

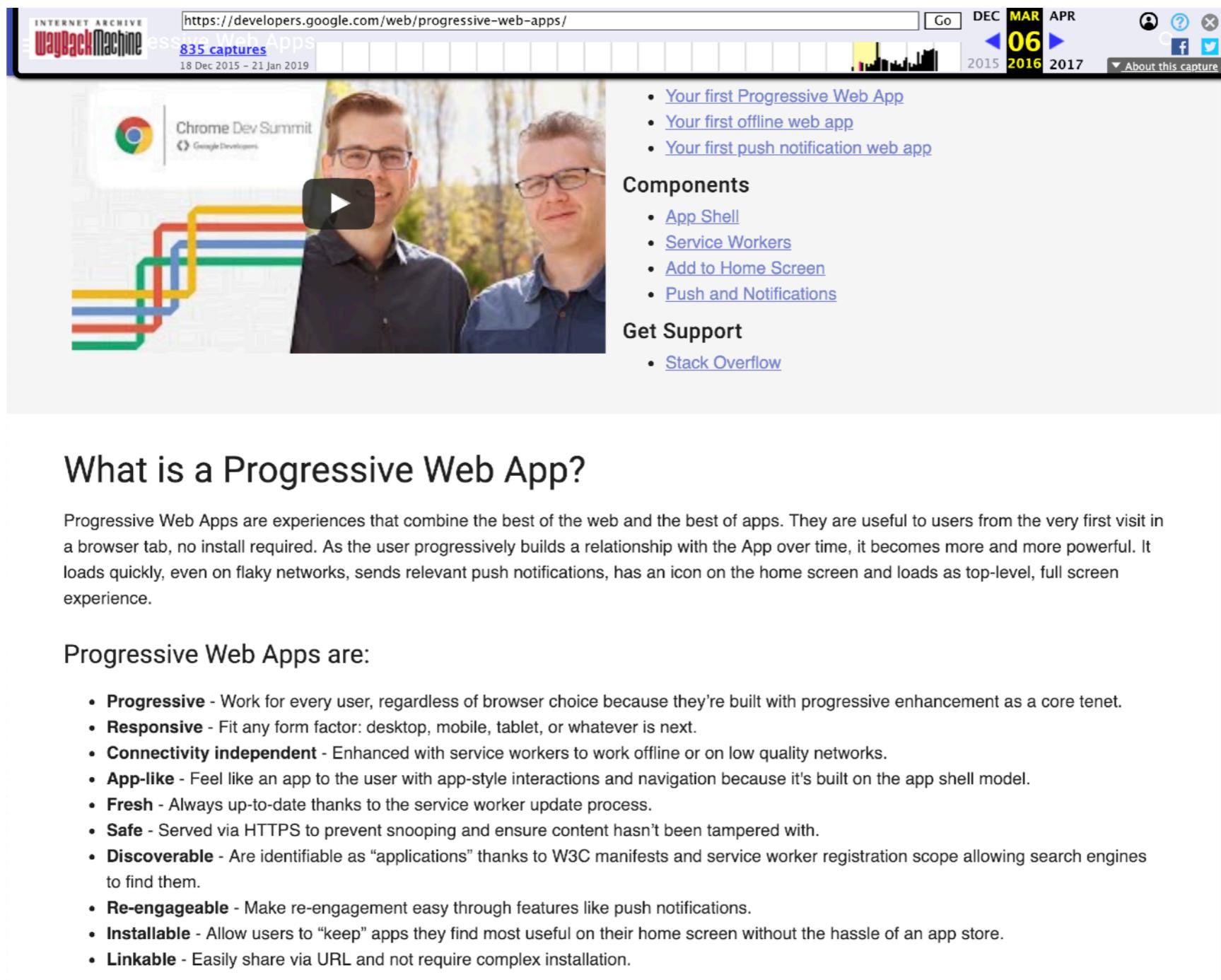
# Progressive web apps

Declan Rek

# WHAT IS A PWA



# 2016



The screenshot shows a Wayback Machine capture of the Google Progressive Web Apps page. The URL is <https://developers.google.com/web/progressive-web-apps/>. The date is set to March 6, 2016. The page features a video player with two men, a Chrome Dev Summit logo, and a graphic of colorful pipes. It includes sections for components like App Shell, Service Workers, Add to Home Screen, and Push and Notifications, along with links to the first PWA, offline web app, and push notification web app.

## What is a Progressive Web App?

Progressive Web Apps are experiences that combine the best of the web and the best of apps. They are useful to users from the very first visit in a browser tab, no install required. As the user progressively builds a relationship with the App over time, it becomes more and more powerful. It loads quickly, even on flaky networks, sends relevant push notifications, has an icon on the home screen and loads as top-level, full screen experience.

Progressive Web Apps are:

- **Progressive** - Work for every user, regardless of browser choice because they're built with progressive enhancement as a core tenet.
- **Responsive** - Fit 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** - Feel like an app to the user with app-style interactions and navigation because it's built on the app shell model.
- **Fresh** - Always up-to-date thanks to the service worker update process.
- **Safe** - Served via HTTPS to prevent snooping and ensure content hasn't been tampered with.
- **Discoverable** - Are identifiable as "applications" thanks to W3C manifests and service worker registration scope allowing search engines to find them.
- **Re-engageable** - Make re-engagement easy through features like push notifications.
- **Installable** - Allow users to "keep" apps they find most useful on their home screen without the hassle of an app store.
- **Linkable** - Easily share via URL and not require complex installation.

# 2016... LATER

The screenshot shows a Wayback Machine interface with a timeline from June 2015 to September 2017. A specific capture from August 19, 2016, is highlighted. The URL in the address bar is <https://developers.google.com/web/progressive-web-apps/>. The page content includes a quote from Amar Nagaram: "that performs as well as our native app. We now feel we have the best of both worlds." Below the quote is a note from Amar Nagaram, an Engineering Director at Flipkart.



## Instant Loading

Service workers allow your apps to load nearly instantly and reliably, no matter what kind of network connection your user is on.



## Add to Home Screen

Web app install banners give you the ability to let your users quickly and seamlessly add your web app to their home screen, making it easy to launch and return to your app.



## Push Notifications

Web push notifications makes it easy to re-engage with users by showing relevant, timely, and contextual notifications, even when the browser is closed.



## Fast

Smooth animations, scrolling, and navigations keep the experience silky smooth.



## Secure

HTTPS secures the connection between you and your users, ensuring your users information is protected and isn't tampered with.



## Responsive

Modern users live on phones, tablets and laptops; your apps and websites should do the same. Learn how to structure and code your content to look great on screens of any size.

# NOW



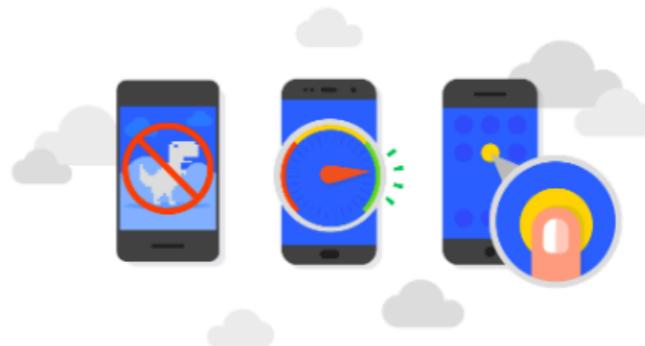
## Progressive Web Apps

A new way to deliver amazing user experiences on the web.

Don't miss out on the action at this year's Chrome Dev Summit, [watch live now!](#)

Progressive Web Apps are user experiences that have the reach of the web, and are:

- **Reliable** - Load instantly and never show the downasaur, even in uncertain network conditions.
- **Fast** - Respond quickly to user interactions with silky smooth animations and no janky scrolling.
- **Engaging** - Feel like a natural app on the device, with an immersive user experience.



This new level of quality allows Progressive Web Apps to earn a place on the users home screen.

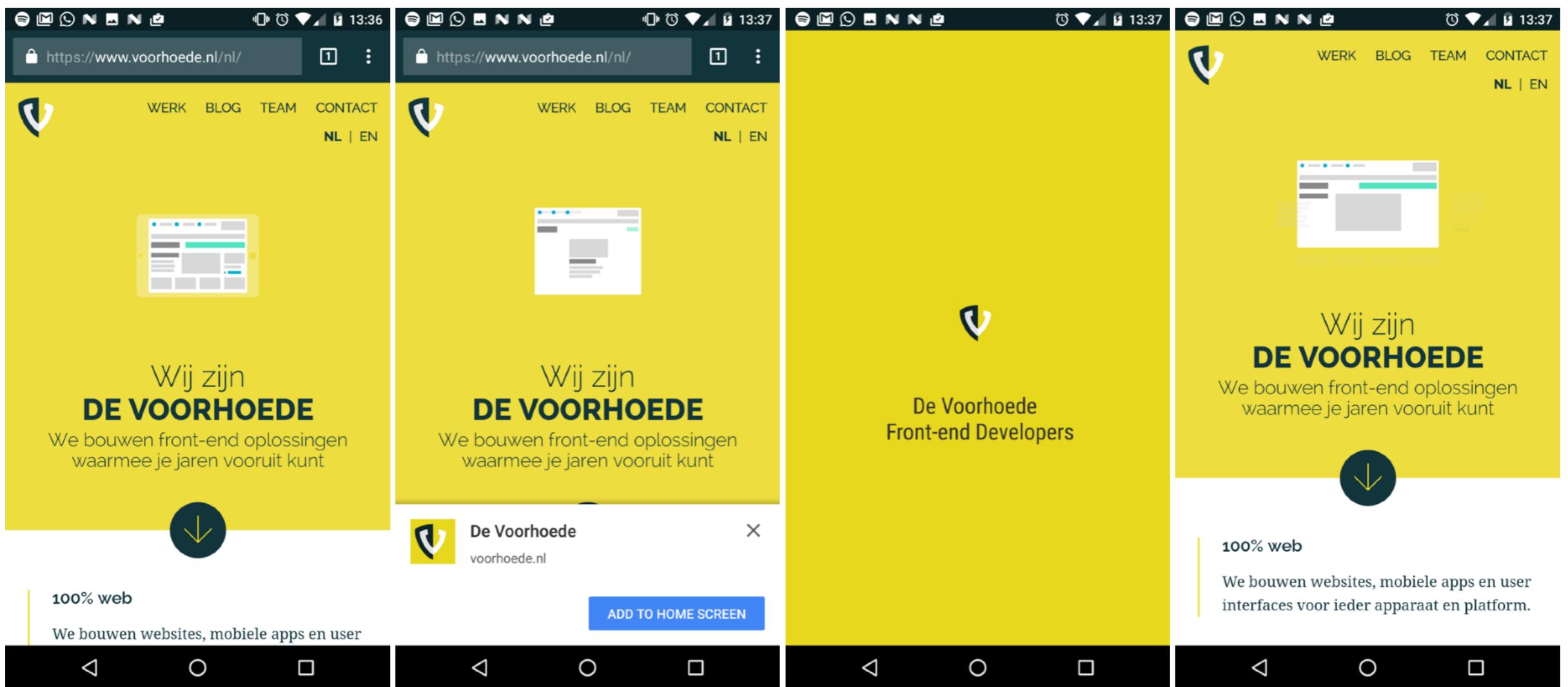
**Progressive Web Apps are  
user experiences that have  
the reach of the web**

**HTTPS**

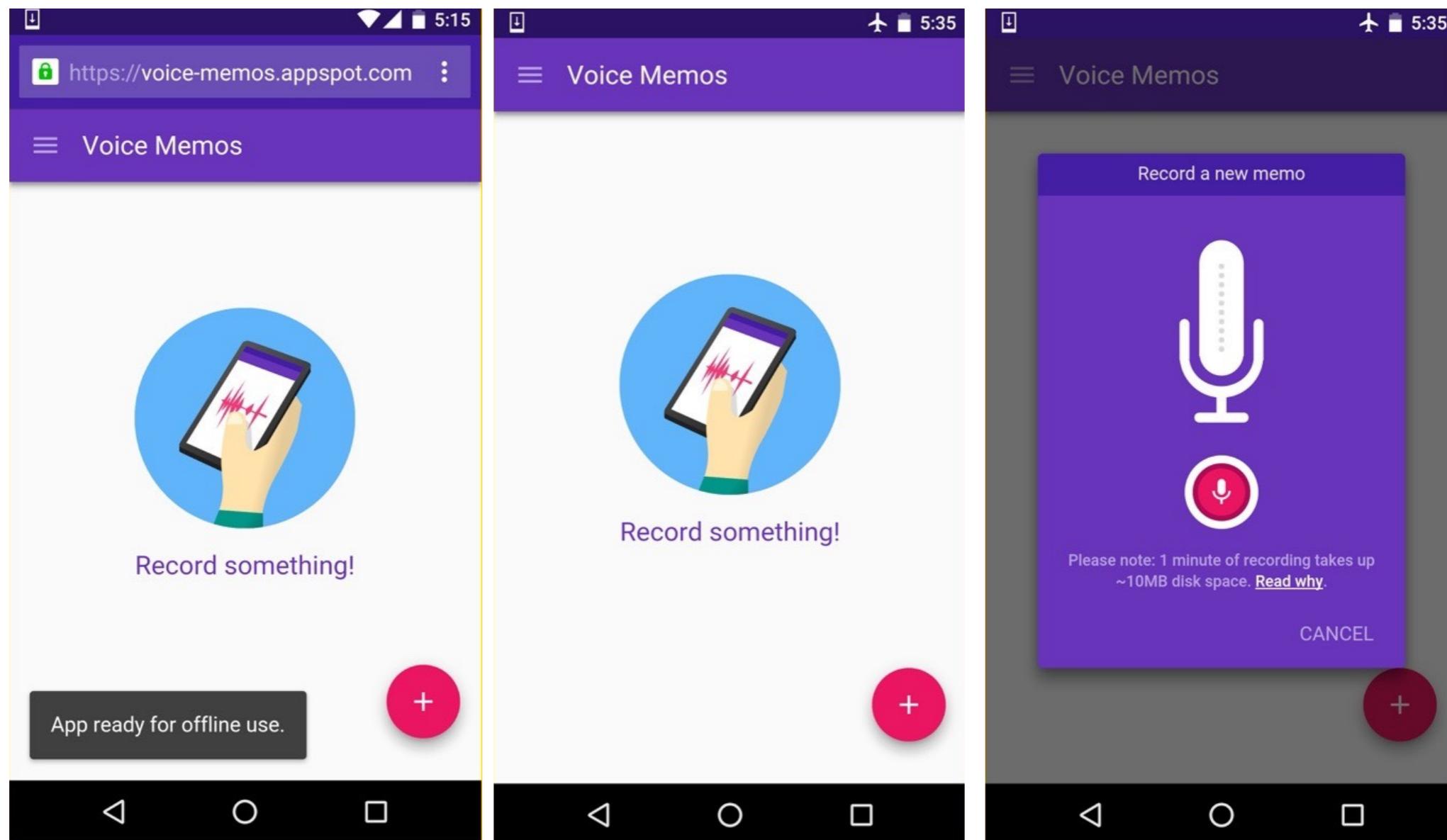
**Service worker**

**Manifest file**

# ADD TO HOMESTEEN

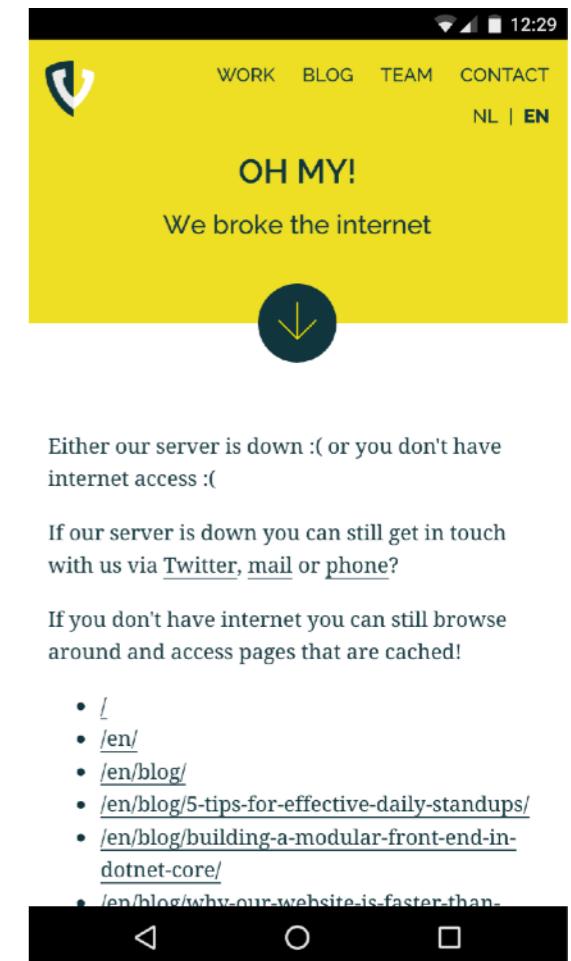
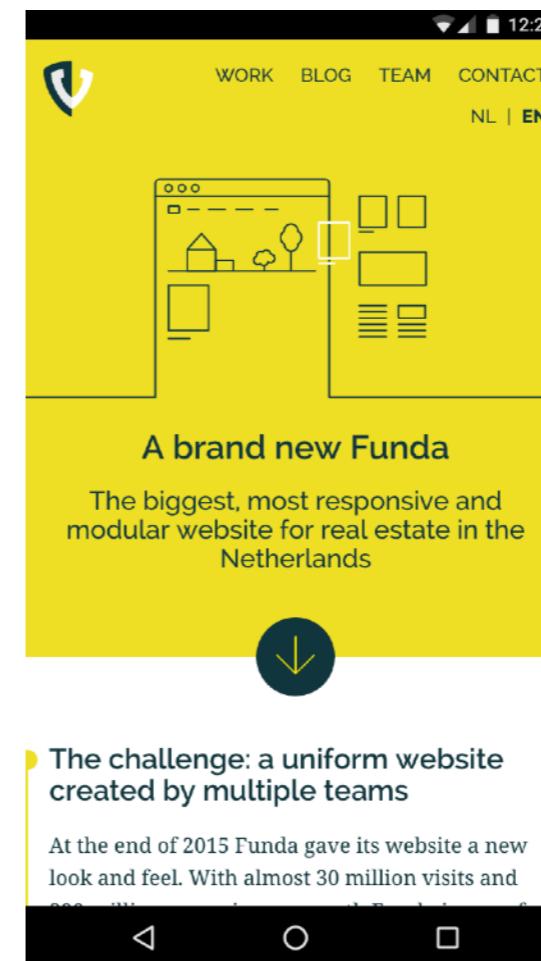
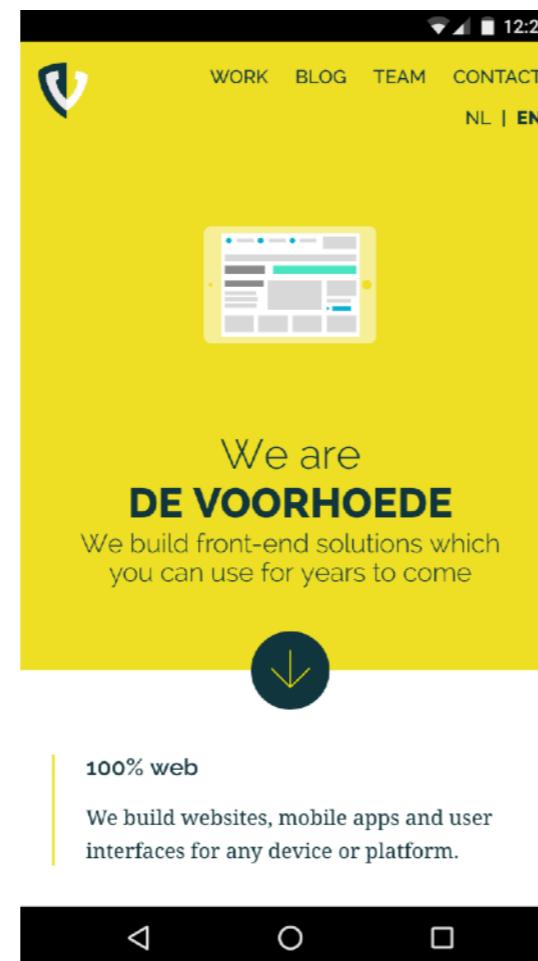
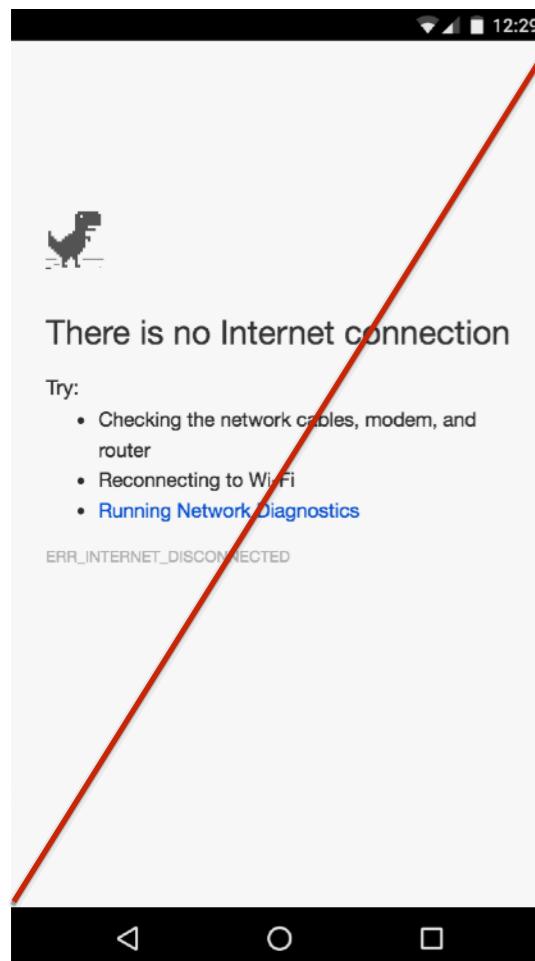


# OFFLINE SUPPORT

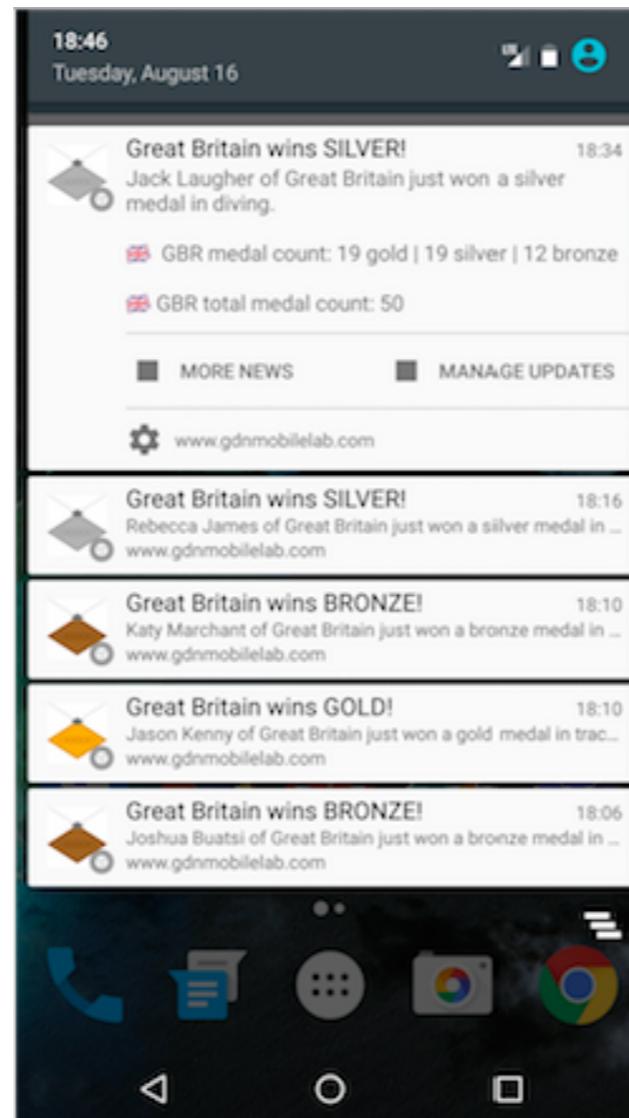
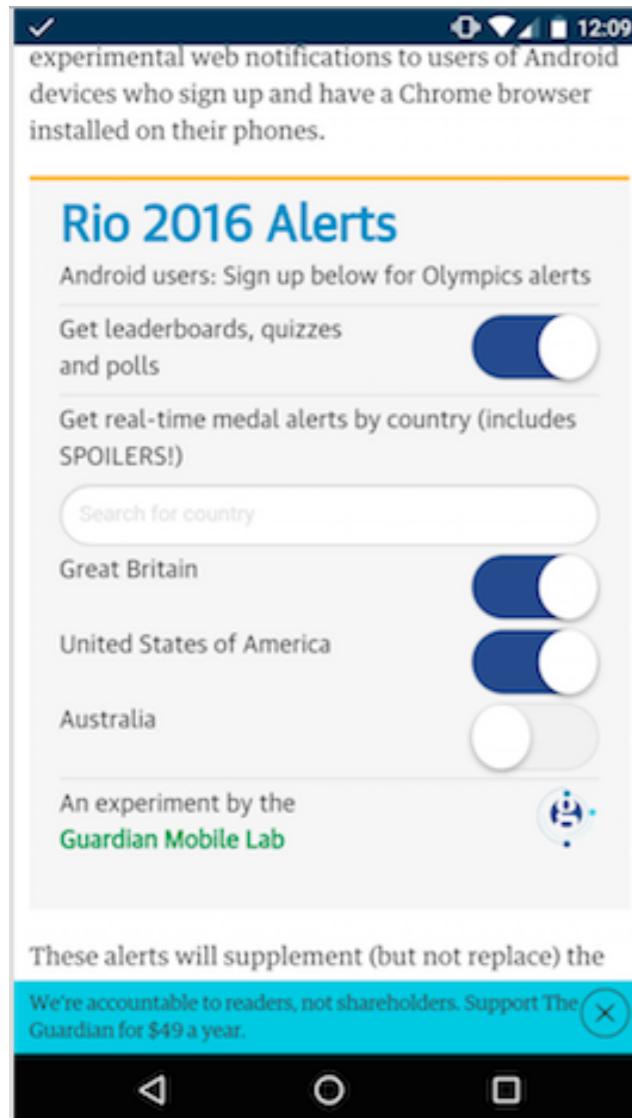


[developers.google.com/web](https://developers.google.com/web)

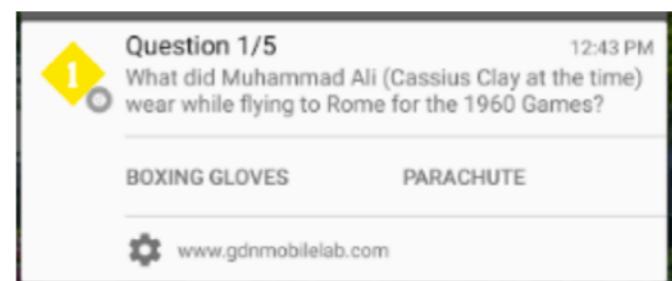
# OFFLINE SUPPORT



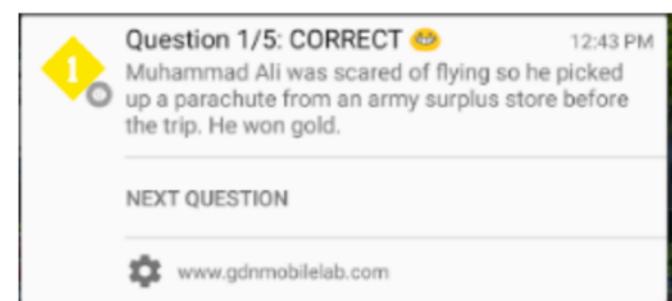
# PUSH NOTIFICATIONS



Users tapped the action button to start the quiz



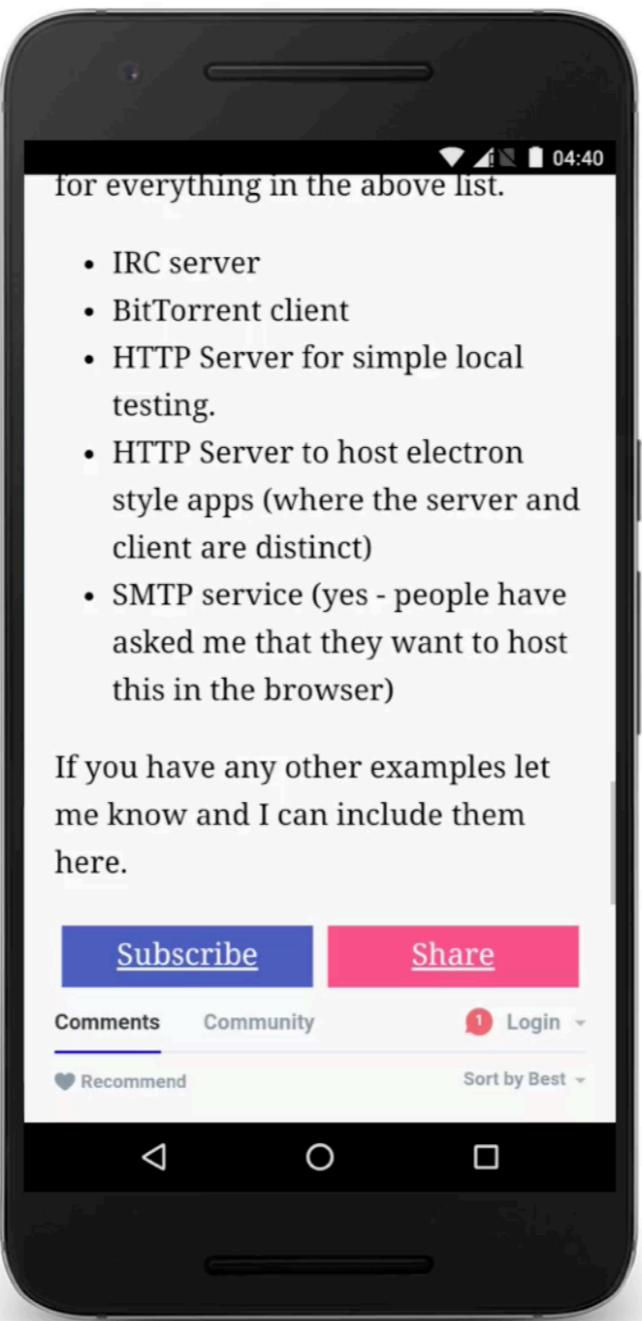
Users tapped on the action button with their answer



The answer was revealed before going to the next question

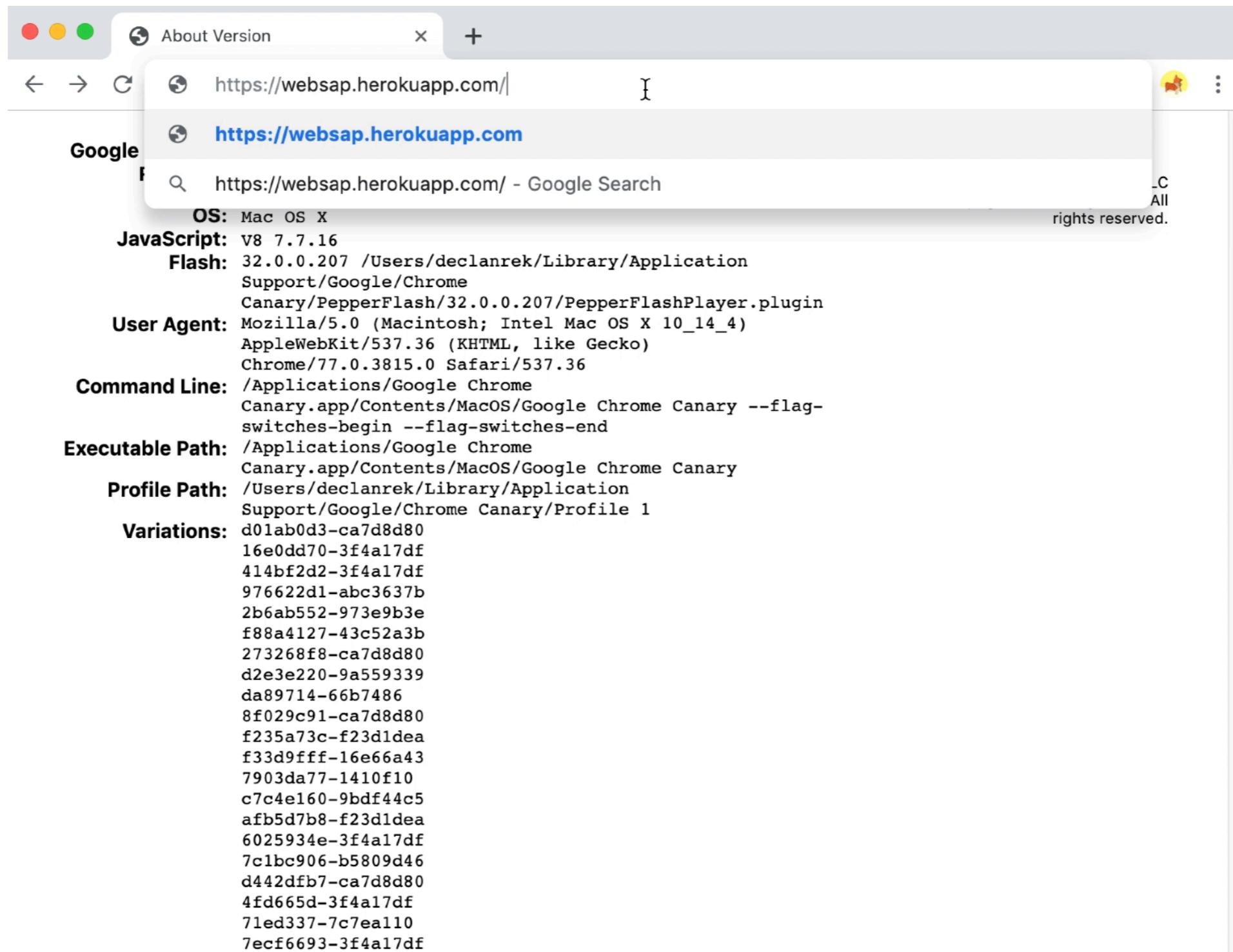
[medium.com/the-guardian-mobile-innovation-lab/](https://medium.com/the-guardian-mobile-innovation-lab/)

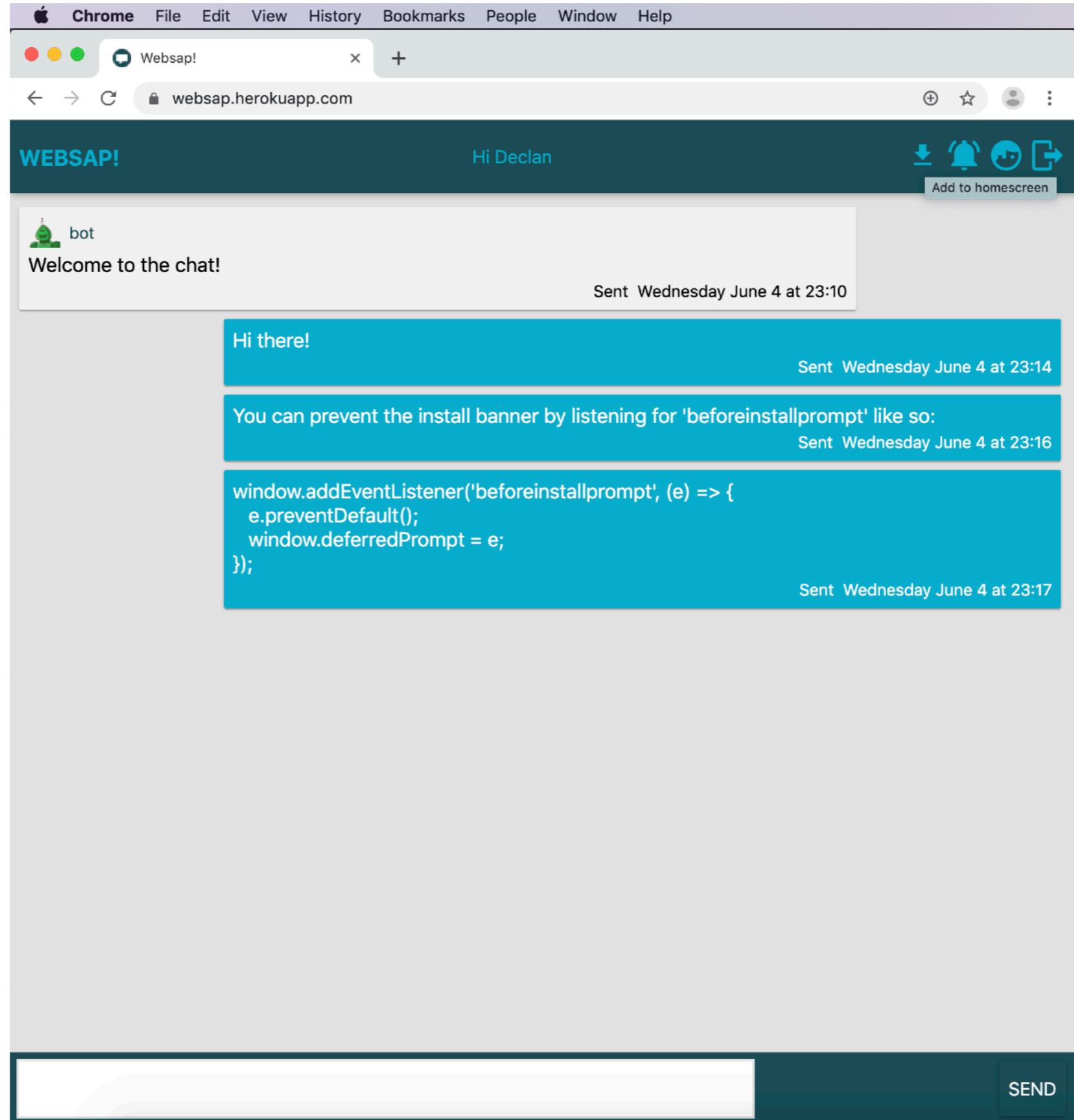
# NATIVE SHARE

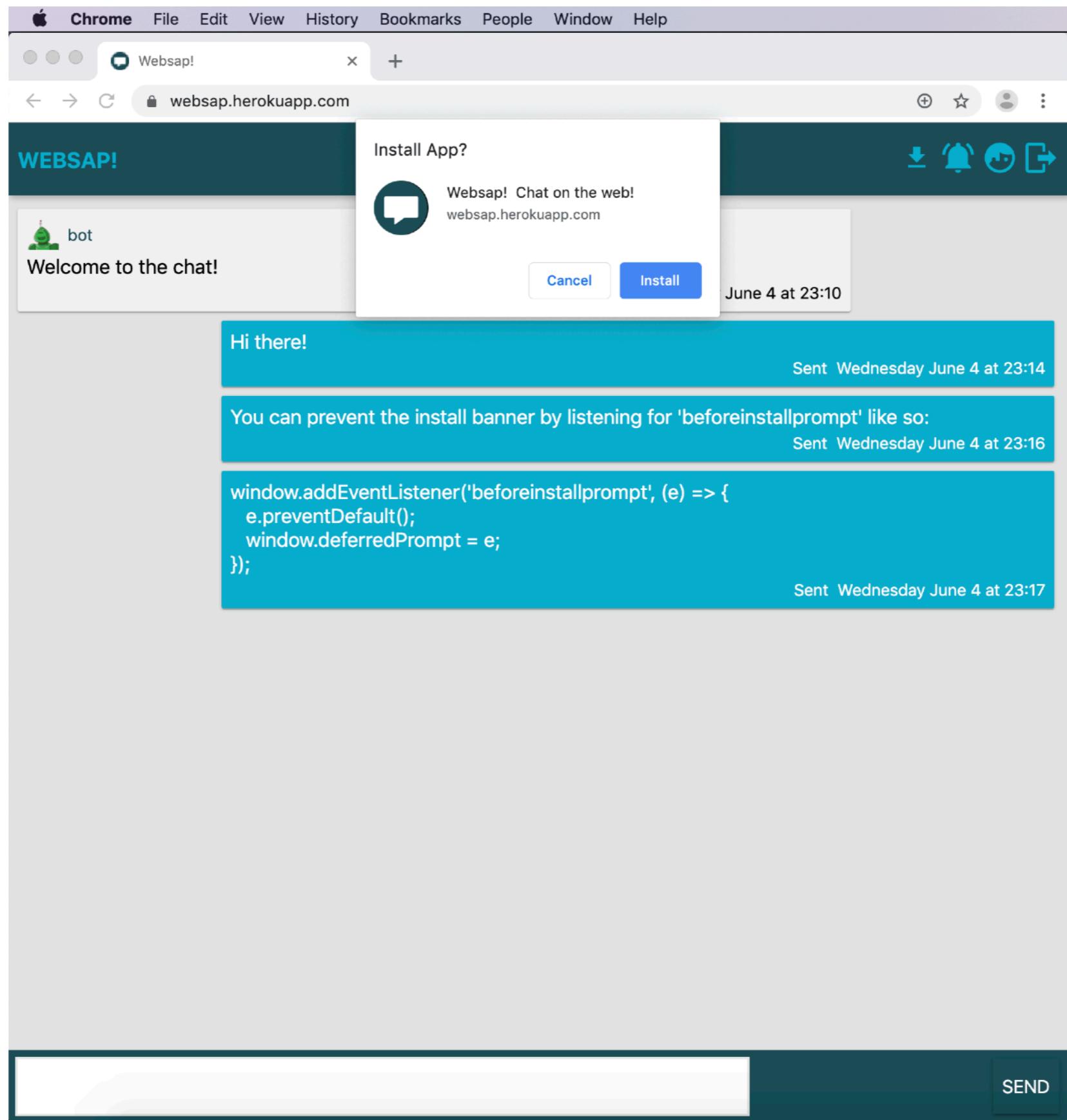


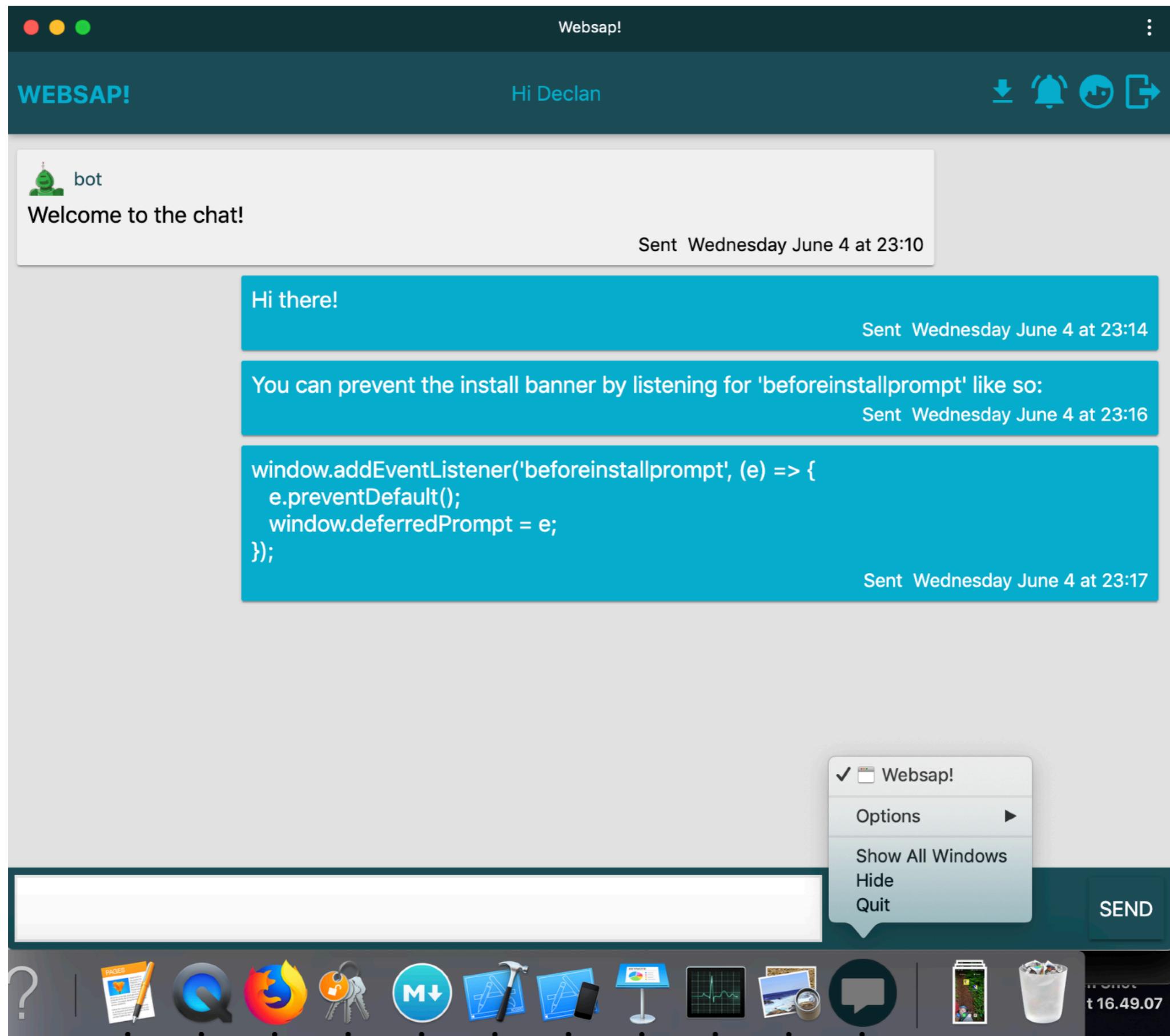
[paul.kinlan.me/navigator.share](http://paul.kinlan.me/navigator.share)

# DESKTOP APPS









# PLAY STORE

The image shows the Google Play Store interface for the app 'Pick Up 10'. The top navigation bar includes the Google Play logo, a search bar, and language selection (De). Below the navigation is a green header bar with the 'Apps' tab selected. The main content area displays the app's details: title 'Pick Up 10', developer 'De Voorhoeve', category 'Lifestyle', PEGI rating '3', and compatibility note 'This app is compatible with your device.' A large green button on the right indicates the app is 'Installed'. Three screenshots of the app are shown at the bottom: the first shows a 3-step challenge process; the second shows the camera interface with instructions to 'Take a pic of litter'; the third shows a results screen with a star rating and a list of items identified.

# MICROSOFT STORE

Microsoft Store

Startpagina Games Entertainment Productiviteit Deals Search ↻ 44 ⋮

This product is installed.

**Pick Up 10**

De Voorhoeve • Lifestyle

Share Wish list

Plastic litter turns into plastic soup and threatens our oceans. The good news: continuing his fight for a plastic-free ocean, this summer the Plastic Soup Surfer will go adventuring with his 3D-printed foil SUP board made of recycled plastic. And you can help! The Plastic Soup Surfer invites everyone to join in his "Pick Up 10" campaign. You pick up More

PEGI 3

Free

Type here to search

14:07 17/01/2020 ENG

**HTTPS**

**Service worker**

**Manifest file**

# MANIFEST.JSON

The screenshot shows a web browser window with the URL [voorhoede.nl/en/](http://voorhoede.nl/en/). The main content area displays the De Voorhoede website, which features a yellow background with blue line-art illustrations of people working on computers and a large white hand holding a stylized path or ribbon. The page includes text about making the internet better and a heading "Front-end specialists with heart and hands". Below this, there's a section titled "What we do" with a description of their services.

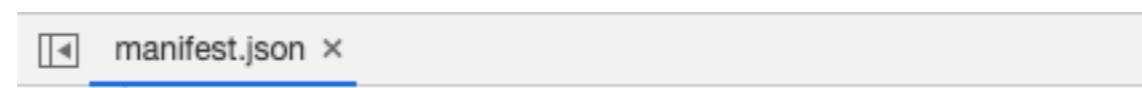
The browser's address bar shows the current URL. The DevTools sidebar is open, specifically the "Application" tab, which is focused on the "Manifest" section. The manifest file, named `manifest.json`, is shown with the following content:

```
{ "name": "De Voorhoede", "short_name": "De Voorhoede", "start_url": "/?homescreen=true", "display": "standalone", "background_color": "#ffe400", "theme_color": "#0000ff", "orientation": "portrait", "icons": [ { "src": "image/png", "size": "36x36px", "type": "image/png" }, { "src": "image/png", "size": "48x48px", "type": "image/png" }, { "src": "image/png", "size": "72x72px", "type": "image/png" }, { "src": "image/png", "size": "96x96px", "type": "image/png" }, { "src": "image/png", "size": "144x144px", "type": "image/png" } ] }
```

The "Manifest" section also lists Service Workers and Storage. The "Installability" section notes that the page is loaded in an incognito window and that no service worker is detected. The "Identity" section shows the name "De Voorhoede" and short name "De Voorhoede". The "Presentation" section includes fields for Start URL, Theme color, Background color, Orientation, and Display. The "Icons" section shows the primary icon used by Chrome and provides links for download at various sizes: 36x36px, 48x48px, 72x72px, 96x96px, and 144x144px.

# MANIFEST.JSON

<link rel="manifest" href="/manifest.json">



```
manifest.json x
1 {
2   "name": "De Voorhoede \n Front-end Developers",
3   "short_name": "De Voorhoede",
4   "start_url": "/?homescreen=true",
5   "display": "standalone",
6   "orientation": "portrait",
7   "gcm_sender_id": "482941778795",
8   "theme_color": "#0000ff",
9   "background_color": "#ffe400",
10  "icons": [
11    {
12      "src": "/images/social/android-36x36.png",
13      "sizes": "36x36",
14      "type": "image/png"
15    },
16    {
17      "src": "/images/social/android-48x48.png",
18      "sizes": "48x48",
19      "type": "image/png"
20    },
21    {
22      "src": "/images/social/android-72x72.png",
23      "sizes": "72x72",
24      "type": "image/png"
25    },
26    {
27      "src": "/images/social/android-96x96.png",
28      "sizes": "96x96",
29      "type": "image/png"
30    },
31    {
32      "src": "/images/social/android-144x144.png",
33      "sizes": "144x144",
34      "type": "image/png"
35    },
36    {
37      "src": "/images/social/android-192x192.png",
38      "sizes": "192x192",
39      "type": "image/png"
40    }
41  ]
42 }
```

# SERVICE WORKER



Service workers essentially act as proxy servers that sit between web applications, the browser, and the network (when available). They are intended, among other things, to enable the creation of effective offline experiences, intercept network requests and take appropriate action based on whether the network is available. They will also allow access to push notifications and background sync APIs.

[MDN - Service worker API](#)

# SERVICE WORKER

JavaScript

Web worker

Runs in the background

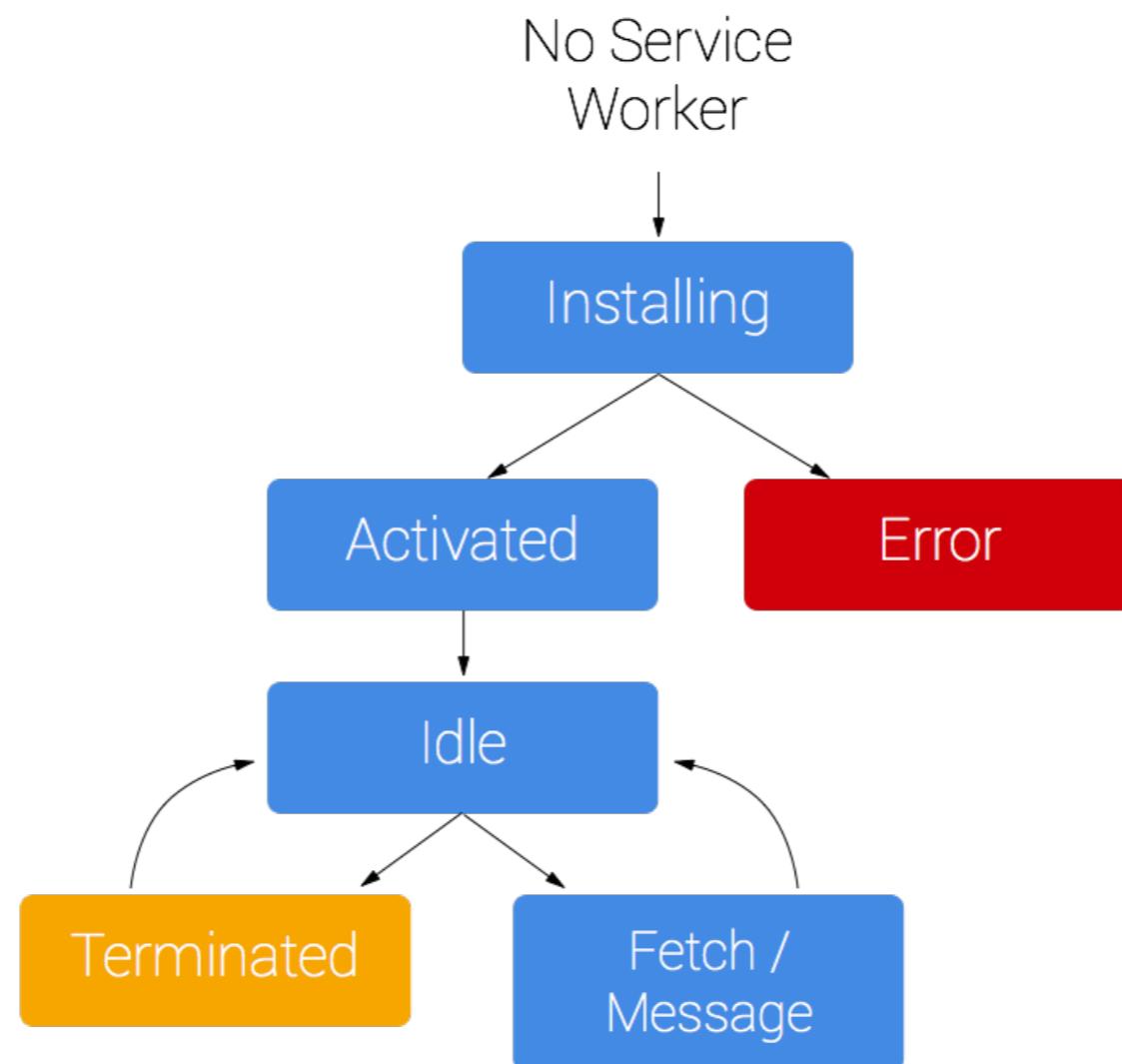
Separate thread

Event based

No access to DOM

No global state

# SERVICE WORKER



[developers.google.com/web/fundamentals/primers/service-workers](https://developers.google.com/web/fundamentals/primers/service-workers)

# REGISTER SW

```
if ('serviceWorker' in navigator) {  
  window.addEventListener('load', function() {  
    navigator.serviceWorker.register('/service-worker.js')  
      .then(function(registration) {  
        return registration.update();  
      })  
  }) ;  
}
```

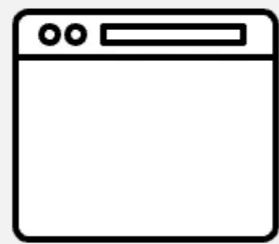
# SERVICE WORKER LIFECYCLE

```
self.addEventListener('install', event => {  
    // do I even exist?  
});
```

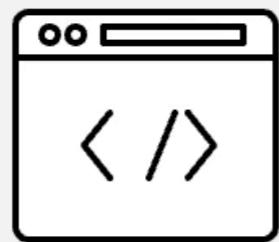
```
self.addEventListener('activate', event => {  
    // am I even active?  
});
```

```
self.addEventListener('fetch', event => {  
    // can I even do cool stuff?  
});
```

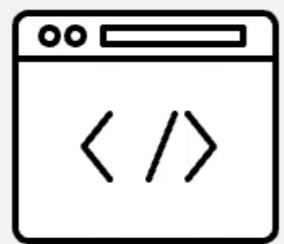
# INITIAL PAGE LOAD



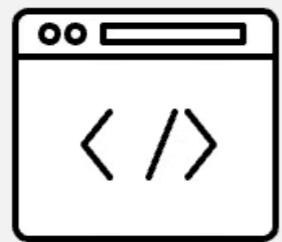
# REGISTER SERVICE WORKER



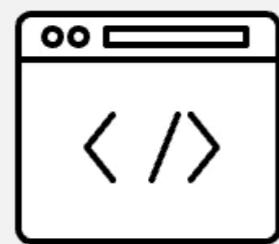
# SERVICE WORKER AS PROXY



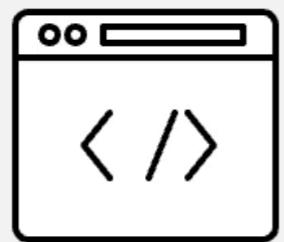
# HIJACK FETCH



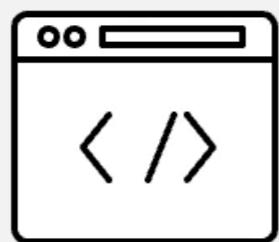
# CACHE ON INSTALL



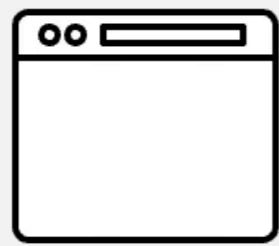
# CACHE ON FETCH



# SERVE FROM CACHE



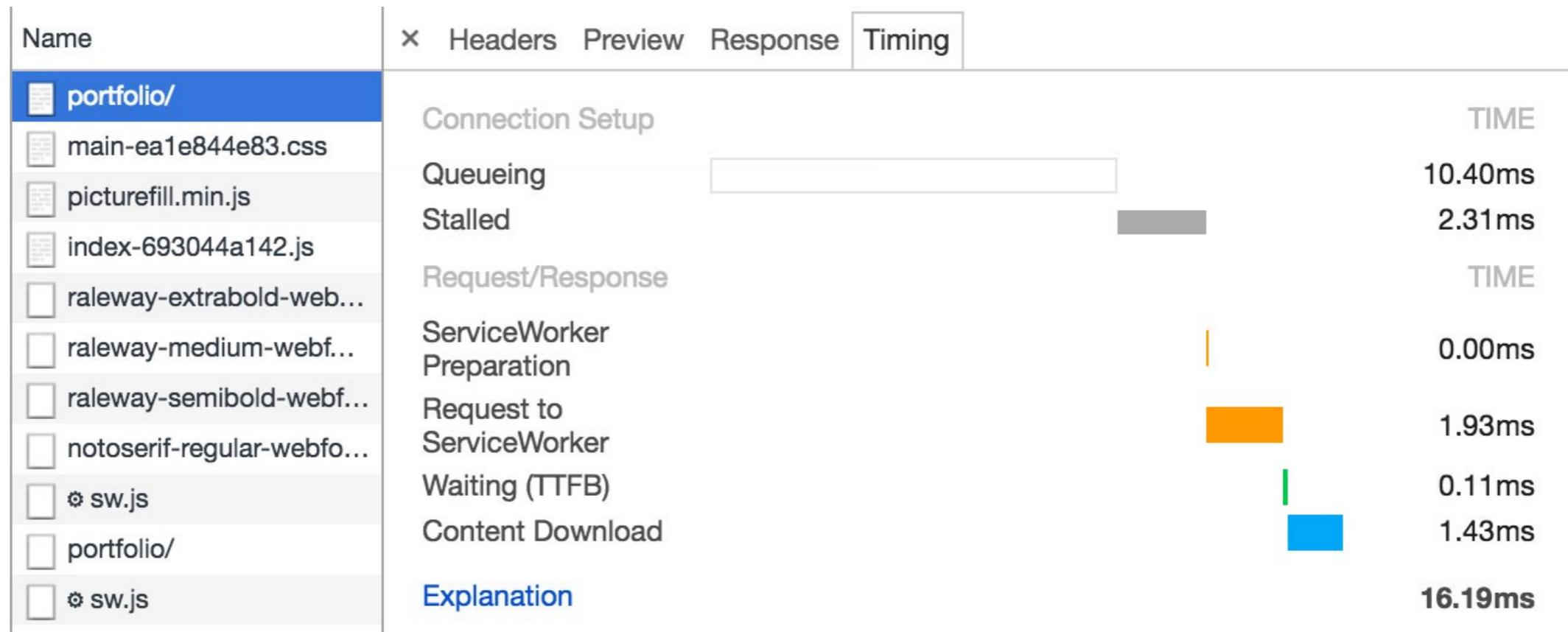
# UPDATE SERVICE WORKER



# FROM



# TO



# CACHING THINGS



# PRECACHING

```
const CORE_CACHE_NAME = 'core-cache';
const CORE_ASSETS = [ /* css, bundle.js, offline page */ ];

self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CORE_CACHE_NAME)
      .then(cache => cache.addAll(CORE_ASSETS))
      .then(() => self.skipWaiting())
  )
});
```

# SERVE FROM CACHE

```
self.addEventListener('fetch', event => {
  const request = event.request;
  if (isInCoreCache(request)) {
    event.respondWith(
      caches.open(CORE_CACHE_NAME)
        .then(cache => cache.match(request.url))
    )
  }
})
```

# RUNTIME CACHING

```
self.addEventListener('fetch', event => {
  const request = event.request;
  if (isSomethingParticular(request) && isNotCached(request)) {
    event.respondWith(
      return fetch(request).then(response => {
        return caches.open('mycache')
          .then(cache => {
            cache.put(request, response.clone())
          })
          .then(() => response)
      })
    )
  }
})
```

# HIJACK FETCH

```
self.addEventListener('fetch', event => {  
  const request = event.request;  
  event.respondWith(  
    '^_^(ツ)_/^\n  );  
});
```

```
self.addEventListener('install', event => {
    // on installation
});

self.addEventListener('activate', event => {
    // on activation
});

self.addEventListener('fetch', event => {
    // on outgoing request
});

self.addEventListener('push', event => {
    // on push message received
});

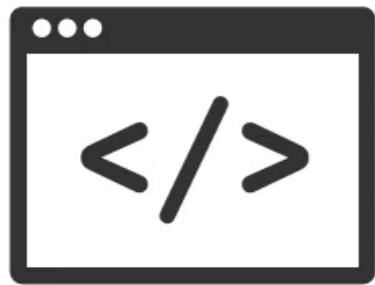
self.addEventListener('message', event => {
    // on message from app
});

self.addEventListener('sync', event => {
    //
});
```

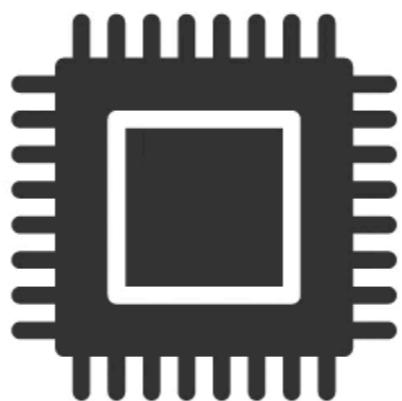
# CACHING STRATEGIES



# CACHE ONLY



Page

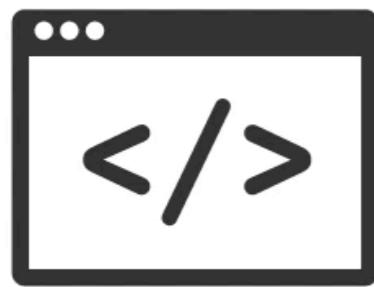


ServiceWorker

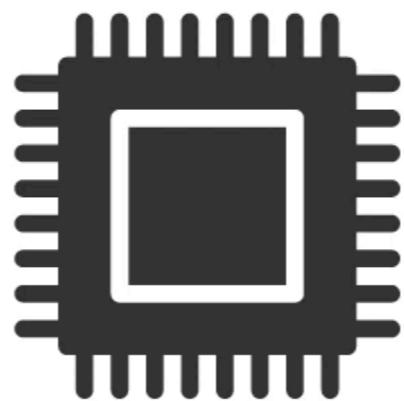


Cache

# NETWORK ONLY



Page



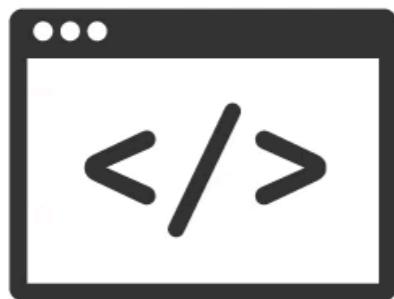
ServiceWorker



Network

[Offline cookbook by Jake Archibald](#)

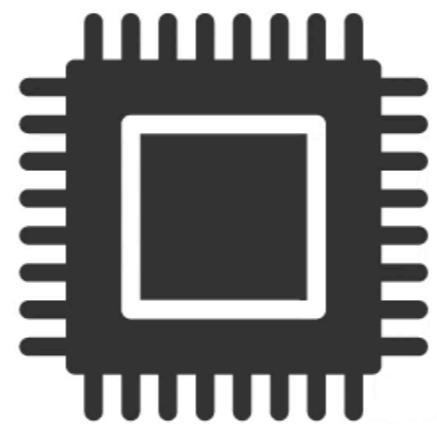
# CACHE FIRST



Page



Network



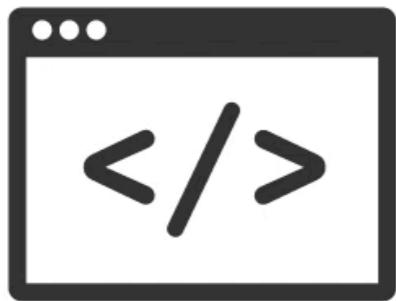
ServiceWorker



Cache

[Offline cookbook by Jake Archibald](#)

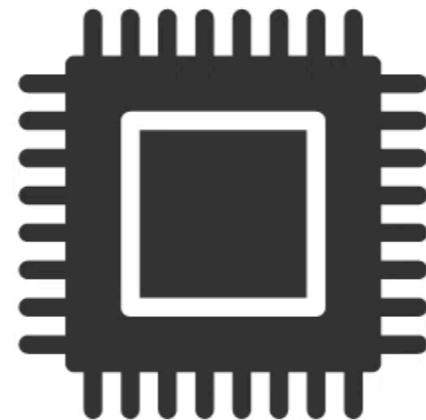
# NETWORK FIRST



Page



Cache



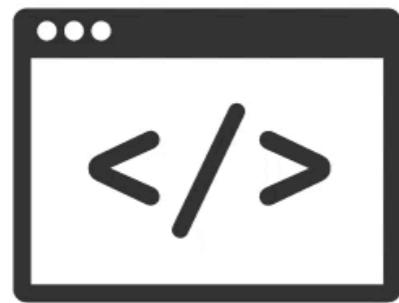
ServiceWorker



Network

[Offline cookbook by Jake Archibald](#)

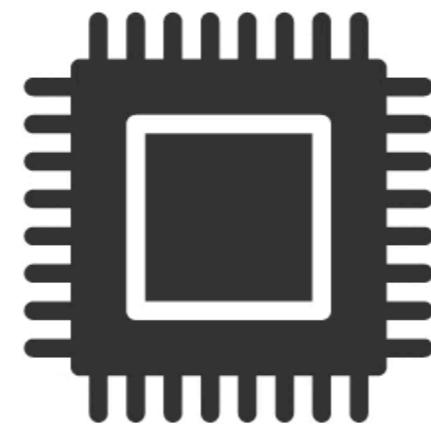
# STALE WHILE REVALIDATE



Page



Cache



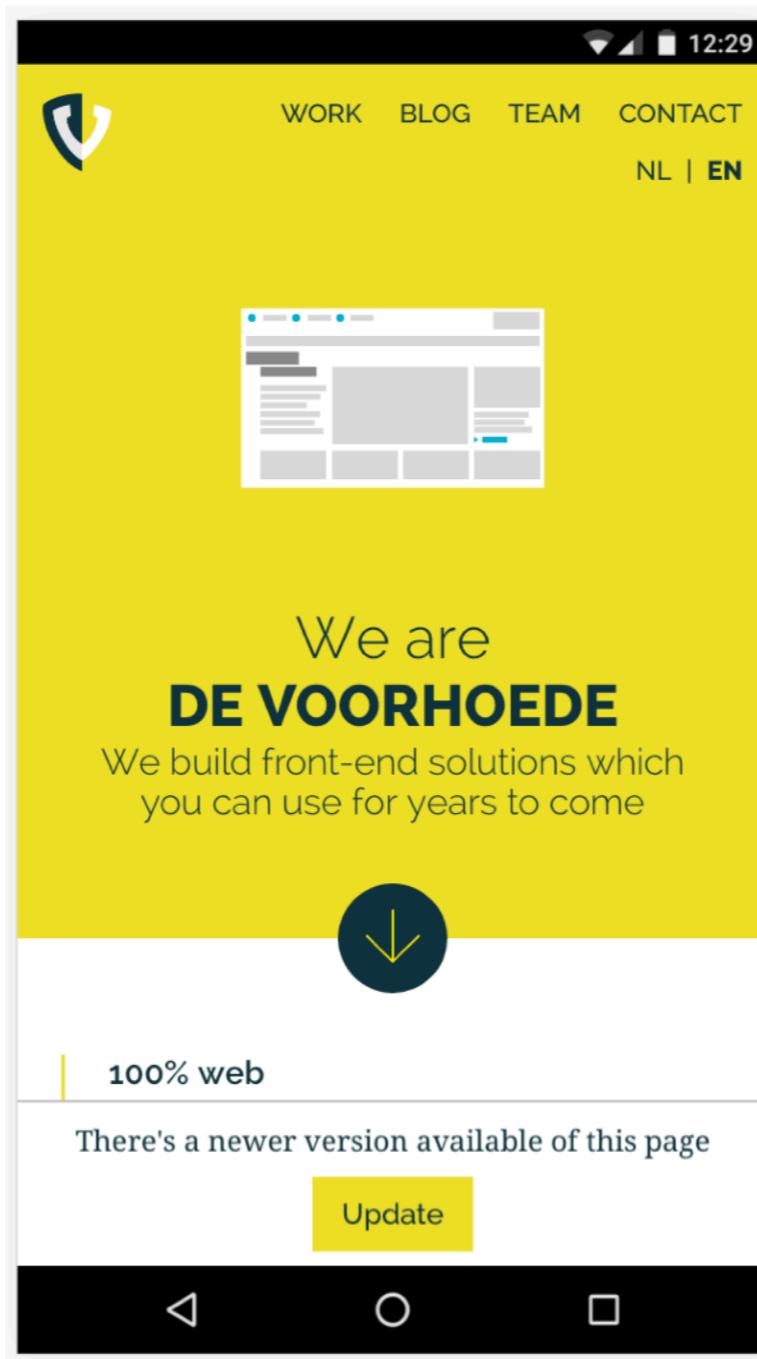
ServiceWorker



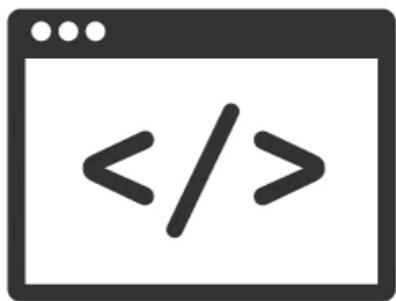
Network

[Offline cookbook by Jake Archibald](#)

# STALE WHILE REVALIDATE



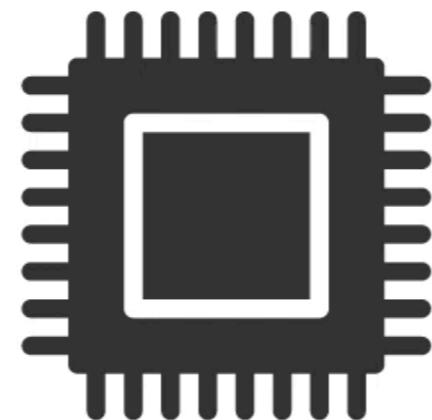
# GENERIC FALBACK



Page



Cache



ServiceWorker



Network

[Offline cookbook by Jake Archibald](#)

# DEVTOOLS & TESTING



# CHROME DEV TOOLS

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the sidebar lists sections: Application (Manifest, Service Workers, Clear storage), Storage (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), Cache (Cache Storage, Application Cache), and Frames (top). The main panel displays the Service Workers section for the URL <http://localhost:7924/>. It shows a service worker named [service-worker.js](#) was received on 14/12/2017, 10:49:40. The status is green, indicating it is activated and running. Two clients are listed: <http://localhost:7924/> and <http://localhost:7924/>. Action buttons for Update, Push, Sync, and Unregister are available.

Application

- Manifest
- Service Workers**
- Clear storage

Storage

- Local Storage
- Session Storage
- IndexedDB
  - chatDb - <http://localhost:7924/>
  - Web SQL
- Cookies
  - <http://localhost:7924/>

Cache

- Cache Storage
  - core-cache - <http://localhost:7924/>
  - Application Cache

Frames

- top

Service Workers

Offline  Update on reload  Bypass for network  Show all

<http://localhost:7924/>

Source [service-worker.js](#)  
Received 14/12/2017, 10:49:40

Status ● #17547 activated and is running [stop](#)

Clients <http://localhost:7924/> [focus](#)  
<http://localhost:7924/> [focus](#)

[Update](#) [Push](#) [Sync](#) [Unregister](#)

# LIGHTHOUSE

The screenshot shows the Lighthouse audit interface with the following scores:

- Progressive Web App: 82
- Performance: 100
- Accessibility: 100
- Best Practices: 92

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

82

2 failed audits

- ▶ Does not redirect HTTP traffic to HTTPS ✗
- ▶ User will not be prompted to Install the Web App ✗

Failures: Manifest start\_url is not cached by a Service Worker.

▶ 9 Passed Audits

▶ Manual checks to verify

**Performance**  
These encapsulate your app's performance.

100

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

131 ms	263 ms	394 ms	526 ms	657 ms	789 ms	920 ms	1.1 s	1.2 s	1.3 s