Game Assignment

# Objective

You will create a Tic Tac Toe game using HTML, CSS, and JavaScript.

# Instructions

1. You must use the functions indicated.
2. You must use the module pattern, or some equivalent that allows private/public functions.
   1. You must choose which functions should be public or private correctly.
3. You must not directly copy any code or algorithms from the web, trust me I’ll know ;).
   1. You can however use other sources as reference to help you solve problems.
4. All JavaScript must be in external files.
5. Events cannot be added in the HTML, they must be unobtrusive.
6. You must submit a ZIP file in the following format: **lastname\_firstName\_studentNumber.zip**

## Welcome Screen

1. When the page loads, a welcome screen should be displayed asking the user if they would like to play a game.
2. There should be a button that says ‘I want to play’, it’s functionality should be in a function called **initGame:**
   1. When the button is pressed a check is done to see if users exist.
      1. If they don’t, in a function called **getUserInfo**:
         1. A prompt should come up asking for X’s player name.
         2. A prompt should come up asking for O’s player name.
            1. Player O’s name cannot be the same as player X’s name.
         3. A user object should be created for each user to store their information.
3. Once the user’s information has been gathered the game board should be created by calling **generateGameBoard**.

## Game Board/Screen

1. The game board should be generated using the JavaScript DOM api:
   1. There should be a function called **generateGameBoard** that is called from within the **initGame** function that creates the game board.
   2. You will have to decide the best way to create this board.
   3. It’s okay to use some hard-coded HTML.

## Game Functionality

1. Each round of the game, the **nextRound** function is called and will:
   1. Display the current user’s information (name) so they know whose turn it is.
   2. If this is the first round of the game the first player should be randomly picked.
      1. This functionality should be placed in a function called **getFirstPlayer**
2. When a user selects a square, the **selectSquare** function is called:
   1. A check is done to see if it’s been selected before
      1. If it has, an alert should inform the user to make a different selection
      2. If it hasn’t, the user’s symbol should be added to the square
   2. A check is done to see if the game is over using the function **checkGameState**
      1. If the game isn’t over, continue to the next round
      2. If the game is over, display the results (Who won, or if tie)
         1. Store the results of each user in their objects (win, loss, tie)
         2. Display a button to play another match (should keep old results)
            1. When clicked, **initGame** should be called

Hint: This time we already have the players.

* + - 1. Display a button that when clicked shows the results of all games played between the current users so far (how many wins, losses, and ties for each user) in a function called, **displayResults**.

1. Have a button that clears current game data and returns user’s to the welcome screen.
   1. This functionality should be in a function called **resetGame**

## Bonus

For bonus marks, create a single player game mode with some simple AI to try and beat the user!

## Deliverables

A zip folder containing all of your applications files.

## Function Details

|  |  |
| --- | --- |
| Function | Description |
| initGame | * Sets up the game * Calls getUserInfo * Calls generateGameBoard |
| getUserInfo | * Uses prompts to create user objects |
| generateGameBoard | * Uses the DOM API to create a game board |
| nextRound | * Displays relevant info for each round (current users/ symbol) * Changes current user |
| selectSquare | * Checks if square is selected * Selects a square |
| checkGameState | * Uses algorithm to check what state the game is in: (won, tied, unfinished) |
| displayResults | * Shows the results of all the games played so far (wins, losses, and ties). |
| resetGame | * Clears all game data and brings back the welcome screen |
| getFirstPlayer | * Randomly selects the first player of the game |

## Folder Structure

1. tictactoe
   1. Scripts
      1. [whatever script files you have]
   2. Styles
      1. [whatever css files you have]
   3. Index.html

## User Object Example

var user = {

name: ‘’,

gameResults: = [ ]

};

## Rubric

|  |  |
| --- | --- |
|  | Marks |
| Code is organized and proper formatting | 8 |
| Comments (useful, help understand the program) | 2 |
| Logical variable names (they need to make sense) | 5 |
| Functions used correctly (private, public, logic in correct function) | 10 |
| Logic is correct (game state works correctly, code functionality works as stated in the assignment, no global variables, etc.) | 10 |
| Front-end HTML and CSS look good and makes sense | 5 |
| Win/Tie algorithm works correctly | 10 |
| Total | 50 marks |
| Bonus marks (This would be super impressive!) | 5 |