O'REILLY®

React Native

NATIVE APPS PARALLEL FÜR ANDROID UND IOS ENTWICKELN

Erik Behrends

React Native

Native Apps parallel für Android und iOS entwickeln

Erik Behrends

O'REILLY®

Erik Behrends

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Satz: III-satz, www.drei-satz.de
Druck und Bindung: Media-Print Informationstechnologie,
mediaprint-druckerei.de

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

ISBN
Print: 978-3-96009-066-3
PDF: 978-3-96010-201-4
ePub: 978-3-96010-202-1
mobi: 978-3-96010-203-8

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

1. Auflage 2018

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Inhalt

Einleitung

Smartphones sind inzwischen allgegenwärtig, und wir sind es gewohnt, ständig einen leistungsfähigen Computer mit Internetzugang bei uns zu haben, den wir überall bedienen können. Von Apps erwarten wir ein hohes Maß an Bedienkomfort, das oftmals nur von nativen Apps gewährleistet werden kann. Dies betrifft nicht nur die Interaktion auf dem Multitouchscreen durch bestimmte Gesten, sondern auch die Unterstützung besonderer Gerätefunktionen wie z. B. die Verarbeitung von Sensordaten.

Soll mit einer nativen App eine möglichst große Anzahl von Smartphone-Benutzern erreicht werden, muss die App für Android-Geräte und iPhones entwickelt werden, denn der Markt für Smartphones wird von diesen beiden Plattformen nahezu vollständig abgedeckt. Die Technologien zur Entwicklung einer nativen App für Android und iOS sind komplex und unterscheiden sich in vielerlei Hinsicht. Ein Softwareunternehmen, das eine native App jeweils für Android und iOS entwickeln möchte, wird deshalb in der Regel auf zwei verschiedene Programmierteams angewiesen sein, wobei jedes dieser Teams sich auf eine Plattform spezialisiert. Diese Verdopplung des Entwicklungsaufwands ist ein erheblicher Kostenfaktor.

Um dieser Herausforderung zu begegnen, kann ein plattformübergreifender Ansatz gewählt werden, um mit weniger Aufwand eine App für Android und iOS entwickeln zu können. Für diesen Zweck stellt React Native ein äußerst attraktives Framework dar. Dabei ist die Tatsache, dass sich mit React Native echte native Apps für Android und iOS mit größtenteils identischem Code entwickeln lassen, nur ein positiver Aspekt von weiteren Vorteilen. React Native zeichnet sich zusätzlich durch eine hohe Produktivität aus, denn bei der Programmierung sind die Reaktionszeiten viel kürzer als bei den herkömmlichen Ansätzen, native Apps zu entwickeln. Darüber hinaus wird in React Native mit JavaScript programmiert, wodurch das Framework unter anderem für Webentwickler relativ leicht zugänglich ist. Neben Android und iOS gibt es z. B. mit Apple TV und Microsofts Universal Windows Platform weitere Zielplattformen, und es besteht die Möglichkeit, nativen Code für Android (Java) und iOS (Swift/Objective-C) einzubinden.

Ziele dieses Buchs

Zunächst möchte ich mit diesem Buch zeigen, dass React Native einen niedrigschwelligen Einstieg in die Entwicklung nativer mobiler Apps bietet. Nach nur wenigen Minuten können Sie Ihre erste App direkt auf dem Smartphone benutzen.

Um komplexere Apps zu programmieren, ist ein grundlegendes Verständnis verschiedener Technologien nötig, auf denen React Native basiert. Dazu gehören neue Features von JavaScript und die deklarative, auf Komponenten basierende Programmierung mit React. Diese Grundlagen stelle ich Ihnen vor.

In der täglichen Arbeit mit dem Framework React Native müssen Sie mit seinen Komponenten und APIs umgehen können. Ich werde einige davon vorstellen und mit praktischen Beispielen beschreiben. Ein übergeordnetes Ziel dieses Buchs ist, Sie in die Lage zu versetzen, dass Sie eigenständig das Framework verwenden können, um z. B. Komponenten einsetzen zu können, die nicht in diesem Buch behandelt werden.

Schließlich bringen mobile Apps einige Besonderheiten mit sich, auf die ich teilweise eingehen möchte. So erfordert die geringe Bildschirmgröße von Smartphones spezielle Ansätze in Bezug auf die Navigation, und für die Bedienung von Apps durch Berührung des Touchscreens sind besondere Formen der Interaktion nötig. Dazu wird es jeweils eigene Kapitel geben.

Über mehrere Kapitel hinweg werden wir eine App für Android und iOS entwickeln, mit der Sie ein Tagebuch führen können. Sie können die App bereits jetzt auf Ihrem Smartphone ausprobieren, wenn Sie neugierig sind, welche Funktionalitäten wir in diesem Buch behandeln werden. Unter *expo.io/@behrends/myjournal* ist die App *MyJournal* zu finden. Dort finden Sie auch Links zur App *Expo*, mit der MyJournal auf Android und iOS ausgeführt werden kann.

Zielgruppe und benötigte Vorkenntnisse

Dieses Buch richtet sich an Entwickler, die lernen möchten, wie mit React Native plattformübergreifend native Apps für Android und iOS programmiert werden können. Dabei wird vorausgesetzt, dass Sie bereits Programmiererfahrung haben, wobei grundlegende JavaScript-Kenntnisse von Vorteil sind. Außerdem sollte Ihnen bekannt sein, wie mit HTML oder XML hierarchische Dokumentstrukturen erstellt werden können und was die dazugehörigen Begriffe wie z. B. Element, Tag oder Attribut bedeuten.

Verwendung und Aufbau dieses Buchs

Mir ist wichtig, dass Sie von Anfang an praktische Erfahrungen mit React Native sammeln. Ich glaube, dass ein wirklicher Lernerfolg nur durch das eigenständige

Programmieren möglich ist. Daher möchte ich Sie ermutigen, die Codebeispiele selbst im Editor einzugeben und die so programmierten Apps zu testen. Das nötige Hintergrundwissen für das konzeptuelle Verständnis wird selbstverständlich an passender Stelle geliefert und aufgebaut.

Ab Kapitel 4 werden wir schrittweise eine App namens *MyJournal* entwickeln, mit der Sie auf dem Smartphone ein Tagebuch führen können. Wir werden verschiedene Aspekte von React Native direkt in dieser App umsetzen. Für die Kapitel 1 - 10 folgt nun eine kurze Beschreibung ihrer Inhalte:

Kapitel 1: React Native kurz vorgestellt
: Zu Beginn stelle ich React Native kurz vor, wobei ich insbesondere verschiedene Vorteile und die Funktionsweise des Frameworks grundlegend beschreibe.

Kapitel 2: Erste Schritte mit React Native
: Hier zeige ich Ihnen, wie eine einfache App in React Native entwickelt werden kann. Einige Konzepte werden zunächst kurz vorgestellt. Im weiteren Verlauf des Buchs gehen wir dann detaillierter auf sie ein.

Kapitel 3: React Native: die Grundlagen
: Dieses Kapitel behandelt relevante Features in JavaScript, die seit der Version ECMAScript 2015 (ES6) eingeführt wurden, und beschreibt die Prinzipien der Webbibliothek React, auf der React Native basiert. Sie können dieses Kapitel auslassen, wenn Sie bereits als Webentwickler mit React gearbeitet haben.

Kapitel 4: Plattformübergreifende UI-Komponenten verwenden
: Wichtige UI-Komponenten von React Native wie `TextInput` und Listen werden behandelt. Zur Veranschaulichung beginnen wir mit der Entwicklung einer App namens MyJournal, mit der Sie ein Tagebuch auf dem Smartphone führen können.

Kapitel 5: Styling des Layouts und des Erscheinungsbilds
: Die Gestaltung von Apps mit der `StyleSheet`-API und das Flexbox-Layout sind die Themen in diesem Kapitel.

Kapitel 6: Fotos mit der Kamera aufnehmen
: Hier lernen Sie, wie Sie Fotos mit der Kamera aufnehmen und durch die `Image`-Komponente in MyJournal darstellen können.

Kapitel 7: Daten lokal speichern und aus dem Web laden
: Die lokale Speicherung der Daten von MyJournal wird in diesem Kapitel mithilfe der API `AsyncStorage` umgesetzt. Zusätzlich besprechen wir, wie der Standort eines Nutzers bestimmt werden kann und wie aktuelle Wetterdaten aus dem Web zur Anzeige in die App integriert werden.

Kapitel 8: Navigation zwischen mehreren Screens mit Tabs
: Zusätzliche Screens für eine Fotogalerie und Einstellungen werden durch eine Navigation mit Tabs in MyJournal eingebaut. Wir verwenden hierfür `TabNavigator` aus der Bibliothek *react-navigation*.

Kapitel 9: Detailansicht und Editor mit StackNavigator einbinden
: Die Navigation wird um eine Detailansicht und einen Editor für Tagebucheinträge erweitert. Dies geschieht mittels `StackNavigator`.

Kapitel 10: Auf Touchgesten reagieren und Animationen anzeigen
: Sie lernen, wie mit den APIs `PanResponder` und `Animated` ein Tagebucheintrag durch eine Wischbewegung gelöscht werden kann.

Entwicklung und Abbildungen der Beispiel-Apps

Ich gehe davon aus, dass Ihnen ein Android-Smartphone oder ein iOS-Gerät zur Verfügung steht, damit Sie die Beispiel-Apps in diesem Buch testen können. Im Prinzip werden die Apps parallel für Android und iOS entwickelt. Es genügt, dass Sie die Apps hauptsächlich auf einem Gerät Ihrer Wahl testen und nur gelegentlich einen Test auf der anderen Plattform durchführen (sofern ein entsprechendes Gerät vorhanden ist). Bitte beachten Sie, dass die Beispiel-Apps nur für Smartphones entwickelt werden – für Tablets mit größeren Bildschirmen werden wir keine speziellen Layouts der Apps erstellen.

Abbildungen dienen regelmäßig der Veranschaulichung der Funktionalitäten, die wir in den Apps implementieren werden. Diese zeigen in der Regel nebeneinander Screenshots der Android-Version und der iOS-Variante an.

Verwendung von Codebeispielen

Im Buch gibt es viele Auszüge aus Programmcode, mit denen relevante Änderungen in den Beispiel-Apps aufgezeigt werden. Dabei werden oftmals nur die Passagen im Code gezeigt, die geändert wurden. Mit Kommentaren wird angedeutet, welche Abschnitte im Code unverändert bleiben:

```
// ... die import-Anweisungen bleiben unverändert ...

render() {
  return <Text>Hello World</Text>;
}

// ... der Rest bleib unverändert ...
```

Von der Webseite zum Buch (*www.behrends.io/react-native-buch*) können Sie für jedes Kapitel die relevanten Codebeispiele herunterladen. Dabei werden die von Änderungen betroffenen Dateien nach einem größeren Entwicklungsschritt zusammengefasst, das heißt, einzelne Beispiele stehen nicht als Download bereit, sondern sind in den Dateien eines bestimmten Stands der Entwicklung enthalten.

Der Code zur Beispiel-App MyJournal, die über den Großteil der Kapitel entwickelt wird, ist online in einem github-Repository zu finden (*github.com/behrends/MyJournal*). Die Commit-Historie spiegelt im Prinzip die im Buch beschriebenen Änderungen wider, sodass Sie diese im Code nachvollziehen können.

Übungen

Am Ende einiger Abschnitte oder Kapitel finden Sie Übungen. Diese geben Ihnen die Möglichkeit, die vorgestellten Konzepte zu wiederholen, anzuwenden und schließlich zu vertiefen. Lösungsansätze und Code zu den Übungen finden Sie auf der Webseite zum Buch (*www.behrends.io/react-native-buch*).

Konventionen, denen dieses Buch folgt

Die folgenden typografischen Konventionen werden in diesem Buch verwendet:

Kursiv
: Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateiendungen.

`Konstante Zeichenbreite`
: Wird für Programmlistings und für Programmelemente in Textabschnitten wie Namen von Variablen und Funktionen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter verwendet.

`Konstante Zeichenbreite, fett`
: Kennzeichnet Befehle oder anderen Text, den der Nutzer wörtlich eingeben sollte.

Dieses Symbol steht für einen Tipp oder eine Empfehlung.

Dieses Symbol steht für einen allgemeinen Hinweis.

Dieses Symbol steht für eine Warnung oder erhöhte Aufmerksamkeit.

Errata und Updates

Ein Buch ist niemals vollständig fehlerfrei. Sie finden daher auf der Webseite unter *www.behrends.io/react-native-buch* eine Liste mit Errata zu diesem Buch.

Wie bei vielen Open-Source-Projekten üblich, kann es sich ergeben, dass sich bestimmte Aspekte in der Konfiguration oder Funktionsweise von React Native nach Erscheinen dieses Buchs grundlegend ändern, sodass es zu Abweichungen gegenüber den Buchinhalten kommen kann. Dabei kann es sich um planmäßige Änderungen und dokumentierte *Breaking Changes* in React Native, aber auch in Android oder iOS handeln. Sollten Sie also auf Probleme bei der Einrichtung von React Native oder im Zusammenhang mit Codebeispielen stoßen, empfiehlt sich ebenfalls ein Blick auf die Webseite zum Buch, auf der ich hierzu nützliche Hinweise geben werde (*www.behrends.io/react-native-buch*).

Wenn Sie in den Codebeispielen zur App MyJournal einen Fehler finden und mir diesen mitteilen wollen, können Sie gern einen *Issue* im entsprechenden github-Repository einstellen (*github.com/behrends/MyJournal*) oder Verbesserungsvorschläge per *Pull Requests* einreichen.

Die Codebeispiele in diesem Buch wurden mit den Versionen 0.48.3 von React Native und 21.0.0 von Expo entwickelt.

Danksagungen

Mein Dank gilt allen, die dieses Buch ermöglicht und bei seiner Erstellung mitgeholfen haben. Dem Verlag O'Reilly danke ich für die Möglichkeit, ein Fachbuch zu React Native zu schreiben und dies im deutschsprachigen Raum zu veröffentlichen. Allen Mitarbeitern des Verlags, die an diesem Projekt beteiligt waren, sei hiermit herzlich gedankt. Ein großes Dankeschön geht an die Lektorin Ariane Hesse, die das Buchprojekt von Anfang an begleitet und in vielerlei Hinsicht unterstützt hat. Mein besonderer Dank gilt meinem Bruder Knut, der sich als erster Leser des Buchs mit den Codebeispielen auseinandergesetzt und dabei viele Unstimmigkeiten gefunden hat, die ich hoffentlich beseitigen konnte. Und natürlich danke ich meiner Frau und meiner Tochter – nicht nur dafür, dass sie mich manchmal nach dem frühmorgendlichen Schreiben vom Laptop losgeeist haben.

Kontakt

Für Rückmeldungen jeder Art können Sie gern direkt mit mir in Kontakt treten. Meine E-Mail-Adresse lautet:

erik@behrends.io

Ich wünsche Ihnen viel Spaß und Erfolg mit React Native!

KAPITEL 1

React Native kurz vorgestellt

React Native ist ein Framework zur plattformübergreifenden Entwicklung nativer Apps mit JavaScript. Es wurde im Frühjahr 2015 zunächst für iOS als Open-Source-Projekt veröffentlicht. Seit 2015 wird auch Android als Zielplattform unterstützt. Zwar wurde React Native von Facebook initiiert, aber seit seiner Veröffentlichung hat sich eine große Gemeinschaft von Programmierern und Unternehmen gebildet, die an der Weiterentwicklung des Frameworks arbeitet.

React Native basiert auf der Webbibliothek React und folgt im Wesentlichen den gleichen Prinzipien zur Erstellung deklarativer Benutzeroberflächen mit JavaScript, wodurch der Zugang zur Entwicklung nativer Apps insbesondere für Webentwickler attraktiv und relativ einfach ist. Nicht zuletzt deswegen hat React Native ähnlich wie React in den letzten Jahren eine große Beliebtheit und eine kritische Masse von Nutzern erreicht. Sowohl React als auch React Native gehören zu den 20 populärsten Projekten auf GitHub.com.[1] Es gibt einige Beispiele von bekannten und viel genutzten Apps, die mit React Native entwickelt wurden. Dazu gehören unter anderem Apps von Facebook, Instagram, Airbnb, Tesla, Uber und Walmart.[2]

Weshalb begeistert und fasziniert React Native? Die wichtigsten Gründe werde ich im Folgenden skizzieren. Anschließend beschreibe ich, wie das Framework prinzipiell funktioniert und wie es dadurch möglich wird, native Apps mit JavaScript zu programmieren.

React Native: viele Vorteile, wenige Einschränkungen

Es gibt verschiedene Vorgehensweisen, um mobile Apps für die beiden wesentlichen Plattformen Android und iOS zu entwickeln. Für die Entwicklung nativer Apps stellen Google und Apple mächtige Werkzeuge zur Verfügung. Die Unter-

1 Ende November 2017 hatten React und React Native mehr als 81.000 bzw. 56.000 Sterne.

2 Weitere Beispiele sind auf der offiziellen Webseite von React Native unter *facebook.github.io/react-native/showcase.html* zu finden.

schiede zwischen den beiden herkömmlichen Ansätzen, eine native App für Android bzw. iOS zu entwickeln, sind jedoch beträchtlich. Dies illustriert folgende Auflistung, die allerdings keinen Anspruch auf Vollständigkeit erhebt:

Tabelle 1-1: Unterschiede in der nativen App-Entwicklung zwischen Android und iOS

	Android	iOS
Programmiersprache	Java oder Kotlin	Objective-C oder Swift
Entwicklungsumgebung	Android Studio	XCode
Unterstützte Betriebssysteme	Linux, macOS, Windows	macOS
Erstellung von UIs	deklarativ mit XML-Layout oder programmatisch mit Java	hauptsächlich durch grafische Tools in XCode (Interface Builder)

Soll nun eine native App für Android und iOS mit den herkömmlichen Ansätzen der Hersteller entwickelt werden, ist es offensichtlich, dass dies in der Regel nur mit eigenständigen Teams möglich sein wird, die sich auf die jeweilige Plattform spezialisieren. Dadurch werden nicht bloß hohe Kosten verursacht, sondern es wird auch ein erhöhter Aufwand in Bezug auf die Projektkoordination erforderlich sein. Dies ist die größte Herausforderung, wenn eine native App für beide Plattformen auf herkömmliche Weise entwickelt werden soll.

Ein weiterer Nachteil der herkömmlichen Ansätze entsteht aus der Tatsache, dass jede Änderung an einer App dazu führt, dass die App kompiliert werden muss. Bevor die Änderung auf dem Smartphone getestet werden kann, vergehen für eine Android-App mindestens einige Sekunden und in einem iOS-Projekt oftmals sogar deutlich mehr Zeit (von den häufig berichteten Abstürzen von XCode ganz zu schweigen). Je komplexer die App wird, desto mehr Zeit verbringt ein Programmierer mit Warten. Den meisten Programmierern ist sicherlich bewusst, dass sich dies negativ auf ihre Produktivität auswirken und eine Belastungsprobe für ihre Gelassenheit darstellen kann.

React Native bietet Abhilfe und reduziert sowohl den Entwicklungsaufwand als auch die Reaktionszeiten bei der Entwicklung nativer Apps. Es ist einerseits ein plattformübergreifendes Framework für Android und iOS, wodurch ein Team in die Lage versetzt wird, eine App für beide Plattformen parallel mit nur einer Codebasis zu entwickeln. So werden wir auch in diesem Buch bei der Entwicklung der App *MyJournal* ab Kapitel 4 vorgehen. Andererseits wird bei der Programmierung mit React Native jede Änderung am Code fast augenblicklich in der App sichtbar. Ermöglicht wird dies durch Features wie *Live Reload* und *Hot Reloading*, die Sie im Laufe des Buchs kennenlernen werden. Neben diesen wichtigen Eigenschaften hat React Native weitere Vorteile, auf die ich kurz eingehen möchte:

Native Benutzeroberflächen
: React Native erzeugt performante Apps mit nativen UI-Elementen für Android und iOS. Dies ist ein großer Vorteil gegenüber hybriden Frameworks wie z. B. Cordova und Ionic, deren Anwendungscode in einem WebView aus-

geführt wird. Die Benutzeroberfläche wird dort mit Webtechnologien erstellt, wodurch UI-Elemente lediglich nachgebildet werden, was sich negativ auf das Erscheinungsbild und die Performance einer App auswirken kann.

Die Basis: JavaScript und React
: Da JavaScript die Programmiersprache für React Native ist, können Webentwickler mit weniger Aufwand Apps erstellen, als wenn sie zuerst eine andere Programmiersprache lernen müssten. Durch JavaScript wird es prinzipiell möglich, allgemeine Geschäftslogik ohne spezifischen Code für die App in Webprojekten wiederzuverwenden. Zusätzlich profitiert React Native von dem deklarativen Programmiermodell in React, das einen eleganten und produktiven Ansatz zur Erstellung von Benutzeroberflächen bietet. Gerade dieser Vorteil ist ein Grund für die wachsende Popularität von React Native in den letzten Jahren.

Aktives Umfeld mit vielen Erweiterungen und Werkzeugen
: React Native wird von vielen Entwicklern und mehreren Unternehmen aktiv weiterentwickelt. Es stehen viele nützliche Erweiterungen und Bibliotheken zur Verfügung, die z.B. UI-Komponenten implementieren, die nicht direkt von React Native angeboten werden. Hervorzuheben ist hierbei das Start-up Expo (*expo.io*), das verschiedene nützliche quelloffene Werkzeuge für React Native bereithält. Um nur ein Beispiel zu nennen: Die Expo-App ermöglicht es in Kombination mit React Native, native Apps für das iPhone ohne Apple-Rechner zu entwickeln. Diese App werden wir daher auch in diesem Buch vorstellen und verwenden.

Weitere Plattformen neben Android und iOS
: Apps plattformübergreifend mit einer Codebasis für Android und iOS programmieren zu können, ist bereits ein großer Produktivitätsgewinn. Es gibt jedoch zusätzliche Plattformen, für die Apps mit React Native entwickelt werden können. Dazu gehören z.B. Apple TV und Microsofts Universal Windows Platform. Architektur und Funktionsweise von React und React Native erlauben es sogar, Anwendungen im zukunftsträchtigen Bereich der virtuellen Realität umzusetzen (siehe React VR unter *facebook.github.io/react-vr*).

Nativer Code kann eingebettet werden
: In React Native kann einerseits nativer Code für Android und iOS eingebettet werden, sodass native Bestandteile anderer Projekte oder Bibliotheken wiederverwendet werden können. Umgekehrt ist es möglich, React Native in bestehenden App-Projekten einzusetzen, die bisher mit den herkömmlichen Ansätzen von Google bzw. Apple entwickelt wurden. Somit ist es möglich, neue Funktionalitäten in diesen Projekten prototypisch und plattformübergreifend mit React Native umzusetzen.

Meines Erachtens ist React Native das einzige Framework für die plattformübergreifende Entwicklung nativer Apps, das solch eine Fülle von Möglichkeiten und Funktionen bietet. Es gibt aber auch Einschränkungen bei dem Framework, die es zu beachten gilt.

Manche Rahmenbedingungen erfordern nativen Code
: Es kann Situationen geben, in denen es nicht möglich ist, bestimmte Funktionalitäten einer App mit React Native umzusetzen. Dies könnte ein besonderes Bedienelement sein, für das es (noch) keine Komponente im Framework gibt. Auch decken die APIs von React Native nicht alle Gerätefunktionen ab, sodass z. B. bestimmte Sensordaten nicht abgefragt werden können. In diesen Fällen kann zwar nativer Code eingebettet werden, aber es bedeutet auch, dass mindestens ein Teammitglied sich mit den nativen Aspekten der App-Entwicklung auskennen sollte. Dieses Wissen wird außerdem dann relevant, wenn tiefer liegende Fehler im Code einer App untersucht werden müssen, falls zukünftig Abwärtskompatibilität sichergestellt werden muss und wenn die App in den App-Stores veröffentlicht werden soll.

Das Framework hat noch keine stabile Version erreicht
: Auch mehr als zwei Jahre seit seiner Einführung befindet sich React Native in aktiver Entwicklung mit monatlichen Veröffentlichungen neuer Versionen. Dabei kommt es gelegentlich zu Änderungen am Framework, wodurch APIs sich ändern oder sogar entfernt werden. Wann eine stabile Version »1.0« erreicht wird, ist im Herbst 2017 noch nicht absehbar. Dennoch bin ich überzeugt davon, dass React Native bereits jetzt ohne große Risiken für App-Projekte eingesetzt werden kann – was auch dadurch belegt wird, dass das Framework von vielen namhaften Unternehmen verwendet wird.

Insgesamt stellt React Native eine faszinierende Technologie dar, und wie das Beispiel von React VR zeigt, könnte es in Zukunft noch weitere spannende Zielplattformen und Anwendungsmöglichkeiten geben. Am Beispiel der in diesem Buch entwickelten App MyJournal können Sie sich ein Bild davon machen, wie eine recht umfangreiche Anwendung komplett mit React Native in JavaScript umgesetzt werden kann, ohne dass nativer Code benötigt wird.

Architektur und Funktionsweise des Frameworks

JavaScript wird in verschiedenen Frameworks zur App-Entwicklung als primäre Programmiersprache eingesetzt. Beispiele beliebter Frameworks sind Cordova und Ionic, die den JavaScript-Code der App in einem WebView ausführen und dort die Benutzeroberfläche mit HTML und CSS ähnlich wie in einem Browser darstellen. Eine solche App entspricht daher vereinfacht gesagt einer lokalen Webanwendung, was zu Einschränkungen in der Verfügbarkeit nativer Funktionalitäten führt. Im Gegensatz dazu erhalten wir mit React Native in JavaScript programmierte Apps, die zur Laufzeit vollen Zugriff auf die nativen UI-Elemente und APIs der jeweiligen Plattform haben. Ermöglicht wird das durch die Architektur des Frameworks React Native. Sein Aufbau lässt sich in drei Ebenen unterteilen:

Laufzeitumgebung des Anwendungscodes (JavaScript)
: In einer React-Native-App werden Anwendungscode und Geschäftslogik in JavaScript programmiert. Für diesen Code muss auf der Zielplattform eine

Laufzeitumgebung für JavaScript existieren. In Android und iOS stehen uns solche Umgebungen zur Verfügung, die es ermöglichen, JavaScript in einer nativen App auszuführen.[3] Dieser Code läuft in einem eigenen Thread. Im JavaScript-Code wird insbesondere die Benutzeroberfläche gemäß den Prinzipien von React deklariert, was zu einer Darstellung echter, nativer UI-Elemente führt.[4]

Bridge
: Die Anweisungen zur Verwendung nativer UI-Elemente oder APIs werden aus der JavaScript-Umgebung als Nachrichten über die sogenannte *Bridge* an native Module kommuniziert. Umgekehrt treten verschiedene Ereignisse (*Events*) in den nativen Modulen der App auf (z. B. das Antippen eines Icons). Diese Events müssen wiederum zur Verarbeitung an den Code in der JavaScript-Umgebung weitergegeben werden, damit das gewünschte Verhalten für eine Benutzeraktion ausgeführt wird (z. B. das Versenden einer E-Mail). Die Kommunikation zwischen JavaScript und den nativen Modulen findet in React Native also in beide Richtungen über die Bridge statt.

Native Module
: Aus der JavaScript-Umgebung gelangen Anweisungen über die Bridge in den nativen Teil der App, in dem insbesondere auf mobilen Plattformen der native UI-Thread läuft. Die Anweisungen werden hier an die zuständigen nativen Module delegiert und von diesen verarbeitet. Dabei kann es sich um native UI-Elemente handeln. Beispielsweise könnte ein Texteingabefeld mit einem bestimmten Wert befüllt werden. Eine Anweisung könnte auch Daten von einer nativen API abfragen, wie z. B. die aktuellen Koordinaten des GPS-Sensors.

Der Aufbau dieser Architektur ist in Abbildung 1-1 skizziert.

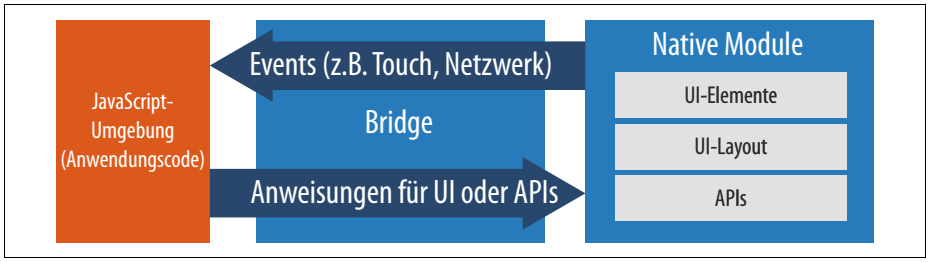

Abbildung 1-1: Die Architektur des Frameworks React Native. Der JavaScript-Code kommuniziert via Bridge mit den nativen Modulen der App.

Die Kommunikation über die Bridge erfolgt in Form von Nachrichten, die im JSON-Format serialisiert werden. Diese werden asynchron übertragen, sodass sich

3 In iOS ist bereits eine Umgebung namens JavaScriptCore vorhanden, und für Android wird mit React Native eine JavaScript-Umgebung mitgeliefert.

4 Die Funktionsweise von React wird im Abschnitt »React: ein deklaratives Programmiermodell für UI-Komponenten« auf Seite 44 beschrieben.

die Threads nicht gegenseitig blockieren, und es finden zusätzliche Optimierungen statt (z. B. werden Nachrichten häufig zusammengefasst und gemeinsam übertragen). Bei der Programmierung mit React Native findet die Kommunikation über die Bridge im Hintergrund statt, sodass wir die konkrete Funktionsweise in der Regel nicht beachten müssen.

Durch diese Architektur entstehen einige Vorteile. Die JavaScript-Umgebung ermöglicht es, plattformübergreifend in nur einer Programmiersprache Apps zu entwickeln. Prinzipiell wäre es möglich, in dieser Architektur Laufzeitumgebungen für andere Programmiersprachen zu verwenden, jedoch ist JavaScript als eine der meistverwendeten Sprachen, die relativ leicht zu erlernen ist, eine naheliegende Wahl und insbesondere für Webentwickler attraktiv.

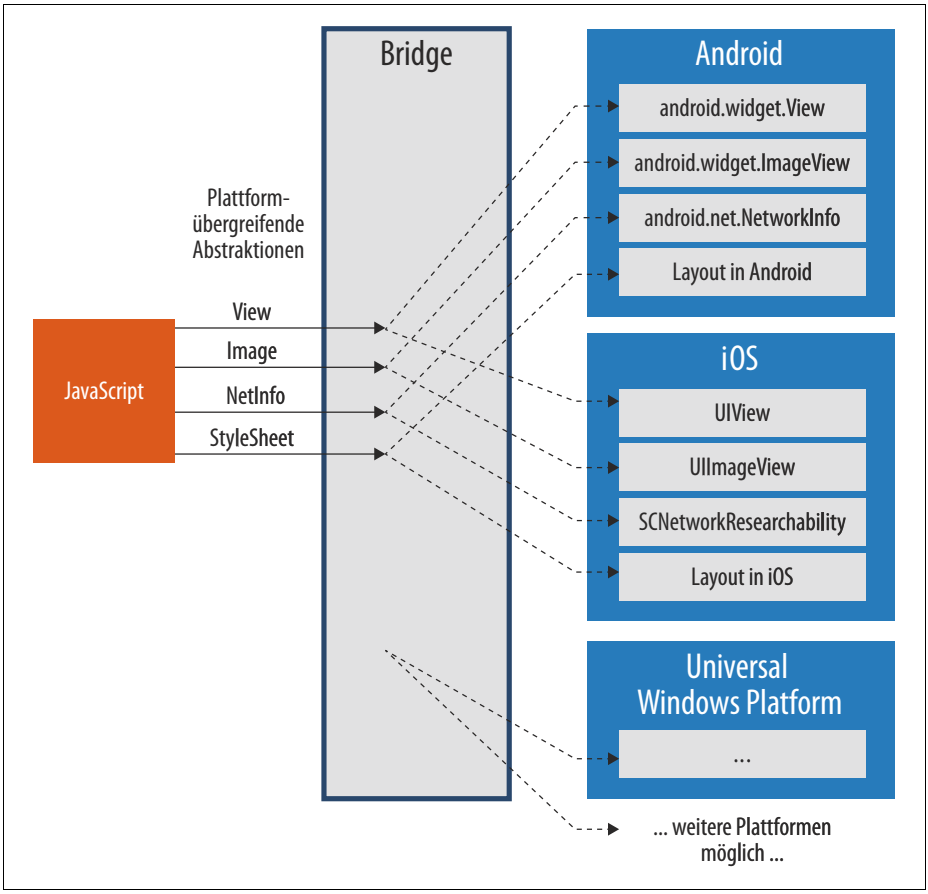

Abbildung 1-2: Die Unterstützung verschiedener Zielplattformen wird durch plattformübergreifende Abstraktionen über die Bridge erreicht.

Zusätzlich verbirgt die Bridge die Implementierungsdetails der nativen Module, sodass diese im Anwendungscode als abstrahierte, plattformübergreifende Kom-

ponenten und APIs zur Verfügung stehen. Wird in JavaScript z. B. ein `View`-Element im UI verwendet, führt dieses in Android zum Einsatz einer Instanz der Klasse `android.widget.View`, und in iOS wird ein `UIView`-Objekt erzeugt. Entsprechend gibt es in React Native für einige Komponenten, APIs und sogar für die Gestaltung und das Layout der Benutzeroberfläche plattformübergreifende Abstraktionen, wie in Abbildung 1-2 angedeutet. Um weitere Zielplattformen durch React Native zu bedienen, muss lediglich die Bridge mit den passenden Abstraktionen erweitert werden, wie z. B. für die Universal Windows Platform geschehen (siehe *github.com/Microsoft/react-native-windows*).

Während der Entwicklung einer React-Native-App arbeitet der Programmierer zum Großteil mit dem Anwendungscode in der JavaScript-Umgebung. Für solche Änderungen am Code muss die App nicht kompiliert werden, denn es reicht aus, lediglich den geänderten JavaScript-Code an die App zu übertragen. Im Entwicklungsmodus verbindet sich daher die React-Native-App mit einem Webserver, der auf dem Rechner des Programmierers läuft und Änderungen automatisch an die App überträgt (dies ist der sogenannte *Packager*). Entwickler erhalten dadurch fast augenblicklich Feedback – viel schneller als bei den herkömmlichen Ansätzen zur nativen App-Entwicklung. Dies werden Sie selbst im folgenden Kapitel erfahren, denn wir werden gleich eine einfache App mit React Native entwickeln.

Zusammenfassung

React Native ist zunächst ein plattformübergreifendes Framework, das bei der Entwicklung nativer Apps für Android und iOS eine einheitliche Codebasis ermöglicht und dadurch Aufwand und Kosten solcher Projekte signifikant reduzieren kann. Zusätzlich zu diesem sofort einleuchtenden Argument bietet die Software jedoch einige weitere Vorteile gegenüber den herkömmlichen Ansätzen der Entwicklung nativer Apps. Hervorzuheben ist hierbei vor allem die Tatsache, dass Änderungen am Code während der Entwicklung sehr schnell in der App erscheinen und getestet werden können. Schließlich lässt die Architektur des Frameworks weitere Zielplattformen zu, sodass React Native das Potenzial hat, in Zukunft auch für Bereiche jenseits mobiler Apps eingesetzt zu werden.

KAPITEL 2

Erste Schritte mit React Native

In diesem Kapitel werden wir zunächst notwendige Vorbereitungen zur Installation von React Native vornehmen, sodass wir anschließend unsere erste App entwickeln können. Es handelt sich um eine einfache App, mit der wir durch das Drücken auf einen Knopf etwas zählen können (z. B. Schritte oder Treppenstufen). Obwohl die Komplexität dieser App aus Benutzersicht als gering erscheinen mag, werden wir während der gemeinsamen Entwicklung dieser App einigen grundlegenden Konzepten des Frameworks React Native begegnen. Dadurch erhalten Sie einen ersten Eindruck davon, wie Apps mit React Native plattformübergreifend für Android und iOS entwickelt werden. Wir nennen diese App *StepCounter*, und sie wird am Ende dieses Kapitels aussehen wie in Abbildung 2-1 dargestellt.

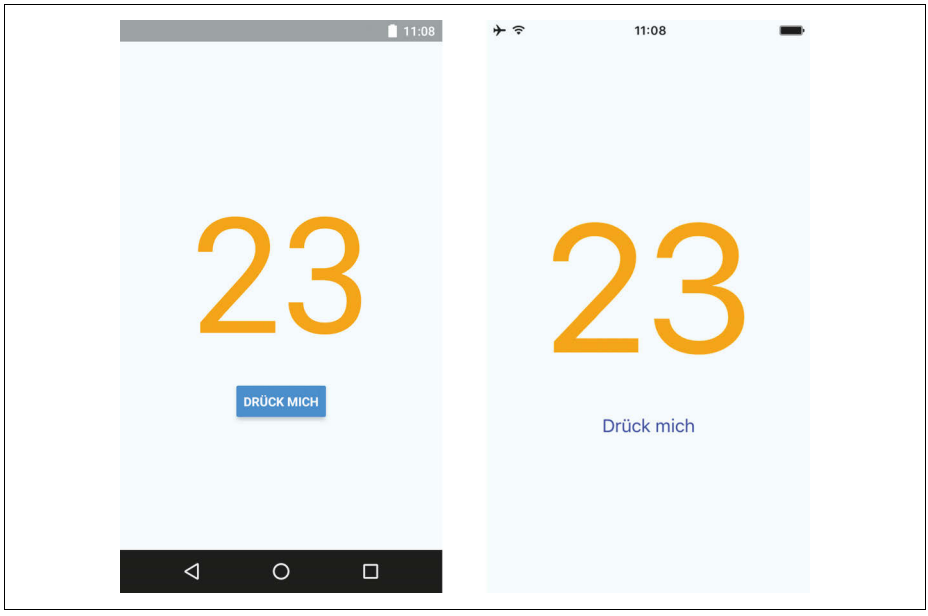

Abbildung 2-1: Die App StepCounter am Ende dieses Kapitels. Die meisten Screenshots in diesem Buch zeigen links die Android-App und rechts die iOS-Version.

Ich habe die Installation von React Native in verschiedenen Betriebssystemen getestet und bin dabei auf einige Stolperstellen gestoßen, auf die ich an den entsprechenden Stellen in diesem Kapitel hinweisen werde. Falls Sie bei der Installation auf Probleme stoßen, die nicht erwähnt werden, könnte das z. B. daran liegen, dass sich das Vorgehen zur Installation von React Native nach Drucklegung des Buchs geändert hat. Prüfen Sie in diesem Fall bitte auf der Webseite zum Buch, ob es dort zusätzliche Informationen und Hilfestellungen gibt (*behrends.io/react-native-buch*).

Vorbereitungen und Installation

React Native lässt sich so installieren, dass Sie nach wenigen Minuten mit der Programmierung mobiler Apps beginnen können. Sie benötigen lediglich einen Rechner mit Internetverbindung und ein iPhone oder ein Android-Gerät. Dabei spielt es keine Rolle, ob das Betriebssystem Ihres Rechners Linux, macOS oder Windows ist. Mit React Native ist es sogar möglich, ohne einen Apple-Rechner mit der Entwicklung nativer Apps für das iPhone zu beginnen – beim herkömmlichen Ansatz wird hierfür das Programm XCode benötigt, das nur für macOS verfügbar ist.

Sollte Ihnen weder ein iPhone noch ein Android-Gerät zur Verfügung stehen, können Sie alternativ einen Android-Emulator verwenden oder auf einem Apple-Rechner den iPhone-Simulator einsetzen. Dazu müssen Sie zuerst die entsprechenden Entwicklungswerkzeuge Android Studio bzw. XCode installieren, wie in der offiziellen Dokumentation zu React Native beschrieben (*facebook.github.io/react-native/docs*).

Es sind einige Vorbereitungen zu treffen, bevor wir mit React Native arbeiten können. Für die Programmierung müssen Sie sich für einen Editor entscheiden, und wir werden die App *Expo* verwenden, um unsere Apps auf dem Smartphone zu testen. Außerdem benötigen Sie die JavaScript-Plattform *Node.js*, die auf Ihrem Rechner mindestens in der Version 6 vorhanden sein muss. Ich werde Ihnen nun die verschiedenen Maßnahmen zur Vorbereitung erläutern.

Einen Editor für die Programmierung auswählen

Die Programmierung von Apps mit React Native ist im Prinzip mit jedem Editor möglich. Auf zwei Editoren möchte ich besonders hinweisen: *Atom* (*atom.io*) und *Visual Studio Code* (kurz *VS Code*, *code.visualstudio.com*). Sowohl Atom als auch VS Code können auf Linux, macOS und Windows eingesetzt werden und werden als kostenlose, quelloffene Downloads angeboten. Beide Editoren sind ausgereift und haben einen vergleichbaren Funktionsumfang. Ich habe zur Entwicklung der Apps in diesem Buch VS Code eingesetzt und kann daher diesen Editor empfehlen. Gleichwohl können Sie ohne Bedenken Ihren bevorzugten Editor verwenden.

Erweiterungen für React Native im Editor

Sowohl Atom als auch VS Code sind erweiterbar, und für beide Editoren können Sie spezielle Erweiterungen für React Native finden. Für Atom gibt es das Package *Nuclide* (*nuclide.io*), das von Facebook entwickelt wurde, und für VS Code steht Ihnen die Erweiterung *React Native Tools* zur Verfügung, die Sie über den Bereich *Extensions* in VS Code suchen und installieren können. Die erwähnten Erweiterungen für Atom und VS Code bieten Ihnen neben Debugging-Funktionalitäten weitere nützliche Funktionen für React Native. Allerdings benötigen Sie für die Arbeit mit diesem Buch nicht unbedingt eine dieser Erweiterungen.

Die App Expo auf das Smartphone laden

Wie bereits am Ende des vorigen Kapitels beschrieben, besteht in der Architektur von React Native eine strikte Trennung zwischen Anwendungscode, der in JavaScript geschrieben wird, und den nativen Modulen für die UI-Elemente und APIs der Zielplattform (siehe Abbildung 2-2). Der Programmcode, der spezifisch für eine React-Native-App ist, besteht also oftmals nur aus JavaScript-Code. Insbesondere in diesem Buch wird das der Fall sein, da wir ausschließlich in JavaScript programmieren und die Einbettung nativen Codes nicht behandeln werden. Daher können wir die App Expo verwenden, die den Einstieg in die Entwicklung mit React Native stark vereinfacht. Der Ablauf der Programmierung mit React Native und Expo ist in Abbildung 2-2 grob skizziert.

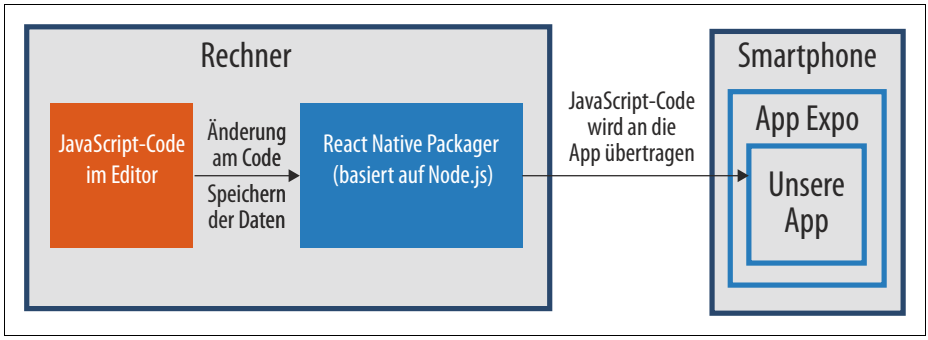

Abbildung 2-2: Änderungen am Code werden vom React Native Packager automatisch und schnell an die App Expo auf dem Smartphone übertragen.

Wenn wir auf dem Rechner programmieren, führen wir Änderungen am JavaScript-Code im Editor durch und speichern die zugehörigen Dateien ab. Während der Entwicklung läuft auf dem Rechner zusätzlich der *React Native Packager*, ein in Node.js geschriebener Webserver, der Änderungen an den Projektdateien überwacht und diese automatisch an die App Expo zum Smartphone überträgt. Dazu müssen wir lediglich die App Expo über das Netzwerk mit dem React Native Packager auf dem Rechner verbinden. Expo ist eine App, die die benötigten nativen

Komponenten und APIs des Frameworks React Native bereitstellt und daher in der Lage ist, den JavaScript-Code unserer App dynamisch auf dem Smartphone anzuwenden. Wie Sie sehen werden, führt Expo unser App-Projekt wie eine »echte App« auf dem Smartphone aus. Expo ist eine etablierte Lösung für den Einstieg in die Programmierung mit React Native.

Aus diesem Grund setzen wir für die Beispiel-Apps in diesem Buch die mobile App Expo ein, die von dem gleichnamigen Start-up Expo als Open-Source-Projekt entwickelt wird und kostenlos im App Store für iOS bzw. im Play Store für Android heruntergeladen werden kann. Navigieren Sie dazu im Browser Ihres Smartphones zur Expo-Webseite mit der URL *expo.io*. Dort finden Sie Links, die zu Expo im App Store bzw. im Play Store führen. Installieren Sie also nun Expo auf Ihrem Smartphone.

Bemerkungen zu Expo

Expo trägt erst seit Anfang März 2017 diesen Namen. Vorher hießen das Projekt sowie das dahinterstehende Unternehmen *Exponent*, sodass in manchen älteren Videos oder Blogs dieser frühere Name noch auftaucht. Sie benötigen für dieses Buch zunächst ausschließlich die mobile App Expo, die Sie bitte auf Ihrem Smartphone aus dem App Store bzw. Play Store herunterladen. Weitere Werkzeuge aus dem Expo-Umfeld brauchen Sie nicht auf Ihrem Rechner zu installieren. Insbesondere die Entwicklungsumgebung Expo XDE werden wir nicht einsetzen.

Node.js auf dem Rechner installieren

Der im vorigen Abschnitt erwähnte React Native Packager basiert auf der JavaScript-Plattform Node.js. Damit Sie Apps mit React Native entwickeln können, muss daher auf Ihrem Rechner Node.js mindestens in der Version 6 vorhanden sein. Um Node.js zu installieren, befolgen Sie die Anweisungen für das Betriebssystem Ihres Rechners, wie auf der Webseite von Node.js beschrieben (siehe *nodejs.org*).

Paketverwaltung in Linux und macOS

Auf einem Mac empfehle ich die Verwendung von Homebrew (*brew.sh*), wodurch sich zusätzliche Programme wie Node.js leicht installieren lassen. Wenn Sie mit Linux arbeiten, können Sie Node.js in Ihrer Linux-Distribution mit der Paketverwaltung installieren. Allerdings ist es in diesem Fall wichtig, dass Node.js mindestens in der Version 6 eingerichtet wird. Sonst wird React Native nicht funktionieren. Stellen Sie zusätzlich sicher, dass das Programm `npm` (*node package manager*) vorhanden ist. Gegebenenfalls müssen Sie `npm` zusätzlich mit der Paketverwaltung installieren.