1. When we want to pass state from app component to grandchild component without having component in between. The grandchild will have access to the app.JS. The application structure looks like {count :0, inc =(), dec()}. The app render activity indicator the move the app to the center and use style component. The Grandchild renders a state count property and have two button to increase and decrease the count.

Answer

App.js

/\*\*installed expo liabraries \*/

import 'react-native-gesture-handler';

import React from 'react';

import { NavigationContainer } from "@react-navigation/native";

import { createStackNavigator } from "@react-navigation/stack";

/\*\*self made imported components \*/

import Home from 'components/Home;

import GrandChild from 'components/Contacts';

export *const* MyContext = React.createContext()

*const* Stack = createStackNavigator();

export default *function* App() {

*const* [state, setStateFun] = React.useState({ count: 0 })

  return (

    <*MyContext.Provider* value={{ count: state.count, setStateFun }}>

      <*NavigationContainer*>

        <*Stack.Navigator* initialRouteName='HOME'>

   <*Stack.Screen* name="HOME" component={Home} options={{ title: "Home" }} />

    <*Stack.Screen* name="GRANDCHILD" component={GrandChild}  />

        </*Stack.Navigator*>

      </*NavigationContainer*>

    </*MyContext.Provider*>

  );

}

grandchild.js

import React, { useEffect } from 'react';

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

import { MyContext } from '../App'

*const* Contactss = () *=>* {

*const* context = React.useContext(MyContext);

*const* handleIncrement = () *=>* {

    context.setStateFun({ count: context.count + 1 })

}

*const* handleDecrement = () *=>* {

    context.setStateFun({ count: context.count - 1 })

}

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

    <*Text*>  {context.count}</*Text*>

    <*Button* title="Increment" onPress={handleIncrement} />

    <*Button* title="Decrement" onPress={handleDecrement} />

</*View*>

}

*const* styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  }

})

export default Contactss;

1. Complete your code for react native functional component using reducer (). The user should be able to add new to-do item. It should be able to toggle with complete Boolean property. The initial state object structure look like this.

Intialstate={ Id: "", todoTextinput: "", Todos: ["voilon ", " gittar "], complete:false) }

Answer:

import React from 'react';

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

*const* initialState = {

id: "",

  todoTextinput: "",

  todos: ["voilon", "gittar"],

  complete: false,

  //lastId: Math.floor(Math.random() \* 100)

}

*const* reducer = (*state* = { initialState }, *action*) *=>* {

  switch (*action*.type) {

    case "ADD":

      return { ...*state*, todoTextinput: *action*.payload }

    case "TODOARRAY":

      return { ...*state*, todos: [...*state*.todos, *state*.todoTextinput] }

      case "TOGGLE":

      return { ...*state*, complete: true }

    default:

      return *state*;

  }

}

*const* MyFunBase = () *=>* {

*const* [state, dispatch] = React.useReducer(reducer, initialState);

  return (

    <*View* style={styles.container}>

      {state.complete ? <*FlatList*

        data={state.todos}

        renderItem={({ *item* }) *=>* {

          return (

            <*View*> <*Text*>{*item*}</*Text*> </*View*>

          )

        }}

        keyExtructor={(*item*) *=>* { *item*.id }}

      /> : null

}

      <*Text*> Name :{state.todoTextinput}</*Text*>

      <*TextInput*

        placeholder="todos.."

        value={state.todoTextinput}

        style={styles.input}

        onChangeText={(*myText*)*=>*dispatch({type:"ADD", payload: *myText*})}

      />

  <*Button* title="Toggel" onPress={() *=>* dispatch({ type: "TOGGLE" })} />

  <*Button* title="Add to array" onPress={() *=>* dispatch({ type: "TODOARRAY" })} />

    </*View*>

  );

}

*const* styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    alignItems: 'center',

    justifyContent: 'center',

  },

  mapStyle: {

    width: Dimensions.get('window').width,

    height: Dimensions.get('window').height,

  },

  input: {

    marginTop: 16,

    paddingVertical: 8,

    borderWidth: 4,

    borderColor: "#20232a",

    borderRadius: 6,

    backgroundColor: "#61dafb",

    color: "#20232a",

    textAlign: "center",

    fontSize: 30,

    fontWeight: "bold",

    width: "80%"

  }

});

export default MyFunBase;

1. Complete the code for the react native functional component about fetching data from the given api and display the title by using flat list component. The response is json object. Response =[{Id :”1”, titile :”first item ”},{Id :”2” , titile :”second item ”}, {Id :”33” , titile :”third item”}]

Answer

import React, { useState } from 'react';

import axios from 'axios';

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

*const* FetchList = () *=>* {

*const* [data, setData] = useState({ data: '' });

  React.useEffect(() *=>* {

    fetchData();

  }, [])

*const* fetchData = async () *=>* {

    try {

*const* response = await axios.get('https://api.kanye.rest/')

      console.log(response.data.quote);

      setData(*prevState* *=>* ({ ...*prevState*, data: response.data.quote }))

      // console.log(response);

    } catch (error) {

      console.log(error.message);

    }

  }

  return (

    <*View*>

      <*Text*>This is Good</*Text*>

     <FlatList

        data={data}

        renderItem={({ item }) => {

          return (

            <View>

              <Text>{item.title}</Text>

            </View>

          )

        }}

        keyExtractor={item =>{item.id}}

      />

export default FetchList;

//////////////////////////////////////////////////////////////////////

      <*ScrollView*>

        {data.data.map(item=>{

          return <View>

            <*Text*>{item.title}</*Text*>

          </View>

        })}

      </*ScrollView*>

    </*View*>

  )

}

export default FetchList;

4. Complete the following to fetch data from the given API

1. when the user click “**QUIZE**” button/page , it redirect you to questions screen,

2. Questions screen should display the questions and their choices all along, and besides should update the navigation header to “QUIZE-QUESTIONS”

3.The questions screen manages a state having the following; {count, count1, index count}. the first state is counting the total questions, the second one is how many questions has been answered correctly. The third one is to be tracking the index of currently displaying question.

4.the screen should update accordingly

5. the screen should also redirect to quizzes if finish with your questions.

Answer:

**Short answer questions**

1. react native have virtual Dom. **True**

**React Native** differs by converting the contents of the **Virtual DOM** description into **native** UI elements.

1. React node used by react native. True

A React node is defined as: a light, stateless, immutable, virtual representation of a DOM node. React is used to define a virtual DOM using React nodes, that fuel React components, that can eventually be used to create a real DOM structured or other structures (e.g., [React Native](https://facebook.github.io/react-native/)).

https://www.reactenlightenment.com/react-nodes/4.1.html

1. React native props can be immutable and not mutating

The simple rule is props should not be changed/ immutable.

https://stackoverflow.com/questions/54825692/immutability-and-components-props-in-react-native-vs-react

1. parent text on text on the top of each other

It will print them in the same line

1. what is reconciliation algorithm and when it is executed?

**Reconciliation algorithm**: Compares the previous V-DOM with the next V-DOM.

Updates the internal Instance which is a Component Tree structure in JavaScript Object(Virtual DOM). And updates the actual DOM only if there is any change.

1. Write a code for componentwillunmount functional component

useEffect(function() {

window.addEventListener(eventName, eventListener)

**return** function cleanup() {

window.removeEventListener(eventName, eventListener);

}

}, [ ]);

1. What is the default value of flex direction: **Column**

Two options **in flexDirection**: row and column. The **default setting** is column

1. What is pass as prop to flatlist ? **data, renderitem, KeyExtructor**
2. What is the difference between react and react native?

**No Browser APIs in React,**

**Navigation System**,

**Styling (no className in RN)**,

**All Base components are different** (not global in scope like react) : where react uses html tags and custom components for its JSX, but react native uses components imported from react native

1. What are the two threads found in every component application?

**JS thread** (application code) and **Main (UI)** thread

1. When to use useState and useReducer.

**useState: when we have simple state**

**useReducer: when our state is complex**