

Mobile Dev 1

CS571: Building User Interfaces

Cole Nelson & Yuhang Zhao

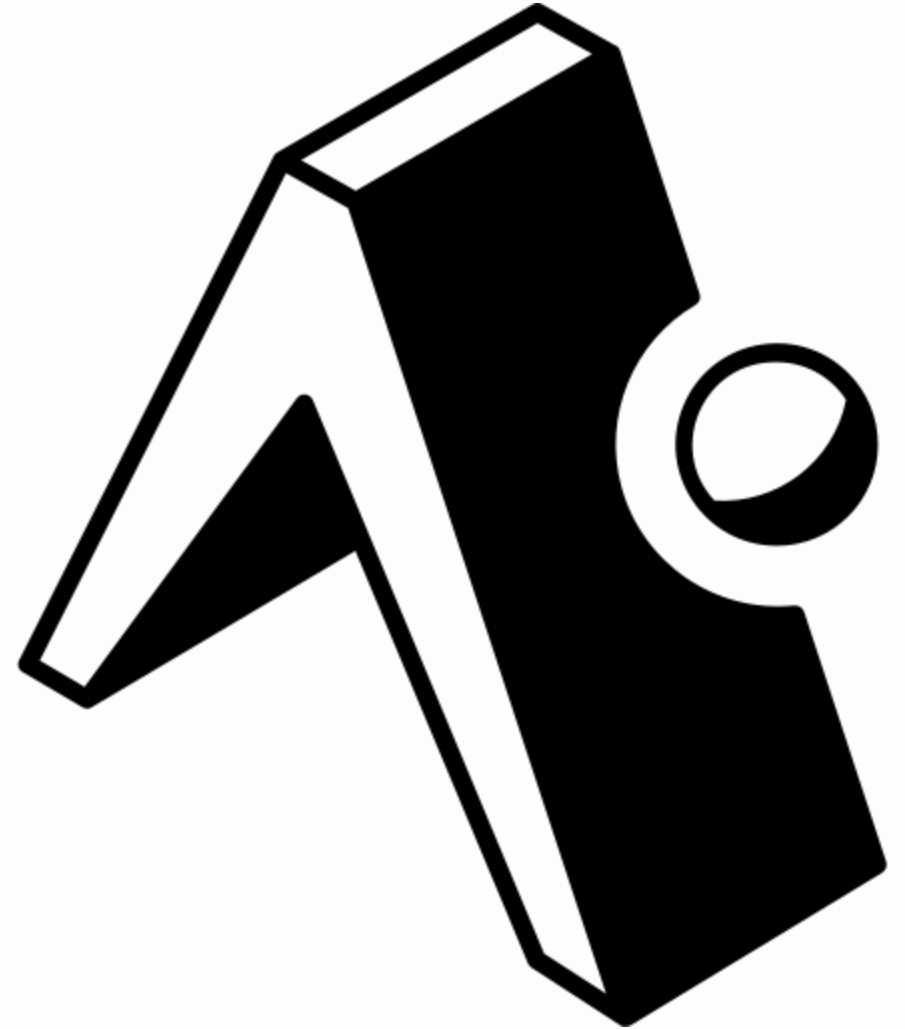
Today's Warmup

- Download Expo for your mobile device (see next slide for details).
- Clone `today's code` to your machine.
 - Run the command `npm install` inside of the `starter` and `solution` folders.
- **Optional:** Set an environment variable called `EXPO_PUBLIC_CS571_BADGER_ID` to be your Badger ID!
 - This may require a restart. For today, just hardcode your Badger ID!

Download Expo

Download Expo!

- Download for [iOS](#)
- Download for [Android](#)
- Don't have a smart phone? You can use an emulator like [AVD](#) or [XCode](#)



Learning Objectives

1. Understand the landscape of mobile development.
2. Be able to identify how "true native" development differs from "React Native" development.
3. Be able to construct a basic, cross-platform mobile application using React Native & Expo.

Mobile Development

Native development and its alternatives

What is "True Native" Development?

Building specifically for the device (e.g. Android or iOS) that you want to support.

iOS: Objective-C or Swift w/ Cocoapods

Android: Java or Kotlin w/ Maven or Gradle

Pros and Cons of True Native

Pros

- Organic User Experience
- Optimized Apps
- Fine-Grained Control

Cons

- Expensive
- Little Code Reuse
- Less Sense of Abstraction

Alternatives to True Native

No mobile app! Do we really need an app? Could a responsive webpage be just as effective?

WebView! Can we take our existing code and just slap it into a WebView? e.g. Apache Cordova

Cross-Platform! Can we use a library or framework that will make our code work natively on Android *and* iOS? e.g. React Native

Who is using React Native?

- Facebook
- Microsoft
- Shopify
- Coinbase
- Discord
- Dave

... among many others. Other companies may be doing pure-native or hybrid development.

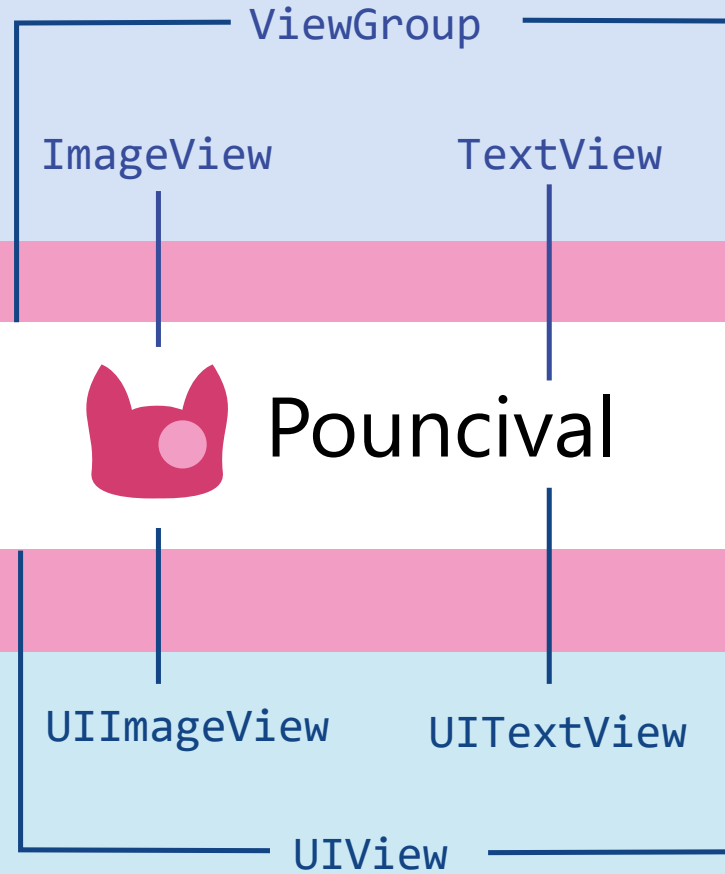
What is React Native?

A JS framework for building native, cross-platform mobile applications using React, developed by Facebook in 2015.

Unlike ReactJS, which was a library, React Native is a framework that includes everything* that we will need to build mobile applications.

React Native supports iOS and Android development.

Android



iOS

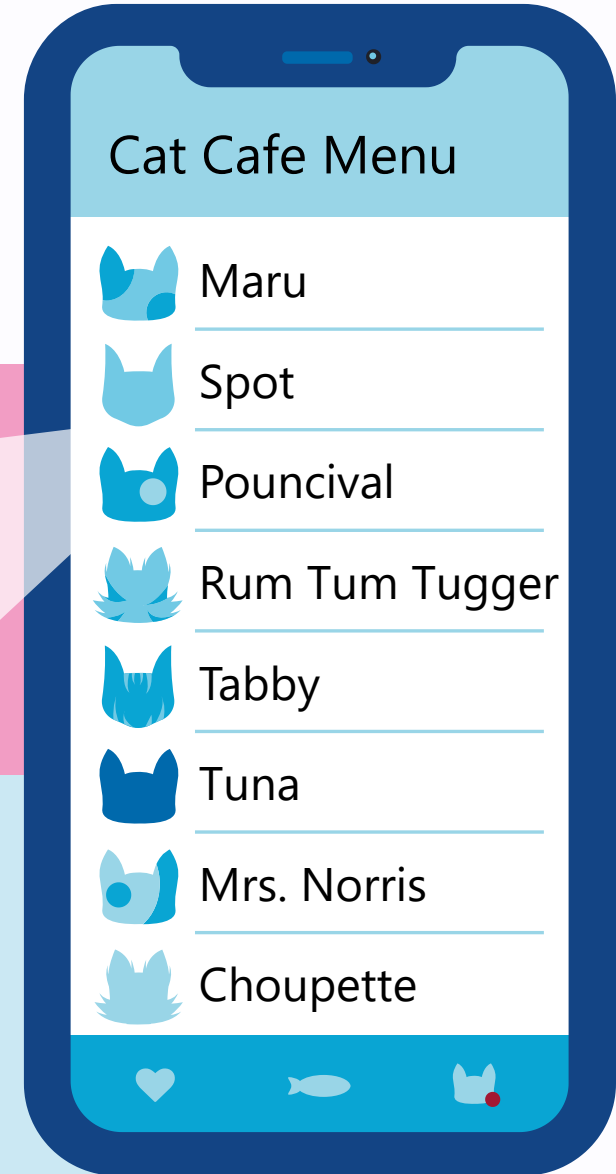


Image Source

React Native

- No more browser!
- No more DOM!
- Hermes is used to translate your JS components to iOS/Android components.

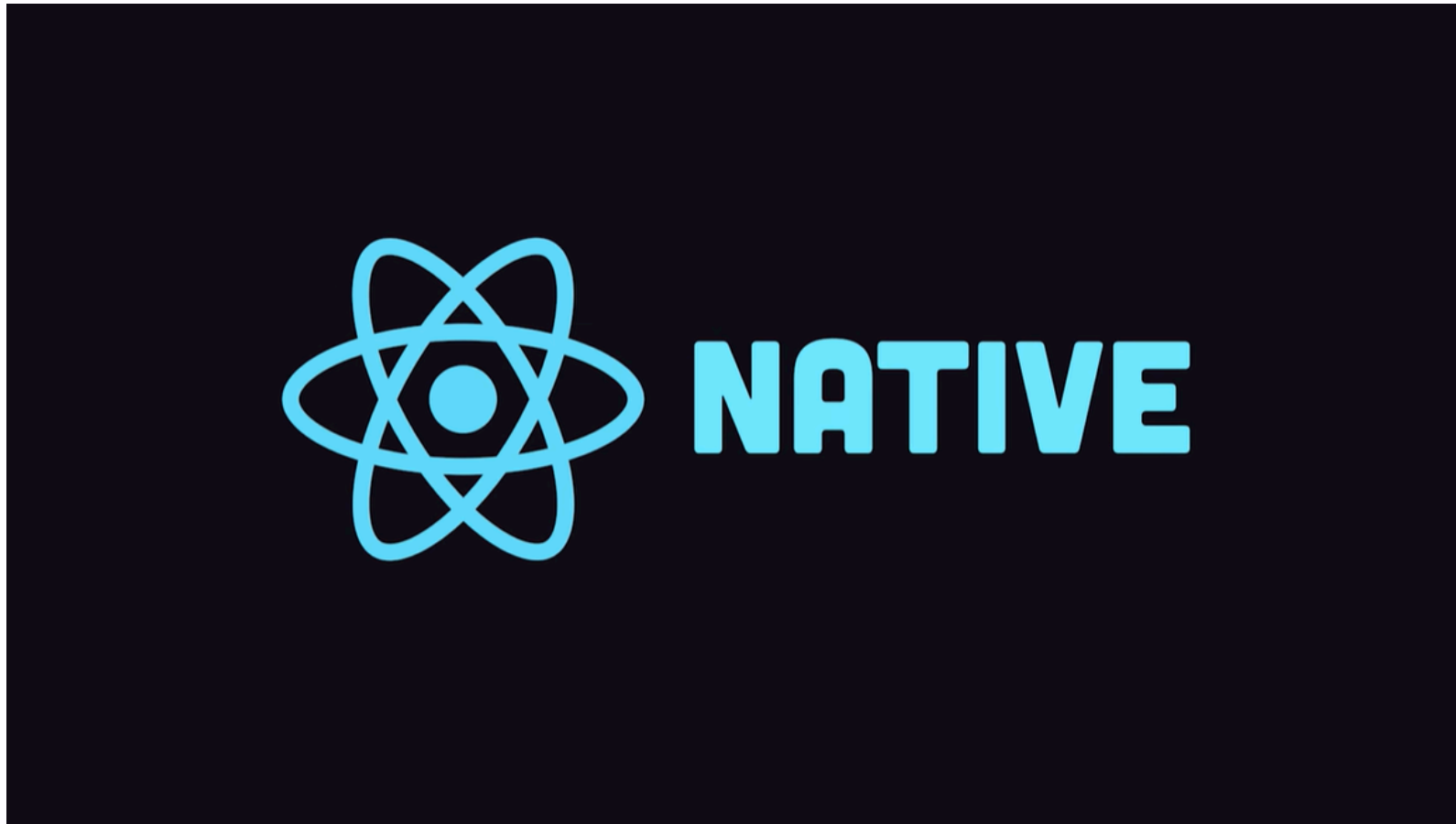
Image Source

CS571 Building User Interfaces | Cole Nelson & Yuhang Zhao | Fall 2024 | Mobile Dev 1



React Native

React for Mobile Devices!



React Native in 100 seconds

A Review of *Implementation* So Far

Lecture Takeaway

Intro	The web runs on HTTP
-------	----------------------

WDB1	Basics of HTML, CSS, and JS
------	-----------------------------

WDB2	APIs and Asynchronous Programming
------	-----------------------------------

WDB3	Declarative Programming and Bootstrap
------	---------------------------------------

A Review of *Implementation* So Far

Lecture Takeaway

React 1	Intro, useState, and useEffect
---------	--------------------------------

React 2	Lifecycle and Controlled Inputs
---------	---------------------------------

React 3	State Management, Context, and Routing
---------	--

React 4	Complex APIs and Secret Management
---------	------------------------------------

React 5	Memoization and Deployment
---------	----------------------------

What stays the same?

- Using NPM for our library management
- Using complex APIs
- Core React features
 - React Hooks (useEffect, useState, etc.)
 - Passing props and state management
 - Controlled vs Uncontrolled Inputs
 - Memoization

What changes?

- This isn't a browser!
 - No more DOM!
 - No more CSS!
 - No more Bootstrap!
 - No more sessionStorage, localStorage, or cookies.
- Wider variety of inputs
 - Sensors
 - Gestures
- **React Navigation** vs **React Router**

Conversions to Know

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
<code><View></code>	<code><ViewGroup></code>	<code><UIView></code>	A non-scrolling <code><div></code>	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
<code><Text></code>	<code><TextView></code>	<code><UITextView></code>	<code><p></code>	Displays, styles, and nests strings of text and even handles touch events
<code><Image></code>	<code><ImageView></code>	<code><UIImageView></code>	<code></code>	Displays different types of images
<code><ScrollView></code>	<code><ScrollView></code>	<code><UIScrollView></code>	<code><div></code>	A generic scrolling container that can contain multiple components and views
<code><TextInput></code>	<code><EditText></code>	<code><UITextField></code>	<code><input type="text"></code>	Allows the user to enter text

Image Source

Other Good Questions to Ask...

- Can we declaratively program using RN? **YES**
- Can we use JSX with RN? **YES**
- Can we use React hooks in RN? **YES**
- Can we do styling in RN? **YES**-ish
- Is it *truly* cross-platform? **MAYBE**-ish

Hello World!

```
import React from 'react';
import { Text, View } from 'react-native';

function MyApp() {
  return (
    <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
      <Text>
        Try editing me! 🎉
      </Text>
    </View>
  );
}

export default MyApp;
```

React Native for React Devs

How can we write our mobile apps with React Native?

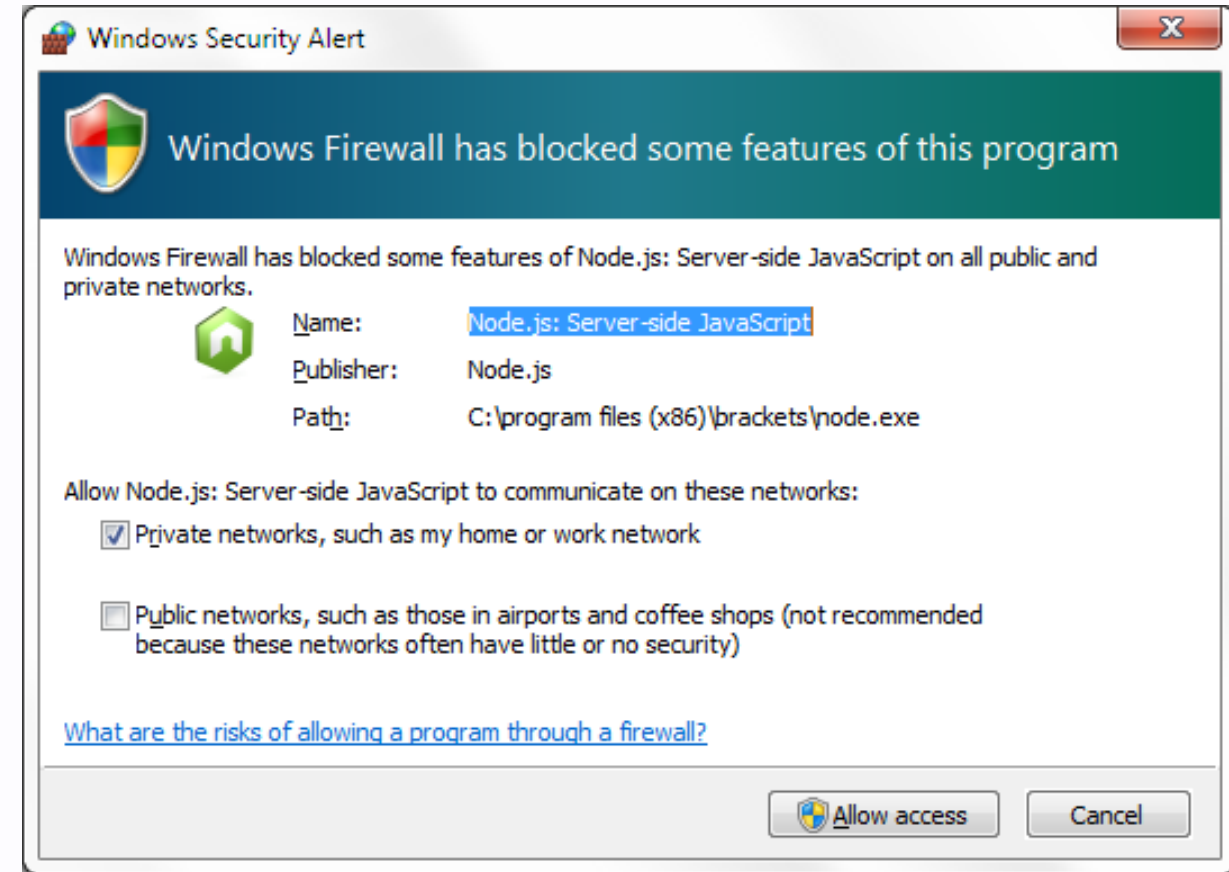
Getting Started

Using **Expo**, similar to vite!

```
npm install  
npm start
```

Getting Started: A Special Note

By default, Expo uses "lan" to host your app. Your computer will act like a server for your phone; be sure to allow connections!



Getting Started: A Special Note

This may cause issues on certain networks. Try using...

If you can wire in...

```
npm start -- --localhost
```

If you want to host over the internet...

```
npm start -- --tunnel
```

Styling

Because React Native does not use a "browser", we can't use CSS styles. Instead, we create JavaScript stylesheets. **These try to emulate CSS.**

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    backgroundColor: '#ecf0f1',
    padding: 40,
  },
  ...
});
```

Styling

Style definitions can be done inline or via stylesheets. You can also combine both methods.

```
<View>  
  <Text style={styles.label}>First label</Text>  
  <Text style={{fontSize: 28, color:'tomato'}}>Second label</Text>  
  <Text style={[styles.label, {fontSize: 20, color:'gray'}]}>Third label</Text>  
</View>
```

Snack Solution

Images

`Image` not `img` (must be imported!)

Must specify a width and height: the default is 0!

`source` not `src` which takes an object (not a string)

```
<Image
  style={{
    width: 100,
    height: 100
  }}
  source={{
    uri: "https://example.com/me.png"
  }}
/>
```

Buttons

Some minor changes...

- `title` is specified with a prop
- `onPress` rather than `onClick`
- Cannot be styled

```
<Button title="Speak!" onPress={doSpeak}/>
```

Also, anything can be made a `Pressable` ... we'll cover this in a bit!

Your Turn!

Using [today's starter code](#)...

1. Get your expo app running!
2. Display the bio data to the phone screen.
3. When the button is pressed, display a message from the API.

<https://cs571api.cs.wisc.edu/rest/f24/ice/mascot>

<https://cs571api.cs.wisc.edu/rest/f24/ice/mascot-messages>

Cross-Platform: By Size

Mobile devices vary significantly in screen size, and we often need to obtain screen dimensions of the device using the `Dimensions` class in `react-native`.

```
const getScreenSize = () => {  
  const screenWidth = Math.round(Dimensions.get('window').width);  
  const screenHeight = Math.round(Dimensions.get('window').height);  
  return { screenWidth: screenWidth, screenHeight: screenHeight };  
}
```

Snack Solution

Cross-Platform: By Platform

React Native provides a number of components that utilize platform capabilities that may not be available in other platforms, thus for cross-platform development, we need to utilize multiple platform-specific components.

e.g. `TouchableNativeFeedback` only works on Android; a *similar* effect can be achieved using `TouchableHighlight` on iOS.

Cross-Platform: By Platform

```
if (Platform.OS === 'android') {  
  return (  
    <TouchableNativeFeedback> ... </TouchableNativeFeedback>  
  );  
} else {  
  return (  
    <TouchableHighlight> ... </TouchableHighlight>  
  );  
}
```

Optionally, create two components e.g.

`MyButton.ios.js` and `MyButton.android.js`.

Snack Solution

Pressable

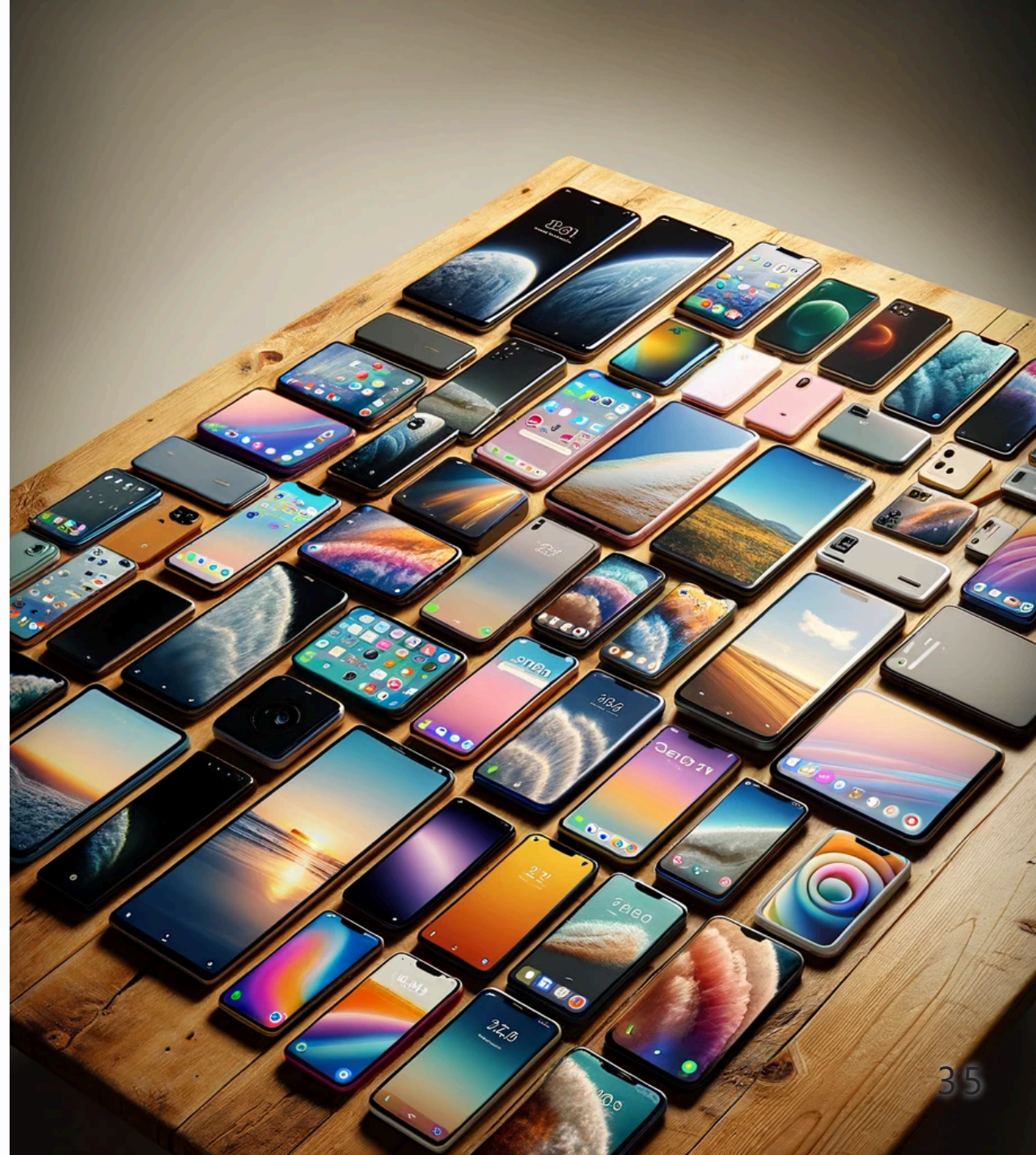
The cross-platform variant! May contain any children that can be "pressed"

```
<Pressable onPress={props.onPress}>
  <Image
    style={{ width: 100, height: 100 }}
    source={{
      uri: "https://example.com/me.png"
    }}
  />
  <Text>Press me!</Text>
</Pressable>
```

Pressable

Mobile Dev HWs

It's difficult to test cross-platform; show us how it works on your device via a demo!



Questions?