PWAs in 2021

Ire Aderinokun @ An Event Apart
“Spring Summit” 2021


od
Phone
Internet

“There’s an app for that.”

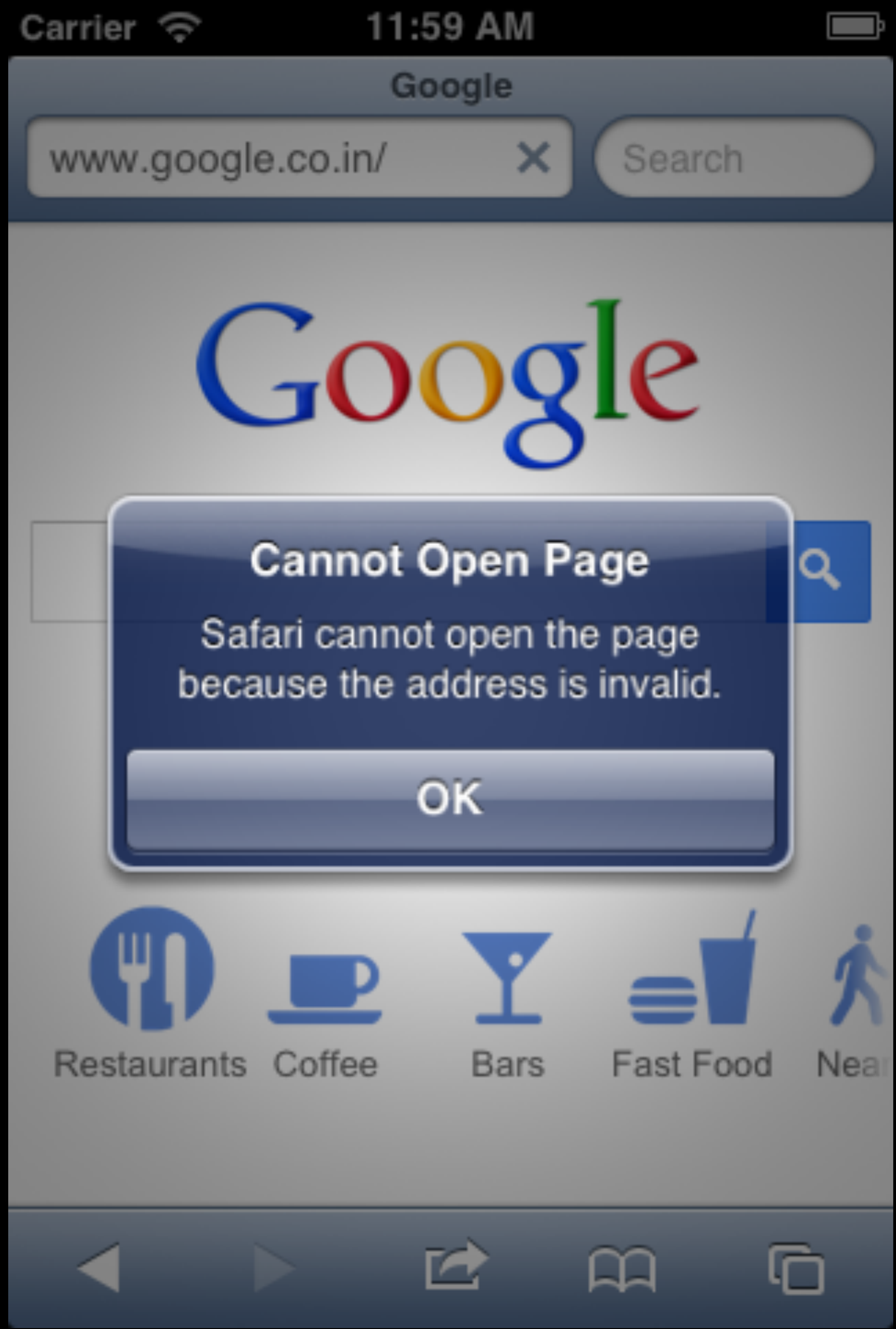
Mobile websites were
less usable

Websites were less capable

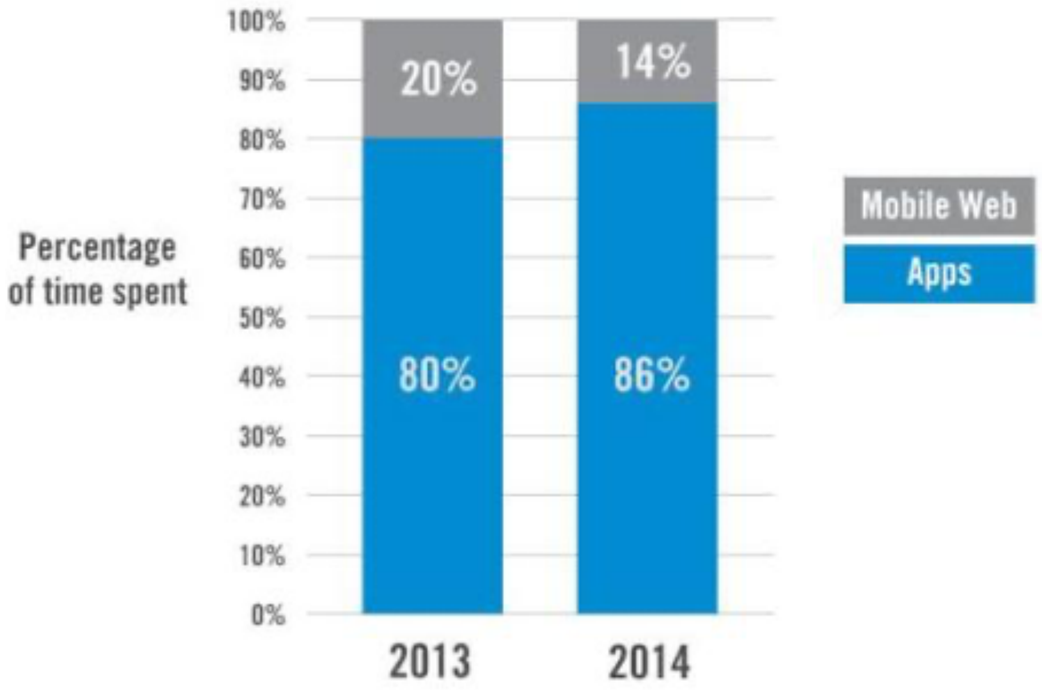
Image from https://dribbble.com/shots/2518631-Notifications-Illustration-Animation

Websites were harder to access

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

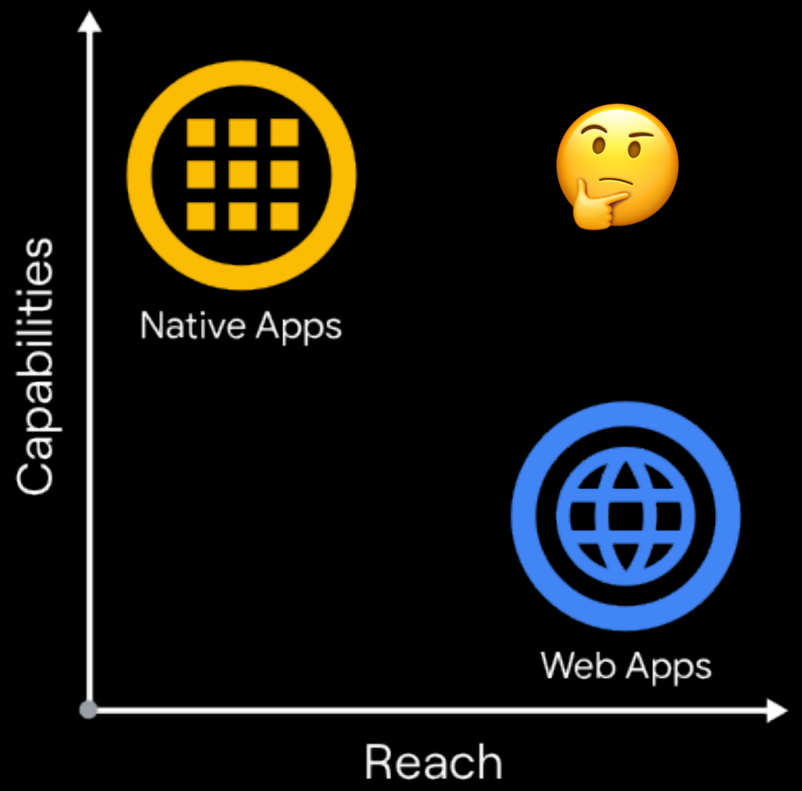


Image from https://web.dev/what-are-pwas

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

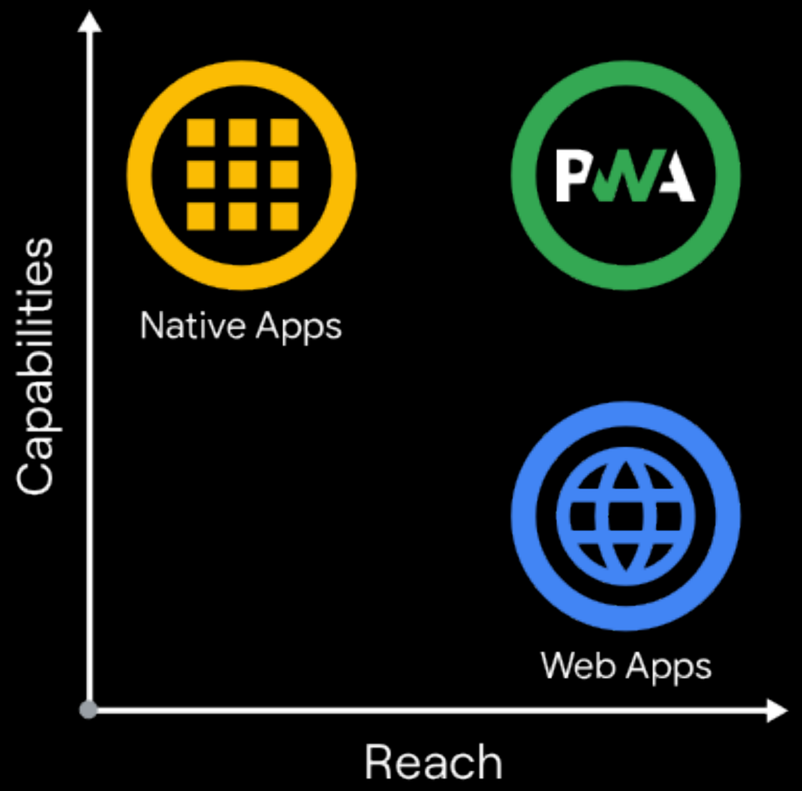


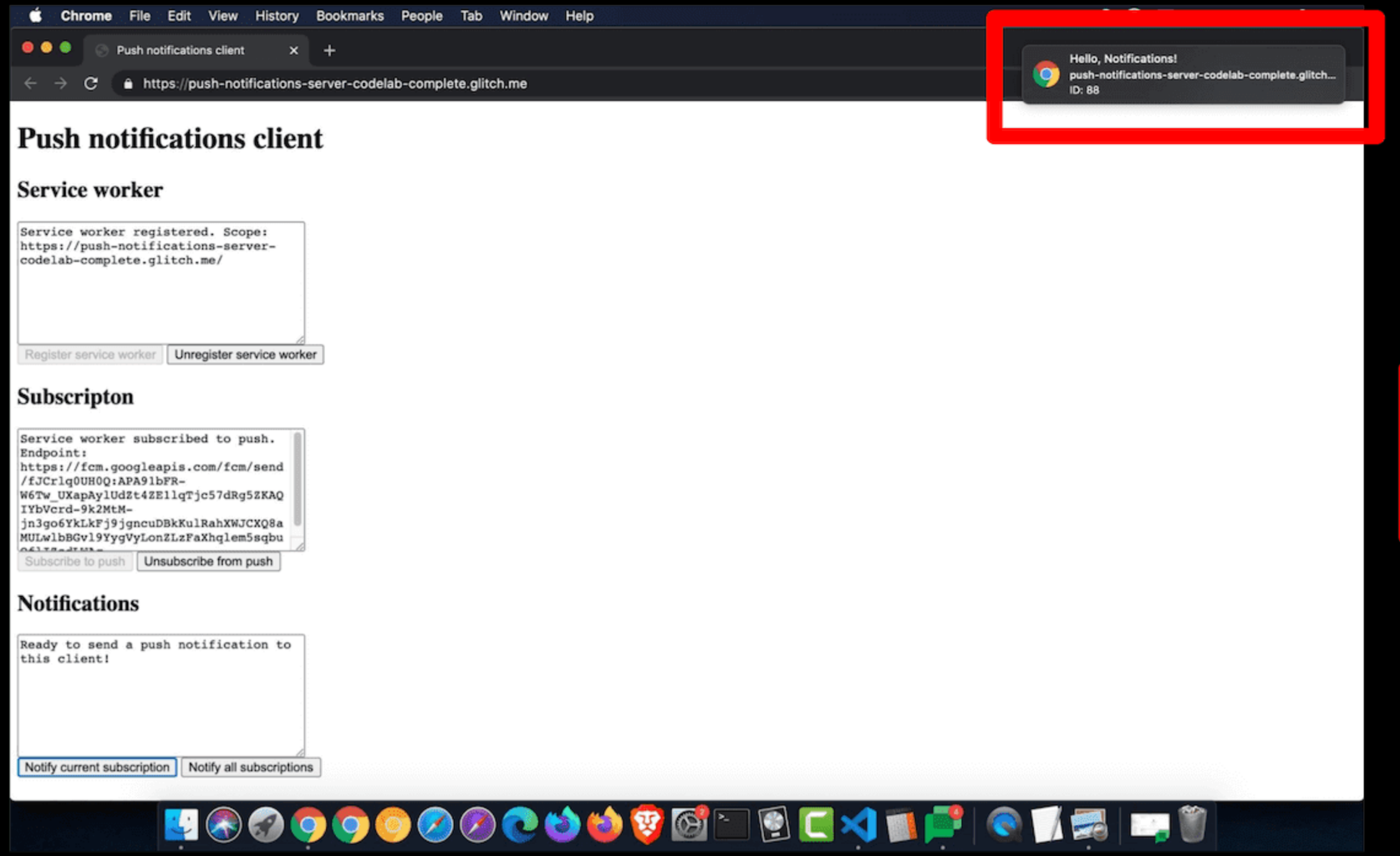
Image from https://web.dev/what-are-pwas

Capable

Reliable

Installable

Capable — uses modern APIs to deliver an app-like experience



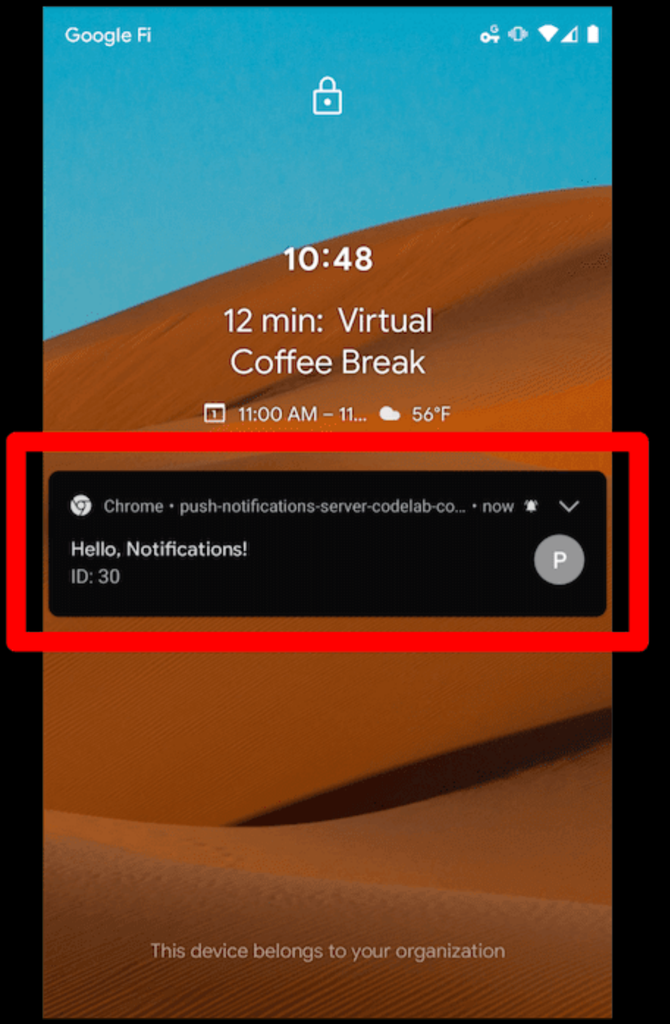
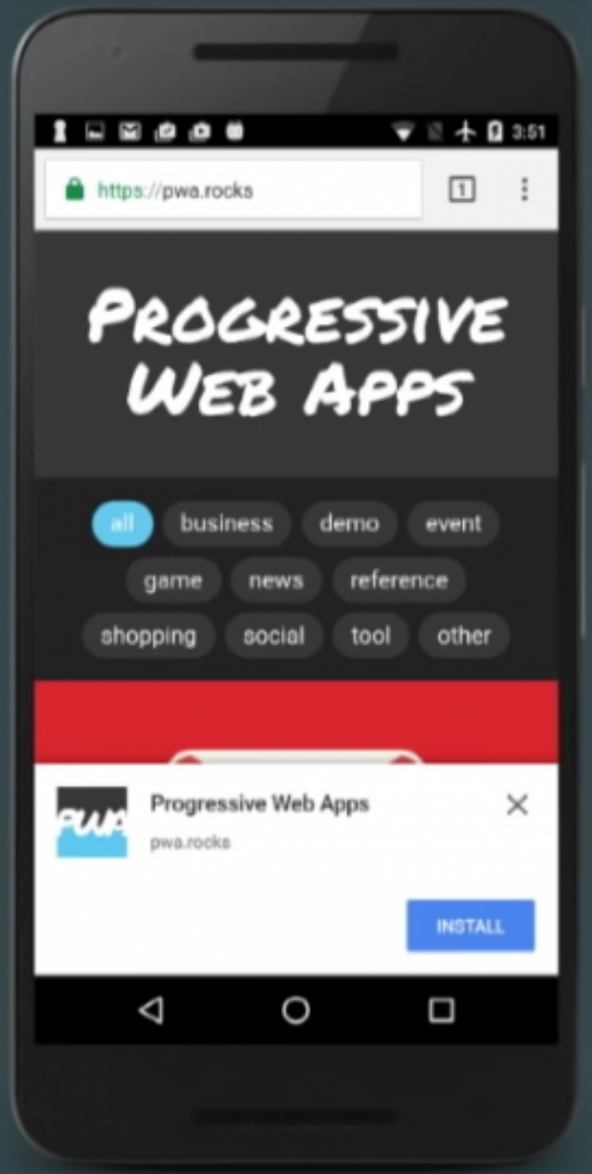


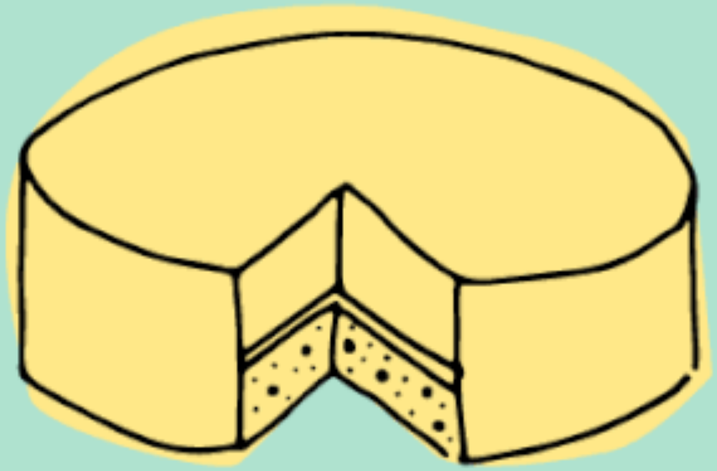
Image from https://web.dev/push-notifications-overview/

Reliable — works well regardless of network

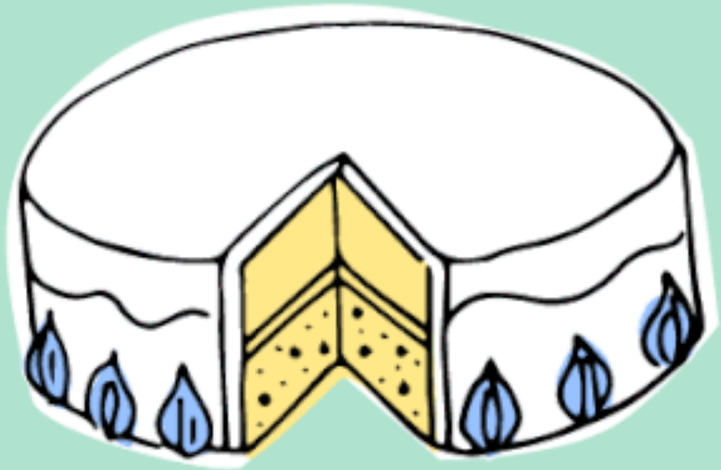

OFFLINE

Installable — can be installed to the device and used outside the browser


3:51
https://pwa.rocks
PROGRESSIVE WEB APPS
all
business
demo
event
game
news
reference
shopping
social
tool
other
Progressive Web Apps
pwa.rocks
INSTALL



Standard website



Fully capable, reliable,
and installable PWA

Image from https://www.shopify.com/partners/blog/what-is-progressive-enhancement-and-why-should-you-care

How did PWAs fare in 2016?

Capable 🤔

Reliable

Installable

Critical capabilities

Push notifications & badging

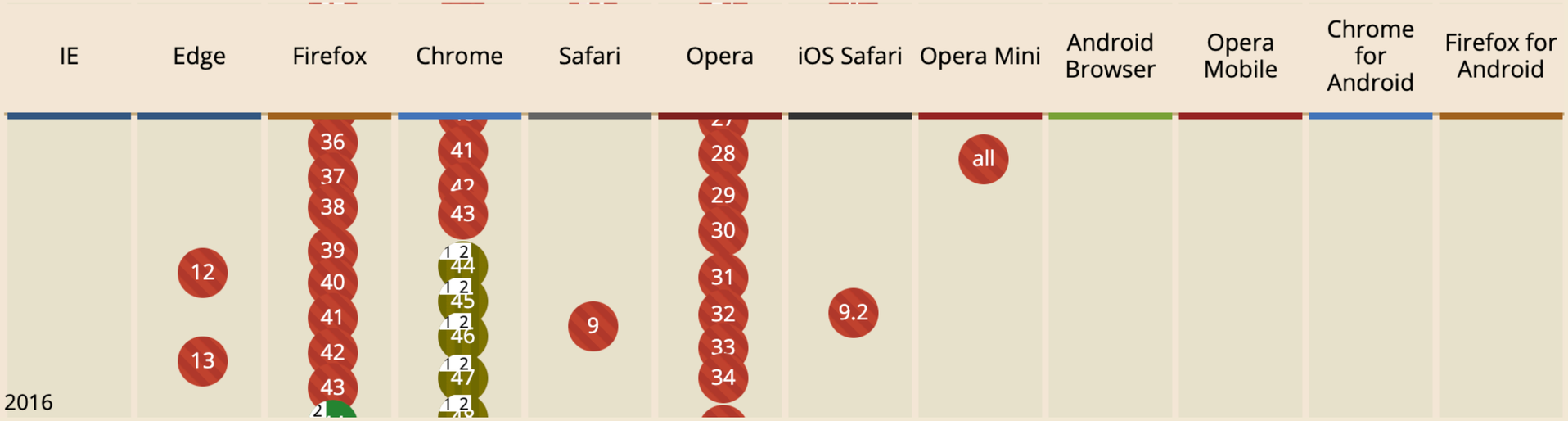
Background sync & fetch

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.

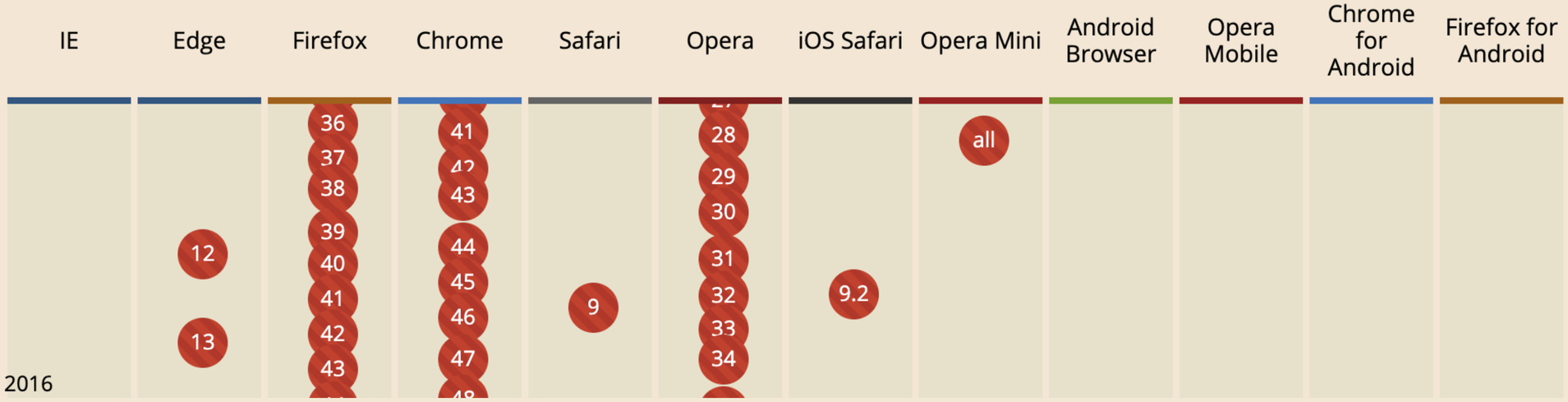
Current aligned | Usage relative | Date relative | Filtered | All



Background Sync API - UNOFF

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

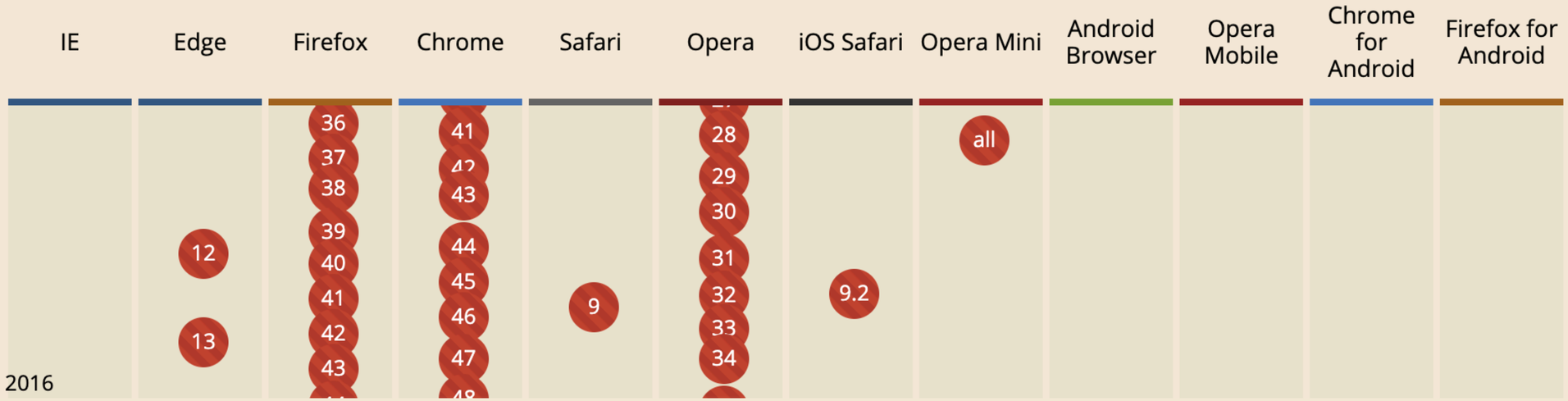
Current aligned | Usage relative | Date relative | Filtered | All



File System Access API - UNOFF

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

Current aligned | Usage relative | Date relative | Filtered | All



```
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

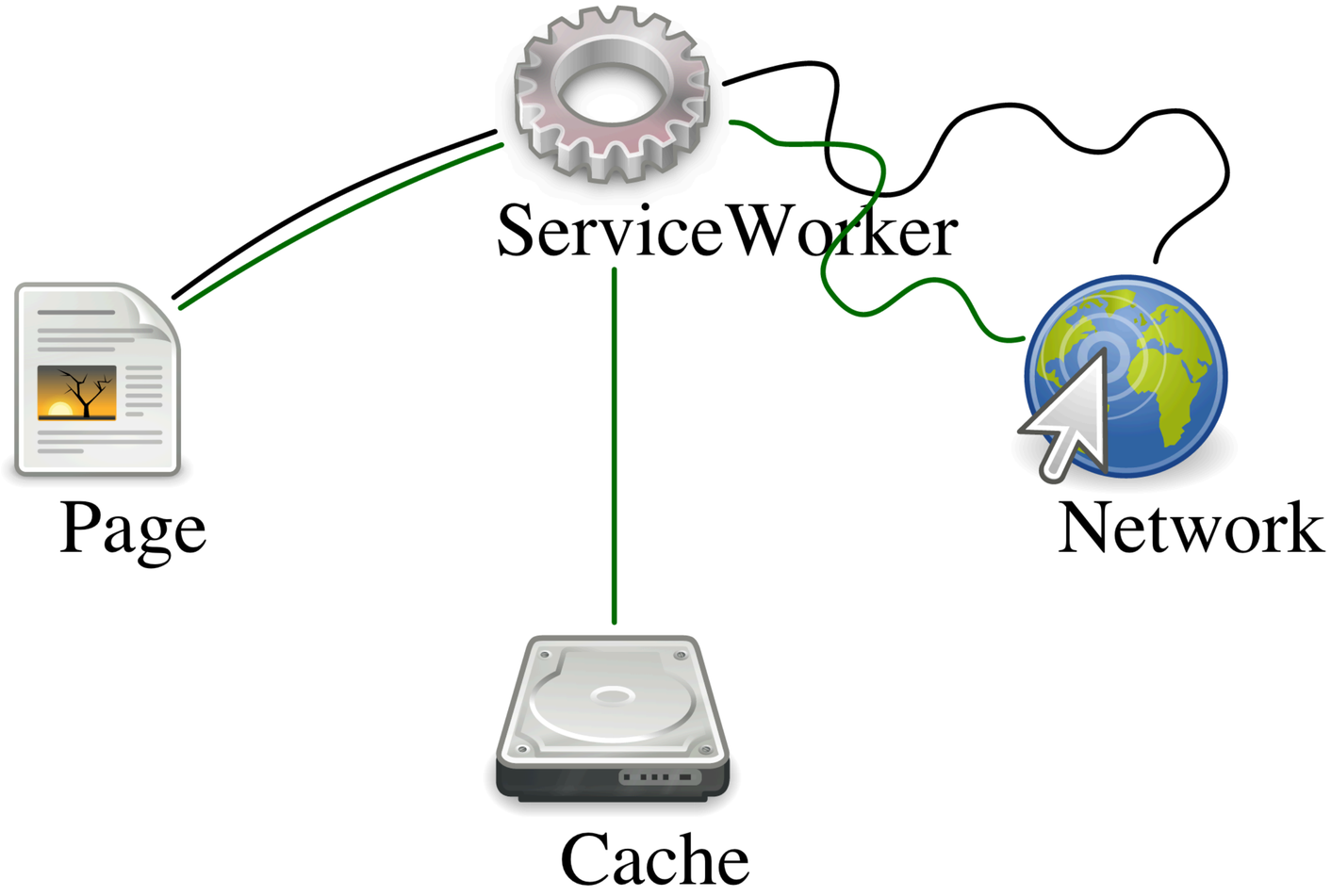


Image from https://pjagajitprusty.github.io/PWD/

Service Workers - CR

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

Current aligned | Usage relative | Date relative | Filtered | All

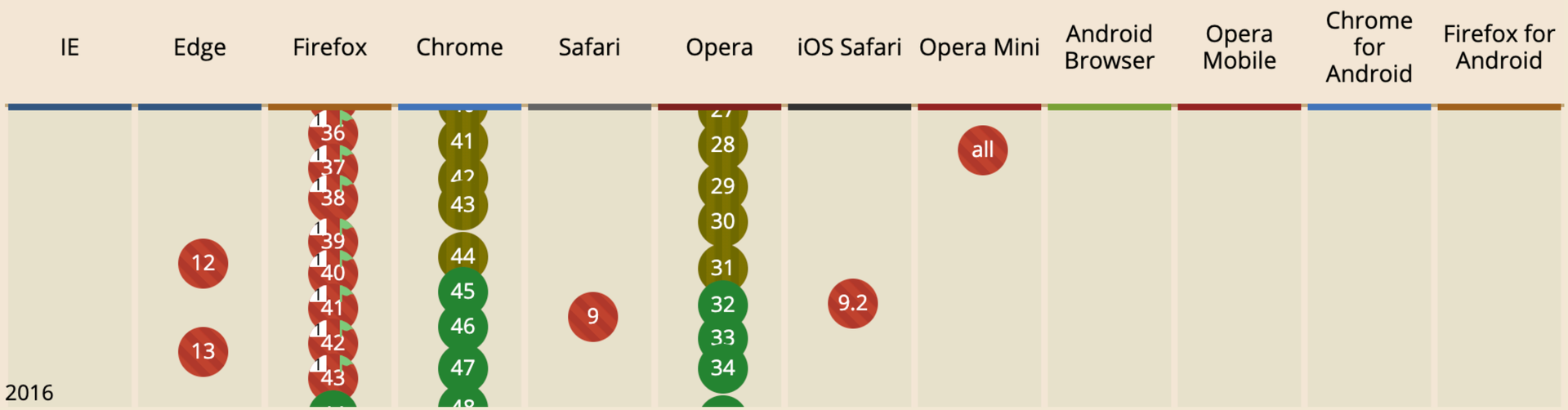
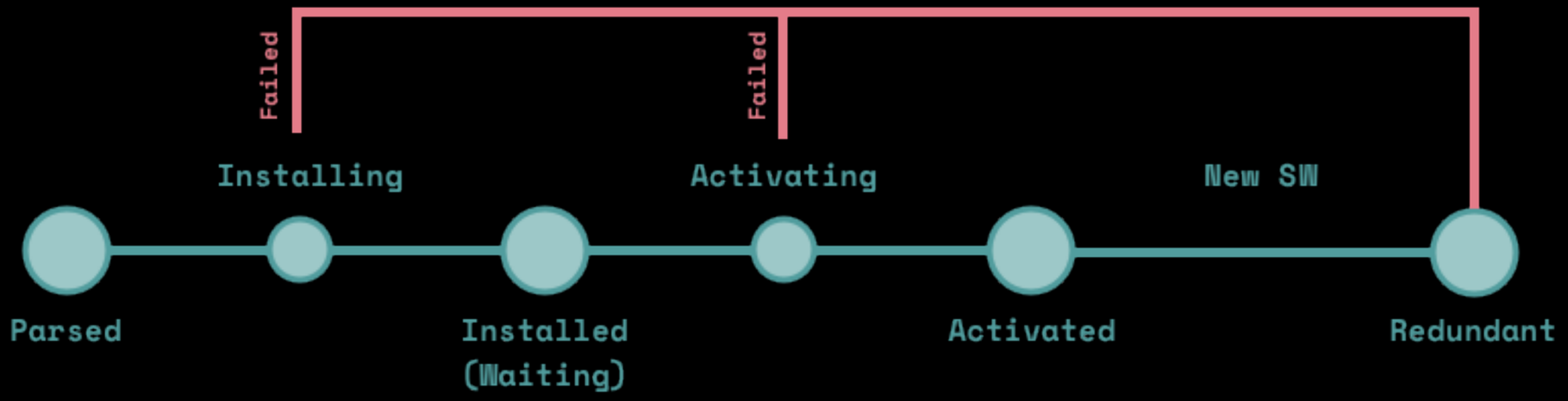
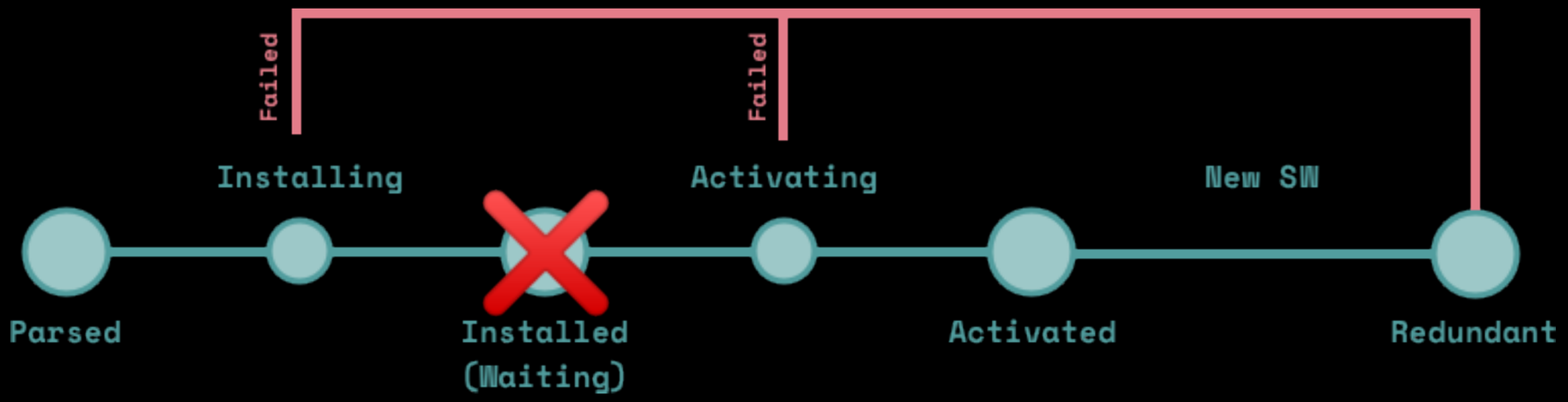


Image from https://explore.easyprojects.net/blog/how-to-deal-with-difficult-team-members


Failed
Failed
Installing
Activating
New SW
Parsed
Installed
(Waiting)
Activated
Redundant

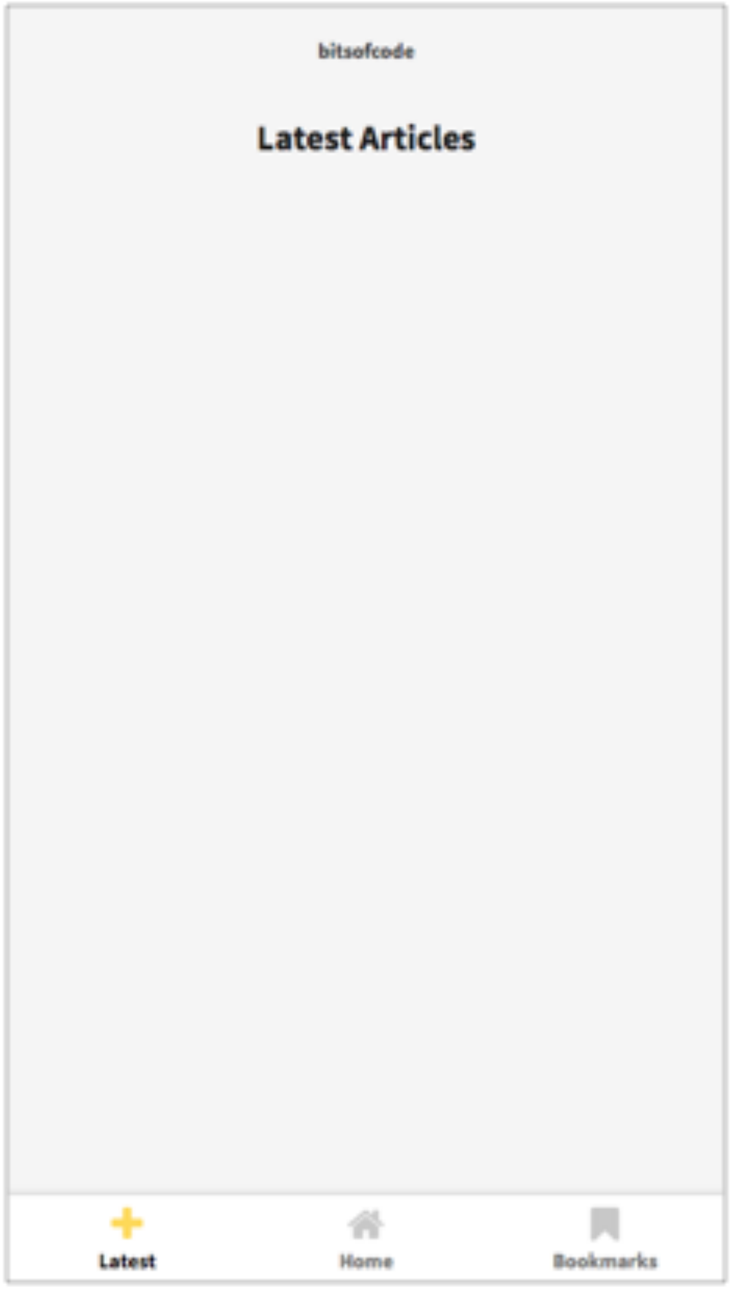


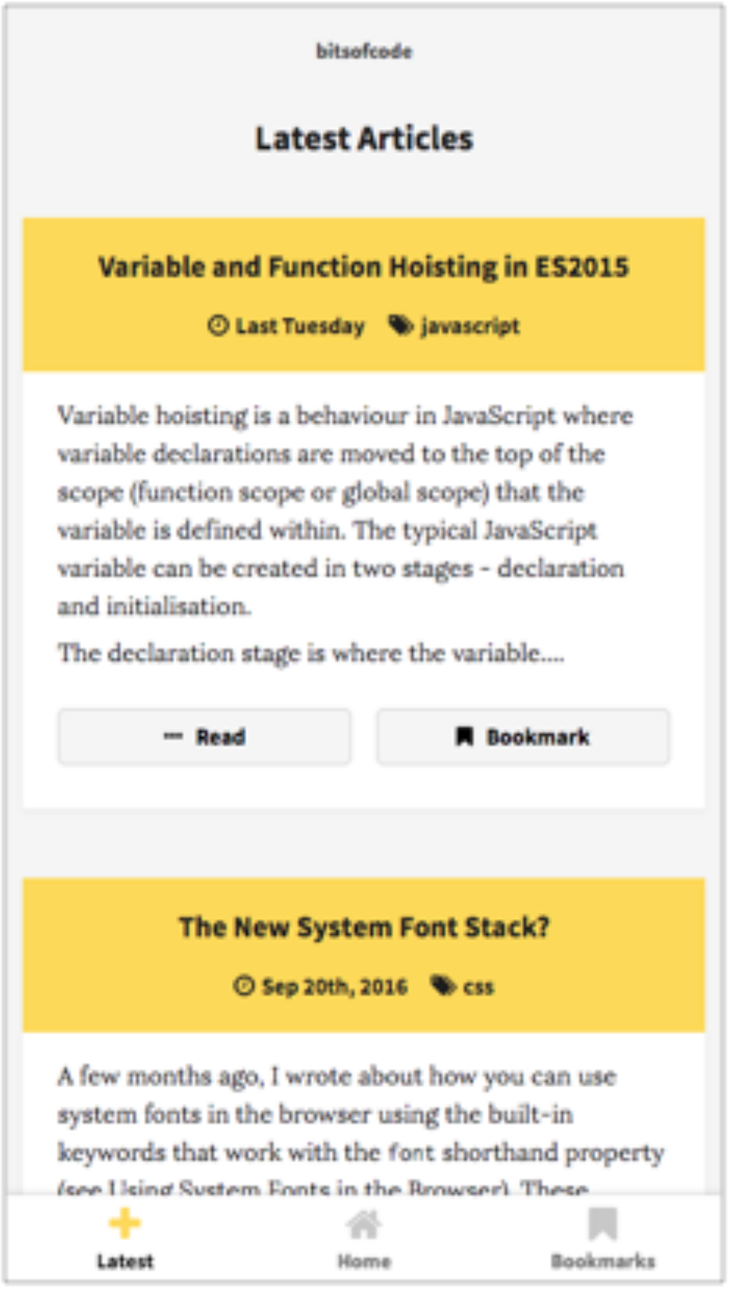
self.skipWaiting()

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


bitsofcode
Latest Articles
Latest
Home
Bookmarks


bitsofcode
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
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));
}
```

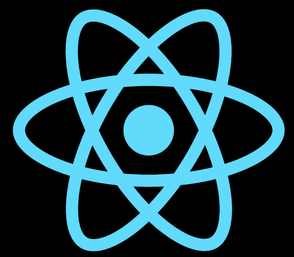
Add content to app shell

```
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));

}
```


PWA


Capable ☹️

Reliable ☹️

Installable

Capable ☹️

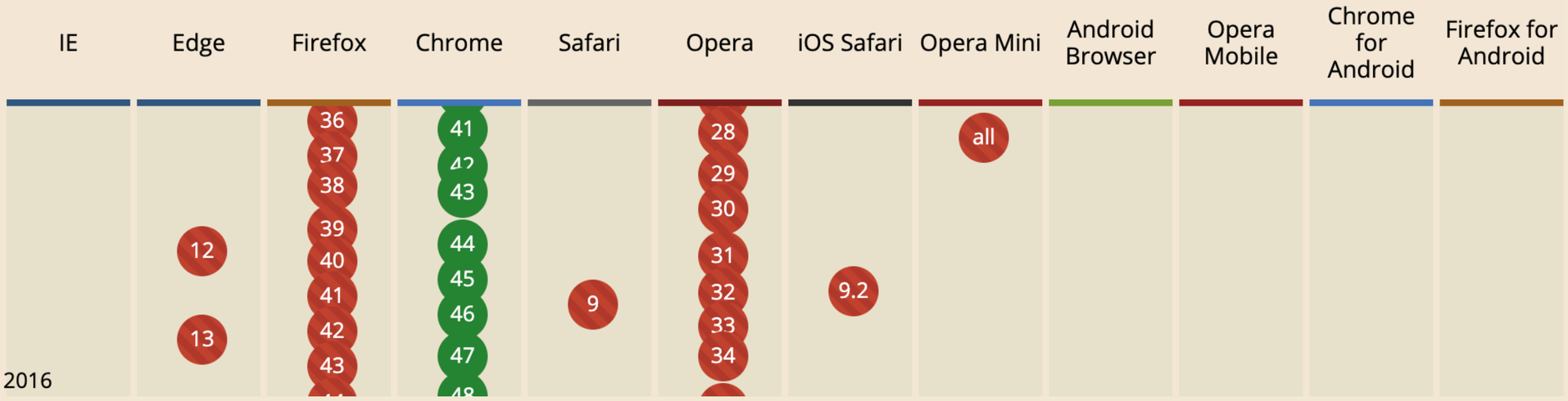
Reliable ☹️

Installable 🤔

Add to home screen (A2HS) - WD

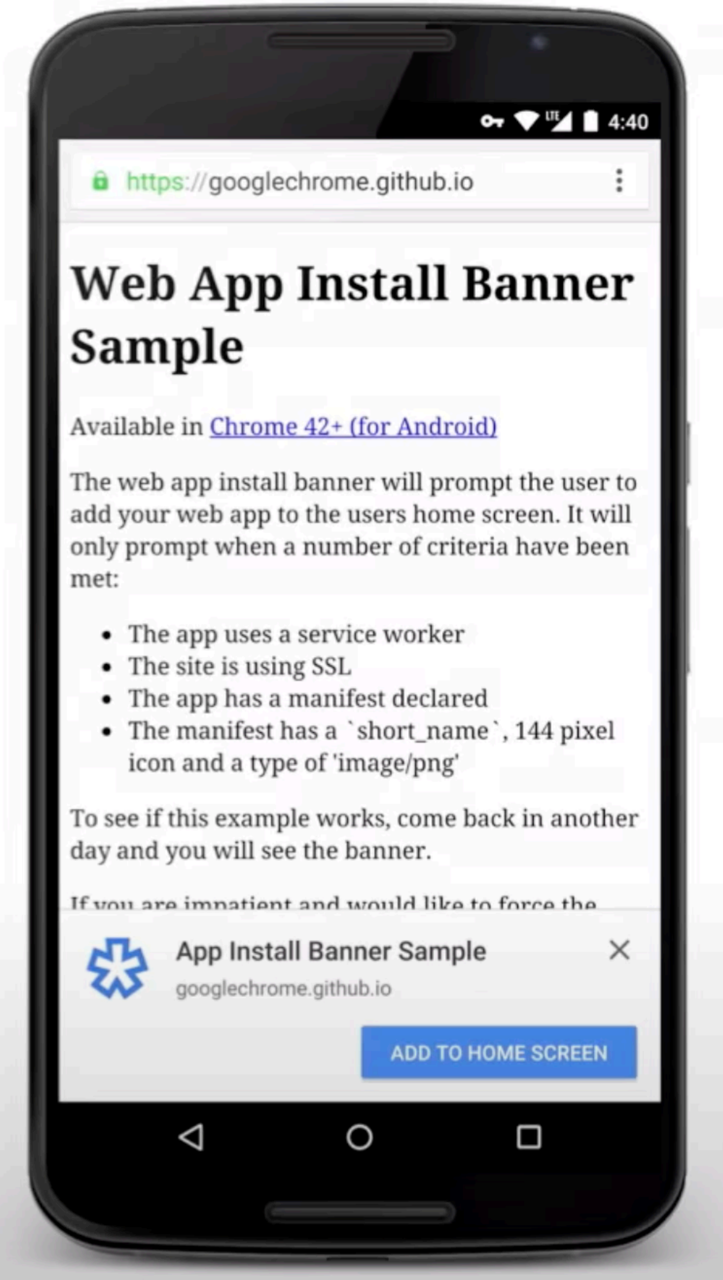
The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load it's assets through a Service Worker.

Current aligned | Usage relative | Date relative | Filtered | All



Requirements for “Add to Home Screen”

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



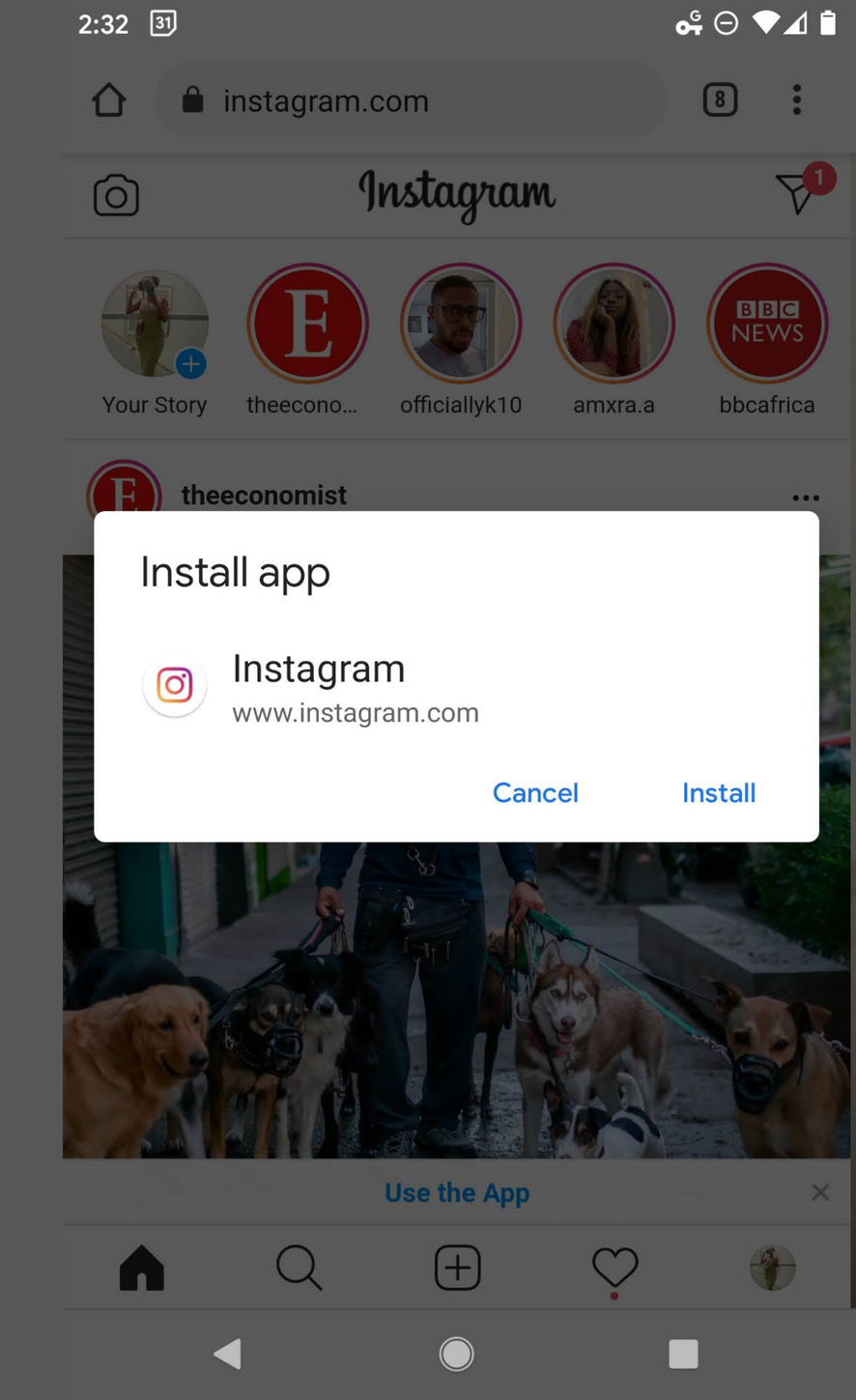
Add to Home Screen !== Install

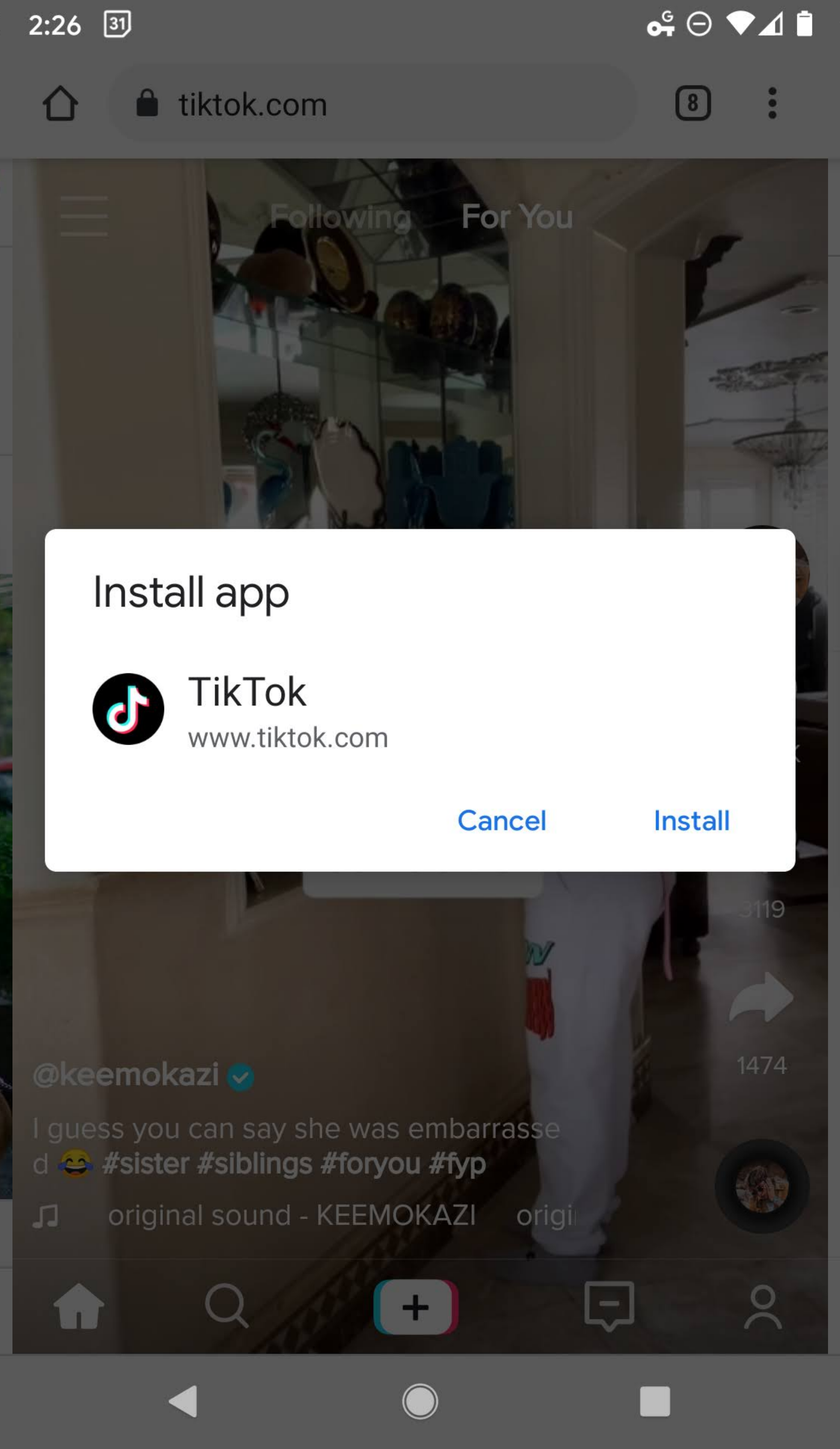
Capable ☹️

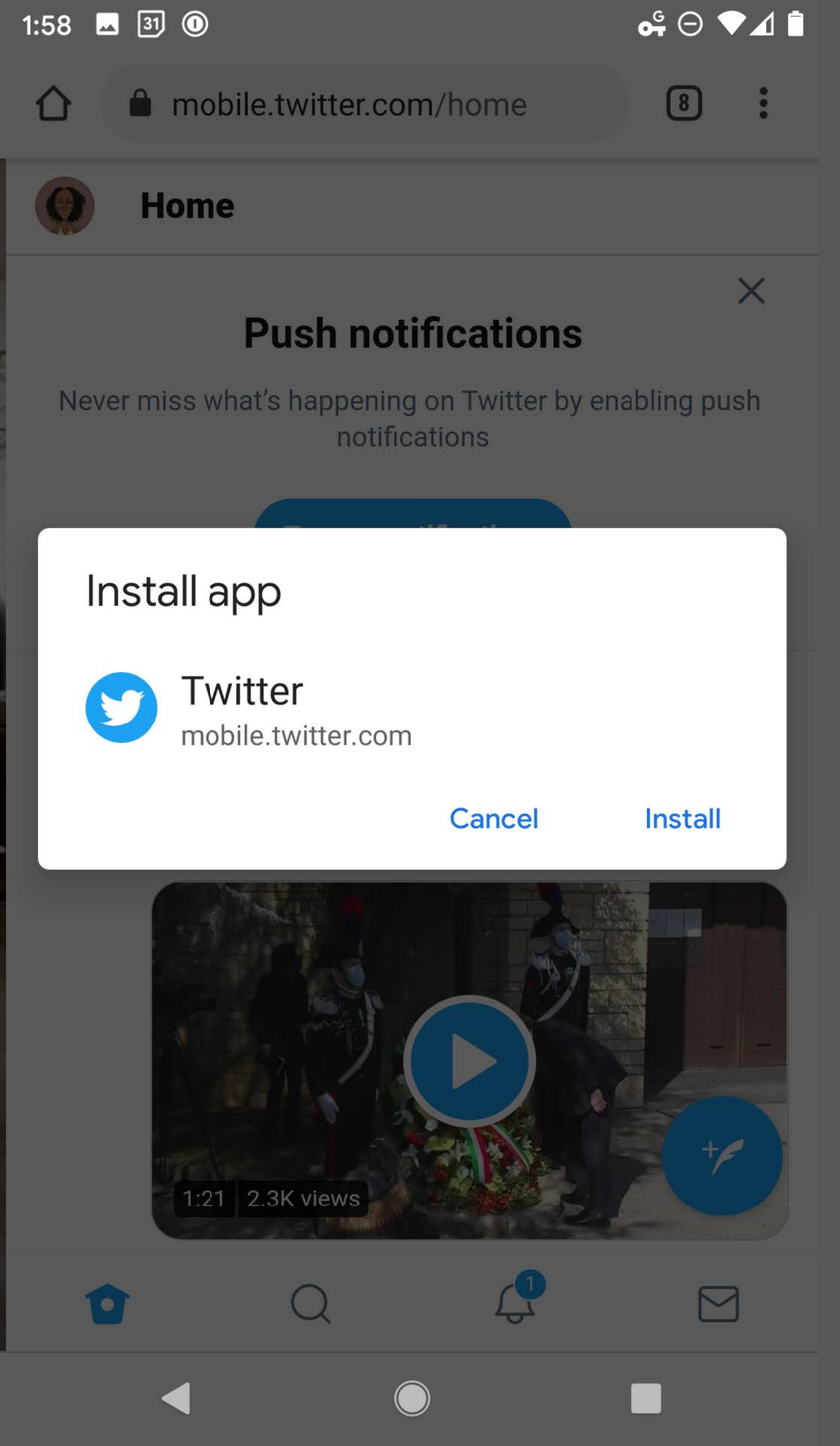
Reliable ☹️

Installable ☹️

PWAs five years on


2:32
instagram.com
Instagram
Your Story
theecono...
officiallyk10
amxra.a
bbcafrica
theeconomist
Install app
Instagram
www.instagram.com
Cancel
Install
Use the App


2:26
tiktok.com
Following
For You
Install app
TikTok
www.tiktok.com
Cancel
Install
@keemokazi
I guess you can say she was embarrasse
d 😂 #sister #siblings #foryou #fyp
original sound - KEEMOKAZI
1474


1:58
mobile.twitter.com/home
Home
Push notifications
Never miss what's happening on Twitter by enabling push notifications
Install app
Twitter
mobile.twitter.com
Cancel
Install
1:21
2.3K views

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

https://fırt.dev/pwa-2021

Capable 🤔

Reliable

Installable

Microsoft

intel

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

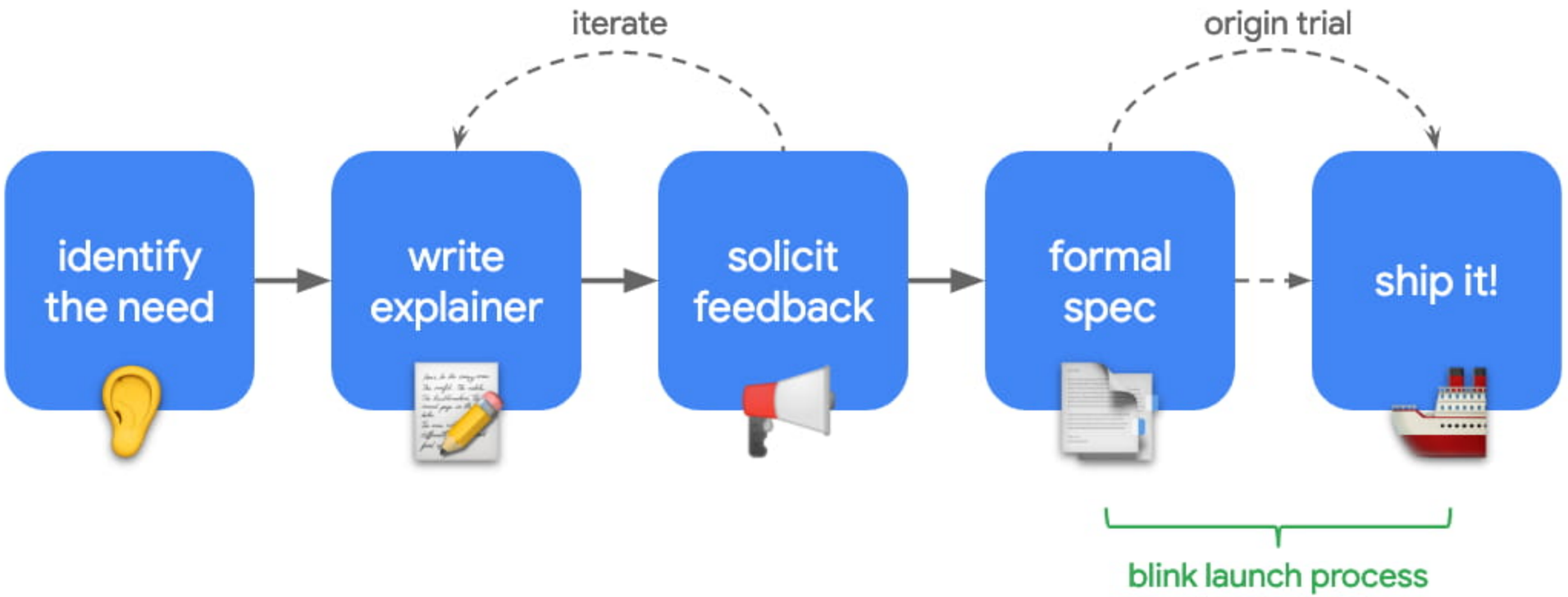


Image from https://developers.google.com/web/updates/capabilities

What has been shipped? 🚀

Critical capabilities

Push notifications & badging

Background sync & fetch

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

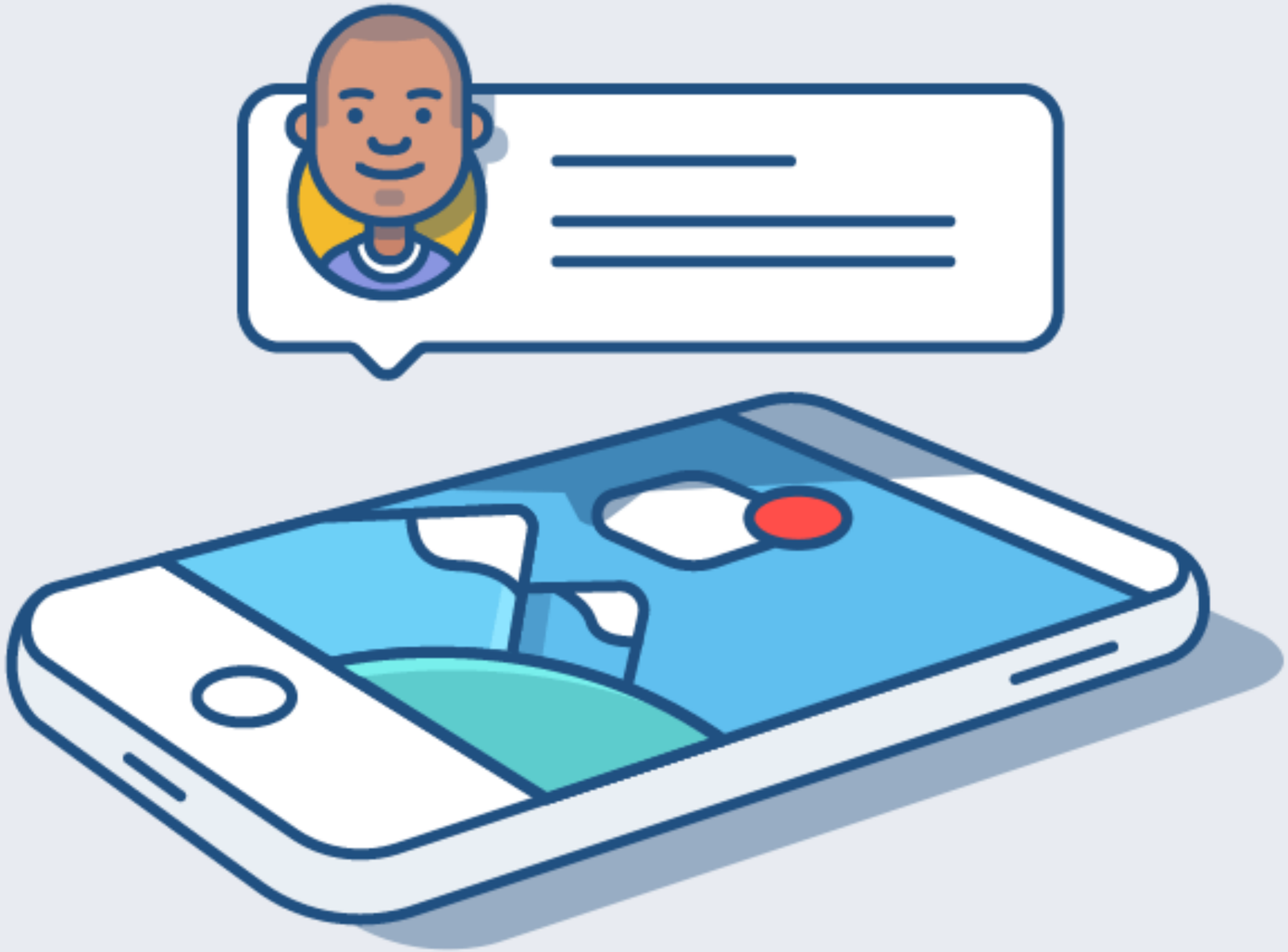


Image from https://dribbble.com/shots/2518631-Notifications-Illustration-Animation

APIs

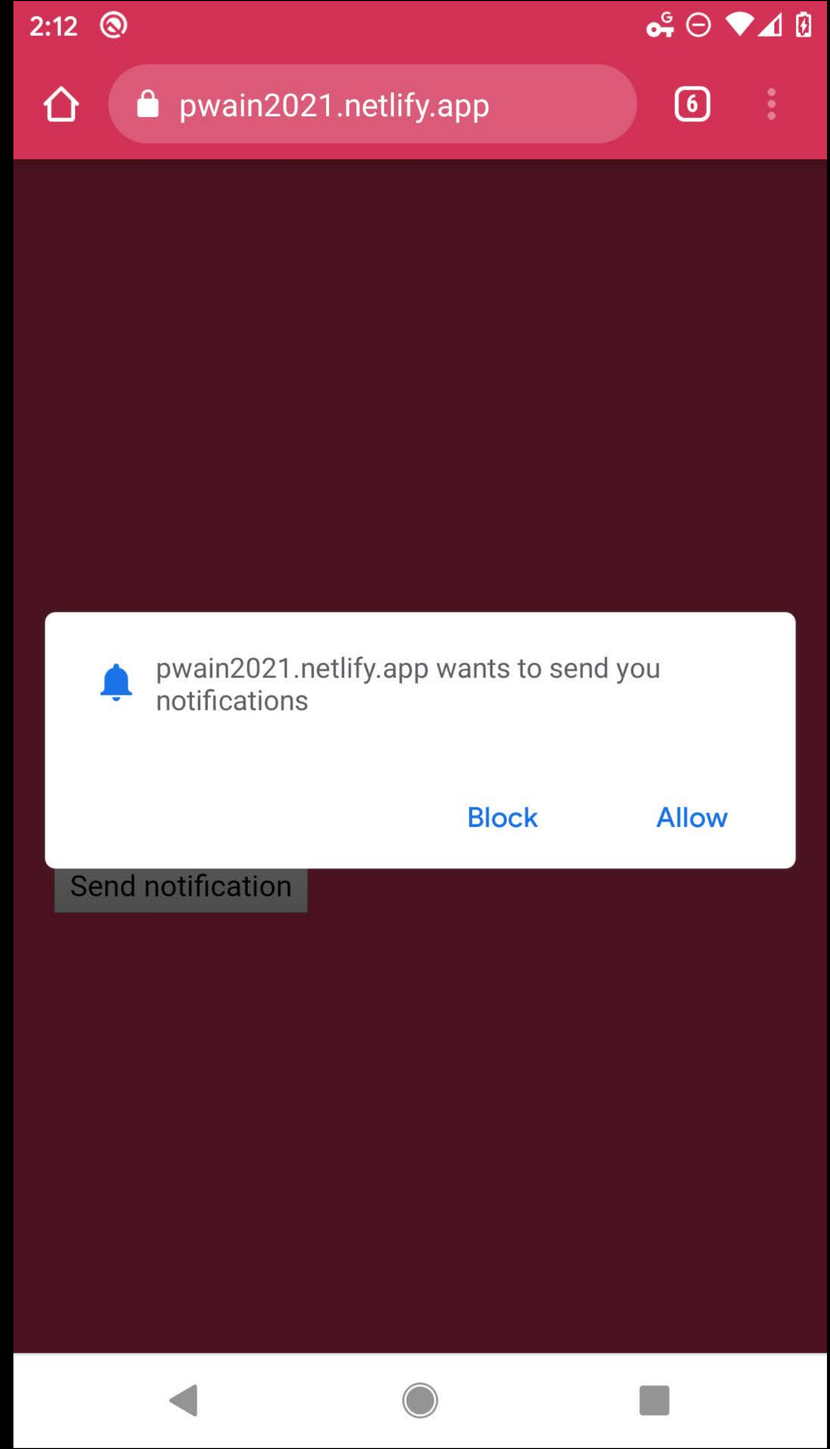
Notifications API — display of notification message

Push API — allows browsers to receive messages from a server

App Badging API — display badge on app icon

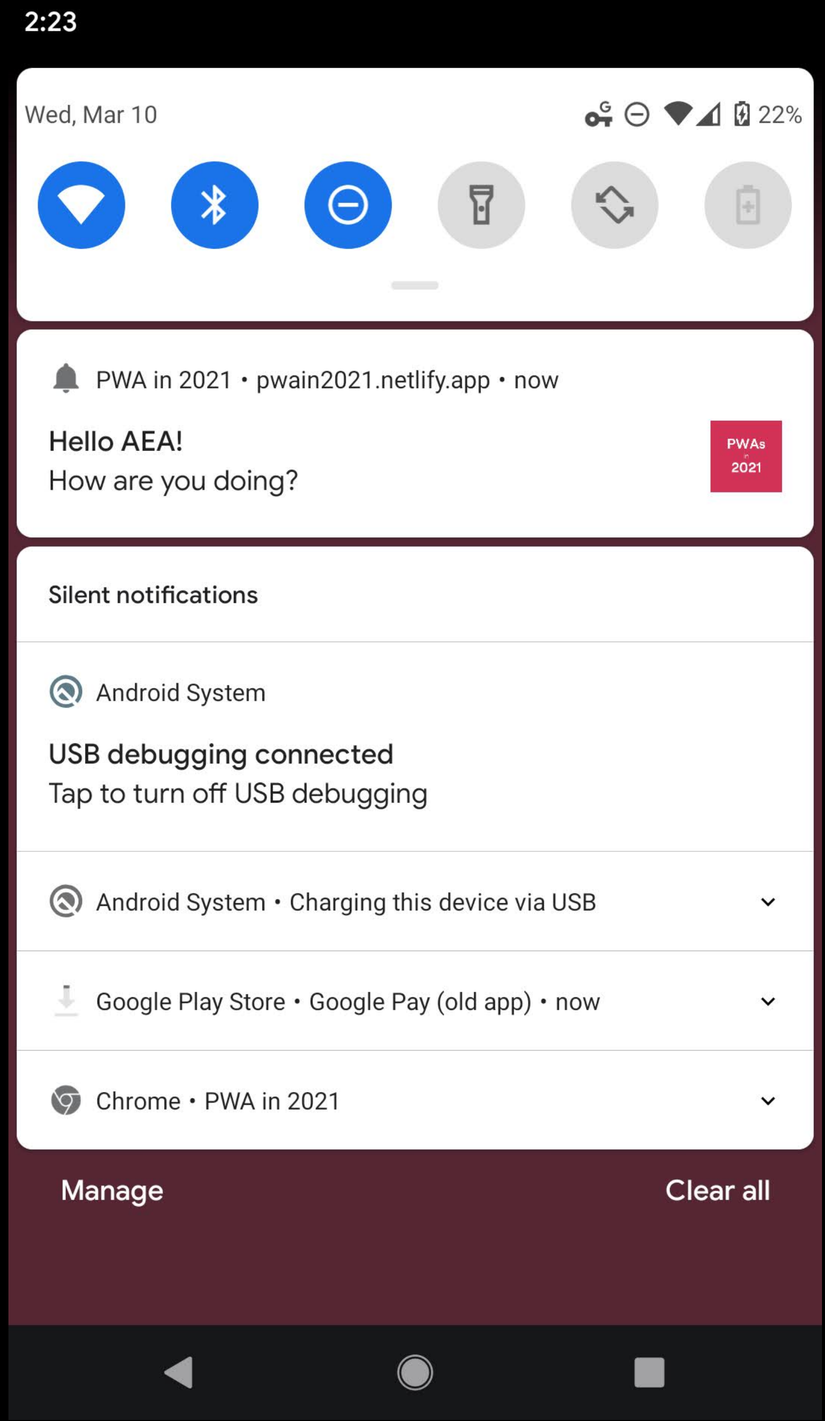
Request permission to display push notifications

```
if ('Notification' in window) {

    Notification.requestPermission();

}
```



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

Image from https://web.dev/push-notifications-overview/

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);
```

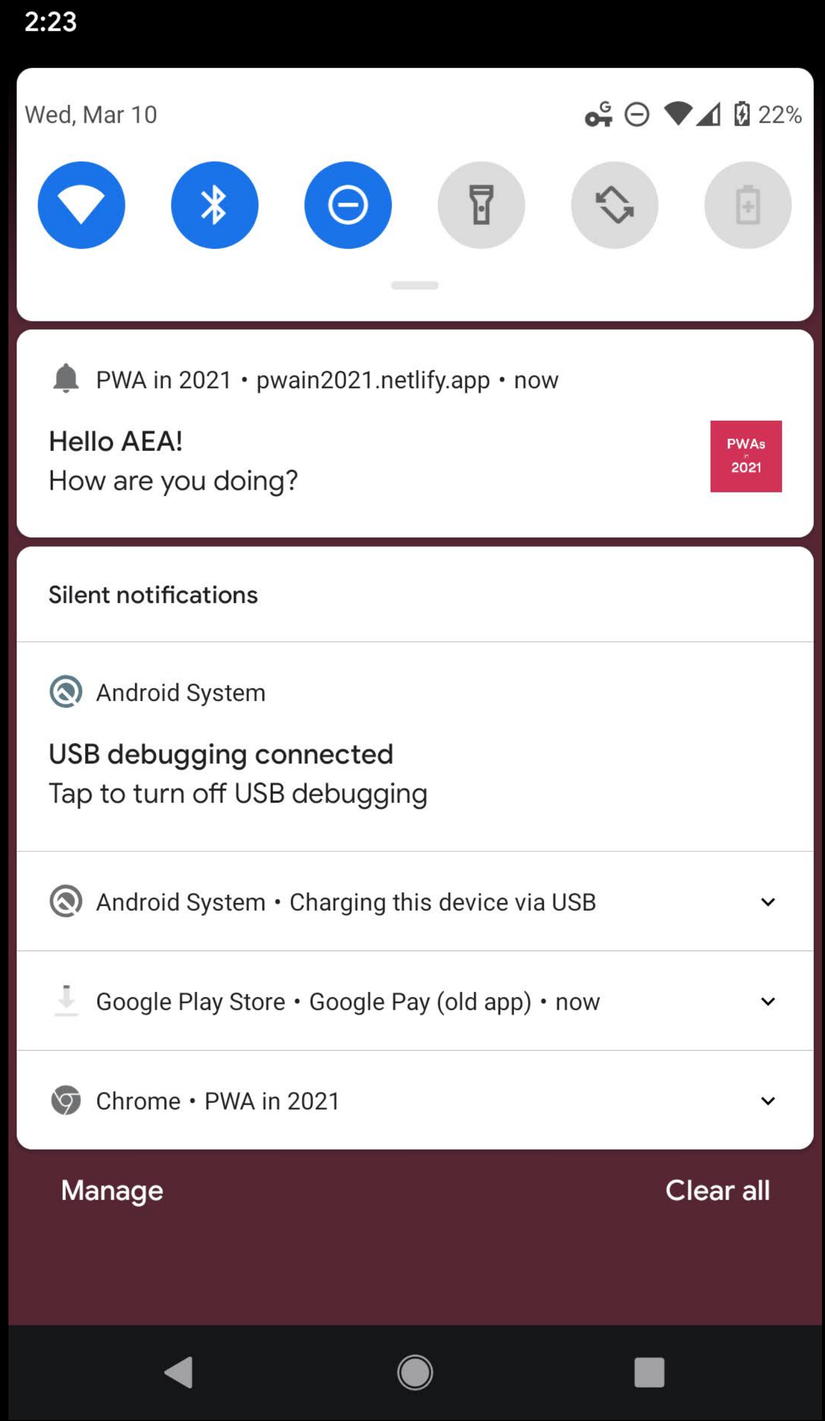
Image from https://web.dev/push-notifications-overview/

Receive notification in service worker

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


});
```

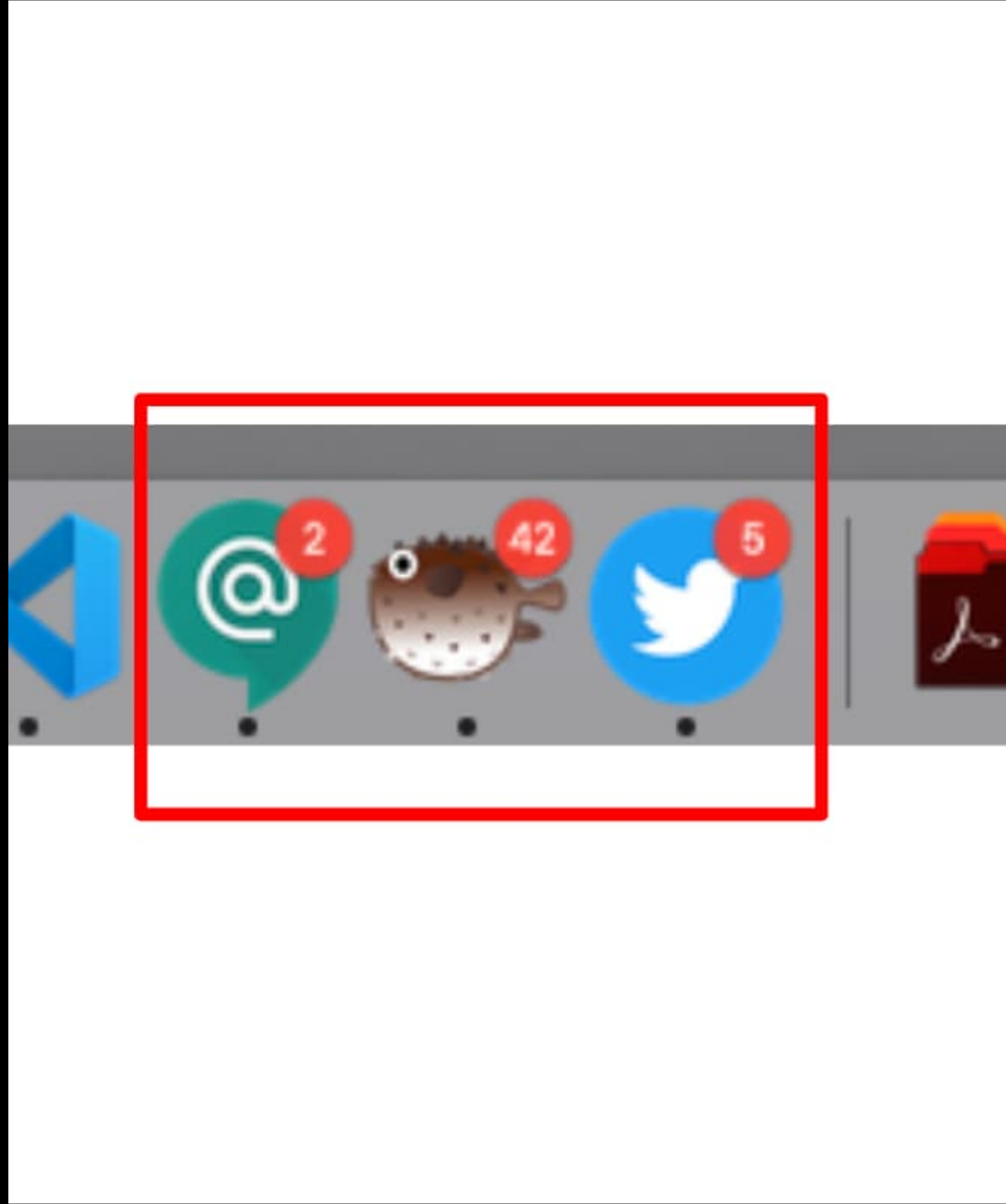
Display notification

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



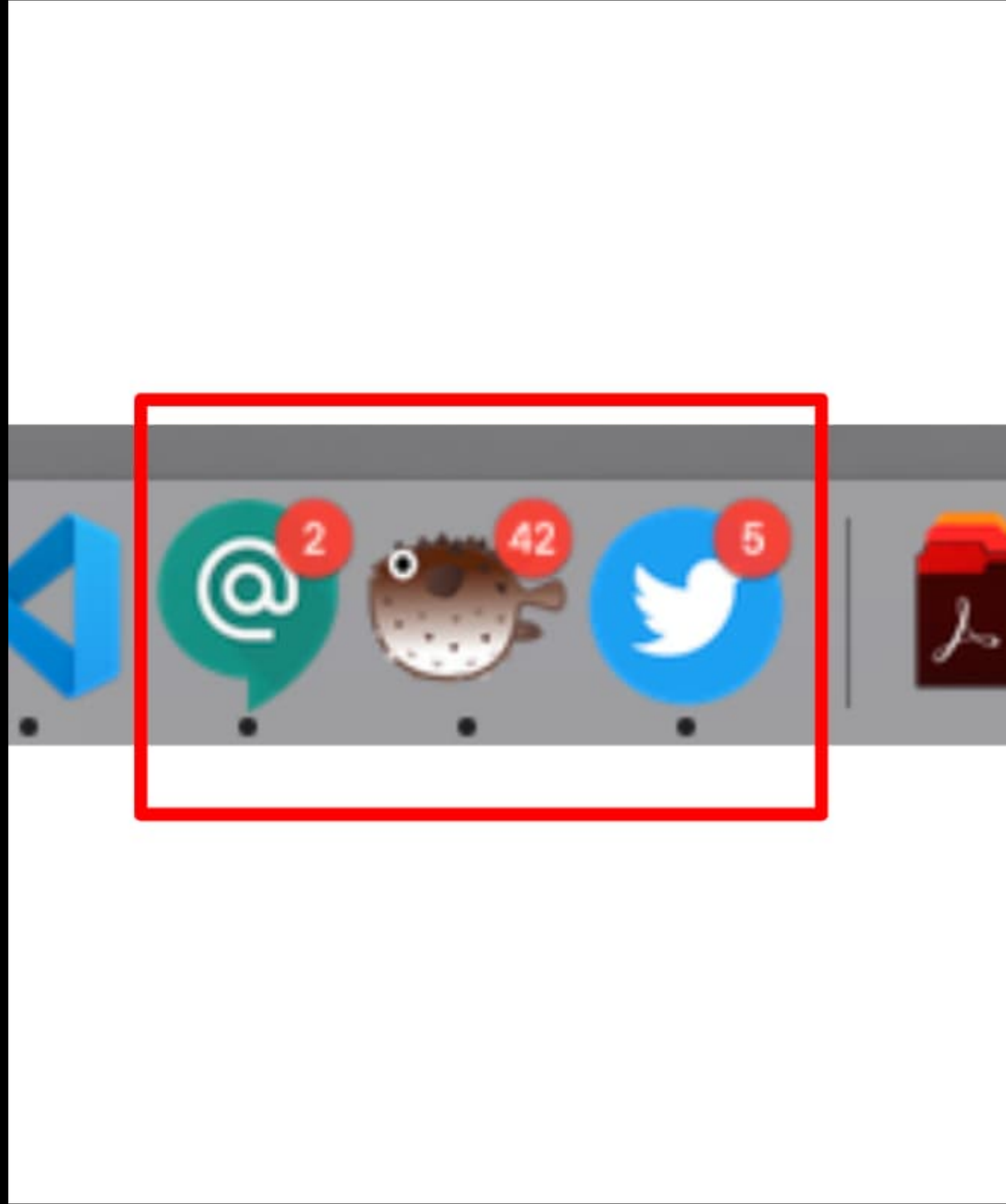
Set app badge

```
navigator.setAppBadge(badgeNumber);

navigator.clearAppBadge();
```



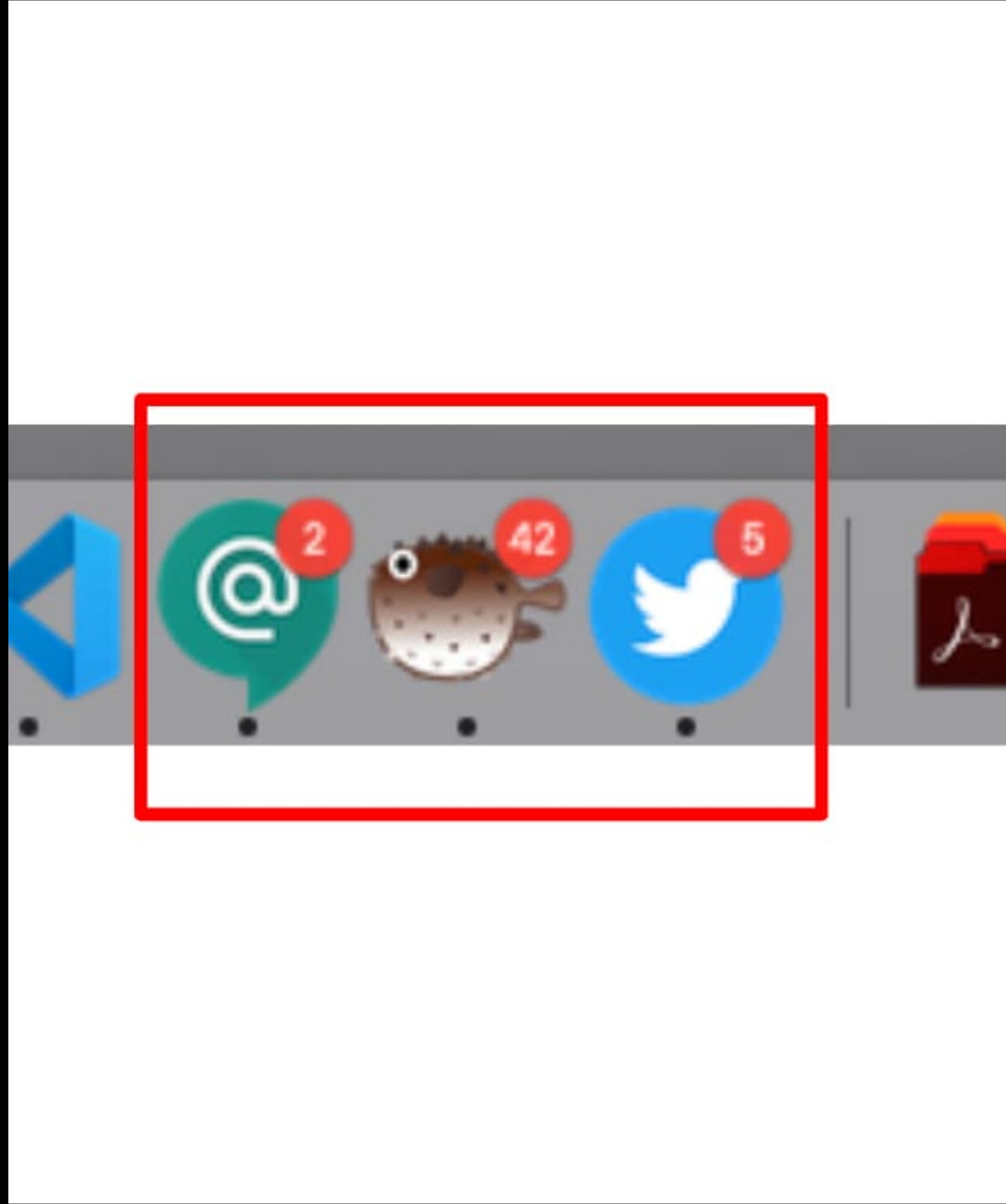
Set app badge

```
navigator.setAppBadge(badgeNumber);

navigator.clearAppBadge();
```



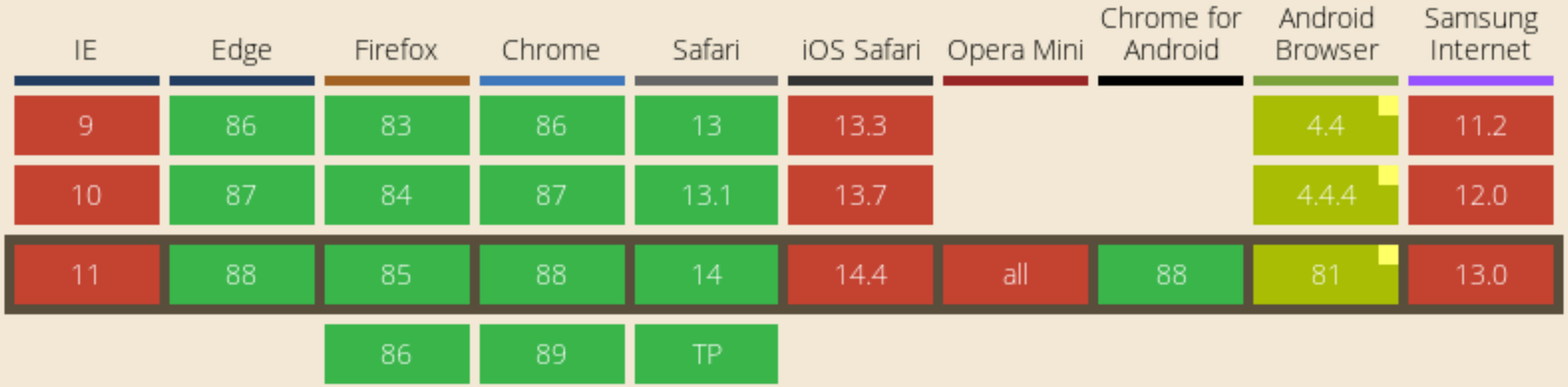
Set app badge

```
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.

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: 76.57% + 0.26% = 76.83%

https://developer.apple.com/videos/play/wwdc2013/614/

Navigator API: setAppBadge

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android
	12-80		4-80								
6-10	81-87	2-84	81-87	3.1-13.1	10-72	3.2-13.7		2.1-4.4.4	12-12.1		
11	88	85	88	14	73	14.4	all	81	59	88	85
		86-87	89-91	TP							

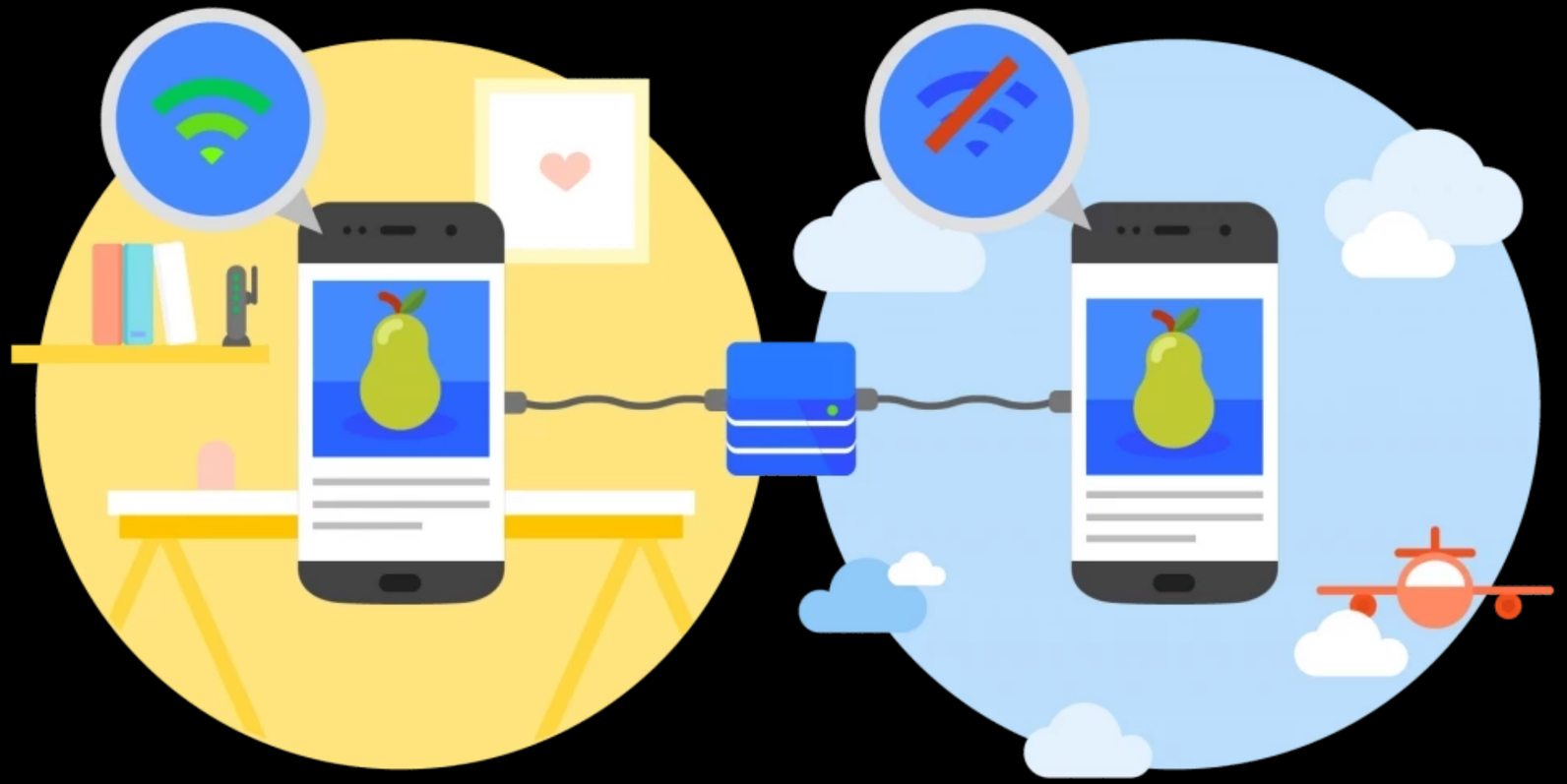


Image from https://robert-askam.co.uk/posts/how-do-make-a-post-request-to-a-server-using-background-sync-and-service-workers

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


OFFLINE

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

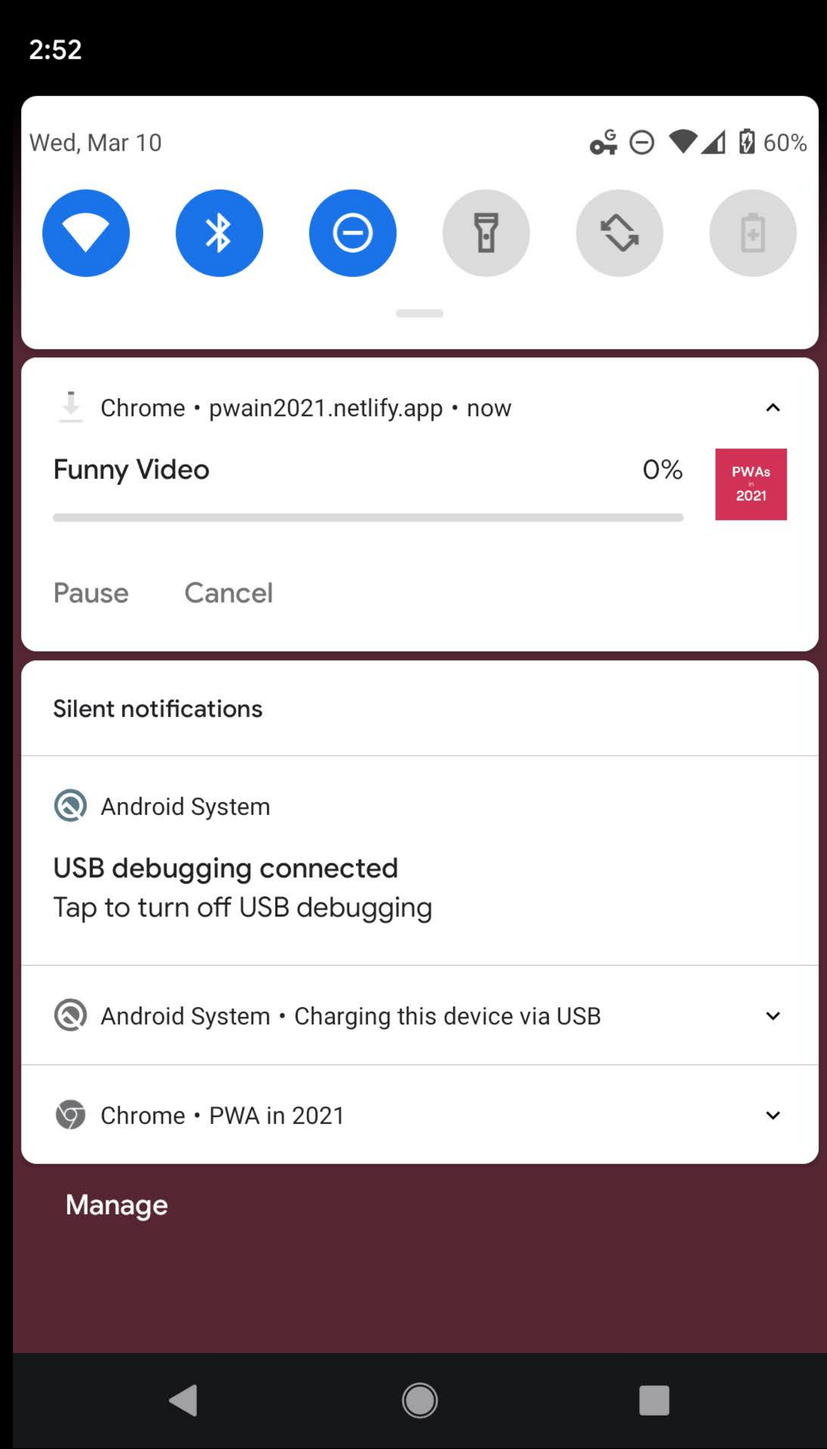
```
navigator.serviceWorker.ready.then(async (registration) => {

    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,

    });

});
```

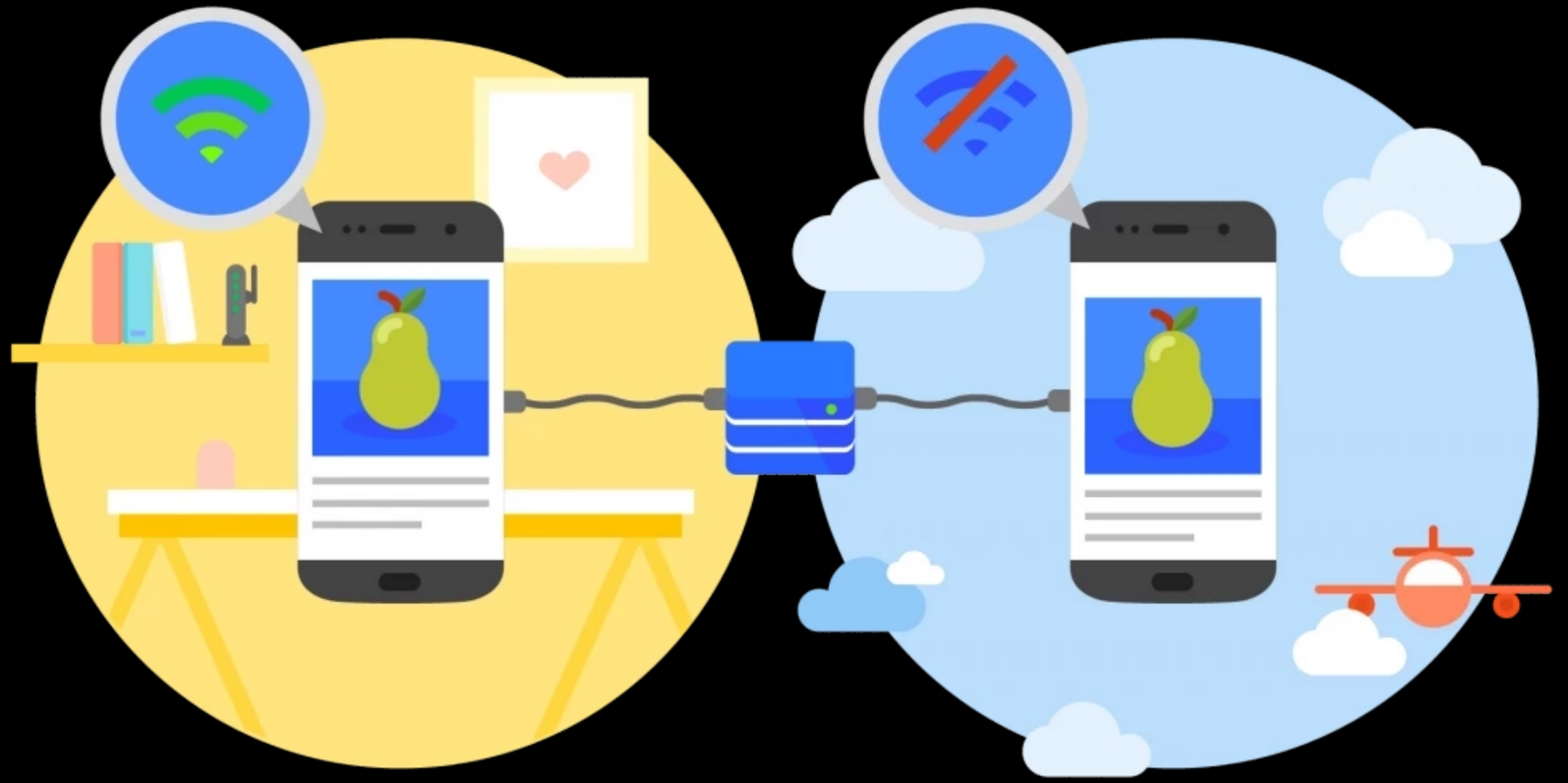
Track progress

```
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

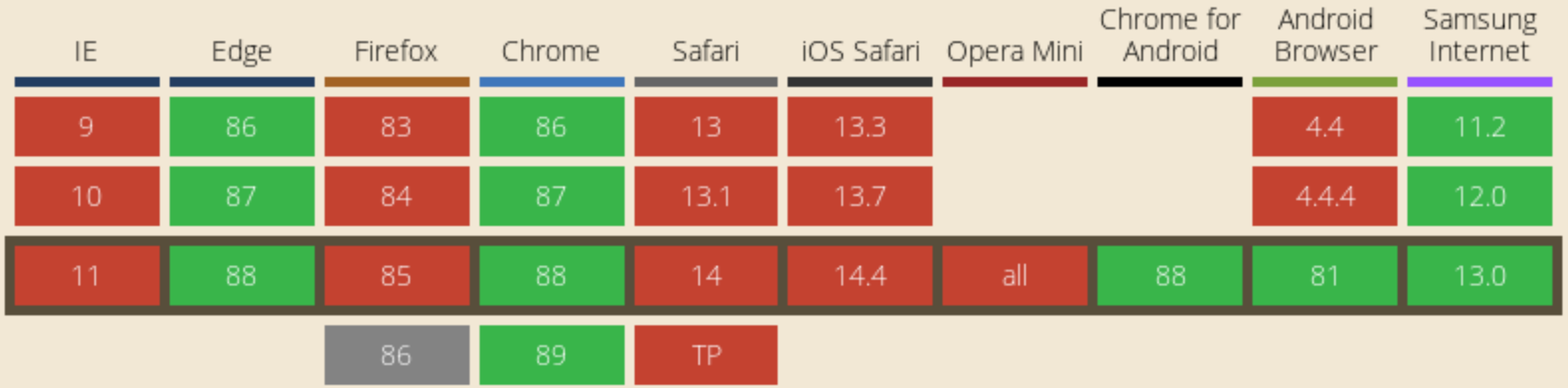
```
self.addEventListener('periodicsync', function(event) {


});
```

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.



✓ ✗ ? Partial Support

Global: 72.7% + 0% = **72.70%**

BackgroundFetchEvent API

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android
			4-73		10-60						
6-10	12-87	2-84	74-87	3.1-13.1	62-72	3.2-13.7		2.1-4.4.4	12-12.1		
11	88	85	88	14	73	14.4	all	81	59	88	85
		86-87	89-91	TP							

PeriodicSyncEvent API

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android
	12-79		4-79		10-66						
6-10	80-87	2-84	80-87	3.1-13.1	67-72	3.2-13.7		2.1-4.4.4	12-12.1		
11	88	85	88	14	73	14.4	all	81	59	88	85
		86-87	89-91	TP							

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

(Some) APIs

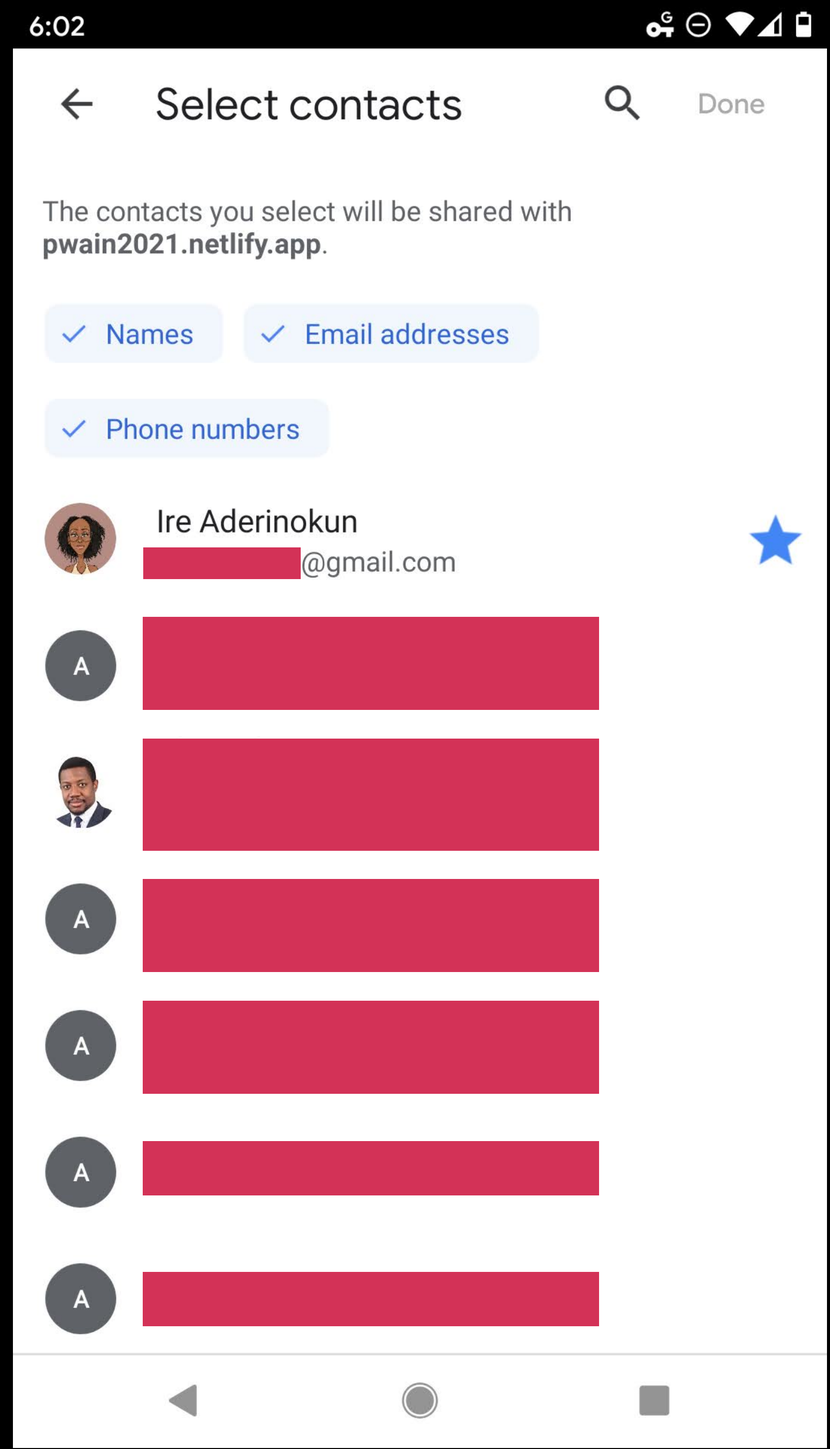
Contact Picker API

Web Share API & Web Share Target API

File System Access API

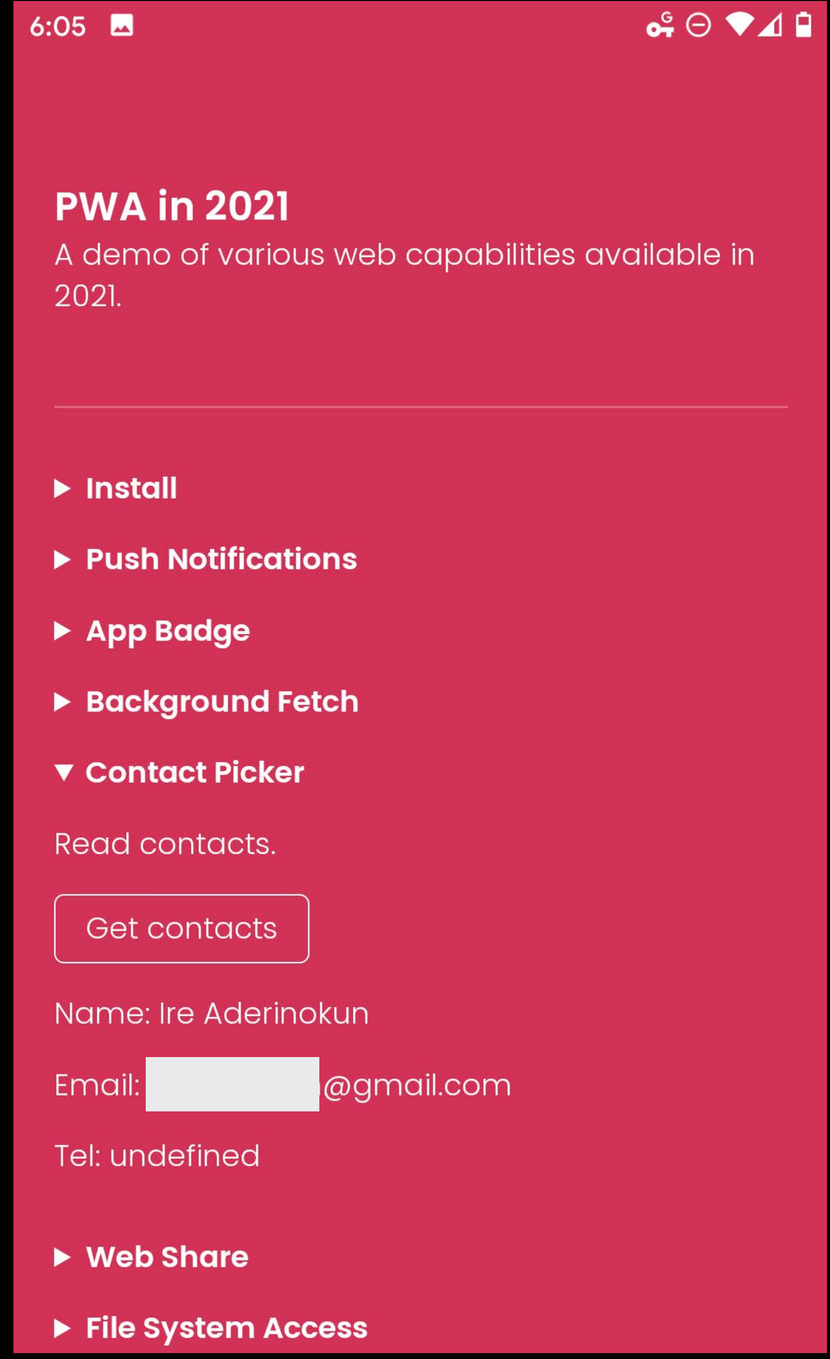
Contact Picker API

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


}
```



Contact Picker API

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

    const contacts = await navigator.contacts.select(

        ['name', 'email', ‘tel', 'icon'],

        {multiple: true}

    );

}
```



Navigator API: contacts

Usage % of all users ?

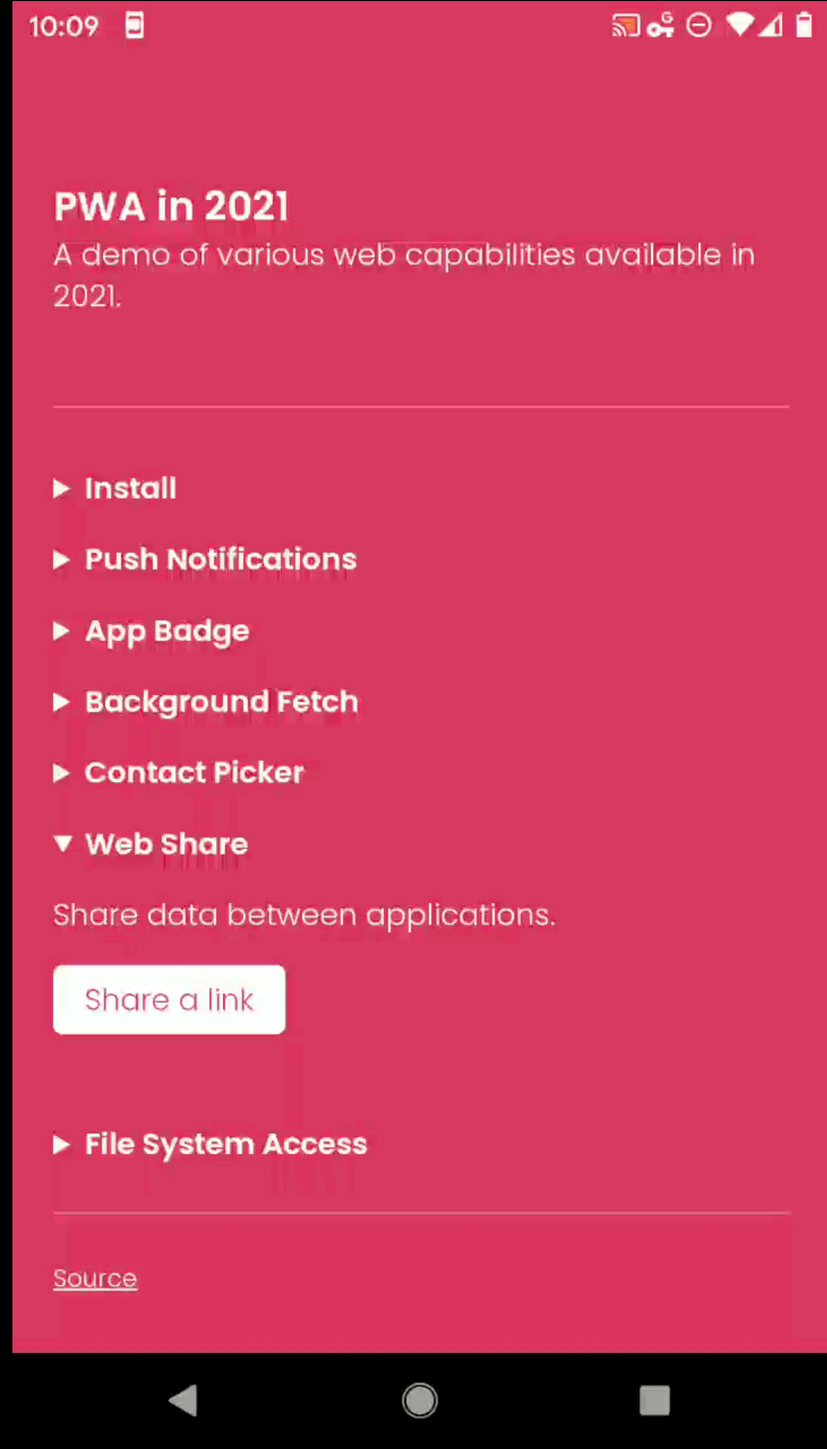
Global 39.69%

Current aligned | Usage relative | Date relative | Filtered | All

iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
3.2-13.7		2.1-4.4.4	12-12.1				4-12.0			
14.4	all	81	59	88	85	12.12	13.0	10.4	7.12	2.5

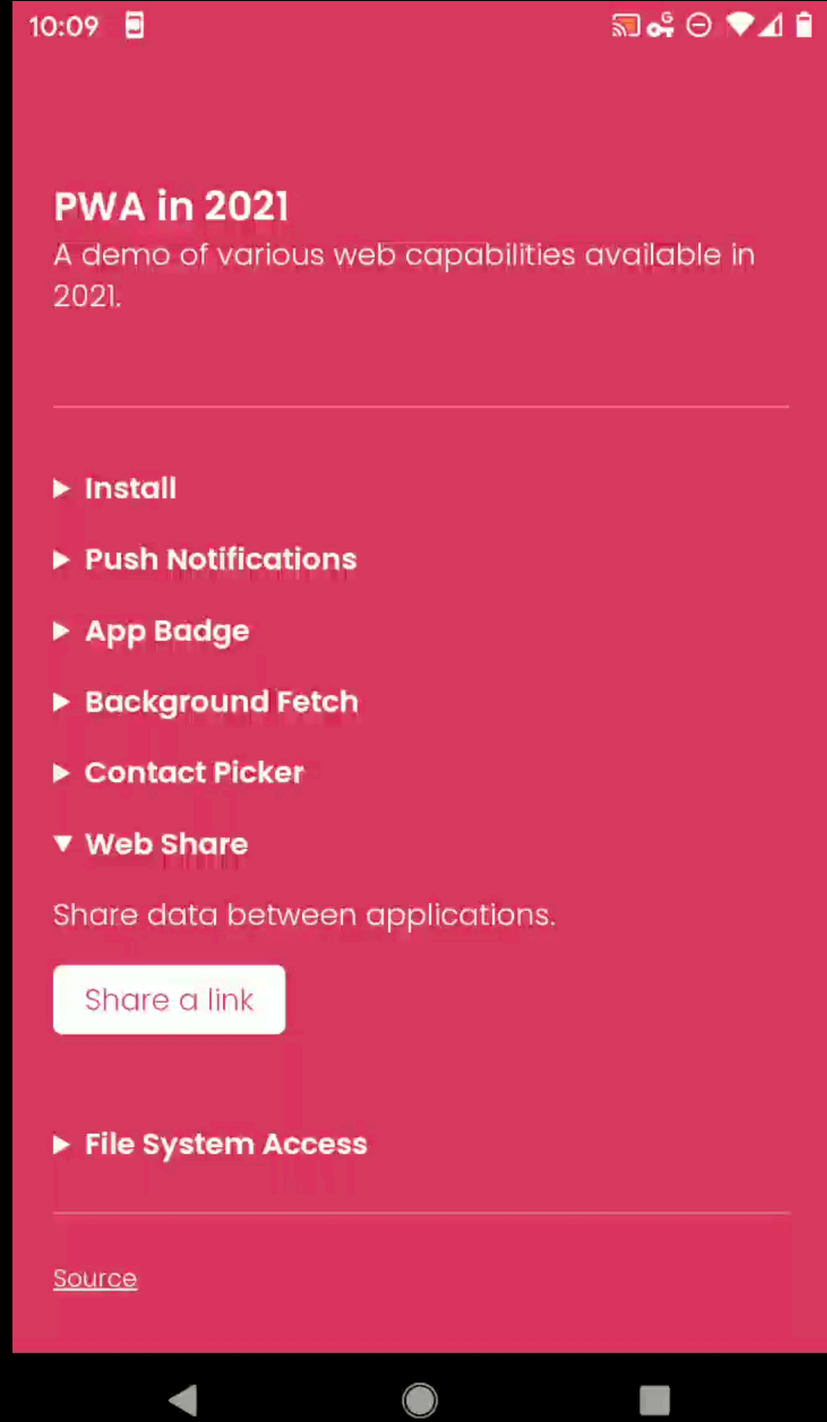
Web Share API

```
if (navigator.share) {


}
```



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',
    });
}
```



Web Share Target API

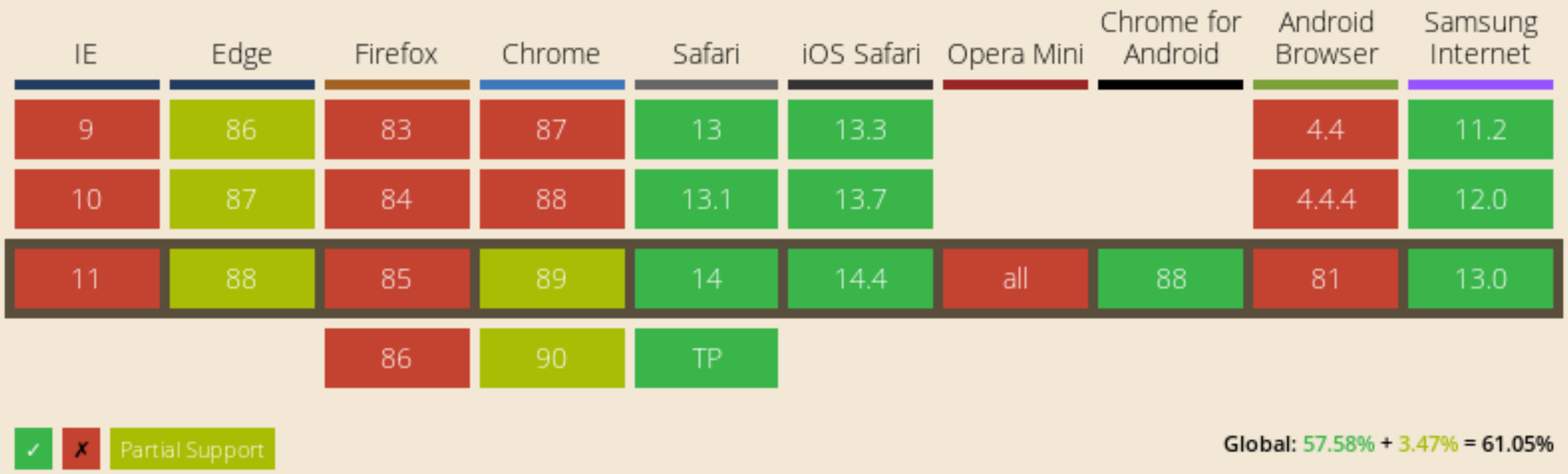
```
"share_target": {

    "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



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

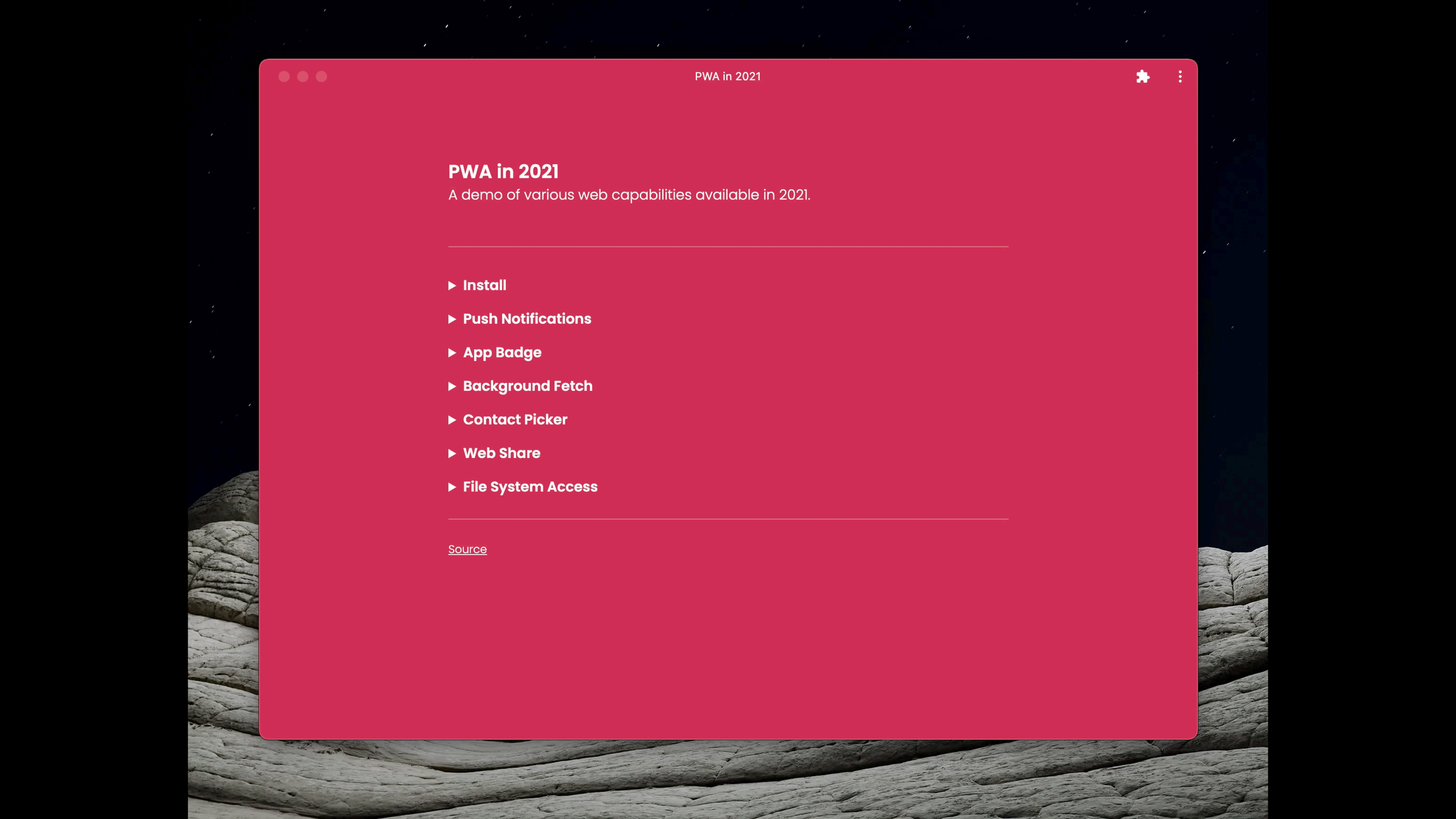
```
document.getElementById('save-file').addEventListener('click', async () => {

    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();

});
```

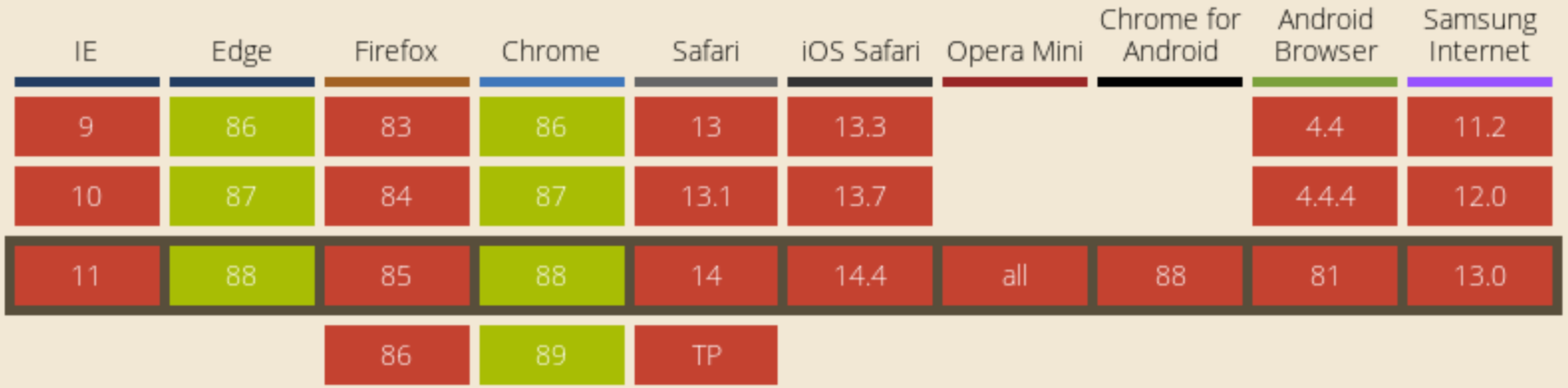
Save to a local file

```
document.getElementById('save-file').addEventListener('click', async () => {

    const writable = await fileHandle.createWritable();

    await writable.write(/* new contents */);

    await writable.close();

});
```


PWA in 2021
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

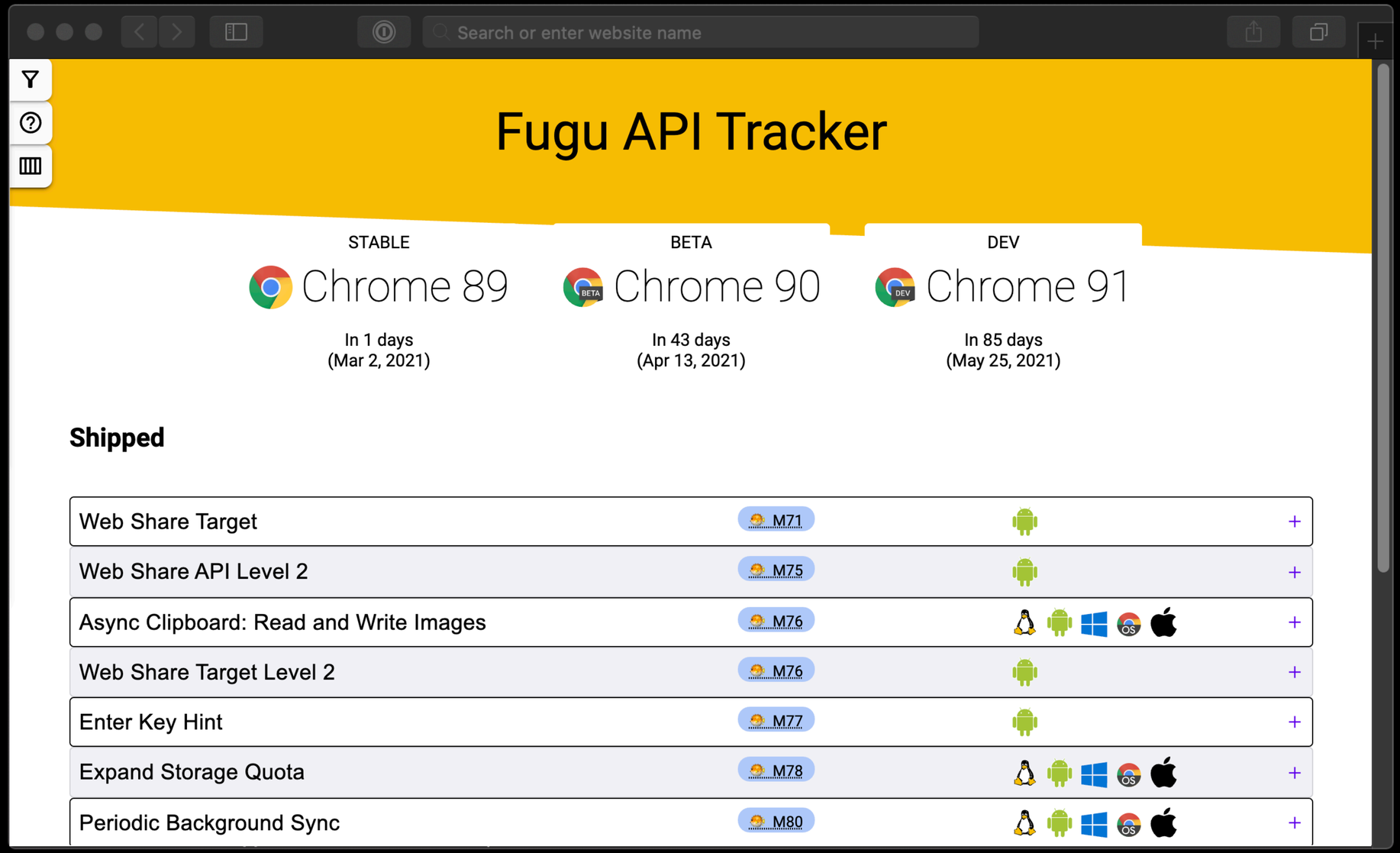
File System Access API

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



https://fugu-tracker.web.app/

Capable 🙂

Reliable

Installable

Capable 🙂

Reliable 🤔

Installable

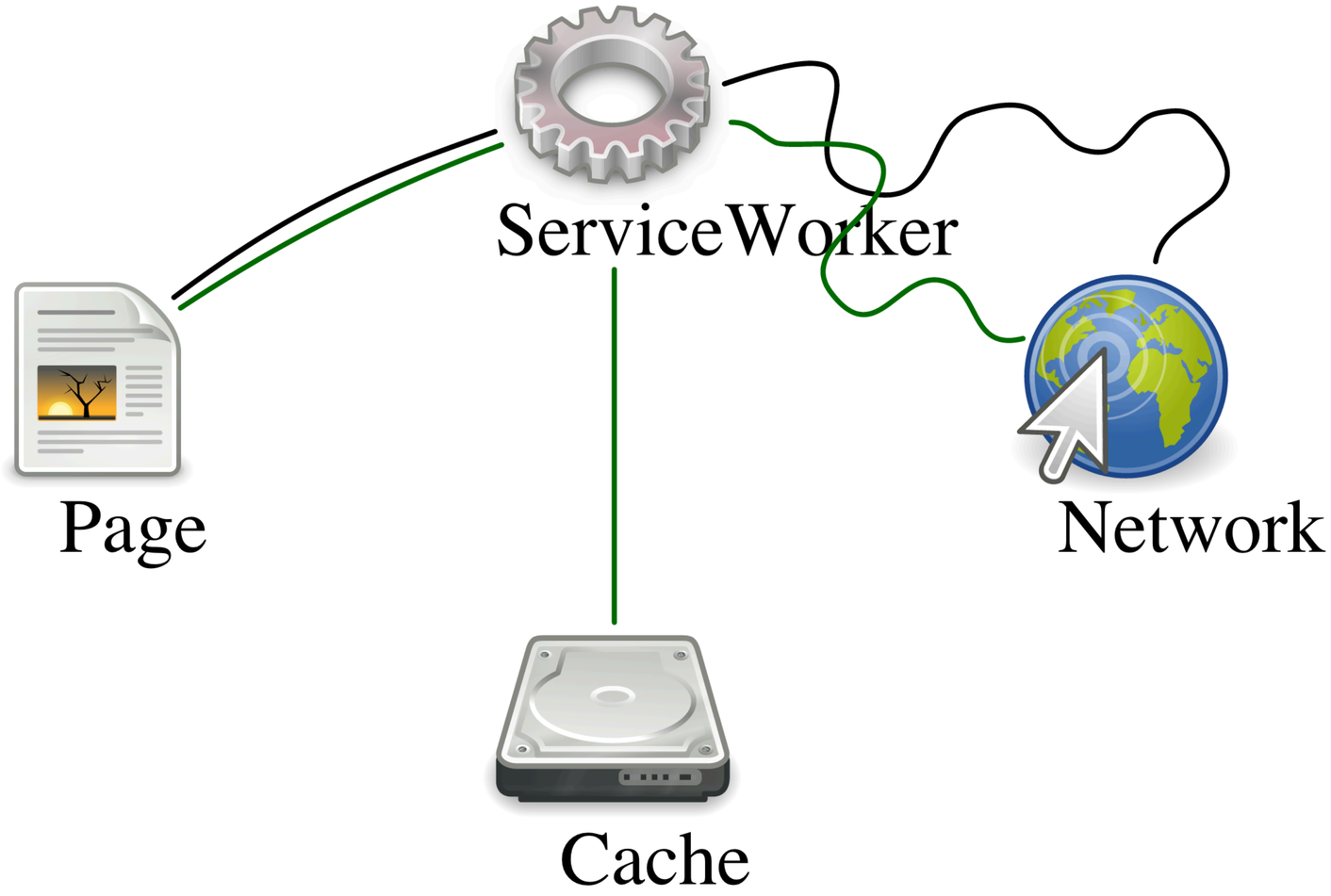


Image from https://pjagajitprusty.github.io/PWD/

Service Workers

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

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	86	83	87	13	13.3			4.4	11.2
10	87	84	88	13.1	13.7			4.4.4	12.0
11	88	85	89	14	14.4	all	88	81	13.0
		86	90	TP					

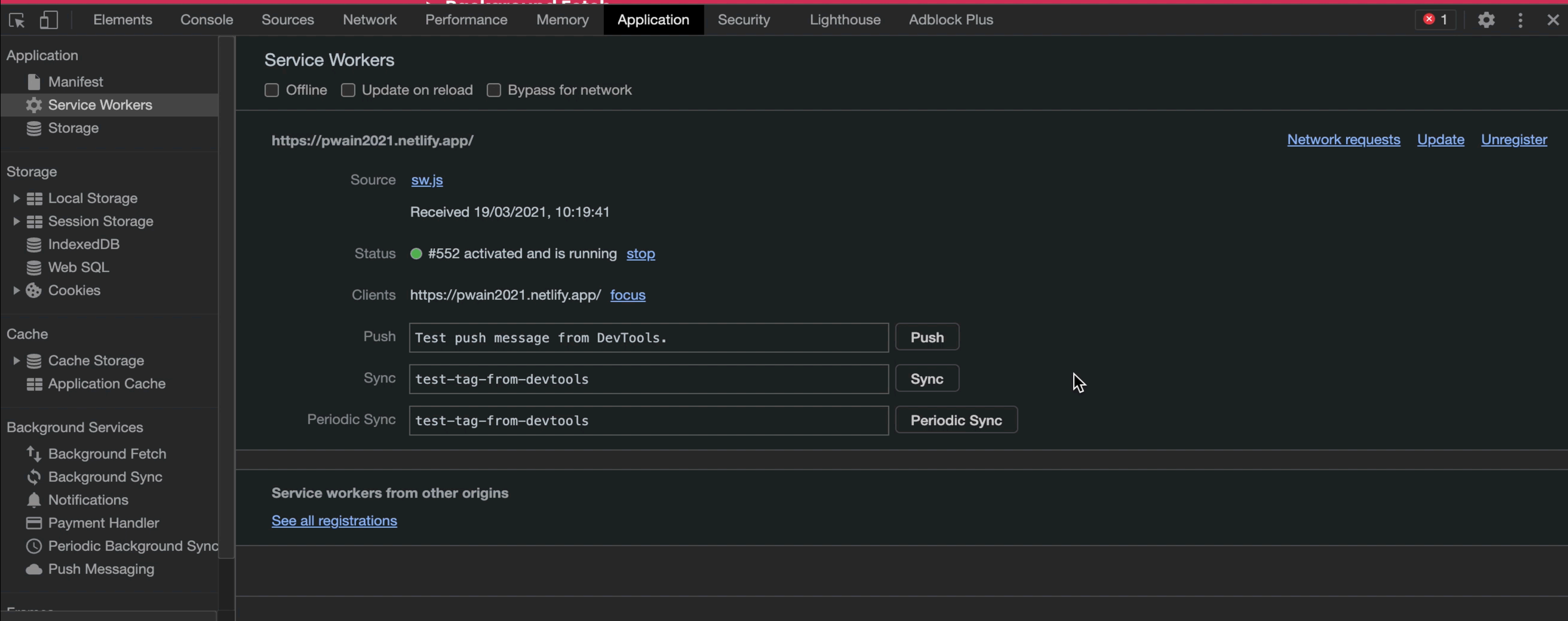
✓ ✗ Partial Support

Global: 94.87% + 0.09% = **94.96%**

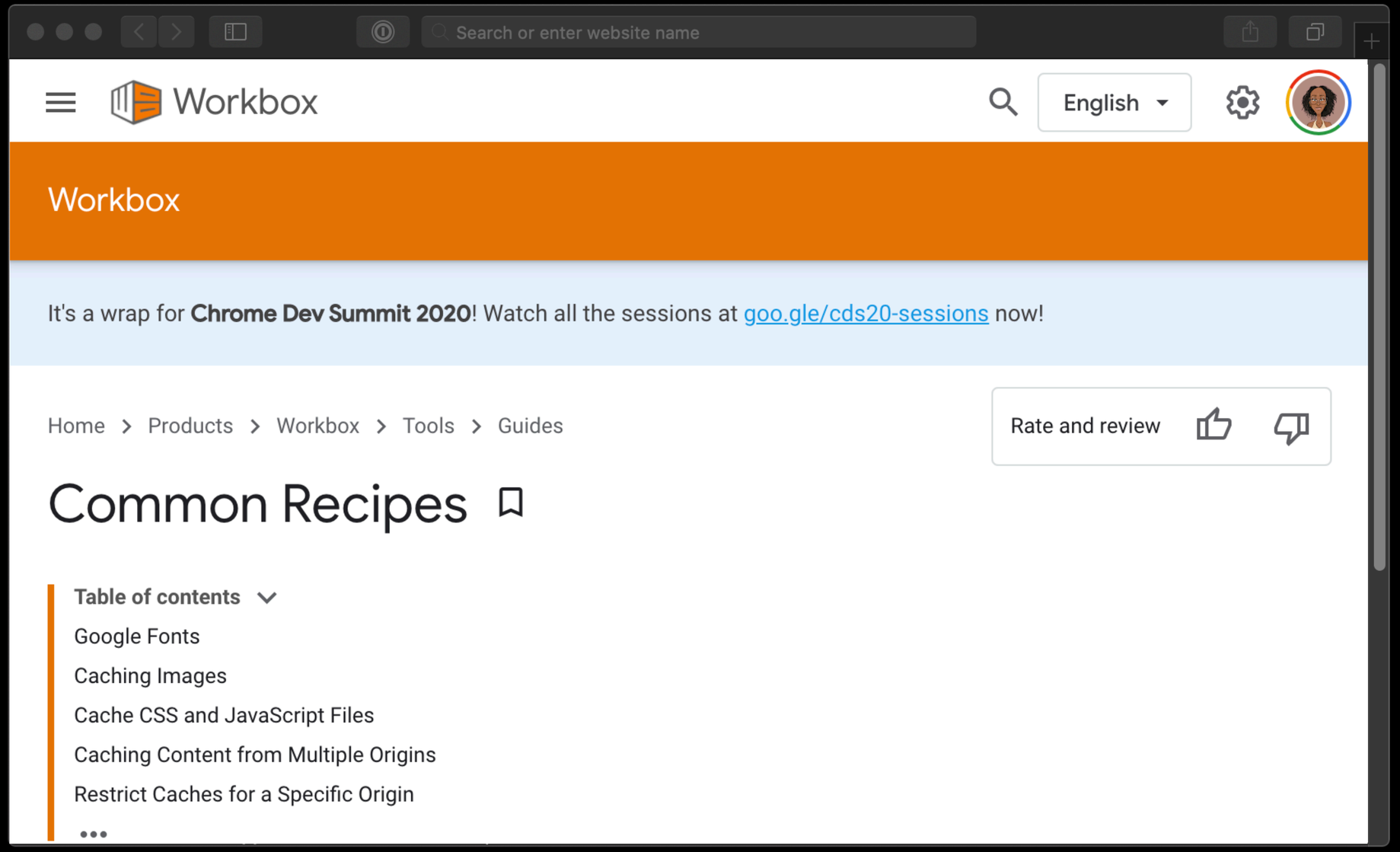
PWA in 2021

A demo of various web capabilities available in 2021.

- ▶ **Install**
- ▶ **Push Notifications**
- ▶ **App Badge**




Workbox



https://developers.google.com/web/tools/workbox/guides/common-recipes

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

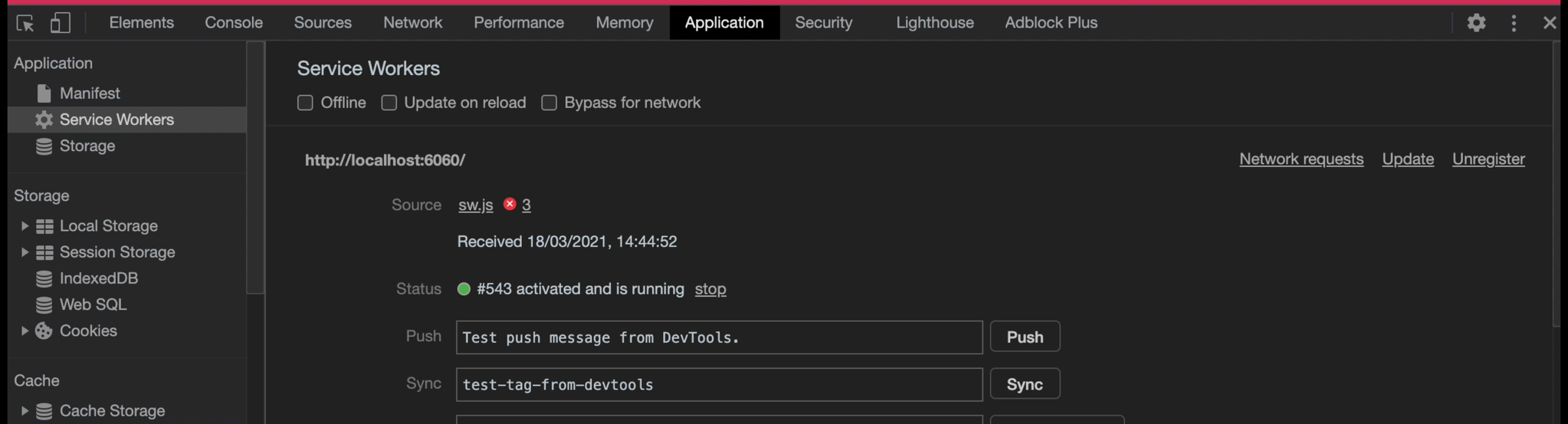
```
const networkOnly = new NetworkOnly();

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) );
```

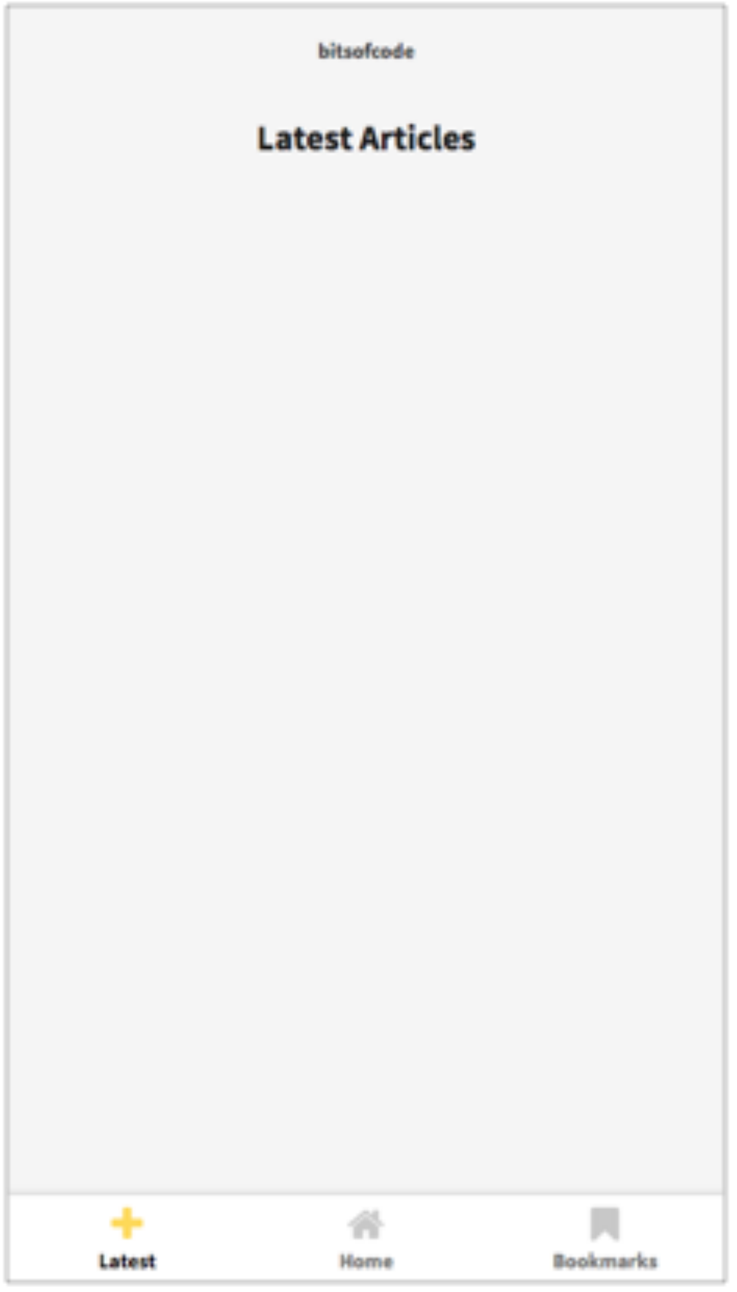
PWA in 2021

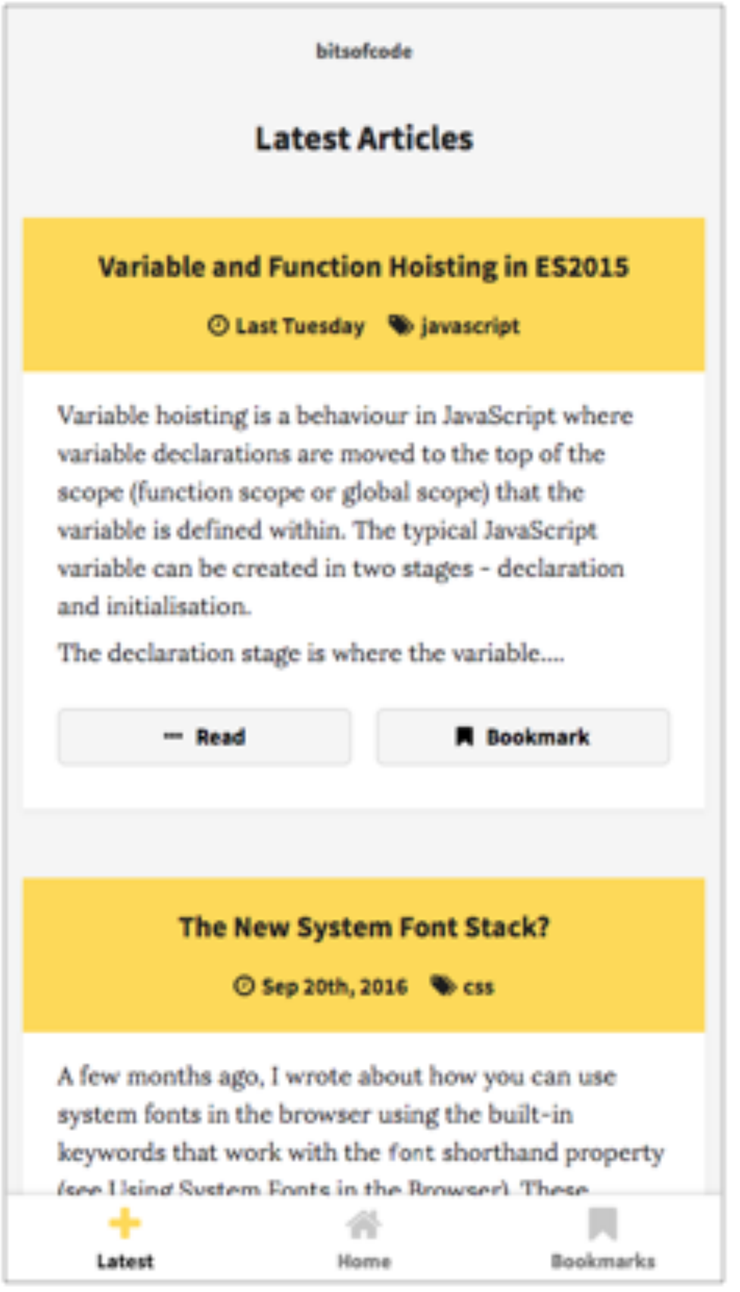
A demo of various web capabilities available in 2021.

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

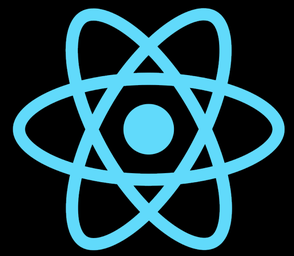
Source

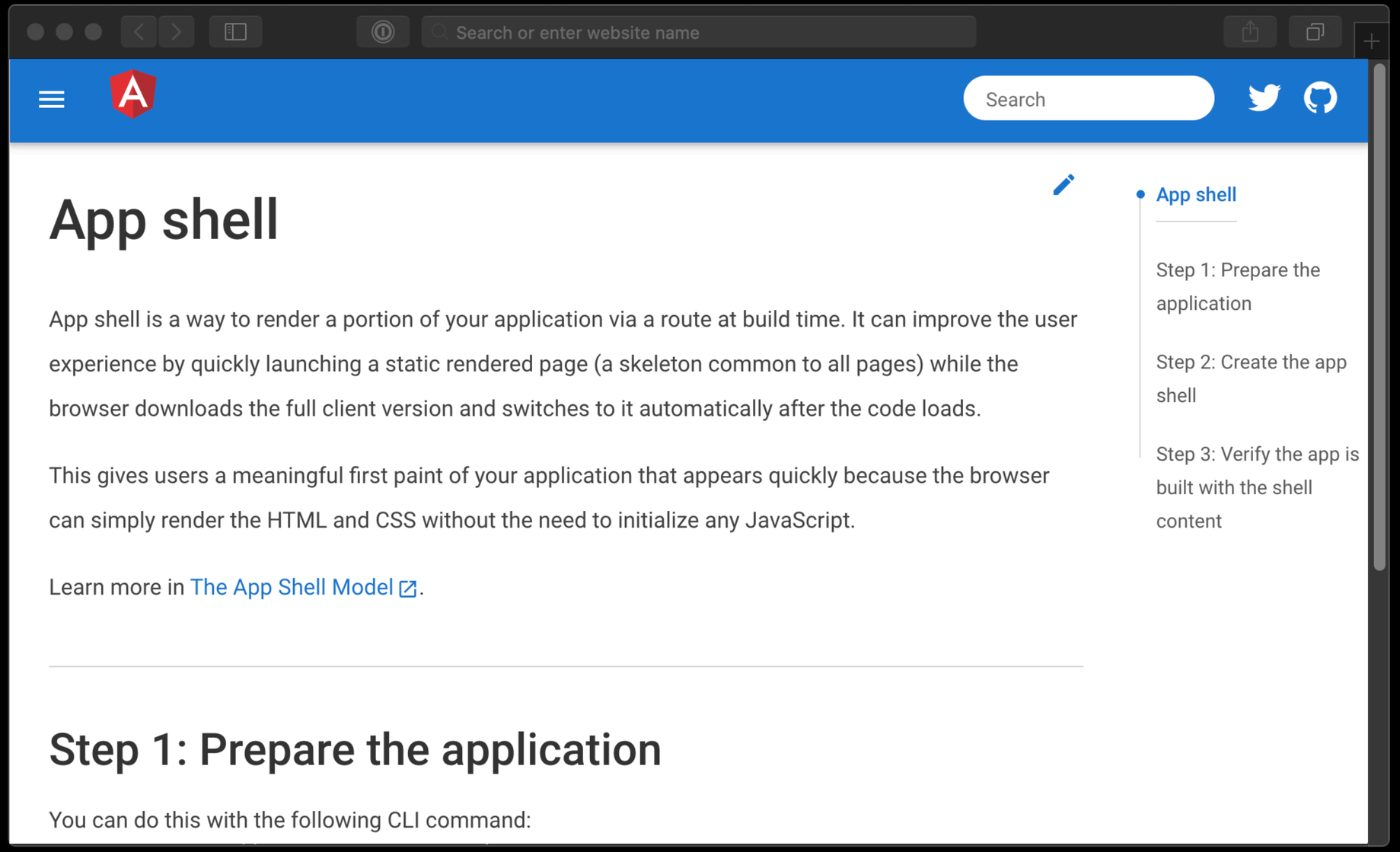



bitsofcode
Latest Articles
Latest
Home
Bookmarks

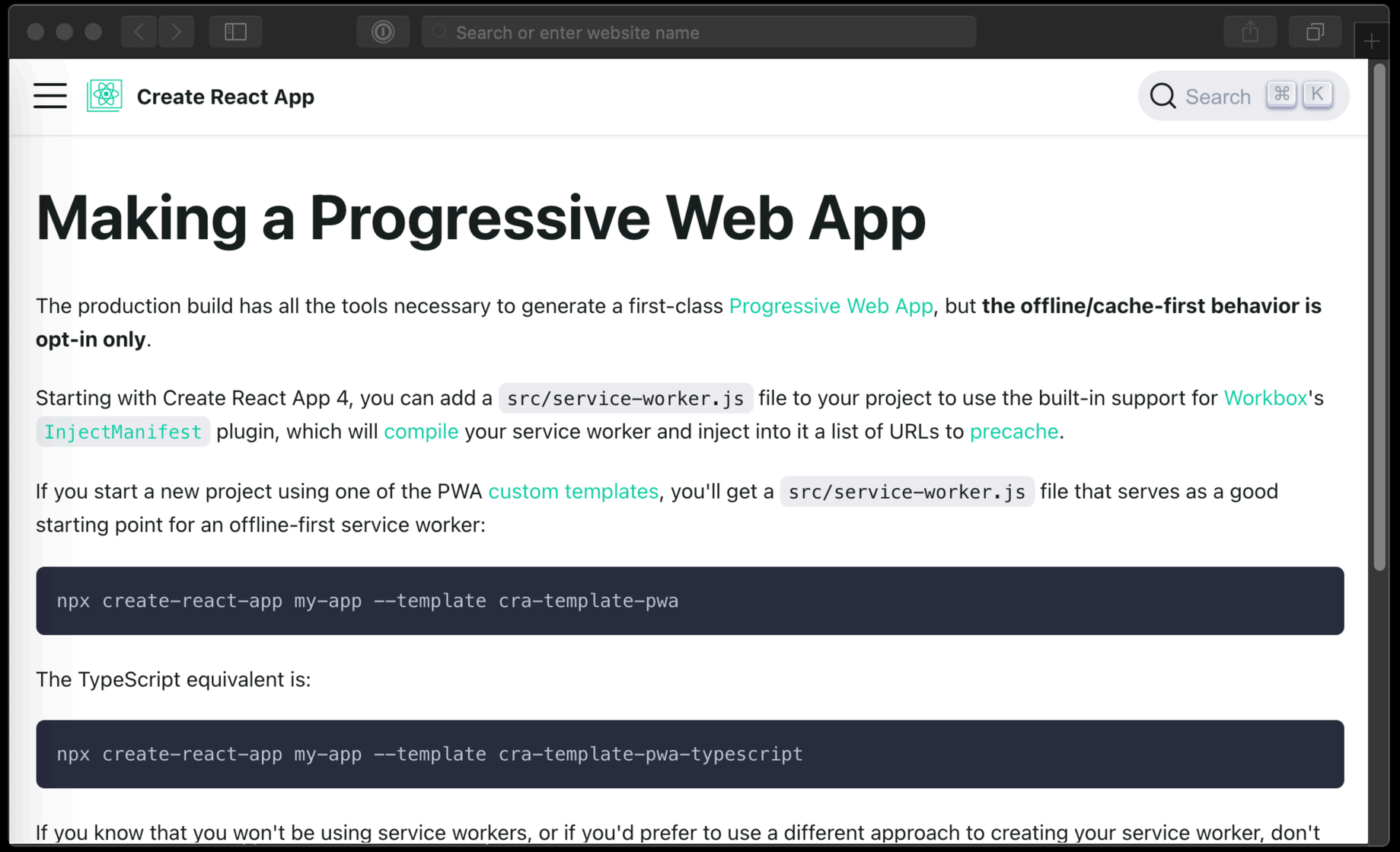

bitsofcode
Latest Articles
Variable and Function Hoisting in ES2015
Last Tuesday
javascript
Variable hoisting is a behaviour in JavaScript where variable declarations are moved to the top of the scope (function scope or global scope) that the variable is defined within. The typical JavaScript variable 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
Latest
Home
Bookmarks


PWA





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



https://create-react-app.dev/docs/making-a-progressive-web-app/

https://cli.vuejs.org/core-plugins/pwa.html

Capable 🙂

Reliable 😀

Installable

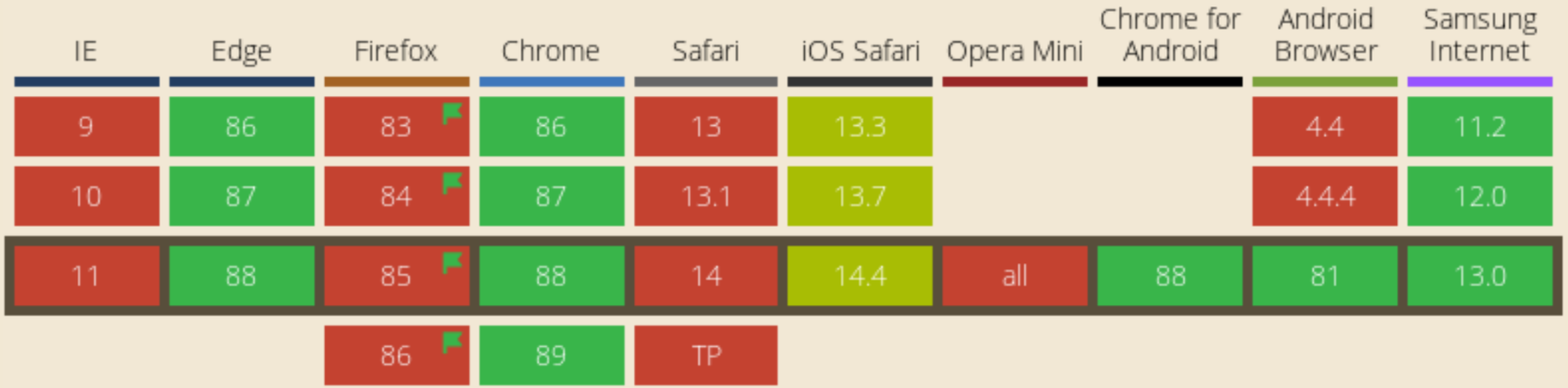
Capable 🙂

Reliable 😀

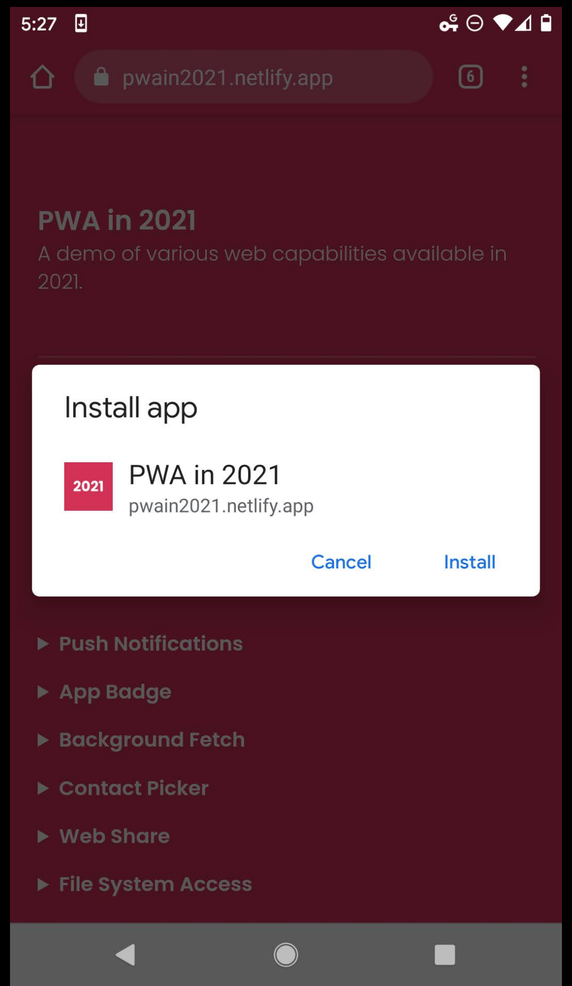
Installable 🤔

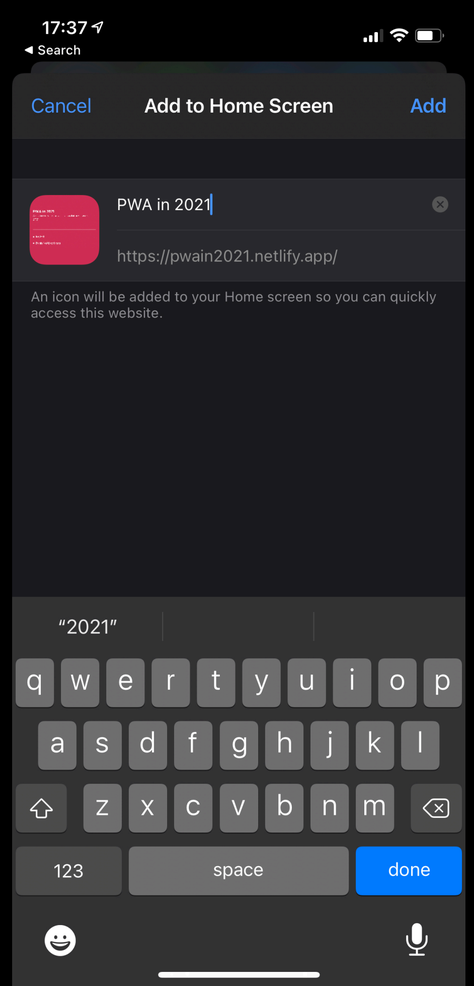
Add to home screen (A2HS)

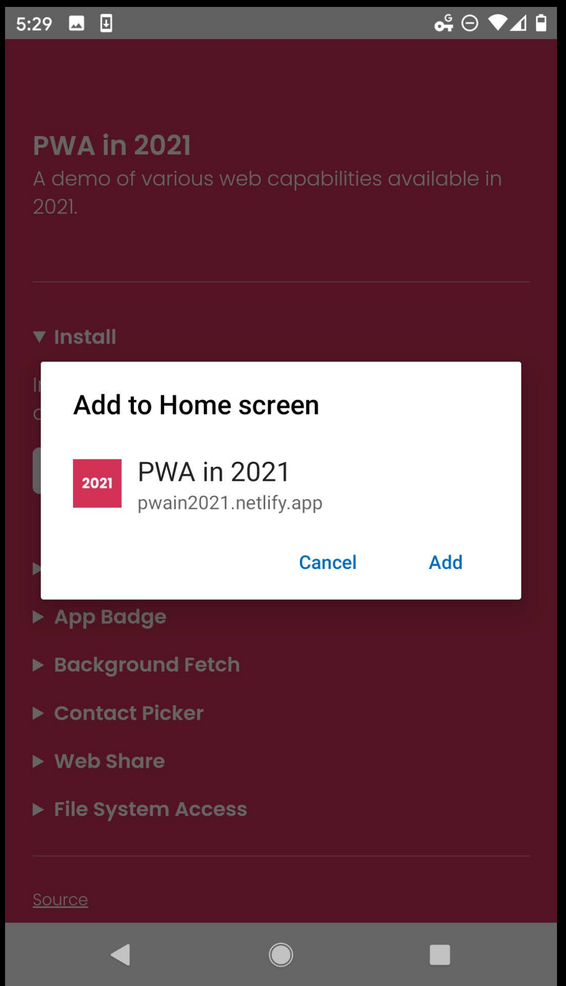
The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load it's assets through a [Service Worker] (https://caniuse.com/#feat=serviceworkers).



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
Search
Cancel
Add to Home Screen
Add
PWA in 2021
https://pwain2021.netlify.app/
An icon will be added to your Home screen so you can quickly access this website.
"2021"
123
space
done

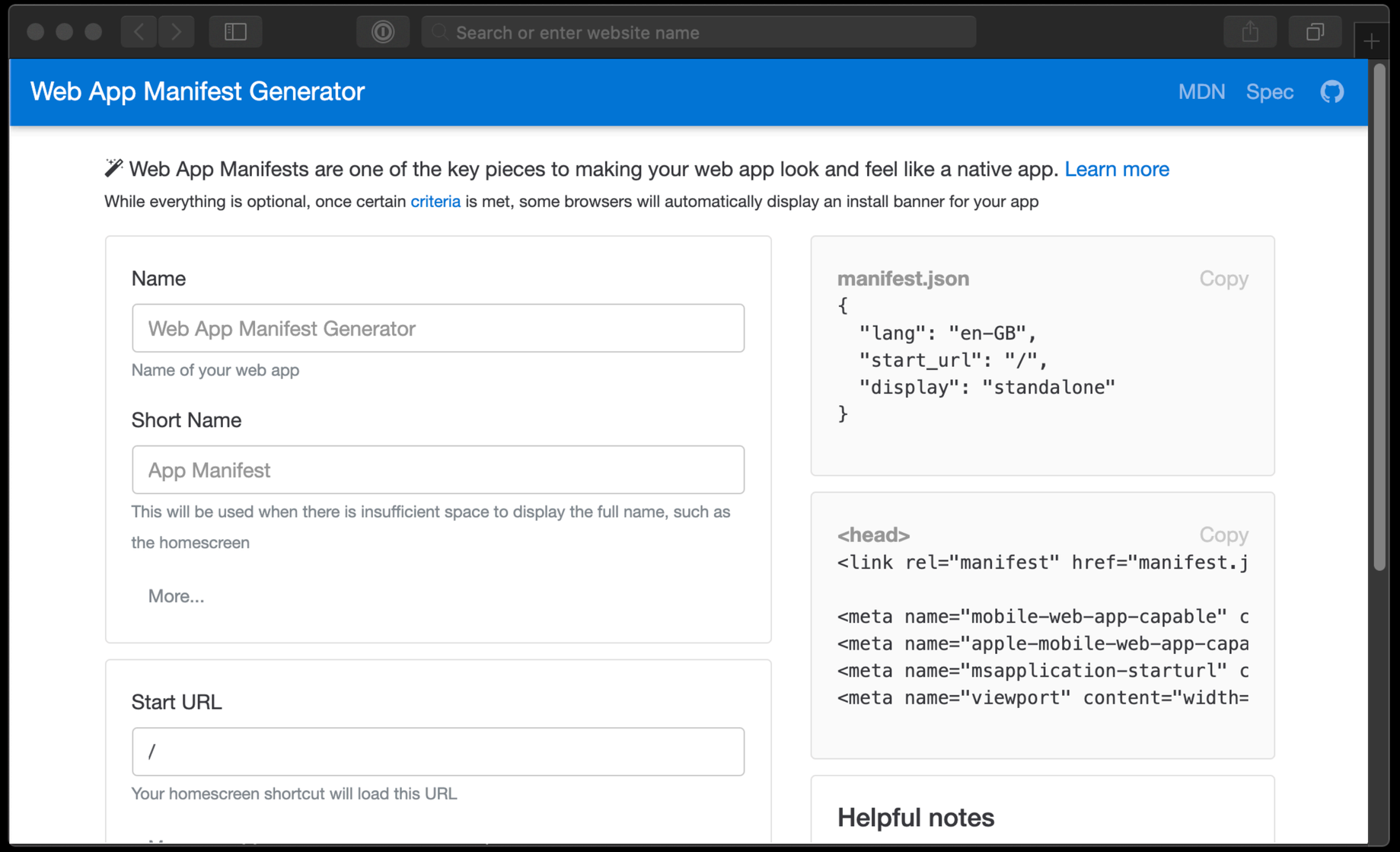

5:29
PWA in 2021
A demo of various web capabilities available in 2021.
Install
Add to Home screen
2021
PWA in 2021
pwain2021.netlify.app
Cancel
Add
App Badge
Background Fetch
Contact Picker
Web Share
File System Access
Source

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": "#ffff",

    "theme_color": "#D33257",

    "start_url": "/",

}
```



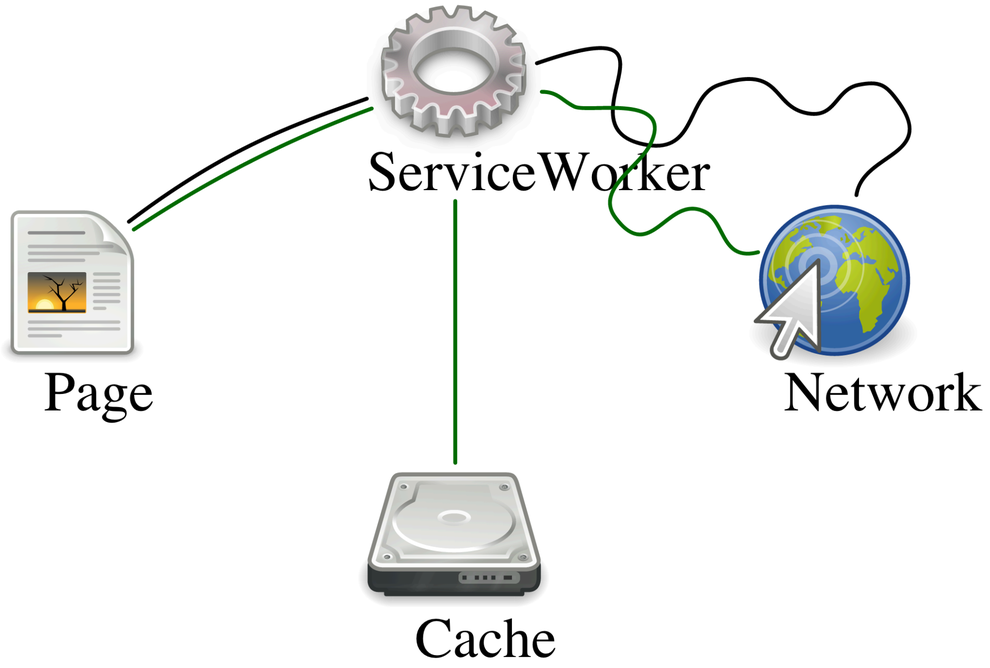
https://tomitm.github.io/appmanifest/

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

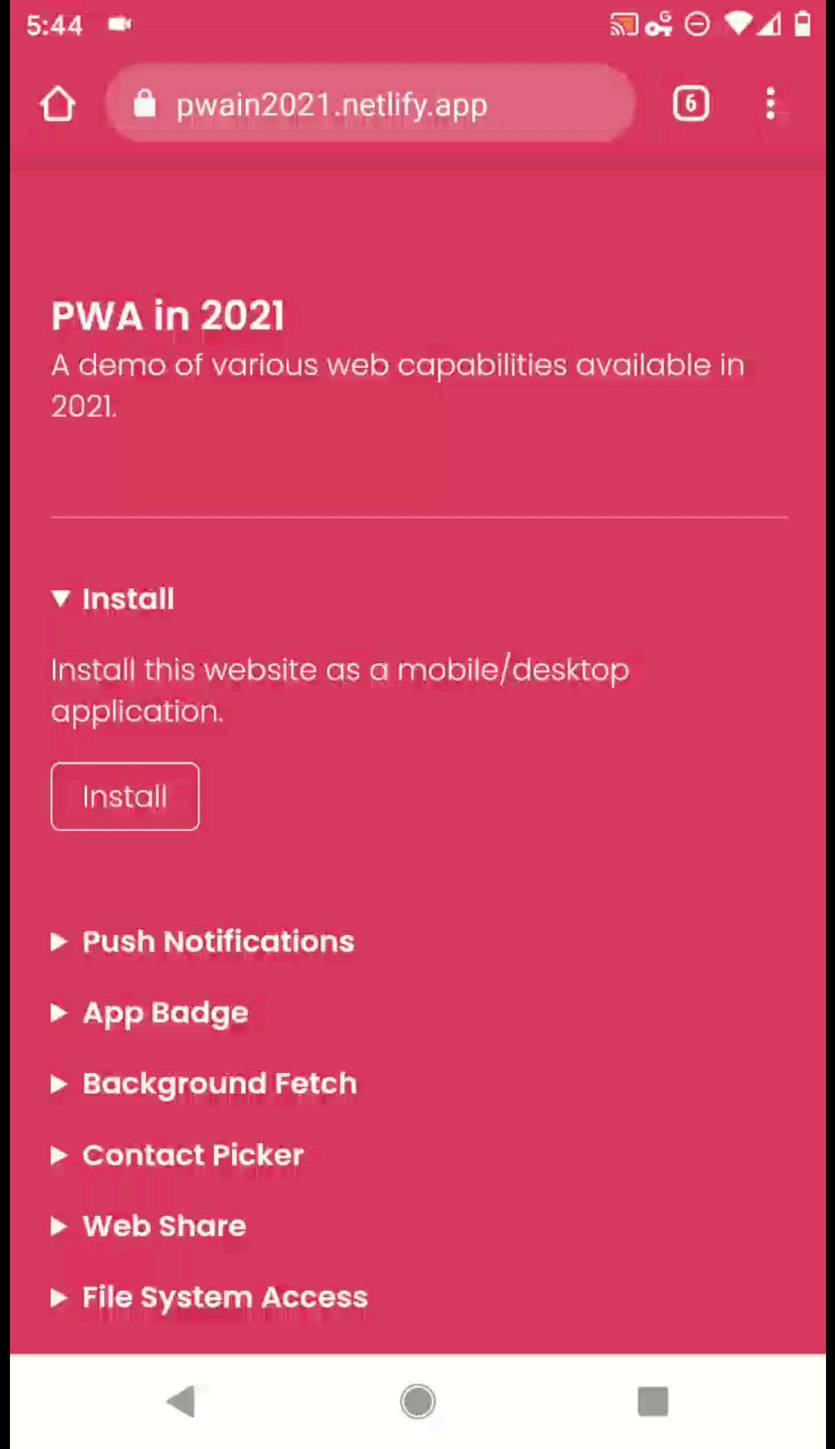
```
customInstallButton.addEventListener('click', async (e) => {

    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
pwain2021.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

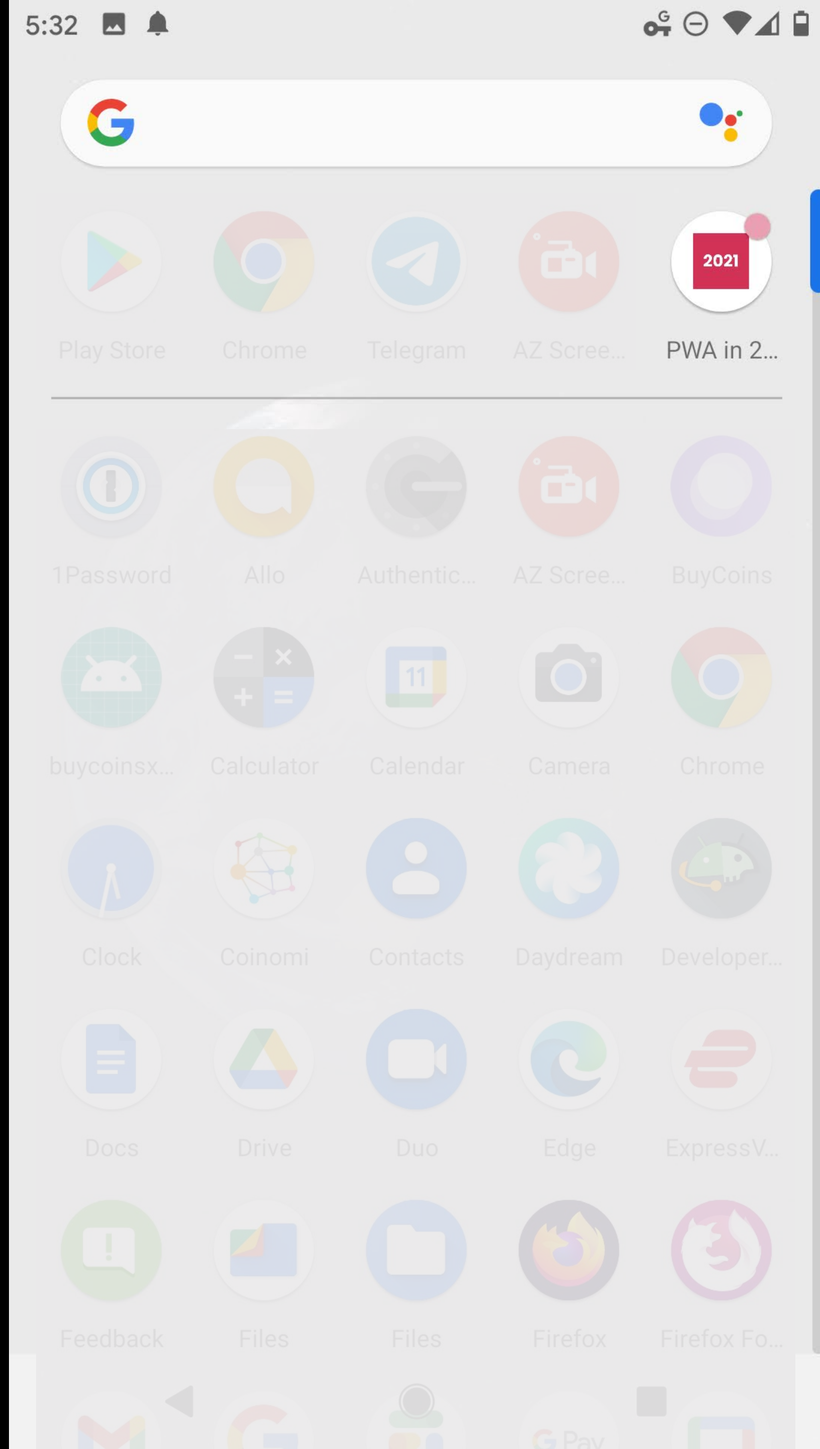
WebAPK

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

Benefits of WebAPK

Benefits of WebAPK

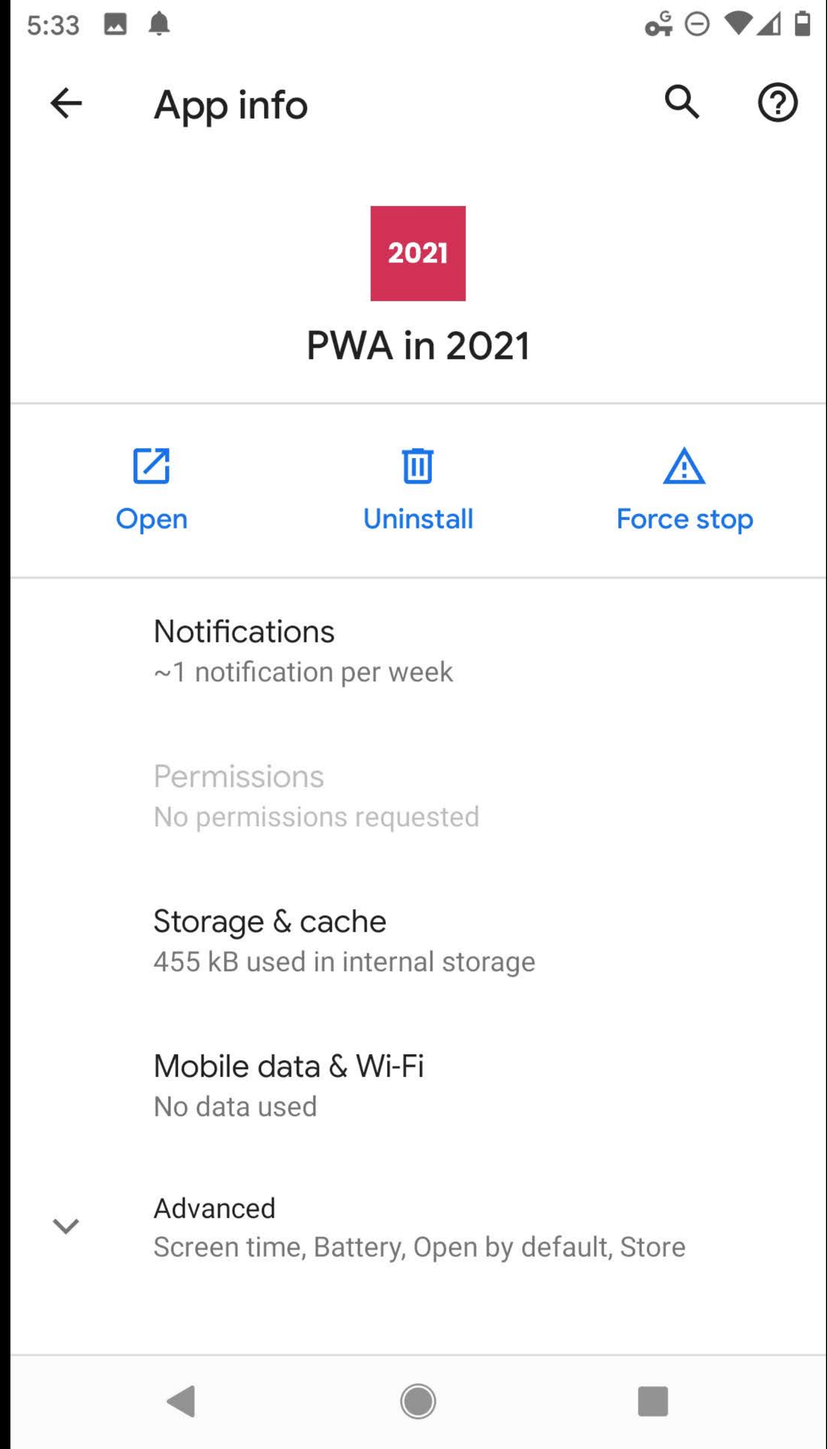
✅ In the app drawer



Benefits of WebAPK

✅ In the app drawer

✅ In app settings

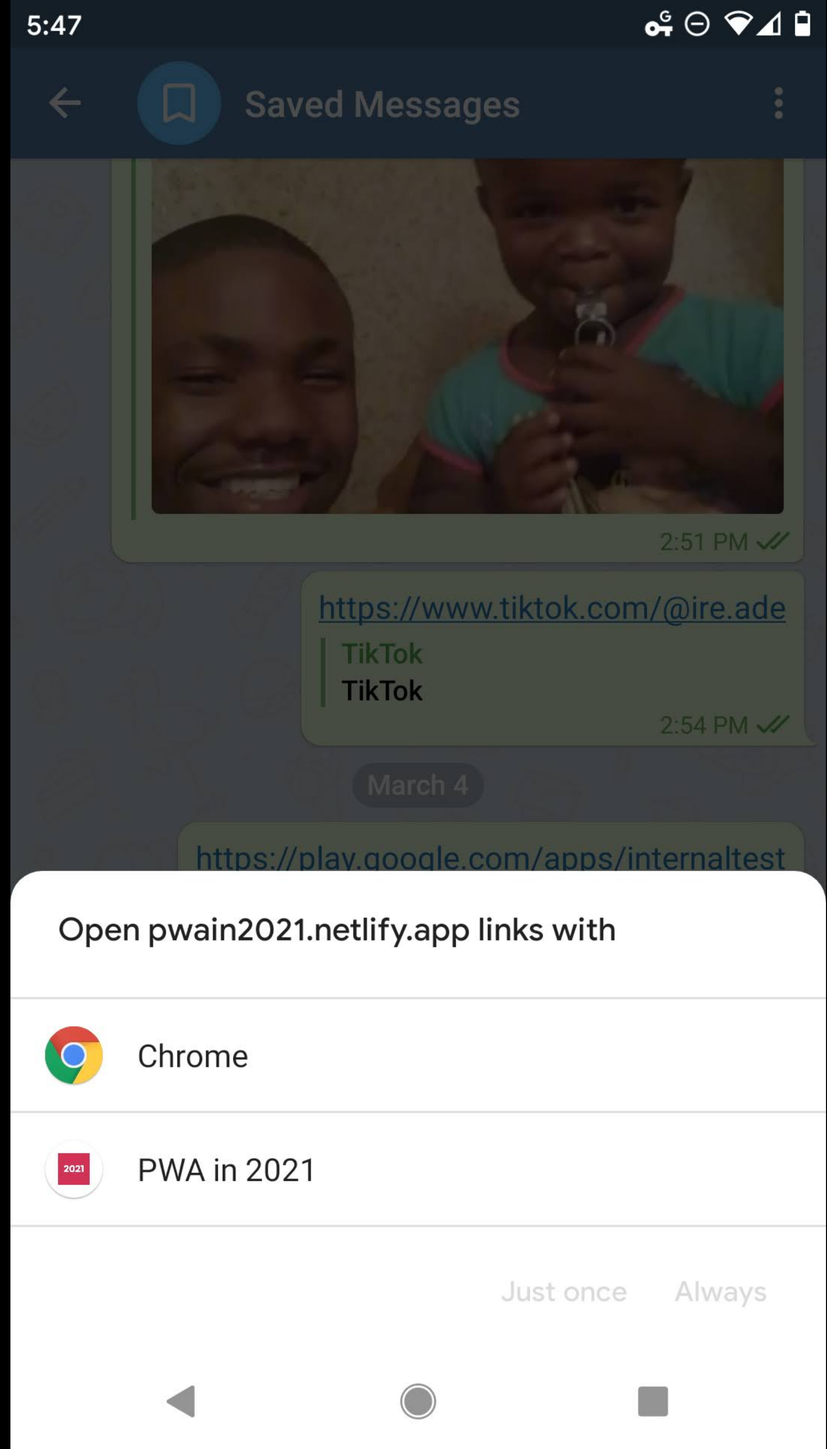


Benefits of WebAPK

✅ In the app drawer

✅ In app settings

✅ Includes intent filters



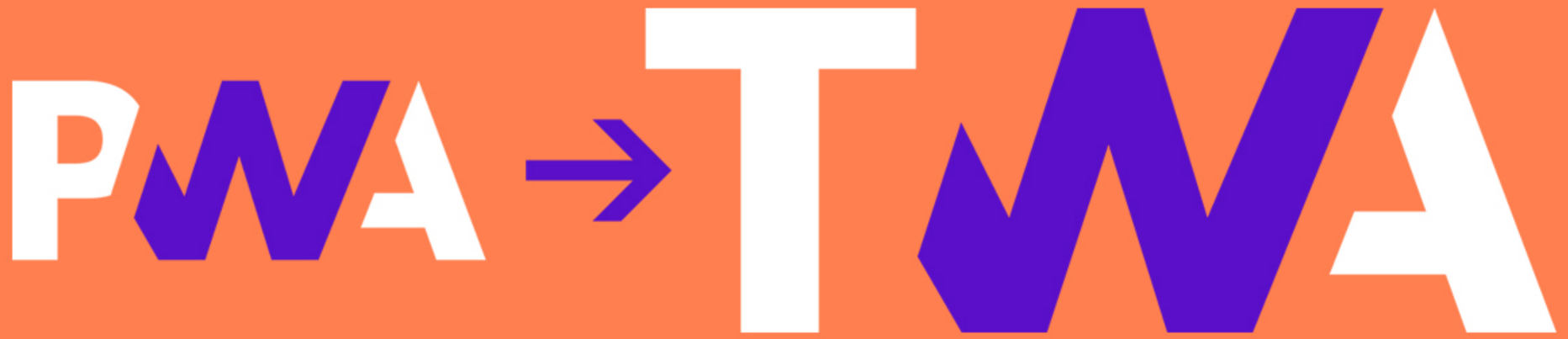


Image from https://www.ateamindia.com/what-are-pwa-and-twa/

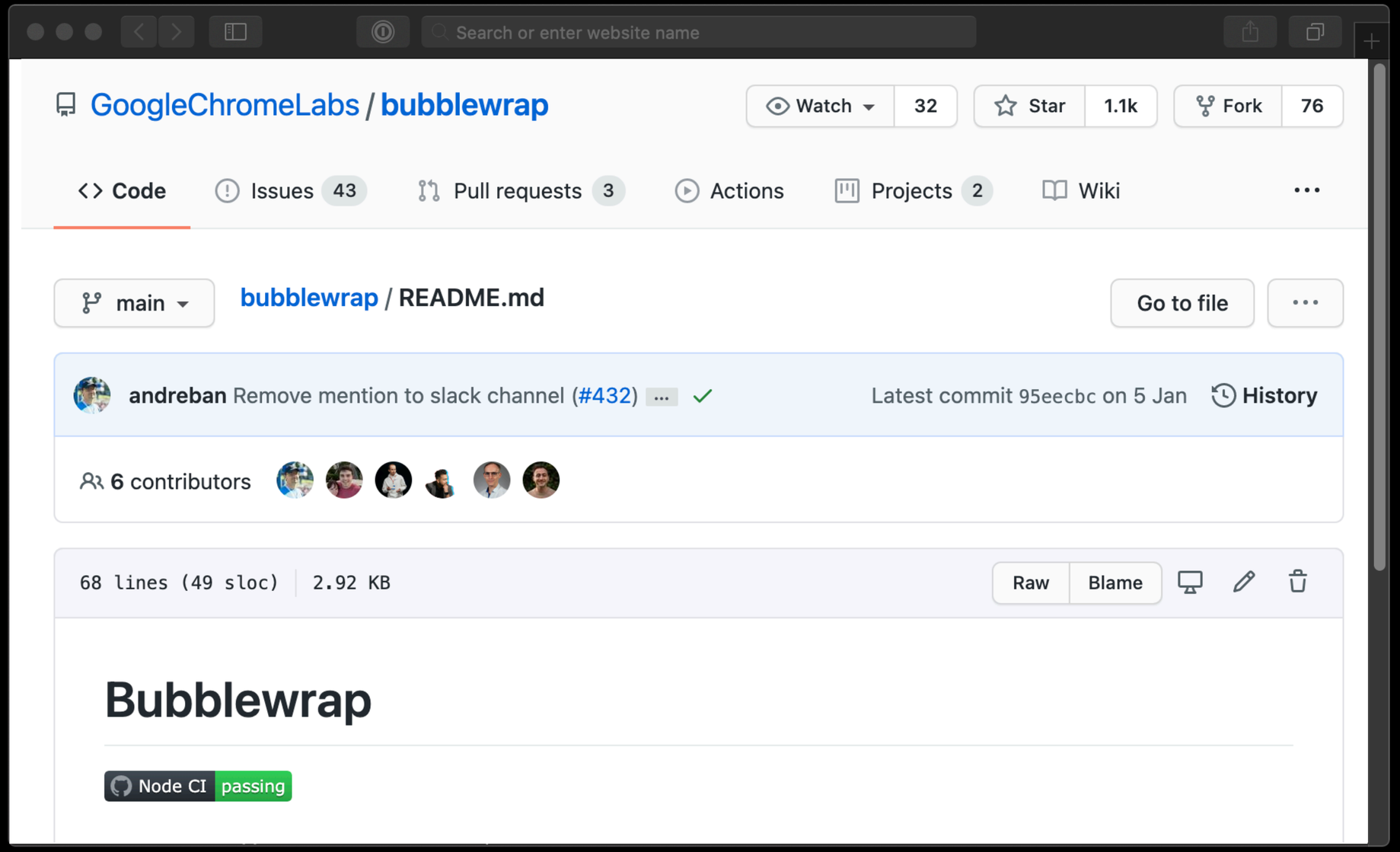
Trusted Web Activity is a way to open your PWA from your Android app using a protocol based on Custom Tabs

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


PWA


PWA


PWA



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

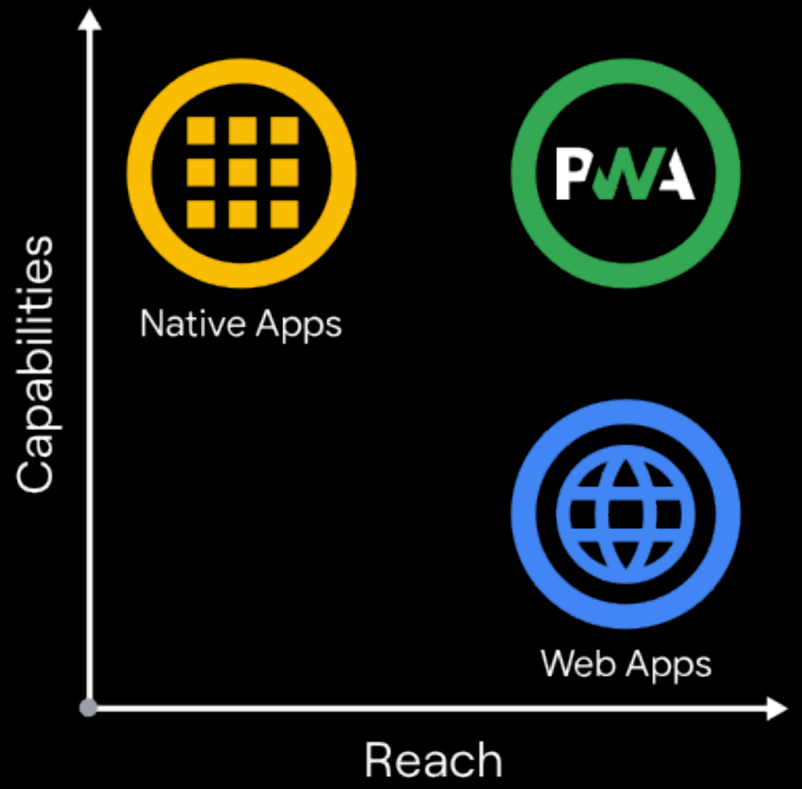
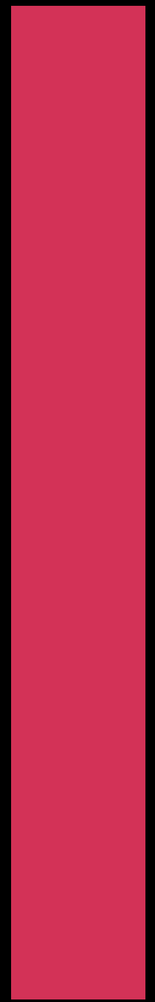


Image from https://web.dev/what-are-pwas

Web on Everything!



Thank you!

Ire Aderinokun 🇳🇬

COO & VP Engineering of BuyCoins

Google Web Expert

ireaderinokun.com

bitsofco.de

@ireaderinokun