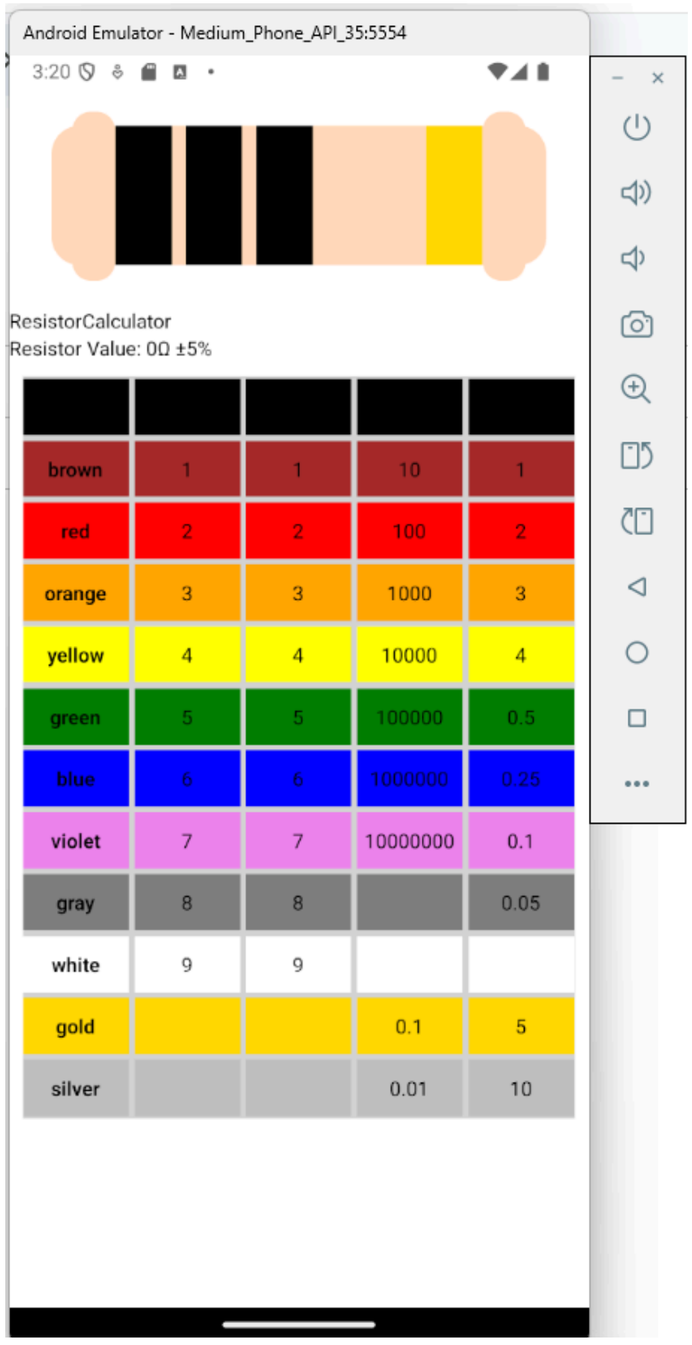
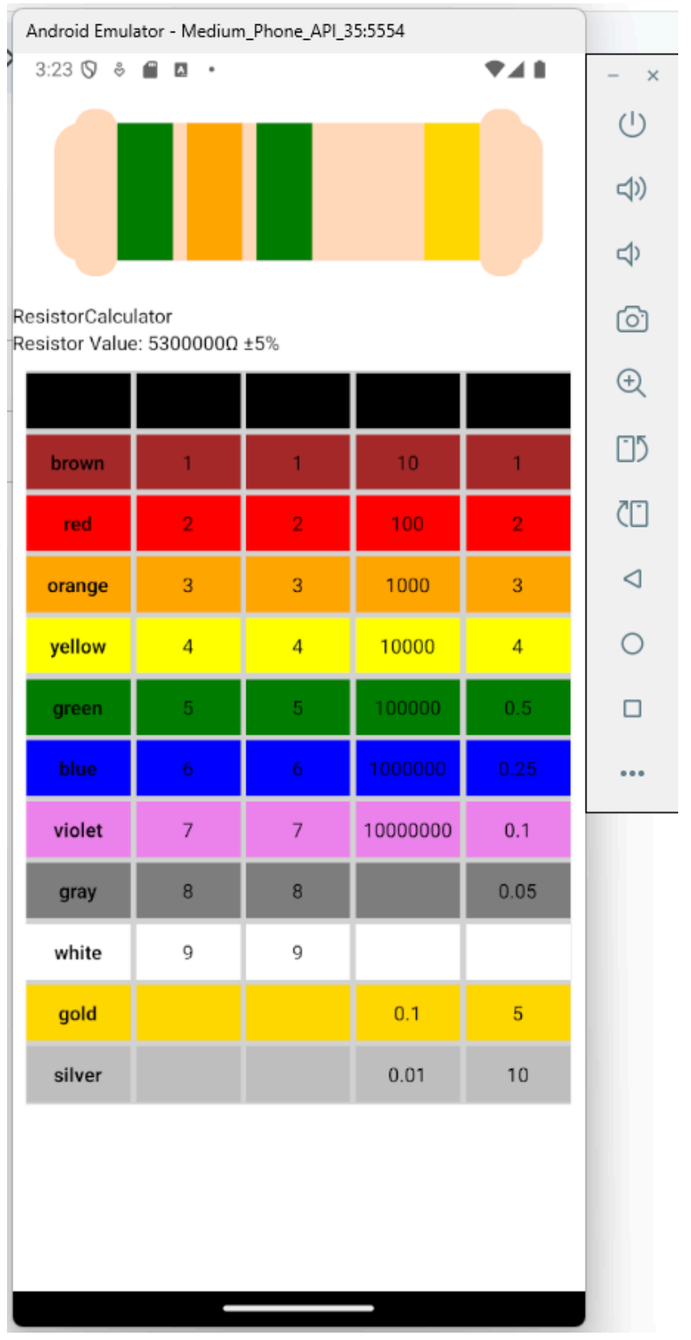
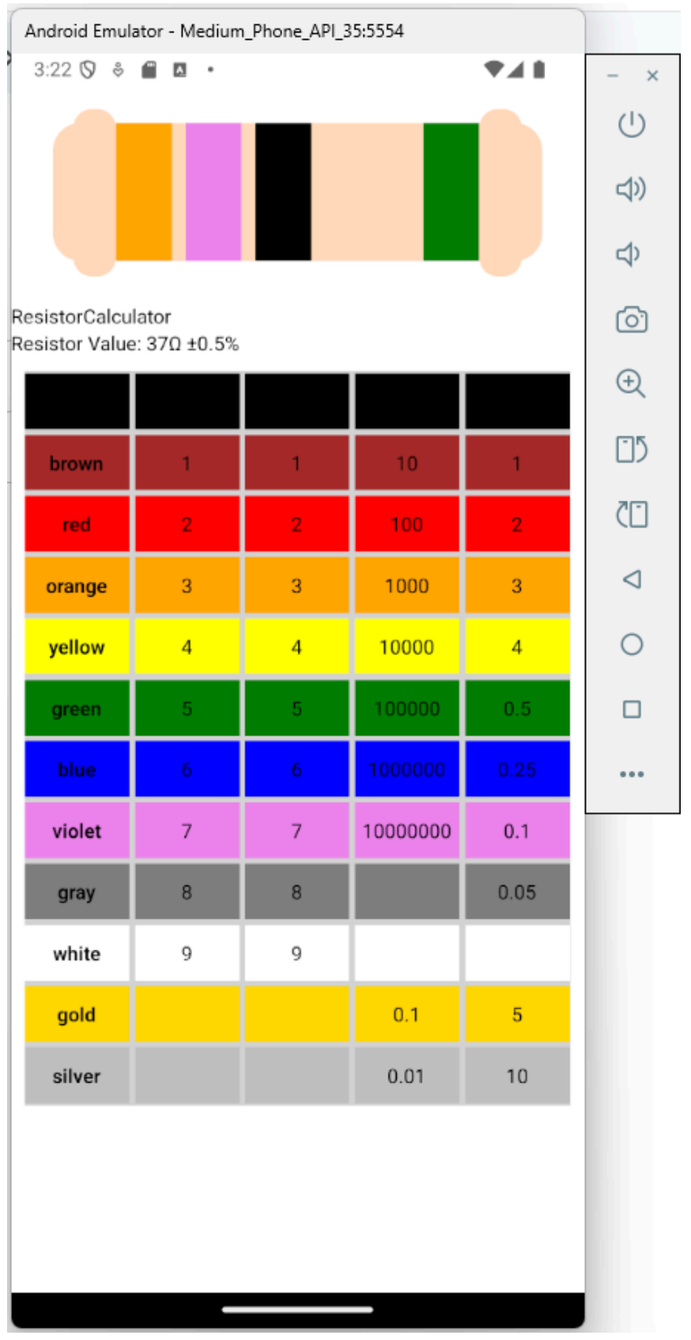



**อาทิตย์ ดิษดำ**

Resistor Calculator

React Native

JS App.js M

JS App.js > ...

You, 11 minutes ago | 2 authors (BANGKHEN\5-903 and one other)

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';
import Colorrr from './Colorrr';
import BandSelector from './BandSelector'

export default function App() {
  return (
    <View style={styles.container}>
      <BandSelector/>
      {/* <Colorrr/> */}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: 20,
    backgroundColor: '#fff',
  },
});

```

Import คอมโพเนนต์ และนำมาแสดง
<BandSelector/>

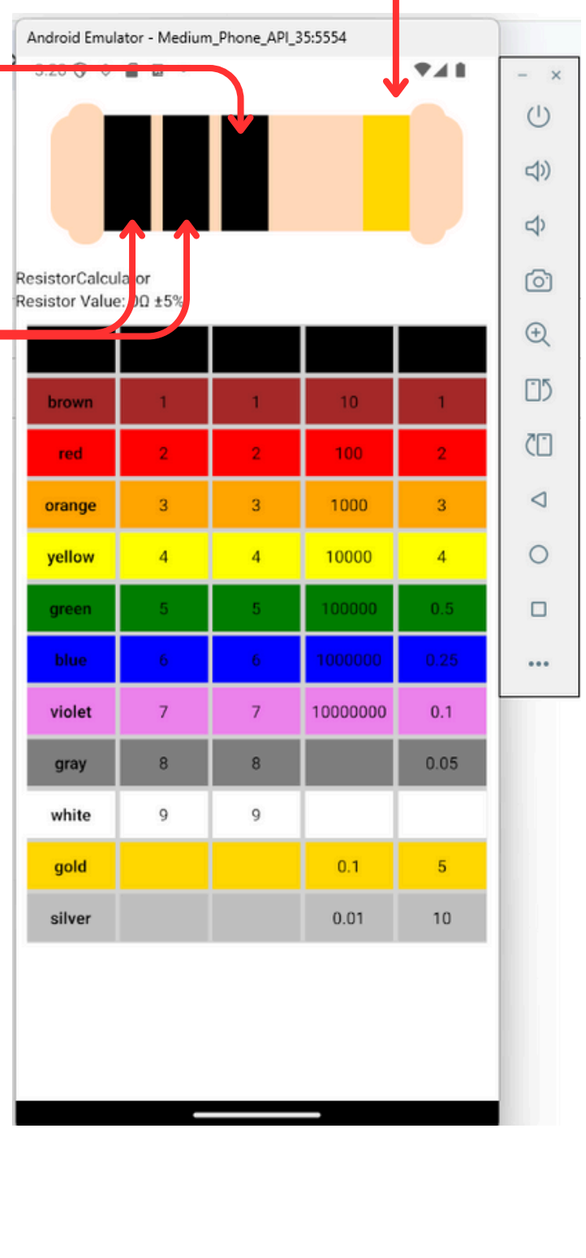
BandSelector.js U

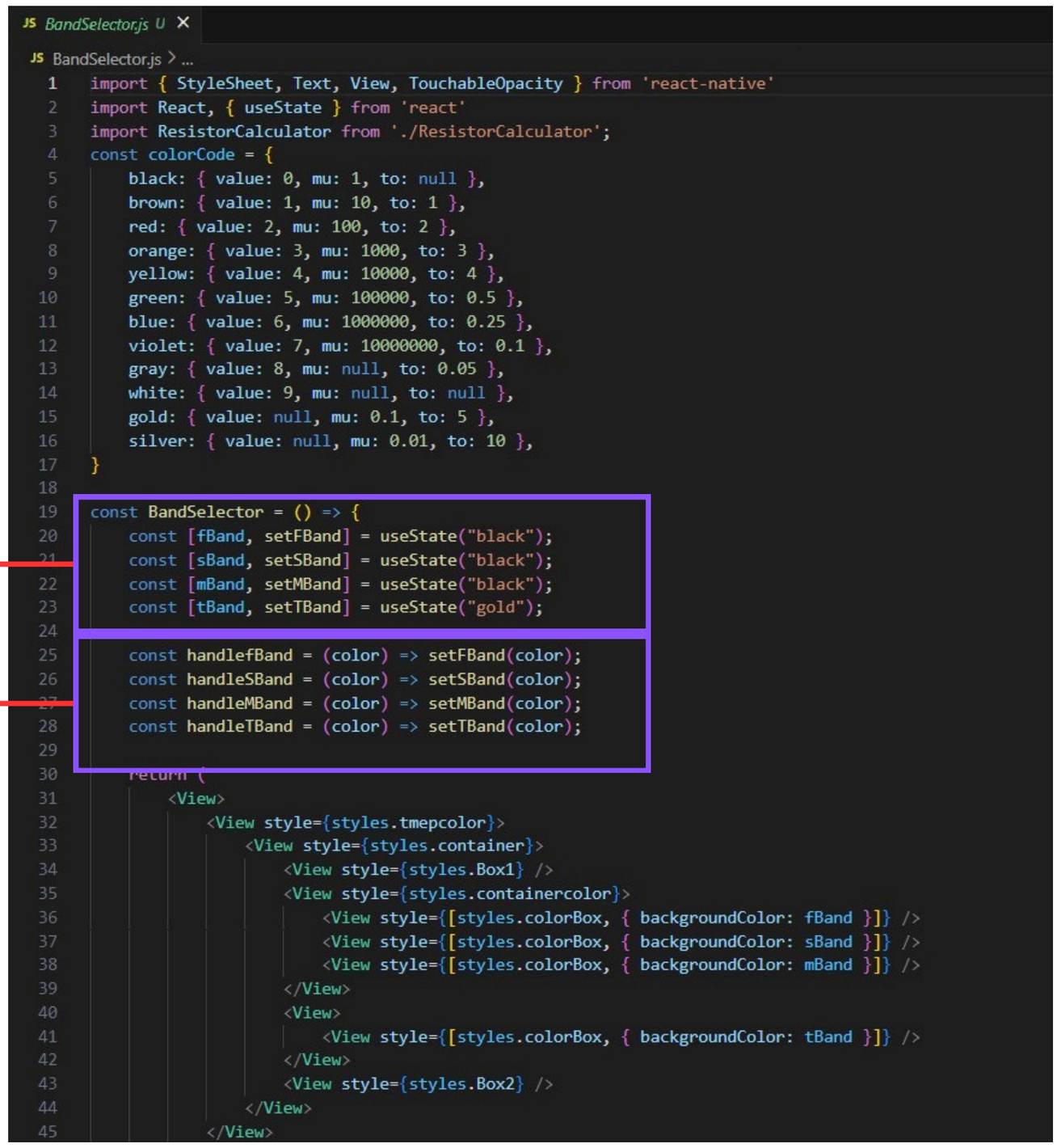
BandSelector.js > ...

```
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native'
import React, { useState } from 'react'
import ResistorCalculator from './ResistorCalculator';
const colorCode = {
    black: { value: 0, mu: 1, to: null },
    brown: { value: 1, mu: 10, to: 1 },
    red: { value: 2, mu: 100, to: 2 },
    orange: { value: 3, mu: 1000, to: 3 },
    yellow: { value: 4, mu: 10000, to: 4 },
    green: { value: 5, mu: 100000, to: 0.5 },
    blue: { value: 6, mu: 1000000, to: 0.25 },
    violet: { value: 7, mu: 10000000, to: 0.1 },
    gray: { value: 8, mu: null, to: 0.05 },
    white: { value: 9, mu: null, to: null },
    gold: { value: null, mu: 0.1, to: 5 },
    silver: { value: null, mu: 0.01, to: 10 },
}

const BandSelector = () => {
    const [fBand, setFBand] = useState("black");
    const [sBand, setSBand] = useState("black");
    const [mBand, setMBand] = useState("black");
    const [tBand, setTBand] = useState("gold");

    const handlefBand = (color) => setFBand(color);
    const handleSBand = (color) => setSBand(color);
    const handleMBand = (color) => setMBand(color);
    const handleTBand = (color) => setTBand(color);

    return (
        <View>
            <View style={styles.tmepcolor}>
                <View style={styles.container}>
                    <View style={styles.Box1} />
                    <View style={styles.containercolor}>
                        <View style={[styles.colorBox, { backgroundColor: fBand }]} />
                        <View style={[styles.colorBox, { backgroundColor: sBand }]} />
                        <View style={[styles.colorBox, { backgroundColor: mBand }]} />
                    </View>
                    <View>
                        <View style={[styles.colorBox, { backgroundColor: tBand }]} />
                    </View>
                    <View style={styles.Box2} />
                </View>
            </View>
```

การทำobject ใน array ใช้เก็บค่าสีแต่ละสีและของตัวต้านทาน เพื่อนำไปทำการคำนวณ และสี

- ค่าสีแต่สี
- (ตัวเลขของตัวต้านทาน) value → ใช้สำหรับแถบที่ 1 และ 2
- (ตัวคูณ) mu → ใช้สำหรับแถบที่ 3
- (ค่าความคลาดเคลื่อน) to → ใช้สำหรับแถบที่ 4






ใช้ useState เพื่อเก็บค่าสีของแต่ละแถบ

- fBand → แถบที่ 1
- sBand → แถบที่ 2
- mBand → แถบที่ 3 (ตัวคูณ)
- tBand → แถบที่ 4 (ความคลาดเคลื่อน)

const handleTBand = (color) => setTBand(color);
ใช้เปลี่ยนค่า state เมื่อกดเลือกสีใหม่จากตารางสี ของทั้ง 4ค่า

```
19  const BandSelector = () => {
20      const [fBand, setFBand] = useState("black");
21      const [sBand, setSBand] = useState("black");
22      const [mBand, setMBand] = useState("black");
23      const [tBand, setTBand] = useState("gold");
24
25      const handlefBand = (color) => setFBand(color);
26      const handleSBand = (color) => setSBand(color);
27      const handleMBand = (color) => setMBand(color);
28      const handleTBand = (color) => setTBand(color);
29
30      return (
31          <View>
32              <View style={styles.tmepcolor}>
33                  <View style={styles.container}>
34                      <View style={styles.Box1} />
35                      <View style={styles.containercolor}>
36                          <View style={[styles.colorBox, { backgroundColor: fBand }]} />
37                          <View style={[styles.colorBox, { backgroundColor: sBand }]} />
38                          <View style={[styles.colorBox, { backgroundColor: mBand }]} />
39                      </View>
40                      <View>
41                          <View style={[styles.colorBox, { backgroundColor: tBand }]} />
42                      </View>
43                      <View style={styles.Box2} />
44                  </View>
45              </View>
```

UI ของตัวต้านทาน (Resistor)

- แสดงแถบตัวต้านทานโดยใช้ View ที่เปลี่ยนสีตามค่าที่ถูกเลือก

แถบสีหลักของตัวต้านทาน

- จะเปลี่ยนสีตามค่าที่เลือกโดยใช้ BackgroundColor ที่อยู่ในอินไลน์ แล้วใช้ค่าใน useState fBand, sBand, mBand, trans ในการเซ็ตค่าสี

```
JS BandSelector.js U
JS BandSelector.js > ...
19  const BandSelector = () => {
45              </View>
46              <Text>ResistorCalculator</Text>
47              <ResistorCalculator fBand={fBand} sBand={sBand} mBand={mBand} tBand={tBand} />
48              <View style={styles.re}>
49                  <View style={styles.colorPalette}>
50                      {Object.keys(colorCode).map((color) => (
51                          <View
52                              key={color}
53                              style={[styles.colorOption, { backgroundColor: color }]}
54                          >
55                              <Text style={styles.txt}>{color}</Text>
56                          </View>
57                      ))}
58                  </View>
59                  <View style={styles.colorPalette}>
60                      {Object.keys(colorCode).map((color) => (
61                          <TouchableOpacity
62                              key={color}
63                              style={[styles.colorOption, { backgroundColor: color }]}
64                              onPress={() => handlefBand(color)}
65                          >
66                              <Text>{colorCode[color].value}</Text>
67                          </TouchableOpacity>
68                      ))}
69                  </View>
70                  <View style={styles.colorPalette}>
71                      {Object.keys(colorCode).map((color) => (
72                          <TouchableOpacity
73                              key={color}
74                              style={[styles.colorOption, { backgroundColor: color }]}
75                              onPress={() => handleSBand(color)}
76                          >
77                              <Text>{colorCode[color].value}</Text>
78                          </TouchableOpacity>
79                      ))}
80                  </View>
81                  <View style={styles.colorPalette}>
82                      {Object.keys(colorCode).map((color) => (
83                          <TouchableOpacity
84                              key={color}
85                              style={[styles.colorOption, { backgroundColor: color }]}
86                              onPress={() => handleMBand(color)}
87                          >
88                              <Text>{colorCode[color].mu}</Text>
```

Object.keys(colorCode).map((color) => ( ... ))

- Object.keys(colorCode) → ดึงเอาชื่อสีทั้งหมดจาก colorCode (เช่น "black", "brown", "red")
- .map((color) => (...)) → วนลูปแต่ละสีและสร้างกล่องสีโดยใช้ View และทีกำหนด style สำหรับแต่ละกล่องสี

<View key={color} style={[styles.colorOption, { backgroundColor: color }]} >

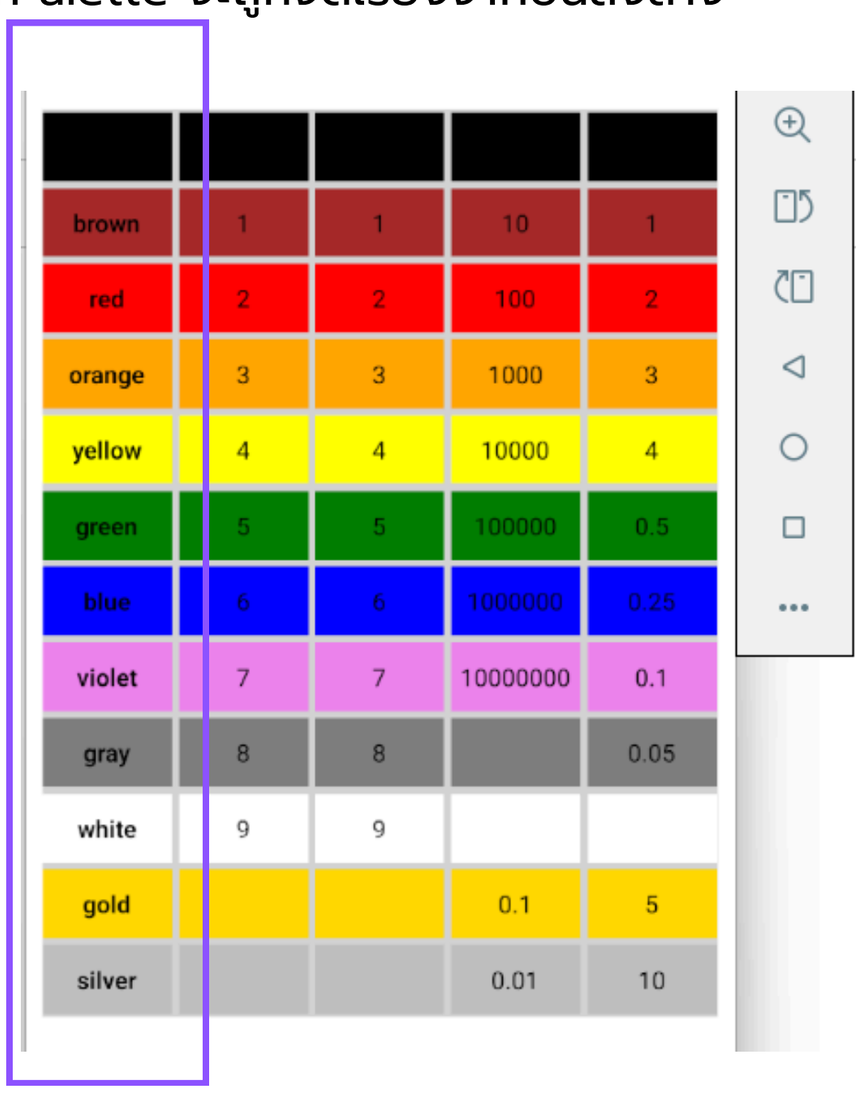
- key={color} → ใช้ color เป็นค่า key เพื่อกำหนดเมื่อใช้ .map()
- style={[styles.colorOption, { backgroundColor: color }]}
  - styles.colorOption → ใช้สไตล์พื้นฐานที่กำหนดไว้ โค้ดstyle อยู่อีกหน้า
  - { backgroundColor: color } → กำหนดสีพื้นหลังให้ตรงกับ color (ทำให้แต่ละ View มีสีที่แตกต่างกัน)

ใช้ map() เพื่อวนลูปแสดงสีทั้งหมดจาก colorCode แต่ละสีจะแสดงเป็น View ที่มี backgroundColor ตรงกับสีนั้น มี Text อยู่ข้างในเพื่อแสดงชื่อสี มีทั้งหมด 5 กล่อง color palette

| | | | | |
|---|---|---|---|---|
| brown | 1 | 1 | 10 | 1 |
| red | 2 | 2 | 100 | 2 |
| orange | 3 | 3 | 1000 | 3 |
| yellow | 4 | 4 | 10000 | 4 |
| green | 5 | 5 | 100000 | 0.5 |
| blue | 6 | 6 | 1000000 | 0.25 |
| violet | 7 | 7 | 10000000 | 0.1 |
| gray | 8 | 8 | | 0.05 |
| white | 9 | 9 | | |
| gold | | | 0.1 | 5 |
| silver | | | 0.01 | 10 |

```
152     re: {
153         flexDirection: 'row',
154         justifyContent: 'center',
155         alignItems: 'center',
156         backgroundColor: '#d3d3d3',
157         margin: 10,
158     },
159     colorPalette: {
160         flexDirection: 'column',
161     },
162     colorOption: {
163         alignItems: 'center',
164         justifyContent: 'center',
165         width: 75,
166         height: 40,
167         margin: 2,
168     },
```

- flexDirection: 'column' → กำหนดให้ เรียงตัวเลือกสีในแนวตั้ง
- แต่ละ View ที่อยู่ใน colorPalette จะถูกจัดเรียงจากบนลงล่าง



| | | | | |
|---|---|---|---|---|
| brown | 1 | 1 | 10 | 1 |
| red | 2 | 2 | 100 | 2 |
| orange | 3 | 3 | 1000 | 3 |
| yellow | 4 | 4 | 10000 | 4 |
| green | 5 | 5 | 100000 | 0.5 |
| blue | 6 | 6 | 1000000 | 0.25 |
| violet | 7 | 7 | 10000000 | 0.1 |
| gray | 8 | 8 | | 0.05 |
| white | 9 | 9 | | |
| gold | | | 0.1 | 5 |
| silver | | | 0.01 | 10 |

- alignItems: 'center' → จัดวาง Text  ให้อยู่ ตรงกลางแนวนอน
- justifyContent: 'center' → จัดวาง Text ให้อยู่ ตรงกลางแนวตั้ง
- width: 75 → กำหนดความกว้างของกล่องสี (75 หน่วย)
- height: 40 → กำหนดความสูงของกล่องสี (40 หน่วย)
- margin: 2 → เพิ่มระยะห่างระหว่างกล่องสีแต่ละอัน

JS ResistorCalculator.js U

JS ResistorCalculator.js > ResistorCalculator

```
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native'
import React from 'react';

const ResistorCalculator = ({ fBand, sBand, mBand, tBand }) => {
    const colorCode = {
        black: { value: 0, mu: 1, to: null },
        brown: { value: 1, mu: 10, to: 1 },
        red: { value: 2, mu: 100, to: 2 },
        orange: { value: 3, mu: 1000, to: 3 },
        yellow: { value: 4, mu: 10000, to: 4 },
        green: { value: 5, mu: 100000, to: 0.5 },
        blue: { value: 6, mu: 1000000, to: 0.25 },
        violet: { value: 7, mu: 10000000, to: 0.1 },
        gray: { value: 8, mu: null, to: 0.05 },
        white: { value: 9, mu: null, to: null },
        gold: { value: null, mu: 0.1, to: 5 },
        silver: { value: null, mu: 0.01, to: 10 },
    };

    const firstBandValue = colorCode[fBand]?.value;
    const secondBandValue = colorCode[sBand]?.value;
    const multiplier = colorCode[mBand]?.mu;
    const tolerance = colorCode[tBand]?.to;

    if (firstBandValue === undefined || secondBandValue === undefined || multiplier === null) {
        return <Text>Invalid Color Code</Text>;
    }

    const resistorValue = (firstBandValue * 10 + secondBandValue) * multiplier;
    const toleranceText = tolerance ? `±${tolerance}%` : '';

    return (
        <Text>
            {`Resistor Value: ${resistorValue}Ω ${toleranceText}`}
        </Text>
    );
};

export default ResistorCalculator;
```

คอมโพเนนต์ ResistorCalculator ใช้สำหรับ คำนวณค่าความต้านทานของตัวต้านทาน (Resistor) จากแถบสีที่เลือก

เป็น Component ที่รับค่าพารามิเตอร์ จาก fBand, sBand, mBand, tBand

- ค่าที่รับมาจะถูกส่งมาจาก BandSelector เพื่อใช้ในการคำนวณ

```
            </View>
            <Text>ResistorCalculator</Text>
            <ResistorCalculator fBand={fBand} sBand={sBand} mBand={mBand} tBand={tBand} />
            <View style={styles.re}>
                <View style={styles.colorPalette}>
                    {Object.keys(colorCode).map((color) => (
                        <View
                            key={color}
                            style={[styles.colorOption, { backgroundColor: color }]}
                        >
                            <Text style={styles.txt}>{color}</Text>
                        </View>
                    ))}
```

JS ResistorCalculator.js U

JS ResistorCalculator.js > ResistorCalculator

```
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native'
import React from 'react';

const ResistorCalculator = ({ fBand, sBand, mBand, tBand }) => {
    const colorCode = {
        black: { value: 0, mu: 1, to: null },
        brown: { value: 1, mu: 10, to: 1 },
        red: { value: 2, mu: 100, to: 2 },
        orange: { value: 3, mu: 1000, to: 3 },
        yellow: { value: 4, mu: 10000, to: 4 },
        green: { value: 5, mu: 100000, to: 0.5 },
        blue: { value: 6, mu: 1000000, to: 0.25 },
        violet: { value: 7, mu: 10000000, to: 0.1 },
        gray: { value: 8, mu: null, to: 0.05 },
        white: { value: 9, mu: null, to: null },
        gold: { value: null, mu: 0.1, to: 5 },
        silver: { value: null, mu: 0.01, to: 10 },
    };

    const firstBandValue = colorCode[fBand]?.value;
    const secondBandValue = colorCode[sBand]?.value;
    const multiplier = colorCode[mBand]?.mu;
    const tolerance = colorCode[tBand]?.to;

    if (firstBandValue === undefined || secondBandValue === undefined || multiplier === null) {
        return <Text>Invalid Color Code</Text>;
    }

    const resistorValue = (firstBandValue * 10 + secondBandValue) * multiplier;
    const toleranceText = tolerance ? `±${tolerance}%` : '';

    return (
        <Text>
            {`Resistor Value: ${resistorValue}Ω ${toleranceText}`}
        </Text>
    );
};

export default ResistorCalculator;
```

- ใช้ค่าจาก fBand, sBand, mBand, tBand มาเช็คเก็บ colorCodeเพื่อนำข้อมูลในobject มาใช้
- ?. (Optional Chaining) กรณีค่าที่รับมาไม่ถูกต้อง

และใช้ IF เช็คค่าจากFIRSTBANDVALUE, SECONDBANDVALUE หรือ MULTIPLIER ไม่มีค่า
แสดงข้อความ "Invalid Color Code"

JS ResistorCalculator.js U

JS ResistorCalculator.js > ResistorCalculator

```
1   import { StyleSheet, Text, View, TouchableOpacity } from 'react-native'
2   import React from 'react';
3
4   const ResistorCalculator = ({ fBand, sBand, mBand, tBand }) => {
5       const colorCode = {
6           black: { value: 0, mu: 1, to: null },
7           brown: { value: 1, mu: 10, to: 1 },
8           red: { value: 2, mu: 100, to: 2 },
9           orange: { value: 3, mu: 1000, to: 3 },
10          yellow: { value: 4, mu: 10000, to: 4 },
11          green: { value: 5, mu: 100000, to: 0.5 },
12          blue: { value: 6, mu: 1000000, to: 0.25 },
13          violet: { value: 7, mu: 10000000, to: 0.1 },
14          gray: { value: 8, mu: null, to: 0.05 },
15          white: { value: 9, mu: null, to: null },
16          gold: { value: null, mu: 0.1, to: 5 },
17          silver: { value: null, mu: 0.01, to: 10 },
18      };
19
20      const firstBandValue = colorCode[fBand]?.value;
21      const secondBandValue = colorCode[sBand]?.value;
22      const multiplier = colorCode[mBand]?.mu;
23      const tolerance = colorCode[tBand]?.to;
24
25      if (firstBandValue === undefined || secondBandValue === undefined || multiplier === null) {
26          return <Text>Invalid Color Code</Text>;
27      }
28
29      const resistorValue = (firstBandValue * 10 + secondBandValue) * multiplier;
30      const toleranceText = tolerance ? `±${tolerance}%` : '';
31
32      return (
33          <Text>
34              {`Resistor Value: ${resistorValue}Ω ${toleranceText}`}
35          </Text>
36      );
37  };
38
39  export default ResistorCalculator;
40
```

นำค่าจากแถบสีที่ 1 (fBand) และแถบสีที่ 2 (sBand) มาคำณวน FIRSTBANDVALUE * 10 + SECONDBANDVALUE

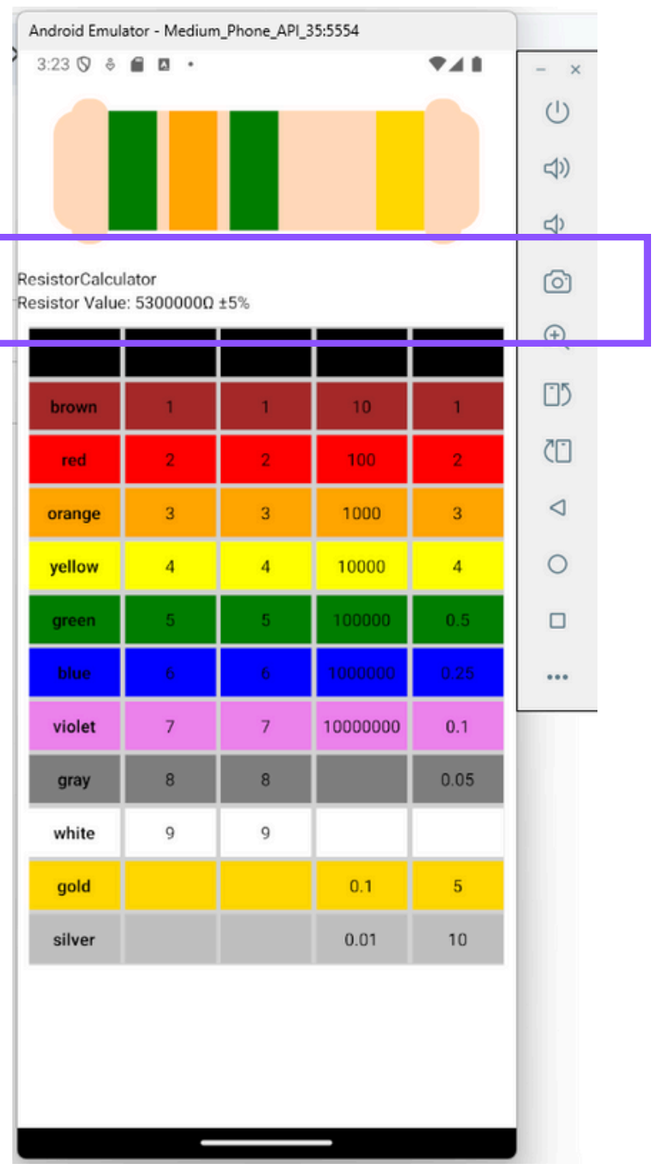
- เช่น
- ถ้าเลือก "RED" (2) และ "BLUE" (6) → 2 * 10 + 6 = 26

และคูณด้วยตัวคูณ (MBAND)

- ถ้า MBAND = 1000 → 26 * 1000 = 26,000 Ω

และนำค่าความคลาดเคลื่อนจาก tolerance(to) → แสดงผลเป็น "±x%" ถ้าไม่มี → แสดงเป็น ""

- แสดงค่าความต้านทานที่คำนวณได้ พร้อมหน่วย Ω (โอห์ม) และแสดงค่าความคลาดเคลื่อน




JS BandSelector.js U

JS BandSelector.js > ...

```
19  const BandSelector = () => {
45              </View>
46              <Text>ResistorCalculator</Text>
47              <ResistorCalculator fBand={fBand} sBand={sBand} mBand={mBand} tBand={tBand} />
48              <View style={styles.re}>
49                  <View style={styles.colorPalette}>
50                      {Object.keys(colorCode).map((color) => (
51                          <View
52                              key={color}
53                              style={[styles.colorOption, { backgroundColor: color }]}
```