Hints 3: Determining Game Status

To do this, you can write a function that checks the values of each of the squares that have been played in. If any three in a row have the same value, then the player associated with that symbol is the winner. If there is no winner and all of the spaces have a symbol, then it is a tie. That's the general idea.

Please go back and read the Hints for Requirement 2 to understand the following variables that have been declared in the game.

- `currentPlayerSymbol`: the symbol of the player that has the next click
- `squareValues`: an array that contains the symbols for each of the squares

With those variables declared (or ones like them), add code that will perform the following functionality.

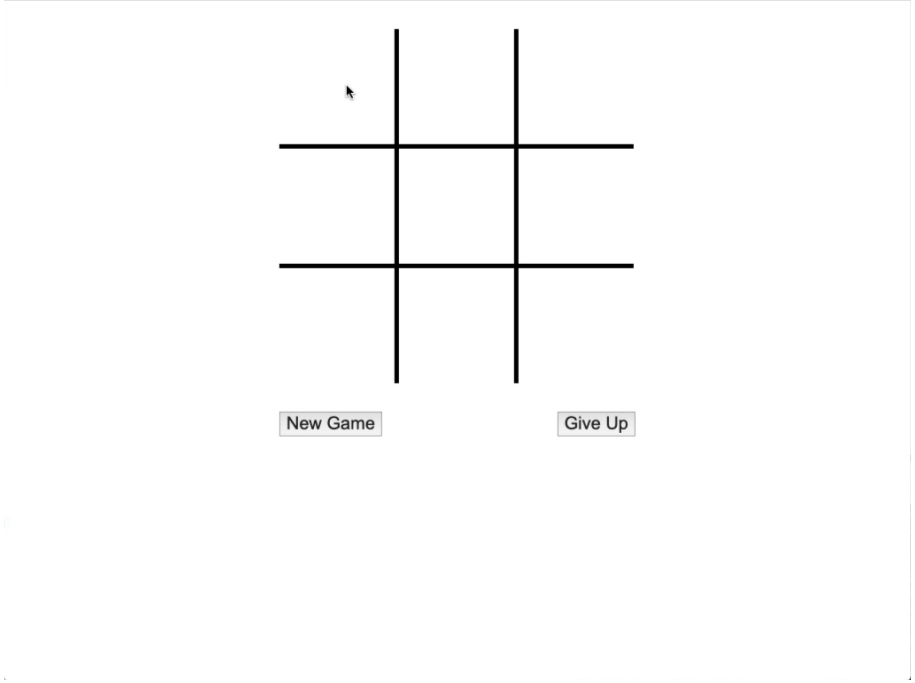
In the JavaScript file:

- Create a top-level variable named `gameStatus` and set it to the empty string.
- Create a top-level function named `checkGameStatus` that will loop through all of the rows and columns, and check the diagonals to determine if there is a winner. In it, perform these checks
    - If there is a winner, set the `gameStatus` value to the uppercase value of the value that is winning symbol.
    - If there is no winner, then check if all of the squares are full. If they are, then set the `gameStatus` equal to "None".
    - If the `gameStatus` is not an empty string, set `document.getElementById('game-status').innerHTML` to the concatenated value of "Winner: " and the value of `gameStatus`. Use the `id` value of `game-status` to get a reference to that element.

- Add a check at the beginning of the click handler for the grid that prevents the rest of the event handler from running if the `gameStatus` value is not an empty string. (This is as simple as adding an `if` statement that tests for it and has a `return` statement inside the curly braces.)

```
if (gameStatus !== '') return; // This just stops
                               // the rest of the function
                               // from running.
```

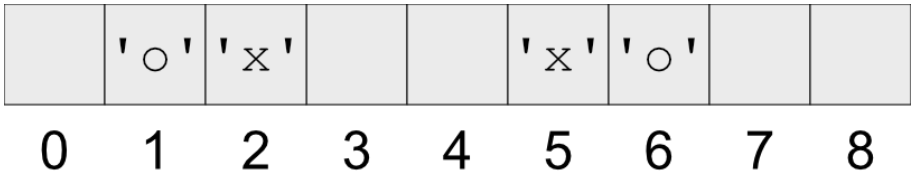
- At the end of the of the click handler for the grid to call the `checkGameStatus` function.

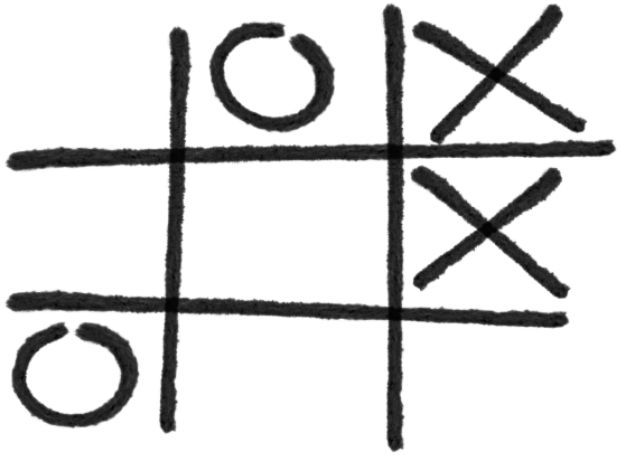


How to find a winning game

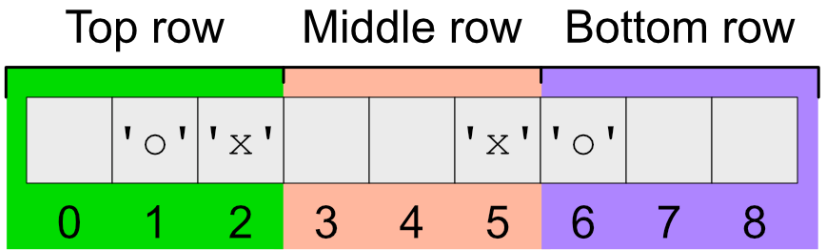
Consider using a one-dimensional array with nine entries to store your representation of the board. Then, you can map each of the nine entries to a square on the board kind of like what you see in the following picture where the

0-index entry in the array is the top-left square of the board, the 1-index entry in the array is the top-middle square, the 2-index entry is the top-righ, the 3-index entry is the middle-left, and so on.





If that's the model that you have, then you can check the rows of the board for a win by comparing the first three entries of the array with one another, the second three entries with one another, and the third three entries with one another. Here's the breakdown of the rows of the tic-tac-toe-board mapped to the entries in the array.

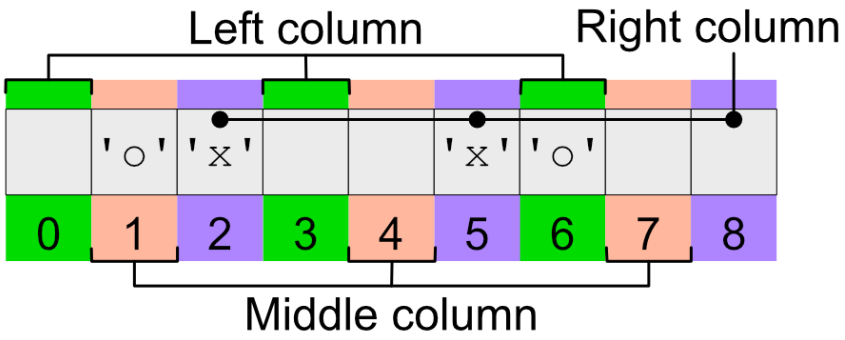


Assume the name of your array that holds the board values is named `board`. You can determine if the first row has a winning combination of entries by doing the following comparison:

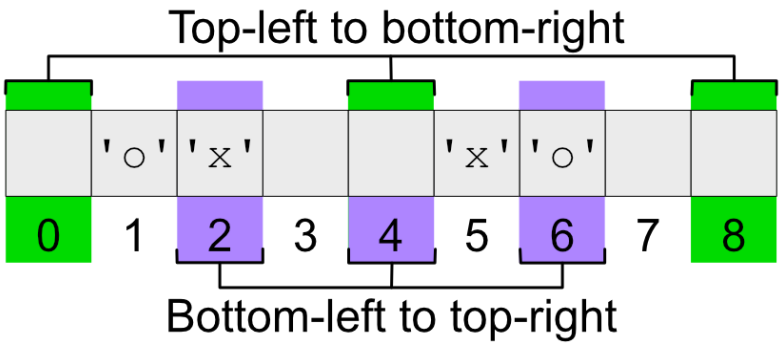
```
if (board[0] === board[1] && board[1] === board[2] && board[2] !== '') {
  // do something because a player won on the first row!
}
```

You can just do the same for the other two rows, too.

To find the columns in the array, you can refer to the following image.



Finally, to find the diagonals, you need to map squares 0, 4, and 9 for the top-left to bottom right and the 2, 4, 6 squares for the top-right to the bottom-left like in the following picture.


Top-left to bottom-right
'o'
'x'
'x'
'o'
0
1
2
3
4
5
6
7
8
Bottom-left to top-right