Project Instructions for “Four in a Row”

Open the project files in your favorite text editor and preview them in the browser. Also, it’s a smart idea to test your project in multiple browsers!

This game is a lot like Tic Tac Toe, except that it’s played on a larger grid and the objective is to get four in a row, instead of three. It’s played on a vertical board with a grid of spaces. Tokens are moved left and right, and dropped with the left, right and down arrow keys. On each turn, the player has a token that they move left or right along the board, eventually dropping it into the desired column of spaces. To see what the game looks like in action, see the attached screenshots.

In order to win, the player must be the first to get *four of their tokens in a row*, horizontally, vertically, or diagonally.

Be sure to properly comment your code; explain the purpose of the method, what the parameters are, and what the method returns. Below are some examples:

```
102     /**
103      * Checks if there a winner on the board after each token drop.
104      * @param   {Object}    target - Targeted space for dropped token.
105      * @return  {boolean}   Boolean value indicating whether the game has been won (true) or not (false)
106      */
107     checkForWin(target){
108         const owner = target.token.owner;
109         let win = false;
110
```

```
18      /**
19       * Creates two player objects
20       * @return  {array}    An array of two player objects.
21       */
22      createPlayers() {
23          const players = [new Player('Player 1', 1, '#e15258', true),
24                           new Player('Player 2', 2, '#e59a13')];
25          return players;
26      }
27
```

- Start out by getting familiar with the game and plan on how you want to organize your code.
  - Inside the js directory, create the app.js file.
  - Inside the js directory, create the following files (be sure to capitalize the names of the files) :
    - Player.js
    - Token.js
    - Board.js
    - Space.js
    - Game.js
  - Inside each of the files, declare the associated class.

Here's an example on how to create a class in JavaScript:

```
1   class Game {
2       constructor() {
3           this.board = new Board();
4           this.players = this.createPlayers();
5           this.ready = false;
6       }
7   }
```

Tutorial on how to create classes in JavaScript: https://www.youtube.com/watch?v=2ZphE5HcQPQ

- Recommended methods for the Player.js file:
  - **Constructor method.**
  - **createTokens()**
    - Creates token objects for player.
  - **unusedTokens()**
    - Gets all tokens that haven't been dropped.
  - **activeToken()**
    - Gets the active token by returning the first token in the array of unused tokens.
  - **checkTokens()**
    - Check if a player has any undropped tokens left.

- Recommended methods for the Token.js file:
  - **Constructor method**.
  - **htmlToken()**
    - Gets the associated htmlToken.
  - **offsetLeft()**
    - Gets the left offset of html element.
  - **drawHTMLToken()**
    - Draws new HTML token.
  - **moveLeft()**
  - **moveRight()**
  - **drop()**
    - Drops html token into targeted board space.

- Recommended methods for the Board.js file:
  - **Constructor method**.
  - **createSpaces()**
    - Generates 2D array of spaces.

- **drawHTMLBoard()**
  - Draws the associated SVG spaces for all game spaces.

- Recommended methods for Space.js file:
  - **Constructor Method.**
  - **owner()**
    - Checks if space has an associated token to find its owner.
    - Returns either null or the owner object of the space's associated token.
  - **drawSVGSpace()**
  - **mark()**
    - Updates space to reflect a token has been dropped into it.

- Recommended methods for Game.js file:
  - **Constructor Method.**
  - **activePlayer()**
    - Returns active player.
  - **createPlayers()**
    - Creates two player objects.
  - **startGame()**
    - Initializes the game.
  - **handleKeydown()**
    - Branches code, depending on what key the player presses.
  - **playToken()**
    - Finds space object to drop token into, drops the token.
  - **updateGameState()**
    - Updates game state after token is dropped.
  - **checkForWin()**
    - Checks if there is a winner on the board after each token drop.
  - **switchPlayers()**
    - Switches active player.

- **gameOver()**
  - Displays winner info.

- Recommended methods for app.js file:
  - Two different event listener methods:
    - One that listens for when the user clicks the “start” button to start the game
      - When the start button is clicked, the **startGame()** method in Game.js is called.
    - One that listens for when the user clicks the left, right, or down arrow keys on their keyboard.