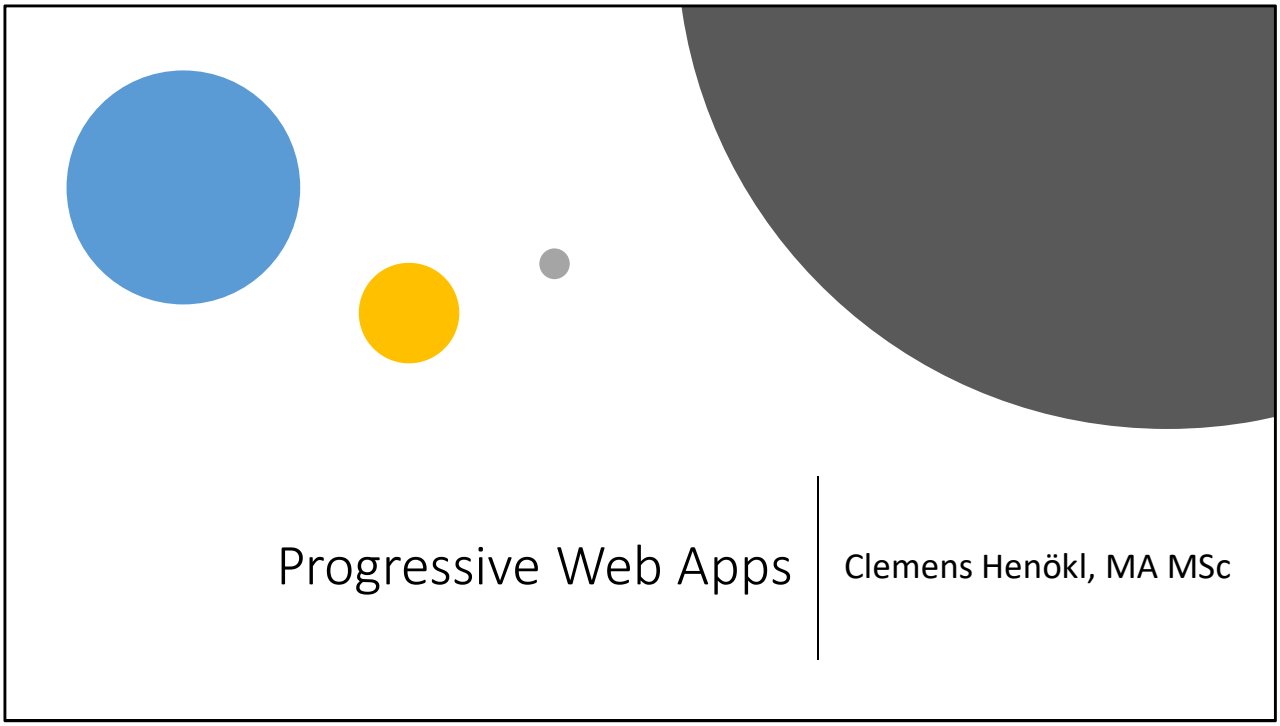

Progressive Web Apps
Clemens Henökl, MA MSc

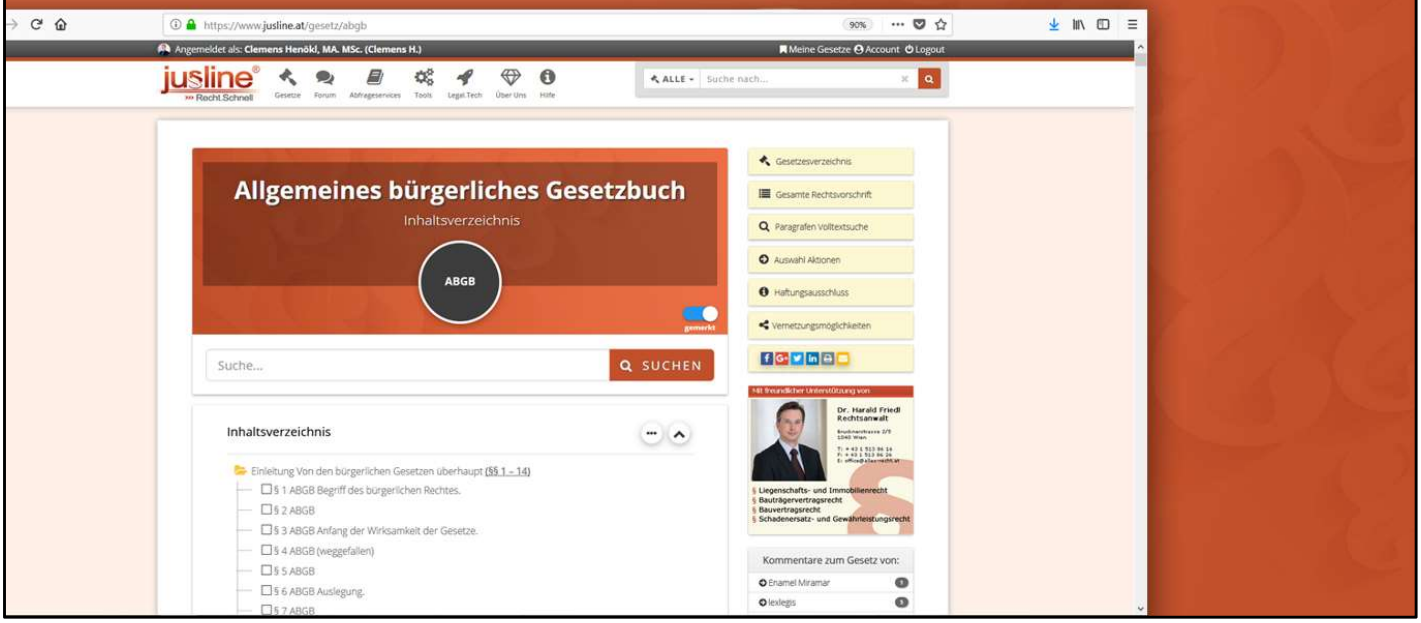

Personal
Introduction

Clemens Henökl

- Full stack web developer and team leader at ADVOKAT Unternehmensberatung GREITER & GREITER GmbH
- Master's degree at the Management Center Innsbruck (MCI) in the program of Management, Communication & IT
- Master's degree at the University of Nebraska Omaha (UNO) in the program of Management Information Systems
- Experience in software development:
  - Frontend: JavaScript, HTML & CSS
  - Backend: PHP
  - Database: MySQL & MSSQL
  - More: VB.NET, ABAP, REST, Regex, ...

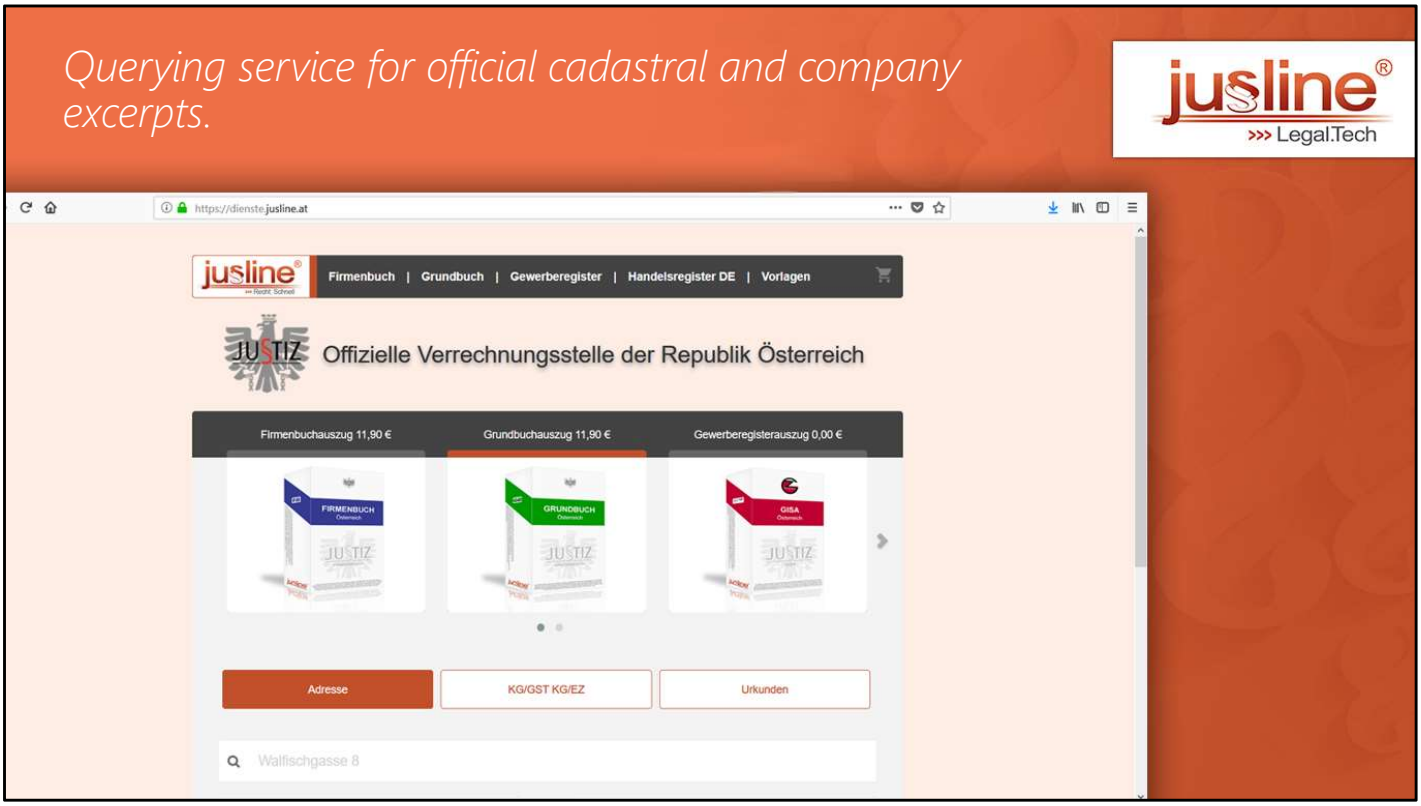
Platform for legal research and knowledge transfer.

jusline®
»» Legal.Tech



- >70.000 Users
- Laws and jurisdictions documents are provided
- Forum to share knowledge about legal topics
- Support for research and structuring of legal documents
- Application of machine learning in the area of natural language processing

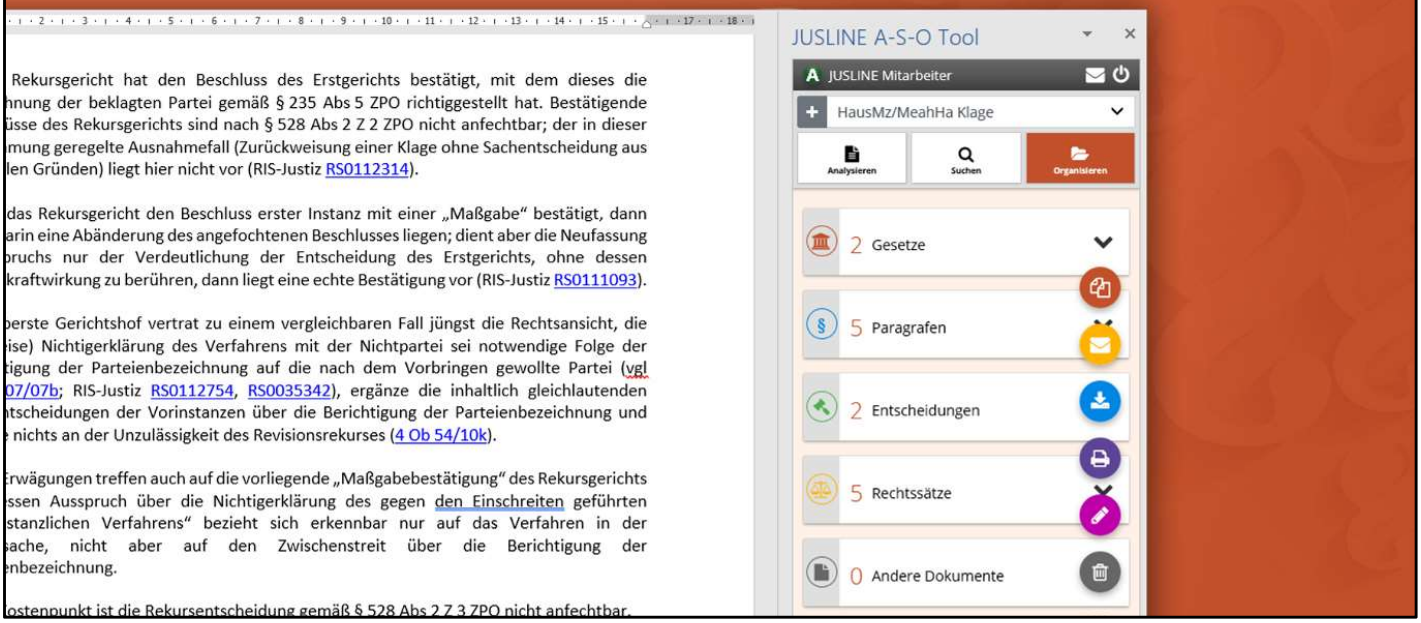
Querying service for official cadastral and company excerpts.



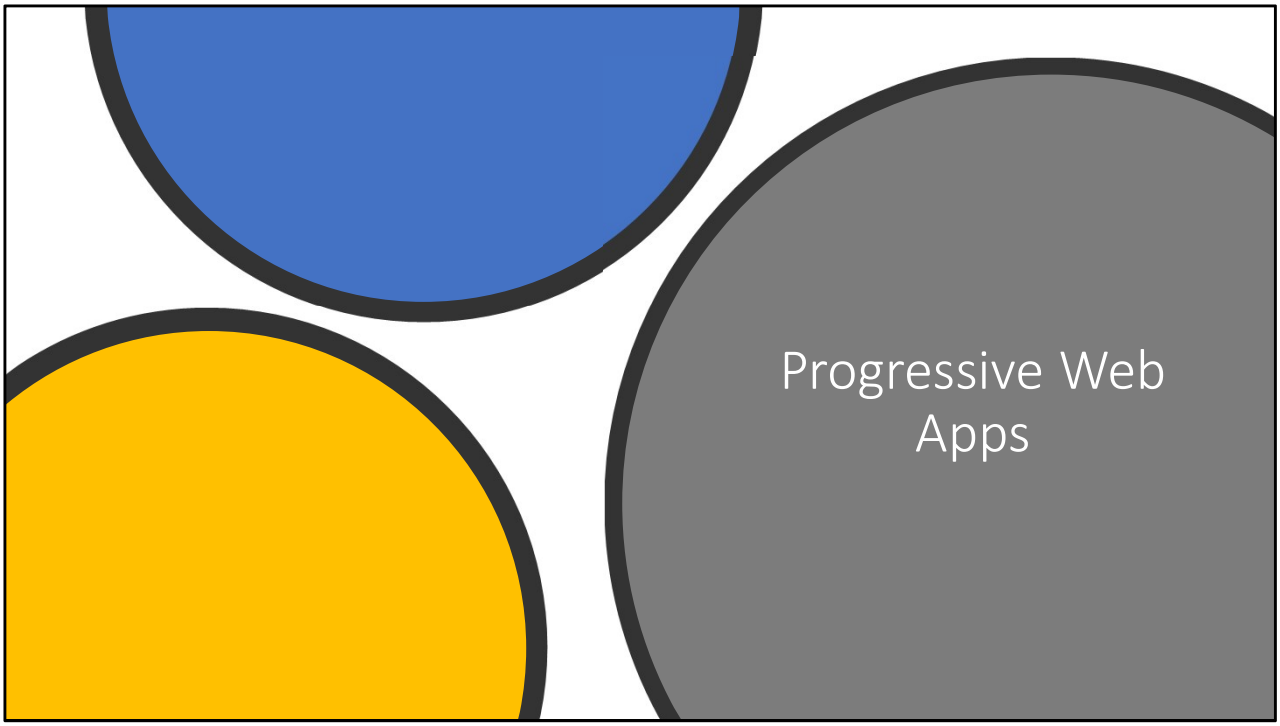
- Excerpt service for official cadastral and company excerpts
- Further standardized documents
- In future also custom contracts such as rental contracts will be available

Automatic text analysis and connection with online resources in Microsoft Word.

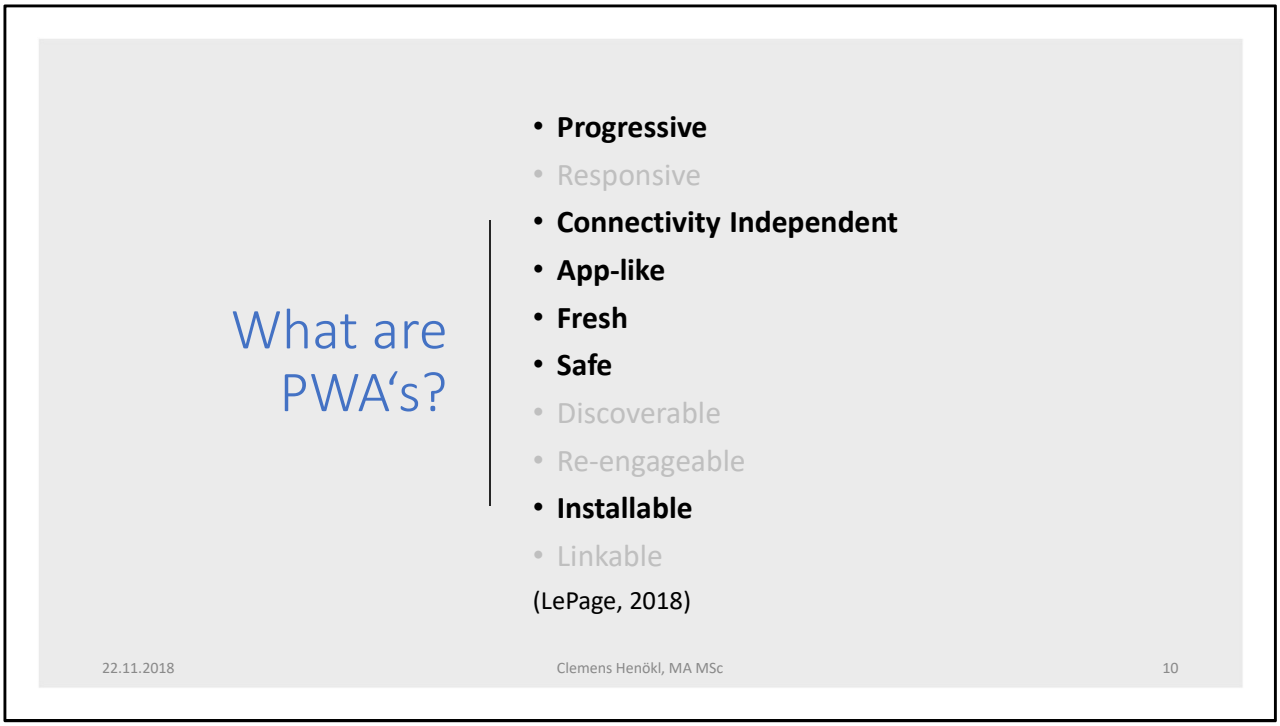
jusline®
>>> Legal.Tech



- Microsoft Word Add-in from 2016
- Automatic analysis of Word documents
- Connection of law and jurisdiction short names with corresponding online resources
- Structured organization of legal documents
- Different export options


Progressive Web
Apps

Progressive Web Apps (PWAs) are currentlc defined with 10 different characteristics stated by Google.



Only 6 of them will be covered in this presentation since they are perceived as the most important.

Progressive

- Works for every user
- Works in every browser
- Follows the progressive enhancement principle

(LePage, 2018; Robert, 2018)

In other words, the web application needs to work for every browser and user since the additional functionality will just be added on top.

Connectivity Independent

It allows

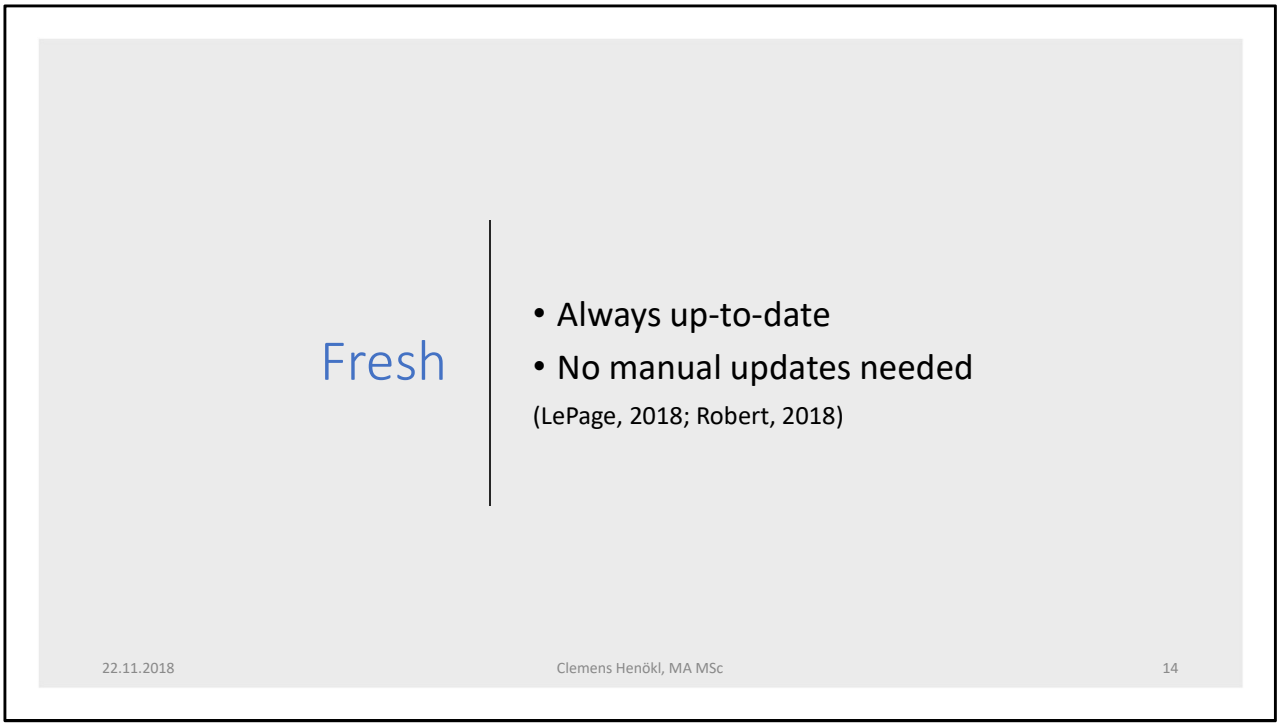
- to work offline
- and on low-quality networks

(LePage, 2018; Robert, 2018)

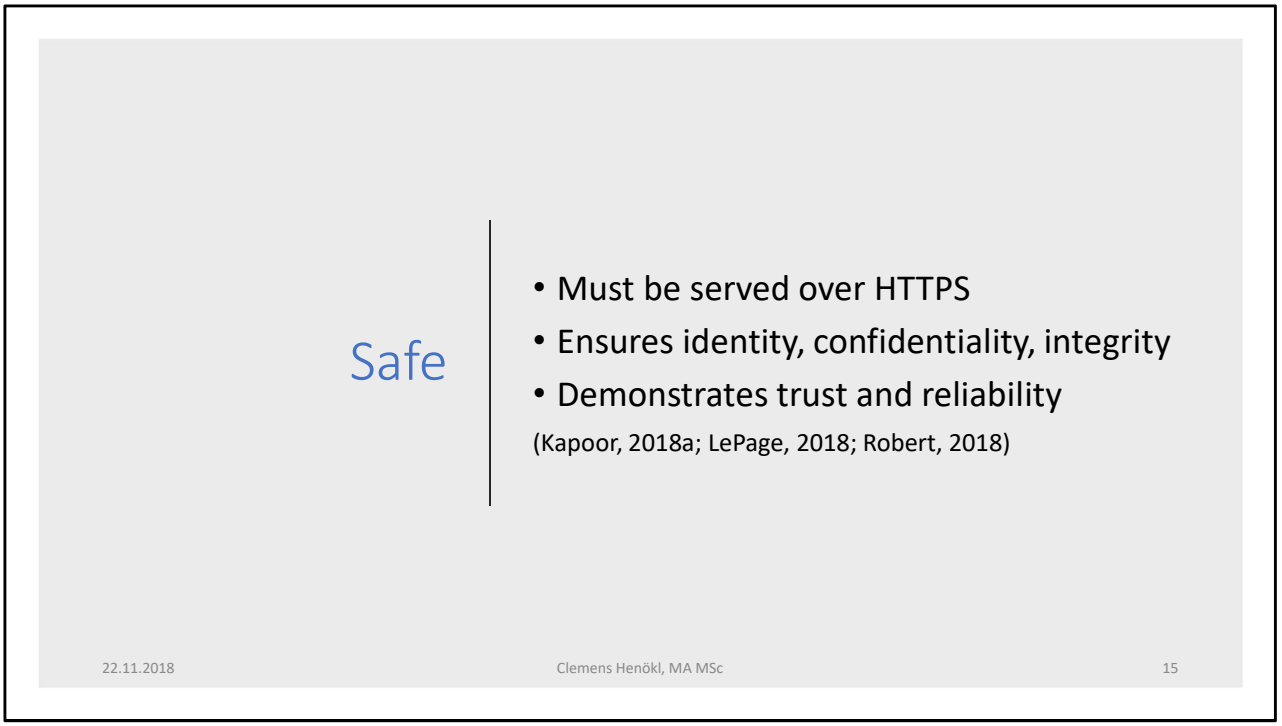
App-like

- Interactions and navigation feel like an app
- Use of app shell architecture
- Separation of functionality and content

(LePage, 2018; Robert, 2018)

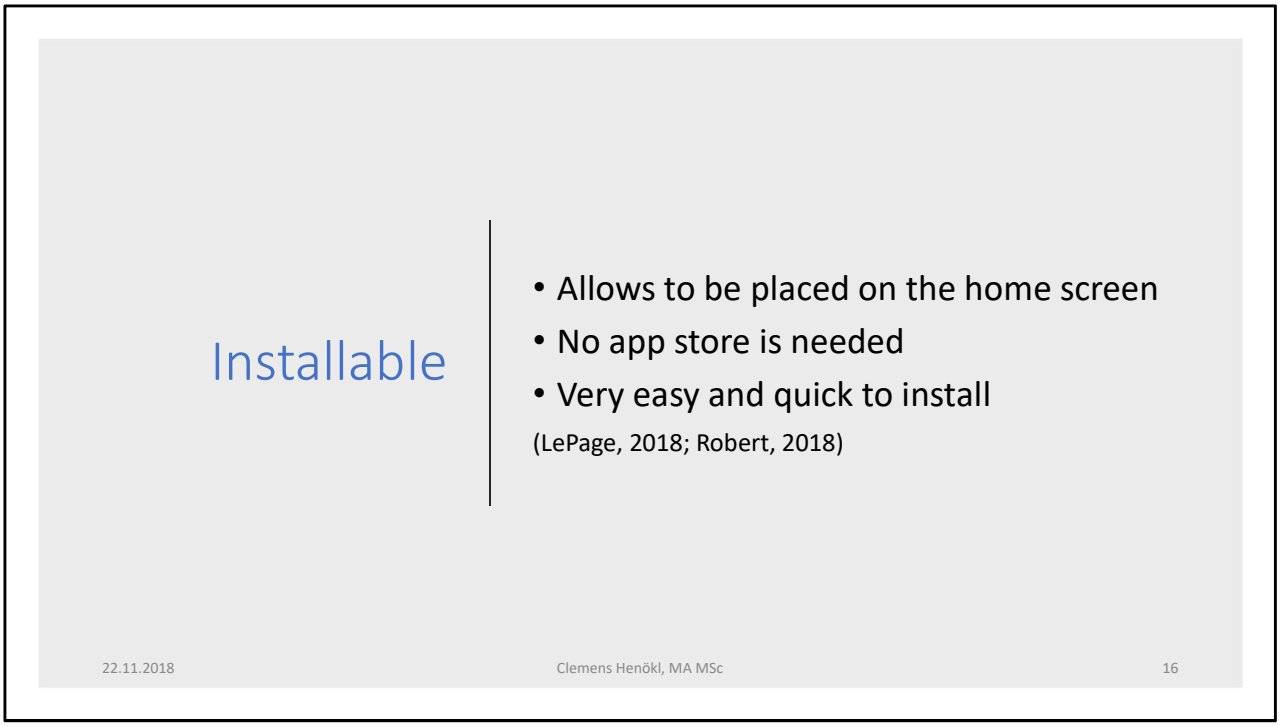


Since no updates are distributed via the app store, users won‘t have to update the app manually. Updates will be published immediately once the app sources are updated on the server (LePage, 2018; Robert, 2018).

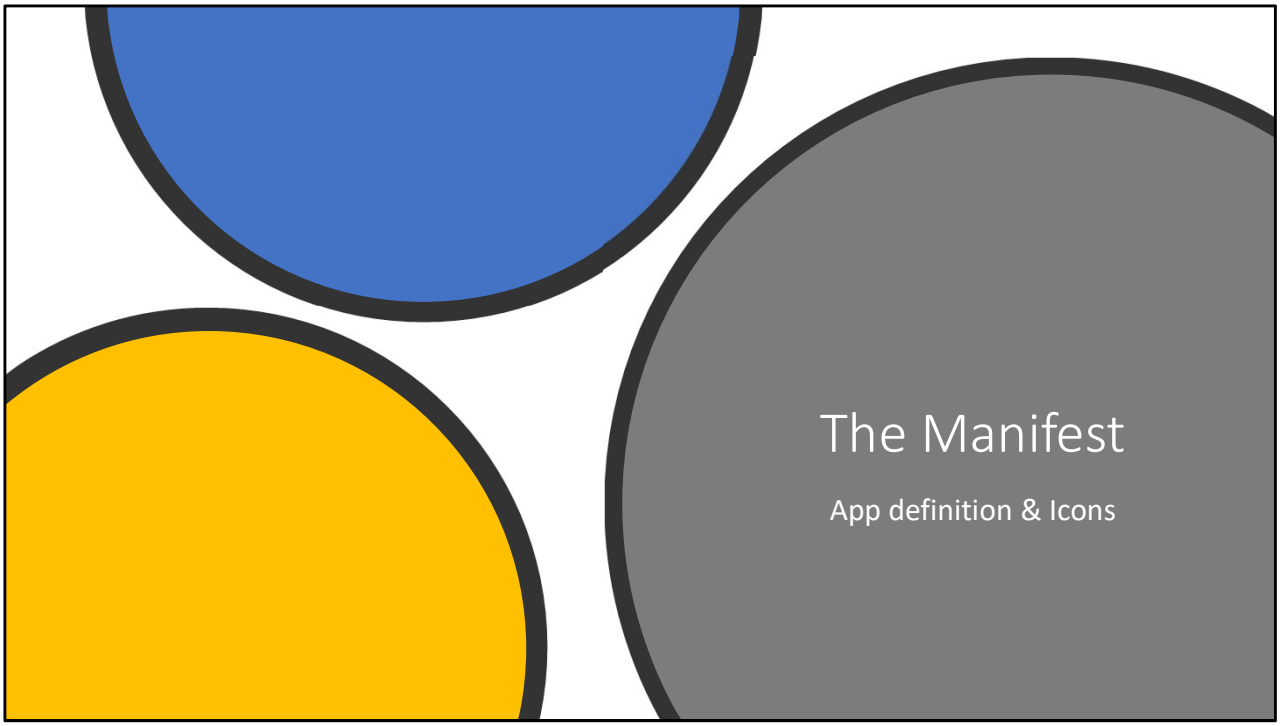


- Content won‘t be manipulated
- Avoid man in the middle attacks.
- Ensures that the identity is correct.

(Kapoor, 2018a; LePage, 2018; Robert, 2018)



Allows to install web apps with an install prompt to the home screen without an app store in an easy and quick manner (LePage, 2018; Robert, 2018).


The Manifest
App definition & Icons

Manifest

manifest.json

```
{
  "name": "PWA",
  "short_name": "PWA",
  "theme_color": "#1976d2",
  "background_color": "#fafafa",
  "display": "standalone",
  "orientation": "any",
  "scope": "/",
  "start_url": "/",
  "icons": [
    {
      "src": "assets/icons/icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
```

Structure of the manifest.json file

index.html

```
<!-- Tags für PWA -->
<link rel="manifest" href="manifest.json">
<meta name="theme-color" content="#1976d2">
```

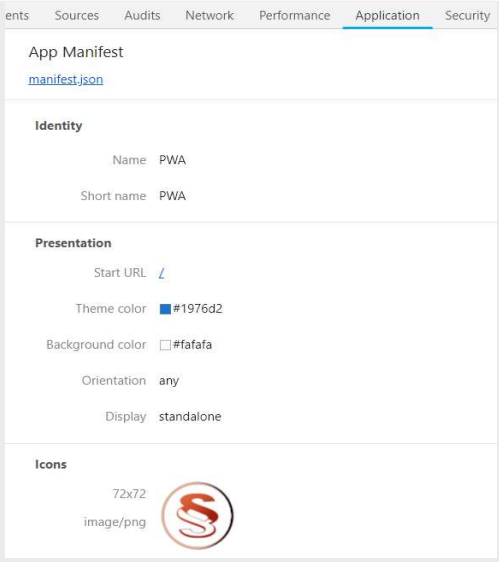
Reference to the manifest in the index.html

It‘s a JSON File which defines the basic app characteristics for the installation.

- Name => App Name
- Short_name => Name for the home screen icon
- Colors for the splash screen or browser bar
- Orientation => landscape, portrait or any
- Scope => To specify region the service worker is allowed to control
- Start_url => URL from which the app is started
- Icons => Different sizes are used for the home screen icon, tab icon, splash screen and further.

(Kapoor, 2018a; Smirnova, 2017)

Test the manifest



Check the manifest and Icons with Chrome Developer Tools (Google Developers, 2018; Smirnova, 2017)

At least one icon needs to be defined as well as its size needs to suite exactly to the stated sizes in the manifest. With Chrome Developer Tools in the Application tab it is possible to check the manifest file (Google Developers, 2018; Smirnova, 2017)


Service Workers
Lifecycle, Development Example & Storage

Registering a Service Worker

Progressive enhancement

Registers only if **serviceWorker** is in the **navigator** of the browser

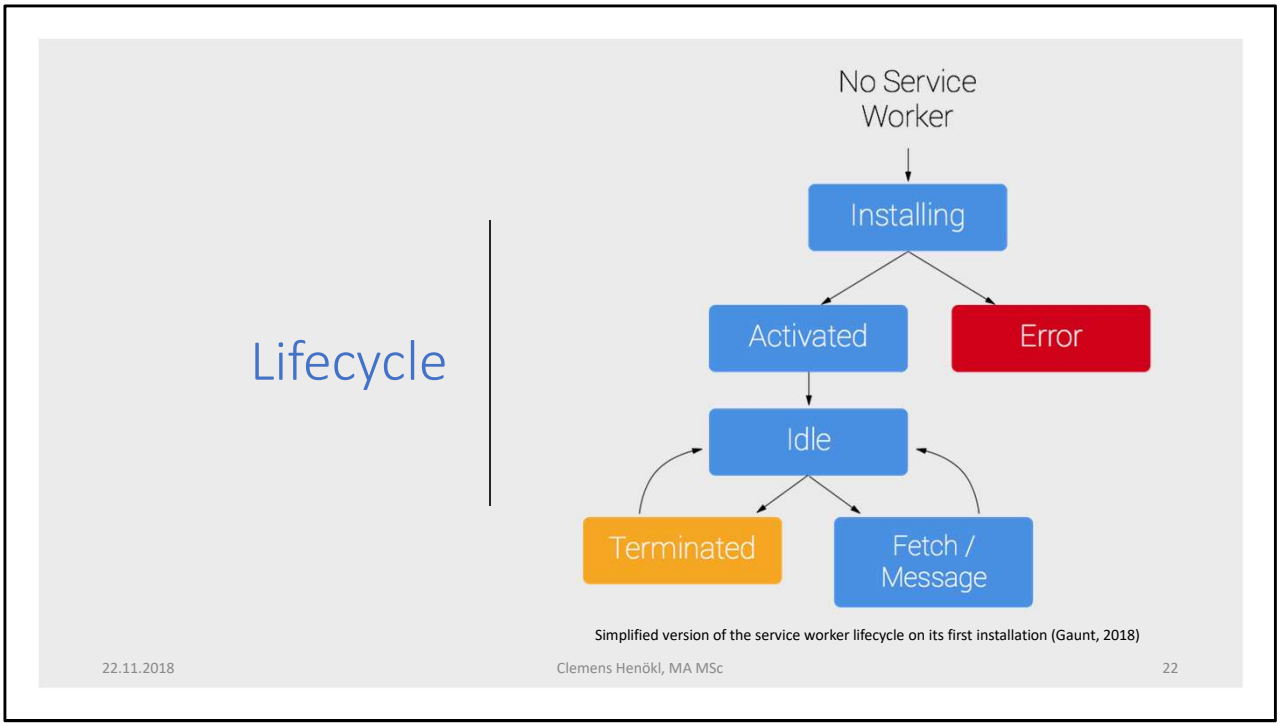
Registers the service-worker.js

```
366  // Register Service Worker
367  if ('serviceWorker' in navigator) {
368    navigator.serviceWorker
369      .register('./service-worker.js')
370      .then(function () {
371        console.log('Service Worker Registered');
372      });
373  }
```

Registering the service-worker.js file (LePage, 2018)

- Example for progressive enhancement
- Only registered if **serviceWorker** is in the **navigator** of the browser
- Registers the service-worker.js file

(LePage, 2018)



- The install event is the first event a service worker gets, and it only happens once.
- If any of the files fail to download and cache, the install step will fail. It will try again the next time the page is refreshed.
- After a successful Installation, the service-worker becomes activated.
- Then it will either be terminated because of an update or it will handle functional events such as fetch, push or sync.

(Archibald, 2018b; Gaunt, 2018)

Install Event

Specifies the cache name

Adds all stated files to the cache

Only triggered once

(Archibald, 2018b)

```
self.addEventListener('install', function (e) {
    console.log('[ServiceWorker] Install');
    e.waitUntil(
        caches.open('app-v1').then(function (cache) {
            console.log('[ServiceWorker] Caching app shell');
            return cache.addAll([
                '/',
                '/index.html',
                '/scripts/app.js',
                '/styles/style.css',
                '/images/icon.png',
            ]);
        })
    );
});
```

Code snipped for Install event in the lifecycle (LePage, 2018)

22.11.2018 Clemens Henökl, MA MSc 23

The install event is the first event which will be triggered after the registration of the serviceWorker. Here it is time to add all necessary files to the cache of the browser. Different caching strategies can be applied at this stage as described in the appendix.

- **Cache as dependency vs. Cache not as dependency**

(Archibald, 2018b; LePage, 2018)

Activate Event

Executed after the service worker has been installed

Time clean up cache

Ready to call functional events such as fetch, push or sync

(Archibald, 2018b)

```
46  self.addEventListener('activate', function(e) {
47    console.log('[ServiceWorker] Activate');
48    e.waitUntil(
49      caches.keys().then(function(keyList) {
50        return Promise.all(keyList.map(function(key) {
51          if (key !== 'app-v1') {
52            console.log('[ServiceWorker] Removing old cache', key);
53            return caches.delete(key);
54          }
55        }));
56      })
57    );
58    return self.clients.claim();
59  });
```

Code snipped for the activate event in the lifecycle (LePage, 2018)

22.11.2018 Clemens Henökl, MA MSc 24

The activate event will be triggered if the installation has been successfully and all dependent files have been loaded into the cache. At this stage it is time to clean up old cache files in case the serviceWorker needs to be updated because dependent app files have been changed. In case everything went well, the serviceWorker becomes active and functional events such as fetch, push or sync can be triggered (Archibald, 2018b; LePage, 2018).

Fetch Event

Executed only if service worker is active

Functional Event

Is triggered for every cached resource

(Archibald, 2018b)

```
71  self.addEventListener('fetch', function(e) {
72    console.log('[ServiceWorker] Fetch', e.request.url);
73    e.respondWith(
74      caches.match(e.request).then(function(response) {
75        return response || fetch(e.request);
76      })
77    );
78  });
```

Code snipped for the fetch event after service-worker has been activated (LePage, 2018)

The fetch event is triggered for every resource which is stored in the cache. Instead of requesting the resource from the network, it will fetch it from the cache. This stage will make the app available all the time, also if there is a network error in place. However, it needs to be provided that all necessary files have been cached. Otherwise the app will break (Archibald, 2018b; LePage, 2018).

Storage usage

- Chrome <6% of free space
- Firefox <10% of free space
- Safari <50MB
- IE10+ <250MB
- Edge Dependent on volume size

(Osmani & Cohen, 2018)

- In Chrome, apps can use up to 6% of free disk space.
- In Firefox, apps can use up to 10% of free disk space, but will prompt the user for further storage requests after 50MB data stored. I
- In mobile Safari, apps can use up to 50MB max, whereas desktop Safari allows unlimited storage (and prompts after 5MB).
- IE10+ maxes out at 250MB and prompts the user at 10MB.

(Osmani & Cohen, 2018)

Browser support

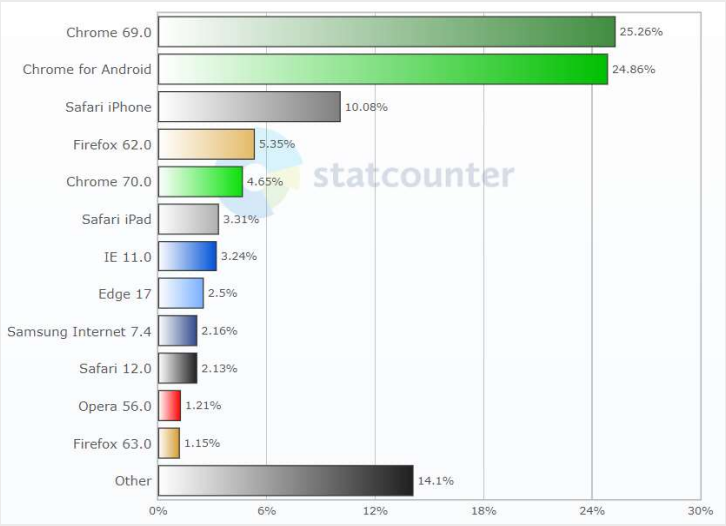
Chrome >40

Safari >11.1

Firefox >44

Edge >17

(vaadin, 2018b)



Browser Version Market Share Europe Oct 2018 (statcounter, 2018)

When it comes down to the browser support, it can be seen, that service workers are supported in all major browsers according to the chart for browser market share in Europe from October 2018. Nevertheless, this is just stating, that it support the basic technology of PWAs. There are still differences in supporting individual functionalities depending on the browser (Firtman, 2018; Gawron, 2018; Robert 2018; vaadin, 2018b).

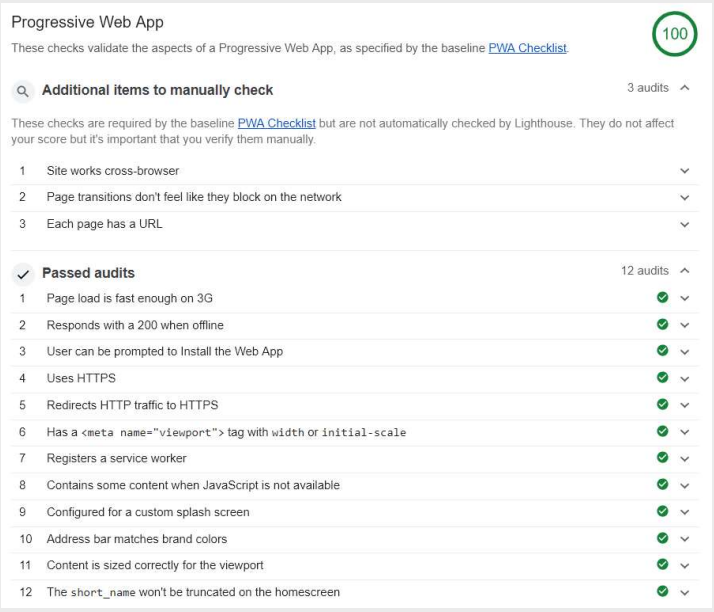
More information on this

More Information on service-workers, their lifecycle and a tutorial can be found in the following resources:

- Gaunt, M. (2018). Service Workers: an Introduction. Retrieved from https://developers.google.com/web/fundamentals/primers/service-workers/
- Archibald, J. (2018b). The Service Worker Lifecycle. Retrieved from https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle
- LePage, P. (2018). Your First Progressive Web App. Retrieved from https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/


PWA in practice

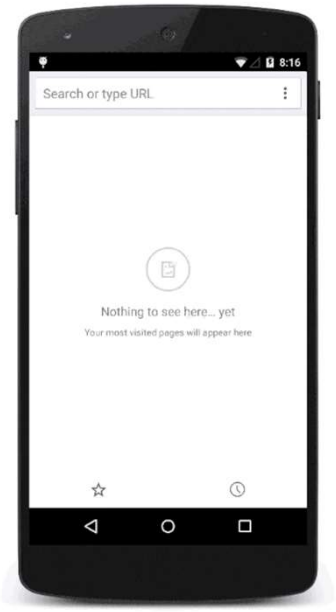
Audit with Lighthouse



Progressive Web App section in Lighthouse audit (Google Developers, 2018; Smirnova, 2017)

- Lighthouse is an open-source, automated tool for improving the quality of web pages.
- You can run it against any web page, public or requiring authentication.
- It has audits for performance, accessibility, progressive web apps, and more.
- In the progressive web apps section it provides a good insight in missing or wrong parts for your app to be a PWA

(Google Developers, 2018)



PWA add to home screen prompt (Gawron, 2018)

Here is an example of how the web app is asking to be installed to the home screen on mobile phones. Once the website is visited, the install prompt will appear if all prerequisites are fulfilled. With a touch on install it will add it with the stated icon and name in the manifest to a free position on the home screen. From now on, the app can be started like any other app in full screen mode (Gawron, 2018).

Summary

- A well defined manifest.json file
- Has at least one icon specified in the manifest.json
- Registers a service worker with at least a fetch event
- App is served over HTTPS

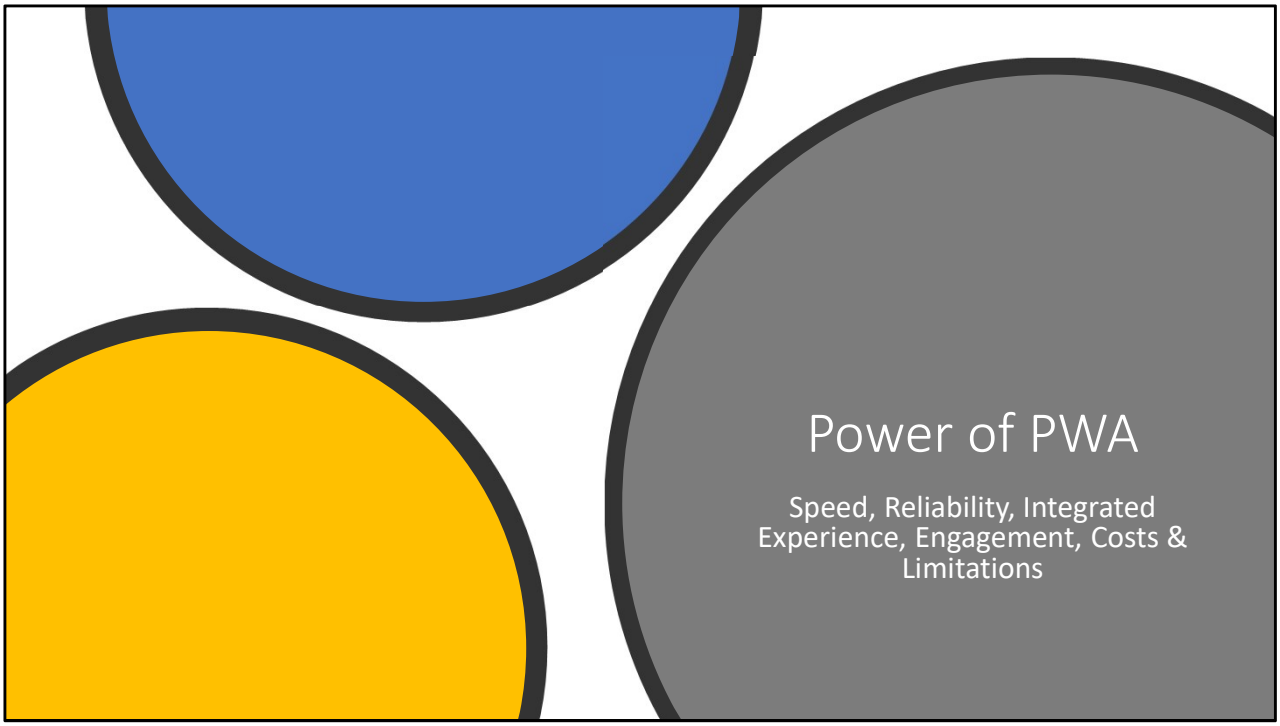
(Kapoor, 2018b)

In order to summarize this, these are the basic requirements for the web app to get the „install on home screen“ prompt and to be available offline (Kapoor, 2018b).

More information on this

Lots of more information to the manifest, audit and tutorials also with Angular 5+ can be found here:

- Angular University. (2018). Angular Service Worker: Step-By-Step Guide for turning your Application into a PWA. Retrieved from https://blog.angular-university.io/angular-service-worker/
- Kapoor, S. (2018). Progressive Web Apps 102: Building a Progressive Web App from scratch. Retrieved from https://medium.freecodecamp.org/progressive-web-apps-102-building-a-progressive-web-app-from-scratch-397b72168040
- Smirnova, N. (2017). Creating PWA with Angular 5. Part 2: Progressifying the application. Retrieved from https://medium.com/@nsmirnova/creating-pwa-with-angular-5-part-2-progressifying-the-application-449e3a706129


Power of PWA
Speed, Reliability, Integrated Experience, Engagement, Costs & Limitations

Reasons for network failure or slow network

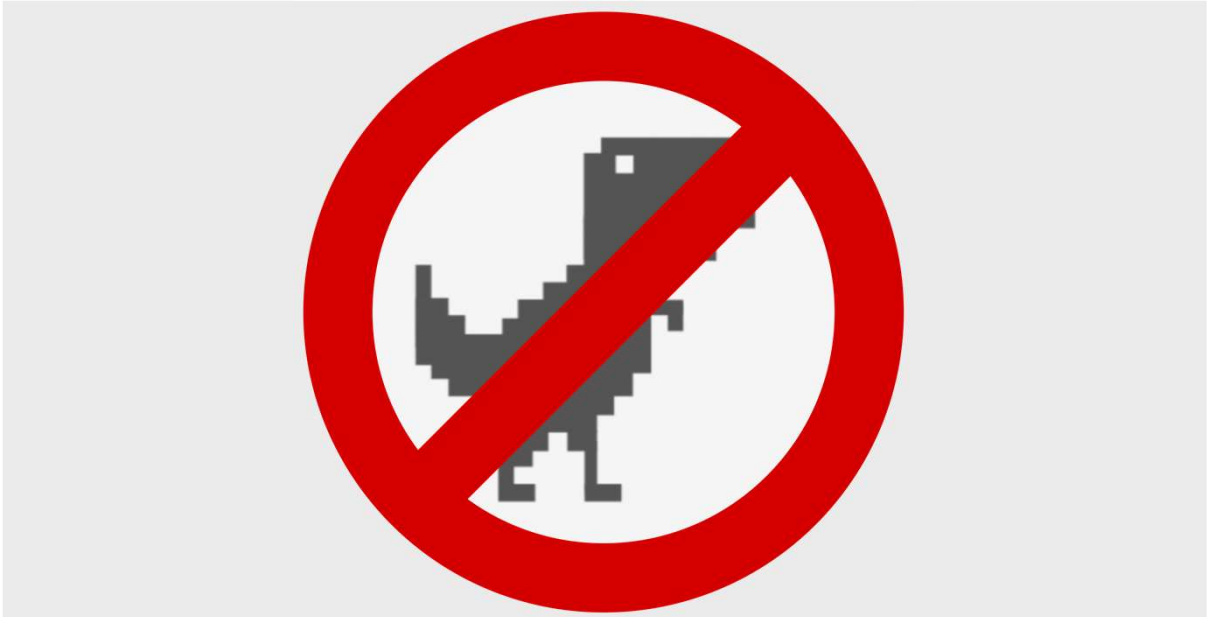
- 60% of the world's population is still using 2G
- Poor Signal
- Misconfigured Proxy
- Faults in the mobile network
- Busy network
- Server being DDoSed
- Bugs in the server
- Moon's Gravitational Pull (at least arguable ;))

(Kapoor, 2018a; Salman, 2018)

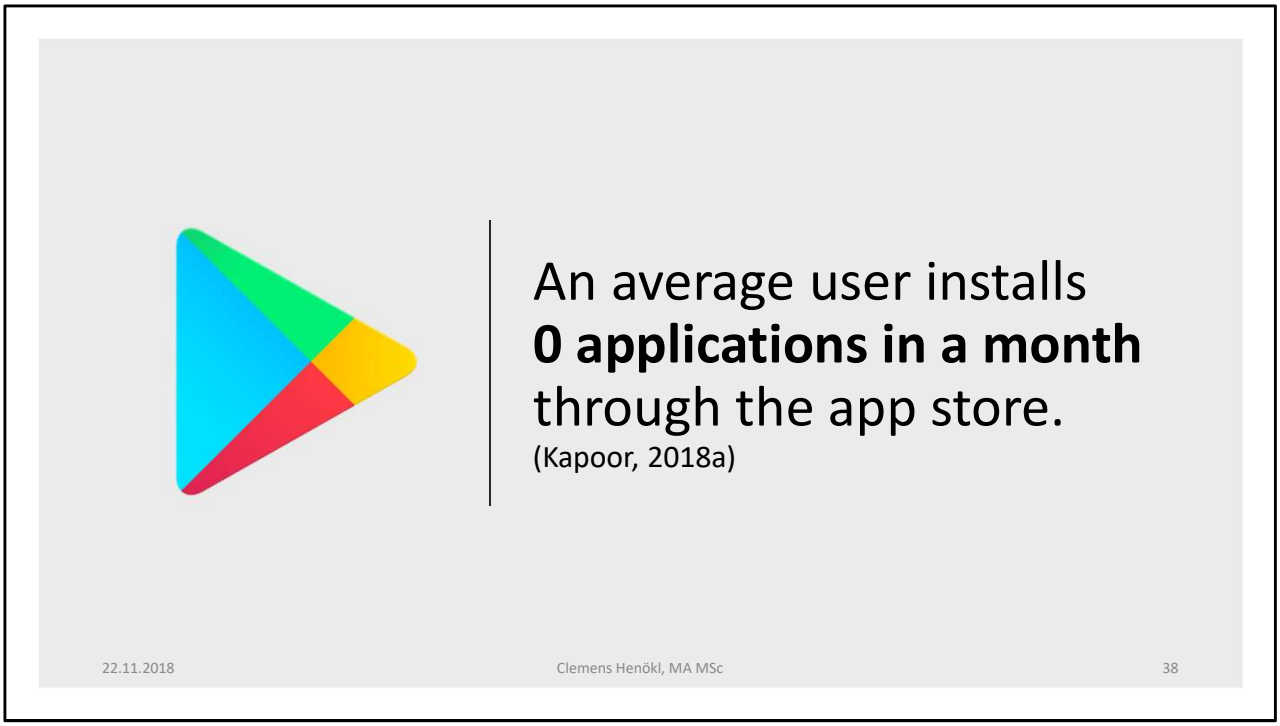
Fast & Reliable

- PWAs caching approach allows to load the app very fast even without network.
- On network failure at least basic app functionality will be provided.
- Content can be stored and displayed as a fallback.

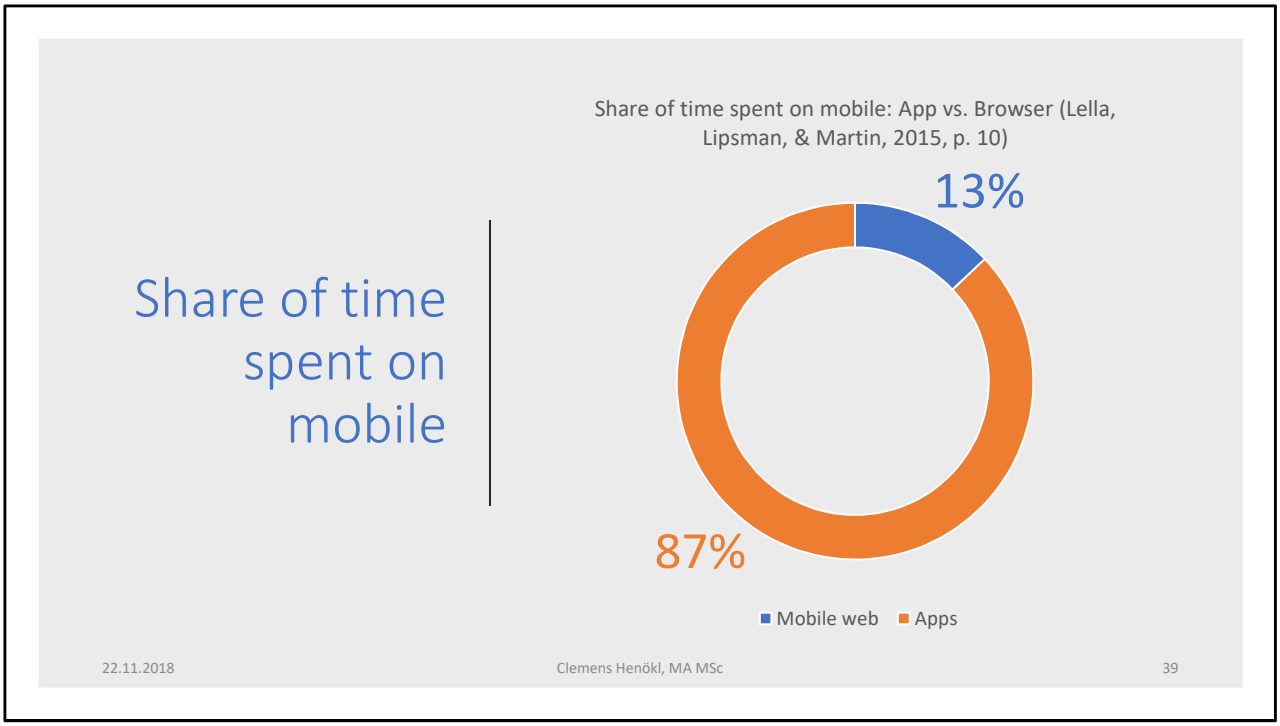
(Kapoor, 2018a, Salman, 2018)



No more dinosaur will be shown if the web app is started in offline mode or on network failure (Kapoor, 2018a, Salman, 2018).



This is concerning mature users who are using their smartphone for a long time. Nevertheless, most of the users are mature users. So it comes down to the point, that most of smartphone users spent 80% of the time on the top three apps on their phone. This situation makes it really hard to publish apps via the app store with a big reach to the users (Google Chrome Developers, 2018).



Most of the time spent on mobile phones is related to app usage whereas browser usage (websites) is quite limited. Which is related to the fact, that users have the perception that apps are more reliable than websites. Especially, when slow network is available (Google Chrome Developers, 2018).

Integrated user experience & engagement

- Feels like native apps
- Placed on the home screen
- Send push notifications
- Access to device's functionalities (e.g.: camera, microphone and further)
- Seamless and integrated experience

(Google Chrome Developers, 2018; Kapoor, 2018a; LePage, 2018)

PWAs want to change the afore mentioned perception with making web apps feel like native apps. This will be reached with the home screen icon, push notifications, access to device functionality and seamless integration to the operating system (e.g.: The task manager in android).

Native app development and distribution is **expensive**.

(Dixit, 2018)

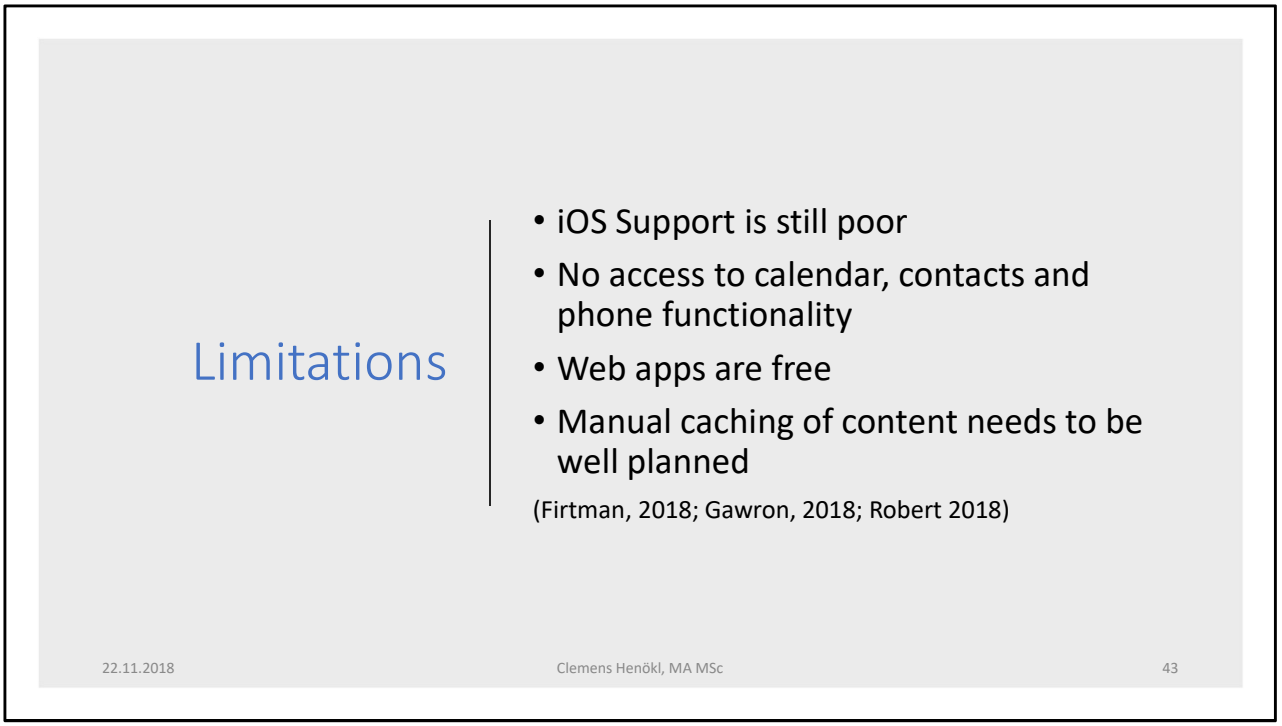
Reduction of Costs

- Average salary of web developers are lower than application developers
- Cross platform development
- Use of open-source frameworks
- Easy and fast to distribute updates
- Customer acquisition costs (CRC) can shrink by as much as 10 times
- Improved conversion rate

(Ajani, 2018; Dixit, 2018, Gawron, 2018; Kapoor, 2018a)

It is hard to get user install your app through the app store. Hereby, lots of consumer education with marketing activities has to be done. Furthermore development costs are higher for native apps because of cross platform development, license costs and time consuming review processes for different app stores. Especially when updates are distributed. Finally, finding specialized developers for different platforms is difficult and full-stack app developers are more expensive than web developers (Ajani, 2018; Dixit, 2018, Gawron, 2018; Kapoor, 2018a).

.



iOS does not support
- no push notifications
- no splash screen,
- lack of support for other browsers than Safari
- No install prompt

(Firtman, 2018; Gawron, 2018)

Moreover, it has to be kept in mind that web apps cannot be monetarized like apps from the app store. There needs to be a different approach for monetarization. This means basically, it won't suit to every business model and needs to be included in defining a PWA strategy (Robert 2018).

Conclusion

- Good opportunity to
  - add further value to your website
  - move closer to the customer
  - and to grow revenue and lower costs
- Strong support by Google
- Most of the browsers support at least the basic technology
- Applicable in the field of regularly updated content (events, blogs, news,…)


Any Questions left?
I‘m happy to start a little discussion with you!

Resources

- Contact Information
  - Clemens Henökl
    Email: clemens.henoekl@advokat.at
    LinkedIn: https://www.linkedin.com/in/clemens-henökl-789843134/
- GitHub Repo
  - https://github.com/chenoekl/web-and-speck-pwa
  - Includes the Angular 6 test app with service worker usage
  - Full presentation with notes
- Test App
  - https://spielwiese.jusline.at

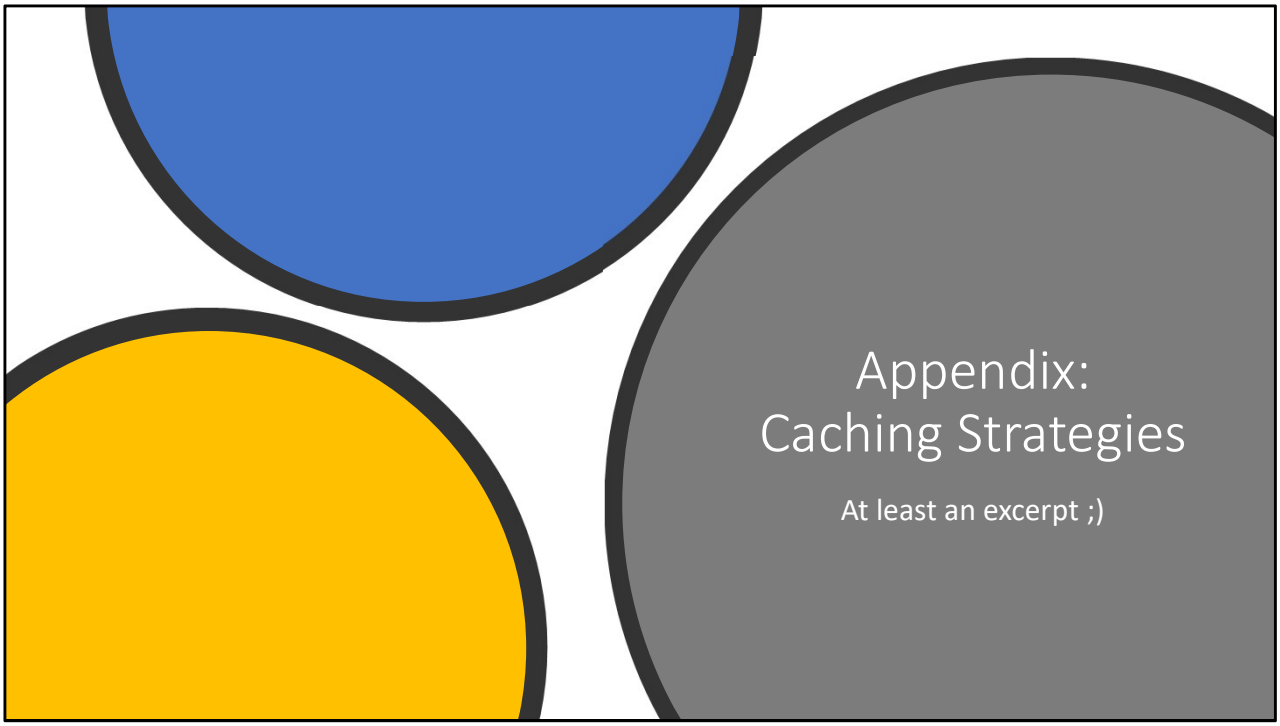
References (I)

- Ajani, N. (2018). How progressive web app can lower the development cost? Retrieved from https://www.atyantik.com/how-progressive-web-app-can-lower-the-development-cost/
- Angular University. (2018). Angular Service Worker: Step-By-Step Guide for turning your Application into a PWA. Retrieved from https://blog.angular-university.io/angular-service-worker/
- Archibald, J. (2018). The Offline Cookbook. Retrieved from https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook
- Archibald, J. (2018). The Service Worker Lifecycle. Retrieved from https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle
- Dixit, P. (2018). Top 12 reasons to develop a progressive web app. Retrieved from https://www.tristatetechnology.com/blog/top-12-reasons-develop-progressive-web-app/
- Firtman, M. (2018). Progressive Web Apps on iOS are here. Retrieved from https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7
- Gaunt, M. (2018). Service Workers: an Introduction. Retrieved from https://developers.google.com/web/fundamentals/primers/service-workers/
- Gawron, K. (2018). "Should I Consider PWA?" Remarkable Possibilities of Progressive Web Apps. Retrieved from https://www.monterail.com/blog/should-i-consider-pwa-remarkable-possibilities-of-progressive-web-apps
- Google Chrome Developers. (2018). *Progressive Web Apps - PWA Roadshow* [YouTube]. Retrieved from https://www.youtube.com/playlist?list=PLNYkxOF6rcICnIOm4cfylT0-cEfytBtYt
- Google Developers. (2018). Lighthouse. Retrieved from https://developers.google.com/web/tools/lighthouse/

References (II)

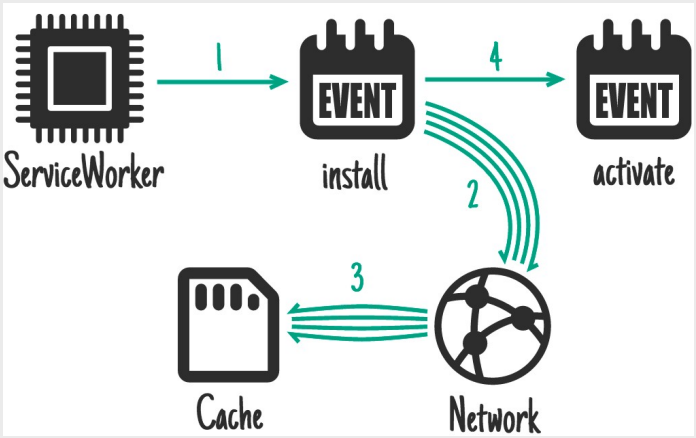
- Kapoor, S. (2018). Progressive Web Apps 101: the What, Why and How: What is a Progressive Web App? Why do we need one? How can we build one? Retrieved from https://medium.freecodecamp.org/progressive-web-apps-101-the-what-why-and-how-4aa5e9065ac2
- Kapoor, S. (2018). Progressive Web Apps 102: Building a Progressive Web App from scratch. Retrieved from https://medium.freecodecamp.org/progressive-web-apps-102-building-a-progressive-web-app-from-scratch-397b72168040
- Lella, A., Lipsman, A., & Martin, B. (2015). *The 2015 U.S. Mobile App Report*. Retrieved from comScore, Inc. website: https://www.liftpointconsulting.com/wp-content/uploads/2015/11/2015_US_Mobile_App_Report-from-comscore.pdf
- LePage, P. (2018). Your First Progressive Web App. Retrieved from https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/
- Osmani, A., & Cohen, M. (2018). Offline Storage for Progressive Web Apps. Retrieved from https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa
- Robert. (2018). Progressive Web Apps (PWA) – Was ist das überhaupt und wie nutzt man sie? Retrieved from https://apptooltester.com/de/progressive-web-apps/
- Salman, A. (2018). Why should you care about Progressive Web Apps (PWAs)? Retrieved from https://codeburst.io/why-should-you-care-about-progressive-web-apps-pwas-3c3f73cb8c92
- Smirnova, N. (2017). Creating PWA with Angular 5. Part 2: Progressifying the application. Retrieved from https://medium.com/@nsmirnova/creating-pwa-with-angular-5-part-2-progressifying-the-application-449e3a706129
- Statcounter. (2018). GlobalStats: Browser Market Share. Retrieved from http://gs.statcounter.com/
- Vaadin. (2018). Progressive Web App Browser Support. Retrieved from https://vaadin.com/pwa/learn/browser-support
- Wagner, J. (2018). Why Performance Matters. Retrieved from https://developers.google.com/web/fundamentals/performance/why-performance-matters/


Thank you for your attention!


Appendix:
Caching Strategies
At least an excerpt ;)

On install - as a dependency

Ideal for: Static files such as CSS, images, fonts, JS,…



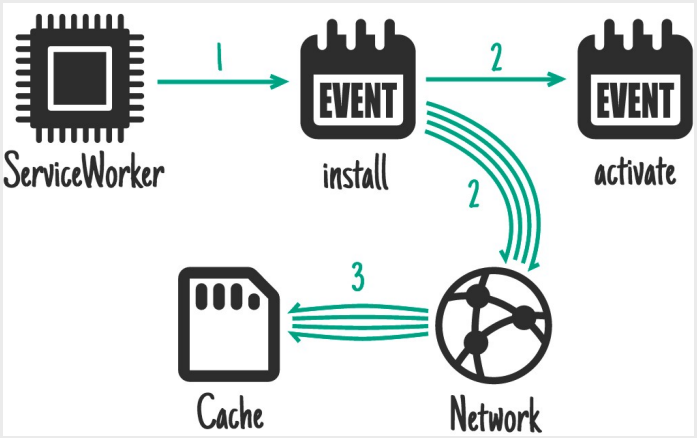
Service Worker code example (Archibald, 2018a)

- ServiceWorker gives you an install event. You can use this to get stuff ready, stuff that must be ready before you handle other events. While this happens any previous version of your ServiceWorker is still running & serving pages, so the things you do here mustn't disrupt that.
- These are things that would make your site entirely non-functional if they failed to fetch, things an equivalent native-app would make part of the initial download.

(Archibald, 2018a)

On install - not as a dependency

Ideal for: Bigger resources that aren't needed straight away



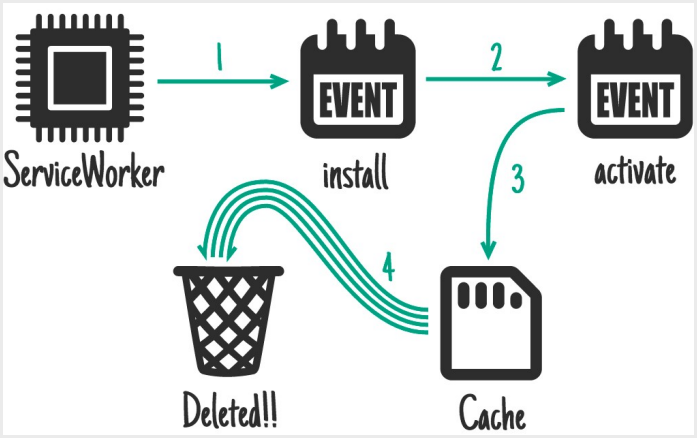
Service Worker code example (Archibald, 2018a)

- Similar to above, but won't delay install completing and won't cause installation to fail if caching fails.
- We're not passing the cache.addAll promise for levels 11-20 back to event.waitUntil, so even if it fails, the game will still be available offline. Of course, you'll have to cater for the possible absence of those levels & reattempt caching them if they're missing.
- The ServiceWorker may be killed while levels 11-20 download since it's finished handling events, meaning they won't be cached. In future we plan to add a background downloading API to handle cases like this, and larger downloads such as movies.

(Archibald, 2018a)

On Activate

Ideal for: Clean-up & migration



Service Worker code example (Archibald, 2018a)

- Once a new ServiceWorker has installed & a previous version isn't being used, the new one activates, and you get an activate event. Because the old version is out of the way, it's a good time to handle schema migrations in IndexedDB and also delete unused caches.
- During activation, other events such as fetch are put into a queue, so a long activation could potentially block page loads. Keep your activation as lean as possible, only use it for things you couldn't do while the old version was active.

(Archibald, 2018a)

More information on this

Lots of more information to caching strategies and coding snippets can be found here:

- Archibald, J. (2018a). The Offline Cookbook. Retrieved from https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook
- Osmani, A., & Cohen, M. (2018). Offline Storage for Progressive Web Apps. Retrieved from https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa