Das Web ist wieder auf dem Weg zu alter Stärke

Autor: Tobias Huber, LumaBIT GmbH

Vor 20 Jahren waren Webanwendungen hip, vor 10 Jahren waren Native Apps der letzte Schrei und seit zwei Jahren gibt es einen neuen Ansatz von Google, der dafür sorgen soll, dass diese beiden gigantischen Mächte zusammengeführt werden. Das Interesse von Webentwicklern hat sich meistens in Grenzen gehalten, wenn es darum ging, sich in die Welt der nativen Appentwicklung einzuarbeiten. Fragen wie, „Wieso soll ich das Gleiche dreimal entwickeln für Web, iOS und Android?“ oder „Fang ich mit iOS oder mit Android an?“, waren an der Tagesordnung und keiner konnte sie wirklich beantworten.

Hier Apps, da Apps, Mobile Apps, Smartphone Apps, iOS, Android, Hybrid, Native und Webapps. Ja was ist jetzt eigentlich eine App? Zum grundsätzlichen besseren Verständnis dieses Artikels werden wir in den folgenden Zeilen von PWA sprechen, wenn es sich um eine Progressive Web App handelt, wir werden von Native App sprechen, wenn es sich um eine Smartphone App handelt, die mit nativen Methoden für iOS oder Android entwickelt wurde und wir werden es Hybrid App nennen, wenn eine App mit Hilfe von Cordova dargestellt wird. Wenn wir einfach nur App schreiben, dann ist der abstrakte Oberbegriff von PWA, Hybrid App und Native App gemeint.

In den ersten Zeilen dieses Artikels konnte ich schonmal kurz anmerken weshalb der aktuelle Stand nicht das Optimum ist und weshalb sich alle Entwickler nach einer besseren Lösung sehnen, damit sie ihre Abende nicht damit verbringen müssen sich ständig in neue Technologien einzuarbeiten, sondern stattdessen auf einer schicken Party über PWAs reden können. Das Thema PWA soll übrigens der Icebreaker 2017 sein, Quelle unbekannt ☺. Jetzt kommt das Beste. PWAs haben nicht nur Vorteile für den Entwickler sondern auch für den Endanwender. Bevor ich auf diese Vorteile genauer eingehe möchte ich aber erst noch etwas über PWAs an sich erzählen, damit wir alle eine klare Vorstellung davon haben was PWA bedeutet und was es ist.

Geschichte

„Geschichte ... vier langweilige Stunden pro Woche in der Schule, oder was das lange her ist, oder immer ohne einen passiert“. In diesem Fall hat Freundeskreis

nicht ganz Recht. Die Geschichte von PWA ist nicht alt und die Idee in keinster Weise langweilig. Ganz abstrakt gesagt kann man sagen, dass die grundsätzliche Idee, die hinter PWAs steckte, folgende war. Es galt die riesige Lücke zwischen Native Apps und Webapps zu schließen. Google hat sich dieser Aufgabe im Jahr 2015 angenommen und die einfache Annahme getroffen, dass eine App nicht zwangsweise als Native App aus dem App Store heraus installiert werden muss, damit sie auf einem Smartphone lauffähig ist, sondern dass auch ganz einfach ein Webbrowser als Container für Apps dienen kann. Schön und gut, aber hat mein Webbrowser nicht viel zu wenig native Funktionalität, um mit einer Native App mitzuhalten? Vor ein paar Jahren hätte ich genickt und DITO gesagt. Heute sieht das aber alles etwas anders aus. Öffne doch einfach mal in Deinem Smartphone den Browser und tippe die URL http://whatwebcando.today ein. In Abbildung 1 habe ich einen Screenshot hinzugefügt mit den Funktionen, die mein Browser beherrscht. Gibts doch nicht??? Steht da tatsächlich Push Messages, GeoLocation, SpeechRecognition und Offline Mode? Google hat also tatsächlich den Chrome Browser dahingehend optimiert, dass er viele mobile native Funktionen nun unterstützt. Es werden zwar noch nicht alle Funktionen unterstützt, aber es wird davon ausgegangen, dass mit der aktuellen Anzahl an unterstützer nativer Funktionalität circa 90 Prozent der Native Apps auch als PWA entwickelt werden können und das ist etwas, dass alle Webentwickler schwärmen lässt.

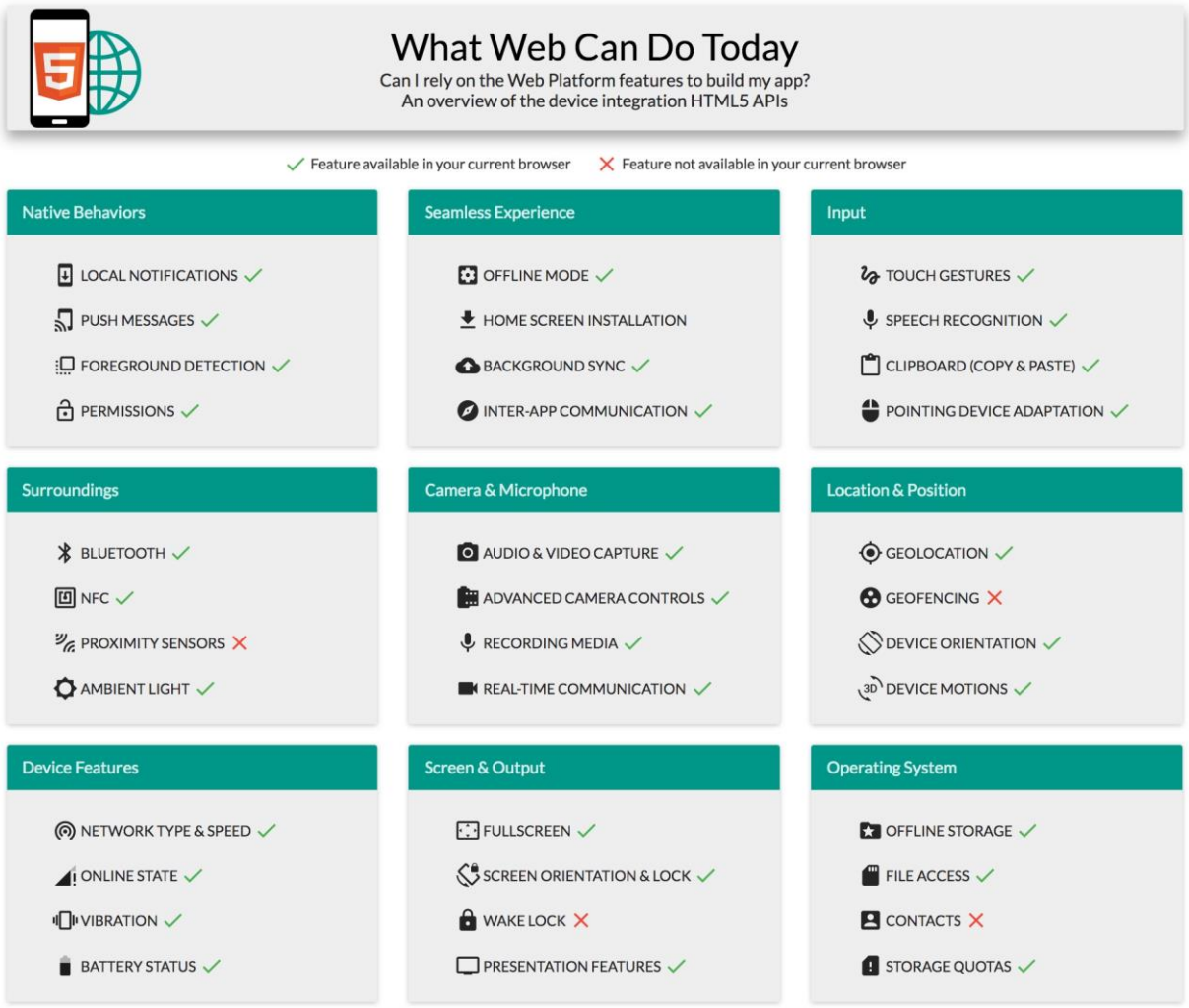

Abbildung 1: Was unterstütz mein Browser alles?

Aktuell einziger Wermutstropfen ist, dass Apple PWAs nicht von Anfang an unterstützt hat. In Cupertino hat man sich lange gegen die Idee von Google gesträubt, was durchaus verständlich ist, da man mit dem Verkauf von Apps über den firmeneigenen App Store jährlich Millionen verdient. Weshalb sollte man also etwas wie PWA unterstützen, das den eigenen App Store zum Großteil überflüssig macht? Mittlerweile hat man bei Apple die Zeichen der Zeit erkannt und sich dazu entschlossen ebenfalls PWA zu unterstützen. Man arbeitet bei Apple aktuell an der Erweiterung des Safari Browsers und des iOS, damit diese in naher Zukunft für die Idee von Google betriebsbereit sind. Mich als iPhone User freut das natürlich besonders.

Was genau sind denn jetzt Progressive Web Apps?

Jetzt gab es so viel Gerede von PWAs und ich als Endanwender habe noch immer keinen blassen Schimmer was mit PWA gemeint ist. Das ist auch gut so, denn dem Endanwender soll der Unterschied zwischen PWA und Native App überhaupt nicht groß auffallen bis auf eine Kleinigkeit, die eine enorme Auswirkung hat. Die App wird zukünftig nicht mehr über den Google Play Store auf dem Handy installiert sondern beim Besuchen der Webapp wird dem Endanwender die Möglichkeit geboten die App direkt vom Browser aus zu installieren. Nach der Installation erscheint das typische Icon zum Starten der App auf dem Desktop. Der Installations- und Startprozess wird in Abbildung 2 auch nochmal visuell dargestellt.

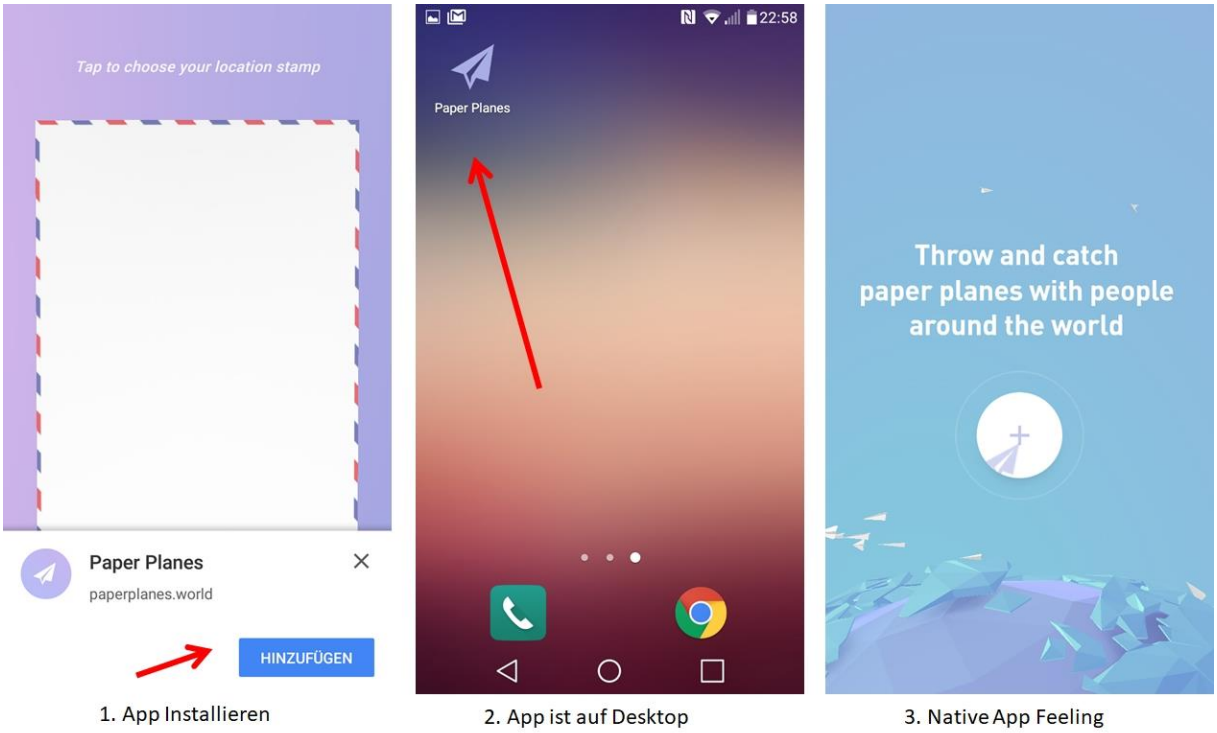

Abbildung 2: Installation einer PWA

Diese einfache Installation hat für uns als Endanwender den Vorteil, dass wir uns nicht durch Millionen von Apps und Bewertungen in den App Stores durchklicken müssen, um diese eine App zu finden, die wir haben wollen. Stattdessen sehen wir eine coole Webapp und können uns genau diese mit einem einzigen Klick auf unserem Smartphone installieren. Für das Deinstallieren der App wird der Standarddeinstallationsprozess des Smartphones verwendet.

Es sollten nicht nur die Entwickler und die Endanwender bei dem Begriff PWA aufhorchen sondern auch die Marketingabteilung. Oh ja. Bei PWA können alle drei Parteien profitieren was unter anderem durch eine interessante Aussage des Google

Manager Gabor Cselle untermautert wird. In einem Blog von 2012 hat er behauptet, dass mit jedem Schritt, den der Endanwender machen muss bevor er einen Nutzern von der App hat, circa 20 Prozent der Endanwender und damit auch der potenziellen Kunden abspringen. Mit Hilfe von PWA fallen da zumindest schonmal die Installationsschritte weg. Abbildung 3 zeigt den Userverlust einer sozialen App wie Instagram anhand einer Statistik auf.

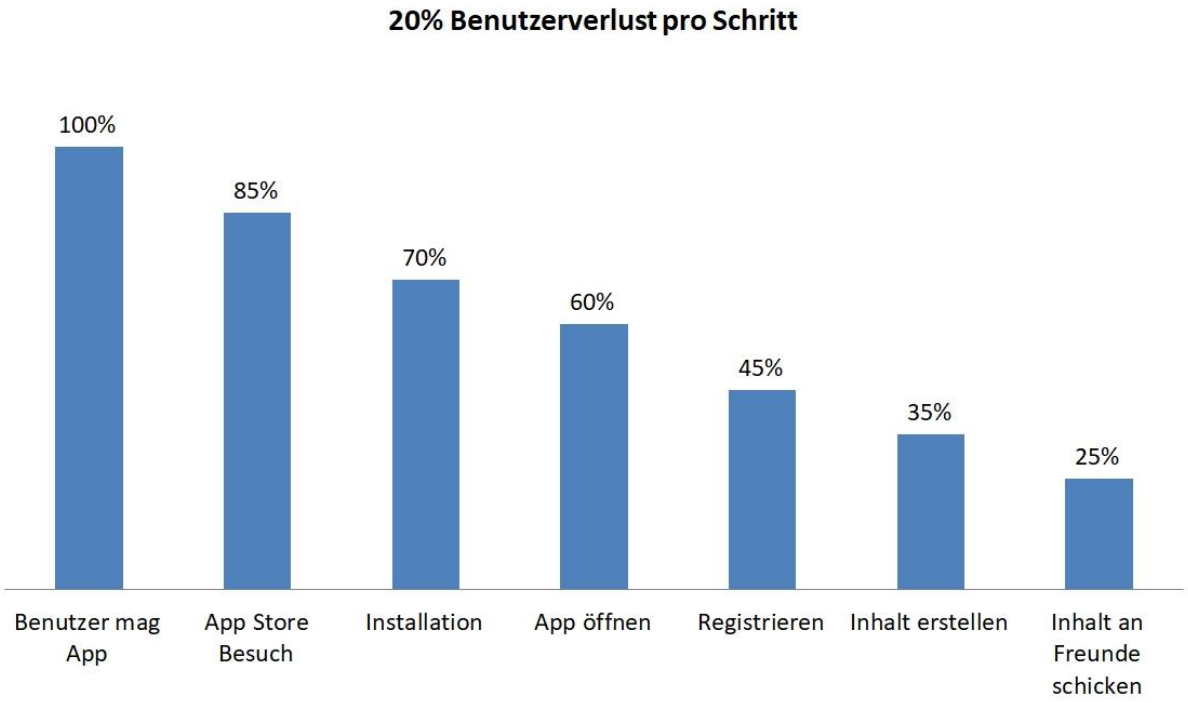

Abbildung 3: Userverlust bei Erstnutzern einer App

Webapp vs. PWA vs. Native App: Fight!

Würde mich jemand fragen was der Unterschied zwischen Webapps und PWAs sind würde ich sagen, dass eine PWA eine Webapp ist, die Responsive Design, Push Notifications und Offline Support bietet. Zusätzlich wird noch die Möglichkeit geboten, dass man eine PWA auf dem Smartphone recht einfach installieren kann, was mit einer Webapp so nicht möglich ist.

Der Unterschied zwischen PWA und Native Apps ist, dass PWAs noch nicht 100 Prozent der nativen Funktionalität beherrschen, was dazu führt, dass die ein- oder andere App, die nach besonderer Funktionalität schreit, nativ entwickelt werden muss. Diese Lücke wird in den nächsten Monaten kleiner werden und spätestens dann muss sich jeder Anbieter bei einer Neuentwicklung mit dem Gedanken beschäftigen, ob er noch Native App gegenüber PWA verwenden möchte und

welchen Vorteil eine native Entwicklung mitbringt. Wieviel mit PWA schon möglich ist sieht man daran, dass große Unternehmen, wie Twitter, ihre Anwendung bereits als PWA anbieten. Für den Anbieter ist der Kostenfaktor auch ein großer Vorteil einer PWA gegenüber einer Native App. Es gibt bei einer PWA nur einen Code zu entwickeln und zu pflegen, der sowohl für das Web als auch für iOS und für Android verwendet werden kann.

Folgende Tabelle stellt nochmals die Möglichkeiten der einzelnen Entwicklungsvarianten dar. Da sich bei PWAs aktuell die Unterstützung nativer mobiler Funktionalität rasend schnell erweitert werde ich die fehlenden Funktionen nicht explizit in der Liste angeben, da diese zum Zeitpunkt der Veröffentlichung dieses Artikels wahrscheinlich schon wieder veraltet wären.

	PWA	Hybrid	Native	Webapp
Push Notification	+	+	+	-
Offline Support	+	+	+	-
Installation über Browser	+	-	-	-
Installation über App Store	-	+	+	-
Regelmäßige App Updates nicht notwendig	+	-	-	-
Abdeckung von 100% nativer Funktionalität	-	+	+	-
iOS Support	-	+	+	-
Android Support	+	+	+	-

Zu einigen Punkte aus der Tabelle möchte ich nochmals kurz etwas erwähnen. Installation über App Store ist bei PWA aktuell nicht möglich. Wie man damit umgeht ist noch nicht ganz klar. Viele sehen den App Store auch als Werbeplattform und gerade für große Firmen ist es Pflicht, dass ihre App eben auch im App Store vorhanden ist. Auch wenn es Twitter nun als PWA gibt, so kann es sich Twitter nicht leisten die Entwicklung der Native App einzustellen, weil Twitter dann nicht mehr im App Store wäre.

Regelmäßige App Updates sind bei PWA nicht mehr notwendig, da es sich unter der Haube noch immer um eine Webanwendung handelt. Die PWA erkennt automatisch,

dass es eine neue Version gibt, die auch gleich verwendet wird. Dieser Updateworkflow ist in dieser Art eben auch möglich, weil PWAs nur einen Bruchteil an Speicherplatz einer nativen App oder gar einer Hybrid App benötigen.

Fehlender iOS Support bedeutet, dass beim Aufruf einer PWA mit einem iOS Browser die PWA ganz normal als Webapp dem Endanwender dargestellt wird. Es kommt hier zu keiner Fehlermeldung.

Alles dreht sich um das Caching

Für den Endanwender ist ein großer Vorteil einer PWA gegenüber einer Webapp der, dass eine PWA wesentlich schneller geladen wird, weil Google für das Caching gute Lösungen gefunden hat. Zum einen gibt es eine App Shell, bei der es sich um statischen Inhalt, bestehend aus HTML, CSS, Bildern und Javascript, handelt, der gecached wird. Dieses Cachen macht Sinn, weil sich dieser Inhalt nur selten ändert. Das Cachen der App Shell sorgt auch dafür, dass dem Endanwender eine PWA schnell angezeigt werden kann, auch wenn nur eine 3G Verbindung da ist. Abbildung 4 deutet die Idee einer App Shell nochmal an. Wenn die PWA dann zum ersten Mal aufgerufen wird, wird gecached. Beim zweiten Aufruf wird die App Shell schon aus dem Cache geladen und der Content wird asynchron nachgezogen.

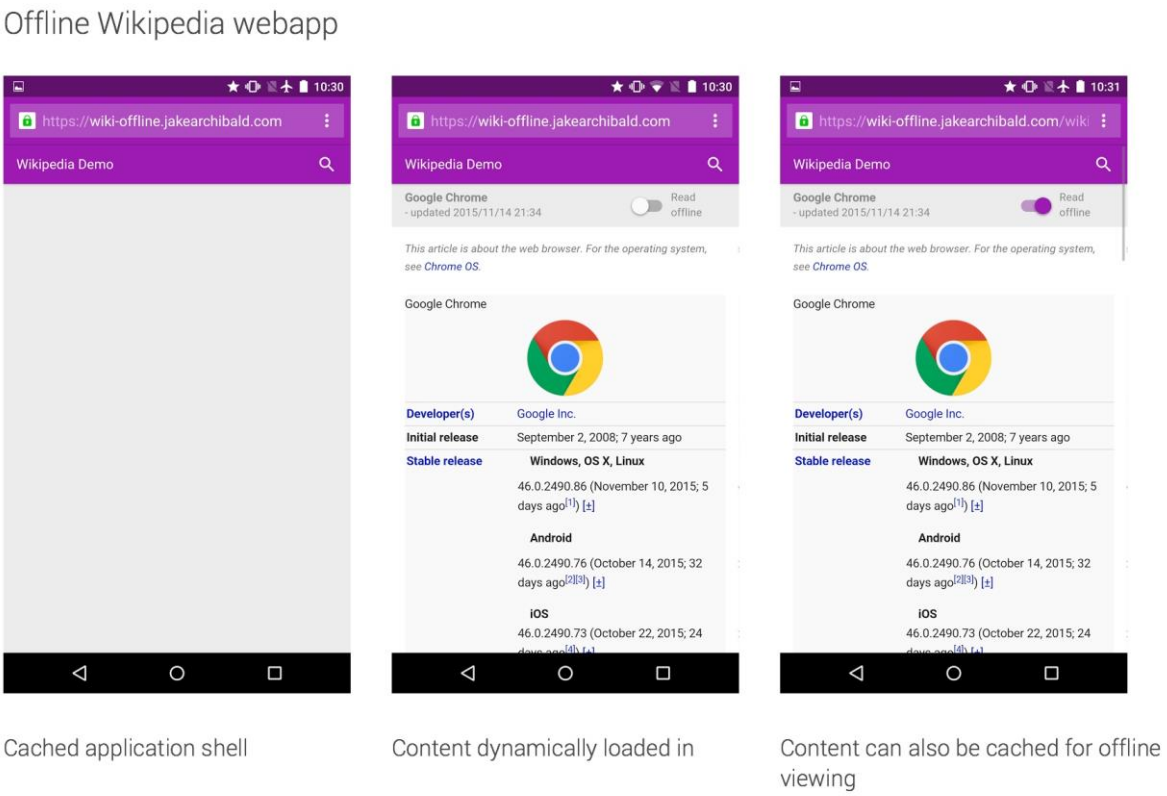

Abbildung 4: Quelle Google – App Shell

Bei PWAs ist es allerdings nicht nur möglich statischen Content zu cachen. Es ist auch möglich die komplette Antwort von GET-Requests zu cachen. Dies führt dazu, dass je länger ein Nutzer sich auf einer Seite aufhält, desto performanter wird die Seite, weil der Cache immer weiter gefüllt wird. Im Normalfall bedeutet dies dann auch, dass selbst bei einer fehlenden Internetverbindung etwas angezeigt wird. Bei einer Newsseite wäre dies dann eben die Ausgabe mit den Artikeln vom Vortag anstatt einer 404 Fehlermeldung.

Das Geheimnis des hochwertigen Caching bei PWA liegt in den Service Workern, die im Hintergrund arbeiten. Ein Service Worker ist eine Javascript Datei, die in der Anwendung läuft, aber in einem separaten Prozess. Dieser separate Prozess hat den Vorteil, dass der Service Worker weiterläuft, auch wenn die Anwendung beendet wird. Dieser Service Worker, der die ganze Zeit im Hintergrund läuft, kann frische Daten asynchron nachziehen oder aber auch auf Push Messages, die vom Server geschickt werden, reagieren. Da es sich bei diesem Artikel weniger um einen entwicklerspezifischen Artikel handelt, möchte ich hier auch mit der Beschreibung

von Service Workern aufhören. Abbildung 5 zeigt das einfache Basisprinzip eines Service Workers ganz gut, in dem dargestellt wird, dass ein Service Worker als Proxy zwischen Web Server und PWA dient.

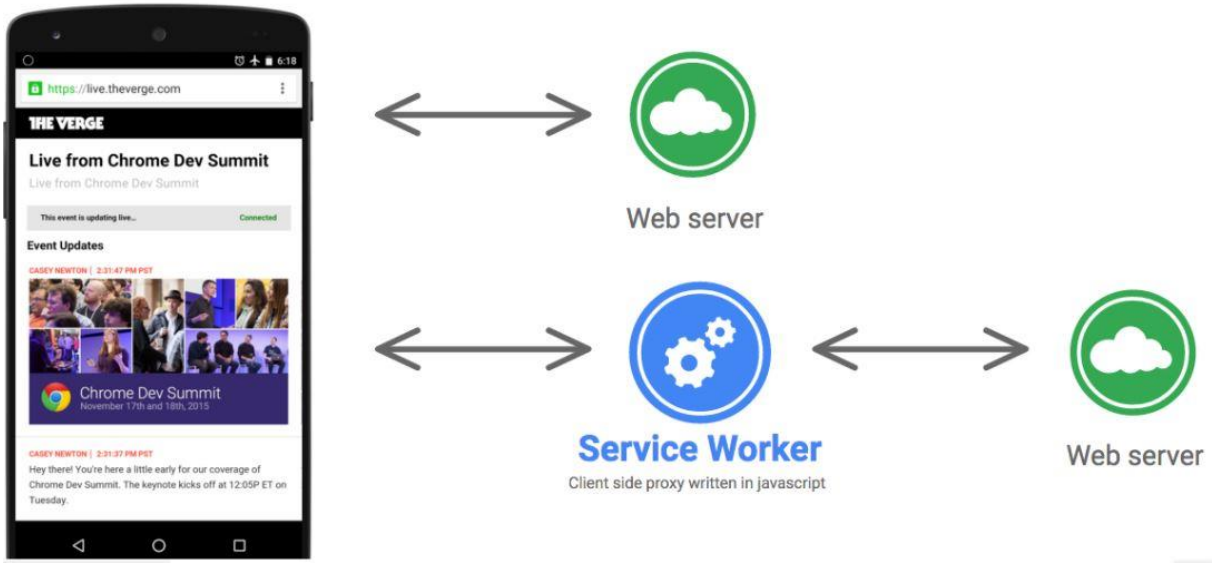

Abbildung 5: Quelle Google – Service Worker

Zusammenfassung

Ich war vor einigen Jahren von Hybrid Apps begeistert und bin nun von PWAs begeistert, weil es Schritte in die richtige Richtung sind. Mehr erreichen mit weniger Arbeit. Eine Frage, die sich mir aktuell allerdings immer wieder stellt, ist, weshalb Google mit PWA etwas vorantreibt, das bei Erfolg den eigenen App Store überflüssig machen könnte. Immerhin verdient Google mit dem App Store sehr viel Geld. Wenn PWAs allgegenwärtig sind, kann rein theoretisch jeder seinen eigenen App Store aufbauen, weil die Installation einer PWA nicht an Android oder iOS gebunden ist, sondern lediglich ein kompatibler Browser notwendig ist.

Kontakt:

Tobias Huber

huber@lumabit.de