Das Zauberwort dafür ist *VAPID („Voluntary Application Server Identification for Web Push“)*. Jeder Browser-Anbieter, der *Push*-Nachrichten in dieser Form anbietet, stellt auch einen eigenen *Messaging*-Dienst zur Verfügung. Damit werden – wie erwähnt – Daten am *eigenen Server* verschlüsselt und *erst am Client* entschlüsselt: sehr sicher und vom Datenschutz her bestens geeignet.

Ein älteres Verfahren heißt *GCM („Google Cloud Messaging“)*, wird per 11. April 2019 eingestellt und von *FCM („Firebase Cloud Messaging“)* abgelöst (https://www.heise.de/developer/meldung/Google-Cloud-Messaging-Abschaltung-am-11-April-2019-4021944.html). Hier führt der Weg der Mitteilung über die Google Server. Google ist eine US-amerikanische Firma und unterliegt dem *Patriot Act*. Ob dabei die Geheimhaltung (Verschlüsselung) lückenlos garantiert wird, darf zumindest hinterfragt werden. Siehe auch https://security.googleblog.com/2018/06/end-to-end-encryption-for-push.html Leider verwenden sehr viele Apps GCM bzw. FCM.

Background Sync

Nachrichten und Daten, die von einem Client (einem Handy) gesendet werden, müssen verlässlich an den Empfänger übermittelt werden. Unabhängig davon, ob wir gerade in einem Versorgungsloch im Waldviertel unterwegs sind, in einen Aufzug einsteigen, eine Tiefgarage aufsuchen oder das WLAN in einem Gebäudeteil nicht funktioniert: sobald die Internetverbindung wieder hergestellt ist, sind die Daten- ohne nochmalige Eingabe oder sonstige Aktionen – zu übertragen.

Auch diese Aufgabe übernimmt der Service Worker. Die Nachricht wird als *Sync Event* dem Service Worker übergeben, der sich verlässlich um die Ausführung kümmert. Na ja – abdrehen darf man das Handy eben nicht...

Node, npm

ECMAScript ist nicht mehr auf den Client (den Browser) beschränkt, sondern wird auch auf der Serverseite eingesetzt. Wichtig ist dabei eine effiziente Implementierung, da ein Server ja viele Anfragen gleichzeitig behandeln muss. Derartig große Programmpakete müssen effizient gewartet werden: Versionen und gegenseitige Abhängigkeiten sind automatisch zu verwalten.

Node ist die Plattform, auf der solche Programm gesammelt werden (https://nodejs.org/en/, https://de.wikipedia.org/wiki/Node.js). Um nun einzelne Programm auf einem Gerät zu installieren, wird *npm* (früher: *Node Package Manager*) (https://www.npmjs.com/, https://de.wikipedia.org/wiki/Npm_(Software)) eingesetzt.

Icons am Startschirm

Eine Kleinigkeit fehlt noch für das *„App Feeling“:* ein Icon am Start-Bildschirm. Aber auch hier sorgen PWAs vor: über eine Manifest-Datei wird festgelegt, was wie angezeigt werden soll. Nach dem ersten Aufruf der PWA wird der Nutzer gefragt, ob er ein Icon installieren will. Mit der positiven Bestätigung ist alles erledigt.

Und was ist daran Progressive?

PWAs können und sollen so geschrieben werden, dass auch ältere Browser eine Nutzung der Webseite ermöglichen. Die Anzeige kann einen geringerem Leistungsumfang oder geringere Geschwindigkeit haben, darf aber nicht ausfallen. Diese Eigenschaft wird als *progressive* bezeichnet.

Das war's jetzt – oder?

Jein! Ja, wer alles brav durchgearbeitet hat, kann schöne PWAs schreiben. Mit der *IndexedDB* bleiben Daten auch wohl geordnet am Mobiltelefon (Tablet, Desktop,...) erhalten. Was aber, wenn ich möchte, dass diese Daten auch auf anderen Geräten von mir oder von anderen verfügbar sein sollen? Und wenn Änderungen an einem Datensatz auch sofort an alle anderen Nutzer dieser Datenbank weiter gegeben werden sollen, also automatisch repliziert werden sollen. Geht das?

Ja! *PouchDB* ist ein Programmpaket, das auf die *IndexedDB* „draufgesetzt“ wird, und selbstständig mit anderen kommuniziert. Die Variante für den Server heißt *CouchDB*. Damit können auch Berechnungen, die am Server durchgeführt werden, schnell an alle Nutzer verteilt werden. Und das geht auch in der umgekehrten Richtung.

CouchDB und PouchDB sind nicht die einzigen Datenbankprogramme für diesen Zwecke – mir erscheint diese Kombination aber besonders effizient.

NoSQL

Viele kennen und arbeiten mit SQL, der *Structured Query Language*. *CouchDB* verwendet aber *NoSQL* (früher als *„No SQL“*, jetzt als *„Not only SQL“* interpretiert). *CouchDB* wird von Apache bereit gestellt. Wer sich noch nie mit *NoSQL* beschäftigt hat, betritt eine völlig neue Welt. So ziemlich alles, was wir über relationale Datenbanken wissen, ist bei *NoSQL* und damit beim Arbeiten mit *CouchDB* zu vergessen. Aber der Ansatz ist faszinierend und der Lohn ist eine Datenbank mit extrem schnellen Abfragen.

Lesen.

ECMAScript, PWAs, DOM, Preact, npm, Node, Cache, Clientseitige Datenbanken, *Service Worker, Push, Notification, Sync, NoSQL.* das wären so die wichtigsten Kapitel, die ich mir für ein Buch wünschen würde. Mit ausführlichen Beispielen, natürlich auch als Source-Code im Internet. Und vielleicht auf Deutsch? Also die eierlegende Wollmilch-Sau für das Erstellen von PWAs. Aber die gibt es leider nicht. Die Anzahl der Bücher zu einzelnen Themen ist überschaubar, zu anderen wieder sehr redundant. Dafür gibt es viele Artikel im Internet, Kommentare, Ratschläge. Dabei stellt sich heraus, dass viele Artikel einander ähneln.

Noch ein Problem: die verwendeten Programmbibliotheken werden weiter entwickelt. In vielen Beträgen fehlt aber eine Angabe zur Version oder einfach ein Datum. Die Suche nach einem Fehler wird nicht einfacher, wenn man auf einen Artikel über eine alte Programmversion zurück greift.

Viele Beiträge stammen von zwei Quellen ab, nicht wenige sind nur abgeschrieben: MDN und Google

Anstelle einer seitenlangen Literaturliste, mit der aus Zeitgründen erst wieder niemand etwas anfangen kann, ist die kurze Liste im Anhang mein Vorschlag, wie ich eine Annäherung an des Thema PWA empfehle.

Zusammenfassung

„Na gut, dann schreibe ich eben jetzt ein paar PWAs.“ Nein, leider, so einfach geht das nicht. In den vorherigen Abschnitten sind Konzepte beschreiben, die alle – zumindest in den Grundzügen – verstanden werden müssen, bevor irgendetwas, das einer PWA ähnlich sieht, entsteht. Es ist zweifellos sehr hilfreich, Beispielprogramme nachzuvollziehen. Aber jeder, der das schon probiert hat, kennt den Effekt: kaum baut man ein paar eigene Ideen ein, gibt es völlig unverständliche Fehlermeldungen, die erst nach langen Recherchen im Internet zu klären sind. Oft folgen dann noch Hinweise, die für den Wissenden völlig klar sind, aber beim Anfänger weitere Recherchen auslösen.

Diese Zeilen sollten aber niemand entmutigen, sondern eher anspornen, nicht aufzugeben. Wenn 2020 schon 50% aller Apps als PWAs geschrieben werden, sollten wir nicht daran vorbei gehen.

Neue Techniken lassen sich in Kursen oft wesentlich schneller erlernen als durch das Studium von Büchern und Webseiten. Viele Leser der PCNEWS sind an Schulen tätig. Die Pädagogischen Hochschulen wären gut beraten, Seminare zum Thema PWA anzubieten.

PWAs und SCHUL.InfoSMS bzw. SCHUL.InfoService

PWAs funktionieren ohne die Installation einer App, brauchen wenig Ressourcen, sind sehr schnell und auch einfach zu bedienen und daher optimal für alle bisherigen und zukünftigen Aufgaben von SCHUL.InfoSMS und SCHUL.InfoService geeignet. Mehr dazu im nächsten Heft der PCNEWS.