

```

1  /* ===== React work behind the seen =====
2
3  1) How Does React Work Behind The Sceen?
4  2) Understanding the Virtual DOM & DOM Updates
5  3) Understanding State & State Updates
6
7  How Does React Work?
8
9  React => "A javaScript Library for building user interfaces"
10
11 React => know only Components and state change nothing elase
12     React does not konw the HTML
13     if the component change then React change the State with previous on
14
15 ReactDOM => interface to the web (what the user sees)
16
17     Context
18     |
19 props => Components <= ReactDOM
20     |
21     State
22
23 ===== Re-Evaluating Components !== Re-Rendering the DOM =====
24
25 Components
26     1) Re-evaluated whenever props, state or context changes
27     2) React executes component functions
28
29 Real DOM
30     1) Changes to the real DOM are only made for differences between evaluations
31
32 ===== Virtual DOM Diffing =====
33 React work like this way
34     1) React manage a Virtual DOM copy of Actual DOM
35     2) If any component changes on the Virtual DOM then
36     3) React Understanding where the changes happened
37     4) then Changes only those component efeciently
38     5) use the Diffing those to DOM and made changes to
39
40 ===== Parent component re Run for State change =====
41
42     1) All child components will be Re Run Again
43     2) it will be cost some performance issue
44     3) we can be control re run for child components run again again
45
46 ===== React.memo(DemoOutput) for stopping unesesery component re run =====
47
48 By the help of memo() we can control re run for child components run again
49
50     1) when the parent component State change the it will run
51     2) if the => export default React.memo(DemoOutput) will chack that component props change or not
52     3) it the component DemoOutput props change then it will re Run again
53     4) the component DemoOutput Re Run. it's child component will also re Run again
54
55 */
56 // ===== Parent component =====
57
58
59 import React, { useState } from 'react';
60 import DemoOutput from './Demo/DemoOutput';
61
62 function App() {
63     const [showParagraph, setShowParagraph] = useState(false);
64     const showParagraphHandler = () =>{
65         setShowParagraph( (prevState) => !prevState);
66     }
67     return (
68         <React.Fragment>
69             <p>Hi, there!</p>
70             <DemoOutput show = {showParagraph} />

```

```

71     <button onClick={showParagraphHandler}>Click Me</button>
72   </React.Fragment>
73
74
75   );
76 }
77 export default App;
78
79
80   //===== child component =====
81
82 import React from 'react';
83
84 function DemoOutput(props){
85   return <p>{props.show ? "This is new" : ""}</p>
86 }
87 export default React.memo(DemoOutput);
88
89
90 /*
91   ===== useCallback() hook =====
92
93   1) to prevent reCreate same function or object again and again when component re run
94   2) useCallback() create a function only one time and reference that function to utilize the function
95   3) let obj1; let obj2; => obj1 = obj2 -> here same object create location indicate
96   4) useCallback() also do the same job for us
97 */
98 import React, {useCallback} from 'react';
99
100   const showParagraphHandler = useCallback( () =>{
101     setShowParagraph( (prevState) => !prevState);
102   } , [])
103 /*
104   here dependency not given that's why this function is created again and again
105
106   where as some use case that time when the dependency is changed that useCallback() re executed and new Function is created
107 */
108
109 import React, {useCallback} from 'react';
110
111 const showParagraphHandler = useCallback( () =>{
112   if(allowToggle){
113     setShowParagraph( (prevState) => !prevState);
114   }
115
116
117 } , [allowToggle])
118
119 const allowToggle = () =>{
120   setAllowToggle(true);
121 }
122
123 /*
124   when re run that time check that allowToggle value is change or not
125   if it is change then the new function will created
126
127 */
128
129
130 /* ===== useMemo() Hook =====
131
132   1) useMemo() hook use to memorize the array or other data which we don't want to re calculate
133   2) Like sort of array that when the component call again and again ... we can optimize that to store calculation
134   3) if new element come only that time calculated the sort data....
135   4) in this case useMemo() hook can help us to memorize
136
137   useMemo( function return , [])
138
139   return value which we want to store...
140 */
141 import React. {useMemo} from 'react':

```

```
141 import React, {useMemo} from 'react';
142
143 function App(){
144   const listItems = useMemo( () => [5, 3, 1, 10, 9], []);
145
146   return (
147     <DemoList items={listItems} />
148   );
149 }
150
151 }
152
153
154 // ..... Demo list child components
155
156 import React, {useMemo} from 'react';
157
158 function DemoList(props){
159
160   // props destructuring
161   const {items} = props;
162
163   const sortedList = useMemo( () =>{
164     return items.sort( (a, b) => a - b);
165   }, [])
166
167   return (
168     {sortedList.map(item) =>
169
170       ..... list reandering
171     }
172   );
173 }
```