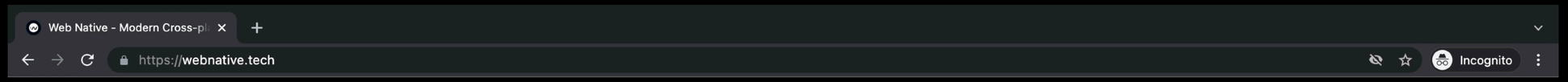



Powered by ionic

It's time we talked about the elephant in the room.

Hybrid mobile apps have been built the same way for almost 10 years.

While great for accelerating app development across teams, the architecture underneath is antiquated and brittle.

https://webnative.tech/


Introduction to
Mobile App Dev

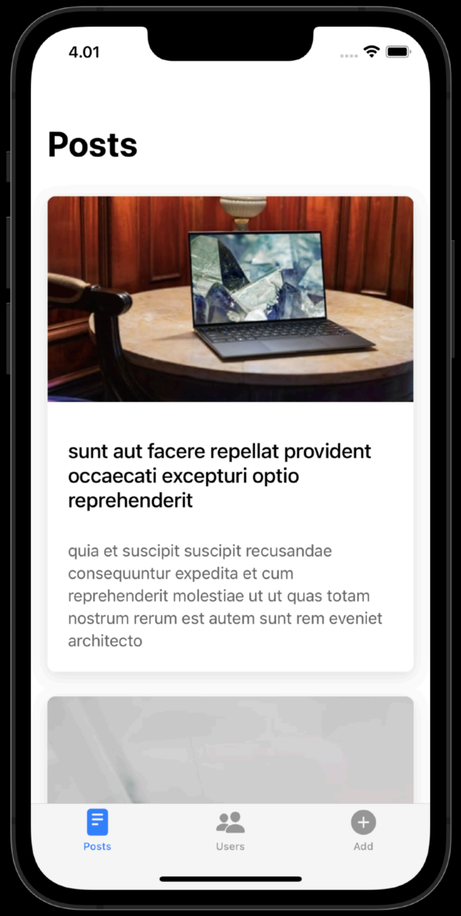

ionic

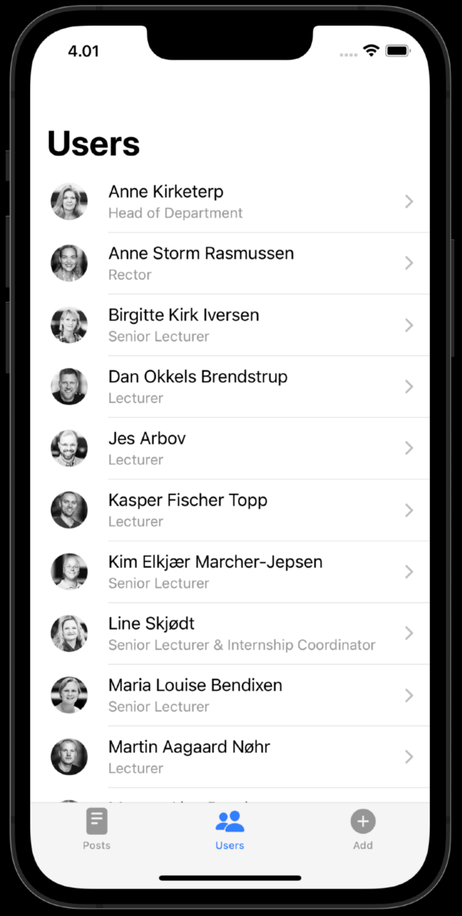

BouTime

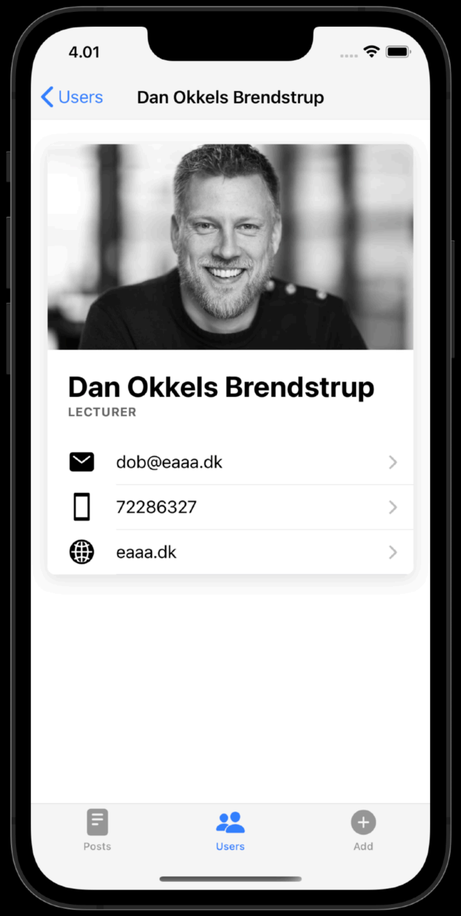
# Agenda

- Course Overview & Practicalities
- What’s mobile app dev?
- Types of Apps
- Ionic & Ionic React
- CLI & Tools
- Getting started w/ Ionic React

# Demo




[ionic-post-app-firebase-rest-v6]

# Mobile App Dev

*... deals with programming techniques, user interface development and web integration which are necessary to develop applications for mobile devices - from idea to fully developed, distribution-ready application for one (or more) platform.*

## Uge 34

| | Mandag d. 22 - 08 | Tirsdag d. 23 - 08 | Onsdag d. 24 - 08 | Torsdag d. 25 - 08 | Fredag d. 26 - 08 |
|---|---|---|---|---|---|
| 08:30 - 10:00 | | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 10:30 - 12:00 | | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 12:30 - 14:00 | | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 14:15 - 15:45 | | | | | |

## Uge 35

| | Mandag d. 29 - 08 | Tirsdag d. 30 - 08 | Onsdag d. 31 - 08 | Torsdag d. 01 - 09 | Fredag d. 02 - 09 |
|---|---|---|---|---|---|
| 08:30 - 10:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 10:30 - 12:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 12:30 - 14:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 14:15 - 15:45 | | | | | |

## Uge 36

| | Mandag d. 05 - 09 | Tirsdag d. 06 - 09 | Onsdag d. 07 - 09 | Torsdag d. 08 - 09 | Fredag d. 09 - 09 |
|---|---|---|---|---|---|
| 08:30 - 10:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 10:30 - 12:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 12:30 - 14:00 | **Advanced Web Programming**<br>**Bemærk:**<br>AWP Project | **Mobile**<br>RACE<br>eaa-R104-1.12 | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 14:15 - 15:45 | | | | | |

## Uge 37

| | Mandag d. 12 - 09 | Tirsdag d. 13 - 09 | Onsdag d. 14 - 09 | Torsdag d. 15 - 09 | Fredag d. 16 - 09 |
|---|---|---|---|---|---|
| 08:30 - 10:00 | **Mobile**<br>**Bemærk:**<br>Ionic Project | **Mobile**<br>**Bemærk:**<br>Ionic Project | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| 10:30 - 12:00 | **Mobile**<br>**Bemærk:**<br>Ionic Project | **Mobile**<br>**Bemærk:**<br>Ionic Project | **Progressive Web Apps**<br>DOB<br>eaa-R104-1.12 | | **Advanced Web Programming**<br>DOB<br>eaa-R104-1.12 |
| | **Mobile** | **Mobile** | **Progressive Web Apps** | | **Advanced Web Programming** |

# Course Overview - Part 1

| Date | Name | Notes |
| --- | --- | --- |
| 23/08/2022 | 1. Introduction to Mobile App Dev | Course Intro |
| 30/08/2022 | 2. Web Native, Tools & UI Components | |
| 06/09/2022 | 3. Native Device Features | Ionic Project Kick Off |
| 13/09/2022 | 4. Ionic Project 1 | Group Work |
| 20/09/2022 | 5. Storage & Native APIs | Internship info at 14.00 |
| 27/09/2022 | 6. Firebase & App Dev | Final Ionic React lecture |
| 04/10/2022 | 7. Ionic Project 2 | Group Work |
| 11/10/2022 | 8. Ionic Project 3 | Group Work |
| 18/08/2022 | 9. Ionic Project 4 | Group Work |

MAD - Course Overview

# Curriculum

https://www.baaa.dk/media/o2riq0gp/curriculum-web-development-elective-catalogue-february-2022.pdf

# Exam

The exam is an individual written exam in the form of a report with a maximum of 15 pages including images, which is prepared on the basis of a group project.

# Two Mobile App Projects

During the Mobile App Dev Course, you will be introduced to two approaches of Mobile App Development (Ionic React and Unity), finalised by a project.

The projects forms the basis of your exam in Mobile App Dev and must be handed in, in order to pass the course and participate in the exam at the end of the semester.

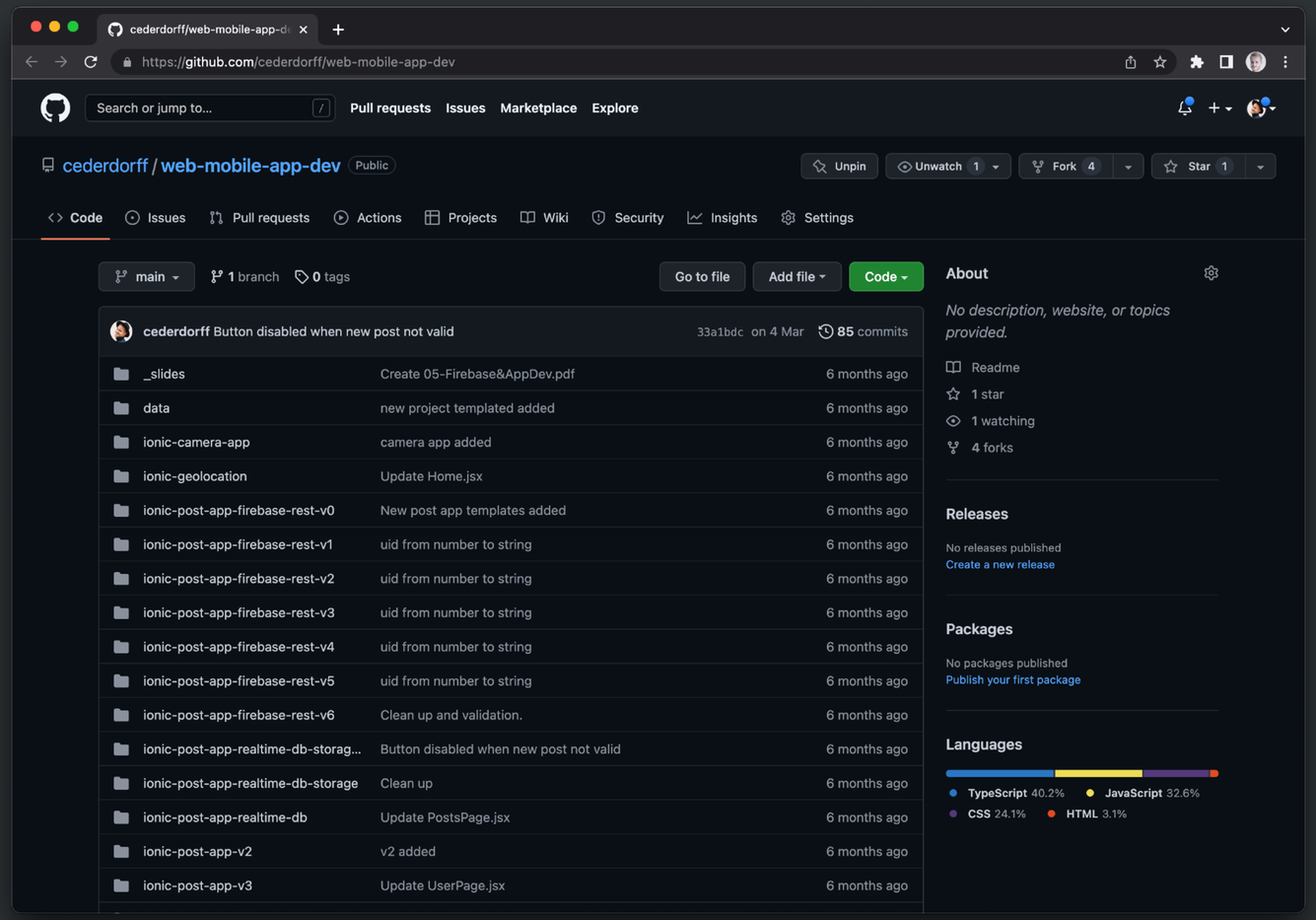
Comparing the different approaches, will be a part of the final individual written exam. Read more about the exam in the curriculum.

# Two Mobile App Project

## Scope

- The exam is an individual written exam in the form of a report with a maximum of 15 pages including images, which is prepared based on either your Ionic App or Unity App group project.
- During the exam, you must review your previous project, and improve and extend your mobile app (Ionic or Unity).
- The technical requirements are equal to the listed in the project descriptions (Ionic App Project or Unity App Project)
- In the exam project, you must compare the two different approaches to mobile app development (Ionic VS Unity) and reflect on the pros and cons.

# GitHub

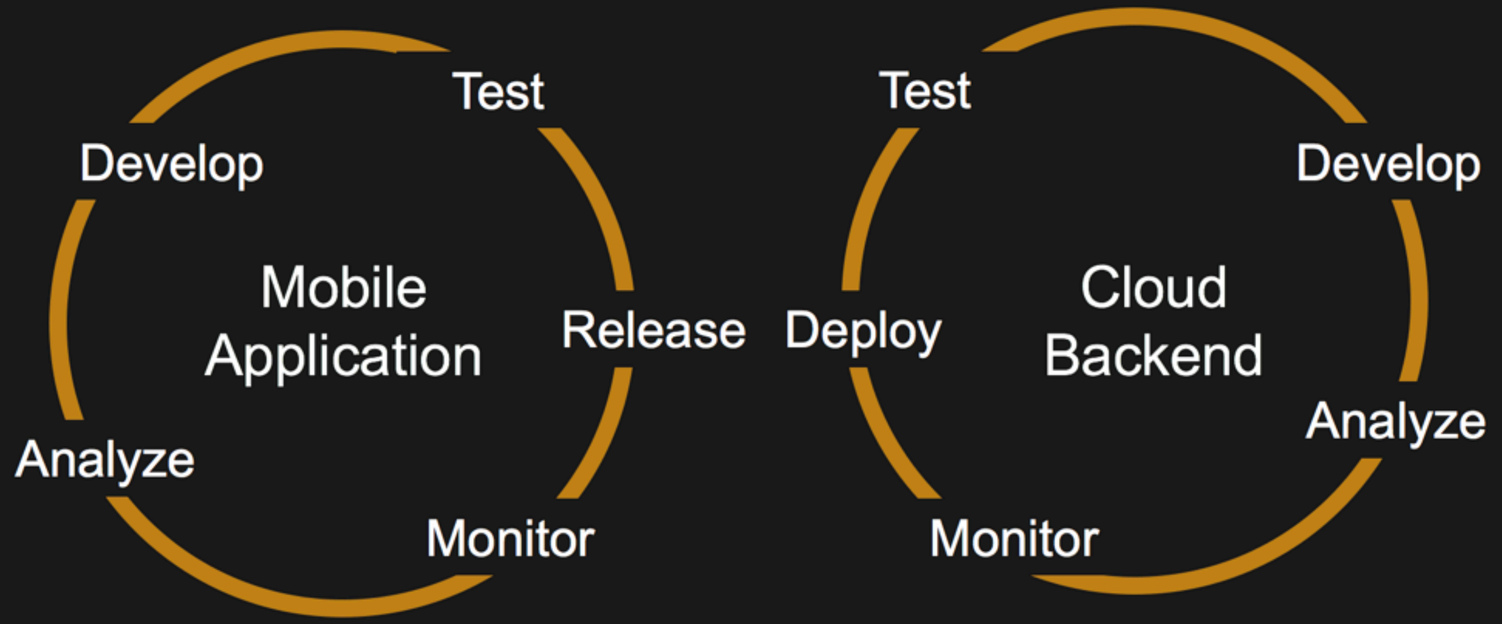


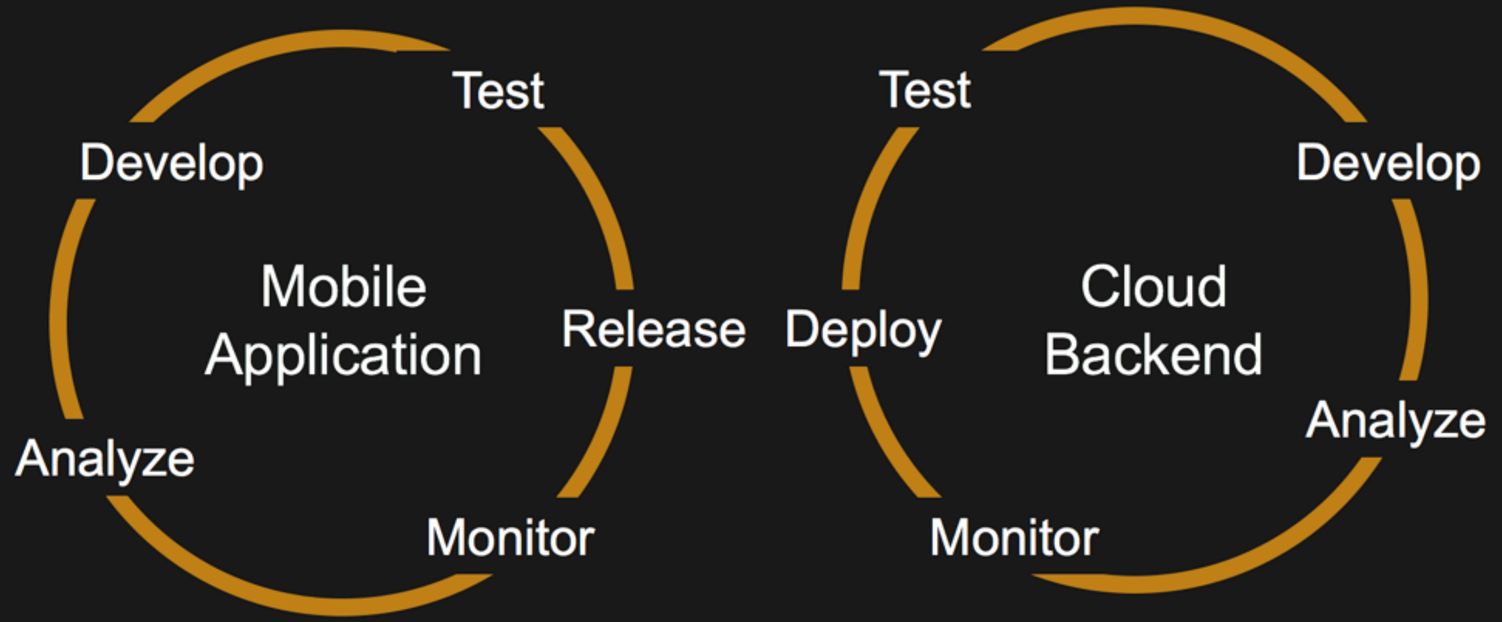

https://github.com/cederdorff/web-mobile-app-dev

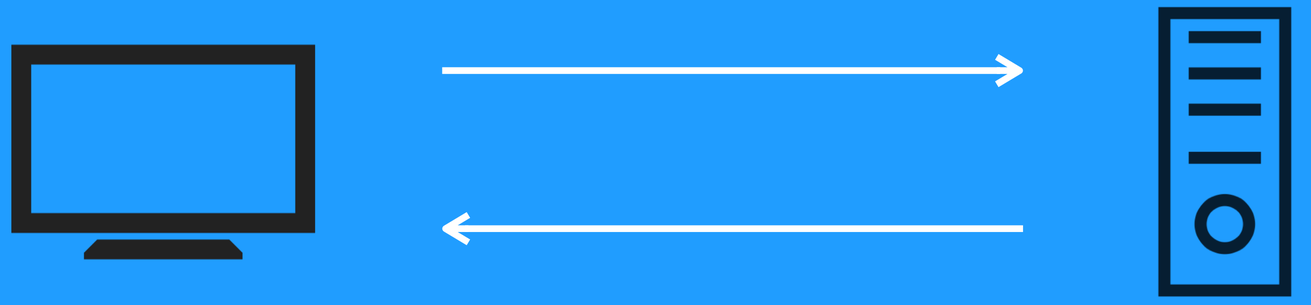
# What’s mobile app dev?

The process of developing software applications that run on a mobile device, such as smartphones, tablets & smartwatches.

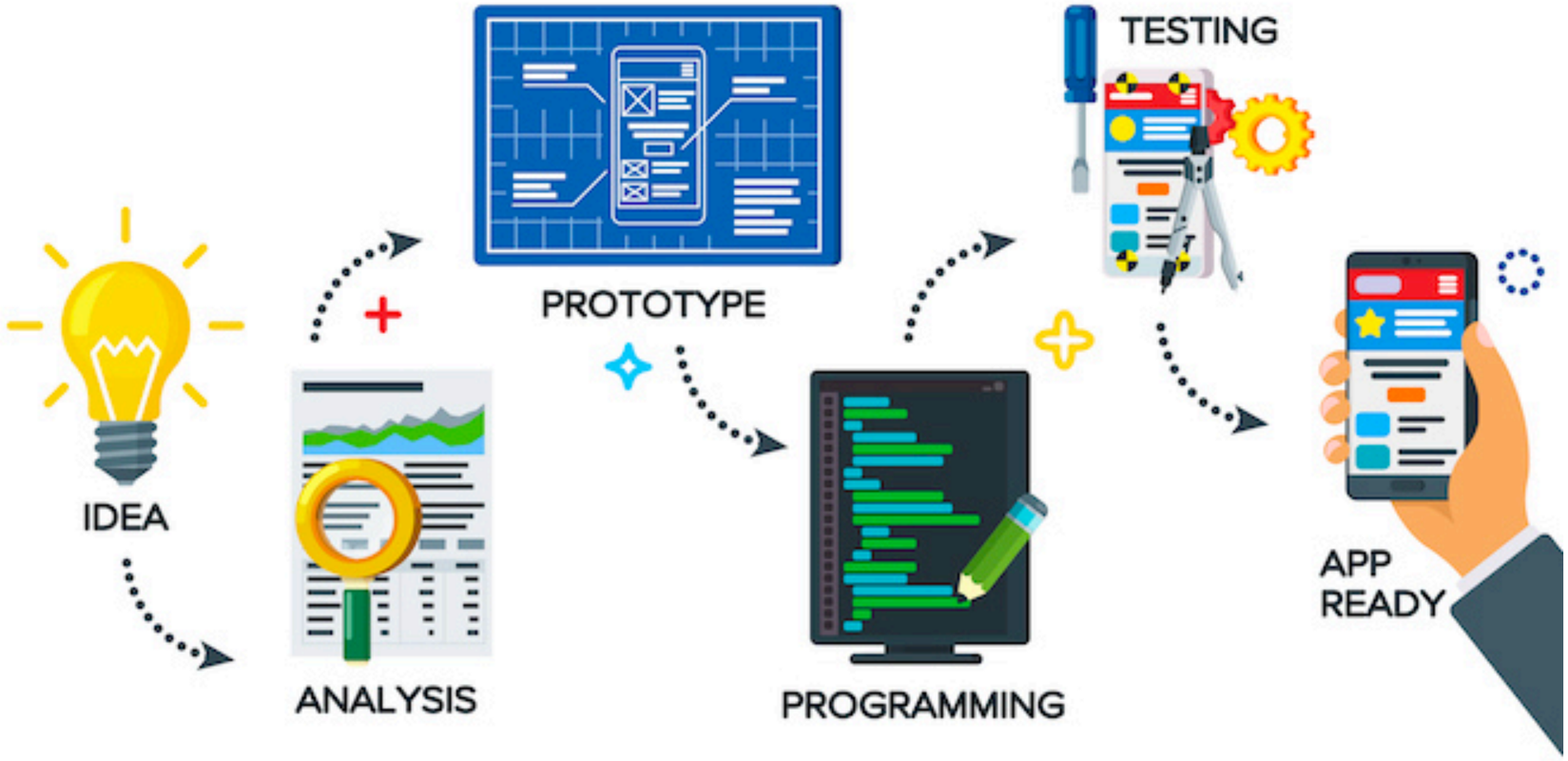
# Mobile Dev Lifecycle




Test
Develop
Mobile
Application
Release
Analyze
Monitor
Test
Develop
Deploy
Cloud
Backend
Analyze
Monitor


CLIENT
SERVER

# “from idea to fully developed, distribution-ready application”



# Download: App Store & Google Play

# Native, Hybrid, Web Native & Web App

Types of Mobile Apps


# Native

Build with native & platform-specific languages & tools.
iOS - Xcode with Swift or Objective-C
Android - Android Studio with Kotlin or Java

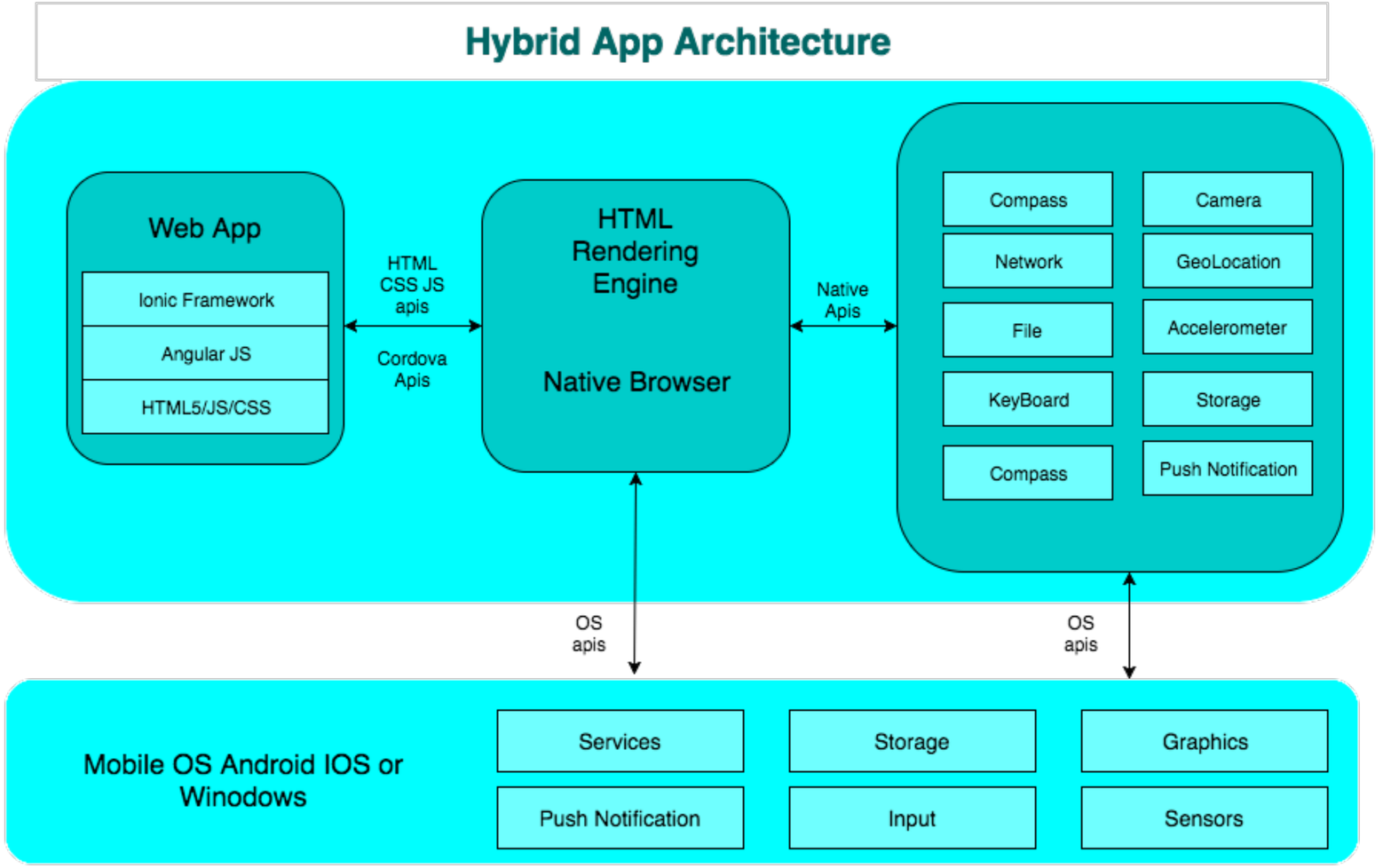
Full and easier access to the device’s capabilities.
“tend to also be more performant since their code is closer to the ‘metal’”.
Native user interface (UI) controls and layouts.
Android Apps cannot run on iOS and vice versa.

# Hybrid

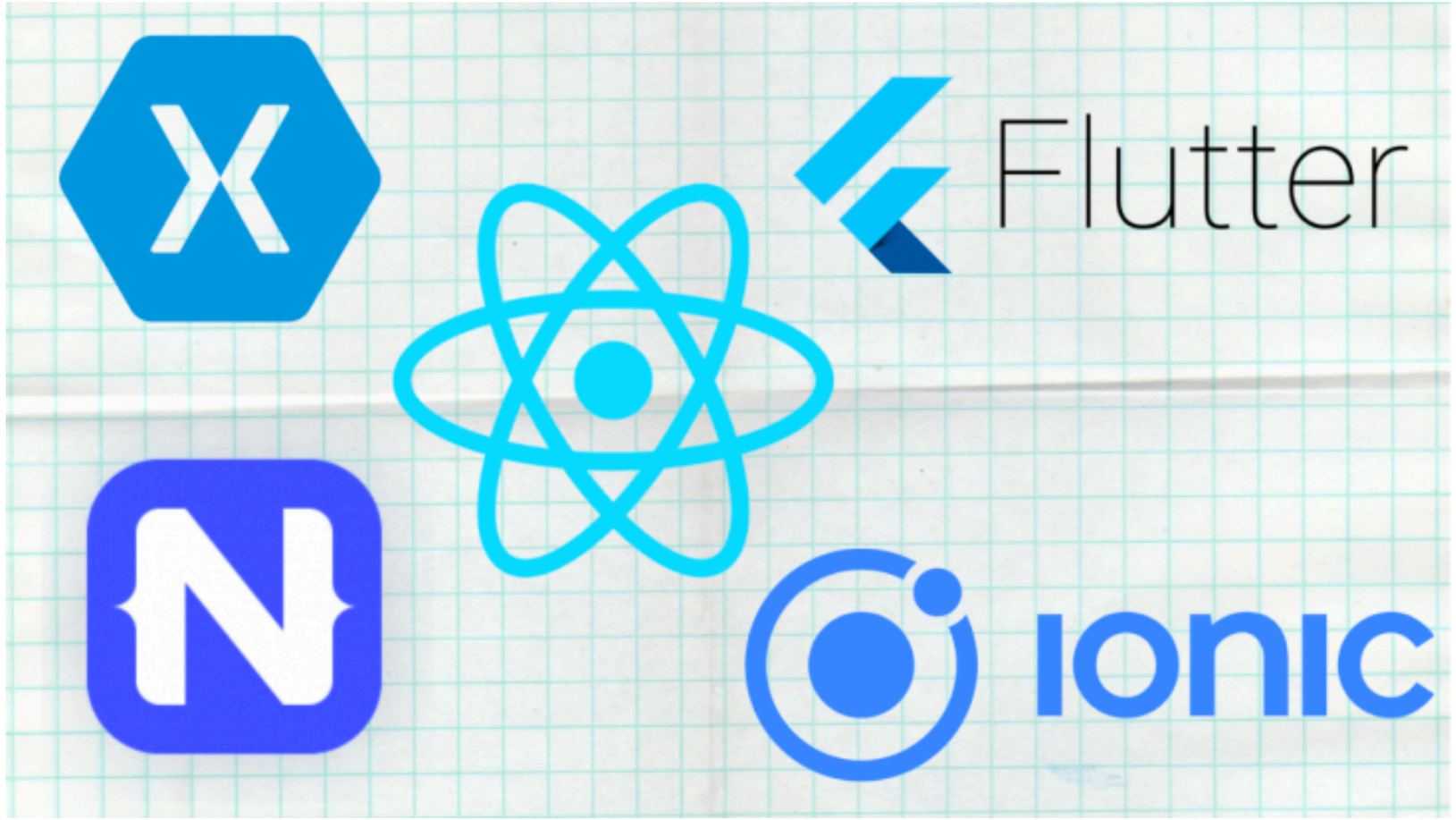
“Hybrid apps are native apps. They’re downloaded from the platform’s app store or marketplace and offer the same native features, offline support, and hardware-based performance acceleration as any app built with a native SDK.”

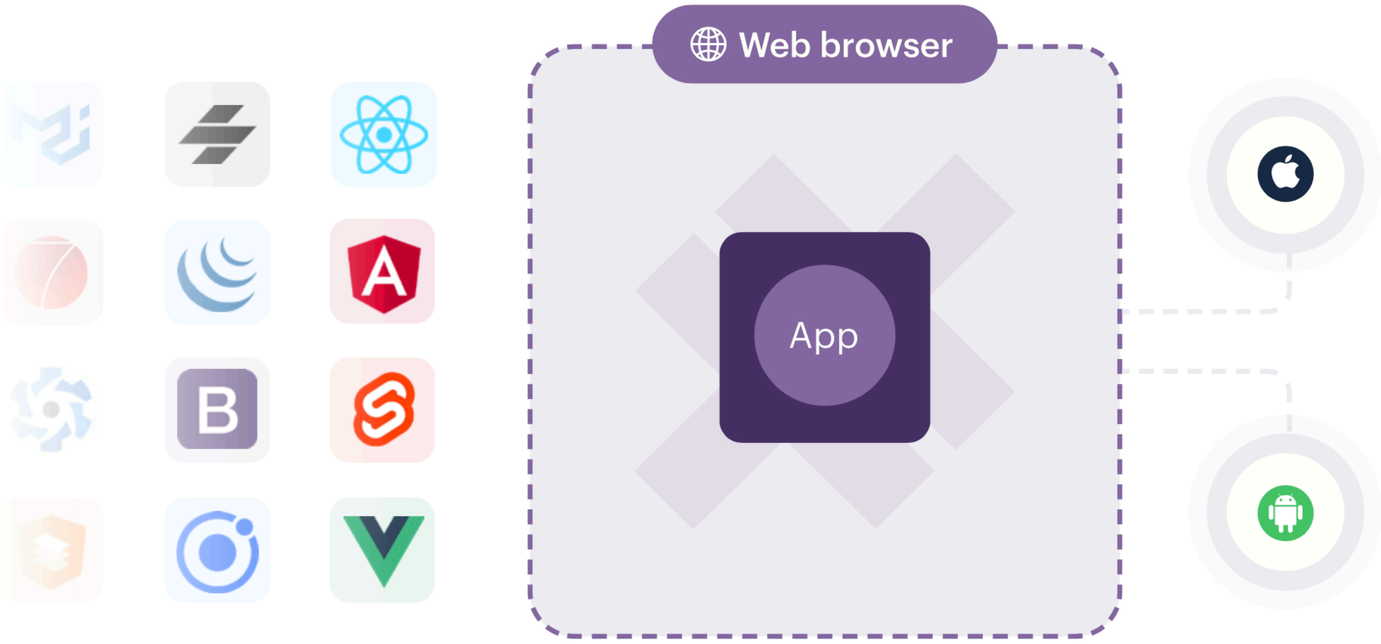
A blend - both native & web.
Build with HTML, CSS & JS (+ framework) & cross-platform like Ionic, NativeScript, Xamarin, React Native.

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
Web App
Ionic Framework
Angular JS
HTML5/JS/CSS
HTML
CSS JS
apis
Cordova
Apis
HTML
Rendering
Engine
Native Browser
Native
Apis
Compass
Camera
Network
GeoLocation
File
Accelerometer
KeyBoard
Storage
Compass
Push Notification
OS
apis
OS
apis
Mobile OS Android IOS or
Winodows
Services
Storage
Graphics
Push Notification
Input
Sensors

# Cross-platforms




Web browser
App

# Web App

Any program executed in the browser.

Stored on a web server.

Accessible anywhere through a URL.

Build with HTML, CSS & JS

# TYPES OF WEB APPS

WEB BASED

BROWSER BASED

API BASED

SERVER APPS

PORTAL APPS

SOFTWARE AS A SERVICE

PROGRESSIVE WEB APP

SPA & MPA

# 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.

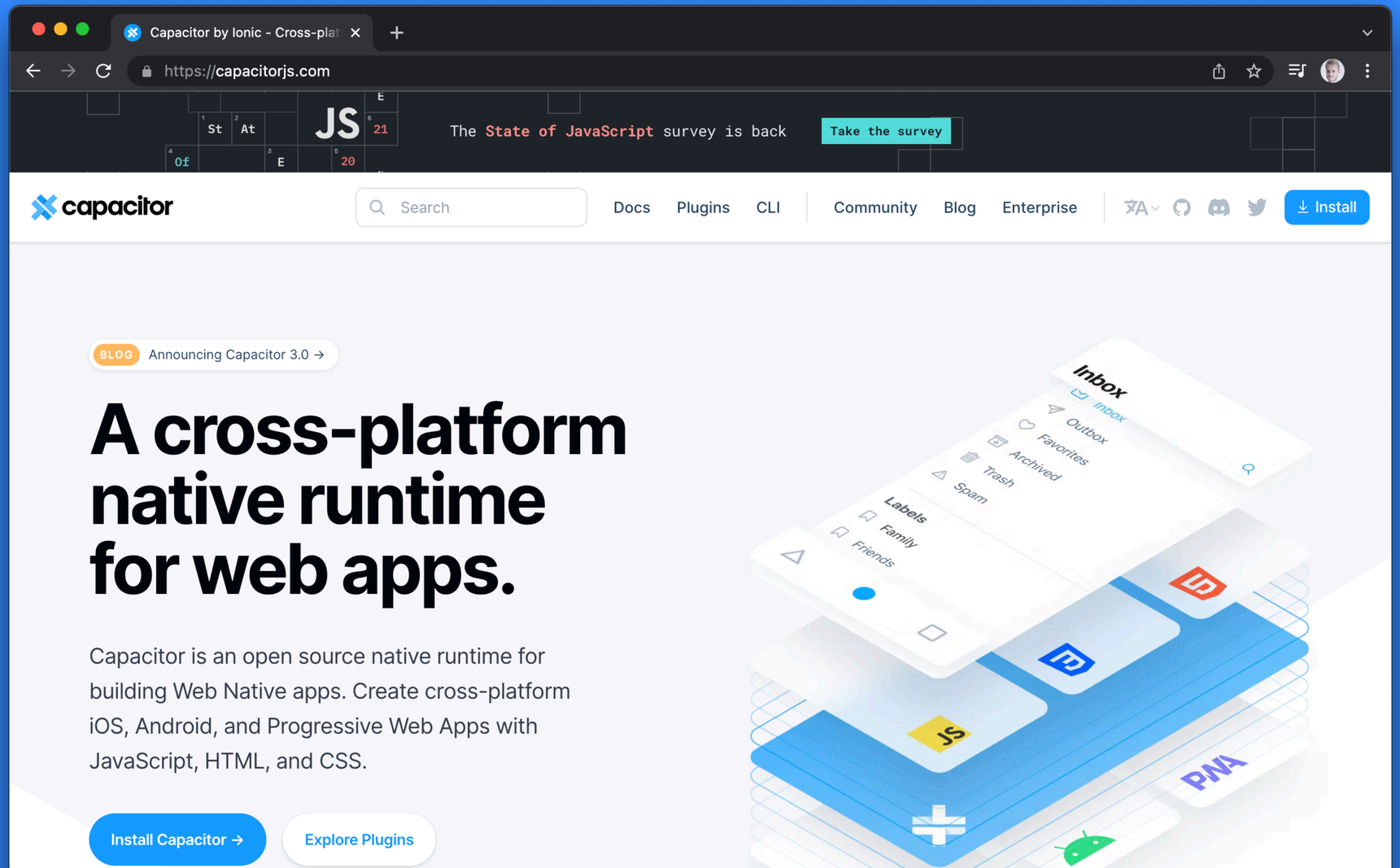
https://webnative.tech/

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

https://webnative.tech/

# Capacitor

A cross-platform native runtime for web apps.



# Back in the 2010s

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

# Today

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

https://capacitorjs.com/

# Native vs Web Native (Hybrid)

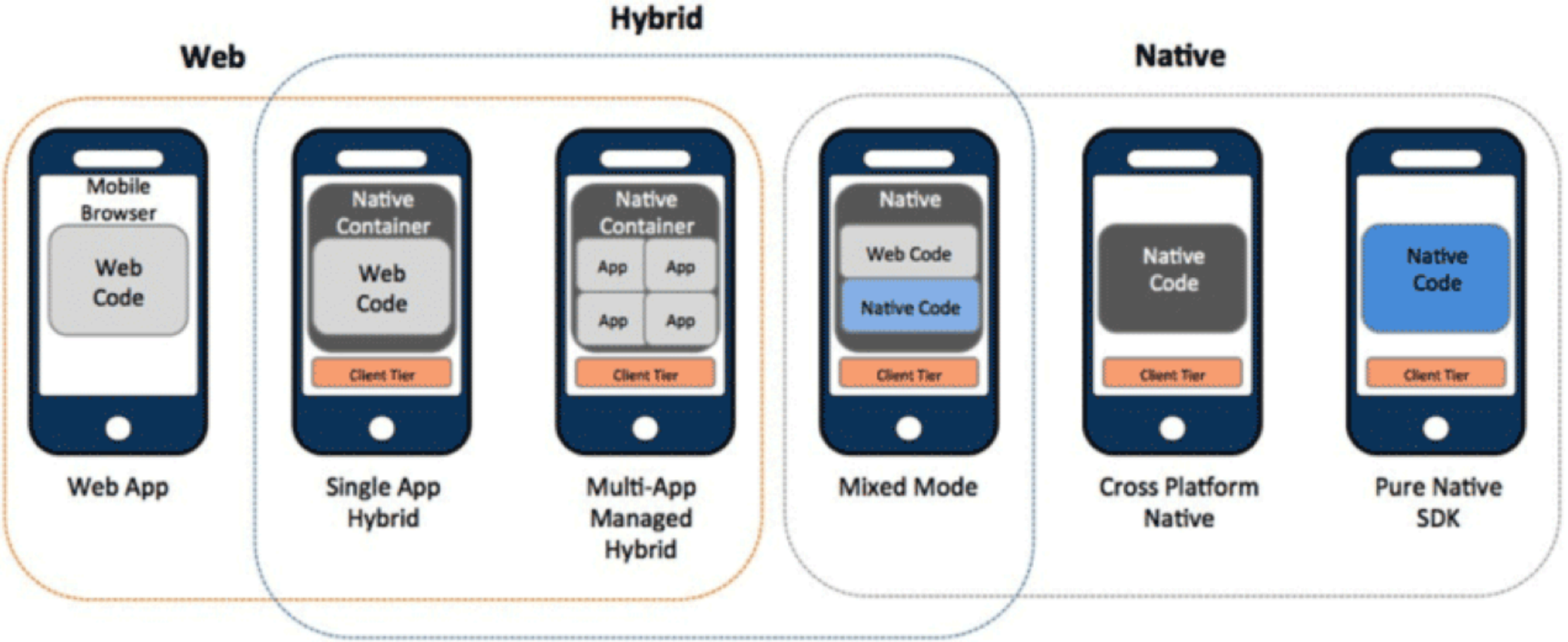
What are the differences? Pros & Cons?

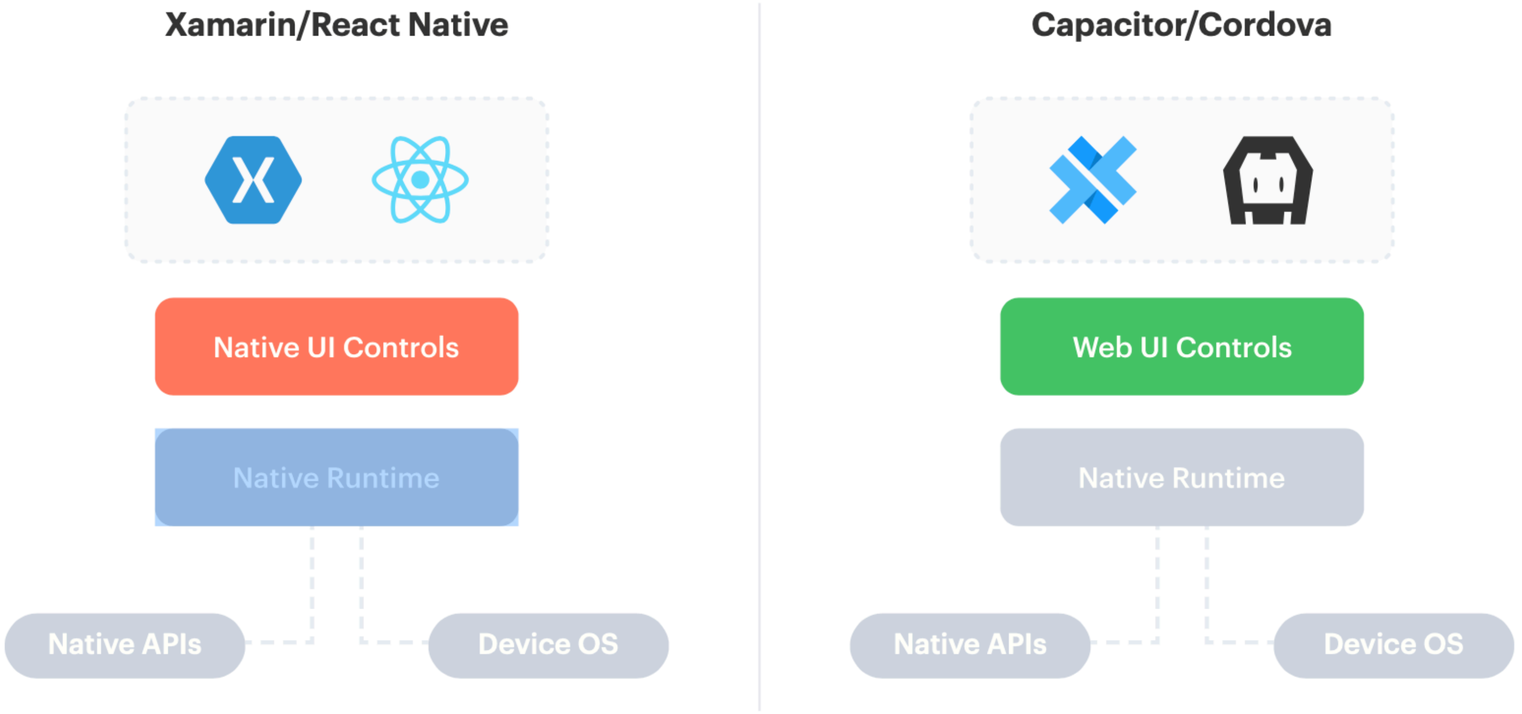
“The key difference is that hybrid apps are built using open web technologies like HTML, CSS, and JavaScript, rather than the proprietary or specialized languages used by iOS, Android, and others. That means anyone with a modern web developer skill-set can begin building an app using the hybrid approach.

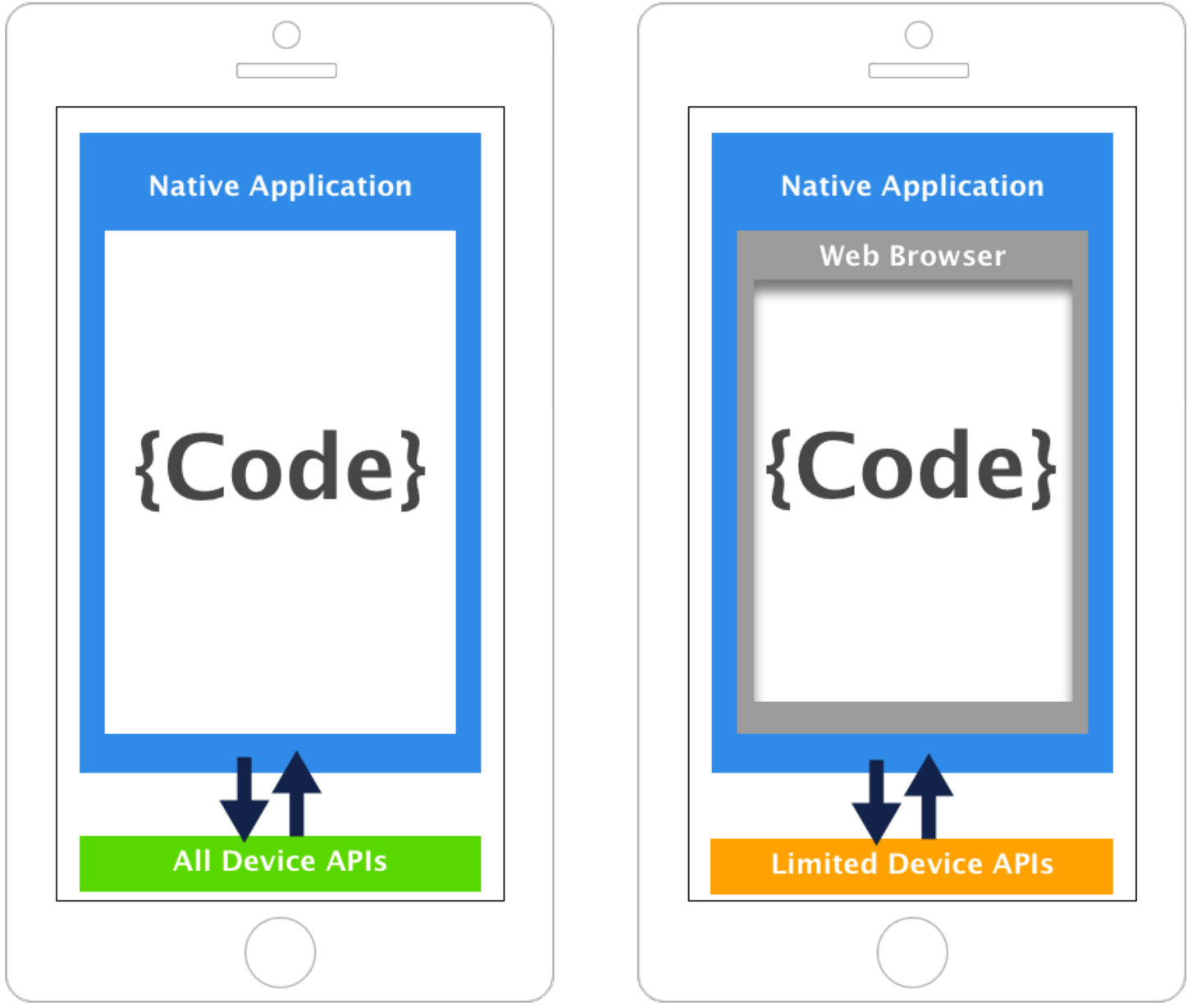
Hybrid apps run in a full-screen browser, called a webview, that is invisible to the user. Through customizable native plugins, they can access the native features of specific mobile devices (such as the camera or touch ID), without the core code being tied to that device.

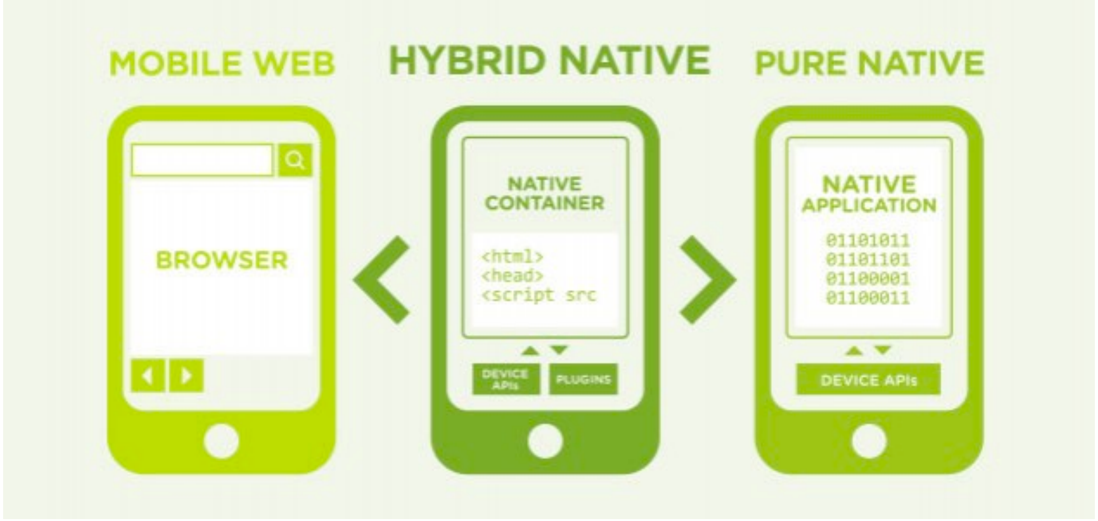
That means cross-platform hybrid applications can run on any platform or device, all from a single codebase, while still delivering native performance.“

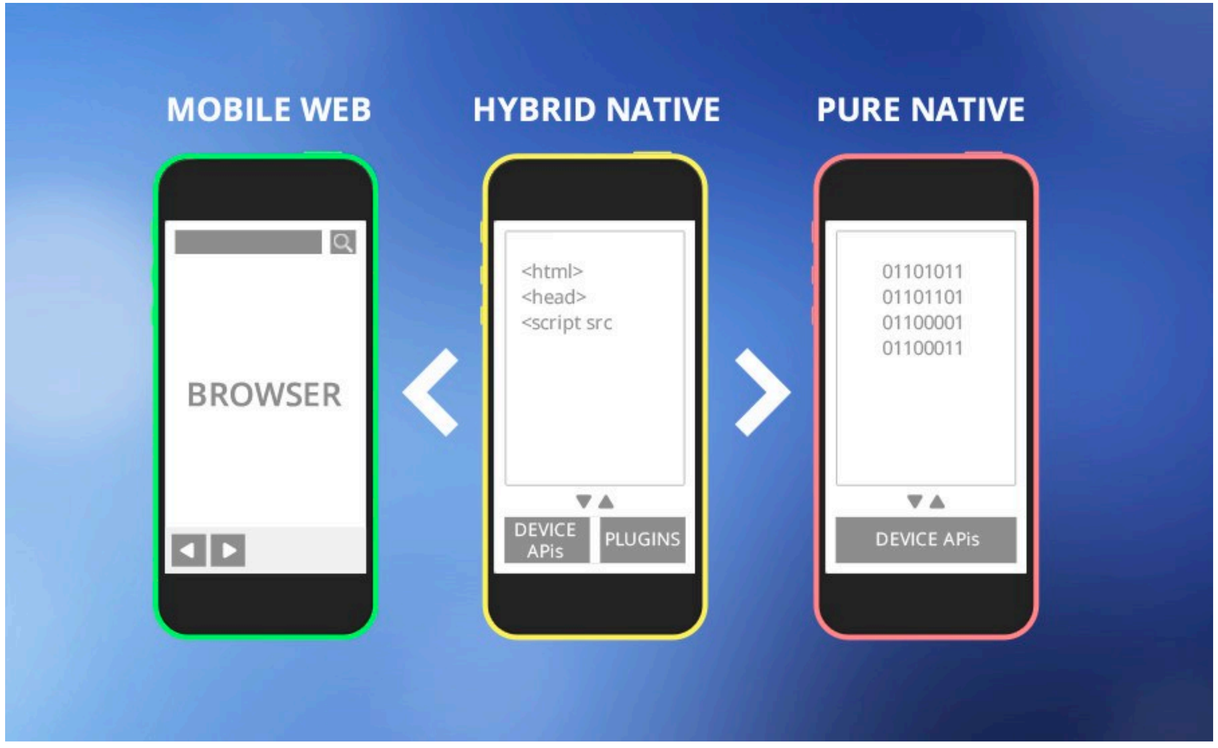

https://go.ionic.io/hybrid-vs-native-guide

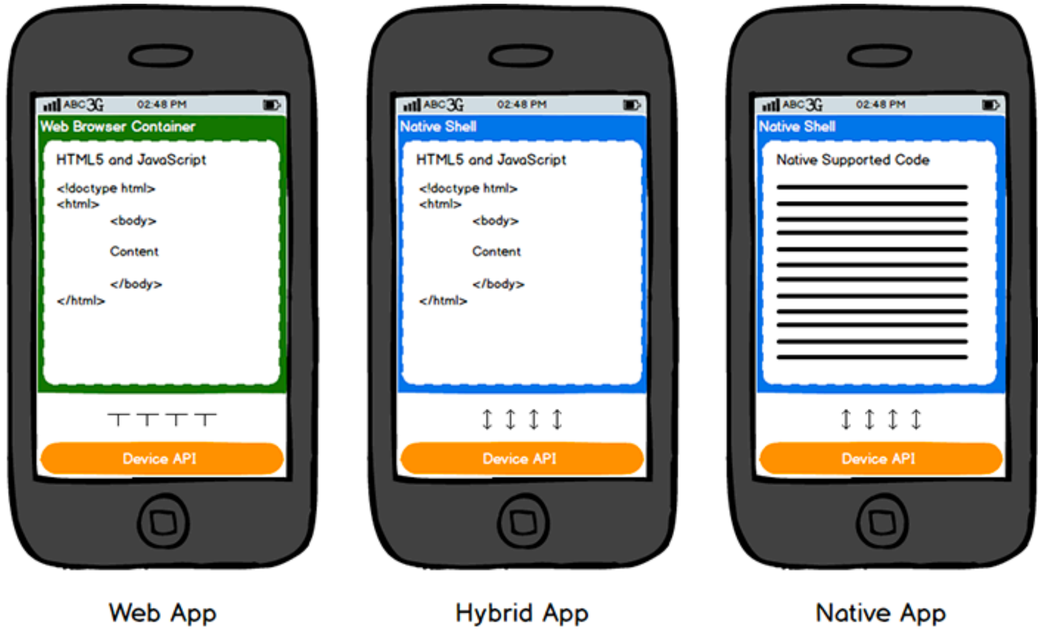

Hybrid
Web
Native
Mobile Browser
Web Code
Native Container
Web Code
Client Tier
Native Container
App
App
App
App
Client Tier
Native
Web Code
Native Code
Client Tier
Native Code
Client Tier
Native Code
Client Tier
Web App
Single App Hybrid
Multi-App Managed Hybrid
Mixed Mode
Cross Platform Native
Pure Native SDK


Xamarin/React Native
Native UI Controls
Native Runtime
Native APIs
Device OS
Capacitor/Cordova
Web UI Controls
Native Runtime
Native APIs
Device OS


Native Application
{Code}
All Device APIs
Native Application
Web Browser
{Code}
Limited Device APIs



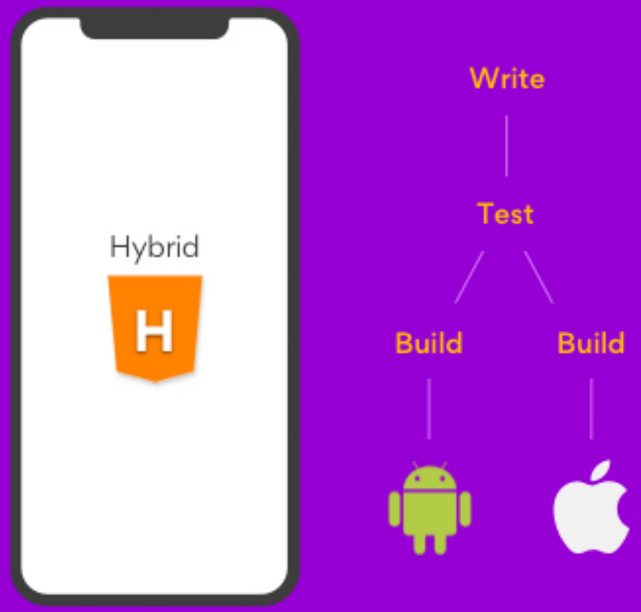




Web App

Hybrid App

Native App

## 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

## Traditional Approach


## Ionic (Web Native) Approach


ionic

Ionic is an open source UI toolkit for building performant, high-quality mobile and desktop apps using web technologies — HTML, CSS, and JavaScript — with integrations for popular frameworks like Angular, React, and Vue.

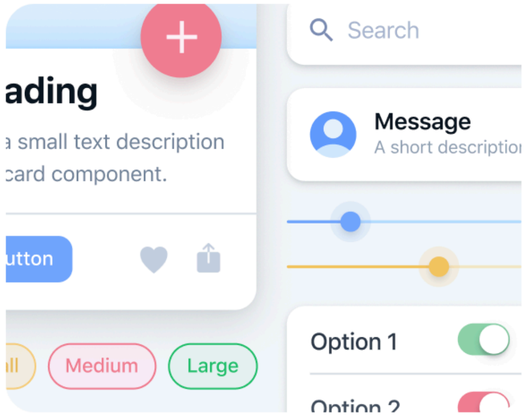
# Ionic is a Cross-platform

And framework that offers mobile-optimized UI components, gestures, and tools for build apps (iOS, Android, Web, PWA & Desktop)

# Ionic is a Cross-platform

Yes, Hybrid (or Web Native)

Component library for building apps that run on iOS, Android, Electron, and the Web.




## Pre-designed UI components

Ionic's UI components look great on all mobile devices and platforms. Start with pre-made components, typography, and a base theme that adapts to each platform.

## Write once, run anywhere

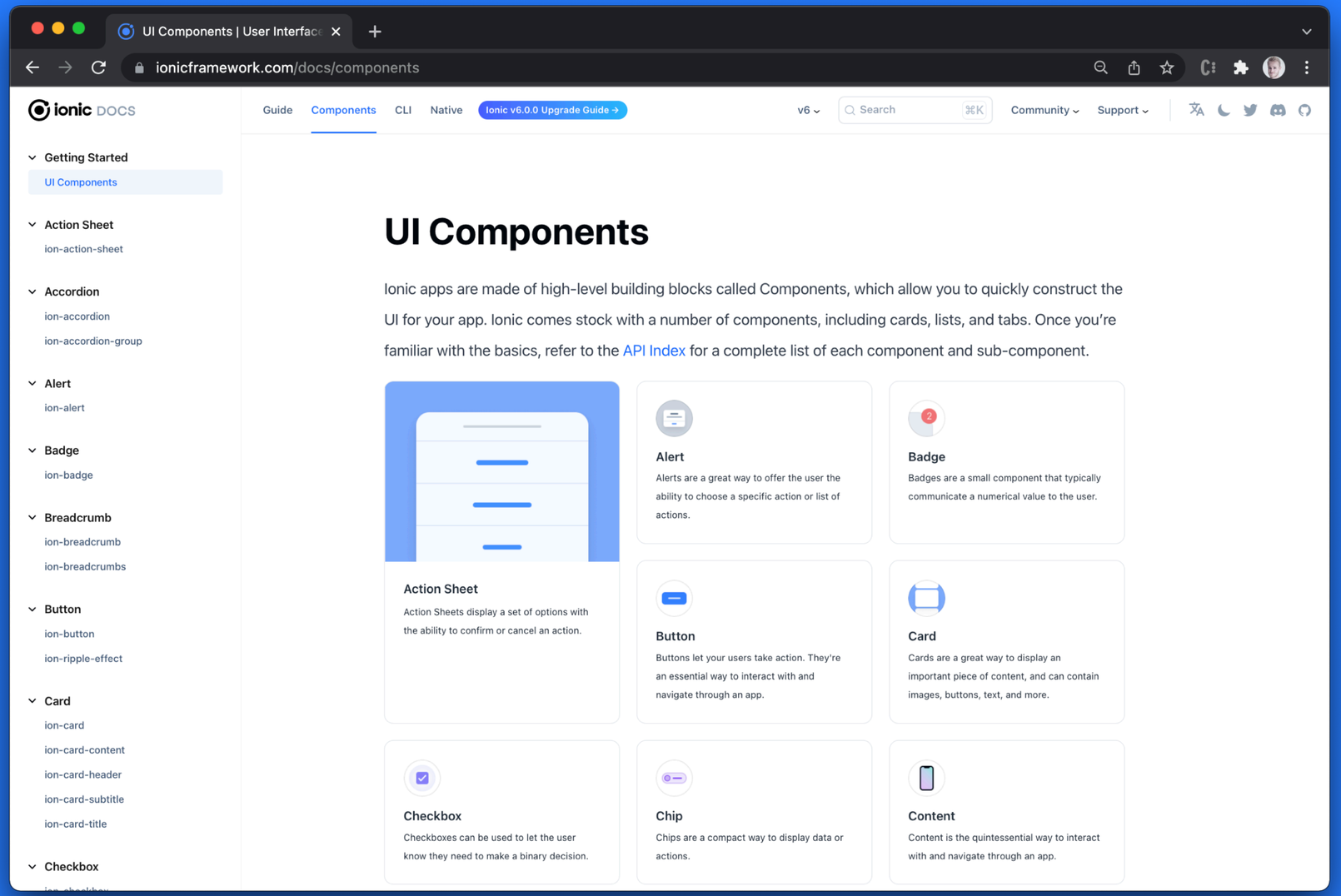
Ionic lets developers to ship apps to the app stores and as a PWA with a single code base. With Adaptive Styling, apps look and feel at home on every platform.

```
$ ionic start
? Project name: My App
? Starter template: tabs
Success!
ionic run ios
Compiled successfully.
Deployed to iOS!
```

## Developer-friendly tooling

Create, build, test, and deploy your app with the Ionic CLI. Take advantage of Live Reload, deployments, integrations, and even use your favorite JS framework's CLI.

# UI Components



# Build with whatever you prefer

Angular, React, Vue, or Vanilla JavaScript

# Ionic Framework

### Installation Guide

Step-by-step guides to setting up your system and installing the framework.

### UI Components

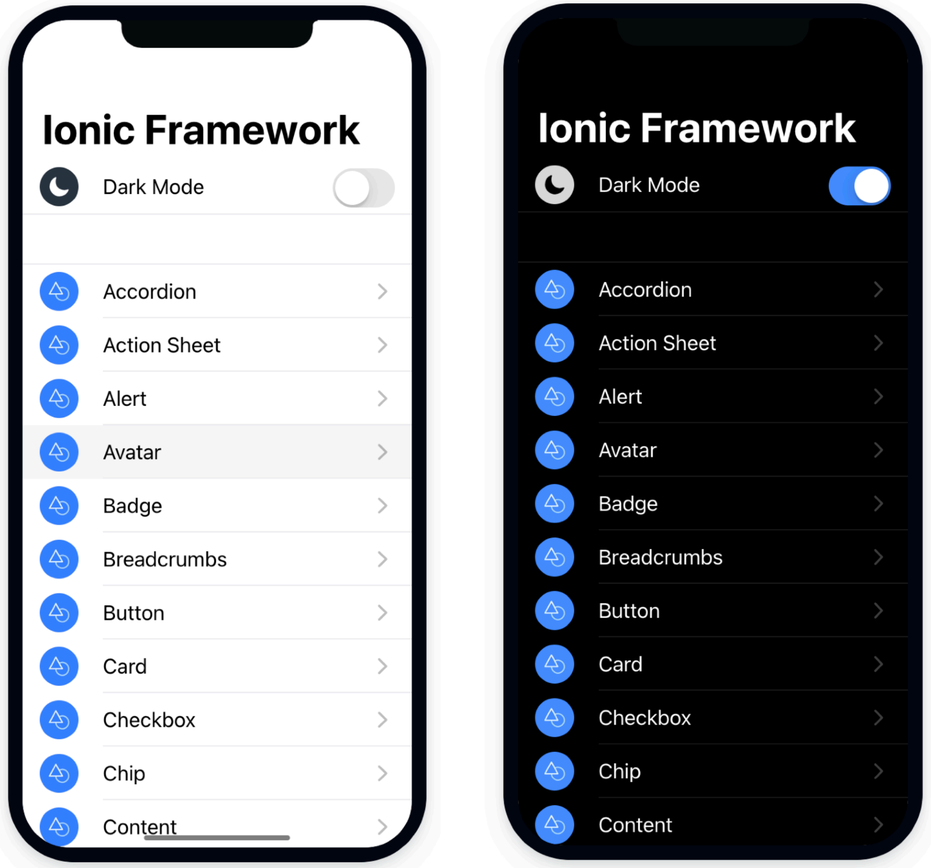
Dive into Ionic beautifully designed UI component library.

### Native Functionality

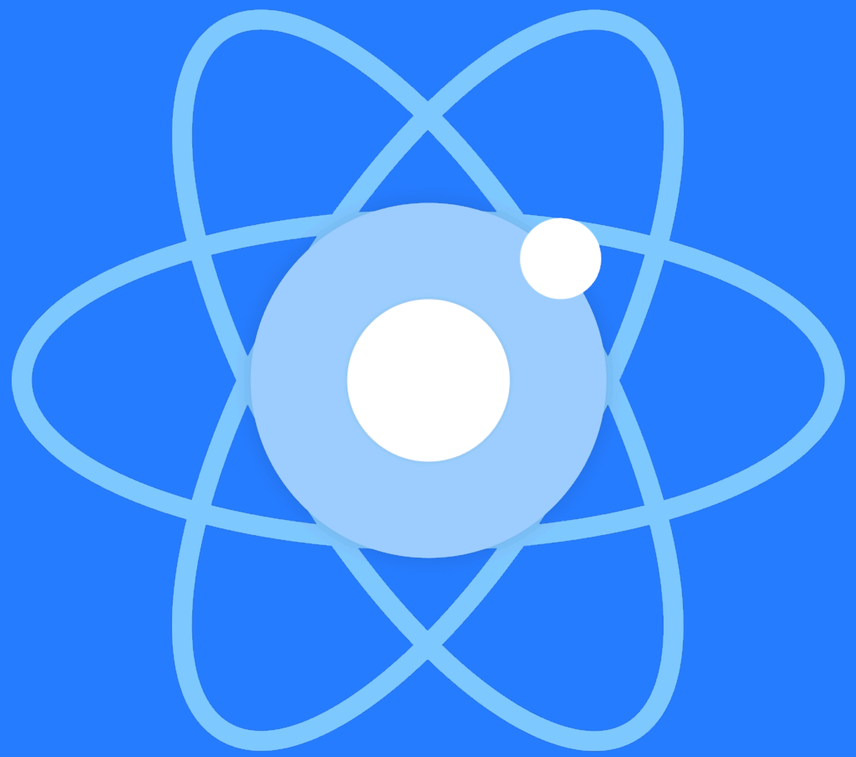
Integrate native device plugins, like Bluetooth, Maps, HealthKit, and more.

### Theming

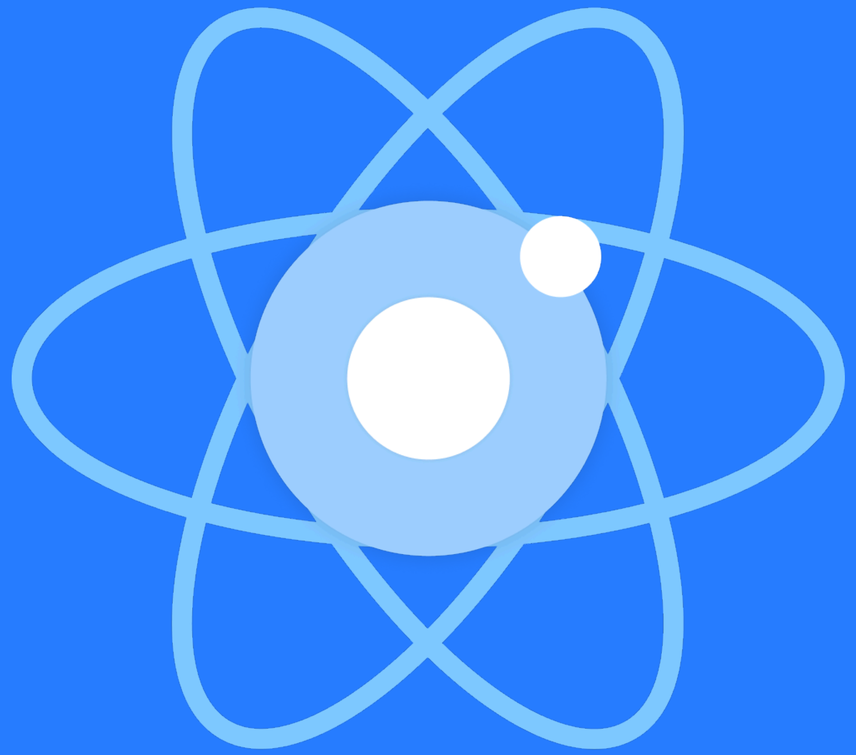
Learn to easily customize and modify your Ionic app's visual design to fit your brand.




https://ionicframework.com/

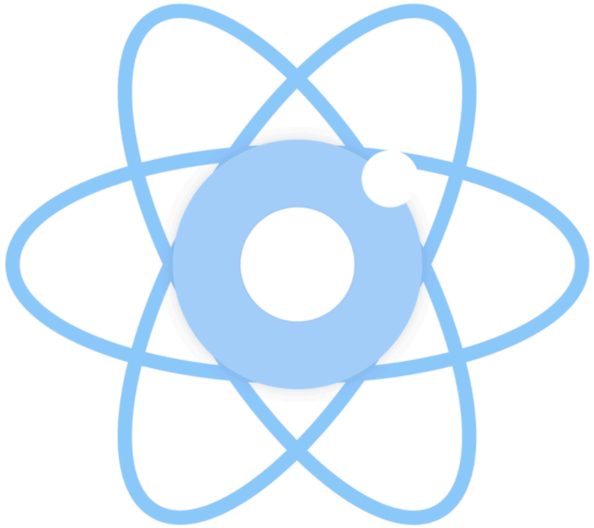


# Ionic React



Build Native and Progressive Web Apps from a single code base with Ionic React.

# One Codebase
# Any Platform
# Just React



✓ 100+ mobile optimized React UI components

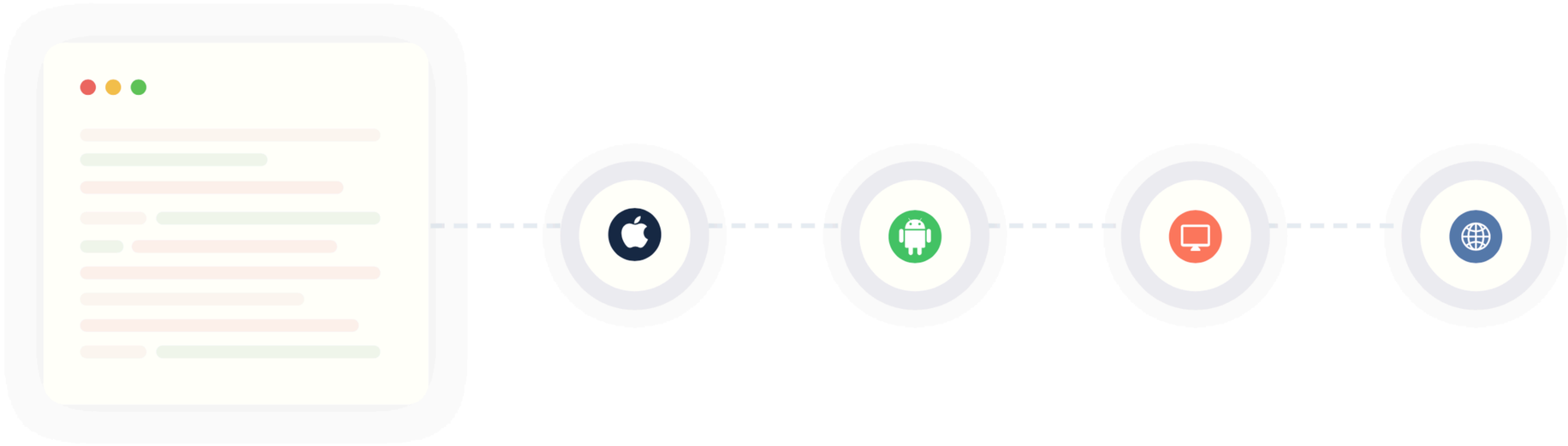
✓ Standard React tooling with react-dom

✓ iOS / Android / Electron / PWA

# Why Ionic?

you
are
web developers!

# Seemless experience across platforms



CARD SUBTITLE

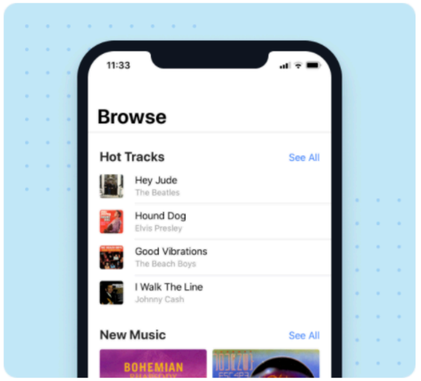
## Card Title

Here's a small text description for the card content. Nothing more, nothing less.

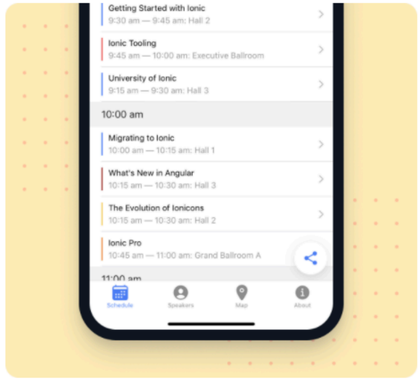
Action

# Predefined UI Components

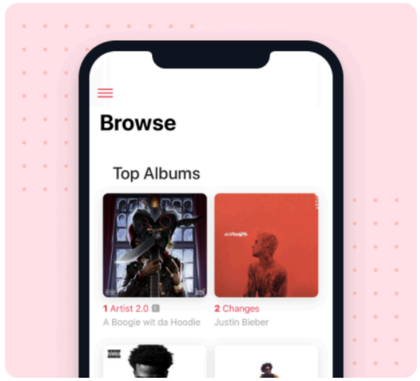
Awesome docs with many examples and sample apps




Music Player

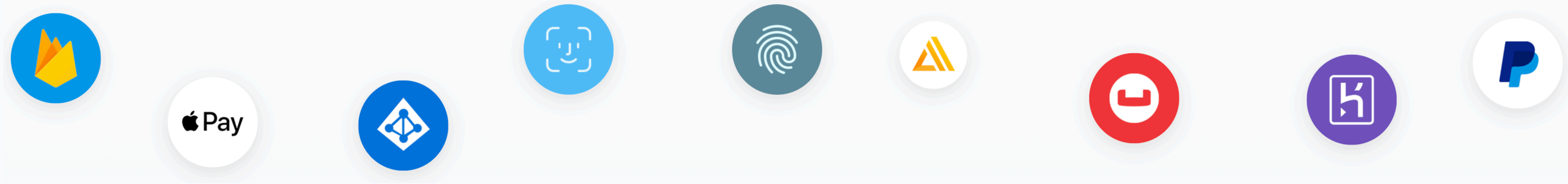



Conference App




StarTrack

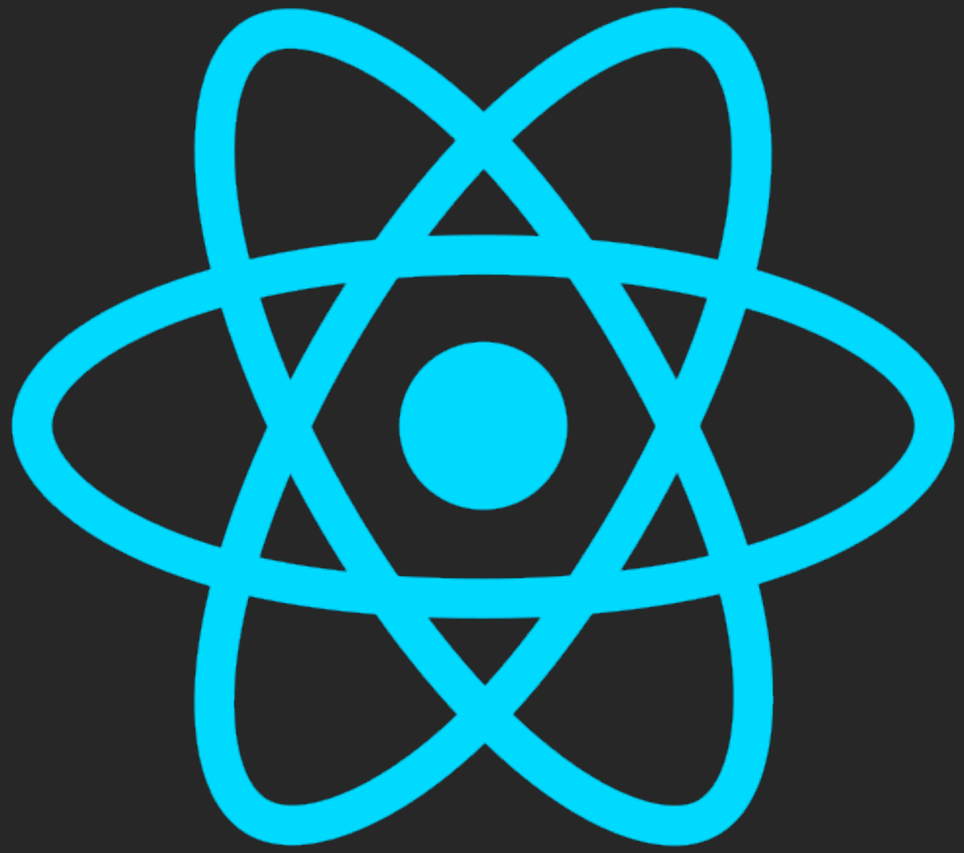
# Integrations



Integrate with any backend tech

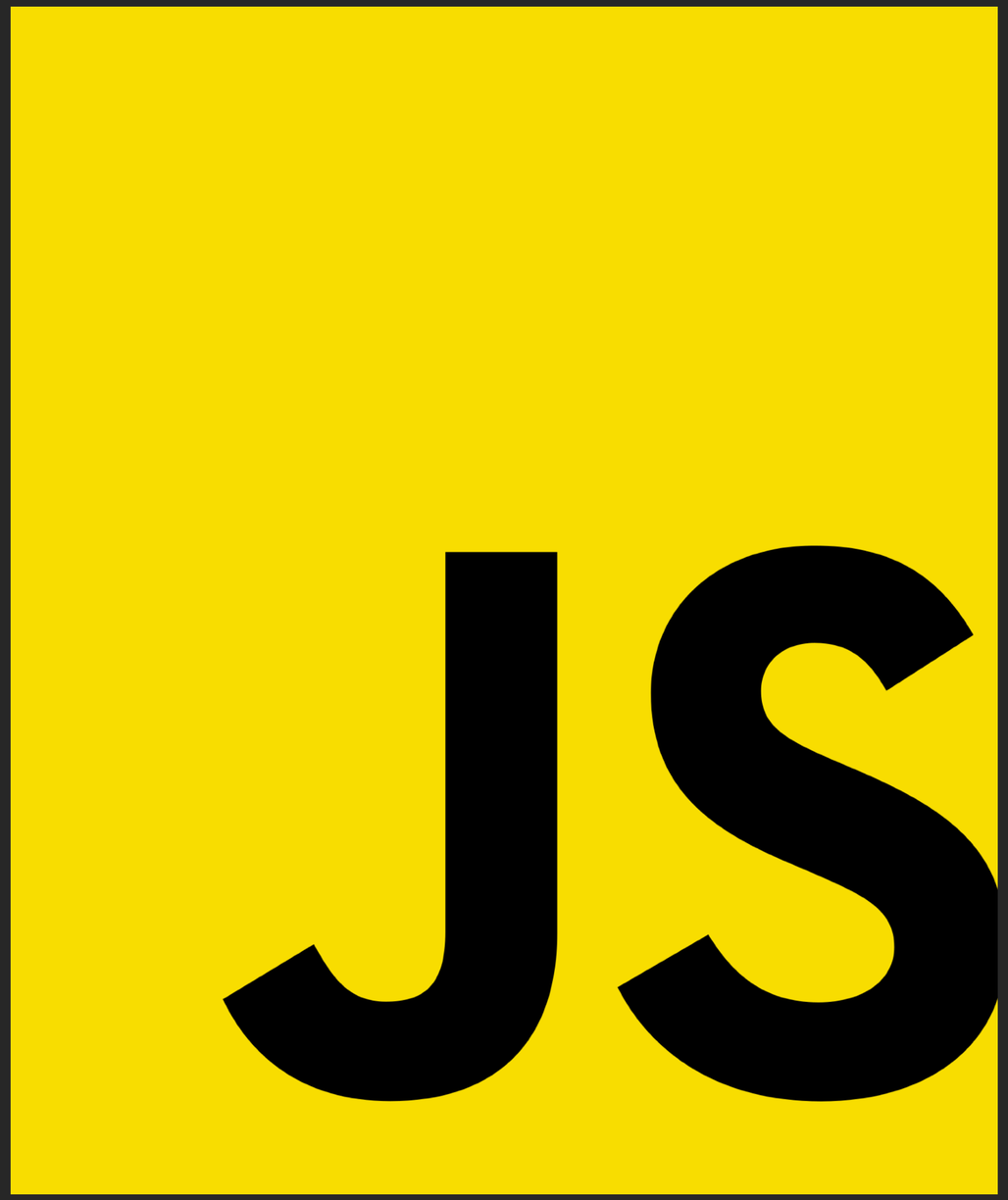

Use your front-end flavor of choice


Deploy your app anywhere



# It’s Just React

Ionic React is just React components

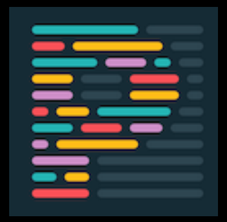


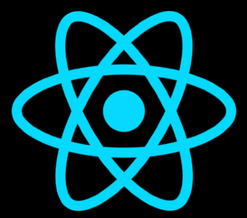

Well, at the end of the day, it is all just JavaScript

🤷

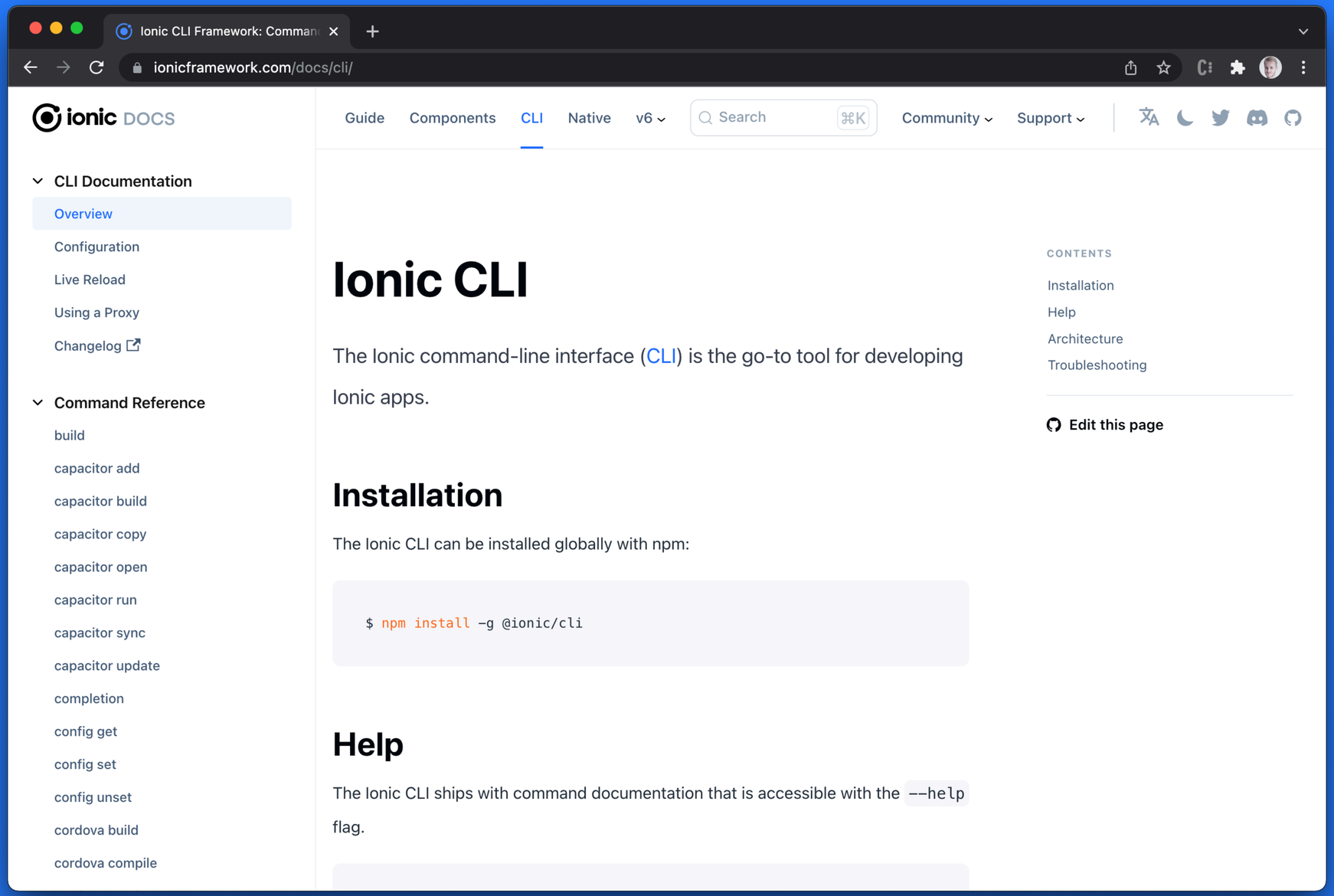
# CLI & Tools







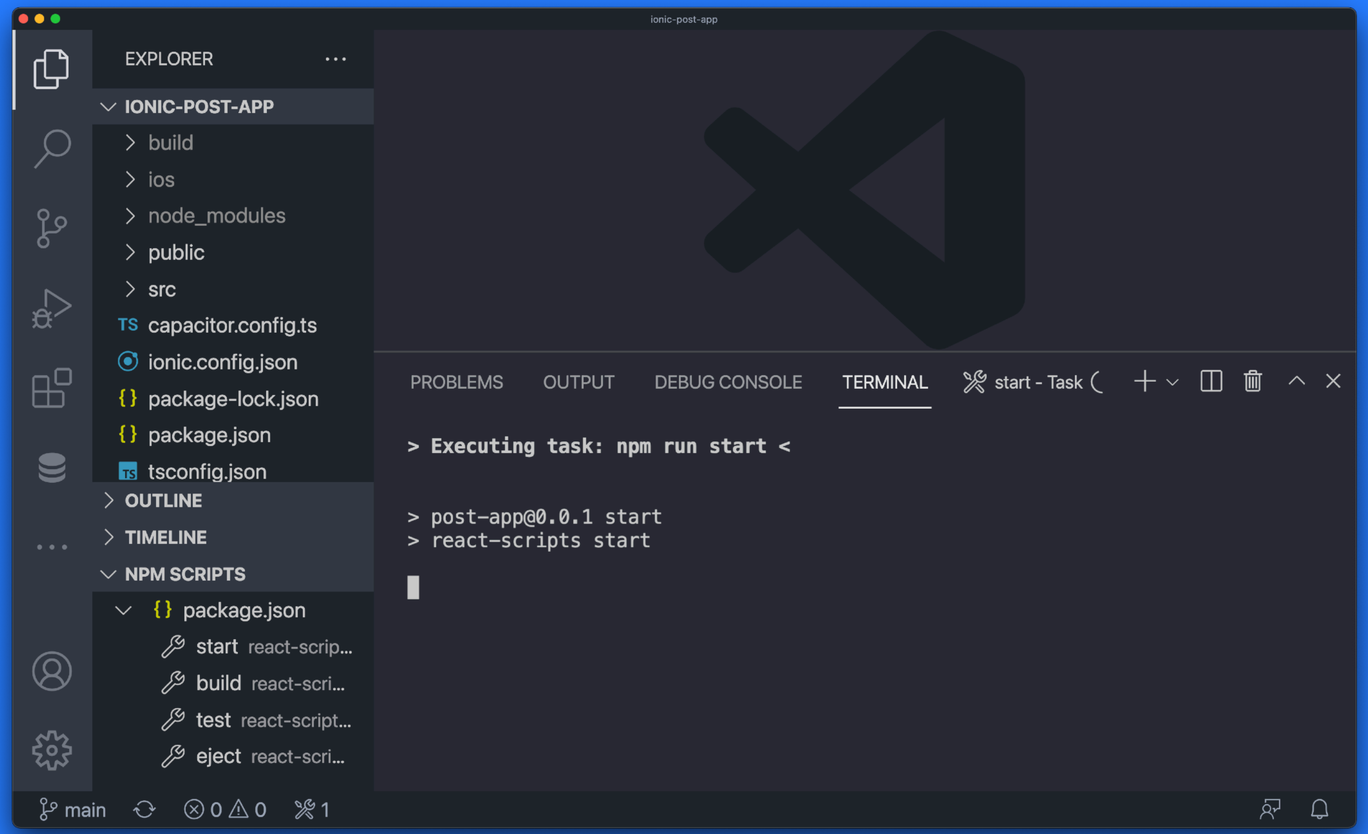


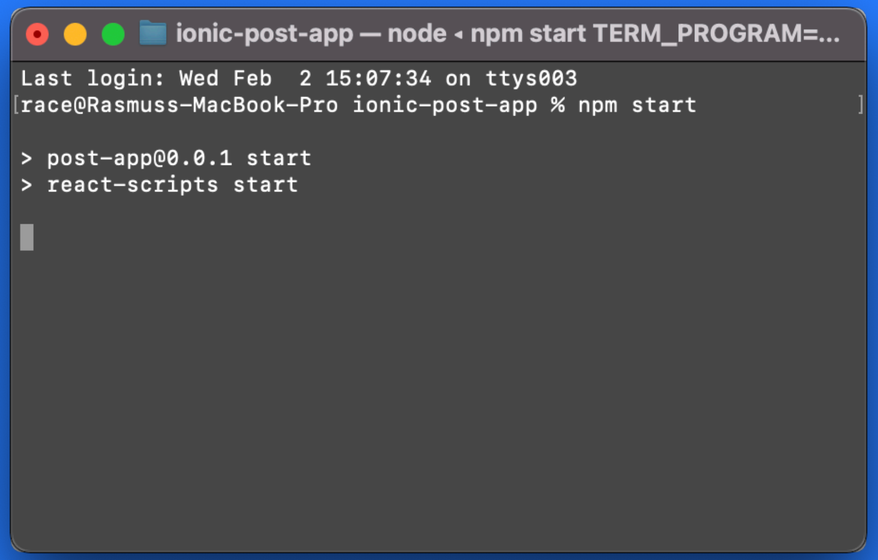



https://ionicframework.com/docs/cli/

# CLI

https://ionicframework.com/docs/intro/cli



# CLI Installation

https://ionicframework.com/docs/intro/cli

# Ionic React Quickstart

https://ionicframework.com/docs/react/quickstart

# Getting Started

https://race.notion.site/Getting-Started-5086f9c2cb684f34b4cd259b4b0c44f0

# Structure

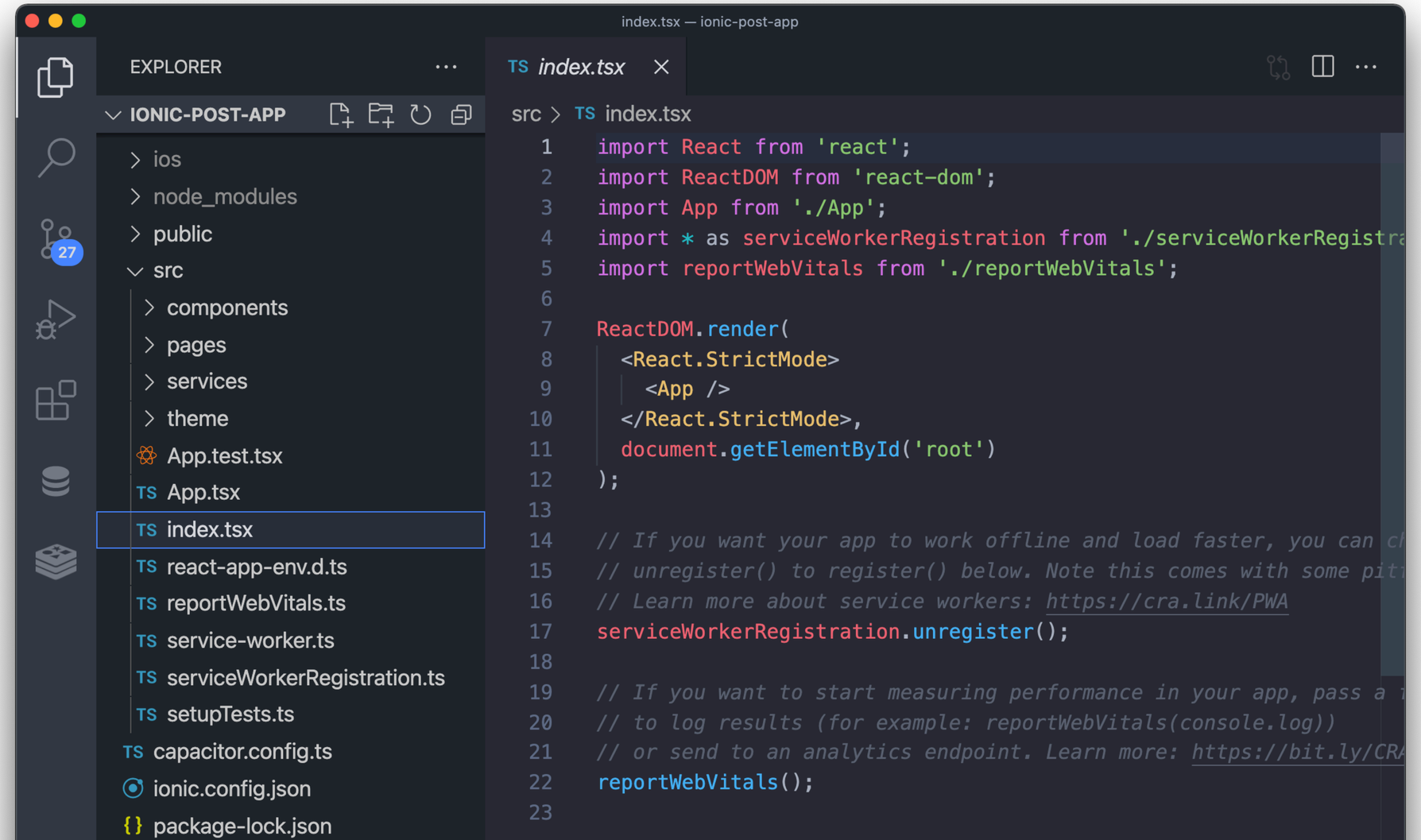
## It’s all components

```tsx
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

```tsx
import React from 'react';
import { Route } from 'react-router-dom';
import { IonApp, IonRouterOutlet } from '@ionic/react';
import { IonReactRouter } from '@ionic/react-router';
import Home from './pages/Home';

/* Core CSS required for Ionic components to work properly */
import '@ionic/react/css/core.css';

const App: React.FC = () => (
  <IonApp>
    <IonReactRouter>
      <IonRouterOutlet>
        <Route path="/home" component={Home} exact={true} />
        <Route exact path="/" render={() => <Redirect to="/home" />} />
      </IonRouterOutlet>
    </IonReactRouter>
  </IonApp>
```

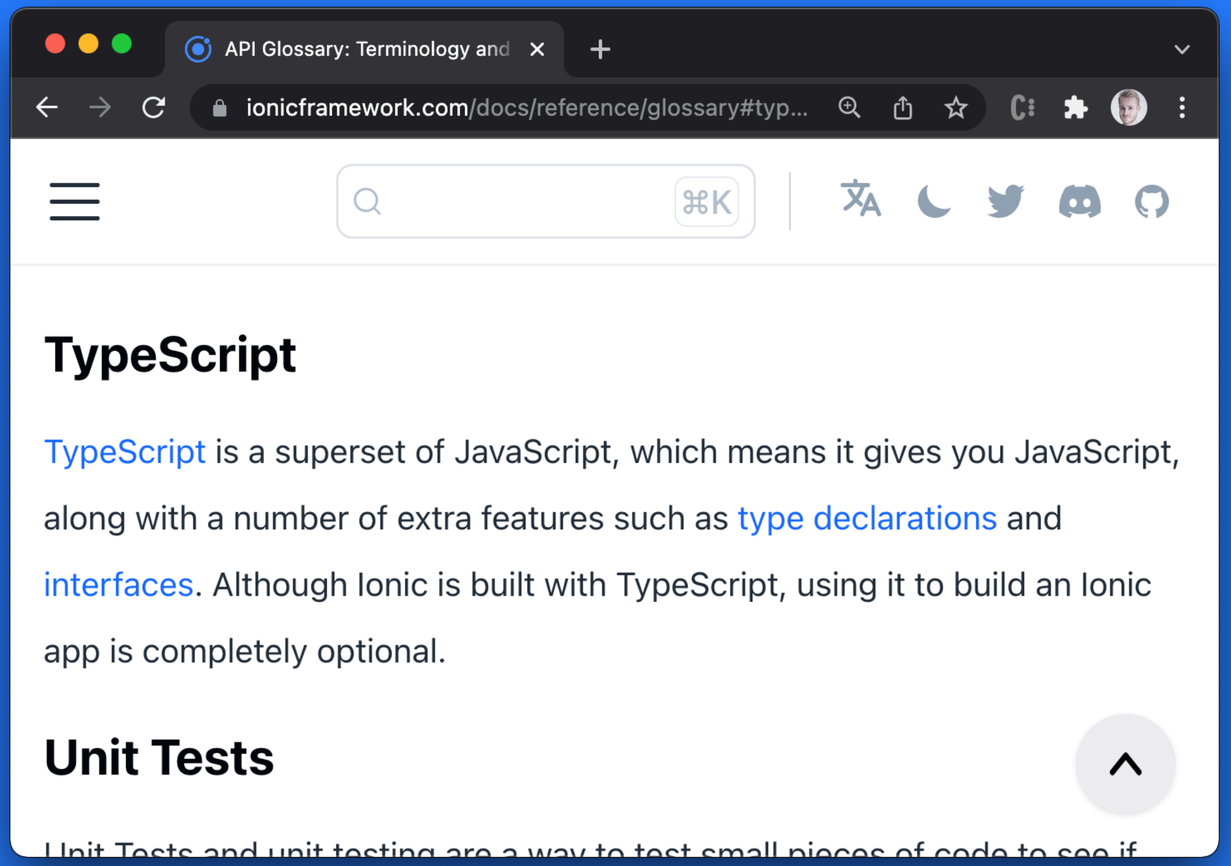
# Structure

## It’s all components



# .tsx?

What the ... TypeScript 🤷




https://ionicframework.com/docs/reference/glossary#typescript

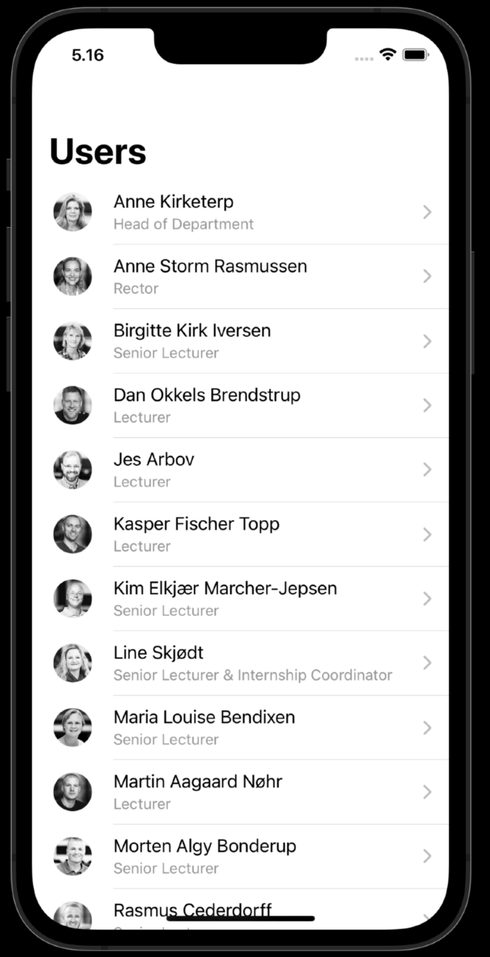
# TypeScript is **JavaScript with syntax for types.**

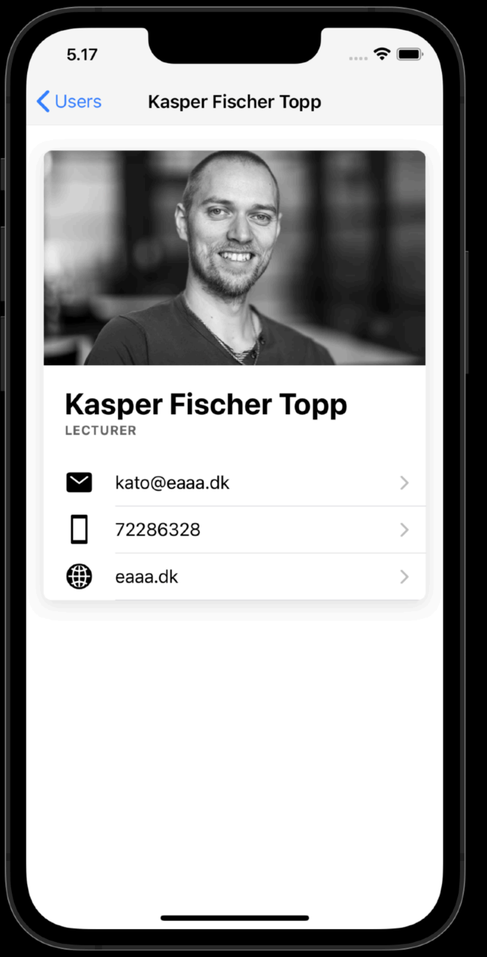
TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

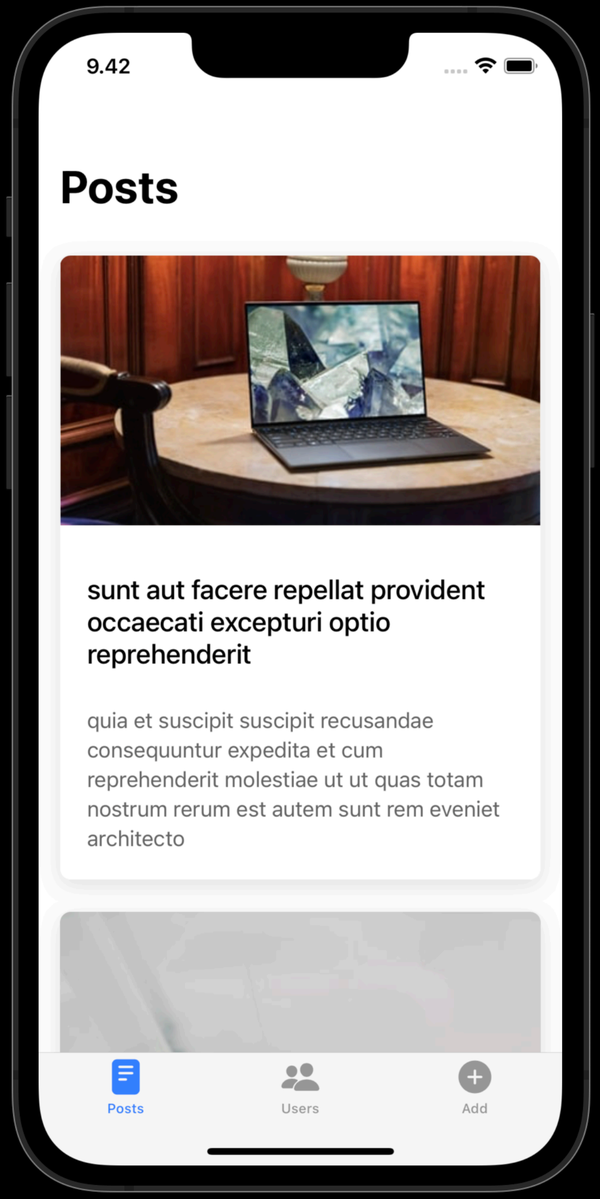
https://www.typescriptlang.org/

# Ionic User List App

https://race.notion.site/User-List-e4d5af63b2de443f946cc2f5a1227716





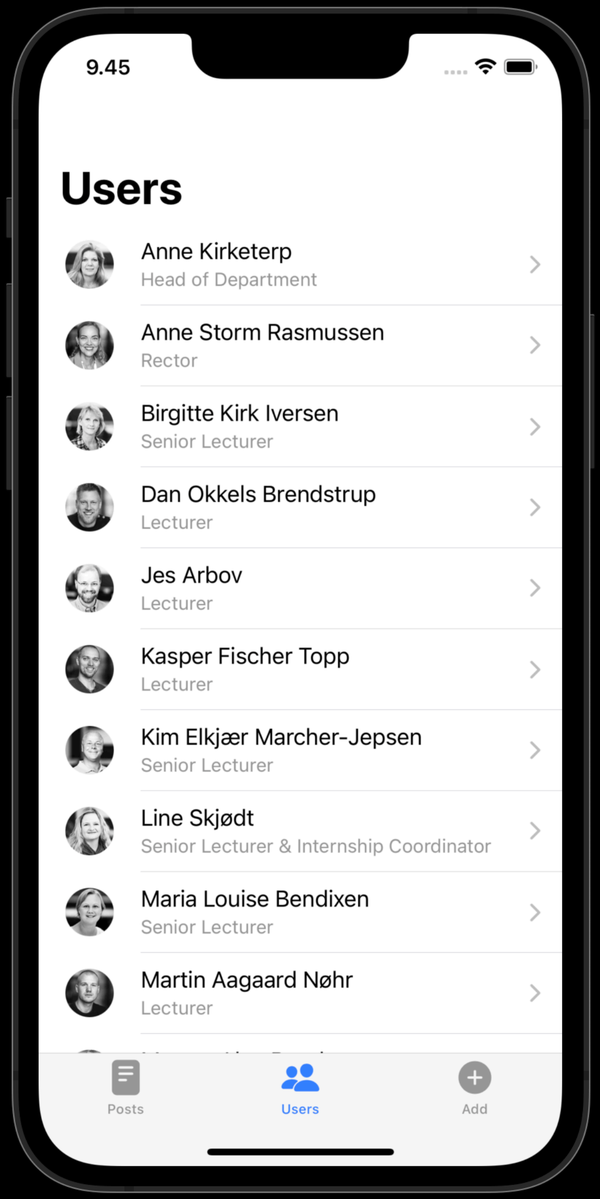

# Ionic Post App

1. Use `tabs` starter template.

2. Customise the template with Posts Page, Users Page and Add Page.

3. In the Posts tab, fetch and display posts from this source.

4. Display image, title and body property using Ionic UI Components.




# Ionic Post App

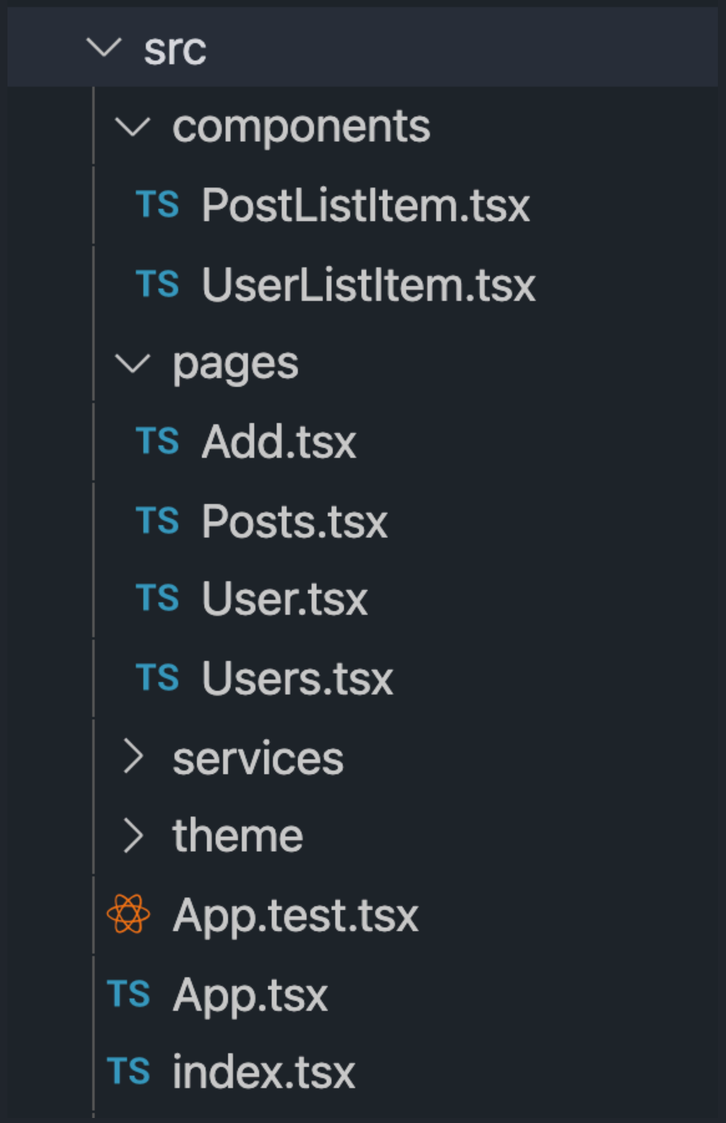
5. Add the list of users in the second tab, Users Page.

6. Make sure you can navigate to the detail view.

# Ionic Post App

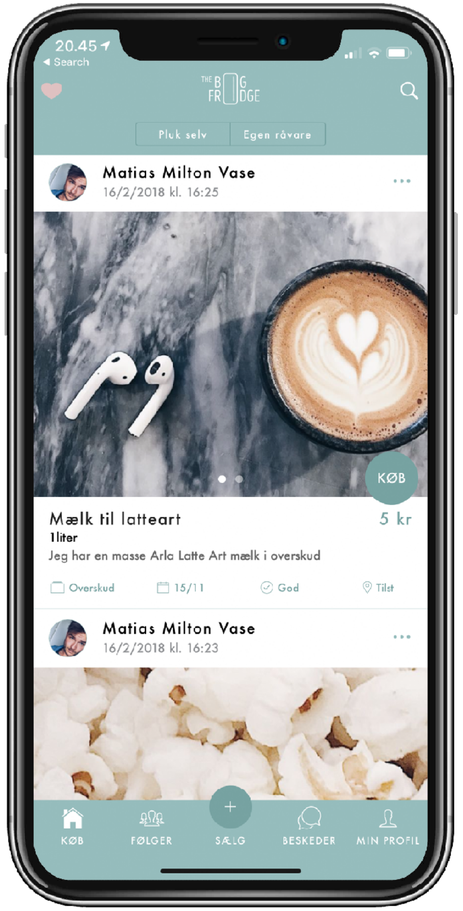
7. Think in components. Is there any logic that might be useful to extract to separate components?

8. Reimplement your pages using components.

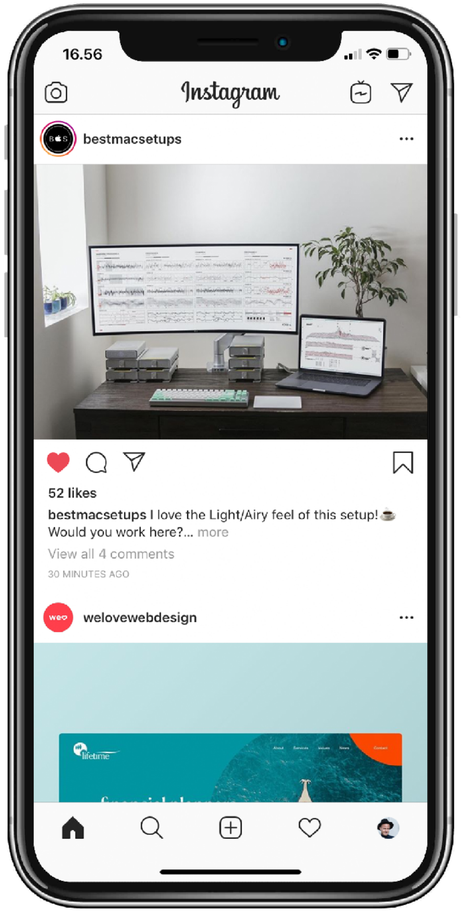


# Display User

## ... on given post (uid)





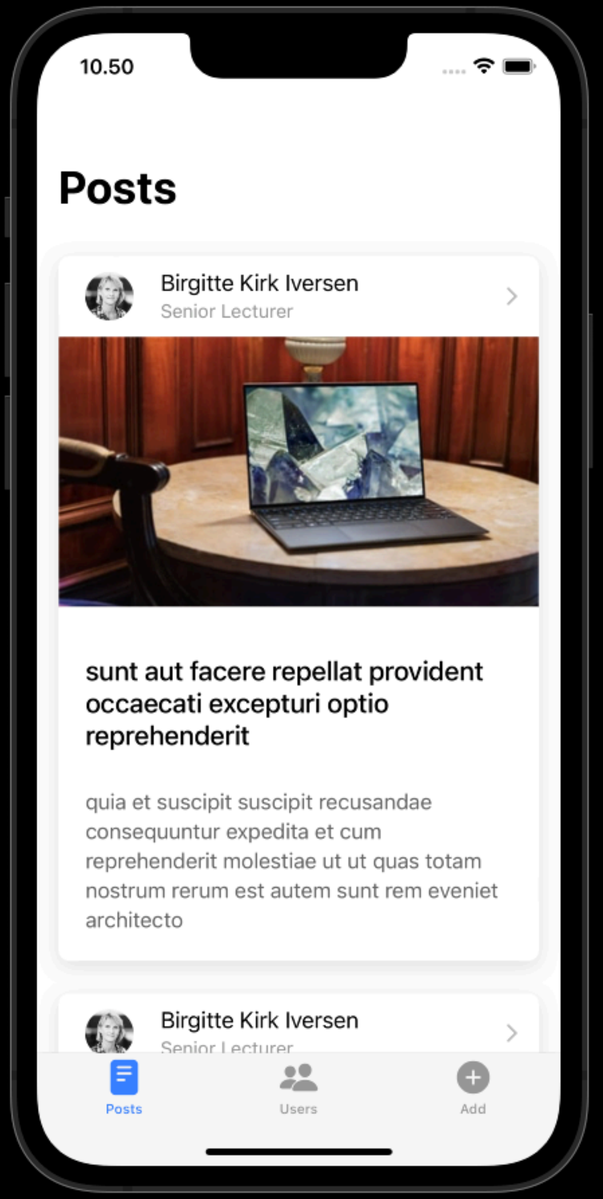



# Ionic Post App

9. Improve your post view by adding user details (post create by...). Start by adding hardcoded user details like user avatar and user name using Ionic UI Components.

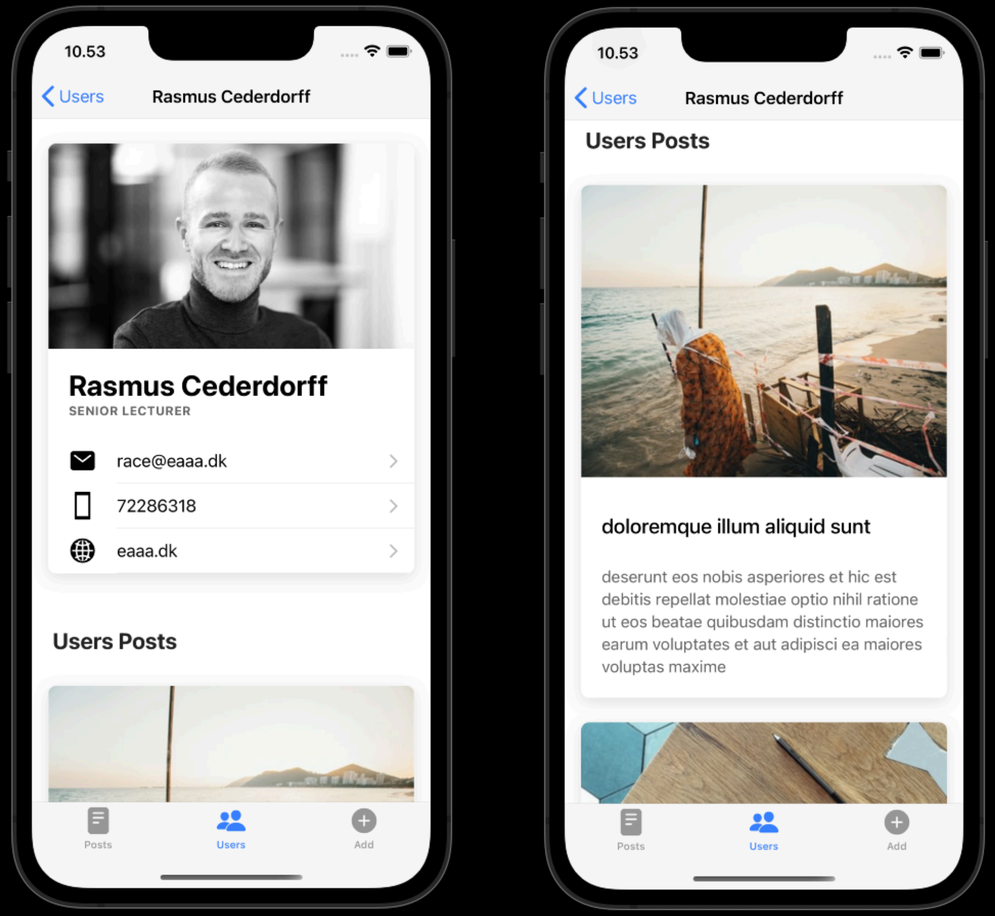
10. Implement logic to concatenate the posts array with user data from users array. All post objects have a `uid` property.

11. Use the concatenated array to display posts with user details dynamically.



# Combine posts with users

```
import userService from "./usersService";

class PostService {
    constructor() {…
    }

    async fetchPosts() {…
    }

    async getPosts() {…
    }

    async getPostsWithUserDetails() {
        if (this.posts.length === 0) {
            await this.fetchPosts();
        }
        const users = await userService.getUsers();

        const postsWithUser = this.posts.map(post => {
            const user = users.find(user => user.id === post.uid);
            post = { ...post, user: user }; // combine objects with spread operator
            delete post.uid; // delete uid - it's inside post.user.id
            return post;
        });
        return postsWithUser;
    }
```




# Ionic Post App

12. Improve the user detail view by displaying all posts created by the user. Again, think of the `uid` property on every post object and filter posts by given `uid`.

13. Components, components, components. When implementing improvements, think of how to (re-)use your existing components. No code duplicating is allowed.

# Ionic Post App

14. Implement the missing tab, Add.

15. Explore Ionic UI Components.

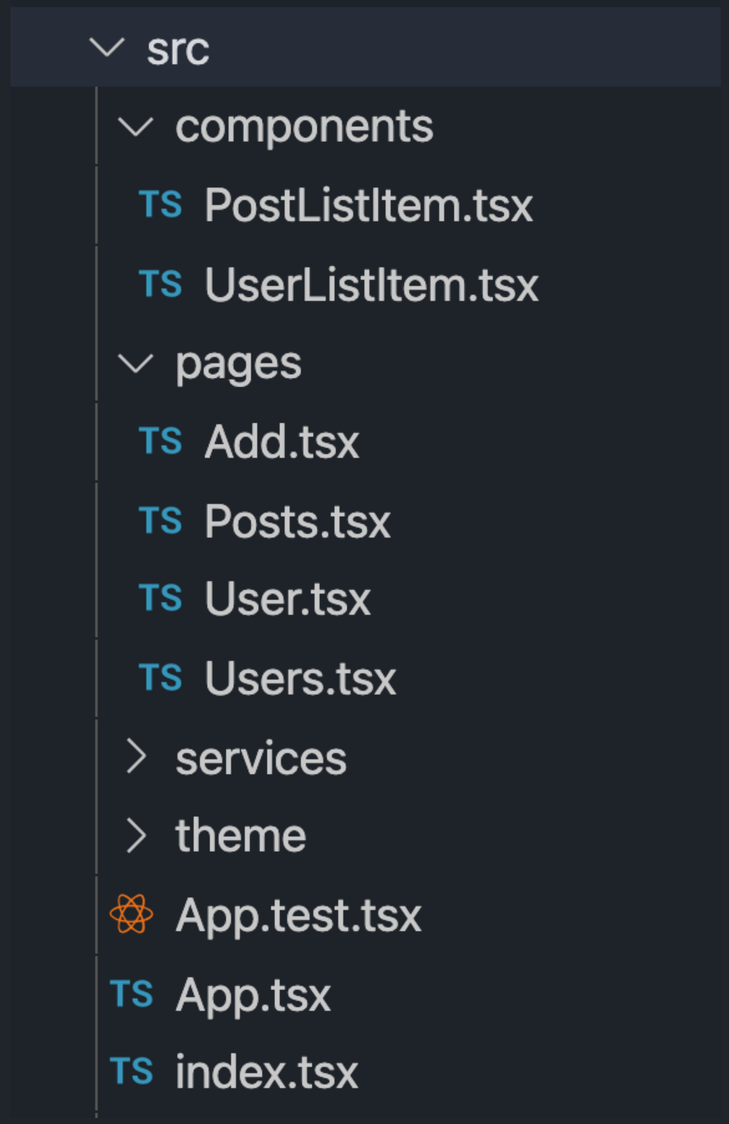
16. Implement the Add tab with UI Components to create a new post object with the following properties: `uid`, `id`, `title`, `body` & `image`.

`uid` can be hardcoded. `image` can be a URL.

17. The new post object must be pushed to the array of posts.

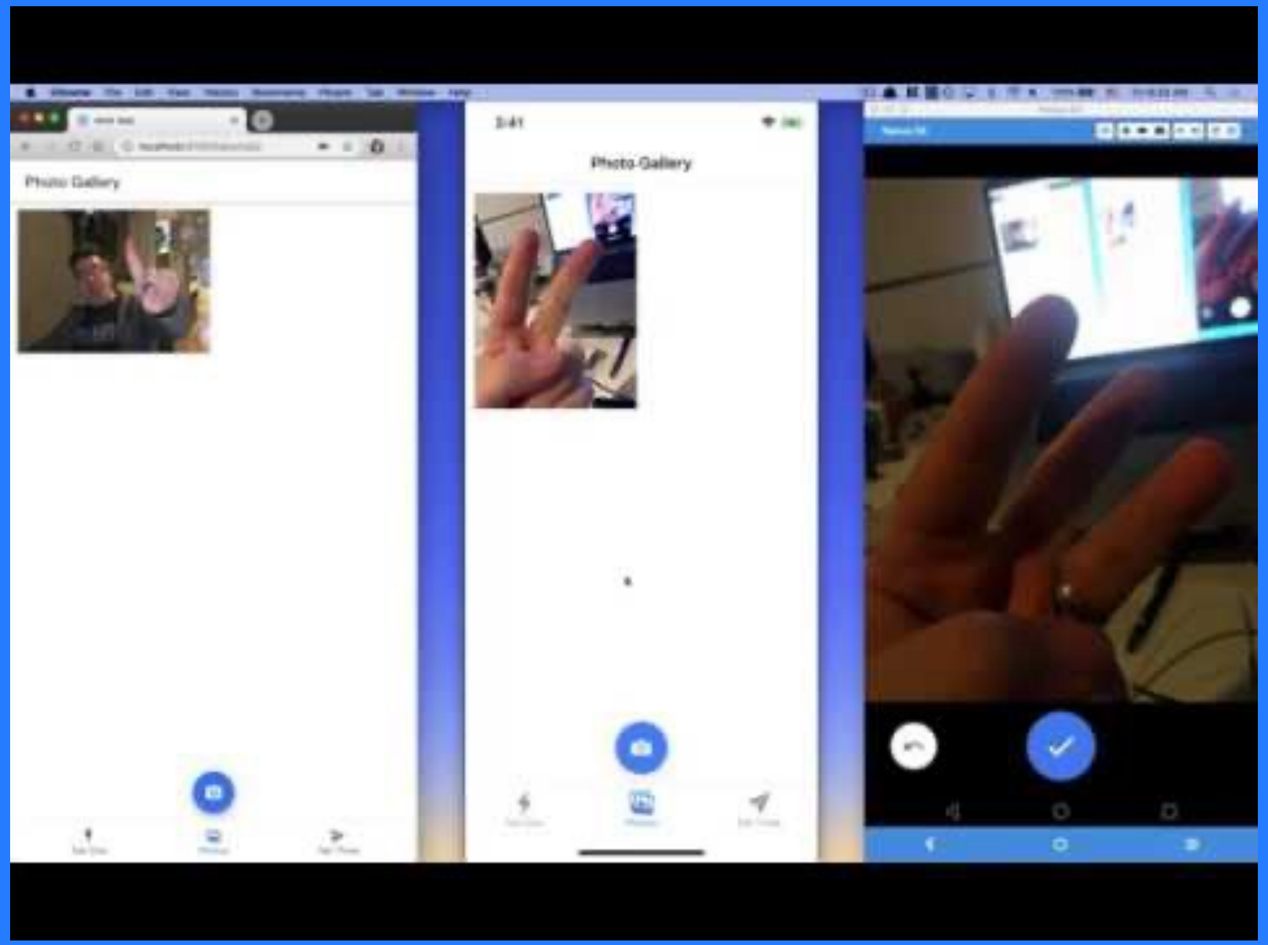
# Components

Think in components. Is there any logic that might be useful to move to separate components?



# Ionic React Camera App

https://ionicframework.com/docs/react/your-first-app#create-an-app



# Next Tuesday