android-hello-world-application-with-PhoneGap/>

<https://groups.google.com/forum/?fromgroups#!forum/PhoneGap>

Motorola Rhodes:

<http://rhomobile.com/products/rhodes/>

<http://stackoverflow.com/questions/3689753/can-anyone-suggest-me-good-links-or-tutorial-for-rhomobile>

<https://app.rhohub.com/>

<https://developer.motorolasolutions.com/welcome>

iUI:

<http://code.google.com/p/iui/>

<http://www.iui-js.org/>

<http://www.iui-js.org/>

iWebkit:

<http://iwebkit.net/>

<http://www.grails.org/iwebkit+Plugin>

<http://drupal.org/project/iwebkit>

<http://gigaom.com/apple/how-to-create-an-iphone-web-app/>

<http://www.youtube.com/watch?v=kFfBmfn2a2A>

<http://doiphone.com/2010/02/how-to-create-iphone-web-apps-iwebkit/>

<http://www.apple.com/webapps/productivity/iwebkit5.html>

XUI:

<http://xuijs.com/>

<http://davidbcalhoun.com/tag/xui>

<http://stackoverflow.com/search?q=xui&submit=search>

<https://github.com/xui/xui>

jQPad:

<http://code.google.com/p/jqpad/>

<https://github.com/dunxrion/jQPad>

www.youtube.com/user/jqpad

www.youtube.com/watch?v=nJ9fRLSgbzo

bput4all.wordpress.com/tag/jqpad/

<http://css.turbolinux.org/tag/jqpad>

jQuery Mobile:

<http://jquerymobile.com/resources/>

<http://www.jqmgallery.com/jquery-mobile-tutorials/>

<http://www.underworldmagazines.com/11-cool-jquery-mobile-tutorials/>

<http://spyrestudios.com/beginners-tutorial-coding-web-apps-with-jquery-mobile/>

<http://leo.4eyes.ch/2012/01/PhoneGap-und-jquery-mobile/>

<https://groups.google.com/forum/?fromgroups=#!search/jquerymobile>

<http://www.peterbe.com/plog/qunit-jquery-mobile-in-full-swing>

Android:

<http://developer.android.com/tools/device.html>

9. Abbildungsverzeichnis

Abbildung 1 PhoneGap-Projektstruktur	52
Abbildung 2 Die Lupen-App.....	55
Abbildung 3 Debug über Build-Seite starten.....	56
Abbildung 4 Elements-Ansicht der Debug-Seite	57
Abbildung 5 Anzeige bei Inspiziertem Header	57
Abbildung 6 Anzeige nach angepasster Header-Grösse	57
Abbildung 7 Übersicht des Online-Builds	59