Ubiquitous and Mobile Computing
CS 528: React Native
Google Analytics

Nathan Hsu, Jonathan Wang, Wen Ge, Maoyu Chien, Guangda Li

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
React Native
A brief history of React Native

1. **SUMMER 2013**
   - Facebook's Hackathon
   - React Native started as Facebook’s internal Hackathon project.

2. **JANUARY 2015**
   - Public preview
   - At react.js con, React Native had its first public preview.

3. **MARCH 2015**
   - Open-source
   - At F8 conference, Facebook announced React Native is open and available on Github.

4. **NOVEMBER 2015**
   - Shoutem & React
   - After giving a lot of thoughts, we decided to switch Shoutem from our current technologies to React Native.

5. **SEPTEMBER 2016**
   - UI Tollkit
   - On september 15th, we launched our open-source UI toolkit for React Native.

6. **SEPTEMBER 2016**
   - Trending at the top
   - For the first time in the past 12 months, React Native surpassed iOS and Android development according to Google trends.

Making apps should be easy.
Why React Native

- Build native app using Javascript, which is ubiquitous nowadays.
- A React Native app is a real mobile app. Not HTML or Hybrid app.
  - WebKit (or similar technology) is still too heavy for mobile device!
- Reload the parts that have been changed without re-compiling entire project.
- Use native code when you need to: Use `Platform.OS === 'android'` and `Platform.OS === 'ios'` to separate platform-specific code.
- Therefore, developers can shared codebase for iOS and Android! (sort of...)
Who use React Native

- Facebook
- Dropbox
- Slack
- Instagram
- F8
- Boombox
- Pinterest
- Skype
- Tesla
- Uber
- Walmart
- Wix
Bloomberg

Search Smarter
Find the answers you need faster with intelligent search.

Make It Yours
Prioritize the features you need most.

Connect Instantly
Tap into 24/7 live customer support and connect to your network with secure collaboration tools.
What is React Native

- Developers create UI in the same way as React (see below).
- React native already has many built-in components: ListView, Image, Button, Picker, TextInput, ... etc.
- It also support Geolocation, various touch events handling.

As the image shown in the right, the declarative UI code (aka. component) is mixed with logic code.
How React Native Works

```
React Component
render: function() {
    return <div>Hi!</div>;
}
```

```
React Component
render: function() {
    return <View>Hi!</View>;
}
```
Native

Don't use it

Use it
When to use React Native

- Small and Poor
- The app doesn’t require platform specific toolkit, e.g. ARkits
- The app doesn’t require user to do interaction intensively, e.g. Social app, e-commerce app, ...etc
And when not to use it

- Big and Rich
- Apps having complex interaction, e.g. Gestures
- Apps that have to communicate with its background services heavily, e.g. rely on binder to do IPC
Airbnb dropped React native

- React Native Immaturity
- React Native Open Source Libraries
- Hard to Debug
Let’s learn some React Native!
Hello World!

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

export default class HelloWorldApp extends Component {
  render() {
    return (
      <View>
        <Text>Hello world!</Text>
      </View>
    );
  }
}
```
class ShoppingList extends React.Component {

    constructor(props) {
        super(props);
    }

    render() {
        return (
            <View>
                <Text>Shopping List for {this.props.name}</Text>
                <FlatList
                    data={this.props.Items}
                    renderItem={({item}) => <Text>{item.key} </Text> }
                />
            </View>
        );
    }
}

export default class App extends Component {
    render() {
        return (
            <View>
                <ShoppingList
                    name="alvin"
                    Items={[{key: "coffee"}, {key: "candy"}]} />

                <ShoppingList
                    name="john"
                    Items={[{key: "beer"}, {key: "egg"}]} />
            </View>
        );
    }
}