Fabian Deitelhoff

Vue.js

Von Grundlagen
bis Best Practices

Für den Um- und Einstieg in Vue 3

dpunkt.verlag

Inhalt

3 Vue 3 im Überblick

In Vue 3 sind zahlreiche Änderungen eingeflossen, die in einem länger andauernden Prozess definiert, der Community im Voraus vorgestellt und mit dieser diskutiert worden sind. Diese Diskussionen hatten nicht immer einen Fokus auf einen sachlichen Diskurs. Die Ankündigungen, erste Request-for-Comments-Dokumente (RFCs) und Beispiele zu den geplanten Änderungen zu Vue 3 haben durchaus kontroverse und heftige Diskussionen ausgelöst. Das betrifft insbesondere die Spezifikation zum Composition API (vorgestellt im Juni 2019). Viele Diskussionen wurden durch Missverständnisse befördert. Zum Beispiel war nicht allen von Anfang an klar, dass das neue Composition API optional ist und das vorhandene Options API nicht ersetzen sollte. Letztendlich zeigen diese Reaktionen deutlich, dass Änderungen an Frameworks aller Art durchaus stressig sind und sich eine Community missverstanden oder angegriffen fühlen kann.

Zahlreiche Neuerungen haben es in die finale Version geschafft, bei anderen wurden verschiedene Implementierungen getestet und wiederum andere Ideen dagegen komplett verworfen. Laut Release-Post (Team V., Release-Post zu v3.0.0 One Piece, 18) auf GitHub hat sich die Performance deutlich verbessert (Team V., Vue 3 vs. Vue 2 Perf comparison, 2022) sowie die Größe der Bibliothek und der Speicherverbrauch verringert.

3.1 Warum neu schreiben?

Eine primäre Frage bei der neuen Vue-Version ist, warum diese zu einem Großteil neu implementiert wurde. Diese Tendenz gibt es bei Softwareprojekten immer wieder, weil sich die Rahmenbedingungen, Anforderungen und technischen Voraussetzungen ändern und die Verlockung groß ist, bei der nächsten Version noch mal von vorne anzufangen.

Das Vue-Team hat sich dennoch dazu entschlossen, die Codebasis von Vue 2 grundlegend zu verändern. Zwei primäre Abwägungen haben zu dieser

Entscheidung geführt: Zunächst die allgemeine Verfügbarkeit vieler neuer JavaScript-Sprach-Features in Browsern, die mittlerweile in der Community umfassend genutzt werden. Darüber hinaus gibt es viele Verbesserungen bei den Design- und Architekturentscheidungen von Vue, die sich über die 2,5 Jahre Entwicklungszeit der Vue-2-Codebasis herauskristallisiert haben und jetzt umgesetzt werden.

Die primären Gründe, Vue 3 neu zu implementieren, sind daher die Nutzung neuer JavaScript-Funktionen, zum Beispiel aus ECMAScript 2015 (ES2015). Am nennenswertesten sind die neuen Proxys, die Vue für das verbesserte Reactivity-System nutzt (siehe Kapitel 5). Zudem wurde die Codebasis von JavaScript auf TypeScript umgestellt, um das Typsystem nutzen zu können. Auf der Architekturseite wurden interne Pakete vom Vue-Kern entkoppelt, um explizite Schnittstellen zu definieren. Das verbessert die Wartbarkeit und ermöglicht es Entwicklern aus der Community, einfacher Änderungen beizusteuern. Umfangreiche Anpassungen wurden bei der Art und Weise vorgenommen, wie das virtuelle DOM erzeugt wird. Der neue Compiler ist in der Lage, Optimierungen durchzuführen, um so viele unnötige Render-Durchläufe wie möglich einzusparen.

Mit dem Rewrite von Vue wurden daher zahlreiche neue Möglichkeiten geschaffen und alte Probleme ausgemerzt, sodass die neue Version von Vue zahlreiche Vorteile gegenüber der Version 2 besitzt.

3.2 Ein mehrschrittiges Release

Vue hat sich in Version 3 deutlich stärker bestehenden Web-Frameworks angenähert, was unter anderem am Composition API liegt, mit dem sich lose gekoppelte Komponenten implementieren lassen. Da sich dieses Buch auf Vue 3 fokussiert, sind alle Beispiele und Erklärungen darauf bezogen. Der gesamte Veröffentlichungsprozess zu Vue 3 war mehrschrittig. Ein Vorabcheck der im Projekt eingesetzten Bibliotheken und Tools ist dennoch Pflicht, damit nicht erst bei der Migration ein Show-Stopper auffällt. Wichtige Punkte einer Migration von Version 2 auf Version 3 behandelt das Buch in Kapitel 17 ebenfalls.

Mit *petite-vue* wird zudem ein spezielles Vue-Release angeboten, das einige besondere Eigenschaften vereint. Diese alternative Distribution von Vue ist für das sogenannte *Progressive Enhancement* optimiert. Dahinter verbirgt sich eine Designphilosophie, die möglichst vielen Nutzern einen Grundstock an wesentlichen Inhalten und Funktionen zur Verfügung stellt, während das bestmögliche Erlebnis nur den Nutzern der modernsten Browser geboten wird,

die den gesamten erforderlichen Code ausführen können. Das Wort *progressiv* in *Progressive Enhancement* bedeutet, dass ein Design geschaffen wird, das für Benutzer älterer Browser und Geräte mit eingeschränkten Möglichkeiten ein einfacheres, aber dennoch brauchbares Erlebnis bietet, während es gleichzeitig ein Design ist, das das Benutzererlebnis für Benutzer neuerer Browser und Geräte mit umfangreicheren Möglichkeiten zu einem überzeugenderen Erlebnis mit vollem Funktionsumfang weiterentwickelt.

Das Projekt bietet die gleiche Template-Syntax und das gleiche Reaktivitäts-Modell wie das Standard-Vue an. Es ist jedoch speziell dafür optimiert, eine kleine Menge an Interaktionen auf eine bestehende HTML-Seite, die von einem Server-Framework gerendert wird, zu streuen. petite-vue ist nur sechs Kilobyte groß und arbeitet direkt auf dem DOM. Es wird eine Untermenge der Funktionalität von Vue angeboten.

3.3 Interne und externe Anpassungen

Ganz grundlegend gab es in Vue interne und externe Änderungen. Die externen fallen bei der täglichen Arbeit mit Vue am stärksten auf. Das sind zum Beispiel Anpassungen am API oder am Verhalten von Tools beziehungsweise Bibliotheken. Oft haben sich Schnittstellen geändert oder sind komplett weggefallen. Interne Anpassungen, die in Vue 3 ebenfalls zuhauf vorhanden sind, bleiben dagegen eher im Verborgenen. Dazu gehört prominent die Modularisierung des Web-Frameworks, in offiziellen Dokumenten häufig mit *Layered Internal Modules* beschrieben. Die Kernfunktionen befinden sich jetzt nicht mehr in einem einzelnen und damit umfangreichen Core, sondern in einzelnen Paketen. Das ermöglicht eine Entkopplung, da alle Module eigenständig sind. Vue besitzt nun *drei* Kernmodule: das *Reactivity*-Modul, das *Compiler*-Modul und das *Renderer*-Modul. Ersteres erlaubt es, reaktive JavaScript-Objekte zu erstellen, die auf Änderungen überwacht werden können. Das Compiler-Modul besitzt die Implementierungen, die notwendig sind, um HTML-Templates zu Render-Funktionen zu kompilieren. Meistens läuft dieser Prozess, wenn unser Vue-Projekt erzeugt wird. Das Renderer-Modul enthält Implementierungen für *drei* verschiedene Phasen, um eine Vue-Komponente in einer Webseite zu rendern: die *Render*-Phase, *Mount*-Phase und *Patch*-Phase.

Die folgende Liste fasst die wichtigsten Änderungen zusammen, die in Vue 3 Einzug gehalten haben. Zusätzlich zeigt die Liste auch wichtige Änderungen im Ökosystem. Ein Deep Dive zu diesen Features ist in den nachfolgenden Abschnitten und Kapiteln enthalten.

- Änderungen beim Mounten (Global Mounting) (Abschnitt 2.1.3)
- Verbesserte Reaktivität (Kapitel 5)
- Fragmente und native Portale durch Teleports (Kapitel 6)
- Anpassungen bei CSS-Variablen (Kapitel 7)
- Composition API (Kapitel 8)

Beim Ökosystem sieht es mit der Unterstützung für Vue 3 ebenfalls sehr gut aus. Die primären Tools und Bibliotheken wie das Vue CLI (Kapitel 4), der Vue Router (Kapitel 9) und Vuex (Kapitel 10) haben sehr früh in Major-Versionen Anpassungen für Vue 3 bekommen. Das gilt ebenso für zahlreiche Plug-ins, zum Beispiel Vetur für die Integration von Vue 3 in Entwicklungsumgebungen (Abschnitt 4.3). Dem Einsatz von Vue 3 steht damit bereits seit dem Frühjahr 2021 nichts mehr im Wege. Im produktiven Kontext ist der Einsatz seit circa Sommer 2021 empfehlenswert.

3.4 Der neue Standard und Framework-Empfehlungen

Wie bereits kurz angesprochen, ist Vue 3 der neue Standard. Das Ökosystem hat sich angepasst und neue Projekte, zum Beispiel über das CLI, werden mit Vue 3 als Standard erzeugt, und die offizielle Dokumentation nimmt auch Vue 3 als Standard an. Das bedeutet auch, dass Repositorys und Links von CDNs nicht mehr auf den *next*-Tag verweisen. Hier sind eventuell Anpassungen im eigenen Projekt notwendig, um dieser Änderung Rechnung zu tragen.

Zusätzlich gibt es eine Reihe von neuen *Framework-Empfehlungen*. Einige davon sind sicherlich überraschend, zeigen allerdings auch, wie viel sich in den letzten Jahren und Monaten getan hat. Die neuen Standard-Empfehlung zu Vue 3 sind die folgenden:

- Neue Versionen von Vue Router, den Devtools und den Test Utils mit Vue-3-Support.
- Beim IDE-Support gibt es einen Wechsel von Vetur auf Volar.
- Die Build-Toolchain wechselt von Vue CLI auf Vite.
- Beim State Management gibt es einen Wechsel von Vuex auf Pinia, das besser mit dem Composition API zusammenarbeitet.
- Den Support auf der Kommandozeile für TypeScript übernimmt jetzt vue-tsc.
- VitePress als Ersatz für VuePress als Generator für statische Seiten.

Das sind zahlreiche Änderungen, die für sich genommen schon einiges Feedback in der Community ausgelöst haben. Zum Beispiel der Wechsel auf Vite an praktisch allen Fronten und der Wechsel auf Pinia für das State Management. Die Änderungen sind aber durchweg positiv und versprechen auch in Zukunft viele Verbesserungen.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z