



Fabian Deitelhoff

Vue.js

Von Grundlagen
bis Best Practices



dpunkt.verlag



Foto: Ailona Kardash/TU Dortmund

Dr. Fabian Deitelhoff arbeitet nach seiner Promotion zu »Source Code Comprehension« als Tech-Lead Domestic an Cloud-Themen bei Miele. Darüber hinaus ist er mit brickobotik in der MINT-Bildung und mit Loosely in der Cross-Plattform-Softwareentwicklung tätig. Seine Schwerpunkte sind Low- und No-Code sowie digitale Geschäftsmodelle. Daneben ist er als freier Autor, Dozent und Softwareentwickler im .NET- und Web-Umfeld tätig. Sie erreichen ihn über deitelhoff.me, unter fabian@deitelhoff.me oder auf Twitter als [@FDeitelhoff](https://twitter.com/FDeitelhoff).

Copyright und Urheberrechte:

Die durch die dpunkt.verlag GmbH vertriebenen digitalen Inhalte sind urheberrechtlich geschützt. Der Nutzer verpflichtet sich, die Urheberrechte anzuerkennen und einzuhalten. Es werden keine Urheber-, Nutzungs- und sonstigen Schutzrechte an den Inhalten auf den Nutzer übertragen. Der Nutzer ist nur berechtigt, den abgerufenen Inhalt zu eigenen Zwecken zu nutzen. Er ist nicht berechtigt, den Inhalt im Internet, in Intranets, in Extranets oder sonst wie Dritten zur Verwertung zur Verfügung zu stellen. Eine öffentliche Wiedergabe oder sonstige Weiterveröffentlichung und eine gewerbliche Vervielfältigung der Inhalte wird ausdrücklich ausgeschlossen. Der Nutzer darf Urheberrechtsvermerke, Markenzeichen und andere Rechtsvorbehalte im abgerufenen Inhalt nicht entfernen.

Fabian Deitelhoff

Vue.js

Von Grundlagen bis Best Practices



dpunkt.verlag

Fabian Deitelhoff
fabian@deitelhoff.me

Lektorat: Melanie Andrisek
Lektoratsassistentz: Anja Weimer
Copy-Editing: Claudia Lötschert, www.richtiger-text.de
Satz: Gerhard Alfes, mediaservice, Siegen, www.mediaservice.tv
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, www.exclam.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 978-3-86490-900-9
PDF 978-3-96910-760-7
ePub 978-3-96910-761-4
mobi 978-3-96910-762-1

1. Auflage 2022
Copyright © 2022 dpunkt.verlag GmbH
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.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

*Für meine wundervolle Frau Sandra -
mit deiner Unterstützung ist alles möglich!*

Inhaltsverzeichnis

Vorwort

1 Vue im Überblick

- 1.1 Was ist ein JavaScript-Frontend-Framework?
- 1.2 Historie und das Team rund um Vue
- 1.3 Was ist Vue und was zeichnet es aus?
- 1.4 Warum Vue?
- 1.5 Unterschiede zu anderen Web-Frameworks
- 1.6 Vue 2 oder Vue 3 lernen?
- 1.7 Der Blick in die Zukunft

2 Die Grundlagen: Ein Vue-Schnellstart

- 2.1 Die erste Vue-Anwendung aufsetzen
 - 2.1.1 Die HTML-Struktur
 - 2.1.2 Einbinden von Vue als Skript
 - 2.1.3 Unsere erste Vue-Instanz
 - 2.1.4 Schleifen und Bedingungen
 - 2.1.5 Benutzereingaben verarbeiten
 - 2.1.6 Mit Komponenten kapseln
 - 2.1.7 Das deklarative Rendering und das virtuelle DOM

3 Vue 3 im Überblick

- 3.1 Warum neu schreiben?
- 3.2 Ein mehrschrittiges Release
- 3.3 Interne und externe Anpassungen
- 3.4 Der neue Standard und Framework-Empfehlungen

4 Vue-Projekte aufsetzen

- 4.1 Scaffolding als Basis
- 4.2 Vue-Projekte erstellen
 - 4.2.1 Das Vue CLI
 - 4.2.2 Die Auswahlmöglichkeiten
 - 4.2.3 Ein Projekt erzeugen und den Webserver starten
 - 4.2.4 Rapid (instant) Prototyping
- 4.3 Vite als neuer Standard
- 4.4 Die Entwicklungsumgebung vorbereiten
- 4.5 Server-Side Rendering (SSR)

5 Das (neue) Reactivity-System

- 5.1 Der Ansatz von Vue
- 5.2 Die Implementierung als Proxy in Vue 3
- 5.3 Probleme bei der Reaktivität

6 Komponenten im Detail

- 6.1 Komponentenorientierte Entwicklung
- 6.2 Komponenten und das Separation-of-Concerns-Prinzip

- 6.3 Options API, Class-Style-Syntax und Composition API
 - 6.3.1 Options API
 - 6.3.2 Composition API
- 6.4 Globale und lokale Komponenten
- 6.5 Single-File Components (SFC)
- 6.6 Lifecycle-Hooks
- 6.7 Daten und Props
- 6.8 Non-Prop-Attribute und Refs
- 6.9 Methoden
- 6.10 Benutzerdefinierte Events
- 6.11 Der Datenfluss zwischen Komponenten
- 6.12 Computed Propertys
- 6.13 Watcher
- 6.14 Eigene Komponenten mit v-model
- 6.15 Standard, Named und Scoped Slots
- 6.16 Dynamische Komponenten
- 6.17 Eigene Direktiven und Plug-ins
- 6.18 Fragmente, Portale und Suspense
- 6.19 Asynchrone und funktionale Komponenten
- 6.20 Dependency Injection mit Provide und Inject
- 6.21 Render-Funktionen und JSX
- 6.22 Vue Components und Web Components
- 6.23 Wiederverwendbarkeit und Komponenten-Patterns

7 Data Binding, Formulare und Styles

- 7.1 Einführung ins Data Binding bei Vue
- 7.2 Template-Syntax und das Binding von Attributen
- 7.3 Bedingtes Rendering mit v-if und v-show
- 7.4 Rendern von Listen mit v-for
- 7.5 Die neue Arbeit mit Arrays
- 7.6 Daten filtern, suchen und sortieren
- 7.7 Daten auf Seiten aufteilen (Pagination)
- 7.8 Formulare mit Daten verknüpfen
- 7.9 Events von Anwenderinnen verarbeiten
- 7.10 Formulare validieren
- 7.11 Class- und Style-Binding
- 7.12 Animationen und Transitionen

8 Das Composition API

- 8.1 Der Hintergrund
- 8.2 Setup, Daten und Props
- 8.3 Script Setup
- 8.4 Methoden und Lifecycle-Hooks
- 8.5 Computed Propertys und Watcher
- 8.6 Composables (Composition Function)
- 8.7 Vorgefertigte Kompositionen in Bibliotheken
- 8.8 ref vs. reactive und Reactivity Transform
- 8.9 Provide und Inject
- 8.10 Render Functions und Templates Refs
- 8.11 Eine komplette Komponente mit dem Composition API

- 8.12 Das Reactivity API
- 8.13 Composition API vs. Options API vs. Class API
- 8.14 Das Composition API und Vue 2

9 Routing mit dem Vue Router

- 9.1 Was ist Routing und was der Vue Router?
- 9.2 Installation und Bestandteile des Vue Router
- 9.3 Routen konfigurieren
- 9.4 Routen nutzen
- 9.5 Dynamische Routen mit Parametern und Props
- 9.6 Das Pattern-Matching
- 9.7 Die verschiedenen History-Modi
- 9.8 Verschachtelte Routen, Redirects und Lazy Loading
- 9.9 Route-Guards und Metadaten
- 9.10 CSS-Klassen, Transitionen und Composition API
- 9.11 Fehler bei der Navigation erkennen
- 9.12 Ein DIY-Router für Minimalistinnen und Minimalisten

10 State Management mit Vuex und Pinia

- 10.1 Was ist State Management und das Flux-Pattern
- 10.2 Andere State-Management-Patterns
- 10.3 Das neue Vuex und die Installation
- 10.4 Die drei Kernprinzipien
- 10.5 Die 4 + 1 Kernkonzepte

- 10.5.1 Der Zustand (State)
- 10.5.2 Zugriff auf den Zustand (Getter)
- 10.5.3 Veränderungen des Zustands (Mutations)
- 10.5.4 Aktionen durchführen (Actions)
- 10.5.5 Einen Store aufteilen (Module)
- 10.6 Einen Store in Komponenten nutzen
- 10.7 Vuex und das Composition API
- 10.8 Die Struktur einer Vuex-Anwendung und Plugins
- 10.9 Einen Store debuggen
- 10.10 State Management mit Pinia
- 10.11 Vuex bzw. Pinia nutzen oder nicht?

11 Mit Vue auf Ressourcen zugreifen

- 11.1 Das Beispiel-API
- 11.2 Zugriff auf ein API über das Fetch API
- 11.3 Axios vs. Fetch
- 11.4 Die Bibliothek Axios als Quasi-Standard
- 11.5 Axios in Vue integrieren
- 11.6 Daten abfragen und anzeigen
- 11.7 Daten hinzufügen und löschen mit POST und DELETE
- 11.8 Axios konfigurieren
- 11.9 Pagination mit API-Aufrufen
- 11.10 Mit Fehlern umgehen
- 11.11 API-Zugriffe bündeln
- 11.12 Axios mit Vue Router und Vuex

11.13 Ein kurzer Blick auf Alternativen

12 Anwendungsentwicklung mit Vue

12.1 Ein Blick über den Tellerrand

12.2 Das Tooling als Grundvoraussetzung

12.3 Eine Vue-Projektstruktur

12.4 Vue-Apps debuggen und Fehlerbehandlung

12.5 Web-Apps auf Basis von UI-Frameworks

12.6 Web-Apps auf Basis von Application Frameworks

12.7 Mobile App-Entwicklung und statische Seiten

12.8 Eine Web-App am Beispiel von Quasar

13 Internationalisierung und Barrierefreiheit

13.1 Installation und Konfiguration von Vue I18n

13.2 Das aktive Gebietschema abrufen und verändern

13.3 Übersetzungen nutzen

13.4 Vue I18n und das Composition API

13.5 Asynchrones Laden von Übersetzungsdateien

13.6 Routen und Router-Links übersetzen

13.7 Vuex und Vue I18n

13.8 Anwendungsentwicklung und das Ökosystem

13.9 Etwas zur Barrierefreiheit

14 Testen von Vue-Anwendungen

14.1 Das Testen und Vue

- 14.2 Vorbereitungen im Projekt
- 14.3 Tests organisieren und strukturieren
- 14.4 Unit-Tests für JavaScript und TypeScript
- 14.5 Unit-Tests für Vue-Komponenten
- 14.6 Unit-Tests für das Composition API
- 14.7 Events testen
- 14.8 Tests für Formulareingaben
- 14.9 Tests für Vue Router
- 14.10 Tests für Vuex
- 14.11 Ende-zu-Ende-Tests
- 14.12 Snapshot-Tests
- 14.13 Tests der Internationalisierung

15 Build und Deployment von Vue-Anwendungen

- 15.1 Development- vs. Production-Build
- 15.2 Den Build im Auge behalten
- 15.3 Deployment auf Netlify und Heroku
- 15.4 CI-/CD-Pipelines mit GitHub Actions und Azure DevOps

16 Performante Vue-Apps entwickeln

- 16.1 Vue 3 als allgemeiner Vorteil
- 16.2 Performance von Komponenten prüfen
- 16.3 Events und Reactivity

17 Migration von Vue 2 auf Vue 3

- 17.1 Die Wege einer Migration und der Migration-Build
- 17.2 Breaking Changes
 - 17.2.1 Globales API
 - 17.2.2 Template-Direktiven
 - 17.2.3 Das Key-Attribut
 - 17.2.4 Komponenten
 - 17.2.5 Render-Funktionen
 - 17.2.6 Viele kleinere Änderungen
- 17.3 Vue Router und Vuex

18 Vue und TypeScript

- 18.1 Vue (3) und TypeScript
- 18.2 Ein neues Vue-Projekt mit TypeScript
- 18.3 TypeScript im Vue-Projekt
- 18.4 Die verschiedenen API-Modelle
 - 18.4.1 Options API
 - 18.4.2 Class-Style Vue Components
 - 18.4.3 Composition API
- 18.5 Vue Router, Vuex und Internationalisierung
- 18.6 Vor- und Nachteile bei Vue mit TypeScript
- 18.7 Migration auf TypeScript

19 Vue und das Backend

- 19.1 Express (Node.js, JavaScript und TypeScript)
- 19.2 Django (Python)
- 19.3 Laravel (PHP)
- 19.4 Spring Boot (Java)

- 19.5 ASP.NET Core (.NET, C#, F# und Visual Basic)
- 19.6 Firebase (Low Code)
- 19.7 Serverless als Motto
- 19.8 Low- und No-Code als Wachstumsmotor

20 Etwas zu Bibliotheken

- 20.1 Composition API
- 20.2 Awesome Vue als umfassender Überblick
- 20.3 Eine Bibliothek für Vue 2 und Vue 3 erstellen

Literaturverzeichnis

Index

Vorwort

Webanwendungen werden zunehmend umfangreicher, interaktiver und müssen mit immer größeren Datenmengen umgehen. Das erfordert bei der Konzeption und Implementierung von Webanwendungen ein Umdenken, insbesondere im Bereich der Frontend-Entwicklung. Dort wird verlangt, immer kompliziertere Funktionen hinzuzufügen und immer leistungsfähigere Werkzeuge zu verwenden - und das in immer kürzeren Release-Zyklen.

Web-Frameworks haben sich aus diesem Grund in den letzten Jahren sehr rasant weiterentwickelt. Sie erlauben es Entwicklern, auch komplexe und hochinteraktive Webanwendungen mit vertretbarem Aufwand umzusetzen. Sie bieten vorgefertigte Features an, können auch mit komplexen Zuständen und vielen Daten umgehen. Außerdem bieten sie zahlreiche weitere Features wie Komponenten, Routing und State Management an.

Vue.js hat sich neben Angular und React in kürzester Zeit zu einem populären Web-Framework entwickelt. Beim Schreiben dieses Buchs besitzt das Repository von Vue 2 auf GitHub über 190.000 Sterne und wird über den Paketmanager npm mehr als 2,5 Millionen Mal pro Woche heruntergeladen. Das Repository des Vue-3-Cores kommt bereits auf fast 30.000 Sterne. Mit der Version 3 des Web-Frameworks kamen viele seit Langem schon gewünschte

Funktionen, Änderungen und Neuerungen hinzu, die das Framework zum einen weiter an andere Frameworks wie React heranrücken und andererseits die Vielseitigkeit von Vue erhöhen. Eine lohnende Version, um ein Projekt mit Vue zu starten oder darauf umzustellen.

Dieses Buch ist für alle gedacht, die sich in die Welt der Web-Frameworks aus Sicht von Vue einarbeiten möchten, um anschließend mit Vue kleine und große Anwendungen entwickeln zu können. Der Fokus liegt auf Vue in der aktuellen Version 3.x mit den Themenschwerpunkten auf allgemeine Framework-Funktionen sowie alles rund um Routing, State Management, Anwendungsentwicklung, Testen, Internationalisierung und Deployment von Vue-Anwendungen. Tipps und Tricks zu Migration von Vue-2-Projekten auf die neue Version sind ebenso enthalten, wie Deep Dives zu den umfangreichsten Änderungen von Vue 3, wie das neue Reactivity-System und Composition API.

Die Inhalte der kommenden Kapitel bauen auf JavaScript, HTML und CSS auf. Wo notwendig, werden Implementierung, Methoden und Vorgehensweise erklärt, zum Beispiel auch zu TypeScript und wichtigen Themen am Rande, wie Scaffolding. Dieses Buch ist auch für alle interessant, die Angular oder React kennen und sich über die Möglichkeiten von Vue informieren möchten.

Die Beispiele und Erläuterungen basieren auf Vue 3. Wo sinnvoll, wird auf die Änderungen zu Version 2 des Web-Frameworks verwiesen. Erstellt sind die Quelltexte und Beispielprojekte mit JavaScript; mit Semikolon hinter jeder Zeile und dem Options API. Wo notwendig, wird auf eine TypeScript-Implementierung verwiesen, da TypeScript im Webumfeld einen immer wichtigeren Platz einnimmt. Auch das Composition API, eine neue Möglichkeit von Vue 3, kommt nicht zu kurz. Alle Beispiele sind in einem Git-Repository auf GitHub verfügbar und nach Kapiteln

organisiert: <https://github.com/fdeitelhoff/vue-3-book>.
Zusätzlich existieren weitere Git-Repositorys, die als Grundlage für die Beispiele in StackBlitz dienen. Diese sind im genannten Repository verlinkt. Als Entwicklungsumgebung kommt Visual Studio Code zum Einsatz, ein kostenfreier Open-Source-Editor, der auf allen Plattformen zur Verfügung steht. Als Paketverwaltung ist die Wahl auf yarn gefallen und beim Module Bundler auf Webpack. Wenn möglich und sinnvoll, gibt es ein Live-Repository auf StackBlitz pro Kapitel für die Beispiele, um diese unkompliziert im Browser ausprobieren zu können.

Ein großes Dankeschön geht außerdem an die beiden Reviewer:



Vanessa Böhner (geborene Otto) arbeitet als Lead Frontend Developer bei Zavvy. Sie hat einen M.Sc. in Medieninformatik und fokussierte sich bei ihrem Studium auf die Mensch-Computer-Interaktion. Vanessa ist Moderatorin für die Podcasts Working Draft, expect(Exception) und Ausbaufähig. Bis 2021 war sie als Chapter Lead für die Region DACH der Front-end Foxes zuständig. Sie war Mitglied des JS-Kongress-Programmkomitees.



Antony Konstantinidis ist seit über zwölf Jahren freiberuflich als Full-Stack-Entwickler, Consultant und Trainer tätig. Hauptsächlich befasst er sich mit der Frontend-Entwicklung und ist super glücklich mit der Entwicklung von und in Vue. Mit vuejs.de hat er letztes Jahr die erste deutsche Anlaufstelle für Vue gegründet.

1 Vue im Überblick

Vue.js, englisch ausgesprochen [vju?], wie *View*, ist ein Framework zur Implementierung von Webanwendungen. Nicht selten wird Vue.js als Bibliothek bezeichnet, weil es sich durch den progressiven Ansatz, der in diesem Kapitel erläutert wird, gut in eigene Projekte jeglicher Größe und Komplexität eingliedert. Der Hauptunterschied zwischen einem Framework und einer Bibliothek ist, dass sich unser Code den Standards und Vorgaben eines Frameworks anpassen muss, während sich eine Bibliothek unseren Standards und Vorgaben anpasst. Da sich Vue.js bereits durch minimale Veränderungen im Projekt integrieren lässt, schwankt die Definition zwischen Framework und Bibliothek. Dieses Buch folgt der Kategorisierung als Framework, da es bei einem konsequenten Einsatz die Architektur der Webanwendung vorgibt - ein unverwechselbares Merkmal eines Frameworks. Im weiteren Verlauf wird Vue.js, je nach Version auch Vue.js 2 oder Vue.js 3 genannt, als Vue bezeichnet. Das *js* als Abkürzung für JavaScript ist im Sprachgebrauch nicht notwendig.

1.1 Was ist ein JavaScript-Frontend-Framework?

Bis in die frühen 2000er-Jahre hinein waren komplexe Webanwendungen im Browser undenkbar. Sowohl von der Performance her als auch vom notwendigen Tooling war an eine Entwicklung von Webanwendungen nicht zu denken. JavaScript war hauptsächlich im Einsatz, um kleinere Funktionalitäten auf Webseiten zu realisieren; weit entfernt vom heutigen Anwendungsfeld und den Möglichkeiten.

Geändert hat sich das mit der Einführung von Google Maps und Gmail im Jahr 2005 in den USA und ab 2006 auch als Webdienst in Deutschland. Zwei Anwendungen, die schon damals bei ihrer Einführung problemlos im Browser liefen - angetrieben durch asynchrone Ajax-Requests, die Anfragen über das Netzwerk möglich machten. Diese Möglichkeiten wurden in den folgenden Jahren konsequent ausgebaut. Browser haben neue Funktionen bekommen, ebenso wie neue und angepasste APIs. Zudem wurde JavaScript als Sprache und Webstandard grundlegend überarbeitet und zum Teil fast neu geschaffen. AngularJS, Ember, Knockout und Backbone.js gehörten zur ersten Welle an neuartigen und modernen Web-Frameworks, um diese rasante Entwicklung von Webanwendungen zu unterstützen. Diese Geburtsstunde von Web-Frameworks hat dazu geführt, dass immer mehr Interaktionen mit dem Browser und dem Document Object Model (DOM) einer Webseite abstrahiert wurden. Das hatte neue Funktionen, bessere Performance und eine verkürzte Entwicklungszeit zur Folge.

Ein Frontend-Framework ist ein Gerüst für das Erstellen eines Frontend. Mit an Bord sind normalerweise Möglichkeiten, Dateien zu strukturieren, zum Beispiel um Komponenten und Daten mit DOM-Elementen zu verknüpfen. Ein Frontend-Framework hilft zudem bei der Trennung der Belange (*Separation of Concerns*), indem es sich ausschließlich um die Anzeige kümmert und Anfragen

bezüglich Daten an ein Backend weiterleitet. Solche Frameworks bieten viele Vorteile, wie bessere Wartbarkeit, saubere Trennung der Belange, höhere Geschwindigkeit, eine gute Zusammenarbeit zwischen Entwicklern und eine breite Community. Es gibt aber auch Nachteile, wie die Einführung einer weiteren Abstraktionsschicht und die damit einhergehende erhöhte Komplexität, eine Lernkurve, die (zu) schnelle Evolution des Ökosystems, die Notwendigkeit, ein Projekt spezifisch nach den Wünschen des Frameworks aufzusetzen, und einen Overkill für kleinere Projekte.

Ein JavaScript-Web-Framework erlaubt es somit, moderne Anwendungen mit modernen Tools zu erschaffen, mittlerweile nicht mehr nur fürs Web im Browser, sondern ebenfalls auf dem Desktop und als mobile Anwendungen auf Smartphones. Wer sich bisher noch nicht sicher war, was ein JavaScript-Front-end-Framework oder Web-Framework bedeutet, hat jetzt hoffentlich einen ersten Einblick bekommen. Zudem ist Vue als Einstieg in die große Welt der Webentwicklung genau richtig.

1.2 Historie und das Team rund um Vue

Vue wurde von *Evan You* erschaffen, der bei Google mit AngularJS an einer Reihe von Projekten gearbeitet hat. In einem Interview sagte er mal, dass er das Beste an AngularJS herauslösen und als leichtgewichtiges Projekt anbieten wollte. Genau das hat er getan. Im Juli 2013 fanden die ersten Commits im Projekt statt. Eine erste öffentliche Version 0.6 erschien im Dezember des gleichen Jahres.

Seitdem ist die Beliebtheit von Vue konstant gewachsen, bis zur Version 3.0, die im September 2020 erschien. Im Jahr 2016 waren es circa 7.500 Sterne, im Jahr 2017 bereits über 36.000. Zudem zählte das Vue-Repository als das populärste Repository im Jahr 2016 ([Stars, 2016](#)). Das GitHub-Repository für Vue 2 ([Team V., GitHub Repository für Vue 2, 2022](#)) mit dem Quelltext des Vue-Projekts zählt mittlerweile über 190.000 Sterne und über 31.000 Forks (beides Stand Februar 2022). Da Vue 3 neuorganisiert wurde, gibt es dazu ein eigenes Repository (Vue, 2022). Das ist eine beachtliche Steigerung. Denn auch wenn Sterne auf GitHub keine verlässliche Metrik für die Beliebtheit sind, sind sie zumindest ein Indikator. Eine aktuelle Auswertung in der Umfrage *The State of JS 2021* kommt zu dem Ergebnis, dass die Beliebtheit und Zufriedenheit etwas abgenommen hat (80%), aber immer noch auf einem hohen Wert verbleibt. Das Interesse und die Nutzung verbleiben bei einem nahezu unveränderten Wert (circa 50%) ([State of JS Team, 2022](#)).

Mittlerweile ist nicht mehr nur Evan You mit der Entwicklung von Vue beschäftigt. Eine ganze Reihe von Personen auf der ganzen Welt arbeitet gemeinsam am Projekt und den zahlreichen offiziellen Erweiterungen.

1.3 Was ist Vue und was zeichnet es aus?

Das Ziel von Vue ([Team V., Vue.js - The Progressive JavaScript Framework, 2022](#)) ist es, die Vorteile reaktiver Datenbindung (*Reactive Data Binding*) und kombinierbarer Komponenten (*Composable Components*) mit einer einfach

zu nutzenden Schnittstelle zu verknüpfen und zur Verfügung zu stellen.

Diese einfache API und die damit verbundene niedrige Hürde für den Einstieg sind ein Merkmal, das Vue auszeichnet und von vielen anderen Web-Frameworks und -Bibliotheken abhebt. Denn andere Frameworks und Bibliotheken gehen oft den Weg, eine Schnittstelle anzubieten, die so umfassend wie möglich ist. Ein weiteres Merkmal ist, dass Vue nicht als das alles umfassende Web-Framework gedacht ist. Vue nutzt das *Model-View-ViewModel*-(MVVM-) Entwurfsmuster und fokussiert sich auf den View-Layer und nur auf diese Schicht – ein weiterer Punkt, warum Vue als einfach zu adaptieren gilt, wenn es um den Einsatz im eigenen Projekt geht. Diese Designentscheidungen führen zudem dazu, dass sich Integration und Kombination mit anderen Bibliotheken einfach umsetzen lassen. Vue sperrt sich dabei selten, und die Community hat eine umfassende Palette an guten Erweiterungen geschaffen, die gerne als First-Party-Erweiterungen bezeichnet werden.

Diese Merkmale bedeuten aber nicht, dass wir mit Vue keine umfangreichen oder anspruchsvollen Webprojekte stemmen können. Mit Vue lassen sich sehr wohl hochkomplexe *Single Page Applications (SPAs)* erzeugen, was die zahlreichen kurz vorgestellten Use Cases gegen Ende dieses Buchs beweisen.

1.4 Warum Vue?

Da mittlerweile zahlreiche Web-Frameworks existieren, steht immer die Frage im Raum, warum dieses oder jenes Framework für das eigene Projekt zum Einsatz kommen soll. In diesem Fall also die simple Frage: Warum Vue?

Das läuft auf die Frage hinaus, was Vue für einen konkreten Mehrwert bietet. Hier kommt der progressive Ansatz von Vue zum Tragen. Die Macher von Vue bezeichnen es als progressives Framework, weil Vue es erlaubt, mit minimalem Aufwand eine Anwendung mit Vue zu starten oder eine bereits bestehende mit einer ersten Vue-Komponente auszustatten. Dazu ist es nicht notwendig, die gesamte Architektur des Projekts von Grund auf neu zu entwickeln, damit es den Anforderungen von Vue gerecht wird. Die minimal notwendigen Schritte sind, Vue einzubinden, im einfachsten Fall als loses Skript, um dann eine Seite mit einer View-Komponenten auszustatten. Diese Änderungen als Grundlage genutzt, lassen sich weitere Komponenten einsetzen, Komponenten miteinander kombinieren und dergleichen. Wenn mit der Zeit die Anforderungen wachsen, lassen sich weitere Komponenten einbeziehen, zum Beispiel für das Routing oder das State Management.

Diese Möglichkeit, Vue in einem Projekt nach und nach einzusetzen, ist unter Entwicklerinnen und Entwicklern ein häufiger Grund, warum die Wahl auf Vue fällt. Insbesondere beim Einsatz des ersten Web-Frameworks ist diese iterative Adaption von Features ein großer Vorteil. Vue passt sich den Fähigkeiten der Entwicklerin oder des Entwicklers an und nicht andersherum. Das wirkt aktiv gegen den Trend vieler Frameworks und Bibliotheken, Neuankömmlinge abzuschrecken, weil sich das gesamte Projekt nach den Vorgaben des Web-Frameworks richten muss, was dazu führen kann, dass sich Einsteiger am Anfang komplett verloren fühlen. Vue wird daher manchmal als das neue jQuery bezeichnet, weil es ebenso einfach einsetzbar ist wie die einstige Vorzeigebibliothek für dynamische Webseiten.

Vue besitzt einige weitere Vorteile. Es ist klein, was die Downloadgröße angeht, besitzt das Prinzip der Single-File Components und damit eine gute Lesbarkeit, ein solides Tooling- und Ökosystem und gilt im Allgemeinen als einfach zu nutzen. Aber natürlich ist nicht alles eitel Sonnenschein. Mit Vue 3 wurde das Reactivity-System verbessert, es ist aber immer noch recht komplex. Außerdem besteht die Gefahr der Über-Flexibilisierung. Wenn Vue ohne große Hürden überall zum Einsatz kommen kann, kann es zu nicht gut wartbarem Code kommen. Hier ist etwas Disziplin gefordert.

1.5 Unterschiede zu anderen Web-Frameworks

Über die Jahre hat sich eine ganze Reihe von Web-Frameworks etabliert. Allen voran sind sicherlich Angular und React zu nennen, die eine umfassende Community und eine Vielzahl prominenter Webprojekte hinter sich vereinen. Vue muss sich allerdings nicht mehr hinter diesen Projekten verstecken. Insbesondere mit Vue 3 sind erhebliche Änderungen in das Projekt eingeflossen, sodass sich die drei großen Web-Frameworks Angular, React und Vue weiter annähern.

Beim Vergleich zwischen Angular und Vue stechen einige Punkte heraus. Das mögen von Projekt zu Projekt Gründe sein, um Vue einem Einsatz von Angular vorzuziehen. In der Regel ist der Einsatz eines Web-Frameworks von Fall zu Fall zu entscheiden. Vue ist beispielsweise deutlich einfacher zu nutzen als Angular, was primär am API-Design liegt. Der Einstieg in Vue gelingt häufiger einfacher und schneller. Zudem ist Vue in vielen

Fällen flexibler und bringt eine weniger starke eigene Meinung mit, wie ein Projekt zu strukturieren ist. Vue ist primär eine Schnittschicht (*Interface Layer*), sodass sich auf Webseiten einige Features von Vue nutzen lassen, ohne direkt alles umbauen und umstrukturieren zu müssen. Ein großer Unterschied zu Angular ist noch, dass Vue im Kern praktisch keine zusätzlichen Features mitbringt, zum Beispiel kein Routing. Der Ansatz von Vue ist die Annahme, dass in einem Web-Framework sowieso ein Bundler wie Webpack für externe Module zum Einsatz kommt. Ein weiterer Unterschied zwischen Angular und Vue ist, dass Vue im Standard auf ein One-Way-Data-Binding zwischen Eltern- und Kindkomponente setzt, wohingegen Angular Two-Way anbietet. Vue trennt zudem ganz klar zwischen Direktive sowie Komponente und hat vielfach die bessere Performance. Ein Grund dafür ist das transparente Tracking von Abhängigkeiten auf Basis eines asynchronen Queuing-Verfahrens. Alle Änderungen werden unabhängig voneinander getriggert, außer sie haben eine explizite Abhängigkeitsbeziehung. Viele dieser Probleme und Unterschiede wurden in vergangenen und werden in zukünftigen Versionen von Angular angepackt, sodass diese Unterschiede von Version zu Version kleiner werden.

React und Vue teilen sich viele Gemeinsamkeiten, da beide Web-Frameworks reaktive und kombinierbare Komponenten für den View-Layer anbieten. React nutzt für Änderungen das Virtual DOM, ein vormals großer Unterschied zu Vue. Mittlerweile implementiert Vue ebenfalls ein virtuelles DOM, eine Light-Version des realen DOM. Hier haben sich React und Vue deutlich angenähert. Aus Sicht des API wird es häufig kritisch gesehen, dass die Render-Funktionen in React viel Logik enthalten und somit nach und nach eher einem Programm gleichen, was sie letztendlich auch sind, anstatt sich nur um die visuelle

Repräsentation zu kümmern. Das mag für Entwickler noch verträglich sein, für Designer kann es die Arbeit mit diesen Templates erschweren. Das React-Team verfolgt zudem das ambitionierte Ziel, React zu einem plattformunabhängigen UI-Entwicklungsparadigma zu entwickeln. Vue konzentriert sich im Gegensatz dazu darauf, eine einfache pragmatische Lösung für das Web zu bieten. React arbeitet zudem gut mit funktionalen Mustern (Patterns) zusammen, was gerade zu Beginn zu einer etwas steileren Lernkurve und mehr Hürden führen kann. Sind diese funktionalen Muster aber bekannt, kann React seine ganze Stärke ausspielen. Vue gilt als gemeinhin einfacher für den Einstieg. Mit Vue 3 und dem Composition API, ebenfalls Thema in diesem Buch, geht aber auch das neue Vue einen deutlich funktionaleren – aber optionalen – Weg. Ein großer Vorteil von React ist das ausgezeichnete State Management. Mit Flux/Redux (([Facebook, 2022](#)) und ([Abramov, Redux, 2022](#))) werden Möglichkeiten angeboten, die lange Zeit allen anderen Frameworks voraus waren. Vue setzt hier, getreu dem Designprinzip des Web-Frameworks, auf externe Bibliotheken wie Vuex, und auch Redux lässt sich mit Vue kombinieren. Als letztes Merkmal ist der Trend bei React zu nennen, alles in JavaScript einzubetten, auch die CSS-Anweisungen. Vue löst das mit dem Feature der Single-File Components, in denen das CSS pro Komponente gekapselt ist, aber als reines CSS bereitsteht. Dadurch können zum Beispiel weiterhin Präprozessoren für CSS zum Einsatz kommen.

Vue grenzt sich auch von Frameworks wie Ember, Polymer und Riot in vielen Aspekten ab. In Ember ist die Reaktivität beispielsweise nur über eigene Ember-Objekte herzustellen, zudem ist die Template-Syntax von Vue einfacher zu nutzen, da sich JavaScript-Anweisungen integrieren lassen. Zudem ist Vue noch immer ein ganzes