Ralph Steyer

Cordova

Entwicklung plattformneutraler Apps

Springer Vieweg

Cordova

Ralph Steyer

Cordova

Entwicklung plattformneutraler Apps

Springer Vieweg

Ralph Steyer
Bodenheim
Deutschland

ISBN 978-3-658-16723-3 ISBN 978-3-658-16724-0 (eBook)
DOI 10.1007/978-3-658-16724-0

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen National-bibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist Teil von Springer Nature
Die eingetragene Gesellschaft ist Springer Fachmedien Wiesbaden GmbH
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Strasse 46, 65189 Wiesbaden, Germany

Vorwort

Als ich im Jahr 2012 erstmals ein Buch zur plattformneutralen Programmierung von Apps geschrieben hatte, hatte ich mir Gedanken gemacht, wie stark sich die Nutzung des Internets über die Jahre verändert hatte. Ganz natürlich ging ich zu der Zeit via einer USB-Tethering-App und meinem mobilen WLAN-Hotspot über mein Smartphone online und hatte über dem damaligen Vorwort gebrütet. Dabei hatte ich am Waldrand unter einem Sonnenschirm mit meinem Notebook im Garten gesessen. Ganz selbstverständlich. Nur wenige Jahren zuvor saß ich beim Schreiben anderer Bücher ebenso oft dort (zumindest wenn es warm und schön war), aber **offline**. Wenn ich damals online sein wollte oder musste, musste ich ins Büro oder zumindest in die Nähe meines festen WLAN-Routers gehen. Diese Zeiten sind lange vergangen und gerade hat mir mein Provider mitgeteilt, dass er die Geschwindigkeit meines mobilen Anschlusses kostenlos verdoppelt.

Warum erzähle ich Ihnen das? Ich will Sie neidisch machen auf meine manchmal ungewöhnlich schönen Arbeitsbedingungen;-). Aber im Ernst – wie bei vielen Personen ist mein Smartphone mittlerweile ein Schweizer Taschenmesser des täglichen Lebens. Der Nutzen liegt weit über der einfachen Telefonie. Von der Verwaltung meiner Kontakte, Notizen und Termine über die Wiedergabe von Musik und Videos, zur Informationsbeschaffung jeder Art samt Onlinegeschäften, Navigationsmöglichkeiten, der ständig verfügbaren Kamera bis hin zu kleinen praktischen Tools wie einer Taschenlampen-App, eines Kompass oder einer Wasserwaage reichen die Anwendungen. Die diversen Spiele sind nicht zu vergessen.

Nun muss ich aber zugeben, dass ich mich lange dagegen gewehrt habe, die Organisation der täglichen Anforderungen meines Lebens solchen mobilen Geräten zu übertragen. Denn eigentlich bin ich – technisch – konservativ. Ich springe fast nie einfach auf einen Trend auf, nur weil er neu ist. Ehrlich gesagt muss ich eingestehen, dass häufig der Spruch „Was der Bauer nicht kennt, frisst er nicht!“ auch für mich gelten könnte. Ich bin jedoch lernfähig. Nur muss ich immer erst von dem Nutzen einer neuen Sache überzeugt sein und vor allen Dingen, dass das Neue auch gut und sinnvoll ist. Entweder überzeugt mich irgendjemand in meinem Umfeld davon oder ich bekehre mich selbst. Letzteres dauert meist länger, ist aber der Regelfall. Wenn ich dann aber von einer Neuentdeckung überzeugt bin, dann beschäftige ich mich intensiv mit einer Sache und vertrete deren Vorteile

später auch konsequent. Oft sogar ganz entgegen meines Standpunktes, den ich vor der Konvertierung vertreten habe.

Ein paar Situationen belegen das ganz gut: Während meines Mathematik-Studiums war ich beispielsweise sicher, dass ich nie „Programmierknecht“ werden wollte. Die Informatiker hatten sich an der Uni bei Beginn meines Studiums gerade mit einem eigenständigen Fachbereich von uns Mathematikern abgespalten und wir wollten nicht mit so etwas schmutzig Praktischem wie Programmierung und Computertechnik in einem Topf geworfen werden. Und außerdem habe ich mich sowieso mehr für Physik interessiert und dieses Fach parallel zu Mathematik studiert.

Als ich dann mit dem Studium fertig war, stand ich nun mit meinem Diplom in Mathematik da. Und was gab es für Mathematiker für Job-Angebote? Hauptsächlich als Programmierer. Aber als ich dann als Programmierer bei einer Versicherung angefangen hatte, bin ich dem Reiz dieser ehemals „verachteten“ Welt erlegen. Sowohl was die Freiheiten eines Programmierers in der Firma anging, aber vor allen Dingen der spannenden Tätigkeit, dem Erschaffen von Dingen (wenn auch nur Programmen) und den logischen Denkweisen.

Nun haben wir bei der Versicherung in meiner Abteilung anfangs prozedural mit Turbo Pascal programmiert. Das Ziel waren DOS-Programme für den Außendienst und unsere Vertriebspartner in Banken und Agenturen. Nach einiger Zeit sollten wir jedoch an einem objektorientierten Projekt arbeiten. Die OOP (objektorientierte Programmierung) kam zu dem Zeitpunkt als Hype auf. Aber diese objektorientierte Programmierung war für mich wieder so eine Neuerung, an die ich anfangs gar nicht heran wollte. Mir war der Nutzen zu dem Zeitpunkt einfach nicht klar. Vor allen Dingen, da wir in der Versicherung mit C/C++ einen hybriden Zugang zu dieser Welt gewählt hatten und man die objektorientierte Denkweise damit wunderbar umgehen konnte (und immer noch kann). Aber als ich 1996 – mittlerweile selbständig als Freelancer – Java programmieren wollte, musste ich mich richtig auf Objektorientierung einlassen und nach einiger Zeit wurde ich zum überzeugten Vertreter der OOP.

Als dann der Hype um Handys aufkam und jeder sich damit wichtigmachen wollte, war meine Aussage: „So etwas brauche ich nicht!“. Ich brauche nicht zu betonen, dass mich irgendwann auch Handys überzeugt haben. Und als alle Welt von mobilem Internet geschwärmt hat …. Soziale Netzwerke, Blogs etc. – alles habe ich erst einmal betrachtet, um es erst dann zu nutzen, wenn ich den Sinn und die Vorteile verstanden habe.

Es scheint in meinem Leben immer so zu sein, dass ich bei einem Zug erst einmal schaue wohin er fährt und ob er nicht gleich beim Verlassen des Bahnhofs entgleist. Aber wenn er abgeht, dann springe ich auf den letzten Wagen auf und arbeite mich sehr oft sogar bis zur Lok vor. Seltsamer Weise klappt das dann meist.

Was nun die Programmierung von Apps angeht, ist hier die Situation aber etwas anders. Wie angedeutet, liegen meine Wurzeln in der Desktop-Programmierung mit klassischen Sprachen, aber auch vor allen Dingen Java und zum Teil C# bzw..NET. Mein zweites Standbein, mit dem ich seit etwa 1996 hauptsächlich gearbeitet und programmiert habe, sind Web-Technologien. Nun bot Java schon sehr früh die Möglichkeit zur Entwicklung

von mobilen Applikationen – die sogenannten MIDlets. Mit diesen habe ich mich bereits beschäftigt, bevor der eigentliche Boom der mobilen Apps und mobilen Endgeräte begonnen hat. Die Erstellung von Apps ist sozusagen eine der wenigen technischen Entwicklungen, bei denen ich zu früh dran war. Denn wie Sie vielleicht wissen, ist der MIDlet-Zug entgleist. Oder anders ausgedrückt – MIDlets und die frühen mobilen programmierbaren Geräte wurden nicht gerade ein Erfolg. Von daher war ich dann wieder vorsichtig als die aktuellen Apps und mobilen Geräte gehyped wurden. Aber ich war immer am Puls der Zeit, denn mit Java als auch C# hatte ich die Programmiergrundlagen seit Jahren bereit. Als dann zusätzlich Web-Technologien immer mehr als Basis für Apps und mobile Seiten bzw. Applikationen propagiert wurden, fügte sich im mobilen Bereich zusammen, was ich bereits über die Jahre in allen möglichen Umfeldern genutzt habe.

Ich bin von der Verwendung von Web-Technologien im mobilen Umfeld überzeugt, obwohl ich auch native Android-Apps mit Java oder Apps für Windows Phone mit C# und VB.NET erstelle. Nur der Einsatz von HTML5, CSS3 und JavaScript erlaubt die Erstellung universell verwendbarer Web-Apps für Android, Apple, Windows Phone und andere mobile Plattformen. Und mit Cordova können Sie auch die speziellen Features mobiler Endgeräte ausnutzen. Sie können etwa auf GPS-Empfänger, die Orientierung, die Kamera, Datenbanken etc. direkt aus JavaScript heraus zugreifen, ohne spezielle native Programmiertechniken lernen zu müssen. Ich zeige Ihnen in diesem Buch, wie das geht.

Herbst/Winter 2016/2017 Ralph Steyer
http://www.rjs.de

Inhaltsverzeichnis

Einleitung und Grundlagen – Um was geht es bei Cordova?

1

Inhaltsverzeichnis

1.1 Was behandeln wir im einleitenden Kapitel?

Im einleitenden Kapitel wollen wir uns erst einmal kompakt ansehen, was Cordova eigentlich ist. Was zeichnet Cordova aus? Wozu ist Cordova überhaupt da und wie ist der Begriff „PhoneGap“ in Hinsicht auf Cordova zu sehen?

R. Steyer, *Cordova*, DOI 10.1007/978-3-658-16724-0_1

1.2 Erste grundlegende Überlegungen zum Umfeld

Die Verwendung mobiler Endgeräte boomt jetzt schon seit einigen Jahren, weil sie cool und (meist) nützlich ist. Mobilität mit Handys, Smartphones oder Tablets über ebook-Reader bis hin zu Netbooks, Ultrabooks und Notebooks ist ein Zeichen unserer Zeit. Und die Grenzen der Gerätetypen verschwimmen. Ebenso kommen immer wieder neue Typen an mobilen Geräten und Plattformen hinzu. Es ist erstaunlich, dass die Industrie immer wieder neue Kategorien erfinden und damit einen (künstlichen) Bedarf schaffen kann. Letztendlich kann die Vielfalt egal sein – Mobilität an sich ist wichtig und „Always-on" dabei mehr oder weniger ein Muss. Spätestens seit erschwingliche Flatrates für mobiles Internet vorhanden sind. Und dann bedarf es geeigneter Anwendungen (Apps), die den Besonderheiten mobiler Endgeräte Rechnung tragen und die ebenfalls entsprechenden Mehrwert bieten, den man mobil gebrauchen kann.

Sind Netbooks, Ultrabooks oder Notebooks dabei noch „echte" Computer, die man mit Tastatur und Maus bedienen kann, müssen ebook-Reader, Smartphones und Tablets meist ohne solche externen Eingabegeräte auskommen. Je kleiner und leichter diese Geräte sind, desto besser sind sie zu transportieren und damit verzichtet man im Hardwaredesign natürlich auf alles, was nicht notwendig erscheint. Die Eingabe von Daten durch die Anwender wird dabei auf wenige Hardwaretasten und vor allen Dingen Touchscreens, Sensoren und Spracheingabe übertragen. Aber das erzwingt auf der einen Seite spezifische Designs der Benutzeroberflächen. Auf der anderen Seite werden gerade Smartphones und Tablets oft in der Hand gehalten und nicht wie klassische Computer feststehend bzw. liegend auf dem Schreibtisch bedient. Ebenso ändert sich oft der Standort des Geräts – während des Betriebs. Ebenfalls besitzen mobile Geräte oft spezifische Hardware, die bei stationären Geräten oder klassischen Computern nicht vorhanden sind oder nur begrenzt Sinn machen. Besondere Hardwaresensoren – etwa zum Erkennen der Orientierung des Bildschirms oder des Standortes eines Benutzers – unterstützten in der Regel diese spezifische Verwendung.

1.2.1 Die mobile Welt – Tausende von Inseln

Nun gibt es allerdings gerade im mobilen Bereich ein extrem heterogenes Umfeld als technische Basis. Sowohl was die Hardware angeht, aber auch die Bedienkonzepte, die verwendeten Programmiersprachen und die Betriebssysteme.

Hintergrundinformation
Im Buch werden wir uns auf die **Referenzsysteme** Android, Windows Phone und iOS konzentrieren. Aber die Ausführungen zu Cordova werden in der Regel auch für weitere mobile Betriebssysteme gelten, soweit Cordova diese unterstützt. Das ist ja gerade das Wesen plattformneutrale Apps.

Native Programmierung von mobilen Applikationen (Apps) bedeutet nun, dass Sie sich auf eine Hardware samt speziellem Betriebssystem oder gar nur eine spezielle Version davon konzentrieren. Dazu verwenden Sie in der Regel mächtige Programmiersprachen

wie Objective C oder Swift, Java, C/C++ oder C#. Wollen Sie mehrere Welten unterstützen, müssen Sie gegebenenfalls für jedes Zielsystem eine eigene App erstellen, die funktional identisch zu einer bereits vorhandenen App für ein weiteres Zielsystem ist. Dabei müssen Sie zudem sehr wahrscheinlich die App für die eine Plattform in der Programmiersprache A und für die andere Welt in der Programmiersprache B erstellen. Das kann einen immensen Aufwand bedeuten, wie man leicht nachvollzieht.

Wenn Sie nun weitgehend neutral von verschiedenen Zielplattformen bleiben wollen, bieten sich bei modernen Gerätetypen alternativ klassische Web-Technologien als Basis einer App an. Insbesondere **HTML** (Hyper Text Markup Language) und **CSS** (Cascading Style Sheets) werden in Verbindung mit *JavaScript* mittlerweile von allen relevanten Herstellern mobiler Endgeräte unterstützt und die neuen Möglichkeiten von HTML5 und CSS3 eröffnen sehr interessante Perspektiven. Erst einmal zwar nur im Rahmen der jeweilig verfügbaren eingebetteten Web-Browser (das sind derzeit fast immer die Engines WebView, Trident oder Webkit), aber diese Unterstützung kann man für „echte“ Apps „missbrauchen“. Damit können Sie den Quellcode für eine App nur einmal entwickeln und (relativ) problemlos auf allen modernen mobilen Systemen einsetzen. Dieser Ansatz mit der Verwendung von klassischen Web-Technologien hat auch den Charme, dass viele Leute bereits HTML, CSS oder JavaScript beherrschen und damit die Einstiegshürde zur Erstellung von Apps viel niedriger als bei nativer Programmierung ist.

Sie müssen sich dann aber auch darüber im Klaren sein, dass Sie eben Web-Apps entwickeln. Mit allen Vorteilen, die wir schon angedeutet haben, aber auch Nachteilen. Um nicht in den Verdacht unkritischer Lobhudelei zu geraten, soll neben einer optisch nicht immer dem nativen Layout entsprechender GUI (Grafical User Interface) insbesondere die manchmal schwache Performance von Web-Apps als Nachteil direkt angeführt werden. Insbesondere kann man mit JavaScript nicht so einfach oder gar nicht direkt die spezifische Hardware des mobilen Geräts nutzen. Aber hier helfen ja gerade Frameworks wie Cordova. Als spezielle Erweiterung für diese mobilen Belange erlaubt Cordova den Zugriff auf die speziellen Hardwarekomponenten mobiler Geräte – auf Basis von Quelltext in HTML, JavaScript und CSS. Wobei – um auch hier die Euphorie etwas zu dämpfen – es bei einigen Geräten und Betriebssystemen bzw. Betriebssystemversionen eine Reihe von Problemen beim Zugriff auf Hardwarebestandteile geben kann. Aber auch hier kann man davon ausgehen, dass das Cordova-Framework als auch die Geräte und Plattformen immer besser und die Probleme abnehmen werden.

Ebenso lassen sich solche Web-Apps wie native Apps auf einem Endgerät installieren und Sie erhalten damit die Möglichkeit, die Apps in einem der spezifischen Marktplätze zu vermarkten. Das ist sicher auch nicht zu verachten.

1.2.2 Was erwartet Sie in dem Buch?

Wie der Titel offensichtlich deutlich macht, beschäftigten wir uns in diesem Buch mit dem Cordova-Framework und wie man damit die Entwicklung Web-basierter, plattformneutraler Apps betreiben kann. Dazu lassen wir uns von geeigneten Mitteln für die

Entwicklung Web-basierter Apps unterstützen und lernen typische Anwendungen mobiler Apps kennen. Von dem ausführlichen Einrichten von Cordova-Projekten (was nicht ganz trivial ist) samt Fragen zur Hardware über die intensive Behandlung des Frameworks und seiner Bibliotheken samt des Aufbaus einer geeigneten Oberfläche einer App, den Zugriff auf typische Elemente eines modernen Smartphones (Geolocation, Orientierung, Kamera, Audiowiedergabe, …) bis hin zum Vertrieb und der Vermarktung von Apps.[1]

▶ **Tipp** Die Quellcodes des Buchs finden Sie nach Kapiteln und darin erstellten Projekten sortiert auf den Webseiten des Verlags. Die Namen der jeweilig aktuellen Dateien bzw. Projekte werden als Hinweise oder direkt im Text vor den jeweiligen Beispielen angegeben und bei Bedarf wiederholt. Ich empfehle allerdings, dass Sie die Beispiele unbedingt alle von Hand selbst erstellen. Das ist für das Verständnis und das Lernen eindeutig besser als ein reines Kopieren oder nur Ansehen.

Beachten Sie auch, dass nur die selbsterstellten Quellcodes und keine vollständigen Projekte oder Bibliotheken zum Download bereitgestellt werden. Denn Sie sollten auf jeden Fall die Apps mit dem eigenen Entwicklungssystem übersetzen. Nur dann können Sie sicherstellen, dass die Apps auch laufen. Und die verwendeten Bibliotheken (im Wesentlichen zu Cordova) benötigen Sie auch für Ihr spezielles Ziel- und Entwicklungssystem – darauf gehen wir ja im Buch ausführlich ein. Ebenso werden die Bibliotheken immer wieder erneuert und Sie sollten die jeweils aktuelle Version verwenden. Sie finden allerdings Hinweise zu den verwendeten Bibliotheken im Buch, damit im Fall einer nicht kompatiblen neuen Version das Problem beseitigt werden kann.

In diesem Buch werden verschiedene Schreibkonventionen eingehalten, die Ihnen helfen sollen, die Übersicht zu bewahren. Wichtige Begriffe werden hervorgehoben. Vor allem sollten Sie erkennen können, ob es sich um normalen Text oder Programmcode handelt. Wenn bestimmte Codepassagen mir besonders wichtig erscheinen, werde ich sie so hervorheben. Ebenso werden Tasten und noch einige weitere Besonderheiten gekennzeichnet. Diese Formatierungen werden konsequent verwendet. Und ebenso werden Sie in dem Buch Bereiche vorfinden, die über die Markierung mit verschiedenen Symbolen besondere Aufmerksamkeit erzeugen sollen.

An verschiedenen Stellen im Buch werden Sie auch Quellangaben finden, die ins Internet verweisen. Zum Zeitpunkt der Bucherstellung wurden die Quellen überprüft, aber es kann natürlich immer sein, dass diese Angaben im Laufe der Zeit nicht mehr aktuell sind.

Wenn im Buch auf Software und Bibliotheken verwiesen wird, sollten Sie ebenso die Versionsnummern beachten (das wurde oben ja schon angedeutet). Ggf. werden auch

[1] Letzteres aber nur als Randthema.

hier neue Versionen verfügbar sein und bestimmte Erklärungen nicht mehr zu 100 % übereinstimmen.

1.3 Was sind Cordova und PhoneGap?

Wenn wir von „Cordova“ reden wollen, müssen wir zuerst „PhoneGap“ nennen. PhoneGap, dessen Webseite Sie unter http://phonegap.com/ finden (Abb. 1.1), ist ein Open-Source-Entwicklungs-Framework zur Programmierung von Apps mit JavaScript und anderen clientseitigen Webtechnologien, das aus einem iPhoneDevCamp-Event in San Francisco hervorging und ursprünglich von der Firma Nitobi (http://www.nitobi.com/) entwickelt wurde.

Im Oktober 2011 wurde diese Firma von Adobe Systems (http://www.adobe.com/de/) übernommen. Gleichzeitig wurde der PhoneGap-Code der Apache Software Foundation für ein neues OpenSource-Projekt namens Apache Cordova (zuerst Apache Callback) bereitgestellt (https://cordova.apache.org/).

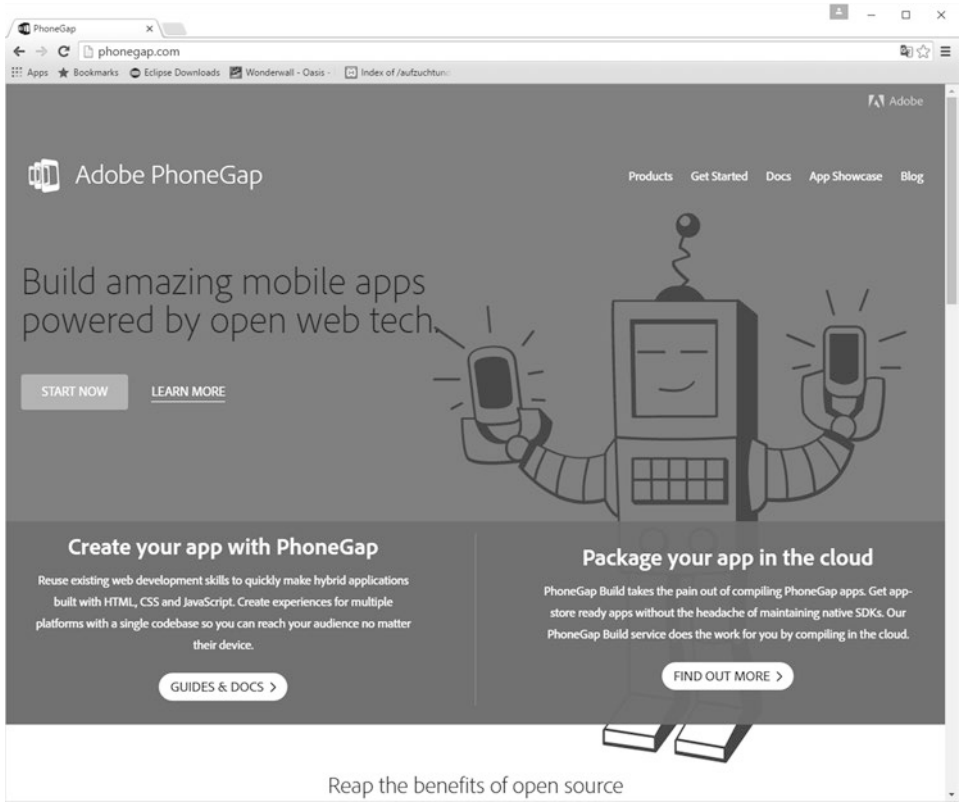

Abb. 1.1 Die zentrale Seite des PhoneGap-Projekts

1.3.1 Cordova oder PhoneGap oder was?

Genaugenommen ist PhoneGap damit eine **Distribution** von Apache Cordova und die verschiedenen Bezeichner sind im wesentlichen lizenzrechtlich von Bedeutung. Oft wird Apache Cordova als die *Engine* zur Ausführung von PhoneGap gesehen – so wie etwa Webkit die Engine von Chrome oder Safari ist. Grundsätzlich kann die eigentliche PhoneGap-Distribution einige zusätzliche Tools für spezifische Adobe-Services enthalten, die nicht im freien Apache-Projekt zur Verfügung stehen. Aber das spielt für unsere Apps keine entscheidende Rolle. Wir werden deshalb PhoneGap und Cordova synonym verwenden. Interessant ist jedoch, dass der Bezeichner „Cordova“ den Bezeichner „PhoneGap“ in der Popularität und Bekanntheit mittlerweile mehr als deutlich überholt hat. Das schlägt sich sowohl im Titel des Buchs als auch in den Trefferlisten bei Suchen im Internet wieder. Bei einer Suche, die ich aktuell durchgeführt habe (das ist natürlich nur eine Momentaufnahme von Sommer 2016), bekam ich etwa 450.000 Treffer zum Suchbegriff „PhoneGap“ und über 46 Millionen Treffer beim Suchbegriff „Cordova“.

Hintergrundinformation
Für das Buch bildet Cordova in der Version 6.x die Basis.

PhoneGap respektive Cordova ermöglicht Entwicklern nun wie gesagt die Programmierung von mobilen Apps per JavaScript, HTML und CSS. Über die Verwendung von diesen Web-Technologien soll die Cross-Plattform Entwicklung mobiler Anwendungen erleichtert werden. Dabei ist Cordova in der Regel keine Konkurrenz zu reinen mobilen Web-Frameworks wie iWebkit, jQTouch, Sencha Touch oder jQuery Mobile, sondern eine Ergänzung. Oder diese sind umgekehrt Ergänzungen zu Cordova. Sie können solche Frameworks und Cordova also gut in Kombination verwenden, was wir in dem Buch teilweise auch machen werden. Nur wozu braucht man Cordova, wenn schon ein Framework wie jQuery Mobile oder Bootstrap umfangreich die Oberfläche und Logik einer App mit HTML, CSS und JavaScript unterstützt?

1.4 Die Architektur, der Cordova-Wrapper und FFI

Cordova erzeugt aus diesem reinen Web-Code, der bei den genannten anderen Frameworks rein über einen Browser auf der mobilen Plattform nutzbar ist, native Applikationen. Oder genauer – Cordova packt sie in native Apps ein. Dazu wird ein **Foreign Function Interface** (FFI) gegenüber einer eingebetteten WebView- oder Webkit-Engine (also grob gesagt dem Web-Browser) auf dem mobilen Gerät bereitgestellt (der sogenannte PhoneGap-Wrapper bzw. Cordova-Wrapper).

▶ Unter **FFI** (auf Deutsch „Fremdfunktionsschnittstelle“) versteht man einen Mechanismus, mit dem ein Programm Routinen oder Dienstleistungen aufrufen oder nutzen kann, die in einer anderen Programmiersprache geschrieben wurden. Der Begriff

stammt aus der Spezifikation für Common Lisp, wird aber auch in anderen Sprachen verwendet. Andere Sprachen verwenden wiederum andere Terminologien, die aber meist die gleichen Mechanismen beschreiben. Wobei sich die konkret bereitgestellten Leistungsumfänge unterscheiden können. Einige FFIs sind auf bestimmte offengelegte Funktionen beschränkt, während andere auch Anrufe von Methoden in einem Objekt oder einer Klasse gestatten und einige FFIs erlauben sogar die Migration von komplexen Datentypen und/oder Objekten über die Sprachgrenze hinweg.

In den meisten Fällen wird ein FFI durch eine Programmierung in einer Programmiersprache auf einer höheren Ebene umgesetzt. Damit kann ein FFI in einer unteren Ebene Sprachdienstleistungen definieren und bereitstellen, die sonst nur in der Regel von einer Systemsprache wie C oder C ++ verwendet werden können. Dies wird in der Regel entweder der Zugang zu Dienstleistungen des Betriebssystems sein, die in dem API (Application Programming Interface) des Betriebssystems definiert sind, oder Leistungsinformationen.

Wenn Sie also mit diesem Framework eine Anwendung für mobile Geräte bauen, ist diese hybrid. Das bedeutet, dass sie weder wirklich nativ sind (das gesamtes Layout-Rendering

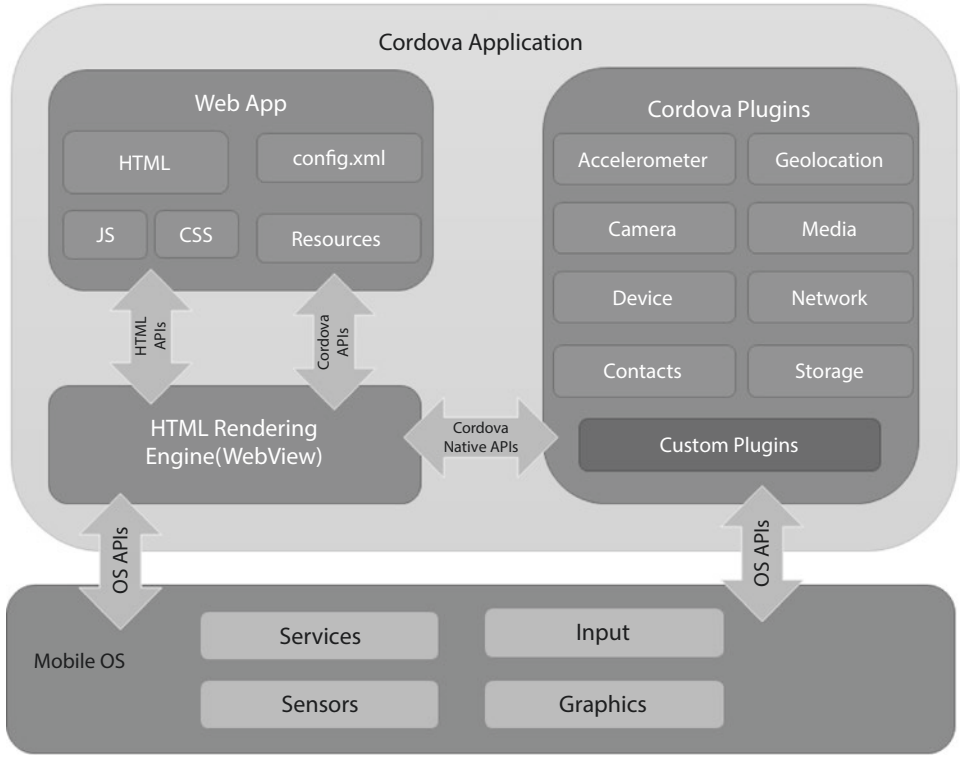

Abb. 1.2 Architektur von Cordova (https://cordova.apache.org/docs/en/latest/guide/overview/index.html)

wird über den eingebetteten Browser anstatt der Plattform mit dem jeweiligen nativen UI-Framework durchgeführt) noch rein web-basiert (sie sind nicht nur Web-Anwendungen, sondern für die Auslieferung in App-Stores verpackt). Genaugenommen läuft eine solche Anwendung unter Cordova in einem unsichtbaren Browserfenster – der reinen Web-Engine auf Basis von WebView – und das HTML-Grundgerüst samt der verwendeten CSS- und JavaScript-Anweisungen werden nicht umgewandelt. Zudem Cordova-Apps Zugriff auf einen großen Teil des jeweiligen spezifischen (API) eines mobilen Geräts. Und das macht es möglich, dass Sie aus JavaScript auf Hardwarekomponenten wie z. B. Kameras, GPS-Empfänger oder Beschleunigungssensoren zugreifen (Abb. 1.2). Wobei man festhalten kann, dass viele Erweiterungen von HTML5 und den darüber direkt in JavaScript verfügbaren Objekten auf einige Komponenten auch direkt Zugriff gestatten (etwa den GPS-Sensor). Diese „nativen" Objekte werden aber nahtlos in das API von Cordova integriert und meist mit speziellen Erweiterungen noch besser nutzbar gemacht.

Hintergrundinformation

Es gibt auch Ansätze zur App-Entwicklung mit JavaScript und anderen Web-Technologien, bei denen die ausgelieferte App letztendlich vollständig nativ ist. Diesen Weg geht etwa das Cross Platform Framework Appcelerator **Titanium** (https://www.appcelerator.com/mobile-app-development-products/). Unter Android wird etwa der gesamte Web-Code in natives Java umgewandelt.

Eine andere Alternativ kommt von Microsoft. Wobei Microsoft dazu eine Firma übernommen hat, die sich auf plattformübergreifende App-Entwicklung spezialisiert hatte – die Firma **Xamarin** (https://www.xamarin.com/). Diese wurde 2011 von den Entwicklern des Open-Source-Projekts Mono gegründet. Mono hatte generell plattformübergreifende Entwicklung unter.NET zum Ziel, während Xamarin die App-Entwicklung mit.NET revolutionieren sollte. Vereinfacht übersetzt Xamarin C#-Code in native Apps und ermöglicht die Wiederverwendung von Code über Plattformgrenzen hinweg – auch für iOS- und Android-Devices. Allerdings beträgt der Anteil von wiederverwendbarem Code nicht 100 %. Insbesondere am Beginn des Projekts war der Anteil deutlich geringer. Das betrifft sowohl die interne Logik als auch vor allen Dingen das Design und die Bedienung. Der einzig wirkliche Vorteil bei Xamarin blieb somit lange Zeit die Verwendung einer einheitlichen Programmiersprache. Mittlerweile gibt es zumindest für die Generierung der Oberfläche ein Konzept mit Namen Xamarin.Forms. Das ist eine zusätzliche Abstraktionsschicht über Benutzeroberflächen. Die Oberfläche wird zunächst plattformunabhängig definiert und anschließend je nach Plattform in das dort übliche native Pendant übersetzt. Durch die Abstraktion steht allerdings nur der kleinste gemeinsame Nenner aller Plattformen zur Verfügung, den man bei Bedarf um plattformabhängige, native Komponenten erweitern muss. Microsoft spricht aktuell auf seinen Webseiten zu Xamarin davon, dass man im Mittel 75 % des App-Codes gemeinsam auf allen mobilen Entwicklungsplattformen nutzen kann, die Benutzeroberflächen aber zu 100 %, sofern Sie sich auf den besagten kleinsten gemeinsamen Nenner aller unterstützten Plattformen beschränken.

1.5 Unterstützte Plattformen bei Cordova

Die aktuellen potentiellen Zielplattformen für Cordova-Apps kann man in einem Buch nur andeuten, denn die Unterstützung bestehender Plattformen wird ständig erweitert. Zudem kommen neuen Zielplattformen hinzu, die voraussichtlich sehr schnell auch unterstützt

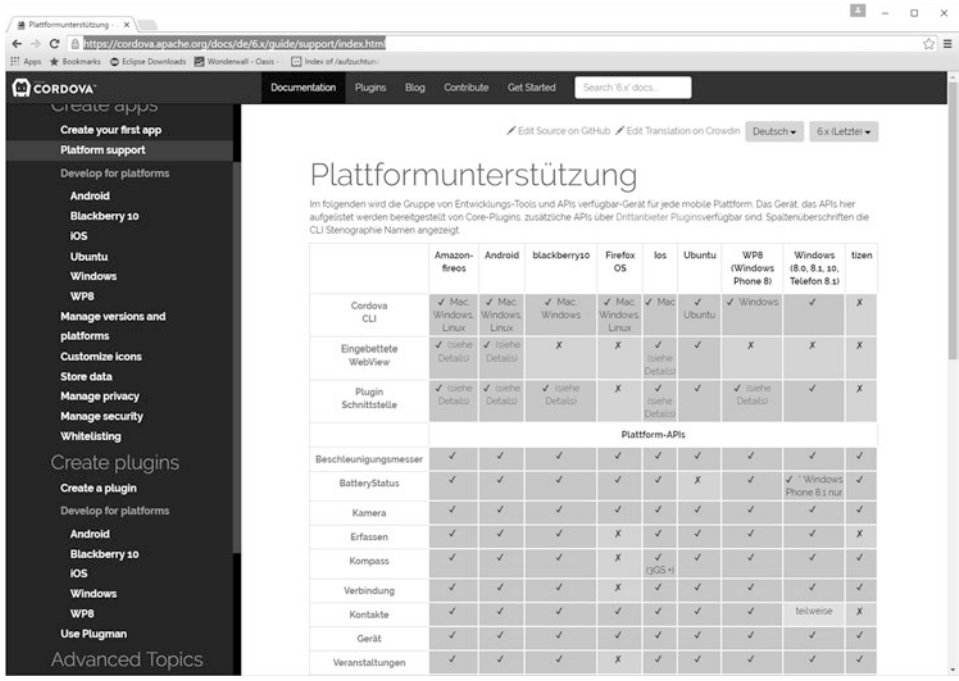

Abb. 1.3 Die Features von Cordova und wie weit sie auf den verschiedenen Plattformen zur Verfügung stehen

werden, wenn sie eine relevante Verbreitung erlangen. Andererseits nimmt auch gelegentlich die Bedeutung von Zielplattformen ab und es ist nicht abzusehen, ob diese dann langfristig mitgezogen werden. Derzeit unterstützt Cordova aber die Entwicklung für die Betriebssysteme u. a. Apple iOS, Google Android, Microsoft Windows Phone, Ubuntu, Firefox OS und RIM blackBerry10. Allerdings kann sich die Unterstützung bestimmter Features in den Betriebssystemen und/oder bestimmter Versionen davon unterscheiden. Auf der Webseite des Projekts (https://cordova.apache.org/docs/de/6.x/guide/support/index.html bzw. https://cordova.apache.org/docs/en/latest/guide/support/index.html – Abb. 1.3) finden Sie die jeweils aktuellen Daten – auch für ältere Versionen von Cordova. Sie werden sehen, dass bestimmte Features nicht auf allen Plattformen bereitstehen, aber auch wenn ein Feature grundsätzlich unterstützt wird, gibt es in Details oft Abweichungen. Diese sind dann aber in der genauen Dokumentation des API beschrieben.

Hintergrundinformation

Da sich mobile Plattformen massiv unterscheiden, stellt Cordova oft sogenannte **Quirks** bereit. Das sind spezielle Anpassungen an einzelne Plattformen. In der Dokumentation von Cordova finden Sie neben der Beschreibung von Standardfunktionalitäten für alle Plattformen meist eine in der Regel noch viel umfangreichere Beschreibung dieser einzelnen Quirks mit Spezialitäten der verschiedenen

Plattformen. Dabei ist der Umfang der Quirks-Dokumentationen umgekehrt proportional zur Bedeutung. Fast immer kann man sich auf die Standardfunktionalitäten für alle Plattformen konzentrieren und braucht nur in ganz wenigen Ausnahmesituationen diese speziellen Features.

1.6 Was sollten Sie bereits wissen?

Klären wir nun noch kurz, was Sie an Kenntnissen haben sollten, damit Sie mit dem Buch effektiv arbeiten können. Da wir mit Cordova explizit JavaScript programmieren, sind für Sie Grundlagenkenntnisse darin sehr sinnvoll. Und da JavaScript eigentlich nur im Rahmen einer HTML-Seite verwendet wird (serverseitiges JavaScript sei außer Acht gelassen), möchte ich ebenfalls voraussetzen, dass Sie HTML kennen. Ebenso sollten Style Sheets und CSS keine Fremdworte für Sie sein. Sie brauchen allerdings keine professionellen Kenntnisse als Basis. Weder in HTML, CSS oder JavaScript noch einer anderen Programmiersprache. Zu diesen drei Themen finden Sie bei Bedarf in Kap. 4 als auch im Anhang auch eine konzentrierte Einführung. Grundsätzlich handelt es sich bei diesem Buch um ein Einsteigerbuch in die mobile Programmierung und das bedeutet, dass zwar nicht bei Null begonnen wird, aber die einzelnen Schritte und Aktionen sehr ausführlich erklärt werden. Insbesondere wenn es sich um Details zu speziellen mobilen Aspekten handelt. Allerdings muss eingeschränkt werden, dass unser eigentlicher Fokus die Programmierung mit Cordova ist und nicht alle Aspekte der allgemeinen App-Entwicklung und -Distribution behandelt werden.

Ansonsten berühren wir bei fortgeschrittenen Themen auch einige Techniken, die in dem Zusammenhang als Grundlagen hilfreich sein könnten. Etwa XML oder SQL, was etwa im Kapitel zum Umgang mit Dateien und Datenbanken vorkommen wird.

1.7 Was sollten Sie haben?

Um die Beispiele und Übungen im Buch nachvollziehen zu können, benötigen Sie natürlich einen Computer zur Entwicklung. Einen Internet-Zugang setze ich als selbstverständlich voraus. Ob Sie nur einen Windows-PC, einen Linux-Rechner oder einen Mac als Entwicklungsrechner verwenden, spielt (eigentlich) keine Rolle – für das Buch. Diese drei Welten werden explizit im Buch als Entwicklungs-Basis betrachtet.

1.7.1 Betriebssysteme

Allerdings wollen wir ja mobile Apps entwickeln und da werden Sie sich vermutlich auf eine Zielplattform erst einmal festlegen, was damit auch die Entwicklungsplattform für Sie beeinflusst. Das soll jetzt kein Widerspruch dazu sein, dass die eigentlichen Apps auf

mehreren Plattformen laufen werden (oder sollen). Aber in manchen Habitaten müssen Sie zwingend einen bestimmten Rechnertyp bzw. ein bestimmtes Betriebssystem als Entwicklungsbasis verwenden, wenn Sie für spezielle mobile Plattform entwickeln wollen, wenn Sie nicht schwere Klimmzüge für eine Alternativlösungen machen wollen. Bei Apple benötigen Sie etwa zur Entwicklung von Apps für das iOS-System auch zwingend einen Mac mit einem aktuellen Betriebssystem. Aber auch für die Entwicklung von Cordova-Apps für Windows Phone sind Sie recht beschränkt, was Ihren Entwicklungsrechner angeht. Ein PC mit Windows 7 oder 8.x mit den letzten Service-Pack und Aktualisierungen sollte als Untergrenze ausreichen, aber im Grunde ist Windows 10 als Entwicklungssystem zu empfehlen. Für Windows Phone Universal-Apps brauchen Sie mindestens Windows 10 in jedem Fall. Für Android sind Sie hingegen recht frei in der Wahl Ihre Entwicklungsbasis, aber auch hier machen ältere Betriebssysteme oft unnötige Probleme.

▶ **Tipp** Im Laufe des Buchs wird Ihnen ein Dienst vorgestellt, mit dem Sie Ihre Apps in einer Cloud für ein beliebiges Zielsystem übersetzen können. Auch wenn Sie keinen passenden Entwicklungsrechner zur Verfügung haben. Allerdings ist der Dienst für kommerzielle Anwendung nicht kostenfrei. Mehr möchte ich an dieser Stelle nicht ausführen, da wir noch weitere Grundlagen als Basis benötigen. Nur sollten Sie im Hinterkopf beachten, dass Sie mit dem Dienst etwa eine App, die Sie auf einem Windows-Rechner geschrieben haben, im Prinzip auch für iOS übersetzen lassen können.

Für das Buch bzw. das Lernen der App-Entwicklung mit Cordova genügt es aber wie gesagt, dass Sie sich erst einmal auf ein Zielsystem mit einem passenden Entwicklungsrechner konzentrieren. Wobei die alternative Betrachtung von den drei Referenzzielplattformen auch dann interessant ist, wenn Sie gar nicht für alle Zielsysteme entwickeln wollen – Sie lernen durch die Vergleiche viel für Ihr eigentliches Zielsystem. Zudem stellt sich die Frage nach den realen mobilen Geräten – wer von Ihnen hat schon 3 oder gar mehr Smartphones oder Tablets mit verschiedenen Betriebssystemen zum Testen zur Verfügung? Ich gehe jedoch davon aus, dass Sie aber mindestens ein Gerät Ihr Eigen nennen, auf dem Sie testen können und für das Sie am Anfang erst einmal mit dem passenden Betriebssystem entwickeln. Aber selbst wenn Sie gar kein Smartphone oder ein anderes mobiles Gerät besitzen, dann gibt es zumindest sogenannte Emulatoren bzw. Simulatoren, über die Sie die Apps aus dem Buch testen können. Ich stelle Ihnen diese Tools im Folgenden gleich noch genauer vor.

Hintergrundinformation
Die konkreten Cordova-Quellcodes im Buch werden – bis auf wenige Ausnahmen – unabhängig von der Entwicklungs- als auch Zielplattform sein. Wenn bestimmte Apps auf einer unserer Referenzsysteme nicht laufen, wird das ausdrücklich erwähnt.

1.7.2 Die Entwicklungs-Software

Die reine Erstellung von Quellcodes erfordert im Grunde außer einem beliebigen Editor keine weitere Software. Bei Cordova-Apps schreiben Sie ja explizit nur HTML-, CSS- und JavaScript-Code. Und selbst für die umgebenden Cordova-Wrapper kann man im Prinzip mit einem reinen Editor arbeiten. Aber Sie müssen ja diese Quellcodes im Editor irgendwann übersetzen, damit daraus lauffähige Apps-Programme werden. Und dazu sind gewissen Programme notwendig. Zudem ist ein reiner Editor selbst bei HTML, CSS und JavaScript wenig komfortabel. Auch für einen angemessenen Programmierkomfort bietet sich eine Integrierte Entwicklungsumgebung (IDE – Integrated Development Environment) an. Allerdings müssen wir hier differenzieren, für welche Zielplattformen Sie entwickeln wollen.

1.7.2.1 Software für die Entwicklung von Android-Apps

Das Android-Betriebssystem von Google ist ein Linux-System. Für Linux kann man in verschiedenen Sprachen native Applikationen programmieren. Aber native Apps für Android sind in der Regel in **Java** geschrieben, zumal das Google selbst mit diversen Tools und APIs unterstützt. Die objektorientierte Programmiersprache Java, die es seit 1995 gibt, wurde von Sun Microsystems (https://www.oracle.com/sun/) erfunden, das vor einigen Jahren von Oracle (https://www.oracle.com/) übernommen wurde. Wenn Sie Cordova-Apps für Android entwickeln, werden diese über einen Java-Wrapper als native Apps in Android eingebunden. Von daher werden wir uns im Buch auch ganz kurz mit Java beschäftigen, obwohl wir uns letztendlich unter Cordova kaum mit der eigentlichen Java-Programmierung auseinander setzen müssen.

Als **Entwicklungsplattform** (also das Betriebssystem, auf dem Sie programmieren) für Java sind Sie in der glücklichen Situation, dass Ihnen mehrere Varianten zur Verfügung stehen. Sei es Linux, Windows oder auch Mac OS in den aktuellen Varianten. Aber dort benötigen Sie für die Entwicklung von Java-Applikationen gewisse Programme und Bibliotheken.

1.7.2.1.1 Das JDK

Als erstes sei das **JDK** (Java Development Kit) genannt. Die gesamte Java-Entwicklung dreht sich um jenes ominöse JDK. Sie finden das JDK beispielsweise für die verschiedenen Betriebssystemen auf den offiziellen Java-Seiten (Java SE Download) von Oracle unter http://www.oracle.com/technetwork/java/javase/downloads/index.html, wie Sie in Abb. 1.4 sehen können.

Statt JDK wird im Umfeld von Java auch gelegentlich vom SDK (Software Development Kit) gesprochen. Hier hat Sun der Vergangenheit leider nicht ganz glücklich agiert, denn diese unterschiedlichen Bezeichner sorgen für unnötige Verwirrung. Wir werden im Buch konsequent auf den Bezeichner JDK zurückgreifen.

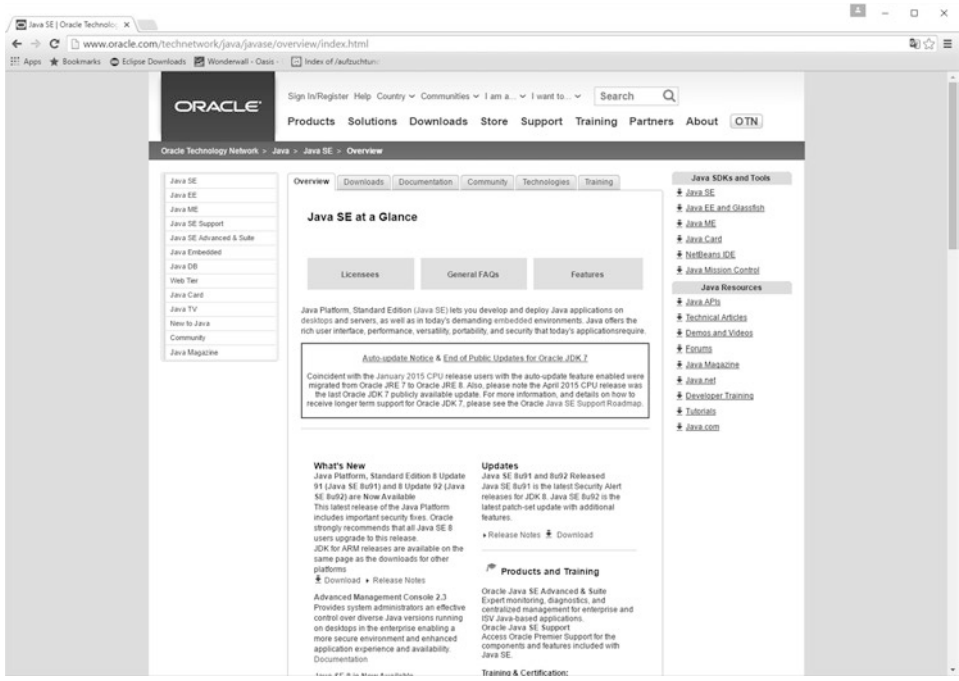

Abb. 1.4 Downloadmöglichkeiten für Java und das JDK

Hintergrundinformation
Diesem Buch liegt die Version 8 des JDK zugrunde. Und zwar die sogenannte Standard Edition.

Wenn Sie das JDK laden wollen, müssen Sie dessen Lizenzbedingungen akzeptieren und die passende Version für Ihr Betriebssystem auswählen. Sie erhalten dann über einen Link eine Installationsdatei zum Speichern auf Ihren Rechner.

Die Installation des JDK ist absolut einfach. Der Installationsassistent führt Sie mit wenigen Schritten zum Erfolg. Nach der Installation möchte Sie Oracle zu einer Registrierung per Internet motivieren, aber das ist nicht zwingend, um mit Java und dem JDK zu arbeiten.

1.7.2.1.2 Das Android Studio und Eclipse

Im Umfang des JDK ist kein Programm zur Erstellung von Quellcode dabei. Lange Zeit war im Umfeld von Android eine IDE mit Namen **Eclipse** (http://www.eclipse.org) das Standardwerkzeug. Diese IDE unter der OpenSource-Lizenz, die ursprünglich von IBM entwickelt, später freigegeben und nun von der Eclipse-Foundation bereitgestellt wird, hatte den Ruf einer universellen Lösung für alle Zwecke im Programmierumfeld. Eclipse

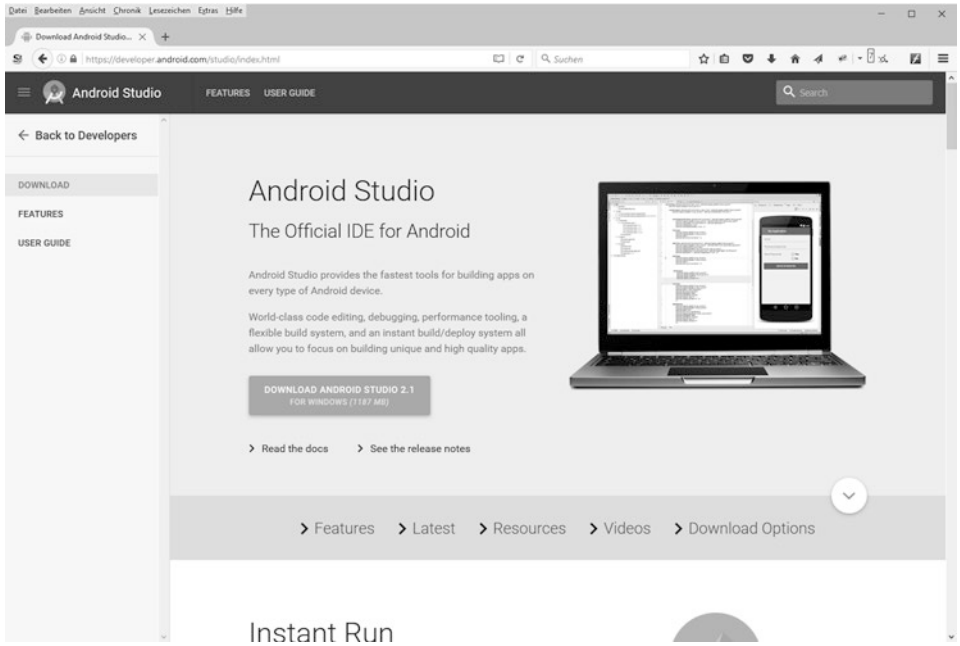

Abb. 1.5 Hier bekommen Sie das Android Studio

gibt es sowohl für Linux und Windows als auch Mac OS. Obwohl Eclipse selbst in Java geschrieben und im Schwerpunkt auf die Entwicklung mit Java ausgerichtet ist, können in Eclipse mithilfe unzähliger PlugIns nahezu alle Programmiertechniken abgedeckt werden. Auch die offizielle Dokumentation von Cordova setzte in den älteren Versionen des Frameworks auf die Verwendung genau dieser IDE. Zur Entwicklung von Android-Apps mit Eclipse benötigte man dann aber noch das Android SDK als Erweiterung, das Google bereitgestellt hatte.

Allerdings hat Google in der letzten Zeit alle Weichen zur Entwicklung von Android-Apps in Richtung des sogenannten Android Studios gestellt, das Sie unter dem Link https://developer.android.com/studio/index.html bekommen – Abb. 1.5. Was jedoch im Gegenzug nahezu die komplette Einstellung der Unterstützung der Android-Entwicklung bei Eclipse zur Folge hatte. Deshalb kann man kaum noch guten Gewissens zum Einsatz von Eclipse bei dieser Art der Applikationen raten.

Wenn man nicht aus persönlichen Gründen unbedingt Eclipse verwenden will, würde ich dringend zum Einsatz des Android Studios raten. Oder dem Visual Studio, worauf ich jetzt noch zu sprechen komme. Denn mittlerweile ist das Visual Studio zu einer universellen IDE geworden.

1.7.2.2 Software für die Entwicklung von Windows-Phone-Apps & mehr

Wie erwähnt, benötigen Sie zur Entwicklung von Windows-Phone-Apps im Normalfall einen geeigneten Windows-PC. Früher hatte man dort die Developer Tools von Microsoft

verwendet, aber auch hier ist die Zeit voran geschritten. Um auf einem PC für Windows Phone eine native App zu entwickeln, laden und installieren Sie derzeit am besten die neuste Version des **Visual Studios**. Im Buch verwenden wir die Version 2015. Die sogenannte **Community Edition** ist kostenlos und beinhaltet alles, was Sie zum Entwickeln von Apps benötigen.[2] Und zwar nicht nur für Windows-Apps. Microsoft hat sich in den letzten Jahren sowohl anderen Plattformen als auch der Open-Source-Gemeinschaft immer mehr geöffnet. Das resultiert unter anderem darin, mit Visual Studio plattformübergreifende Entwicklung betreiben können. Sie können also mittlerweile Apps für Android-, iOS- oder Windows-Geräte mithilfe von Visual Studio erstellen. Außerdem können Sie verbundene Dienste wie Office 365, Azure Mobile Services oder Application Insights problemlos den Apps hinzufügen, was wir aber nicht verfolgen werden. Insbesondere erhalten Sie mit dem Visual Studio einen Emulator für Apps – den Windows Phone Emulator. Diesen können Sie natürlich zum Testen der nativen Windows Phone-Apps, aber auch zum Testen von beliebigen mobilen Webseiten oder Web-Apps unter Realbedingungen verwenden, denn der Emulator bringt einen integrierten Browser mit.

Visual Studio erlaubt so mittlerweile die Erstellung von Apps auf verschiedene Weisen. Sie können die Apps mithilfe von C# oder VB.NET und mit dem .NET Framework, C++ oder auch mithilfe von HTML und JavaScript erstellen. Der letzte Punkt führt damit zu Cordova.

Sie sollten jedoch bei der Installation von Visual Studio darauf achten, dass Sie die Features für die plattformübergreifende Entwicklung mit auswählen. Achten Sie auf den Bezeichner „Cordova“ und wählen Sie auch alle Web-Features aus, die Ihnen angeboten werden. Es gibt zudem eine ganze Reihe an optionalen Erweiterungen rund um Cordova, die Sie nachträglich auch noch hinzufügen können. Damit wird Visual Studio zu einer allumfassenden IDE für App-Entwicklung.

1.7.2.3 Software für die Entwicklung von iOS-Apps

Die Entwicklung von Apps für mobile Apple-Geräte setzt einen geeigneten Mac voraus, auf dem Sie die **Xcode Developer Tools** (https://developer.apple.com/xcode/ – Abb. 1.6) installieren sollten.

Dieses Paket bietet alles, was Sie brauchen, um Anwendungen für Mac, aber auch iPhone und iPad zu erstellen. Xcode ist eng mit dem Cocoa und Cocoa Touch-Frameworks verbunden. Das Xcode-Toolset enthält die Xcode IDE mit dem Interface Builder Design-Tool und dem voll integrierten Apple LLVM Compiler sowie der üblichen weiteren unterstützenden Entwickler-Tools. Von besonderer Bedeutung ist die Integration eines iOS-Simulators, auf dem Ihre Anwendung in der gleichen Weise abläuft, als wenn ein tatsächliches iOS-Gerät die Basis wäre. Apple versteht also in Hinsicht auf die Anwendung durch den Programmierer unter dem Simulator das, was Google und Microsoft mit Emulator bezeichnen. Wir werden in der Folge einheitlich nur den Begriff Emulator verwenden.

[2] Natürlich können Sie auch die kostenpflichtigen Vollversionen einsetzen.

Abb. 1.6 Die Entwicklungs-Tools von Apple

1.7.2.4 Software für die neutrale Web-Entwicklung

Grundsätzlich sollten die bisher beschriebenen Tools alles beinhalten, was Sie zum Erstellen von Apps benötigen. Aber wir wollen ja Web-Apps erstellen und das bedeutet, dass wir ziemlich viel mit HTML, CSS und JavaScript arbeiten. Es kann durchaus sinnvoll sein, dass Sie zusätzliche Programme verwenden, in denen Sie diese Codefragmente extern bearbeiten und dann erst die Web-Strukturen in die eigentlichen Apps-IDEs hineinkopieren. Ich mache das oft, wenn die IDEs mir zu schwergewichtig sind oder nicht das bieten, was ich gerne hätte. Ich persönlich arbeite bei reinen Web-Sprachen sehr gerne mit Notepad++ (http://notepad-plus-plus.org/) als Editor. Aber auch PlugIns wie Aptana (http://www.aptana.com/) für Eclipse können Ihnen bei Web-Technologien viel helfen, wenn Standardfunktionalitäten nicht genügen bzw. optimal sind. Sehr sinnvoll ist auch ein lokaler Webserver, von dem Sie externe Inhalte zum Testen in Ihre Apps laden können. Eine gute Wahl ist XAMPP (http://www.apachefriends.org/de/xampp.html). Das ist eine Zusammenstellung von diversen freier Software rund um den Webserver Apache. Ziel ist das einfache Installieren und Konfigurieren des Webservers Apache in Verbindung mit den Datenbanksystemen MySQL (mittlerweile in der voll kompatiblen Distribution MariaDB im Einsatz) bzw. SQLite und den serverseitigen Skriptsprachen Perl und PHP sowie einigen weiteren Tools. XAMPP ist bewusst nicht für den Einsatz als Produktivsystem