

# WEB NATIVE, TOOLS & UI COMPONENTS





# Agenda



Ionic React & TypeScript  
Ionic Core Concepts  
UI Components  
Mobile Navigation & Routing  
(Mobile UI & UX)

# Your First Ionic Apps

Ionic User List App

Thinking in React?

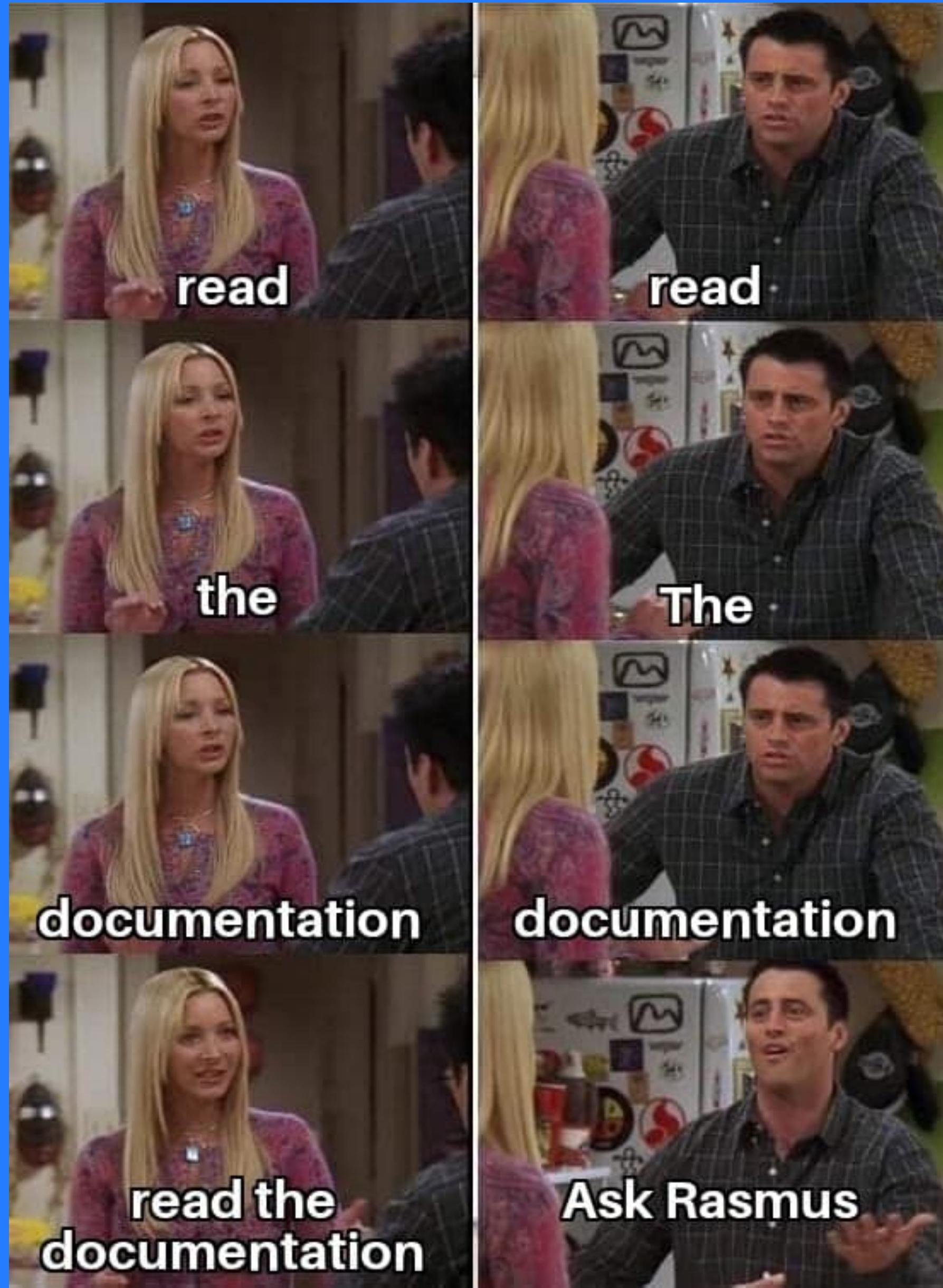
Components, components, components

Why lifecycle methods?

Ionic Post App

Ionic Camera App





# Hands-on & Documentation

WU-E21s 2nd sem | Calendar [+ New](#)

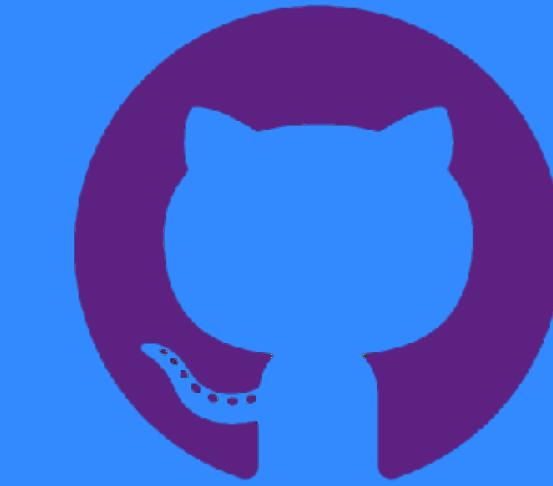
<https://bewildergeist.notion.site/bewildergeist/00356a53759946969b2cae1dfc28cd3?v=36388c84f6bd472a8b6bec609520767e>

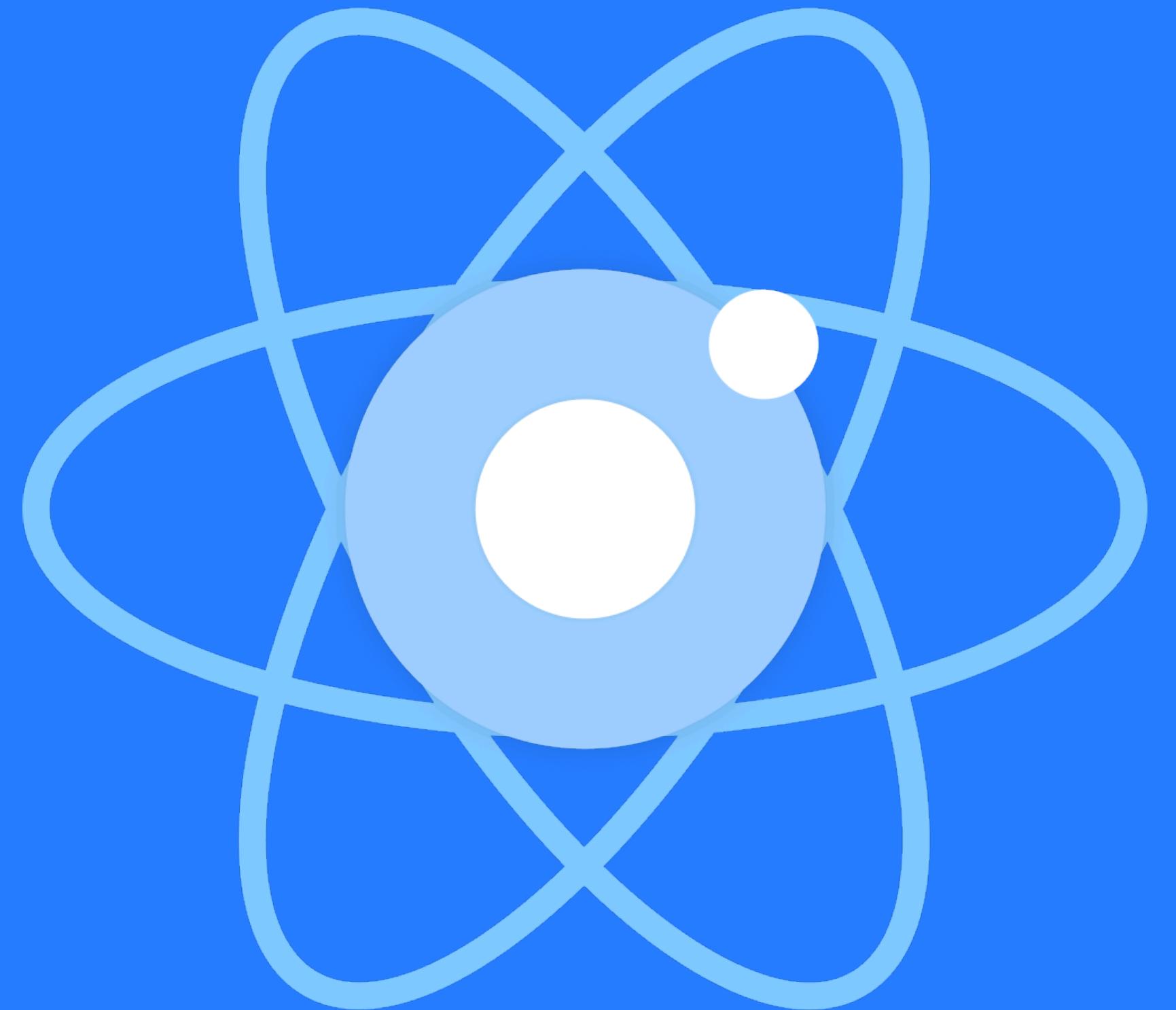
Search Duplicate [...](#) Try Notion

February 2022 < Today >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31 1. PWAs and Progressi... <a href="#">PWA</a> <a href="#">DOB</a>	Feb 1	2 1. Becoming Advanced... <a href="#">AWP</a> <a href="#">DOB</a>	3 1. Introduction to M... <a href="#">MAD</a> <a href="#">RACE</a>	4	5
6	7 2. Frontend performan... <a href="#">PWA</a> <a href="#">DOB</a>	8	9 2. Understanding React <a href="#">AWP</a> <a href="#">DOB</a>	10 2. Web Native, Tool... <a href="#">MAD</a> <a href="#">RACE</a>	11	12
13	14 3. HTTP caching for p... <a href="#">PWA</a> <a href="#">DOB</a>	15	16 3. Understanding Remix <a href="#">AWP</a> <a href="#">DOB</a>	17 3. Native Device Fe... <a href="#">MAD</a> <a href="#">RACE</a>	18	19
20	21 4. Modern responsive ... <a href="#">PWA</a> <a href="#">DOB</a>	22	23 4. Styling: CSS vs. Tail... <a href="#">AWP</a> <a href="#">DOB</a>	24 4. Storage & Native ... <a href="#">MAD</a> <a href="#">RACE</a>	25	26
27	28 5. Async Javascript + ... <a href="#">PWA</a> <a href="#">DOB</a>	Mar 1	2 5. Fetch, Request, Res... <a href="#">AWP</a> <a href="#">DOB</a>	3 5. Firebase & App D... <a href="#">MAD</a> <a href="#">RACE</a>	4	5
6	7 6. Service workers de... <a href="#">DWA</a>	8	9 6. HTML Forms for dat... <a href="#">AWD</a>	10 6. Ionic App Project <a href="#">MAD</a>	11	12

# CLI & Tools





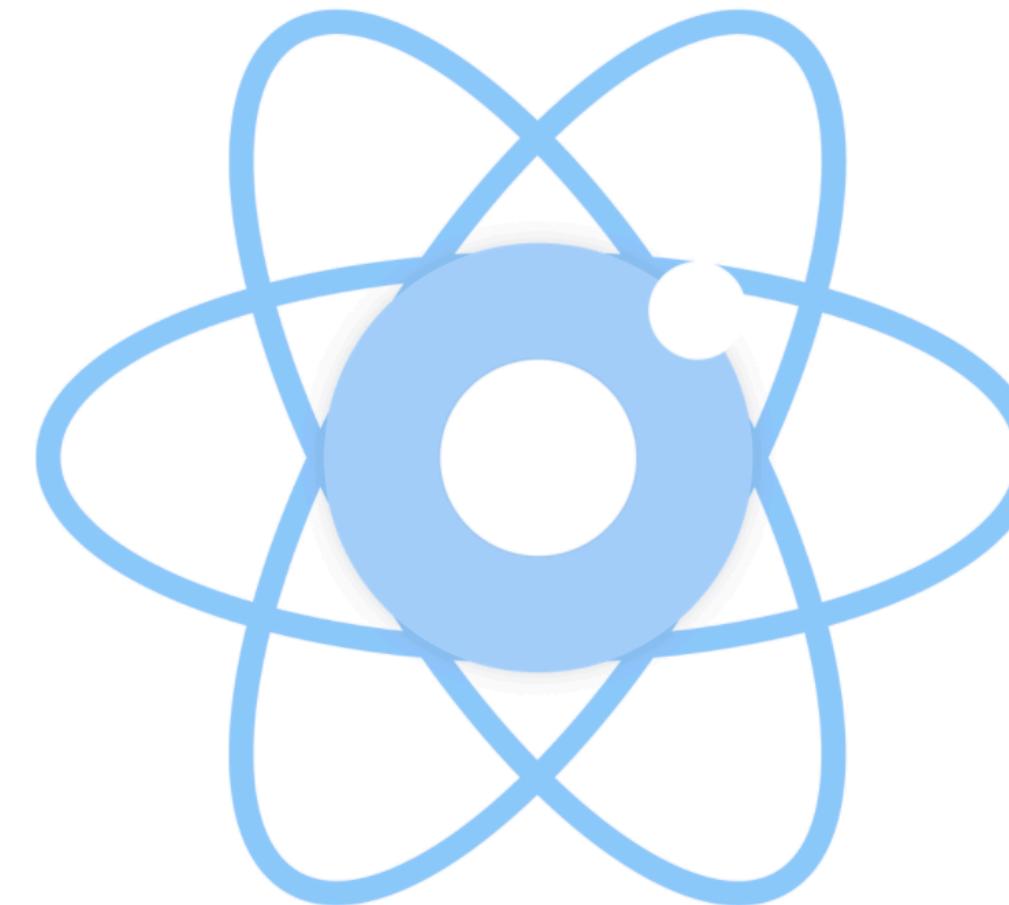
# Ionic React

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

# Ionic is a Cross-platform

**One Codebase  
Any Platform  
Just React**

- ✓ 100+ mobile optimized React UI components
- ✓ Standard React tooling with react-dom
- ✓ iOS / Android / Electron / PWA



**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.



## Native Functionality

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



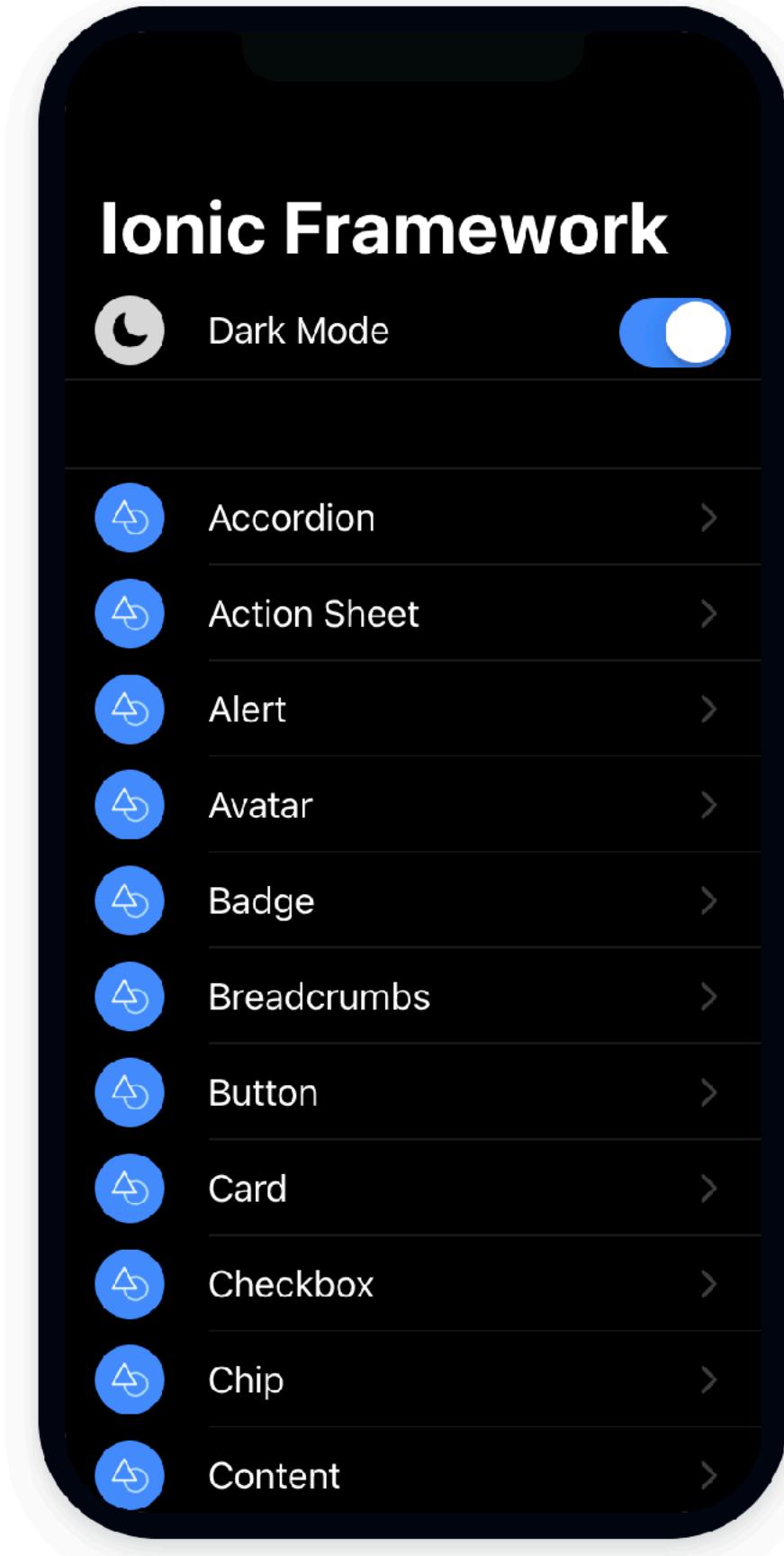
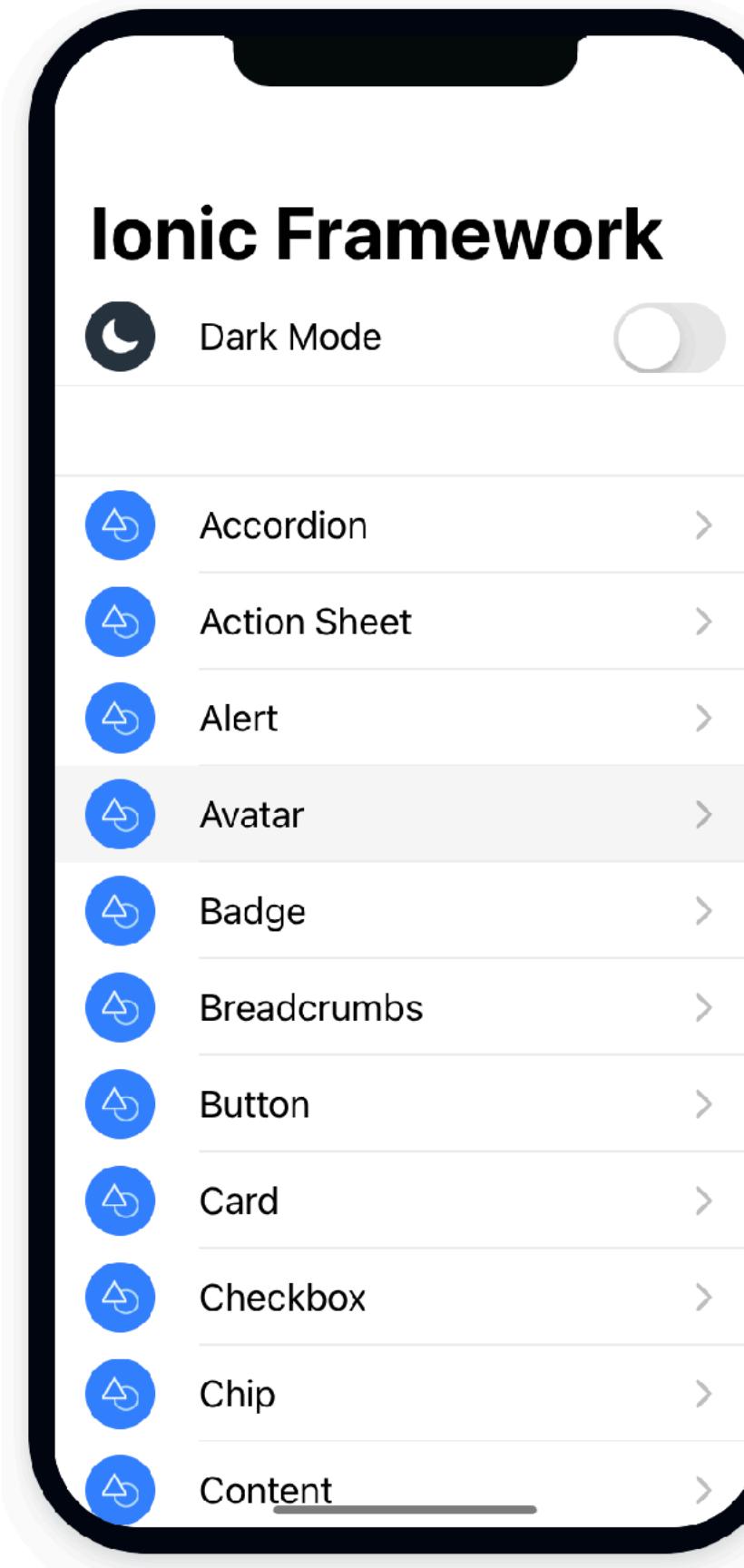
## UI Components

Dive into Ionic beautifully designed UI component library.



## Theming

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



<https://ionicframework.com/>

# Web Native

“Web Native is the idea that teams should build modern Web Apps and combine them with tools like Capacitor that unlock powerful Native APIs to the app for the platform its running on.”

“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.”

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

<https://webnative.tech/>

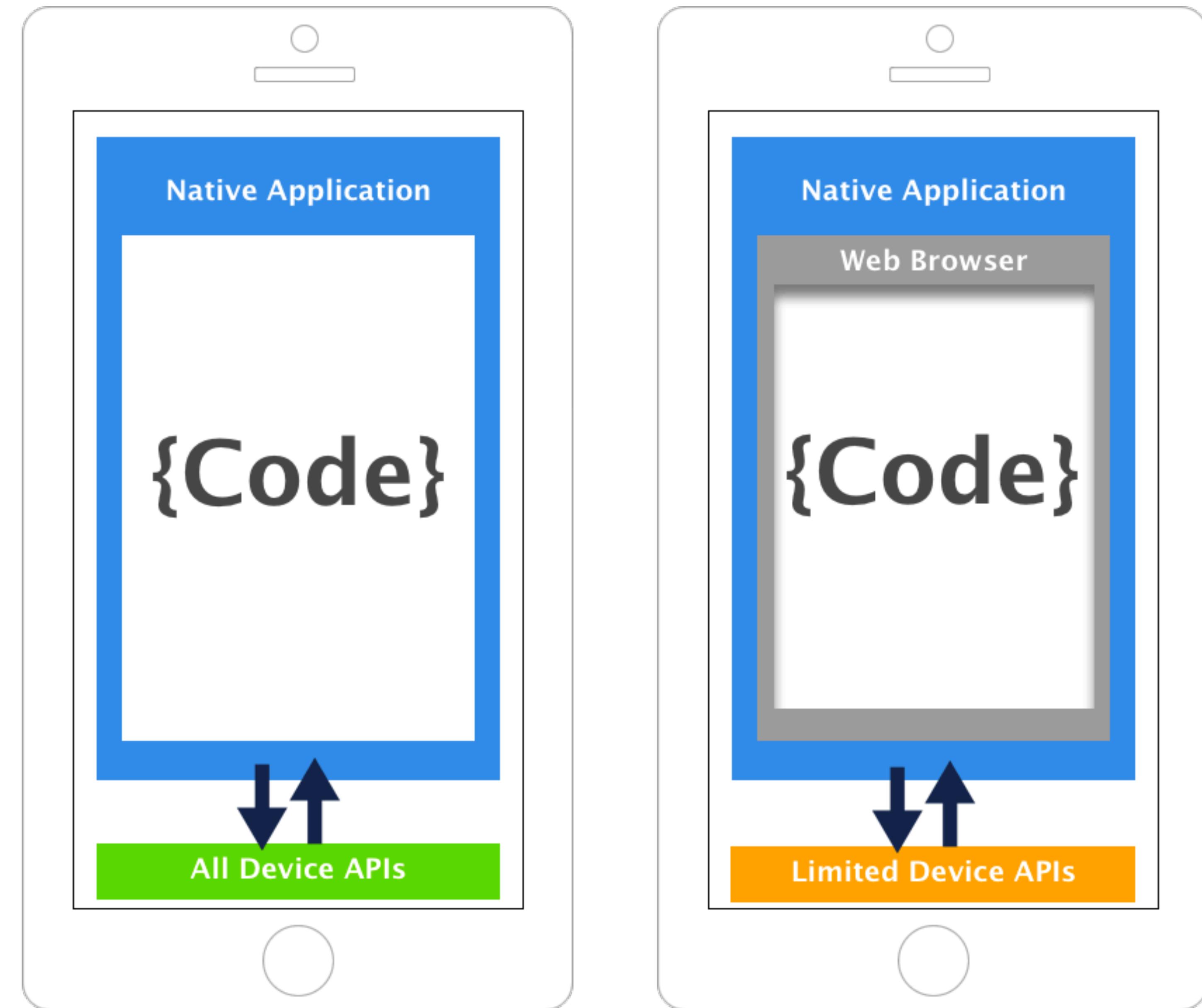


# Capacitor

A cross-platform native runtime for web apps.



<https://capacitorjs.com/>



# Back in the 2010s (Cordova)

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



Runs web apps  
across multiple  
platforms

Hybrid Apps

# Today (Capacitor)

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



Runs modern web apps  
natively on iOS, Android,  
Electron and Web  
multiple platforms.

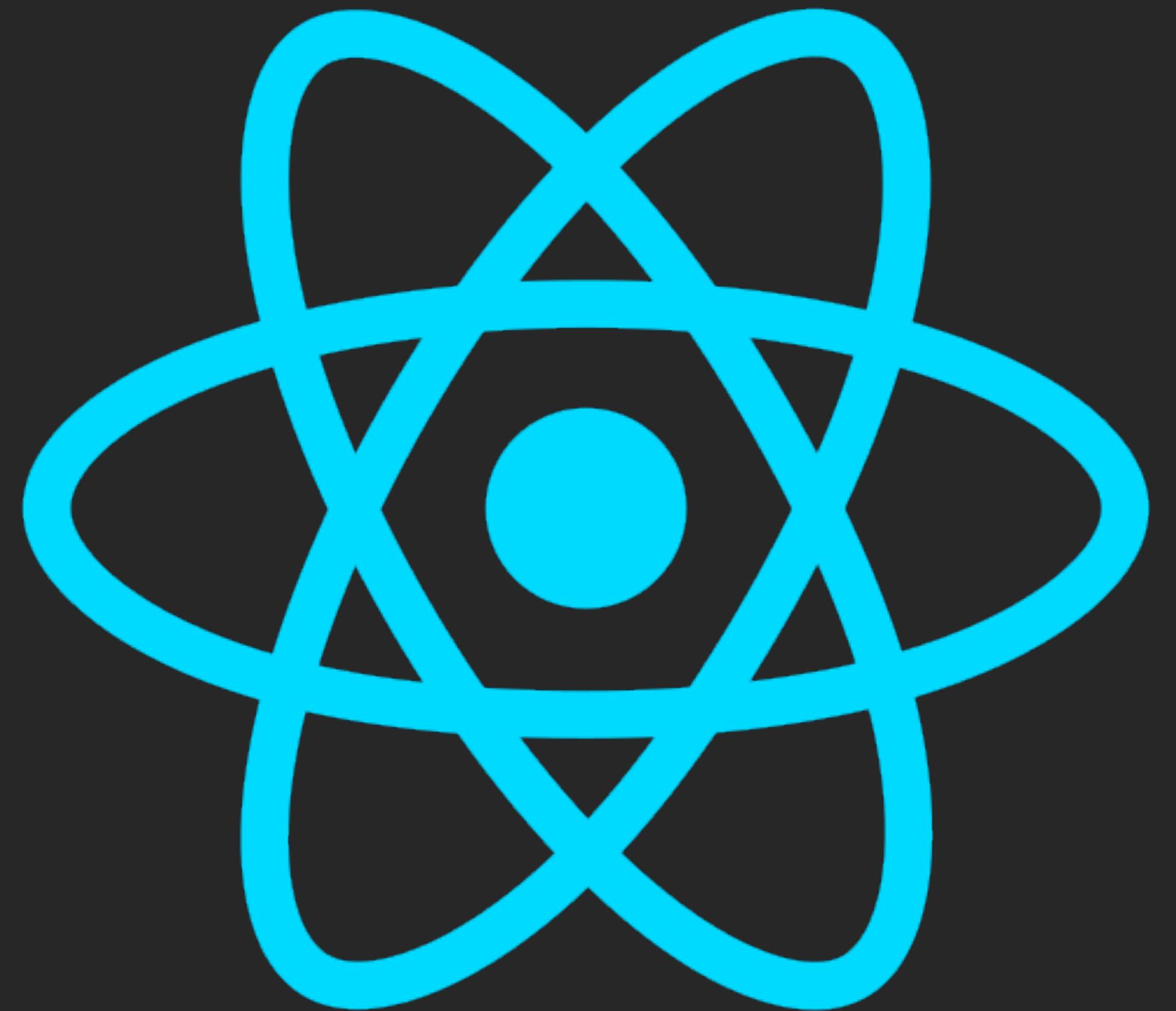
Web Native Apps & PWA

## Traditional Approach



## Ionic (Web Native) Approach





**It's Just React**  
Ionic React is just React  
components

JS

Well, at the end of  
the day, it's just  
JavaScript



# WHAT IS TYPESCRIPT?



A screenshot of a web browser window. The title bar says "API Glossary: Terminology and". The address bar shows "ionicframework.com/docs/reference/glossary#typ...". Below the address bar is a navigation bar with icons for search, refresh, and other functions. The main content area has a sidebar icon and a search bar. The main text is about TypeScript being a superset of JavaScript with type declarations and interfaces. It also mentions that using TypeScript is optional. There is a section titled "Unit Tests" with a note about testing small pieces of code.

**TypeScript**

TypeScript is a superset of JavaScript, which means it gives you JavaScript, along with a number of extra features such as [type declarations](#) and [interfaces](#). Although Ionic is built with TypeScript, using it to build an Ionic app is completely optional.

**Unit Tests**

Unit Tests and unit testing are a way to test small pieces of code to see if

<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/>



# Usage

Angular   Javascript   **React**   Stencil   Vue

ios   Android

## Action Sheet

ion-action-sheet

## Accordion

ion-accordion

ion-accordion-group

## Alert

ion-alert

## Badge

ion-badge

## Breadcrumb

ion-breadcrumb

ion-breadcrumbs

## Button

ion-button

ion-ripple-effect

## Card

ion-card

ion-card-content

ion-card-header

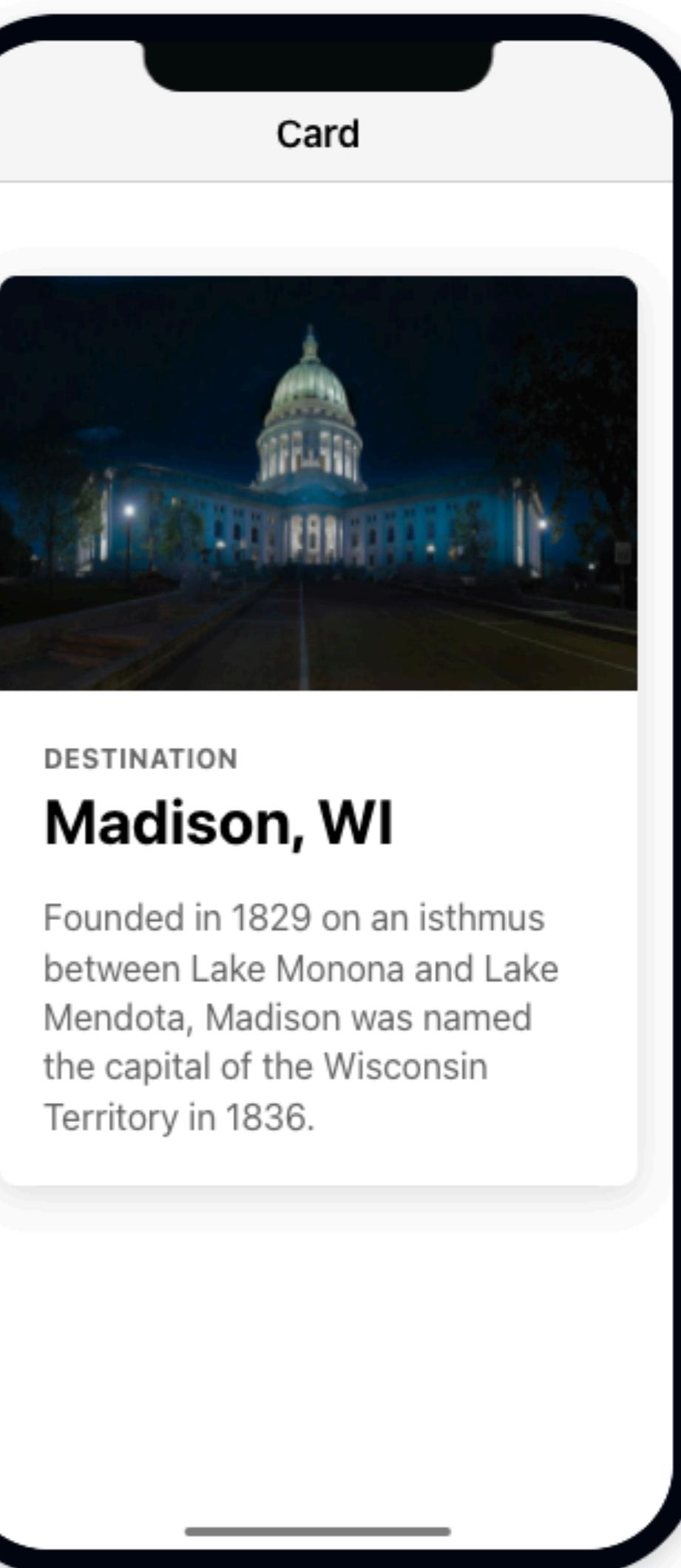
ion-card-subtitle

```
import React from 'react';
import { IonContent, IonHeader, IonPage, IonTitle, IonToolbar, IonCard, IonCardContent } from '@ionic/react';
import { pin, wifi, wine, warning, walk } from 'ionicons/icons';
```

```
export const CardExamples: React.FC = () => {
  return (
    <IonPage>
      <IonHeader>
        <IonToolbar>
          <IonTitle>Card Examples</IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent>
        <IonCard>
          <IonCardHeader>
            <IonCardSubtitle>Card Subtitle</IonCardSubtitle>
            <IonCardTitle>Card Title</IonCardTitle>
          </IonCardHeader>
          <IonCardContent>
            Keep close to Nature's heart... and break clear away, once in awhile  

            and climb a mountain or spend a week in the woods. Wash your spirit
          </IonCardContent>
        </IonCard>
        <IonCard>
          <IonItem>
            <IonIcon icon={pin} slot="start" />
            <IonLabel>ion-item in a card, icon left, button right</IonLabel>
            <IonButton fill="outline" slot="end">View</IonButton>
          </IonItem>
        </IonCard>
      </IonContent>
    </IonPage>
  );
}
```

Copy



# TypeScript for JavaScript Programmers

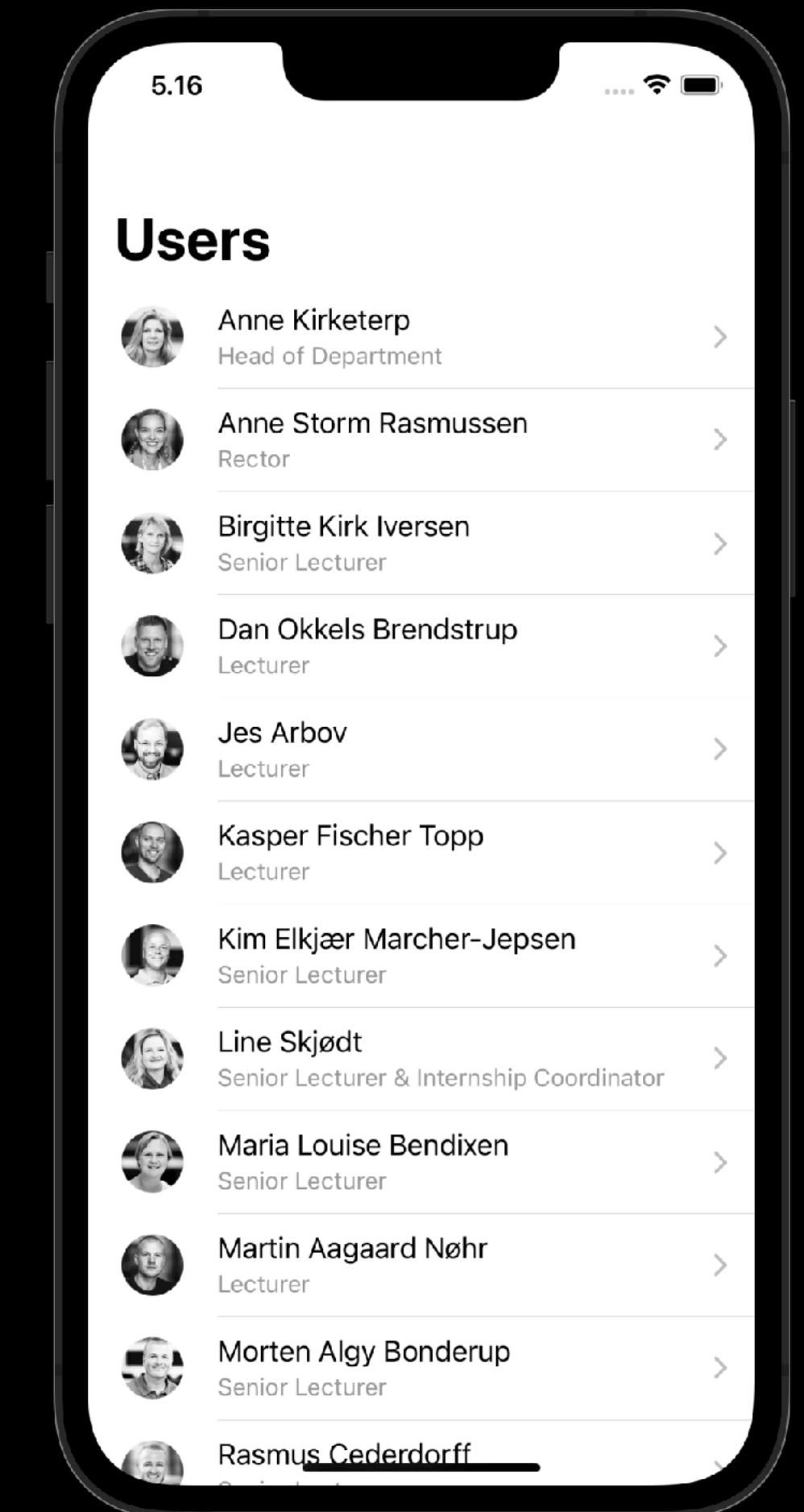
<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>

# Ionic User List App

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

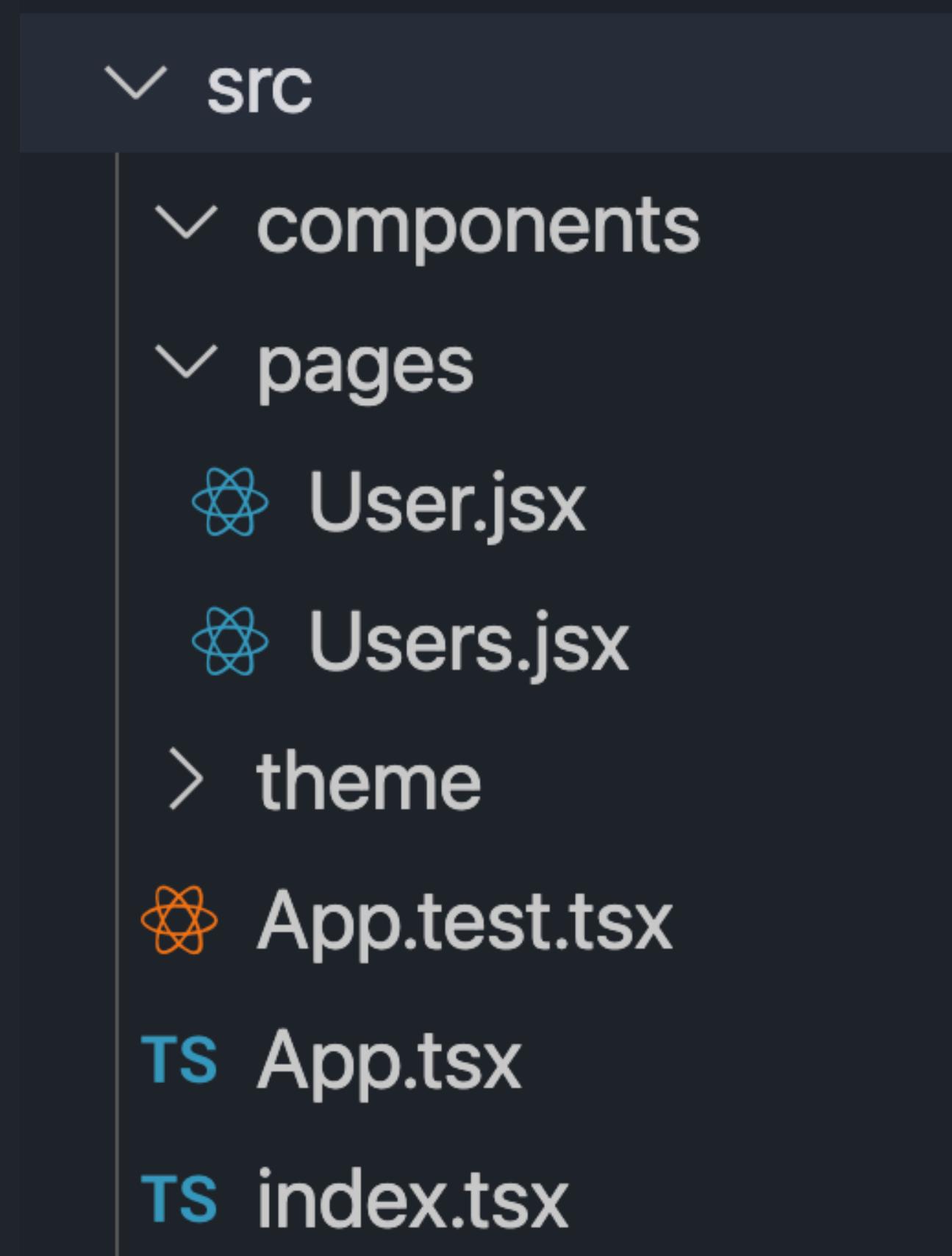
[ionic-user-list](#)

[ionic-user-list-no-ts](#)

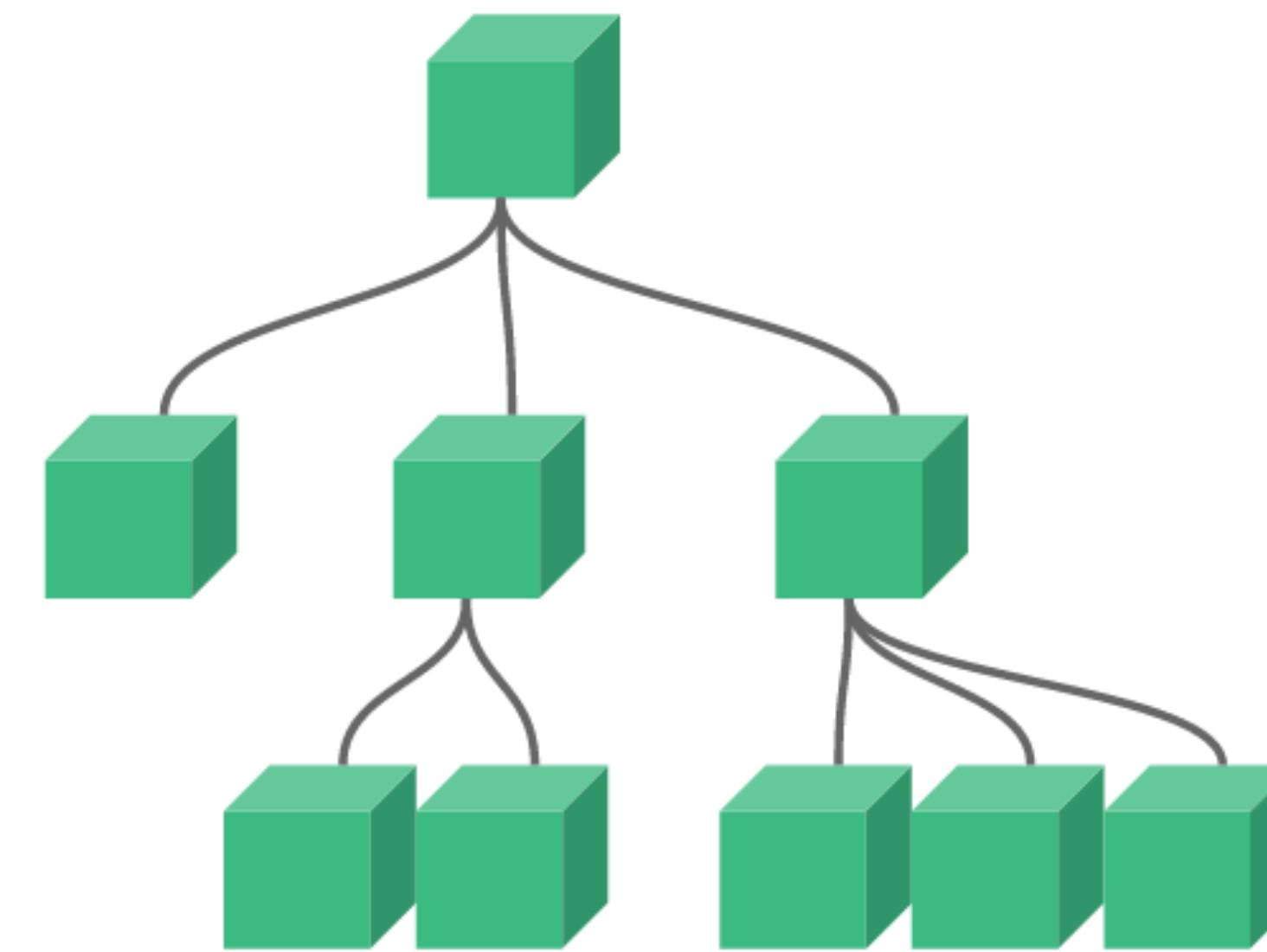
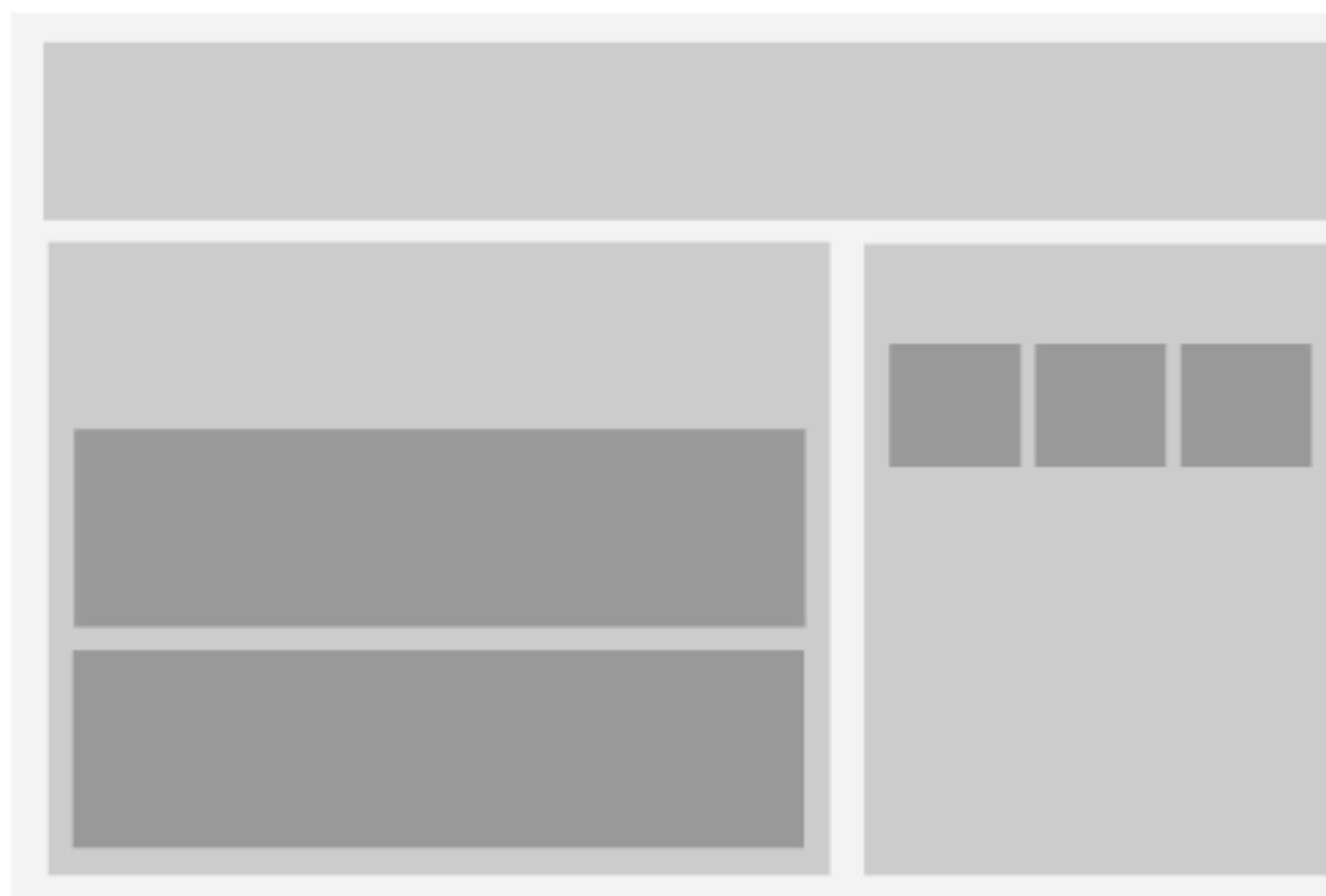


# Thinking in React?

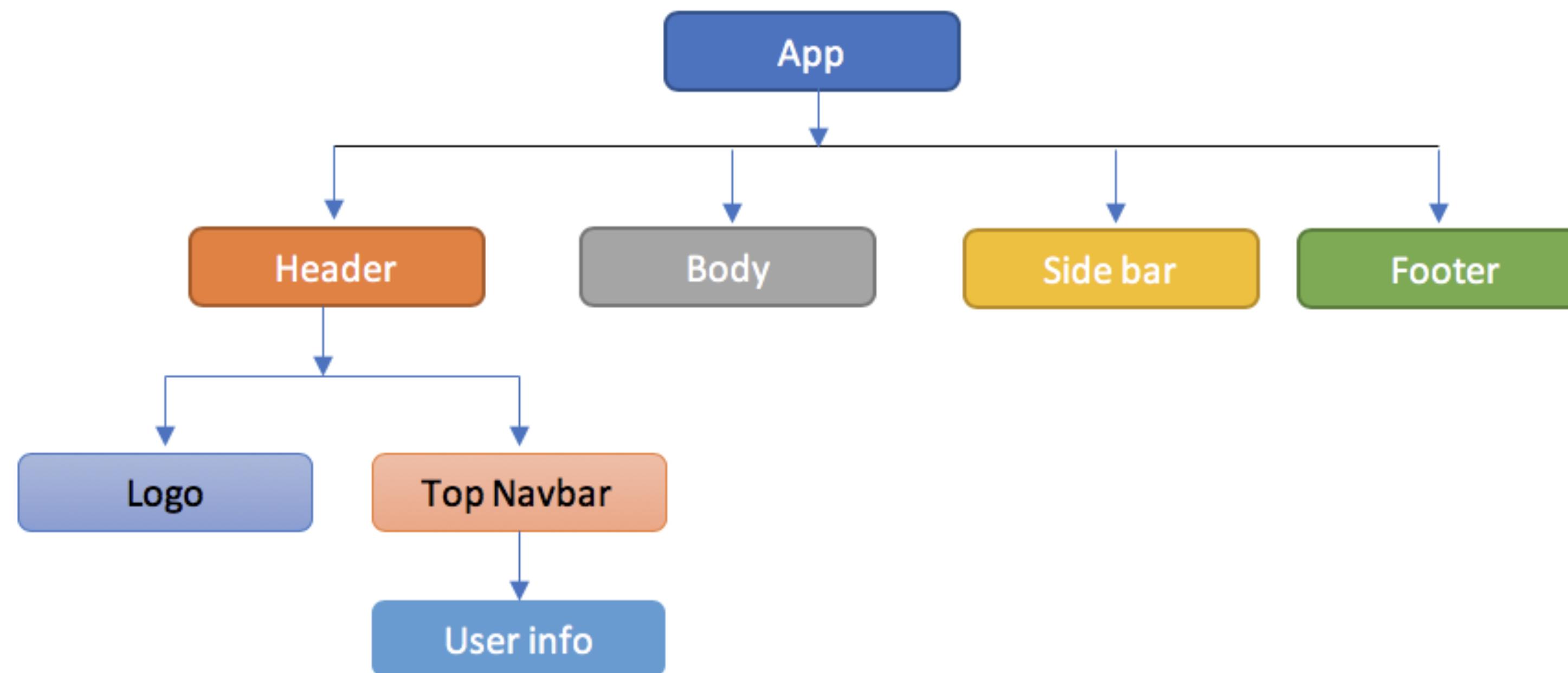
[https://beta.reactjs.org/learn/  
thinking-in-react](https://beta.reactjs.org/learn/thinking-in-react)



# COMPONENTS



# APP STRUCTURED IN COMPONENTS



# Structure

It's all components

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

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

```
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>
)
```

# Components, components, components

1. Review your Users component.
2. Think of how to restructure the component and divide it into several components.
3. Create a file and a new component named UserListItem or whatever you prefer.
4. Extract the IonItem to the newly created component. Refactor what's needed.
5. Import and use the new component in Users and make it pass needed props.

```
src
  components
    TS UserListItem.tsx
  pages
    TS User.tsx
    TS Users.tsx
  services
  theme
  App.test.tsx
  TS App.tsx
  TS index.tsx
```

A screenshot of a file explorer window showing a directory structure. The root folder is 'src'. It contains 'components' (containing 'UserListItem.tsx'), 'pages' (containing 'User.tsx' and 'Users.tsx'), 'services', 'theme', 'App.test.tsx', 'App.tsx', and 'index.tsx'. The 'UserListItem.tsx' file is highlighted with a blue background and has a 'TS' icon next to its name. The 'User.tsx' and 'Users.tsx' files also have 'TS' icons next to them. The 'App.test.tsx', 'App.tsx', and 'index.tsx' files have orange test icon next to them. The 'services' and 'theme' folders have greater than signs next to them, indicating they are collapsed.

# Ionic Core Concepts

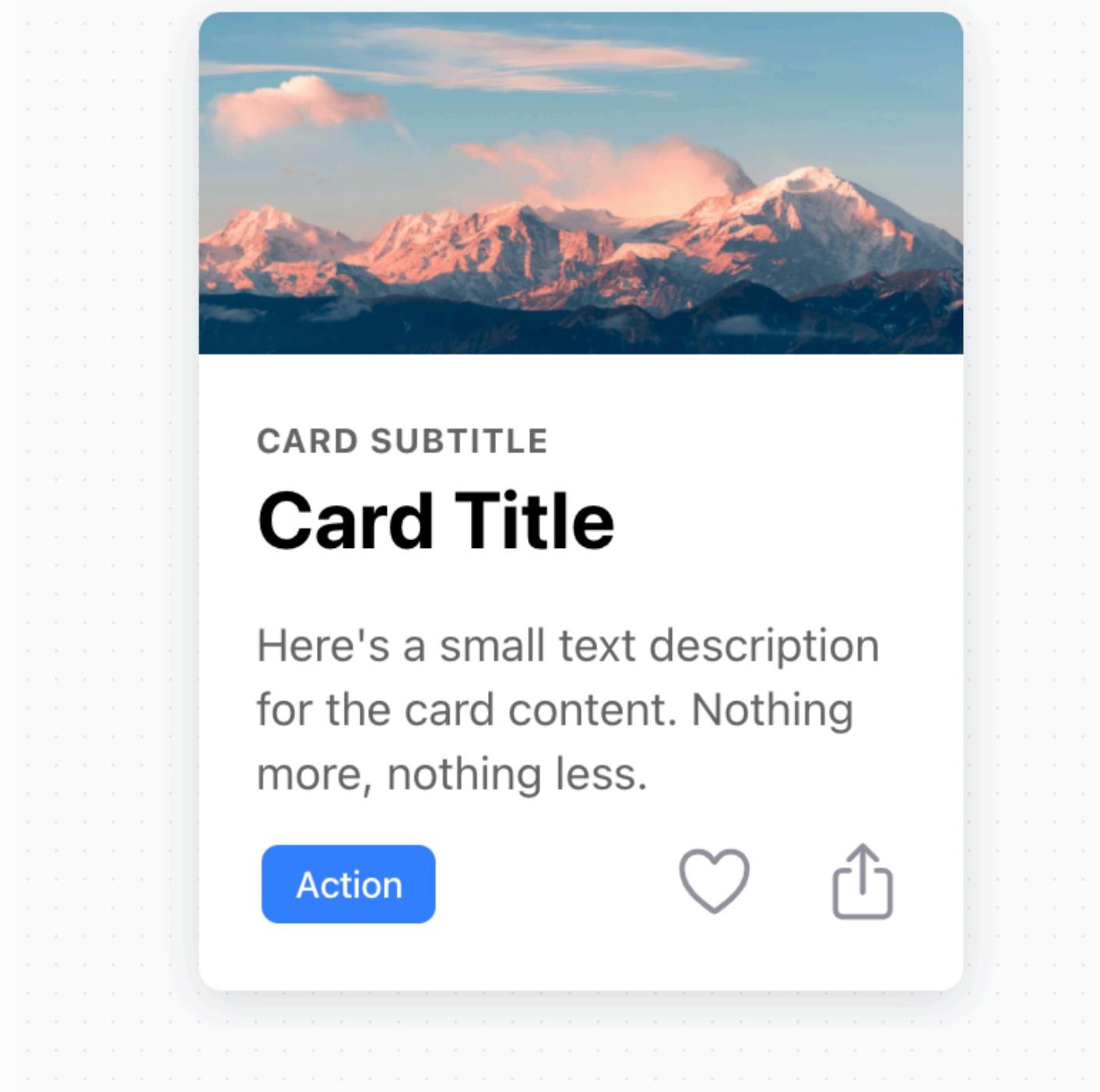
<https://ionicframework.com/docs/core-concepts/fundamentals>

# UI Components

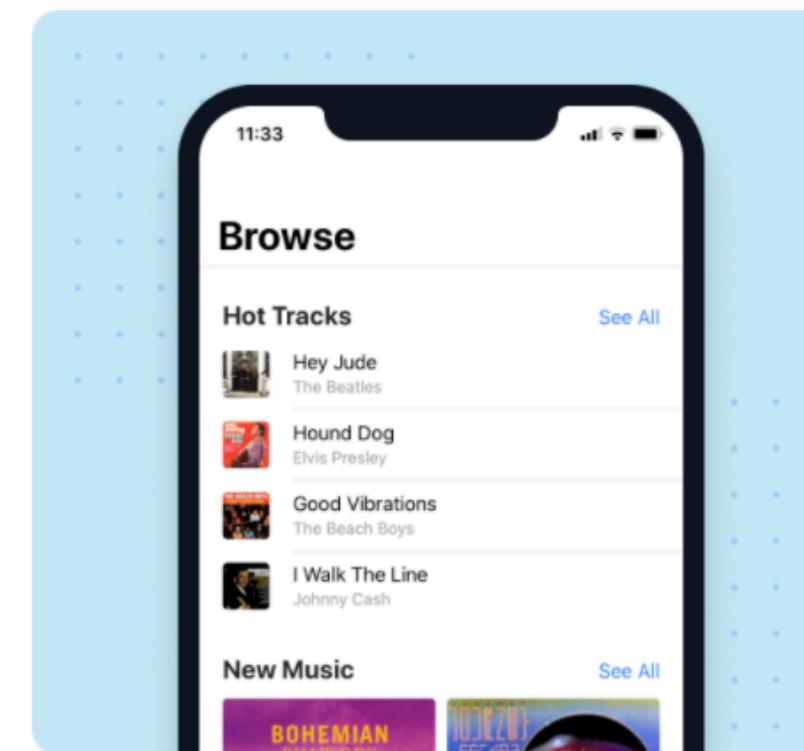
The screenshot shows a web browser window displaying the Ionic UI Components documentation at [ionicframework.com/docs/components](https://ionicframework.com/docs/components). The page has a dark header with the Ionic logo and navigation links for Guide, Components (which is the active tab), CLI, Native, and an Upgrade Guide. The main content area features a large title "UI Components" and a paragraph explaining that Ionic apps are built using high-level building blocks called Components. Below this, there's a grid of cards, each representing a different UI component:

- Action Sheet**: Shows a card with three horizontal bars. Description: Action Sheets display a set of options with the ability to confirm or cancel an action.
- Alert**: Shows a card with a list icon and a red badge with the number 2. Description: Alerts are a great way to offer the user the ability to choose a specific action or list of actions.
- Badge**: Shows a card with a red badge icon. Description: Badges are a small component that typically communicate a numerical value to the user.
- Button**: Shows a card with a blue button icon. Description: Buttons let your users take action. They're an essential way to interact with and navigate through an app.
- Card**: Shows a card with a card icon. Description: Cards are a great way to display an important piece of content, and can contain images, buttons, text, and more.
- Checkbox**: Shows a card with a checked checkbox icon. Description: Checkboxes can be used to let the user know they need to make a binary decision.
- Chip**: Shows a card with a chip icon. Description: Chips are a compact way to display data or actions.
- Content**: Shows a card with a smartphone icon. Description: Content is the quintessential way to interact with and navigate through an app.

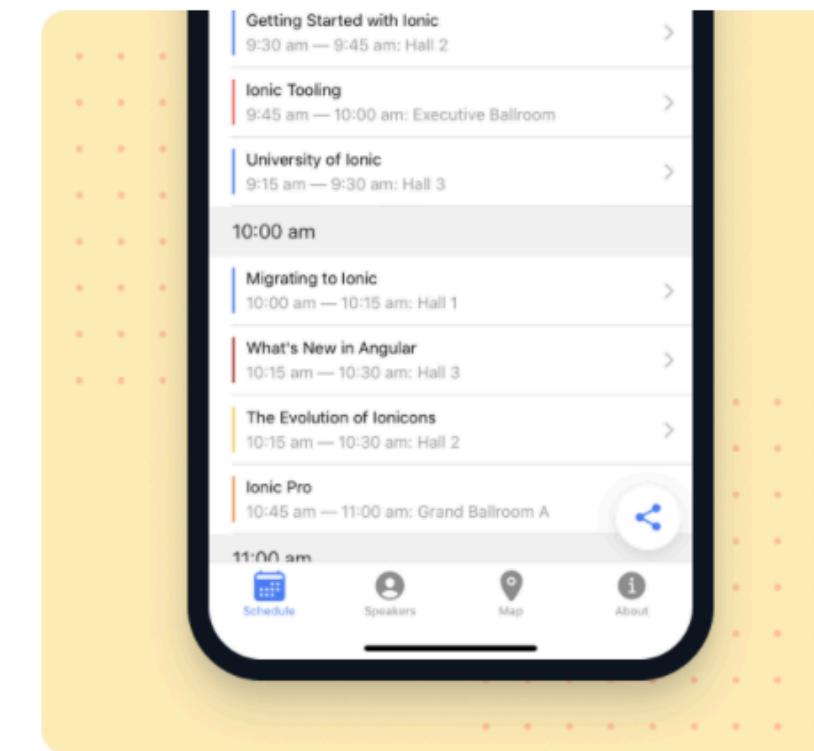
# Predefined UI Components



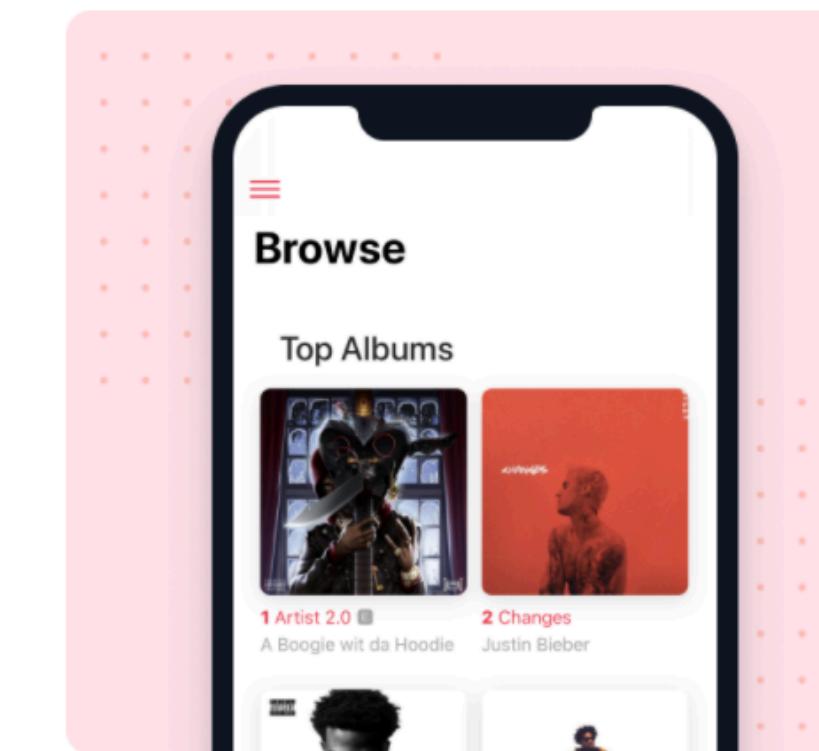
Awesome docs with many examples  
and sample apps



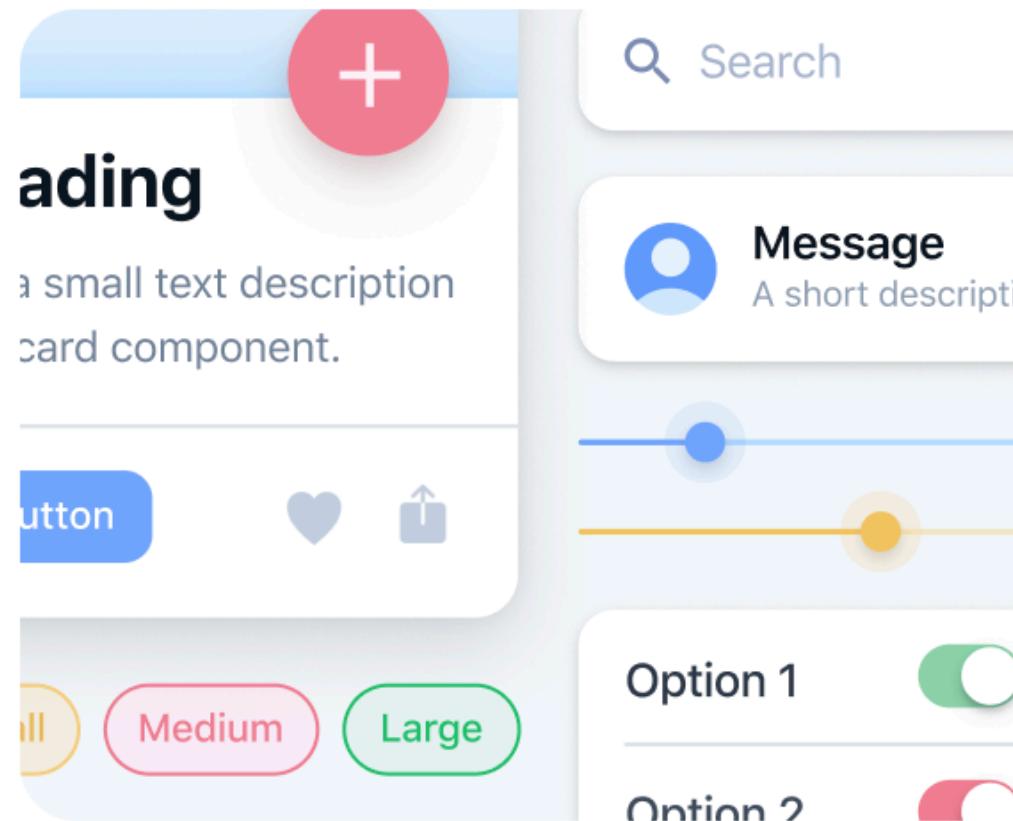
Music Player



Conference App



StarTrack



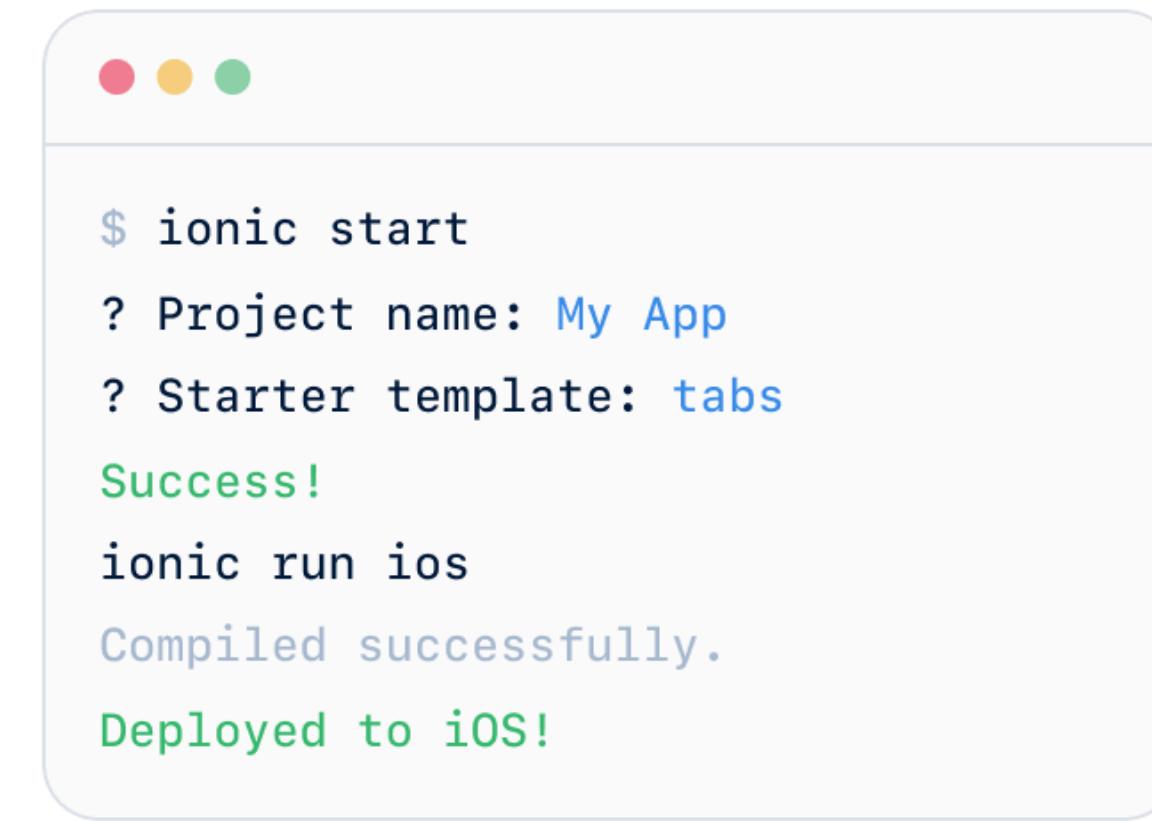
## Pre-designed UI components

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.



## 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.

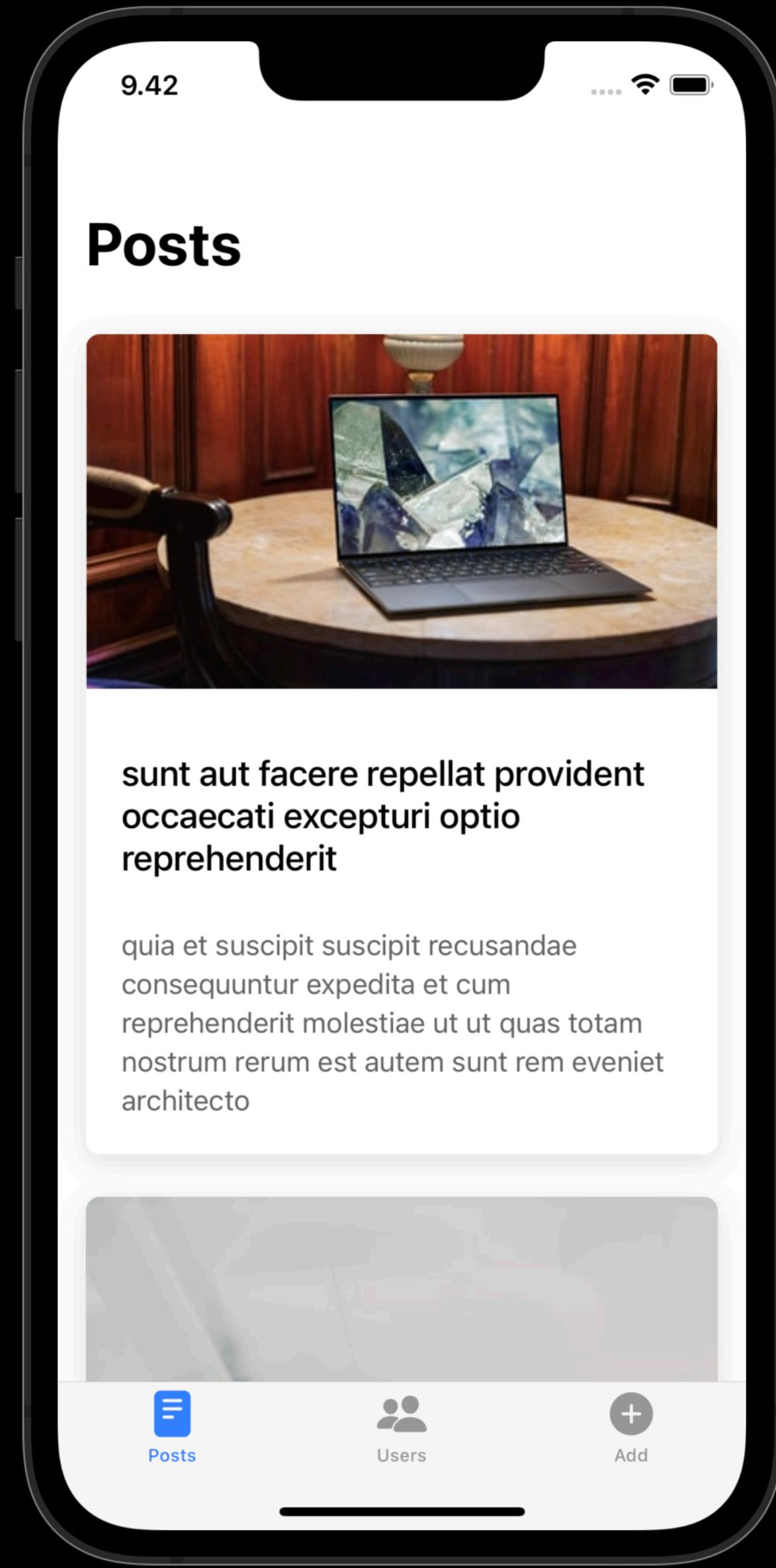
# Lifecycle of ion (React) Component

Lifecycle hooks: <https://ionicframework.com/docs/react/lifecycle>

# Why lifecycle methods?

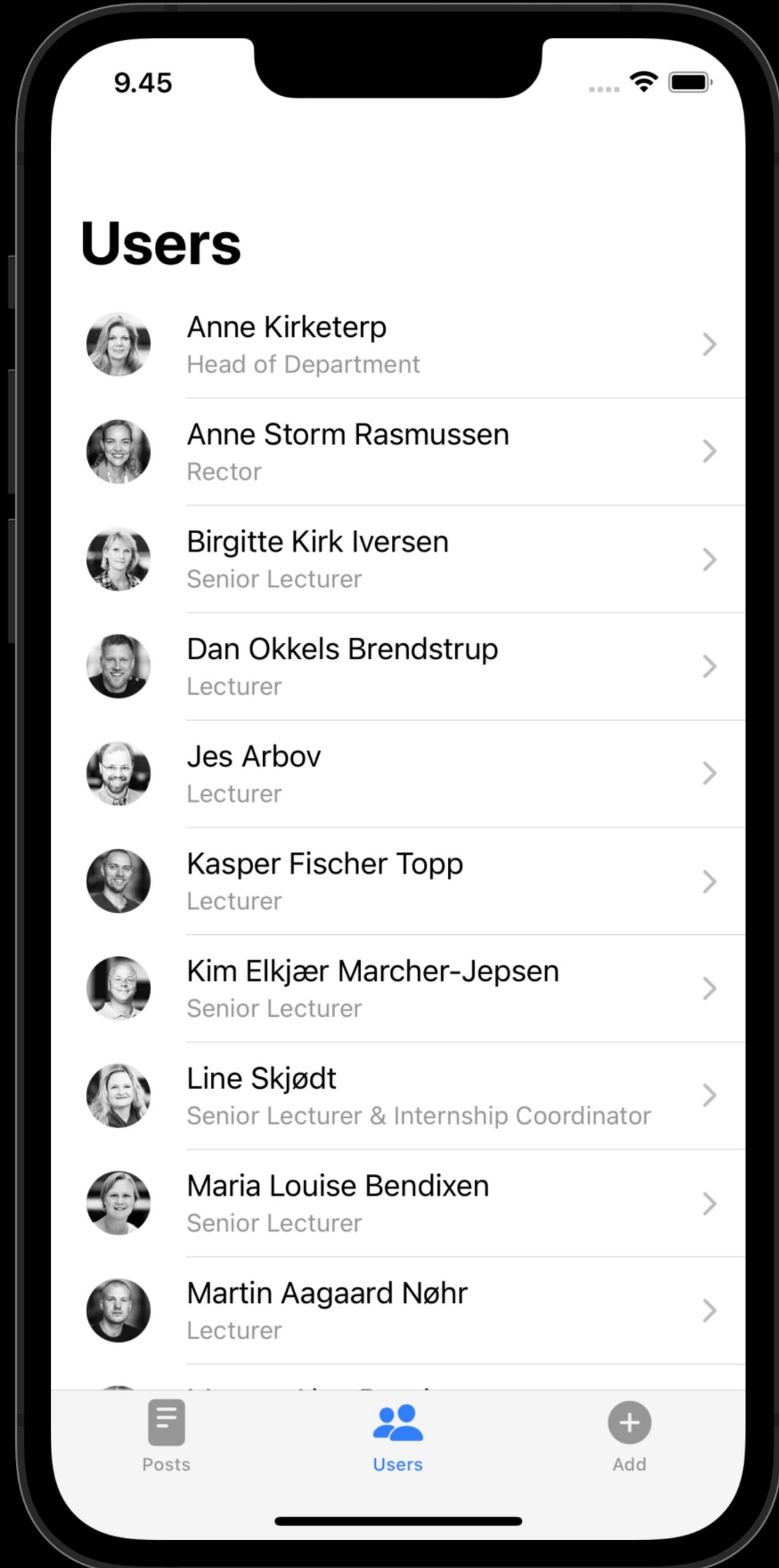
<https://ionicframework.com/docs/react/lifecycle#lifecycle-methods-in-functional-components>

1. Reimplement Users and User component using one of the Ionic React Lifecycle Hooks.
2. Consider the use and differences between `useEffect(...)` and `useIonViewWillEnter(...)`.



# 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.

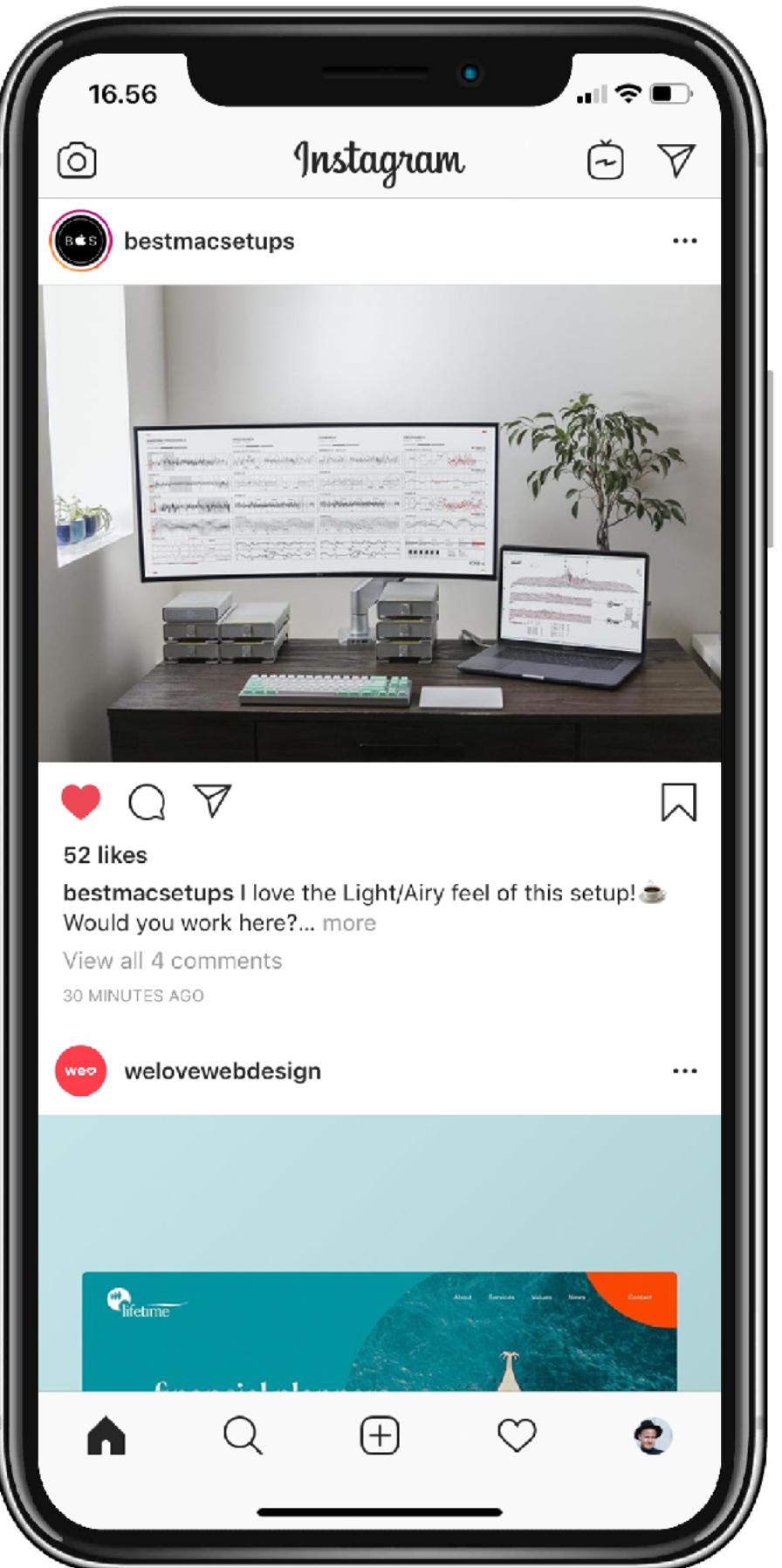
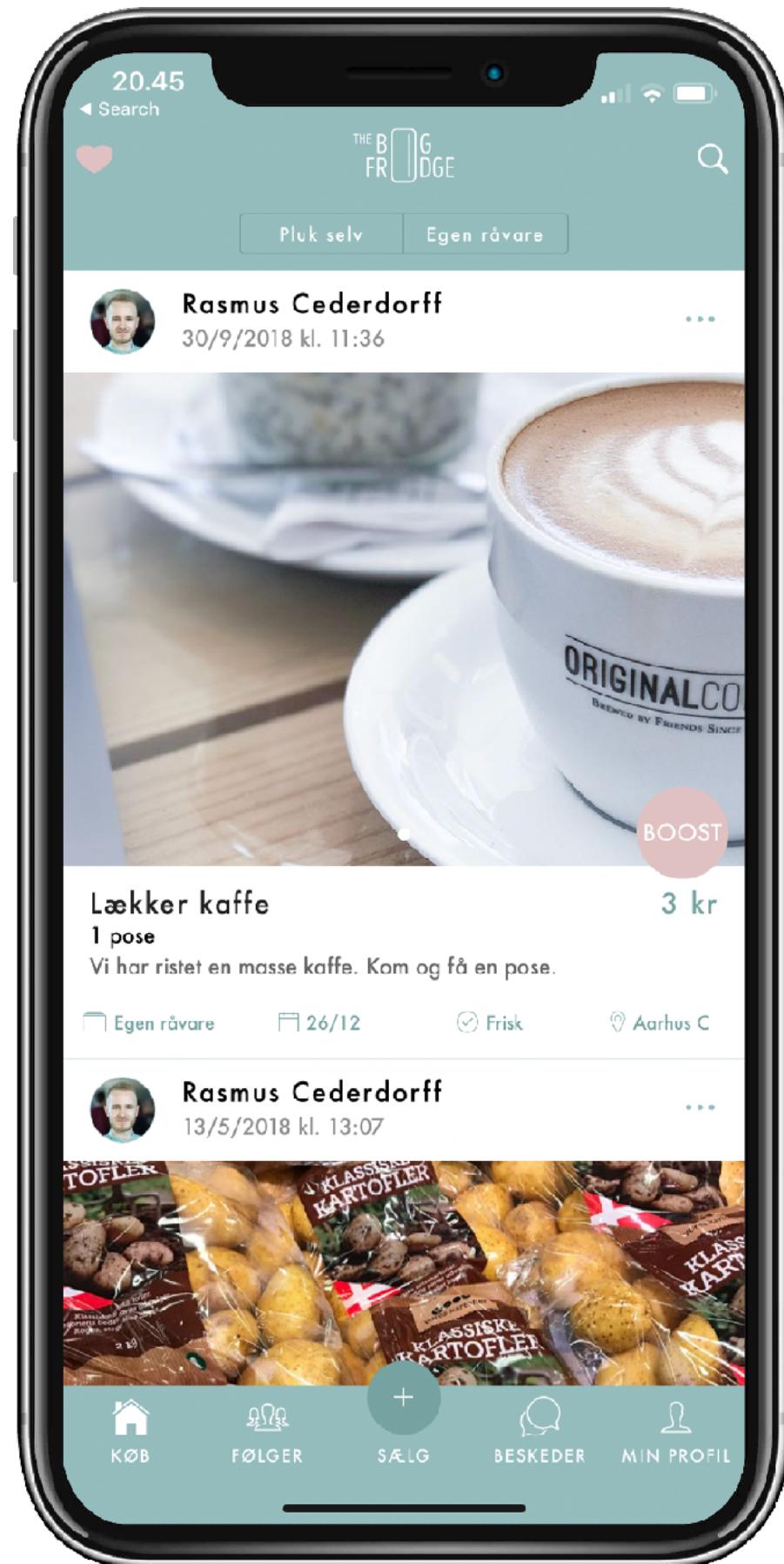
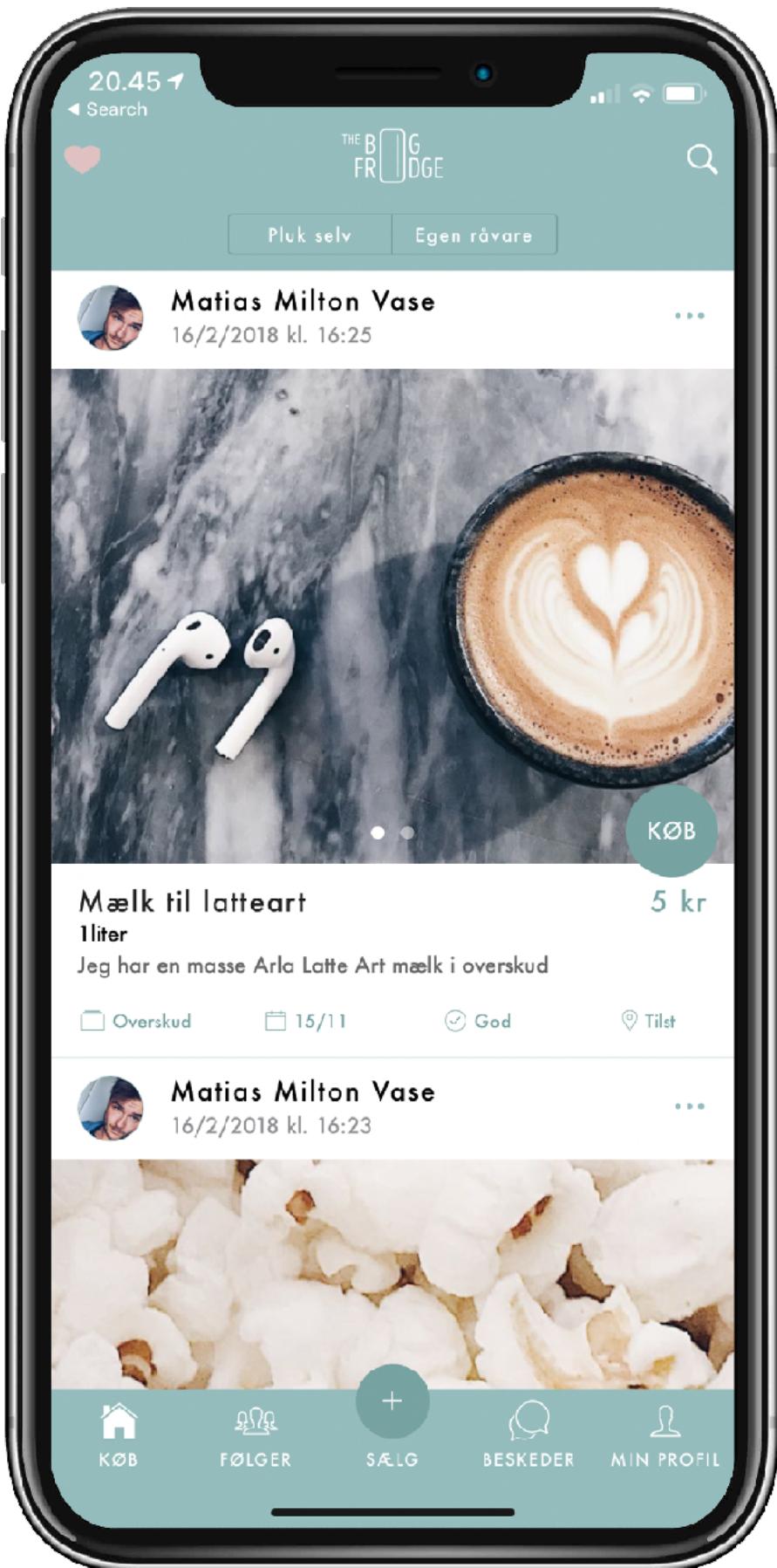
# Ionic Post App

7. Think in components. Is there any logic that might be useful to extract to separate components?
8. Reimplement your pages using components.

```
    <div>
      <ul>
        <li>src</li>
        <li><ul><li>components</li>
          <li>PostListItem.tsx</li>
          <li>UserListItem.tsx</li>
        </ul></li>
        <li><ul><li>pages</li>
          <li>Add.tsx</li>
          <li>Posts.tsx</li>
          <li>User.tsx</li>
          <li>Users.tsx</li>
        </ul></li>
        <li>> services</li>
        <li>> theme</li>
        <li>App.test.tsx</li>
        <li>App.tsx</li>
        <li>index.tsx</li>
      </ul>
    </div>
```

# 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";

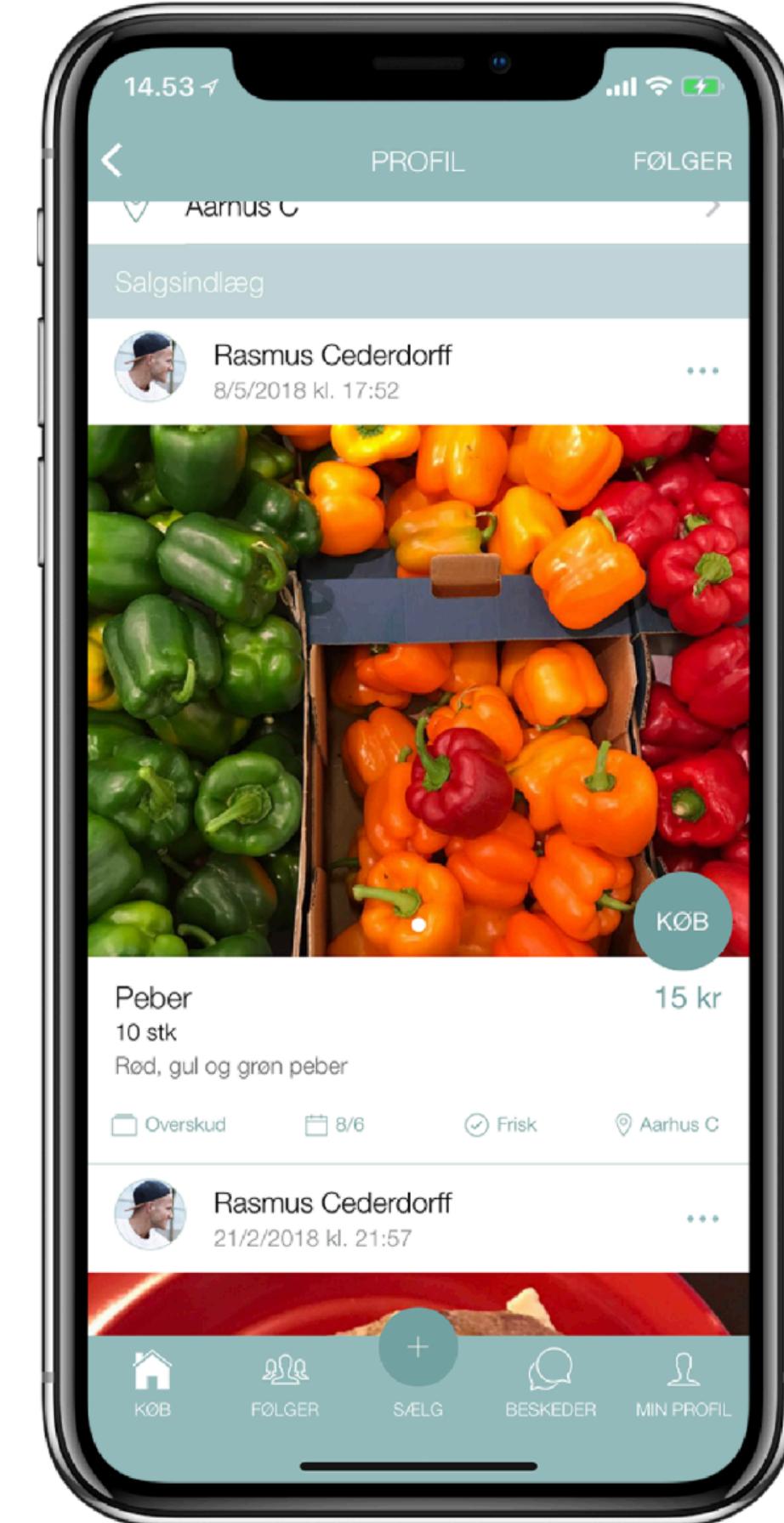
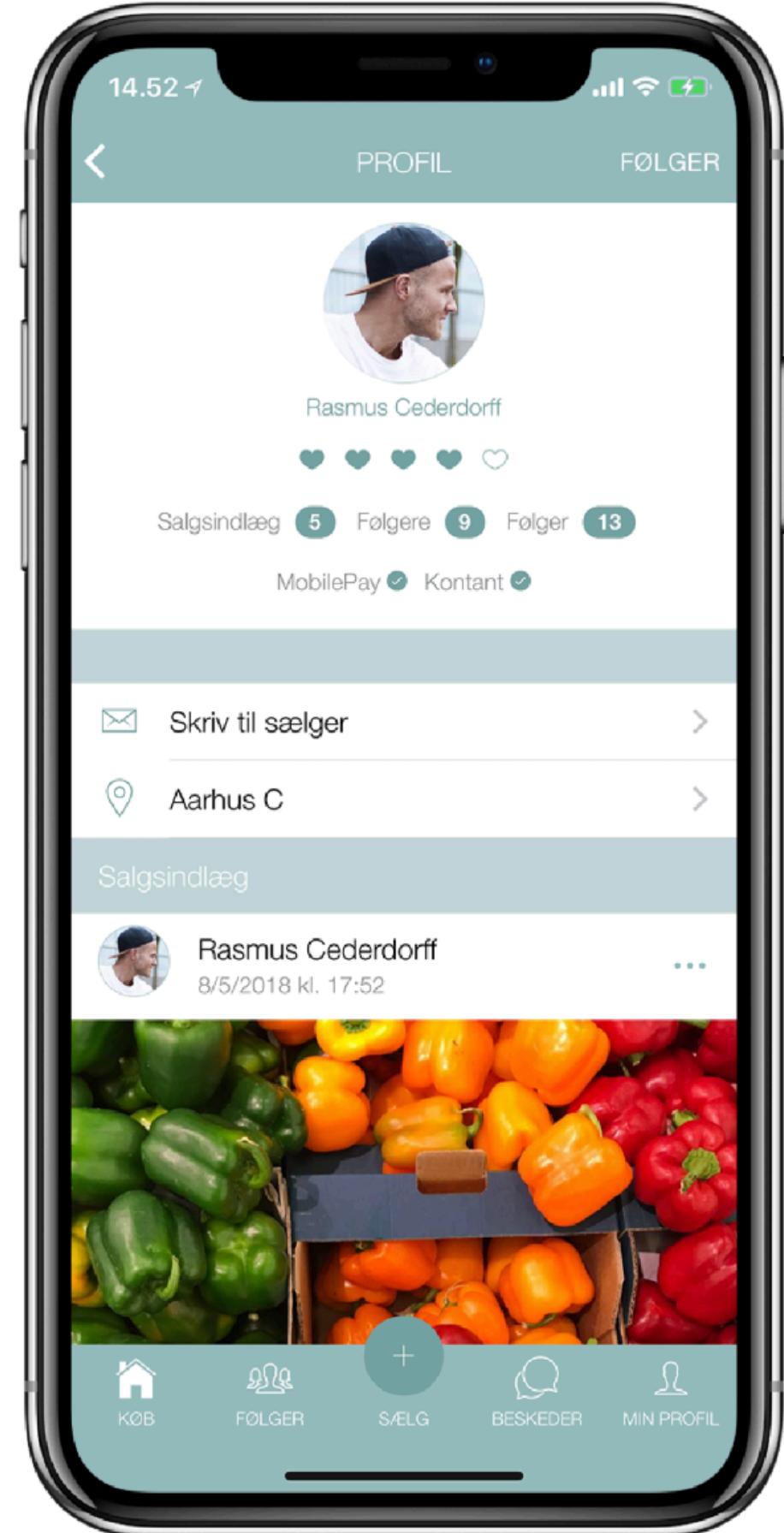
class PostService {
  >   constructor() { ... }

  >   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.



# Navigation & Routing

<https://ionicframework.com/docs/react/navigation>

# ion-router

<https://ionicframework.com/docs/api/router>

# ion-nav

<https://ionicframework.com/docs/api/nav>

# Add New Page

1. Add a new function component (a new page).
2. Add a new route in App.tsx to handle routing of your new page.
3. Test and make sure the navigation works as expected.
4. Next, change the redirect so your new page is the default page of your app.

# Types of Mobile Navigation

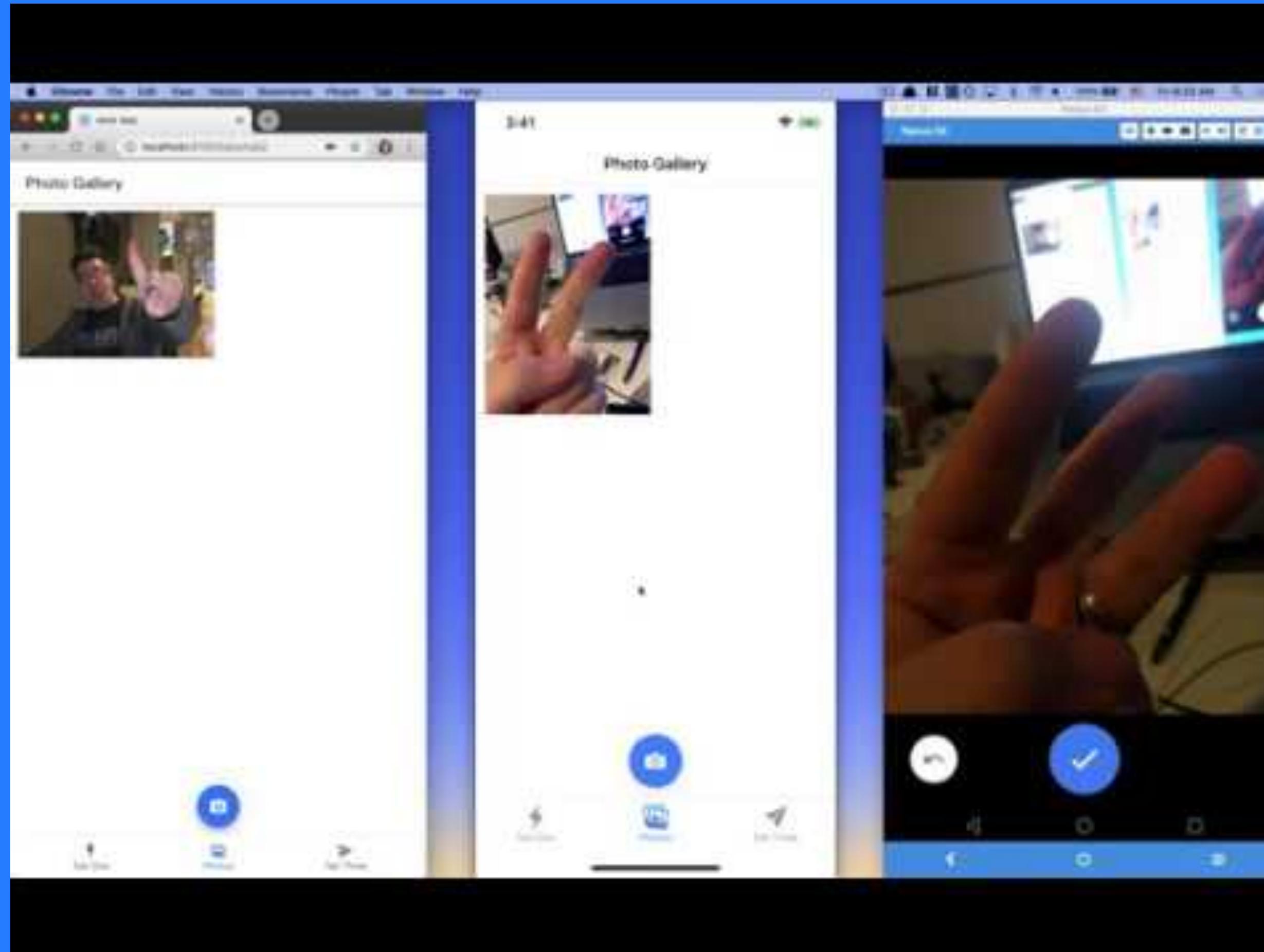
What kind of types do you know?

# Add a modal

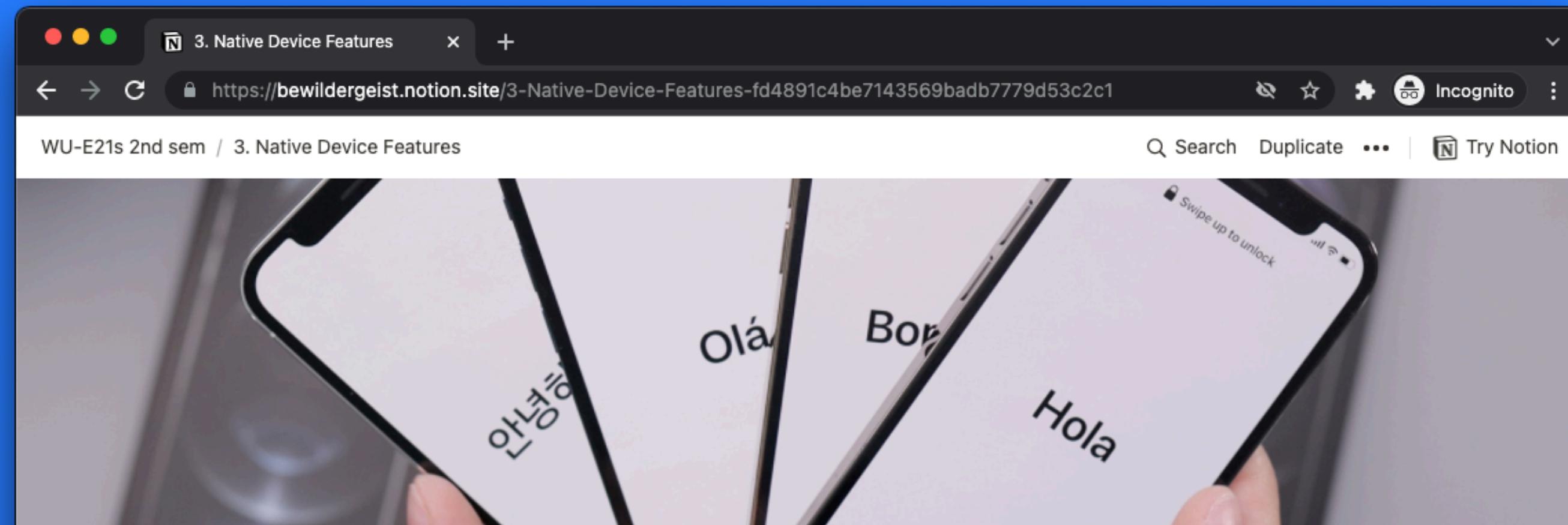
<https://ionicframework.com/docs/api/modal#usage>

# Ionic Camera App

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



# Next Thursday



The image shows a Notion page titled "3. Native Device Features". The page header includes the URL <https://bewildergeist.notion.site/3-Native-Device-Features-fd4891c4be7143569badb7779d53c2c1> and an "Incognito" button. The main content features a large image of three smartphones side-by-side, each displaying a different greeting in a different language: "Olá" (Portuguese), "Boas" (Portuguese), and "Hola" (Spanish). Below the image, the section title "3. Native Device Features" is displayed in bold. Underneath the title, there are three metadata entries: "Date" (17/02/2022), "Teacher" (RACE), and "Course" (MAD). At the bottom of the page, under the heading "Agenda/ Themes", is a bulleted list of topics:

- Build, emulate & compile for the Web, Android & iOS
- Capacitor Setup
- Native Device Features
- Using the Native APIs
- Android Studio, Xcode & Livereload
- Ionic CLI & Tools