

Progressive Web App

Studienarbeit

des Studiengangs IT Automotive
an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Emmelie Beitlich, Finn Freiheit

Oktober 2020

Bearbeitungszeitraum
Matrikelnummer, Kurs
Gutachter

12 Wochen
2533282, ITA19
Dipl.-Ing. (FH) Peter Pan

Erklärung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Thema: *Progressive Web App* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Stuttgart, Oktober 2020

Emmelie Beitlich, Finn Freiheit

Abstract

Inhaltsverzeichnis

| | |
|--|------------|
| Abkürzungsverzeichnis | IV |
| Abbildungsverzeichnis | V |
| Tabellenverzeichnis | VI |
| Listings | VII |
| 1 Einleitung | 1 |
| 1.1 Motivation | 1 |
| 1.2 Begriffsklärung | 1 |
| 1.3 Aufbau der Arbeit | 3 |
| 2 Theoretische Grundlagen | 4 |
| 2.1 Progressive Web App Grundlagen | 4 |
| 2.1.1 Web-App-Manifest | 4 |
| 2.1.2 Service Workers | 7 |
| Literatur | 11 |
| Anhang | 13 |

Abkürzungsverzeichnis

| | |
|-------------|-----------------------------------|
| PWAs | Progressive Web Apps |
| PWA | Progressive Web App |
| JSON | JavaScript Object Notation |
| URL | Uniform Resource Locator |
| API | Applikation Programming Interface |
| DOM | Document Object Model |

Abbildungsverzeichnis

| | | |
|-----|---|---|
| 2.1 | display in Standalone Einstellung | 5 |
| 2.2 | display in Minimal User Interface Einstellung | 5 |
| 2.3 | Theme Color auf Weiß geändert | 6 |
| 2.4 | Entwicklereinstellungen Web-App-Manifest | 6 |
| 2.5 | Laden einer Applikation ohne Service Worker | 8 |
| 2.6 | Laden einer Applikation ohne Internetverbindung | 9 |

Tabellenverzeichnis

| | | |
|-----|---|---|
| 1.1 | Plattformen und die dazu benötigten Programmiersprachen | 1 |
|-----|---|---|

Listings

1 Einleitung

1.1 Motivation

Folgendes Szenario soll ein Einblick in die Vorteile von Progressive Web Apps ([PWAs](#)) aufzeigen.

Ein junges Startup aus IT-Studenten hat eine Idee für eine Applikation. Ihr Ziel ist es diese Applikation an so viele Nutzer wie möglich zu verbreiten. Die Applikation soll daher für folgende Plattformen, siehe Tabelle [1.1](#) erhältlich sein.

| Plattform | Programmiersprache |
|----------------|--------------------------|
| IOS | Swift |
| Android | Java oder Kotlin |
| MacOS | Swift |
| native Windows | C++ oder C# |
| Webbrowser | JavaScript, HTML und CSS |

Tabelle 1.1: Plattformen und die dazu benötigten Programmiersprachen

Wie man anhand der Tabelle sehen kann, wird eine Vielzahl an unterschiedlichen Programmiersprachen benötigt, um die Applikation über mehrere Plattformen zu verbreiten. Das junge Startup verfügt leider nicht über die Kapazitäten um die Applikation in jeder dieser Programmiersprachen zu implementieren und zu warten.

Aus diesem Grund entscheidet sich das Startup dafür eine Progressive Web App ([PWA](#)) zu erstellen. Eine PWA ist eine Webanwendung mit erweiterten Funktionen. Die Besonderheit dieser erweiterten Webanwendung liegt darin, dass sie einmal implementiert auf sämtliche Plattformen installiert werden kann. Sie ist somit Plattform unabhängig.

1.2 Begriffsklärung

der Begriff *Progressive Web App* setzt sich aus den Begriffen *Web App* und *Progressive Enhancement* zusammen. Eine Web App (deutsch Webanwendung) ist eine mithilfe von

JavaScript, HTML und CSS entwickelte Applikation. Der zweite Begriff wurde von Steve Champeon im Jahre 2003 in seiner Publikation mit dem Titel *progressive enhancement and the future of web design* geprägt [Cha].

Unter dem Begriff *Progressive Enhancement* (deutsch Progressive Verbesserung) verbirgt sich das Ziel Webseiten so zur Verfügung zu stellen, dass jeder Webbrowser in der Lage ist, die grundlegendste Form einer Webseite darzustellen. Hierbei ist es unabhängig über welche Version der Browser oder das Endgerät verfügt. Alle zusätzlichen Funktionalitäten, die eventuell erst mit modernen Browsern und Endgeräte genutzt werden können, werden erst im anschluss in form von Skripten eingebunden.

Um PWAs nutzen zu können werden die neusten Funktionen der modernen Webbrowser benötigt, darunter *service workers* und *web app manifests* (Referenz Kapitel).

Google hat das Konzept von PWAs im Jahr 2015 vorgestellt und ist seit dem maßgeblich an der Entwicklung beteiligt. Das Ziel bei der Entwicklung von PWAs liegt darin die Vorteile von Native Applikation mit den Vorteilen von Webanwendung zu kombinieren.

Native Applikation beziehungsweise Plattformspezifische Applikation sind sehr Funktionsreich und zuverlässig. Weitere Vorteile sind, das sie :

- Netzwerkunabhängig funktionieren,
- lokale Dateien aus dem Dateisystem lesen und schreiben können,
- auf Hardwareschnittstellen wie USB und bluetooth zugreifen können,
- mit Daten des Gerätes interagieren können, wie zum Beispiel Fotos oder aktuell spielende Musik.

Webapplikationen wiederum sind sehr gut erreichbar, sie können verlinkt, über Suchmaschinen gefunden und geteilt werden.

Mithilfe von PWAs können Applikation erzeugt werden, die:

- Installierbar sind, Kapitel,
- auf Geräteschnittstellen zugreifen können, Kapitel
- Netzwerkunabhängig funktionieren, Kapitel
- Push-Notifikations versenden können, Kapitel

PWAs sind somit laut Sam Richard und Pete LePage das beste aus zwei Welten [Sam]. Mithilfe von progressiver Verbesserung werden die modernen Funktionen von Browsern genutzt um die Vorteile von Plattformspezifischen Anwendungen nutzen zu können. Sind die dafür benötigten Funktionen wie zum Beispiel *service workers* nicht vorhanden, können dennoch die Grundfunktionen der Anwendung im Web genutzt werden.

1.3 Aufbau der Arbeit

2 Theoretische Grundlagen

2.1 Progressive Web App Grundlagen

Wie in Kapitel 1.2 erwähnt verfügt eine PWA unter anderem über folgende Funktionen:

- Installierbar,
- zugriff auf Geräteschnittstellen,
- Netzwerkunabhängig,
- Push-Notifikations.

Im folgenden werden die theoretischen Grundlagen erläutert, die benötigt werden, um diese Funktionalitäten zu realisieren.

2.1.1 Web-App-Manifest

The web app manifest is a JSON file that defines how the PWA should be treated as an installed application, including the look and feel and basic behavior within the operating system [Devb].

Eine PWA kann auf ein Endgerät wie zum Beispiel ein Desktop oder Handy installiert werden. Um diese Funktion zu realisieren, müssen zusätzliche Informationen wie zum Beispiel der Name und das Icon der installierten Applikation in einer Datei festgehalten werden.

Bei dieser Datei handelt es sich um das sogenannte Web-App-Manifest. Die Informationen sind im JavaScript Object Notation (JSON)-Format¹ angegeben.

Ohne Das Manifest ist die Applikation nicht installierbar, somit ist die Datei eine zwingende Voraussetzung für eine PWA. Das Manifest muss mindestens ein **name**-Schlüssel und ein **String**-Wert aufweisen. Neben dem Namen der Applikation kann ein Manifest über folgende Informationen verfügen:

¹ durch Komma getrennte Schlüssel-Wert paare

short_name

Unter **short_name** kann ein kurzer Name der Applikation angegeben werden. Dieser Name wird verwendet, falls das Endgerät nicht über genügend platz verfügt, um den Originalen Namen anzuzeigen.

icons

Unter **icons** wird ein Array¹ mit Bildobjekten gespeichert. Ein Bildobjekt besteht aus einem Dateipfad, unter dem das anzuzeigende Bild gespeichert ist, einer Typ Beschreibung des Bildes zum Beispiel *png* oder *svg*, eine Informationen über die Auflösung des Bildes und optional noch eine Angabe welchem Zweck das Bild dient.

Die Gespeicherten Bilder werden als App-Icon auf dem Desktop oder Handy angezeigt.

start_url

Die angegebene **start_url** ist jene Uniform Resource Locator ([URL](#)) die geöffnet wird, sobald der Nutzer das installierte Icon auswählt und somit die Applikation startet. Wird keine explizite Startadresse angegeben, so wird die URL verwendet, von der die PWA installiert wurde.

display

Beim Auswählen des Installierten Icons wird die PWA in einem neuen Fenster geöffnet. Unter **display** kann angegeben werden, wie das Betriebssystem das Fenster darstellen soll. Es kann zwischen **Fullscreen**, **Standalone** und **Minimal User Interface** unterschieden werden. Der unterschied zwischen den einzelnen Auswahlmöglichkeiten liegt bei den Navigationselementen, siehe Abbildung 2.1 und 2.2.



Abbildung 2.1: display in Standalone Einstellung

¹ Datentyp, das mehrere Werte speichern kann

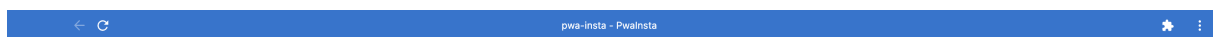


Abbildung 2.2: display in Minimal User Interface Einstellung

theme_color

Mit Hilfe dieser Einstellung kann die Farbe der oberen Navigationsleiste angepasst werden, wie in Abbildung 2.3 dargestellt ist. Hierbei ist jedoch darauf zu achten, das die Applikation nicht den meta tag theme-color definiert.

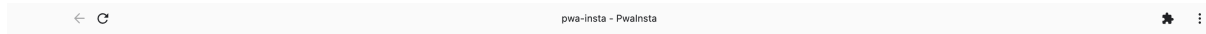


Abbildung 2.3: Theme Color auf Weiß geändert

Debugging vom Manifest

Neben den oben aufgezählten Grundeinstellungen sind viele weitere Möglich. Um nachzuvollziehen, ob alle Einstellungen den Anforderungen entsprechen kann das Manifest mithilfe der Browser Entwicklerwerkzeuge untersucht werden. Unter Google Chrome kann unterm Reiter *Application* das Manifest ausgewählt werden. Darauf hin erhält man folgende Ansicht, siehe Abbildung 2.4.

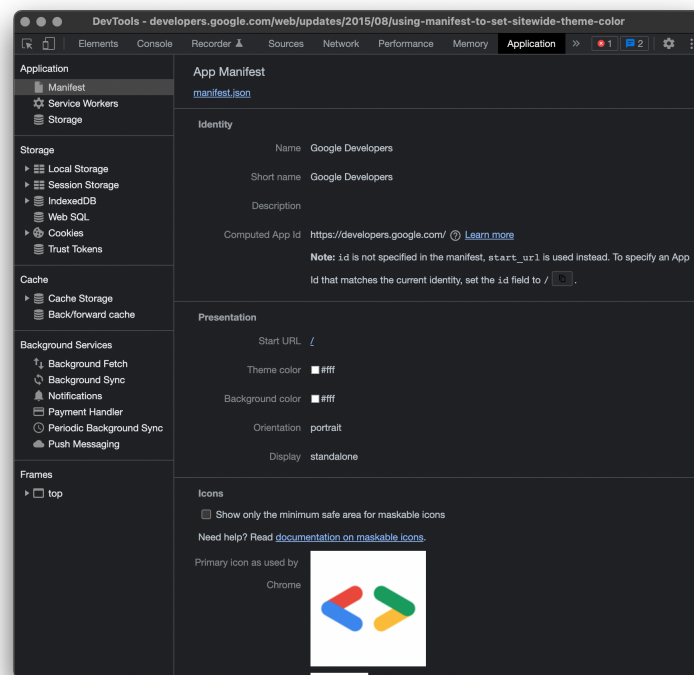


Abbildung 2.4: Entwicklereinstellungen Web-App-Manifest

2.1.2 Service Workers

Service workers are a fundamental part of a PWA. They enable fast loading (regardless of the network), offline access, push notifications, and other capabilities [Deva].

Der *Service Worker* ist ein wichtiger Grundbaustein um Funktionalitäten wie Push-Notifikationen, Hintergrund-Synchronisation und die Möglichkeit, auch Offline die Anwendung auszuführen, zu realisieren.

Bei dem *Service Worker* handelt es sich um ein script das im Hintergrund des Browsers, unabhängig von der Webanwendung, läuft [Gau]. Entstanden sind die service worker aus der Verwendung des Application Caches. Die service Worker Applikation Programming Interface (API) wächst kontinuierlich und bietet zunehmende weitere Funktionalitäten.

Bei der Verwendung eines Service Worker sollten folgenden Eigenschaften berücksichtigt werden:

- Ein service worker kann zwar nicht direkt das Document Object Model (DOM) einer Seite manipulieren, kann aber auf Requests der Seite mit Responses reagieren und die Seite selbst kann daraufhin ihr DOM ändern
- Ein service worker ist ein „programmierbarer“ Proxy, der steuert, wie Requests von der Webseite behandelt werden.
- Service workers verwenden die IndexedDB API, um client-seitig strukturierte Daten persistent zu speichern.
- Service workers verwenden Promises.

Der Lebenszyklus eines Service Workers

Der Lebenszyklus eines Service Workers beginnt mit dem registrieren. Die Registrierung erfolgt im JavaScript-Quellcode der PWA. Der Service Worker wird vom Browser heruntergeladen und installiert, sobald die PWA mit registrierten Service Worker das erste mal aufgerufen wird.

Nach einer erfolgreichen Installation wird der Service Worker aktiviert und verwaltet ab dann sämtliche Requests der Applikation. Durch den Aktivierten Service Worker kann die Application auch ohne eine bestehende Internetverbindung geladen werden. Der Service Worker verarbeitet den Initialen Request der Applikation, leitet diesen jedoch nicht an den Webserver weiter, siehe Abbildung 2.5, sondern lädt die Applikation aus dem Cache Storage, siehe Abbildung 2.6.

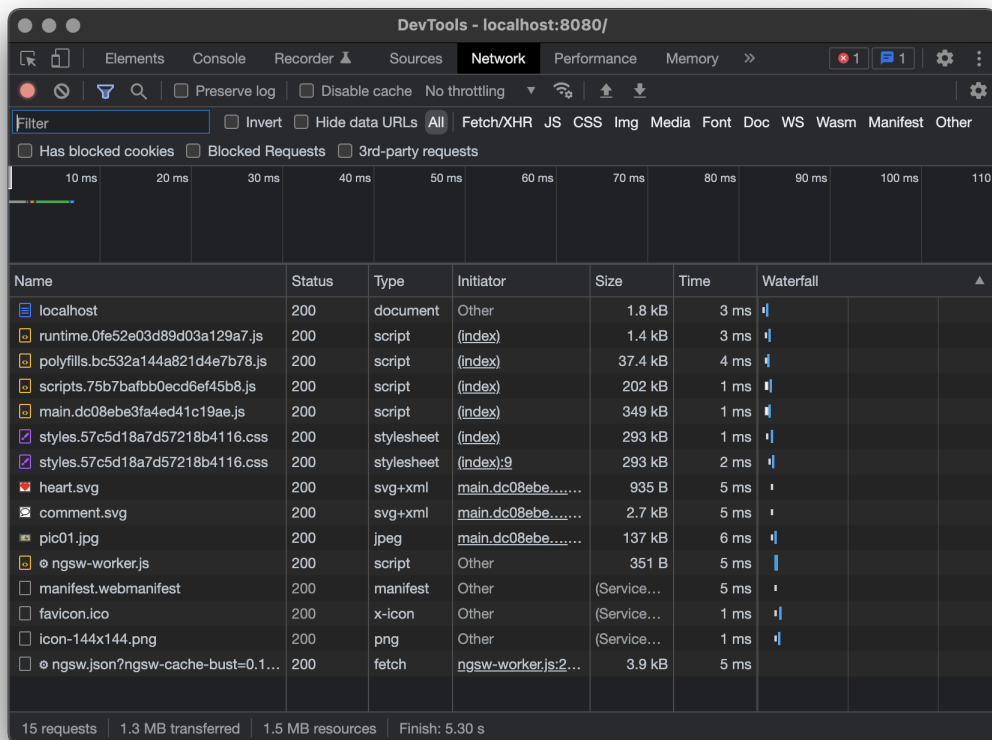


Abbildung 2.5: Laden einer Applikation ohne Service Worker

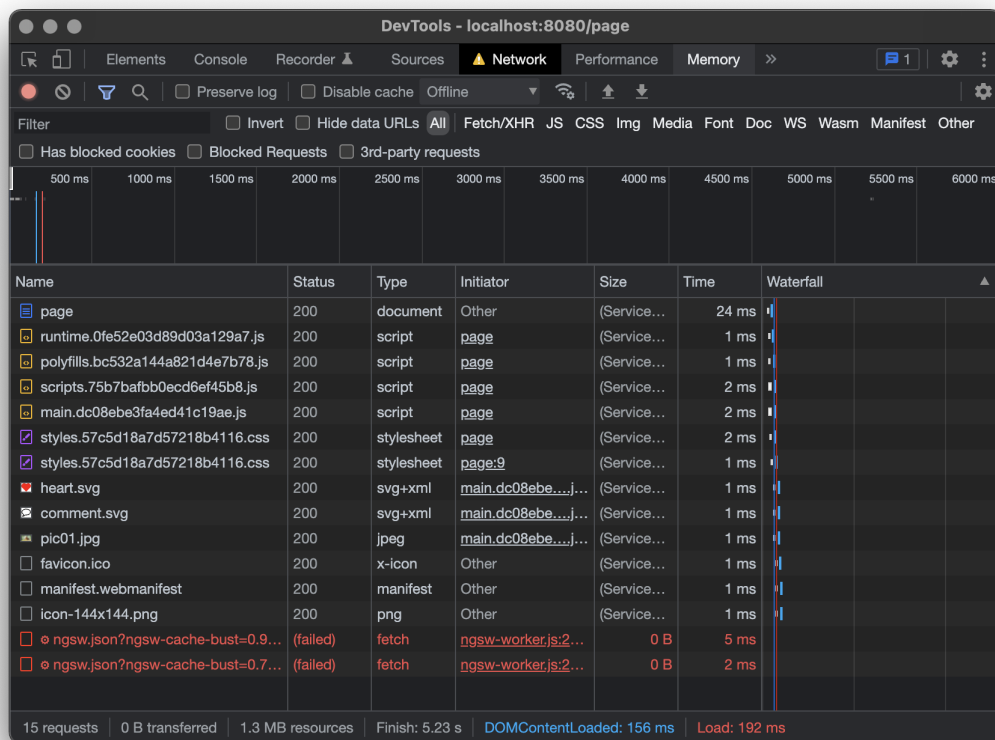


Abbildung 2.6: Laden einer Applikation ohne Internetverbindung

Push Notifikations

Eine Notifikation ist eine Nachricht, die auf dem Endgerät des Nutzers angezeigt wird. Dies kann als Reaktion auf eine Nutzereingabe geschehen, es kann jedoch auch unabhängig vom Nutzer eine Nachricht von einem Server „gepusht“ werden und das auch ohne dass die Applikation aktiv ist. Um push Notifikations zu erzeugen werden zwei APIs benötigt. Die *Notifications API* visualisiert die Nachricht für den Nutzer und die *Push API* erlaubt es den Service Worker, Kapitel 2.1.2 Nachrichten zu verwalten, die vom Server gesendet wurden während die Applikation nicht aktiv war.

Literatur

- [Cha] Steve Champeon. *PROGRESSIVE ENHANCEMENT AND THE FUTURE OF WEB DESIGN*. URL: http://www.hesketh.com/progressive_enhancement_and_the_future_of_web_design.html.
- [Deva] Google Developers. *Service workers*. URL: <https://web.dev/learn/pwa/service-workers/>.
- [Devb] Google Developers. *Web app manifest*. URL: <https://web.dev/learn/pwa/web-app-manifest/>.
- [Gau] Matt Gaunt. *Service Workers: an Introduction*. URL: <https://developers.google.com/web/fundamentals/primers/service-workers>.
- [Sam] Sam Richard, Pete LePage. *What are Progressive Web Apps?* URL: <https://web.dev/progressive-web-apps/>.

Anhang