Explanation for the winner calculations

Since we just copied our winner calculation to our game, this file will be explaining the process (this is **not required** thing to learn, it’s just here for the people that are interested in how the calculation works).

const checkWinner = () => {

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 (let i = 0; i < lines.length; i++) {

        const [a, b, c] = lines[i];

        if (gameState[a] && gameState[a] === gameState[b] && gameState[a]

=== gameState[c]) {

            return gameState[a];

        }

    }

    return null;

}

The checkWinner function checks for a winner in a Tic-Tac-Toe game where there are exactly 3 x 3 squares.

For reference here is a 3 x 3 Tic-Tac-Toe board with each square numbered with their index

**| 0 | 1 | 2 |**

**| 3 | 4 | 5 |**

**| 6 | 7 | 8 |**

The lines array has all the unique possible lines on a 3 x 3 board. Line being three variables in a row (when you win a game).

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],

        ];

**All the duplicate lines are removed** from the array. For example, these two lines would be **duplicates**:

Line one: 0 -> 2

Line two: 2 -> 0

There is no need to check both lines because if a winning combination (three **X’s** or three **O’s**) is on line one, then the winning combination has to be on line two as well.

So, the lines array has all the unique line combinations. We then **loop through each of them** and check if all the squares (variables **a**, **b** and **c**) have the same value (for example, three **X**’s):

for (let i = 0; i < lines.length; i++) {

    const [a, b, c] = lines[i];

    if (gameState[a] && gameState[a] === gameState[b] && gameState[a]

=== gameState[c]) {

        return gameState[a];

    }

}

return null;

If at any point this condition is **true**, then we return the value that is stored in the gameState at the first square (**return gameState[a]** in the code).

If we go through all the lines and the winner is never found, then we return **null** (A draw or the game is still in progress).

**Hope this cleared the process to the ones that are interested!** 😊