ToiletPaper #163

jambit
WHERE INNOVATION WORKS

Trink nicht so viel: Frontend-Code reduzieren

Autor: Santo Pfingsten / Senior Software Architect / Standort Leipzig

✕ Einleitung

Stell dir vor du hast einen Hammer, der super Nägel in die Wand befördert. Das macht der so gut, dass du anfängst damit auch Schrauben in die Wand zu hauen und deinen Abwasch von hartnäckigen Resten zu befreien. Kommt dir bekannt vor? Nein? Ersetze mal Hammer mit einem Entwicklungstool deiner Wahl und Schraube durch ein Projekt, das damit umgesetzt wird.

✓ Hydration & Partial Hydration

Um mit React das ausgelieferte HTML im Browser interaktiv zu machen (vereinzelte Stellen, wie Menü, Header, Videos, etc.), wird das Feature **Hydration** genutzt. Dabei werden von Haus aus viel mehr Daten und Code an den Browser geschickt als eigentlich notwendig. Es gibt **Opt-Out**-Verfahren, mit denen man Komponenten als "statisch" markieren kann, so dass diese nur vom Server erzeugt werden (HTML), aber Daten (JSON) und Code (.js) für diese Teile nicht mit ausgeliefert werden müssen. Das Ganze nennt sich **Partial Hydration**. Dabei schließt man jedoch auch Interaktivität aus, was bei jeder Komponente erstmal die Frage aufstellt, ob das so einfach möglich ist.
Neben der Größe der ausgespielten Inhalte ist Performance auch ein wichtiges Thema. Die "Time To Interactive" ist bei React in umfangreichen Seiten sehr hoch und auch das Server-Side-Rendering dauert entsprechend lang.

➔ Beispiel aus einem größeren Projekt

Ich bin seit einiger Zeit im Umfeld einer News-Webseite, welche mit einem React-basiertem Framework auf dem Server HTML erzeugt und dieses im Browser hydriert. Leider hatten wir auf diese Technologiewahl keinen Einfluss. Das Problem dabei: Von Haus aus muss alles, was man im Server zum Rendern nutzt, auch im Browser landen, damit die Interaktivität hergestellt werden kann. Wir reden hier sowohl vom Code (.js), als auch von den Daten (JSON), da diese ja nicht nochmal vom Browser angefragt werden sollten. Wir waren zeitweise bei über **20 MB** in der ersten Anfrage (reines HTML + eingebettetes JSON).
Mit viel Arbeit (und unschönen Workarounds) haben wir über Partial Hydration die erste Anfrage auf unter **1 MB** gedrückt. Optimal ist das aber längst nicht.

➔ Alternativen

Bei der Technologiewahl sollte man immer bedenken, was man eigentlich gerade baut. Eine News-Webseite ist keine Single-Page-Application, sondern eine Multi-Page-Application. Fokus sollten ein schneller Seitenaufbau und gute SEO-Werte sein.
Um das zu erreichen, sollten via **Opt-In** nur die Features, die wirklich interaktiv sein müssen, mit Code verknüpft werden. Früher war das ganz normal. Heutzutage im SPA-Hype vergisst man das jedoch gerne mal. Ich habe meinen Hammer und erschlage damit alles. Schlechte Performance im Server wird dabei versucht, mit CDNs zu kaschieren. Bei schlechter Performance im Browser hilft das allerdings nicht.
Man muss nun auch nicht zurück in die Steinzeit zu PHP & JQuery. Es gibt durchaus moderne Alternativen. Sie sind nur schwer zu finden. Hilfreiche Suchbegriffe sind **Partial Hydration** und **Progressive Enhancement**.

+ Interessante Tools & Frameworks

- https://astro.build/ ist ein Build-Tool, welches man weiterhin mit bekannten SPA-Frameworks nutzen kann, dabei aber ein Opt-In-Verfahren zur Verfügung gestellt bekommt. Meiner Meinung nach geht das Ganze noch nicht weit genug, da ich denke, dass diese SPA-Frameworks nicht dafür gedacht und deutlich langsamer sind als notwendig.
- https://nanojsx.io/ geht mir schon mehr in die Richtung, die ich mir vorstelle: Ein neues, leichtgewichtiges Framework auf Basis der JSX-Syntax. Allerdings scheint es noch in den Kinderschuhen zu stecken.
- https://stimulus.hotwired.dev/ geht einen stark anderen Ansatz, welcher komplett außenvor lässt, wie das HTML erzeugt wird und sich lediglich damit befasst, wie das HTML später interaktiv wird. Hier fehlt dann natürlich ein Framework, um erstmal das HTML serverseitig zu erzeugen. Es lässt sich zudem gut in Kombination mit https://turbo.hotwired.dev/ verwenden, welches den Seitenwechsel in zumeist statischen Webseiten stark beschleunigt.
- Purist*innen können auch custom-elements benutzen, um vorhandenes HTML mit JavaScript Code zu verknüpfen. Den Shadow-Dom sollte man dabei jedoch möglichst vermeiden, damit die Webseite auch ohne JavaScript Inhalte sinnvoll darstellt.
- Bei React sollen demnächst Server Components kommen. Das zieht sich jedoch schon ein Weilchen und es sind noch einige Fragen offen, ob und wie das mit CDNs funktioniert. Zudem wird es in der ersten Version nicht direkt, sondern über separate Projekte wie Next.js unterstützt: https://reactjs.org/blog/2020/12/21/data-fetching-with-react-server-components.html
- https://remix.run/ sieht sehr spannend aus, auch wenn es das Problem des Partial Hydration bisher ignoriert, so gehört es dennoch in die Kategorie Progressive Enhancement.