CLASS 69

QR Code Scanner

1. Before we start, let's try to understand what is a Barcode and QR code and what is the difference between them.
2. You might have seen a Barcode on all products that you purchase - books, packed food items, etc. These codes have information embedded in them. They can contain strings / text which convey information like - name of the product, price, etc. The advantage of QR code and Barcode is that they are machine readable and machines can read information printed on them very quickly. You can learn more about QR code and Barcode through the reference links provided below.

   https://commons.wikimedia.org/wiki/File:Japan-qr-code-billboard.jpg

   https://commons.wikimedia.org/wiki/File:Code93.png

3. QRCode: A QR code (short for "quick response" code) is a type of Barcode that contains a matrix of dots. All QR codes have a square shape. It can be scanned using a QR scanner or a smartphone with a built-in camera. Once scanned, software on the device converts the dots within the code into numbers or a string of characters.
4. Barcode is a printed series of parallel bars or lines of varying width used to enter data into a computer system—mainly used for stock keeping in shops. The bars are typically black on a white background, and their width and quantity vary according to application. The numbers represented by a barcode are also printed out at its base.
5. Let's get started in building our own Barcode/QR code scanner. Import the below link in snack.
   https://github.com/whitehatjr/e-library-PRO-C68

6. Currently our Transaction screen only has text inside the View component. Create a Button (using TouchableOpacity) which will trigger the QR code scanner and display the scanned output as text.
7. Add some styling to our TouchableOpacity and Text using StyleSheet.

8. We will be using an expo package (library) which will help us build the QR code scanner in our application.
9. You might have noticed that our app needs to ask for Camera permissions first before using it to scan QR codes. Let's import the Barcode scanner component and permissions.

```
import React, { Component } from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";
import * as Permissions from "expo-permissions";
import { BarCodeScanner } from "expo-barcode-scanner";

```

10. Let's define three states in our application:
    - domState : “ => This will tell if the app is in scanner mode or scanned mode.
    - hasCameraPermissions: null => This will tell if the user has granted camera permission to the application.
    - scanned: false => This will tell if scanning has been completed or not.
    - scannedData: '' => This will hold the scanned data that we get after scanning.

11. When we press our Scan QR Code Button in our application, we need to request for camera permission. Write a function - getCameraPermission - which can request for camera permission.
12. Note that this function needs to be asynchronous because it takes time for the user to give camera permission to the application. The Permission component which we imported has a predefined function called .askAsync() which can request for various permissions. We will use this to request for camera permission inside our function and change the state of hasCameraPermissions.
13. Note that .askAsync() returns an object with a 'status' key containing the status of the permission granted by the user. If the user grants permission, status changes to 'granted' *Note: {status} automatically extracts the value from the object with key 'status'.
14. Let's call the function getCameraPermissions when the TouchableOpacity Button is pressed using its onPress prop.

```
      scannedData:
    };
  }

  getCameraPermissions = async domState => {
  const { status } = await Permissions.askAsync(Permissions.CAMERA);

  this.setState({
    /*status === "granted" is true when user has granted permission
      status === "granted" is false when user has not granted the permission
    */
    hasCameraPermissions: status === "granted",
    domState: domState,
    scanned: false
  });
 };

 render() {
   return (
     <View style={styles.container}>
       <TouchableOpacity
         style={[styles.button, { marginTop: 25 }]}
         onPress={() => this.getCameraPermissions("scanner")}>
         <Text style={styles.buttonText}>Scan QR Code</Text>
       </TouchableOpacity>
     </View>
   );
 }
}
```

15. Pressing the button makes the application ask for camera permission.
16. The camera permission is asked only once. If you grant the permission, the application will remember it.
17. We need to tell our application what to do if, hasCameraPermissions is 'null' or 'false'. Let's say that we want to display a text "Request for camera permission" if hasCameraPermissions is false or null.
18. If hasCameraPermissions is 'true', we will display whatever text is inside the scannedData state. Currently scannedData is an empty string. Let us write code for this in our render() function.
19. Note that we can write JSX only in the return function. If we are writing JavaScript code, we do it inside the curly {} brackets.

```
      */
      hasCameraPermissions: status === "granted",
      domState: domState,
      scanned: false
    });
  };

  render() {
    const { domState, hasCameraPermissions, scannedData, scanned } = this.state;
    return (
      <View style={styles.container}>
        <Text style={styles.text}>
          {hasCameraPermissions ? scannedData : "Request for Camera Permission"}
        </Text>
        <TouchableOpacity
          style={[styles.button, { marginTop: 25 }]}
          onPress={() => this.getCameraPermissions("scanner")}>
          <Text style={styles.buttonText}>Scan QR Code</Text>
        </TouchableOpacity>
      </View>
    );
  }
}

const styles = StyleSheet.create({
```

20. Now, we want to display our BarCodeScanner when our Scan Button is clicked. We have already created a domState that keeps track if the button has been clicked.
21. ● domState will be 'normal' when the application starts.
    ● When the button is clicked to get camera permissions, domState should change to 'scanner'.

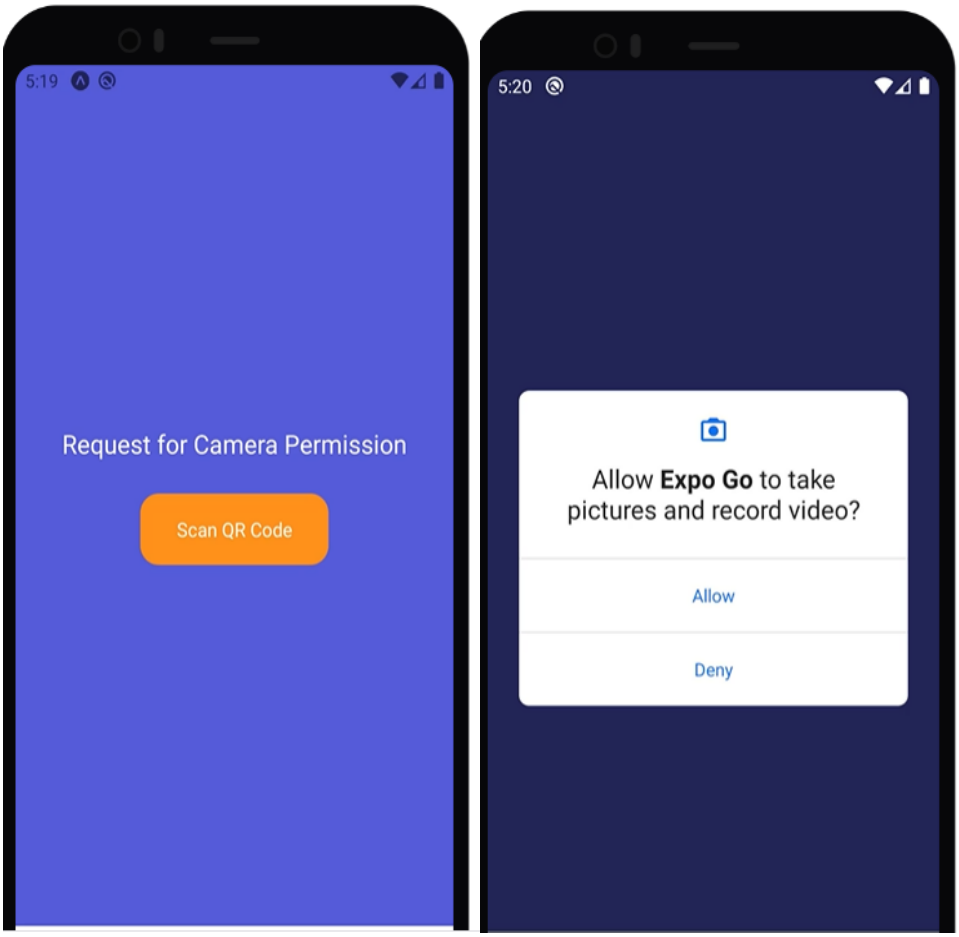
22. Now, we want to return a BarCodeScanner component when the button is clicked and the user has given camera permissions. The BarCodeScanner component automatically starts scanning using the Camera. It has a prop called onBarCodeScanned which can call a function to handle data received after scanning. We want to call this function only when the scanned is false.

23. Write a function called handleBarCodeScanned which is called when the scan is completed.
    ● This function automatically receives the type of barcode scanned and the data inside the barcode. We can set the scannedData here to be equal to the data received after scanning. ● Once the scan has been completed, we also want to set the scanned state to true. We also want to change the state for the domState to make it back to normal when the scan is completed.

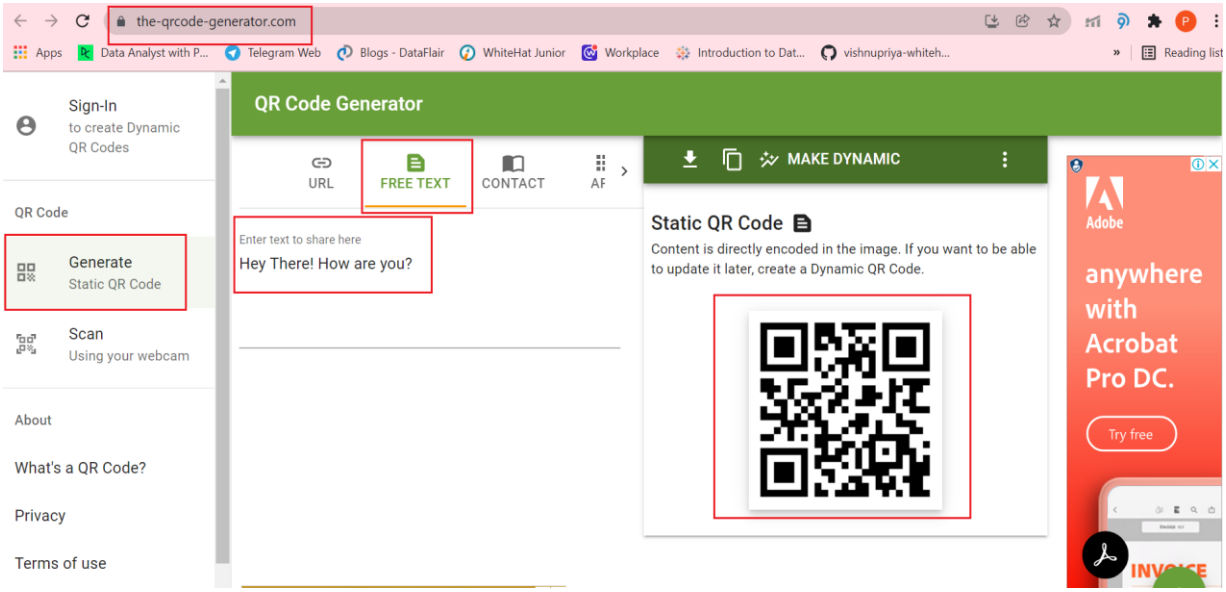
```
tion.js
nents
ch.js
saction.js
onfig.js
e.json
E.md
ck

24          hasCameraPermissions: status === "granted",
25          domState: domState,
26          scanned: false
27        });
28      };
29
30      handleBarCodeScanned = async ({ type, data }) => {
31        this.setState({
32          scannedData: data,
33          domState: "normal",
34          scanned: true
35        });
36      };
37
38      render() {
39        const { domState, hasCameraPermissions, scannedData, scanned } = this.state;
40        if (domState === "scanner") {
41          return (
42            <BarCodeScanner
43              onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
44              style={StyleSheet.absoluteFillObject}
45            />
46          );
47        }
48
49        return (
50          <View style={styles.container}>
51            <Text style={styles.text}>
52              {hasCameraPermissions ? scannedData : "Request for Camera Permission"}
```

24. Execute the application and see how it works.



25. Steps to generate random QR code- Use the website https://www.the-qrcode-generator.com/
26. Click on Generate on left hand side panel
27. You can select any freetext/URL option in the working space and write any message.
28. The QR code will gte generated on the right panel. Scan that code with the app we created and see the results. The app should show you the same message for which you have created the QR code.



29. The reference code for this class are present at- https://github.com/whitehatjr/e-library-v2-PRO-C69