Day 1

11-04-2022

React JS and React Native

Angular is framework. Angular is part of google. Angular is heavy. Angular follow standard. It follow design pattern ie MVC.

Angular Framework is component base architecture.

Angular we use one and two way data binding.

Base upon the typescript.

React JS a library. It is part of Facebook. Library is light weighted. In MVC react is View.

React JS provide virtual dom.

In React JS we use one way data binding.

In React JS we use JSX (JavaScript and XML).

React : it module provide set of library to do the operation on virtual.

ReactDOM : this module provide the bridge between react module and actual dom.

React JS a open source light weighted library which help to create the UI Component.

Component is control the view or part of view.

Angular and React is use to create SPA application.

In React JS we can create the component

1. Using function style : ES5 style
2. Using arrow style : ES6 style
3. Using class style : ES6 style

React JS or React Native we can do using JS or TS.

Html code

<div id="obj1">

</div>

Css code

div {

color: "#abc123"

}

JS code

//ReactDOM.render(<h1>Welcome to Simple React JSProgram</h1>,document.getElementById("obj1"))

function MyComponent() {

// coding......

return <div>

<Header></Header>

Welcome to React JS with function style

</div>

}

function Header() {

// logic...

return <h1>This is header component </h1>

}

ReactDOM.render(<MyComponent/>,document.getElementById("obj1"))

**Using node JS creating react JS Application**

npm install –g create-react-app

create-react-app welcome-app

React component contains two types of variable

1. State : state variable use to describe the behaviour of the component. State variable local that component. State variable known as a mutable property. You can change state variable using setState functions.

Syntax of state variable declaration

this.state={key:value,key:value};

old version of react we can create state variable only class component. Because of this reason class component is known as state full component.

From new version of react we can declare state variable function style component using hook ie useState.

1. Props : props are use to pass value from one component to another component ie parent to child component. Props can’t change. Props is known as immutable properties.

Day 2

12-04-2022

We can declare state variable inside a class component.

State variable we have to declare inside a constructor using syntax as

this.state ={key:value,key:value}

We can change the state variable using setState function.

New project using command as

**create-react-app login-app**

Day 3

13-04-2022

Component communication

Every component contains set variable. Set variable are local to that component. So we can use those variable within that component(inside the JSX). If we want to share the state variable from one component to another component.

Relationship between two component.

Parent – child relationship : using props we have pass the value form one component like @Input decorator in Angular Framework.

Child parent relationship : using props with callback function we can pass the value from child to parent like @Output decorator with EventEmitter or @ViewChild decorator.

Sibling relationship : child1 pass the value to parent using props with callback and parent pass the value to child2 using props.

Creating React JS application using typescript

npx create-react-app react-with-typescript –-template typescript

Day 4

14-04-2022

React routing :

To do routing in react we have install below module.

npm install react-router-dom

if we want to do routing we have to use pre-defined API

BrowerRouter. BrowserRouter we have to wrap for parent component.

React Native

React Native is base upon the React JS.

React Js is library base upon JavaScript.

React JS is use to create the Web Application

React Native is use to crate the hybrid mobile application or native application for android and iOS.

Using React Native we can access native platform API.

React JS + React Native = help to create hybrid mobile application.

Android –Java

iOS – Objective C language

react-native

React JS Android iOS React Native

Dom android.view UIView View Component

Div

P

H1

Etc

As well as base upon our requirement we can downloads external module or plugin

React Native Component compiled to Native code base upon Device.

React Native Component + Logic

To creating the react native application we can create using two ways

Expo cli : third party module which help to deploy the application on actual device or emulator

React Native cli : to the application we require run time environment.

To enable expo module we have to install

npm install –g expo-cli

after installed please check the expo command.

To create the project using expo we have write the command as

expo init welcome-app

once project created move inside a project folder using command as

cd project-name

React native can’t understand HTML as well as CSS properties directly.

JSX return using DOM Elements.

Div

P

H1

<div>

Welcome to Simple HTML Web page

</div>

Or

<div>

<p>Welcome to Simple HTML Web page </p>

</div>

Day 5

18-04-2022

Create new project using command as

expo init basic-element

option – typescript

Normal CSS we can’t use in react native.

Inline CSS

<tagName style=”color:red”></tagName>

Inline CSS syntax in React Native is

<ReactNativeComponent style={{property:value}}>

</ReactNativeComponent>

All Reactive native component doesn’t provide style attribute like Button component doesn’t provide style attribute.

State variable in function component with Button Component

Using react hook we can use state variable in function component.

React hook provided one of the pre-defined function is useState() which help create the state variable in function component.

const [variableName,setFunctionName]=useState(defaultValue);

View

Text

Button

TextInput component

$ echo fs.inotify.max\_user\_watches=524288 | sudo tee -a /etc/sysctl.conf  
and  
$ sudo sysctl -p

Day 6

19-04-2022

React Native Layout :

React Native layout helps us in structure the application which makes the user interface (UI) more attractive. Reactive native use flex layout style.

Flexbox work in the same way as in work over CSS.

The default flex direction to column instead of row (in CSS) and only a single number is supported by the flex parameter

Flex-direction

Row wise

Columns wise

**expo init react-flex-layout**

const styles = StyleSheet.create({

  container: {

    flex: 1,

    //flexDirection:'row',

    //backgroundColor: '#fff',

    alignItems: 'center',           // top to bottom

    justifyContent: 'center',       // left to right

    paddingTop:0,

    backgroundColor:'orange',

    paddingBottom:0

  },

  subView1:{

    //flex:1

    flexDirection:'row'

  },

  subView2:{

    //flex:1

    flexDirection:'row-reverse'

  },

  firstBox: {

    flex:3,

    backgroundColor:"red",

    padding:10

  },

  secondBox: {

    flex:2,

    backgroundColor:"green",

    padding:10

  },

  thirdBox: {

    flex:1,

    backgroundColor:"blue",

    padding:10

  },

  fourthtBox: {

    flex:1,

    backgroundColor:"yellow",

    padding:30

  }

});

View the items from array (it hold primitive value or complex object).

expo init employee-view-items

ScrollView : It is known as scrollable container, which scrolls multiple child component and views inside it.

ScrollView load all the data on device at time.

Which will affect the performance.

FlatList : react native providing FlatList type of scroll view which load the data one demand.

Touchable component : using this component we can create custom button with user-defined styles button.

20-04-2022

expo init task-tracker

Alert message in react native

Add, Display and Delete task in one component

**App.tsx**

import { useState } from 'react';

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

export default function App() {

  const [task,setTask]=useState<string>("");

  const [taskDetails,setTaskDetails]=useState<string[]>([]);

  //const [result,setResult]=useState<boolean>(false);

  const handleTask=(data:string)=> {

        setTask(data);

  }

  const addTask= ()=> {

      setTaskDetails(previousTask=>[...previousTask,task]);   // using spread operator we are adding new task.

      setTask("");

      Alert.alert("Info","Task Added successfully");

  }

  const deleteTask=(data:any)=> {

    //console.log(data.index);

    Alert.alert("warning","Do you want to delete",[{text:"Yes",onPress:()=> {

      setTaskDetails((taskInfo:string[])=>{

           return taskInfo.filter((v:any)=>v!=data.item);

      })

    }},{text:"No"}])

  }

  return (

    <View style={styles.container}>

     <View style={styles.inputView}>

     <TextInput placeholder='Enter your task' style={styles.inputText}

         value={task} onChangeText={handleTask}

     />

     </View>

    <TouchableOpacity onPress={addTask}>

          <Text style={styles.myButton}>Add Task</Text>

    </TouchableOpacity>

    <FlatList

     data={taskDetails}

     keyExtractor={(item,index)=> {

       return index.toString()

     }}

     renderItem={(tt)=> {

return  <View><TouchableOpacity onPress={()=>deleteTask(tt)}><Text style={styles.task}>{tt.item}</Text></TouchableOpacity></View>

     }}

    />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

    margin:30

  },

  inputView: {

    //flex:1,

    flexDirection:'row',

    borderWidth:2,

    paddingTop:5,

    borderColor:"red",

    borderRadius:15

  },

  inputText: {

    fontSize:25,

    padding:5,

    width:250

  },

  myButton : {

      borderWidth:1,

      borderColor:"orange",

      backgroundColor:"#fa12bb",

      width:150,

      textAlign:'center',

      marginTop:5,

      padding:10,

      borderRadius:15,

      fontSize:20

  },

  task: {

    margin:10,

    borderWidth:2,

    width:300,

    padding:5,

    paddingLeft:5,

    borderRadius:10,

    fontSize:25,

    color:'white',

    backgroundColor:"gray"

  }

});

21-04-22

**Context API**

Context API provide clean and easy way to share the state between two component (class style or function style).

Context API provide global state concept which help to share the data between two component very easily.

Context API build block or component

1. Context
2. Provider
3. Consumer

Context : React provide createContext function which help to initialize the global state value.

number, boolean, string, array or complex object type.

Provider : After created context, Provider provides the capability to access context which wrapped form it. Provider provides the data and function to pass down to all the component. The component level can be n level.

Consumer : Consumer allow to access the value to child component which parent that is wrapped by provider

App.tx

First fourth

Second

Sixth

Third

Fifth

Redux :

Angular with redux : ngrx

Redux is state management tool. Redux is a predicable state container for JavaScript app.

Redux keep the data (state) in a store and any component can access those data from a store.

Redux

Action

Reducer

Store

State : global state

We can make local state as a global state using

Redux store concept.

Reducer : it is normal JavaScript function which help to changes on store base up on action provided by component using dispatcher.

Action is responsible to change the state from a store.

Create simple project

expo init simple-redux-app

npm install redux

npm install react-redux

Context.tsx

import { createContext } from "react";

type myContextType = {

    num?:number;

}

const AppContext = createContext<myContextType>({

    num:100

})

export default AppContext;

**MyConsumer1.tsx**

import { useContext } from "react";

import { Text, View } from "react-native";

import AppContext from "../context/Context";

export default function MyConsumer1() {

const contextType = useContext(AppContext)

    return(

        <View>

           <Text>{contextType.num}</Text>

        </View>

    )

}