React – native

In this lab, you will create a React app to run on mobile devices.

# Objectives

In this lab, you will

* Install react-native on your laptop
* Create a mobile app
* Create the TODOs app for mobile
* Run the app

# Create the First Mobile App

1. NOTE: Xcode must be installed on the Mac computers to create iOS apps. You should have received an email detailing machine setup for this part of the course.
2. Change to the lab folder and run

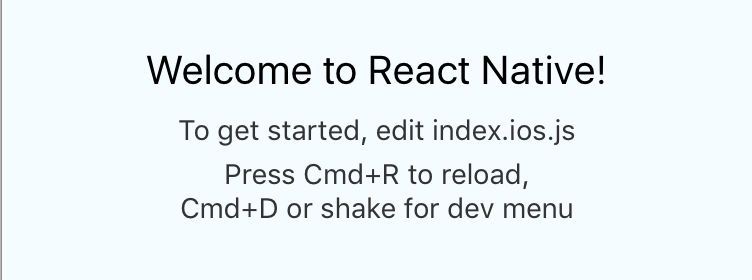
npm install –g react-native-cli

react-native init TodoList

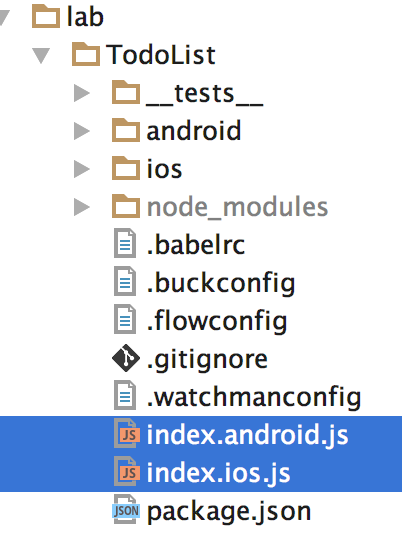
cd mobile

react-native run-ios

1. Be patient. Eventually the mobile phone emulator will appear with the following message:



1. Notice the files created:



1. The two JS files indicated are identical except for the text messages displayed.
2. Edit the index.ios.js file, add the following text line (in yellow)

**export default class** TodoList **extends** Component {  
 render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**Text style=**{***styles***.**welcome**}>  
 Welcome to React Native!  
 </**Text**>  
 <**Text style=**{***styles***.**instructions**}>  
 To get started, edit index.ios.js  
 {**'\n'**}  
 This is FUN!!!  
 </**Text**>  
 <**Text style=**{***styles***.**instructions**}>  
 Press Cmd+R to reload,{**'\n'**}  
 Cmd+D or shake for dev menu  
 </**Text**>  
 </**View**>  
 );  
 }  
}

1. Type CMD-R to refresh the app. You should see your changes on the emulator.
2. Wow, the emulator works great. Let’s create the TodoList app.

# Create the TodoList App

1. Edit the /index.ios.js and /index.android.js files and replace the contents to the following:

*/\*\*  
 \* Sample React Native App  
 \* https://github.com/facebook/react-native  
 \** ***@flow*** *\*/***import** {  
 AppRegistry  
} **from 'react-native'**;  
  
**import** TodoList **from './TodoList'**;  
  
AppRegistry.registerComponent(**'TodoList'**, () => TodoList);

1. NOTE: both files contain the same content. We will create the TodoList component and import it into each mobile file.
2. Let’s create the TODO app. Create a file, /TodoList.js and add the following content.

*/\*\*  
 \* Sample React Native App  
 \* https://github.com/facebook/react-native  
 \** ***@flow*** *\*/***import** React, { Component } **from 'react'**;  
**import** {  
 StyleSheet,  
 Text,  
 View,  
 ListView,  
 TextInput,  
 ActivityIndicator,  
 TouchableOpacity,  
 TouchableHighlight,  
 Image  
} **from 'react-native'**;  
  
**export default class** TodoList **extends** Component {  
  
 constructor(props) {  
 **super**(props);  
 **this**.**state** = {  
 **data**: [**'Todo 1'**, **'Todo 2'**, **'Todo 3'**, **'Todo 5'**, **'Todo 6'**, **'Todo 9'**],  
 **text**: **''** };  
 }  
  
 render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**View style=**{***styles***.**navbar**}>  
 <**Text style=**{***styles***.**welcome**}>  
 React-Native ***Todo*** </**Text**>  
 </**View**>  
 </**View**>  
 );  
 }  
}  
  
**const *styles*** = StyleSheet.**create**({  
 **container**: {  
 **flex**: 1,  
 **justifyContent**: **'center'**,  
 **padding**: 10,  
 **backgroundColor**: **'#F5FCFF'** },  
 **input**: {  
 **height**: 40,  
 **borderColor**: **'gray'**,  
 **borderWidth**: 1,  
 **marginBottom**: 20, **paddingLeft**: 10  
 },  
 **welcome**: {  
 **fontSize**: 20,  
 **textAlign**: **'center'**,  
 **margin**: 10,  
 **color**: **'#666666'**,  
 **fontWeight**: **'bold'** },  
 **instructions**: {  
 **textAlign**: **'center'**,  
 **color**: **'#333333'**,  
 **marginBottom**: 5  
 },  
 **navbar**: {  
 **paddingTop**: 17,  
 **height**: 70  
 },  
 **rowContainer**: {  
 **flex**: 1,  
 **flexDirection**: **'row'**,  
 **padding**: 5  
 },  
 **leftContainer**: {  
 **flex**: 20  
 },  
 **rightContainer**: {  
 **flex**: 1  
 }  
});

1. The above creates a simple React Component that displays the title on the screen. Note that most of the lines are the styles used in the UI. Save the files and type CMD-R to refresh the app.
2. Notice the constructor creates the state with two variables, data which holds the TODO list strings, and text which will hold the input text.
3. Now change the render() method and add the renderRows() method with the following:

renderRows() {  
 **return** (  
 <**View**>  
 {**this**.**state**.**data**.map((item, idx) => <**Text key=**{idx}>{item}</**Text**>)}  
 </**View**>  
 )  
 }  
  
 render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**View style=**{***styles***.**navbar**}>  
 <**Text style=**{***styles***.**welcome**}>  
 React-Native ***Todo*** </**Text**>  
 </**View**>  
 <**View**>  
 {**this**.renderRows()}  
 </**View**>  
 </**View**>  
 );  
 }

1. Refresh the emulator and see the list of TODO items.
2. Let’s add the text input for adding new TODO items. Change the render() method to the following.

render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**View style=**{***styles***.**navbar**}>  
 <**Text style=**{***styles***.**welcome**}>  
 React-Native ***Todo*** </**Text**>  
 </**View**>  
 <**TextInput  
 style=**{***styles***.**input**}  
 **onSubmitEditing=**{**this**.submitEdit.bind(**this**)}  
 **onChangeText=**{(text) => **this**.setState({text})}  
 **placeholder="Enter TODO text here..."  
 value=**{**this**.**state**.**text**}  
 />  
 <**View**>  
 {**this**.renderRows()}  
 </**View**>  
 </**View**>  
 );  
}

1. Add the event handler, submitEdit(), after the render() method and refresh the emulator:

submitEdit(event) {  
 **let** data = **this**.**state**.**data**;  
 **let** text = event.**nativeEvent**.**text**;  
 **this**.setState({  
 **data**: data.concat(text),  
 **text**: **''** });  
}

1. Enter a couple of TODOs to test the app.
2. Managing the list of components is so common that react-native has a core component, ListView, to handle it. Let’s refactor our application to use ListView.
3. ListView needs a DataSource to wrap the input data. Let’s add it in the object as follows:

**export default class** TodoList **extends** Component {  
  
 **dataSource** = **new** ListView.**DataSource**({rowHasChanged: (r1, r2) => r1 !== r2})  
  
 constructor(props) {

1. Now change the render() method to use the ListView as follows:

render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**View style=**{***styles***.**navbar**}>  
 <**Text style=**{***styles***.**welcome**}>  
 React-Native ***Todo*** </**Text**>  
 </**View**>  
 <**TextInput  
 style=**{{**height**: 40, **borderColor**: **'gray'**, **borderWidth**: 1, **marginBottom**: 20}}  
 **onSubmitEditing=**{**this**.submitEdit.bind(**this**)}  
 **onChangeText=**{(text) => **this**.setState({text})}  
 **placeholder="Enter TODO text here..."  
 value=**{**this**.**state**.**text**}  
 />  
 <**ListView  
 dataSource=**{**this**.**dataSource**.cloneWithRows(**this**.**state**.**data**)}  
 **renderRow=**{(rowData) => <**Text**>{rowData}</**Text**>} />  
 </**View**>  
 );  
}

1. Notice the renderRow attribute above wraps the row data with a <Text> component. Refresh the emulator and add some more TODO items.

# Add the Delete Button

1. Add the following methods to add a delete button to the application. Add the following code above the render() method.

\_onPressButton(rowData, rowID, sectionID) {  
 **let** data = **this**.**state**.**data**;  
 data.splice(sectionID, 1);  
 **this**.setState({  
 **data**: data  
 });  
}  
  
renderRow(rowData, rowID, sectionID) {  
 **return** (  
 <**View style=**{***styles***.**rowContainer**}>  
 <**View style=**{***styles***.**leftContainer**}>  
 <**Text style=**{***styles***.**text**}>  
 {rowData}  
 </**Text**>  
 </**View**>  
 <**View style=**{***styles***.**rightContainer**}>  
 <**TouchableOpacity  
 onPress=**{() => **this**.\_onPressButton(rowData,rowID, sectionID)}>  
 <**Image source=**{***require***(**'./delete.png'**)}/>  
 </**TouchableOpacity**>  
 </**View**>  
 </**View**>  
 )  
}

1. The above will render one TODO item along with the delete button. The ListView component will call this instead of it’s own render function.
2. Copy the /solution/TodoList/delete.png file to the /lab/TodoList folder. This file contains the image of the delete button.
3. Modify the ListView component to use the above renderRow() method as follows:

render() {  
 **return** (  
 <**View style=**{***styles***.**container**}>  
 <**View style=**{***styles***.**navbar**}>  
 <**Text style=**{***styles***.**welcome**}>  
 React-Native ***Todo*** </**Text**>  
 </**View**>  
 <**TextInput  
 style=**{***styles***.**input**}  
 **onSubmitEditing=**{**this**.submitEdit.bind(**this**)}  
 **onChangeText=**{(text) => **this**.setState({text})}  
 **placeholder="Enter TODO text here..."  
 value=**{**this**.**state**.**text**}  
 />  
 <**ListView  
 dataSource=**{**this**.**dataSource**.cloneWithRows(**this**.**state**.**data**)}  
 **renderRow=**{**this**.renderRow.bind(**this**)} />  
 </**View**>  
 );  
}

1. Save everything and refresh the emulator. Add and remove some TODO items.

Congratulations. You have completed this lab.