React States and Hooks - Practice Exercises

Questions on React State

1. Create a simple click counter app using React states. You will need a class component called "**MyCounter.js**" that will depend on another component called, "**CounterDisplayer.js**". "**MyCounter.js**" keeps track of the click count value in a state and this is where you will initialize your state that will hold your click counts. It is also under "**MyCounter.js**" that you will write a function, called "**allClicksCounter()**" that will update/increase your state by one whenever a button is clicked. Under "**MyCounter.js**" you will also need a button which calls the method "**allClicksCounter()**" when it is clicked. Please display the number of clicks right below the button. The main purpose of "**CounterDisplayer.js**" is to display the number of times the button on the "**MyCounter.js**" component is clicked. Meaning, you will need to pass the updated click data from "**MyCounter.js**" to "**CounterDisplayer.js**"" using props.

2. Create another class component named "**EvenCounterDisplayer.js**". Please note that "**MyCounter.js**" will also depend on "**EvenCounterDisplayer.js**" to display <u>ONLY</u> the even number of click counts. Meaning, your "**EvenCounterDisplayer.js**" tracks and displays only the even number of times the button found under "**MyCounter.js**" is clicked. Meaning, when the button is clicked just once, your "**MyEvenCounter.js**" component should just say **"Clicked 0 times"**. But when the button is clicked again it should say "**Clicked 2 times**". When the button is clicked for a third time it should just say "**Clicked 2 times**". When clicked again for the fourth time it should say "**Clicked 4 times**" and so forth. Display the "**MyEvenCounter.js**" component right below your "**MyCounter.js**"' component so that you can see both counters together

   Watch this demo clip for question 1 & 2, showing what your app should look like

Question on React Hooks

3. Create a simple click counter app using React Hooks. You will need a functional component called "**IncreaseDecreaseCount.js**". There will be three buttons; a button to increase, a button to decrease and a button to reset click count values and you will need to implement the useState() Hook to update the clicks. Right above the buttons, there will be a count displayer with an initial click value of 0. When the button to increase is clicked, the click value increases by one, when the decrease button is clicked, the click value will decrease by one and when a button to reset is clicked, the click value resets to initial value which is 0. Whenever any button is clicked, the change in the click value will be shown on the count displayer.

   ** Watch this demo clip for question 3, showing what your app should look like **

4. Create a simple click counter app using React Hooks. You will need a functional component called "**UseEffectForTitle.js**". There will be one button and right below the button, there will be a count displayer with an initial click value of 0. When this button is clicked, the click value will increase by one and the displayer will show the changed count value. You will need to implement the useState() Hook to update the click values. When the button is clicked and the click value changes, you will need to change the document's title to show the count value as well. Basically, the change in the document's title is a side-effect to the change in the count value due to the button's click. **Hint**: You will need to use useEffect() to write the logic/function that will change the document's title to the current value of the click count.
   - When the component is rendered for the first time (or when it is mounted), display on the screen an alert text that says "*Component is mounted*". **Hint**: You will need to use the useEffect() Hook to show the alert message and the message should display only when the component is mounted, not every time the button is clicked.

   ** Watch this demo clip for question 4, showing what your app should look like**

********* Steps to follow for each question *********

Steps to follow for question 1

- create a folder called “**Week-6**” in your **“Phase-3”** folder
- In your “**Week-6**” folder create a folder named “**StatesAndHooksProject**”
- Clone the GitHub repository starer code from here and save it under your “**StatesAndHooksProject**”
- Open the React app folder you cloned with VSC
- The modules this React app depends on are not included in the GitHub repo, so you must install the dependencies
  - To install the dependencies, run this command on your terminal: npm i
  - Wait until React finishes installing all the dependencies
- Go to the “**Components**” folder located in the repo and create 2 class based components, named "**MyCounter.js**" and “**CounterDisplayer.js**”
- Do not forget to do the following in your "**MyCounter.js**" component:
  - Remember to call your constructor() function in your component
  - Make sure to call the super() method in your constructor
  - Initialize your state that will count the clicks to 0 in the constructor ()”
  - Create the “**allClicksCounter()**” function that will update/increase your state by one. You will need to use the setState() method in your **allClicksCounter()** function to update your state
  - Render a button which calls the method all “**ClicksCounter()**” when this button is clicked
  - Pass the updated counter value to the “**CounterDisplayer.js**’” component to display the number of clicks counted
  - Do not forget to import the "**CounterDisplayer.js**" in your "**MyCounter.js**"
- **Hint:**
  - The "**MyCounter.js**" component is going to be dependent on the "**CounterDisplayer.js**'"
  - The "**MyCounter.js**" component should be the one to be called on the App.js component, not the “**CounterDisplayer.js**”
  - Do not forget to import your "**MyCounter.js**" under “**app.js**”

    - You would need to use both states and props for this question

Steps to follow for question 2

- Go to the “**Components**” folder and create a component named “**EvenCounterDisplayer.js**”
- Do not forget to do the following in your "**MyCounter.js**" component:
    - Create the “**evenClicksCounter()**” function that will return the click counts that are only even. You will need to use the setState() method in your “**allClicksCounter()**” function to update your state to only contain the even counts
    - Pass the updated counter value to the “**EvenCounterDisplayer.js**”” component to display the even number of clicks counted
    - Do not forget to import the "**EvenCounterDisplayer.js**" in your "**MyCounter.js**"
- **Hint:**
    - The "**MyCounter.js**" component is going to be dependent on the “**MyEvenCounter.js**”
    - Do not forget to import your“**MyEvenCounter.js**” under "**MyCounter.js**"
    - The “**MyCounter.js**” component should be the one to be called on the App.js component, not the “**MyEvenCounter.js**”
    - There will be only one button to click, the button found under "**MyCounter.js**". When that button is clicked, both “**allClicksCounter()**” function and the “**evenClicksCounter()**” function get executed. Meaning, when the button is clicked, the "**MyCounter.js**" component will show both of the click counters right below the button.
    - **Hint**: You would need to use both states and props for this question

Steps to follow for question 3

- Go to the “**Components**” folder and create a function component called "**IncreaseDecreaseCount.js**"
- Don’t forget to do/include these under your "**IncreaseDecreaseCounter.js**" component
    - Importing useState Hook from React
    - Declaring the state variable for the component to hold the increased/decreased click value into your component
    - Declaring an updater function that will update the current click count value

    - Passing in your initial state value of as an argument to the useState() Hook
    - Writing the logic/functions that will increase, decrease and reset your click count values
    - Attaching each click event with the respective event handler function
- Import "**IncreaseDecreaseCount.js**" in your App.js and
- Make sure to call the "**IncreaseDecreaseCounter.js**" component in your App.js

<u>Steps to follow for question 4</u>

- Go to the "**Components**" folder and create a function component called "**UseEffectForTitle.js**"
- Don't forget to do/include these under your "**UseEffectForTitle.js**"' component
    - Importing useState and UseEffect Hooks from React
    - Declaring the state variable for the component to hold the click value into your component
    - Declaring an updater function that will update the current click count value
    - Passing in your initial state value of as an argument to the useState() Hook
    - Writing the logic/function that will increase the click count by 1 whenever there is a button click
    - Attaching the click event with the event handler function
    - Writing a function that will alert the "*Component is mounted*" message when the component is mounted (rendered for the first time). You will need to use useEffect() Hook to show this effect
    - Writing a function that will change the document's title to be the current count value whenever there is a button click. You will need to use useEffect() Hook to show this effect
- Import "**UseEffectForTitle.js**" in your App.js and
- Make sure to call the "**UseEffectForTitle.js**" component in your App.js

Happy coding 😀