TÌM HIỂU REACT-NATIVE

1. **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.
* import React, { Component } from 'react';
* import { AppRegistry, Image } from 'react-native';
* class Bananas extends Component {
* render() {
* let pic = {
* uri: 'https://upload.wikimedia.org/wikipedia/commons/d/de/Bananavarieties.jpg'
* };
* return (
* <Image source={pic} style={{width: 193, height: 110}}/>
* );
* }
* }
* AppRegistry.registerComponent('Bananas', () => Bananas);
* 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:
* import React, { Component } from 'react';
* import { AppRegistry, Text, View } from 'react-native';
* class Greeting extends Component {
* render() {
* return (
* <Text>Hello {this.props.name}!</Text>
* );
* }
* }
* class LotsOfGreetings extends Component {
* render() {
* return (
* <View style={{alignItems: 'center'}}>
* <Greeting name='Rexxar' />
* <Greeting name='Jaina' />
* <Greeting name='Valeera' />
* </View>
* );
* }
* }
* AppRegistry.registerComponent('LotsOfGreetings', () => LotsOfGreetings);
* 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.
* 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.

1. **State**

* Nếu như **Props** được cài đặt bởi cha của nó và chúng sẽ được cố định trong suốt thời gian hiện thị của **Component**. Muốn dữ liệu có thể thay đổi được chúng ta sẽ 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.
* import React, { Component } from 'react';
* import { AppRegistry, Text, View } from 'react-native';
* class Blink extends Component {
* constructor(props) {
* super(props);
* this.state = {showText: true};
* // Toggle the state every second
* setInterval(() => {
* this.setState({ showText: !this.state.showText });
* }, 1000);
* }
* render() {
* let display = this.state.showText ? this.props.text : ' ';
* return (
* <Text>{display}</Text>
* );
* }
* }
* class BlinkApp extends Component {
* render() {
* return (
* <View>
* <Blink text='I love to blink' />
* <Blink text='Yes blinking is so great' />
* <Blink text='Why did they ever take this out of HTML' />
* <Blink text='Look at me look at me look at me' />
* </View>
* );
* }
* }
* AppRegistry.registerComponent('BlinkApp', () => BlinkApp);
* Với một ứng dụng thực tế, bạn không nên cài đặt **state** với một timer. Bạn có thể cài đặt **state** khi 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**.

1. **Text**

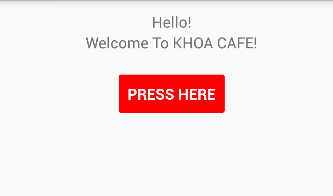
Để có thể dùng Text trong React-native chúng ta import thư viên Text.



Và để xuất các dòng Text ra màn hình ứng dụng ta để trong lệnh <Text>…</Text>



Kết quả khi chạy

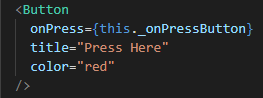


1. **Button**

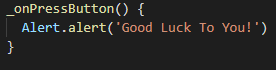
Để có thể dùng Button trong React-native chúng ta import thư viện Button.



Và để xuất các Button ra màn hình ứng dụng ta để trong lệnh <Button…/>



onPress: gọi hàm \_onPessButton khi người dùng nhấn vào Button



title: hiện chữ của trên button

color: định dạng màu cho button.

