TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN

--------------------------------

BÁO CÁO KẾT THÚC MÔN HỌC

Lập trình di động 3

**QUẢN LÝ NHÂN VIÊN**

Giảng viên hướng dẫn: TRƯƠNG BÁ THÁI

Sinh viên thực hiện:

1. Nguyễn Đăng Khoa
2. Nguyễn Đức Lai
3. Phan Văn Khải
4. Bùi Đức Thảo

Ngành: Công nghệ thông tin Lớp: Di Động 3 Khoá: 2015

*Tp. Hồ Chí Minh*, ngày 12 tháng 10 năm 2018

**NHẬT KÝ HOẠT ĐỘNG NHÓM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stt** | **Họ và tên** | **Công việc đã thực hiện** | **Tự đánh giá** | **Nhóm đánh giá** | **Chữ ký** |
| 1 | Nguyễn Đăng Khoa | 1. Tìm Hiểu Cài Đặt react native  2. Tìm Hiểu Class  3. Tìm hiểu Props , Plex layout , vẽ giao diện  4. Tìm hiểu image , TouchableOpacy , TextInput  5. Tìm hiểu listview  6. Tìm hiểu navigator  7. Tìm hiểu Webservice |  |  |  |
| 2 | Nguyễn Đức Lai | 1. Tạo Gmail,Gifhub  2. Tìm hiểu sự khác nhau Var & Let  3. Tìm hiểu width và height  4. Tìm hiểu listview  5. Tìm hiểu TextInput |  |  |  |
| 3 | Phan Văn Khải | 1. Tìm Hiểu Object  2. Tìm hiểu State  3. Tìm Hiểu TextInput , Tổng hợp báo cáo  4. Tìm hiểu TouchableOpacy  5. Tìm hiểu navigator , checkbox  6. Viết báo cáo |  |  |  |
| 4 | Bùi Đức Thảo | 1. Tìm hiểu Style  2. Tìm hiểu listview  3. Tìm hiểu TouchableOpacy  4. Viết báo cáo  5. Tìm hiểu checkbox |  |  |  |

# CHƯƠNG 1. TỔNG QUAN VỀ REACT NATIVE

## 1. Giới thiệu tổng quan về React native

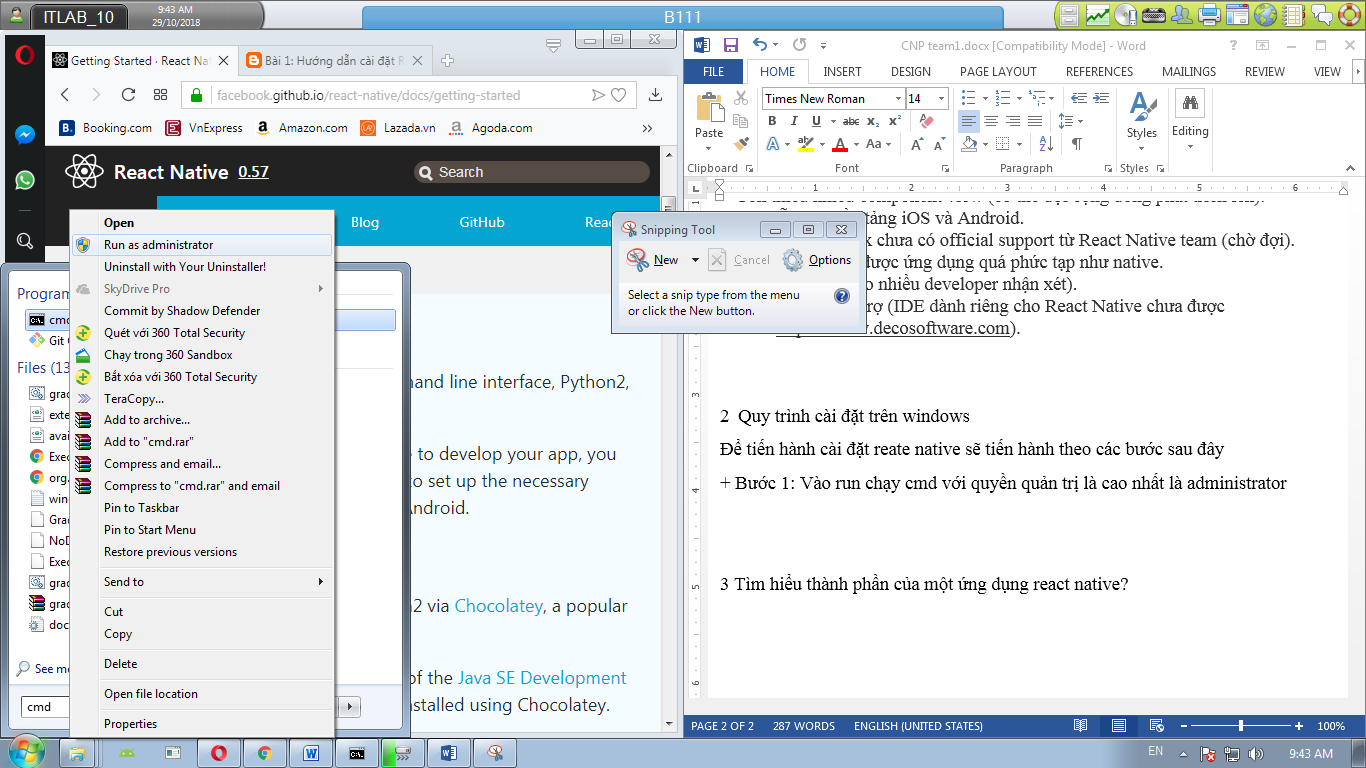
Ưu điểm ,nhược điểm

* Ưu điểm:
* Làm việc với vấn đề test giao diện: Nó cực kì dễ để viết các test case giao diện vì virtual DOM được cài đặt hoàn toàn bằng JS.
* Hiệu năng cao đối với các ứng dụng có dữ liệu thay đổi liên tục, dễ dàng cho bảo trì và sửa lỗi.
* Reactjs cực kì hiệu quả: Reactjs tạo ra cho chính nó DOM ảo – nơi mà các component thực sự tồn tại trên đó. Điều này sẽ giúp cải thiện hiệu suất rất nhiều. Reactjs cũng tính toán những thay đổi nào cần cập nhật len DOM và chỉ thực hiện chúng. Điều này giúp Reactjs tránh những thao tác cần trên DOM mà nhiều chi phí.
* Nó có nhiều công cụ phát triển: Khi bạn bắt đầu Reactjs, đừng quên cài đặt ứng dụng mở rộng của Chrome dành cho Reactjs. Nó giúp bạn debug code dễ dàng hơn. Sau khi bạn cài đặt ứng dụng này, bạn sẽ có cái nhìn trực tiếp vào virtual DOM như thể bạn đang xem cây DOM thông thường.
* Render tầng server: Một trong những vấn đề với các ứng dụng đơn trang là tối ưu SEO và thời gian tải trang. Nếu tất cả việc xây dựng và hiển thị trang đều thực hiện ở client, thì người dung sẽ phải chờ cho trang được khởi tạo và hiển thị lên. Điều này thực tế là chậm. Hoặc nếu giả sử người dung vô hiệu hóa Javascript thì sao? Reactjs là một thư viện component, nó có thể vừa render ở ngoài trình duyệt sử dụng DOM và cũng có thể render bằng các chuỗi HTML mà server trả về.
* Nhược Điểm:
* Reactjs chỉ phục vụ cho tầng View.  
  React chỉ là View Library nó không phải là một MVC framework như những framework khác. Đây chỉ là thư viện của Facebook giúp render ra phần view. Vì thế React sẽ không có phần Model và Controller, mà phải kết hợp với các thư viện khác. React cũng sẽ không có 2-way binding hay là Ajax
* Tích hợp Reactjs vào các framework MVC truyền thống yêu cầu cần phải cấu hình lại.
* React khá nặng nếu so với các framework khác  
  React có kích thước tương tương với Angular (Khoảng 35kb so với 39kb của Angular). Trong khi đó Angular là một framework hoàn chỉnh
* Khó tiếp cận cho người mới học Web

Quy trình cài đặt react native ( windows)(Nguyễn Đăng Khoa)

Để tiến hành cài đặt reate native sẽ tiến hành theo các bước sau đây

+ Bước 1: Vào run chạy cmd với quyền quản trị là cao nhất là administrator

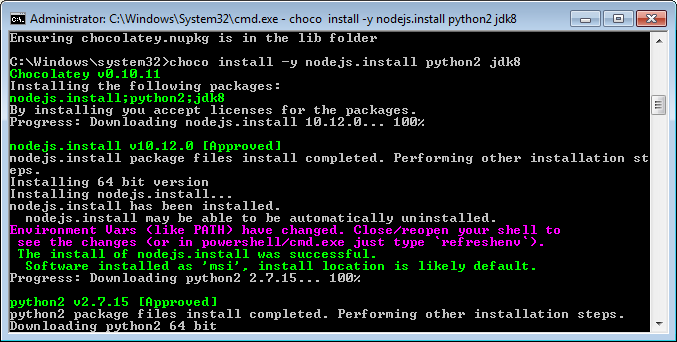


+ Bước 2: Tải và cài đặt các công cụ cần thiết các công cụ cần cài như sau: Nodejs, Python, android sdk. Chúng ta cài Chocolatery trên windows qua link sau:

<https://chocolatey.org/install>

Sau đó ta mở cửa sổ cmd ta đã mở từ trước và copy đoạn lệnh sau đây vào để hệ thống download và tiến hành cài đặt (Chocolatery, Node.js, python, JDK)

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"



- Sau khi cài đặt xong chocolatey ta đóng cửa sổ vừa chạy lệnh lại, sau đó mở CMD và chạy với quyền quản trị để tiến hành cài tải và cài đặt node, pyhon và jdk thông qua choco với câu lệnh

choco install -y nodejs.install python2 jdk8

+ Bước 3: Tiến hành cài React native

Chúng ta mở cmd và bỏ lệnh sau để tiến hành cài đặt cmd

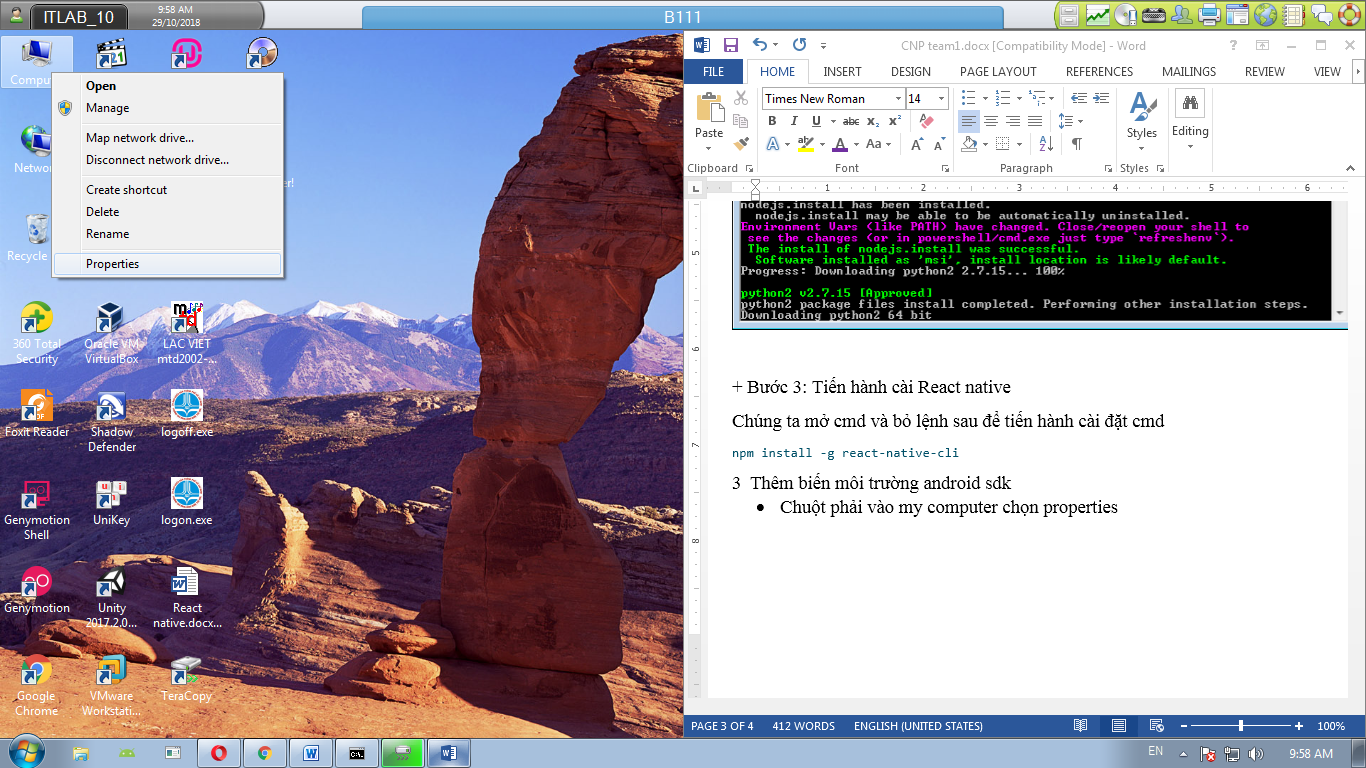
npm install -g react-native-cli

+ Bước 4: Thêm biến môi trường android sdk

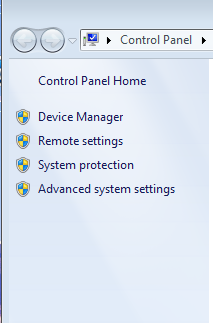
Ta cài đặt android sdk thông qua cài đặt android studio đây là bước bắt buộc.

Sau đó ta cấu hình ANDROID\_HOME cho windows qua các bước sau:

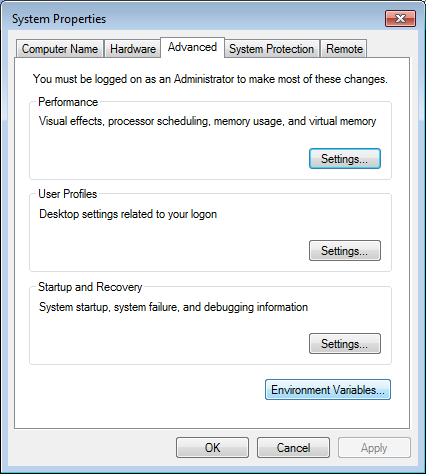
* Chuột phải vào my computer chọn properties



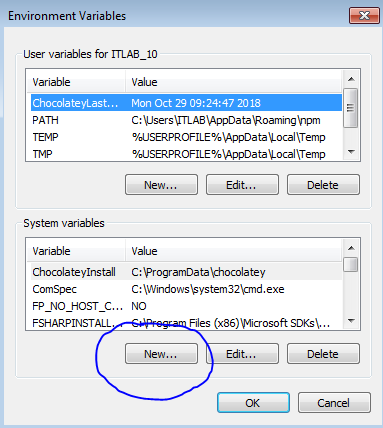
* Sau đó chọn advanced system settings ở bên trái



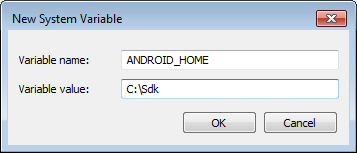
* Khi system properties hiện lên ta chọn vào Environment Variables



* Chọn new trong khung System variables



* Sau đó ta nhập Varible name là “ANDROID\_HOME” và ở Variable value chúng ta sẽ truyền đường dẫn chứa thư mục sdk



(Ở ví dụ trên sdk ở ổ C)

* Sau đó nhấn OK để hoàn tất tạo biến môi trường

Bước 5: Tạo một project react native

Mở cmd lên với quyền quản trị nhập lệnh sau:

react-native init Cool

// Cool đây là tên của project

Bước 6: Để chạy project

Ở đây chúng ta sẽ chạy project bằng máy ảo nên trước tiên chúng ta cần khởi chạy máy ảo lên (Ở đây dung genymotion).

Nhập lệnh vào cmd để chuyển tới project vừa tạo

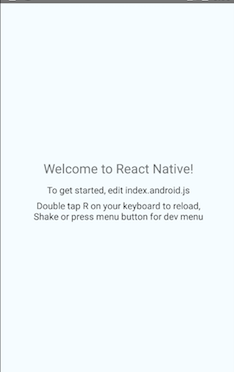
cd Cool

// Ở đây Cool là tên của project mà ta đã tạo ở trên

Để chạy project lên máy ảo ta nhập tiếp lệnh vào cmd

react-native run-android

Nếu thành công ta sẽ thấy giao diện của ứng dụng ở trên máy ảo



## 2. Kiến thức cơ bản về ES6 trong React native

Biến Let, Var, Const trong ES6 (Nguyễn Đức Lai)

* Biến let có phạm vi sử dụng trong { } ở ngoài không thể sử dụng biến let

Ví dụ:

function foo() {

let x = 10;

if (true) {

var x = 20; *// x ở đây cũng là x ở trên*

console.log(x); *// in ra 20*

}

console.log(x); *// vẫn là 10*

}

Biến let không tạo ra thuộc tính mới cho global scope(this) //global scope là không nằm trong function nào.

let y = 'global';

console.log(this.y); *// undefined*

* Biến var có phạm vi sử dụng trong 1 function chứa nó.

Ví dụ:

function foo() {

var x = 10;

if (true) {

var x = 20; *// x ở đây cũng là x ở trên*

console.log(x); *// in ra 20*

}

console.log(x); *// vẫn là 20*

}

Biến var tạo thuộc tính mới cho global scope (this)

var y = 'global';

console.log(this.y); *// global*

* Biến const sử dụng để khai báo 1 hằng số mà trong quá trình chạy chương trình giá trị không thay đổi.

Ví dụ:

*const pi = 3.14;*

*pi = 1; // Lỗi Uncaught TypeError: Assignment to constant variable*

Kiểu dữ liệu Object (Phan Văn Khải)

JavaScript hỗ trợ mở rộng các kiểu dữ liệu.

Một **đối tượng** là một cá thể chứa một tập hợp các cặp giá trị khóa. Không giống như các kiểu dữ liệu nguyên thủy, các đối tượng có thể biểu thị các giá trị nhiều hoặc phức tạp và có thể thay đổi theo thời gian của chúng. Các giá trị có thể là các giá trị vô hướng hoặc các hàm hoặc thậm chí là các mảng của các đối tượng khác.

Các biến thể cú pháp để xác định một đối tượng được thảo luận thêm.

Khởi tạo đối tượng

Giống như các kiểu nguyên thủy, các đối tượng có một cú pháp theo nghĩa đen: **xoăn bracesv** ({và}). Sau đây là cú pháp để xác định một đối tượng.

var identifier = {

Key1:value, Key2: function () {

//functions

},

Key3: [“content1”,” content2”]

}

Nội dung của một đối tượng được gọi là **các thuộc tính** (hoặc các thành viên), và các thuộc tính bao gồm một **tên** (hoặc khóa) và **giá trị** . Tên thuộc tính phải là chuỗi hoặc ký hiệu và giá trị có thể là bất kỳ loại nào (kể cả các đối tượng khác).

Giống như tất cả các biến JavaScript, cả tên đối tượng (có thể là biến bình thường) và tên thuộc tính phân biệt chữ hoa chữ thường. Bạn truy cập các thuộc tính của một đối tượng với một dấu chấm đơn giản.

Sau đây là cú pháp để truy cập Object Properties.

objectName.propertyName

Ví dụ: Trình khởi tạo đối tượng

var person = {

firstname:"Tom",

lastname:"Hanks",

func:function(){return "Hello!!"},

};

//access the object values

console.log(person.firstname)

console.log(person.lastname)

console.log(person.func())

Ví dụ trên, định nghĩa một đối tượng. Đối tượng có ba thuộc tính. Thuộc tính thứ ba đề cập đến một hàm.

Đầu ra sau được hiển thị khi thực hiện thành công mã trên.

Tom

Hanks

Hello!!

Trong ES6, gán giá trị thuộc tính khớp với tên thuộc tính, bạn có thể bỏ qua giá trị thuộc tính.

Thí dụ

var foo = 'bar'

var baz = { foo }

console.log(baz.foo)

Đoạn mã trên định nghĩa một đối tượng **baz** . Đối tượng có một thuộc tính **foo**. Giá trị thuộc tính được bỏ qua ở đây vì ES6 ngầm gán giá trị của biến foo cho khóa foo của đối tượng.

Sau đây là ES5 tương đương với mã trên.

var foo = 'bar'

var baz = { foo:foo }

console.log(baz.foo)

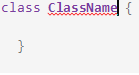
Đầu ra sau được hiển thị khi thực hiện thành công mã trên.

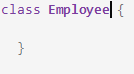
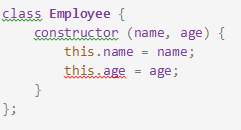
bar

Với cú pháp viết tắt này, công cụ JS nhìn vào phạm vi chứa cho một biến có cùng tên. Nếu nó được tìm thấy, giá trị của biến đó được gán cho thuộc tính. Nếu nó không được tìm thấy, một lỗi tham chiếu được ném.

Khai báo Class trong ES6 ( Nguyễn Đăng Khoa)

- Để tạo một Class được khai báo theo chuẩn ES6 thì sử dụng cú pháp như sau:



Trong đó ClassName là tên của đối tượng được khởi tạo.  
VD: Tạo đối tượng Employee:  
  
- Để khai báo constructor trong ES6 thìchỉ cần khai báo một phương thức có tên là constructor :  


- Cách khai báo getter và setter



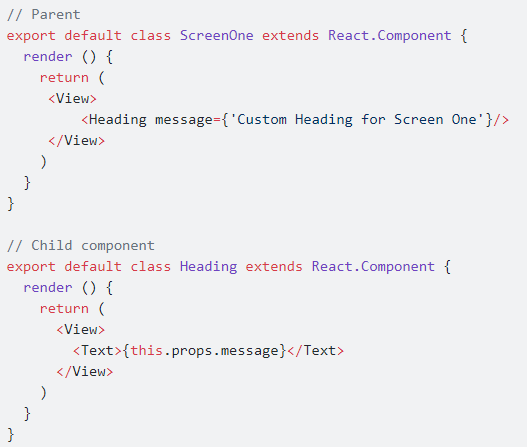
- Để kế thừa từ Class khác, chúng ta sử dụng từ khóa extends  


- Để thực hiện gọi các phương thức trong lớp cha khi đang ở lớp con, mà phương thức đó đã bị rewritetrong lớp con rồi thì các bạn sử dụng keyword super  


- Để khai báo các thành phần tĩnh cho Class, sử dụng từ khóa static ở trước tên phương thức.  


- Để khởi tạo Class được khai báo theo chuẩn ES6 thì các bạn sử dụng từ khóa new với cú pháp như sau:  

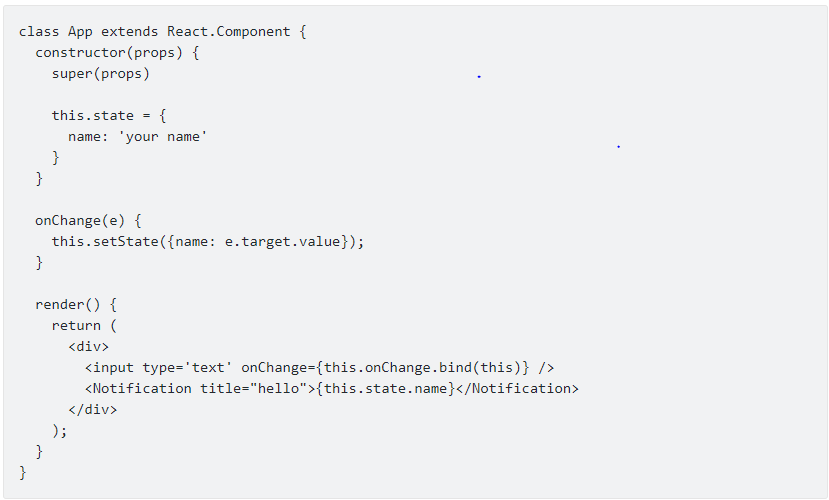

Props (Nguyễn Đăng Khoa)  
- Props  là viết tắt của Properties, ý tưởng của Props  đó là việc trừu tượng hoá các component để có thể sử dụng được ở nhiều chỗ khác nhau trong app.

VD:   


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



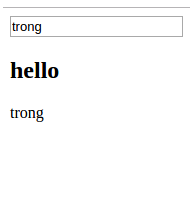
Ở đây chúng ta khởi tạo state với name: 'your name'

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

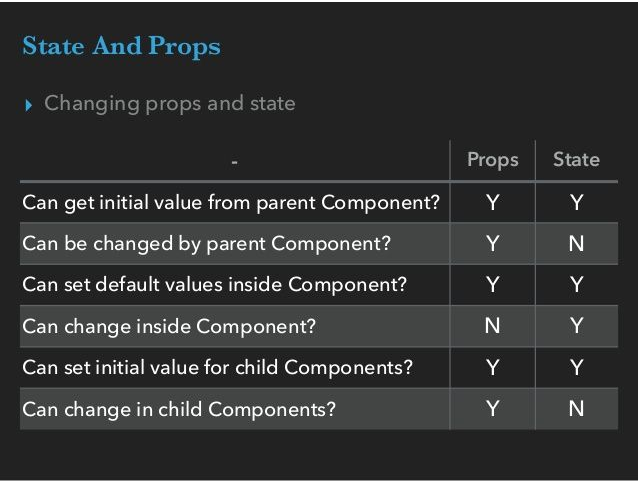
state.name được truyền vào component Notification thông qua props.children

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

Kết quả:



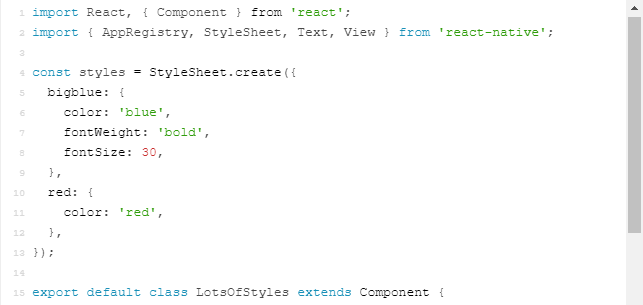
Điểm khác nhau giữa props và state:

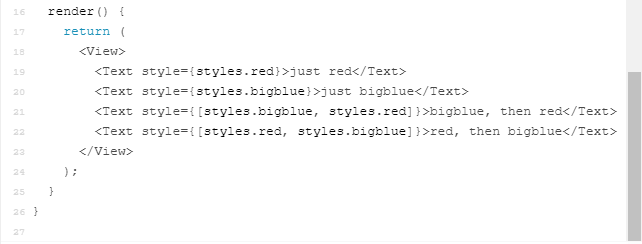


Style (Bùi Đức Thảo)  
Với React Native, bạn không sử dụng ngôn ngữ hoặc cú pháp đặc biệt để xác định kiểu. Bạn chỉ cần tạo kiểu cho ứng dụng của mình bằng JavaScript. Tất cả các thành phần cốt lõi chấp nhận một prop có tên là Style. Tên và giá trị kiểu thường khớp với cách CSS hoạt động trên web, ngoại 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:





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>

);

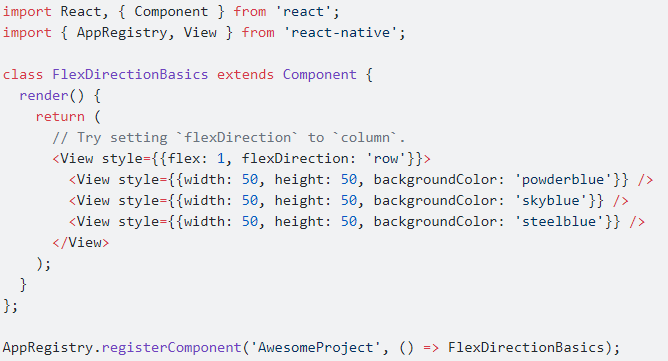
}

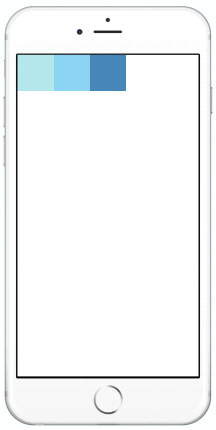
};

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

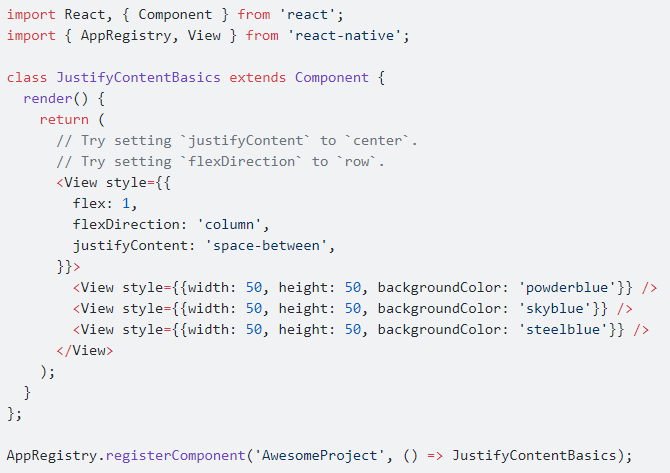
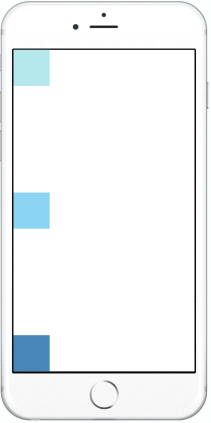
Flexbox with Layout (Nguyễn Đăng Khoa)

- 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.

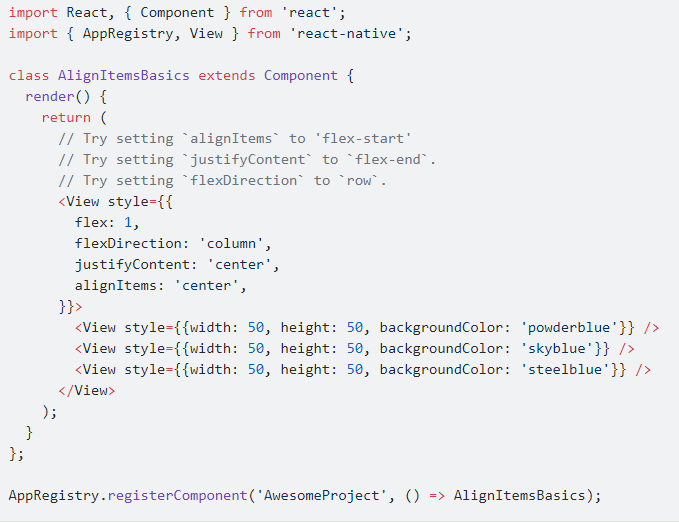
**\* Flex Decrection**  
- Thêm thuộc tính flexDecrection vào style của một thành phần để thay đổi hướng sắp xếp các thành phần con của nó. Những thành phần con nên được sắp xếp trên cùng một hàng (row) hoặc hoặc trên cùng một cột (column)? Mặc định thì các thành phần con sẽ được sắp xếp nằm trên cùng một cột (column).   
VD:  


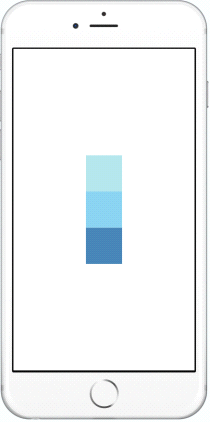
Kết Quả:  


**\*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:  


Tìm hiểu về Button

## Button

Tùy vào hệ điều hành dùng button này sẽ hiển thị với giao diện khác nhau.

VD: Sử dụng button

import { Button } from 'react-native';

...

<Button

onPress={onPressLearnMore}

title="Learn More"

color="#841584"

accessibilityLabel="Learn more about this purple button"

/>

**Các props (Các thuộc tính đó)**

- onPress

// Gọi phương thức khi người dùng chạm vào button (Yêu cầu)

- title

// Hiển thị tên nút (Yêu cầu)

- accessibilityLabel

// Văn bản hiện thị các trợ năng (Không yêu cầu)

- color

// Màu sắc của chử trong ios nhưng đây là màu nền của phím trong android

- disabled

// Tắt tương tác cho nút

- testID

// Được sử dụng để truy cập view này

- hasTVPreferredFocus

// Chỉ dành cho nền tảng ios

Tìm hiểu về Text(Đăng Khoa)

Là một thành phần để hiện thị text. Nó cũng xử lý lồng nhau, kiểu sự kiện touch handling.

VD

import React, { Component } from 'react';

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

export default class **TextInANest** extends **Component** {

constructor(props) {

super(props);

this.state = {

titleText: "Bird's Nest",

bodyText: 'This is not really a bird nest.'

};

}

**render**() {

return (

<**Text** *style*={styles.baseText}>

<**Text** *style*={styles.titleText} *onPress*={this.onPressTitle}>

{this.state.titleText}{'\n'}{'\n'}

</**Text**>

<**Text** *numberOfLines*={5}>

{this.state.bodyText}

</**Text**>

</**Text**>

);

}

}

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

baseText: {

fontFamily: 'Cochin',

},

titleText: {

fontSize: 20,

fontWeight: 'bold',

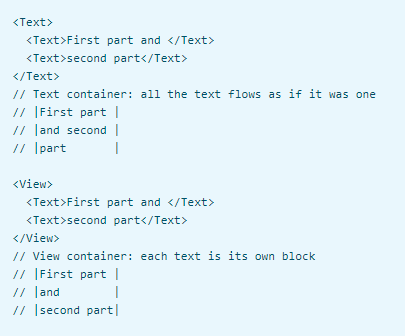
},

});

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

AppRegistry.**registerComponent**('TextInANest', () => TextInANest);

**Các thành phần trong Text**



**Các Props (Thuộc tính đó)**

- selectable

// Để cho phép người dùng chọn văn bản, để sử dụng chức năng sao chép và dán. (Không yêu cầu) giá trị là kiểu bool

- accessibilityHint (String)

// Gợi ý trợ năng giúp người dùng hiểu điều gỉ xảy ra khi họ thực hiện hành động trên phần tử (Không yêu cầu)

- accessibilityLabel (node)

// Ghi dè văn bản được đọc bởi trình đọc màn hình khi người dùng tương tác với phần tử. (Không yêu cầu)

- accessible (Bool)

Khi được đặt thành true cho biết chế độ xem là yếu tố trợ năng. Giá trị mặc định cho phần tử văn bản là đúng (Không yêu cầu)

- ellipsizeMode

- TYPE

- nativeID (String)

Được sử dụng để định vị chế độ xem này từ mã gốc (Không yeu cầu)

- numberOfLines (Số)

- onLayout

{nativeEvent: {layout: {x, y, width, height}}}

- onLongPress

// Gọi sự kiện khi nhấn giữ gọi một phương thức trong hàm hoặc một arrow function

- onPress

// Gọi sự kiện khi nhấn

VD

onPress={() => console.log('1st')}

- pressRetentionOffset

- allowFontScaling (bool)

- style (style)

// Định dạng các css trong đây đó

+ color

+fontStyle  enum('normal', 'italic')

+ fontWeight enum('normal', 'bold', '100', '200', '300', '400', '500', '600', '700', '800', '900')

+ lineHeight

+ textAlign enum('auto', 'left', 'right', 'center', 'justify')

+ textDecorationLine enum('none', 'underline', 'line-through', 'underline line-through')

+ fontFamily

+ letterSpacing (Số)

- testID (String)

// Sử dụng cho cục bộ muốn gọi view này

- disabled (Bool)

- selectionColor (color)

// Màu highlight của text

Tìm hiểu về TextInput(Đức Lai)

Dùng để nhập dữ liệu

VD

import React, { Component } from 'react';

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

export default class **UselessTextInput** extends **Component** {

constructor(props) {

super(props);

this.state = { text: 'Useless Placeholder' };

}

**render**() {

return (

<**TextInput**

*style*={{height: 40, borderColor: 'gray', borderWidth: 1}}

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

*value*={this.state.text}

/>

);

}

}

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

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

**Các Props**

- allowFontScaling (bool)

// Chỉ định xem font chữ có nên mở rộng hỗ trợ cài đặt trợ năng hay không. mặc định là true

- autoCapitalize

// Có thể yêu cầu text input tự động viết hoa các ký tự nhất định.

- autoCorrect (bool)

// Mặc định đã được bật

- autoFocus (bool)

- blurOnSubmit (bool)

- clearTextOnFocus (bool) (ios)

- TYPE (IOS)

- defautlValue (String)

- disableFullscreenUI (bool)

- keyboardType

// Quy định kiểu hiện thỉ của bàn phím

+ default

+ nuber-pad

+ decimal-pad

+ numeric

+ email-address

+ phone-pad

- maxLength (số)

- multiline (bool)

- numberOfLines

- onBlur (function)

- onChange (function)

// Gọi phương thức khi text bắt đầu thay đổi

- onChangeText (Function)

// Gọi lại khi văn bản đầu thay đổi

- onEndEditing (function)

// Dược gọi khi nhập xong

- onFocus (function)

// Gọi sự kiện chọn input

- onKeyPress (function)

- onScroll (Function)

- onSubmitEditing (Function)

- placeholder (String)

- placeholderTextColor (color)

- value (String)

- inlineImageLeft

Dùng để hiển thị hình ảnh bên trái text box

<TextInput

inlineImageLeft='search\_icon'

/>

**Các methods**

- clear();

- isFocused();

Tìm hiểu về checkbok(Đức Lai, Đức Thảo)

Đây là thành phần không có sẵn trong thư viện mặc định khi tạo ứng dụng mà ta phải import từ bên ngoài

import { CheckBox } from 'react-native-elements'

<CheckBox

title='Click Here'

checked={this.state.checked}

/>

<CheckBox

center

title='Click Here'

checked={this.state.checked}

/>

<CheckBox

center

title='Click Here'

checkedIcon='dot-circle-o'

uncheckedIcon='circle-o'

checked={this.state.checked}

/>

<CheckBox

center

title='Click Here to Remove This Item'

iconRight

iconType='material'

checkedIcon='clear'

uncheckedIcon='add'

checkedColor='red'

checked={this.state.checked}

/>

<CheckBox

checkedIcon={<Image source={**require**('../checked.png') />}

uncheckedIcon={<Image source={**require**('../unchecked.png') />}

checked={this.state.checked}

onPress={() => this.**setState**({checked: !this.state.checked})}

/>

**Cac props (Thuộc tính của thẻ)**

- iconType (String)

// Có thể thêm biểu tượng

- component

// Chỉ định thành phần react cho nút chính.

- checked (bool)

// Kiểm tra thẻ đã được check chưa

- size (Số)

// Kích thước của checkbox mặc định là 24

- iconRight (bool)

// Duy chuyển icon check sang bên phải của check mặc định giá trị này là false

- right (bool)

Căn chình checkbox sang bên phải

- center

// Căn chình checkbox sang giữa

- title (string)

// Đây là tiêu đề của textbox đây là thuộc tính yêu cầu

- containerStyle (object)

// Kiểu của thành phần chính

- textStyle

// Kiểu của chữ

- onLongPress (function)

Khi nhấn giữ vào check box

- onLongIconPress (function)

Khi nhấn giữ lâu vào check box

- onPress (function)

Khi nhấn vào check box

- onIconPress (function)

khi nhấn vào check box đây là thuộc tính yêu cầu

- checkedIcon

Icon mặc định khi check box chưa được nhấn chúng ta có thể dùng icon từ font awesome icon để thay thê và icon mặc định của giá trị này là check-square-o

- uncheckedIcon

Icon khi ta check vào nó mặc định là square-o

- checkedColor

Màu mặc định khi được check. Mặc định là màu green

- fontFamily

// Chỉ định font chữ

Tìm hiểu về radiobutton

Đây cũng là thành phần mà ta phải cài đặt từ bên ngoài

**Để thêm radio button ta phải chạy câu lệnh sau**

npm i radio-button-react-native --save

Sau đó import nó vào đồ án

**import** RadioButton **from** 'radio-button-react-native';

Cấu trúc của radio button sẽ giống như sau

render(){

**return**(

**<**View**>**

**<**RadioButton currentValue**=**{this.state.value} value**=**{0} onPress**=**{this.handleOnPress.bind(this)}**>**

**<**Text**>**Button1**</**Text**>**

**</**RadioButton**>**

**<**RadioButton currentValue**=**{this.state.value} value**=**{1} onPress**=**{this.handleOnPress.bind(this)}**>**

**<**Text**>**Button2**</**Text**>**

**</**RadioButton**>**

**<**RadioButton currentValue**=**{this.state.value} value**=**{2} onPress**=**{this.handleOnPress.bind(this)}**>**

**<**Text**>**Button3**</**Text**>**

**</**RadioButton**>**

**</**View**>**

    );

**Cơ bản của một Radio Button**

**<**RadioForm

  radio\_props**=**{radio\_props}

  initial**=**{0}

  onPress**=**{(value) => {this.setState({value**:**value})}}

**/>**

**Đầy đủ hơn thì**

**<**RadioForm

  radio\_props**=**{this.state.types}

  initial**=**{0}

  formHorizontal**=**{false}

  labelHorizontal**=**{true}

  buttonColor**=**{'#2196f3'}

  animation**=**{true}

  onPress**=**{(value) => {this.setState({value**:**value})}}

**/>**

**Các Props của RadioForm**

- radio\_props

// Giá trị của radio button này giá trị truyền dùng làm nhãn là một mãng.

- onPress (function)

Gọi phương thức này khi radio button được check.

- buttonColor

Màu sắc của radion button

**<**RadioForm

  radio\_props**=**{radio\_props}

  initial**=**{0}

  buttonColor**=**{'#50C900'}

**/>**



- labelColor

Màu sắc của nhãn của radio button

**<**RadioForm

  radio\_props**=**{radio\_props}

  initial**=**{0}

  labelColor**=**{'#50C900'}

**/>**

- formHorizontal (bool)

Giúp xác định chiều hiển thị của radio button mặc định giá trị này là false nên nó sẽ hiển thị ngang còn khi giá trị là true thì nó sẽ hiển thị dọc

**<**RadioForm

  radio\_props**=**{radio\_props}

  initial**=**{0}

  labelHorizontal**=**{false}

**/>**





- animation

Đương nhiên đây là hiệu ứng khi ta click vào radion button rồi mặc định giá trị này là true

- accessible

Cho biết khả năng truy cập riêng lẽ cho từng thành phần nút radio và nút radio riêng lẽ

- accessibilityLabel

- testID

Được sử dụng để thiết lập testId cho đầu vào nút radio và thành nhấn của nó

= [testID]Label[index].

Tìm hiểu về list view (Đăng Khoa)

ListView là một thành phần cốt lõi được thiết kế để hiện thị hiệu quả đối với dữ liệu cần hiển thị dạng danh sách các dữ liệu sẽ được hiển thị theo chiều dọc.

Có hai thuộc tính quan trong list view đó là dataSource và renderRow. Datasource là dữ liệu sẽ được hiển thị lên. RenderRow sẽ trả về giao diện giữa các list item.

VD

class **MyComponent** extends **Component** {

constructor() {

super();

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

this.state = {

*// Tạo một đối tượng data source với dữ liệu là một mảng*

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

};

}

**render**() {

return (

<**ListView**

*// Set nguồn dữ liệu của list view*

*dataSource*={this.state.dataSource}

*// Lấy từng dữ liệu từ mảng hiển thị lên mỗi dòng sẽ là một thẻ text*

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

/>

);

}

}

**Các props (Các thuộc tính trong thẻ list view)**

- dataSource

Đây là thuộc tính bắt buộc với giá trị là ListViewDataSource

var ds = new ListView.DataSource({rowHasChanged: this.\_rowHasChanged});

- initialListSize (Số)

Quy định số hàng hiển thị trên màn hình lúc đầu.

- onEndReachedThreshold (Số)

Quy định số pixel hiển thị trên thiết bị vật lý

- pageSize (Số)

Số dòng sẽ được render cho mỗi lần loop (Mỗi lần kéo). Đây là thuộc tính yêu cầu

- renderRow (Function)

(rowData, sectionID, rowID, highlightRow) => renderable

Lấy một mục dữ liệu từ nguồn và các id của nó và sẽ trả về một thành phần có thể hiển thị dưới dạng hàng. Theo mặc định dữ liệu chính là những gì được đưa vào nguồn dữ liệu. List view có thể đánh dấu hàng được chọn bằng highlighRow(sectionID, rowID).

Những thiết lập giá trị boolean này của adjacentRowHighlighted in renderSeparator cho phép bạn điều khiển các dòng riêng biệt trên và dưới highlightedrow.

- renderScrollComponent (function)

(props)=> renderable

Một phương thức trả về thành phần được cuộn trong danh sách các hàng đã được render. Mặc định để để trả về một ScrollView với giá trị được gắn vô props.

- scrollRenderAheadDistance (Số)

Hiển thị sớm các hàng trên màn hình trước khi nó sẵn sàng hiện dữ liệu

- stickyHeaderIndices

**Các methods**

- getMetrics()

Mở rộng một vài dữ liệu.

- scrollTo()

scrollTo(...args: Array)

Cuộn tới x, y với hiệu ứng

- scrollToEnd()

scrollToEnd(([options]: object));

- flashScrollIndicators()

flashScrollIndicators();

**2 Tìm hiểu về navigation (Đăng Khoa , Văn Khải)**

Naviagation đây là thành phần giúp chúng ta chuyển đổi giữa các màn hình cũng là thành phần giúp chúng ta chuyển đổi dữ liệu giữa các màn hình với nhau.

**Để sử dụng navigation chúng ta phải trải qua các bước sau:**

- Bước 1: Cài gói navigation vào project thông qua lệnh cmd trỏ vào project

npm install --save react-navigation

npm install --save react-native-gesture-handler

react-native link react-native-gesture-handler

Bước 2: Tiến hành tạo stack navigator có bao nhiêu màn hình thì chúng ta đều khai báo tại đây

VD

import React from 'react';

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

import { createAppContainer, createStackNavigator, StackActions, NavigationActions } from 'react-navigation'; *// Version can be specified in package.json*

class **HomeScreen** extends **React**.**Component** {

**render**() {

return (

<**View** *style*={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<**Text**>Home Screen</**Text**>

<**Button**

*title*="Go to Details"

*onPress*={() => {

this.props.navigation.**dispatch**(StackActions.**reset**({

index: 0,

actions: [

NavigationActions.**navigate**({ routeName: 'Details' })

],

}))

}}

/>

</**View**>

);

}

}

class **DetailsScreen** extends **React**.**Component** {

**render**() {

return (

<**View** *style*={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>

<**Text**>Details Screen</**Text**>

</**View**>

);

}

}

const AppNavigator = **createStackNavigator**({

Home: {

screen: HomeScreen,

},

Details: {

screen: DetailsScreen,

},

}, {

initialRouteName: 'Home',

});

<https://reactnavigation.org/docs/en/hello-react-navigation.html>

Bước 3: Sau khi khai báo đây là cách di chuyển giữa các màn hình

- Để di chuyển sang màn hình khác tại sự kiện của component nào đó ta gọi cú pháp như sau:

this.props.navigation.**navigate**('TenManHinhDaDinhNghia')

// Nếu muốn load lại màn hình sau khi đã chuyển đến màn hình đó rồi thì theo cú pháp sau:

his.props.navigation.**push**('ManHinhMuonKhoiTaoLai')

// Trường hợp muốn back lại trường hợp trước đó

this.props.navigation.**goBack**()

// Trường hợp muốn chuyển dữ liệu giữa các màn hình

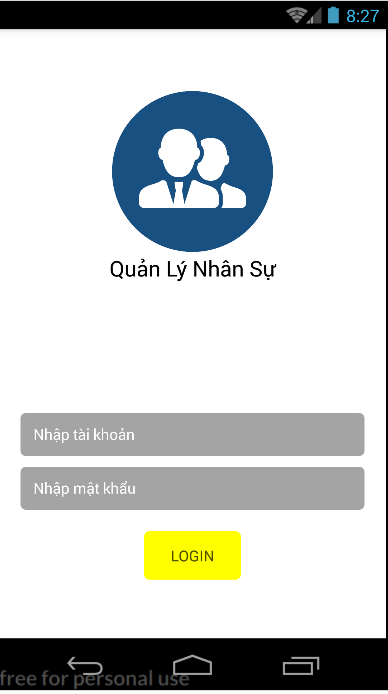
this.props.navigation.**navigate**('TenManHinhDaDinhNghia', {key: value})

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1 Phân tích hệ thống

### 2.1.1. Feature/Component #1: Màn hình Login( Phan Văn Khải)

#### 2.1.1.1 User Interfaces



*Image1: Login*

#### 2.1.1.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Login screen** | Màn hình Login có 5 thành phần:   * Image * Label * TextInput Tài Khoản * TextInput Mật Khẩu * Button |  |  |
| **Image** |  |  |  |
| **Label** |  |  |  |
| **TextInput Nhập Tài Khoản** | Cho người dùng nhập tài khoản | Bấm TextInput | Người dùng có hể nhập tài khoản |
| **TextInput Nhập Mật Khẩu** | Cho người dùng nhập tài mật khẩu | Bấm TextInput | Người dùng có thể nhập mật khẩu |
| **Button Login** | Dùng để đăng nhập vào ứng dụng | Bấm nút | Đăng nhập vào trang chủ |

### 2.1.2. Feature/Component #2: Màn hình Trang Chủ ( Nguyễn Đăng Khoa)

#### 2.1.2.1 User Interfaces



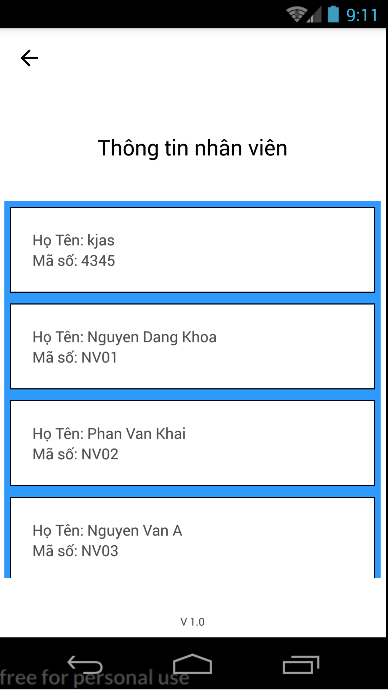
*Image2: Trang Chủ*

#### 2.1.2.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình trang chủ** | Màn hình Trang chủ có 3 thành phần:   * Label * Images * Text |  |  |
| **Image Danh sách nhân viên** | Chuyển đến màn hình Danh sách nhân viên | Bấm vào Image | Hiển thị màn hình Danh sách nhân viên |
| **Image Thêm nhân viên** | Chuyển đến màn hình Thêm nhân viên | Bấm vào Image | Hiển thị màn hình Thêm nhân viên |
| **Image Tìm kiếm nhân viên** | Chuyển đến màn hình Tìm kiếm nhân viên | Bấm vào Image | Hiển thị màn hình Tìm kiếm nhân viên |
| **Image Thoát** | Thoát về màn hình Login | Bấm vào Image | Hiển thị màn hình Login |
| **Text Danh sách nhân viên** | Chuyển đến màn hình Danh sách nhân viên | Bấm vào Text | Hiển thị màn hình Danh sách nhân viên |
| **Text Thêm nhân viên** | Chuyển đến màn hình Thêm nhân viên | Bấm vào Text | Hiển thị màn hình Thêm nhân viên |
| **Text Tìm kiếm nhân viên** | Chuyển đến màn hình Tìm kiếm nhân viên | Bấm vào Text | Hiển thị màn hình Tìm kiếm nhân viên |
| **Text Thoát** | Thoát về màn hình Login | Bấm vào Text | Hiển thị màn hình Login |
| **Label** |  |  |  |

### 2.1.3. Feature/Component #3: Màn hình Danh sách nhân viên ( Nguyễn Đức Lai)

#### 2.1.3.1 User Interfaces



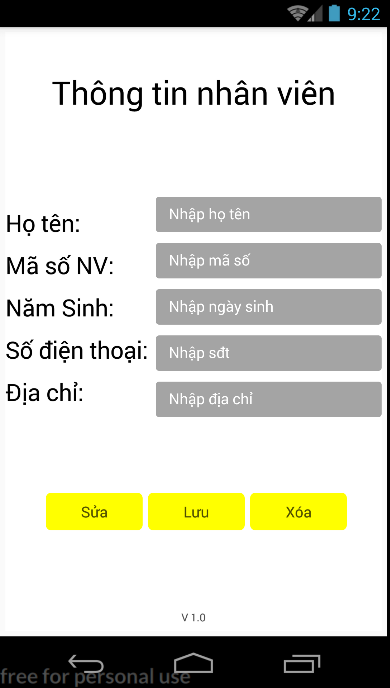
*Image3: Danh Sách Nhân Viên*

#### 2.1.3.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình Danh sách nhân viên** | Màn hình Trang chủ có 4 thành phần:   * Label * ListView * Text * ItemListView |  |  |
| **Label** |  |  |  |
| **ListView** | Hiển thị danh sách các nhân viên | Khi đi đến màn hình Danh sách nhân viên | Danh sách các nhân viên hiển thị trên list view |
| **Text Họ tên** | Hiển thị họ tên nhân viên | Khi đi đến màn hình Danh sách nhân viên | Họ tên chủa nhân viên được hiển thị |
| **Text Mã số** | Hiển thị mã số nhân viên | Khi đi đến màn hình Danh sách nhân viên | Mã số của nhân viên được hiển thị |
| **ItemListView** | Hiển thị nhân vien , dùng để chuyển tới màn hình xóa sửa | Bấm vào Item | Hiển thị màn hình xóa sửa |

### 2.1.4. Feature/Component #4: Màn hình Xóa sửa nhân viên ( Nguyễn Đức Lai)

#### 2.1.4.1 User Interfaces



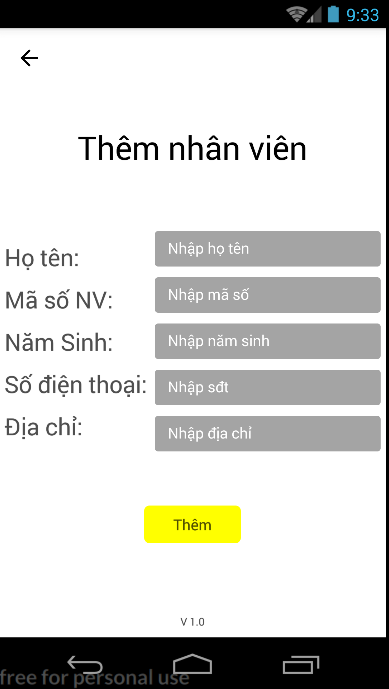
*Image4: Xóa Sửa Nhân Viên*

#### 2.1.4.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình Xóa sửa** | Màn hình Trang chủ có 4 thành phần:   * Label * TextInput * Text * Button |  |  |
| **Label** |  |  |  |
| **Text Năm sinh** |  |  |  |
| **Text Họ tên** |  |  |  |
| **Text Mã số** |  |  |  |
| **Text Số điện thoại** |  |  |  |
| **Text Địa chỉ** |  |  |  |
| **TextInput Họ tên** | Hiển thị họ tên nhân viên | Khi đến màn hình Xóa Sửa | Họ tên nhân viên được hiển thị trên TextInput này |
| **TextInput Mã số** | Hiển thị mã số nhân viên | Khi đến màn hình Xóa Sửa | Mã số nhân viên được hiển thị trên TextInput này |
| **TextInput Năm sinh** | Hiển thị năm sinh nhân viên | Khi đến màn hình Xóa Sửa | Năm sinh nhân viên được hiển thị trên TextInput này |
| **TextInput Số điện thoại** | Hiển thị số điện thoại nhân viên | Khi đến màn hình Xóa Sửa | Số điện thoại nhân viên được hiển thị trên TextInput này |
| **TextInput Địa chỉ** | Hiển thị địa chỉ nhân viên | Khi đến màn hình Xóa Sửa | Địa chỉ nhân viên được hiển thị trên TextInput này |
| **Button Sửa** | Sửa Thông tin nhân viên | Bấm vào Button | Sửa thông tin nhân viên |
| **Button Lưu** | Lưu Thông tin nhân viên | Bấm vào Button | Lưu thông tin đã sửa lên DataBase |
| **Button Xóa** | Xóa nhân viên | Bấm vào Button | Xóa nhân viên trên DataBase |

### 2.1.4. Feature/Component #5: Màn hình Xóa sửa nhân viên ( Nguyễn Đăng Khoa)

#### 2.1.4.1 User Interfaces



*Image5: Thêm Nhân Viên*

#### 2.1.4.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình Xóa sửa** | Màn hình Trang chủ có 4 thành phần:   * Label * TextInput * Text * Button |  |  |
| **Label** |  |  |  |
| **Text Năm sinh** |  |  |  |
| **Text Họ tên** |  |  |  |
| **Text Mã số** |  |  |  |
| **Text Số điện thoại** |  |  |  |
| **Text Địa chỉ** |  |  |  |
| **TextInput Họ tên** | Nhập họ tên nhân viên | Bấm vào TextInput | Nhập họ tên nhân viên |
| **TextInput Mã số** | Nhập mã số nhân viên | Bấm vào TextInput | Nhập họ tên mã số |
| **TextInput Năm sinh** | Nhập năm sinh nhân viên | Bấm vào TextInput | Nhập họ tên năm sinh |
| **TextInput Số điện thoại** | Nhập số điện thoại nhân viên | Bấm vào TextInput | Nhập họ tên số điện thoại |
| **TextInput Địa chỉ** | Nhập địa chỉ nhân viên | Bấm vào TextInput | Nhập họ tên địa chỉ |
| **Button Thêm** | Thêm nhân viên | Bấm vào Button | Nhân viên được thêm trên DataBase |

### 2.1.5. Feature/Component #5: Màn hình Danh sách nhân viên ( Bùi Đức Thảo)

#### 2.1.5.1 User Interfaces



*Imag5: Tìm kiếm Nhân Viên*

#### 2.1.5.2. Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình Danh sách nhân viên** | Màn hình Trang chủ có 4 thành phần:   * Label * Button * ListView * Text * TextInput |  |  |
| **Label** |  |  |  |
| **ListView** | Hiển thị danh sách các nhân viên vào danh sách các nhân viên được tìm kiếm | Khi đi đến màn hình Tìm kiếm nhân viên và tìm nhân viên | Danh sách các nhân viên được hiển thị trên list view |
| **Text Mã số nhân viên** |  | Khi đi đến màn hình Danh sách nhân viên | Họ tên chủa nhân viên được hiển thị |
| **TextInput Nhập mã** | Nhấn vào nhập mã nhân viên | Bấm vào textinput | Nhập mã số nhân viên cần tìm |
| **Button Tìm kiếm** | Tìm kiếm nhân viên | Bấm vào Button | Tìm kiếm nhân viên |