**BÁO CÁO TUẦN 2.2**

**Tên nhóm:** Nhóm 4

**Thành viên**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | Email | SDT |
| 1 | Nguyễn Hữu Định | huudinh1401.tdc@gmail.com | 0961341401 |
| 2 | Nguyễn Văn Hải | nguyenhai1449@gmail.com | 0967678646 |
| 3 | Đặng Phương Đông | dpdong225@gmail.com | 0358846070 |

Email nhóm: ltdd3nhomhaidinh@gmail.com

Pass: nhomhaidinh

Github: <https://github.com/NhomHaiDinh/LTDD3-Project>

Đề tài: Ứng dụng dự báo thời tiết

**Tìm hiểu :**

1. **Tìm hiểu props và state**
   1. **Props**
      * Các thuộc tính riêng của các thẻ có thể được sử dụng như giá trị cho các thẻ con mà có sử dụng giá trị đó. Điều này cho phép chúng ta tạo ra một thành phần giao diện duy nhất nhưng lại có giá trị khác nhau trong ứng dụng của chúng ta.
      * Để gọi props: this.props trong phương thức của render.
      * Với props các thành phần Text, Image, View chúng ta có thể xây dựng một loại màn hình tĩnh khác nhau.

import React, { Component } from 'react';

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

class **Greeting** extends **Component** {

**render**() {

return (

<**Text**>Hello {this.props.name}!</**Text**>

);

}

}

export default class **LotsOfGreetings** extends **Component** {

**render**() {

return (

<**View** *style*={{alignItems: 'center'}}>

<**Greeting** *name*='Rexxar' />

<**Greeting** *name*='Jaina' />

<**Greeting** *name*='Valeera' />

</**View**>

);

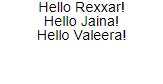
}

}

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => LotsOfGreetings);

// Kết quả là :



* + - Ở ví dụ trên chúng ta sử dụng name như là một prop cho phép chúng ta tùy chỉnh thành phần Greeting, vì vậy chúng ta có thể tái sử dụng thành phần đó cho mỗi lời chào của chúng ta.
  1. **State**
     + Trong react native chúng ta có hai loại để kiểm soát các thành phân đó là Props và State. Props được thiết lập giá trị và chúng sẽ cố định trong suốt vòng đời của thành phần. Còn với State chúng ta có thể khiến dữ liệu của nó thay đổi tùy theo hành động của chúng ta xử lý trong chương trình.
     + Chúng ta nên khởi tạo giá trị state trong hàm khởi tạo và setState khi ta muốn thau đổi giá trị đó.
     + Giả sử chúng ta muốn làm cho một văn bản nhấp nháy mọi lúc. Bản thân văn bản được đặt một lần khi thành phần nhấp nháy được tạo, và chính bản thân văn bản là prop.

import React, { Component } from 'react';

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

class **Blink** extends **Component** {

constructor(props) {

super(props);

this.state = {isShowingText: true};

*// Toggle the state every second*

**setInterval**(() => {

this.**setState**(previousState => {

return { isShowingText: !previousState.isShowingText };

});

}, 1000);

}

**render**() {

let display = this.state.isShowingText ? this.props.text : ' ';

return (

<**Text**>{display}</**Text**>

);

}

}

export default 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**>

);

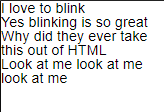
}

}

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => BlinkApp);

//Kết quả:



1. **Style trong React Native**
   * + Với react native chúng ta không sử dụng ngôn ngữ đặc biệt hoặc một cú pháp đặc biệt để định nghĩa styles. style của ứng dụng sử dụng bằng JavaScript. Các giá trị để thiết lập style có giá trị giống với CSS tương tự với web. Nhưng các thuộc tính trong CSS lại viết theo kiểm đó là camelCasing. VD backgroundColor.
     + Chúng ta có thể sử dụng style như là một mảng.
     + Chúng ta có thể định nghĩa style tại một nơi khác thường đặt là StyleSheet.create
     + VD:

import React, { Component } from 'react';

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

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**>

);

}

}

const styles = **StyleSheet**.**create**({

bigblue: {

color: 'blue',

fontWeight: 'bold',

fontSize: 30,

},

red: {

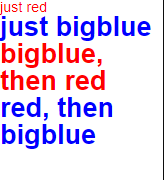
color: 'red',

},

});

*// skip this line if using Create React Native App*AppRegistry.**registerComponent**('AwesomeProject', () => LotsOfStyles);

**//Kết quả là:**



1. **Width, Height và Flex**
   * + Width, Height, Flex là thuộc tính css cơ bản được sử dụng trong thuộc tính style của các thẻ.
     + Dùng width dùng để quy định chiều rộng của thẻ đối tượng giao diện.
     + Dùng Height để quy định chiều cao của thẻ đối tượng giao diện.
     + VD: Width và Height

import React, { Component } from 'react';

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

export default 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**>

);

}

}

*// skip this line if using Create React Native App*

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

 //Kết quả là:

**Thuộc tính Flex**

* Đây cũng là một thuộc tính thuộc style của các thẻ component được set style.
* Giá trị của nó là một số nguyên dương có tác dụng chia tỉ lệ cho các thành phần được set flex.
* Sử dụng flex trong syle của component để mở rộng và thu hẹp tự động dựa trên không gian có sẵn. Giống như chia tỉ lệ các thành phần trong android vậy đó.
* Thông thường chúng ta sử dụng flex: 1 nó sẽ giúp fill tất cả không gian còn lại với điều kiện các thành phần khác chúng ta không chia flex.
* Nếu có nhiều flex trong các thẻ component khác nhau thì thẻ nào có giá trị flex lớn hơn sẽ chiếm nhiều không gian hơn thành phần khác.

import React, { Component } from 'react';

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

export default 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**>

);

}}

 //Kết quả là:

1. **FlexBox**

* Đây là một component có thể xác định và bố trí các thẻ con của nó bằng cách sử dụng các thuộc tính flex box.
* Flexbox được thiết kế để cung cấp bố cục nhất quan trên các kích thước màn hình khác nhau.
* Chúng ta thường sử dụng các loại flexbox sau: flexDirection, alignItems, justifyContent để đạt được bố cục mà chúng ta mong muốn.

1. **Flex Direction**

Thuộc tính Flex Direction là một thành phần trong thẻ style có hai giá trị định nghĩa hai hướng đó là horizontally với giá trị row và vertically với giá trị column . Dùng định hướng cho các thành phần con trong view đó như cho phép mở rộng các thành phần con theo chiều ngang hoặc chiều dọc tương tự như cách sử dụng LinearLayout trong android.

import React, { Component } from 'react';

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

export default 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**>

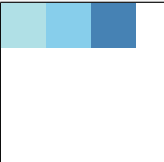
);

}

};

*// skip this line if using Create React Native App*

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

 //Kết quả là:

1. **Justify Content**

* Sử dụng JustifyContent vào component cha để xác định sự phân bố của các thẻ con dọc theo trục chính chiều của trục thì phụ thuộc vào giá trị của thuộc tính FlexDirection. Các thẻ con có thể gắn ở đầu, ở giữa hoặc cách đều nhau.
* Các giá trị của thuộc tính này là: flex-start (Gắn ở đầu thẻ cha), center (Gắn vào giữa thẻ cha), flex-end (Gắn vào cuối thẻ cha), space-around , space-between (Căn đều các thẻ con sao cho chiếm hết toàn bộ không gian thẻ cha), space-evenly.

import React, { Component } from 'react';

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

export default 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**>

);

}

};

*// skip this line if using Create React Native App*

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

//Kết quả là:



1. **Align Items**

* Sử dụng alignItems để xác định kiểu căn chỉnh của một thành phần các thẻ con theo định hướng của thẻ cha.
* Các giá trị của thuộc tính này là: flex-start, center, flex-end, stretch.

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);