React Native

1. Môi trường làm việc

* Cài chocolatey:

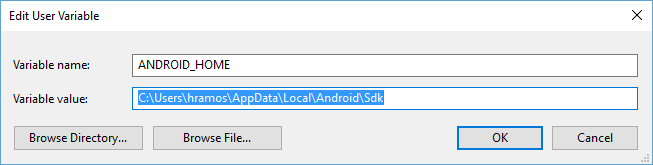
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

* choco install -y nodejs.install python2 jdk8

cài đặt expo

* npm install exp –global.
* Cài đặt Yarn
* npm install -g yarn
* tạo app qua lệnh
* npm install -g create-react-native-app
* xem đường dẫn của lệnh
* yarn add --dev babel-cli babel-preset-flow
* [Download Android Studio](https://developer.android.com/studio/index.html).
* Custom:
* Android SDK
* Android SDK Platform
* Performance (Intel ® HAXM)
* Android Virtual Device (phần này có thể thay thế bởi Genymotion: https://www.genymotion.com/)

Chọn cài đặt bản Android Sdk 9 (Pie), khi bật Android Sdk Manage hãy chọn tối thiểu 2 gói sau

* Android SDK Platform 28
* Intel x86 Atom\_64 System Image or Google APIs Intel x86 Atom System Image
* Cấu hình ANDROID\_HOME environment variable
* Các công cụ React Native yêu cầu một số biến môi trường được thiết lập để xây dựng các ứng dụng với Native Code.
* 
* *Cài đặt biến môi trường ANDROID\_HOME trên windows*
* Thay thế giá trị trên bằng SDK của bạn, Sdk được cài đặt trong thư mục sau
* C:\Users\YOUR\_USERNAME\AppData\Local\Android\Sdk
* Bạn có thể tìm vị trí thực tế của SDK trong Android Studio bằng cách: click vào “Preferences” và chọn menu *Appearance & Behavior → System Settings → Android SDK.* Sau đó copy và dán vào ô bên trên
* Thêm platform-tools vào Path
* Hãy vào phần biến môi trường của Windows như bước trên, và chọn biến **Path** , sau đó click **Edit** và chọn **New** để nhập thêm platform-tools vào biến Path. Đường dẫn của thư mục platform-tools có cấu trúc như sau
* C:\Users\YOUR\_USERNAME\AppData\Local\Android\Sdk\platform-tools

exp init MemoryGame

exp start

//function viết theo es6 hàm thực hiện nhiều code thì {} và phải có return

let xinchao =(a,b)=>{

var c=5;

var tong=0;

tong=a+b\*c;

return tong ;

}

//function viết theo es6 hàm không có giá trị trả về ruturn thì

let xinchao =(a,b)=>{

console.log(“xin chao”)

}

xinchao();

// hàm 1 tham số

let xinchao =a=>a

console.log(xinchao(“giatrị”))

//sử dụng map duyệt từng phần tử

let arrays=["a","b","c"];

arrays.map(function(value){

console.log(value);

})

//sử dụng hàm es6

let arrays=["a","b","c"];

arrays.map((value)=>console.log(value))

//

bmi=(chieucao,cannang)=>{

bmi\_=0;

bmi\_=cannang/chieucao\*cannang; //không sử dụng được this

return bmi\_;

}

console.log(bmi(162,60))

//

var nhanvien={

a:"Chỉ số BMI: ",

b:"của tôi",

bmi:(chieucao,cannang)=>

{

bmi\_=0;

bmi\_=nhanvien.a + cannang/chieucao\*cannang; //không sử dụng được this

return bmi\_;

}

}

console.log(nhanvien.bmi(162,60) + nhanvien.b)

//arrow

let connguoi={

tay:"tay dai",

chan:"chân dài",

mat:"mắt nhỏ"}

console.log(connguoi.tay + "\_" + connguoi.chan)

//sử dụng mảng trong object

let connguoi={

tay:"tay dai",

chan:"chân dài",

mat:"mắt nhỏ",

xinchao: (a,b)=>{ return “chào bạn" + a + b }}

console.log(connguoi.xinchao(“mập”, “béo”))

//sử dụng hàm trong object

var nhanvien={

cannang:'60',

cannangsauan: (a,b)=>nhanvien.cannang \*2 , //không sử dụng được this

cannangtruocan(){

return this.cannang \* 2; //hàm này mới sử dụng được this.

}}

console.log(nhanvien.cannangtruocan())

//phân mảnh trong object

var nhanvien={

cannang:'60',

tuoi:30,

cannangsauan: (a,b)=>nhanvien.cannang + nhanvien.tuoi

}

let {cannang:cannangcuaban,tuoi:tuoicuaban,cannangsauan:cangnangsauancuatoi}=nhanvien;

console.log(cannangcuaban + " \_ "+tuoicuaban +"\_"+ cangnangsauancuatoi())

//phân rả mảng sử dụng dấu ngoặc vuông

let arrays=["a","b","c"];

let [gt1,gt2,gt3]=arrays ; //

console.log(gt2)

// khai báo hàm mà nhận vào 1 mảng

let tinhtong=(...nthamso)=>{

let tong=0;

nthamso.map((value)=>tong+=value);

return tong;

}

console.log(tinhtong(1,2,3,4,5))

// số a,b và n tham số

let tinhtong=(a,b,...nthamso)=>{

let tong=0;

nthamso.map((value)=>tong+=value);// nó tự bỏ biến a và b hay giá trị 1,2

return tong;

}

console.log(tinhtong(1,2,3,4,5))

//sử dụng class

class connguoi{

constructor(){

this.tay="ngắn";

this.chan="dài"

}

}

connguoi1=new connguoi();

console.log(connguoi1.tay);

**//kế thừ từ class cha**

class connguoi{

constructor(){//biến trong constructor không khai kiểu cho nó. để là this.

this.tay="ngắn";

this.chan="dài";

this.xinchao=()=>"che cong";

}

xuatten(ten){

return "xin chao" + ten;

}

}

class contrai extends connguoi {

constructor(){

super();// từ khóa hàm kế thừa phương thức cha

this.gioitinh="nam";

}

}

connguoi1=new contrai();

console.log(connguoi1.xuatten("che cong binh"));

console.log(connguoi1.xinchao());

**//Duyệt mảng**

mang=["a","b","c"];

mangobject=[

{

ten:"che cong",

tuoi:10

},

{

ten:"nghiep",

tuoi:5

}

]

for(let giatri1 of mang)

{

console.log(`giá trị mãng ${giatri1}`)

}

for(let giatri of mangobject)

{

console.log(`giá trị mãng ${giatri.ten}`)

}

//tìm phần tử đầu tiên trong object dùng hàm find(), lọc tất cả phần tử fillter, findIndex tìm vị trí mảng

mangobject=[

{

ten:"che cong",

tuoi:10

},

{

ten:"nghiep",

tuoi:5

},

{

ten:"che cong",

tuoi:7

}

]

let giatitim=mangobject.filter((value)=>{

//cach 1

//if(value.ten==="che cong"){

// return value

// }

//

//cach 2: return thẳng luôn

return value.ten==="che cong"

})

console.log(giatitim);

//mảng map là mảng có key và value

let mangmap= new Map([

//[key,value]

["ten","che cong"],

["tuoi",10],

["toc","ngắn"]

]);

console.log(mangmap.get("ten"));

//get điều key.kiểm tra giá trị //has kiểm tra key. nếu có true, ko false //size kích thước file

//phân rả cấu trúc mảng

Cách 1:

let mangmap= new Map([

//[key,value]

["ten","che cong"],

["tuoi",10],

["toc","ngắn"]

]);

for(let key of mangmap){

//mangmap: thì lấy key và values;mangmap.values: lấy giá trị; mangmap.key:lấy key

console.log(key);

}

Các 2:

let mangmap= new Map([

//[key,value]

["ten","che cong"],

["tuoi",10],

["toc","ngắn"]

]);

for(let [key,values] of mangmap){//

console.log(key +"\_"+values);

}

Cách 3:

let mangmap= new Map([

//[key,value]

["ten","che cong"],

["tuoi",10],

["toc","ngắn"]

]);

mangmap.forEach((values,key)=>{

console.log(key +"\_"+values);

})

**//set, delete**

let mangmap= new Map([

//[key,value]

["ten","che cong"],

["tuoi",10],

["toc","ngắn"]

]);

mangmap.set("mui","to");// gán thêm giá trị vào map

mangmap.delete("ten");

mangmap.forEach((values,key)=>{

console.log(key +"\_"+values);

})

//mảng set là mảng chỉ có value

let arrays=["a","b","c"];

let gt=new Set(arrays);

gt.add("d");

gt.delete("a");

for(let item of gt)

{

console.log(item);

}

//Gọi class và truyền prop, state

* 1. cách gọi class tên class phải viết Hoa ký tự đầu.
  2. sự kiện<Text onPress={this.\_OnPress.bind(this)}>click</Text>

\_OnPress(){this.setState({state1:"Thanh Loan"})

* 1. props:

class con: <Text>Xin chào {this.props.ten}</Text>

Class cha: gọi class con<Con ten="AN"></Con>

* 1. state

clas cha phải có constructor

constructor(){

    super();

    this.state={state1:'Loan'};

class con: <Text >{this.state.state1}</Text>

* 1. stylesheet: khai báo
* const styles=StyleSheet.create({
* container:{
* flex:1,
* backgroundColor:'#fff',
* alignItems:'center',
* justifyContent:'center'
* }
* })

Gọi: C1: <View style={styles.container}>

C2: <Text style={{color:'red'}}>Xin chào </Text>

export default class Main extends React.Component{

  render(){

    return(

      <View style={styles.container}>

        <Text>wellcome to React Native</Text>

        <Con ten="AN"></Con>

      </View>

    )

  }

}

class Con extends React.Component{

  constructor(){

    super();

    this.state={state1:'Loan'};

  }

  \_OnPress(){

    this.setState({state1:"Thanh Loan"})

  }

  render(){

    return(

      <View>

        <Text>Xin chào {this.props.ten}</Text>

        <Text >{this.state.state1}</Text>

        <Text onPress={this.\_OnPress.bind(this)}>click</Text>

      </View>

    )

  }

}

const styles=StyleSheet.create({

  container:{

    flex:1,

    backgroundColor:'#fff',

    alignItems:'center',

    justifyContent:'center'

  }

})

import React, {Component} from 'react';

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

export default class MainFlex extends Component{

    render(){

        return(

            <View style={styles.container}>

                <TextInput placeholder={'nhập họ và tên'}></TextInput>

                <View style={styles.bluecontain}></View>

                <View style={styles.yellowcontain}></View>

            </View>

        )

    }

}

const styles=StyleSheet.create({

container:{

flex:1,

justifyContent:'center',

alignItems:'center'

},

bluecontain:{

width:50,

height:50,

backgroundColor:'blue'

},

yellowcontain:{

    width:50,

    height:50,

    backgroundColor:'yellow'

    }

})

* Text: ellipsizeModel enum (string):
* Head: …abc
* Middle: abc…yz
* Tail: abc…

ellipsizeMode={‘head’} numberOfLines={1}

Lưu ý: ellipsizeMode chỉ hoạt động cùng props numOfLines.

* NumberOfLines number (int): quy định số dòng đoạn text hiển thị
* onLayout function (): khởi tạo ỏ Mount và khi layout thay đổi sẽ phát sinh event là object chứa thông số liên quan đến kích thước của Component. onLayout ((event=>event.nativêvent.laout.x)
* onLongPress function(): sự kiện nhấn giữ.
* onPress function():nhấn
* Style:
* fontFamily: Number: font
* fontStyle: (normal,italic: kiểu chữ
* fontWeight: (normal,bold,100->9000: in đậm
* lineHeight: (number) : chiều cao text
* textAlign: (auto, left, center, justify): canh lề
* textDecorationLine: (none, underline, line-through, underline-through) ; bố trí đường gạch ngang cho text
* textShadowColor: mã màu: màu đổ bóng
* textShadowOffset: width: (number), height: (number) :kích thước đổ bóng
* textShadowRadius: number: bo gốc cho bóng
* textAlignVertical: (auto, top, bottom, center): canh lề text
* fontVariant: small-caps, oldstyle-num, lining-num, tablular-num, proportinal-nums: thay đổi font chữ ios
* testID: kiểu string xác định vị trí kết thúc đoạn text
* selectable: kiểu boolean: là copy paste.
* TextInput: onChangeText, OnSubmitEditting và onFocus

Vd: Style={{height:40, borderColor:’gray’, borderWidth:1}}

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

value={this.state.text}

* autoCapitalize enum(string): tự động viết hoa (characters: tất cả ký tự, words: chữ đầu tiên của mỗi từ, sentences: từ đầu tiên của câu, none: ko viết hoa.
* autoCorrect bool: Nếu true thì bật chức năng auto-correct (mặc định là true)
* autoFocus bool: Nếu true thì textinput sẽ được focus khi componentĐiMount được gọi (mặc định là false)
* blurOnSubmit bool: nếu true thì focus khi submit. Mặc định là true
* editable bool: nếu false không cho người dùng nhập văn bản
* keyboardType (string): khai báo keyboard sẽ được mở: default, email-address, numeric, phone-pad, ascii-capable, numbers-and-punctuation, url, number-pad, name-phone-pad, decimal-pad, twither, web-search.

Các tham số làm việc đa nền tảng: default, numeric, email-address, phone-pad.

* maxLength: giới hạn ký tự được phép nhập
* multiline: true đoạn text nhập nhiều dòng
* placeholder: nhập vào chuỗi nhắc nhỡ
* placeholderTextColor: (color) quy định màu text
* returnKeyType: các tham số đa nền tảng:done, go, next, search, send. Tham số làm việc trên android: none, previous.
* SecureTextEntry: true thì các đoạn text sẽ thay thế bằng dấu \*
* selectTextOnFocus: true đoạn text được gán thuộc tính này sẽ tự focus.
* Selection {start:number, end:number} sẽ chọn văn bản tương ứng ở vị trí truyền vào
* inlineImageLeft: nếu được khai báo thì sẽ render hình ảnh bên trái textinput
* inlineImagePadding: canh padding cho hình ảnh inline
* numberOfLines số dòng được phép hiển thị
* returnKeyLabel: trả về key của label
* underlineColorAndroid: màu border bottom textinput
* clearButtonMode: sẽ xuất hiện bên phải của textinput. Các tham số: never, while-diting, unless-editing, aways(IOS)

EVENT

* onBlur function: được gọi khi bị mất focus
* onChange:
* onChangeText:
* onContentSizeChange: được gọi khi kích thước của text thay đổi và trả về obj {nativeEvent:{contentSize:{width,height}}}
* onEndEditing: kết thúc nhập text
* onFocus: được gọi khi đoạn text được focus
* onLayout: được khởi tạo và layout thay đổi {x,y,width,height}
* onScroll: khi nội dung được scroll sẽ trả ra tham số là obj {nativeEvent:{contentOffset:{x,y}}}
* onSubmitEditing: được gọi khi người dùng nhấn submit
* Image Component
* <image source={require(‘./my-icon.png’)}
* Var con=this.props.active? require (‘./my-con-active.png’):require (‘incon-inactive.png’);

<Image source={icon}/>

* Event:

OnError: load ảnh thất bại

onLayout: dc gọi khi ứng dụng được Mount

onLoad: dc gọi khi hình ảnh được