**Component**

Giống như trong iOS thì mình sử dụng các UIButton, UITextView,... thì trong React Native những cái kiểu vậy gọi là các Component. Để sử dụng được các component chúng ta cần khai báo thư viện như sau

import React from 'react';

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

Trong đó:

**react**: để hỗ trợ giao tiếp giữa các component

**react-native**: lấy output của các component để bỏ lên màn hình điện thoại.

Chúng ta dùng component nào thì nhớ phải khai báo trong chỗ import này nhé, không là sẽ bị lỗi. Tạo một component đơn giản, ví dụ ở đây mình muốn tạo một dòng chữ hiển thị trên màn hình, chúng ta viết vào file App.js như sau:

import React, { Component } from 'react';

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

export default class HelloWorld extends Component {

render() {

return (

<View>

<Text>Hello world!</Text>

</View>

);

}

}

Màn hình sẽ hiển thị lên dòng chữ Hello world! ở trên cùng. Đấy là các component chúng ra sử dụng của thư viện react-native. Tuy nhiên chúng ta có thể tự tạo một component. Ví dụ dưới đây mình tự tạo 1 component có tên là ScreenA như sau:

export default class ScreenA extends Component {

render() {

return(

<View>

<Text> ScreenA </Text>

</View>

)

}

}

Sau đó mình sử dụng ScreenA giống như các component View hay Text bình thường

export default class HelloWorld extends Component {

render() {

return (

<View>

<ScreenA />

</View>

);

}

}

Component bản chất của nó cũng là các class, do vậy nó cũng có các thuộc tính. Trong react native có 2 loại thuộc tính là Prob và state. Sau đây mình sẽ tìm hiểu xem nó là cái gì nhé.

Button:

import { Button } from 'react-native';

...

<Button

onPress={onPressLearnMore}

title="Learn More"

color="#841584"

accessibilityLabel="Learn more about this purple button"

/>

## Style

Trong React Native, bạn không sử dụng một ngôn ngữ đặc biết hoặc ngôn ngữ có cấu trúc để định nghĩa style. Bạn chỉ cần sử dụng JavaScript để định nghĩa style cho ứng dụng của bạn. Tất cả các thành phần cơ bản sẽ sử dụng là một tham số có tên là style. Những tham số style và giá trị của chúng towngj tự với cách hoạt động của CSS trên môi trường Web, điểm khách biệt đó là tên các tham số tuân theo quy tắc Lamda (hay một số người vẫn gọi là Lạc đà), ví dụ như backgroundColorthay vì sử dụng background-color trên CSS.

Thuộc tính style là một thuộc tính đã có từ lâu trong đối tượng JavaScript. Đây là cách đơn giản nhất và cũng là những thứ mà chúng ta sẽ dùng trong code ví dụ. Bạn động thời cũng có thể khai báo một mảng các style, style cuối cùng được khai báo trong mảng style sẽ được ưu tiên, như vậy bạn hoàn toàn có thể sử dụng style theo hình thức kế thừa.

Dường như một thành phần được tạo ra sẽ phức tạp. Vậy nên nó thường được làm rõ bằng cách sử dụng StyleSheet.create để định nghĩa các style và đặt chúng tại một nơi để dễ quản lý. Dưới đây là ví dụ

import React, { Component } from 'react';

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

class LotsOfStyles extends Component {

render() {

return (

<View>

<Text style={styles.red}>just red</Text>

<Text style={styles.bigblue}>just bigblue</Text>

<Text style={[styles.bigblue, styles.red]}>bigblue, then red</Text>

<Text style={[styles.red, styles.bigblue]}>red, then bigblue</Text>

</View>

);

}

}

const styles = StyleSheet.create({

bigblue: {

color: 'blue',

fontWeight: 'bold',

fontSize: 30,

},

red: {

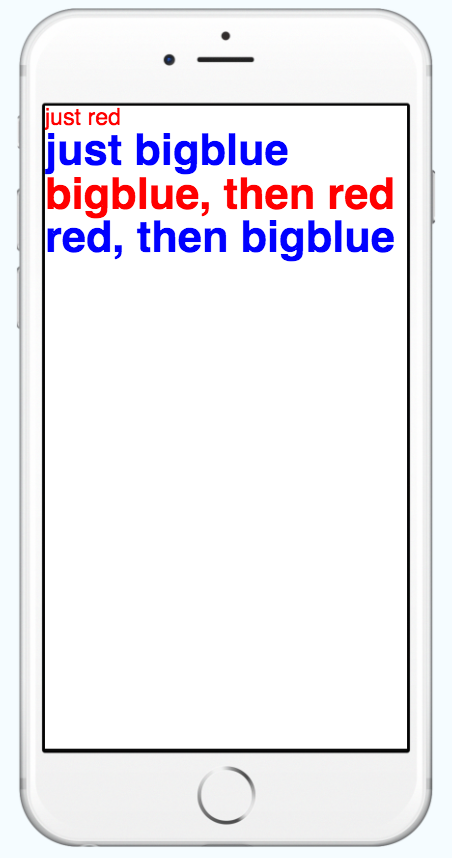
color: 'red',

},

});

AppRegistry.registerComponent('LotsOfStyles', () => LotsOfStyles);

và kết quả nhận được ở dưới đây



Một cách thức chung đó là cho tạo các thành phần của bạn cho phép tham số style truy cập trong khi nó được sử dụng như một phần phần con của style. Bạn có thể sử dụng bằng cách tạo ra các style theo mô hình thác nước mà người ta vẫn hay sử dụng trong CSS.

## Height & Width

Chiều rộng và dài của các thành phần sẽ xác định kích thước của các thành phần giao diện trên màn hình.

### Kích thước cố định

Cách đơn giản nhất để chỉ định kích thước của các thành phần là sử dụng các tham số kích thước cố định width và height trong style. Tất cả kích thước được sử dụng trong React Native đều tuân theo đơn vị pixel khi hiển thị lên trên màn hình.

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

class FixedDimensionsBasics extends Component {

render() {

return (

<View>

<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<View style={{width: 100, height: 100, backgroundColor: 'skyblue'}} />

<View style={{width: 150, height: 150, backgroundColor: 'steelblue'}} />

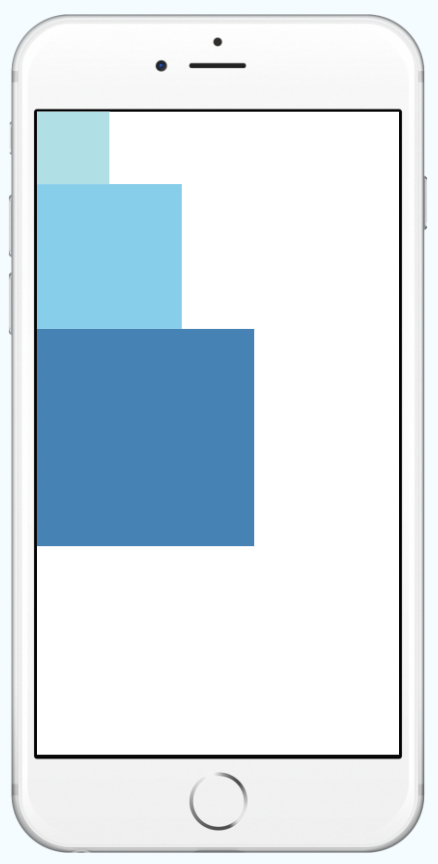
</View>

);

}

};

AppRegistry.registerComponent('AwesomeProject', () => FixedDimensionsBasics);



Để cài đặt việc hiển thị có sự tương tự nhau đối với nhiều loại kích thước màn hình các bạn đọc phần tiếp theo ngay dưới đây

### Kích thước linh động

Sử dụng flex trong style của các thành phần để thanh đổi vùng hiển thị một cách linh động dựa trên vùng hiển thị có thể. Thông thường, bạn sẽ sử dụng flex: 1 khi mà bạn muốn một thành phần trải rộng hết mức có thể so với các thành phần khác có chung một view cha. Đây là giá trị tỷ lệ cao nhất mà flex cung cấp. Với các tỷ lệ khác thì nó sẽ so sánh với các thành phần ngang hàng với nó để đưa ra tỷ lệ hiển thị trong cùng một view cha.

Một thành phần chỉ có thể mở rộng ra hết khoảng trống có thể nếu như view cha của nó có kích thước lớn hơn 0. Nếu như view cha không được gán các kích thước cố định như width và height hoặc flex, thì kích thước của view cha sẽ có kích thước là 0 và khi sử dụng flex tại view con thì nó cũng sẽ không được hiển thị.

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

class FlexDimensionsBasics extends Component {

render() {

return (

// Try removing the `flex: 1` on the parent View.

// The parent will not have dimensions, so the children can't expand.

// What if you add `height: 300` instead of `flex: 1`?

<View style={{flex: 1}}>

<View style={{flex: 1, backgroundColor: 'powderblue'}} />

<View style={{flex: 2, backgroundColor: 'skyblue'}} />

<View style={{flex: 3, backgroundColor: 'steelblue'}} />

</View>

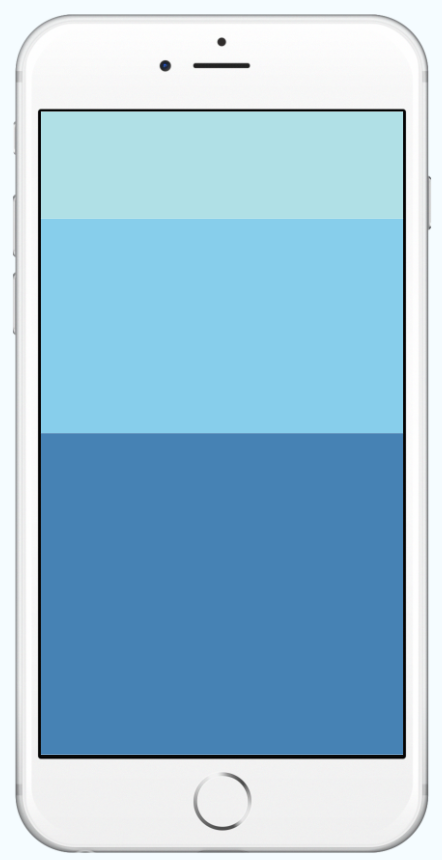
);

}

};

AppRegistry.registerComponent('AwesomeProject', () => FlexDimensionsBasics);

và kết quả chúng ta thu được ở hình dưới đây



Sau khi bạn đã có thể điều kiển được kích thước của các thành phần, bước tiếp theo chúng ta sẽ tìm hiểu cách làm thế nào để có thể thay đổi bố cục của các thành phần ở ngay phần dưới đây.

## Điều chỉnh Layout với Flexbox

Một thành phần có thể chỉ định rõ bố cục cho các view con của nó bằng cách sử dụng thuật toán của flexbox. Flexbox được thiết kế để cung cấp giao diện tương đồng trên những màn hình có kích thước khác nhau.

Bạn sẽ thường phải sử dung với các thuộc tính flexDirection, alignItems, và justifyContent để tạo ra được một giao diện đúng ý muốn.

Flexbox làm việc với React Native tương tự với cách mà CSS làm việc trên Web. nhưng cũng có một vài ngoại lệ. Giá trị mặc định là khác nhau, ví dụ như đối với thuộc tính flexDirection giá trị mặc định của nó sẽ là column thay thế cho giá trịnh row của CSS trên bản web, và giá trị của alignItems có giá trị mặc định là stretch chứ không phải là flex-start và tham số flexchỉ hỗ trợ số nguyên.

### Flex Direction

Thêm thuộc tính flexDirection vào style của một thành phần để thay đổi hướng sắp xếp các thành phần con của nó. Những thành phần con nên được sắp xếp trên cùng một hàng row hoặc hoặc trên cùng một cột column? Mặc định thì các thành phần con sẽ được sắp xếp nằm trên cùng một cột column.

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

class FlexDirectionBasics extends Component {

render() {

return (

// Try setting `flexDirection` to `column`.

<View style={{flex: 1, flexDirection: 'row'}}>

<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />

</View>

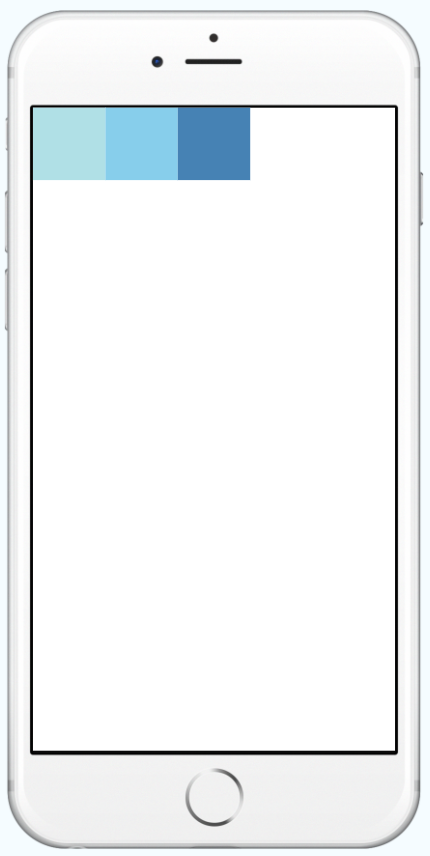
);

}

};

AppRegistry.registerComponent('AwesomeProject', () => FlexDirectionBasics);

Chúng ta cùng xem kết quả thu được nhé



### Justify Content

Thêm thuộc tính justifyContent tới style của một thành phần sẽ phân bổ theo vị trí tương đối của chúng trong cùng một hướng sắp đặt. Vậy các thanh phần con nên được phân bổ theo hướng từ điểm bắt đầu đến kết thúc, hay nằm chính giữa, hay từ điểm kết thúc lên điểm bắt đầu, hay ngang hàng nhau? Những tùy chọn đó sẽ tương ứng với những giá trị sau đây flex-start, center, flex-end, space-around, và space-between.

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

class JustifyContentBasics extends Component {

render() {

return (

// Try setting `justifyContent` to `center`.

// Try setting `flexDirection` to `row`.

<View style={{

flex: 1,

flexDirection: 'column',

justifyContent: 'space-between',

}}>

<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />

</View>

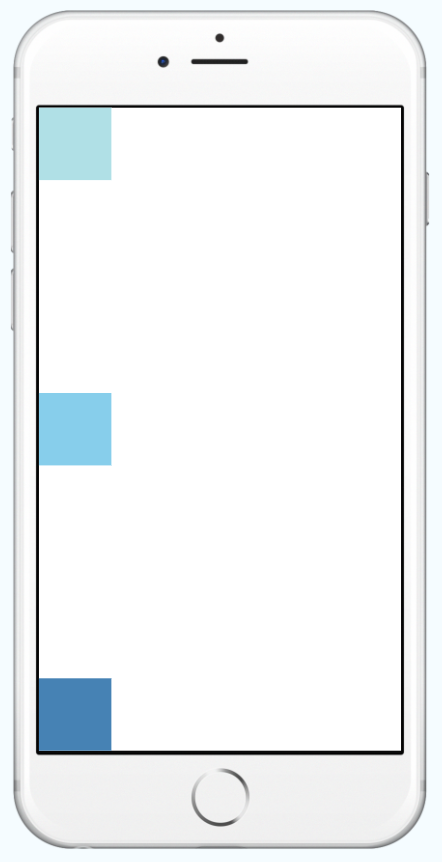
);

}

};

AppRegistry.registerComponent('AwesomeProject', () => JustifyContentBasics);

Kết quả thu được sẽ như hình dưới đây:



### Align Items

Thêm thuộc tính alignItems tới style của một thành phần để thay đổi các đối tượng con theo hướng sắp đặt còn lại(ví dụ như nếu như hướng sắp xếp lúc đầu để là row thì hướng sắp xếp còn lại sẽ là theo column, và ngược lại thì tương tự). Vậy những thành phần con nên được đặt theo đầu đến cuối, ở trung tâm, hay tự động co giãn để lấp đầy đối tượng cha? Những giá trị tùy chọn đó sẽ tương ứng ở đây flex-start, center, flex-end, và stretch.

Để cho thuộc tính stretch có thể thể hiện được, thì những view con cần phải có một kích thước cố định trong hướng sắp đặt còn lại. Như ở ví dụ bên dưới đây, việc cài đặt alignItems: stretch sẽ không hoạt động trừ khi thuộc tính width: 50 bị xóa khỏi style của các view con.

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

class AlignItemsBasics extends Component {

render() {

return (

// Try setting `alignItems` to 'flex-start'

// Try setting `justifyContent` to `flex-end`.

// Try setting `flexDirection` to `row`.

<View style={{

flex: 1,

flexDirection: 'column',

justifyContent: 'center',

alignItems: 'center',

}}>

<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />

<View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />

</View>

);

}

};

AppRegistry.registerComponent('AwesomeProject', () => AlignItemsBasics);

