**Answer Codebase**: https://github.com/EdocEdoc/SerinoCodingExam  
**Answer Expo App**: exp://exp.host/@vagenodipa/SerinoCodingExam?release-channel=default  
  
=====================================================================================

**Question #1 - React - Identify the Problem and Refactor**

=====================================================================================

Please identify the problems and tell us what the problems are, then improve this React Component by coding your own version!

It would be a plus point if you can convert/refactor them into React hooks.

class MyComponent extends React.Component {

constructor(props) {

// set the default internal state

this.state = {

clicks: 0

};

}

componentDidMount() {

this.refs.myComponentDiv.addEventListener('click', this.clickHandler);

}

componentWillUnmount() {

this.refs.myComponentDiv.removeEventListener('click', this.clickHandler);

}

clickHandler() {

this.setState({

clicks: this.clicks + 1

});

}

render() {

let children = this.props.children;

return (

<div className="my-component" ref="myComponentDiv">

<h2>My Component ({this.state.clicks} clicks})</h2>

<h3>{this.props.headerText}</h3>

{children}

</div>

);

}

}

=====================================================================================

ANSWER

=====================================================================================

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

import React, { useState, useEffect } from "react";

const QuestionOneScreen = ({ headerText, children }) => {

  const [clicks, setClicks] = useState(0);

  const clickHandler = () => {

    setClicks(clicks + 1);

  };

  useEffect(() => {

*// removed the event listener for "click"*

*// will be using the built in event listener from react native touchable opacity => onPress*

*// components will mount*

    console.log("components will mount");

    return () => {

*// components will unmount*

      console.log("components will unmount");

    };

  }, []);

  return (

    <TouchableOpacity onPress={clickHandler}>

      <Text>My Component ({clicks} clicks)</Text>

      <Text>{headerText}</Text>

      {children}

    </TouchableOpacity>

  );

};

*// setting defaults for required component props*

QuestionOneScreen.defaultProps = {

  headerText: "Default Header Text",

};

export default QuestionOneScreen;

=====================================================================================

**Question #2 - React - Solve the Problem**

=====================================================================================

Complete the following <TodoList> component to display todos and allow for adding and removing of todo items

const todosReducer = (state, action) => {

switch (action.type) {

case 'ADD\_TODO':

case 'REMOVE\_TODO':

}

};

const TodoList = () => {

const [todos, dispatch] = useReducer(todosReducer, []);

return (

<div>

<ul>

{todos.map((todo) => (

<li><button>Remove todo</button></li>

))}

</ul>

<button>Add todo</button>

</div>

);

};

=====================================================================================

**ANSWER**

=====================================================================================

import {

  Pressable,

  ScrollView,

  StyleSheet,

  Text,

  TextInput,

  View,

} from "react-native";

import React, { useReducer, useState } from "react";

const todosReducer = (state, action) => {

  switch (action.type) {

    case "ADD\_TODO":

      return [...state, { index: new Date(), name: action.payload }];

    case "REMOVE\_TODO":

      return state.filter((item) => item.index != action.payload.index);

    default:

      return state;

  }

};

const QuestionTwoScreen = () => {

  const [toDos, dispatch] = useReducer(todosReducer, []);

  const [toDO, setToDo] = useState("");

  const submit = () => {

    if (toDO) {

      dispatch({ type: "ADD\_TODO", payload: toDO });

    }

    console.log(toDos);

    setToDo("");

  };

  const remove = (item) => {

    dispatch({ type: "REMOVE\_TODO", payload: item });

  };

  return (

    <View style={styles.container}>

      <ScrollView>

        {toDos.length > 0 && (

          <Text style={{ marginTop: 10 }}>To DO Items:</Text>

        )}

        <View>

          {toDos.length > 0 ? (

            toDos.map((item, index) => (

              <Pressable key={index} onPress={() => remove(item)}>

                <Text style={[styles.inputTxt, { borderColor: "green" }]}>

                  {item.name}

                </Text>

              </Pressable>

            ))

          ) : (

            <Text>No to Do Item</Text>

          )}

        </View>

        <Text style={{ marginTop: 20 }}>Add To DO Items:</Text>

        <TextInput

          value={toDO}

          onChangeText={setToDo}

          style={styles.inputTxt}

          onSubmitEditing={submit}

        />

        <Pressable style={styles.btnOne} onPress={submit}>

          <Text style={{ color: "white" }}>Add Todo</Text>

        </Pressable>

      </ScrollView>

    </View>

  );

};

export default QuestionTwoScreen;

const styles = StyleSheet.create({

  inputTxt: {

    borderWidth: 1,

    borderColor: "black",

    borderRadius: 5,

    padding: 10,

    marginTop: 10,

  },

  container: {

    flex: 1,

    backgroundColor: "#fff",

    padding: 20,

  },

  btnOne: {

    alignItems: "center",

    justifyContent: "center",

    paddingVertical: 12,

    paddingHorizontal: 32,

    borderRadius: 4,

    elevation: 3,

    backgroundColor: "black",

    marginTop: 20,

  },

});