```
/* ========== React work behind the seen ===============

    1) How Does React Work Behind The Sceen?
    2) Understanding the Virtual DOM & DOM Updates
    3) Understanding State & State Updates

    How Does React Work?

    React => "A javaScript Library for building user interfaces"

    React => know only Components and state change nothing elase
          React does not konw the HTML
          if the component change then React change the State with previous on

    ReactDOM => interface to the web (what the user sees)

                        Context
                          |
            props =>      Components      <=  ReactDom
                        |
                        State

       =============== Re-Evaluating Components !== Re-Rendering the DOM ============

       Components
          1) Re-evaluated whenever props, state or context changes
          2) React executes component functions

       Real DOM
          1) Changes to the real DOM are only made for differences between evaluations

    ===================== Virtual DOM Diffing ============
    React work like this way
       1) React manage a Virtual DOM copy of Actual DOM
       2) If any component changes on the Virtual DOM then
       3) React Understanding where the changes happened
       4) then Changes only those component efeciently
       5) use the Diffing those to DOM and made changes to

    ================== Parent component re Run for State change ============

       1) All child components will be Re Run Again
       2) it will be cost some performance issue
       3) we can be control re run for child components run again again

    ====================== React.memo(DemoOutput) for stopping unesesery component re run ==============

    By the help of memo() we can control re run for child components run again

    1) when the parent component State change the it will run
    2) if the => export default React.memo(DemoOutput) will chack that component props change or not
    3) it the component  DemoOutput props change then it will re Run again
    4) the component DemoOutput Re Run. it's child component will also re Run again

*/
       // ================= Parent component =============


       import React, { useState } from 'react';
       import DemoOutput from './Demo/DemoOutput';

       function App() {
         const [showParagraph, setShowParagraph] = useState(false);
         const showParagraphHandeler = () =>{
           setShowParagraph( (prevState) => !prevState);
         }
         return (
           <React.Fragment>
             <p>Hi, there!</p>
             <DemoOutput show = {showParagraph} />
```

```
            <button onClick={showParagraphHandeler}>Click Me</button>
          </React.Fragment>


        );
      }
      export default App;


      //=========== child component ==============

import React from 'react';

function DemoOutput(props){
   return <p>{props.show ? "This is new" : ""}</p>
}
export default React.memo(DemoOutput);


/*
   ============== useCallback() hook ===========

   1) to prevent reCreate same function or object again and again when component re run
   2) useCallback() create a function only one time and reference that function to utilize the function
   3) let obj1; let obj2; => obj1 = obj2 -> here same object create location indicate
   4) useCallback() also do the same job for us
*/
   import React, {useCallback} from 'react';

     const showParagraphHandeler = useCallback( () =>{
       setShowParagraph( (prevState) => !prevState);
     } , [])
/*
     here dependency not given that's why this function is created again and again

     where as some use case that time when the dependency is changed that useCallback() re executed and new Function is created
*/

import React, {useCallback} from 'react';

const showParagraphHandeler = useCallback( () =>{
   if(allowToggle){
      setShowParagraph( (prevState) => !prevState);
   }


} , [allowToggle])

const allowToggle =  () =>{
   setAllowToggle(true);
}

/*
   when re run that time check that allowToggle value is change or not
   if it is change then the new function will created

*/


/*   ================ useMemo() Hook =============

   1) useMemo() hook use to memorize the array or other data which we don't want to re calculate
   2) Like sort of array  that when the component call again and agian ... we can optimize that to store calculation
   3) if new element come only that time calulated the sort data....
   4) in this case useMemo() hook can help us to memorize

   useMemo( function return , [])

   return value which we want to store...
*/
import React, {useMemo} from 'react';
```

```
import React, {useMemo} from 'react';

function App(){
    const listItems = useMemo( () => [5, 3, 1, 10, 9], []);

    return (

        <DemoList items={listItems} />
    );

}


// ......... Demo list child components

import React, {useMemo} from 'react';

function DemoList(props){

    // props destructuring
    const {items} = props;

    const sortedList = useMemo( () =>{
        return items.sort( (a, b) => a - b);
     }, [])

    return (
        {sortedList.map(item) =>

            ..... list reandering
        }
    );
}
```