# BÁO CÁO ĐẦU MÔN HỌC

* Bộ môn: Lập trình di động 3
* Tên nhóm: Nhóm 1
* Thành viên:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ Và Tên | Email | SDT |
| 1 | Lê Đình Cường | [ledinhcuongse@gmail.com](mailto:ledinhcuongse@gmail.com) | 0359752007 |
| 2 | Đoàn Thị Yến Nhi | [Doanthiyennhi1001@gmail.com](mailto:Doanthiyennhi1001@gmail.com) | 0388102525 |
| 3 | Trần Đức Phương | [phuongtranitnl@gmail.com](mailto:phuongtranitnl@gmail.com) | 0937320311 |
| 4 | Bùi Đức Cơ | [Rongtrang137@gmail.com](mailto:Rongtrang137@gmail.com) | 01889361179 |

* Email nhóm: [ncpdidong3@gmail.com](mailto:ncpdidong3@gmail.com)
* Pass: Aa359752007
* Github:
* Đề tài: Xây dựng ứng dụng lưu ghi chú bằng React Native

Task tuần 1: Làm quen với React Native và thiết lập môi trường lập trình

## 1. Tổng quan về react native?

React native là:

+ là công nghệ được tạo bởi Facebook,

+ tạo các ứng dụng di động chỉ với Javascript và React. React Native có kiến trúc thiết kế tương tự như React, cho phép bạn thiết kế, tái sử dụng các UI phong phú từ các component mặc định và các component đến từ đóng góp của người dùng.

+ Là nền tảng để phát triển ứng dụng có thể chạy trên hai nền tảng android và ios

Ưu điểm:

* Học một lần sử dụng nhiều nơi.
* Build nhanh chóng.
* Khả năng tái sử dụng code cao.
* Mang lại trải nghiệm giống như native.

Nhược điểm:

+ Còn thiếu nhiều component view (có thể đợi cộng đồng phát triển lên).

+ Chỉ hỗ trợ 2 nền tảng iOS và Android.

+ Windows và linux chưa có official support từ React Native team (chờ đợi).

+ Không thể build được ứng dụng quá phức tạp như native.

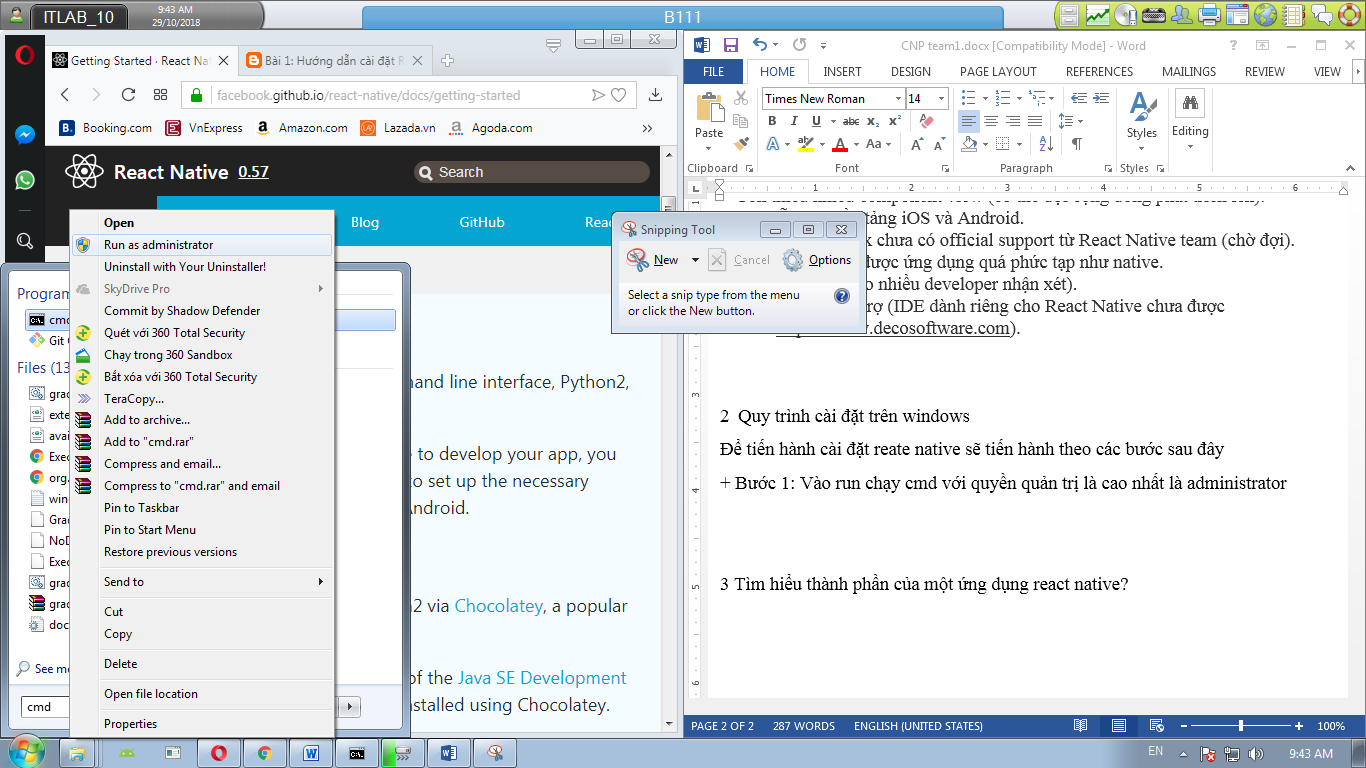
+ Khó tiếp cận (theo nhiều developer nhận xét).

+ Chưa có IDE hỗ trợ (IDE dành riêng cho React Native chưa được release https://www.decosoftware.com).

## 2. Quy trình cài đặt trên windows

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

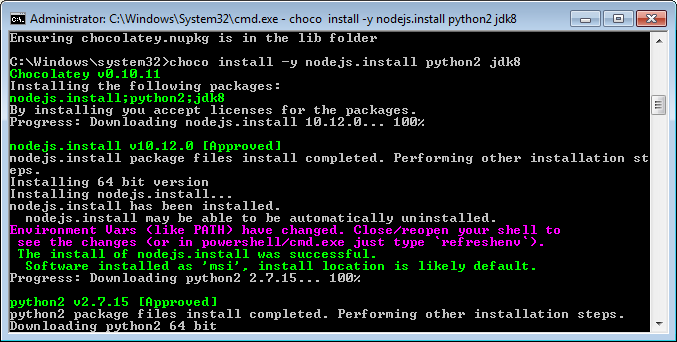


+ Bước 2: Tải và cài đặt các công cụ cần thiết các công cụ cần cài như sau: Nodejs, Python, android sdk. Chúng ta cài Chocolatery trên windows qua link sau:

<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

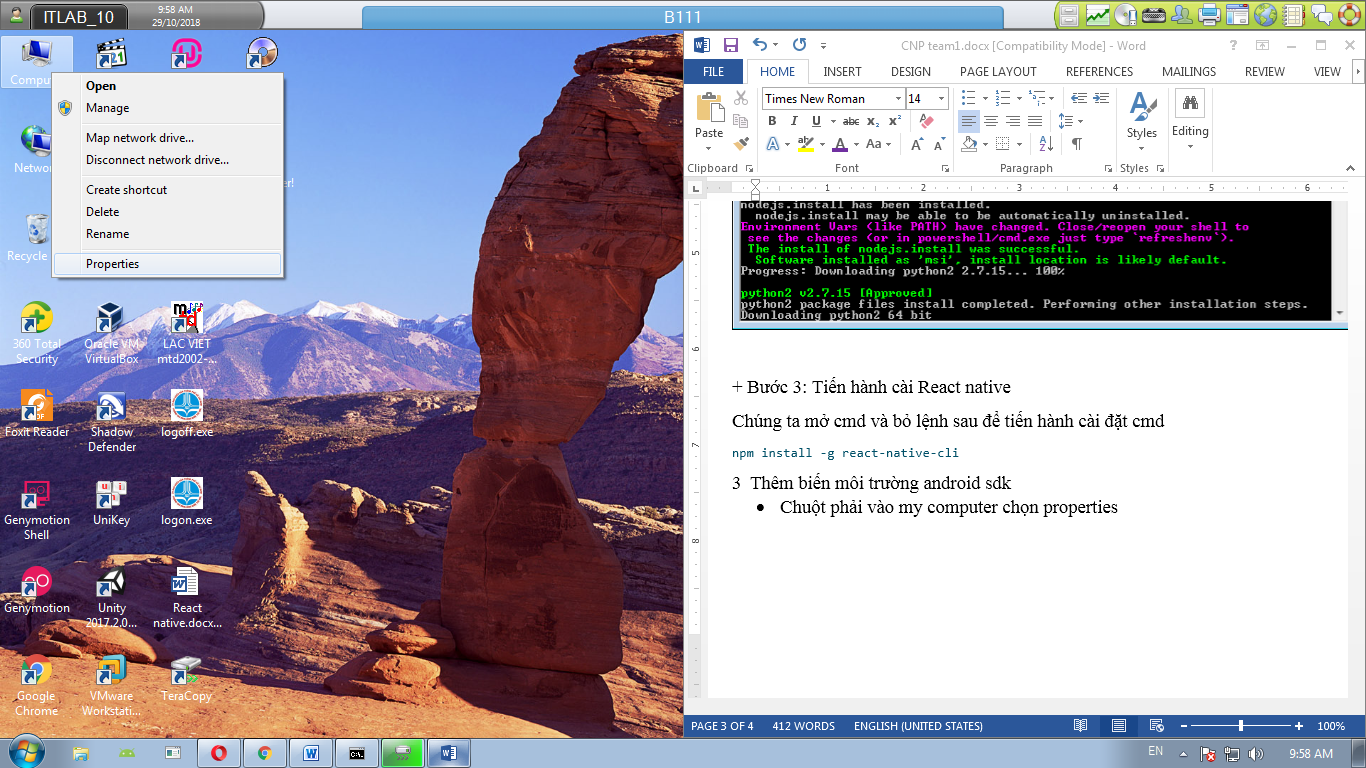
npm install -g react-native-cli

+ Bước 4: Thêm biến môi trường android sdk

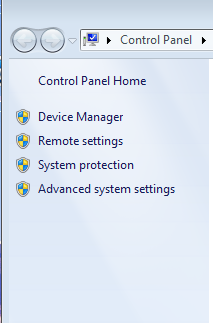
Ta cài đặt android sdk thông qua cài đặt android studio đây là bước bắt buộc.

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

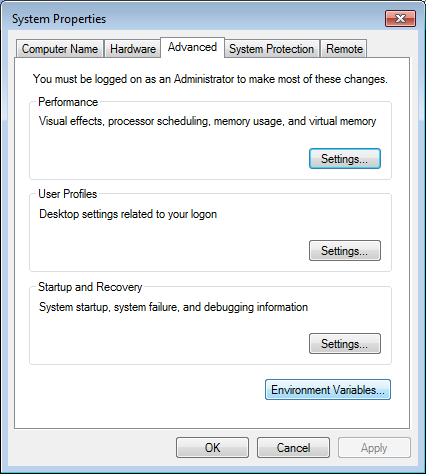
* Chuột phải vào my computer chọn properties



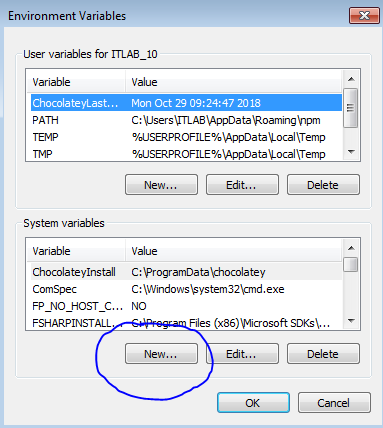
* Sau đó chọn advanced system settings ở bên trái



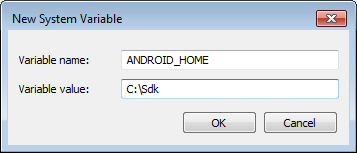
* Khi system properties hiện lên ta chọn vào Environment Variables



* 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 AwesomeProject

// AwesomeProject 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

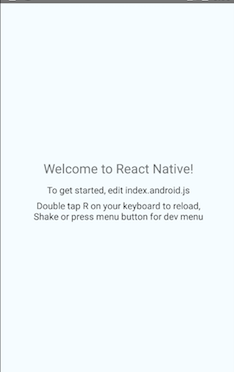
cd AwesomeProject

// Ở đây AwesomeProject 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



## 3. Tìm hiểu thành phần của một ứng dụng react native?

Chúng ta sẽ phân tích file App.js được tạo ra khi chúng ta tiến hành tạo đồ án react native

*/\*\**

*\* Sample React Native App*

*\* https://github.com/facebook/react-native*

*\**

*\* @format*

*\* @flow*

*\*/*

import React, {Component} from 'react';

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

const instructions = Platform.**select**({

ios: 'Press Cmd+R to reload,\n' + 'Cmd+D or shake for dev menu',

android:

'Double tap R on your keyboard to reload,\n' +

'Shake or press menu button for dev menu',

});

type **Props** = {};

export default class **App** extends **Component**<**Props**> {

**render**() {

return (

<**View** *style*={styles.container}>

<**Text** *style*={styles.welcome}>Welcome to React Native!</**Text**>

<**Text** *style*={styles.instructions}>To get started, edit App.js</**Text**>

<**Text** *style*={styles.instructions}>{instructions}</**Text**>

</**View**>

);

}

}

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

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: '#F5FCFF',

},

welcome: {

fontSize: 20,

textAlign: 'center',

margin: 10,

},

instructions: {

textAlign: 'center',

color: '#333333',

marginBottom: 5,

},

});

Phần 1: Đấy là phần chúng ta import các thành phần mà chúng ta sử dụng và các file tài nguyên và các components (Text, InputText, Button,…).

import React, {Component} from 'react';

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

Phần 2: Đây là phần class gọi giao diện sẽ có một phương thức render() đây là phương thức load giao diện trong nó lên màn hình.

export default class **App** extends **Component**<**Props**> {

**render**() {

return (

<**View** *style*={styles.container}>

<**Text** *style*={styles.welcome}>Welcome to React Native!</**Text**>

<**Text** *style*={styles.instructions}>To get started, edit App.js</**Text**>

<**Text** *style*={styles.instructions}>{instructions}</**Text**>

</**View**>

);

// export default thường đặt cho class mà sẽ được gọi trong chương trình

Đây là những thẻ giao diện để chương trình hiển thị mỗi sẽ có các thuộc tính style và props khác nhau.

<**View** *style*={styles.container}>

<**Text** *style*={styles.welcome}>Welcome to React Native!</**Text**>

<**Text** *style*={styles.instructions}>To get started, edit App.js</**Text**>

<**Text** *style*={styles.instructions}>{instructions}</**Text**>

</**View**>

Dưới đây là phần khai báo các style sheet các thành phần trong nó sẽ được gọi tại các thẻ View

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

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: '#F5FCFF',

},

welcome: {

fontSize: 20,

textAlign: 'center',

margin: 10,

},

instructions: {

textAlign: 'center',

color: '#333333',

marginBottom: 5,

},

});

Tổng quát các thành phần chính của react native

1 Props

Ví dụ một thành phần react native cơ bản là Image. Khi chúng ta tạo một hình ảnh chúng ta có thể sử dụng một nguồn được đặt tên để kiểm soát hình ảnh mà nó hiển thị.



import React, { Component } from 'react';

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

export default 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}}/>

);

}

}

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

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

// {pic} được bao quanh bởi các dấu ngoặc để nhúng các biến ảnh vào jsx. Chúng ta có thể đặt bất kỳ biểu thức JavaScript vào bên trong dấu ngoặc ôm trong jsx.

Các thành phần của riêng bạn cũng có thể sử dụng như đạo cụ (Props). Điều này cho phép chúng ta tạo một thành phần duy nhất được sử dụng ở nhiều vị trí khác nhau trong ứng dụng của bạn với các thuộc tính khác nhau ở mỗi vị trí.

Để gọi chỉ cần this.props trong phương thức render.

**Ví dụ dưới đây sẽ làm rõ “Props” hơn.**

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**>

);

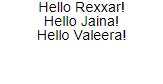
}

}

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

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

// Kết quả



// 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.

Vói props và các thành phần Text, Image, View chúng ta có thể xây dựng một loại màn hình tĩnh khác nhau.

**2 State**

Có hai loại dữ liệu kiểm soát một thành phần đó là Props và State.

Props được thiết lập bởi parent và chúng được cố định trong suốt vòng đời của một thành phần.

State sử dụng đối với dữ liệu sẽ thay đổi.

Nói chung, chúng ta nên khởi tạo state trong hàm khởi tạo và sau đó setState khi ta muốn thay đổi nó.

VD

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**>

);

}

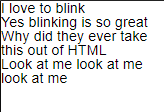
}

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

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

// Kết quả

Màn hình

Màn hình sẽ liên tục nhấp nháy dòng chứ này

Trong thực tế ứng dụng, chúng ta không cần thiết lập trạng thái với thời gian. Mà chúng ta sẽ thiết lập state khi chúng ta có dữ liệu mới xuất hiện từ server hoặc từ dữ liệu nhập vào từ người dùng. Chúng ta có thể sử dụng một state container giống như một Redux hoặc Mobx để điều khiển dòng dữ liệu. Trong trường hợp bạn sử dụng Redux hoặc Mobx để định nghĩa state của bạn chứ không phải gọi setState trực tiếp.

Khi setState được gọi thì BlinkApp sẽ re-render thành phần của nó. Bằng cách gọi setState mà không thời gian, thành phần sẽ re-render mỗi time được kích hoạt.

3 Style

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: Sử dụng style

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);

