1 Einleitung

1.1 Was ist React?

React[1] ist eine von Facebook entwickelte JavaScript-Bibliothek, mit der du Benutzeroberflächen für das Web entwickeln kannst. Facebook stellte das Framework 2013 als Open-Source-Lösung zur Verfügung und verwendet das Framework auch selbst als Basis für eine ganze Reihe von seinen Webseiten. Zum Beispiel ist Instagram komplett auf Basis von React entwickelt. Auch auf anderen großen Webseiten, wie zum Beispiel Netflix, Airbnb oder Wall Street Journal, ist React im Einsatz.

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.

Dabei ist der Kern von React erst einmal losgelöst vom Web: React kann in unterschiedlichen Szenarien funktionieren, auch in nativen Anwendungen. Allerdings ist neben dem reinen Konzept ebenso die Programmiersprache JavaScript festes Element. React ist in JavaScript geschrieben und auch deine Programme müssen entweder in JavaScript geschrieben oder nach JavaScript übersetzt werden.

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. Entweder renderst du deine Anwendung auf dem Server – das erfordert natürlich eine JavaScript-Engine auf der Serverseite – oder du tust dies im Browser. React unterstützt beides, sogar beide Möglichkeiten zusammen: Du renderst die Anwendung auf dem Server, überträgst das fertige HTML an den Browser und der bringt das HTML in den

1. *http://reactjs.com*

DOM. Von da an werden alle Updates im Browser gerendert. Das ist sehr praktisch zum Beispiel für eine schnelle erste Anzeige der Anwendung.

React 15

Mit dem React Release 15 (Stand Drucklegung April 2016) wechselt React auf ein neues Versionierungsschema. Während die vorherigen Versionen 0.13 und 0.14 hießen, passt Facebook sich in React 15 bestehenden Realitäten an: Sprünge von 0.13 auf 0.14 waren tatsächlich eher Major-Releases und trotz Major-Nummer 0 ist React seit 2013 stabil und produktiv nutzbar[a].

Als vielleicht wichtigstes neues Feature von React 15[b] werden nun alle SVG-Tags und -Attribute unterstützt. Die data-reactid Attribute verschwinden ebenso wie span-Elemente um als Expression expandierte Textblöcke. Als Nebenprodukt wird React 15 dabei um ca. 10 % schneller als 0.14. Dazu gibt es ein paar Bugfixes, weitere Warnungen und einige neue Deprecations. Die Trennung von React auf ReactDOM ist mit React 15 abgeschlossen und Funktionen, die dazu in 0.14 deprecated wurden, sind nun ganz verschwunden.

a. *http://facebook.github.io/react/blog/2016/02/19/new-versioning-scheme.html*
b. *http://facebook.github.io/react/blog/2016/03/07/react-v15-rc1.html*

Komponenten

Keine Templates

Das zentrale Konzept bei React ist eine Komponente. Eine React-Komponente enthält alles Notwendige, um sich darzustellen. Dabei trennt React nicht zwischen dem Template und der Darstellungslogik, beides ist in der Komponente. Genauer gesagt: Es gibt kein Template, es gibt nur eine einfache Funktion, die für die Darstellung zuständig ist. Dies hier könnte die einfachste mögliche React-Komponente als reine Funktion sein:

Komponente als Funktion

```
function HelloMessage() {
  return React.createElement
    ('h1', null, 'Hello, World');
}
```

Komponente als ES6-Klasse

Alternativ dazu kann eine Komponente auch als ES6-Klasse geschrieben werden. Zu den Unterschieden kommen wir im Laufe des Buchs. An dieser Stelle zeigen wir nur das Pendant unserer Komponente als ES6-Klasse[2]. Wichtig dabei ist, dass die Methode, die die Darstellung zurückliefert, `render` heißt:

2. ES6-Klassen beschreiben wir in Anhang B.

```
class HelloMessage extends React.Component {
   Render() {
    return React.createElement
      ('h1', null, 'Hello, World');
  }
}
```

Damit definieren wir die Komponente `HelloMessage`, die bei ihrem Aufruf ein `h1`-Element mit dem Inhalt `Hello World` rendert. Dafür benutzen wir `createElement`, das ein HTML-Element erzeugt. Der Tag-Name des Elements lautet gemäß des ersten Parameters `h1`. Wir haben keine weiteren Properties für `h1` und drücken das mit dem zweiten Parameter `null` aus. Alle weiteren Parameter sind die Kinder-Elemente, in diesem Fall also einfach nur ein Text.

JSX

Wir haben also eine Komponente erzeugt, die sich mit `<h1>Hello, World</h1>` ausgibt. Obwohl es in React keine explizite Template-Sprache gibt, kannst du die Ausgabe doch viel kürzer schreiben:

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

Du kannst dein JavaScript mit HTML-artigen Code-Schnipseln anreichern und ihn damit deutlich lesbarer und kürzer machen. Dieser Code tut nämlich genau dasselbe wie das Beispiel oben.

JavaScript-Erweiterung
JSX

Die Erweiterung von JavaScript, mit der du HTML-artigen Code schreiben kannst, heißt JSX. Wenn du diese Erweiterung nutzen willst, brauchst du zwingend einen Übersetzer, der JSX-Ausdrücke in reguläres JavaScript übersetzt. Für das JSX in diesem Beispiel kommt dabei fast wörtlich der Code weiter oben heraus. Wir kommen später auf diesen Übersetzer zurück; hier ist nur wichtig, dass es ihn gibt und seine Nutzung kein Problem darstellt.

Darstellung im Browser

Es ist sehr einfach, eine solche Komponente in eine HTML-Seite einzubetten:

```
const mountNode = document.getElementById('example');
const element = <HelloMessage />;
ReactDOM.render(element, mountNode);
```

Neu ist hier `ReactDOM.render`, das wir zur Darstellung einer Komponente nutzen. Genauer gesagt, erzeugen wir hier über JSX eine Instanz der `HelloMessage`-Komponente von oben. Die Methode `document.get`

`ElementById` ist keine Funktion aus React, sondern Teil der DOM-API des Browsers. Mit ihr suchen wir uns einen DOM-Knoten heraus, in dem wir unser Element darstellen wollen. Der Inhalt des Knotens wird dabei komplett ersetzt.

Komponentenbäume

Eine größere Anwendung in React baust du, indem du Komponenten zusammensteckst (komponierst). Dabei kannst du einer Komponente auch Properties mitgeben, die zum Beispiel Informationen darüber enthalten, was sie ausgeben soll oder wie sie dies tut. Wir wollen nun unsere Komponente so umbauen, dass sie eine andere Komponente nutzt und dieser Properties mitgibt. Als Erstes zeigen wir dir den neuen Code unserer `HelloMessage`-Komponente:

```
function HelloMessage()
  return <h1><Message name='World' /></h1>;
}
```

Properties

Anstatt den Text direkt auszugeben, delegieren wir diese Aufgabe nun an eine zweite Komponente mit dem Namen `Message`. Diese Komponente bekommt ein Property mit dem Namen `name` und dem Wert `'World'`. Wie bei XML und XHTML kannst du dir bei JSX das schließende Tag sparen, indem du das öffnende mit `/>` beendest. Schauen wir uns dazu nun die `Message`-Komponente an, um zu sehen, wie wir mit einem solchen Property umgehen können:

```
function Message(props) {
  return <span>{'Hello, ' + props.name}</span>;
}
```

Diese Komponente ist wieder als eine einfache Funktion definiert. Sie wird aufgerufen, wenn eine Instanz der Komponente gerendert, also dargestellt werden soll. Dabei bekommt sie als Parameter ein Objekt übergeben, das wir hier `props` nennen. In diesem `props`-Objekt befinden sich alle übergebenen Properties, in unserem Fall also das `name`-Property. Ebenfalls neu ist hier das Mischen von JSX und reinem JavaScript. Wenn du einen JavaScript-Ausdruck in JSX einbetten willst, musst du ihn mit geschweiften Klammern umschließen. Das heißt, `'Hello, ' + props.name` ist einfaches JavaScript. Das `span`-Element ist notwendig, da jede `render`-Methode genau ein Element zurückgeben muss.

Das soll an dieser Stelle erst einmal reichen. In Kapitel 2 kommen wir noch auf den Zustand einer Komponente zu sprechen. Änderungen am Zustand führen zur erneuten Darstellung eines Komponentenbaums. Für die Aktualisierung des Zustands werden wir dort auch Event-Handler schreiben.

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 ein sehr gutes Framework 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:
  Die Einführung von Änderungen an der React-API erfolgt sehr behutsam. Inkompatible Änderungen werden im Vorwege gut kommuniziert und die Abschaltung von APIs erfolgt erst, wenn es Ersatz dafür gibt.

- Integrierbarkeit:
  Da React ein reines View-Framework ist, kannst du selbst bestimmen, wie und mit welchen Techniken du die anderen Teile deiner Anwendung baust. React macht sehr wenige Annahmen über seine Umgebung, so dass es sich auch sehr gut in bestehenden Projekten einsetzen lässt und eine sanfte Migration erlaubt.

- Ökosystem:
  Um React herum ist bereits ein sehr lebendiges Ökosystem mit weiteren Tools und Frameworks für diverse Einsatzszenarien entstanden. Obwohl React nur ein View-Framework 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 Frameworks 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, aber auch Lösungen für Webpack, die das automatische Aktualisieren ganzer Anwendungen im Browser erlauben, ohne dass dabei der Anwendungszustand verloren geht.

- Praxiserprobte Technologie:
  Obwohl React erst seit Mitte 2013 als Open-Source-Lösung zur Verfügung steht, ist es schon sehr praxiserprobt. Zum einen wurde das Framework zuvor bereits intern von Facebook für die eigenen

Anwendungen verwendet. Zum anderen setzen seit der Veröffentlichung auch diverse namhafte Websites React ein. Dazu zählen zum Beispiel AirBnB und Netflix und seit Ende 2015 auch Wordpress.

1.3 Beziehung zu anderen Technologien

Wie du nun gesehen hast, ist React eine Bibliothek, die dabei hilft, User-Interfaces auf Basis von Komponenten zu entwickeln. Dabei ist sie selten ganz allein im Einsatz. Insbesondere brauchst du für eine komplette Anwendung zumindest noch

- einen Übersetzer von JSX und gegebenenfalls ES6 nach ES5,
- ein Modulsystem zur Strukturierung deiner Anwendung,
- ein Build-System.

Für viele größere Anwendungen kommen noch hinzu:

- eine architektonische Idee davon, wie eine Anwendung aufgebaut ist, und
- Routing (Navigation von URLs auf Komponenten).

Passende Techniken wollen wir nun kurz beleuchten.

1.3.1 ES6 (ECMAScript 2015), ES7 und Babel

Wie du gesehen hast, brauchen wir zwingend einen Übersetzer, wenn wir JSX nutzen wollen. Dabei ist Babel[3] das von den React-Entwicklern empfohlene Werkzeug und auch wir nutzen es hier. Babel ist in der Lage, neben JSX auch sehr viel ES6-Sprachfeatures und ebenso einige ES7-Vorschläge nach ES5 zu übersetzen.

ECMAScript 6, ECMAScript 2015, Harmony ...?!

Wir beziehen uns in diesem Buch auf unterschiedliche Versionen von JavaScript. Unterschiedliche Begriffen 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.

3. *https://babeljs.io/*

Von der Spezifikation gibt es unterschiedliche Versionen. Seit Juni 2015 ist ECMAScript 2015 der derzeit aktuellste Standard. Sein Vorgänger war ECMAScript 5 (kurz ES5) und lange Zeit war der offizielle Name von ECMAScript 2015 daher auch konsequenterweise ECMAScript 6 oder kurz ES6. Die Begriffe ECMAScript 6, ES6 und ECMAScript 2015 beziehen sich also auf dieselbe Sprachversion.

Dazu kommt noch der Begriff Harmony. Dieser war der ursprüngliche Arbeitstitel für ECMAScript 2015 und wird fast nicht mehr verwendet. Dort, wo er noch verwendet wird, steht er eher für alle zukünftigen Versionen von ECMAScript, nicht nur ECMAScript 2015.

Die nächste Version von ECMAScript, die sich gerade in der Spezifikationsphase befindet, hat bisher keinen Namen und in Ermangelung dessen wird sie pragmatisch ES7 genannt. Manchmal erhält sie auch die Bezeichnung ECMAScript 2016, um anzudeuten, dass jedes Jahr eine neue Spezifikation fertig werden soll.

Obwohl ECMAScript 2015 der offizielle Name ist, werden wir wegen ihrer Kürze in diesem Buch ab jetzt nur noch die Begriffe ES5, ES6 und ES7 verwenden.

Die von uns verwendeten neuen Konzepte von ES6 (und ES7) haben wir zusammengefasst im Anhang erläutert.

1.3.2 Webpack und Browserify

Modul-Loader

Wir strukturieren unsere Anwendung mit ES6-Modulen. Diese und ES6-Imports brauchen aber einen Modul-Loader, der Abhängigkeiten auflöst. Mit Webpack[4] und Browserify[5] stehen uns hier zwei weit verbreitete Werkzeuge zur Verfügung. Beide lösen ES6-Importe auf und erzeugen eine einzige Ausgabedatei mit allen Abhängigkeiten. Browserify wird im Build-Prozess von React selbst genutzt, während Webpack in vielen neueren Projekten zum Einsatz kommt.

Hot Reloading

Da Webpack auch ein spezielles Hot Reloading für React bietet, geben wir in diesem Buch Webpack den Vorzug. Eine Einführung in die Entwicklung eines Build-Prozesses auf Basis von Webpack findest du im Anhang.

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 ES6 geschaffen. Du kannst damit jeden Parameter, jedes Property und jede Variable mit einem Typ annotieren. Dazu gibt es Interfaces und Funktionstypen. 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.

TypeScript Compiler

Microsoft liefert auch gleich einen Compiler mit, der diese Typannotationen checkt und in ES6, ES5 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.

Ein prominentes 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. Diese Überprüfung ist an manchen Stellen sogar mächtiger als das, was der TypeScript-Compiler tut.

1.3.4 React Router

URLs auf Komponenten abbilden

Wie erwähnt, kommt React ohne einen Router. Ein Router dient dazu, für eine URL eine oder mehrere passende Komponenten anzuzeigen. Dazu wird typischerweise eine Abbildung von einem Pfad in einer URL auf eine oder mehrere Komponenten angegeben.

Der React Router[10] 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 ein komplettes Kapitel in unserem Buch widmen.

---

6. *http://www.typescriptlang.org/*
7. *https://www.jetbrains.com/webstorm/*
8. *https://code.visualstudio.com/*
9. *http://flowtype.org/*
10. *https://github.com/reactjs/react-router*

1.3.5 Flux

Architekturmodell für React-Anwendungen

Flux[11] 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. Dazu mehr im letzten Teil dieses Buchs.

1.3.6 Redux

Redux[12] ist die Flux-Implementierung, die wir für dieses Buch ausgewählt haben. Redux spielt sehr gut mit dem React Router zusammen, unterstützt das Rendering auf Client und Server und erlaubt Hot Reloading von allen Teilen der Anwendung. Als Besonderheit wird der Zustand der Anwendung an einer einzigen zentralen Stelle der Anwendung gehalten und die Verarbeitung geschieht durch pure Funktionen – sogenannte Reducers. Diese Ideen sind unter anderem von Elm inspiriert, auf das wir weiter hinten eingehen werden.

1.3.7 React Developer Tools

Entwickertools für Chrome und Firefox

Für die Entwicklung von React-Anwendungen stellt Facebook die React Developer Tools zur Verfügung[13]. 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, so dass du schnell ausprobieren kannst, wie sich deine Komponente mit anderen Properties verhalten würde.

11. *https://facebook.github.io/flux/docs/overview.html*
12. *https://github.com/reactjs/redux*
13. *https://github.com/facebook/react-devtools*

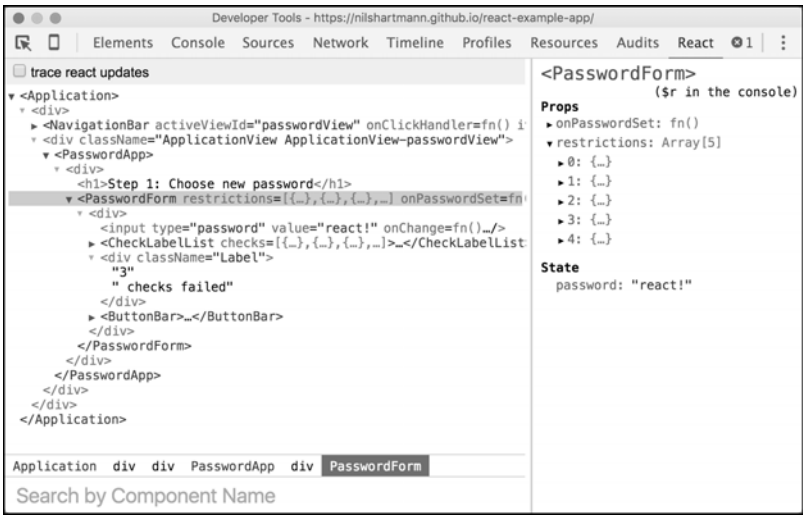

Abb. 1–1
Die React Developer Tools

1.4 Vergleich mit anderen Webtechnologien

Neben React gibt es eine ganze Reihe weiterer aktueller Webtechnologien. Manche dieser Techniken sind ergänzend und andere wieder für denselben Einsatzzweck gedacht und damit Konkurrenz. 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 werden der kommende Standard für eine ganze Reihe von Techniken sein[14]. Jede dieser Techniken kann man in Kombination mit den anderen oder für sich allein verwenden.

Custom Elements[15] 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*[16], in dem sich die Komponente darstellt, wird die Lokalität der Komponente realisiert.

HTML Templates[17] 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 ver-

14. *https://developer.mozilla.org/en-US/docs/Web/Web_Components*
15. *https://developer.mozilla.org/en-US/docs/Web/Web_Components/Custom_Elements*
16. *https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM*
17. *https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template*

sehen und im Browser-DOM darzustellen. Obwohl diese Erweiterung als Teil von Web Components begonnen hat, wird sie nun als Teil der HTML-Spezifikation weiterentwickelt.

HTML Imports

Über *HTML Imports*[18] können Web Components oder allgemein HTML-Seiten in andere importieren werden. Allerdings kündigte Mozilla bereits an, dieses Feature nicht zu implementieren. Ihre Argumentation lautet, dass es bereits einen Import für JavaScript in der Version ECMAScript 2015 gibt, der diese Funktionalität abdeckt. Damit kann man davon ausgehen, dass diese Technik nicht im endgültigen Standard enthalten sein wird.

Sind Web Components komplementär oder Konkurrenz zu React?

Verwendung von Web Components in React-Komponenten

Seit einiger Zeit ist es möglich, Web Components wie jedes andere HTML-Element in einer React-Komponente zu nutzen. Diese Möglichkeit war eine bewusste Entscheidung der React-Entwickler. Hier sperrt man sich also nicht gegen Web Components[19].

Andere wiederum sehen in React eher eine Konkurrenz zu Webkomponenten, sind aber mit React nicht zufrieden, weil es kein Standard ist[20]. 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[21].

1.4.2 Ember

Ember[22] ist im Gegensatz zu React ein komplettes Framework mit vielen Konventionen und impliziten Annahmen. Funktionen und Zusammenhänge ergeben sich häufig aus den Namen der einzelnen Teile der Anwendung und den jeweiligen Ordnern, in denen sie liegen.

React ist selbst nicht unmittelbar von Ember beeinflusst. Allerdings ist der React Router, den wir auch in diesem Buch behandeln werden, durch den Ember-Router inspiriert und beruht auf denselben Ideen.

---

18. *https://developer.mozilla.org/en-US/docs/Web/Web_Components/HTML_Imports*
19. *http://webcomponents.org/presentations/complementarity-of-react-and-web-components-at-reactjs-conf/*
20. *http://christianheilmann.com/2015/07/01/over-the-edge-web-components-are-an-endangered-species/*
21. *https://github.com/facebook/react/issues/5052#issuecomment-145594782*
22. *http://emberjs.com/*

1.4.3 Angular 1 und 2

Der Hype »vor React«

Angular kommt in zwei unterschiedlichen Versionen, die mehr oder weniger unterschiedlich und inkompatibel sind. Angular 1[23] gibt es schon seit einer ganzen Weile und man kann es ohne Wenn und Aber den »Hype vor React« nennen. Angular 2[24] hingegen ist noch sehr neu und derzeit nur als Beta-Version verfügbar. Es bricht mit einer ganzen Reihe von Konzepten aus Angular 1.

Sowohl Angular 1 als auch Angular 2 sind jeweils ein komplettes Framework, inklusive Router und klarer Vorstellungen von einer Architektur und der Art und Weise, wie getestet wird.

Im folgenden Kapitel haben wir einen ausführlicheren Vergleich der beiden Frameworks mit React für dich zusammengestellt.

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.

Da sich React Native noch in den Anfängen befindet, werden wir React Native in dieser Auflage nicht behandeln.

1.4.5 Elm

Der nächste Hype?

Elm[26] ist eine funktionale, reaktive Programmiersprache, die für die Entwicklung von Benutzerschnittstellen im Browser entwickelt wurde. Auch sie nutzt statische Typisierung wie TypeScript. Anders als bei TypeScript weicht ihre Semantik mit Absicht deutlich von JavaScript ab.

Zusammen mit den Paketen für HTML[27] und Virtual-DOM[28] erlaubt Elm eine Entwicklung mit sehr ähnlichen Konzepten, wie wir sie in diesem Buch beschreiben werden. Während Elms Virtual-DOM von React inspiriert ist, steht anders herum Redux unter dem Einfluss

23. *https://angularjs.org/*
24. *https://angular.io/*
25. *http://trac.webkit.org/wiki/JavaScriptCore*
26. *http://elm-lang.org/*
27. *https://github.com/evancz/elm-html*
28. *https://github.com/Matt-Esch/virtual-dom*

der architekturellen Ideen[29] Elms. Elm bietet neben den erwähnten optionalen Typen unveränderliche Datentypen (über Persistent Data Structures[30]), einen Debugger mit Time-Travel-Funktion (wie Redux) und eine Schnittstelle zu nativem JavaScript.

Manche Menschen glauben, dass Elm der nächste Hype im Bereich Frontend-Technologien werden könnte.

1.4.6 Om und Om next

Om[31] ist eine Integration von React mit ClojureScript[32]. ClojureScript erlaubt den Einsatz von Clojure[33] im Browser. Dafür gibt es einen Compiler, der Clojure nach JavaScript übersetzt. Viele Ideen von Om sind so auch im Flux-Framework Redux zu finden, das wir ebenfalls in diesem Buch vorstellen. Insbesondere die Idee von einem zentralen Zustand ähnelt einer Idee aus dem Om-Framework[34], das dafür eine atom-Referenz[35] aus ClojureScript nutzt.

Die nächste Version von Om, genannt Om Next[36], hat eine ganze Reihe von Inspirationen aus der React-Ecke aufgenommen. Hier zeichnet sich also eine ganz neue Richtung von Anwendungsentwicklung im funktionalen Bereich ab.

29. *https://github.com/evancz/elm-architecture-tutorial/*
30. *https://en.wikipedia.org/wiki/Persistent_data_structure*
31. *https://github.com/omcljs/om*
32. *https://github.com/clojure/clojurescript*
33. *http://clojure.org/*
34. *https://github.com/omcljs/om/wiki/Basic-Tutorial#om-basics*
35. *http://clojure.org/atoms*
36. *https://github.com/omcljs/om/wiki#om-next*

1.5 Die Beispielanwendung

Node.js und npm

Für den serverseitigen Code und die Buildtools verwenden wir die Node.js Platform[a], die du auf einem Rechner installiert haben solltest. Wir haben die Beispiele mit der Node.js-Version 4.2.4 getestet.

Die verwendeten Module (sowohl Client-Frameworks wie React, React Router als auch Tools für die Entwicklung (Webpack, Babel)) installieren wir mit dem Node Package Manager[b] (npm). Hier verwenden wir die Version 2.14.x. Mit der runderneuerten Version 3 haben wir die Beispiele nicht getestet!

a. *https://nodejs.org*
b. *https://docs.npmjs.com/*

Node.js und npm

In Zusammenhang mit diesem Buch haben wir eine Beispielanwendung entwickelt, die wir mit dir gemeinsam Schritt für Schritt in jedem Kapitel weiterentwickeln werden. Die Anwendung findest du in unserem GitHub-Repository git clone `https://github.com/reactbuch/vote-example.git`. Wenn du möchtest, kannst du die Anwendung bereits jetzt lokal klonen. Danach kannst du die Anwendung mit den folgenden Schritten zum Laufen bringen:

Installation und Start der Anwendung

1. In das Verzeichnis `app` wechseln
2. Abhängigkeiten installieren: `npm install`
3. Den Server starten: `npm start`
4. Die Anwendung im Browser öffnen: `http://localhost:3000`

Jetzt solltest du unsere Beispielanwendung »Vote as a Service« mit einigen voreingestellten Umfragen sehen. Wir werden auf die Anwendung genauer in Kapitel 3 eingehen. Dort beschreiben wir auch, wie wir die Anwendung im Laufe des Buchs Schritt für Schritt entwickeln und was du wissen musst, um die einzelnen Schritte auszuprobieren.

1.6 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.

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 haben wir im folgenden Kapitel zunächst ein sehr kleines Beispiel entwickelt, 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. Bevor wir schließlich in die Details von React einsteigen, zeigen wir dir am Ende des Kapitels noch die JavaScript-Spracherweiterung JSX.
- 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.
- Danach zeigen wir dir im dritten Teil, wie »echte« bzw. größere React-Anwendungen aussehen können. Das erklären wir anhand des React Routers und des Flux-Frameworks Redux. Außerdem sehen wir uns an, wie du mit den genannten Technologien Universal Webapps entwickeln kannst, also Anwendungen, die auf dem Client und auf dem Server ausgeführt werden können.
- Am Schluss folgt noch der Anhang. Hier zeigen wir dir einen exemplarischen Build-Prozess für React-Anwendungen und gehen auf ES6-Grundlagen ein. Wir setzen für dieses Buch zumindest gute ES5-Kenntnisse voraus. Alle ES6-Features, die wir im Buch verwenden werden, sind im Anhang beschrieben.

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 darin 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. Mehr dazu wieder in Kapitel 3.

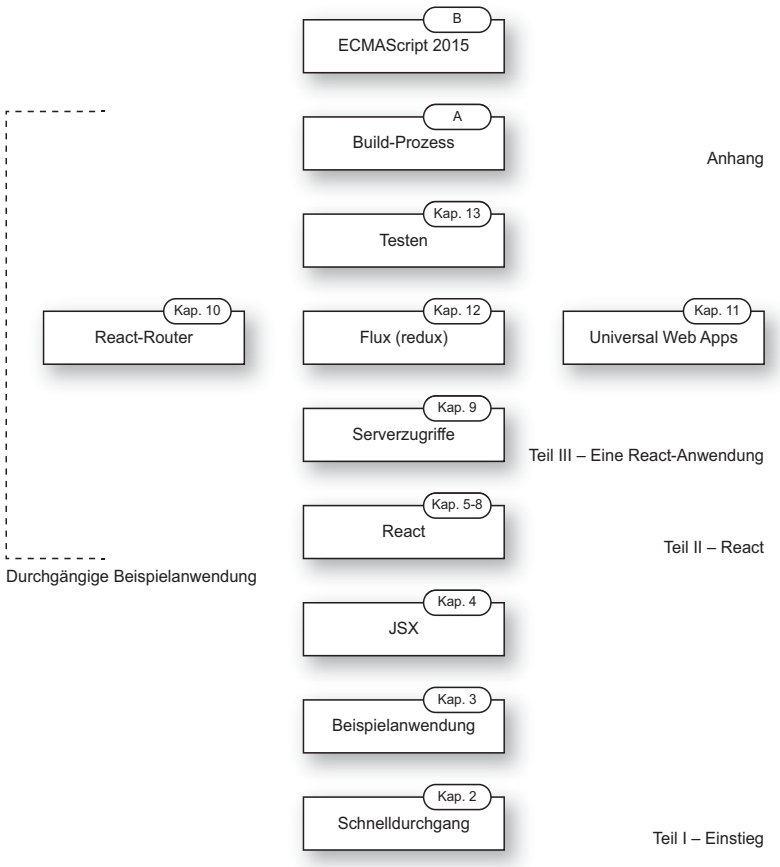

Abb. 1–2
Gliederung des Buchs

Die Kapitel des Buchbauen aufeinander auf. Ab Kapitel 5 verwenden wir durchgehend unsere Beispielanwendung, um die Konzepte von React zu demonstrieren. Insbesondere im dritten Teil sind die vorgestellten Technologien oder Frameworks teilweise so komplex, dass wir sie dir 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 das Framework auf unsere Anwendung auswirkt.

Die Auswahl der in Teil drei von uns eingesetzten Technologien ist subjektiv. Diese Technologien sind nicht Bestandteil von React, sondern bauen darauf auf oder lassen sich gut mit React integrieren. Es gibt aber keinen »Standard« und keine Referenzimplementierungen oder etwas Ähnliches, die besagen, wie genau eine React-Anwendung gebaut wird. Die vorgestellten Frameworks lassen sich auch alle unabhängig voneinander benutzen. Aus diesem Grunde zeigen wir in den Kapiteln jeweils die isolierte Funktion des jeweiligen Frameworks und dann die Integration mit den weiteren vorgestellten Frameworks.

1.7 Voraussetzungen für dieses Buch

ES5-Kenntnisse notwendig

Wir verwenden in diesem Buch konsequent die aktuelle JavaScript-Version ECMAScript 2015. 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, so dass 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 noch mit npm in der Version 2 getestet!

Node.js

Für den serverseitigen Beispielcode in Teil III verwenden wir Node.js. Wenn du diese Teile ausprobieren möchtest, brauchst du eine Node.js-Installation auf deinem Rechner. (Wir haben mit Version 4.2.x getestet.)

1.8 Website zum Buch

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

http://reactbuch.de

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

autoren@reactbuch.de

1.9 Danksagungen

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

Dabei möchten wir insbesondere Dave Brotherstone erwähnen, der uns immer wieder bei zahlreichen Fragen rund um React weitergeholfen hat. Unser Dank gilt auch denjenigen, die das Buch schon vor der Fertigstellung gelesen und kommentiert haben!

Darüber hinaus möchten wir ganz herzlich unserem Lektor René Schönfeldt vom dpunkt.verlag danken, der uns stets mit Rat und Tat zur Seite gestanden hat.

Dem dpunkt.verlag gilt unser Dank dafür, dass er das Risiko eingegangen ist, ein Buch zu verlegen, das Technologien beschreibt, die noch sehr stark im Fluss sind. Danke für euer Vertrauen!