**Exploring Pressable**

In an earlier video, you explored how to use the Pressable component. You learned how to build a **ViewMenu** button within the Little Lemon app which displays the menu items when clicked. You will be using the **Pressable** component within the **MenuItems** component. This reading will thoroughly explore the code used to do this.

**Imports**

First, you will import the **Pressable** component from the React Native package. As it is a core component, it can be directly imported from the React Native package.

1

import { View, Text, StyleSheet, SectionList, Pressable } from 'react-native';

In order to keep track of the **ViewMenu** button state and the show or hide menu items, you should use the **useState** hook. This will be used to keep track of the local state of the **ViewMenu** button within this component.

Make sure to first import the **useState** hook to make it available.

1

import React, { useState } from 'react';

**Setting up Local State**

Within the **MenuItems** component, you can set up the local state as needed.

1

const [showMenu, setShowMenu] = useState(false);

Here **showMenu** is a Boolean, and its default value has been set to false. The **setShowMenu** can be used to update the value of **showMenu***.*

**Using the Pressable**

The **Pressable** component can accept a child or any number of children. These children can be any React Node, such as boolean, number, a React element, String, or an array of any of these.

Within the **MenuItems** component, the text **ViewMenu** is wrapped within the **Pressable** component. Also, the **onPress** handler calls the **setShowMenu** method to toggle the Boolean value of **showMenu***.*

When the user clicks on the **ViewMenu** text, the **showMenu** Boolean gets toggled to true immediately at the **onPress** handler of the **Pressable***.*

Also, notice that the text changes, based on the **showMenu** value. If the **showMenu** is true, the button text reads **Home** If it is false, then it would read **View Menu**.

Note that the **Pressable** component inherits all the styles of the **View** component, which means you can style it with all the styles you have available for a **View** component.

1

2

3

4

5

6

7

8

9

<Pressable

  style={menuStyles.button}

  onPress={() => {

    setShowMenu(!showMenu);

  }}>

    <Text style={menuStyles.buttonText}>

      {showMenu ? 'Home' : 'View Menu'}

    </Text>

</Pressable>

**Putting it all together**

Below is the complete code for **App.js** with the **Pressable** component inserted.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

import React, { useState } from 'react';

import { View, Text, StyleSheet, SectionList, Pressable } from 'react-native';

const menuItemsToDisplay = [

  {

    title: 'Appetizers',

    data: [

      'Hummus',

      'Moutabal',

      'Falafel',

      'Marinated Olives',

      'Kofta',

      'Eggplant Salad',

    ],

  },

  {

    title: 'Main Dishes',

    data: ['Lentil Burger', 'Smoked Salmon', 'Kofta Burger', 'Turkish Kebab'],

  },

  {

    title: 'Sides',

    data: [

      'Fries',

      'Buttered Rice',

      'Bread Sticks',

      'Pita Pocket',

      'Lentil Soup',

      'Greek Salad',

      'Rice Pilaf',

    ],

  },

  {

    title: 'Desserts',

    data: ['Baklava', 'Tartufo', 'Tiramisu', 'Panna Cotta'],

  },

];

const Separator = () => <View style={menuStyles.separator} />;

const Footer = () => (

  <Text style={menuStyles.footerText}>

    All Rights Reserved by Little Lemon 2022

  </Text>

);

const Item = ({ name }) => (

  <View style={menuStyles.innerContainer}>

    <Text style={menuStyles.itemText}>{name}</Text>

  </View>

);

const MenuItems = () => {

  const [showMenu, setShowMenu] = useState(false);

  const renderItem = ({ item }) => <Item name={item} />;

  const renderSectionHeader = ({ section: { title } }) => (

    <Text style={menuStyles.sectionHeader}>{title} </Text>

  );

  return (

    <View style={menuStyles.container}>

      {!showMenu && (

        <Text style={menuStyles.infoSection}>

          Little Lemon is a charming neighborhood bistro that serves simple food

          and classic cocktails in a lively but casual environment. View our

          menu to explore our cuisine with daily specials!

        </Text>

      )}

      <Pressable

        style={menuStyles.button}

        onPress={() => setShowMenu(prevState => !prevState)}>

        <Text style={menuStyles.buttonText}>

          {showMenu ? 'Home' : 'View Menu'}

        </Text>

      </Pressable>

      {showMenu && (

        <SectionList

          keyExtractor={(item, index) => item + index}

          sections={menuItemsToDisplay}

          renderItem={renderItem}

          renderSectionHeader={renderSectionHeader}

          ListFooterComponent={Footer}

          ItemSeparatorComponent={Separator}></SectionList>

      )}

    </View>

  );

};

const menuStyles = StyleSheet.create({

  container: {

    flex: 0.95,

  },

  innerContainer: {

    paddingHorizontal: 40,

    paddingVertical: 20,

    backgroundColor: '#333333',

  },

  sectionHeader: {

    backgroundColor: '#fbdabb',

    color: '#333333',

    fontSize: 34,

    flexWrap: 'wrap',

    textAlign: 'center',

  },

  itemText: {

    color: '#F4CE14',

    fontSize: 32,

  },

  separator: {

    borderBottomWidth: 1,

    borderColor: '#EDEFEE',

  },

  footerText: {

    color: '#EDEFEE',

    fontSize: 20,

    flexWrap: 'wrap',

    textAlign: 'center',

  },

  button: {

    fontSize: 22,

    padding: 10,

    marginVertical: 8,

    margin: 40,

    backgroundColor: '#EDEFEE',

    borderColor: '#EDEFEE',

    borderWidth: 2,

    borderRadius: 12

  },

  buttonText: {

    color: '#333333',

    textAlign: 'center',

    fontSize: 32,

  },

  infoSection: {

    fontSize: 24,

    padding: 20,

    marginVertical: 8,

    color: '#EDEFEE',

    textAlign: 'center',

    backgroundColor: '#495E57',

  },

});

export default MenuItems;