React Native 1

CS571: Building User Interfaces

Cole Nelson

What will we learn today?

- An Overview of Mobile Development
- An Introduction to React Native
- Programming w/ React Native & Expo

Mobile Development

Native Development and its Alternatives

What is "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 Native Development

Pros

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

Cons

- Expensive
- Little Code Reuse
- Less Sense of Abstraction

Alternatives to Native Development

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

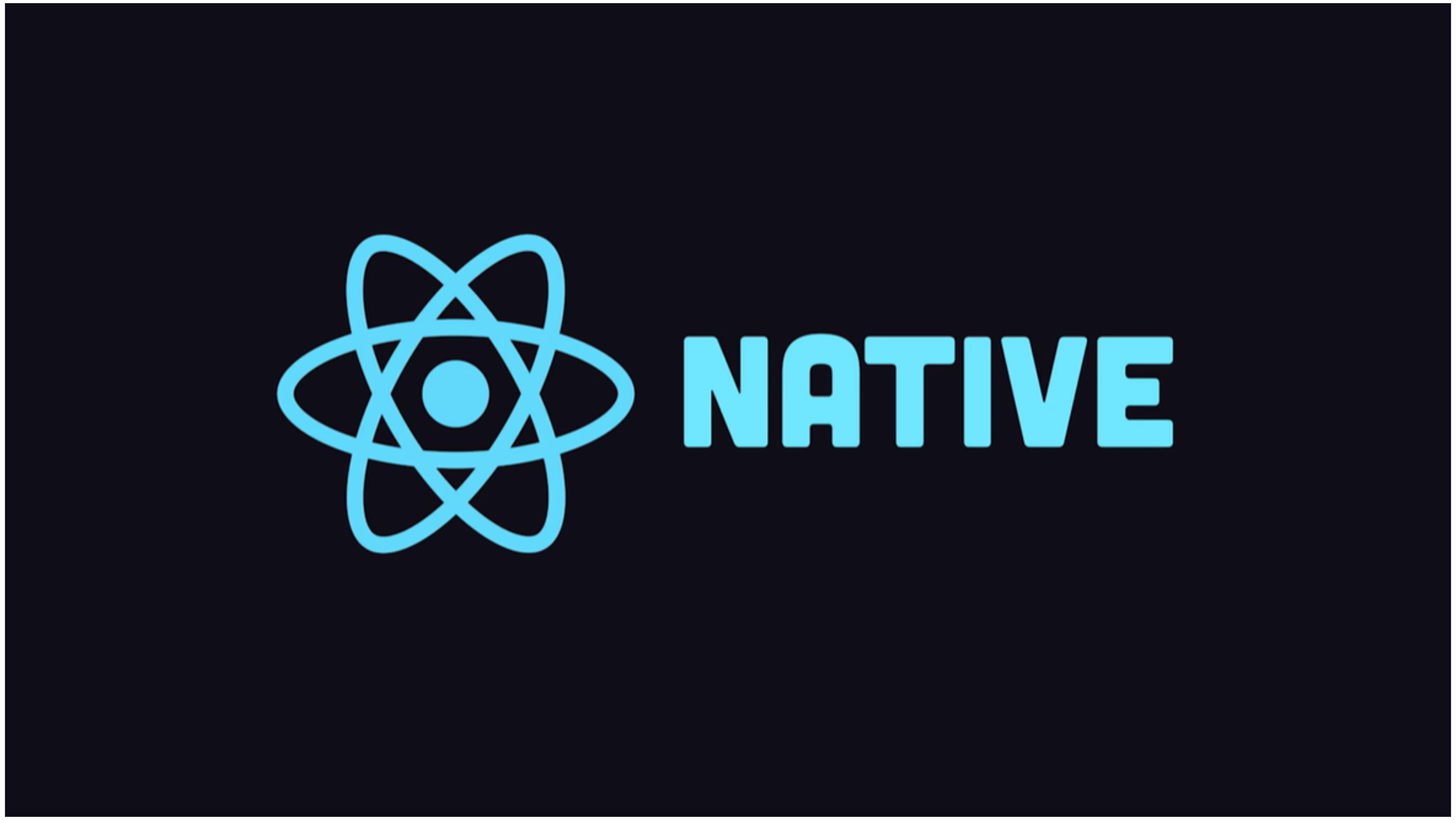
Who is using React Native?

- Facebook
- Microsoft
- Shopify
- Coinbase
- Discord

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

React Native

React for Mobile Devices!



React Native in 100 seconds

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.

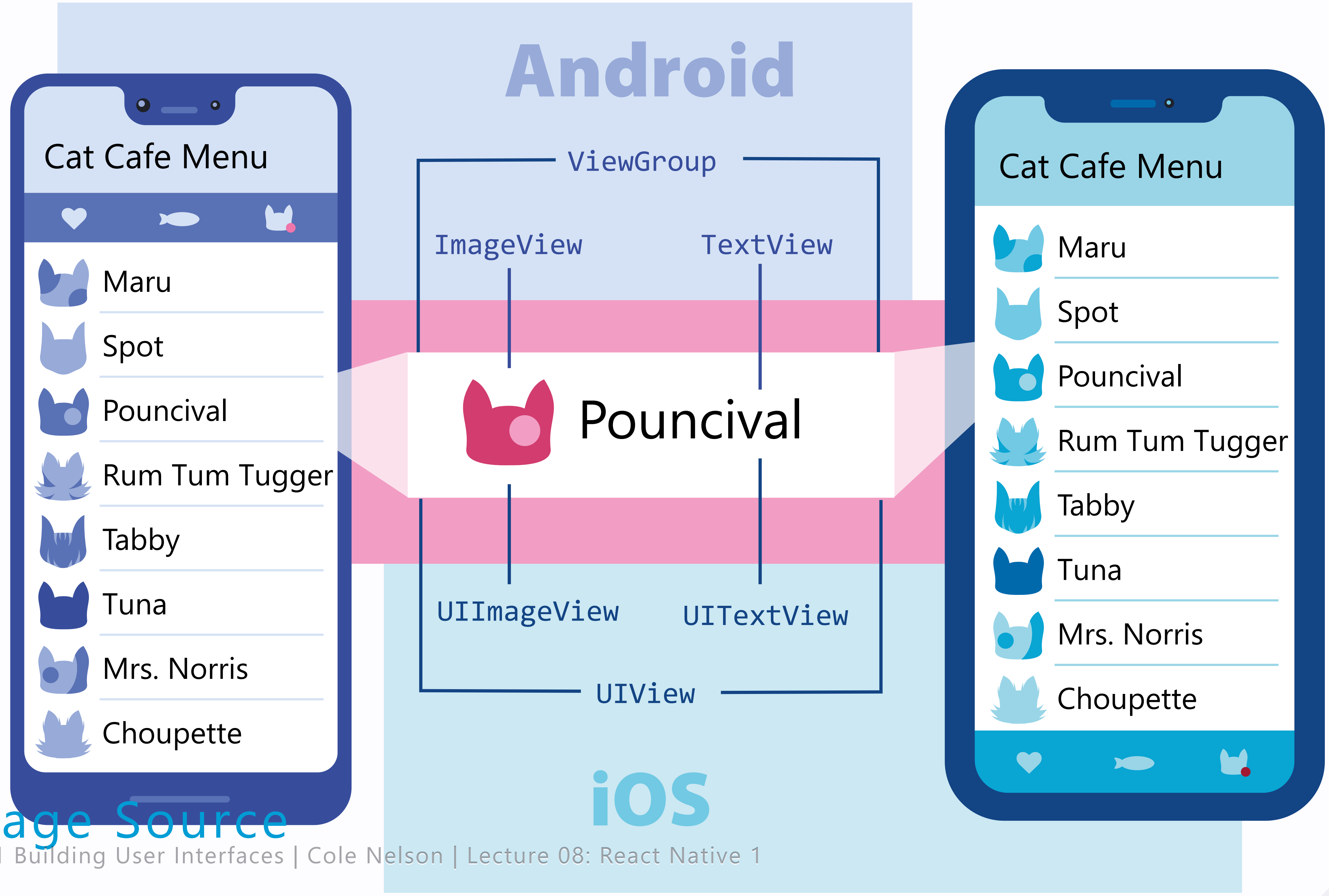
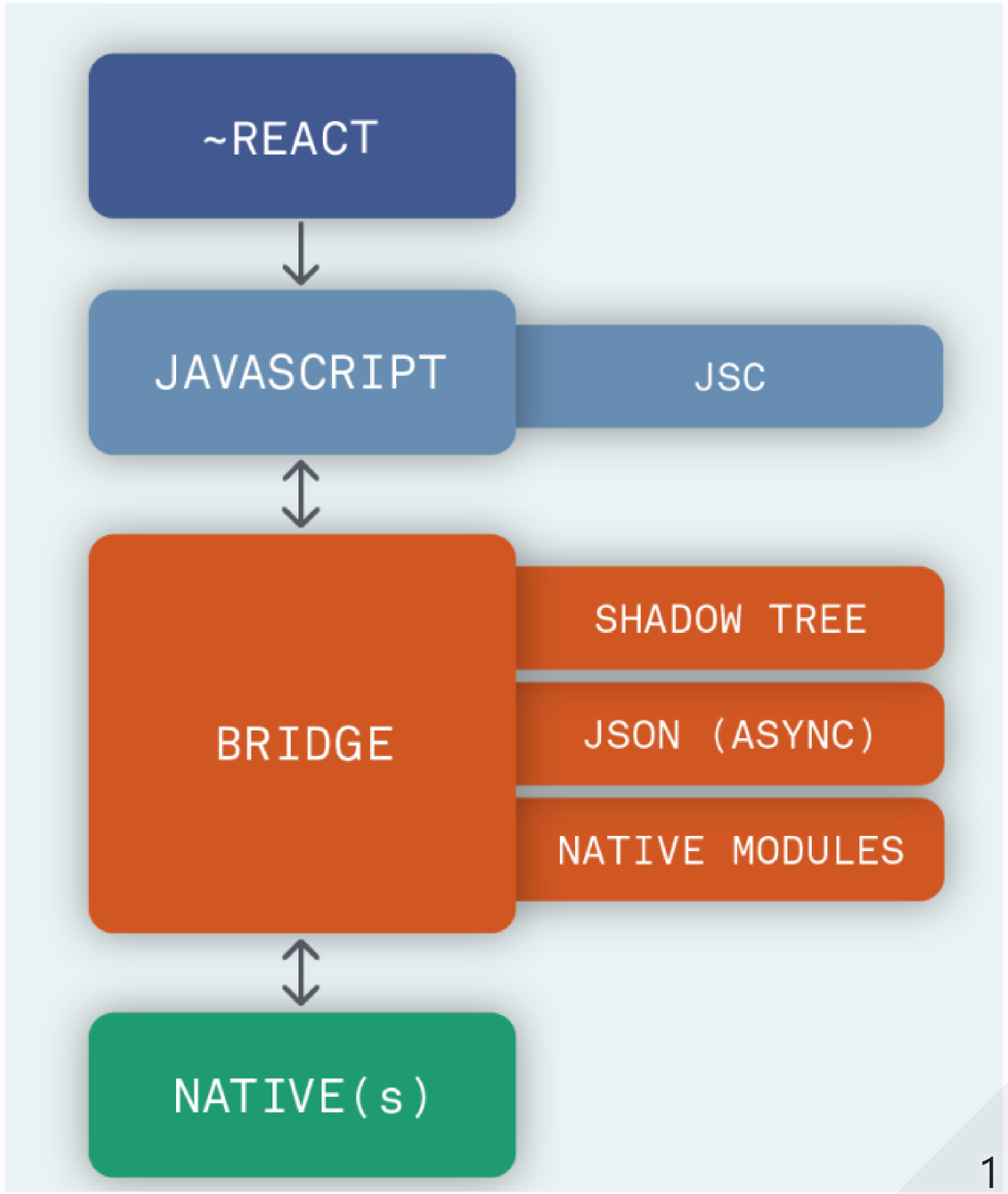


Image Source

React Native

- No more DOM or browser capabilities!
- Connects with native components using a "bridge"

Image Source



React Native

- The use of a bridge causes a slight hit to performance.
- Will soon be remedied with "The New Archeticture" and Hermes!

Image Source

React Native for React Devs

How can we write our mobile apps with React Native?

Getting Started

Using Expo, similar to create-react-app!

Run one-time...

```
npm install expo-cli --global
```

Run for each project...

```
expo init my-new-project
cd my-new-project
npm start
```

Getting Started: A Special Note

By default, expo uses "lan" to host your app. This may cause issues on certain networks. Try using "localhost" or "tunnel" by modifying scripts of `package.json` ...

```
"scripts": {
  "start": "expo start --localhost",
  "android": "expo start --android",
  "ios": "expo start --ios",
  "web": "expo start --web"
}
```

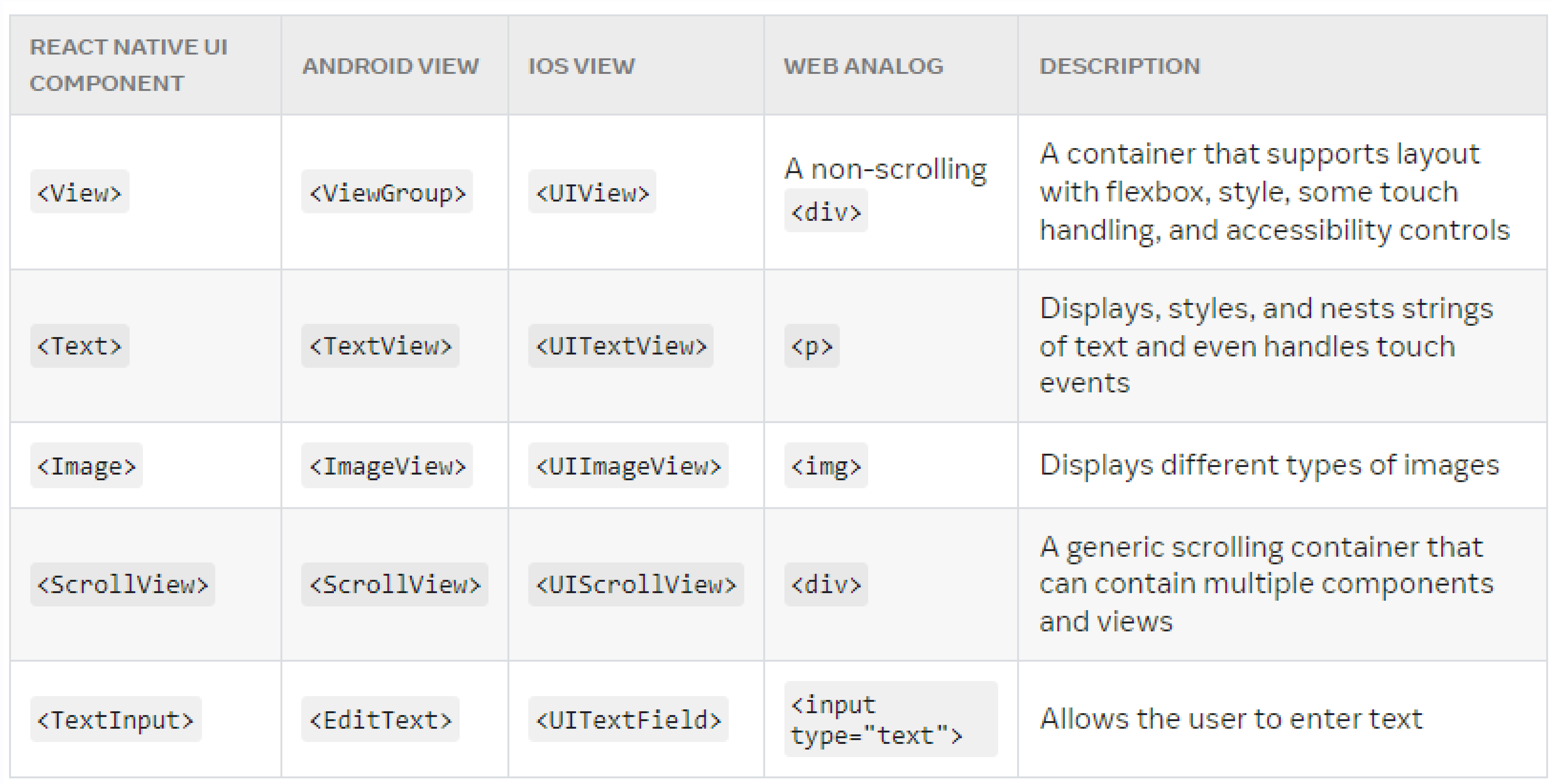
Expo Demo

Setting up your first React Native app!

Download Expo for iOS or Android.

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
- Can we use cookies, sessionStorage, and localStorage in RN? **NO**



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

Image Source

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;
```

Code Source

Styling

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

```
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

CROSS-PLATFORM

"Cross-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.

Differentiating 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

Cross-Platform: Dimensions

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

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

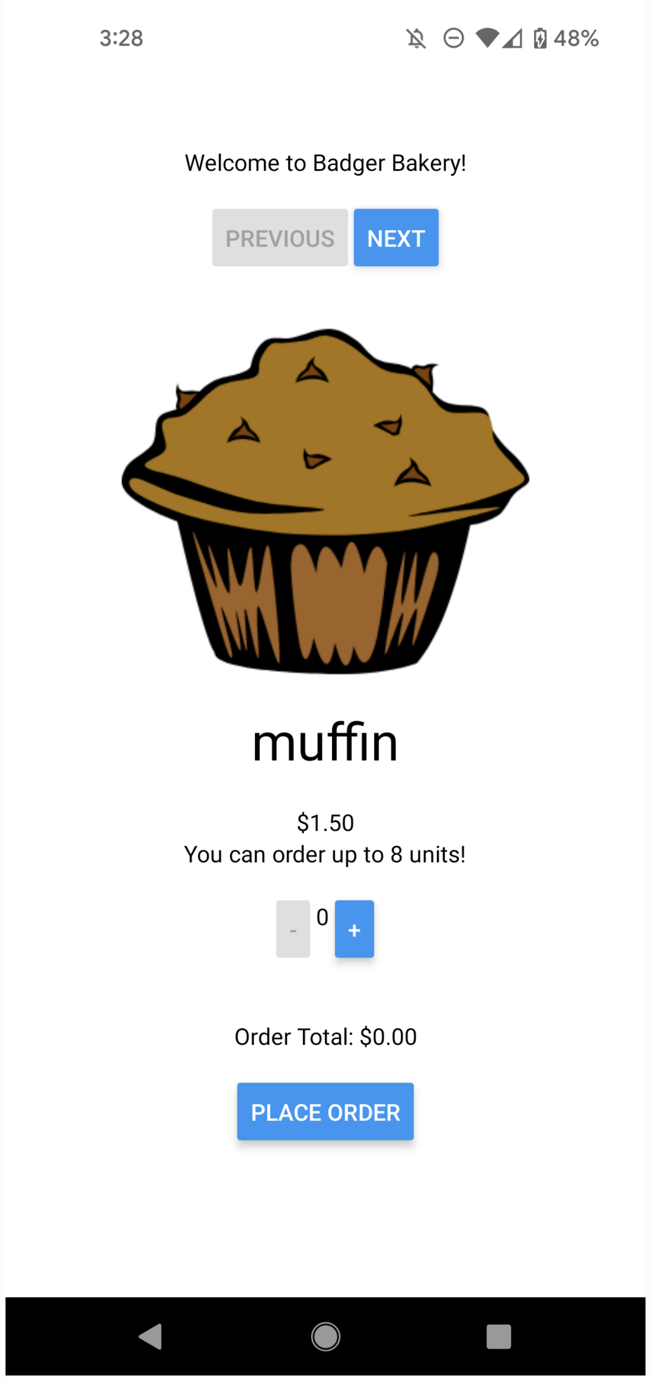
"Find My Badger" Demo

Everything that we learned in React... w/ React Native!

Snack Solution

Homework

- Much more open-ended!
- Requires you to record a demo.
- Start early!



Welcome to Badger Bakery!

muffin

$1.50
You can order up to 8 units!

Order Total: $0.00

What did we learn today?

- An Overview of Mobile Development
- An Introduction to React Native
- Programming w/ React Native & Expo

On to Mobile Design! 🚀