

# PWAs in 2021

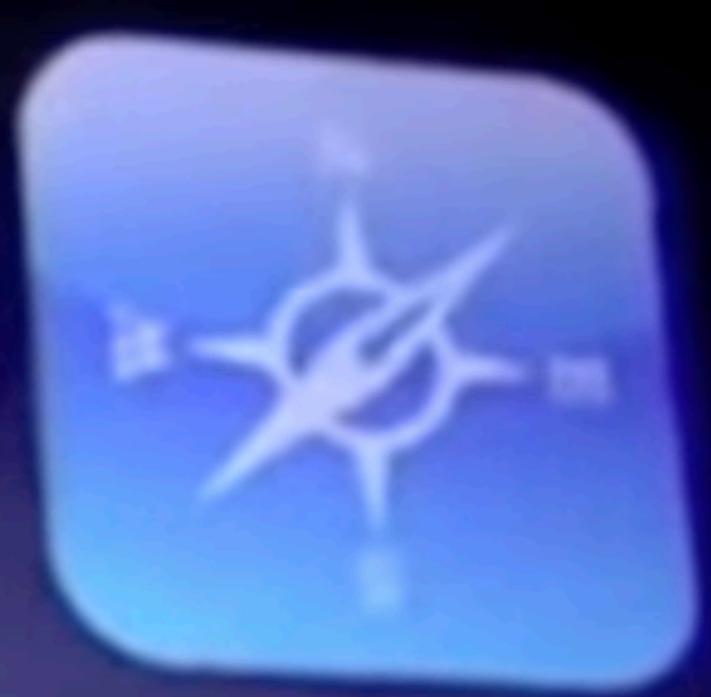
Ire Aderinokun @ An Event Apart  
“Spring Summit” 2021





od

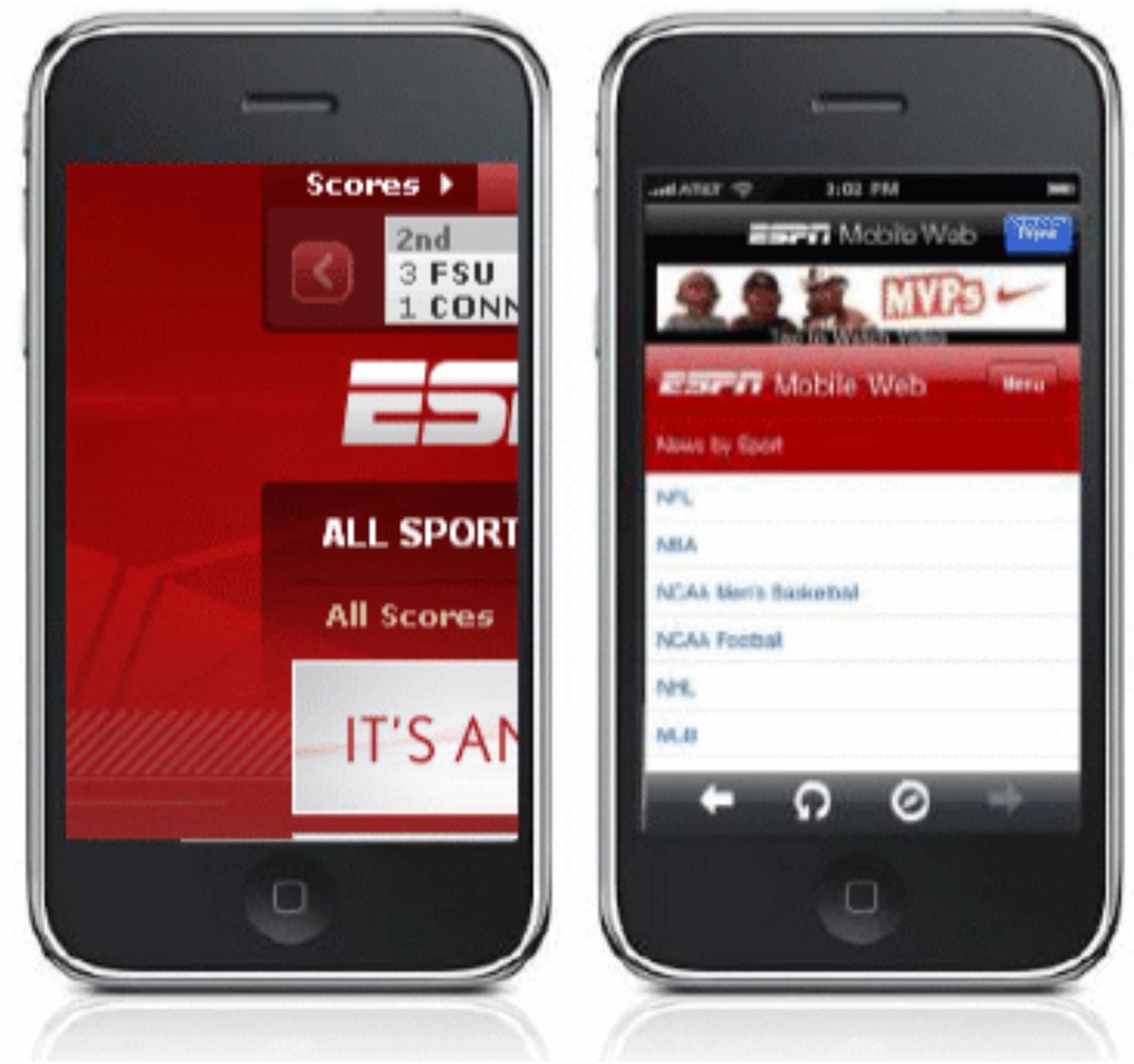
Phone



Internet

**"There's an  
app for that."**

Mobile websites were  
less usable



Websites were **less**  
capable

Image from <https://dribbble.com/shots/2518631-Notifications-Illustration-Animation>



Websites were harder  
to access



Google

www.google.co.in/



Search

Google

Cannot Open Page

Safari cannot open the page  
because the address is invalid.

OK



Restaurants



Coffee



Bars



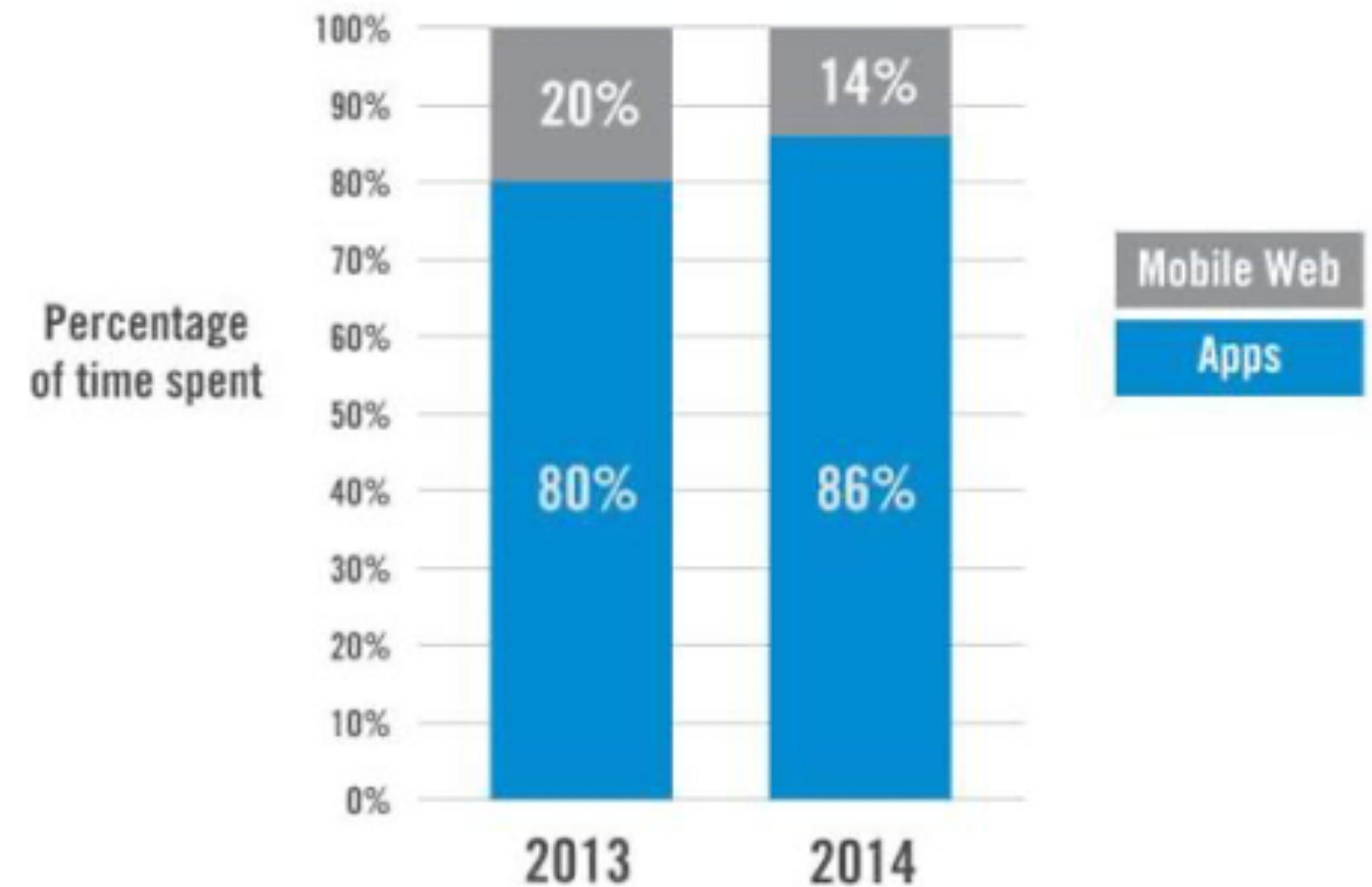
Fast Food



Near

Websites had no  
offline support

On mobile, people spent more time on apps compared to the web



Graph from <https://www.flurry.com/blog/apps-solidify-leadership-six-years-into-the-mobile>

# Rise of the “mobile-first” apps



# Websites in 2016

 Broad reach

# Mobile Apps in 2016

 More usable

 More capable

 Easier to access

 Available offline

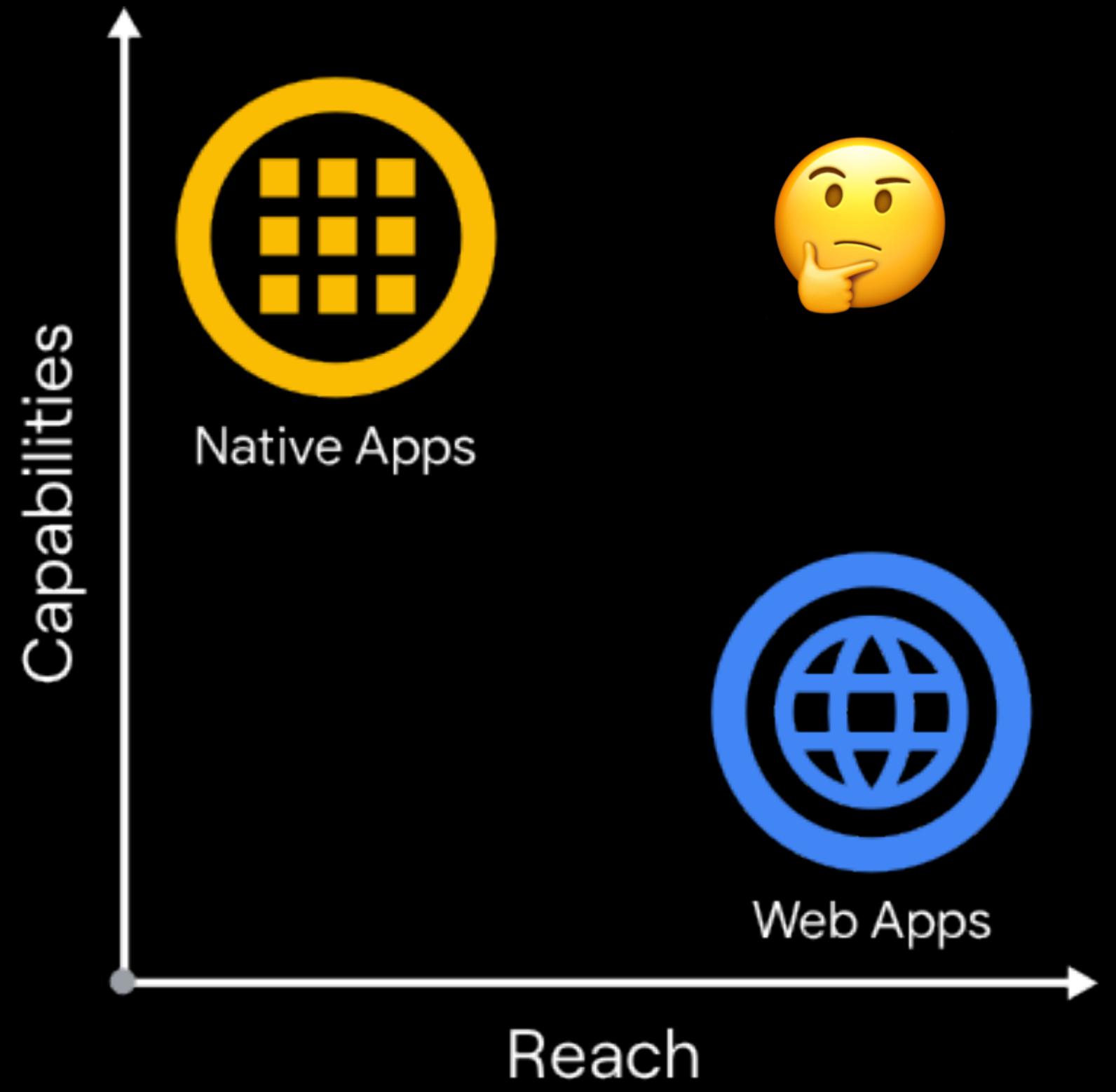


Image from <https://web.dev/what-are-pwas>

Progressive Web Apps use modern web capabilities to deliver an app-like user experience. They evolve from pages in browser tabs to immersive, top-level apps, maintaining the web's low friction at every moment.

[https://developers.google.com/web/updates/2015/12/getting-started-pwa#what\\_is\\_a\\_progressive\\_web\\_app](https://developers.google.com/web/updates/2015/12/getting-started-pwa#what_is_a_progressive_web_app)

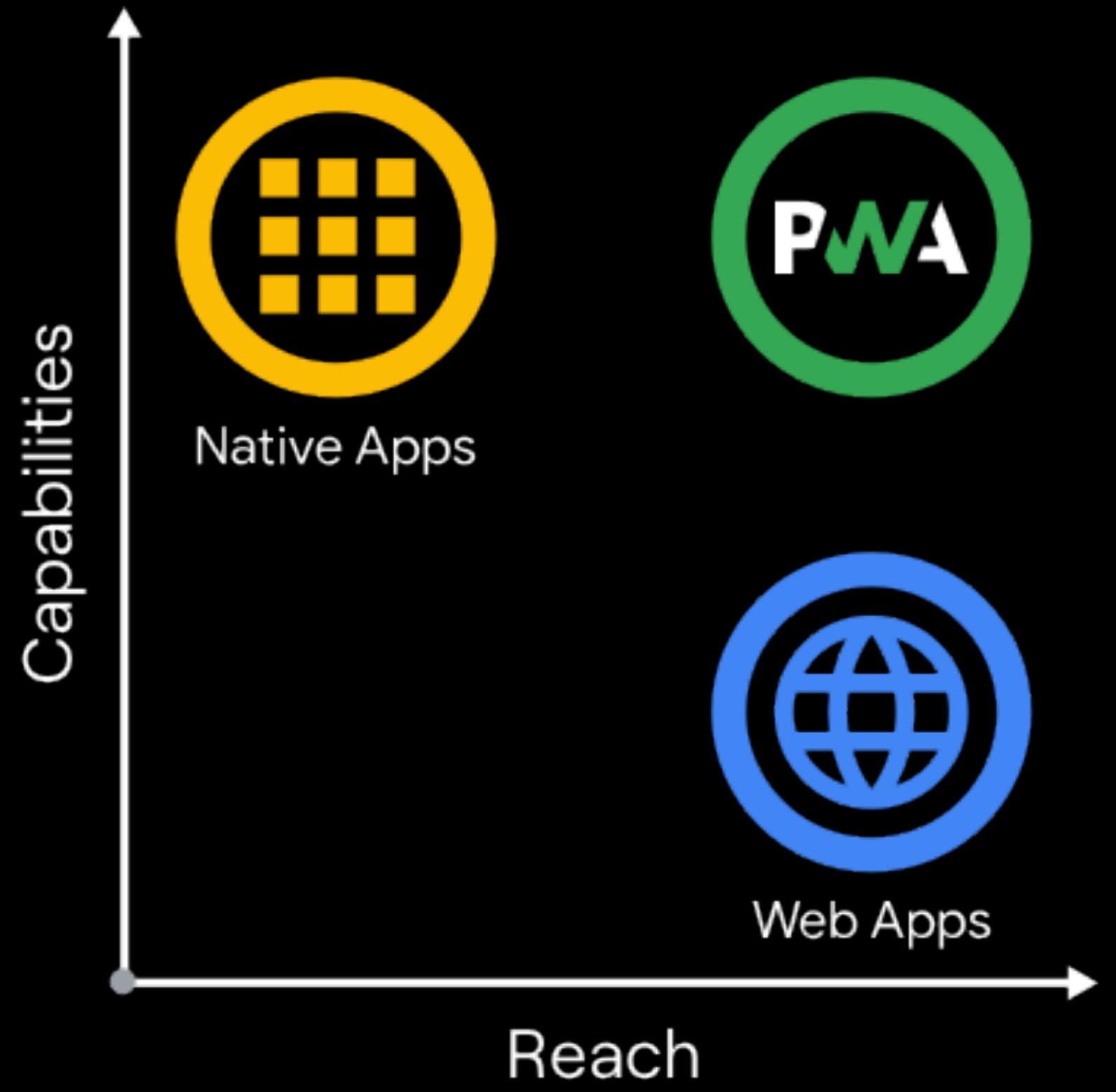


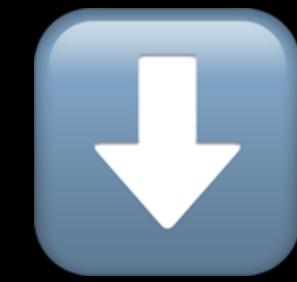
Image from <https://web.dev/what-are-pwas>



Capable

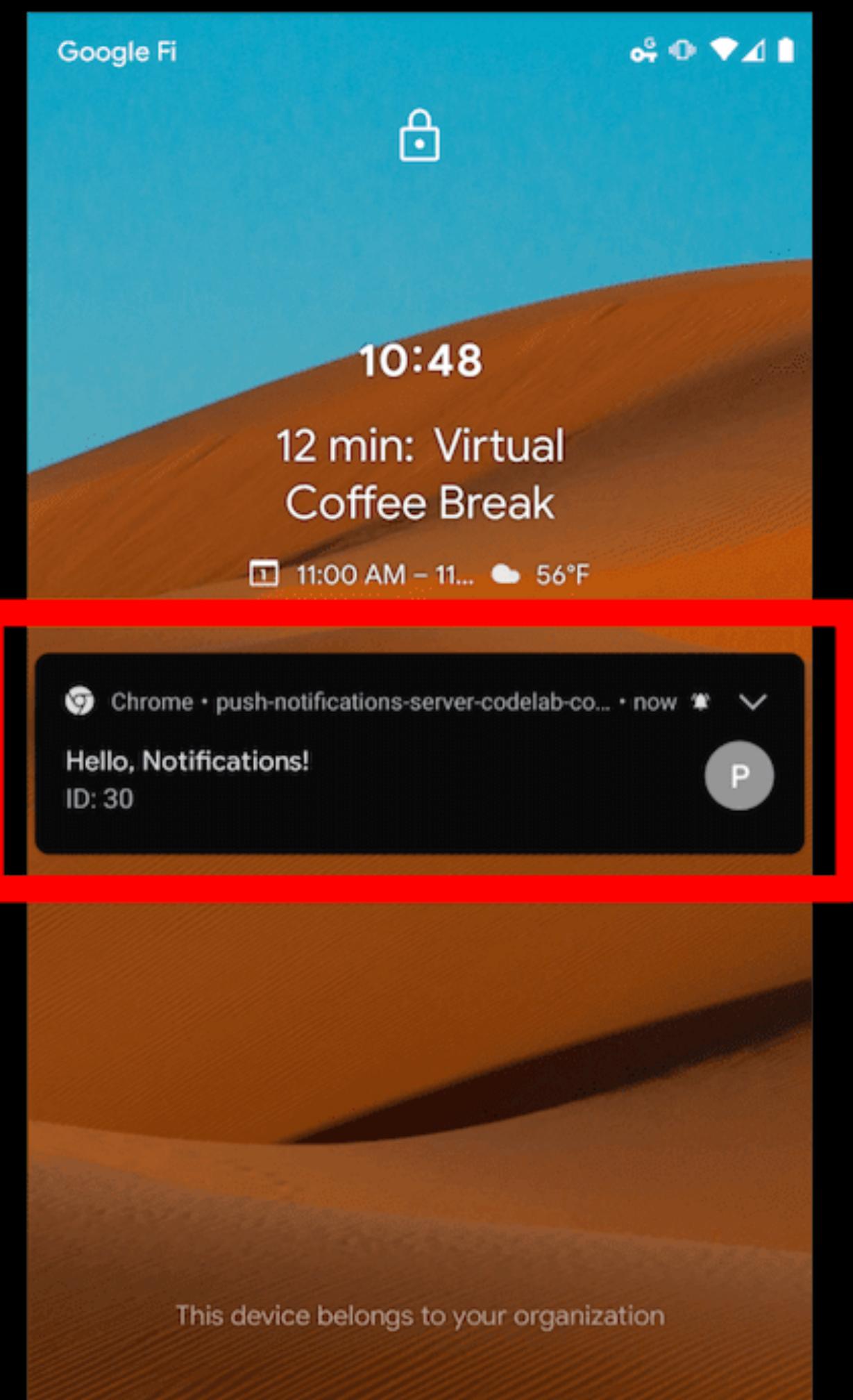
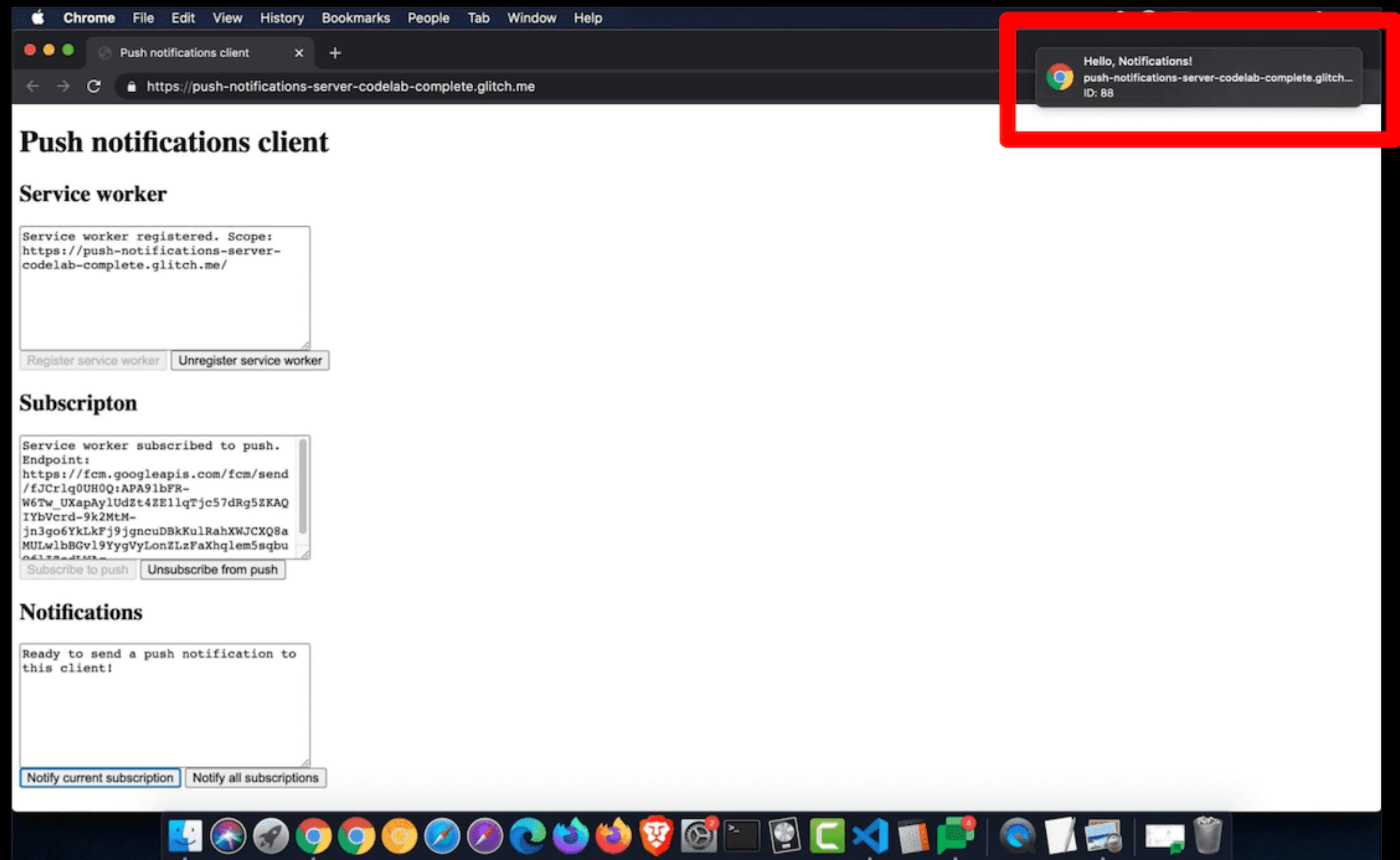


Reliable



Installable

**Capable** – uses modern APIs to deliver  
an app-like experience

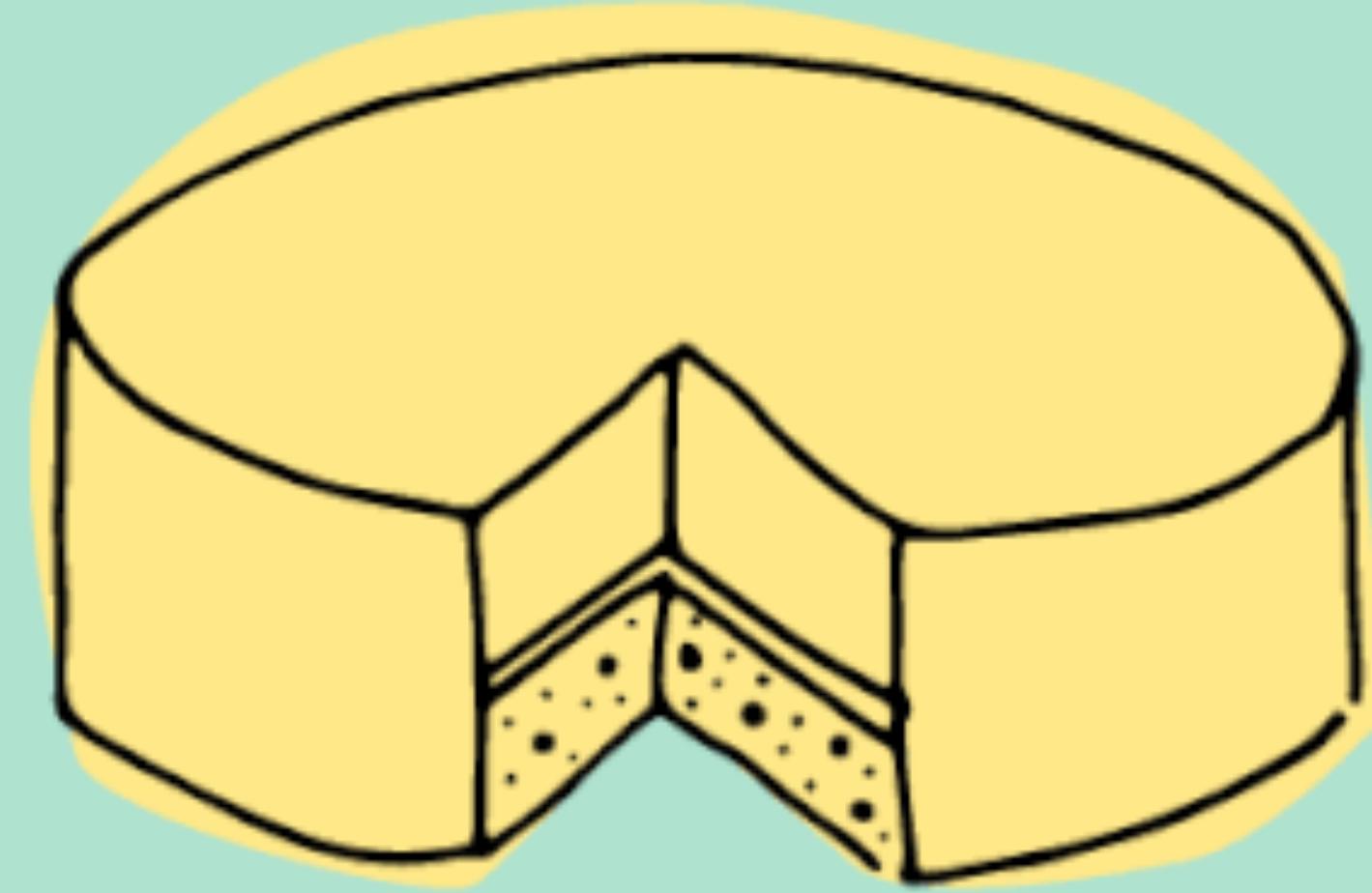


**Reliable** – works well regardless of  
network

OFFLINE

**Installable** – can be installed to the device and used outside the browser





Standard website



Fully capable, reliable,  
and installable PWA

How did PWAs fare in 2016?

Capable



Reliable

Installable

# Critical capabilities

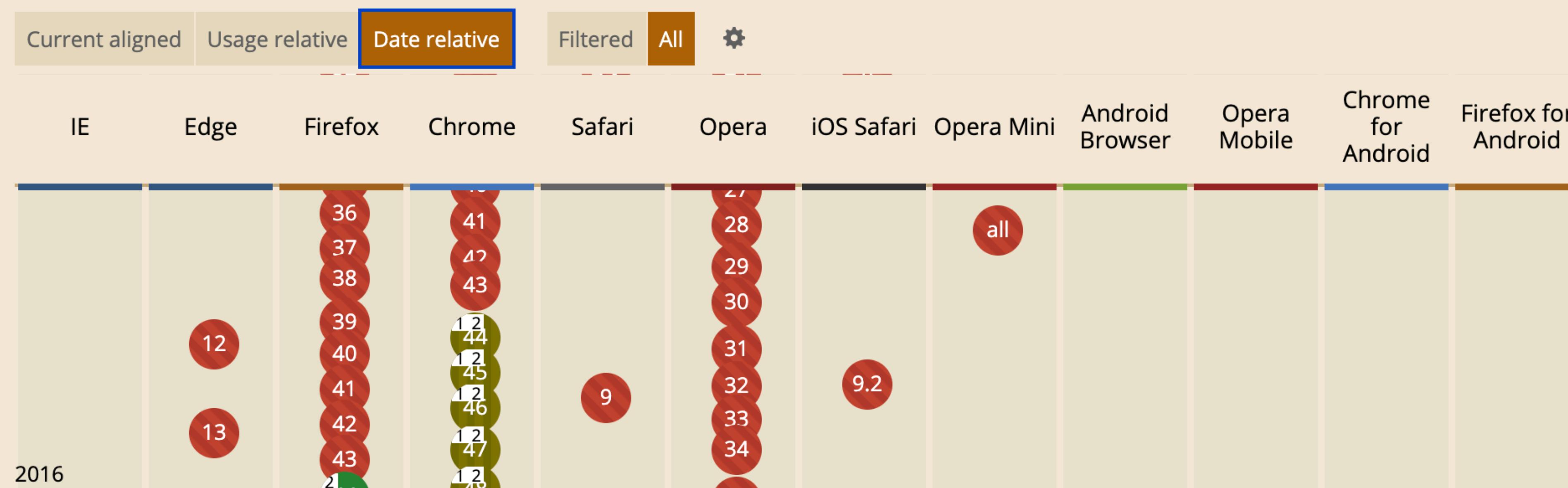
Push notifications & badging

Background sync & fetch

Access to device - files, contacts, settings, etc.

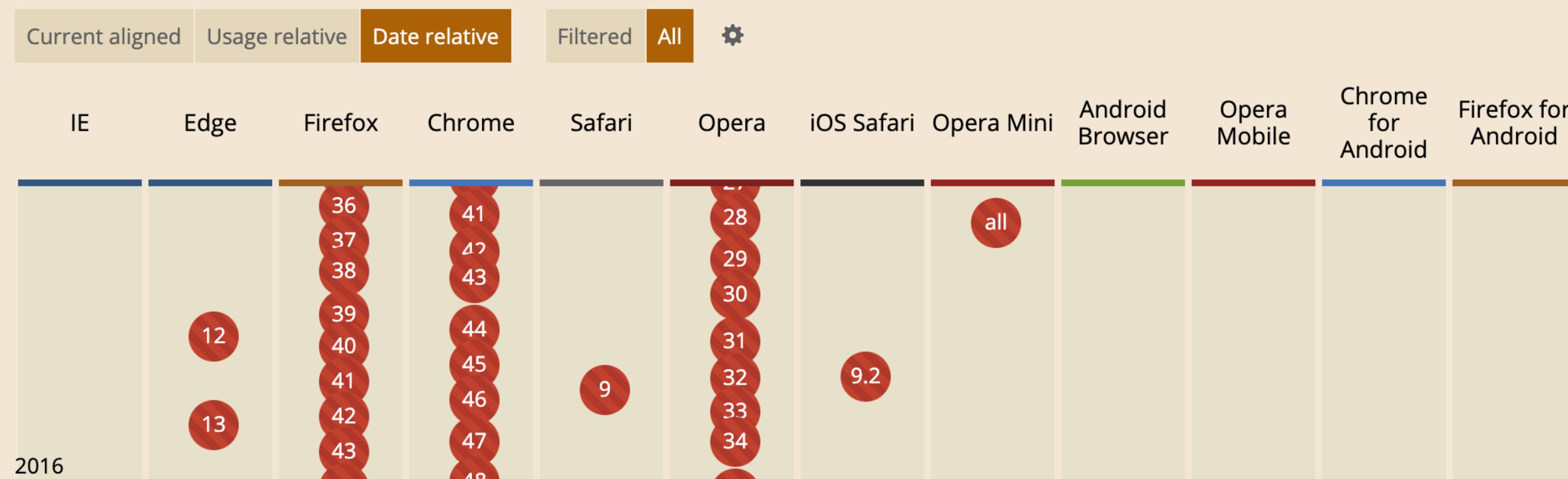
## Push API 📄 - WD

API to allow messages to be pushed from a server to a browser, even when the site isn't focused or even open in the browser.



# Background Sync API

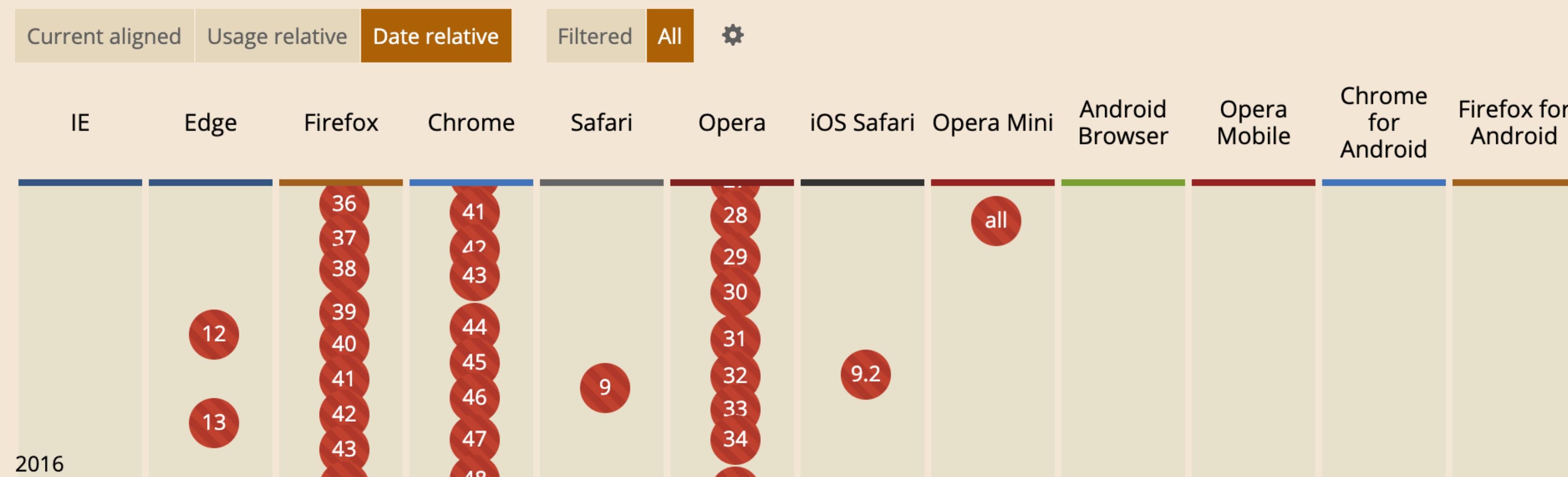
Provides one-off and periodic synchronization for Service Workers with an `onsync` event.



# File System Access API

📄 - UNOFF

API for manipulating files in the device's local file system (not in a sandbox).



```
if (awesomeFeature in window) {  
    // do awesome thing  
}  
else {  
    // same old thing  
}
```

```
if (awesomeFeature in window) {
```

```
    // do awesome thing
```

```
} else {
```

```
    // same old thing
```

```
}
```

Capable



Reliable

Installable

Capable



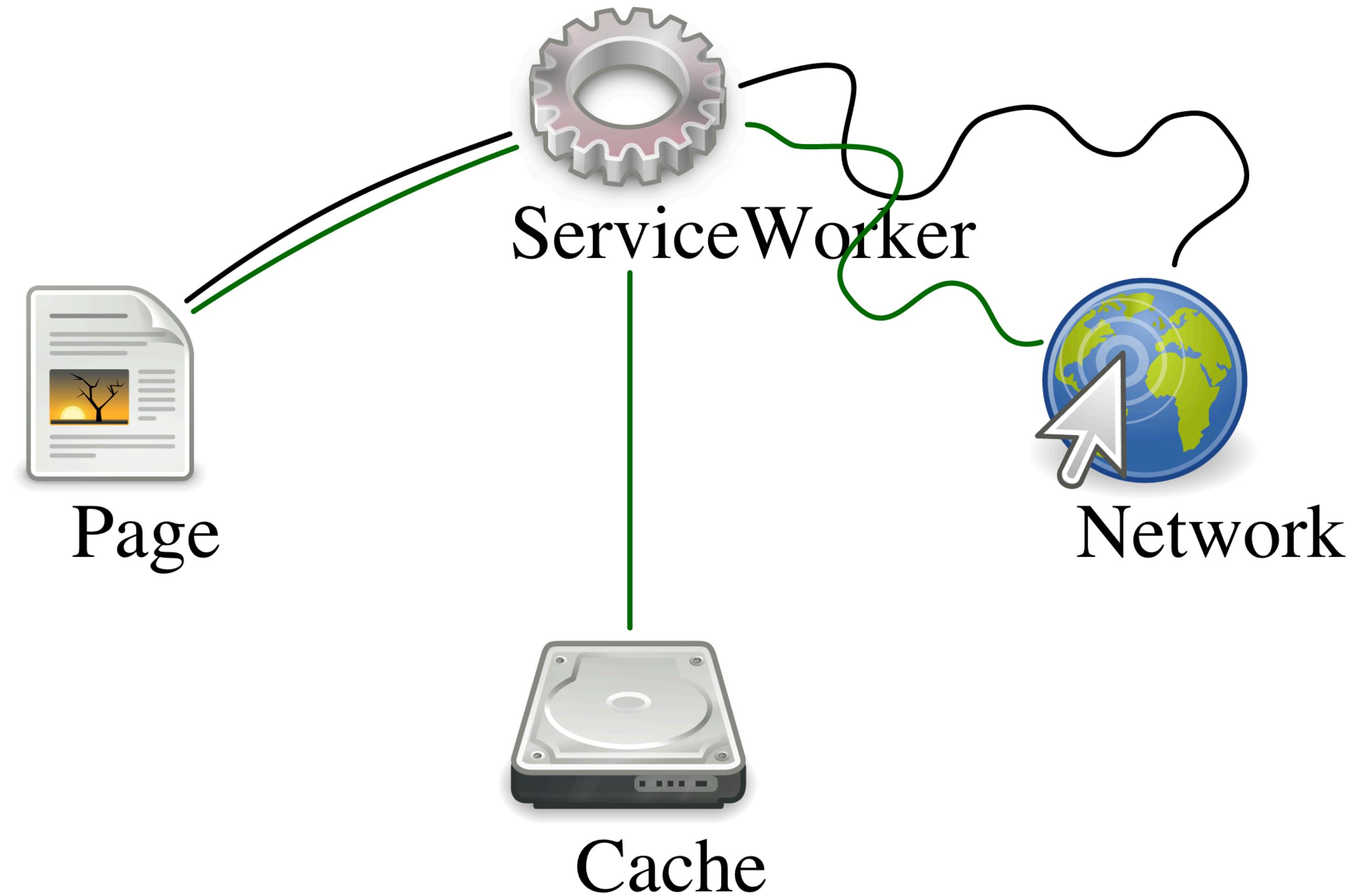
Reliable



Installable

A service worker is a script that your browser runs in the background, separate from a web page, opening the door to features that don't need a web page or user interaction

<https://developers.google.com/web/fundamentals/primers/service-workers>



## Service Workers 📁 - CR

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

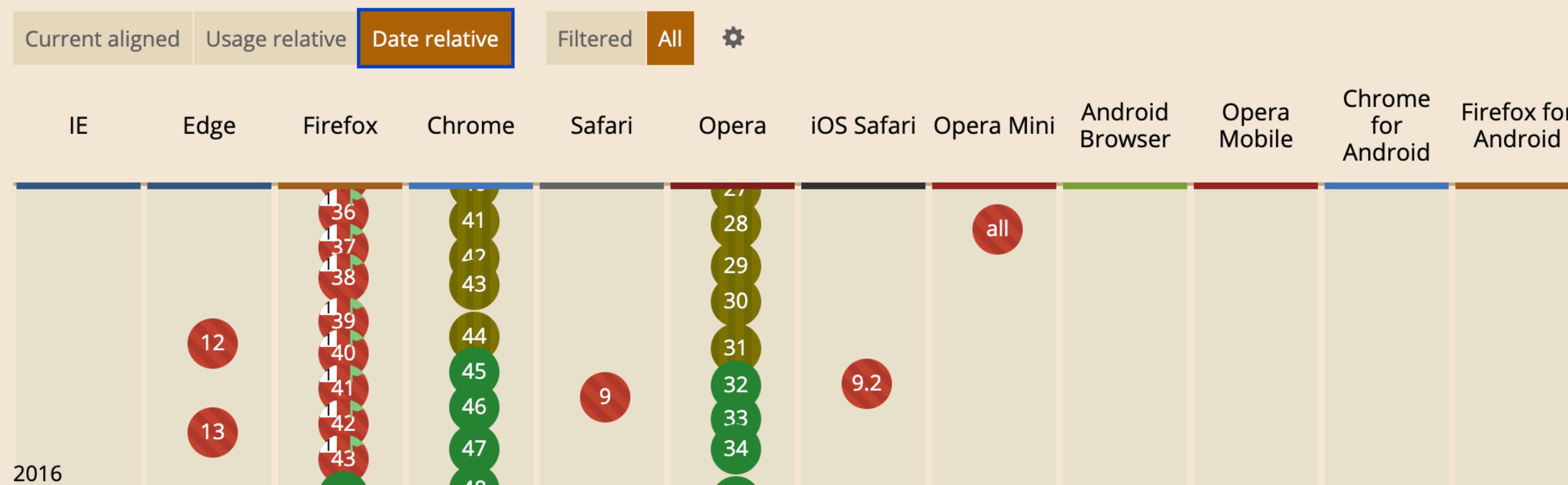
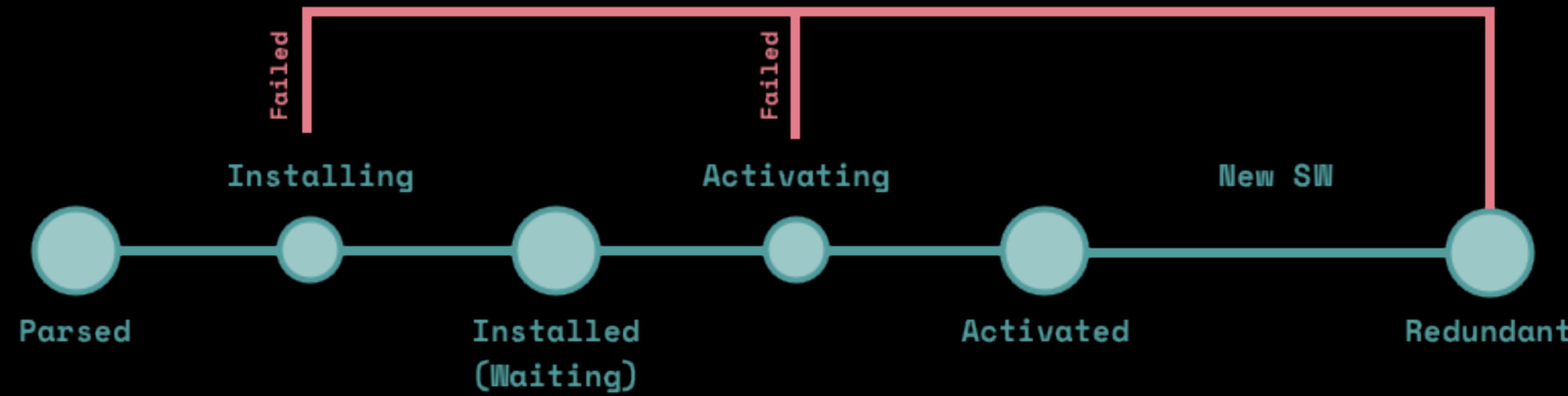
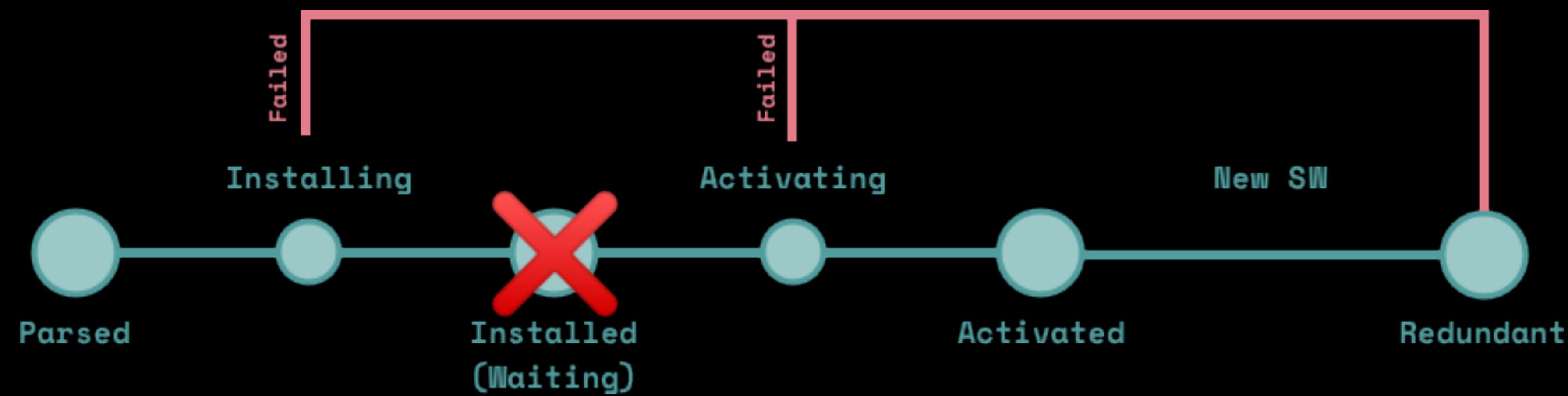




Image from <https://explore.easyprojects.net/blog/how-to-deal-with-difficult-team-members>





self.skipWaiting()

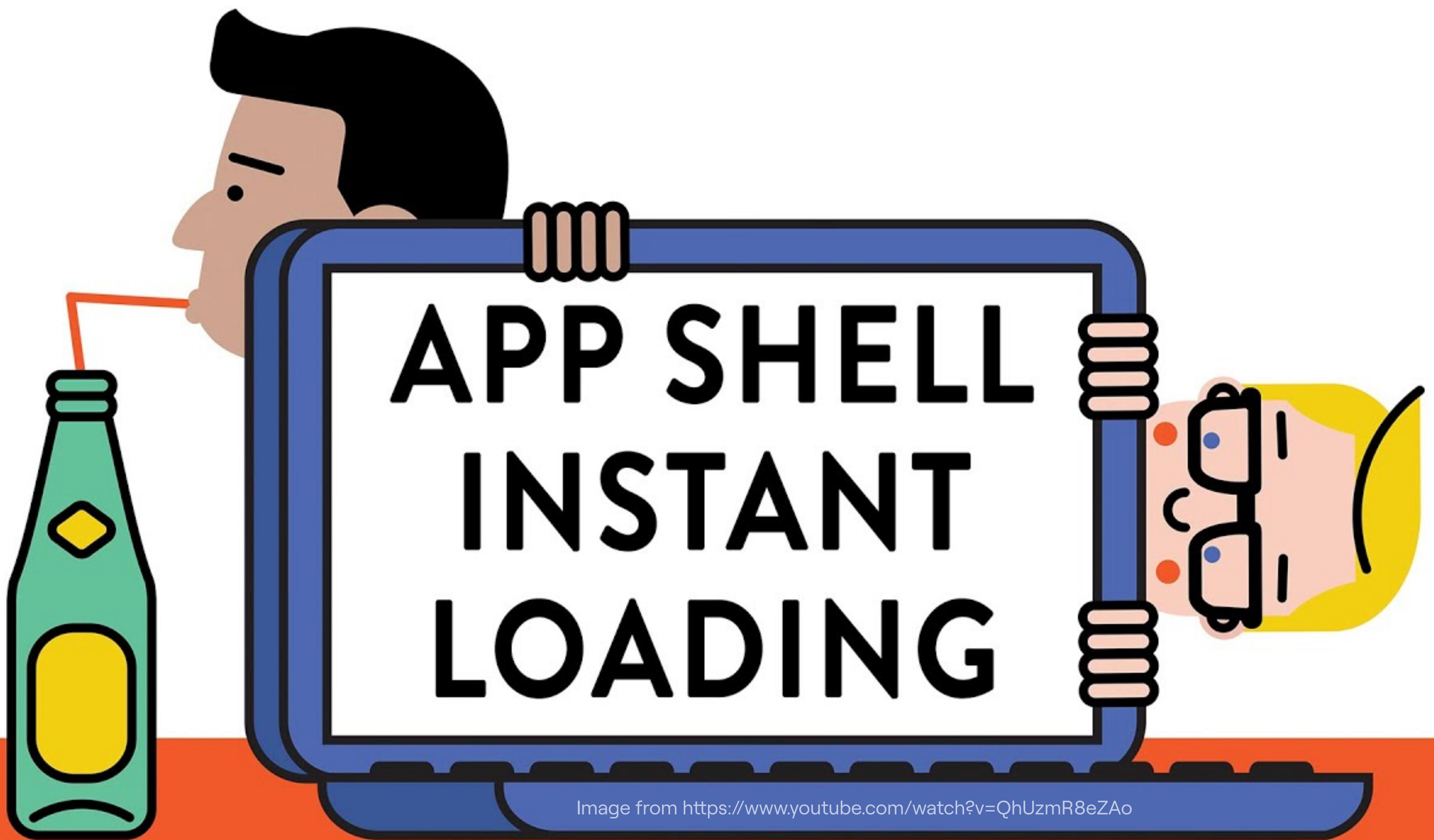


Image from <https://www.youtube.com/watch?v=QhUzmR8eZAo>

The app "shell" is the **minimal HTML, CSS and JavaScript required to power the user interface** and when cached offline can ensure instant, reliably good performance to users on repeat visits. This means the application shell is not loaded from the network every time the user visits. Only the necessary content is needed from the network.

<https://developers.google.com/web/fundamentals/architecture/app-shell>

## Latest Articles



Latest



Home



Bookmarks

## Latest Articles

### Variable and Function Hoisting in ES2015

⌚ Last Tuesday 📷 javascript

Variable hoisting is a behaviour in JavaScript where variable declarations are moved to the top of the scope (function scope or global scope) that the variable is defined within. The typical JavaScript variable can be created in two stages - declaration and initialisation.

The declaration stage is where the variable....

— Read

■ Bookmark

### The New System Font Stack?

⌚ Sep 20th, 2016 📷 css

A few months ago, I wrote about how you can use system fonts in the browser using the built-in keywords that work with the font shorthand property ([See Using System Fonts in the Browser](#)). These



Latest



Home



Bookmarks

# App shell

```
<main>

  <h2>Latest Articles</h2>

  <section id="excerpts">

    <!-- loading icon -- >

  </section>

</main>
```

# Respond with cached page

```
self.addEventListener('fetch', (event) => {  
  event.respondWith(  
    caches.match(event.request)  
      .then((cachedResponse) => cachedResponse || fetch(event.request))  
  );  
});
```

# Add content to app shell

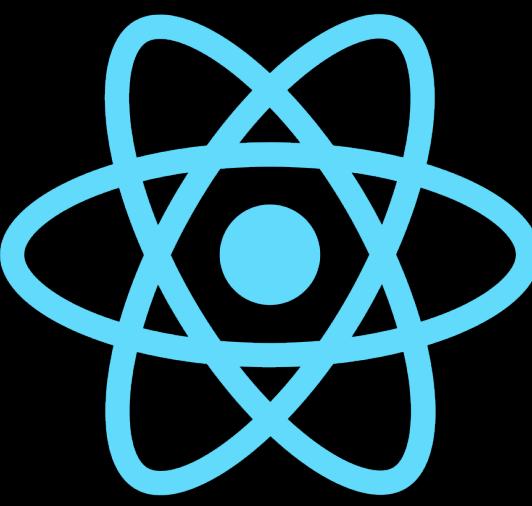
```
if ( 'serviceWorker' in navigator ) {  
  
  getArticlesFromIndexedDB()  
  
    .then((articles) => displayArticles(articles))  
  
    .then(() => updateArticlesInBackground());  
  
} else {  
  
  fetchArticlesFromNetwork()  
  
    .then((articles) => displayArticles(articles));  
  
}
```

# Add content to app shell

```
if ( 'serviceWorker' in navigator ) {  
  
  getArticlesFromIndexedDB()  
  
  .then((articles) => displayArticles(articles))  
  
  .then(() => updateArticlesInBackground());  
  
} else {  
  
  fetchArticlesFromNetwork()  
  
  .then((articles) => displayArticles(articles));  
  
}
```

# Add content to app shell

```
if ( 'serviceWorker' in navigator ) {  
  
  getArticlesFromIndexedDB()  
  
  .then((articles) => displayArticles(articles))  
  
  .then(() => updateArticlesInBackground());  
  
} else {  
  
  fetchArticlesFromNetwork()  
  
  .then((articles) => displayArticles(articles));  
  
}
```



Capable



Reliable



Installable

Capable



Reliable

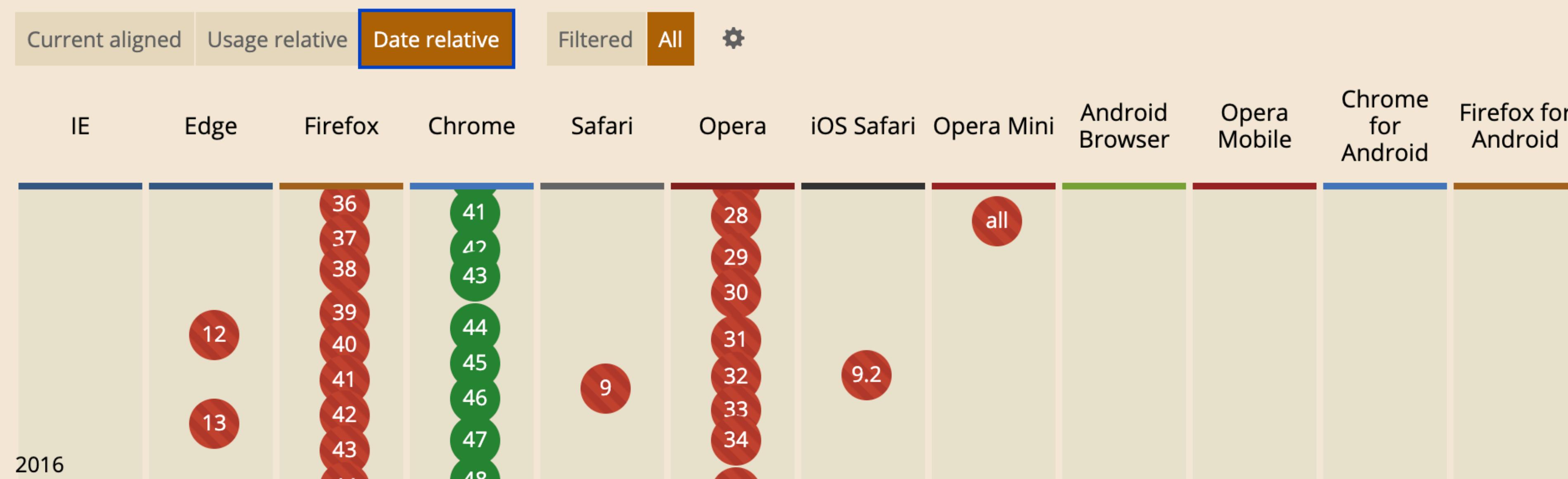


Installable



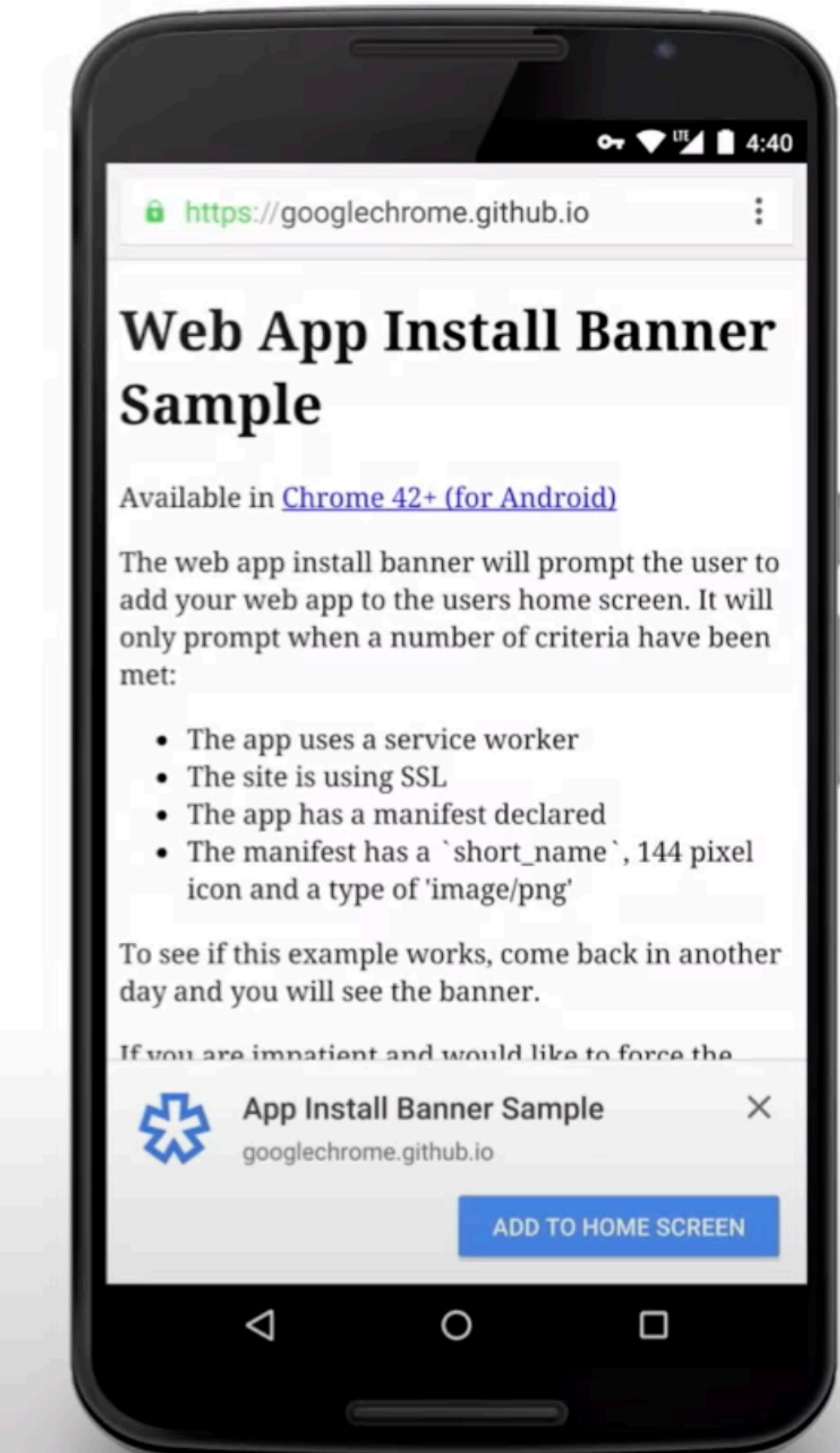
## Add to home screen (A2HS) - WD

The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load its assets through a [Service Worker](#).



# Requirements for “Add to Home Screen”

1. A manifest.json file
2. A service worker
3. Visit frequency heuristics 🤔



Add to Home Screen != Install

Capable



Reliable



Installable



# PWAs five years on

[instagram.com](https://instagram.com)[tiktok.com](https://tiktok.com)[mobile.twitter.com/home](https://mobile.twitter.com/home)

Instagram



Your Story



theecono...



officiallyk10



amxra.a



bbcafrica

**E** theeconomist

Install app



Instagram

[www.instagram.com](https://www.instagram.com)[Cancel](#)[Install](#)

Install app



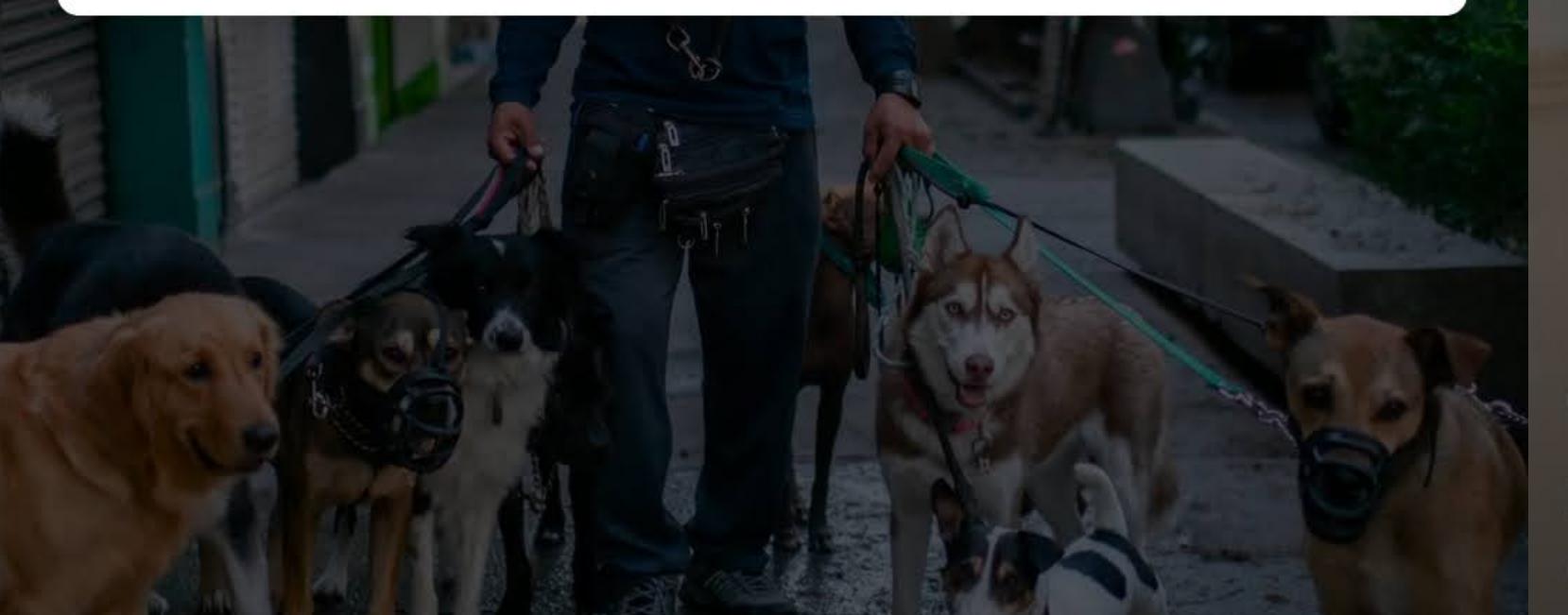
TikTok

[www.tiktok.com](https://www.tiktok.com)[Cancel](#)[Install](#)

Install app



Twitter

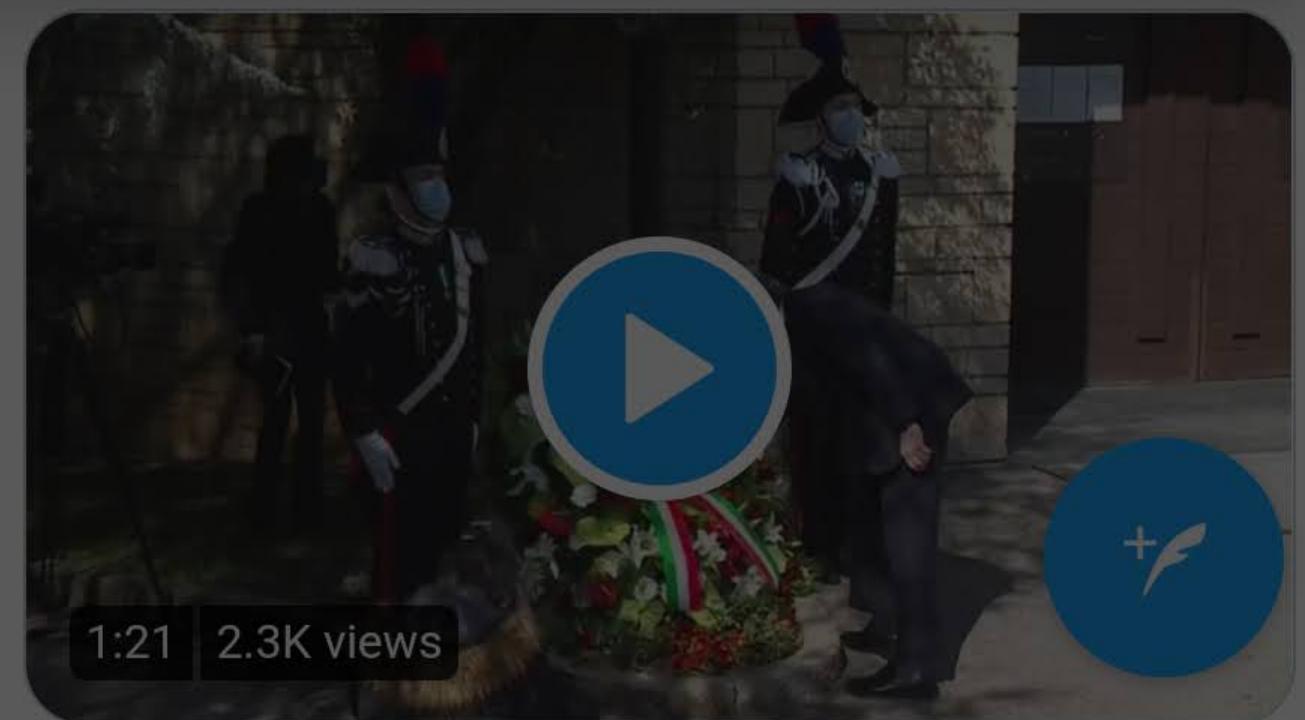
[mobile.twitter.com](https://mobile.twitter.com)[Cancel](#)[Install](#)[Use the App](#)

@keemokazi

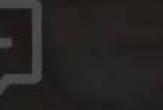
I guess you can say she was embarrassed 😂 #sister #siblings #foryou #fyp

original sound - KEEMOKAZI

origin



1:21 2.3K views



~1,500,000 websites may be installable on  
mobile home screens, offering an app  
experience

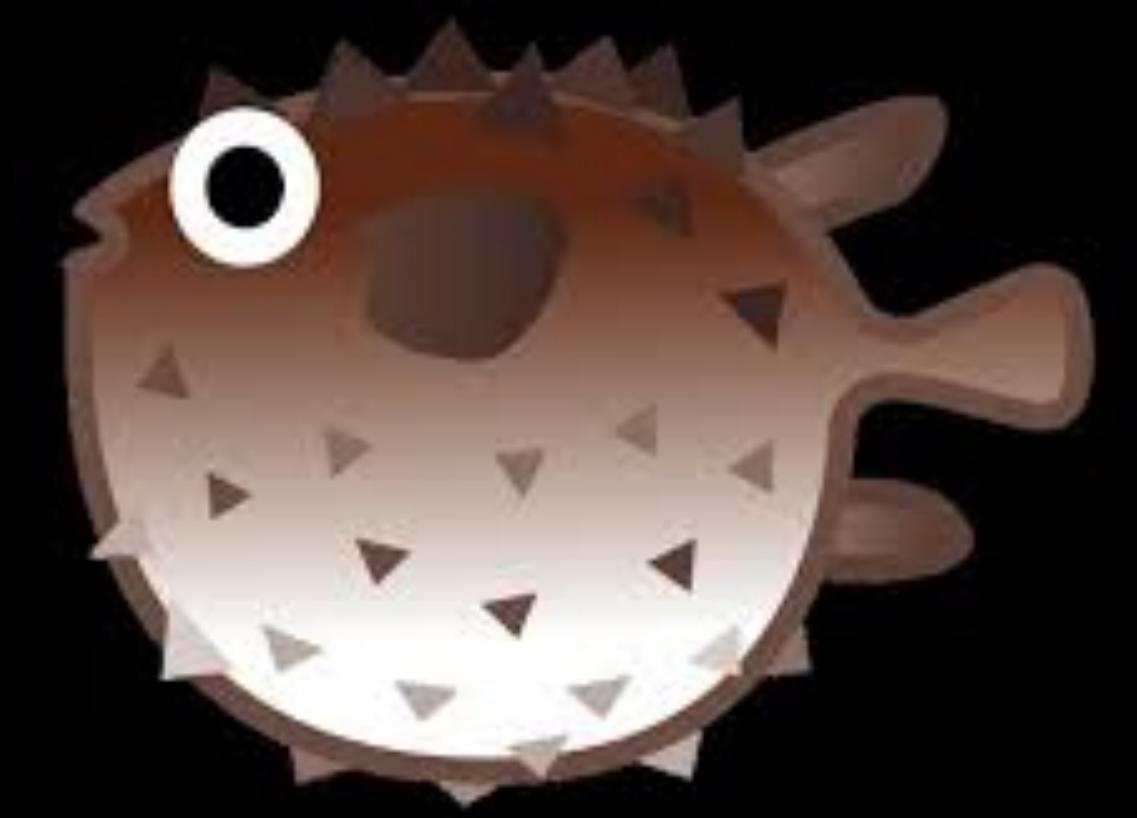
<https://firt.dev/pwa-2021>

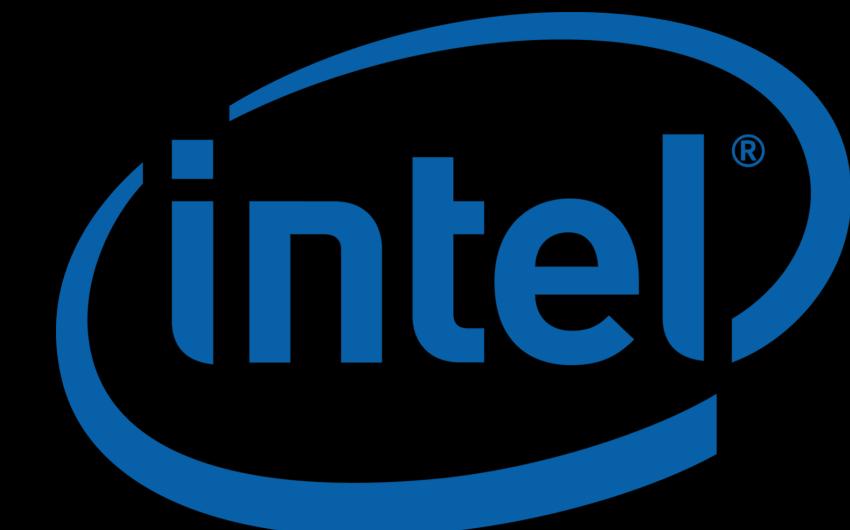
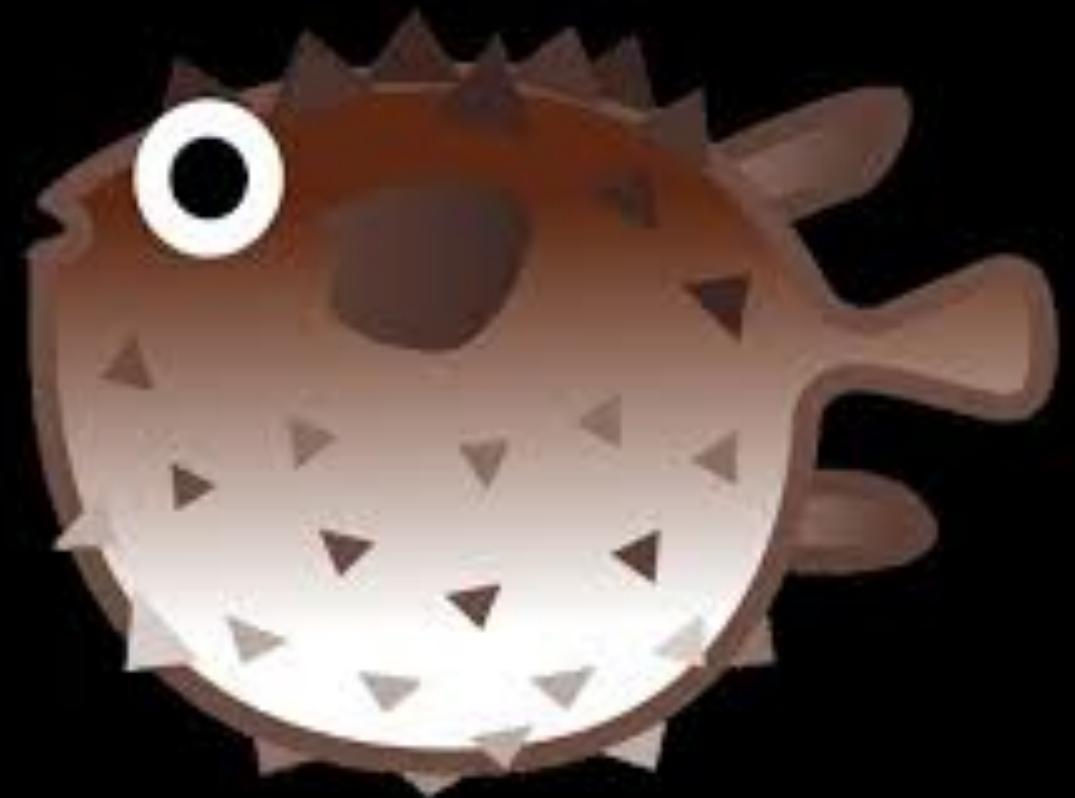
Capable



Reliable

Installable





SAMSUNG

Project Fugu is an effort to **close gaps in  
the web's capabilities** by enabling new  
classes of applications to run on the web

<http://www.chromium.org/teams/web-capabilities-fugu>

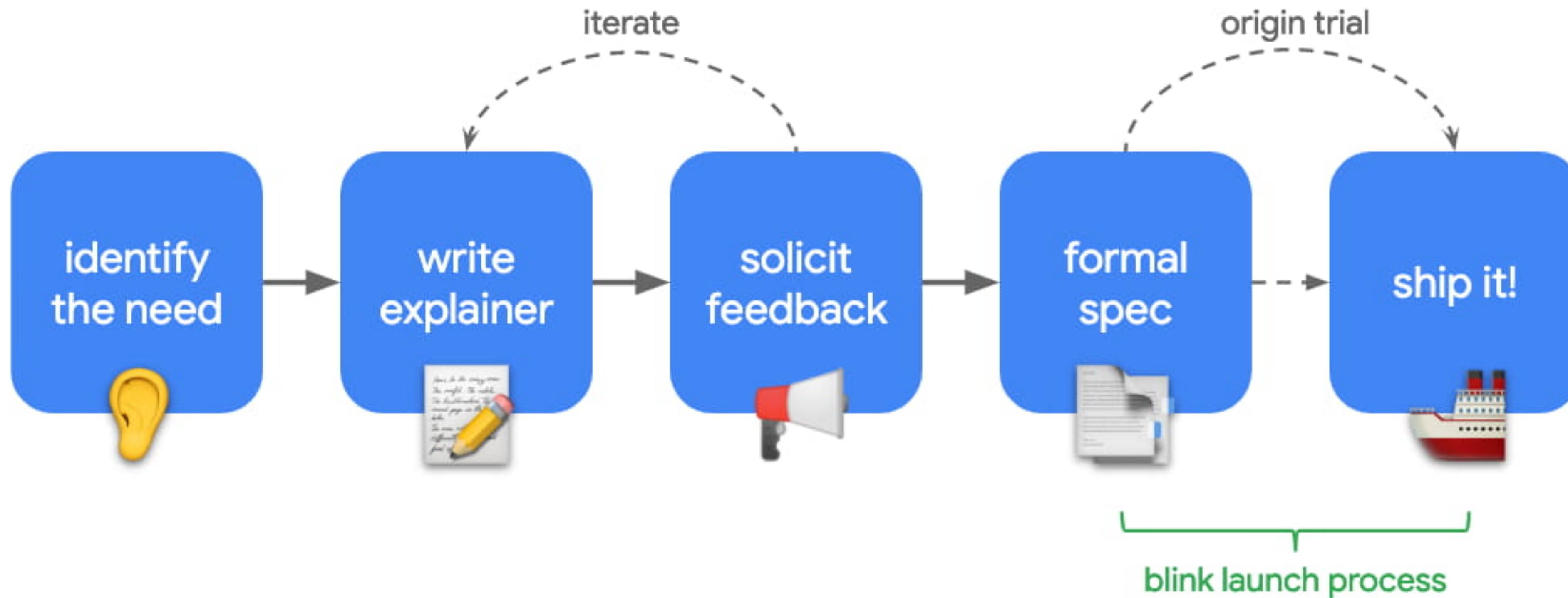


Image from <https://developers.google.com/web/updates/capabilities>

What has been shipped? 

# Critical capabilities

Push notifications & badging

Background sync & fetch

Access to device - files, contacts, settings, etc.

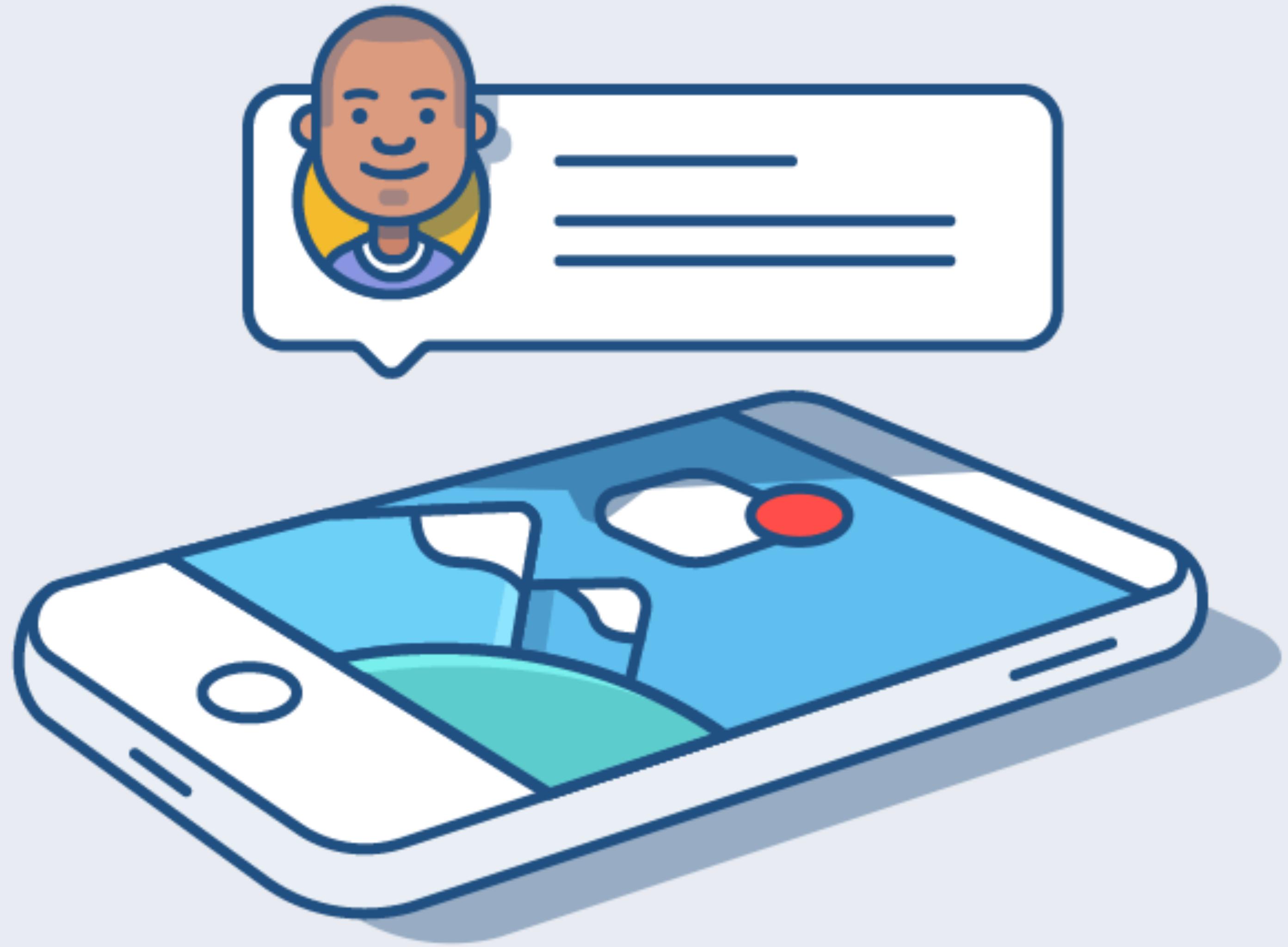


Image from <https://dribbble.com/shots/2518631-Notifications-Illustration-Animation>

# APIs

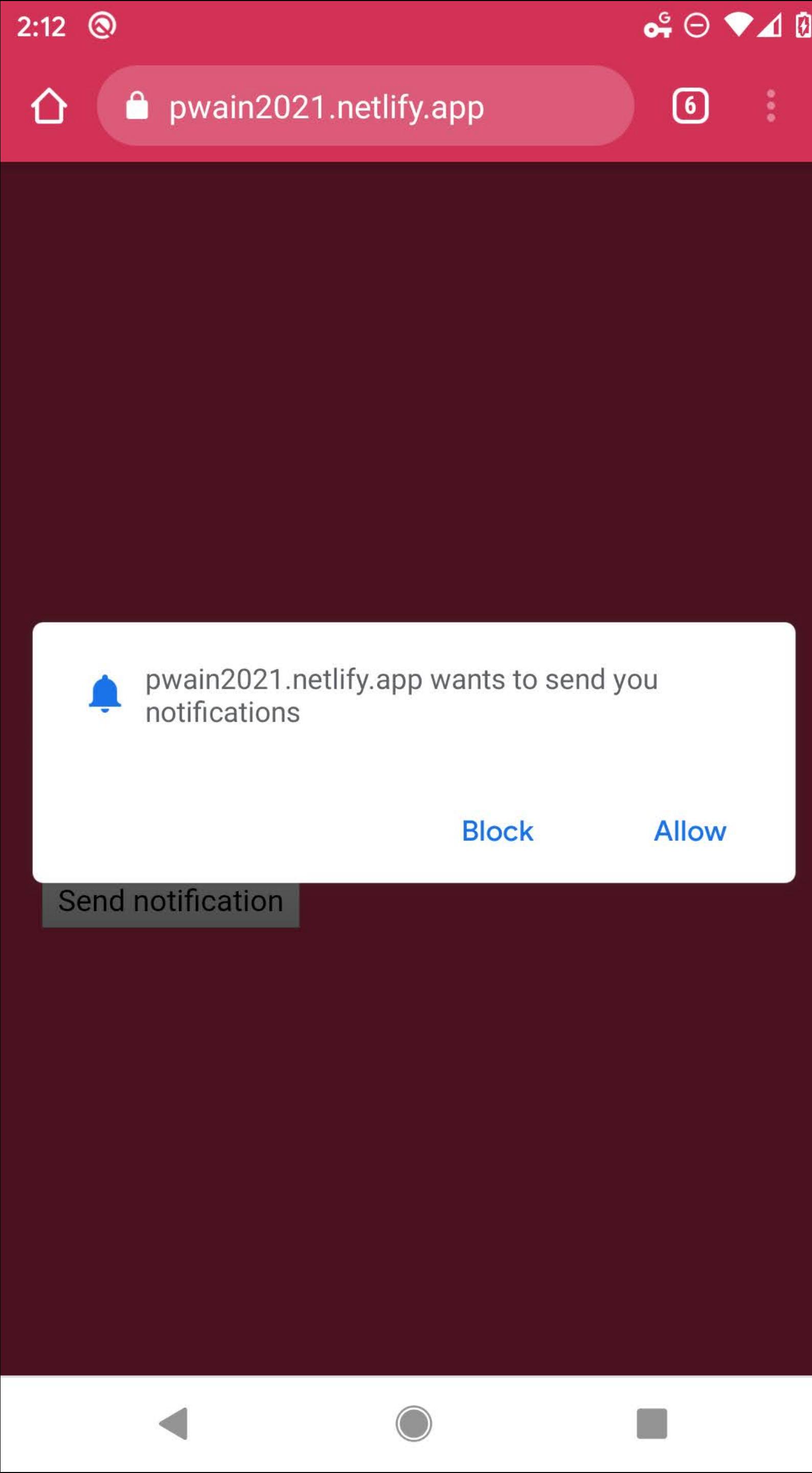
Notifications API – display of notification message

Push API – allows browsers to receive messages from a server

App Badging API – display badge on app icon

# Request permission to display push notifications

```
if ('Notification' in window) {  
  
  Notification.requestPermission();  
  
}  
}
```



2:23

# Display notification

```
navigator.serviceWorker.ready.then((registration) => {
```

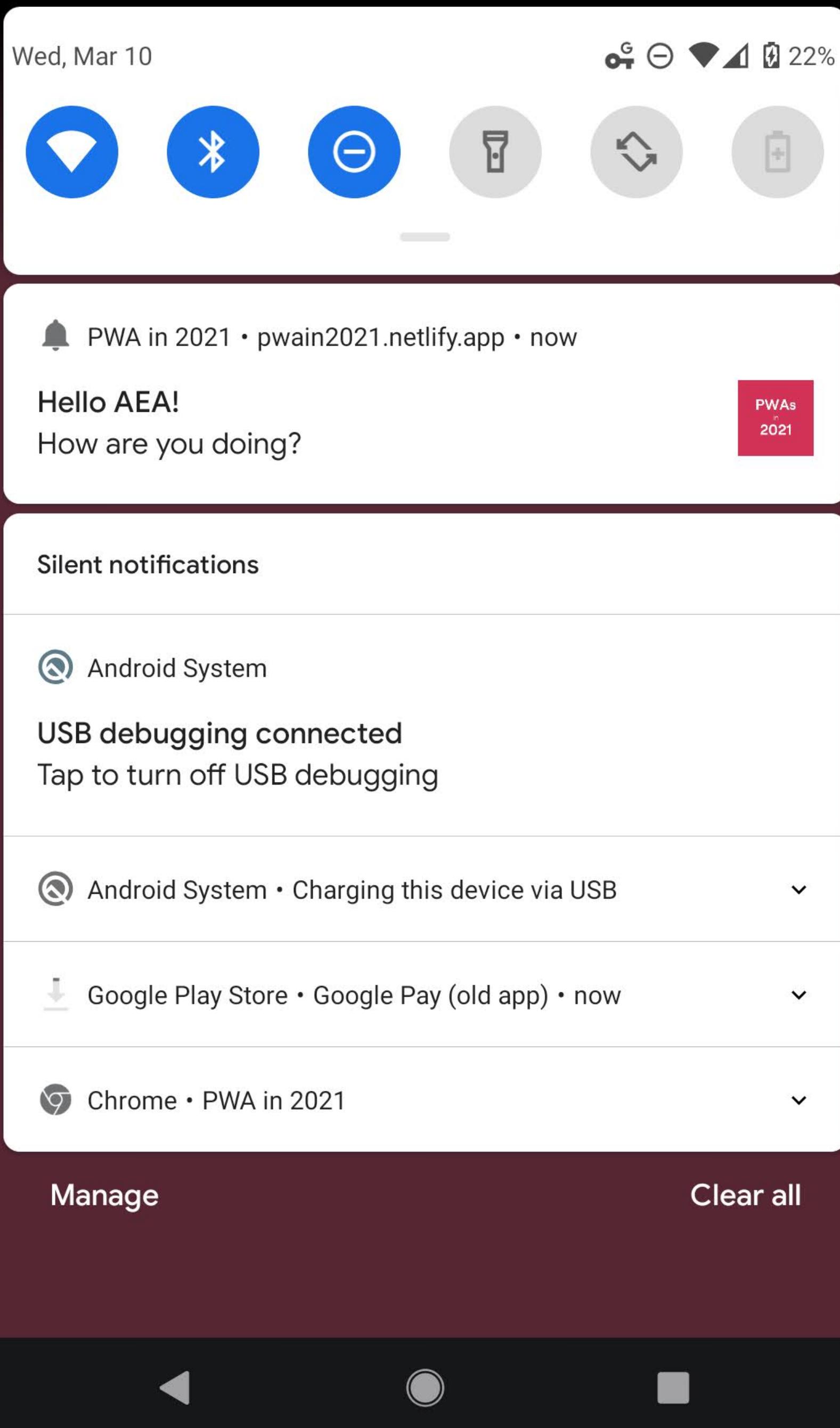
```
  registration.showNotification(
```

```
    'Hello AEA!',
```

```
    { body: 'How are you doing?' }
```

```
);
```

```
});
```





*1. Get Permission to  
Send Push Messages*



*2. Get PushSubscription*



*3. Send PushSubscription  
to Your Server*

# Get and send PushSubscription to server

```
const registration = await navigator.serviceWorker.register('/service-worker.js');

const pushSubscription = await registration.pushManager.subscribe({
    userVisibleOnly: true,
    applicationServerKey: /* VAPID key */
});

saveToServer(pushSubscription);
```

# Get and send PushSubscription to server

```
const registration = await navigator.serviceWorker.register('/service-worker.js');

const pushSubscription = await registration.pushManager.subscribe({
  userVisibleOnly: true,
  applicationServerKey: /* VAPID key */
});

saveToServer(pushSubscription);
```

# Get and send PushSubscription to server

```
const registration = await navigator.serviceWorker.register('/service-worker.js');

const pushSubscription = await registration.pushManager.subscribe({
  userVisibleOnly: true,
  applicationServerKey: /* VAPID key */
});

saveToServer(pushSubscription);
```



Your Server



*Web Push Protocol  
Request*



Push Service



Message Arrives  
on the Device

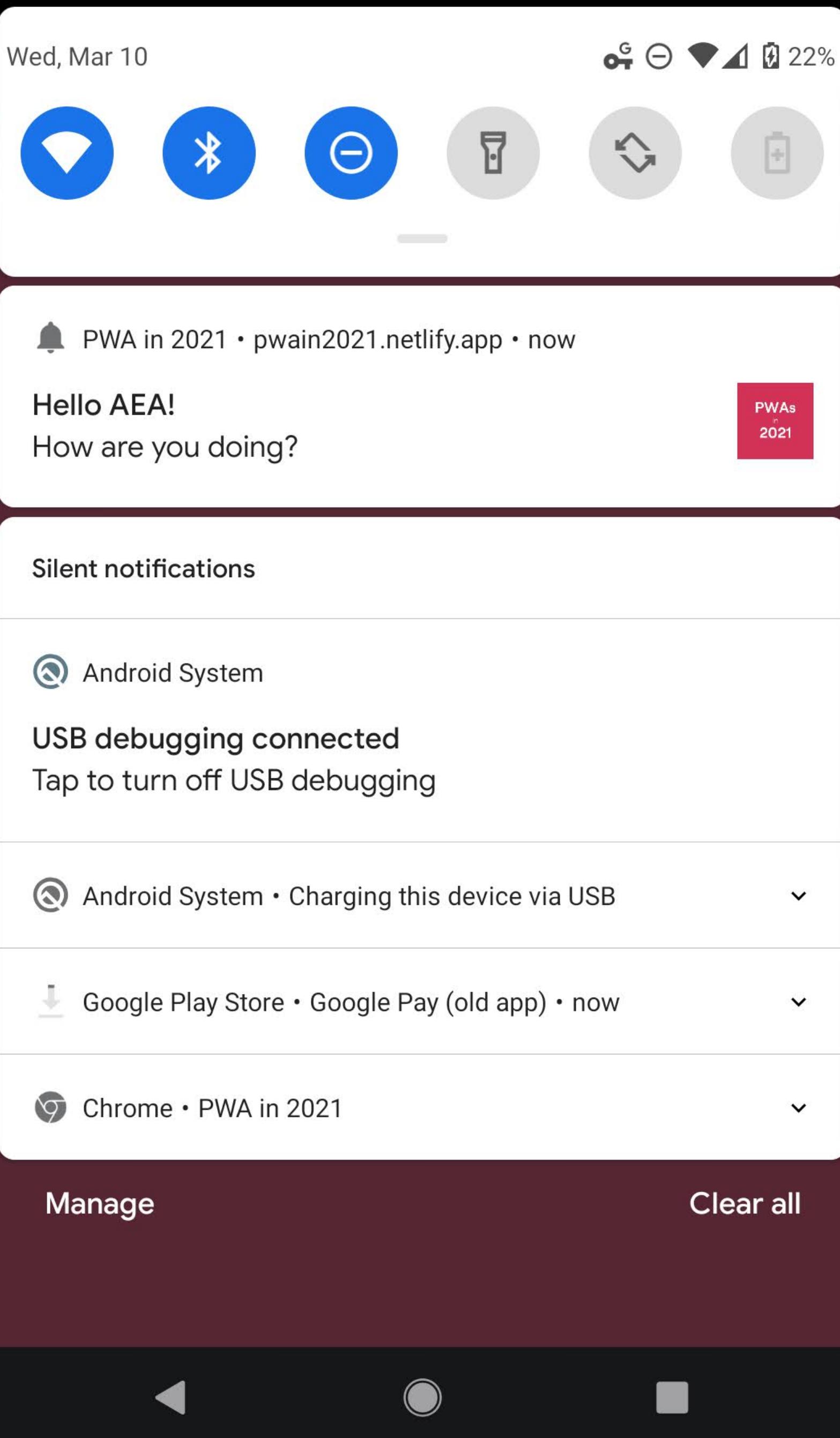
# Receive notification in service worker

```
self.addEventListener('push', function(event) {  
});
```

2:23

# Display notification

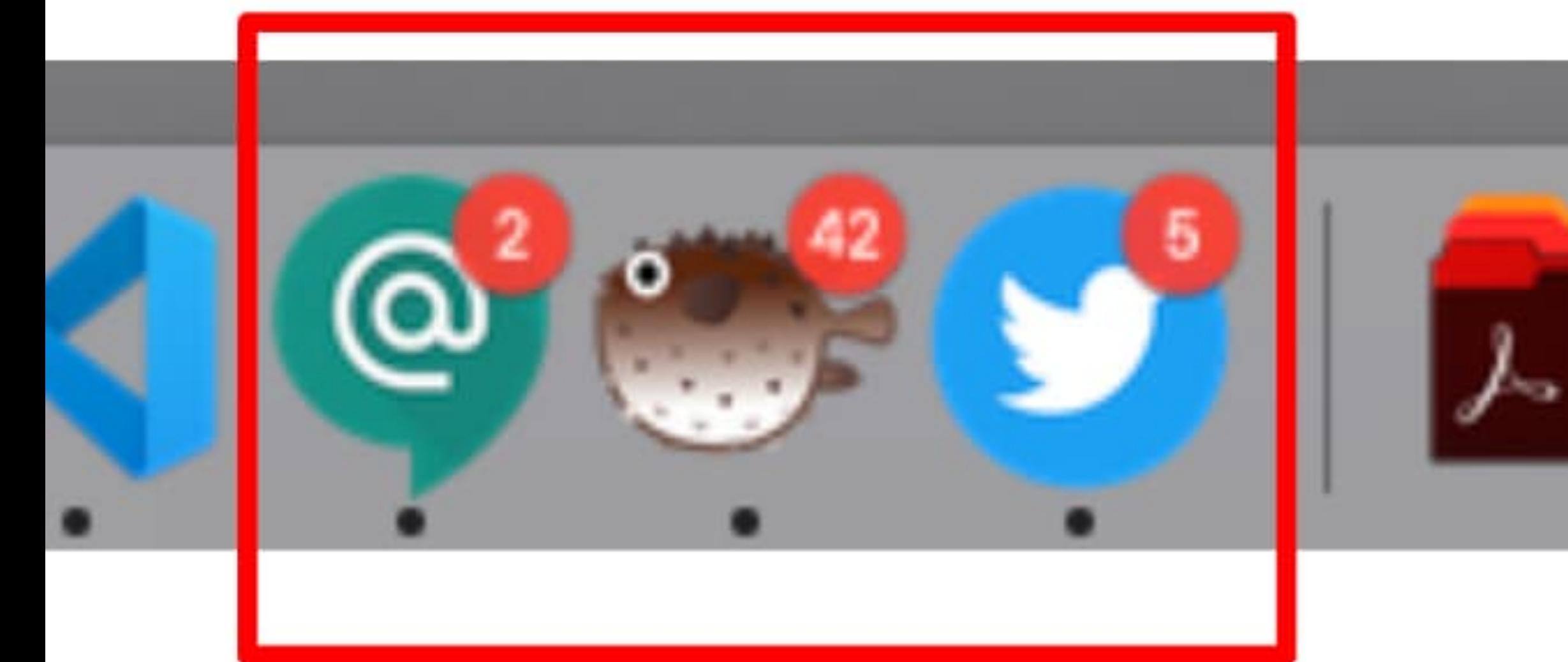
```
self.addEventListener('push', function(event) {  
  event.waitUntil(  
    self.registration.showNotification(  
      'Hello AEA!',  
      { body: 'How are you doing?' }  
    )  
  );  
});
```



# Set app badge

```
navigator.setAppBadge(badgeNumber);
```

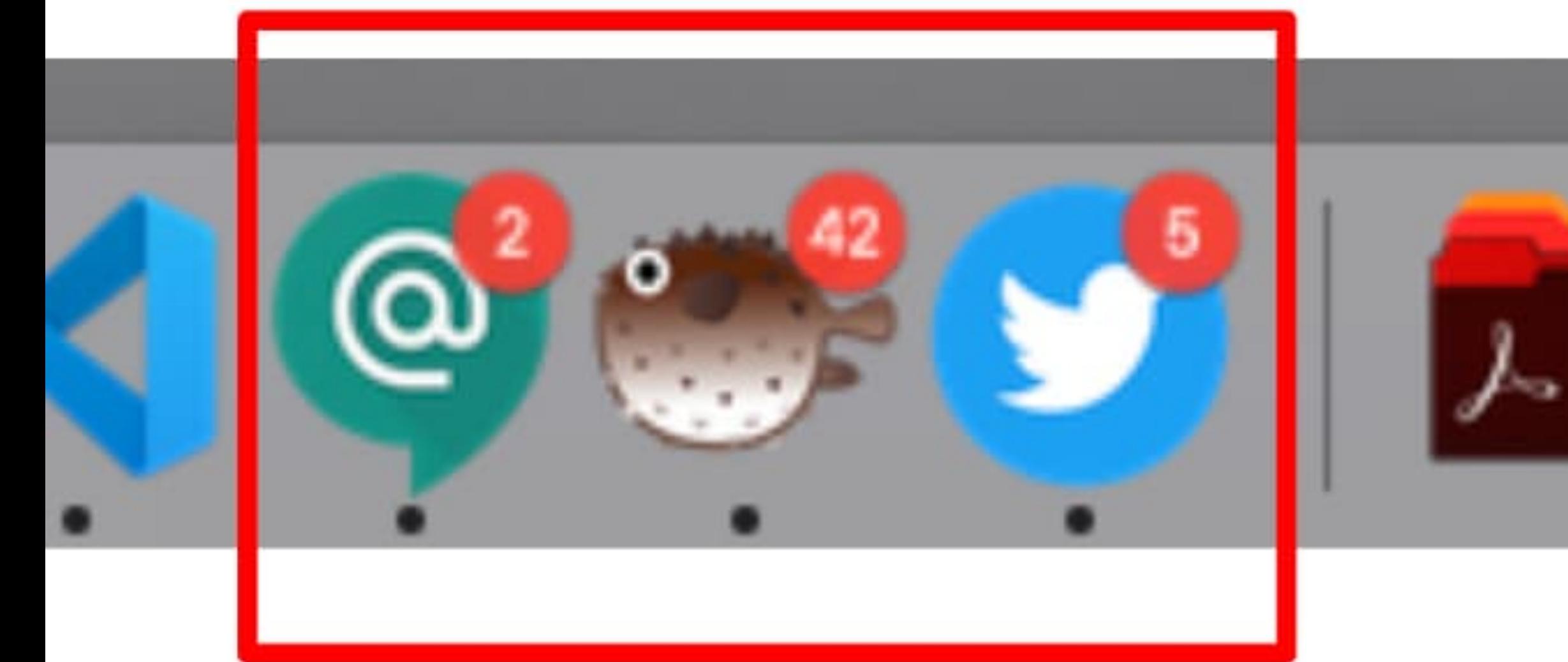
```
navigator.clearAppBadge();
```



# Set app badge

```
navigator.setAppBadge(badgeNumber);
```

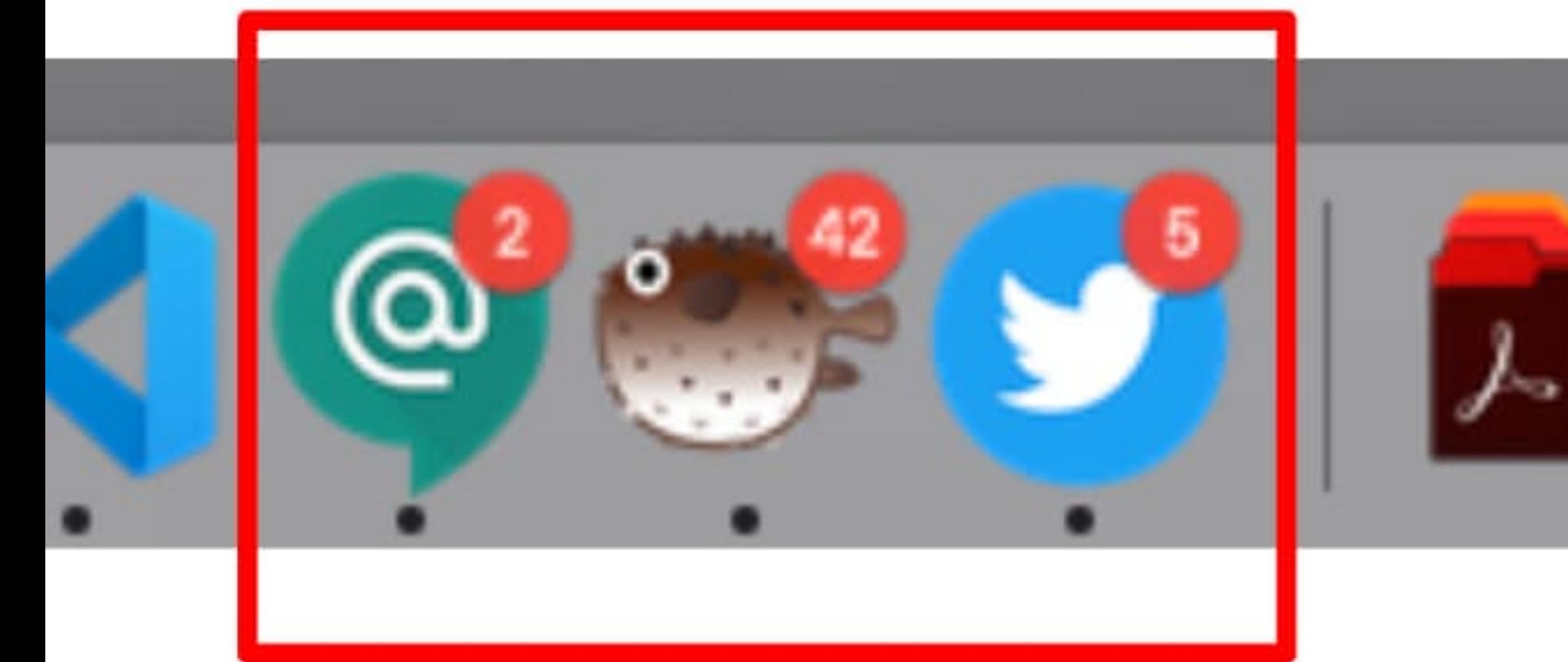
```
navigator.clearAppBadge();
```



# Set app badge

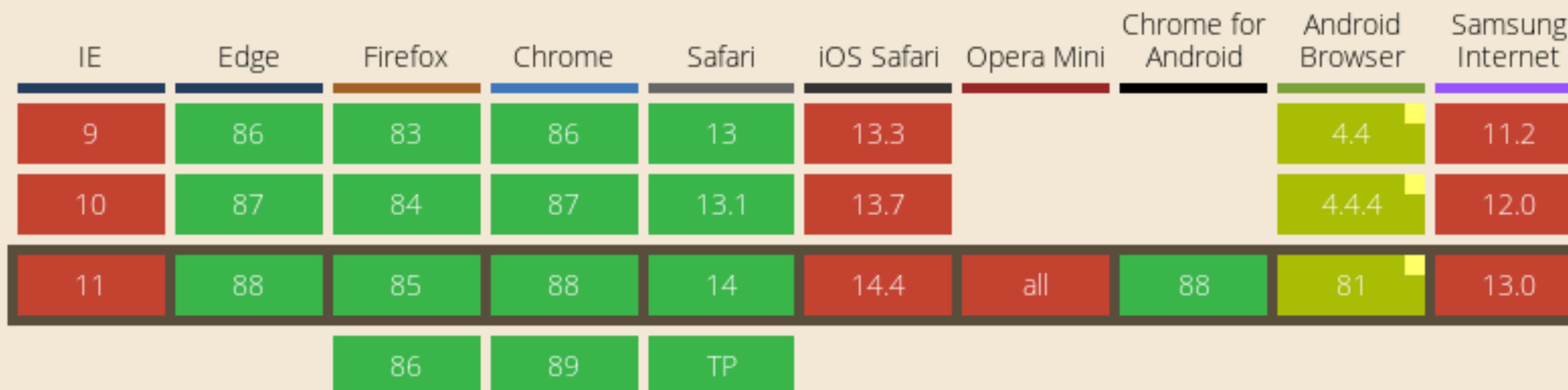
```
navigator.setAppBadge(badgeNumber);
```

```
navigator.clearAppBadge();
```



## Web Notifications

Method of alerting the user outside of a web page by displaying notifications (that do not require interaction by the user).

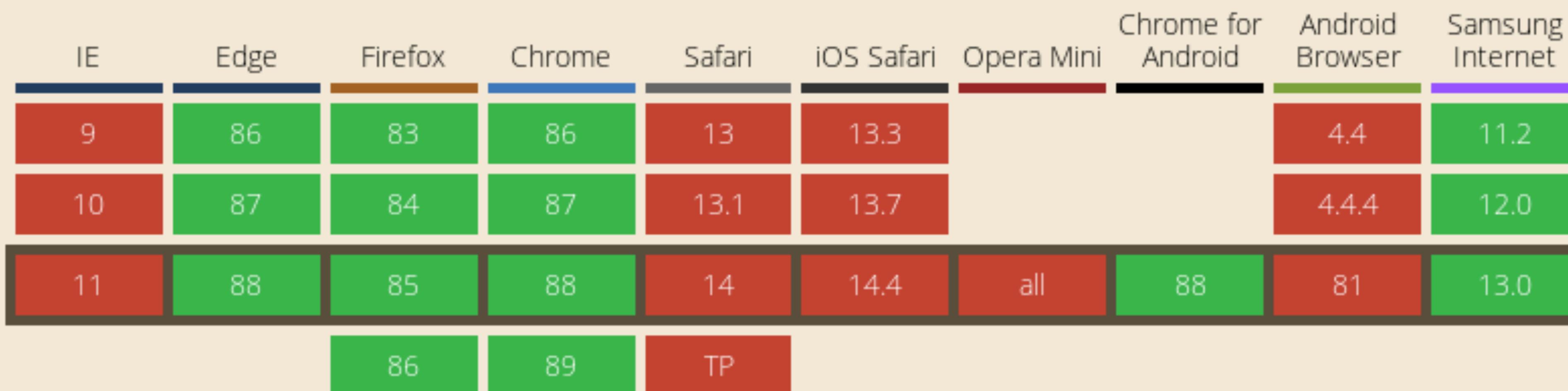


✓ Partial Support ✘ Prefixed

Global: 76.07% + 0.49% = 76.56%

## Push API

API to allow messages to be pushed from a server to a browser, even when the site isn't focused or even open in the browser.



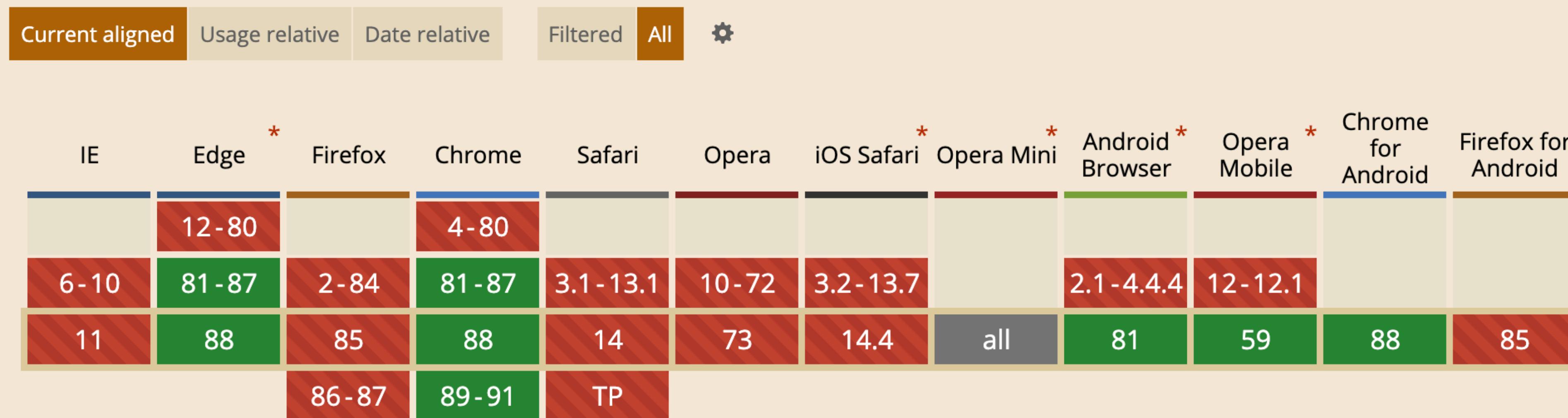
✓   ✗   Partial Support



Global: 76.57% + 0.26% = 76.83%



# Navigator API: setAppBadge



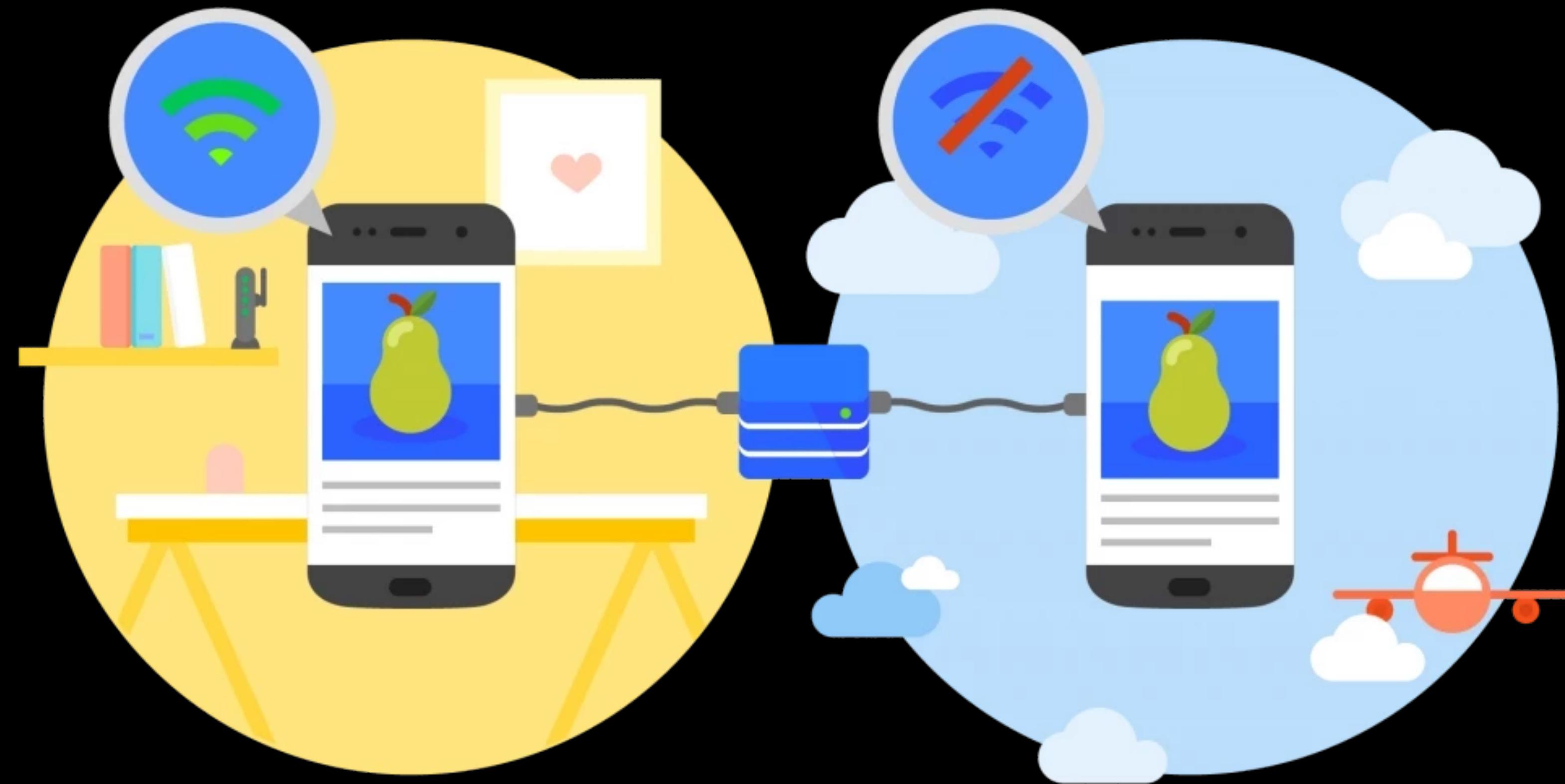


Image from <https://robert-askam.co.uk/posts/how-do-make-a-post-request-to-a-server-using-background-sync-and-service-workers>

# APIs

Background Sync API – defer actions until stable connection

Background Fetch API – perform fetches in the background

Periodic Background Sync API – sync data in the background



# Register a sync task

```
navigator.serviceWorker.ready.then(async (registration) => {  
  await registration.sync.register('send-chat-message');  
});
```

# Detect connectivity

```
self.addEventListener('sync', function(event) {  
});
```

# Complete task

```
self.addEventListener('sync', function(event) {  
  
  if (event.tag = 'send-chat-message') {  
  
    event.waitUntil( /* send chat message */ );  
  
  }  
});
```



# Start a background fetch

```
navigator.serviceWorker.ready.then(async (registration) => {  
  
  const videoFetch = await registration.backgroundFetch.fetch('video-fetch', ['/video.mp4'], {  
  
    title: 'Funny Video',  
  
    icons: [{ sizes: '300x300', src: '/thumbnail.png', type: 'image/png' }],  
  
    downloadTotal: 60 * 1024 * 1024,  
  
  });  
  
});
```

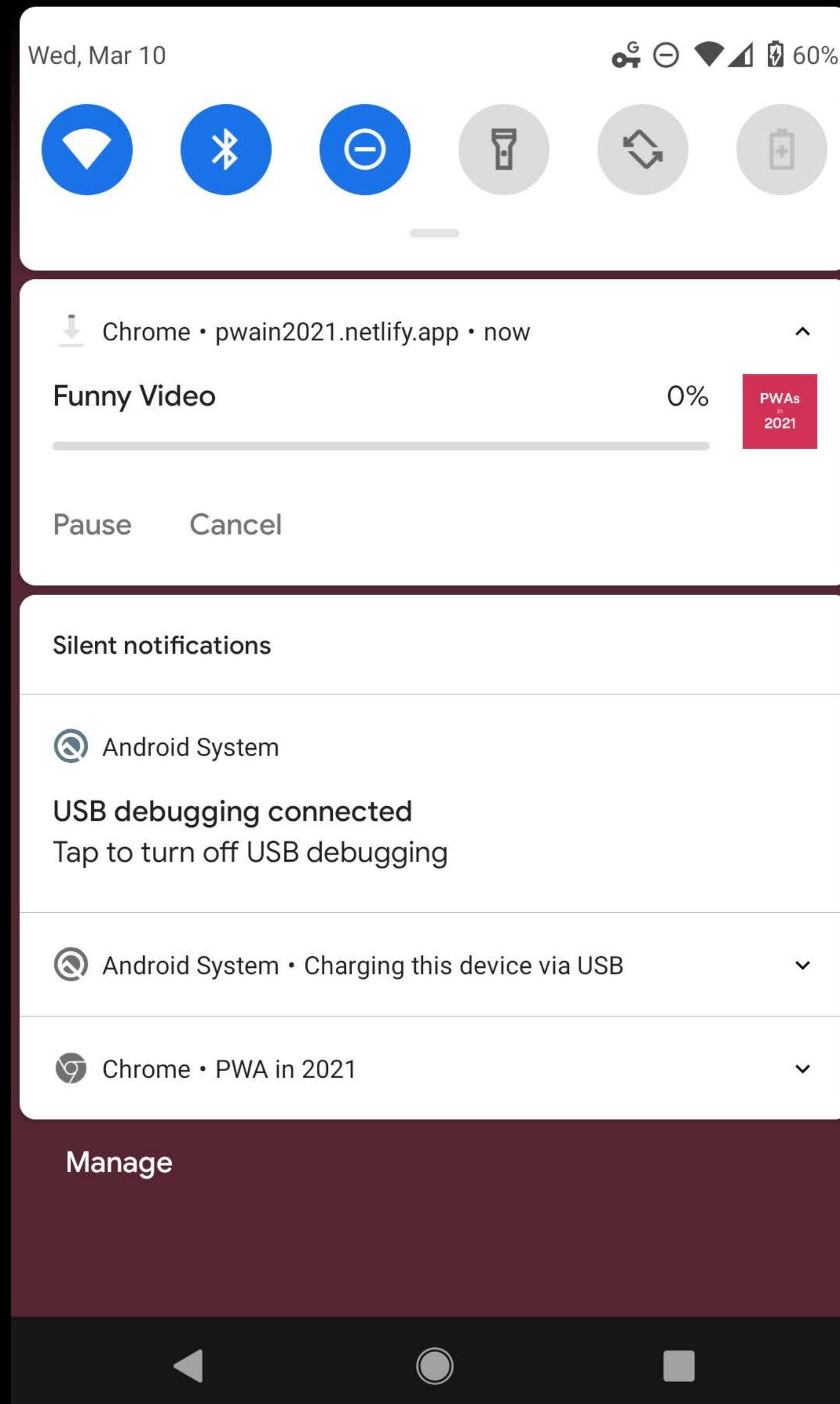
# Start a background fetch

```
navigator.serviceWorker.ready.then(async (registration) => {  
  
    const videoFetch = await registration.backgroundFetch.fetch('video-fetch', ['/video.mp4'], {  
  
        title: 'Funny Video',  
  
        icons: [{ sizes: '300x300', src: '/thumbnail.png', type: 'image/png' }],  
  
        downloadTotal: 60 * 1024 * 1024,  
  
    }) ;  
  
});
```

# Start a background fetch

```
navigator.serviceWorker.ready.then(async (registration) => {  
  
    const videoFetch = await registration.backgroundFetch.fetch('video-fetch', ['/video.mp4'], {  
  
        title: 'Funny Video',  
  
        icons: [{ sizes: '300x300', src: '/thumbnail.png', type: 'image/png' }],  
  
        downloadTotal: 60 * 1024 * 1024,  
  
    }) ;  
  
});
```

2:52



# Track progress

```
videoFetch.addEventListener('progress', () => {  
  const percent = Math.round(videoFetch.downloaded /  
videoFetch.downloadTotal * 100);  
  console.log(`Download progress: ${percent}%`);  
});
```

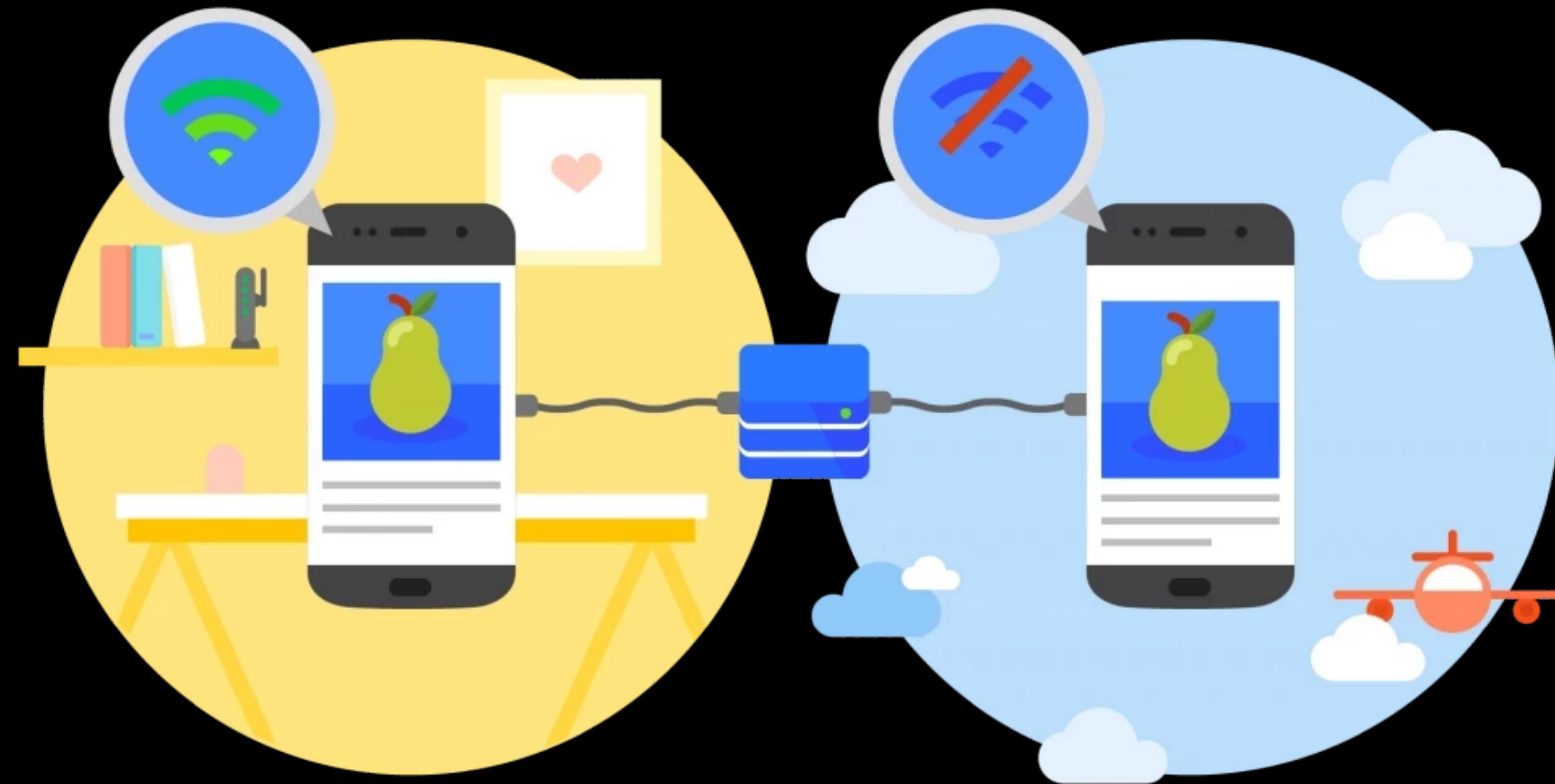
# Service worker events

```
self.addEventListener('backgroundfetchsuccess' () => /* */ );
```

```
self.addEventListener('backgroundfetchfailure' () => /* */ );
```

```
self.addEventListener('backgroundfetchabort' () => /* */ );
```

```
self.addEventListener('backgroundfetchclick' () => /* */ );
```



# Register a periodic sync task

```
navigator.serviceWorker.ready.then(async (registration) => {  
  await registration.periodicSync.register('chat-sync', {  
    minInterval: 24 * 60 * 60 * 1000, // Once a day  
  });  
});
```

# Register a periodic sync task

```
navigator.serviceWorker.ready.then(async (registration) => {  
  await registration.periodicSync.register('chat-sync', {  
    minInterval: 24 * 60 * 60 * 1000, // Once a day  
  });  
});
```

# Register a periodic sync task

```
navigator.serviceWorker.ready.then(async (registration) => {  
  await registration.periodicSync.register('chat-sync', {  
    minInterval: 24 * 60 * 60 * 1000, // Once a day  
  });  
});
```

# Detect periodic sync event

```
self.addEventListener('periodicsync', function(event) {  
});
```

# Complete task

```
self.addEventListener('periodicsync', function(event) {  
  
  if (event.tag = 'chat-sync') {  
  
    event.waitUntil( /* fetch and update chat messages */ );  
  
  }  
});
```

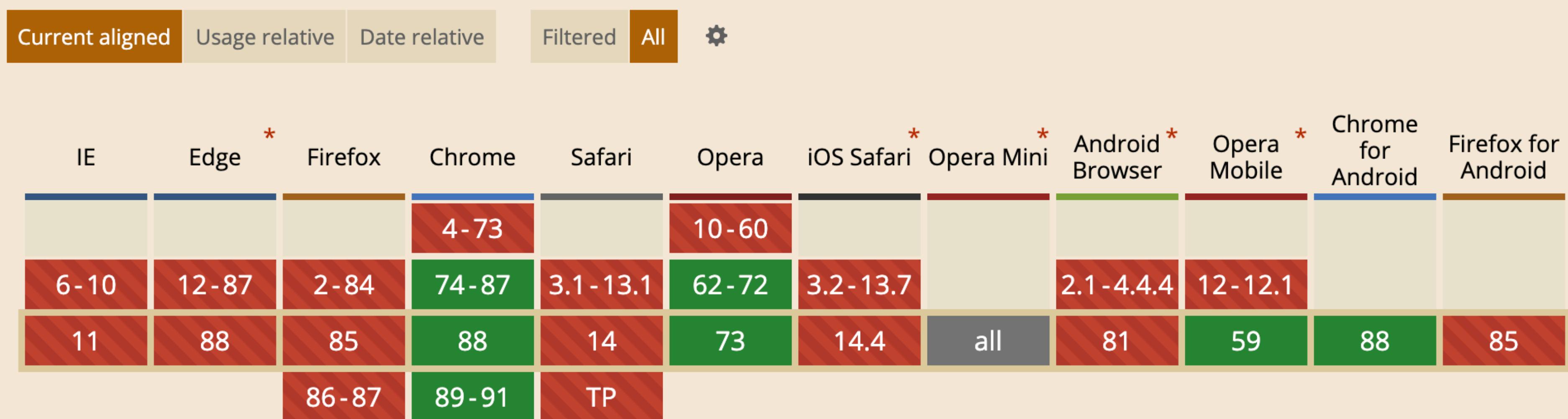
## Background Sync API

Provides one-off and periodic synchronization for Service Workers with an `onsync` event.

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	86	83	86	13	13.3			4.4	11.2
10	87	84	87	13.1	13.7			4.4.4	12.0
11	88	85	88	14	14.4	all	88	81	13.0
	86	89	TP						
✓	✗	?	Partial Support						

Global: 72.7% + 0% = 72.70%

# BackgroundFetchEvent API



# PeriodicSyncEvent API

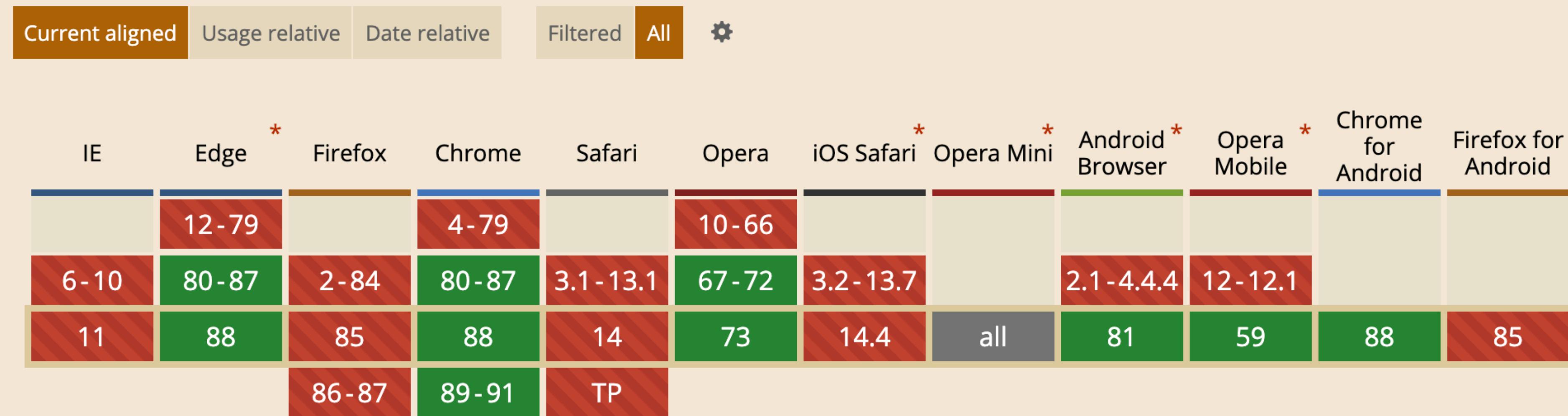




Image from <https://www.simicart.com/blog/pwa-hardware-access/>

# (Some) APIs

Contact Picker API

Web Share API & Web Share Target API

File System Access API

# Contact Picker API

```
if ('contacts' in navigator) {  
}  
}
```

← Select contacts Done

The contacts you select will be shared with [pwain2021.netlify.app](http://pwain2021.netlify.app).

Names  Email addresses  Phone numbers

	Ire Aderinokun [REDACTED]@gmail.com	
	[REDACTED]	

# Contact Picker API

```
if ('contacts' in navigator) {  
  
  const contacts = await navigator.contacts.select(  
    ['name', 'email', 'tel', 'icon'],  
    {multiple: true}  
  
  );  
  
}
```

## PWA in 2021

A demo of various web capabilities available in 2021.

- ▶ **Install**
- ▶ **Push Notifications**
- ▶ **App Badge**
- ▶ **Background Fetch**
- ▼ **Contact Picker**

Read contacts.

Get contacts

Name: Ire Aderinokun

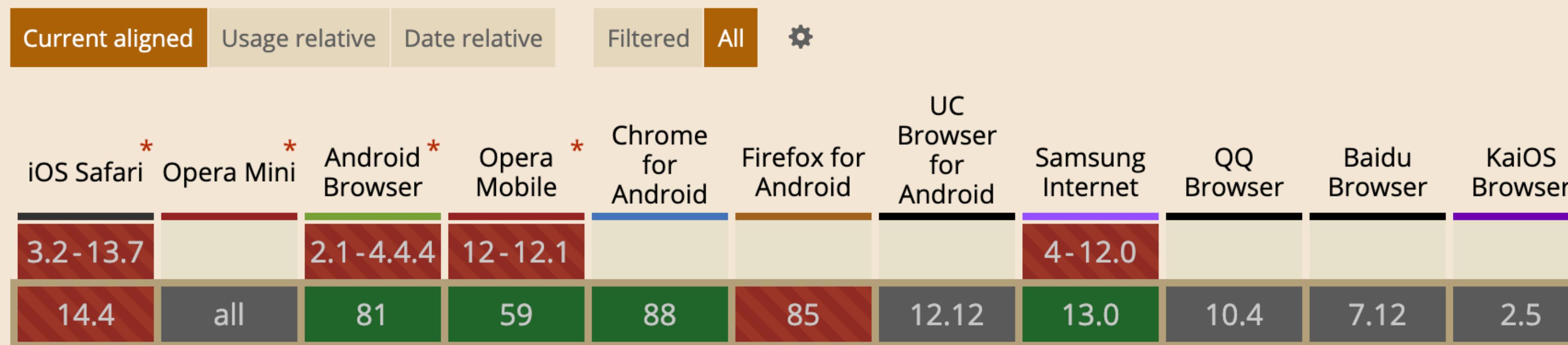
Email: [REDACTED]@gmail.com

Tel: undefined

- ▶ **Web Share**
- ▶ **File System Access**

# Navigator API: contacts

Usage % of all users ?  
Global 39.69%



# Web Share API

```
if (navigator.share) {  
}  
}
```

## PWA in 2021

A demo of various web capabilities available in 2021.

- ▶ [Install](#)
- ▶ [Push Notifications](#)
- ▶ [App Badge](#)
- ▶ [Background Fetch](#)
- ▶ [Contact Picker](#)

### ▼ Web Share

Share data between applications.

[Share a link](#)

- ▶ [File System Access](#)

[Source](#)

# Web Share API

```
if (navigator.share) {  
  
  navigator.share({  
  
    title: 'An Event Apart Spring Summit',  
  
    text: 'Check out An Event Apart\'s latest conference!',  
  
    url: 'https://aneventapart.com/event/spring-summit-2021',  
  
  });  
}
```

## PWA in 2021

A demo of various web capabilities available in 2021.

- ▶ [Install](#)
- ▶ [Push Notifications](#)
- ▶ [App Badge](#)
- ▶ [Background Fetch](#)
- ▶ [Contact Picker](#)
- ▼ [Web Share](#)

share data between applications.

[Share a link](#)

- ▶ [File System Access](#)

[Source](#)

# Web Share Target API

```
"share_target": {  
  
  "action": "/share-target.html",  
  
  "method": "GET",  
  
  "params": {  
  
    "title": "title",  
  
    "text": "text",  
  
    "url": "url"  
  
  }  
  
}
```



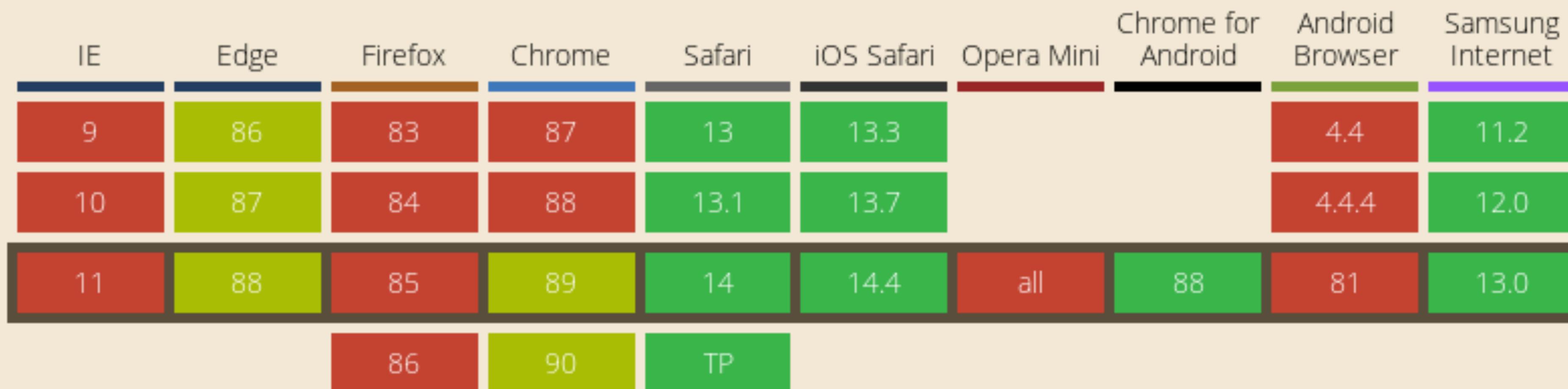
# Web Share Target API

```
window.addEventListener('DOMContentLoaded', () => {  
  const { searchParams } = new URL(window.location);  
  
  const title = searchParams.get('title');  
  
  const text = searchParams.get('text');  
  
  const url = searchParams.get('url');  
});
```



## Web Share API

A way to allow websites to invoke the native sharing capabilities of the host platform



Partial Support

Global: 57.58% + 3.47% = 61.05%

## Web Share Target



# File System Access API

```
if ('showOpenFilePicker' in window) {  
}  
}
```

# Read a local file

```
let fileHandle;

document.getElementById('open-file-picker').addEventListener('click', async () => {

  [fileHandle] = await window.showOpenFilePicker();

  const file = await fileHandle.getFile();

  const contents = await file.text();

});
```

# Read a local file

```
let fileHandle;

document.getElementById('open-file-picker').addEventListener('click', async () => {

  [fileHandle] = await window.showOpenFilePicker();

  const file = await fileHandle.getFile();

  const contents = await file.text();

});
```

# Read a local file

```
let fileHandle;

document.getElementById('open-file-picker').addEventListener('click', async () => {

  [fileHandle] = await window.showOpenFilePicker();

  const file = await fileHandle.getFile();

  const contents = await file.text();

});
```

# Read a local file

```
let fileHandle;

document.getElementById('open-file-picker').addEventListener('click', async () => {

  [fileHandle] = await window.showOpenFilePicker();

  const file = await fileHandle.getFile();

  const contents = await file.text();

});
```

# Save to a local file

```
document.getElementById('save-file').addEventListener('click', async () => {  
  
  const writable = await fileHandle.createWritable();  
  
  await writable.write(/* new contents */);  
  
  await writable.close();  
  
});
```

# Save to a local file

```
document.getElementById('save-file').addEventListener('click', async () => {  
  const writable = await fileHandle.createWritable();  
  
  await writable.write(/* new contents */);  
  
  await writable.close();  
});
```

# Save to a local file

```
document.getElementById('save-file').addEventListener('click', async () => {  
  
  const writable = await fileHandle.createWritable();  
  
  await writable.write(/* new contents */);  
  
  await writable.close();  
  
});
```

# Save to a local file

```
document.getElementById('save-file').addEventListener('click', async () => {  
  const writable = await fileHandle.createWritable();  
  
  await writable.write(/* new contents */);  
  
  await writable.close();  
});
```

## PWA in 2021

A demo of various web capabilities available in 2021.

- 
- ▶ [Install](#)
  - ▶ [Push Notifications](#)
  - ▶ [App Badge](#)
  - ▶ [Background Fetch](#)
  - ▶ [Contact Picker](#)
  - ▶ [Web Share](#)
  - ▶ [File System Access](#)
- 

[Source](#)

## File System Access API

API for manipulating files in the device's local file system (not in a sandbox).

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	86	83	86	13	13.3			4.4	11.2
10	87	84	87	13.1	13.7			4.4.4	12.0
11	88	85	88	14	14.4	all	88	81	13.0
			86	89	TP				



Partial Support

Search or enter website name

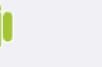
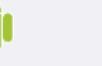
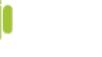
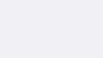
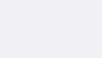
# Fugu API Tracker

STABLE      BETA      DEV

 Chrome 89     Chrome 90     Chrome 91

In 1 days  
(Mar 2, 2021)      In 43 days  
(Apr 13, 2021)      In 85 days  
(May 25, 2021)

## Shipped

Web Share Target	 M71		
Web Share API Level 2	 M75		
Async Clipboard: Read and Write Images	 M76	    	
Web Share Target Level 2	 M76		
Enter Key Hint	 M77		
Expand Storage Quota	 M78	    	
Periodic Background Sync	 M80	    	

Capable



Reliable

Installable

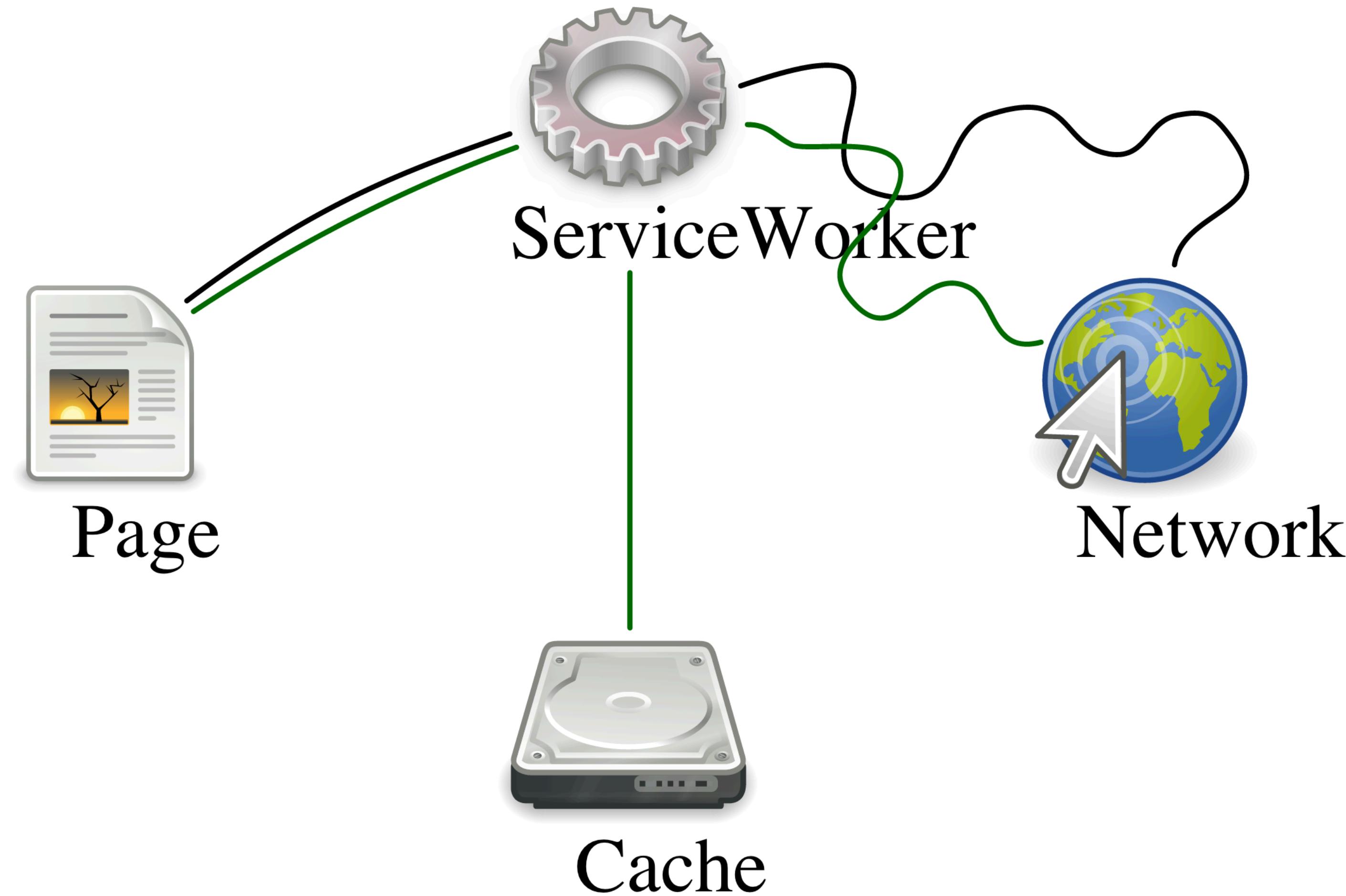
Capable



Reliable

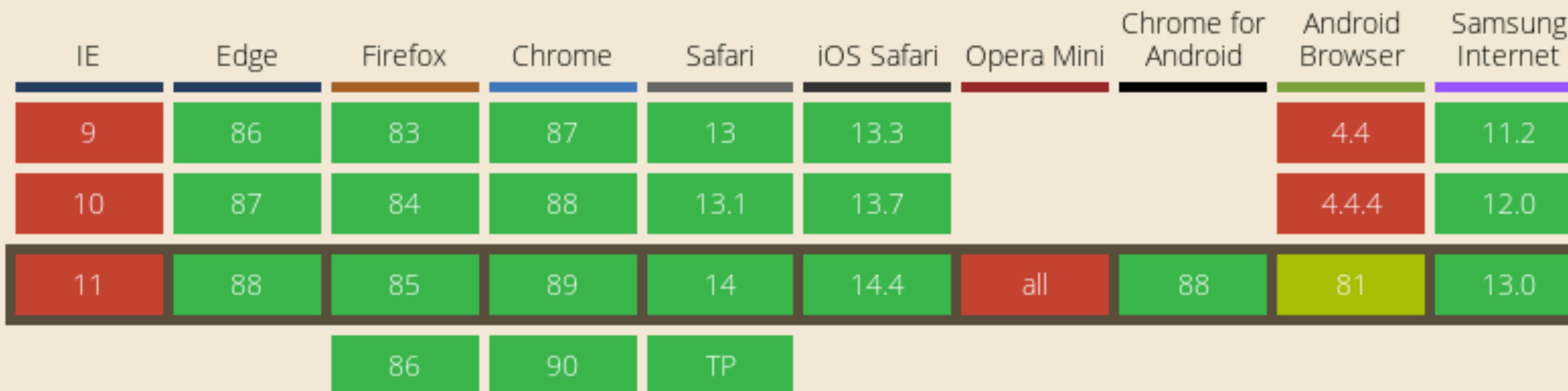


Installable



## Service Workers

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.



Partial Support

Global: 94.87% + 0.09% = 94.96%

▶ **Install**

▶ **Push Notifications**

▶ **App Badge**

Elements    Console    Sources    Network    Performance    Memory    **Application**    Security    Lighthouse    Adblock Plus    ✖ 1    ⚙    ⋮    ×

**Application**

- Manifest
- Service Workers**
- Storage

**Storage**

- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies

**Cache**

- Cache Storage
- Application Cache

**Background Services**

- Background Fetch
- Background Sync
- Notifications
- Payment Handler
- Periodic Background Sync
- Push Messaging

**Service Workers**

Offline     Update on reload     Bypass for network

**https://pwain2021.netlify.app/**

Source [sw.js](#)  
Received 19/03/2021, 10:19:41

Status ● #552 activated and is running [stop](#)

Clients <https://pwain2021.netlify.app/> [focus](#)

Push Push

Sync Sync

Periodic Sync Periodic Sync

**Service workers from other origins**

[See all registrations](#)

✖ ✖ ✖ ✖ ✖ ✖



**Workbox**

A screenshot of a web browser displaying the Workbox website. The browser interface includes a top bar with navigation icons, a search bar, and a user profile icon. The main header features the Workbox logo and navigation links. A prominent orange banner at the top of the page announces the end of the Chrome Dev Summit 2020 and directs users to watch sessions at [goo.gle/cds20-sessions](https://goo.gle/cds20-sessions). Below the banner, the page shows a breadcrumb trail (Home > Products > Workbox > Tools > Guides) and a "Rate and review" button with thumbs up and down icons. The main content area is titled "Common Recipes" and includes a sidebar with a "Table of contents" dropdown menu containing links to various caching recipes.

≡  Workbox

Search or enter website name

English 

Workbox

It's a wrap for **Chrome Dev Summit 2020!** Watch all the sessions at [goo.gle/cds20-sessions](https://goo.gle/cds20-sessions) now!

Home > Products > Workbox > Tools > Guides

Rate and review  

# Common Recipes

Table of contents ▾

- [Google Fonts](#)
- [Caching Images](#)
- [Cache CSS and JavaScript Files](#)
- [Caching Content from Multiple Origins](#)
- [Restrict Caches for a Specific Origin](#)
- ...

# Cache JS & CSS

```
import {registerRoute} from 'workbox-routing';

import {StaleWhileRevalidate} from 'workbox-strategies';

registerRoute(
  ({request}) => request.destination === 'script' ||
    request.destination === 'style',
  new StaleWhileRevalidate()
);
```

# Cache JS & CSS

```
import {registerRoute} from 'workbox-routing';
import {StaleWhileRevalidate} from 'workbox-strategies';

registerRoute(
  ({request}) => request.destination === 'script' ||
    request.destination === 'style',
  new StaleWhileRevalidate()
);
```

# Cache JS & CSS

```
import {registerRoute} from 'workbox-routing';

import {StaleWhileRevalidate} from 'workbox-strategies';

registerRoute(
  ({request}) => request.destination === 'script' ||
    request.destination === 'style',
  new StaleWhileRevalidate()
);
```

# Cache JS & CSS

```
import {registerRoute} from 'workbox-routing';

import {StaleWhileRevalidate} from 'workbox-strategies';

registerRoute(
  ({request}) => request.destination === 'script' ||
    request.destination === 'style',
  new StaleWhileRevalidate()
);
```

# Offline Google Analytics

```
import * as googleAnalytics from 'workbox-google-analytics';

googleAnalytics.initialize();
```

# Offline Google Analytics

```
import * as googleAnalytics from 'workbox-google-analytics';

googleAnalytics.initialize();
```

# Offline Google Analytics

```
import * as googleAnalytics from 'workbox-google-analytics';

googleAnalytics.initialize();
```

# Cache offline page

```
const CACHE_NAME = 'offline-html';

const FALLBACK_HTML_URL = '/offline.html';

self.addEventListener('install', async (event) => {

  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => cache.add(FALLBACK_HTML_URL))
  );
});
```

# Cache offline page

```
const CACHE_NAME = 'offline-html';

const FALLBACK_HTML_URL = '/offline.html';

self.addEventListener('install', async (event) => {

  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => cache.add(FALLBACK_HTML_URL))
  );
});
```

# Cache offline page

```
const CACHE_NAME = 'offline-html';

const FALLBACK_HTML_URL = '/offline.html';

self.addEventListener('install', async (event) => {

  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => cache.add(FALLBACK_HTML_URL))
  );
});
```

# Cache offline page

```
const CACHE_NAME = 'offline-html';

const FALLBACK_HTML_URL = '/offline.html';

self.addEventListener('install', async (event) => {

  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => cache.add(FALLBACK_HTML_URL))
  );
});
```

# Serve offline page

```
const networkOnly = new NetworkOnly();

const navigationHandler = async (params) => {

  try {

    return await networkOnly.handle(params);

  } catch (error) {

    return caches.match(FALLBACK_HTML_URL, { cacheName: CACHE_NAME });

  }

};

registerRoute( new NavigationRoute(navigationHandler) );
```

# Serve offline page

```
const networkOnly = new NetworkOnly();

const navigationHandler = async (params) => {

  try {

    return await networkOnly.handle(params);

  } catch (error) {

    return caches.match(FALLBACK_HTML_URL, { cacheName: CACHE_NAME });

  }

};

registerRoute( new NavigationRoute(navigationHandler) );
```

# Serve offline page

```
const networkOnly = new NetworkOnly();

const navigationHandler = async (params) => {

  try {

    return await networkOnly.handle(params);

  } catch (error) {

    return caches.match(FALLBACK_HTML_URL, { cacheName: CACHE_NAME });

  }

};

registerRoute( new NavigationRoute(navigationHandler) );
```

# Serve offline page

```
const networkOnly = new NetworkOnly();

const navigationHandler = async (params) => {

  try {

    return await networkOnly.handle(params);

  } catch (error) {

    return caches.match(FALLBACK_HTML_URL, { cacheName: CACHE_NAME });

  }

};

registerRoute( new NavigationRoute(navigationHandler) );
```

# Serve offline page

```
const networkOnly = new NetworkOnly();

const navigationHandler = async (params) => {

  try {

    return await networkOnly.handle(params);

  } catch (error) {

    return caches.match(FALLBACK_HTML_URL, { cacheName: CACHE_NAME });

  }

};

registerRoute( new NavigationRoute(navigationHandler) );
```

# PWA in 2021

A demo of various web capabilities available in 2021.

▶ [Install](#)

▶ [Push Notifications](#)

▶ [App Badge](#)

▶ [Background Fetch](#)

▶ [Contact Picker](#)

▶ [Web Share](#)

▶ [File System Access](#)

[Source](#)

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, there's a sidebar with sections for Application (Manifest, Service Workers, Storage), Storage (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), and Cache (Cache Storage). The Service Workers section is currently active, showing the status of a service worker at `http://localhost:6060/`. The service worker is identified as `sw.js` and has been received on 18/03/2021, 14:44:52. Its status is green, indicating it is activated and running. A red error icon with the number 3 is present. Below this, there are fields for testing push notifications and sync operations. A 'Push' button is associated with the message 'Test push message from DevTools.', and a 'Sync' button is associated with the message 'test-tag-from-devtools'.

## Latest Articles



Latest



Home



Bookmarks

## Latest Articles

### Variable and Function Hoisting in ES2015

⌚ Last Tuesday 📷 javascript

Variable hoisting is a behaviour in JavaScript where variable declarations are moved to the top of the scope (function scope or global scope) that the variable is defined within. The typical JavaScript variable can be created in two stages - declaration and initialisation.

The declaration stage is where the variable....

— Read

■ Bookmark

### The New System Font Stack?

⌚ Sep 20th, 2016 📷 css

A few months ago, I wrote about how you can use system fonts in the browser using the built-in keywords that work with the font shorthand property ([See Using System Fonts in the Browser](#)). These



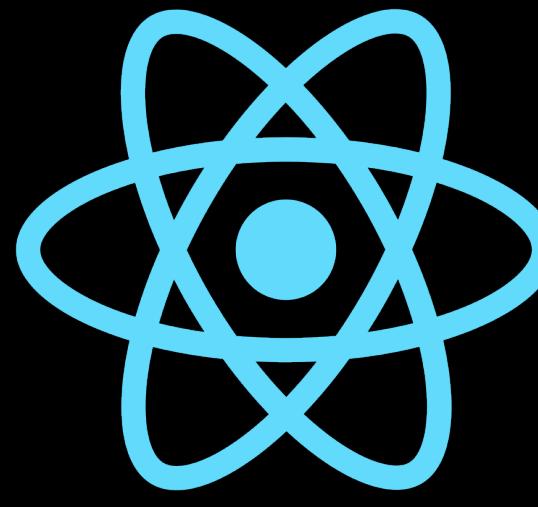
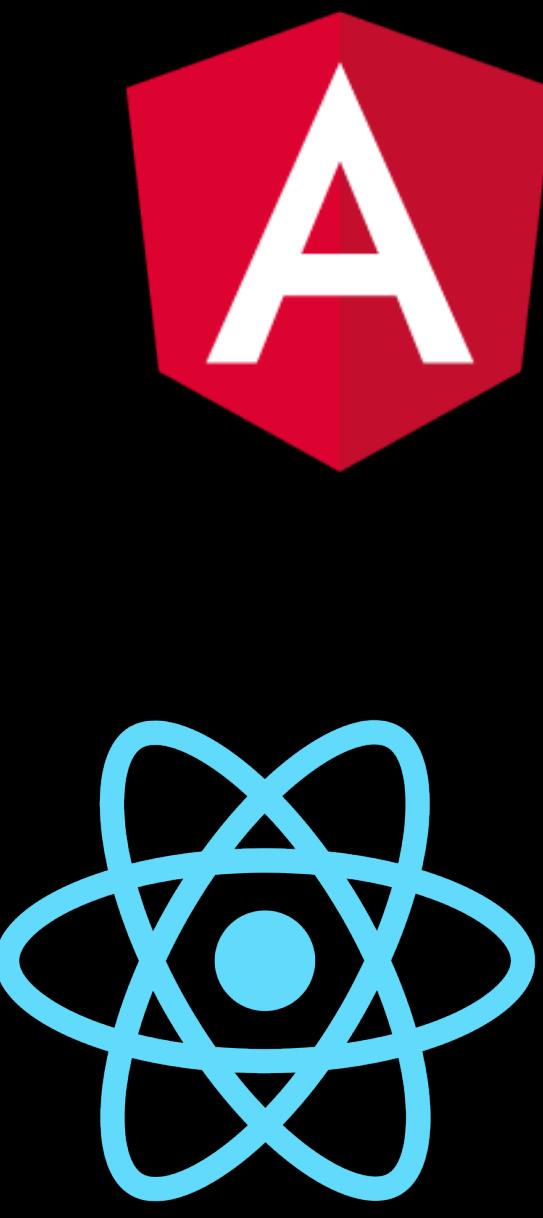
Latest



Home



Bookmarks



The screenshot shows a web browser window with the Angular website open. The title bar includes standard icons for back, forward, search, and tabs. The Angular logo is in the top left, and a search bar with a Twitter and GitHub icon is in the top right. The main content area has a blue header with the title "App shell". Below the header, there is a large heading "App shell" and a detailed explanation of what it is and how it improves user experience. A sidebar on the right lists three steps: "Step 1: Prepare the application", "Step 2: Create the app shell", and "Step 3: Verify the app is built with the shell content".

# App shell

App shell is a way to render a portion of your application via a route at build time. It can improve the user experience by quickly launching a static rendered page (a skeleton common to all pages) while the browser downloads the full client version and switches to it automatically after the code loads.

This gives users a meaningful first paint of your application that appears quickly because the browser can simply render the HTML and CSS without the need to initialize any JavaScript.

Learn more in [The App Shell Model](#).

## Step 1: Prepare the application

You can do this with the following CLI command:

<https://angular.io/guide/app-shell>

The screenshot shows a web browser window with the following details:

- Address Bar:** Search or enter website name
- Title Bar:** Create React App
- Search Bar:** Search (with keyboard shortcut ⌘K)
- Content Area:**
  - # Making a Progressive Web App
  - The production build has all the tools necessary to generate a first-class [Progressive Web App](#), but **the offline/cache-first behavior is opt-in only.**
  - Starting with Create React App 4, you can add a `src/service-worker.js` file to your project to use the built-in support for [Workbox's InjectManifest plugin](#), which will [compile](#) your service worker and inject into it a list of URLs to [precache](#).
  - If you start a new project using one of the PWA [custom templates](#), you'll get a `src/service-worker.js` file that serves as a good starting point for an offline-first service worker:

```
npx create-react-app my-app --template cra-template-pwa
```
  - The TypeScript equivalent is:

```
npx create-react-app my-app --template cra-template-pwa-typescript
```
  - If you know that you won't be using service workers, or if you'd prefer to use a different approach to creating your service worker, don't

The screenshot shows a web browser displaying the Vue CLI documentation. The URL in the address bar is <https://cli.vuejs.org/core-plugins/pwa.html>. The page title is "@vue/cli-plugin-pwa". The main content area contains the following text:

# @vue/cli-plugin-pwa

pwa plugin for vue-cli

The service worker added with this plugin is only enabled in the production environment (e.g. only if you run `npm run build` or `yarn build`). Enabling service worker in a development mode is not a recommended practice, because it can lead to the situation when previously cached assets are used and the latest local changes are not included.

Instead, in the development mode the `noopServiceWorker.js` is included. This service worker file is effectively a 'no-op' that will reset any previous service worker registered for the same host:port combination.

If you need to test a service worker locally, build the application and run a simple HTTP-server from your build directory. It's recommended to use a browser incognito window to avoid complications with your browser cache.

On the left sidebar, under "Core Vue CLI Plugins", the following items are listed:

- @vue/cli-plugin-babel
- @vue/cli-plugin-typescript
- @vue/cli-plugin-eslint
- @vue/cli-plugin-pwa**
- Configuration
- Example Configuration
- Installing in an Already Created Project
- Injected webpack-chain Rules
- @vue/cli-plugin-unit-jest
- @vue/cli-plugin-unit-mocha

A sidebar advertisement for "Authentic" is visible on the left side of the page.

Capable



Reliable



Installable

Capable



Reliable

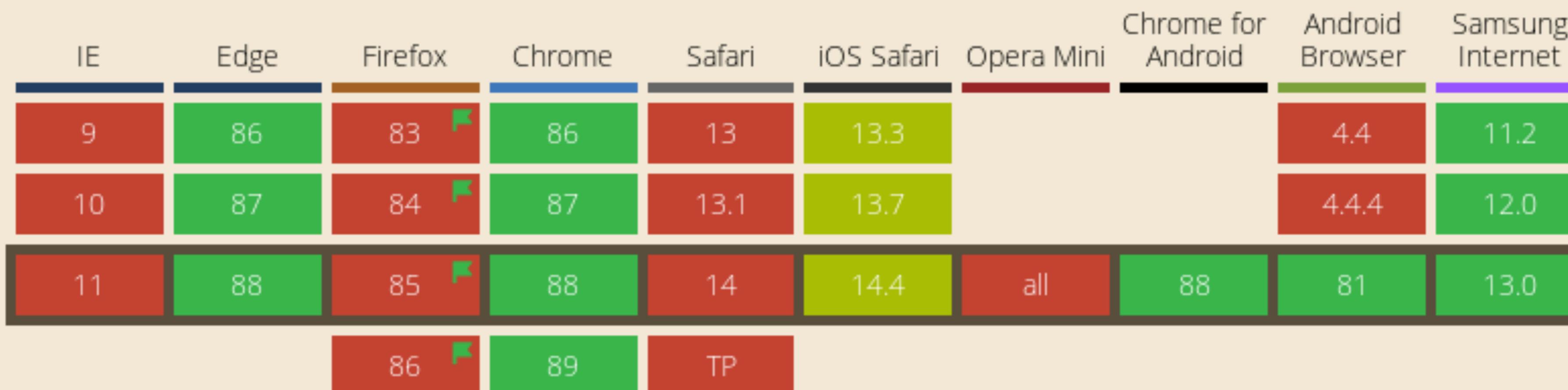


Installable



## Add to home screen (A2HS)

The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load its assets through a [Service Worker] (<https://caniuse.com/#feat=serviceworkers>).



Partial Support



Behind a Flag

Global: 72.91% + 14% = 86.91%



5:27

pwain2021.netlify.app

## PWA in 2021

A demo of various web capabilities available in 2021.

Install app

2021 PWA in 2021  
pwain2021.netlify.app

Cancel Install

▶ Push Notifications  
▶ App Badge  
▶ Background Fetch  
▶ Contact Picker  
▶ Web Share  
▶ File System Access

17:37

Add to Home Screen

PWA in 2021  
https://pwain2021.netlify.app/

An icon will be added to your Home screen so you can quickly access this website.

Cancel Add

"2021"

q w e r t y u i o p  
a s d f g h j k l  
z x c v b n m  
123 space done  
smiley face microphone

5:29

## PWA in 2021

A demo of various web capabilities available in 2021.

▼ Install

Add to Home screen

2021 PWA in 2021  
pwain2021.netlify.app

Cancel Add

▶ App Badge  
▶ Background Fetch  
▶ Contact Picker  
▶ Web Share  
▶ File System Access

Source

# Requirements for “Add to Home Screen”

# Requirements for “Add to Home Screen”

## 1. Web app manifest

```
{  
  "name": "PWA in 2021",  
  
  "short_name": "PWA in 2021",  
  
  "description": "A demo of what we can do with PWAs in  
  2021",  
  
  "scope": "/",  
  
  "display": "standalone",  
  
  "background_color": "#fffff",  
  
  "theme_color": "#D33257",  
  
  "start_url": "/"  
}
```

The screenshot shows a web browser window with the title "Web App Manifest Generator". The page content includes a note about Web App Manifests, configuration fields for Name and Short Name, and generated manifest files in JSON and HTML formats.

**Name**

Web App Manifest Generator

Name of your web app

**Short Name**

App Manifest

This will be used when there is insufficient space to display the full name, such as the homescreen

More...

**Start URL**

/

Your homescreen shortcut will load this URL

**manifest.json**

```
{  
  "lang": "en-GB",  
  "start_url": "/",  
  "display": "standalone"  
}
```

**<head>**

```
<link rel="manifest" href="manifest.j  
<meta name="mobile-web-app-capable" c  
<meta name="apple-mobile-web-app-capa  
<meta name="msapplication-starturl" c  
<meta name="viewport" content="width=
```

**Helpful notes**

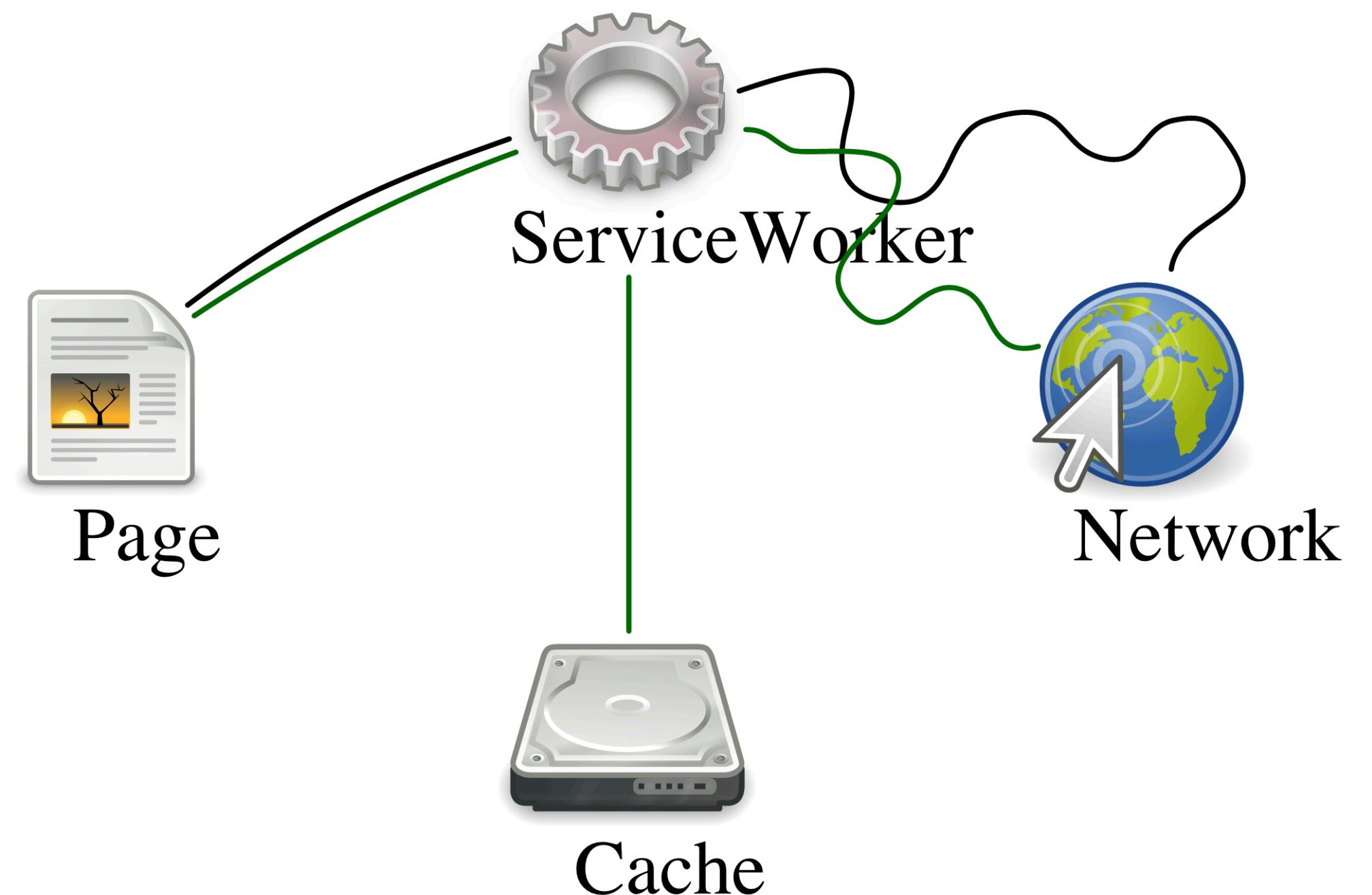
# Requirements for “Add to Home Screen”

1. Web app manifest
2. HTTPS



# Requirements for “Add to Home Screen”

1. Web app manifest
2. HTTPS
3. Service worker (which returns a 200 response when offline)



# Requirements for “Add to Home Screen”

1. Web app manifest
2. HTTPS
3. Service worker (which returns a response when offline) \*
4. User engagement heuristic

More control over “Install” flow

# Detect if PWA can be installed

```
window.addEventListener('beforeinstallprompt', (e) => {  
});
```

# Save install prompt event

```
let installPrompt;  
  
window.addEventListener('beforeinstallprompt', (e) => {  
  e.preventDefault();  
  
  installPrompt = e;  
  
});
```

# Save install prompt event

```
let installPrompt;  
  
window.addEventListener('beforeinstallprompt', (e) => {  
  e.preventDefault();  
  
  installPrompt = e;  
  
});
```

# Show a custom install prompt

```
let installPrompt;  
  
window.addEventListener('beforeinstallprompt', (e) => {  
  e.preventDefault();  
  
  installPrompt = e;  
  
  showCustomInstallPrompt();  
});
```

# Trigger install prompt

```
customInstallButton.addEventListener('click', async (e) => {  
  installPrompt.prompt();  
  
  const { outcome } = await installPrompt.userChoice;  
});
```

# Trigger install prompt

```
customInstallButton.addEventListener('click', async (e) => {  
  installPrompt.prompt();  
  
  const { outcome } = await installPrompt.userChoice;  
});
```

# Cleanup

```
window.addEventListener('appinstalled', () => {  
});
```

5:44

wain2021.netlify.app

6 :

## PWA in 2021

A demo of various web capabilities available in 2021.

---

▼ **Install**

Install this website as a mobile/desktop application.

[Install](#)

► **Push Notifications**

► **App Badge**

► **Background Fetch**

► **Contact Picker**

► **Web Share**

► **File System Access**

Add to Homescreen == Install

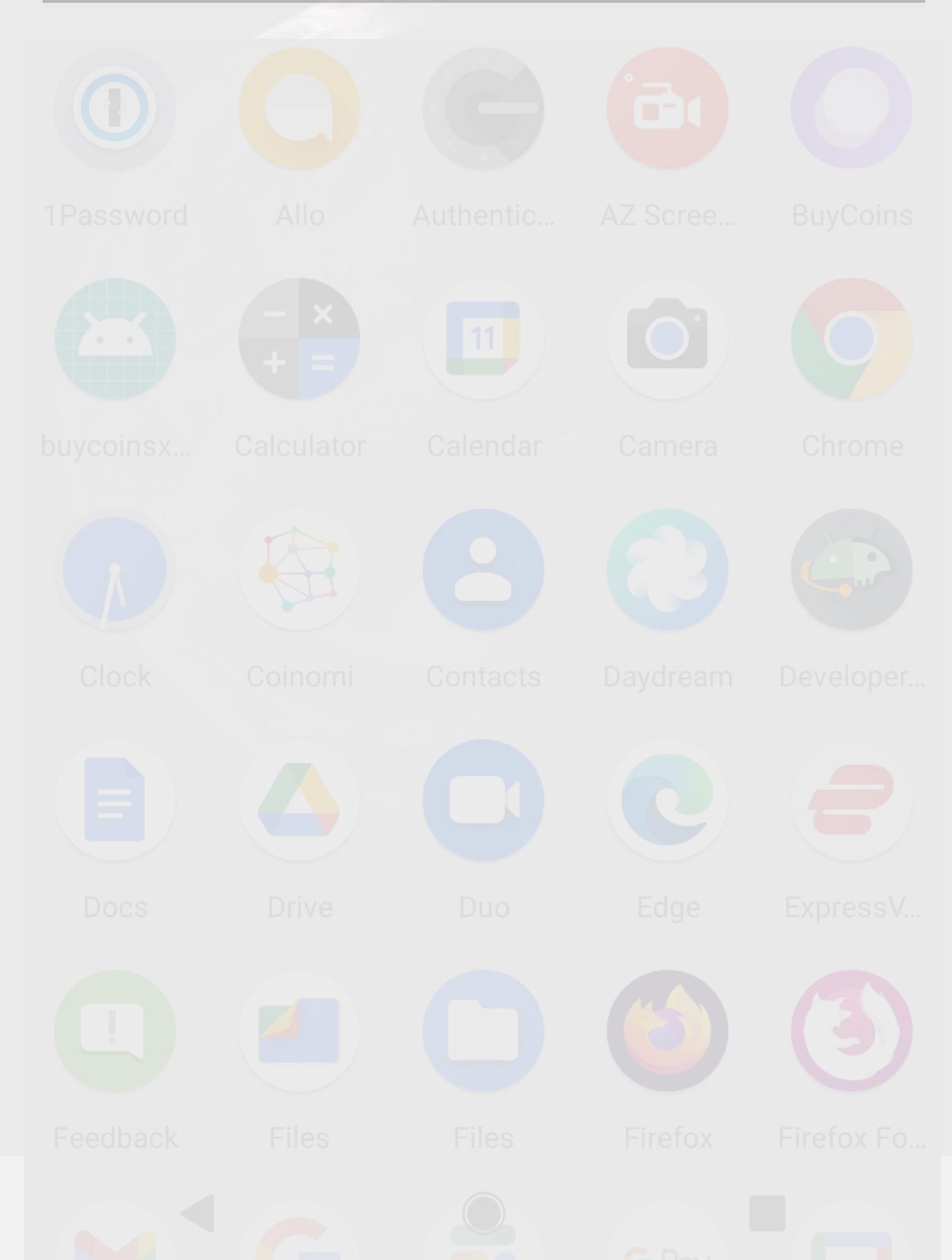
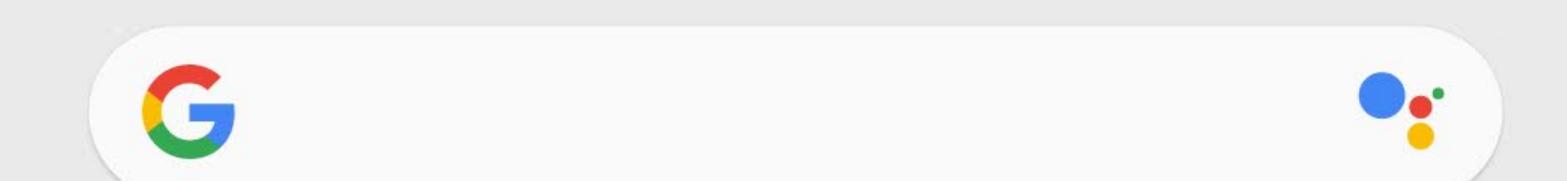
# WebAPK

A WebAPK is an Android Application Package (APK) **automatically generated** from a PWA and installed to the device.

# Benefits of WebAPK

# Benefits of WebAPK

✓ In the app drawer



# Benefits of WebAPK

 In the app drawer

 In app settings

5:33

G ⊖

App info



2021

PWA in 2021



Open



Uninstall



Force stop

Notifications

~1 notification per week

Permissions

No permissions requested

Storage & cache

455 kB used in internal storage

Mobile data & Wi-Fi

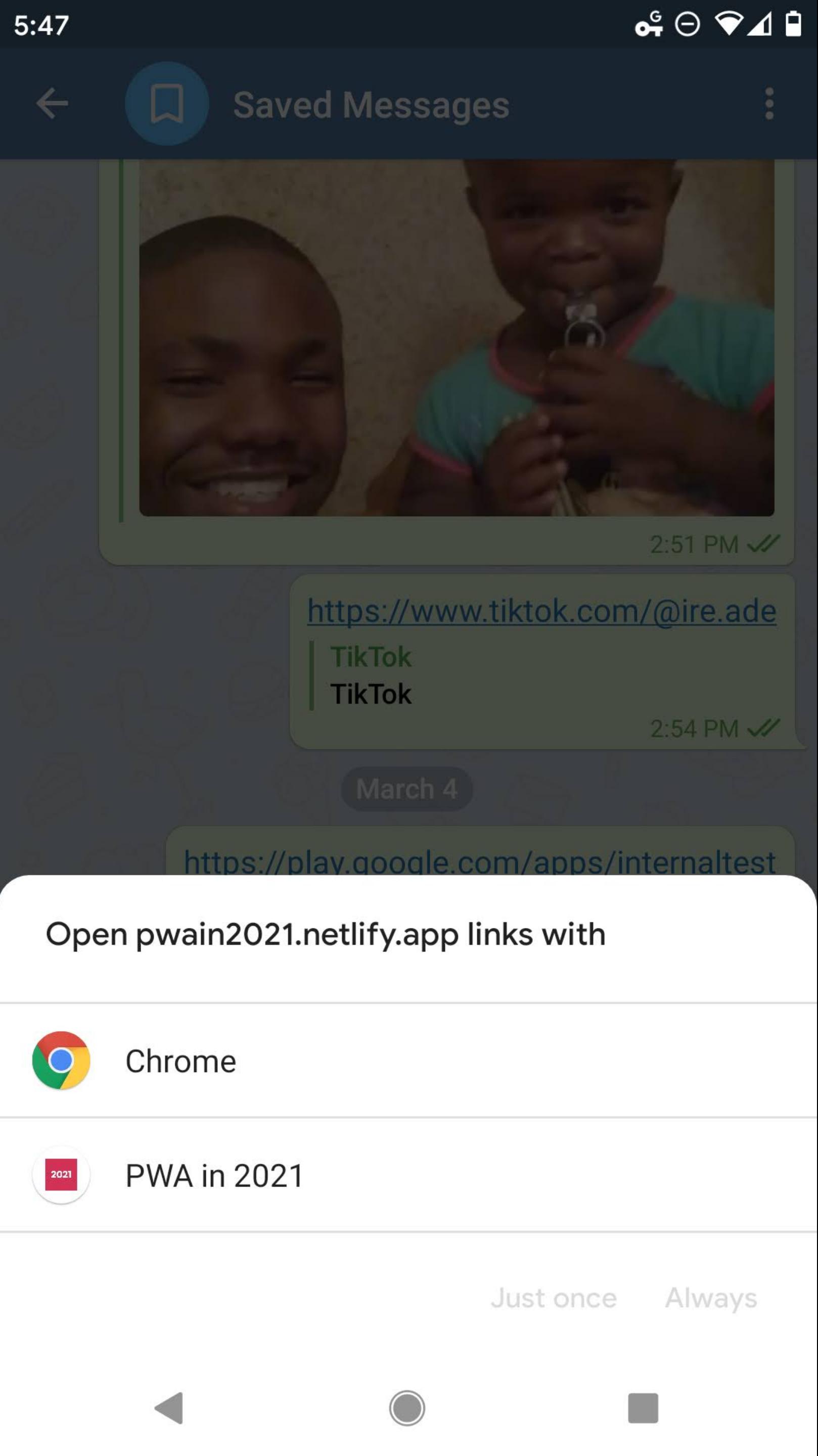
No data used

Advanced

Screen time, Battery, Open by default, Store

# Benefits of WebAPK

- In the app drawer
- In app settings
- Includes intent filters



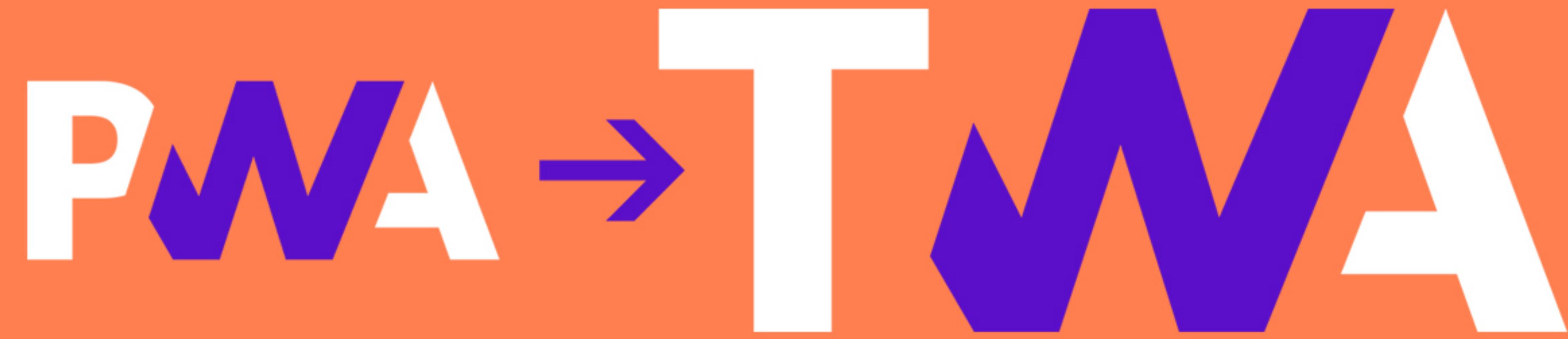


Image from <https://www.ateamindia.com/what-are-pwa-and-twa/>

Trusted Web Activity is a way to open  
your PWA from your Android app using a  
protocol based on Custom Tabs

<https://developers.google.com/web/android/trusted-web-activity>







A screenshot of a GitHub repository page for `GoogleChromeLabs / bubblewrap`. The page includes a navigation bar with icons for back, forward, search, and file operations. The repository name is displayed in the top left, with a star count of 1.1k and a fork count of 76. Below the header are navigation links for Code, Issues (43), Pull requests (3), Actions, Projects (2), Wiki, and more. A dropdown menu shows the `main` branch. The main content area displays the `bubblewrap / README.md` file, showing a commit by `andrebau` to remove a mention to slack channel (#432). It also lists 6 contributors and provides options to view raw code, blame history, or copy/paste. The file content itself is titled **Bubblewrap** and includes a green Node CI status badge indicating it is passing.

GoogleChromeLabs / bubblewrap

Watch 32 Star 1.1k Fork 76

Code Issues 43 Pull requests 3 Actions Projects 2 Wiki ...

main bubblewrap / README.md Go to file ...

andrebau Remove mention to slack channel (#432) ... ✓ Latest commit 95eecbc on 5 Jan History

6 contributors

68 lines (49 sloc) | 2.92 KB Raw Blame

# Bubblewrap

Node CI passing

<https://github.com/GoogleChromeLabs/bubblewrap>

Capable



Reliable



Installable



What's next?

Capable



Reliable



Installable



# (Some) Upcoming APIs

Local font access

Tabbed application mode

Detecting orientation change events

Widgets

Detect/block screenshots

Install-time permissions

Splash screen

Web on



Web on



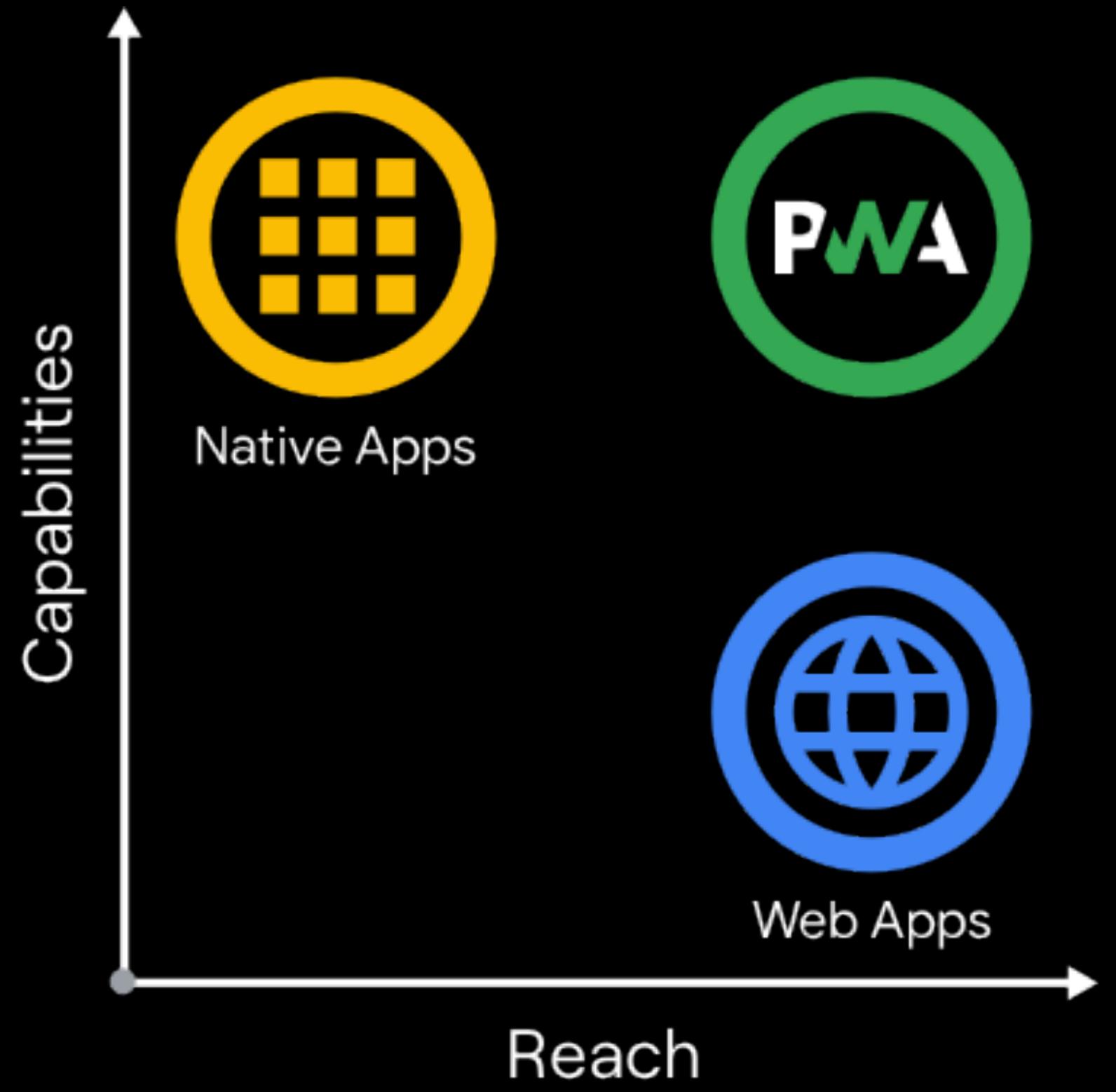
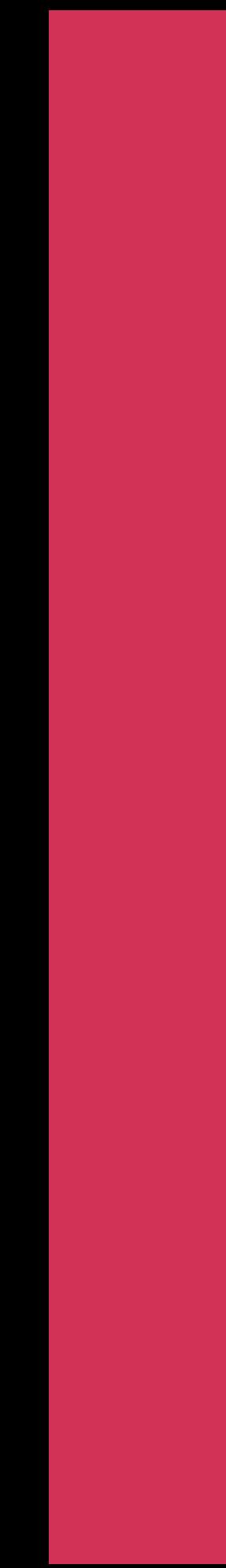


Image from <https://web.dev/what-are-pwas>

Web on Everything!



# Thank you!

Ire Aderinokun 

COO & VP Engineering of BuyCoins

Google Web Expert

[ireaderinokun.com](http://ireaderinokun.com)

[bitsofco.de](http://bitsofco.de)

@ireaderinokun

