**COMPONENT TRONG REACT NATIVE**

* **Nhóm 3**
* **Thành viên:**
  + **Nguyễn Quang Hiểu**
  + **Phạm Trọng Duyễn**
  + **Vũ Thái Ngân**
  + **Hoàng Văn Giang**

**Tổng quan về Component**

1. **Props và State( Hoàng Văn Giang)**

+ Props

Hầu hết các components đều có thể được tùy chỉnh khi chúng ta tạo ra nó, với những tham số khác nhau. Những tham số này được gọi là props.

Ví dụ, một thành phần cơ bản trong React Native là Image. Khi bạn tạo ra một image, bạn có thể sử dụng một prop có tên là source để điểu khiển image nào sẽ được hiển thị ra.



Chú ý rằng {pic} được bao ngoài bởi dấu móc, để hiển thị biến pic trong JSX, Bạn có thể đặt bất cứ câu lệnh JavaScript trong kết cấu của JSX.

Các components của bạn đồng thời có thể sử dụng props. Điều này cho phép bạn tạo ra các components đơn để sử dụng ở nhiều nơi khác nhau trong ứng dụng của bạn với tính chất khác nhau ở mỗi một nơi mà bạn dùng. Chỉ việc gọi đến this.props trong phương thức render của bạn. Dưới đây là ví dụ:



Sử dụng name như một prop cho phép chúng ta tùy chỉnh component Greeting, vì vậy chúng ta có thể sử dụng lại các thành phần cho mỗi Greeting của chúng ta. Ví dụ này cũng sử dụng các component Greeting trong JSX, giống như một building in component. Lợi ích của việc này là làm cho React tuyệt vời hơn, nếu như bạn tìm thấy những thứ bản thân mong muốn và bạn muốn cài đặt những thứ khác nhau trong giao diện thì bạn chỉ cần tạo ra ra một component mới.

Điều mới mẻ khác xảy ra ở đây là component <View>. Component View rất hữu dụng, nó là một container cho các component khác, để giúp điều khiển style và giao diện.

Với props và các components cơ bản như Text, Image, và View bạn có thể xây dựng một loạt các màn hình tĩnh. Để tìm hiểu làm thế nào để thực hiện thay đổi ứng dụng của bạn theo thời gian, bạn cần phải tìm hiểu về State ở ngay phần bên dưới đây.

+ State

Có hai kiểu dữ liệu để điều khiển một component: props và state. props đươc cài đặt bởi cha của nó và chúng sẽ được cố định trong suốt thời gian hiển thị của Component, Để dữ liệu có thể thay đổi được, chúng ta sẽ cần phải sử dụng đến state.

Thông thường, bạn nên khởi tạo state trong hàm khởi tạo, và sau đó bạn có thể gọi setState khi bạn muốn thay đổi nó.

Ví dụ như, nếu bạn muốn làm cho một văn bản nhấp nháy liên tục trong toàn bộ thời gian, Văn bản đó sẽ được cài đặt một lần khi component được tạo ra, vậy bản thẩn nó sẽ là một prop. Và thở thời điểm này thì băn bản sẽ được hiển thị mà không bị thay đổi trong quá trình chạy, thế nên nó lên giữ thêm một state.





Với một ứng dụng thực tế, bạn không nên cài đặt state với một timer. Bạn có thể cài đặt statekhi bạn có dữ liệu mới nhận được từ server hoặc từ phía người dùng nhập vào. Bạn đồng thời cũng có thể dử dụng một state container giống như Redux để điều khiển luồng dữ liệu. Trong trường hợp này bạn nên sử dụng Redux để thay đổi state của bạn sẽ hay hơn là việc gọi điều hướng đến setState.

State làm việc giống với cách mà nó đã làm việc trên React. thế thế để xem chi tiết hơn về việc điều khiển state bạn cso thể xem trong [React.Component API]

1. **Style trong React Native(Phạm Trọng Duyễn).**

Với React-Native hỗ trợ style cho tất cả các components . Nó đơn giản như bạn viết CSS cho Web. Tuy nhiên có một số thay đổi về quy định cách thức viết các selector, hay cách gọi css…các bạn cần nắm rõ các quy định đó. Ví dụ trong CSS sử dụng cho HTML chúng ta viết background-color thì sang React-Native chúng ta sử dụng backgroundColor Thao tác với css cho đối tượng, tức là bạn sẽ thao tác trong class hiển thị ra phần view. Có 3 kiểu khai báo styles:

1.Khai báo trực tiếp trong Component

import React, { Component } from 'react';

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

class StylesInComponent extends Component {

render() {

return (

<View style={{

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: 'red',

}}>

<Text style={{

fontSize: 20,

textAlign: 'center',

margin: 10,

color: 'white',

}}>

Welcome to React Native!

</Text>

</View>

);

}

}

2. Khai báo ngoài Component nhưng vẫn nằm trong một file Đầu tiên bạn cần import thư viện StyleSheet Tiếp theo bên ngoài Component bạn khai báo biến styles kiểu StyleSheet. Bạn sẽ tạo các selector CSS :

* container -> View
* welcome -> Text
* const styles = StyleSheet.create({
* container: {
* flex: 1,
* justifyContent: 'center',
* alignItems: 'center',
* backgroundColor: '#F5FCFF',
* },
* welcome: {
* fontSize: 20,
* textAlign: 'center',
* margin: 10,
* },
* });

Quay vào trong class cha chứa các component, bạn đơn giản chỉ cần gọi biến styles để lựa chọn các selector tương ứng cho component

class StylesOutComponent extends Component {

render() {

return (

<View style={styles.container}>

<Text style={styles.welcome}>

Welcome to React Native!

</Text>

</View>

);

}

}

});

**3**. Khai báo vào file riêng

Tạo file có tên Styles.js với nội dung như sau :

* import { StyleSheet,} from 'react-native';
* const styles = StyleSheet.create({
* container: {
* flex: 1,
* justifyContent: 'center',
* alignItems: 'center',
* backgroundColor: '#F5FCFF',
* },
* welcome: {
* fontSize: 20,
* textAlign: 'center',
* margin: 10,
* }
* });
* export default styles;

Đầu tiên bạn cần import thư viện StyleSheet Tương tự bạn tạo cái selector CSS : container, welcome cho biến styles Quan trọng nhất là bạn phải thêm dòng export default styles; vào cuối file. Khi bạn đã export thì có nghĩa là bạn có thể import styles ở bất kì đâu để sử dụng. Sử dụng câu lệnh import styles from './Styles' . Trong đó './Styles' là đường dẫn của tới Styles.js

Ví dụ:

import styles from './Styles'

class StylesNewFile extends Component {

render() {

return (

<View style={styles.container}>

<Text style={styles.welcome}>

Welcome to React Native!

</Text>

</View>

);

}

}

1. **Width, Height và Flex(Vũ Thái Ngân)**

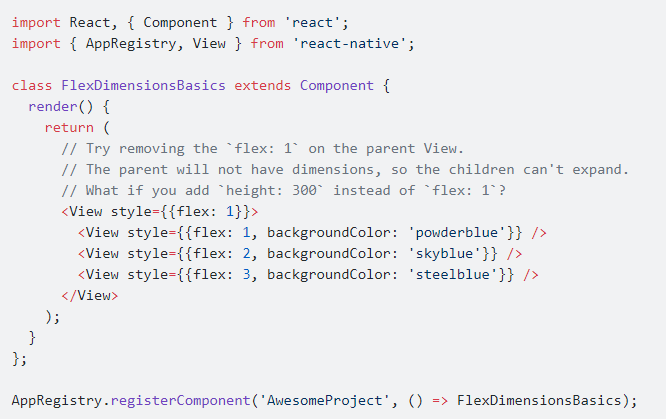
## + Height & Width: Chiều rộng và dài của các thành phần sẽ xác định kích thước của các thành phần giao diện trên màn hình. Cách đơn giản nhất để chỉ định kích thước của các thành phần là sử dụng các tham số kích thước cố định width và height trong style. Tất cả kích thước được sử dụng trong React Native đều tuân theo đơn vị pixel khi hiển thị lên trên màn hình.

## 

**\* Flex** :

- 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.



1. **FlexBox(Nguyễn Quang Hiểu)**

Một thuộc tính giúp căn chỉnh vị trí tương đối của các thành phần con trong thành phần cha.

Có 2 đối tượng chính trong Flexbox:

+ Đối tượng bao quanh (container) chính là view cha

+ Đối tượng con bên trong (items) là các view con

Phần 1: Flexbox cho đối tượng cha (container)

* 1. *flexDirection*

Sắp xếp các đối tượng con theo column (chiều dọc) hoặc row (chiều ngang). 2 giá trị của nó là : row hoặc column.

Cú pháp khai báo: flexDirection: ‘chiều khai báo’.

* 1. *justifyContent*

Căn vị trí tương đối của đối tượng con theo trục chính của flex.

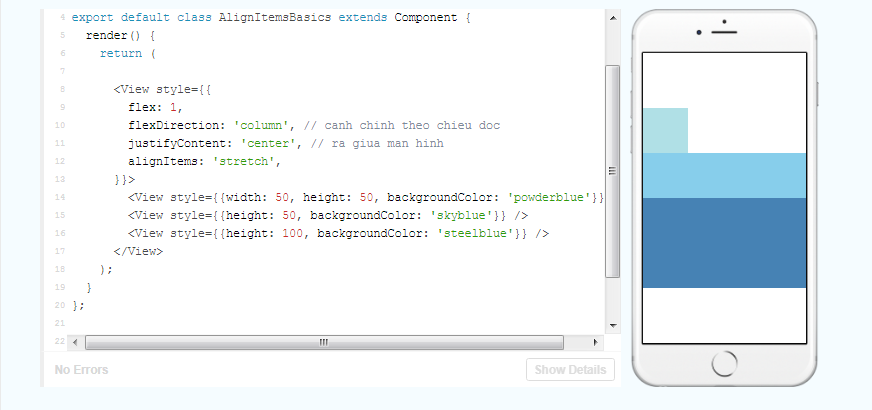
Gồm các giá trị: flex-start, center, flex-end, space-around, space-between

* 1. *alignItems*

Căn vị trí tương đối của các đối tượng con theo trục còn lại. Gồm các giá trị: flex-start, flex-end , center, strecht

* 1. *Flex-wrap* : wrap | no-wrap

Thuộc tính này cho phép container bọc các items con mặc dù kích thước của chúng vượt qua kích thước của container. Măc định là no-wrap.



Phần 2: Flexbox cho đối tượng con (items)

1.1 Oder

Sắp xếp vị trí các items theo giá trị nhập

1.2 Flex-grow

Phân định kích thước của các đối tượng con tương đối bằng tỉ lện giữa các giá trị flex-grow.

1.3 Flext-shrinlk

Giá trị này sẽ quyết định tỉ lệ co giãn giữa các items khi co giãn màn hình.