```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Tic Tac Toe Game</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<!--
    Author: Professor
    Date created: Summer 2016
    Date updated: May 2019
    Version: 19.519.9
        FIRST VERSION: procedural unobtrusive Javascript, using console output
        SECOND VERSION: procedural unobtrusive Javascript, using DOM methods and
        unmodified HTML
        THIRD VERSION: functional unobtrusive Javascript, using DOM methods to create
        HTML
        FORTH VERSION: object-based unobtrusive Javascript, uinsg DOM methods to create
        HTML
    Purpose:
        Demonstrate the approach and rigor required in designing and implementing a web
        application. Start with documentation to provide the framework for coding the
        solution.
    NOTE!  THIS IS NOT A FINAL VERSION!
           You need to continue to improve it as you get closer to the complete
           version. Continue to refactor the code. Use TODO: DONE: TOFIX: FIXED: until
           the application is complete, tested, and ready for production.
           REMEMBER: there are many solutions and multiple ways of doing any single
           task. Write down all your observations and alternatives (which you can
           expand on and rearrange throughout the process) so that you can validate and
           verify you chose the appropriate one at some point.
    Copyright:
        This work is the intellectual property of Sheridan College. Any further copying
        and distribution outside of class must be within the copyright law. Posting to
        commercial sites for profit is prohibited.
    Objective:
        design a generic tic tac toe game (technology-gnostic) and then implement with
        HTML5, CSS3, and Javascript (client-side).
    Citations:
        [1] https://en.wikipedia.org/wiki/Tic-tac-toe
        [2] https://www.quora.com/Why-is-a-draw-game-in-tic-tac-toe-called-a-cats-game
        [3] Course Material, Section 7846x, SYST10199, Summer 2019, Sheridan College
    Description:
        A two-player game based on [1].
        The players are presented with a gameboard of three-by-three grid and a button
        to start a new game.
        Once the game starts, players take turn selecting one of the nine
        open squares (available or empty cells) until one player wins or the game is "a
        scratch" [2]. A player wins when they complete a row, a column, or a diagonal
        first. When a winning state occurs, a message is presented that the game is
        over and who the winner is.

        * A message is displayed, specifying which player's turn is presented during
        the game.
        * A gameboard representation used

            0 | 1 | 2
           ---+---+---
            3 | 4 | 5
           ---+---+---
            6 | 7 | 8

    Algorithm: see Readme.pdf file for earlier versions.

    Algorithm (pass N-1):
        (event-driven)
        1. Start game
        2. Select cell
        3. Check if winner
        3a.   if winner, End game
        3b.   if no winner and cells available, switch player
        4. Repeat 2-3 until no cell empty
        5. End game

    Algorithm (pass N):
        (event-driven)
        1. Start game
        2. Repeat until no cell empty
        3.   Select cell
        4.   Check if winner
        4a.    if winner, End game
        4b.    if no cells available, End game
        4c.    if no winner and cells available, switch player
        5. End game

        CAUTION:  x can complete a win combo using the last available cell

    Testing:
        create test cases...

        "When considering only the state of the board, and after taking into account
        board symmetries (i.e. rotations and reflections), there are only 138 terminal
        board positions. A combinatorics study of the game shows that when "X" makes
        the first move every time, the game is won as follows:
        91 distinct positions are won by (X)
        44 distinct positions are won by (O)
        3 distinct positions are drawn (often called a "cat's game")" [1]

            o |   | o          | o |            | o |
           ---+---+---      ---+---+---      ---+---+---
              |   | o        o | o |            | o |
           ---+---+---      ---+---+---      ---+---+---
              | o |            |   | o        o |   | o
-->
<style>
body {
    width: 680px;
    margin: 0 auto;
    text-align: center;
}
table {
     margin: auto;
}
td {
    border: 1px solid blue;
    height: 50px;
    width: 50px;
    font-size: 1.4em;
}
tr:first-child td {
    border-top: none;
}
tr:nth-child(3) td {
    border-bottom: none;
}
td:first-child {
    border-left: none;
}
td:last-child {
    border-right: none;
}
</style>
</head>
<body>
    <header><h1>Tic Tac Toe</h1></header>
    <table>
        <tr>
            <td>X</td>
            <td>O</td>
            <td>O</td>
        </tr>
        <tr>
            <td>X</td>
            <td>X</td>
            <td>O</td>
        </tr>
        <tr>
            <td>O</td>
            <td>O</td>
            <td>X</td>
        </tr>
    </table>
    <p>Player <span id="player">X</span> go!
        <button id="reset">Start A New Game</button>
    </p>
    <div id="message"></div>
    <footer>Web Programming &copy; Sheridan College</footer>
<script>
/*  ***
    Tic Tac Toe Game functionality
    FIRST VERSION: procedural unobtrusive Javascript, using console output
    SECOND VERSION: procedural unobtrusive Javascript, using DOM methods and unmodified
    HTML
    THIRD VERSION: functional unobtrusive Javascript, using DOM methods to create HTML
    FORTH VERSION: object-based unobtrusive Javascript, uinsg DOM methods to create HTML

    set up all variable and data structures
    - current player: X or O
    - array (collection) of 9 objects
    - all winning combinations, 3 rows, 3 columns, 2 diagonals
    - number of available (empty) cells
    - game not in session (false if in process)
    - handle to <span id="player">
    - no handle to <button id="reset"> USED only once
    - handle to <div id="message">

    playTicTacToe
    - call gameReset() to start a game
    - when cell is clicked,
            call function cellWasClicked(whichCell){}
    - function cellWasClicked(whichCell){}
            calls function checkIfCurrentPlayerIsWinner(lastCellPlayed)
    - function checkIfCurrentPlayerIsWinner(lastCellPlayed)
            calls displayWhoWon() when there is a winner or a scratch (ends game)
*/



/*  ***
    function displayWhoWon() is called when the game is over and
```

```
        the results are displayed: "Game Over! " + player  + " wins."
    */
    function displayWhoWon() {
    }


    /*  ***
        function checkIfCurrentPlayerIsWinner() is called to check all winning combinations
        calls displayWhoWon() if one of them is found true.
    */
    function checkIfCurrentPlayerIsWinner(lastCellPlayed) {
        // loop through all combos and check if one has the same values
        //      but it is not empty
        // display the end of the game message - game over
        // highlight the winning combo on screen
        // if no empty cells, it is a scratch; display game over
        // check if there are any remaining empty cells
        // display the end of the game message - game over
        // start with checking every time; then filter to lastCellPlayed
    }


    /*  ***
        Function cellWasClicked(whichCell) is called
            when the event listeners for the "td" cells fire which occurs
            when the user clicks on one of the nine cells of the board
        1. sets the content of the clicked cell to the current player's mark
        2. checks whether or not there is a winner
        3. flips (changes) the current player
        4. updates the message to the current player
        TODO: 1-4 should occur only when the selected cell is empty !!
    */
    function cellWasClicked(whichCell) {
        // conditional on game not being over and cells available
        // place the user character
        // one less cells is available
        // check if there is a winning combination
        // update player turn and display
    }


    /*  ***
        function gameReset() is called when user clicks on the "Start A New Game" button
        1. sets content of all 9 cells to nothing
        2. sets the starting player (this version, X always starts the game)
        3. updates the message to the current player
        4. resets the number of empty cells to 9
        5. sets the game over flag to false to indicate that the game is in progress
        6. reset font color
    */
    function gameReset() {
    }


    /*  ***
        Set up event listeners
        1. when user clicks on the reset button (id="reset")
        2. when user clicks on one of the 9 cells on the board
    */


    /*  ***
        Further enhancements
        - TODO: change the background of the last cell played to indicate what was the last move
        - TODO: display and style overlays with messages
        - TODO: create the board (table) with Javascript
        - TODO: function playTicTacToe() to load and initialize entire game on "DOMContentLoaded" event
        - TODO: (optional) make the starting player random
        - TODO: (optional) keep track of statistics (how many times X wins, etc.)
        - TODO: constrain checking for winner after the forth turn (start on turn 5 when x places third mark)
        - TODO: constrain checking only the row, column, and diagonal(s) containing the last selected cell
        - TODO: create and destroy the board with Javascript instead of hard-coding it with HTML
        - TODO: convert to object-oriented version
    */
    </script>
    </body>
    </html>
```