**NHÓM 7**

Thành Viên:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ Và Tên | Email | SDT |
| 1 | Nguyễn Đức Lai | nguyenduclai1997@gmail.com | 0338001520 |
| 2 | Nguyễn Đăng Khoa | jeremi220497@gmail.com | 0967286764 |
| 3 | Phan Văn Khải | khajphan8@gmail.com | 0326027536 |
| 4 | Bùi Đức Thảo | thaobui290896@gmail.com | 0798775789 |

Email nhóm: didong3nhom7@gmail.com

Pass: Didong1234657

Đề tài: Ứng dụng quản lý nhân viên

Tuần 2.2

Ngày 07/11/2018

**1/Props (Nguyễn Đăng Khoa)**  
- Props  là viết tắt của Properties, ý tưởng của Props  đó là việc trừu tượng hoá các component để có thể sử dụng được ở nhiều chỗ khác nhau trong app.

VD:   


**2/ State (Phan Văn Khải)**

State biểu diễn trạng thái của component, state là private chỉ có thể thay đổi bên trong bản thân của chính component đó. Chúng ta có thể change states bằng cách gọi this.setState()

Một ví dụ sử dụng state



Ở đây chúng ta khởi tạo state với name: 'your name'

Hàm onChange được gọi mỗi khi giá trị input thay đổi và nó sẽ setState name bằng giá trị input

state.name được truyền vào component Notification thông qua props.children

Mỗi khi state.name thay đổi thì component Notification sẽ được render lại.

Kết quả:



Điểm khác nhau giữa props và state:



**3/Style (Bùi Đức Thảo)**  
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 prop có tên là Style. Tên và giá trị kiểu thường khớp với cách CSS hoạt động trên web, ngoại trừ tên được viết bằng cách sử dụng vỏ camel, ví dụ: màu background chứ không phải màu nền.

Style bảo vệ có thể là đối tượng JavaScript object cũ. Đó là đơn giản nhất và những gì chúng ta thường sử dụng cho mã ví dụ. Bạn cũng có thể truyền một mảng style – styles cuối cùng trong mảng có ưu tiên, vì vậy bạn có thể sử dụng để kế thừa style.

Là thành phần phát triển phức tạp, thường sử dụng StyleSheet.create để xác định một số style ở một nơi. Ví dụ:





**4/Width và Height (Nguyễn Đức Lai)**  
Chiều rộng và chiều dài của các thành phần sẽ xác định kích thước của các thành phần giao diện trên màn hình.

+ Sử dụng với kiểu cố định

* Cách đơn giản để chỉ định kích thước của các thành phần là sử dụng tham số kích thước cố định width và height trong style. Tất cả kích thước được sử dụng trong React Native sử dụng đơn vị pixel khi hiển thị lên màn hình.
* Nên hạn chế sử dụng kiểu cố định vì không có sự tương thích nhiều với tất cả các loại kích thước màn hình khác nhau, sẽ gây ra thiếu thẫm mĩ hoặc “vỡ” giao diện khi sử dụng.

Vd:

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);

5/ Flexbox with Layout (Nguyễn Đăng Khoa)

- 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.  
- Thường phải sử dung với các thuộc tính flexDerection, justifyContent, alignItems để tạo ra được một giao diện đúng ý muốn.

**\* Flex Decrection**  
- Thêm thuộc tính flexDecrection 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).   
VD:  


Kết Quả:  


**\*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. Có những tùy chọn tương ứng với những giá trị sau đây flex-start (hướng từ điểm bắt đầu đến kết thúc), center (nằm chính giữa), flex-end (từ điểm kết thúc lên điểm bắt đầu), space-around và space-between (ngang hàng nhau.

VD:  
  
Kết quả:  


**\* Align Item**  
- Thêm thuộc tính alignItem 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 (đặt theo đầu đến cuối),center (ở trung tâm),flex-end (đặt theo cuối đến đầu) và strench (tự động co giãn để lấp đầy đối tượng cha).

VD:  


Kết quả:  
