**Báo Cáo**

**Tên: Đặng Phương Đông**

**Nhóm: 4**

**Tìm hiểu về các Component cơ bản**

1. **TextInput Component**
   1. **Khái niệm**

Là một thành phần cho phép người dùng nhập văn bản.

* 1. **Ví dụ**

import \* as React from 'react';

import { TextInput } from 'react-native-paper';

export default class MyComponent extends React.Component {

state = {

text: ''

};

render(){

return (

<TextInput

label='Email'

value={this.state.text}

onChangeText={text => this.setState({ text })}

/>

);

}

}

* 1. **Props**
     + [allowFontScaling](https://facebook.github.io/react-native/docs/textinput#allowfontscaling)
* Chỉ định xem phông chữ có nên mở rộng để tôn trọng cài đặt trợ năng Kích thước văn bản hay không. Mặc định là đúng.
  + - [autoCapitalize](https://facebook.github.io/react-native/docs/textinput#autocapitalize)
* Có thể yêu cầu TextInput tự động viết hoa các ký tự nhất định. Thuộc tính này không được hỗ trợ bởi một số loại bàn phím như tên-điện thoại-pad.
  + - [autoCorrect](https://facebook.github.io/react-native/docs/textinput#autocorrect)
* Nếu sai, tắt tự động sửa. Giá trị mặc định là true.
  + - [autoFocus](https://facebook.github.io/react-native/docs/textinput#autofocus)
* Nếu đúng, hãy tập trung đầu vào vào componentDidMount. Giá trị mặc định là sai.
  + - [blurOnSubmit](https://facebook.github.io/react-native/docs/textinput#bluronsubmit)
* Nếu đúng, trường văn bản sẽ mờ khi được gửi. Giá trị mặc định là đúng cho các trường một dòng và sai cho các trường đa dòng. Lưu ý rằng đối với các trường nhiều dòng, hãy đặt blurOnSubmit thành true có nghĩa là nhấn return sẽ làm mờ trường và kích hoạt sự kiện onSubmitEditing thay vì chèn một dòng mới vào trường.
  + - [caretHidden](https://facebook.github.io/react-native/docs/textinput#carethidden)
* Nếu đúng, dấu mũ bị ẩn. Giá trị mặc định là sai.
  + - [clearButtonMode](https://facebook.github.io/react-native/docs/textinput#clearbuttonmode)
* Khi nút rõ ràng sẽ xuất hiện ở phía bên phải của chế độ xem văn bản. Thuộc tính này chỉ được hỗ trợ cho thành phần TextInput một dòng. Giá trị mặc định là không bao giờ.
  + - [clearTextOnFocus](https://facebook.github.io/react-native/docs/textinput#cleartextonfocus)
* Nếu đúng, hãy xóa trường văn bản tự động khi chỉnh sửa bắt đầu.
  + - [contextMenuHidden](https://facebook.github.io/react-native/docs/textinput#contextmenuhidden)
* Nếu đúng, menu ngữ cảnh bị ẩn. Giá trị mặc định là sai.
  + - [defaultValue](https://facebook.github.io/react-native/docs/textinput#defaultvalue)
* Cung cấp giá trị ban đầu sẽ thay đổi khi người dùng bắt đầu nhập. Hữu ích cho các trường hợp sử dụng đơn giản mà bạn không muốn đối phó với việc nghe các sự kiện và cập nhật giá trị prop để giữ trạng thái được điều khiển đồng bộ.
  + - [editable](https://facebook.github.io/react-native/docs/textinput#editable)
* Nếu sai, văn bản không thể chỉnh sửa được. Giá trị mặc định là true.
  + - [enablesReturnKeyAutomatically](https://facebook.github.io/react-native/docs/textinput#enablesreturnkeyautomatically)
* Nếu đúng, bàn phím sẽ tắt phím quay lại khi không có văn bản và tự động bật nó khi có văn bản. Giá trị mặc định là sai.
  + - …

1. **Listview**
   1. **Khái niệm**

React Native cung cấp một bộ component cho việc xuất ra danh sách các phần tử. Nói chung, bạn sẽ cần sử dụng “FlatList” hay “SectionList”.

Component “FlatList” xuất ra một danh sách cuộn có thể thay đổi tương tự với cấu trúc dữ liệu. “FlatList” làm việc tốt đối với những danh sách dữ liệu dài, nơi mà số hạng mục thay đổi theo thời gian. Không giống với “ScrollView” , “FlatList” chỉ trả về các phần tử đang hiển thị trên màn hình, không phải là tất cả các phần tử cùng lúc.

“FlatList” yêu cầu 2 thuộc tính: “data” và “renderItem”. “data” chứa các nguồn thông tin bên trong danh sách còn “renderItem” lấy một item trong đó và trả về một định dạng để hiển thị.

* 1. **Ví dụ**

class MyComponent extends Component {

constructor() {

super();

const ds = new ListView.DataSource({rowHasChanged: (r1, r2) => r1 !== r2});

this.state = {

dataSource: ds.cloneWithRows(['row 1', 'row 2']),

};

}

render() {

return (

<ListView

dataSource={this.state.dataSource}

renderRow={(rowData) => <Text>{rowData}</Text>}

/>

);

}

}

* 1. **Props**
     + dataSource
* Một phiên bản của ListView.DataSource để sử dụng
  + - initialListSize
* Có bao nhiêu hàng để kết xuất trên mount thành phần ban đầu. Sử dụng công cụ này để làm cho nó có giá trị dữ liệu màn hình đầu tiên xuất hiện cùng một lúc thay vì qua nhiều khung.
  + - onEndReachedThreshold
* Ngưỡng tính bằng pixel (ảo, không phải vật lý) để gọi onEndReached.
  + - pageSize
* Số lượng hàng để kết xuất trên mỗi vòng lặp sự kiện. Lưu ý: nếu 'hàng' của bạn thực sự là các ô, tức là chúng không trải rộng toàn bộ chiều rộng của chế độ xem của bạn (như trong ListViewGridLayoutExample), bạn nên đặt kích thước trang là bội số của mỗi ô, nếu không bạn có thể thấy các khoảng trống ở rìa ListView khi các trang mới được tải.
  + - renderRow
* (rowData, partID, rowID, highlightRow) => renderable
* Lấy một mục nhập dữ liệu từ nguồn dữ liệu và id của nó và sẽ trả về một thành phần có thể kết xuất được hiển thị dưới dạng hàng. Theo mặc định, dữ liệu chính xác là những gì được đưa vào nguồn dữ liệu, nhưng cũng có thể cung cấp các trình trích xuất tùy chỉnh. ListView có thể được thông báo khi một hàng đang được tô sáng bằng cách gọi highlightRow (phầnID, rowID). Điều này đặt giá trị boolean của liền kềRowHighlighted trong renderSpayator, cho phép bạn điều khiển các dấu phân cách ở trên và dưới hàng được tô sáng. Trạng thái được tô sáng của một hàng có thể được đặt lại bằng cách gọi highlightRow (null).
  + - renderScrollComponent
* Một hàm trả về thành phần có thể cuộn trong đó các hàng danh sách được hiển thị. Mặc định để trả về một ScrollView với các đạo cụ đã cho.
  + - scrollRenderAheadDistance
* Làm thế nào sớm để bắt đầu kết xuất hàng trước khi chúng xuất hiện trên màn hình, tính bằng pixel.
  + - stickyHeaderIndices
* Một loạt các chỉ số con xác định những đứa trẻ nào được neo vào đầu màn hình khi cuộn. Ví dụ: việc chuyển stickHeaderInices = {[0]} sẽ khiến đứa trẻ đầu tiên được cố định vào đầu chế độ xem cuộn. Thuộc tính này không được hỗ trợ kết hợp với horizonta = {true}.
  + - enableEmptySections
* Cờ cho biết liệu tiêu đề phần trống sẽ được hiển thị. Trong bản phát hành tương lai, các tiêu đề phần trống sẽ được hiển thị theo mặc định và cờ sẽ không được chấp nhận. Nếu các phần trống không mong muốn được hiển thị, các chỉ mục của chúng sẽ được loại trừ khỏi đối tượng partID.
  + - onEndReached
* Được gọi khi tất cả các hàng đã được hiển thị và danh sách đã được cuộn đến bên trong onEndReachedThrưỡng của phía dưới. Sự kiện cuộn bản địa được cung cấp.
  + - …

1. **Style**
   1. **Khái niệm**

Với React Native, bạn không sử dụng ngôn ngữ hoặc cú pháp đặc biệt để xác định kiểu. Bạn chỉ cần tạo kiểu cho ứng dụng của mình bằng JavaScript. Tất cả các thành phần cốt lõi chấp nhận một phong cách tên prop. Tên và giá trị kiểu thường khớp với cách CSS hoạt động trên web, ngoại trừ tên được viết bằng vỏ lạc đà, ví dụ như phông nền thay vì màu nền.

Kiểu prop có thể là một đối tượng JavaScript cũ đơn giản. Đó là cách đơn giản nhất và những gì chúng ta thường sử dụng ví dụ mã. Bạn cũng có thể vượt qua một loạt các kiểu - kiểu cuối cùng trong mảng được ưu tiên, vì vậy bạn có thể sử dụng kiểu này để kế thừa kiểu.

* 1. **Ví dụ**

import React, { Component } from 'react';

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

const styles = StyleSheet.create({

bigblue: {

color: 'blue',

fontWeight: 'bold',

fontSize: 30,

},

red: {

color: 'red',

},

});

export default 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>

);

}

}

// skip this line if using Create React Native App

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

1. **Layout with Flexbox**
   1. **Khái niệm**

Một thành phần có thể chỉ định bố cục của các phần tử con của nó bằng thuật toán flexbox. Flexbox được thiết kế để cung cấp một bố cục nhất quán trên các kích thước màn hình khác nhau.

Thông thường bạn sẽ sử dụng kết hợp flexDirection, alignItems và justifyContent để đạt được bố cục phù hợp.

* 1. **Ví dụ**

import React, { Component } from 'react';

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

export default 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: 'stretch',

}}>

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

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

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

</View>

);

}

};

// skip this line if using Create React Native App

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