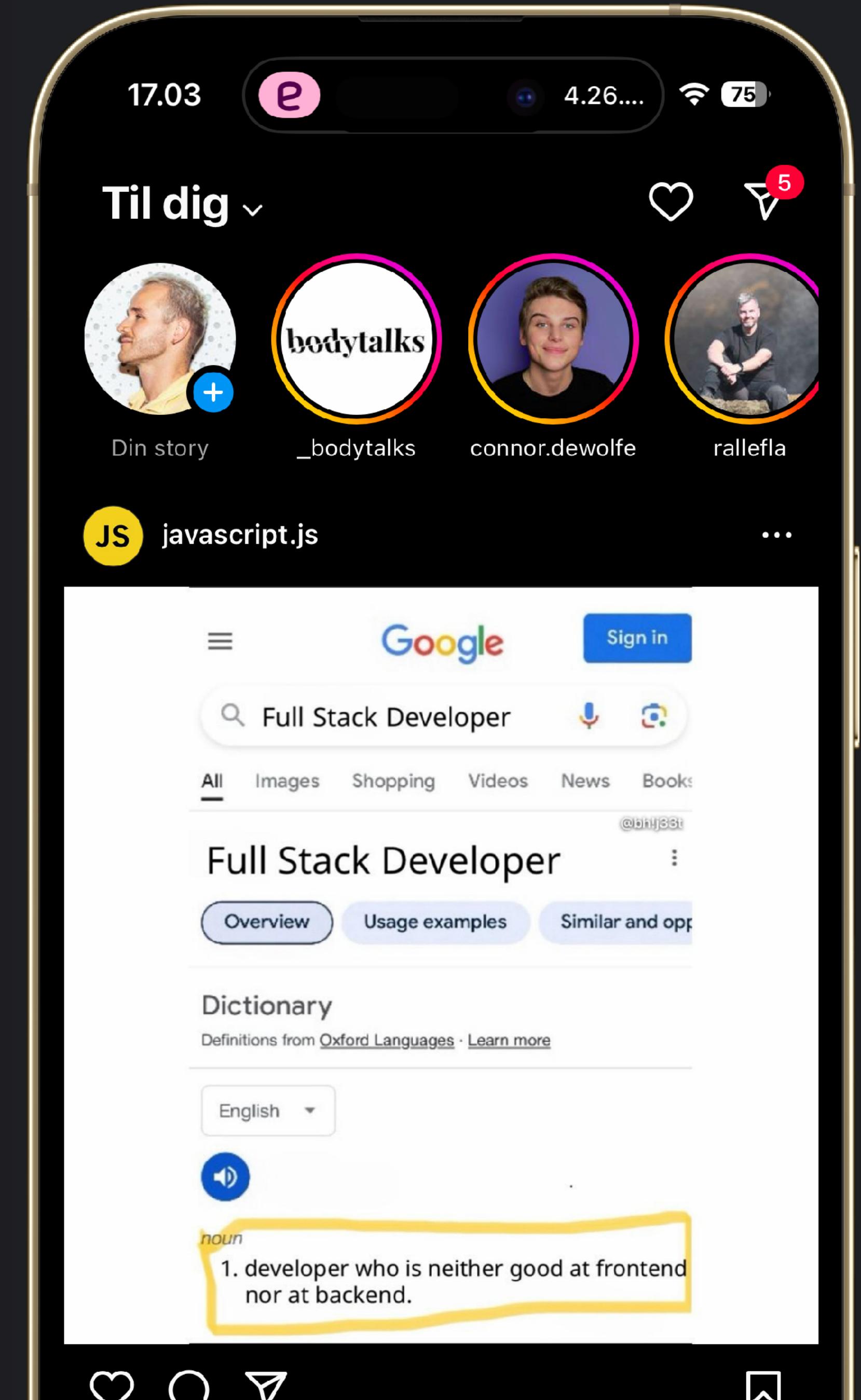


Er du

# Fullstack Developer?



PWA

Progressive Web App

# PWA

“[...] is basically just a website with some added features, which enable it to provide an app-like user-experience. “

<https://whatpwacando.today/info>

# PWA

“This means it can work practically just like a native iOS or Android app. It can be installed to the home screen of your mobile device, work offline and receive push notifications, among other things.”

<https://whatpwacando.today/info>

**100** *SECONDS OF* 

**PWA**

***AND BEYOND!***

<https://www.youtube.com/watch?v=sFsRyICQblw>



BouTime - BouMatic A/S & SA boutime.boumatic.dk/home

### BOUMATIC SAC DANMARK

**Begivenheder**

M	T	O	T	F	L	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Denne uge  
08/04/2024  
Langtidsrask kommer på besøg 14.MAJ 11.JUNI i tidsrummet kl. 12.00-13.00. Vi  
er at finde i kantinen i Skjern.  
07:48

08/04/2024  
Donorbussen holder på parkeringspladsen torsdag d. 11.04.2024  
08:22

**Fraværende**

I dag

- Birgitte Sillesen (BS) 05/04/2024 - 12/04/2024 FERIE
- Helle Holm (HHO) 10/04/2024 - 12/04/2024 FERIE

**Kalender**

Rediger begivenhed

Beskrivelse: Langtidsrask kommer på besøg 14.MAJ 11.JUNI

Dato: 08/04/2024 07:48

Startdato: 08/04/2024 07:48

Slutdato: 08/04/2024 07:48

Hele dagen:

Oprettet af: Laila Rewaldt Nielsen (LRN)  
Executive Secretary/HR Officer  
Management

**Hjem** **Kalender** **Fraværende**

boutime.boumatic.dk/home

### Min Side REDIGER

Rasmus Cederdorff (RACE)  
Web App Developer  
Andet

**Kontaktoplysninger**

Telefon: 53789600

Mail: rasmus@cederdorff.com

**Private kontaktoplysninger**

Adresse: Langørvej 79B, 8381 Tilst

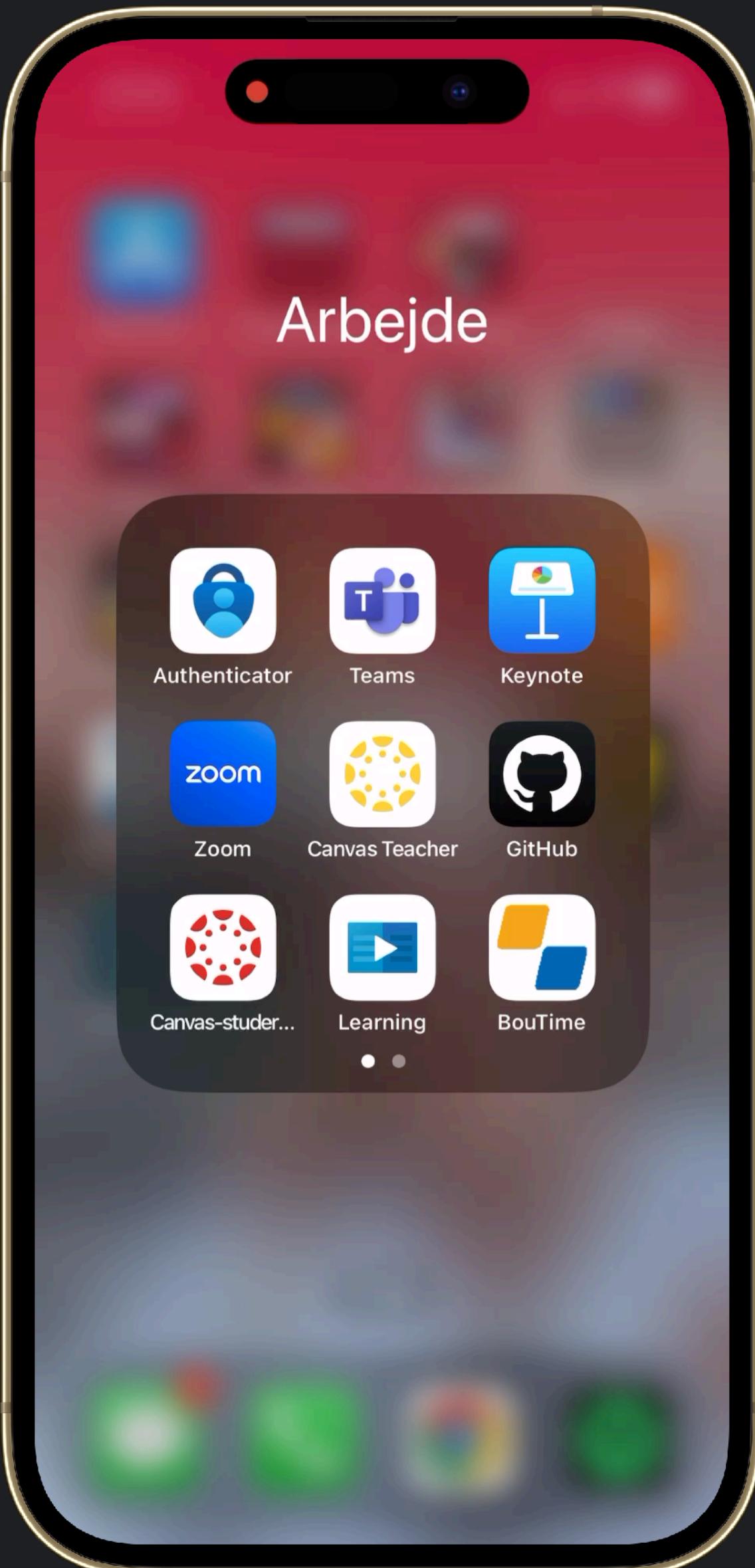
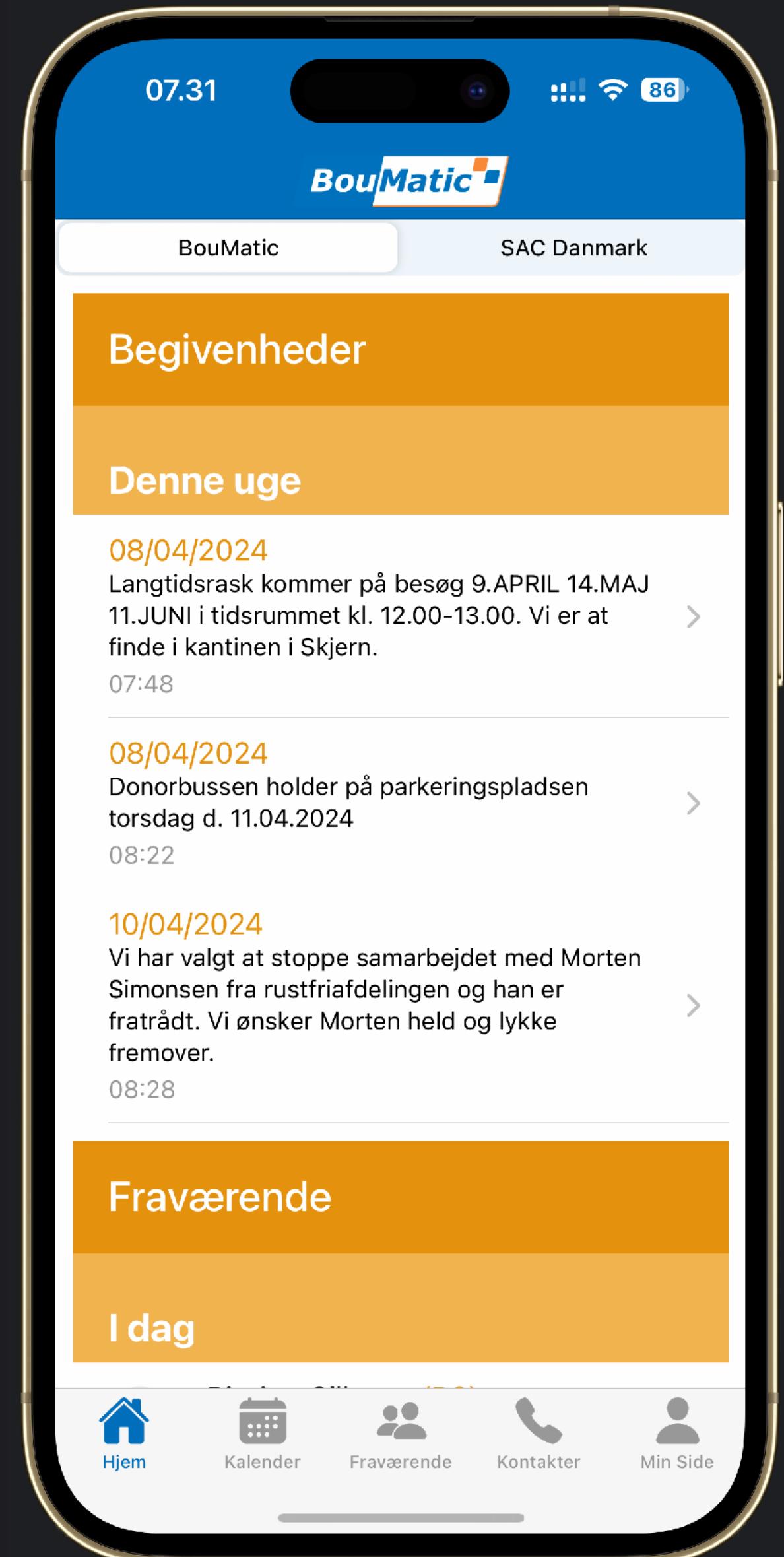
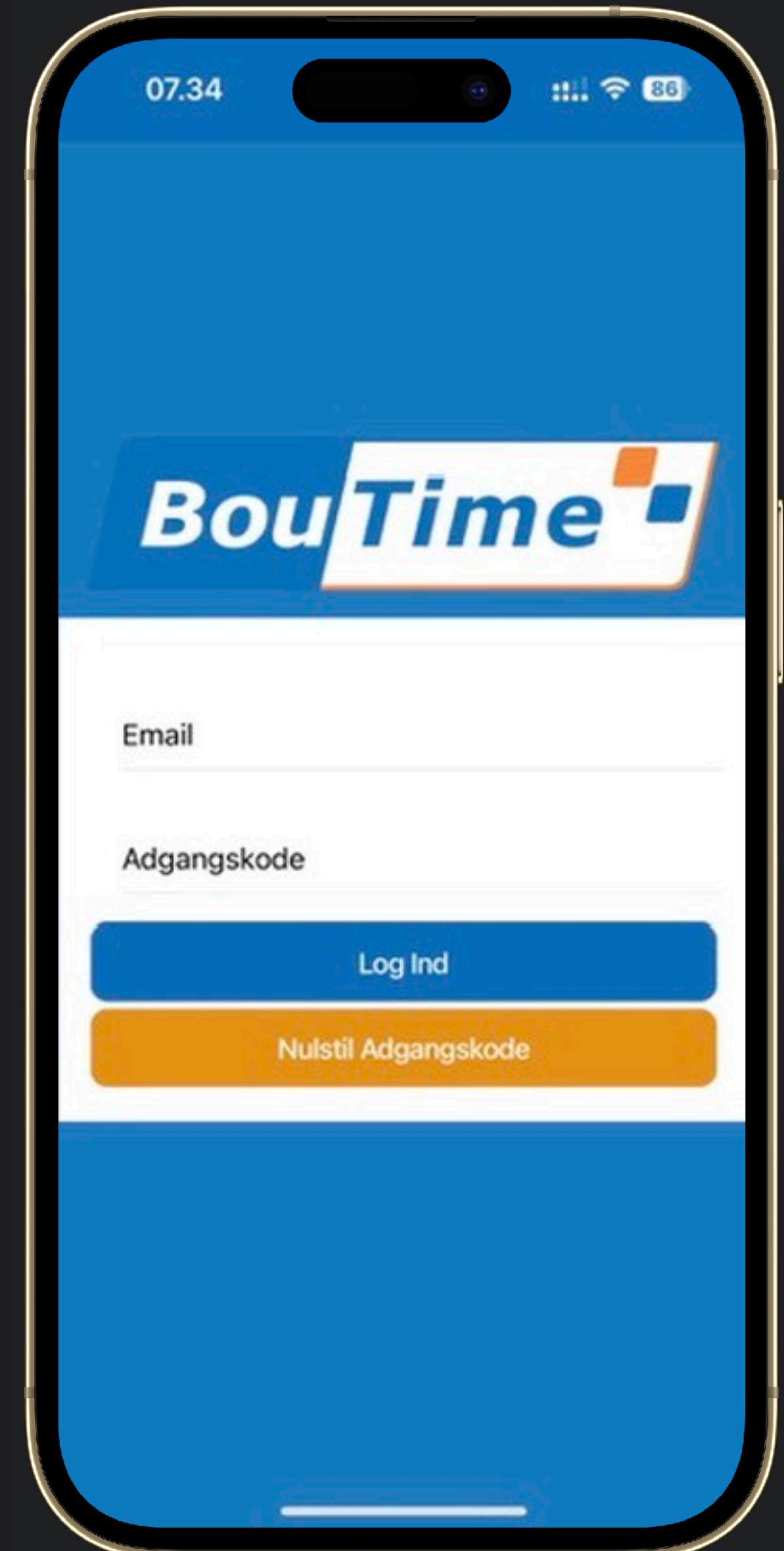
**Bruger**

Se fravær:

Nulstil din adgangskode:

Log Ud:

**Hjem** **Kalender** **Fraværende** **Kontakter** **Min Side**



# (Re)Use the expertise you already have in-house

Traditional Approach



Web (Native) Approach



# Combining the best of web and native apps



“[...] feels as good to  
use as a platform-  
specific application [...]”

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

One Codebase  
Any Platform

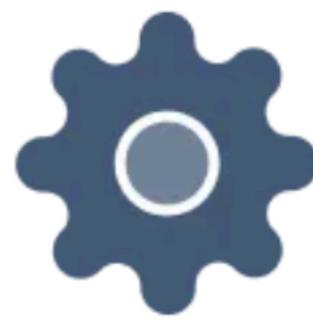
# What are PWAs?

The three pillars of PWA design



## Reliable

Works offline and performs well on low-quality networks.



## Capable

Web apps are more capable than ever given modern APIs.



## Installable

Runs as a standalone window and launches from the home screen.

Website, Web App, Native  
App, Hybrid App & Web Native

# Website

The user wants to get something!

# Web App

The user wants to do something!

Remix Post App

localhost:3000/posts

## POSTS

# Posts

 **Dan Okkels Brendstrup**  
Lecturer



A cozy morning with coffee

Likes: 785

Tags: morning, coffee, cozy, food

 **Rasmus Cederdorff**  
Senior Lecturer



Serenity of the forest

Likes: 799

Tags: forest, nature, serenity

 **Maria Louise Bendixen**  
Senior Lecturer



A beautiful morning in Aarhus

Likes: 99

Tags: morning, Aarhus, beautiful, AROS

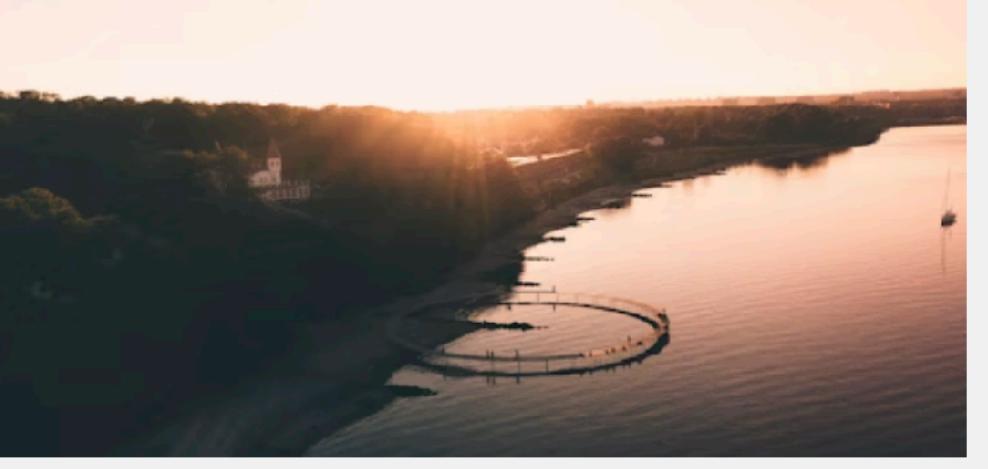
 **Anne Kirketerp**  
Head of Department



 **Rasmus Cederdorff**  
Senior Lecturer



 **Maria Louise Bendixen**  
Senior Lecturer



Remix Post App

localhost:3000/posts

POSTS ADD POST PROFILE

# Posts

Search by caption

Filter by tag

Sort by

Search

select tag

newest

 **Dan Okkels Brendstrup**  
Lecturer



A cozy morning with coffee

Likes: 785

Tags: morning, coffee, cozy, food

 **Rasmus Cederdorff**  
Senior Lecturer



Serenity of the forest

Likes: 799

Tags: forest, nature, serenity

 **Maria Louise Bendixen**  
Senior Lecturer

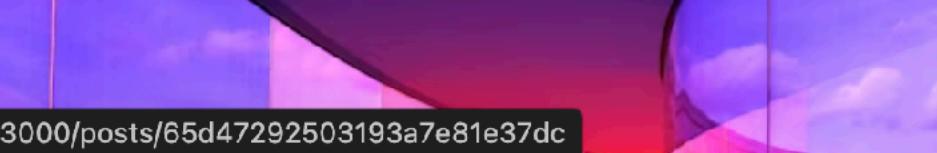


A beautiful morning in Aarhus

Likes: 99

Tags: morning, Aarhus, beautiful, AROS

 **Anne Kirketerp**  
Head of Department



localhost:3000/posts/65d47292503193a7e81e37dc

 **Rasmus Cederdorff**  
Senior Lecturer



 **Maria Louise Bendixen**  
Senior Lecturer



# Web App



- Any program executed in the browser.
- Any program that performs a specific function by using a web browser as its client.
- The user wants to complete a task - to do something.
- Stored on a web server.
- Accessible anywhere, at any time.
- Build with HTML, CSS & JS.
- Not just about mobile devices.

# Types of Web Apps

WEB BASED

BROWSER BASED

API BASED

SERVER APPS

PORTAL APPS

SOFTWARE AS A SERVICE

PROGRESSIVE WEB APP

SPA & MPA

# Native



iOS

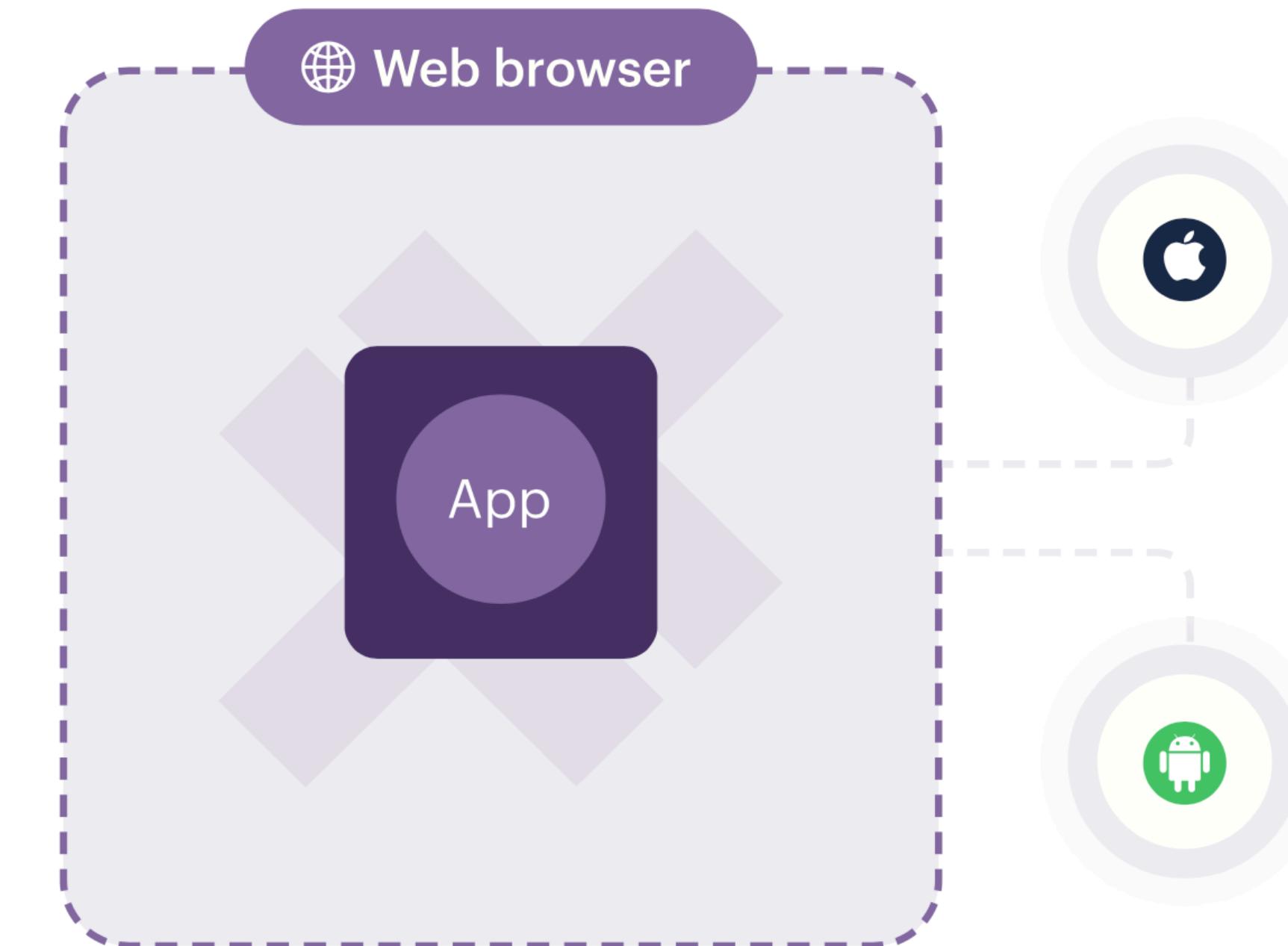
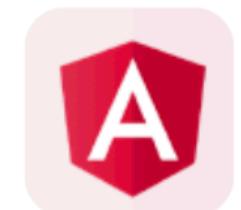
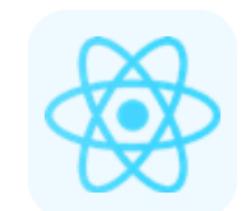
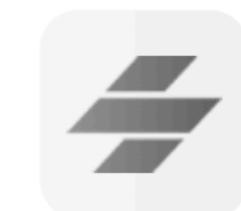
Android

- Build with native & platform-specific languages & tools.
- iOS - Xcode with Swift or Objective-C
- Android - Android Studio with Kotlin or Java
- Full and easier access to the device's capabilities.
- “tend to also be more performant since their code is closer to the ‘metal’”.
- Native user interface (UI) controls and layouts.
- Android Apps cannot run on iOS and vice versa.

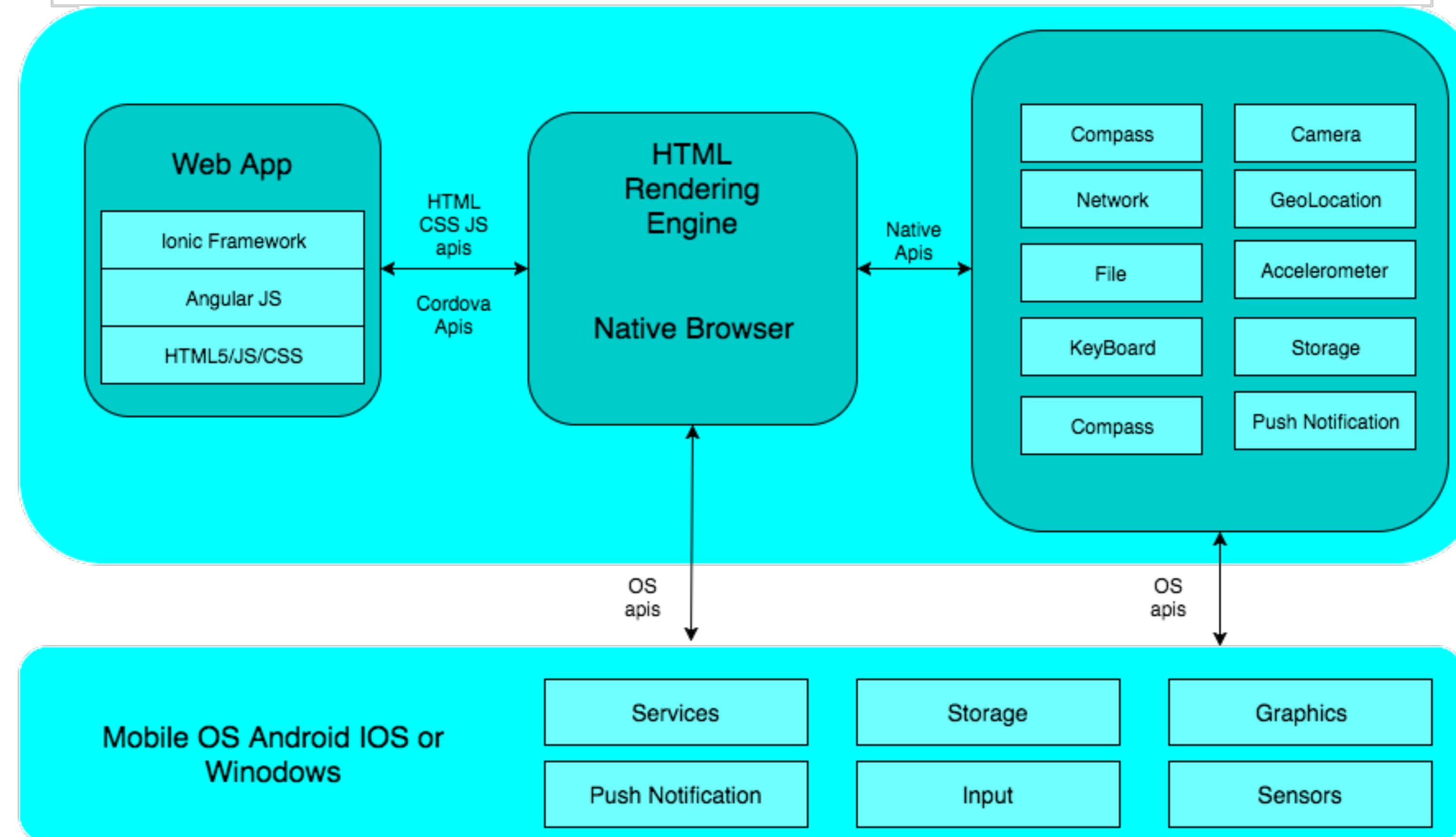
# Hybrid

- “Hybrid apps are like native apps. They’re downloaded from the platform’s app store or marketplace and offer the same native features, offline support, and hardware-based performance acceleration as any app built with a native SDK.”
- A blend - both native & web.
- Build with HTML, CSS & JS (+ framework) & cross-platform like Ionic, NativeScript, Xamarin, React Native.
- With plugins: full access to native features.
- Embedded in WebView.
- With a UI Framework: feels and looks like a pure native.
- Bridge & plugins - Cordova & Capacitor.

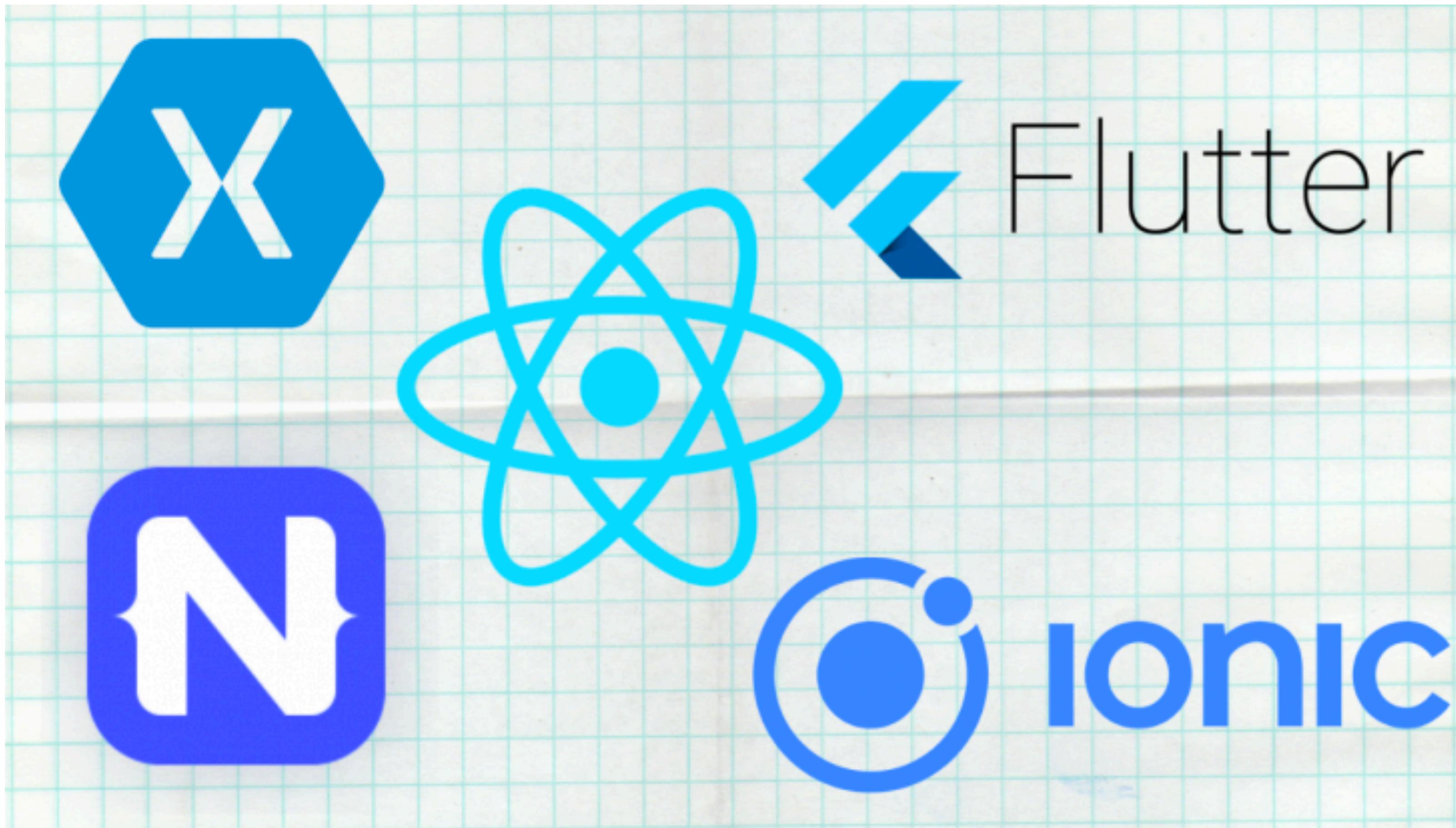




## Hybrid App Architecture



# Cross-platforms



# Web Native

More than *just* Hybrid

# Web Native

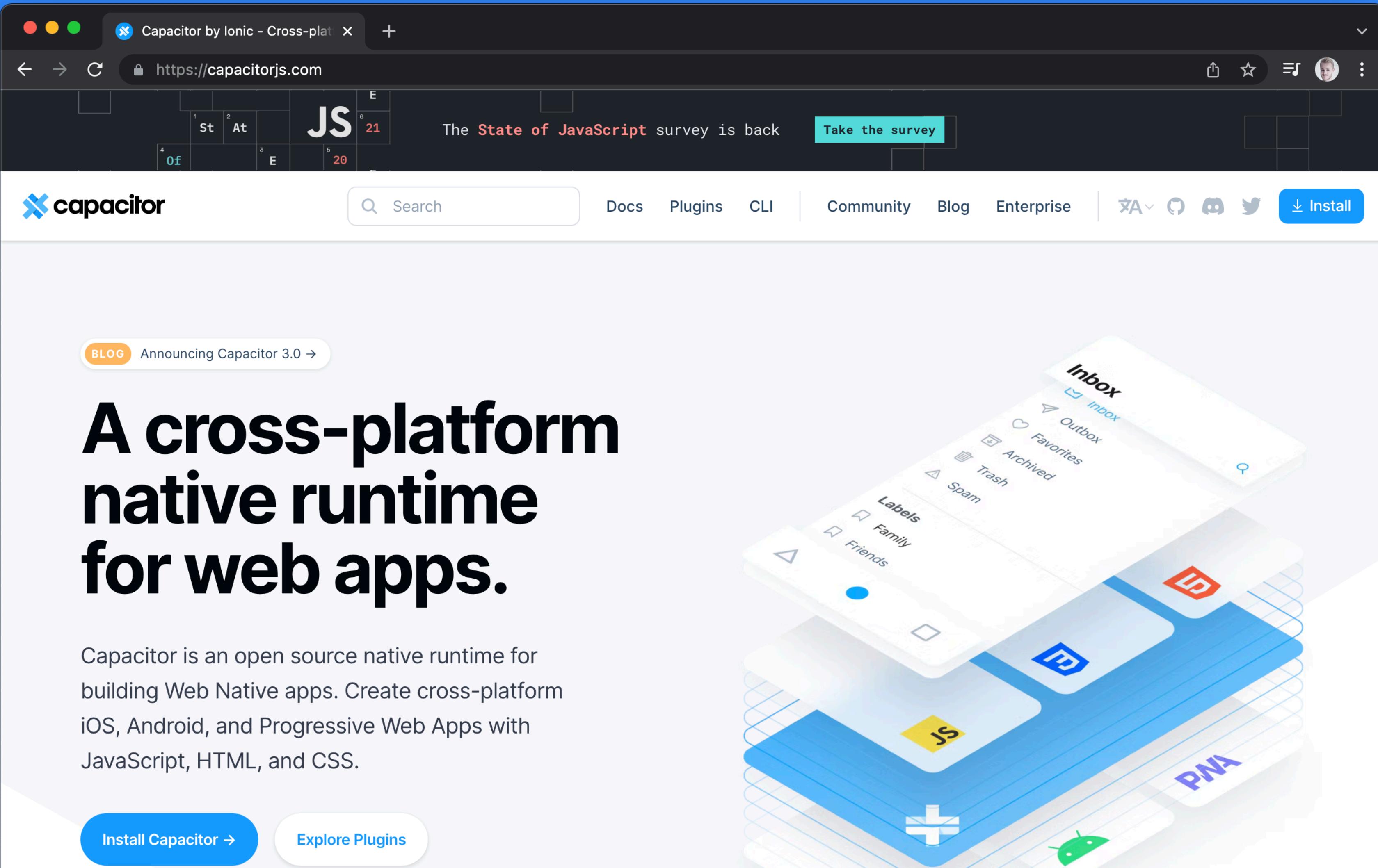
Is the full capability and access to developers of the Web Platform, with the full functionality and performance benefits of traditional native apps.

<https://webnative.tech/>

Web Native is "hybrid" done right, and it's the future of mobile app development.

# Capacitor

A cross-platform native runtime for web apps.



The screenshot shows a web browser displaying the Capacitor website at <https://capacitorjs.com>. The page features a large, bold headline: "A cross-platform native runtime for web apps.". Below the headline is a paragraph describing Capacitor as an open source native runtime for building Web Native apps. To the right of the text is a graphic of three smartphones showing different app interfaces: one with an "Inbox" screen, one with a "Labels" screen, and one with a "PNA" screen. At the bottom of the page are two call-to-action buttons: "Install Capacitor →" and "Explore Plugins". The browser's header includes the title "Capacitor by Ionic - Cross-plat", a search bar, and various browser controls.

BLOG Announcing Capacitor 3.0 →

# A cross-platform native runtime for web apps.

Capacitor is an open source native runtime for building Web Native apps. Create cross-platform iOS, Android, and Progressive Web Apps with JavaScript, HTML, and CSS.

Install Capacitor → Explore Plugins

# Back in the 2010s

Feature	Native	Web-only	Hybrid
<b>Device Access</b>	Full	Limited	Limited
<b>Performance</b>	High	Medium to High	Low
<b>Development Language</b>	Platform Specific	HTML, CSS, Javascript	HTML, CSS, Javascript
<b>Cross-Platform Support</b>	No	Yes	Yes
<b>User Experience</b>	High	Medium to High	Low
<b>Code Reuse</b>	No	Yes	Yes

# Today

Feature	Native	Web-only	Hybrid
<b>Device Access</b>	Full	Limited	Full (with plugins)
<b>Performance</b>	High	Medium to High	Medium to High
<b>Development Language</b>	Platform Specific	HTML, CSS, Javascript	HTML, CSS, Javascript
<b>Cross-Platform Support</b>	No	Yes	Yes
<b>User Experience</b>	High	Medium to High	Medium to High
<b>Code Reuse</b>	No	Yes	Yes

# Capacitor

A cross-platform native runtime for Web Native Apps.

The Web View and the native app communicate through the use of Capacitor or Cordova plugins. Plugins provide native APIs such as camera, geolocation, and filesystem access to your web app.



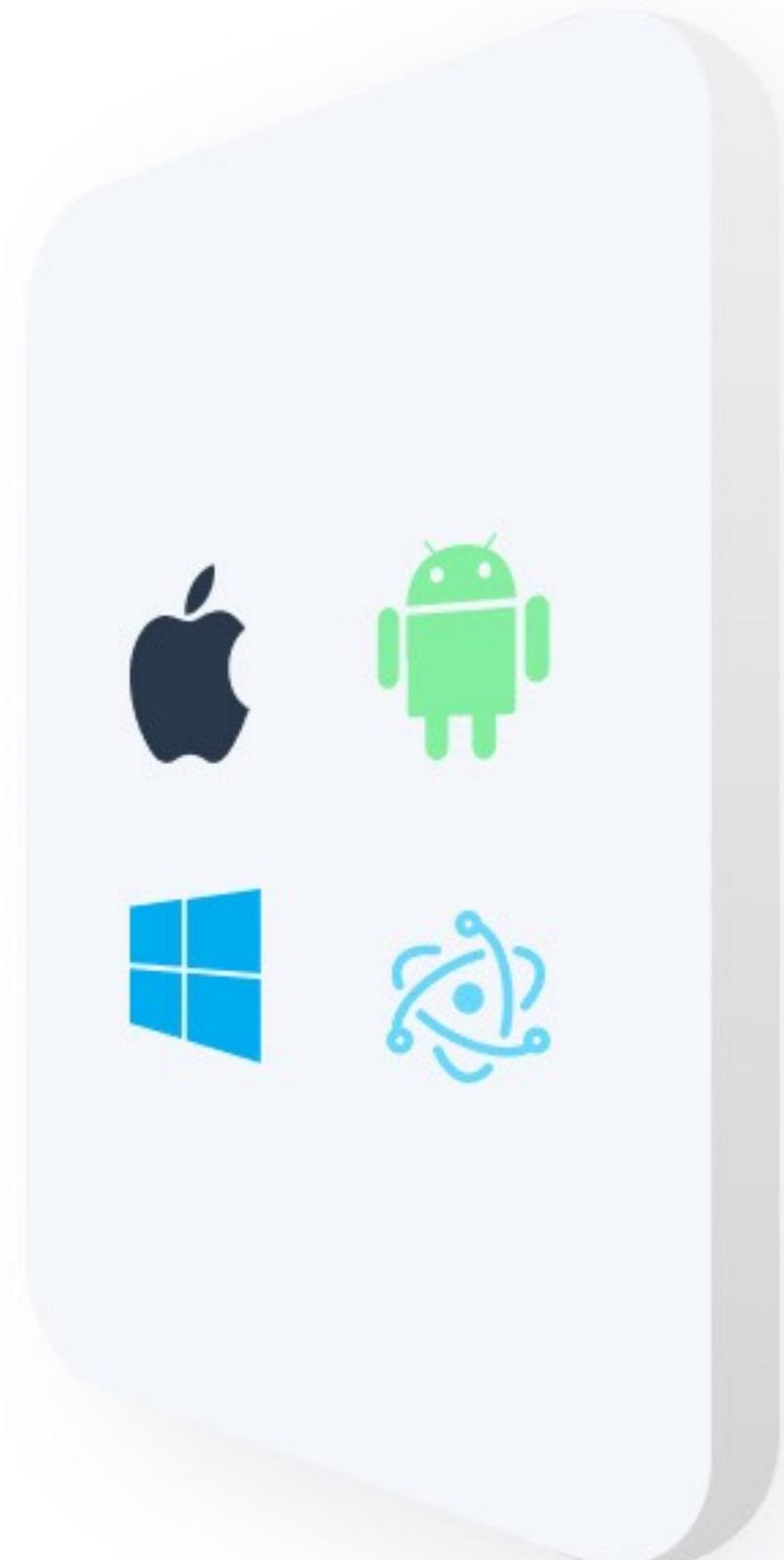
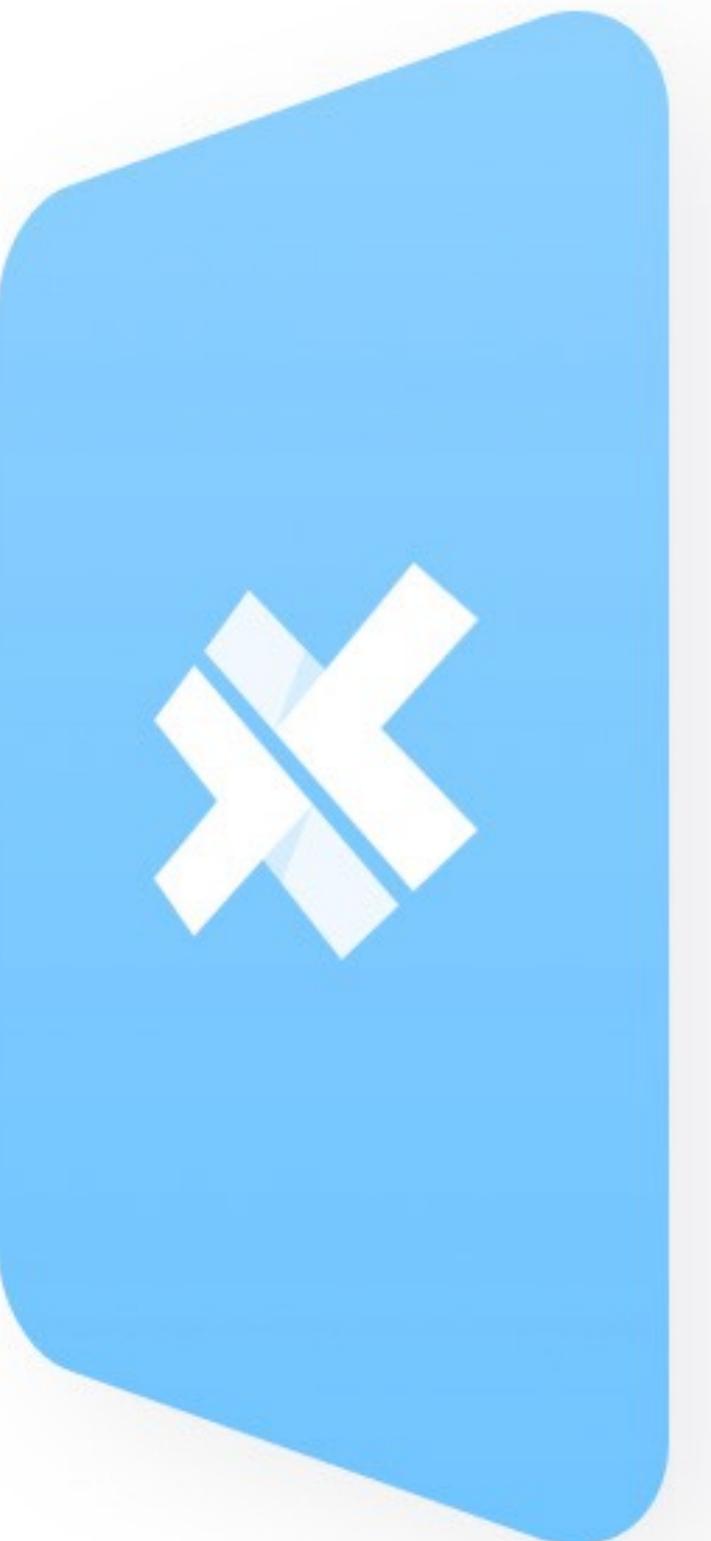
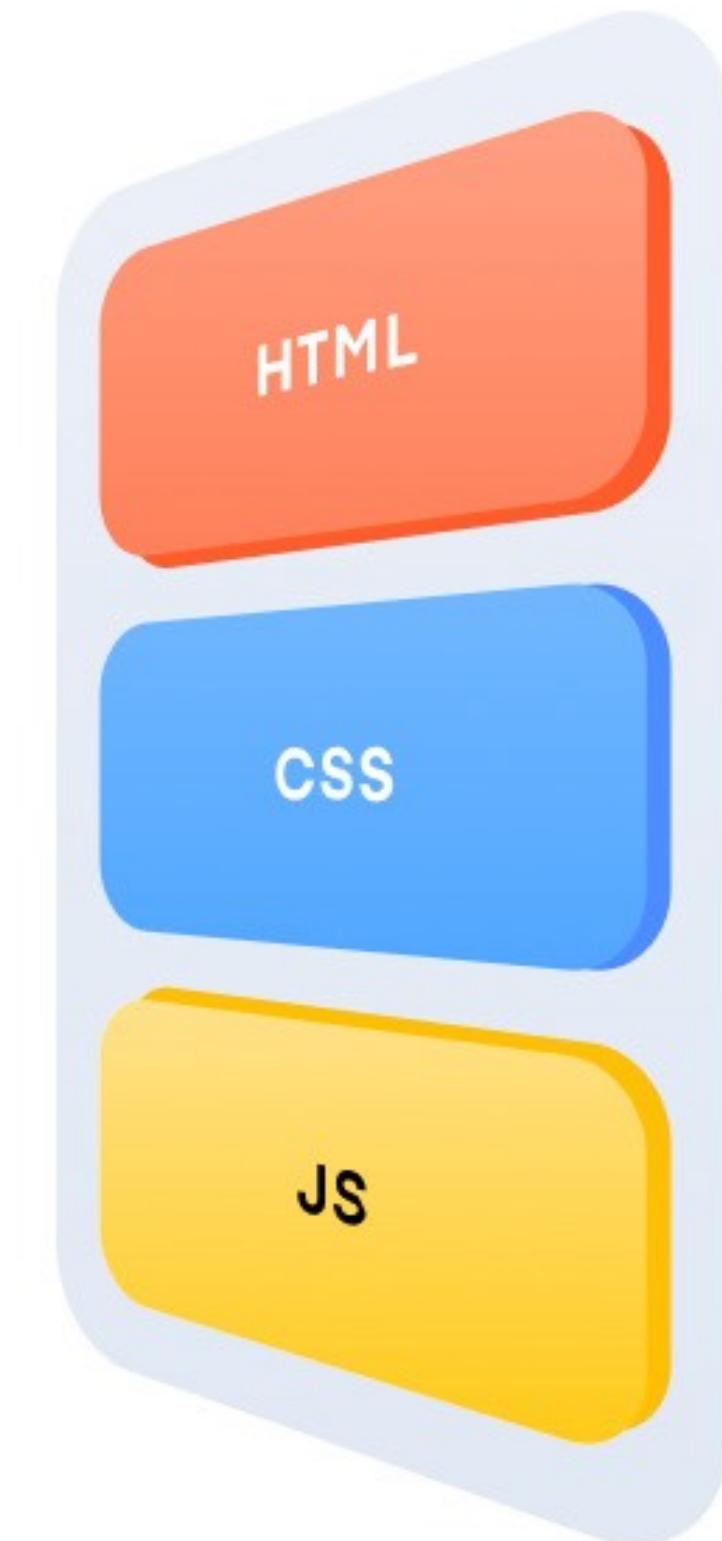
UI (Web View)

JS, HTML & CSS

Ionic React

Capacitor

iOS, Android & PWA



YOUR APP (ANGULAR, REACT, VUE...)

UI CONTROLS (IONIC)

NATIVE ACCESS (CAPACITOR)

DISTRIBUTION PLATFORMS

# Native vs Web Native (Hybrid)

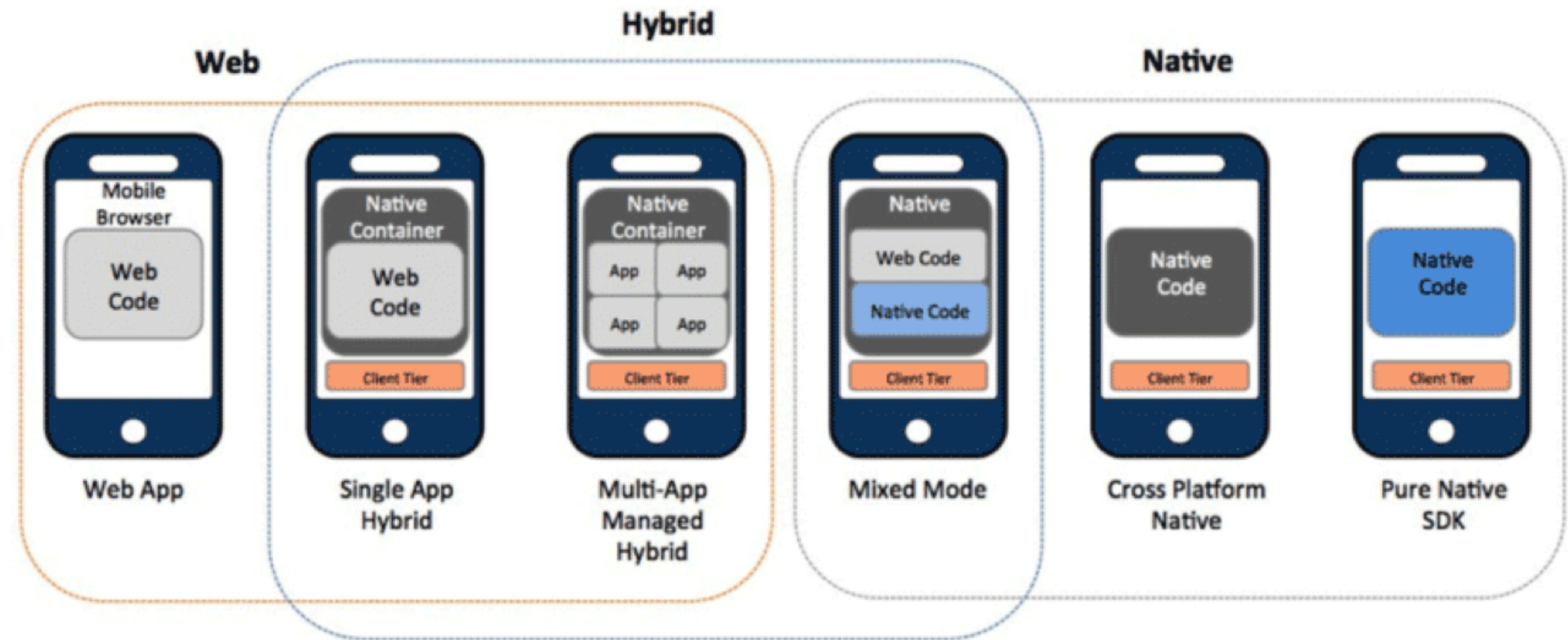
What are the differences? Pros & Cons?

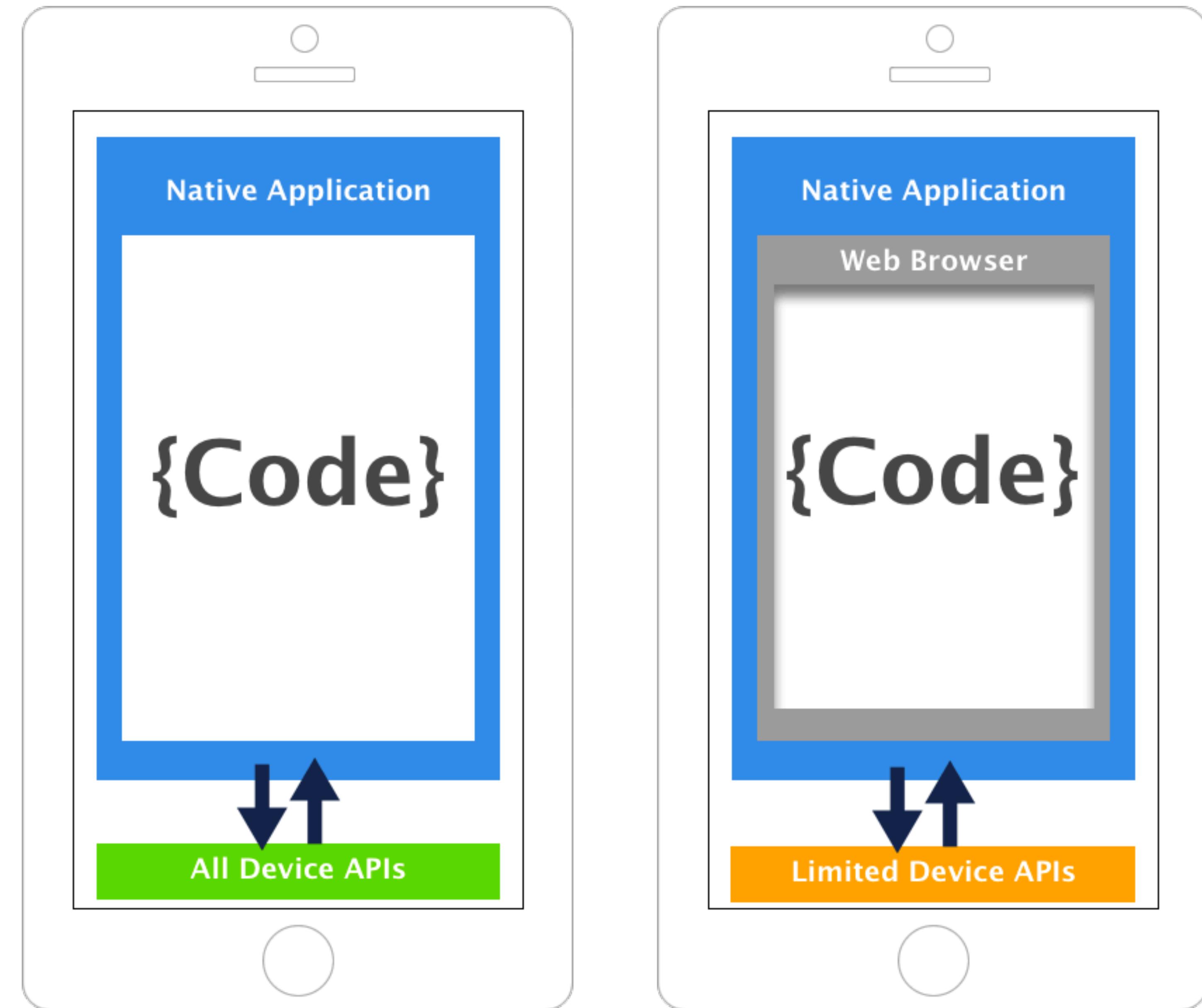
“The key difference is that hybrid apps are built using open web technologies like HTML, CSS, and JavaScript, rather than the proprietary or specialized languages used by iOS, Android, and others. That means anyone with a modern web developer skill-set can begin building an app using the hybrid approach.

Hybrid apps run in a full-screen browser, called a webview, that is invisible to the user. Through customizable native plugins, they can access the native features of specific mobile devices (such as the camera or touch ID), without the core code being tied to that device.

That means cross-platform hybrid applications can run on any platform or device, all from a single codebase, while still delivering native performance.“







## HYBRID APPS



## NATIVE APPS



# Why Native? Why Hybrid?

Hybrid vs Native (ebook): <http://go.ionic.io/hybrid-vs-native-guide>

# Why hybrid?

Let's take a look at each of these.



Write once, run anywhere



Use the talent you  
already have



Deliver a great user  
experience across  
platforms



Build for the future

# Core Features of PWAs

- **Responsive:** Work on any device (desktop, tablet, smartphone).
- **Connectivity Independent:** Service workers allow work offline or on low-quality networks.
- **App-like:** Feel like a native app due to the use of an app shell model.
- **Fresh:** Always up-to-date due to the service worker update process.



# Core PWA Checklist

- Starts fast, stays fast
- Works in any browser
- Responsive to any screen size
- Provides a custom offline page
- Is installable



<https://web.dev/articles/pwa-checklist#core>

# Optimal PWA Checklist

- Provides an offline experience
- Is fully accessible
- Can be discovered through search
- Works with any input type
- Provides context for permission requests
- Follows best practices for healthy code

<https://web.dev/articles/pwa-checklist#optimal>

# Development Tools & Technologies

- Languages: HTML, CSS, JavaScript (React, Angular, Vue for SPAs).
- Tools: Lighthouse (for auditing), Workbox (for service workers), PWA Builder.
- Platforms: Ex. Firebase for hosting and backend services.



JS

At the end of  
the day, it is all  
just JavaScript



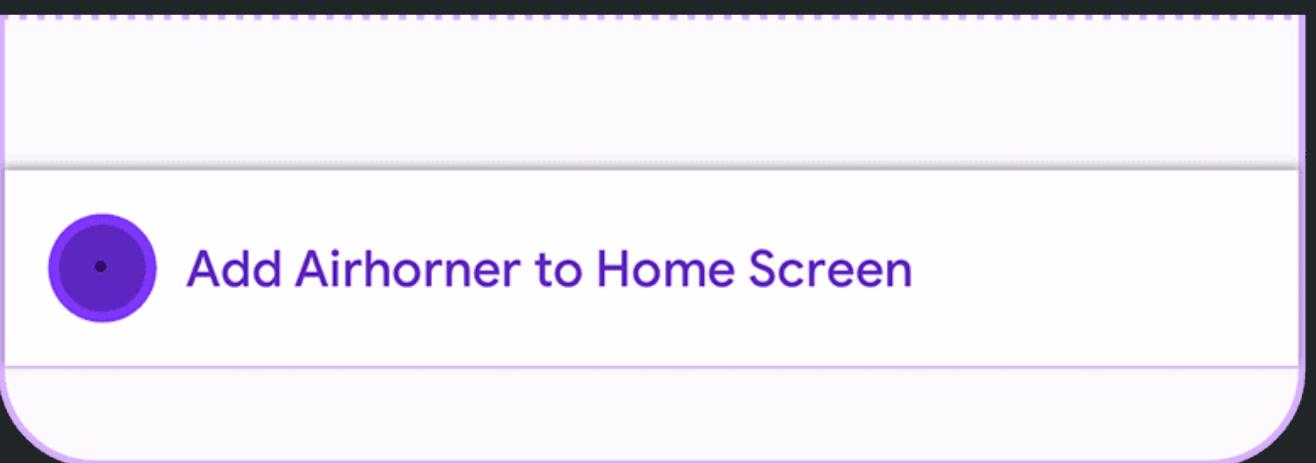
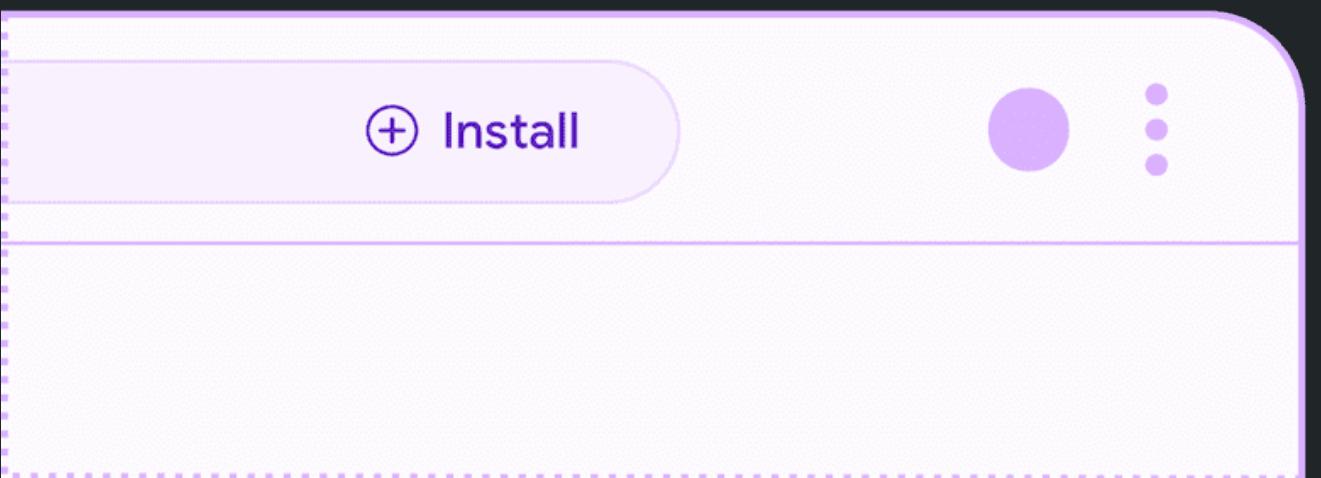
# Installable Web Apps

*When a Progressive Web App moves out of a tab and into a standalone app window, it transforms how users think about it and interact with it.*

<https://web.dev/articles/what-are-pwas#installable>

# Installable Web Apps

- Manifest File
- Logo
- HTTPS
- (Service Worker)



<https://web.dev/articles/install-criteria>

# Technical Foundations of PWAs

- Service Workers: For offline functionality and background tasks.
- Manifest File: For adding the web app to the home screen.
- HTTPS: For secure data transactions.



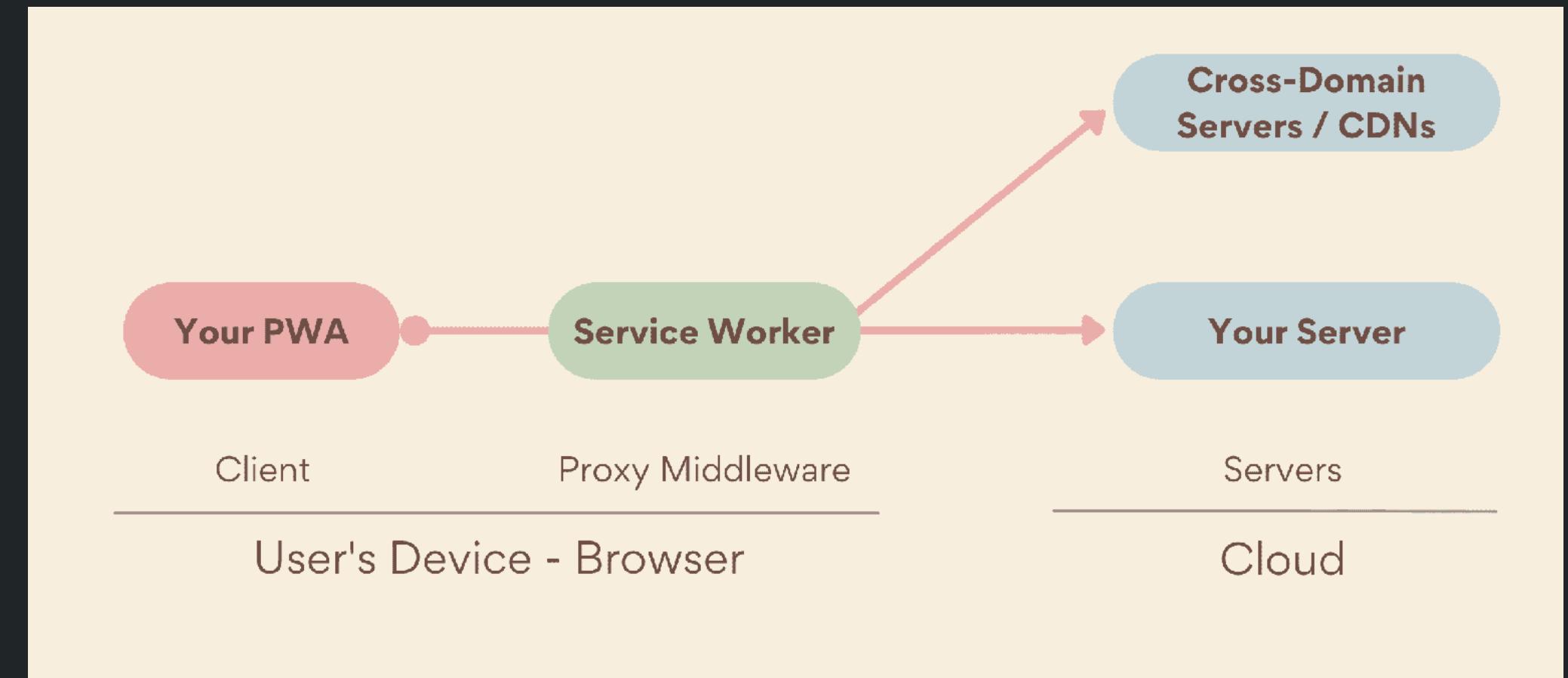
[7 Web Features You Didn't Know Existed](#)

<https://web.dev/articles/install-criteria>

# What is a Service Worker?

Service workers enhance web app performance and user experience.

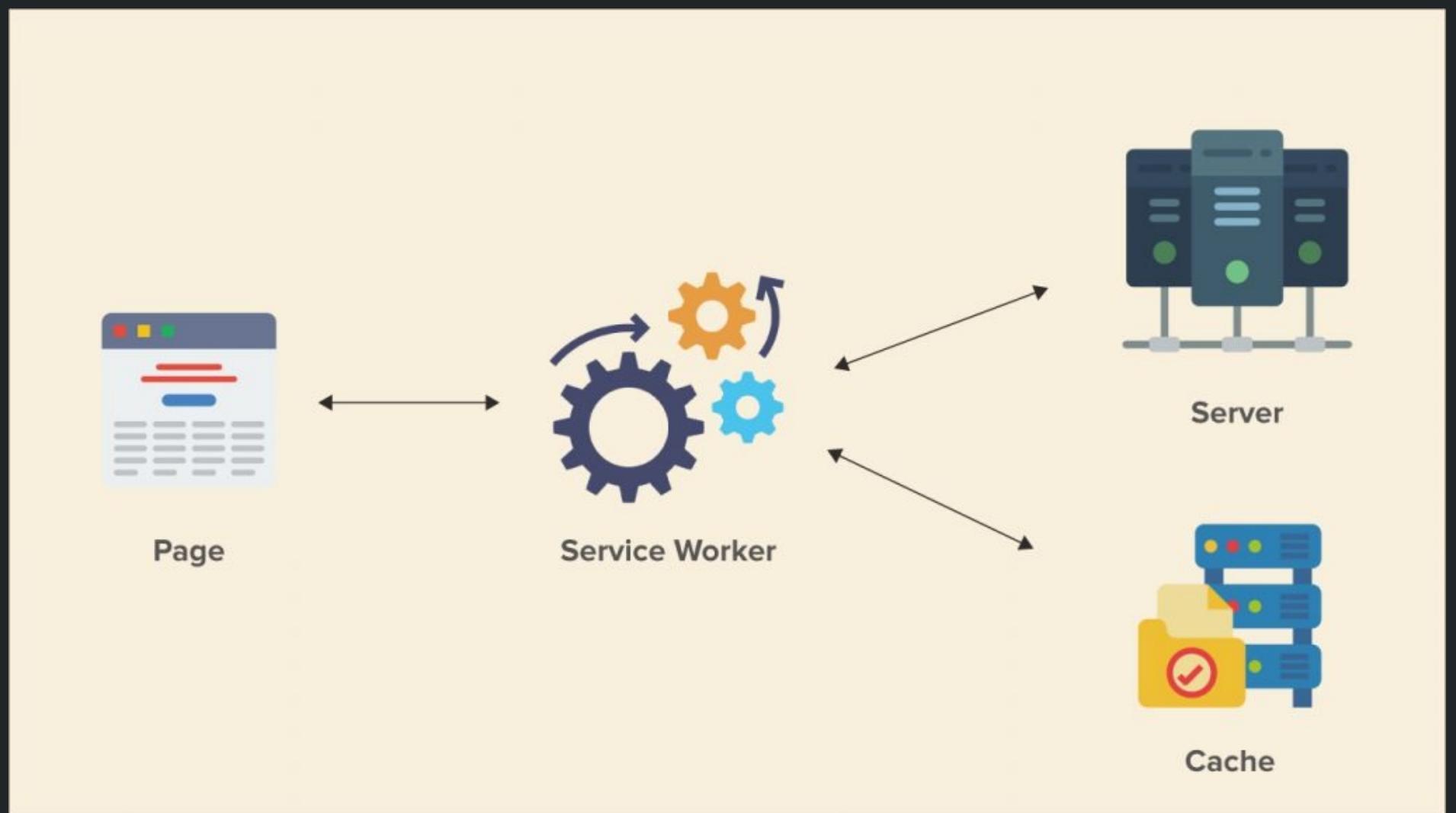
- A script that runs in the background, separate from a web page.
- Acts as a proxy between the browser, the network, and the web application.
- Enables features like push notifications, background sync, and offline functionality.



<https://web.dev/learn/pwa/service-workers>

# Service Worker: Core Functionalities

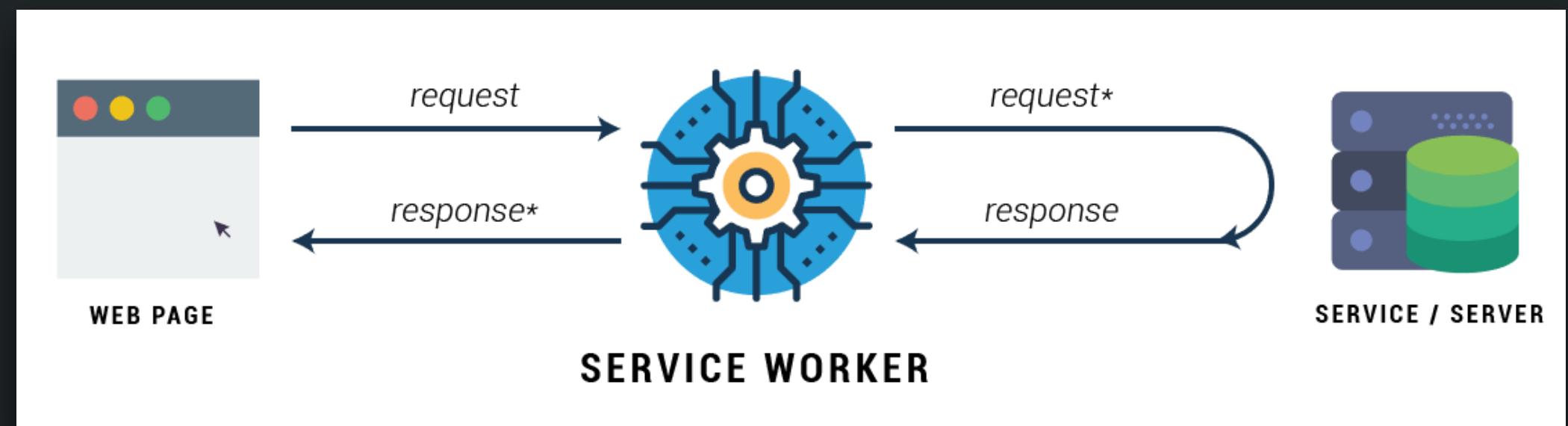
- Proxy Server: Intercepts network requests to serve custom responses.
- Offline Capability: Caches assets and data for app use without an internet connection.
- Background Updates: Updates cached content in the background.
- Push Notifications: Sends notifications to re-engage users.



<https://web.dev/learn/pwa/service-workers>

# Service Worker

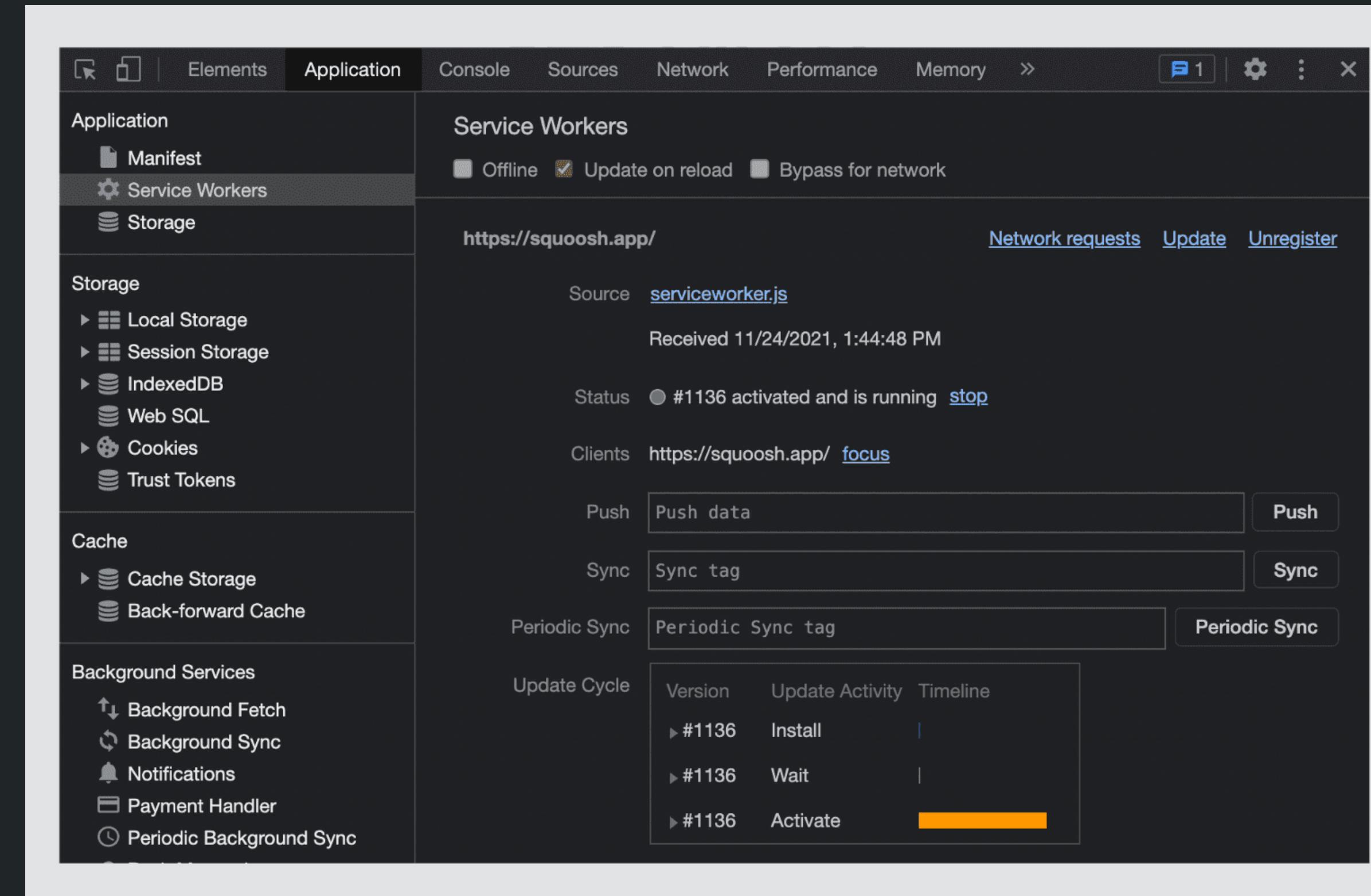
- Key Features
  - Event-Driven: Operates based on events like fetch, install, and activate.
  - Life Cycle: Independent from the web page, includes installation, activation, and updates.
  - Async Operations: Utilizes promises for asynchronous actions.
- Requirements & Limitations
  - HTTPS Required: Needs a secure context to operate, except on localhost for development.
  - Compatibility: Not all browsers support service workers fully.
  - Complexity: Requires careful design to manage caching and updates effectively.



<https://web.dev/learn/pwa/service-workers>

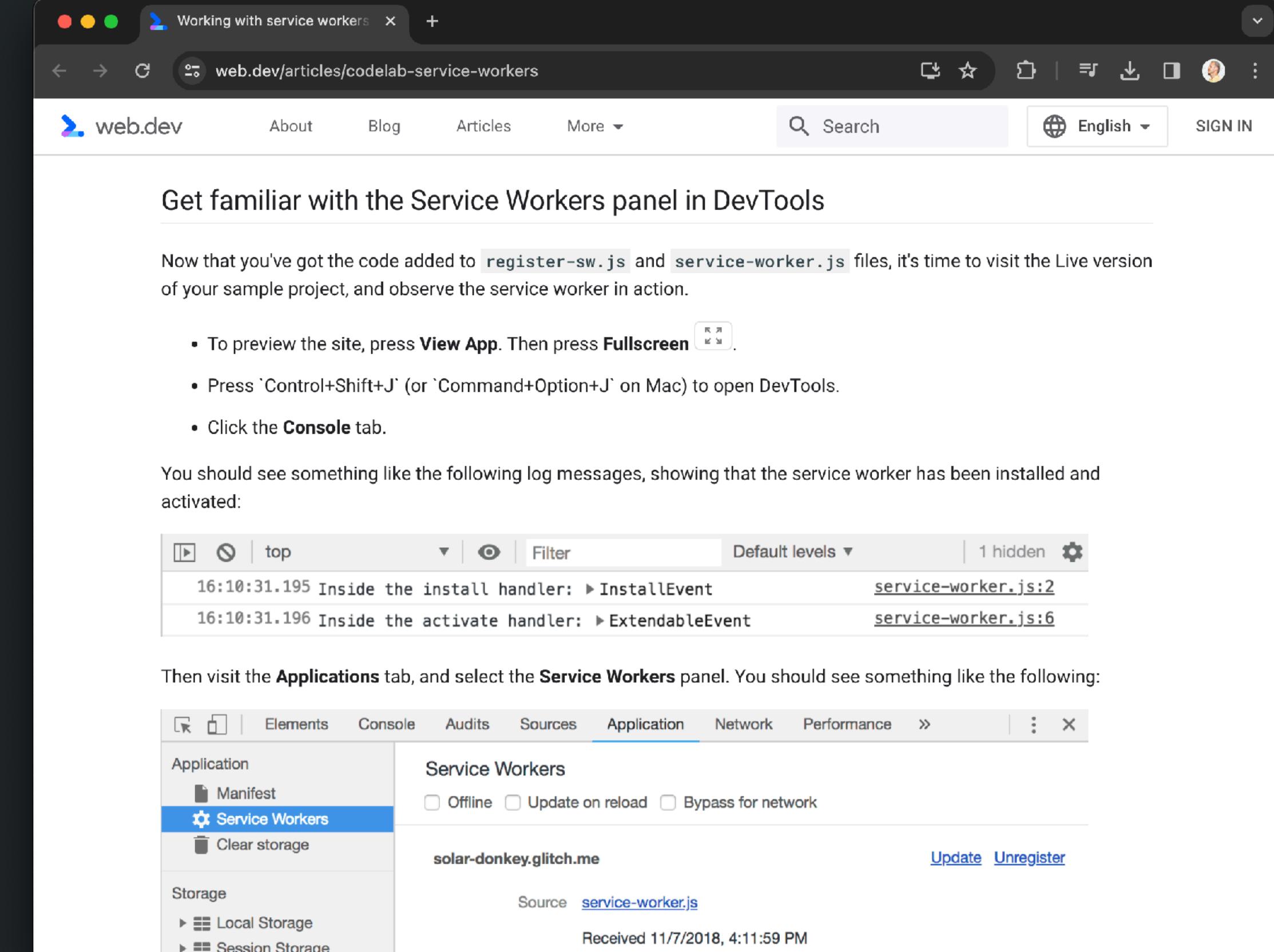
# Service Worker: Tools

- Lighthouse: For auditing and performance metrics.
- Workbox: A set of libraries to simplify common service worker tasks and caching.
- Service Workers menu



<https://web.dev/learn/pwa/service-workers>

# Working with Service Workers



The screenshot shows a web browser window with the title "Working with service workers". The URL in the address bar is "web.dev/articles/codelab-service-workers". The page content includes a heading "Get familiar with the Service Workers panel in DevTools", instructions for previewing the site using DevTools, and a log message in the browser's developer tools console indicating the service worker has been installed and activated. Below the console, the "Application" tab of the developer tools is shown, specifically the "Service Workers" section, which lists a registered service worker for the domain "solar-donkey.glitch.me".

Get familiar with the Service Workers panel in DevTools

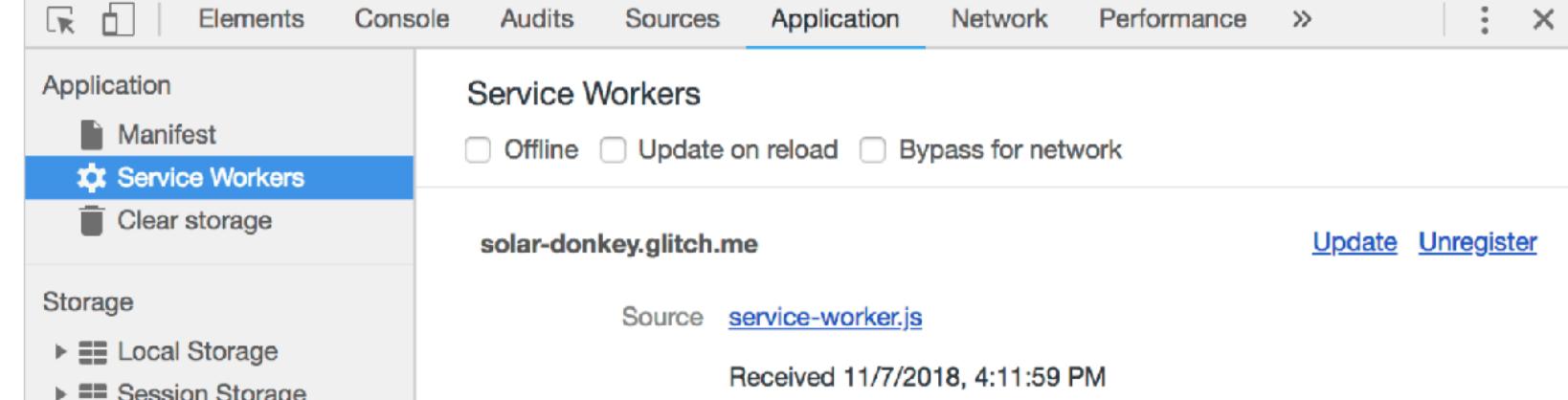
Now that you've got the code added to `register-sw.js` and `service-worker.js` files, it's time to visit the Live version of your sample project, and observe the service worker in action.

- To preview the site, press **View App**. Then press **Fullscreen**.
- Press `Control+Shift+J` (or `Command+Option+J` on Mac) to open DevTools.
- Click the **Console** tab.

You should see something like the following log messages, showing that the service worker has been installed and activated:

```
16:10:31.195 Inside the install handler: > InstallEvent          service-worker.js:2
16:10:31.196 Inside the activate handler: > ExtendableEvent      service-worker.js:6
```

Then visit the **Applications** tab, and select the **Service Workers** panel. You should see something like the following:



The screenshot shows the "Application" tab of the developer tools with the "Service Workers" panel selected. It displays a list of registered service workers, including one for the domain "solar-donkey.glitch.me". The "Source" field shows the file "service-worker.js" and the "Received" timestamp is "11/7/2018, 4:11:59 PM".

<https://web.dev/articles/codelab-service-workers>

# Benefits of PWAs

- Improved Performance: Faster load times and smoother interactions.
- Increased Engagement: Features like push notifications.
- Cost-Effective: Development and maintenance costs lower than native apps.

<https://web.dev/articles/install-criteria>

# Benefits of PWAs

- It's just a website! You don't need to build separate apps anymore. If you have a website, you can easily turn it into an iOS and Android app as well!
- A PWA is much smaller than a native app. Your users no longer need to install tens of megabytes of code
- No need to get your app into the App Store or Play Store. Just share the link to your website and users can install it as an app
- There's no need to get users to install updates anymore. When you release a new version of your app, all your users automatically get the new version
- By default, PWAs are served over HTTPS and are therefore safe and secure
- PWAs are lightweight and offer high performance
- Especially on Android, a PWA can almost do anything a native app can



# Challenges & Considerations

- Browser Compatibility: Not all PWA features are supported across all browsers.
- Performance Optimization: Requires careful consideration, especially for complex applications.



# Real-world Examples of PWAs

- Twitter
- Youtube
- Pinterest
- Uber
- Starbucks
- Canvas

