Magdalena Mues & Matthias Böhm

Progressive Web-Apps – ein Leitfaden

Eine progressive Web-App (PWA) ist eine Webseite, die zusätzliche Funktionen basierend auf der Geräteunterstützung bereitstellt. Zu diesen Funktionen gehören Offlinefähigkeit, Push-Benachrichtigungen, ein nahezu natives Erscheinungsbild, hohe Geschwindigkeit sowie lokales Caching von Ressourcen.
Progressive Web-Apps werden in der Entwicklung mobiler Anwendungen mit Web-Technologien verwendet. Der Begriff PWA bezeichnet eine Reihe von Techniken, die das Ziel haben, ein besseres Erlebnis für webbasierte Anwendungen zu schaffen.

Was sind progressive Web-Apps?

Progressive Web-Apps haben bereits jetzt den unangefochtenen Ruf, die unverzichtbaren Web-Anwendungen der Zukunft zu sein. Dabei könnten sie sogar das mobile Web vorantreiben, die Parität zu Web- und nativen Anwendungen herstellen und mobilen Entwicklern helfen, mehr Benutzer über die Grenzen der App-Stores hinaus zu erreichen.

Die PWA-Technik wurde ursprünglich von Google im Jahr 2015 eingeführt und bietet sowohl dem Entwickler als auch den Nutzern viele Vorteile.

Mit einem Web-Stack können Entwickler Anwendungen erstellen. Das ist wesentlich einfacher und kostengünstiger als die Erstellung nativer Anwendungen, insbesondere, wenn man die Wartung der plattformübergreifenden Anwendungen berücksichtigt.

Eine progressive Web-App ist eine Website, die technisch so entwickelt wurde, dass sie fast wie eine native App funktioniert. Das mobile Erlebnis ist so viel angenehmer als bei einer normalen mobil optimierten Website.

Eine PWA bietet die folgenden Funktionen:

- Offline-Support
- Lädt schnell
- Ist sicher
- Ist in der Lage, Push-Benachrichtigungen auszusenden
- Hat eine immersive, bildschirmfüllende Benutzeroberfläche ohne URL-Leiste

Mobile Plattformen bieten eine zunehmende Unterstützung für progressive Web-Apps bis hin zur Aufforderung an den Benutzer, die App dem Startbildschirm hinzuzufügen, sobald sie eine Webseite als PWA erkennen.

Progressive Web-Apps – Alternativen

Wodurch unterscheidet sich eine PWA von anderen App-Arten, wenn es darum geht, ein mobiles Erlebnis zu schaffen? Betrachtet man die Vor- und Nachteile der Alternativen, ist klar zu sehen, für welche Fälle PWAs am besten geeignet sind.

Native Mobile Apps

Native mobile Apps sind der offensichtlichste Weg, um eine mobile App zu erstellen, z. B. Objective-C oder Swift auf iOS, Java/Kotlin auf Android und C# auf Windows Phone.

Alle Plattformen haben ihr eigenes User Interface (UI) und eine individuelle User Experience (UX), sodass die nativen Widgets das Erlebnis bieten, das der Nutzer erwartet. Diese können über die Plattform-App-Stores bereitgestellt werden.

Die größte Herausforderung bei nativen Apps liegt darin, dass eine plattformübergreifende Entwicklung das Erlernen, Beherrschen und Aktualisieren vieler verschiedener Methoden und Best Practices erfordert. Hat man ein kleines Team oder ist sogar Solo-Entwickler und erstellt eine App für drei Plattformen, muss man viel Zeit damit verbringen, die jeweilige Techno-

DIE AUTORIN

Magdalena Mues ist eine der Gründer/-innen und Geschäftsführer/-innen von CLANEO. Mit ihrem Team berät sie Unternehmen in verschiedenen Online-Marketing-Themen, insbesondere im SEO, SEA und im Content-Marketing.

DER AUTOR

Matthias Böhm ist SEO Consultant bei CLANEO. Seine Schwerpunkte sind technisches SEO sowie das Entwickeln von Konzepten zur Visualisierung von Daten und Kennzahlen.

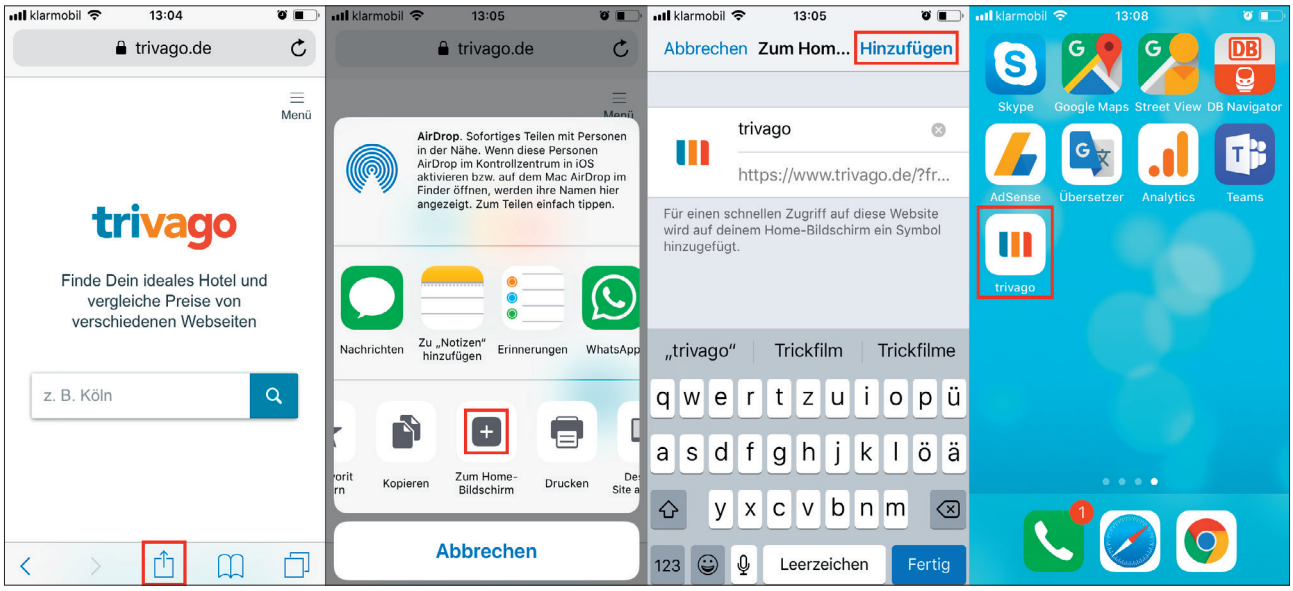

Abb. 1: App auf dem Startbildschirm installieren

logie und die IT-Umgebung zu erlernen, verschiedene Bibliotheken zu verwalten und diverse Workflows zu verwenden.

Hybrid-Apps

Hybride mobile Apps funktionieren wie alle anderen Apps, die man auf seinem Smartphone vorfindet. Sie werden auf dem Gerät installiert. Man kann sie in App-Stores downloaden. Auf ihnen kann man Spiele spielen, seine Freunde über soziale Medien ansprechen, Fotos machen, seine Gesundheit verfolgen und vieles mehr.

Hybrid-Mobile-Apps werden wie herkömmliche Websites im Internet mit einer Reihe von Web-Technologien wie HTML, CSS und JavaScript erstellt. Der wesentliche Unterschied besteht darin, dass Hybrid-Apps innerhalb einer nativen Anwendung bereitgestellt werden, die auf der Grundlage des WebView einer mobilen Plattform arbeitet. (Man kann sich den WebView als ein transparentes Browserfenster vorstellen, das typischerweise für die Ausführung im Vollbildmodus konfiguriert wurde.) Auf diese Weise kann der Nutzer auf Gerätefunktionen wie Beschleunigungs-Sensor, Kamera, Kontakte und mehr zugreifen. Dies sind Funktionen, die oft auf den Zugriff aus mobilen Browsern beschränkt werden. Darüber hinaus können hybride mobile Anwendungen bei Bedarf native UI-Elemente beinhalten.

Die gängigsten Plattformen sind Phonegap, Xamarin, Ionic Framework und viele andere, bei denen es sich in der Regel um ein WebView handelt.

Der wichtigste Aspekt von Hybrid-Apps ist das Konzept Write Once, Run Anywhere, das die plattformübergreifenden Vorteile in JavaScript beschreibt. Der unterschiedliche Plattformcode wird zur Buildzeit generiert, also zur gleichen Zeit, wie die Daten für das Endgerät bereitgestellt werden. Man entwickelt Apps mit JavaScript, HTML und CSS, was fantastisch ist. Die Gerätefunktionen (Mikrofon, Kamera, Netzwerk, GPS ...) werden über JavaScript-APIs bereitgestellt.

Progressive Web-Apps – Merkmale

Was können PWAs also im Vergleich zu Native- und Hybrid-Apps und was sind ihre wichtigsten Merkmale?

Features

Progressive Web-Apps sind mithilfe von Suchmaschinen auffindbar. Wenn ein Benutzer auf Ihre Website gelangt, die über PWA-Funktionen verfügt, fragen Browser und Gerät, ob der Benutzer die App auf dem Startbildschirm installieren möchte. Dies ist sehr wichtig, weil sich eine regelmäßige Suchmaschinenoptimierung auf Ihre PWA beziehen kann, was zu einer viel geringeren Abhängigkeit von bezahlter Werbung führt.

PWAs sind im Grunde HTML5-Anwendungen mit Schlüsseltechnologien, die einige der wichtigsten Funktionen ermöglichen. Wenn man sich zurückerinnert, wurde das iPhone ohne die Möglichkeit angeboten, native Apps zu entwickeln. Die Entwickler wurden stattdessen angewiesen, mobile HTML5-Apps zu entwickeln, die auf dem Startbildschirm installiert werden konnten. Doch leider war damals die Technologie noch nicht dafür bereit.

Progressive Web-Apps werden offline ausgeführt.

Der Einsatz von Service Worker (eine moderne Browsertechnologie, die mittels JavaScript einen Proxy zwischen dem Webbrowser und dem Server bereitstellt) ermöglicht es der App, immer neue Inhalte zu präsentieren und diese im Hintergrund zu laden. Er unterstützt Push-Benachrichtigungen, um breitere Möglichkeiten der erneuten Kontaktaufnahme zu bieten.

Außerdem können Benutzer die App ganz einfach teilen, da nur eine URL benötigt wird.

Vorteile

Warum sollten sich Benutzer und Entwickler also um progressive Web-Apps kümmern?

1. PWAs sind leichtgewichtiger: Native Apps können 200 MB oder mehr umfassen, während eine PWA im KB-Bereich liegen kann.
2. Man benötigt keinen nativen Plattformcode.
3. Es ist viel einfacher, einen Benutzer davon zu überzeugen, eine Website zu besuchen als eine App zu installieren.
4. Eine PWA bedarf deutlich weniger Aufwand bei der Erstellung und Veröffentlichung von Updates.
5. Es gibt deutlich mehr Unterstützung für Deep-Links als bei App-Store-Apps.

Nachteile

Hier sollte man sich als Entwickler die Frage stellen, ob bestimmte Funktionen benötigt werden.

1. Die doppelten Menüs (Browser-Menü und App-Menü) sind unter Umständen für den Nutzer verwirrend.
2. Die App gleicht einem weiteren offenen Browser-Tab, dadurch kann der Nutzer schnell wechseln.
3. Die PWAs haben keinen Zugriff auf die Kontakte und Kalender.

Kernkonzepte

Diese zehn Konzeptpunkte verdeutlichen, warum PWAs für Entwickler und Nutzer so wertvoll sind.

1. **Reaktionsschnell:** Die Benutzeroberfläche passt sich der Bildschirmgröße des Geräts an.
2. **App-ähnliches Erscheinungsbild:** PWAs sehen nicht wie Webseiten aus, sondern annähernd gleich wie eine App.
3. **Offline-Unterstützung:** PWAs nutzen den Gerätespeicher, um Offline-Erfahrung zu ermöglichen.
4. **Installierbar:** Der Gerätebrowser fordert den Benutzer auf, die App zu installieren.
5. **Wiederaufnahme:** Push-Benachrichtigungen helfen Benutzern, die Anwendung nach der Installation wiederzuentdecken.
6. **Erreichbar:** Suchmaschinen können viel mehr Nutzer ansprechen als der App-Store.
7. **Aktuell:** Die App aktualisiert sich selbst und die Inhalte werden online gestellt.

8. **Sicher:** PWAs verwenden HTTPS.
9. **Fortschrittlich:** PWAs funktionieren auf allen Geräten, auch auf älteren, auch wenn die Funktionen eingeschränkter sind (z. B. nur als Webseite, kann nicht installiert werden)
10. **Verlinkbar:** URLs können einfach verlinkt werden.

Ein großer Vorteil von progressiven Web-Apps ist, dass diese auch offline arbeiten können. Dieses Thema soll nun im nächsten Abschnitt genauer ausgeführt werden.

Service Worker

Ein Teil der Definition der progressiven Web-App ist, dass sie offline arbeiten kann. In diesem Fall bedeutet dies, dass Service Worker ein unverzichtbarer Bestandteil einer progressiven Web-App sind.

Ein Service Worker ist eine JavaScript-Datei, die als Vermittler zwischen der Web-App und dem Netzwerk fungiert. Aus diesem Grund kann ein Service Worker Cache-Dienste bereitstellen, das App-Rendering beschleunigen und die Benutzerfreundlichkeit verbessern.

Aus Sicherheitsgründen können nur HTTPS-Seiten Service Worker in Anspruch nehmen, was ein wichtiger Grund dafür ist, dass eine progressive Web-App über HTTPS bereitgestellt werden muss.

App-Manifest

Das App-Manifest ist eine JSON-Datei, mit der dem Gerät Informationen über die progressive Web-App zur Verfügung gestellt werden.

Dazu wird ein Link zum Manifest in der Kopfzeile aller Webseiten hinzugefügt:

```
<link rel="manifest"
href="/manifest.webmanifest">
```

In dieser Datei wird dem Endgerät mitgeteilt, wie die PWA dargestellt werden muss:

» Der Name und die Kurzbezeichnung der App
» Die Positionen der Symbole in verschiedenen Größen
» Die Start-URL, bezogen auf die Domain
» Die Standardausrichtung
» Der Splash-Screen

```
{
  „name“: „Die Wetter App“,
  „short_name“: „Wetter“,
  „description“: „Das ist ein Progressive Web App Beispiel“,
  „icons“: [{
    „src“: „images/icons/icon-128x128.png“,
      „sizes“: „128x128“,
      „type“: „image/png“
    }, {
      „src“: „images/icons/icon-144x144.png“,
      „sizes“: „144x144“,
      „type“: „image/png“
    }, {
      „src“: „images/icons/icon-152x152.png“,
      „sizes“: „152x152“,
      „type“: „image/png“
    }, {
      „src“: „images/icons/icon-192x192.png“,
      „sizes“: „192x192“,
      „type“: „image/png“
    }, {
      „src“: „images/icons/icon-256x256.png“,
      „sizes“: „256x256“,
      „type“: „image/png“
    }],
  „start_url“: „/index.html?utm_source=app_manifest“,
  „orientation“: „portrait“,
  „display“: „standalone“,
  „background_color“: „#3E4EB8“,
  „theme_color“: „#2F3BA2“
}
```

App-Manifest. Das App-Manifest ist ein W3C-Working-Entwurf: *https://www.w3.org/TR/appmanifest/*

App-Shell

Die App-Shell ist keine Technologie, sondern ein Designkonzept, das darauf abzielt, zuerst den Web-App-Container und kurz darauf den eigentlichen Inhalt zu laden und wiederzugeben. So wird dem Benutzer ein ästhetischer, App-ähnlicher Eindruck vermittelt.

Dies entspricht den Apple HIG(Human Interface Guidelines)-Empfehlungen, einen Splash-Screen zu verwenden, der der Benutzeroberfläche ähnelt.

Caching

Die App-Shell wird getrennt vom Inhalt zwischengespeichert, und sie ist so eingerichtet, dass das Abrufen der Shell-Bausteine aus dem Cache sehr wenig Speicherplatz in Anspruch nimmt.

Mehr über die App-Shell direkt bei Google unter: *http://einfach.st/gdev44*

Im Folgenden soll nun erörtert werden, wie SEO für progressive Web-Apps funktioniert. Unter anderem soll auf das Lighthouse-Chrome-Erweiterungs-Tool

und auf die Aspekte, die man bei der Erstellung von PWAs aus SEO- Sicht beachten sollte, eingegangen werden.

SEO für PWA

PWAs sind indizierbar. Das bedeutet, dass sie in den Suchergebnissen erscheinen können. Es gibt jedoch viele technische SEO-Aspekte, die berücksichtigt werden müssen, um sicherzustellen, dass eine PWA auffindbar und für Suchmaschinen richtig optimiert ist.

Die folgende Zusammenstellung geht auf einige der häufigeren unmittelbaren SEO-Probleme ein, die bei der Entwicklung einer PWA und der Aufbereitung für Nutzer und Crawlbots bestehen.

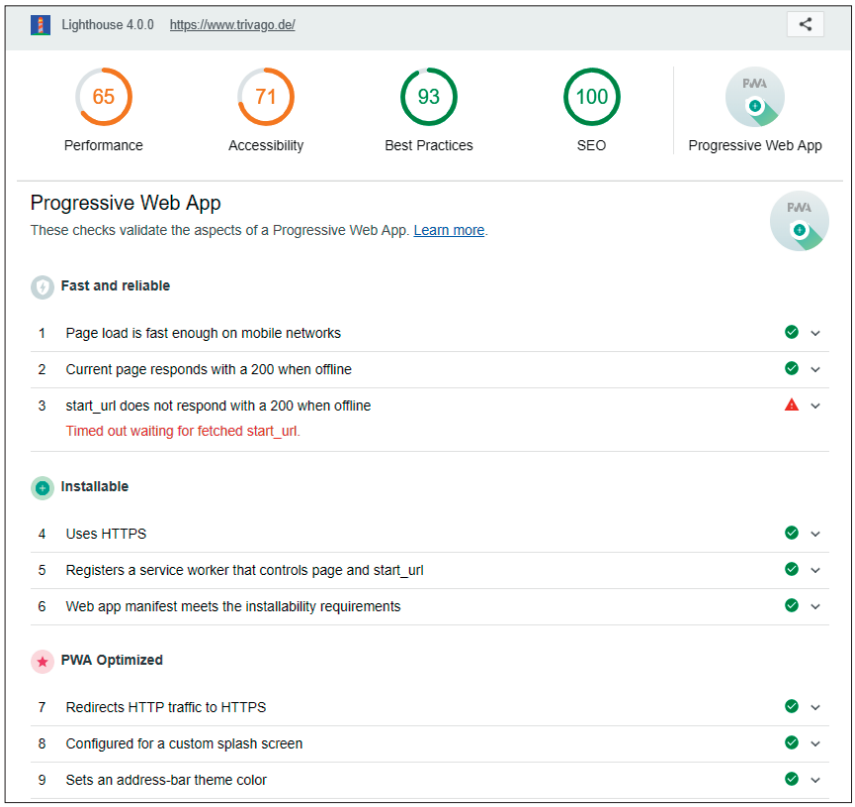

Abb. 2: Google Lighthouse-Tool

Installation von Lighthouse

Sowohl im Hinblick auf die Gesamterfahrung der App als auch auf SEO gibt es Einiges im Hinterkopf zu behalten: Viele der unmittelbaren Fragestellungen bei PWAs betreffen Maßnahmen zur Sicherstellung, dass die Website mit HTTPS geschützt ist, ein ansprechendes Design hat, offline geladen werden kann (ein wichtiger Aspekt von PWAs) und vieles mehr.

Das Lighthouse-Chrom-Erweiterungs-Tool ist ideal, um eine Basismessung Ihrer PWA vorzunehmen. Es wird nach den folgenden Komponenten gesucht:

» Offline-Zugänglichkeit: Hier wird auch berücksichtigt, ob der Nutzer schlechten Empfang hat.
» Seitenladegeschwindigkeiten: Der Bericht von Lighthouse listet eine Reihe verschiedener Geschwindigkeitswerte für Themen wie „Geschwindigkeitsindex“, „Geschätzte Eingabelatenz“ und mehr auf.
» Sicherheit der Netzwerkverbindung: Es wird überprüft, ob die PWA über ein SSL-Zertifikat für eine sichere Verbindung verfügt.
» Funktion „Zum Homescreen hinzufügen“: Wenn die PWA-Erfahrung besonders hilfreich ist, werden die Nutzer ein Verknüpfungssymbol zu ihren Homescreens hinzufügen wollen, sodass die Webanwendung diese Funktion ermöglichen muss.
» Mobiltaugliches Design: Es wird erkannt, ob die PWA so ansprechend gestaltet ist, dass sie auf einer großen Anzahl von Geräten angemessen funktioniert.
» Präsenz der Service Worker: Service Worker sind das Rückgrat von PWAs, da sie eine Reihe von Funktionen/Features handhaben, die mit Webanwendungen gleichzusetzen sind, wie Push-Benachrichtigungsfunktionen, Offline-Caching und Geolokalisierungsdaten.

Im folgenden Bild ist das Lighthouse-Chrome-Erweiterungs-Tool zu sehen, hier wurde die URL trivago.de überprüft.

Stellt Lighthouse bei der ersten Messung Probleme fest, wird das Chrome-Erweiterungs-Tool die Entwickler darüber informieren und einen Einblick in die Lösung aufzeigen.

Die Website mit gängigen Browsern testen

Ein Aspekt, den Lighthouse nicht selbst überprüfen kann, ist die Darstellung und Funktionsweise der PWA in verschiedenen Browsern. Das müssen die Entwickler selbst durchführen, indem die URL in jedem Browser aufgerufen und auf Funktionalität getestet wird. Obwohl dies kein direktes SEO-Problem ist, könnte ein Browser-Problem zu hohen Bounce-Raten bei den Nutzern führen.

Seiten der Webanwendung mit strukturierten Daten auszeichnen

Mit dieser Methode wird es Google erleichtert, jeden Teil der Webanwendung präzise zu identifizieren. Schema.org-strukturierte Daten sind dafür ideal, da sie die wichtigsten Teile der PWA mit Metadaten zusammenfassen und Google den Zweck jeder Seite mitteilen können.

Open-Graph- und Twitter-Cards-Metadaten berücksichtigen

Wie bereits erwähnt, ist einer der großen Vorteile von PWAs die Tatsache, dass sie gemeinsam genutzt werden können. Dies war schon immer ein großes Problem für native Apps. Um eine traditionelle App zu empfehlen, muss ein Nutzer den App-Store besuchen, die App finden, sie herunterladen, Berechtigungen akzeptieren und so weiter.

PWAs sind hingegen sofort über eine URL zugänglich. So können PWAs und ihre Funktionen über Social-Media-Kanäle geteilt werden.

Doppelte Inhalte vermeiden

Viele PWAs werden entwickelt, um eine bestehende Website zu ergänzen. Daher entstehen hier inhaltliche Überschneidungen.

Um sicherzustellen, dass man keine doppelten Inhalte vergibt, sollte man testen, ob alle sich wiederholenden Seiten über die notwendigen <link rel=canonical>-Tags verfügen, die auf die kanonische Quelle des Inhalts verweisen.

Fazit

Wer nun die Welt der Web-Anwendungen verbessern und funktionaler gestalten will, sollte sich ab diesem Punkt mit progressiven Web-Apps beschäftigen.

Progressive Web-Apps sind einfach zu entwickeln, zu entdecken, zu navigieren und zu installieren. Sie bieten sowohl Desktop- als auch mobilen Benutzern aller Plattformen ein neues Web-Erlebnis.

Zu den wichtigsten Funktionen von progressiven Web-Apps gehören die innovativen Funktionen, Offlinefähigkeit, Push-Benachrichtigungen, ein nahezu natives Erscheinungsbild von Apps, schnelle Ladezeiten sowie lokales Caching von Ressourcen. Darüber hinaus sind PWAs über die Grenzen der App-Stores leicht erreichbar. ¶