

W03-P1: Implement checkWin(player) using three different cases

=> player o wins

The screenshot displays a web browser window with a Tic-tac-toe game titled "Tictactoe -- htc". The game board is a 3x3 grid. The first column contains three green 'o' characters, indicating a win for player 'o'. The second column contains a green 'o', a red 'x', and a green 'o'. The third column contains a red 'x', a red 'x', and a green 'o'. Below the board is a "Reset Game" button. The browser's developer tools are open, showing the source code of the game. The code defines a `checkWin` function that checks for a win by player 'o' or 'x' based on the current state of the board. The function is called with the player 'o' as an argument, and the result is logged to the console.

```

1  const o = 'o';
2  const x = 'x';
3  let turn = 0;
4  let done = false;
5
6  const container = document.querySelector('.container');
7  const alertBox = document.querySelector('.alert');
8  const allLi = document.querySelectorAll('.board li');
9  const resetBtn = document.querySelector('.reset');
10
11 console.log(alertBox);
12 console.log('allLi', allLi);
13
14 const checkWin = (player) => {
15   let p = [];
16   allLi.forEach((item) => {
17     p.push(item.classList.contains(player));
18   });
19   console.log('p', p);
20   const [p1, p2, p3, p4, p5, p6, p7, p8, p9] = p;
21   if(
22     (p1 && p2 && p3) ||
23     (p4 && p5 && p6) ||
24     (p7 && p8 && p9) ||
25     (p1 && p4 && p7) ||
26     (p2 && p5 && p8) ||
27     (p3 && p6 && p9) ||
28     (p1 && p5 && p9) ||
29     (p3 && p5 && p7)
30   ) return true;
31   else return false;
32 };
33
34 > const winMessage = (player) => { ...
48 };
49
50 > const tieMessage = () => { ...
55 };
56
57 console.log('checkWin('o')', checkWin('o'));
58
59 const reset = () => {

```

=> player x wins

Tictactoe Demo

127.0.0.1:5500/demo/w03_tictactoe_26/tictactoe_26.html

Tictactoe -- htc

O

+

X

O

X

+

X

+

O

Reset Game

Console

Failed to load :5500/favicon.ico:1 resource: the server responded with a status of 404 (Not Found)

1132-2N-DEMO-26

tictactoe_26.js

demo > w03_tictactoe_26 > JS tictactoe_26.js > ...

14 const checkWin = (player) => {

18 };

19 console.log('p',p);

20 const [p1, p2, p3 ,p4 ,p5 ,p6 ,p7 ,p8 ,p9] = p;

21 if(

22 (p1 && p2 && p3) ||

23 (p4 && p5 && p6) ||

24 (p7 && p8 && p9)||

25 (p1 && p4 && p7)||

26 (p2 && p5 && p8)||

27 (p3 && p6 && p9)||

28 (p1 && p5 && p9)||

29 (p3 && p5 && p7)

30) return true;

31 else return false;

32 }

33

34 > const winMessage = (player) => { ...

48 };

49

50 > const tieMessage = () => { ...

55 };

56

57 console.log(' checkWin('o')',checkWin('o'));

58 console.log(' checkWin('x')',checkWin('x'));

59

60 const reset = () => {

61 alertBox.style.display = 'none';

62 container.style.backgroundColor = '#666';

63 allLi.forEach((item) => {

64 item.textContent = '+';

65 item.classList = '';

66 });

67 };

68

69 resetBtn.addEventListener('click', reset);

70

71

72

73 /*

74 <ul class="board">

75 +

=> no player wins

Tictactoe Demo

127.0.0.1:5500/demo/w03_tictactoe_26/tictactoe_26.html

Tictactoe -- htc

O	X	O
O	O	X
X	O	X

Reset Game

Console

▶ div.alert

tictactoe_26.js:11

allLi

tictactoe_26.js:12

▶ NodeList(9)

[li.o, li.x, li.o, li.o, li.o, li.x, li.x, li.o, li.x]

▶ (9)

[true, false, true, true, true, false, false, true, false]

checkWin('o')

false

tictactoe_26.js:57

▶ (9)

[false, true, false, false, false, true, true, false, true]

checkWin('x')

false

tictactoe_26.js:58

1132-2N-demo-26

tictactoe_26.js

9 const resetBtn = document.querySelector('.reset');

10

11 console.log(alertBox);

12 console.log('allli',allli);

13

14 const checkWin = (player) => {

15 let p = [];

16 allli.forEach((item)=>{

17 p.push(item.classList.contains(player));

18 });

19 console.log('p',p);

20 const [p1, p2, p3 ,p4 ,p5 ,p6 ,p7 ,p8 ,p9] = p;

21 if(

22 (p1 && p2 && p3) ||

23 (p4 && p5 && p6) ||

24 (p7 && p8 && p9)||

25 (p1 && p4 && p7)||

26 (p2 && p5 && p8)||

27 (p3 && p6 && p9)||

28 (p1 && p5 && p9)||

29 (p3 && p5 && p7)

30) return true;

31 else return false;

32 };

33

34 > const winmessage = (player) => { ...

48 };

49

50 > const tieMessage = () => { ...

55 };

56

57 console.log('`checkWin('o')`,checkWin('o')`);

58 console.log('`checkWin('x')`,checkWin('x')`);

59

60 const reset = () => {

61 alertBox.style.display = 'none';

62 container.style.backgroundColor = '#666';

63 allli.forEach((item) => {

64 item.textContent = '+';

65 item.classList = '';

66 });

67 };

6dba4ca 1131-sweb-demo21341032 Sun Mar 9 18:07:39 2025 +0800 Implement checkWin(player) using 1

W03-P2: play TicTacToe successfully

=> player o wins

The screenshot shows a web browser window displaying a TicTacToe game titled "Tictactoe -- htc". The game board is a 3x3 grid. The current state is "Player o wins". The board contains the following symbols:

X	+	O
X	O	+
O	+	+

The VS Code editor shows the JavaScript code for the game. The code is in a file named "tictactoe_26.js". The code includes a "reset" function and a "go" function. The "go" function is highlighted with a red box. The code logic for the "go" function is as follows:

```
const go = (item, player, text) => {
  item.textContent = text;
  //item.classList = 'o disabled';
  item.classList.add(player, 'disabled');
  if (checkWin(player)) {
    winMessage(player);
    done = true;
  }
};

allLi.forEach((item) => {
  item.addEventListener('click', () => {
    if (item.classList.contains('disabled')) {
      alert('already filled');
    } else {
      if (turn % 2 === 0) {
        go(item, 'o', 'o');
      } else if (turn % 2 === 1) {
        go(item, 'x', 'x');
      }
    }

    if (!done && turn < 8) {
      turn++;
    } else if (!done && turn >= 8) {
      tieMessage();
    }
  });
});

resetBtn.addEventListener('click', reset);
```

=> player x wins

The image displays a web browser window on the left and a VS Code editor on the right, both showing a Tic-tac-toe game interface.

Browser Window (Left):

- Address bar: 127.0.0.1:5500/demo/w03_tictactoe_26/tictactoe_26.html
- Page Title: Tictactoe -- htc
- Game State: Player x wins
- Game Board (3x3 grid):
 - Row 1: +, +, X
 - Row 2: O, O, X
 - Row 3: O, +, X
- DevTools is now available in Chinese! (Notification)
- Elements Panel: Shows the HTML structure of the game board, including the 3x3 grid of cells and the footer.
- Styles Panel: Shows the computed styles for the selected element, including margin, border, and padding.

VS Code Editor (Right):

- File Explorer: Shows the project structure with files tictactoe_26.js, w03.md, and tictactoe_26.css.
- Code Editor: Displays the JavaScript code for the game, including the reset function, the go function, and the reset button click event listener.

=> tie

The screenshot displays a web browser window on the left and a VS Code editor on the right. The browser shows a page titled "Tictactoe -- htc" with a 3x3 grid. The grid contains 'X' and 'O' pieces, and a red box highlights the center cell, which is empty. Above the grid, the text "Tie" is displayed. Below the grid, there is a "Reset Game" button. The VS Code editor shows the source code for the game. The file explorer on the left lists files like "w01_dom_26", "w02_dom_26", "w03_dom_26", "w03-p1-1.png", "w03-p1-2.png", "w03-p1-3.png", "w03-p2-1.png", "w03-p2-2.png", "w03.md", "note", "w01_dom_26", "w02_dom_26", "w03_tictactoe_26", "tictactoe_26.css", "tictactoe_26.js", and "tictactoe_26.js". The main editor shows the JavaScript code for the game. The code includes functions for checking win conditions, displaying messages, and resetting the game. A red box highlights the "tieMessage" function, which is called when the game is a tie. The code also includes a "reset" function and a "go" function. The "go" function is called when the user clicks on a cell. The "go" function checks if the cell is empty and if the game is not over. If the cell is empty and the game is not over, the "go" function updates the cell with the current player's piece and checks for a win or tie. If there is a win, the "go" function displays a message and sets the "done" flag to true. If there is a tie, the "go" function displays a message and sets the "done" flag to true. If the cell is not empty or the game is over, the "go" function does nothing.

```
const tieMessage = () => {
  showAlert.style.backgroundColor = '#888';
  showAlert.style.color = '#ddd';
  showAlert.style.display = 'block';
  showAlert.textContent = 'Tie';
};

const reset = () => {
  showAlert.style.display = 'none';
  container.style.backgroundColor = '#666';
  allLi.forEach((item) => {
    item.textContent = '+';
    item.classList = '';
  });
  turn = 0;
  done = false;
  showAlert.style.display = 'none';
};

const go = (item, player, text) => {
  item.textContent = text;
  //item.classList = 'o disabled';
  item.classList.add(player, 'disabled');
  if (checkWin(player)) {
    winMessage(player);
    done = true;
  }
};

allLi.forEach((item) => {
  item.addEventListener('click', () => {
    if (item.classList.contains('disabled')) {

```

6dba4ca 1131-sweb-demo21341032 Sun Mar 9 18:07:39 2025 +0800

Implement checkWin(player)