/* App.jsx of TicTacToe */

```
import React, { useState, useEffect } from 'react';
import './App.css';
import Confetti from 'react-confetti';

const App = () => {
  const [board, setBoard] = useState(Array(9).fill(null));
  const [isXNext, setIsXNext] = useState(true);
  const [showConfetti, setShowConfetti] = useState(false);

  const winnerInfo = calculateWinner(board);
  const isDraw = !winnerInfo && board.every((square) => square !== null);

  const handleClick = (indexa) => {
    if (board[index] || winnerInfo || isDraw) return;
    const newBoard = [...board];
    newBoard[index] = isXNext ? 'X' : 'O';
    setBoard(newBoard);
    setIsXNext(!isXNext);
  };

  const resetGame = () => {
    setBoard(Array(9).fill(null));
    setIsXNext(true);
    setShowConfetti(false); // Reset confetti state
  };

  // Use useEffect to show confetti when there's a winner
  useEffect(() => {
    if (winnerInfo) {
      setShowConfetti(true);
```

```
  } else {
    setShowConfetti(false); // Ensure confetti is hidden if there's no winner
  }
}, [winnerInfo]); // Run this effect when winnerInfo changes

return (
  <div className="App">
    {showConfetti && <Confetti />}
    <h1>Tic Tac Toe</h1>
    <div className="board">
      {board.map((value, index) => (
        <Square
          key={index}
          value={value}
          onClick={() => handleClick(index)}
          isWinningSquare={winnerInfo && winnerInfo.line && winnerInfo.line.includes(index)}
        />
      ))}
    </div>
    <div className="info">
      {winnerInfo ? (
        <h2>{`Winner: ${winnerInfo.winner}`}</h2>
      ) : isDraw ? (
        <h2>It's a Draw!</h2>
      ) : (
        <h2>{`Next Player: ${isXNext ? 'X' : 'O'}`}</h2>
      )}
      <button onClick={resetGame}>Reset Game</button>
    </div>
  </div>
);
```

```
};

const Square = ({ value, onClick }) => (
  <button className="square" onClick={onClick}>
    {value}
  </button>
);

const calculateWinner = (board) => {
  const lines = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6],
  ];

  for (const line of lines) {
    const [a, b, c] = line;
    if (board[a] && board[a] === board[b] && board[a] === board[c]) {
      return { winner: board[a], line }; // Return both winner and winning line
    }
  }

  return null;
};

export default App;
```

/* App.css of TicTacToe */

```
.App {
  text-align: center;
  font-family: Arial, sans-serif;
  background-color: #121212;
  color: #ffffff;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  margin-left: 500px;



}

.board {
  display: grid;
  grid-template-columns: repeat(3, 10rem);
  grid-gap: 5px;
  justify-content: center;
  align-items: center;
}

.square {
  width: 10rem;
  height: 10rem;
  font-size: 2rem;
  font-weight: bold;
  cursor: pointer;
  background-color: #1e1e1e;
```

```
  border: 2px solid #333;
  color: #ffffff;
  transition: background-color 0.2s ease;
}

.square:hover {
  background-color: #333;
}

.square.winning {
  background-color: #4caf50;
  border-color: #388e3c;
}

.info {
  margin-top: 20px;
}

button {
  padding: .625rem;
  font-size: 1rem;
  background-color: #333;
  color: #ffffff;
  border: 2px solid #555;
  cursor: pointer;
  border-radius: .3125rem;
}

button:hover {
  background-color: #555;
}
```

```
button[disabled] {
  background-color: #555;
  cursor: not-allowed;
}

h1 {
  color: #e0e0e0;
}

h2 {
  color: #b0b0b0;
}

/* Confetti styles */
.confetti {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  pointer-events: none;
}
```