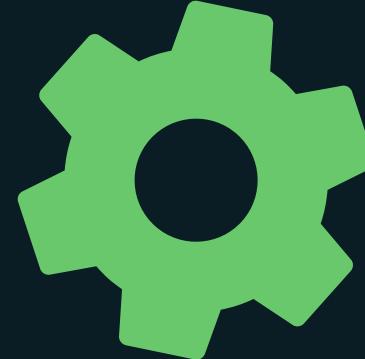


Progressive Web Apps

Manifest & Service Worker API



<https://giftkugel.github.io/talks/>
or
<https://bit.ly/2JdoLoG>

About me

Simon Skoczylas
Senior Software Engineer
Karakun.



 *giftkugel*

 www.skoczylas.net



Karakun Developer Hub

<https://dev.karakun.com>



Karakun DevHub_

@*giftkugel*

Progressive

Add functionality step by step

Web

Web standards, high reach and best availability

Applications

Can be installed, network independent, secure



Google PWA labs definition 1/2

<https://codelabs.developers.google.com/codelabs/your-first-pwapp>

Progressive

Works for every user, regardless of browser choice because it's built with progressive enhancement as a core tenet.

Responsive

Fits any form factor: desktop, mobile, tablet, or whatever is next.

Connectivity independent

Enhanced with service workers to work offline or on low-quality networks.

App-like

Feels like an app, because the app shell model separates the application functionality from application content.

Fresh

Always up-to-date thanks to the [service worker](#) update process.



Google PWA labs definition 2/2

<https://codelabs.developers.google.com/codelabs/your-first-pwapp>

Safe

Served via HTTPS to prevent snooping and to ensure content hasn't been tampered with.

Discoverable

Is identifiable as an "application" thanks to [W3C manifest](#) and [service worker registration](#) scope, allowing search engines to find it.

Re-engageable

Makes re-engagement easy through features like push notifications.

Installable

Allows users to add apps they find most useful to their home screen without the hassle of an app store.

Linkable

Easily share the application via URL, does not require complex installation.



Why?

Worthy of being on the home screen

Work reliably, no matter the network conditions

Increased engagement

Improved conversions

And at least ... because of the Web 😊 😃



PWA Case Studies

<https://developers.google.com/web/showcase/>



PWA or native application?

No competition because of different purposes
Go for a native app, if web standards don't fit your needs

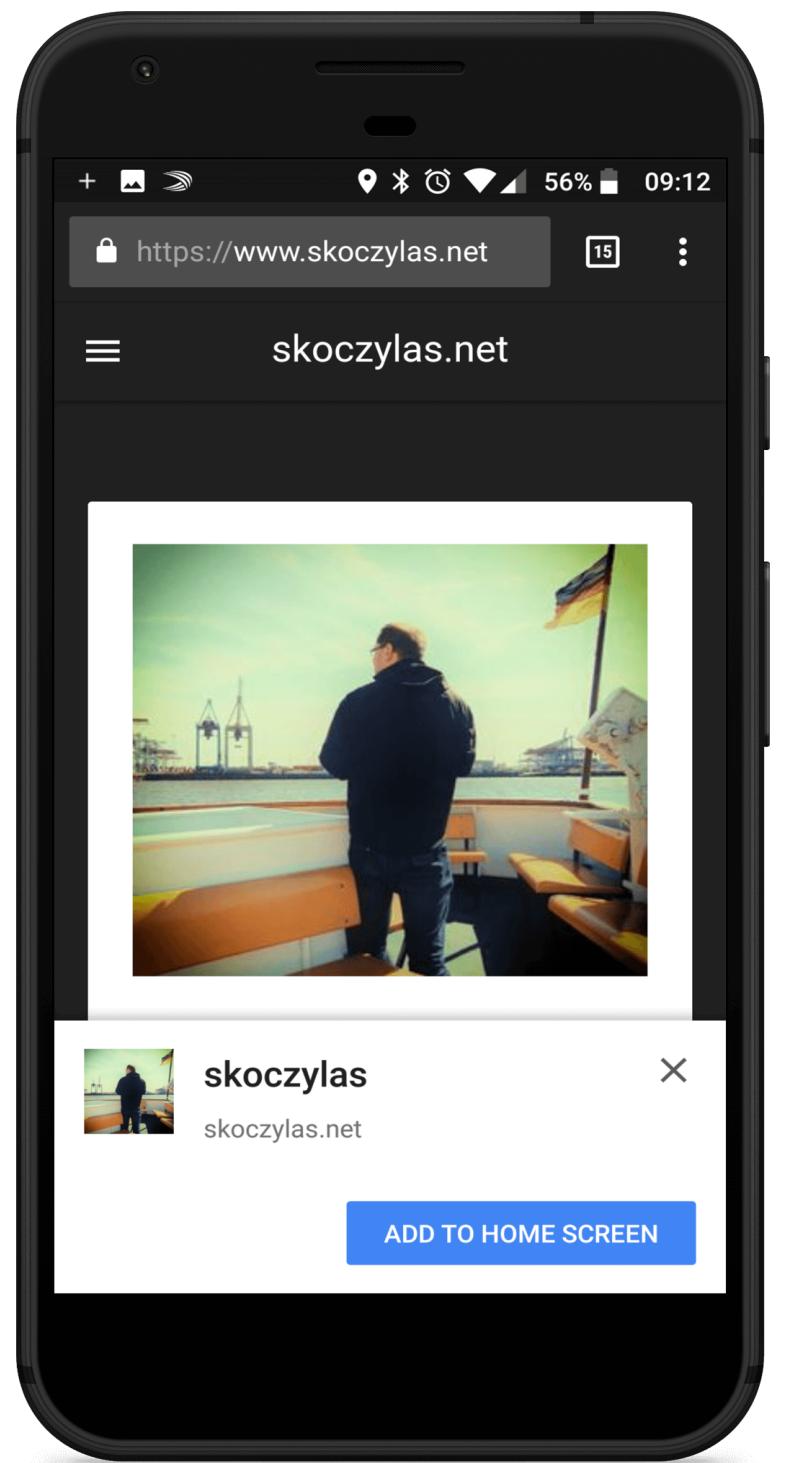
Do you really know all [Web APIs](#)?



Example

Take a look at my PWA
<https://www.skoczylas.net>





skoczylas
skoczylas.net

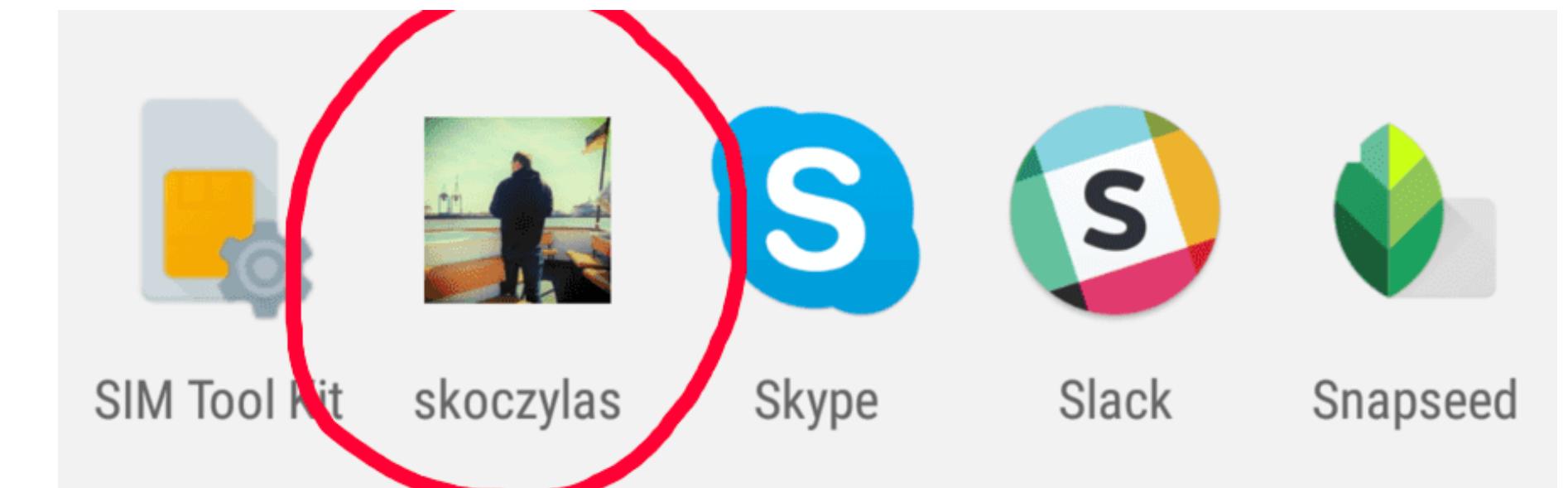
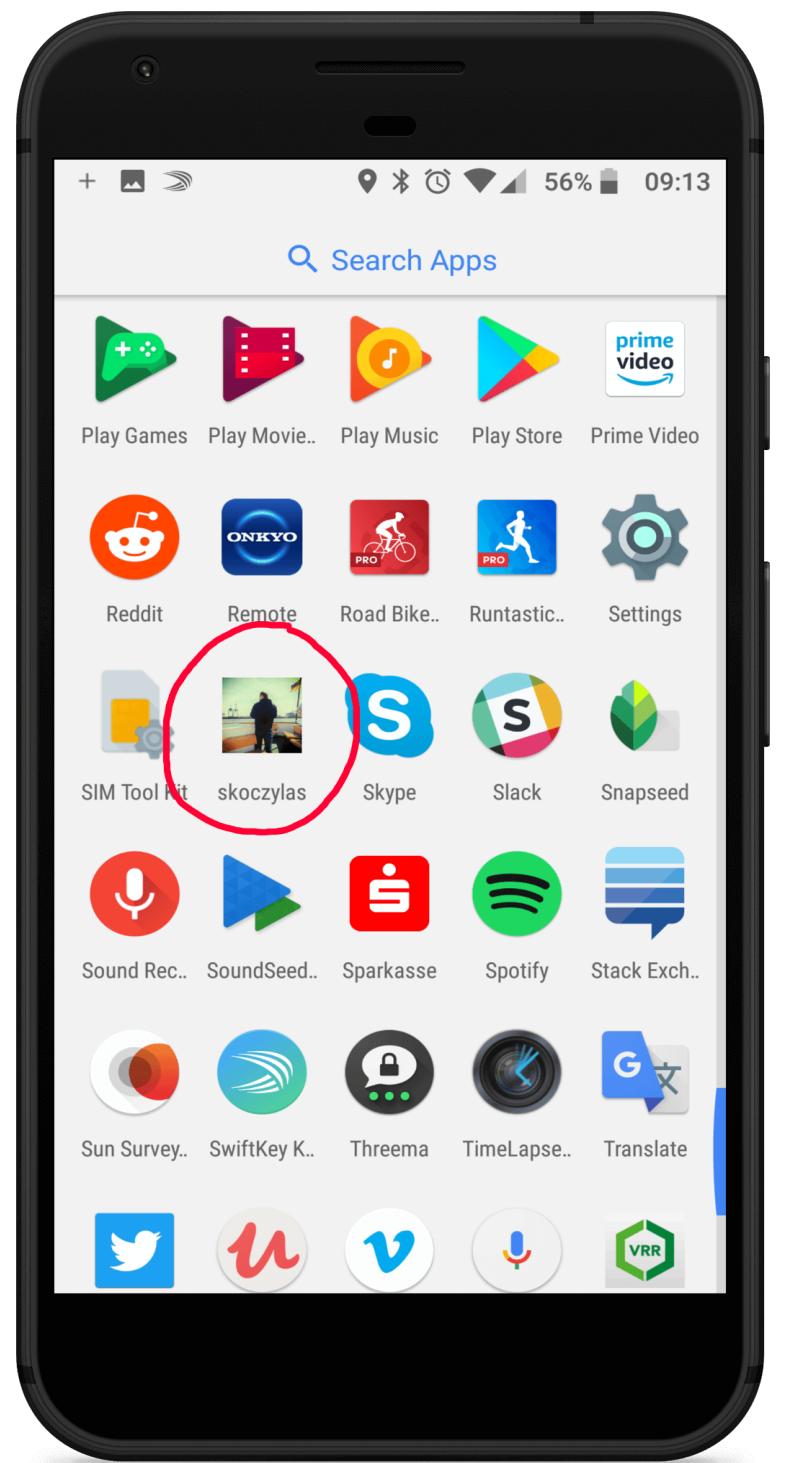
ADD TO HOME SCREEN

Add to home screen



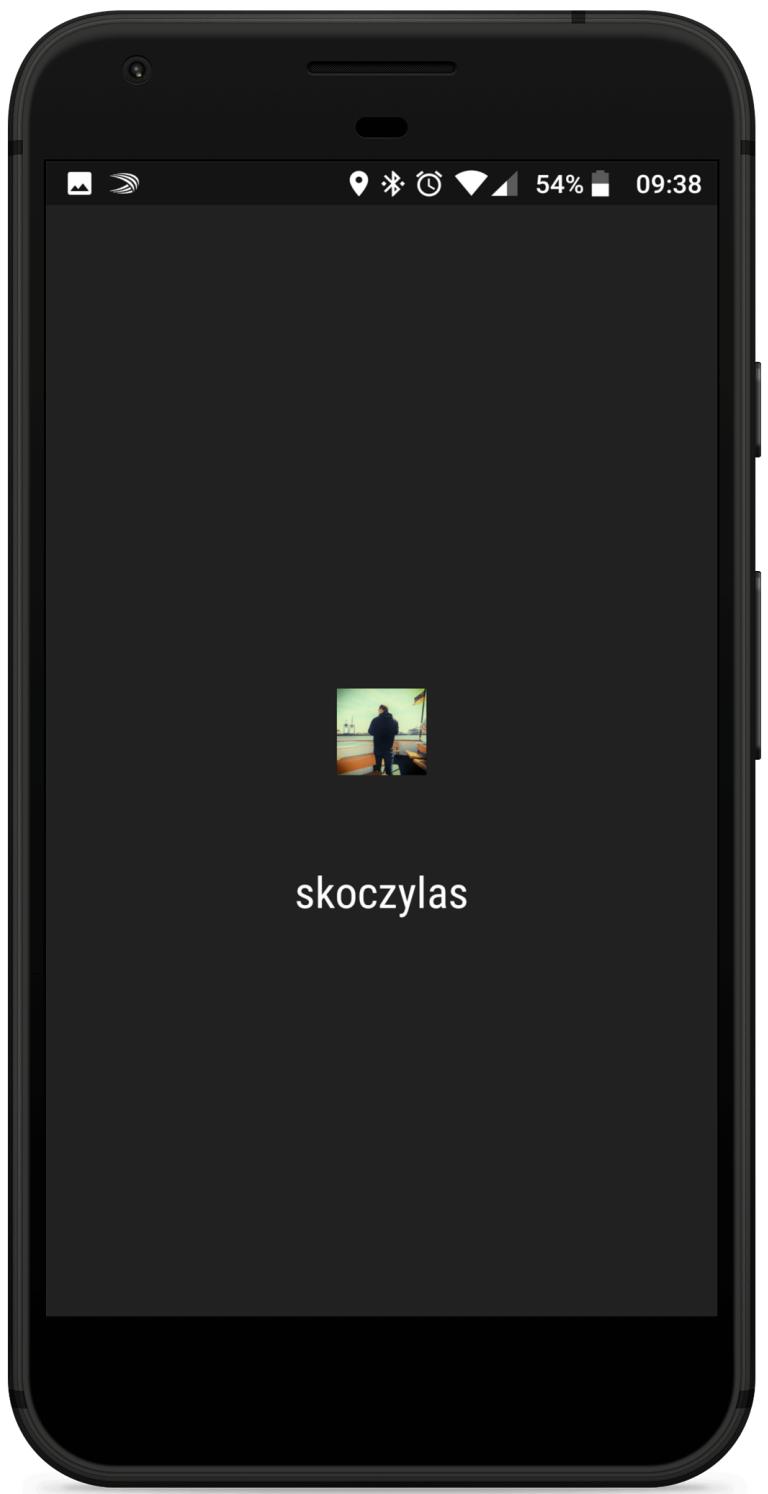
Karakun DevHub_

@gftkugel



Listed as application in the launcher



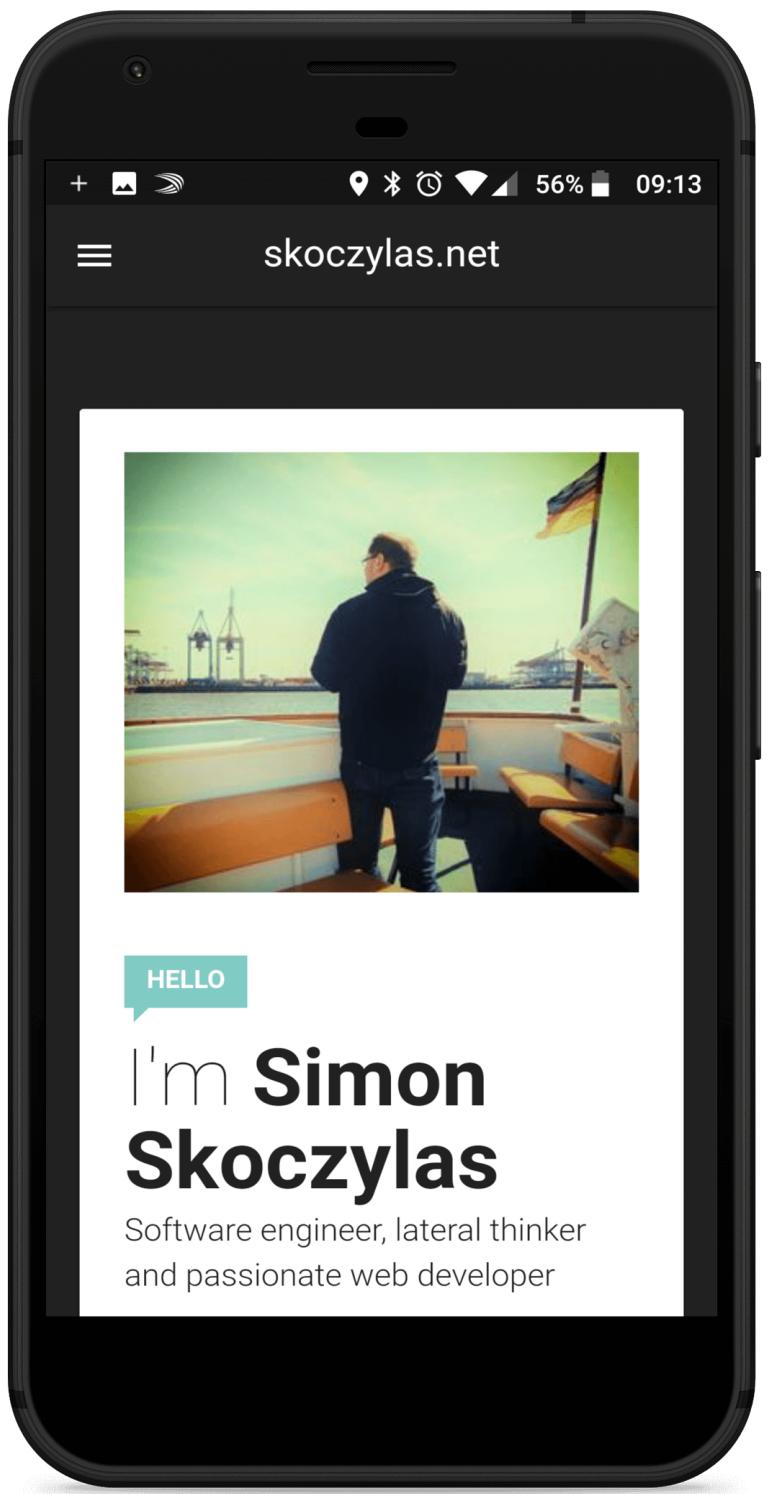


Splashscreen



Karakun DevHub_

@gftkugel

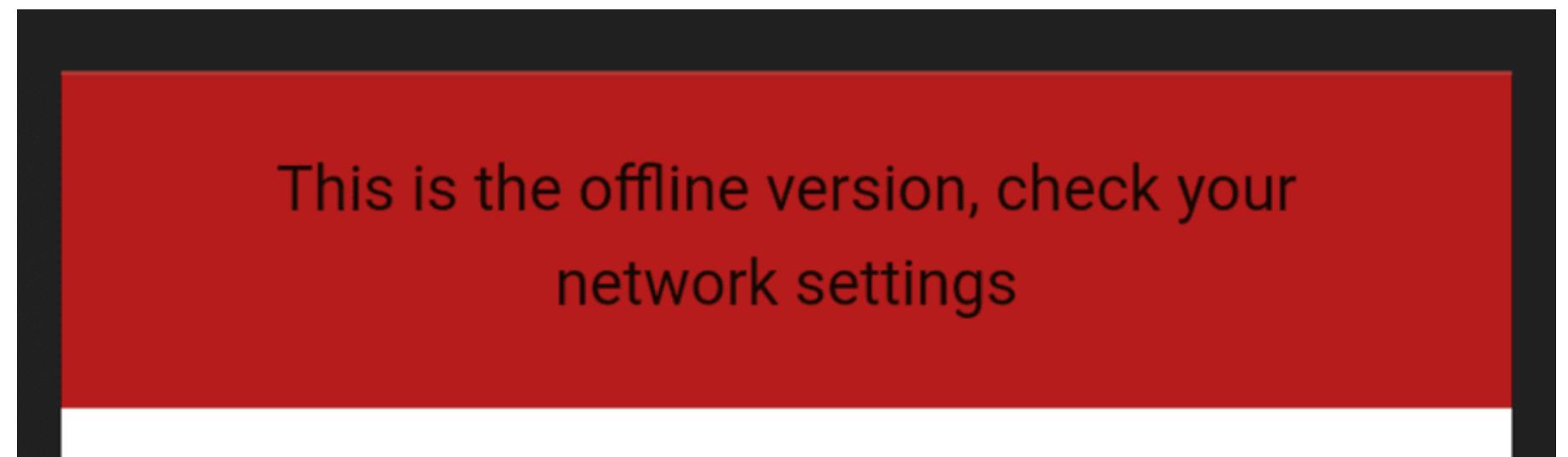
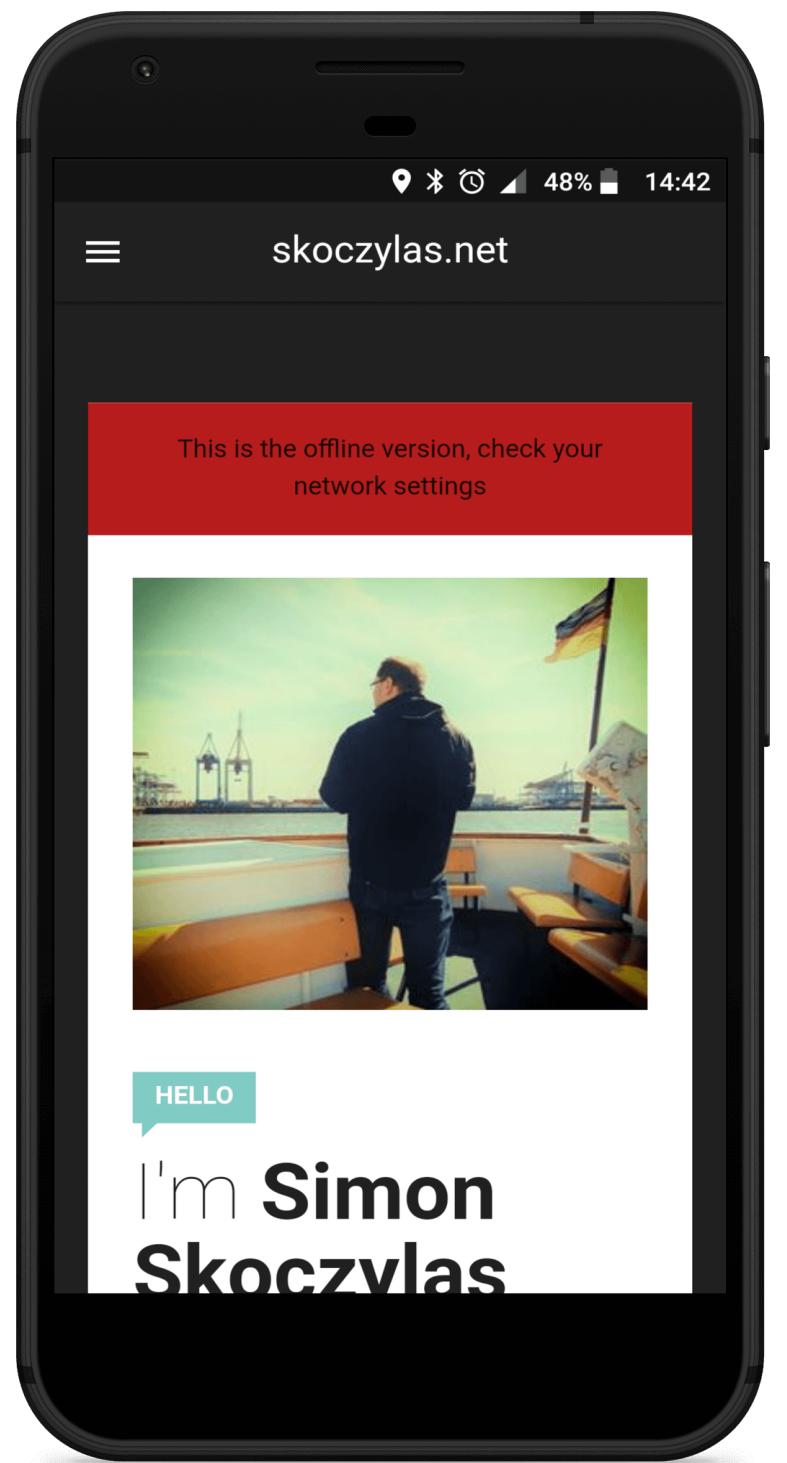


No browser visible



Karakun DevHub_

@gftkugel

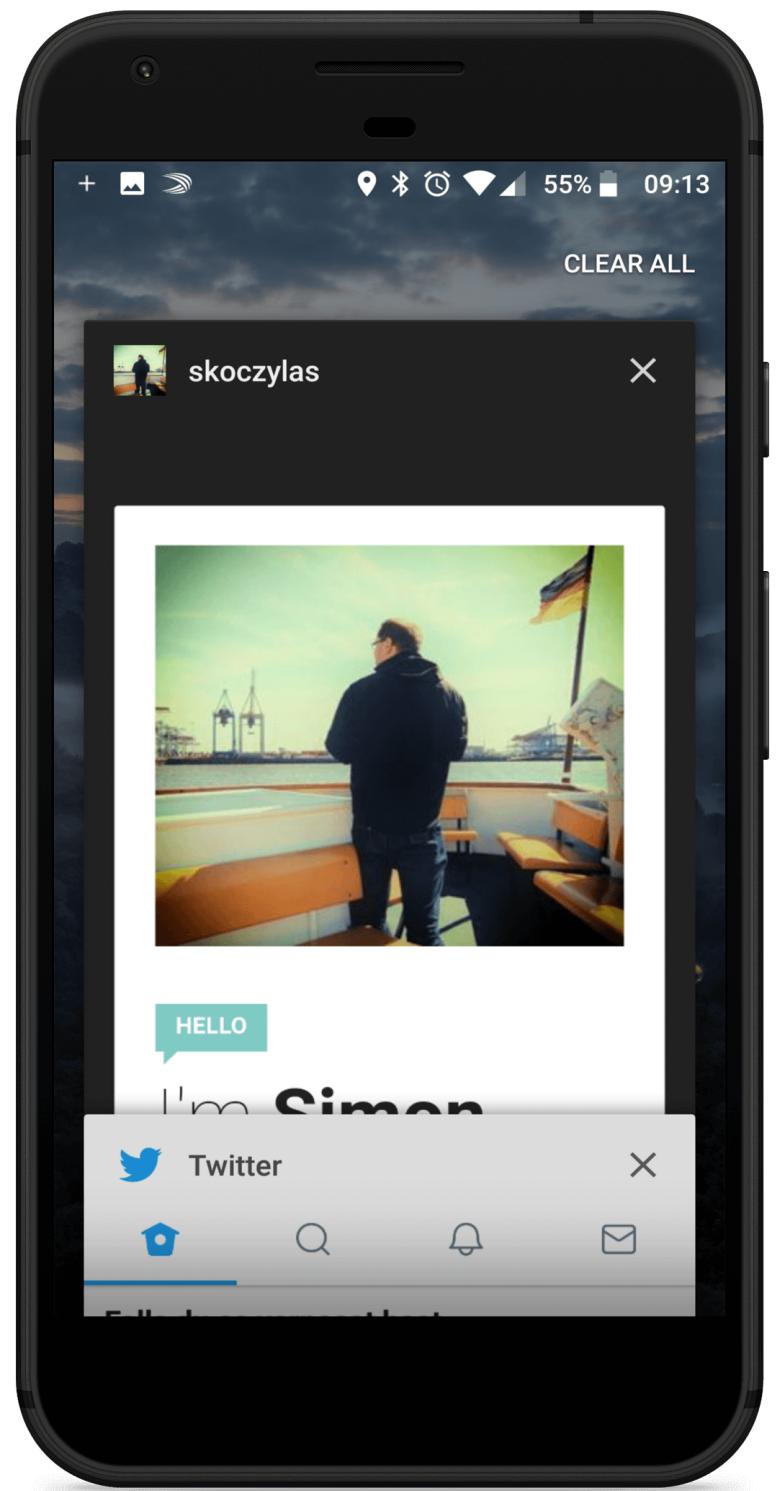


Behaves differently when offline



Karakun DevHub_

@gftkugel



Listed as currently running application



Karakun DevHub_

@gftkugel

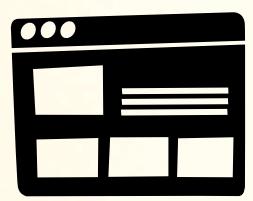
Let's build a PWA

In just a few steps



Karakun DevHub_

@gftkugel



Got a website?

First step accomplished!





Using HTTPS?

Perfect, keep going!



Take the next step

Add a manifest

Web App Manifest

JSON file that provides information about your application

<https://w3c.github.io/manifest/>

General information
name, short_name, description

Colors
background_color, theme_color

Icons
Array of Icons with *src, sizes, type*

and more...
lang, orientation, display, scope, start_url, dir, related_applications



Deploy a manifest

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

Content-Type
application/manifest+json 🤔
application/json



Web App Manifest

Example

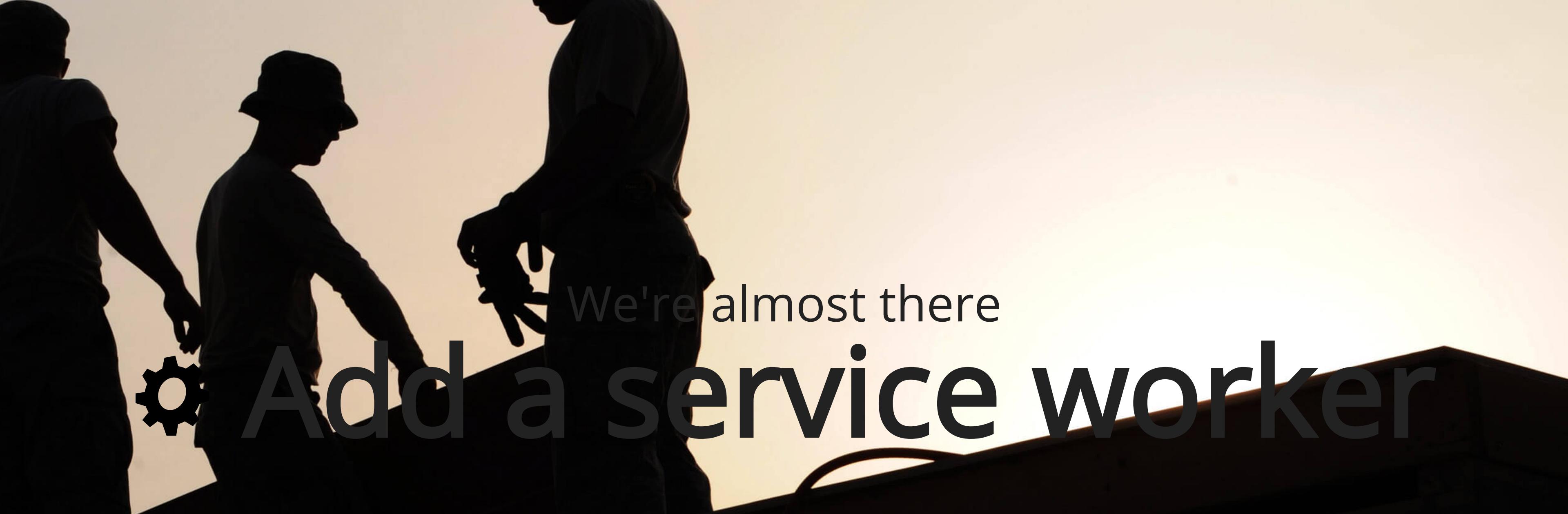
```
{  
  name: "skoczylas",  
  short_name: "skoczylas",  
  icons: [  
    {  
      src: "img/fav/android-icon-36x36.png",  
      sizes: "36x36",  
      type: "image/png",  
      density: "0.75"  
    },  
    {  
      src: "img/fav/android-icon-48x48.png",  
      sizes: "48x48",  
      type: "image/png",  
      density: "1.0"  
    }  
  ],  
  lang: "en",  
  start_url: "index.html",  
  display: "standalone",  
  orientation: "portrait",  
  theme_color: "#212121",  
  background_color: "#212121"  
}
```





Add a service worker

We're almost there



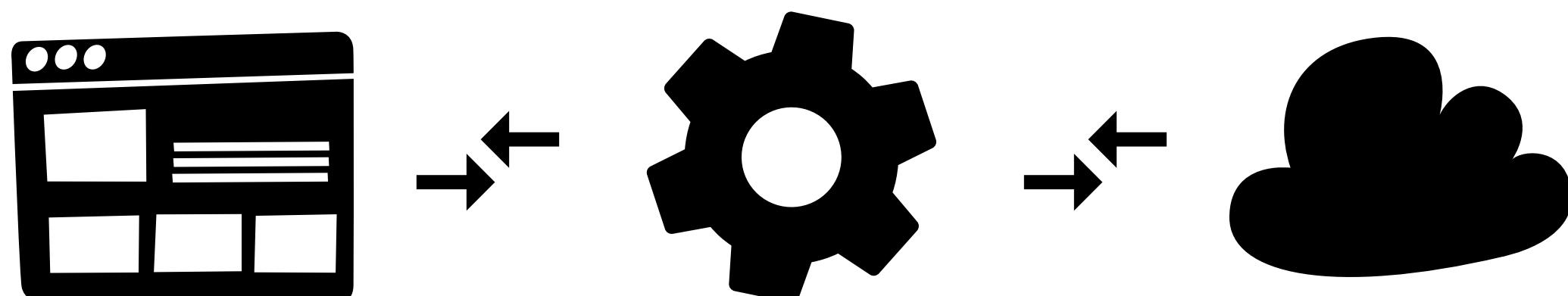
Service Worker API

Service Worker is a script that runs in the background
A special Web Worker (Thread)

As a Worker it can't access the DOM directly

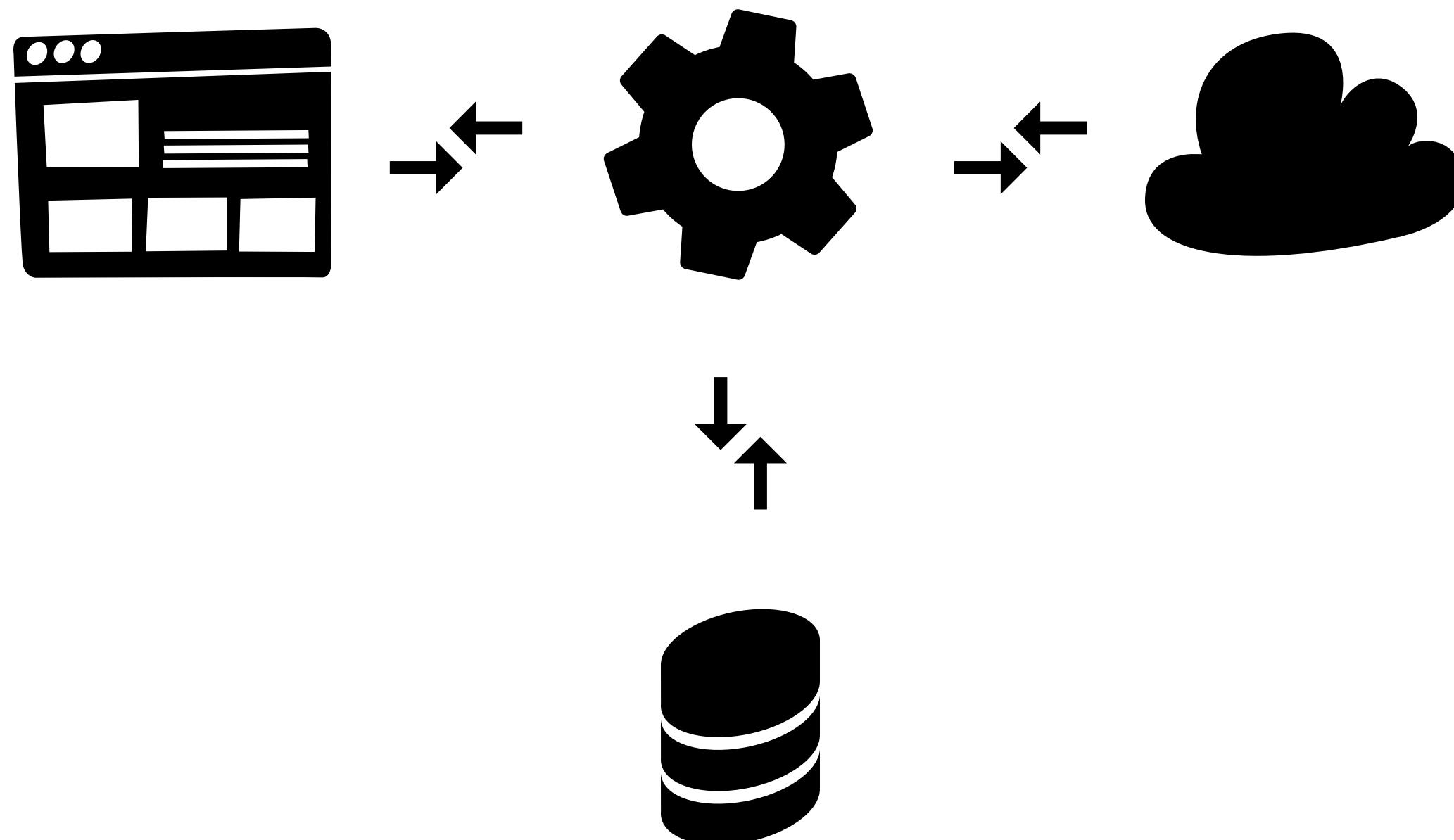
Can communicate with the page via the [postMessage](#) interface

It's a programmable network proxy, allowing you to control how network requests are handled



Service Worker API

It can access the Cache API to read and write assets



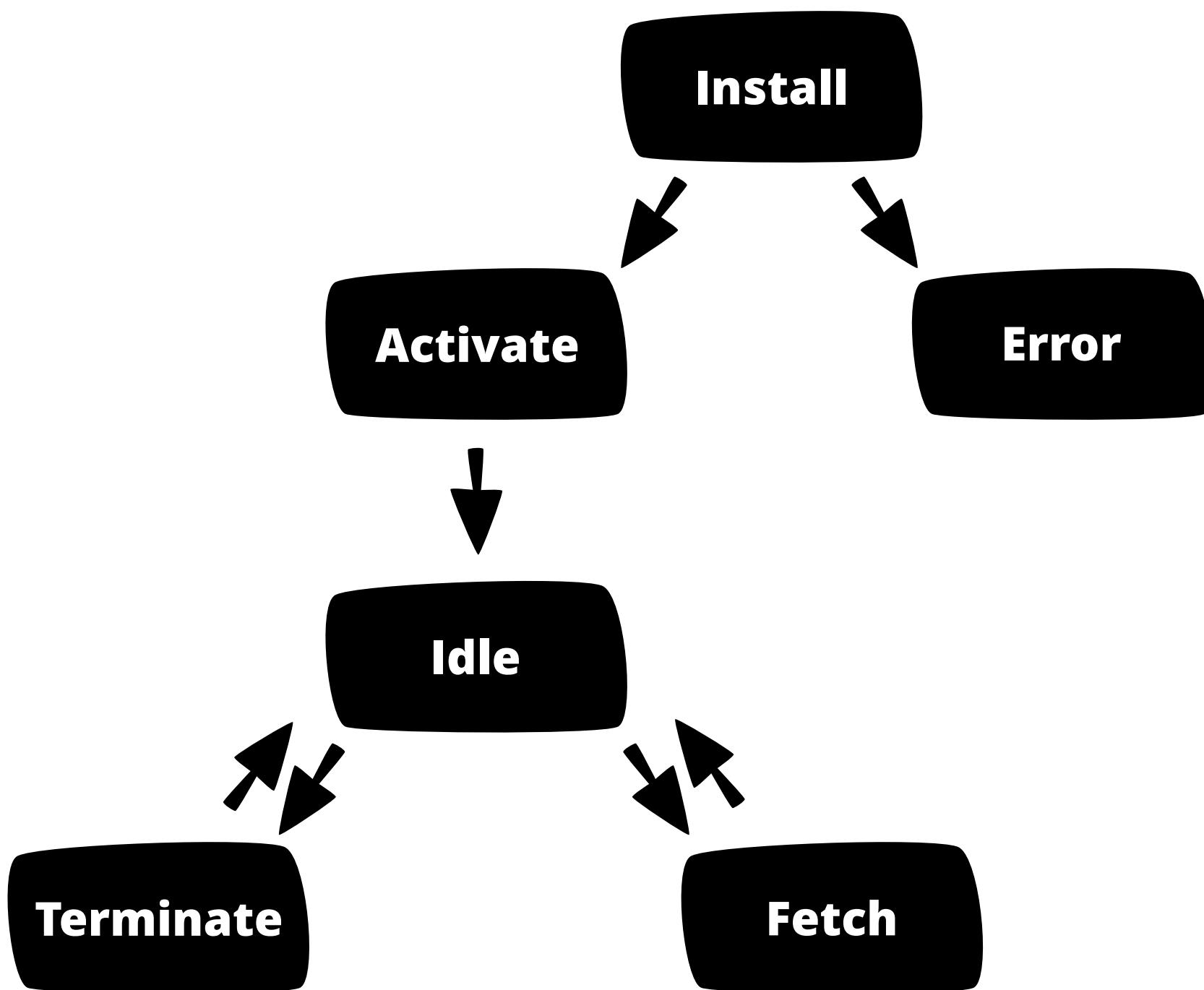
Requires HTTPS

Works without HTTPS on localhost (for development)

Requirement can be disabled through browser settings 😊



Lifecycle



The Navigator object

Register your service worker

```
if ('serviceWorker' in navigator) {  
  navigator.serviceWorker.register('/sw.js');  
}
```

Easy 😍



The Navigator object

Maybe add a scope, wait for an event and handle result?

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/sw.js', {scope: './'})
      .then((registration) => { ... })
      .catch((error) => { ... });
  });
}
```

ServiceWorkerRegistration



this and self

Both points to the [ServiceWorkerGlobalScope](#)

Use *self*, because you never get a guarantee about *this* 😱



Service Worker Events

```
self.addEventListener('install', (event) => { ... }); // InstallEvent  
  
self.addEventListener('activate', (event) => { ... }); // ExtendableEvent  
  
self.addEventListener('message', (event) => { ... }); // MessageEvent
```

aka ExtendableEvent



Service Worker Functional Events

```
self.addEventListener('fetch', (event) => { ... }); // FetchEvent  
  
self.addEventListener('sync', (event) => { ... }); // SyncEvent  
  
self.addEventListener('push', (event) => { ... }); // PushEvent
```

aka ExtendableEvent



It's my turn now!

```
self.skipWaiting();
```

Skip waiting on install event (be aware!)



I am working, don't terminate me...

```
event.waitUntil(promise);
```

on ExtendableEvent



Store assets in cache

```
event.waitUntil(  
  self.caches.open(name)  
  .then(cache => cache.addAll([ ... ]))  
) ;
```

Cache stuff during installation (App shell)



Get asset from cache

```
event.respondWith(  
  self.caches.match(event.request)  
  .then((response) => {  
    if (response) {  
      return response; // cached response  
    }  
  
    const fetchRequest = event.request.clone();  
  
    return fetch(fetchRequest); // network response  
  })  
);
```

Check cache on fetch event



Get asset from cache, or store missing asset in cache

```
event.respondWith(  
  self.caches.match(event.request)  
  .then((response) => {  
    if (response) {  
      return response; // cached response  
    }  
    const fetchRequest = event.request.clone();  
    return fetch(fetchRequest).then((response) => {  
      if(!response || response.status !== 200 || response.type !== 'basic') {  
        return response; // network reponse which should not be cached  
      }  
      const responseToCache = response.clone();  
      self.caches.open(name)  
      .then((cache) => cache.put(event.request, responseToCache)); // update cache  
      return response; // network reponse  
    })  
  })  
)
```

Check cache on fetch event, refresh cache over network



Caching strategies

<https://serviceworkerjs.com/caching-strategies.html>

Network or cache

The service worker tries to retrieve the most up to date content from the network but if the network is taking too much, it will serve cached content instead.

Cache only

Always answering from cache on fetch events.



Caching strategies 2/2

Cache and update

Responding from cache to deliver fast responses and also updating the cache entry from the network.

Cache, update and refresh

Responding from cache to deliver fast responses and also updating the cache entry from the network. When the network response is ready, the UI updates automatically.

Embedded fallback

Serving an embedded content fallback in case of missing resources.

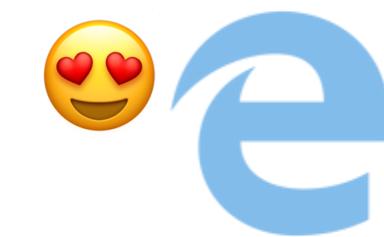
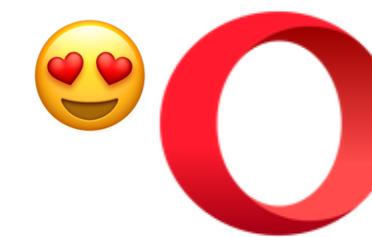
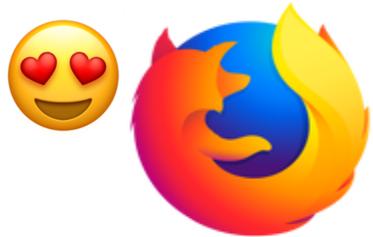
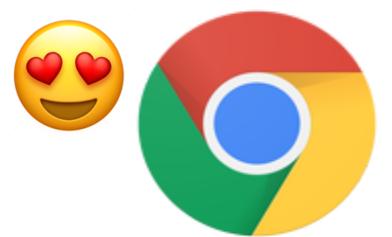
See also

<https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/>



Is Service Worker ready?

<https://jakearchibald.github.io/isserviceworkerready/>





Keep going further with
HTTP/2

Push notifications



Tools 1/2

Lighthouse 

<https://developers.google.com/web/tools/lighthouse/>

Workbox

<https://developers.google.com/web/tools/workbox/>

Webpack OfflinePlugin

<https://github.com/NekR/offline-plugin>

PWA Builder

<https://www.pwabuilder.com>

Service Worker Toolchain, you even get a Mock 

<https://github.com/pinterest/service-workers>



Tools 2/2

Service Worker Precache & Toolbox
<https://github.com/GoogleChromeLabs>

Create React App
<https://github.com/facebook/create-react-app>

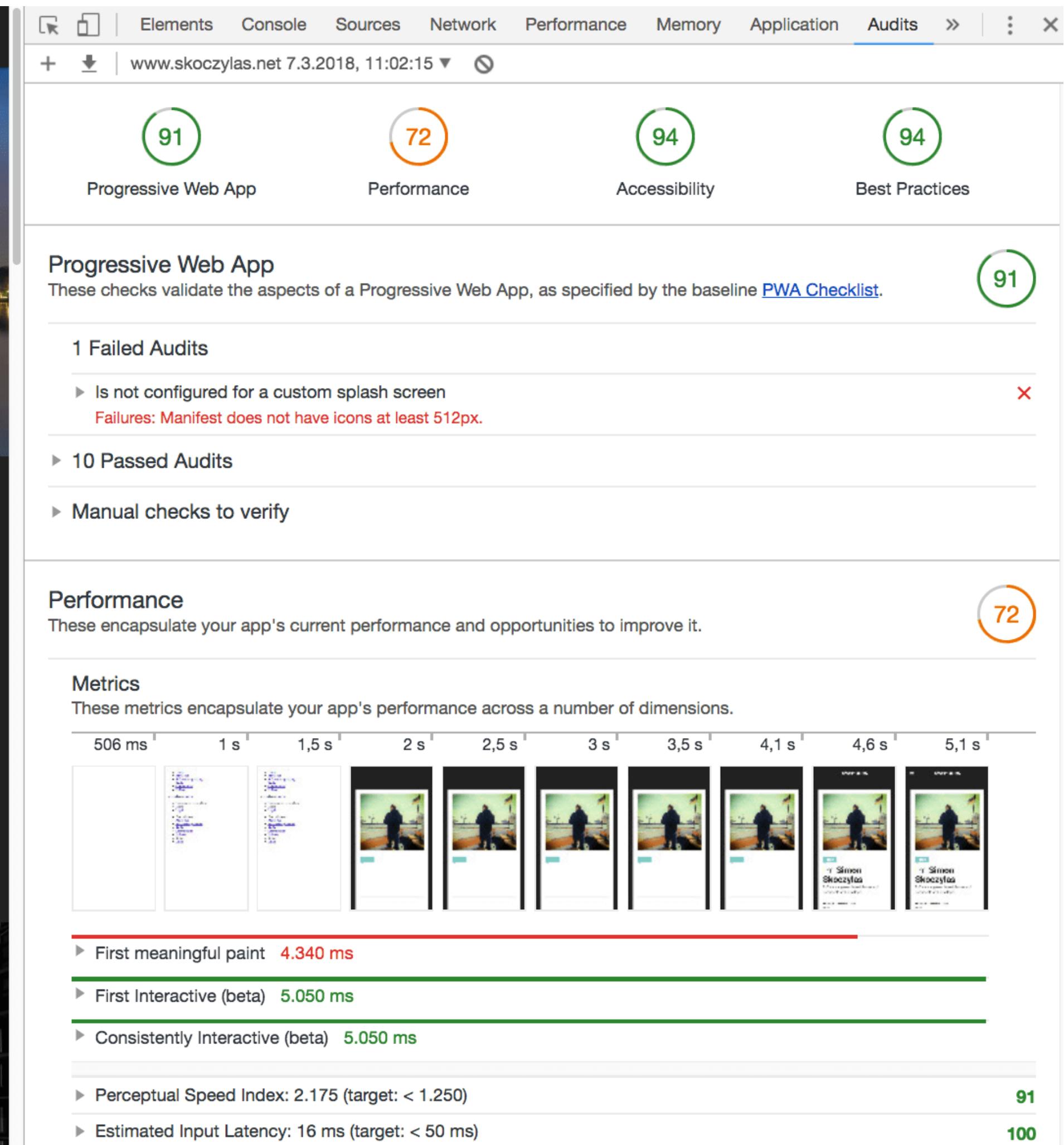
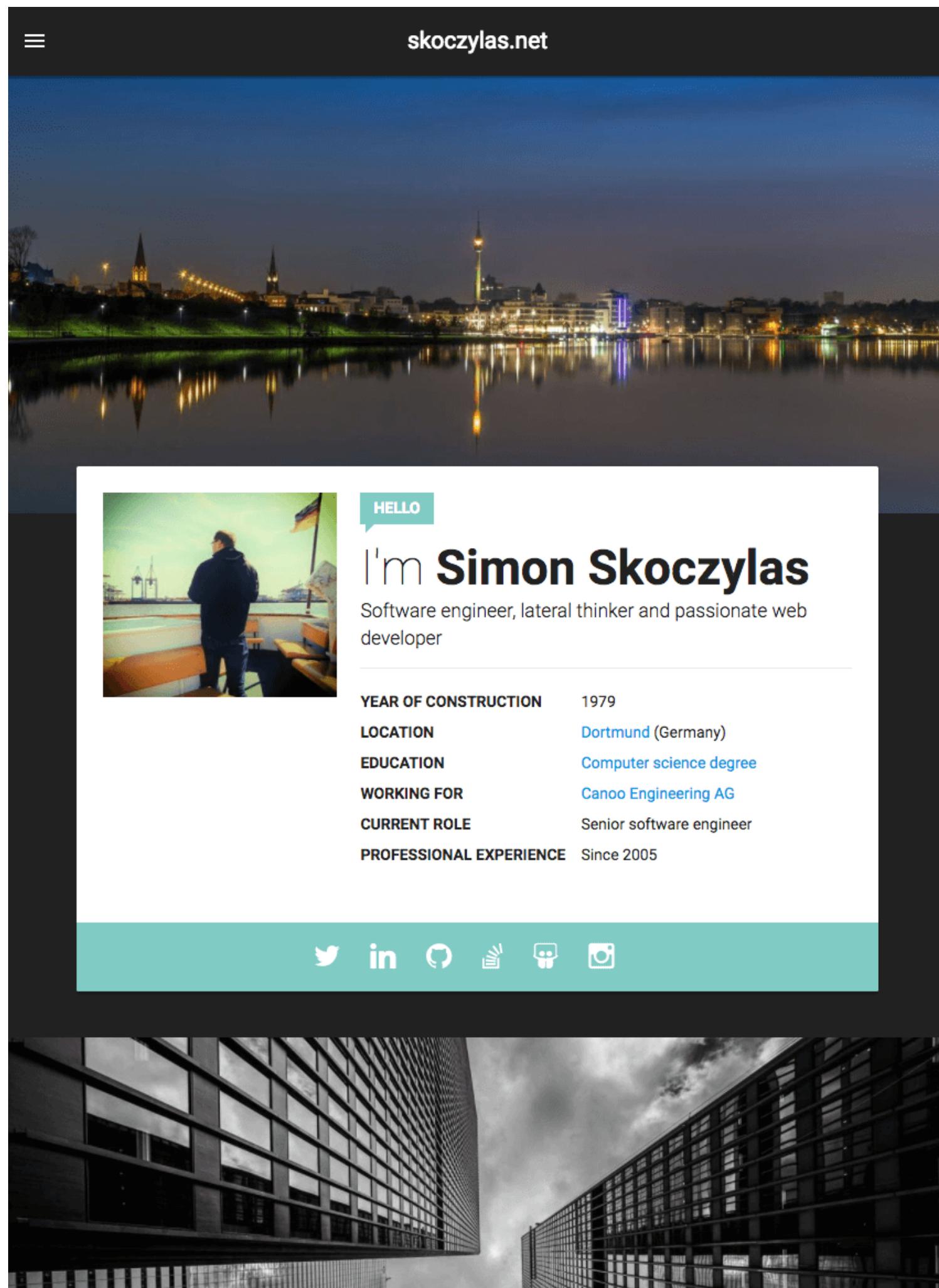
Polymer App Toolbox
<https://www.polymer-project.org/2.0/toolbox/>

Vue.js PWA Template
<https://github.com/vuejs-templates/pwa>

Angular CLI
<https://cli.angular.io/>



Lighthouse analysis



The screenshot shows the Lighthouse audit results for the website www.skoczylas.net at 7.3.2018, 11:02:15. The audit scores are:

- Progressive Web App: 91
- Performance: 72
- Accessibility: 94
- Best Practices: 94

Progressive Web App
These checks validate the aspects of a Progressive Web App, as specified by the baseline [PWA Checklist](#).

1 Failed Audit

- ▶ Is not configured for a custom splash screen
Failures: Manifest does not have icons at least 512px.

▶ 10 Passed Audits

▶ Manual checks to verify

Performance
These encapsulate your app's current performance and opportunities to improve it.

Metrics
These metrics encapsulate your app's performance across a number of dimensions.

Timeline chart showing performance metrics over time:

Time	Event
0.0 s	First meaningful paint
0.0 s	First Interactive (beta)
0.0 s	Consistently Interactive (beta)
0.0 s	Perceptual Speed Index (target: < 1.250)
0.0 s	Estimated Input Latency (target: < 50 ms)

Key performance metrics:

- First meaningful paint: 4.340 ms
- First Interactive (beta): 5.050 ms
- Consistently Interactive (beta): 5.050 ms
- Perceptual Speed Index: 2.175 (target: < 1.250)
- Estimated Input Latency: 16 ms (target: < 50 ms)





Let's take a look at some code



Ξ PWA Checklists

<https://developers.google.com/web/progressive-web-apps/checklist>

Baseline Checklist

Indexability & social

User experience

Performance

Caching

Push notifications

Additional features



Further reading

Web Fundamentals

<https://developers.google.com/web/fundamentals/>

Progressive Web Apps Training



<https://developers.google.com/web/ilt/pwa/>

Making A Service Worker: A Case Study

<https://www.smashingmagazine.com/2016/02/making-a-service-worker/>

ServiceWorker Cookbook by Mozilla

<https://serviceworke.rs>

Progressive Web Apps ILT - Codelabs

<https://www.gitbook.com/book/google-developer-training/progressive-web-apps-ilt-codelabs/details>

Caching best practices & max-age gotchas

<https://jakearchibald.com/2016/caching-best-practices/>



Thank you for listening

PWA are cool with at least those Web APIs

Service Worker API

Web App Manifest

Cache API

Fetch API

Notification API

Indexed DB

BackgroundSync API

Credentials API

Payment Request API

Web Share API

Stay in touch @ dev.karakun.com



Karakun DevHub_

@gftkugel

Legal stuff

Images (CC0) taken from

<https://www.pexels.com/photo/camera-cctv-control-monitoring-274895/>
<https://www.pexels.com/photo/silhouette-of-woman-raising-her-hands-615334/>
<https://www.pexels.com/photo/man-in-white-shirt-using-macbook-pro-52608/>
<https://www.pexels.com/photo/bind-blank-blank-page-business-315790/>
<https://www.pexels.com/photo/building-construction-site-work-38293/>
<https://www.pexels.com/photo/road-pavement-229014/>

Phone mockups generated with

<https://mockuphone.com/#android>

Icons

<https://material.io/icons/>
<http://mariodelvalle.github.io/CaptainIconWeb/>
<https://github.com/alrra/browser-logos>

