Fabian Deitelhoff

Vue.js

Von Grundlagen bis Best Practices

Für den Um- und Einstieg in Vue 3

dpunkt.verlag

Foto: Allona Kardash/TU Dortmund

Dr. Fabian Deitelhoff arbeitet nach seiner Promotion zu »Source Code Comprehension« als Tech-Lead Domestics 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*.

Fabian Deitelhoff

Vue.js

Von Grundlagen bis Best Practices

dpunkt.verlag

Fabian Deitelhoff
fabian@deitelhoff.me

Lektorat: Melanie Andrisek
Lektoratsassistenz: 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*
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 978-3-86490-900-9
PDF 978-3-96910-760-7
ePub 978-3-96910-761-4
mobi 978-3-96910-762-1

1. Auflage 2022

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

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

Inhaltsverzeichnis

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 einem 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 moder-

nen 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-Frontend-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 ei-

ner 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 anbelangt, 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 Schnittstellenschicht (*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, das 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 Stückchen performanter. Bei Polymer ist zum Beispiel ein Nachteil, dass das Framework auf dem aktuellen Feature der Web Components fußt. Für Browser, die das nicht unterstützen, sind umfangreiche und nicht-triviale Polyfills notwendig. Im Vergleich zu Riot gibt es eine ganze Reihe von Unterschieden. Riot rendert zum Beispiel alle if-Verzweigungen immer und blendet je nach Bedingung dann lediglich die nicht benötigten aus. Dadurch entstehen Performance-Bottlenecks. Zudem sind das Tooling und die externen Module, wie das Routing, in Vue deutlich fortschrittlicher und mächtiger.

Dieser Überblick kann nur ein Ausschnitt aus der Welt der Web-Frameworks darstellen, da sich das Web und die Technologien darin beständig weiterentwickeln. Für einen vollständigen Überblick gibt es zu viele Frameworks und Bibliotheken, und die Geschwindigkeit, mit der Änderungen eingebracht werden, ist schwindelerregend.

1.6 Vue 2 oder Vue 3 lernen?

Wer gerade anfängt, Vue kennenzulernen, sollte direkt mit der Version 3 starten. Das ist diejenige, die in Zukunft weiterentwickelt wird und die neuen Konzepte inklusive vieler Verbesserungen nutzt. Zudem sind zahlreiche Funktionalitäten optional, lassen sich also gut graduell einsetzen.

In der Praxis ist die Entscheidung aber gar nicht so leicht. Es hängt zum Beispiel davon ab, ob in einem Team gearbeitet wird, das Vue 2 verwendet und keine Pläne für eine Migration auf Vue 3 hat. Es hängt auch vom eigenen Zeitplan ab und ob die eigene App Abhängigkeiten benötigt, die noch nicht mit Vue 3 kompatibel sind. Das wird sich zwar über die Zeit aller Voraussicht nach lösen, aber in solchen Situationen ergibt es vielleicht keinen Sinn, die neueste Syntax und APIs von Vue 3 zu lernen, wenn diese nicht in der täglichen Arbeit zum Einsatz kommen können. Bis auf Mixins kann Vue 3 so eingesetzt werden, wie es von Vue 2 bekannt ist. Es gibt daher keinen triftigen Grund, noch mit Vue 2 zu starten, wenn die in Vue 3 fehlenden Mixins kein Argument sind.

Der Einstieg mit Vue 3 ist also keine schlechte Idee und zukunftssicher. So ist sichergestellt, dass wir nicht in Kürze abgehängt sind, weil Vue 2 dann doch schneller ausläuft als angenommen.

1.7 Der Blick in die Zukunft

Auch wenn die Veröffentlichung von Vue 3 und der Wechsel zur Version 3 als offizielle Version im Februar 2022 zwei Events zum Feiern im *Vueniverse* waren, so blieb die Entwicklung danach natürlich nicht einfach stehen. In Zukunft sind weitere neue Features geplant.

Dazu gehört *Reactivity Transform*, der finale Baustein für das Reactivity-System und das Script Setup. Damit wird es deutlich einfacher, reaktive Daten zu erstellen, und es verkleinert die Einstiegshürde für alle, die von Vue 2 kommen.

Auch im Ökosystem sind noch einige Änderungen zu erwarten. Zum Beispiel die stabile Version von Nuxt 3 mit zahlreichen Änderungen im Gepäck. So ähnlich sieht es bei Vuetify aus, einem populären Framework basierend auf dem Material Design. Das Team rund um Vuetify arbeitet aktuell an der finalen Unterstützung von Vue 3.

2 Die Grundlagen: Ein Vue-Schnellstart

Trotz der zahlreichen Anpassungen, die Vue über die Jahre erfahren hat, ist dem Web-Framework der progressive Charakter nicht abhandengekommen. Ganz im Gegenteil wurde dieses primäre Designmerkmal weiter gepflegt. Das wird sichtbar, wenn es darum geht, eine erste minimale Vue-Anwendung zu erstellen: ohne großes Aufsehen und ohne großen Aufwand.

Vue besitzt einige Kernfunktionen, mit denen der generelle Funktionsumfang des Web-Frameworks gerne beschrieben wird. Die Vue-Instanz ist eine davon. Jede Anwendung besitzt mindestens eine dieser Instanzen. Grundsätzlich sind aber beliebig viele Vue-Instanzen pro HTML-Datei erlaubt.

Eine weitere Kernfunktion sind Komponenten. Vue-Instanzen lassen sich durch Komponenten erweitern, die sich zudem verschachteln lassen. Eine Komponente enthält das Template, bestehend aus HTML-Elementen, um eine Webseite oder einen Teil davon zu definieren. Komponenten können im Projekt hervorragend wiederverwendet werden. Mit dem neuen Composition API (siehe Kapitel 8) von Vue 3 sogar noch deutlich besser als das bisher der Fall war.

Des Weiteren nutzt Vue eine deklarative auf HTML basierende Template-Syntax, um das DOM an die Daten einer Vue-Instanz zu binden. Zum Beispiel lassen sich dadurch Daten mit geschweiften Klammern an eine Variable binden, die dann in der View ausgegeben werden.

Das Binding und die Direktiven werden ebenso häufig im Zusammenhang mit Vue genannt. Sowohl Attribute als auch Methoden oder Variablen können mit wenig Aufwand gebunden werden, sodass sich Datenänderungen oder Benutzerinteraktionen direkt auf das zugrunde liegende Modell auswirken. Direktiven sorgen dafür, dass zum Beispiel das Rendern von HTML-Elementen an Bedingungen geknüpft werden kann.

2.1 Die erste Vue-Anwendung aufsetzen

Eine erste Vue-Anwendung aufzusetzen, ist nicht viel Aufwand. In diesem Abschnitt liegt der Fokus auf den Grundlagen, mit denen Vue schnell im eigenen Projekt integriert ist: ohne das vollständige Tooling rundherum zu nutzen. Wie

neue Vue-Projekte aufgesetzt werden, ist Inhalt von Kapitel 4. Die nachfolgenden Beispiele zeigen, wie Vue als einfaches Skript einzubinden ist, um die Vue-Application-Instanz zu nutzen, sowie HTML und die darin enthaltenen Daten anzuzeigen. Jede Vue-Anwendung beginnt damit, so eine Instanz zu erzeugen. Genau genommen ist diese *Application Instance* eine Komponente, die als Wurzel-Komponente die Anwendung definiert. Das macht deutlich, wie wenig Overhead Vue verursacht, wenn wir Vue lediglich als Skript einbinden, ohne einen Build-Prozess vorauszusetzen. Zudem zeigt es, dass alles auf Komponenten basiert.

2.1.1 Die HTML-Struktur

Zunächst ist ein HTML-Dokument notwendig. Dazu reicht ein Standard-HTML5-Dokument völlig aus. Das verdeutlicht, dass Vue praktisch keine Anforderungen an die HTML-Elemente und die HTML-Struktur stellt. Ein Head, ein Body, und los geht es.

```
<!DOCTYPE html>

<html lang="de">
  <head>
    <meta charset="utf-8" />

    <title>Vue-Grundlagen</title>
    <meta name="description" content="Das Vue-3-Buch" />
    <meta name="author" content="Dr. Fabian Deitelhoff" />
  </head>

  <body>
    <p>Ein wenig Inhalt als einfacher Test...</p>
  </body>
</html>
```

Um Vue in diesem Beispiel zum Laufen zu bekommen, ist das Einbinden von Vue als Skript und ein Skript-Element zum Erzeugen unserer ersten App notwendig. Außerdem benötigen wir ein HTML-Element zum Andocken (*mounten*) unserer Vue-App. Diese Schritte zeigen die nachfolgenden Abschnitte in diesem Kapitel.

2.1.2 Einbinden von Vue als Skript

Für unseren einfachen Anwendungsfall gibt es zwei Möglichkeiten, Vue einzubinden: entweder als lokales Skript, das zuvor von der Vue-Webseite heruntergeladen wurde, oder direkt über ein *Content Delivery Network* (CDN). In diesem Beispiel kommt die zweite Variante zum Einsatz. Das nachfolgende Beispiel zeigt, wie das HTML-Dokument im Body um ein paar Anweisungen erweitert werden muss, damit Vue als Skript eingebunden ist. Zudem kamen das HTML-Element zum Mounten der Anwendung und das Skript-Element zum Implementieren des kleinen Beispiels hinzu.

Der Standardweg zu Vue

In komplexeren Projekten wird in der Regel eine Paketverwaltung wie npm oder yarn genutzt oder eine Projektstruktur automatisch über die Kommandozeile erzeugt. Es gibt unterschiedliche Skripte und Pakete, zum Beispiel für das Server-Side-Rendering, ohne Bundler, ohne Compiler und dergleichen. Wie eine Projektstruktur für Vue erzeugt wird, zeigt Kapitel 4. Das ist der etablierte Standardweg außerhalb von kleinen Test- oder Beispielanwendungen wie unser Beispiel hier. Bei Projekten zur Entwicklung umfangreicher Anwendungen werden zudem gerne Tools wie Vite, Vuetify oder Quasar genutzt. Damit lassen sich Anwendungen komfortabel und schnell entwickeln. Zu diesen Themen, mit einem Fokus auf Quasar, verrät das Kapitel 12 weitere Informationen.

```
<body>
  <p>Ein wenig Inhalt als einfacher Test...</p>

  <!-- HTML-Element (div) zum Mounten der App. -->
  <div id="first-app"></div>

  <!-- Die Referenz auf die Vue-Bibliothek. -->
  <script src="https://unpkg.com/vue@3"></script>

  <!-- Das Script-Element für die erste Vue-App. -->
  <script></script>
</body>
```

Für diese Demo ist es ausreichend, das Skript von `/vue@3` einzubinden. Dass es sich hierbei um eine Vue-Version für die Entwicklung handelt, meldet auch prompt die Konsole, wenn die entsprechende HTML-Datei im Browser angezeigt wird:

```
You are running a development build of Vue.

Make sure to use the production build (*.prod.js) when deploying for
production.
```

Das ist eine wichtige Information. Zudem ist es ratsam, im produktiven Einsatz beim Einbinden auf eine spezifische Version zu verweisen. Das verhindert unerwartete Probleme, wenn sich Implementierungen zwischen den verschiedenen Versionen ändern.

Unser Beispiel nutzt den globalen Build von Vue, in dem alle APIs unter dem globalen Vue-Variablen verfügbar sind. Wer etwas mehr Konsistenz zu der ES-Modul-Syntax möchte, kann Vue auch als Modul im Browser einbinden:

```
<script type="importmap">
  {
    "imports": {
      "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
    }
  }
</script>
```

Anschließend stehen die bekannten Imports zur Verfügung; als wäre ein Build-Prozess mit Build-Tools im Einsatz. Diese Art, Vue einzubinden, ist in der Regel nur für Demozwecke gut geeignet. Später ergeben Build-Tools und ein umfassenderes Projekt-Setup mehr Sinn.

2.1.3 Unsere erste Vue-Instanz

Für unsere erste Vue-Instanz binden wir zunächst ein kleines Fragment in das HTML-div-Element ein, das zum Mounten der Anwendung dient. Dieses sieht dann wie folgt aus:

```
<!-- HTML-Element (div) zum Mounten der App. -->
<div id="first-app">
  <p>{{ info }}</p>
</div>
```

Vue nutzt, wie viele andere Web-Frameworks auch, eine auf HTML-basierende Template-Syntax. Elemente des daraus erzeugten *Document Object Model* (DOM) werden an die darunterliegende Vue-Instanz gebunden.

Im einfachsten Fall kann eine Interpolation dazu genutzt werden, um Daten im HTML-Dokument anzuzeigen. Vue nutzt dazu, ebenfalls wie viele andere Bibliotheken und Frameworks auch, die sogenannte *Mustache-Syntax* (Team M., 2022) mit zwei öffnenden und schließenden, geschweiften Klammern. In dieser Demo `{{ info }}`. Ohne eine aktive Vue-Instanz wird dieses Fragment als normales HTML interpretiert, und der Browser zeigt es einfach an. Denn bei allen Vue-Templates handelt es sich um valides HTML, das vom Browser verarbeitet und interpretiert werden kann. Die Dynamik kommt erst durch die Vue-Instanz ins Spiel.

Das Skript-Tag, das zuvor als Vorbereitung für die Vue-Instanz in das HTML-Dokument eingebaut wurde, wird nun wie folgt angepasst:

```
<!-- Das Script-Element für die erste Vue-App. -->
<script>
  const FirstVueApp = {
    data() {
      return {
        info: 'Inhalt aus der Vue-Instanz...',
      };
    },
  };

  Vue.createApp(FirstVueApp).mount('#first-app');
</script>
```

Options API in diesen Beispielen

Dieses und die nachfolgenden Beispiele nutzen das Options API (siehe Abschnitt 6.3.1) von Vue 3. Von der `data`-Eigenschaft wird erwartet, dass diese als Funktion implementiert ist und ein reines JavaScript-Objekt zurückgibt. Wir können an dieser Stelle eine Pfeil-Funktion (Arrow Function) nutzen, allerdings ist `this` dann nicht mehr die Instanz der Komponente. Zugriff auf diese bekommen wir dann über den ersten Parameter der Funktion.

Mit diesem Codefragment wird ein simples JavaScript-Objekt mit dem Namen `FirstVueApp` erzeugt. Dieses Objekt wird anschließend an die `createApp`-Methode übergeben. Das ist das sogenannte Bootstrapping einer Vue-App. Das JavaScript-Objekt `FirstVueApp` wird als Vue-Instanz-Definition angesehen und zurückgeliefert. Das Resultat des Aufrufs kann dann über das Fluent Interface mit der `mount`-Methode über einen CSS-Selektor an ein HTML-Element angehängt werden. In diesem Beispiel wird das Element mit der ID `#first-app` genutzt. Das Objekt, das wir an `createApp` weiterreichen, ist in Wahrheit bereits unsere erste Komponente und auch gleichzeitig die Wurzel-Komponente.

Achtung bei Vue 2

In Vue 2 sieht der Teil mit dem Mounting der globalen Vue-Instanz wie folgt aus.

```
const app = new Vue({
  render: (h) => h(App),
}).$mount('#app');
```

Der Import der benötigten Komponenten wird der Übersicht halber in beiden Code-Snippets weggelassen.

In Vue 3 gibt es keine einzelne globale Instanz mehr. Ein wesentlicher Grund für diese Änderung ist, dass dadurch nicht mehr das globale Vue-Objekt verändert wird, zum Beispiel wenn Plug-ins oder Mixing von Drittanbietern an diese globale Instanz gebunden werden. Das ist insbesondere beim automatisierten Testen ein Problem, wenn die globale Instanz über mehrere Tests hinweg existiert. Dann ist nicht sichergestellt, dass jeder Test die gleichen Voraussetzungen nutzt, nämlich eine unveränderte Vue-Instanz. Auch das Einbinden eines Routers (siehe Kapitel 9) und Vuex (siehe Kapitel 10) geschieht nun als einfaches Plug-in. Das erhöht die Transparenz deutlich. Dieses *Global Mounting*, wie der Prozess genannt wird, hat sich von Vue 2 zu 3 deutlich geändert. Unsere erste Vue-App ist damit bereits erstellt.

2.1.4 Schleifen und Bedingungen

Das Beispiel mit der ersten Vue-Instanz hat gezeigt, dass sich Vue mit wenig Aufwand in ein HTML-Dokument integrieren lässt; inklusive Data Binding von Variablen aus dem Datenobjekt im HTML-Template. Dadurch eröffnen sich sofort weitreichende Möglichkeiten: zum Beispiel durch Schleifen und Bedingungen. Denn ein Datenobjekt einer Vue-Instanz kann nicht nur einfache Daten wie Zeichenketten und numerische Werte enthalten, sondern auch Arrays und weitere komplexe Objekte.

Die folgende Liste von Elementen enthält vier JavaScript-Objekte mit zwei Attributen und Beispieldaten, die im Datenobjekt der Vue-Instanz hinterlegt sind, direkt unter dem `info`-Attribut.

```
events: [
  { id: 1, name: 'Veranstaltung 1', participantCount: 7 },
  { id: 2, name: 'Veranstaltung 2', participantCount: 9 },
  { id: 3, name: 'Veranstaltung 3', participantCount: 13 },
  { id: 4, name: 'Veranstaltung 4', participantCount: 22 },
],
```

Das Beispiel zeigt die Dateneigenschaft `events` und die enthaltenen Objekte. In diesem Fall einfache JavaScript-Objekte mit zwei Eigenschaften. Diese Liste mit JavaScript-Objekten kann zum Beispiel im HTML-Template durchlaufen und die enthaltenden Daten anzeigen. Das dafür notwendige HTML-Markup ist überschaubar und beinahe selbsterklärend. Die notwendigen Merkmale und Funktionen von Vue sind schnell beschrieben. Die folgenden HTML-Deklarationen sind im Dokument neu hinzugekommen, um die Informationen im Array darzustellen.

```
<div v-for="event in events">
  <p>
    ID: {{ event.id }}
    <br />
    Name: {{ event.name }}
    <br />
    Anzahl Teilnehmende: {{ event.participantCount }}
  </p>
</div>
```

Dieses HTML-Snippet realisiert die Ausgabe aller Demo-Events in der Eigenschaft `events` unserer Vue-Instanz. Eine Schleife ist in Vue mit der Direktive `v-for` realisiert. Bei der Syntax sind ein Array als Quelle und ein Alias für das aktuelle Element eines Schleifendurchlaufs gefordert. In diesem Fall ist `events` das Quell-Array und `event` der Alias für das aktuelle Element. Beim Schleifendurchlauf wird das HTML-Element mit der Direktive `v-for` pro Element im Array dupliziert. Im Beispiel gibt es daher vier `div`-Element mit jeweils einem `p`-Element. Bei einer HTML-Liste müsste das `v-for` daher beim `li`-Element platziert werden, damit es der Array-Größe entsprechend viele Listenelemente gibt: