

Lập trình Mobile

Ths. Đỗ Phúc Thịnh

Tóm tắt bài trước





Tóm tắt bài trước

3

1. Giới thiệu môn học
2. Cài đặt React Native (Expo CLI)
3. Thực hành thiết kế một số giao diện cơ bản

→ Bài này sẽ học

1. ECMA Script
2. StyleSheet

ECMAScript



- Với sự bùng phát mạnh mẽ của làng JAV (**Javascript**), một nhân giả đã đứng lên thống nhất làng, đó là I Ét Đệ Lục (**ECMAScript 6 – ES6**)
- **Với cách** thi triển nhãn thuật (**code**) **ngắn gọn** của mình, I Ét Đệ Lục (**ES6**) đã dần trở thành khuôn mẫu cho dân làng JAV (**Javascript**)

- Để khai báo biến trong ES6, sử dụng từ khóa:
 - **var**
 - **let**
 - **const**
- Để khai báo hàm trong ES6, sử dụng từ khóa:
 - **function**
- Để in kết quả ra màn hình console, sử dụng hàm **console.log()**

- Ví dụ:

```
var thu = 'Thứ hai'
let ngay = 2
let thang = 2
const nam = 2222

function thungaythangnam(){
  console.log(thu + ' ngày ' + ngay
    + ' tháng ' + thang + ' năm ' + nam)
}

thungaythangnam()
```

- Output

Thứ hai ngày 2 tháng 2 năm 2222

Kiểu dữ liệu mảng – Array

- Kiểu dữ liệu lưu trữ một mảng dữ liệu
- Cú pháp:

```
const mang = [ 'Phần tử 1', 'Phần tử 2', 'Phần tử 3' ]
```

- Truy xuất từng phần tử của mảng: `Tênmảng[Vị trí]`

Ví dụ:

```
mang[0]
```

→ Output:

Phần tử 1

- Truy xuất theo phần tử

```
const [i1, i2, i3] = mang
```


Kiểu dữ liệu mảng – Array

- Một số hàm quan trọng trong Array:
 - filter(): Lọc phần tử thỏa điều kiện
 - find(): Tìm phần tử thỏa điều kiện
 - concat(): Nối mảng
 - splice(): Cắt phần tử
 - some(): Tìm kiếm có phần tử hay không?
 - push(): Thêm phần tử vào cuối mảng
 - pop(): Lấy ra phần tử cuối mảng
 - map(): Ánh xạ mảng thành mảng khác...

- Cấu trúc rẽ nhánh bao gồm: **Cấu trúc điều khiển và vòng lặp**
- Cấu trúc điều khiển: **if, if...else,...**
- Vòng lặp: **for, while, switch,...**
- Cú pháp của các lệnh này tương tự như trong C++, C#

Kiểu dữ liệu Object

- Lưu trữ thông tin theo dạng: **key: value**
- Ví dụ:

```
const SinhVien = {  
  hoten: 'Nguyễn Văn A',  
  mssv: '123456789',  
  ngành: 'Công nghệ thông tin',  
  dihoc: ()=> {  
    console.log('Đi học')  
  },  
}  
console.log(SinhVien.hoten)  
SinhVien.dihoc()
```

- Các object có thể lồng nhau
- Ví dụ:

```
const SinhVien = {  
  hoten: 'Nguyễn Văn A',  
  diem: {  
    toan: 10,  
    van: 5,  
    hoa: 8,  
  }  
}  
console.log(SinhVien.diem.toan)
```

- **Template Literals** hay còn gọi là **Template Strings** là một **cú pháp mới** để khai báo chuỗi trong Javascript được giới thiệu trong **ES6**
- Cú pháp:
 - Đặt trong dấu nháy ngược ``
 - Sử dụng dấu đô-la và cặp ngoặc nhọn **`${}`**
- Ví dụ:

```
let a = 5
let b = 6
console.log(`Kết quả là ${a + b}`)
```

- **Arrow Function** (hàm mũi tên) là một trong những tính năng mới của ES6 **giúp viết hàm ngắn gọn** hơn.
- Cú pháp:
const “Tên hàm” = (“Tên tham số nếu có”) => {“Nội dung hàm”}
- Ví dụ: Không có tham số

Arrow function

```
const chao = () => {  
  console.log('Xin chào')  
}
```

Function

```
function chao() {  
  console.log('Xin chào')  
}
```

Arrow function

- Ví dụ: Có 2 tham số

```
const cong = (a, b) => {  
  |   return a + b  
  |  
  }  
console.log(cong(5, 4))
```

- Ví dụ: Có 1 tham số

```
let binhphuong = (a) => {  
  |   return a * a  
  |  
  }  
console.log(binhphuong(6))
```

```
const binhphuong = a => a * a
```

StyleSheet



There is no CSS!

Inline Styles

StyleSheet Objects

Written in JavaScript

Based on CSS Syntax, but only subset of properties & features is supported!

Preferred!

- Inline Styles:

```
export default function App() {  
  return (  
    <View style={{backgroundColor: 'red', marginTop: 50}}>  
      <Text style={{fontSize: 50, textAlign: 'center'}}>Xin chào các bạn</Text>  
    </View>  
  );  
}
```

- StyleSheet Objects:

```
export default function App() {  
  return (  
    <View style={styles.view}>  
      <Text style={styles.text}>Xin chào các bạn</Text>  
    </View>  
  );  
}  
  
const styles = StyleSheet.create({  
  view: {  
    backgroundColor: 'red',  
    marginTop: 50  
  },  
  text: {  
    fontSize: 50,  
    textAlign: 'center'  
  },  
});
```

Flexbox



- Flex là một thuộc tính trong React Native.
- Khi một component sử dụng flex trong style (điều kiện là kích thước lớn hơn 0) thì component đó sẽ có kích thước là giãn đầy view đang chứa nó.

- Ví dụ 1:

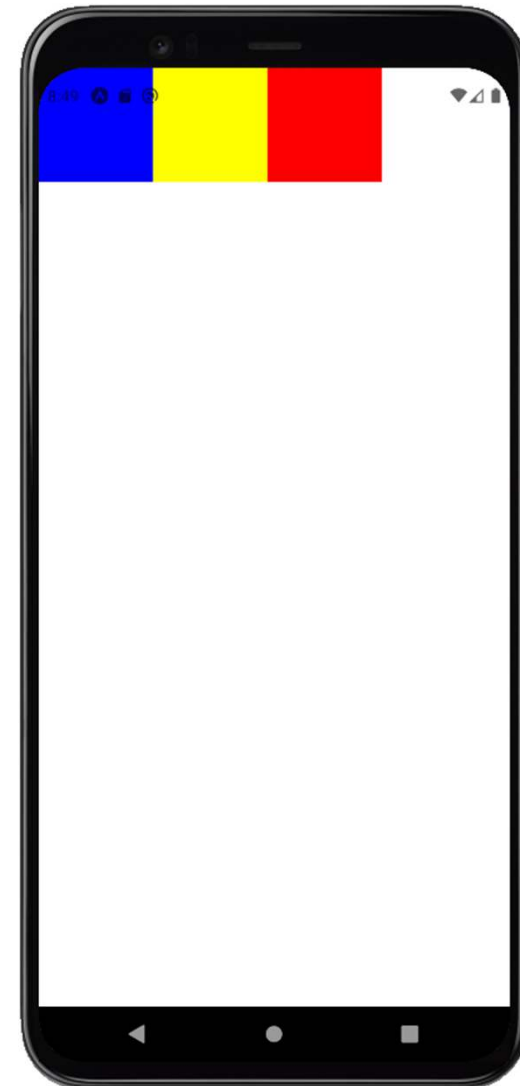
```
export default function App() {  
  return (  
    <View style={{flex: 1}}>  
      <View style={{flex: 1, backgroundColor: 'red'}}></View>  
    </View>  
  );  
}
```

- Ví dụ 2:

```
export default function App() {  
  return (  
    <View style={{flex: 1}}>  
      <View style={{flex: 1, backgroundColor: 'red'}}></View>  
      <View style={{flex: 2, backgroundColor: 'blue'}}></View>  
    </View>  
  );  
}
```

- Những thuộc tính cơ bản thường dùng với flex:
 - **flexDirection**
 - **alignItems**
 - **justifyContent**
 - ...

- Ví dụ:



- Thêm thuộc tính **flexDirection** vào style của một thành phần để **thay đổi hướng sắp xếp** (theo dòng hoặc cột) **các thành phần con** của nó.
- Mặc định thì các thành phần con sẽ được sắp xếp theo cột

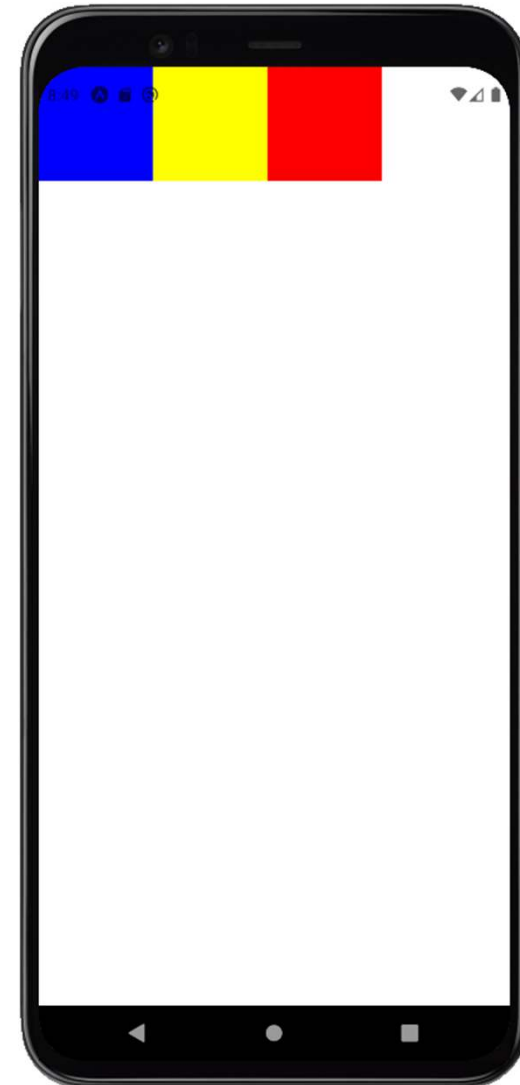
- Ví dụ:

```
export default function App() {  
  return (  
    <View style={styles.view}>  
      <View style={styles.viewxanh}></View>  
      <View style={styles.viewvang}></View>  
      <View style={styles.viewdo}></View>  
    </View>  
  );  
}
```

```
const styles = StyleSheet.create({  
  view: {  
    flex: 1,  
    flexDirection: 'row'  
  },  
  viewdo: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'red',  
  },  
  viewxanh: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'blue',  
  },  
  viewvang: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'yellow'  
  },  
});
```

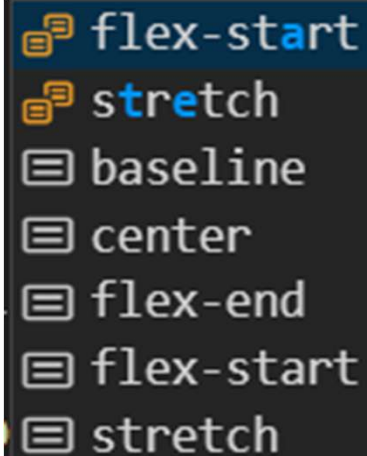
- Ví dụ:

```
export default function App() {  
  return (  
    <View style={styles.view}>  
      <View style={styles.viewxanh}></View>  
      <View style={styles.viewvang}></View>  
      <View style={styles.viewdo}></View>  
    </View>  
  );  
}
```



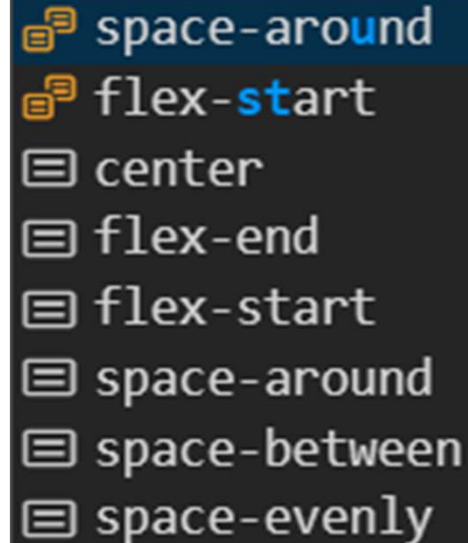
- **alignItems** và **justifyContent** sẽ canh chỉnh các thành phần con bên trong.
- Một số thuộc tính của **alignItems** và **justifyContent**

alignItems



```
flex-start  
stretch  
baseline  
center  
flex-end  
flex-start  
stretch
```

justifyContent



```
space-around  
flex-start  
center  
flex-end  
flex-start  
space-around  
space-between  
space-evenly
```

alignItems và justifyContent

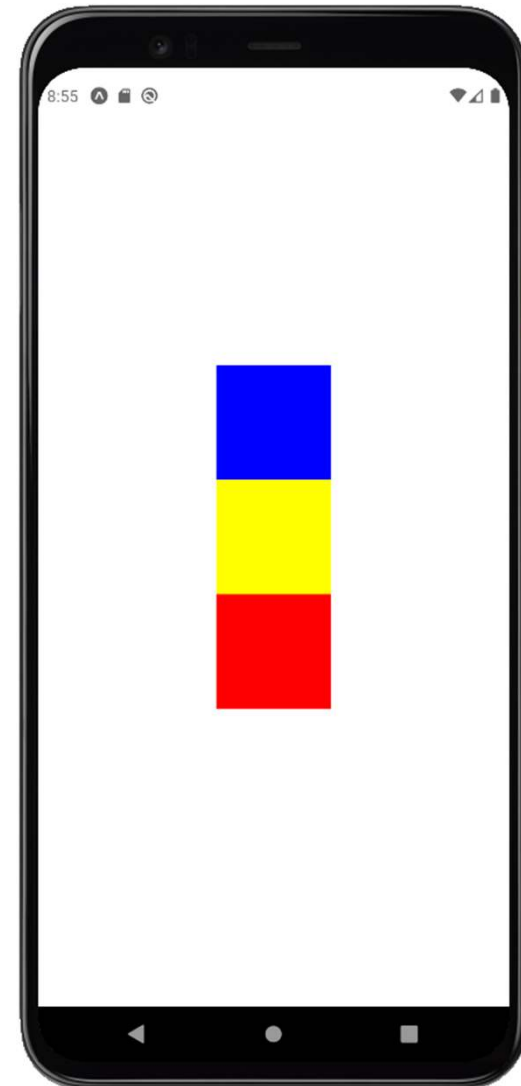
- Ví dụ:

```
export default function App() {  
  return (  
    <View style={styles.view}>  
      <View style={styles.viewxanh}></View>  
      <View style={styles.viewvang}></View>  
      <View style={styles.viewdo}></View>  
    </View>  
  );  
}
```

```
const styles = StyleSheet.create({  
  view: {  
    flex: 1,  
    flexDirection: 'column',  
    alignItems: 'center',  
    justifyContent: 'center'  
  },  
  viewdo: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'red',  
  },  
  viewxanh: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'blue',  
  },  
  viewvang: {  
    height: 100,  
    width: 100,  
    backgroundColor: 'yellow'  
  },  
});
```

- Ví dụ:

```
export default function App() {  
  return (  
    <View style={styles.view}>  
      <View style={styles.viewxanh}></View>  
      <View style={styles.viewvang}></View>  
      <View style={styles.viewdo}></View>  
    </View>  
  );  
}
```



- Đọc thêm ví dụ ở:

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Tách các thành phần



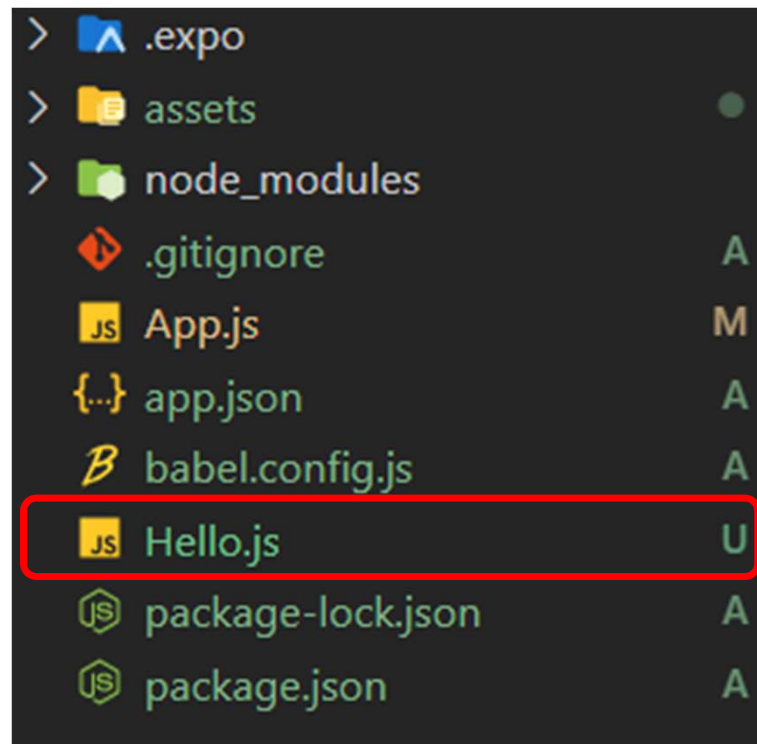
- Trong thực tế, người ta sẽ **tách ứng dụng** thành nhiều **file .js**
- Để có thể sử dụng **các file .js** này, ta sẽ **import** vào
- Cú pháp import:

import TenClass **from** './TenFile';

- Cú pháp sử dụng:

<TenClass />

1. Tạo một file **.js** mới để xây dựng giao diện



2. Viết nội dung file Hello.js như sau (Chú ý export)

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

export default class Hello extends Component {
  render() {
    const name = 'World'
    return (
      <View>
        <Text> Hello {name} </Text>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  });
```

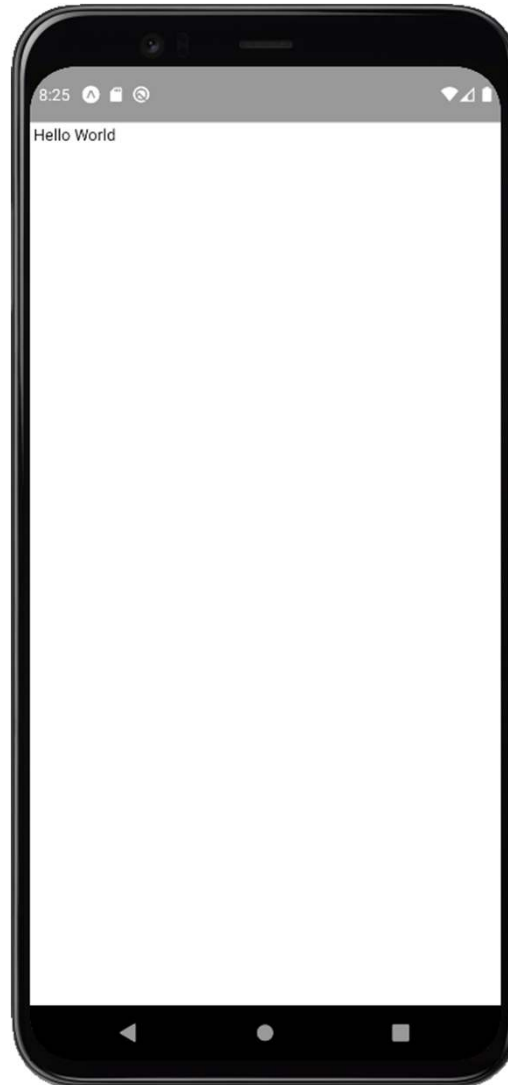
3. Nhúng file .js vừa tạo vào file .js khác (Ví dụ App.js)

```
import Hello from './Hello';  
  
export default function App() {  
  return (  
    <View style={{marginTop: 50}}>  
      <Hello/>  
    </View>  
  );  
}
```

Tên class

Tên file .js

- Kết quả



Bài tập thực hành



1. Viết hàm mũ tên nhận vào hai số a , b và trả về $a^2 + b^2$

Yêu cầu: Dùng **Template Strings** để hiển thị chuỗi

“Kết quả của $a^2 + b^2$ là **xxx**” (**xxx** là kết quả)

2. Viết hàm mũ tên in các số lẻ trong khoảng 50 đến 100 ra màn hình console. Đếm số số lẻ tìm được.

Yêu cầu: Dùng **Template Strings** để hiển thị chuỗi

“Số số lẻ trong khoảng 50 đến 100 là **xxx**” (**xxx** là kết quả)

3. Cho mảng:

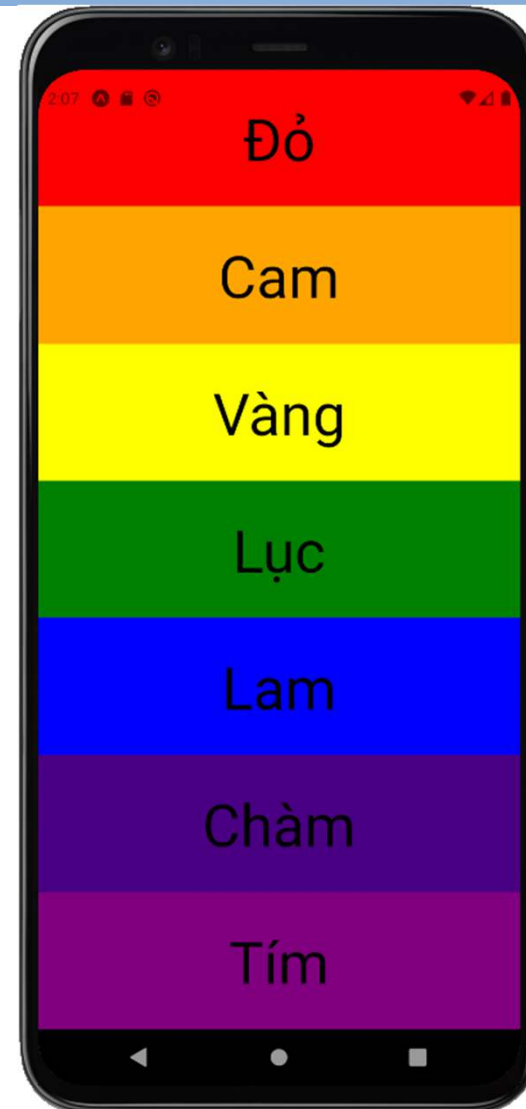
```
const data = [  
  { id: 1, ten: 'Sách Toán', loai: 'Sách' },  
  { id: 2, ten: 'Sách Văn', loai: 'Sách' },  
  { id: 3, ten: 'Conan', loai: 'Truyện' },  
  { id: 4, ten: 'Sherlock Holmes', loai: 'Tiểu thuyết' },  
  { id: 5, ten: 'Doraemon', loai: 'Truyện' }  
]
```

3.1. Dùng hàm filter() lọc ra các loại Sách

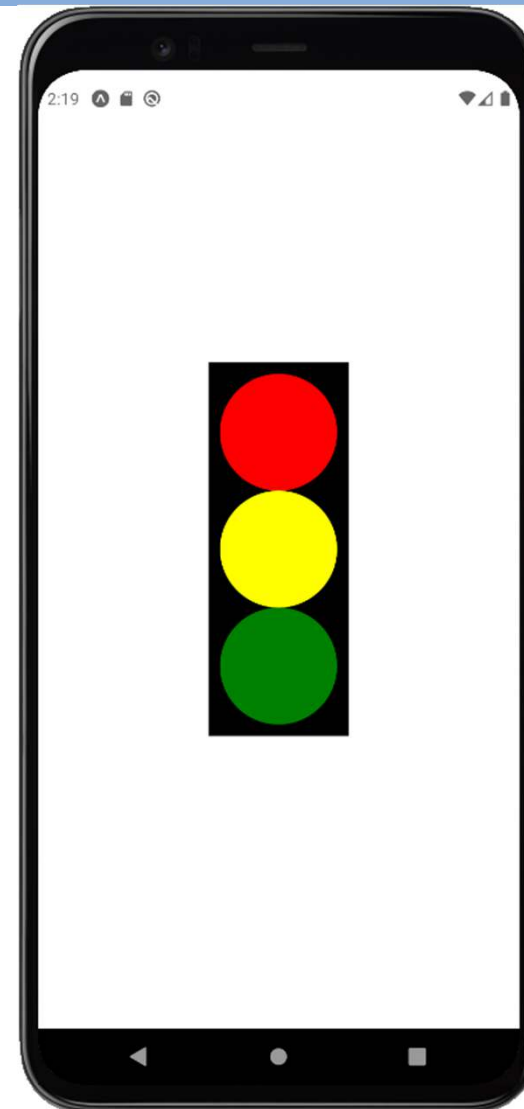
3.2. Dùng hàm find() tìm Sách có id 3

3.3. Dùng hàm some() để kiểm tra xem có **loại Tiểu thuyết** trong mảng hay không?

4. Tạo giao diện như sau
(Yêu cầu: Sử dụng flex)



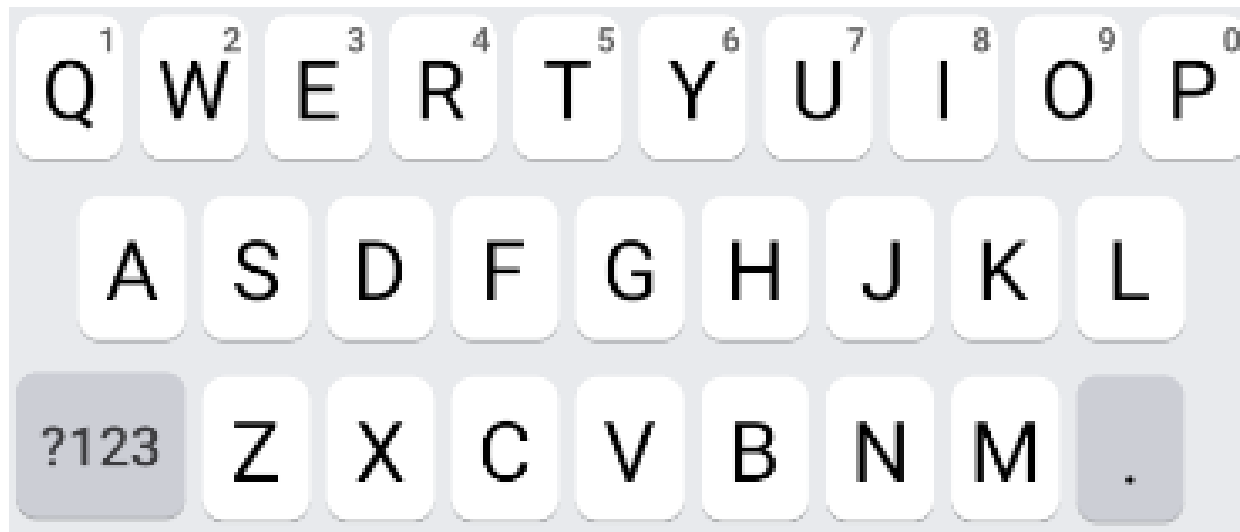
5. Tạo giao diện như sau



6*. Làm giao diện như hình sử dụng flexbox để canh chỉnh

Tham khảo thêm ở:

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>



- **Chụp hình, mô tả** quá trình làm và kết quả đạt được vào trong file word
- Lưu và đặt tên file word theo dạng: **MSSV.doc** hoặc **MSSV.docx**
- Nộp file word lên hệ thống **Canvas** theo tuần