Nils Hartmann · Oliver Zeigermann

React

Grundlagen, fortgeschrittene Techniken und Praxistipps – mit TypeScript und Redux

dpunkt.verlag

Nils Hartmann ist freiberuflicher Softwareentwickler, -architekt, Trainer und Coach. Er hat langjährige Erfahrung in der Entwicklung mit Java sowie JavaScript/TypeScript, ist Autor von Fachartikeln und unterstützt Teams mit Trainings und Coaching beim Einstieg in JavaScript und die Entwicklung von modernen Webanwendungen. Mehr unter: *https://nilshartmann.net*

Oliver Zeigermann ist Entwickler, Architekt, Berater und Coach aus Hamburg. Er hat über Jahrzehnte in vielen unterschiedlichen Sprachen und mit vielen Technologien und Ansätzen Software entwickelt. Er ist Autor zahlreicher Fachbücher im JavaScript und React-Bereich, sowie Experte für Machine und Deep Learning. Mehr unter: *http://zeigermann.eu/*

Nils Hartmann · Oliver Zeigermann

React

Grundlagen, fortgeschrittene Techniken und Praxistipps – mit TypeScript und Redux

2., überarbeitete und erweiterte Auflage

dpunkt.verlag

Nils Hartmann, Oliver Zeigermann

Lektorat: René Schönfeldt
Projektkoordinierung/Lektoratsassistenz: Anja Weimer
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Gerhard Alfes, mediaservice, Siegen, *www.mediaservice.tv*
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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 978-3-86490-552-0
PDF 978-3-96088-419-4
ePub 978-3-96088-420-0
mobi 978-3-96088-421-7

2., überarbeitete und erweiterte Auflage 2020

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.

5 4 3 2 1 0

Inhaltsverzeichnis

Teil II React

Teil I

Einstieg

1 Einleitung

1.1 Was ist React?

React[1] ist eine Open-Source-JavaScript-Bibliothek, mit der du Webanwendungen, sogenannte Single-Page-Anwendungen (kurz SPAs), erstellen kannst. Die Bibliothek wird von Facebook entwickelt und sowohl für facebook.com als auch für Instagram benutzt. Aber auch darüber hinaus ist React sehr weit verbreitet und wird zum Beispiel von Netflix, Airbnb, Twitter oder der Online-Ausgabe des Wall Street Journals verwendet. Da React kaum Bedingungen an seine Umgebung stellt, ist es sehr flexibel einsetzbar. So gibt es auch Webseiten, die React nur für einige interaktive Teile ihres Angebots verwenden.

Single-Page-Anwendungen

Single-Page-Anwendungen (SPAs) zeichnen sich dadurch aus, dass sie vollständig auf dem Client, also im Browser, laufen. Sämtliche Interaktionen werden auf dem Client bearbeitet und die Darstellung der Seite wird mittels JavaScript erzeugt bzw. aktualisiert, um eine möglichst flüssige Darstellung zu erreichen. Mit dem Server werden JavaScript-Code, statische Assets (Bilder, Fonts etc.) und Daten ausgetauscht, aber kein HTML-Code. Zur Entwicklung von Single-Page-Anwendungen haben sich spezialisierte Frameworks und Bibliotheken (in erster Linie React, Angular, Vue oder Web Components) herausgebildet, die von der zugrunde liegenden DOM-API abstrahieren und die Entwicklung dieser Anwendungen vereinfachen.

Im Gegensatz zu SPAs findet bei serverseitig gerenderten Websites und Anwendungen (auch »klassische Webanwendungen« genannt) bei Interaktionen ein Server-Roundtrip statt, bei dem fertiger HTML-Code zurückgeliefert wird. Aus diesem Grund eignet sich dieser Ansatz nur für zumeist statische Websites, auch wenn diese mit JavaScript an einzelnen Stellen um interaktive Features erweitert werden können.

1 *http://reactjs.com/*

Den Kern von React bilden Komponenten und ihre Komposition zu einer Anwendung. Durch eine solche Komposition wird bestimmt, was dargestellt werden soll oder – aus einer anderen Perspektive – wie man den Zustand einer Anwendung in ihre Darstellung transformiert. Beispiele für den Zustand einer Anwendung können fachliche Dinge sein, wie die Anzahl ungelesener Nachrichten, ein Blogpost, der gerade in der Anwendung bearbeitet wird, oder der zurzeit angemeldete User. Aber auch technische Dinge, wie die Informationen, welches Menü gerade aufgeklappt oder welcher Eintrag in einer Combobox ausgewählt ist, sind Zustand der Anwendung.

Der Kern von React ist übrigens losgelöst vom Web: React kann in unterschiedlichen Szenarien funktionieren, so auch in nativen Anwendungen (zum Beispiel iOS oder Android).

In diesem Buch werden wir uns aber auf die Webentwicklung beschränken und daher auch davon ausgehen, dass du eine Webanwendung mit React bauen willst. Eine solche Anwendung wird früher oder später im DOM des Browsers – der Objektrepräsentation der dargestellten Elemente – gerendert werden. Dazu gibt es zwei Möglichkeiten. In der Regel wird die Anwendung auf Clientseite, also im Browser, gerendert. Du kannst deine Anwendung aber zusätzlich auf dem Server rendern lassen, sofern du dort eine JavaScript-Engine laufen lässt. Dann wird auf dem Server fertiger HTML-Code erzeugt, der zum Browser geschickt wird und dort vom Browser nur noch in den DOM umgewandelt und angezeigt werden muss. Von da an werden alle Updates im Browser gerendert. Das ist sehr praktisch zum Beispiel für eine schnelle erste Anzeige der Anwendung.

1.1.1 Komponenten

React-Anwendungen werden in Komponenten aufgeteilt. Eine Komponente enthält alles Notwendige, um sich darzustellen. Dabei trennt React nicht zwischen Template und Logik; sowohl UI und Logik sind in der Komponente enthalten. Als Ersatz für das Template wird in React der UI-Code direkt in den JavaScript-Code einer Komponente geschrieben. Das geschieht mit der React-eigenen Spracherweiterung JSX, die es ermöglicht, HTML-artigen Code in JavaScript einzubinden.

Ohne an dieser Stelle schon in die Details zu gehen, könnte eine einfache Komponente, die in React als Funktion implementiert werden kann, wie folgt aussehen:

```
import React from "react";

export default function HelloMessage() {
  return <h1>Hello, World</h1>;
}
```

Wenn diese Komponente in deiner Anwendung verwendet wird, sorgt React dafür, dass ein `h1`-Element in den DOM eingebaut wird, das den String »Hello, World« beinhaltet.

Deklarative Komponenten

»UI as a Function«

Du siehst an dieser Stelle ein weiteres, wichtiges Konzept von React: Komponenten werden ausschließlich deklarativ beschrieben. Eine Komponente gibt demnach immer nur genau die UI zurück, die dargestellt werden soll. Man spricht daher von »UI as a Function« in der React-Entwicklung, da – genau wie bei einer mathematischen Funktion – zu demselben Eingabeparameter (React: Zustand) immer derselbe Wert (React: UI) zurückkommt.

Zur Laufzeit kümmert sich React dann darum, dass auch der DOM im Browser entsprechend angepasst wird – und zwar unabhängig davon, wie der DOM zuvor aussah. Die einzelnen Übergänge von einer UI zur nächsten, also das jeweilige Aktualisieren des DOM im Brower, übernimmt React für dich und geht dabei sehr effizient und optimiert vor. Funktionen zum Erzeugen oder Löschen von Elementen (wie beispielsweise `createElement` oder `createAttribute` aus der DOM-API) brauchst du in React nicht.

Durch die deklarative Programmierung entfällt eine Menge Komplexität und Fehleranfälligkeit, die aus anderen Ansätzen bekannt sind, da eine wesentliche Fehlerquelle, die diversen Übergänge innerhalb einer Anwendung, nicht implementiert werden müssen. Auch das Testen von Komponenten wird dadurch sehr einfach. In deinem Test übergibst du deiner Komponente ihre Parameter und prüfst hinterher, dass die richtige UI zurückgeliefert wird – genau wie bei einer »normalen« Funktion, nur dass du UI-Elemente überprüfst und keine numerischen oder anderen Werte.

Eigenschaften einer Komponente

Im Folgenden sind die Eigenschaften einer Komponente zusammengefasst:

- *Komponenten in React repräsentieren einen fachlichen Teil einer Anwendung.* Das kann etwas sehr Einfaches (z.B. ein Eingabefeld) oder auch etwas Komplexes sein (z.B. ein gesamtes Formular). In der Regel werden einfache Komponenten zu komplexeren Komponenten und schließlich zu ganzen Anwendungen aggregiert. Eine Anwendung in React ist technisch genau dasselbe wie eine »einfache« Komponente.
- *Der Code einer Komponente besteht aus UI und Logik.* Es gibt in React in der Regel keine Trennung nach Techniken wie in anderen UI-Frameworks (z.B. Trennung in Model und View).

- *Es gibt in React keine Template-Sprache.* Stattdessen wird der UI-Code in JavaScript geschrieben. Um das möglichst einfach zu machen, wird der UI-Code mit der Spracherweiterung JSX geschrieben und sieht dann wie »HTML in JavaScript« aus. Der Tooling-Support (Editoren, IDEs etc.) für JSX ist sehr gut, sodass du trotzdem mit deinem Lieblingseditor (weiter) arbeiten kannst.
- *Komponenten werden deklarativ beschrieben.* Du gibst in der JSX-Notation immer an, welche UI nach dem Rendern der Komponente im Browser dargestellt werden soll. Dabei brauchst du dich aber nicht darum zu kümmern, wie es zu dieser Darstellung kommt. Unabhängig davon, was vor dem Rendern deiner Komponente im Browser angezeigt wurde, kümmert sich React darum, diese alte Darstellung in die neue, von dir gewünschte Darstellung zu transformieren.
- *Es gibt in React kein 2-Wege-Databinding zum Abgleich zwischen Model und UI.* Stattdessen aktualisierst du – z.B. nach einer Benutzerinteraktion – den Zustand deiner Komponente und React rendert die komplette Komponente mitsamt ihrer Unterkomponenten neu. Dadurch kann es nie zu Inkonsistenzen in der Darstellung kommen, weil du z.B. vergessen hast, einen Listener (rechtzeitig) an- oder abzumelden. Es wird auch davon gesprochen, dass React die UI als Funktion betrachtet: Abhängig von einem Parameter (dem Model) wird in React immer die gleiche UI zurückgeliefert. Genau wie in einer Funktion, die – sofern sie frei von Seiteneffekten ist – für denselben Parameter immer dasselbe Ergebnis zurückliefert.
- *Daten in Komponenten sind entweder Properties oder ein interner Zustand.* Properties werden der Komponente von außen, vom Verwender, übergeben. Der Zustand (in React »State« genannt) ist nur komponentenintern und enthält veränderliche Daten, wie z.B. Benutzereingaben. Das Ändern des Zustands führt in React zum erneuten Rendern der gesamten Komponente sowie aller ihrer Unterkomponenten.
- *Technisch werden Komponenten entweder als Funktion oder als ES6-Klasse geschrieben.* Lange Zeit waren die Klassen der einzige vollwertige Weg, Komponenten zu schreiben. Auch wenn es schon länger möglich war, Komponenten als Funktion zu entwickeln, ist dies erst seit Anfang 2019 (mit der React-Version 16.8) durch die »Hooks-API« eine vollwertige Alternative zu Klassen geworden. Da wir davon ausgehen, dass sich die funktionsbasierte API langfristig durchsetzen wird, werden wir in diesem Buch auch hauptsächlich diese API behandeln. Im Anhang haben wir aber ein Kapitel, das auch die Klassen-API vorstellt. Unabhängig davon, wie du deine Komponenten baust, kannst du innerhalb einer Anwendung

Klassen- und Funktionskomponenten beliebig und transparent mischen. Die Frage, Funktions- oder Klassenkomponente, ist nur für die Implementierung der Komponente relevant, aber nicht für deren Verwendung.

1.2 Warum React?

In diesem Abschnitt möchten wir dir in aller Kürze Gründe nennen, warum man sich aus unserer Sicht mit React beschäftigen sollte und warum wir glauben, dass React eine sehr gute Bibliothek ist.

- Einfachheit:
  React hat eine überschaubare API und lässt sich sehr schnell erlernen. Das Architekturmodell hinter React, in dem der Anwendungszustand zentralisiert gehalten wird und die Daten nur in eine Richtung fließen, macht auch große Anwendungen noch gut verständlich und nachvollziehbar.

- Kontinuität und Stabilität:
  Die Einführung von Änderungen an der React-API erfolgt sehr behutsam. Inkompatible Änderungen werden bereits frühzeitig gut kommuniziert und die Abschaltung von APIs erfolgt erst, wenn es Ersatz dafür gibt. Trotz des relativ konservativen Releasezyklus werden aber beständig neue Features eingebaut. Ein gutes Beispiel dafür ist das derzeit (Stand Ende 2019) aktuelle Major-Release 16 von React, das bereits im September 2017 veröffentlicht wurde[2] und jetzt also schon über zwei Jahre (!) aktiv ist. Diese Stabilität bedeutet allerdings nicht, dass keine neuen Features hinzukommen – im Gegenteil! Innerhalb dieses Major-Release sind diverse grundlegende neue Funktionalitäten hinzugekommen, etwa die Hooks-API, die wir in diesem Buch ganz besonders betrachten werden. Diese Änderungen waren allerdings alle abwärtskompatibel, sodass man sie in seiner bestehenden Anwendung einsetzen konnte, aber nicht musste. Du wirst also nicht alle paar Monate oder gar Wochen zu Updates deiner Anwendung gezwungen.

- Integrierbarkeit:
  React konzentriert sich im Wesentlichen auf die Verwaltung des Zustands deiner Anwendung und insbesondere dessen Darstellung. Dabei macht React nur sehr wenige Annahmen über seine Umgebung, sodass die Bibliothek sich auch sehr gut in bestehenden Projekten und Websites einsetzen lässt und eine sanfte Migration be-

2 *https://reactjs.org/blog/2017/09/26/react-v16.0.html*

reits existierender Anwendungen erlaubt. Außerdem kannst du deinen Code so strukturieren, dass UI-unabhängige Logik von React getrennt implementiert wird und somit auch in anderen Projekten mit anderen Technologien weiterverwendet werden kann.

- Ökosystem:
  Um React herum ist bereits ein sehr lebendiges Ökosystem mit weiteren Tools und Bibliotheken für diverse Einsatzszenarien entstanden. Obwohl React nur ein eine UI-Bibliothek ist, musst du in deinem Projekt also nicht bei »null« anfangen, wenn du z.B. einen Router benötigst. In Teil III dieses Buchs gehen wir auf einige dieser Bibliotheken ein.

- Entwicklertools:
  Es gibt für React sehr gute Tools zur Entwicklung, nach dem Motto: ohne gute Tools keine guten Anwendungen. Dazu zählen zum Beispiel die React Developer Tools für Chrome und Firefox, ein Kommandozeilen-Werkzeug zum Erzeugen neuer React-Projekte (Create React App) sowie sehr gute Unterstützung in allen populären Editoren und IDEs.

- Praxiserprobte Technologie:
  React ist sehr weit verbreitet und kommt auf vielen großen und kleinen Websites und Anwendungen im Internet zum Einsatz (neben Facebook zum Beispiel auch bei Netflix, Jira oder Microsoft Outlook) und auch als Lösung für In-House-Anwendungen im Intranet wird React gerne benutzt. Zur Stabilität von React trägt bei, dass neue Features in der Regel zunächst auf der riesigen facebook.com-Codebasis getestet und erst danach als Beta- und dann als finale Version veröffentlicht werden.

1.3 Beziehung zu anderen Technologien

React hilft dir, Anwendungen für das Web zu bauen. Dabei ist React selten ganz allein im Einsatz. Insbesondere brauchst du für eine komplette Anwendung zumindest noch:

- einen Compiler, der JSX-, modernen JavaScript- und/oder TypeScript-Code nach ES5 übersetzt,
- ein Modulsystem zur Strukturierung deiner Anwendung sowie
- ein Build-System.

In größeren Anwendungen kommen meistens noch hinzu:

- eine architektonische Idee davon, wie eine Anwendung aufgebaut ist,
- Routing (Mappen von URLs auf Komponenten für die gewohnte Navigation im Browser) und
- ein Type-Checker (z.B. TypeScript).

Die dazu passenden Techniken wollen wir nun kurz beleuchten.

1.3.1 ES.Next-JavaScript und Babel

Wie du gesehen hast, brauchen wir zwingend einen Compiler, wenn wir JSX nutzen wollen, denn kein Browser unterstützt den JSX-Code nativ und wird es vermutlich auch nie tun. Zum Übersetzen von JSX ist Babel[3] das empfohlene Werkzeug. Babel ist in der Lage, neben JSX die jeweils aktuellen und vielfach auch die kommenden JavaScript-Sprachfeatures, als ES.Next bezeichnet, nach ES5 zurückzuübersetzen, bis diese von allen Browsern unterstützt werden, für die du deine Anwendung zur Verfügung stellen willst.

ES5, ES6, ES.next, ... Historie der JavaScript-Sprachversionen

Wir beziehen uns in diesem Buch auf unterschiedliche Versionen von JavaScript. Unterschiedliche Begriffe können dabei leicht zu Verwirrung führen.

Zuerst zum Unterschied zwischen ECMAScript und JavaScript: ECMAScript ist die Spezifikation der Sprache, die von der Organisation Ecma International veröffentlicht wird. JavaScript ist die Implementierung dieser Spezifikation. In der Praxis spielt dieser Unterschied kaum eine Rolle, und so werden – wenn es um die Benennung einer Version geht – »JavaScript« und »ECMAScript« häufig synonym verwendet.

Von der Spezifikation der Sprache gibt es unterschiedliche Versionen. Bis Juni 2015 war ECMAScript 5 (kurz: ES5) die aktuellste Version, die zu dem Zeitpunkt auch von praktisch allen Browsern unterstützt wurde. Das ist auch der Grund, warum diese Version häufig noch als Zielversion beim Kompilieren verwendet wird (der Internet Explorer 11 zum Beispiel unterstützt weiterhin fast keine neueren JavaScript-Features).

→

3 *https://babeljs.io/*

Im Juni 2015 wurde dann der Nachfolger von ES5 veröffentlicht, der häufig als ES6 bezeichnet wird, offiziell aber ECMAScript 2015 heißt. Die Begriffe ECMAScript 6, ES6 und ECMAScript 2015 beziehen sich auf dieselbe Sprachversion. Mit dieser Version gab es sehr viele neue Sprachfeatures (wie Modulsystem, Klassen, Promises, Block Scoping mit let und const, Arrow-Funktionen etc.), sodass man dieses JavaScript auch als »modernes JavaScript« bezeichnet. Die Lern- und Adaptionskurve für dieses Release war und ist dementsprechend hoch.

Aus diesem Grund wird seit 2015 regelmäßig einmal pro Jahr eine neue Sprachversion veröffentlicht (ES5 ist bereits 2009 erschienen), die dafür weniger Features enthält. Diese Versionen werden nach dem Jahr ihres Erscheinens benannt, häufig aber auch noch als ES7, ES8 etc. bezeichnet.

Dazu kommt noch der Begriff ES.Next. Dieser bezeichnet die jeweils kommende, noch unveröffentlichte Version (in Babel kannst du beispielsweise einstellen, dass du ES.Next-Features in deinem Code verwenden und kompilieren möchtest).

Die von uns verwendeten neuen Konzepte ab ES6 haben wir zusammengefasst im Anhang C erläutert. Im React-Umfeld ist es gängig, neue und teilweise auch unveröffentlichte Sprachfeatures recht schnell einzusetzen, was dank Babel auch kein großes Problem ist.

1.3.2 Webpack und Browserify

Modul-Loader

Wir strukturieren unsere Anwendung mit ES6-Modulen. Diese und ES6-Imports brauchen aber einen Modul-Loader (auch Bundler genannt), der Abhängigkeiten auflöst, da der Browsersupport für ES6-Imports noch nicht ausreichend ist. Mit Webpack[4] und Browserify[5] stehen uns hier zwei weitverbreitete Werkzeuge zur Verfügung. Beide lösen ES6-Importe auf und erzeugen je nach Konfiguration eine oder mehrere Ausgabedateien mit allen benötigten Abhängigkeiten. Webpack ist weit verbreitet und kommt auch bei Projekten zum Einsatz, die mit dem offiziellen React-Tool Create React App angelegt wurden.

Code-Splitting

Webpack unterstützt zudem Code-Splitting, das bedeutet, es kann deinen Code im Build in mehrere kleinere Ausgabedateien zerteilen, sodass der Browser initial nicht deine ganze Anwendung laden und interpretieren muss, sondern nur das, was zu einem Zeitpunkt wirklich benötigt wird. Code-Splitting wird auch von React unterstützt; mehr dazu findest du in Kapitel 5.6.

4 *http://webpack.github.io/*
5 *http://browserify.org/*

1.3.3 TypeScript und Flow

Typensystem für JavaScript

Mit TypeScript[6] hat Microsoft eine getypte Erweiterung von JavaScript geschaffen. Du kannst damit für jeden Parameter, jedes Property und jede Variable einen Typ angeben, außerdem kannst du die Struktur von Objekten mit Interfaces beschreiben. Solche Typannotationen haben viele Vorteile. Neben der besseren Lesbarkeit kann dich auch die IDE besser unterstützen. Durch Typannotationen wird auch in der JavaScript-Welt verlässliches Refactoring, Codeanalyse und Code-Completion möglich.

Gerade für größere Projekte, die über einen längeren Zeitraum laufen, halten wir einen Type-Checker für unumgänglich.

TypeScript-Compiler

Microsoft liefert auch gleich einen Compiler mit, der diese Typannotationen checkt und in ES5-Code (oder wahlweise sogar ES3) zurückübersetzt. Der übersetzte Code bleibt dabei lesbar. Da dieser Compiler ebenfalls JSX unterstützt, kannst du den TypeScript-Compiler auch als Alternative zu Babel verwenden.

Man kann diesen Stil der Typannotationen aber auch ohne den TypeScript-Compiler verwenden. Babel ist in der Lage, diese Annotationen herauszufiltern oder per Plug-in auch in der Ausgabe als Kommentar beizubehalten. Als Checker dient dann z.B. die IDE (WebStorm[7] und Visual Studio Code[8] haben einen sehr guten Support dafür) oder andere Kommandozeilen-Werkzeuge. In Kapitel 11 zeigen wir, wie du TypeScript in deiner React-Anwendung einsetzen kannst.

Ein weiteres Beispiel eines solchen Werkzeugs ist Flow[9], das ebenfalls von Facebook entwickelt wird. Flow übersetzt das annotierte JavaScript nicht – das macht ja wie erwähnt bereits Babel –, sondern führt lediglich eine Überprüfung der Typen durch.

6 *http://www.typescriptlang.org/*
7 *https://www.jetbrains.com/webstorm/*
8 *https://code.visualstudio.com/*
9 *http://flowtype.org/*

Sourcecode formatieren mit Prettier

Prettier[10] ist ein Tool zum Formatieren von JavaScript-, TypeScript- und anderem Sourcecode, das in der React-Community eine hohe Verbreitung erfahren hat. Es gibt dafür Integrationen in zahlreiche IDEs und andere Tools. Die Idee hinter Prettier ist, dass unabhängig davon, wer mit welchem Tool den Sourcecode editiert, dieser immer gleich formatiert ist und somit zum Beispiel »überflüssige« Diffs durch unterschiedliche Formatierungen in der Versionsverwaltung vermieden werden.

Auch wenn Prettier nicht dazu gedacht ist, Fehler im Code zu finden, gibt es doch manchmal einen Hinweis darauf, dass sich Syntaxfehler eingeschlichen haben. Wenn der Code beim Speichern nicht automatisch formatiert wird, ist die Wahrscheinlichkeit hoch, dass sich irgendwo ein Syntaxfehler (fehlende Klammer o.Ä.) eingeschlichen hat.

1.3.4 Router

URLs auf Komponenten abbilden

Auch in Single-Page-Anwendungen sollte das Navigieren über die URL des Browsers sowie des Back-Buttons funktionieren. Das Konzept dazu heißt Router: Dieser setzt einerseits die URL in der Navigationszeile des Browsers, sobald ein anderer Anwendungsteils aufgerufen wird. Andererseits kann der Router die zur URL passenden Komponenten ermitteln und rendern lassen.

Der React Router[11] ist nur eine von vielen Router-Implementierungen, die mit React benutzt werden können. Allerdings hat er sich als De-facto-Standard etabliert. Daher werden wir dem React Router auch das komplette Kapitel 9 in unserem Buch widmen.

1.3.5 Flux

Architekturmodell für React-Anwendungen

Flux[12] ist ein von Facebook entwickeltes Architekturmodell, das beschreibt, wie mit React große Anwendungen gebaut werden können. Es definiert dazu ein Anwendungsmodell, bei dem Daten immer nur in eine Richtung durch die Anwendung fließen und dabei einen Kreislauf beschreiben. Das Modell ist weder auf React beschränkt noch ist es Bestandteil von React. Es gibt eine ganze Reihe von Frameworks, die die Flux-Architekturidee ausimplementieren. Die prominenteste Implementierung, die zumindest stark von dieser Idee inspiriert ist, ist Redux (siehe folgenden Abschnitt).

10 *https://prettier.io/*
11 *https://github.com/ReactTraining/react-router/*
12 *https://facebook.github.io/flux/*

1.3.6 Redux

Redux[13] ist ein Werkzeug zum »externen Statemanagement«, mit dem du den Zustand deiner Anwendung verwalten kannst, auch wenn diese sehr groß wird. Es ist inspiriert von der Flux-Architektur und bringt eine Reihe von Architekturvorgaben mit. So wird der Zustand der Anwendung an einer einzigen zentralen Stelle gehalten und die Verarbeitung (Logik) geschieht durch pure Funktionen – sogenannte Reducer. Redux stellen wir dir in Kapitel 10 vor.

1.3.7 Create React App

Das Aufsetzen eines neuen React-Projekts ist aufwendig, da eine ganze Reihe von Tools, insbesondere für den Build, notwendig ist. Das Konfigurieren und Pflegen des Tool-Stacks mit allen Abhängigkeiten kann einen nicht unerheblichen Aufwand auch während der Projektlaufzeit darstellen.

Um dieses Problem zu umgehen, gibt es das Kommandozeilentool Create React App[14], mit dem du ein neues Projekt anlegen kannst. In diesem Projekt sind alle wichtigen Abhängigkeiten und das Tooling bereits eingetragen und vorkonfiguriert, sodass du sofort mit der Entwicklung beginnen kannst. Die Konfigurationen der verwendeten Tools (zum Beispiel für den Produktionsbuild), die dann für dein Projekt eingesetzt werden, entsprechen den React-Best-Practices und werden ständig erweitert und verbessert. Du kannst die Versionen der Konfiguration jederzeit in deinem Projekt aktualisieren, um die neuesten Features zu bekommen.

Das Tool Create React App stellen wir dir in Abschnitt 2.2 vor.

1.3.8 React Developer Tools

Entwickertools für Chrome und Firefox

Für die Entwicklung von React-Anwendungen stehen die React Developer Tools zur Verfügung[15]. Dabei handelt es sich um eine Erweiterung für die Entwicklertools des Chrome- und Firefox-Browsers.

Die React Developer Tools zeigen in einem eigenen Tab die React-Komponenten einer Webseite sehr übersichtlich mitsamt ihren Properties und des aktuellen Zustands an. Die aktuellen Werte lassen sich darüber ebenfalls verändern, sodass du schnell ausprobieren kannst, wie sich deine Komponente mit anderen Properties verhalten würde.

---

13 *https://github.com/reactjs/redux/*
14 *https://create-react-app.dev*
15 *https://github.com/facebook/react-devtools/*

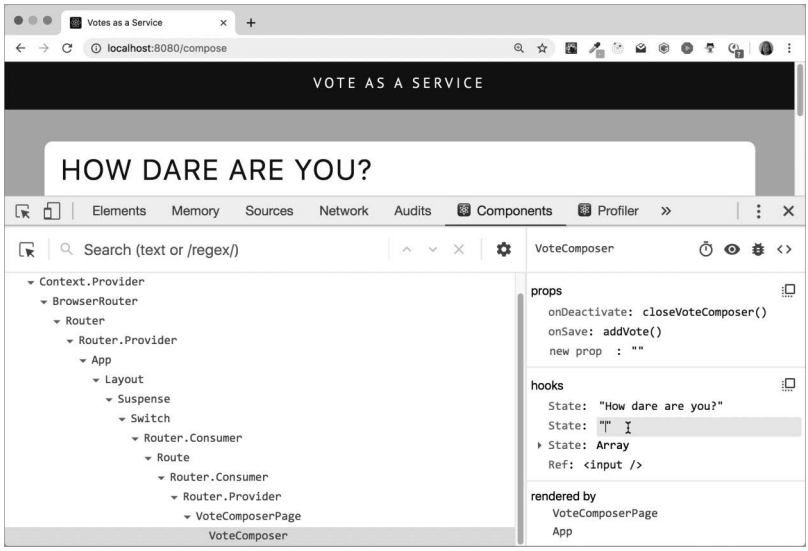

Abb. 1–1
Die React Developer Tools

1.4 Vergleich mit anderen Webtechnologien

Neben React gibt es eine ganze Reihe weiterer Webtechnologien. Manche dieser Techniken sind ergänzend und andere wiederum für denselben Einsatzzweck gedacht und stellen damit eine Konkurrenz dar. Ein Vergleich kann für eine Einordnung spannend sein und dir beim Einstieg helfen, wenn du bereits Erfahrungen mit einer der folgenden Techniken hast.

1.4.1 Web Components

Web Components

Web Components sind der Standard für eine ganze Reihe von Techniken sein[16]. Jede dieser Techniken kann man in Kombination mit den anderen oder für sich allein verwenden.

Custom Elements[17] erlauben die Definition von eigenen HTML-Tags inklusive lokalem CSS und JavaScript. Du kannst also sowohl Aussehen als auch Verhalten für ein solches neues HTML-Tag definieren.

Shadow DOM

Über das sogenannte *Shadow DOM*[18], in dem sich die Komponente darstellt, wird die Lokalität der Komponente realisiert.

16 *https://developer.mozilla.org/en-US/docs/Web/Web_Components/*
17 *https://developer.mozilla.org/en-US/docs/Web/Web_Components/Custom_Elements/*
18 *https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM/*

HTML Templates[19] lassen dich HTML in die Seite einbinden, ohne dass es unmittelbar auf der Seite dargestellt wird. Dies kann nützlich sein, um das Template später durch ein Skript mit Werten zu versehen und im Browser-DOM darzustellen. Obwohl diese Erweiterung als Teil von Web Components begonnen hat, wird sie nun als Teil der HTML-Spezifikation weiterentwickelt.

Die Programmierung von Web Components erfolgt imperativ und mit Listenern. Die Bibliothek Lit Elements[20] stellt für Web Components ein deklaratives Programmiermodell zur Verfügung, das vom Konzept an React erinnert.

Sind Web Components komplementär oder Konkurrenz zu React?

Verwendung von Web Components in React-Komponenten

Web Components können wie jedes andere HTML-Element in einer React-Komponente genutzt werden. Diese Möglichkeit war eine bewusste Entscheidung der React-Entwickler. Hier sperrt man sich also nicht gegen Web Components, auch wenn es bei der Integration einige technische Schwierigkeiten gibt[21].

Andere wiederum sehen in React eher eine Konkurrenz zu Web Components, sind aber mit React nicht zufrieden, weil es kein Standard ist. Laut der Aussage eines React-Kern-Entwicklers wird sich React nicht an Web Components annähern, allerdings auch nicht den Gebrauch zusammen mit React ausschließen, sondern ihn so weit wie möglich unterstützen[22].

1.4.2 Angular

Der Hype »vor React«

Angular kommt in zwei unterschiedlichen Versionen daher, die mehr oder weniger unterschiedlich und inkompatibel sind. Angular 1[23] (»AngularJS«) gibt es schon sehr lange und man kann es ohne Wenn und Aber den »Hype vor React« nennen. Angular in der Version 2 hat mit einer ganzen Reihe von Konzepten aus Angular 1 gebrochen. Die Versionen ab Angular 2 werden deswegen auch »Angular« genannt (im Gegensatz zu »AngularJS«, was sich auf Version 1 bezieht).

19 *https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template/*
20 *https://lit-element.polymer-project.org/*
21 *https://custom-elements-everywhere.com/libraries/react/results/results.html*
22 *https://github.com/facebook/react/issues/5052#issuecomment-145594782/*
23 *https://angularjs.org/*

Sowohl AngularJS als auch Angular sind jeweils ein komplettes Framework, inklusive Router und klaren Vorstellungen von einer Architektur und der Art und Weise, wie getestet wird. So gibt es neben Komponenten zum Beispiel ein Modulkonzept, Services und Dependency Injection. Angular ist mit TypeScript entwickelt und auch Angular-Anwendungen werden darin geschrieben.

1.4.3 Vue

Vue[24] ist ebenfalls ein komponentenbasiertes JavaScript-Framework zur Entwicklung von Single-Page-Anwendungen. Im Gegensatz zu React kommen hier viele Bausteine mit, etwa ein Router und eine offiziell unterstützte Redux-Version für das Zustandsmanagement.

Vue verwendet genau wie Angular eine Template-Sprache, hat aber eine schlankere API und macht weniger Vorgaben, wie die Anwendung strukturiert sein soll. Das Framework wurde 2014 veröffentlicht, wird von einer Open-Source-Community entwickelt und hat sehr schnell eine große Verbreitung gefunden. Die Version 3 von Vue wird genau wie Angular in TypeScript entwickelt.

1.4.4 React Native

Native Anwendungen mit React Native

React Native bietet eine Möglichkeit, mit React native Komponenten nicht nur für das Web, sondern auch für andere Systeme (unter anderem Android und iOS) zu entwickeln. Dabei wird eine andere Version des virtuellen DOM verwendet, die eine React-Komponente nicht für den DOM aufbereitet, sondern auf native Komponenten des Betriebssystems abbildet. Als JavaScript-Engine kommt sowohl bei Android als auch bei iOS Webkits JavaScript-Engine JavaScriptCore[25] zum Einsatz.

1.5 Wie man dieses Buch benutzt

Die in diesem Buch behandelten Themen sind komplex und umfangreich, zudem befinden sie sich in vielen Teilen im Fluss. Alle Themen auf dem neuesten Stand in einem Buch zu behandeln, ist daher nicht möglich. Stattdessen nehmen wir dich als Leser an die Hand und führen dich an die zentralen Themen der React-Welt heran. Für einige Details und weiterführende Informationen verweisen wir mithilfe von Links auf die aktuellsten Quellen im Internet.

---

24 *https://vuejs.org/*
25 *http://trac.webkit.org/wiki/JavaScriptCore/*

Die Teile des Buchs

Das Buch gliedert sich in drei Teile.

- Im ersten Teil geben wir dir einen kurzen Überblick darüber, was React überhaupt ist. Dazu führen wir im folgenden Kapitel zunächst ein sehr kleines Beispiel ein, damit du einmal kompakt sehen kannst, wie React sich »anfühlt«. Für den weiteren Verlauf des Buchs entwickeln wir dann eine größere Anwendung, die wir dir ebenfalls in diesem Teil vorstellen.
- Im zweiten Teil leiten wir die grundlegenden Konzepte von React her. Wir sehen uns an, wie Komponenten entwickelt werden, welche Eigenschaften und Funktionen Komponenten haben und wie diese zusammenspielen. Außerdem werfen wir einen Blick darauf, welche Besonderheiten es gibt, wenn mit React Benutzereingaben, z.B. in Formularen, verarbeitet werden sollen und wie du Daten über eine REST API lesen und schreiben kannst.
- Im dritten Teil blicken wir über den Tellerrand hinaus und sehen uns das Ökosystem von React an, also Technologien und Bibliotheken, die häufig in React-Projekten zum Einsatz kommen, aber nicht Bestandteil von React selbst sind. Neben der Frage, wie du React-Anwendungen testen kannst, gehören dazu die Bibliotheken Redux, React Router, GraphQL und die Sprache TypeScript.
- Am Schluss folgt noch der Anhang. Hier geben wir dir einen Überblick, wie du deine React-Anwendung serverseitig rendern kannst. Außerdem werfen wir einen Blick auf die Klassen API von React und geben dir eine Einführung in TypeScript, in neuere JavaScript-Features, die für React wichtig sind, sowie in die Abfragesprache GraphQL.

Hands-on-Abschnitte

Jedes Kapitel in Teil II und III beginnt mit einem Hands-on-Abschnitt. In diesem werden anhand eines praktischen Beispiels die wichtigsten Konzepte des Kapitels erläutert. In den weiteren Teilen des Kapitels wird dann je nach Schwerpunkt des Kapitels die Anwendung weiterentwickelt und/oder auf weitere Details eingegangen.

Auch wenn du die in den Hands-on-Teilen beschriebenen Schritte nicht selber nachprogrammierst, kannst du damit einen praktischen Überblick über das vorgestellte Thema bekommen. Daher haben wir in den Teilen auch relativ viel Code abgedruckt, damit du zum Nachvollziehen nicht vor dem Computer sitzen musst.

Die Kapitel des Buchs bauen aufeinander auf. Ab Kapitel 4 verwenden wir durchgehend unsere Beispielanwendung, um die Konzepte von React zu demonstrieren. Insbesondere im dritten Teil sind die vorgestellten Technologien und Bibliotheken teilweise so komplex, dass wir sie am Anfang des Kapitels im Hands-on-Teil an einem kleinen, separaten Beispiel zeigen. Im Laufe des Kapitels beschreiben wir dann in Ausschnitten, wie sich die jeweilige Technologie auf unsere Anwendung auswirkt.

1.6 Voraussetzungen für dieses Buch

ES5-Kenntnisse notwendig

Wir verwenden in diesem Buch konsequent die JavaScript-Versionen ECMAScript 2015 und neuer (ES.Next). Es genügt aber, wenn du mit der vorangegangenen Version ES5 vertraut bist. Die wichtigsten von uns verwendeten Neuerungen der Sprache haben wir im Anhang zusammengestellt, sodass du dort bei Bedarf nachschlagen kannst.

HTML und DOM

Neben ES5 solltest du außerdem HTML und das Document Object Model (DOM) kennen.

npm

Wenn du die Beispiele ausprobieren möchtest, muss auf deinem Rechner der Node Package Manager npm installiert sein. Mit npm werden wir die Module (Frameworks und Bibliotheken) für unsere Anwendung installieren und die Anwendung auch starten. Achtung: Die Beispiele sind mit npm in der Version 6.9.x getestet!

Node.js

Für den serverseitigen Beispielcode verwenden wir Node.js. Wenn du diese Teile ausprobieren möchtest, brauchst du eine Node.js-Installation auf deinem Rechner. (Wir haben mit Version 10.16.x getestet.)

1.7 Änderungen gegenüber der ersten Auflage

Die zweite Auflage wurde komplett überarbeitet und stellt jetzt die Hooks-API in den Vordergrund. Wir zeigen die wichtigsten React Hooks sowie die Hooks-API von React Router, Redux und GraphQL.

Neu hinzugekommen sind außerdem Beschreibungen diverser React-APIs, wie Suspense, Caching oder Code-Splitting, Tests mit der React Testing Library und das Tool Create React App. Außerdem betrachten wir die Entwicklung von React-Anwendungen mit TypeScript. Als Grundlage dafür gibt es im Anhang eine Einführung in die Sprache TypeScript.

1.8 Website zum Buch

Wir haben zu diesem Buch eine Website eingerichtet, über die du unter anderem zu unserem GitHub-Repository mit dem Beispielcode findest. Auf der Seite werden wir aber auch Korrekturen und Ergänzungen zum Buch und zu den Beispielen veröffentlichen, insbesondere auch zu Neuerungen in React. Die Adresse der Seite lautet:

https://reactbuch.de

Anregungen, Kommentare und Fragen nehmen wir jederzeit gerne entgegen. Du kannst uns erreichen unter der E-Mail-Adresse:

reactbuch@nilshartmann.net

oder über Twitter:

Nils Hartmann *@nilshartmann*

Oliver Zeigermann *@DJCordhose*

Alternativ kannst du uns auch im GitHub-Repository der Beispielanwendung gerne einen Issue einstellen (*https://github.com/reactbuch/vote-example-v2/issues*).

1.9 Danksagungen

Bei der Erstellung dieses Buchs haben uns eine ganze Reihe von Menschen geholfen, denen wir ganz herzlich danken wollen!

Unser Dank gilt dabei auch denjenigen, die das Buch schon vor der Fertigstellung gelesen und kommentiert haben und die auch für diese zweite Auflage sehr gutes Feedback gegeben haben.

Darüber hinaus möchten wir uns ganz herzlich beim Team vom dpunkt.verlag und allen an der Produktion Beteiligten bedanken, die uns nach Kräften geholfen und mit Rat und Tat zur Seite gestanden haben.