






Build 25 React JS Projects in 10 Hours | React JS Course | React Interview Questions 2023

Generated on March 16, 2024

Summary

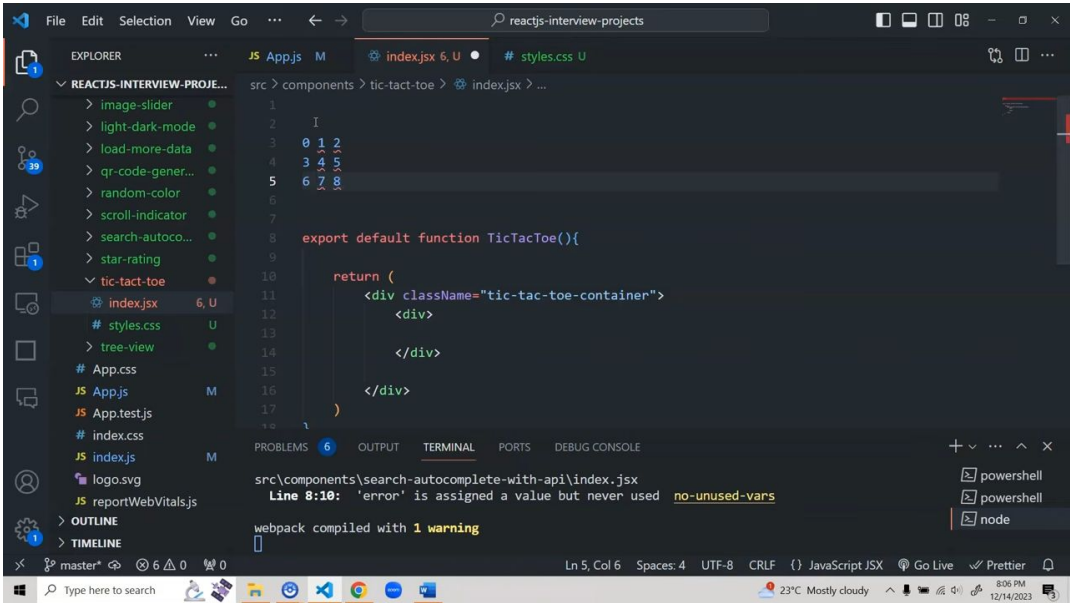
| Notes | Screenshots | Bookmarks |
|--|--|---|
|  30 |  40 |  0 |

we will create a component, and do the following. ss ss.

▶ 3:55:20

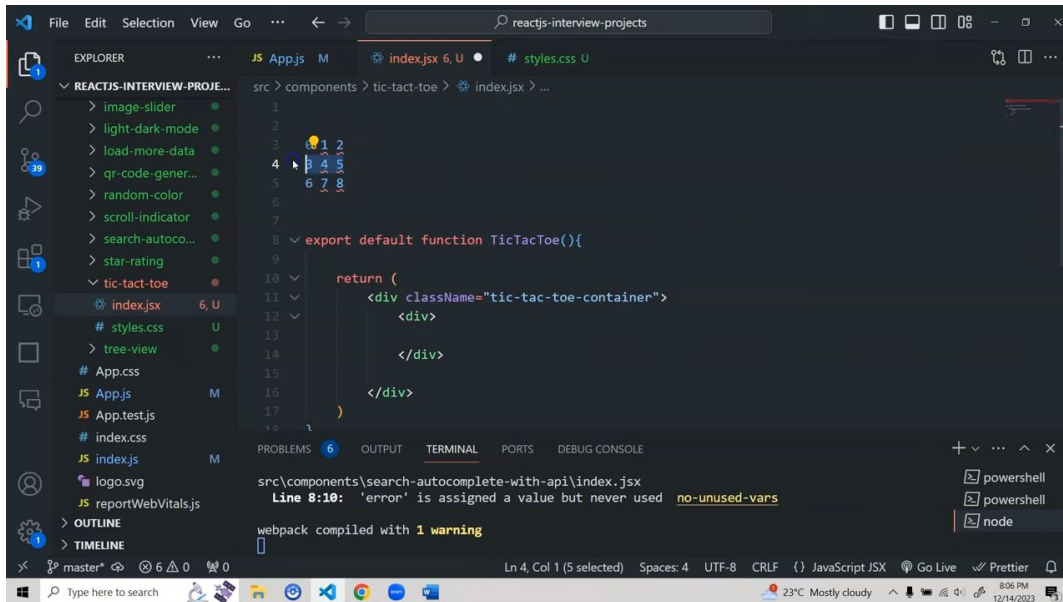
we basically have to create 9 boxes. each box will have an index. see ss.

▶ 3:55:52



✦ New logic for creating rows in a three-row system has been developed based on the sequence 0 1 2, 3 4 5, and 6 7 8.

▶ 3:55:52



```
1 2 3 4 5 6 7 8
2
3
4 1 2
5 3 4 5
6 6 7 8
7
8 export default function TicTacToe(){
9
10 return (
11 <div className="tic-tac-toe-container">
12 <div>
13 </div>
14 </div>
15 )
16 </div>
17 )
```

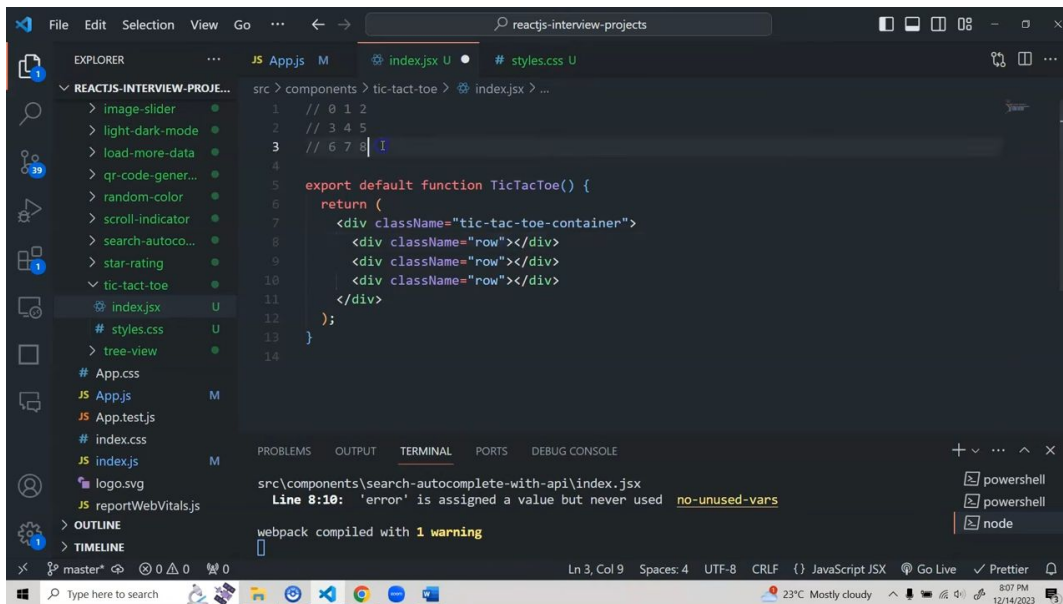
Line 8:10: 'error' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

▶ 3:55:54

since there are 3 rows we will create 3 divs. see ss.

▶ 3:56:04



```
1 // 0 1 2
2 // 3 4 5
3 // 6 7 8
4
5 export default function TicTacToe() {
6 return (
7 <div className="tic-tac-toe-container">
8 <div className="row"></div>
9 <div className="row"></div>
10 <div className="row"></div>
11 </div>
12 );
13 }
14
```

Line 8:10: 'error' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

✨ New component "Square" created to display three rows of three squares each for a new feature in the app.

▶ 3:56:25

```
src > components > tic-tac-toe > index.jsx > Square
3 // 6 7 8
4
5
6 function Square({value}){
7   return <button>{value}</button>
8 }
9
10
11 export default function TicTacToe() {
12   return (
13     <div className="tic-tac-toe-container">
14       <div className="row"></div>
15       <div className="row"></div>
16       <div className="row"></div>
17     </div>
18   );
19 }
20
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

src\components\search-autocomplete-with-api\index.jsx
Line 8:10: 'error' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

✦ Developing a new component to render value from the parent component.

▶ 3:56:52

```
src > components > tic-tac-toe > index.jsx > TicTacToe
14 <div className="row">
15   <Square/>
16   <Square/>
17   <Square/>
18 </div>
19 <div className="row">
20   <Square/>
21   <Square/>
22   <Square/>
23 </div>
24 <div className="row"></div>
25 </div>
26 );
27 }
28
```

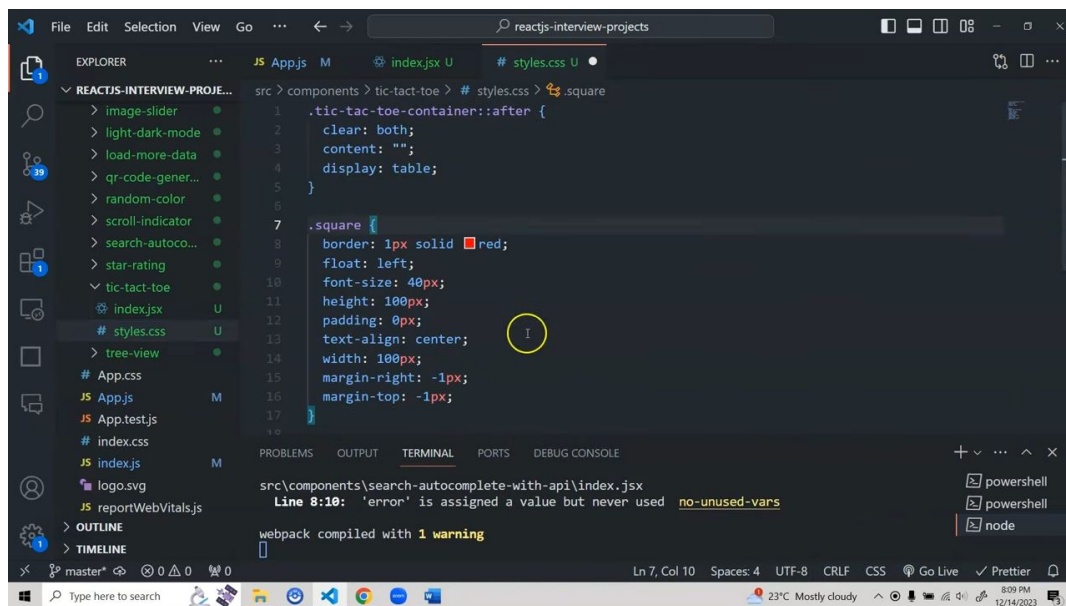
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

src\components\search-autocomplete-with-api\index.jsx
Line 8:10: 'error' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

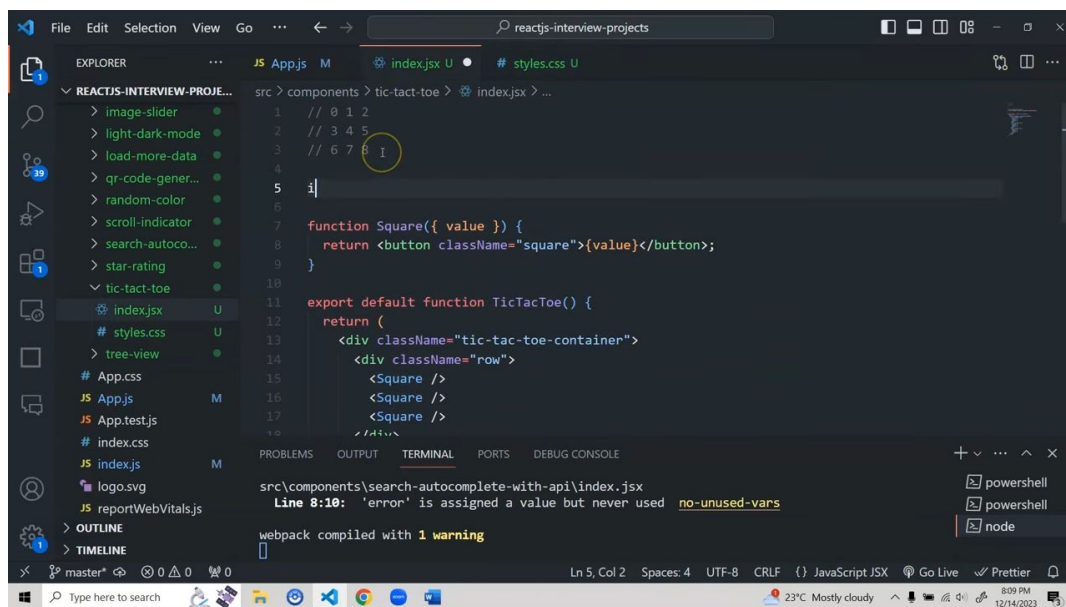
✦ A tutorial on creating a grid of squares using CSS and HTML.

▶ 3:57:12



✦ New styles and cursor changes have been implemented in the latest update of the website.

▶ 3:58:55



▶ 3:59:02

Imagine you have a tic-tac-toe board, and you can either start with putting a cross (X) or a zero (O) in any square. Each square on the board can be empty, have a cross (X), or have a zero (O).

When you click on a square to put a cross or a zero, you need to remember which square you clicked and what symbol (cross or zero) you put there. We use something called "state" to

remember this information.

So, for each square on the board, we need a space or a state to remember if it's empty, has a cross, or has a zero. When you click on a square, we update its state in the state of the game. For example, if you click on the first square and put a cross, we update the state of that square to have a cross.

▷ 4:00:37

either we can start with cross or zero. also, for each and every squares there will be some space or state we need to take so that we can fill that later. let say we are clicking in the first box with cross, i have to set square value for this particular element in the state.

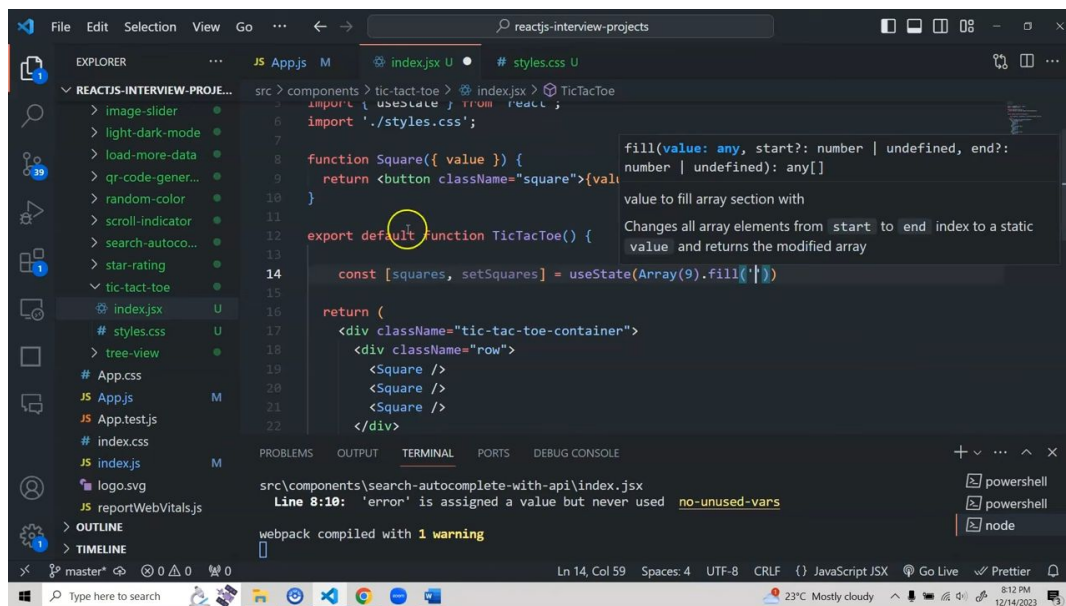
▷ 4:00:37

now, we have to decide that from which we have to start. x or zero. we will go with zero.

▷ 4:00:52

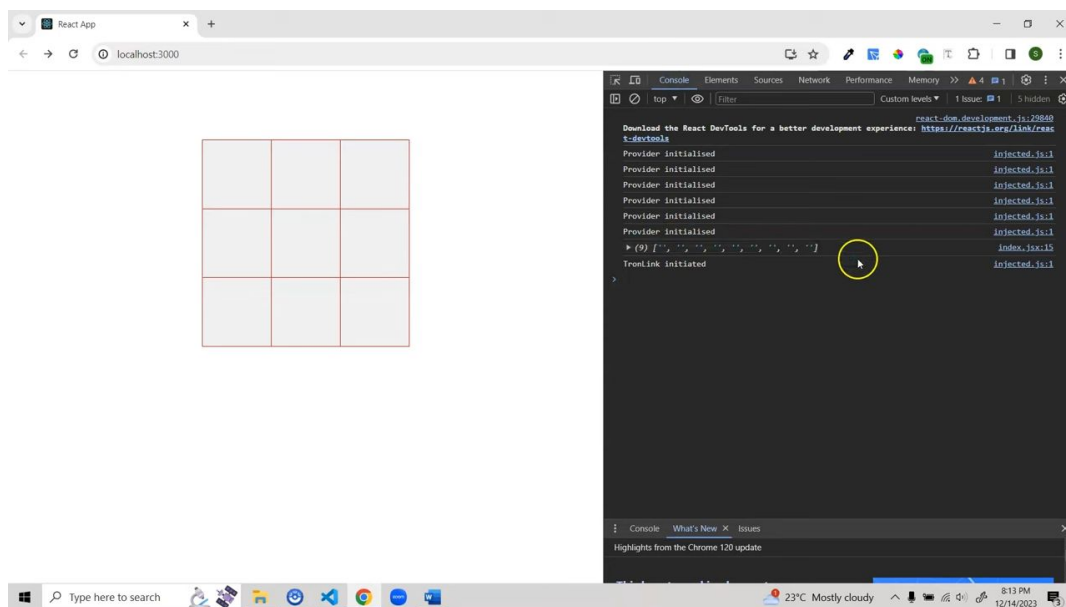
we need a state for each box, so that we can fill that later with either x or zero. for this, we will create a state, squares. in this state, we will take an array with length 9 (because we have 9 boxes) and fill them with empty string. see ss.

▷ 4:01:06



✦ A new method for filling a 9x9 square with empty strings is being introduced.

▶ 4:01:44

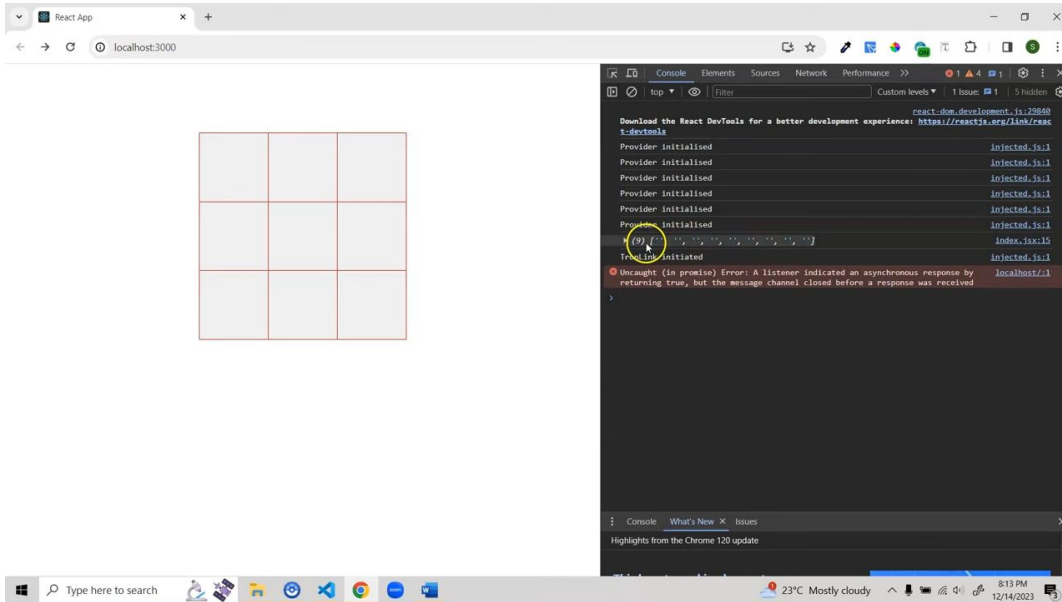


✦ A new console feature allows users to easily fill in elements with either a zero or a cross.

▶ 4:02:28

if i click on the first box, that means i have click on the 0 index.
so i have to fill that with cross. then, if i click on any other box,
its mean i have clicked on the any other index of square state
and i have to fill that with zero.

▶ 4:02:33



✦ A new interactive game has been developed that involves clicking on elements to determine a winner or a draw.

▶ 4:02:49

if there is a winner, i will display below it. if all the buttons are filled and there is no winner, i will display a message about draw. to display x or zero in the box, we will create another state. isXTurn and first time it will be true(becasue we are starting with x)

▶ 4:02:57

```
src > components > tic-tac-toe > index.jsx > TicTacToe
1 import { useState } from 'react';
2 import './styles.css';
3
4 function Square({ value }) {
5   return <button className="square">{value}</button>;
6 }
7
8 export default function TicTacToe() {
9
10  const [squares, setSquares] = useState(Array(9).fill(''));
11  const [isXTurn, setIsXTurn] = useState(true);
12
13  return (
14    <div className="tic-tac-toe-container">
15      <div className="row">
16        <Square />
17        <Square />
18        <Square />
19      </div>
20    </div>
21  );
22 }
```

src\components\tic-tac-toe\index.jsx
Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

✦ In a game of tic-tac-toe, the first turn will always be for X.

▶ 4:03:23

next, we will add an onClick to the each Square component, and write the onClick function. we will pass the onClick in the Square component, and add callback in the each Square component.

▶ 4:03:26

```
src > components > tic-tac-toe > index.jsx > Square
1 // 0 1 2
2 // 3 4 5
3 // 6 7 8
4
5 import { useState } from 'react';
6 import './styles.css';
7
8 function Square({ value, onClick }) {
9   return <button onClick={onClick} className="square">{value}</button>;
10 }
11
12 export default function TicTacToe() {
13
14  const [squares, setSquares] = useState(Array(9).fill(''));
15  const [isXTurn, setIsXTurn] = useState(true);
16
17  return (
18    <div className="tic-tac-toe-container">
19      <div className="row">
20        <Square />
21        <Square />
22        <Square />
23      </div>
24    </div>
25  );
26 }
```

src\components\tic-tac-toe\index.jsx
Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

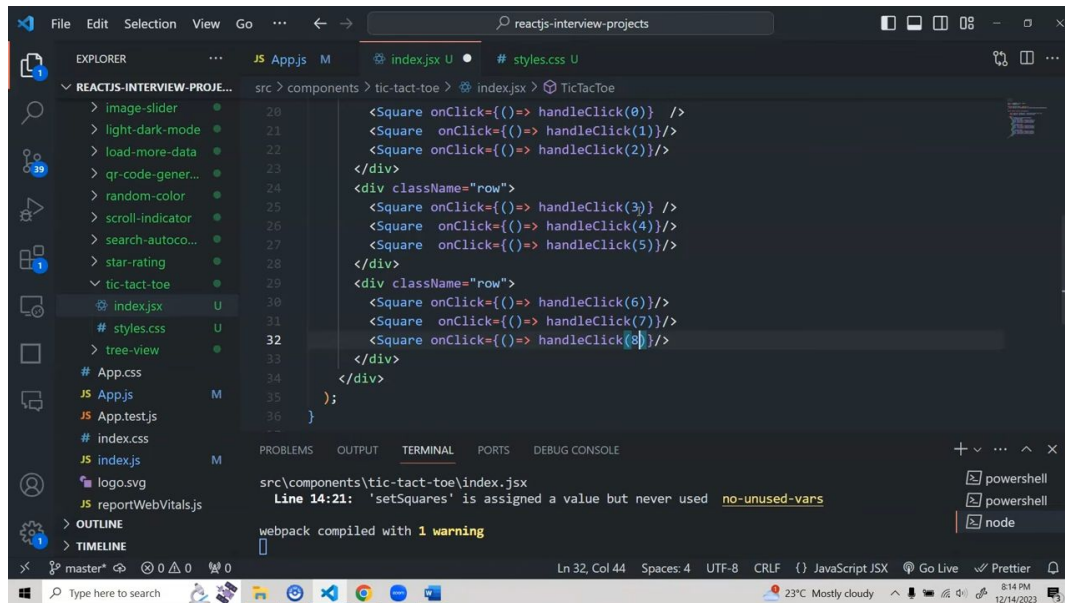
webpack compiled with 1 warning

✦ Developing a method to handle click events and pass them throughout the program is essential for efficient user interaction.

▶ 4:03:45

we will pass the index of square to the each handleClick.

▶ 4:03:55



```
20 <Square onClick={() => handleClick(0)} />
21 <Square onClick={() => handleClick(1)} />
22 <Square onClick={() => handleClick(2)} />
23 </div>
24 <div className="row">
25 <Square onClick={() => handleClick(3)} />
26 <Square onClick={() => handleClick(4)} />
27 <Square onClick={() => handleClick(5)} />
28 </div>
29 <div className="row">
30 <Square onClick={() => handleClick(6)} />
31 <Square onClick={() => handleClick(7)} />
32 <Square onClick={() => handleClick(8)} />
33 </div>
34 </div>
35 );
36 }
```

Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

✦ A new function called handle click is being created to assign identifiers to each square in a game.

▶ 4:04:21

now, we will write the code for function handleClick. pass the getCurrentSquare as parameter(which will represent index). since we have to fill the squares state, will create the copy of squares state using spread operator. then we will pass the index to the copy of squares state, and check if xTurn is true, insert x in that index of state, or insert zero. see ss

▶ 4:04:34

```
export default function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(""));
  const [isXTurn, setIsXTurn] = useState(true);

  function handleClick(getCurrentSquare){

    let cpySquares = [...squares];
    cpySquares[getCurrentSquare] = isXTurn ? 'X' : 'O';

  }

  return (
    <div className="tic-tac-toe-container">
      <div className="row">
        <Square onClick={() => handleClick(0)} />
        <Square onClick={() => handleClick(1)} />
        <Square onClick={() => handleClick(2)} />
      </div>
    </div>
  );
}
```

Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

✦ The process of determining whether it is X or O's turn in a game of squares involves checking and reversing the current state.

▶ 4:05:17

since the default state of isXTurn is true, and we have to insert zero when it is false, we will turn the state. means if it is true, make it false and vice versa.

▶ 4:05:17

```
function handleClick(getCurrentSquare){

  let cpySquares = [...squares];
  cpySquares[getCurrentSquare] = isXTurn ? 'X' : 'O';
  setIsXTurn(!isXTurn);

}

return (
  <div className="tic-tac-toe-container">
    <div className="row">
      <Square onClick={() => handleClick(0)} />
      <Square onClick={() => handleClick(1)} />
      <Square onClick={() => handleClick(2)} />
    </div>
  </div>
);
```

Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

webpack compiled with 1 warning

▶ 4:05:29

at the third line of the function, set the state of squares.

▶ 4:05:35

```
export default function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(""));
  const [isXTurn, setIsXTurn] = useState(true);

  function handleClick(getCurrentSquare) {
    let cpySquares = [...squares];
    cpySquares[getCurrentSquare] = isXTurn ? "X" : "O";
    setIsXTurn(!isXTurn);
    setSquares(cpySquares);
  }

  return (
    <div className="tic-tac-toe-container">
      <div className="row">
        <Square onClick={() => handleClick(0)} />
        <Square onClick={() => handleClick(1)} />
        <Square onClick={() => handleClick(2)} />
      </div>
    </div>
  );
}
```

Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

✨ In a game of tic-tac-toe, players take turns to place their X or O in the squares until one player wins.

▶ 4:05:37

now, we have to pass the value, you can see that we are DE structuring the value in the Square component. we will pass the value in the parent component like this. see ss.

▶ 4:05:41

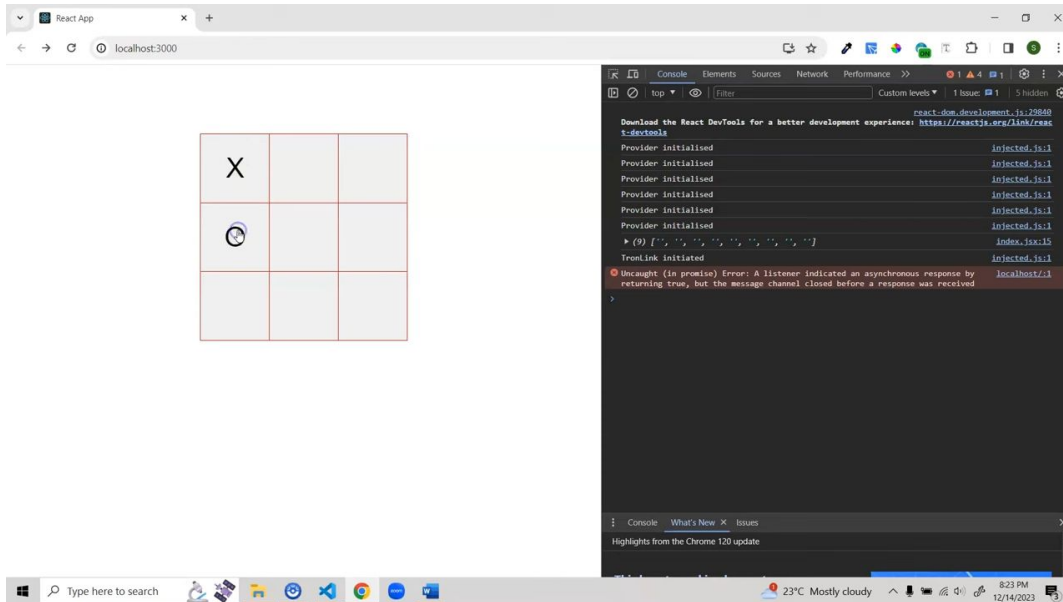
```
return (
  <div className="tic-tac-toe-container">
    <div className="row">
      <Square value={squares[0]} onClick={() => handleClick(0)} />
      <Square value={squares[1]} onClick={() => handleClick(1)} />
      <Square value={squares[2]} onClick={() => handleClick(2)} />
    </div>
    <div className="row">
      <Square value={squares[3]} onClick={() => handleClick(3)} />
      <Square value={squares[4]} onClick={() => handleClick(4)} />
      <Square value={squares[5]} onClick={() => handleClick(5)} />
    </div>
    <div className="row">
      <Square value={squares[6]} onClick={() => handleClick(6)} />
      <Square value={squares[7]} onClick={() => handleClick(7)} />
      <Square value={squares[8]} onClick={() => handleClick(8)} />
    </div>
  </div>
);
```

Line 14:21: 'setSquares' is assigned a value but never used no-unused-vars

✨ Dynamic allows users to create and pass values dynamically in order

to see real-time changes.

▶ 4:06:16



▶ 4:06:24

you can see, x and zero is appearing by clicking in the square box. now, if i click on the filled square, its value is getting changed. this should not happen. so, we have to check if the square is full with the value, do not change the value, we have to return. so in the second line of handle click, if copy of squares state has the index, (means it is clicked) return it.

▶ 4:06:33


```
export default function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(""));
  const [isXTurn, setIsXTurn] = useState(true);

  function handleClick(getCurrentSquare) {
    let cpySquares = [...squares];
    if(cpySquares[getCurrentSquare]) return;
    cpySquares[getCurrentSquare] = isXTurn ? "X" : "O";
    setIsXTurn(!isXTurn);
    setSquares(cpySquares);
  }

  return (
    <div className="tic-tac-toe-container">
      <div className="row">
        <Square value={squares[0]} onClick={() => handleClick(0)} />
        <Square value={squares[1]} onClick={() => handleClick(1)} />
        <Square value={squares[2]} onClick={() => handleClick(2)} />
      </div>
    </div>
  );
}
```

Line 8:10: 'error' is assigned a value but never used no-unused-vars

✦ A new rendering feature has been implemented to check for the value of a current index.

▶ 4:07:05

now, we have to calculate the winner. for this, we will create another function, pass the squares state as argument. now, we have to determine the winning patterns. create a const and store all the winning patterns. see ss. **winningPatterns** is an array of arrays in JavaScript.

▶ 4:07:30

```
export default function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(""));
  const [isXTurn, setIsXTurn] = useState(true);

  function getWinner(squares){
    const winningPatterns = [
      [0,1,2],
      [3,4,5],
      [6,7,8],
      [2,5,8],
      [0,4,8],
      [2,4,6],
      [0,3,6],
      [1,4,7]
    ];
  }
}
```

Line 8:10: 'error' is assigned a value but never used no-unused-vars

✦ A new winning pattern has been identified in the latest lottery draw, offering players a chance to win big.

▷ 4:09:00

now, we will add a for loop, its length will be equal to the length of winning pattern array. and de structure these three values of each array inside.

▷ 4:09:19

- **winningPatterns** is an array of arrays. Each inner array represents a winning pattern in a tic-tac-toe game.
- The **for** loop iterates over each winning pattern in **winningPatterns**.
- In each iteration, the inner array at index **i** of **winningPatterns** is destructured into three variables: **x**, **y**, and **z**.
- The destructuring assignment **[x, y, z]** means that the first element of the inner array is assigned to **x**, the second element to **y**, and the third element to **z**.
- So, in each iteration of the loop, **x**, **y**, and **z** represent the indices of three squares in a tic-tac-toe grid that form a winning pattern.

For example, in the first iteration:

- **x** will be assigned the value of the first element of the first inner array in **winningPatterns**.
- **y** will be assigned the value of the second element.
- **z** will be assigned the value of the third element.

The last line of code is a destructuring assignment that assigns the elements of the inner array at index **i** of **winningPatterns** to the variables **x**, **y**, and **z**.

▷ 4:09:33

```
src > components > tic-tac-toe > index.jsx > TicTacToe > getWinner
> image-slider 22 [0, 1, 2],
> light-dark-mode 23 [3, 4, 5],
> load-more-data 24 [6, 7, 8],
> qr-code-gener... 25 [2, 5, 8],
> random-color 26 [0, 4, 8],
> scroll-indicator 27 [2, 4, 6],
> search-autoco... 28 [0, 3, 6],
> star-rating 29 [1, 4, 7],
> tic-tac-toe 30 ];
index.jsx 31
# styles.css 32 for(let i =0 ;i <winningPatterns.length; i++){
33     const [x,y,z] = winningPatterns[i]
34 }
35
# App.css 36
JS App.js 37 function handleClick(getCurrentSquare) {
JS App.test.js 38     let cpySquares = [...squares];
# index.css
JS index.js
logo.svg
reportWebVitals.js
OUTLINE
TIMELINE
master* 0 0 0
Ln 33, Col 43 Spaces: 4 UTF-8 CRLF {} JavaScript JSX Go Live Prettier
```

🌟 Researchers are working on deconstructing values to determine winning patterns in a new study on game theory.

▶ 4:09:33

This line of code is a conditional check within a tic-tac-toe game to determine if there is a winning combination of squares.

- `squares[x]`, `squares[y]`, and `squares[z]` represent the values of the squares at indices `x`, `y`, and `z`, respectively.

- The condition `squares[x] === squares[y] & & squares[x] === squares[z]` checks if the values of the squares at indices `x`, `y`, and `z` are all equal.

- If the condition is true, it means that the values of the squares in the current winning pattern are all the same, indicating that a player has filled those squares with their symbol (either "X" or "O").

- In that case, the function returns the value of any of the squares (since they are all the same in a winning pattern). This value represents the symbol ("X" or "O") of the player who has won the game by filling that particular winning pattern.

In summary, this line of code checks if the squares in a particular winning pattern have the same value, indicating a win for a player, and returns the symbol of that player.

▶ 4:10:31

```
src > components > tic-tact-toe > index.jsx > TicTacToe > getWinner
26  [0, 4, 6],
27  [0, 3, 6],
28  [1, 4, 7],
29  ];
30
31  for(let i = 0 ; i < winningPatterns.length; i++) {
32    const [x,y,z] = winningPatterns[i]
33
34    if(squares[x] && squares[x] === squares[y] && squares[x] === squares[z]) {
35      return squares[x]
36    }
37  }
38
39  function handleClick(getCurrentSquare) {
40
41  src\components\search-autocomplete-with-api\index.jsx
42  Line 8:10: 'error' is assigned a value but never used no-unused-vars
43  }
```

✦ A new algorithm has been developed to calculate the winner in a game based on specific conditions.

▶ 4:10:31

this will basically calculate the winner

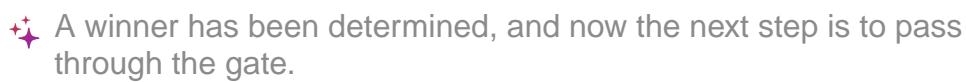
▶ 4:10:51

now, we will add a useEffect, and add its dependencies. see ss.

▶ 4:11:05

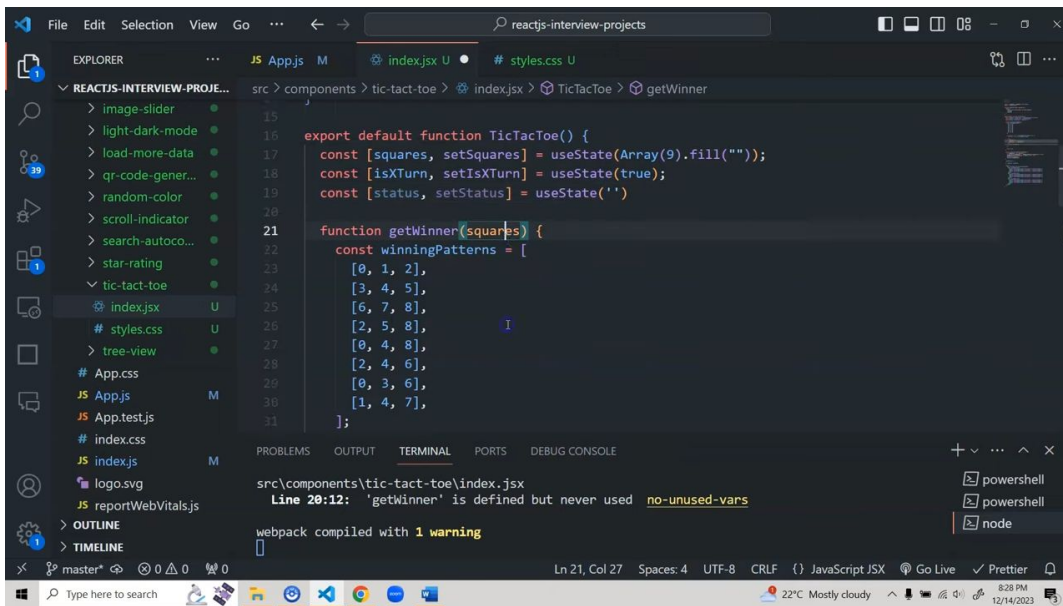
now, if x or zero has won and there are empty squares left, they should remain empty. by clicking on them, values should not be added in them. for this, we will add the following in the second line of handleClick. this line says, if the getWinner function has the cpySquares, return it. see ss.

▶ 4:11:18



now, to maintain the status, we will take another state. by default, it will be empty.

▶ 4:11:44



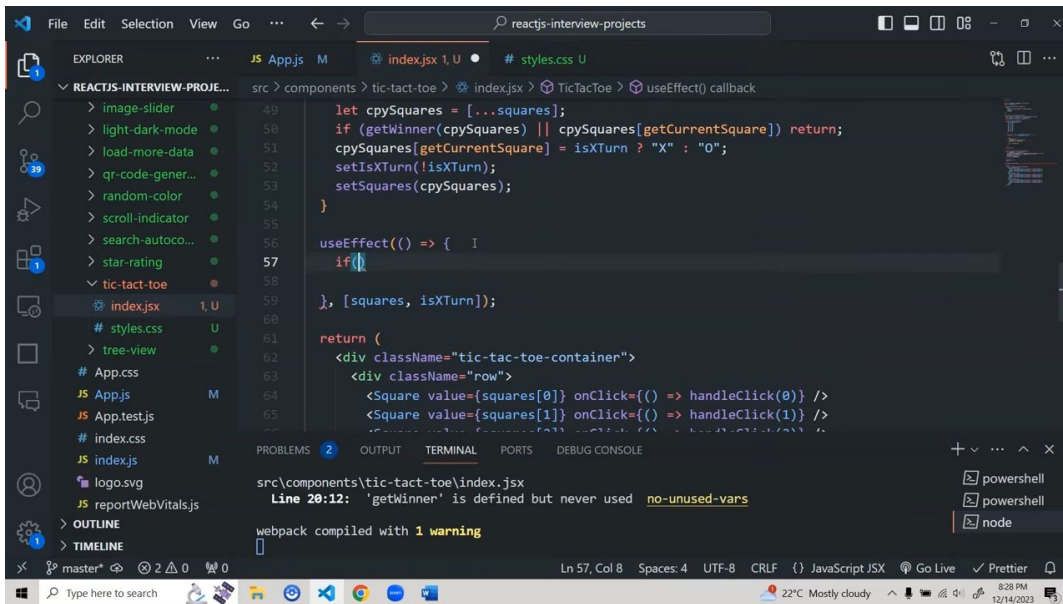
```
export default function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(""));
  const [isXTurn, setIsXTurn] = useState(true);
  const [status, setStatus] = useState('')

  function getWinner(squares) {
    const winningPatterns = [
      [0, 1, 2],
      [3, 4, 5],
      [6, 7, 8],
      [2, 5, 8],
      [0, 4, 8],
      [2, 4, 6],
      [0, 3, 6],
      [1, 4, 7],
    ];
  }
}
```

Line 20:12: 'getWinner' is defined but never used no-unused-vars

webpack compiled with 1 warning

▶ 4:11:45



```
let cpySquares = [...squares];
if (getWinner(cpySquares) || cpySquares[getCurrentSquare]) return;
cpySquares[getCurrentSquare] = isXTurn ? "X" : "O";
setIsXTurn(!isXTurn);
setSquares(cpySquares);
}

useEffect(() => {
  if()
}, [squares, isXTurn]);

return (
  <div className="tic-tac-toe-container">
    <div className="row">
      <Square value={squares[0]} onClick={() => handleClick(0)} />
      <Square value={squares[1]} onClick={() => handleClick(1)} />
    </div>
  </div>
)
```

Line 20:12: 'getWinner' is defined but never used no-unused-vars

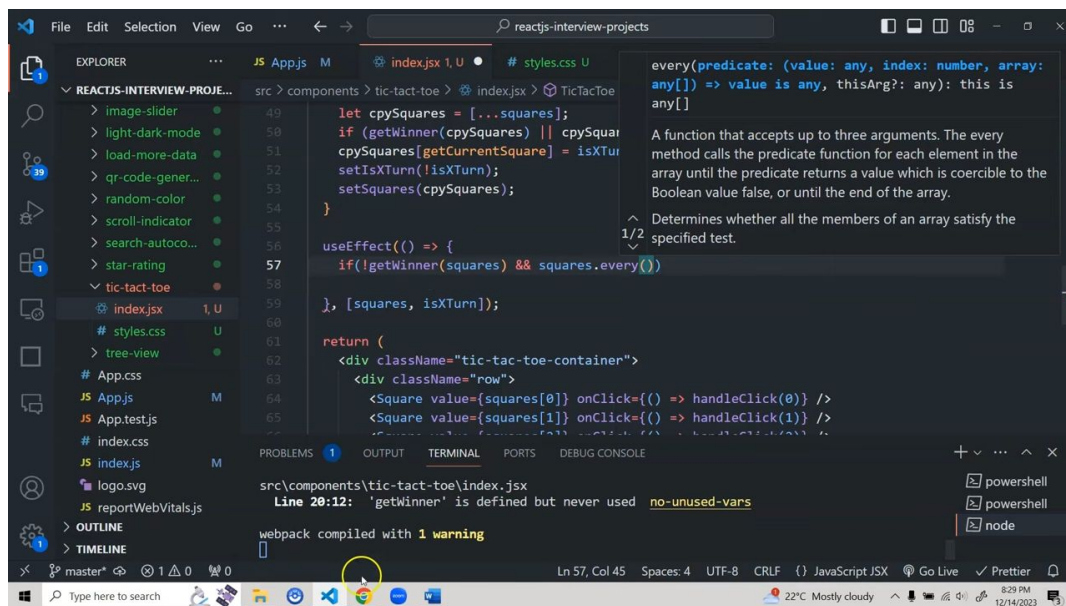
webpack compiled with 1 warning

✦ The status and use state of the empty squares will determine if there is a winner in the game.

▶ 4:11:52

in the useEffect, we will check if there is no winner, and every square is full, that means its draw.

▶ 4:11:56

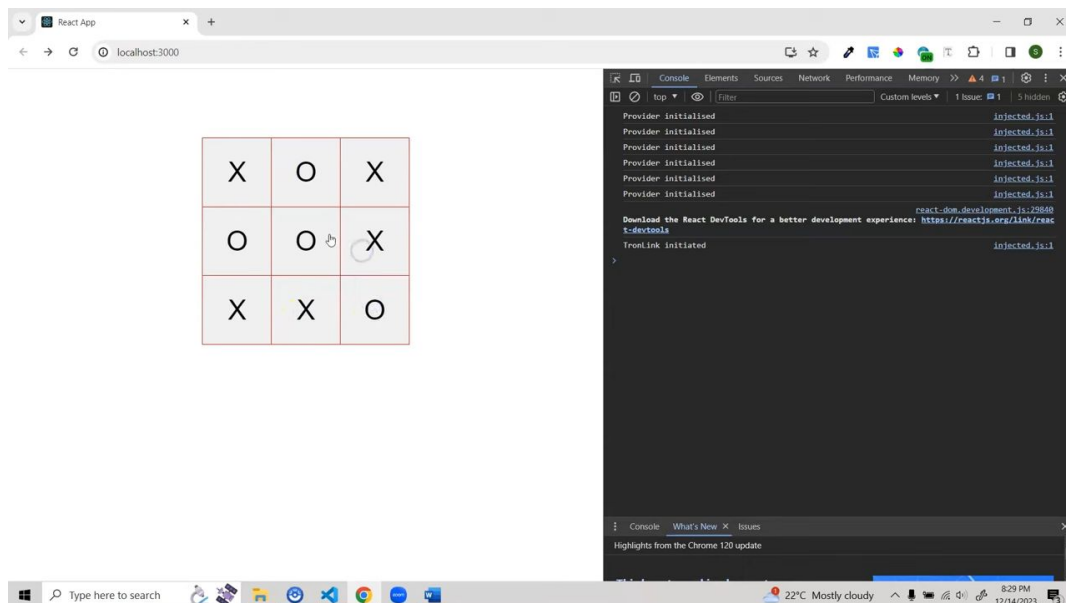


✦ In a game of tic-tac-toe, the winner is determined by three squares in a row being filled with the same symbol.

▶ 4:12:10

in this position, you can see no one is winner and all the squares are full. that's mean it is draw.

▶ 4:12:12



✦ In a new scenario, all squares are full and there is no winner, resulting in a draw.

▶ 4:12:30


```
useEffect(() => {
  if(!getWinner(squares) && squares.every(item=> item !== '')){
    setStatus('This is a draw ! Please restart the game')
  } else if(getWinner(squares)){
    setStatus('Winner is ${getWinner(squares)}')
  } else {
    setStatus(value: React.SetStateAction<string>): void
    setStatus('Next player is ${isXTurn ? 'X' : 'O'}')
  }
}, [squares, isXTurn]);

return (
  <div className="tic-tac-toe-container">
    <div className="row">
      <Square value={squares[0]} onClick={() => handleClick(0)} />
      <Square value={squares[1]} onClick={() => handleClick(1)} />
      <Square value={squares[2]} onClick={() => handleClick(2)} />
    </div>
    <div className="row">
      <Square value={squares[3]} onClick={() => handleClick(3)} />
      <Square value={squares[4]} onClick={() => handleClick(4)} />
      <Square value={squares[5]} onClick={() => handleClick(5)} />
    </div>
    <div className="row">
      <Square value={squares[6]} onClick={() => handleClick(6)} />
      <Square value={squares[7]} onClick={() => handleClick(7)} />
      <Square value={squares[8]} onClick={() => handleClick(8)} />
    </div>
    <h1>{status}</h1>
  </div>
);
```

Line 20:12: 'getWinner' is defined but never used no-unused-vars

✦ In a game of tic-tac-toe, the next player's move will determine whether X or O is rendered on the board.

▶ 4:14:01

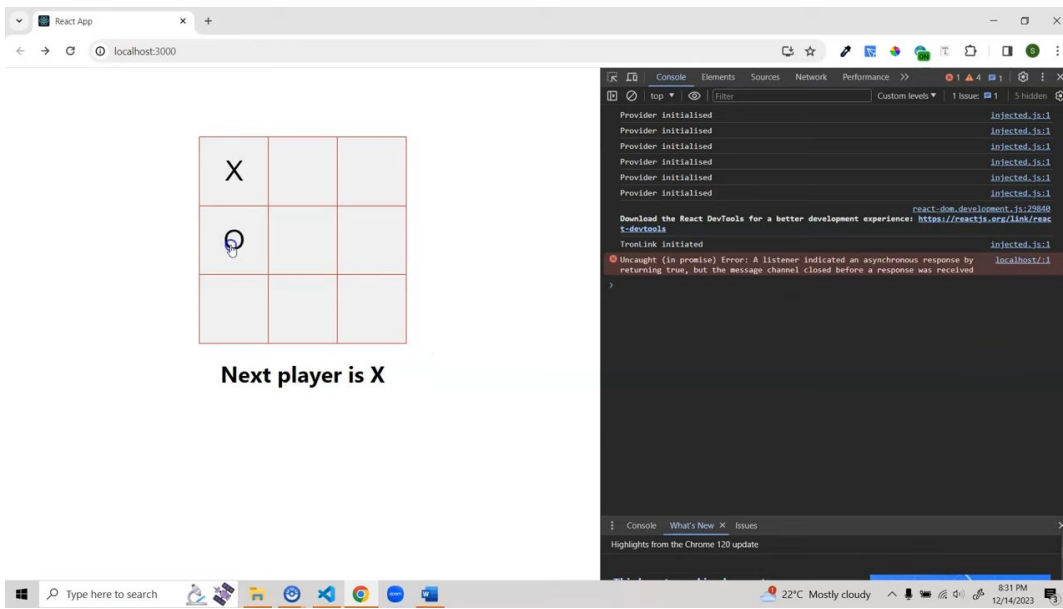
then, if no one is winner yet and it is not draw, that means the game is running. in this situation, set status like this. it will show that whose turn is now. see ss.

▶ 4:14:03

```
<div className="row">
  <Square value={squares[3]} onClick={() => handleClick(3)} />
  <Square value={squares[4]} onClick={() => handleClick(4)} />
  <Square value={squares[5]} onClick={() => handleClick(5)} />
</div>
<div className="row">
  <Square value={squares[6]} onClick={() => handleClick(6)} />
  <Square value={squares[7]} onClick={() => handleClick(7)} />
  <Square value={squares[8]} onClick={() => handleClick(8)} />
</div>
<h1>{status}</h1>
</div>
);
```

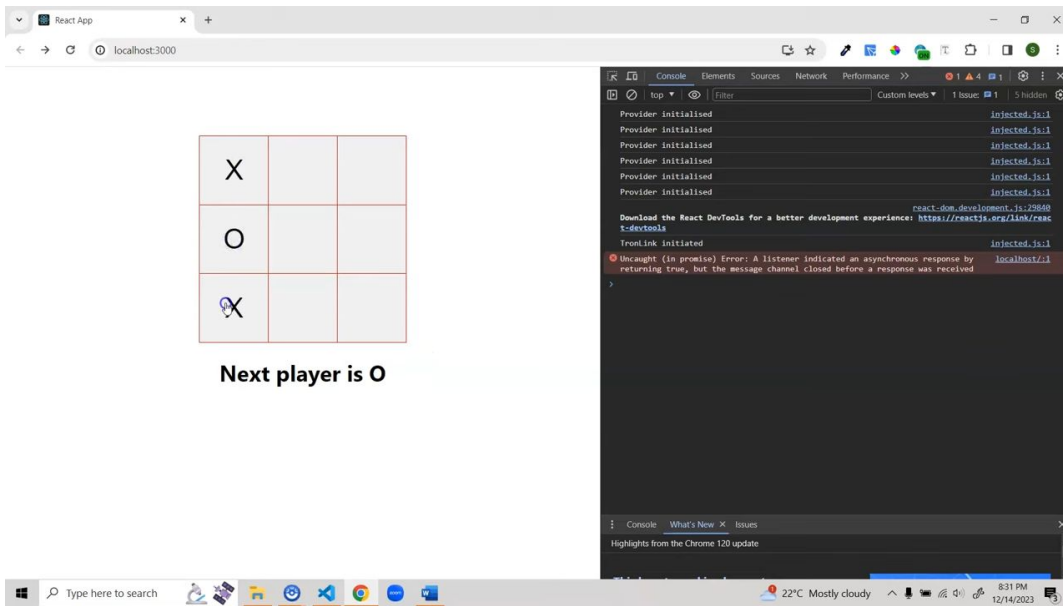
Line 20:12: 'getWinner' is defined but never used no-unused-vars

▶ 4:14:10

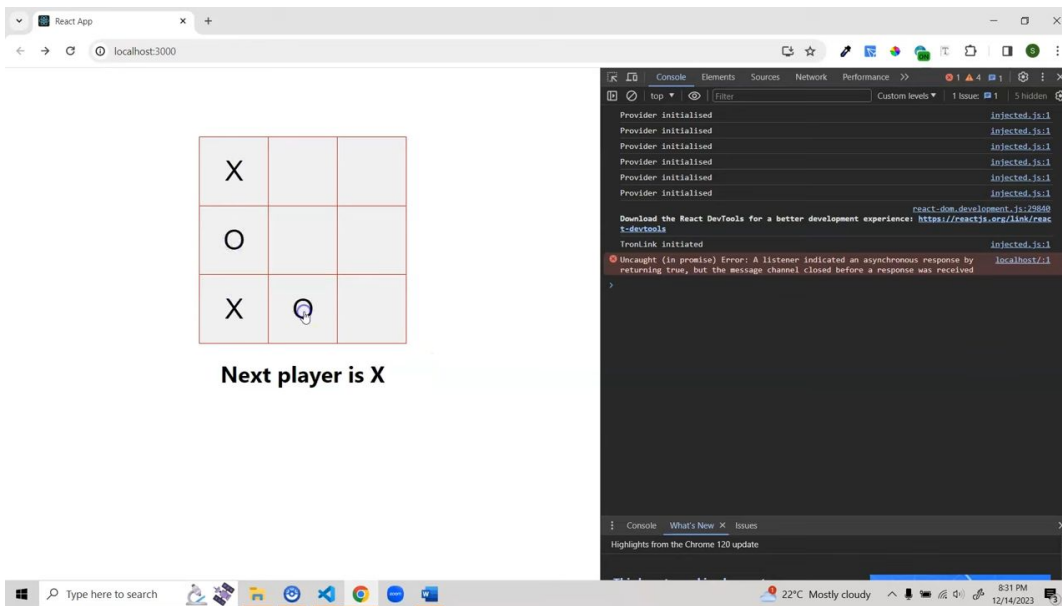


✨ X wins the game in a demonstration of the new rendering feature.

▶ 4:14:17



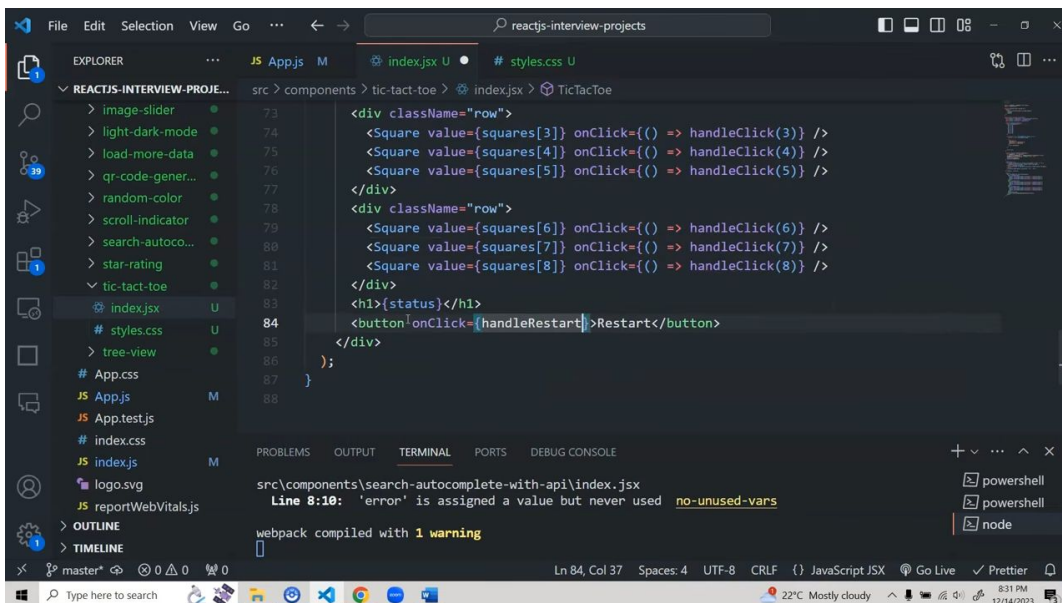
▶ 4:14:18



▶ 4:14:20

now we will add a button that will restart the game.

▶ 4:14:54

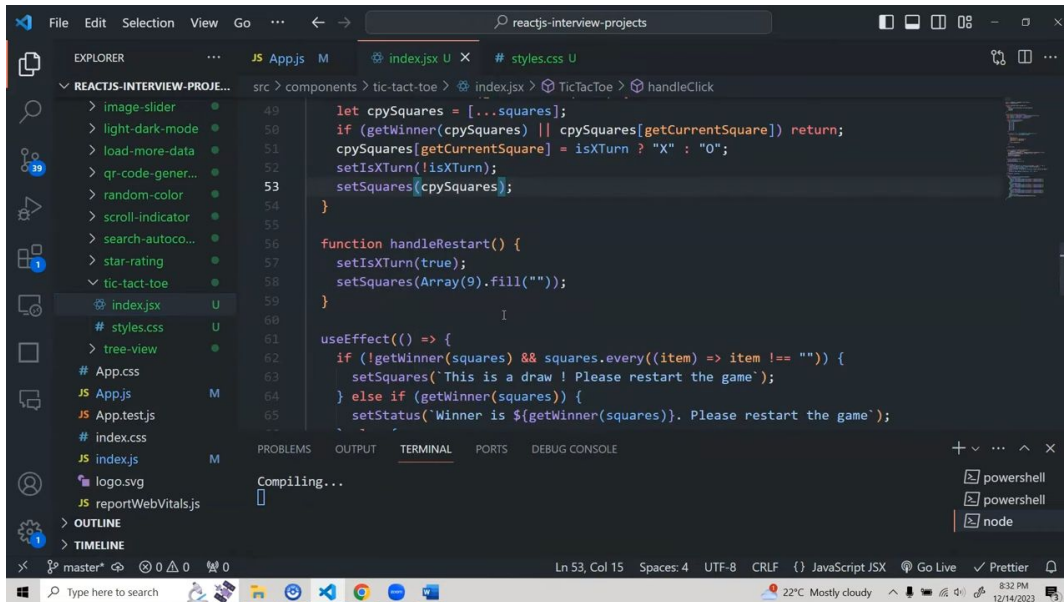


✨ The game will be restarted with a simple button to handle the restart process.

▶ 4:14:57

in the onClick function, we will set the xturn state true, then setSquares will be at its default position.(where it is array with length 9 and each index have empty string)

▶ 4:15:20



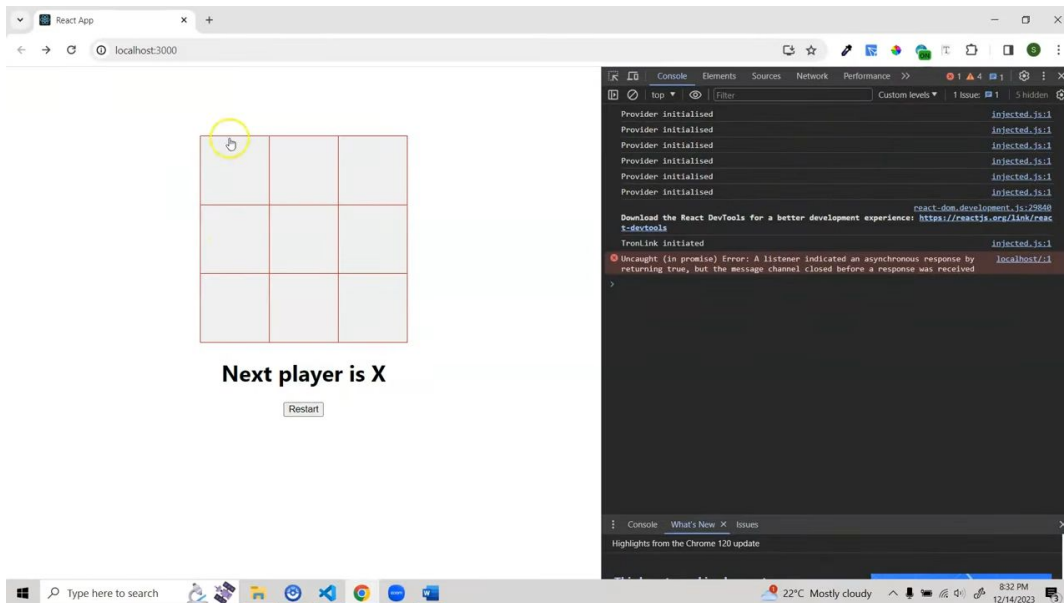
```
let cpySquares = [...squares];
if (getWinner(cpySquares) || cpySquares[getCurrentSquare] return;
cpySquares[getCurrentSquare] = isXTurn ? "X" : "O";
setIsXTurn(!isXTurn);
setSquares(cpySquares);
}

function handleRestart() {
  setIsXTurn(true);
  setSquares(Array(9).fill(""));
}

useEffect(() => {
  if (!getWinner(squares) && squares.every((item) => item !== "")) {
    setSquares('This is a draw ! Please restart the game');
  } else if (getWinner(squares)) {
    setStatus('Winner is ${getWinner(squares)}. Please restart the game');
  }
}
```

✨ In order to restart the game, first ensure that the set is X turn and then fill the array of squares with empty values.

▶ 4:15:20



▶ 4:15:25

React App

localhost:3000

| | | |
|---|---|---|
| X | | |
| O | O | O |
| X | X | |

Winner is O. Please restart the game

Restart

Console

Provider initialised
Provider initialised
Provider initialised
Provider initialised
Provider initialised
Provider initialised
TronLink initiated
Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>
Uncaught (in promise) Error: A listener indicated an asynchronous response by returning true, but the message channel closed before a response was received

▶ 4:15:31

React App

localhost:3000

| | | |
|--|--|--|
| | | |
| | | |
| | | |

Next player is X

Restart

Console

Provider initialised
Provider initialised
Provider initialised
Provider initialised
Provider initialised
Provider initialised
TronLink initiated
Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>
Uncaught (in promise) Error: A listener indicated an asynchronous response by returning true, but the message channel closed before a response was received

▶ 4:15:33