**Documentation for TIC-TAC-TOE Game in ReactJS**

1. Mainly four components of ReactJS are used in this game:

* Components
* Props
* State
* Time Travel

1. First of all, the Board is build using HTML and CSS. And all the things are defined in the board Componnt.
2. All the data is passed among various components and functions using the **PROPS** .
3. Mainly there are two components:

* **Board.js**
* **Square.js**

IMPORTANT POINTS:

* Each component state is private, in our case we cannot change Board component state directly. Instead we will pass a function down to the square component from board component.
* JavaScript supports closures which means an inner function (e.g. handleClick) has access to variables and functions defined in a outer function (e.g. Board).

1. The **USESTATE** is used to manage the state within a component as it stores the state of each square to make it responsive to user interaction.
2. Overall game state is managed by lifting state up to the Board component.
3. A function is passed from the board component to the Square component which will handle the clicks.
4. The handleClick() function can read the squares state and call the setSquares method because they are both defined inside of the Board function.
5. We cannot directly pass the value to the handleClick() function as it will call the function again and again leading to an infinite loop.
6. Instead of defining different functions according to the values, we will iterate them using an arrow function.
7. State needs to be updated in the Board component to reflect changes in the game board.

Important:

* The DOM <button> element’s onClick attribute has a special meaning to React because it is a built-in component. For custom components like Square, the naming is up to you. You could give any name to the Square’s onSquareClick prop or Board’s handleClick function, and the code would work the same. In React, it’s conventional to use onSomething names for props which represent events and handleSomething for the function definitions which handle those events.

1. The concept of immutability is used in this game by performing all the operations on the copy of the array instead of modifying the original array. Copy is done using the **.slice()** method in the **handleClick()** function.
2. Immutability makes it easier to debug the code and to implement undo and redo functionalities easily.