React Advanced Topics:

Since we have seen how react renders and react has only one div as returned.

-- Here we have div which is a component.

-- You include all the components in App file and at the end of a day that is rendered in Index.js which is entry point of your program.

useState:

-- even though on click your function is changed. Because of virtual dom we have can not render the elements which have not changed. Example of simple element rendered is great for that.

-- useState is itself is a function and not module that is why we return it as a {} name in file.

-- The function returns an array of value and a function attached with that value. So, you can preserve the value of that variable.

useState with object:

One of the problems I have inquired here is basically, when you are accessing an array or an object and you want to change it, when you are trying to set a particular property from object you essentially wipe out the whole object.

Now we can use spread operator to access the inside properties and change it using setPerson function.

OR, you can just give them individual set properties all together.

Complex Counter:

Normal Counter :

How I made it? Well a common global value which will get updated on 3 button using setValue function and those setValue functions are just arrow function so get executed just when you click on them

The problem occurs when you want to delay your output using setTimeout and in this case of counter, even though you click it 3 times, it updates value just for once and that is a problem. In order to avoid that use use setValue functions with parameters.

Here, you use arrow functions and parameter as prevState which ensures that we get an old value from the function.

So, we have learnt here that we can set our own function in second argument in react and example can be seen in ueStateCounter example.

Activity – Done – Birthday Reminder

2. UseEffect

-- Multiple things came to play here.

-- useEffect by default rerender every single time of load.

-- Again you can not place hooks into if else statement, that will be a next topic of conditional rendering.

-- useState was used to manage state of any variable like any type of variable(I mean we need conditional rendering even with any variable, lets say a variable crosses a limit then I am going to rerender but maybe I am wrong IDK for now)

-- Update on the point above(Lest say you want to update after counter goes above 1 well if else can be put in the hooks itself )

-- Coming to useEffect it manages all the side effects.(More on that later)

-- Now you can have a condition inside the useEffect that is valid but the rerender still happens.

-- **ONE OF THE DOUBTS I HAD**

**--** I had doubt about why console.log outside the useEffect was called and it turns out that console statement are called over and over again but if you are setting up component and functions ingeenral that does not rerender.

-- Now, as useState takes 2 arguments useEffect also takes 2 arguments (2nd argumend not needed) but if we put 2nd argument an empty **array it will just initially rerender and then stops.**

**--** We can conditionally render element based on value changed or not. You can inject second argument of a useEffect function and that will rerender only when dependency have been changed.

-- And you can have as many as useEffect as you want.

**Golden Rule of Hooks**

**--- useState to manage hooks and useEffect when something changes and you want to capture it.**

**2.** Cleanup Function

-- This helps you to basically track the resizing of window object and how you can do it..

-- You have to attach useEffect with a call back function of event Listner and that goes with callback function which actually use useState second argument for setting up the size to new Size.

-- **DOUBT** I have used dependency to track change in window size and setSize function to basically setSize once again but nothing happens)

**--- Possible Explanation(Maybe size is not changing and that is separate from window.innerwidth)**

**---** The idea here is there are multiple issues with useEfefct and we can see that we rerender the component but what if we are adding an event listener to window resize that can trigger response every time we resize our window that can result into memory leak.

--- The solution to this issue is basically removeEvent lIstner at the same time so even we trigger useeffect we return a removal of EventListener at the same time.

3. FetchData

-- Project is to setup a call function which you can use to fetch data from API.

-- You can not async await useEffect as a function. The reason is useEffect don’t expect promise to be returned.

-- Now if we setUsers in async await. And useEffect has triggeres function then it is troublesome.The reason is getUsers calls setUsers and it goes in infinite loop of rerender.