Schon gewusst? Single Page Applications (SPAs)

Christiane Kunisch

Single Page Applications sind zweifellos leistungsstarke Webanwendungen, aber sie bringen auch eine Reihe von SEO-Fallstricken mit sich, die man verstehen und berücksichtigen sollte. Wie SPAs funktionieren und wie man sie trotzdem SEO-freundlich umsetzt, soll im Folgenden erläutert werden.

Was ist noch einmal eine Single Page Application?

Ganz einfach gesagt ist eine Single Page Application (SPA) ein Typ von Webanwendung, der darauf ausgelegt ist, auf nur einer einzigen URL zu funktionieren, anstatt mehrere verschiedene Seiten zu laden.

Eine herkömmliche Website lädt zum Beispiel beim Klick auf einen Link oder Button eine neue URL im Browser. Bei SPAs dagegen wird die gesamte Anwendung in einer einzigen HTML-Seite geladen, ohne dass sich die URL ändert. Bei einer Interaktion werden die Inhalte also dynamisch nachgeladen, ohne die Seite an sich neu zu laden.

Im Endeffekt ist eine SPA also eine Rich Applikation bzw. Web-App im Browser, wie man es beispielsweise von der Webversion von Netflix, Google Maps oder Facebook kennt.

Welche Vorteile hat eine SPA?

Eine SPA ermöglicht prinzipiell eine flüssigere und schnellere User-Experience, da nur die Teile der Seite aktualisiert werden, die sich tatsächlich ändern, anstatt die gesamte Seite neu zu rendern. Sie sind demnach besonders gut für Anwendungen geeignet, die viele Benutzerinteraktionen und Aktualisierungen des Inhalts erfordern, wie zum Beispiel Social-Media-Plattformen, die ein kontinuierliches Updaten des Feeds benötigen.

Grundsätzliche Vorteile im Überblick

- **Schnelles „snappy" User-Interface:** Da SPAs nur einmal geladen werden und anschließend nur Teile der Seite aktualisiert werden müssen, reduzieren sie die Ladezeiten und bieten deshalb in der Regel eine performante User-Experience. Die Reaktionszeiten auf User-Input, wie Klicken, Scrollen oder Eingaben, sind demnach kürzer und können meist schneller verarbeitet werden.
- **Weniger Serveranfragen und Datentransfer:** Da SPAs Daten nach Bedarf und nicht „pauschal" laden, können sie die Anzahl der Server-Requests reduzieren. Dadurch werden insgesamt die Netzwerklast und der Server-Load verringert und Bandbreite wird gespart.
- **Nutzung von Browser-Funktionen:** SPAs können Browser-Funktionen wie Caching, Local Storage und Offline-Zugriff besser nutzen, um eine reibungslose Nutzung auch bei schlechter Internetverbindung zu überbrücken.

Schnelle Lade- und Reaktionszeiten sind natürlich auch auf ganz regulären Mulitpage Applications zu erzielen – ebenso kann man eine SPA so „schlecht" bauen, dass sie langsam lädt.

Beliebte Technologien zur SPA-Entwicklung

Die meisten ahnen es an dieser Stelle – bei Dingen, die dynamisch nachladen, geht natürlich

DER AUTOR

Christiane (Chrissy) Kunisch ist Expert SEO Consultant bei diva-e am Standort München. Dort beschäftigt sie sich am liebsten mit Technical SEO und kümmert sich um die Trainee-Ausbildung neuer SEO-Talente.

Foto: Mykola Lishchyshyn / gettyimages.de

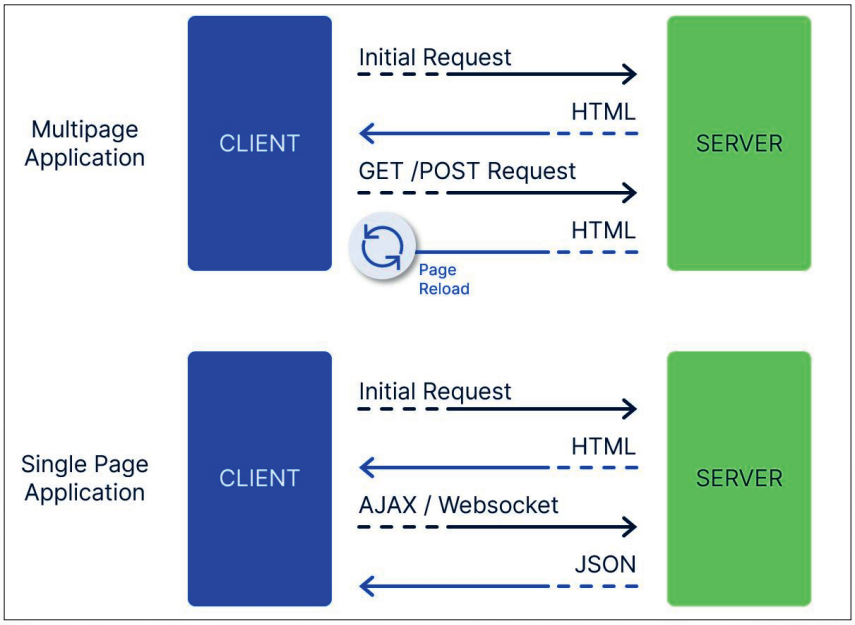

Abb. 1: Prinzip der Requests – Mulitpage vs. Single Page Application

kein Weg an JavaScript vorbei. Deshalb basieren SPAs selbstverständlich auf JavaScript. Die drei beliebtesten und gängigsten Frameworks, die für SPAs eingesetzt werden, sehen Sie unten in einem kurzen „oberflächlichen" Überblick.

Was ist überhaupt (URL-) Routing?

Grundsätzlich müssen bei der Datenübertragung Pakete von A nach B geschickt werden. Das Routing ist der Fahrplan, der festlegt, über welchen Pfad ein Paket an sein Ziel kommt. Im Prinzip könnte man es mit einer Telefonnummer mit Durchwahlen vergleichen. Dass ein Anruf in der Firmenzentrale rauskommt, ist das, was das Domain Name System (DNS) regelt: Die angefragte IP-Adresse wird einer Domain zugeordnet, auf die ihr geleitet werdet. Wollt ihr allerdings bei einem bestimmten Mitarbeiter anrufen, muss die Firma selbst in der Telefonanlage festlegen, welche Durchwahl welchem Mitarbeiter zugeordnet wird. Heißt, welcher eingegebene URL-Pfad auf welchen Inhalt führt, müsst ihr auf eurem Server oder in eurer Applikation selbst festlegen. Die Routes sind also alles, was über die Domain selbst „hinausgeht".

Durch das URL-Routing können also Regeln festgelegt werden, welche Inputs aus der aufgerufenen URL zu welchem „Ergebnis" führen. Wird also eine URL angefragt, wird geprüft, ob eine Route definiert ist, die hier „greift". Die Anfrage wird entsprechend weiterverarbeitet oder ein 404-Fehler wird zurückgegeben, wenn es beispielsweise kein Match gab.

Das klassische Beispiel wäre ein Apache-Server, der beim Aufruf eines Folders prüft, ob es darin eine index.html-Datei gibt und diese standardmäßig als „Startseite" für das Verzeichnis öffnet. Das Routing auf dem Apache-Server kann entsprechend konfiguriert werden, dass dieses Verhalten überschrieben oder individualisiert wird. So kann beispielsweise als Fallback

React ist eine Open-Source-JavaScript-Bibliothek, die von Facebook (Meta) entwickelt wurde und sich auf die Erstellung von User-Interfaces konzentriert. React wurde erstmals für den Newsfeed von Facebook und für Instagram eingesetzt. Es verwendet eine virtuelle DOM-Technik, um Änderungen effizient zu verwalten. React ist im Vergleich eine eher „schlanke" Bibliothek. Daher benötigt man möglicherweise zusätzliche Bibliotheken für Routing und Zustandsverwaltung (das heißt Plug-ins oder zusätzliche Selbstentwicklung nötig).

Angular ist ein von Google entwickeltes Open-Source-Framework, das als Basis die Sprache TypeScript nutzt. Es ist auf die Entwicklung von SPAs ausgerichtet. Angular bringt im Gegensatz zu React bereits Werkzeuge für beispielsweise Routing und Zustandsverwaltung mit. Es generiert besonders gut strukturierten Code und ist auf gute Testbarkeit ausgelegt, kann anfangs aufgrund seiner Eigenheiten für Entwickler allerdings komplexer sein. Für kleinere Projekte ist es gegebenenfalls überdimensioniert.

Vue.js ist das dritte unter den beliebtesten Open-Source-JavaScript-Frameworks für die Entwicklung von SPAs. In diesem Fall kein Produkt eines Tech-Giganten, sondern eines Softwareentwicklers namens Evan You. Der Fokus liegt hier auf einer einfachen Integration und Anwendung. Es basiert ebenfalls auf TypeScript und hat einen komponentenbasierten Ansatz. Man könnte sagen, es bietet einen Mittelweg zwischen dem Funktionsumfang von React und Angular.

TypeScript ist eine Skriptsprache, die quasi eine etwas strikte Variante von JavaScript ist. Sie stammt aus dem Haus Microsoft. JavaScript an sich lässt viele Freiheiten, die dann zu schwer lesbarem Code oder Legacy-Problemen führen können – das vermeidet TS zum Beispiel durch die Anforderung, für jede Variable einen „Type" zu vergeben.

GOOD TO KNOW: VIRTUELLER DOM

Damit React performant die Website im Browser darstellen kann, wird mit einem Algorithmus das Rendern optimiert (also der Prozess, in dem die Website grafisch im Browser angezeigt wird). Während dieses Prozesses wird bei jeder Art von Website das Domain Object Model (DOM) – also quasi der Bauplan, der aus HTML und CSS vorskizziert wurde – für den User optisch dargestellt. Bei jeder Änderung (zum Beispiel durch nachträgliches JavaScript, das die Darstellung ändert) muss dieser Bauplan neu arrangiert werden – das benötigt Zeit. Ideal wäre es also, diese Veränderungen so gering wie möglich zu halten. Genau hier setzt React an: Nur die Teile werden neu gerendert, die sich ändern, der Rest bleibt, wie er ist. Wie geht das? React greift auf das aktuelle DOM zu und erstellt daraus ein virtuelles DOM – also eine Art Kopie des aktuellen Bauplans, die dann im Browser-Cache abgelegt wird. Wenn eine Änderung ansteht, nimmt React diese Kopie zum Abgleich und initiiert nur für Bereiche, die sich ändern sollen, ein neues Rendering.

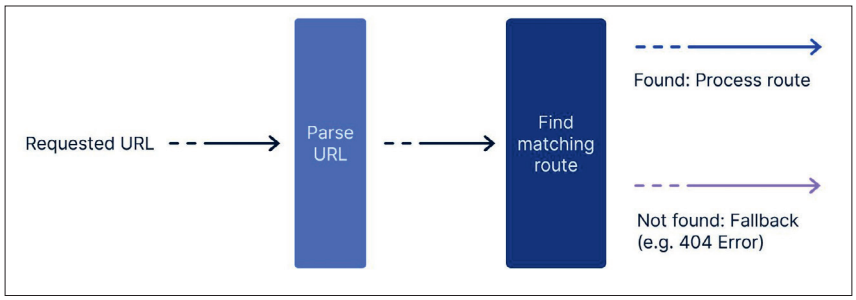

Abb. 2: URL-Routing – Prinzip

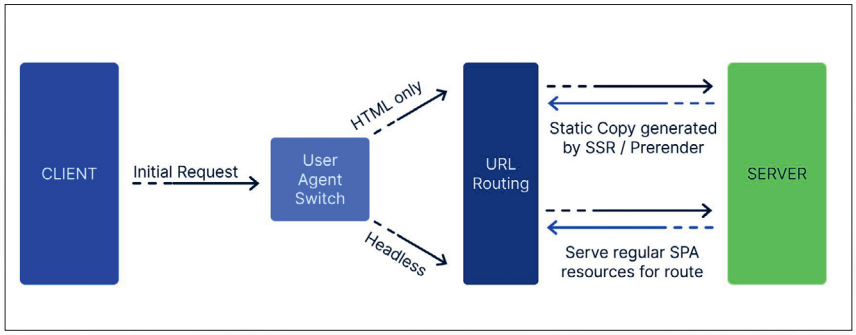

Abb. 3: User-Agent-Switch für Server-Side-Rendering – Prinzip

definiert werden, dass, falls keine index.html in einem Ordner vorliegt, ersatzweise eine Directory-Liste mit allen Files, die in diesem Ordner liegen, im Browser angezeigt wird.

Keine URLs, kein SEO: Warum ist das Routing ein kritischer Faktor bei SPAs?

URLs sind die „unique Identifier", anhand derer Inhalte im Google-Index gespeichert und als Ergebnis für eine Suchanfrage ausgespielt werden können. In einer klassischen SPA benötigt man per se erst einmal keine Routes. Es passiert ja alles auf einer einzelnen Seite. Es werden nur verschiedene „Ansichten" derselben Seite generiert. Sobald man allerdings die Möglichkeit haben möchte, auf Inhalte „von außen" zu verlinken – geschweige denn Inhalte SEO-tauglich zu machen –, ist das Routing eines der essenziellen Themen, die es zu regeln gilt. Es gibt ja sonst gar keine verschiedenen URLs.

Der initiale Request führt also nach DNS-Zuordnung dazu, dass die Applikation gestartet wird. Ab dann hat der Webserver nichts mehr mit dem Routing zu tun, sondern das Regelwerk für die Routes wird einzig und allein vom JavaScript der SPA geregelt. Das heißt zum einen, dass die Routes alle in der SPA definiert werden müssen (je nach Framework mit bestehenden Modulen oder durch Entwicklung entsprechender Komponenten). Beim Entwickeln der SPA muss ebenfalls berücksichtigt werden, die angezeigte URL im Browser optisch zu manipulieren, da die Seite ja nicht per HTTP-Request neu geladen wird und sich somit die URL nicht automatisch sichtbar ändert.

(Bei einer Mulitpage Application mit einem CMS-System wie WordPress wird zwar auch das Routing vom Webserver als Aufgabe quasi ans CMS übergeben, es wird allerdings dann nicht auf JavaScript-Basis, sondern im Fall von WordPress zum Beispiel mit PHP umgesetzt.)

Wenn die URLs entsprechend konfiguriert sind, heißt das zum anderen aber trotzdem, dass nur ein Headless-Fullstack-Crawler bzw. -Browser – also ein Client, der JavaScript ausführen kann – diese URLs überhaupt auch finden und aufrufen kann. Für den User kein Problem – für Suchmaschinen durchaus ein Problem: Selbst Google, die fortgeschrittenste Suchmaschine in Sachen Headless Crawler, kann nur bedingt Seiten mit JavaScript-Execution crawlen. Es ist schlichtweg zu ressourcenaufwendig und somit zu teuer, das gesamte Web regelmäßig in dieser „Ausführlichkeit" zu crawlen (Aufwand eines „HTML only"-Abrufs: unter eine Sekunde, mit JS-Rendering bis zu fünf Sekunden pro Seite).

SPAs trotzdem SEO-gerecht servieren: Server-Side-Rendering

Der einzige Weg, sicherzustellen, dass eine SPA auch für SEO-Zwecke funktioniert, ist die Implementierung eines Mechanismus, der die Seiten bereits auf Serverseite rendert und somit Clients, die kein JavaScript „können", extra bedient. Die Aufgabe, das JS „zusammenzubauen" und zu verstehen, übernimmt also bereits der Server. Das löst die oben beschriebene Thematik mit den durch JS geregelten URLs, aber natürlich auch das Problem, dass der statische Quelltext von React, Angular

oder Vue in nativer Form leer ist (es passiert erst etwas, nachdem JavaScript ausgeführt wurde).

Hier kommt jetzt das berühmte Server-Side-Rendering (SSR) oder Pre-Rendering ins Spiel. Idealerweise wird also eine User-Agent-Weiche erstellt, die bei einer Anfrage erkennt, welche Art von Client einen Inhalt abrufen möchte. Ist es ein regulärer User, der ohne Probleme JavaScript ausführen kann, bekommt er die Seite als „Bausatz in Einzelteilen zum Selbstzusammenbauen". Ist der User-Agent zum Beispiel der Googlebot, bekommt er die „fertig montierte" Seite ausgeliefert. Das muss natürlich auch bei der Konfiguration des Routings berücksichtigt werden.

Pre-Rendering: Es kann beispielsweise immer, wenn etwas live geht, eine statische Variante im Deployment-Prozess erzeugt und auf dem Server abgelegt werden. Denkbar wäre auch ein Cronjob, der regelmäßig die statischen Varianten erzeugt und ablegt, um möglichst aktuelle Kopien sicherzustellen. Diese statischen Varianten werden dann allen Clients ausgeliefert, die von der User-Agent-Weiche als „Sonderfall" identifiziert wurden.

Server-Side-Rendering: Genauso kann natürlich auch zur Laufzeit auf dem Server zum Beispiel ein Headless Chrome gestartet werden, der die aktuell angefragte Seite für einen Nicht-JS-Client direkt rendert und das fertige Ergebnis zurückschickt. Ebenso kann natürlich auf Dienste wie Prerender.io und Co. gesetzt werden.

React, Angular und Vue sind alle in der Lage, das „umzusetzen" (für Vue kommt hier zum Beispiel Nuxt.js ins Spiel, wer diesem Begriff schon öfter begegnet ist).

Single Page Application: Smash or Pass?

Na ja – wie immer halt (im SEO): It depends. Sollte man ein neues Projekt starten, wägt man am besten ab, was das oberste Ziel der Seite ist und wie man das am besten erreicht. Braucht es wirklich die SPA mit Angular, wo allein der Entwickler ein Vermögen kostet, oder tut es eine einfache Multipage-Lösung mit Typo3, die gegebenenfalls ein etwas günstigerer PHP-Freelancer betreuen kann (keine Wertung, nur eine Analogie für das klassische „Mit Kanonen auf Spatzen schießen"-Phänomen). Befindet man sich in einer bestehenden Anwendung, die bereits als SPA gebaut ist, aber organischen Traffic generieren soll, führt kein Weg am Rendering auf Serverseite vorbei. Letztendlich haben SPAs absolut ihre Daseinsberechtigung und das Thema SEO ist nicht immer die oberste Priorität – wichtig ist, den Zusammenhang zu verstehen und dann richtig zu entscheiden. Happy SPA-Coding!

GOOD TO KNOW: WAS IST ZUSTANDSVERWALTUNG?

HTTP ist ein zustandsloses Protokoll, bei dem der Server keine Verbindung zwischen einzelnen Browser-Anfragen herstellt. Webanwendungen erfordern jedoch normalerweise eine sogenannte Zustandsverwaltung, um Anfragen dem richtigen Anwendungszustand zuzuordnen. Was ist ein „Zustand"? Beispielsweise, dass ein User eingeloggt ist oder der Inhalt eines Warenkorbs, der sich im Verlauf der Sitzung ändert. Diese Informationen werden herkömmlicherweise entweder über „GET & POST"-Requests mittels AJAX oder Websockets („AJAX-Standleitung") übermittelt oder zum Beispiel in Cookies gespeichert und an den Server kommuniziert. Bei einer SPA wird beispielsweise im Frontend die Veränderung im Warenkorb bereits mit JavaScript „optisch" angepasst und somit für den User sichtbar, das führt aber noch zu keiner Kommunikation mit dem Server. Das muss vom Entwickler in der SPA extra definiert und getriggert werden, da keine klassischen HTTP-Requests in einer SPA nach dem initialen Aufruf mehr erfolgen.