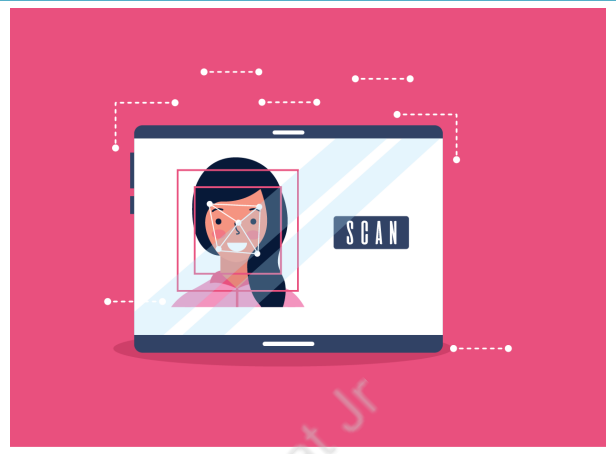


## FACE DETECTION



### What is our GOAL for this MODULE?

We learned about face detection to build a face filters app using react native.

### What did we ACHIEVE in the class TODAY?

- We learned about face detection

### Which CONCEPTS/CODING BLOCKS did we cover today?

- expo-permissions
- expo-camera
- expo-face-detector
- <Camera/>, <SafeAreaView/> components

### How did we DO the activities?

1. Set up the project using expo init

```
npm
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

D:\WhiteHatJr\Classes\Class 180>expo init frapp
✓ Choose a template: » blank          a minimal app as clean as an empty canvas
✓ Downloaded and extracted project files.
Using npm to install packages.
\ Installing JavaScript dependencies.
```

2. Install the following dependencies inside the project folder

- expo install expo-camera
- expo install expo-face-detector
- expo install expo-permissions

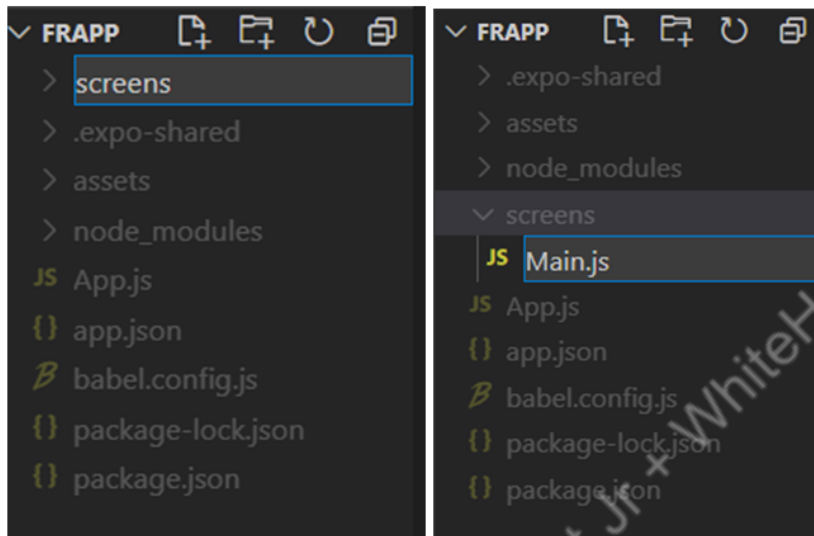
```
C:\Windows\System32\cmd.exe

D:\WhiteHatJr\Classes\Class 180>cd frapp
D:\WhiteHatJr\Classes\Class 180\frapp>expo install expo-camera_
```

```
D:\WhiteHatJr\Classes\Class 180\frapp>expo install expo-face-detector
```

```
D:\WhiteHatJr\Classes\Class 180\frapp>expo install expo-permissions
```

3. Create a new folder called **screens** in our root directory and then create a file **Main.js** inside this folder



4. Write the “Main” class component and import it in the **App.js**.

```
export default class Main extends React.Component {  
  
}
```

```
import React from 'react';  
import Main from "../screens/Main";  
  
export default function App() {  
  return (  
    <Main />  
  )  
}
```

5. **Import** react/expo modules in **Main.js**

```
import React from 'react';
import { StyleSheet, Text, View, SafeAreaView, StatusBar, Platform } from 'react-native';
import { Camera } from 'expo-camera';
import * as Permissions from "expo-permissions";
import * as FaceDetector from 'expo-face-detector';
```

6. Set the **state variable** values

```
constructor(props) {
  super(props)
  this.state = {
    hasCameraPermission: null,
    faces: []
  }
}

componentDidMount() {
  Permissions
    .askAsync(Permissions.CAMERA)
    .then(this.onCameraPermission)
}

onCameraPermission = (status) => {
  this.setState({ hasCameraPermission: status.status === 'granted' })
}

onFacesDetected = (faces) => {
  this.setState({ faces: faces })
}

onFaceDetectionError = (error) => {
  console.log(error)
}
```

7. Write the **Main** class component:
  - Create StyleSheet for the components to render.

```
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  droidSafeArea: {
    marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
  },
  headingContainer: {
    flex: 0.1,
    alignItems: 'center',
    justifyContent: 'center'
  },
  titleText: {
    fontSize: 30
  },
  cameraStyle: {
    flex: 0.65
  },
  filterContainer: {},
  actionContainer: {}
});
```

- Check if the camera has permissions
- If yes, then render the App Name, Camera and Filters component
- If no, then show a text message with “No access to camera”

```
render() {  
  const { hasCameraPermission } = this.state;  
  if (hasCameraPermission === null) {  
    return <View />  
  }  
  if (hasCameraPermission === false) {  
    return (  
      <View style={styles.container}>  
        <Text>No access to camera</Text>  
      </View>  
    )  
  }  
  console.log(this.state.faces)  
  return (  
    <View style={styles.container}>  
      <SafeAreaView style={styles.droidSafeArea} />  
      <View style={styles.headingContainer}>  
        <Text style={styles.titleText}>FRAPP</Text>  
      </View>  
      <View style={styles.cameraStyle}>  
        <Camera  
          style={{ flex: 1 }}  
          type={Camera.Constants.Type.front}  
          faceDetectorSettings={{  
            mode: FaceDetector.Constants.Mode.fast,  
            detectLandmarks: FaceDetector.Constants.Landmarks.all,  
            runClassifications: FaceDetector.Constants.Classifications.all  
          }}  
          onFacesDetected={this.onFacesDetected}  
          onFacesDetectionError={this.onFacesDetectionError}  
        />  
      </View>  
      <View style={styles.filterContainer}>  
      </View>  
      <View style={styles.actionContainer}>  
      </View>  
    </View>  
  )  
}
```

8. Test the output: We will get an array of **faces** data.

```
Object {
  "faces": Array [
    Object {
      "bottomMouthPosition": Object {
        "x": 382.6530616066672,
        "y": 444.0436235138864,
      },
      "bounds": Object {
        "origin": Object {
          "x": 276.54545454545456,
          "y": 329.6498106060606,
        },
        "size": Object {
          "height": 139.3090909090909,
          "width": 115.09090909090908,
        },
      },
      "faceID": -1,
      "leftCheekPosition": Object {
        "x": 408.44373529607594,
        "y": 408.0980912642045,
      },
      "leftEarPosition": Object {
        "x": 414.4398671930486,
        "y": 407.6263656616211,
      },
      "leftEyeOpenProbability": 0.770969033241272,
      "leftEyePosition": Object {
        "x": 385.6348342895507,
        "y": 378.9945583227909,
      },
      "leftMouthPosition": Object {
        "x": 398.6883628151633,
        "y": 427.18644503969136,
      },
      "noseBasePosition": Object {
        "x": 369.7868021184747,
        "y": 400.79015294855293,
      },
      "rightCheekPosition": Object {
        "x": 327.262656471946,
        "y": 426.2369415283203,
      },
      "rightEarPosition": Object {
        "x": 295.1419289328835,
```

We have successfully learned to use FaceDetector API in expo.

## What's NEXT?

In the next class, we will learn to apply face filters based on data collected after face detection.

## EXTEND YOUR KNOWLEDGE:

- You can refer to the link below to explore more about expo FaceDetector  
[FaceDetector](#)

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr