Slot 15 – Các thành phần cơ bản của React Native
JSX
Component
hook
state, props
style
navigation
module
==========
1.JSX(Javascript XML): có thể viết code giông snhw html (vẫn là javascript)
ví dụ:
return(
<View></View>
);
2.Component:
-Function Component: phổ biến hơn do cách viết ngắn gọn
-Thường kết hợp với các hook
-Class component: dài, ít được sử dụng trong react native; không cần dùng các hook, mà dùng các hàm như contructor,…
ví dụ:

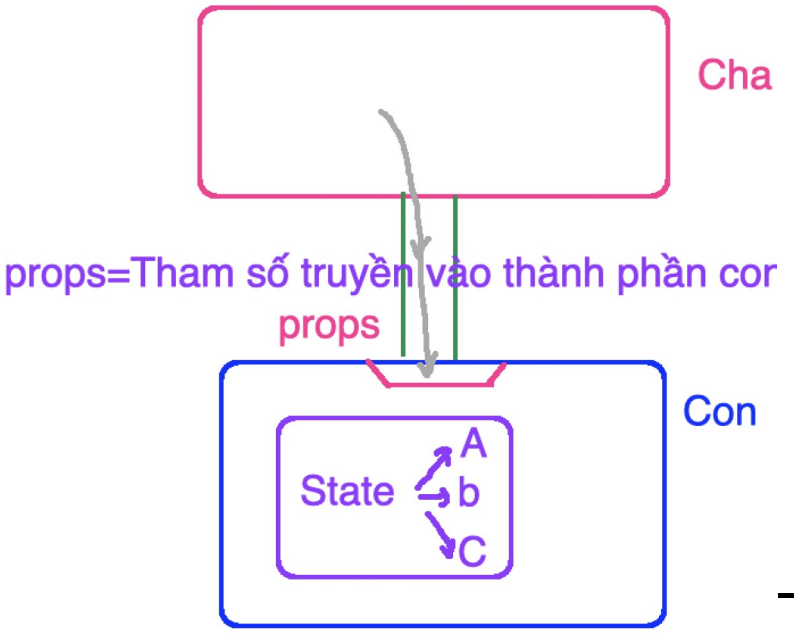
```
const welcome = ()=>{
      return <Text>Xin chao cac ban</Text>
}
```

3. props: là dữ liệu được truyền từ thành phần cha sang thành phần con

```
const Con = (props)=>{
      return <Text>Xin chao {props.name}</Text>
}
const Cha = () =>{
      return(
            <View>
                  <Con name=”Nguyen Van A” />
            </View>
      );
}
```

State:

Quản lý sự thay đổi dữ liệu của thành phần con.
Mỗi khi thành phần con thay đổi dữ liệu thì state phải cập nhật.



-

```
import React,{useState} from "react";
import { Text,View,Button } from "react-native";
const Counter = () =>{
    const [dem,setDem] = useState(0);
    return(
        <View>
            <Text>Dem so lan click</Text>
            <Button title="Click me" onPress={()=>setDem(dem+1)}/>
        </View>
    );
}
export default Counter;
```

2. Các thành phần giao diện trong React Native

Text
View
Image
Button
TextInput
ScrollView
FlatList

3. Style: dựa trên hệ thống định dạng bằng javascript

4. Hook: là cách sẻ dụng state và các tính năng khác mà không cần đến constructor như trong class component => hook chỉ dùng cho function component

useState: quản lý sự thay đổi giá trị của biến trong component

useEffect: render khi có dữ liệu thay đổi

```
import React,{useState,useEffect} from "react";
import { Text,View } from "react-native";
const Timer = ()=>{
    const [seconds,setSeconds] = useState(0);
    useEffect(()=>{
        const interval = setInterval(()=>{
            setSeconds(p=>p+1);
        },1000);
        return ()=>clearInterval(interval);
    },[]);
    return(
        <View>
            <Text>Timer: {seconds} seconds</Text>
        </View>
    );
}
export default Timer;
```

Navigation:

Điều hướng các màn hình

npm I @react-navigation/stack

npm I @react-navigation/native

Module:

React naive có thể sử dụng các module viết bằng ngôn ngữ khác như Objective-C, java, swift

Xử lý sự kiện:

```
import React,{useState} from "react";
import { Text,View,TextInput,Button } from "react-native";
const VD3=()=>{
    const [name,setName]=useState('');
    return(
        <View>
            <TextInput placeholder="Nhap ten" onChange={text=>setName(text)}/>
            <Button title="Submit" onPress={()=>alert(`Xin chao ${name}`)}/>
        </View>
    );
}
export default VD3;
```

Hàm mũi tên:

ngắn gọn, không tạo ra ngữ cảnh riêng mà dùng ngữ cảnh của môi trường nó đang hoạt động

- không dùng từ khóa new để tạo đối tượng mới

```
import React from "react";
import { Text } from "react-native";
const VD4 = ()=>{
    //thuc hien o day
    return "Xin chao";
}
const VD41 = () => "Xin chao";
```

Gọi hàm trong react native:

gọi hàm trong các thành phần của react native

```
import React from "react";
import { Button,Alert, View } from "react-native";
const VD5 = ()=>{
    const handlePress = ()=>{
        Alert.alert("Button Pressed");
    }
    return(
        <View>
            <Button title="Press me" onPress={handlePress}/>
        </View>
    );
}
export default VD5;
```

Hàm Callback:

hàm callback là hàm được truyền dưới dạng 1 tham số vào 1 hàm A,và được gọi lại khi hàm A thực hiện xong.

Nếu không xử lý đến API thì thông thường hàm callback là hàm đồng bộ.

Ví dụ: khi xử lý onPress thì hàm được gọi là hàm callback

```
import React from "react";
import { Button,Alert, View } from "react-native";
const VD6 = ()=>{
    //dinh nghia ham callback
    const docDuLieu = (callback) =>{
        //mo phong xu ly cuj the o day
        setTimeout(()=>{
            callback("Da nhan duoc du lieu")
        },2000);
    };
    //goi ham callback, ten la message
    docDuLieu((message)=>{
        console.log(message);
    });
}
export default VD5;
```

Hàm bất đồng bộ:

thông thường là các hàm xử lý đọc dữ liệu từ API

async: đánh dấu hàm là bất đồng bộ

await: lệnh sau từ khóa await sẽ dừng thực thi cho đến khi công việc thành công hoặc thất bại.

```
//ham bat dong bo
const fetchData = async () =>{
    try {
        let response = await fetch('https://');
        let data=await response.json();
    } catch (error) {

    }
};
```

Tham số hàm mũi tên:

- tham số là biến

```
//tham so la bien
const ham1 = (name) =>{
    console.log(`Xin chao ${name}`);
};
```

Tham số là đối tượng:

```
//dinh nghia ham co tham so la doi tuong
const ham2 = (person) =>{
    console.log(`name: ${person.name}, Age: ${person.age}`);
};
//goi ham co tham so la doi tuong
const person={name:'NVA',age:20};
ham2(person);
```

Tham số là mảng:

```
//dinh nghia ham co tham so la mang
const sumArray=(numbers)=>{
    return numbers.reduce((total,num)=>total+num,0);
};
//goi ham co tham so la mang
const numbers = [1,2,3,4];
console.log(sumArray(numbers));
```

tham số là callback:

```
    //dinh nghia ham co tham so la callback
    const thucthiCallback = (callback)=>{
        console.log("Truoc callback");
        callback();
        console.log("sau callback");
    };
    //goi ham co tham so la callback
    thucthiCallback(()=>console.log("chuong trinh goi ham callback"));
    //ket qua:
    //truoc callback
    //chuong trinh goi ham callback
    //sau callback
```

tham số là đối tượng với cấu trúc phân rã

```
//======
//khai bao ham có tham số la doi tuong destructuring
const inThongTin = ({name,age})=>{
    console.log(`Name: ${name}, Age: ${age}`);
};
//goi ham có tham số la doi tuong destructuring
const person1 ={name:"nva",age:35};
inThongTin(person1);
```

tham số là mảng với cấu trúc phân rã:

```
//-----------
//khai bao ham có tham số la Mang destructuring
const inMang = ([first,second])=>{
    console.log(`first: ${first}, second: ${second}`);
};
//goi ham có tham số la Mang destructuring
const mangso = [1,2,3,4,5,6,7,8,9,10];
inMang(mangso);//in ra 1 va 2
```

tham số là Rest parameter (nhiều tham số)

```
////goi ham có tham số la Rest parameter
const sum = (...numbers)=>{
    return numbers.reduce((total,num)=>total+num,0);
};
console.log(sum(1,2,3,4));//10
```

======
