Artur Kosch

JavaScript-SEO: der richtige Umgang mit JavaScript für eine saubere Indexierung

Das Zusammenspiel von JavaScript und SEO ist ein recht unerforschtes Gebiet. Es gibt viele Fallstricke und wichtige Maßnahmen zu beachten, um Inhalte, die durch JavaScript erzeugt werden, für Suchmaschinen bereitzustellen. Welche Schwierigkeiten Suchmaschinen in Sachen JavaScript haben, wie eine saubere Crawlbarkeit und Indexierung sichergestellt werden kann, welche Tools beim Auditing hilfreich sind und weitere wichtige Best-Practice-Hilfestellungen erläutert Artur Kosch für Sie in diesem Beitrag.

Wieso stellt JavaScript ein großes Problem für Suchmaschinen und SEO-Verantwortliche dar?

Wieso tun sich Suchmaschinen so schwer mit JavaScript? Wo liegt der Unterschied im Vergleich zum herkömmlichen Handling mit HTML und CSS? Vorab ist es wichtig zu verstehen, wie ein Web-Browser (Client) mit einem Web-Server in Bezug auf das Handling mit JavaScript interagiert. Im Folgenden und in Abbildung 1 wird dieser Ablauf vereinfacht dargestellt:

1. Der Browser stellt eine GET-Anfrage an den Server.
2. Der Server führt das PHP-Script aus (z. B. beim Einsatz eines CMS).
3. Der Server gibt den HTML-Quellcode an den Browser zurück.
4. Der Browser führt das JavaScript aus.

Im vierten Schritt ist zu sehen, dass JavaScript nicht vom Server, sondern vom Browser (Client) ausgeführt wird. Genau hier liegt das große Problem! Damit Suchmaschinen Inhalte, die durch JavaScript ausgeführt werden, crawlen können, müssen diese die Arbeit eines Browsers übernehmen, inklusive aller nötigen Technologien und Ressourcen, wie einer Rendering-Engine. Diese benötigten Ressourcen, Technologien und der Einsatz einer Rendering-Engine stellen einen großen Mehraufwand für Suchmaschinen dar. Auch Tools, die beim Auditing unterstützen, müssen über diese Technologien verfügen, um valide Daten zu liefern.

Eine weitere Herausforderung für Suchmaschinen beim Crawlen und Indexieren von JavaScript-Inhalten liegt darin, dass nicht der Crawler (bei Google der Googlebot) für das Rendering von Webseiten verantwortlich ist, sondern der Indexer (bei Google Caffeine). Um das Problem zu erkennen, ist wichtig zu verstehen, wie der Googlebot und Caffeine im Zusammenspiel crawlen, rendern und indexieren.

Vereinfachter Ablauf beim Crawlen von HTML-Inhalten:

1. Googlebot lädt HTML-Dateien herunter.
2. Links werden aus dem Quellcode entnommen und weitere URLs werden gecrawlt.
3. CSS-Dateien werden runtergeladen.
4. Googlebot sendet alle Ressourcen zu Caffeine.
5. Caffeine indexiert ggf. die Inhalte.

Das Zusammenspiel zwischen Googlebot und Caffeine ist bei diesem Ablauf simpel. Nachdem der Googlebot alle Ressourcen erfasste und crawlen konnte, werden diese an Caffeine gesendet. Caffeine übernimmt nun das Rendering und kann die Inhalte ggf. indexieren. Beim Einsatz von JavaScript sieht es etwas anders aus.

Vereinfachter Ablauf beim Crawlen von HTML-Inhalten:

1. Googlebot lädt HTML-Dateien herunter.
2. CSS- und JavaScript-Daten werden parallel runtergeladen.
3. Googlebot sendet Ressourcen zu Caffeine.

DER AUTOR

Artur Kosch ist Online-Unternehmer und SEO-Experte. Als geschäftsführender Gesellschafter von Kosch Klink Performance publiziert er regelmäßig Fachartikel und hält Vorträge auf Konferenzen rund um das Thema Online-Marketing mit dem Fokus auf SEO.

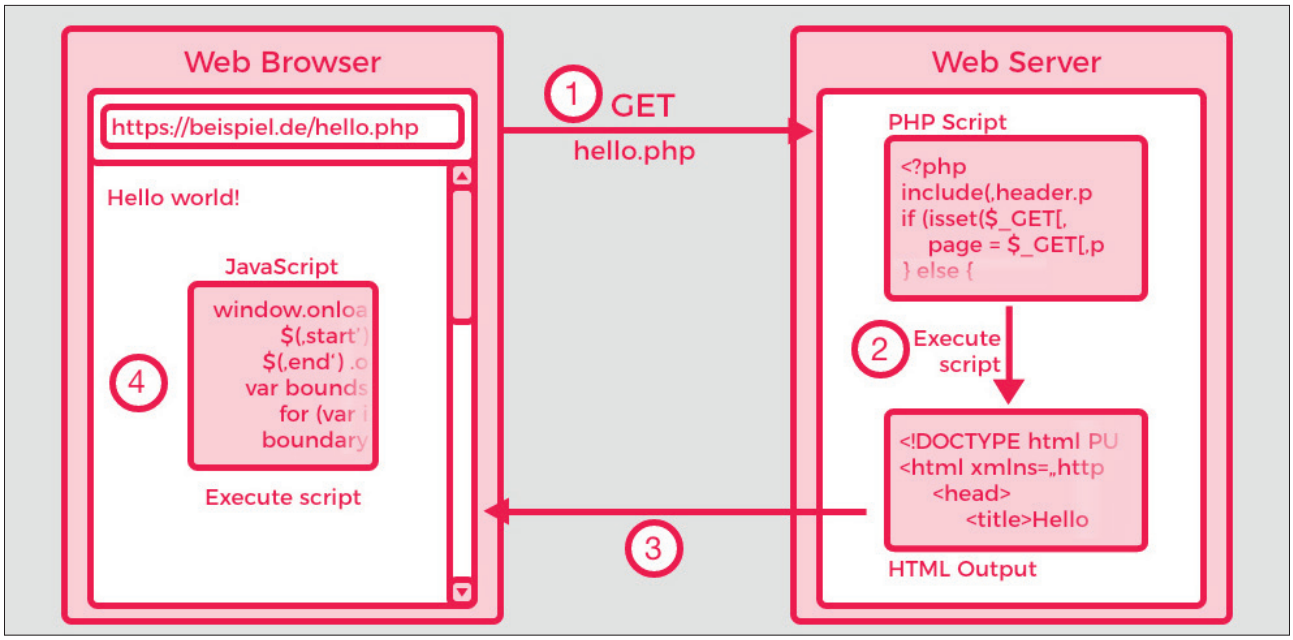

Abb.1: Handling von JavaScript zwischen Web-Browser und Web-Server

4. Caffeine rendert JavaScript, um Inhalte darzustellen, und indexiert diese ggf.
5. Erst jetzt kann Google neue Links crawlen.
6. Links werden aus DOM entnommen und können gecrawlt werden.
7. Bei der nächsten URL beginnt der Ablauf von vorne.

Da der Googlebot kein JavaScript rendert, sondern Caffeine dafür zuständig ist, ist der Googlebot von Caffeine abhängig, um die Inhalte zu erfassen und damit neue Links zu besuchen. Damit beginnt der Ablauf beim Crawlen, Rendern und Indexieren für jede URL von vorne – eine weitere Herausforderung, die nicht nur Google meistern muss, sondern auch SEO-Verantwortliche bei dem Umgang mit JavaScript immer im Hinterkopf behalten müssen.

Welche Crawler sind in der Lage, JavaScript zu rendern?

Google setzt bereits eine Rendering-Engine ein, um JavaScript zu rendern. Wie sieht es aber mit anderen Suchmaschinen aus? Um diese Frage zu beantworten, hat der geschätzte Kollege Bartosz Góralwicz ein interessantes Experiment durchgeführt. Das Ziel des Experimentes war es herauszufinden, wie unterschiedliche Suchmaschinen mit welchen JavaScript-Frameworks wie umgehen. Interessant zu sehen ist, dass bis auf Google keine anderen Suchmaschinen in der Lage sind, JavaScript zu rendern. Google ist bis dato auch der einzige Suchmaschinen-Konzern, der Informationen zum Rendering von Webseiten publiziert hat.

Laut Gerüchten soll die Suchmaschine von Microsoft „Bing“ lediglich bei großen Webseiten und Brands JavaScript-Rendering anwenden. Dieses Gerücht konnte bis dato aber nicht bestätigt werden.

Abgesehen von Suchmaschinen-Konzernen wie Bing, Yahoo, Yandex etc. ist auch bekannt, dass Crawler von Facebook, Twitter, LinkedIn oder Xing aktuell auch kein JavaScript rendern. Deswegen ist derzeit lediglich der Blick auf Google in Sachen JavaScript und SEO möglich.

Welche Technologien nutzt Google, um Webseiten zu rendern?

Erst im August 2017 wurden von Google detailliertere Einblicke in die Technologie beim Rendern von Webseiten veröffentlicht. Vorher war über das Rendering kaum etwas bekannt und lediglich die Google Search Console diente als valide Quelle für Debugging und Testing. Folgende Punkte sollten daher beachtet werden im Zusammenspiel mit JavaScript und SEO:

- Google nutzt zum Rendern von Webseiten den Web Rendering Service (WRS). Dieser basiert auf Chrome 41. Diese Version von Chrome stammt aus dem Jahr 2015!
- Keine Unterstützung des Transfer-Protokolls HTTP/2. Sollte aber kein Problem darstellen, da HTTP/2 abwärtskompatibel ist.
- Keine Unterstützung für Web-Socket-Protokoll, IndexedDB und WebSQL-Interfaces zur Datenhaltung im Browser.
- Inhalte, die erst nach Zustimmung des Users angezeigt werden, können ebenfalls nicht erfasst werden.
- Inhalte des Local Storages sowie Session Storages werden gelöscht und beim Aufruf einer neuen URL werden keine HTTP-Cookies weitergegeben.

Der wichtigste Punkt ist hier, dass Google beim Rendern von Webseiten Technologien von Chrome 41 nutzt, einem bereits drei Jahre alten Standard. Deswegen sollte beachtet werden, dass Google lediglich Web-Features aus dem Jahr 2015 unterstützt. Welche Features

genau genutzt werden können und weitere Einblicke in Chrome 41 bieten die Plattformen „caniuse.com“ und „Chrome Platform Status“ von Google selbst.

Auditing von JavaScript-Webseiten und hilfreiche Tools

Betrachtung des DOM und nicht des Quellcodes

Bei einem Blick auf den Quellcode einer JavaScript-basierten Webseite wird schnell deutlich, dass die vom Browser angezeigten Inhalte im Quellcode nicht zu sehen sind. Das liegt daran, dass kein JavaScript ausgeführt wurde, also der DOM der Webseite nicht gerendert wurde. Erst nachdem JavaScript ausgeführt wurde, werden bei korrekter Implementierung alle Inhalte auch im Quellcode sichtbar.

In der linken Spalte in Abbildung 2 ist der Quellcode einer JavaScript-basierten Webseite zu sehen. Hier ist gut zu erkennen, dass im Quellcode keinerlei Inhalte zu sehen sind. In der rechten Spalte wurde JavaScript bereits ausgeführt, daher sind hier alle Inhalte, die im Browser zu sehen sind auch im Body-Bereich des gerenderten Codes vorhanden.

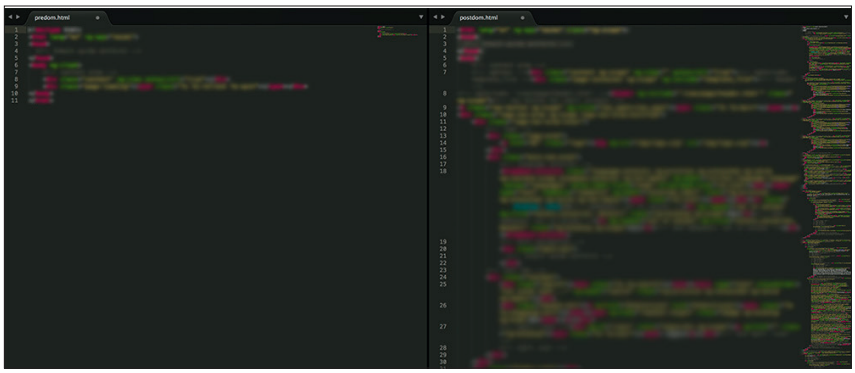

Abb. 2: Der Quellcode einer Rich-JavaScript-Webseite im Vergleich zum DOM

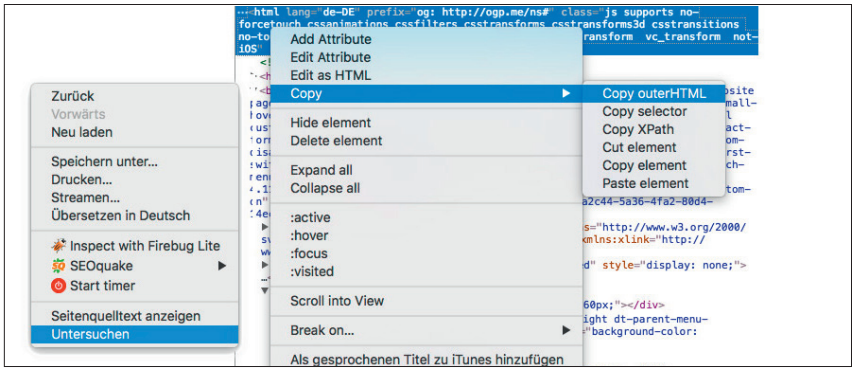

Abb. 3: Gerenderten Code einsehen und analysieren mit den Google Developer-Tools

Chrome Developer-Tool zur Betrachtung des gerenderten Codes

Um den gerenderten Code einer Rich-JavaScript-Webseite zu betrachten, sollte das Chrome Developer-Tool genutzt werden. Dazu wird folgender Ablauf empfohlen (Abbildung 3 dient als weitere Hilfestellung):

1. In einen „leeren Bereich“ der Webseite rechts klicken und „Untersuchen“ auswählen.
2. Um den gesamten Inhalt zu erhalten, den HTML-Tag im rechten Bereich des Browsers auswählen.
3. Mit einem Rechtsklick auf den HTML-Tag auf „Copy“ und „Copy OuterHTML“ navigieren, um den gesamten Inhalt zu kopieren.
4. Der kopierte HTML-Code kann nun in das Textprogramm eingefügt werden, um den Inhalt einzusehen und zu analysieren.

Dieser Vorgang kann auch verwendet werden, um Inhalte einer JavaScript-basierten Webseite auf OnPage-Faktoren zu überprüfen.

Chrome 41 – Googles Stand der Technologien nutzen

Wie bereits erwähnt, dient Googles eigener Browser, Chrome in der Version 41, zum Testing von JavaScript-Webseiten, da Google selbst diese Standards zum Rendern nutzt. Hierbei sollte beachtet werden, dass Features, die neuer als Chrome 41 sind, von Google (Caffeine WRS) nicht unterstützt werden. Hierbei bieten die Fehlermeldungen in der Konsole der Chrome Developer-Tools Aufschlüsse über mögliche Problemstellungen bei JavaScript und sollten behoben werden, um einen reibungslosen Ablauf beim Rendern bei Google sicherzustellen.

Fetch-and-Render-Tool der Google Search Console

Naheliegend ist es natürlich, das Fetch-and-Render-Tool der Google Search Console zu nutzen. Hierbei sollte aber darauf geachtet werden, dass dieses Tool lediglich Auskünfte darüber gibt, ob eine Webseite technisch gerendert werden kann. Timeouts und Performance-Ansprüche des Googlebots werden hierbei nicht berücksichtigt. Diese Tatsache bestätigt ein spannendes Experiment von Tomasz Rudzi im Elephante-Blog.

In diesem Experiment wurden Textinhalte über JavaScript erzeugt, die erst nach zwei Minuten auf der Webseite sichtbar wurden. Bei der Betrachtung im Fetch-and-Render-Tool der Google Search Console wurden diese Inhalte, die mit einer Verzögerung von zwei Minuten vom JavaScript erzeugt wurden, einwandfrei dargestellt. Bei der Betrachtung einer Site-Abfrage konnte aber festgestellt werden, dass Google beim Indexieren nicht so geduldig war. Die verzögerten Inhalte wurden von Google dementsprechend nicht indexiert!

TIPP: WELCHE PUNKTE SOLLTEN BEI JAVASCRIPT UND SEO BESONDERS BEACHTET WERDEN?

- **Fünf-Sekunden-Regel:** Google erstellt nach etwa fünf Sekunden einen Screenshot der Webseite. Alle wichtigen Inhalte sollten in dieser Zwischenzeit dargestellt werden.
- **Load-Event:** Alle wichtigen Inhalte, die indexiert werden sollen, müssen während des Load-Events geladen werden. Inhalte, die durch ein User-Event (z. B. onClick) geladen werden, ignoriert Google.
- **Eigenständige URLs:** Beim Aufruf einer neuen Seite muss eine neue URL mit serverseitiger Unterstützung aufgerufen werden.
- **pushState:** pushState-Fehler sollten vermieden werden, damit die Original-URL mit serverseitiger Unterstützung weitergegeben wird.
- **Hash und Hash-Bang:** Kein Hash (#) oder Hash-Bang (#!) in URLs nutzen (oftmals Standard in einigen JavaScript-Frameworks).
- **A-Href:** Links über A-Href realisieren. Auch hier beachten, dass die URLs nicht über ein User-Event eingefügt werden.
- **Canonical:** Canonical-Empfehlungen im Header-Bereich sollten ohne JavaScript-Rendering bereits im Quellcode vorhanden sein.
- **Meta-Informationen:** Da andere Crawler (Bing, Facebook, Twitter, Xing etc.) kein JavaScript rendern, ist es sinnvoll, wichtige Meta-Informationen ohne JavaScript-Rendering bereitzustellen.
- **AJAX-Schema:** Das AJAX-Crawling-Schema „_=escaped_fragment_=to“ wird von Google nicht mehr unterstützt.
- **Lazy-Loading von Bildern:** Beim Einsatz von Lazy-Loading ist es wichtig, dass die URLs der Bilder über <img src=“bild.jpg“> (noscript) oder JSON-LD dargestellt werden (schema.org/image).

Site-Abfrage in der Google-Suche

Um sicherzustellen, ob Google Inhalte, die von JavaScript erzeugt werden, indexieren kann, ist eine Site-Abfrage inkl. Ausschnitt des zu überprüfenden Textes immer zu empfehlen. Eine Site-Abfrage kann wie folgt in der Google-Suche gestartet werden: site:webseite.de „Ausschnitt des Textinhaltes, der indexiert werden soll“. Erst wenn hier festzustellen ist, dass Google die Inhalte indexiert hat, kann eine Indexierung zu 100 % sichergestellt werden.

Google-Rich-Media-Tester und Google-Mobile-Friendly-Tester

Laut eigenen Aussagen von Google (John Mueller) weist das Fetch-and-Render-Tool der Google Search Console noch einige Schwächen auf, wenn es um die Betrachtung des DOM geht. Deswegen werden der Google-Rich-Media-Tester für Desktop und der Google-Mobile-Friendly-Tester für Mobile empfohlen. Veränderungen, die durch JavaScript am DOM nachträglich vorgenommen werden, können mit dem Google-Rich-Media-Tester und dem Google-Mobile-Friendly-Tester dargestellt werden. Auch CSS-Anweisungen, die z. B. durch Resizing verursacht werden, können der Google-Rich-Media-Tester und der Google-Mobile-Friendly-Tester darstellen. Die Betrachtung des DOM, die der Google-Rich-Media-Tester aktuell bietet, ist laut eigenen Aussagen von Google auch für das Fetch-and-Render-Tool der Google Search Console geplant.

Screaming-Frog – eigene Rendering-Engine mit Vorsicht genießen

Eine Hilfestellung bietet der Screaming Frog SEO Spider. Dieser Crawler beinhaltet bereits die Funktion, JavaScript zu rendern, und erstellt ähnlich wie der Googlebot einen Screenshot der Webseite nach dem Load-Event.

Unter dem Menüpunkt „Configuration » Spider » Rendering“ kann hier der Punkt „JavaScript“ ausgewählt werden. Nach dem erfolgreichen Crawlen der Webseite kann jede URL als „Rendered Page“ betrachtet werden. Dieser Screenshot kann bereits erste Aufschlüsse darüber geben, ob es Probleme beim Rendern der Webseite gibt. Zu beachten ist, dass diese Funktion lediglich in der Paid-Version des Screaming Frog SEO Spiders verfügbar ist.

Obwohl sich der Screaming Frog SEO Spider ähnlich wie der Googlebot verhält, ist leider damit nicht zu gewährleisten, ob der Googlebot exakt so verfährt und die Inhalte so indexieren würde.

Des Weiteren ist vom Screaming Frog SEO Spider bekannt, dass dieser zwar die Rendering-Engine „Blink“ vom Chromium Project nutzt, diese sich aber von der des Googelbots (WRS) unterscheidet. Außerdem benötigt das Rendern (inkl. Timeout von fünf Sekunden) von JavaScript mit dem Screaming Frog SEO Spider große Ressourcen, was das Crawlen von JavaScript-Seiten mit einer großen Anzahl an URLs unzumutbar macht.

Searchmetrics Suite – Chrome- und PhantomJS-basiertes JavaScript-Rendering

Die Searchmetrics Suite bietet mit der Site-Optimization ein JavaScript-Crawling an, welches auf Chrome 41 und PhantomJS basiert. Damit bietet dieses Tool ein praktisches Werkzeug, um SEO-Verantwortlichen die Arbeit zu erleichtern. Bei den Crawling-Einstellungen kann bereits gewählt werden, ob der Crawler JavaScript mitberücksichtigen soll. Da Searchmetrics bei diesem Crawl JavaScript ausführt, kann die Webseite hinsichtlich OnPage-Faktoren wie jede herkömmliche Webseite betrachtet werden, was eine große Hilfe für SEO-Verantwortliche darstellt.

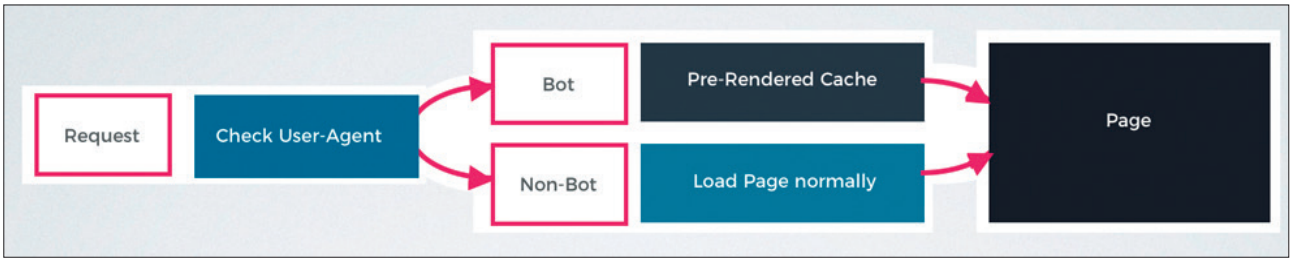

Abb.4: Einsatz von Prerendering für Rich-JavaScript-Webseiten

Prerendering – serverseitiges Rendering von JavaScript

Da andere Suchmaschinen außer Google und andere Crawler nicht in der Lage sind, JavaScript zu rendern, empfiehlt sich der Einsatz von serverseitigem Rendering bzw. Prerendering von JavaScript. Dadurch ermöglicht man den Zugriff auf Inhalte für alle Crawler und Google muss die Arbeit, die normalerweise der Browser übernimmt, nicht leisten. Des Weiteren kann mit serverseitigem Rendering sichergestellt werden, dass alle gewünschten Inhalte gecrawlt und ggf. indexiert werden.

Wie funktioniert serverseitiges Rendering für SEO und wie lässt es sich einsetzen?

Prerendering führt dazu, dass JavaScript bereits vom Server vorgerendert (Cache) wird. Dadurch werden dem Crawler bereits vorgerenderte Inhalte bereitgestellt und ein Rendering von JavaScript seitens des Crawlers ist nicht mehr nötig. Das bringt den großen Vorteil mit sich, dass ganz genau sichergestellt werden kann, welche Inhalte dem Crawler bereitgestellt werden.

Bei einer Anfrage an den Server wird überprüft, welcher User-Agent hinter einer Anfrage steckt. Wenn es sich hier um einen Bot wie z. B. den Googlebot handelt, wird diesem der bereits zwischengespeicherte (Cache), also gerenderter Code zur Verfügung gestellt (Abbildung 4). Damit erspart sich der Crawler das Rendern von JavaScript bzw. ermöglicht es Crawlern, die nicht in der Lage sind, JavaScript auszuführen, die Inhalte der Webseite zu erfassen.

Google selbst schlägt bei großen Webseiten, deren Inhalte sich in einem kurzen Zyklus oft ändern, den Einsatz von serverseitigem Rendering von JavaScript vor. Auf Google I/O 2018 hat John Mueller diese Methode, die bei Google „Dynamic Rendering“ genannt wird, als mögliche Lösung präsentiert. Hier wird zusätzlich die Empfehlung ausgesprochen, für das serverseitige Rendering gesonderte Ressourcen zu nutzen, um den Server zu entlasten.

Wie lässt sich serverseitiges Rendering für SEO am besten realisieren?

Um serverseitiges Rendering einzusetzen, gibt es die Wahl zwischen kostenpflichtigen oder Open-Source-Lösungen, die selbst implementiert werden müssen. Bei kostenpflichtigen Lösungen variiert der Preis zwischen der Anzahl der Seiten der Webseite und der Aktualität des Caches (Wie oft soll der Cache aktualisiert werden?). Bei eigenen Open-Source-Lösungen empfiehlt sich Phantom.JS, Headless Chrome in Kombination mit Puppeteer oder Renderton.

Bei der Entscheidung für oder gegen serverseitiges Rendering steht immer der Kosten-Nutzen-Faktor im Vordergrund und die Wichtigkeit für den Kanal SEO. Die Vorteile durch serverseitiges Rendering liegen aber klar auf der Hand: Eine Sicherheit, dass Inhalte korrekt an Google weitergegeben werden und andere Crawler und Dienste diese ebenfalls erfassen können.

Die Zukunft von JavaScript und SEO

Im Hinblick auf die Zukunft von JavaScript-SEO gibt es einige Punkte, die kritisch hinterfragt werden sollten. Welche Lösungen müssen gefunden werden, wenn andere Konzerne das Rendering von JavaScript ermöglichen? Jedes Unternehmen wird seine eigenen Technologien und Rendering-Engines einsetzen, um JavaScript zu rendern. Diese Vielfalt wird es SEO-Verantwortlichen deutlich erschweren, jedem System gerecht zu werden, um eine Indexierung zu garantieren.

Des Weiteren müssen Tool-Anbieter nachziehen bzw. Kosten und Mühen investieren, um das Thema JavaScript-SEO abzudecken. Lohnt sich der Aufwand überhaupt für die Anbieter? Diesbezüglich ist eine höhere Transparenz der Suchmaschinen-Konzerne nötig, damit Tool-Anbieter ihre Crawler und das Rendering-Verhalten an die Gegebenheiten und Technologien der Suchmaschinen-Konzerne anpassen können. Nur so lässt sich mit großer Wahrscheinlichkeit dem Kunden ein valides Ergebnis präsentieren.

Auditing und Analysen sind bei Rich-JavaScript-Webseiten weitaus aufwendiger und benötigen ein größeres technisches Verständnis und eine reibungslose Zusammenarbeit zwischen SEO-Verantwortlichen und Entwicklern. Dies bedeutet, ohne ein kompetentes Entwickler-Know-how bzw. ein Entwickler-Team lassen sich JavaScript und SEO nur sehr schwer verheiraten.

JavaScript-SEO bringt neue Herausforderungen mit sich, diese benötigen neue Ansätze und Lösungswege. ¶