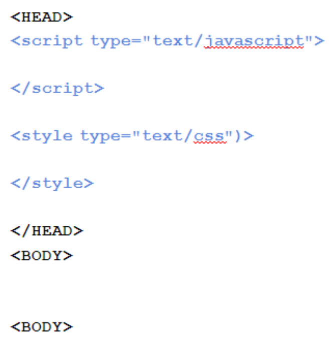
Assignment 3: Tic Tac Toe
Due: Check Syllabus

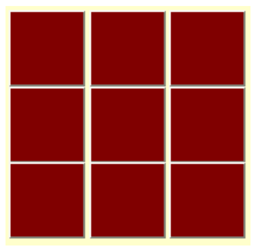
How to write the Tic Tac Toe JavaScript code?

In order for me to help you complete and develop this program, I am going to help you work through this systematically. This will provide you a chance to see how to develop programs in general, especially for web applications. We will try and develop the skeleton/outline and then we will fill in each part of the outline methodically.

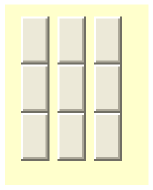


```
<HEAD>
<script type="text/javascript">

</script>

<style type="text/css")>

</style>

</HEAD>
<BODY>


<BODY>
```

Step 1: Set up your html page
Include the script headers for the style sheets and the Javascripts. You do not have to fill in anything at this time. You should use this for a template for the rest of the semester. Unless you need to use external style sheets and external javascript files.



Step 2: Create your initial Tic Tac Toe game (BODY)
The first thing you should do is just get nine boxes to appear as shown. Since you want to be able to click the box and make the box change to an X and an O, the GUI object that needs to be defined for the box will be a button. However, try not to be too ambitious yet.



Step 2a: First, we create the "vanilla" buttons
We can always improve how it looks. Once you can create the 9 buttons, you can change how it looks by redefining the buttons specifically for the game by using style sheet later. Name each of the buttons as *b1*, *b2*, …, *b9*.

Step 2b: Create the nice looking buttons by using style sheets
I have provided a style sheet that you can use for defining box like buttons (see POP/TART exercise).

Step 3: Get the button to show an X.
There are two ways to do this. The first way is to physically assign a function for each of the nine buttons and using the function *changeBox1()*, *changeBox2()*, …, *changeBox9()* for each of the 9 boxes. But this would be a very inefficient way of writing this code because you have to duplicate all the codes and checks for each of the 9 boxes. The alternative that has been discussed in this class is to write just one function *changeBox(this)* that has "this" as a parameter. The only challenge is how to change the button to show X and O. See POP/TART exercise for lots of hints. To do step 3, break it down into smaller pieces. First let's write the function that the event-handler will call ie. *onclick=changeBox(this)* when you click the button. Note that in the body/form of HTML: you use "this" while in the javascript file you use "cell" as the parameter.

```
function changebox(cell) {
// add code in here

}
```

Just write an empty function like that. Do not worry about filling at the moment.

Step 3a: Write code in *changeBox(cell)* to try and get the button to show X when you click it. It does not matter if it is occupied etc. This is to allow you to check that you are able to write the correct event handler function. In this case just do a *cell.value*="*X*" just to have a feel of the function.

Step 3b: Next try and add more code so that you are able to show X and O alternatively. Again, do not worry if the box is occupied before you display the X or O. Try to use a global variable such as *symbol* that stores the character that you want to display ie. X or O. Initialize *symbol*="*X*" at the beginning of the game. Instead of *cell.value*="*X*" written above, now you have an instruction that says *cell.value* = *symbol* instead. Now, all you have to do is to determine that the *symbol*="X" sometimes and *symbol*="O" alternatively. Again, see POP/TART example to help you. You will need to write a *changeSymbol()* function that changes from X to O after **every legitimate move** describe as such:

```
// this function is called to change the symbol from X to O and vice versa
function changeSymbol() {
       if (symbol== "X") {
       // change the symbol to "O"

       }
       else {
       // change the symbol to "X"

       }
}
```

Step 3c: Now that you can alternatively change the *symbol* to X and O, you have to check the button is occupied. It is very simple to check because of the "cell" parameter. By checking *cell.value* you will know if it is empty ie. "". If it is, you will do the following.

- You will display the new symbol on the button
  ```
  cell.value = symbol;
  ```

- You will check if the new symbol created a winning strategy for you. You will write a function *checkWin()* that returns a value *true* if it is a winning strategy. If it returns *false* then there is no winning strategy. See Step 4 for details of *checkWin()* function. Right now you just need a simple *checkWin()* function perform a return in Step 4, your code in Step 3 is simplified to have in it the following:

  ```
  if (checkWin()) {
         // alert the user that he/she won

         return;
  }
  else {
         // change the symbol from X to O or O to X for the next player
         changeSymbol();
         return;
  }
  ```

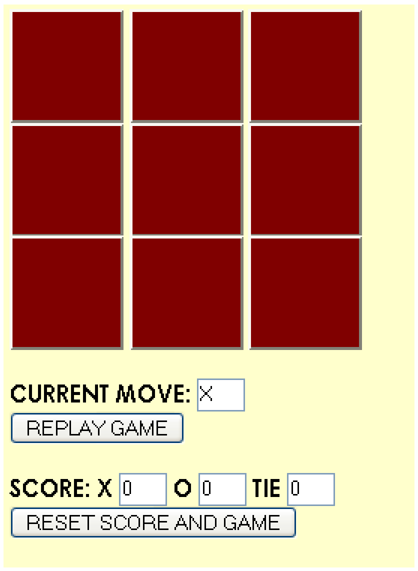
- if it is a WIN strategy, alert the player that he won.

- otherwise change the symbol to the alternative symbol.

If it is occupied, then you alert the player that it is an illegal move and try again. MAKE SURE YOU DO NOT CHANGE THE SYMBOL for an illegal move.

Step 4: function checkWin()
```
function checkWin() {
       // check to see if the current move created a WINNING strategy
       // insert code here to check

       // otherwise return false
       return false;
}
```
For this function, the check is straight forward. Assuming you are "X" and it is your move, it is reasonable to assume that the current *symbol* that we have been discussing above at the moment is also "X" because it is your move. After you were successful in making a move, you have to check to see if the move that you just made created a WINNING strategy for you. To do so, you will do something like this. (Assuming you name the buttons for the first row as *b1*, *b2* and *b3* respectively). The reason that we use *document.simpleForm* using the *with* command is that it will shorten our code. Otherwise, we have to type similar commands *document.simpleForm.b1.value* for all the buttons (there are 24 of them). Let's make life easy on you.

```
with (document.simpleForm) {
       // check the first row
       if ((b1.value == symbol) && (b2.value==symbol) && (b3.value == symbol))
              return true

       //check the second row
       //check the third row
       //check the first column
       //check the second column
       //check the third column
       //check the NW-SE diagonal
       //check the NE-SW diagonal
}
```



Step 5: You are almost done. You just have to incorporate the rest of the buttons for the Tic Tac Toe. First you create all the necessary text output box and buttons in the body of the HTML page. Use the loan calculator exercise if you need help with the commands.

Current Move: Tells you what move is it X or O
Replay Game: It is to reset the current game.
Score X: The number of times X wins
Score O: The number of times O wins
Tie: The number of times that it is a tie
Reset Score and Game: It resets the game and resets the scores to 0s.

Step 5a: Current Move
Inside the *changeSymbol()* function, after you change the symbol, you will also display the new symbol that indicate the current move.

Step 5b: Replay Game
This is just to make all the buttons display "" and to initialize the current symbol to "X". I suggest you write a simple function *replayGame()* to put all the code inside. You also need to initialize counters etc. DO NOT reset the number of times X wins or O wins here.

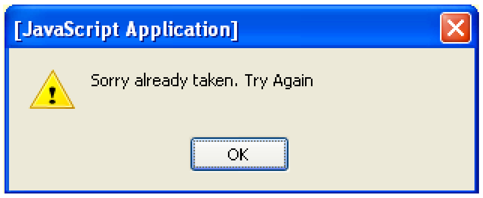
Step 5c: Scores
If it is a WIN, check if the current symbol is X or O. if it is an X then increment *xCount* by one and then display *xCount*. If the current symbol is an O, then increment *oCount* by one and then display *oCount*.

How do I know if it is a tie? In order to know if it is a draw, you need to count in each game the number of moves. Use the variable *numMoves*. *numMoves* is incremented after every valid move is detected (ie. the button is not occupied see Step 3c). It is reset to zero when you perform a replay. You have to check *numMoves* in Step 3c to determine a DRAW after checking for WINNING strategy.

```
if (checkWin()) {
// alert the user that he/she won

       if (symbol == "X" ) {
              // increment the xCount by one

              // display the xCount in the appropriate box

       }
       else {
              // increment the oCount by one

              // display the oCount in the appropriate box

       }

       replayGame(); //resets the game
       return;
}

else if (numMoves == 9)  {
       // alert that the game is tied.

       // increment the xTie by one

       // display the xTie in the appropriate box

       replayGame(); // resets the game
       return
}

else {
       // change the symbol from X to O or O to X for the next player
       changeSymbol();
       return;
}
```

Step 5d: Reset Scores and Game
Write a function *resetScore()* that initializes all the *count*s and *numMoves*. It also calls *replayGame()* to reset all the button values and moves. Once it is initialized, the display boxes in the forms should show the new values etc. 0 or "".

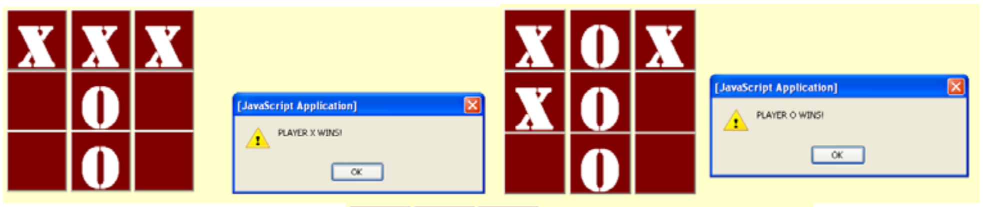
Here are some of the responses and alerts that you have to implement for the game:


[JavaScript Application]

Sorry already taken. Try Again

OK

Trying to put a symbol on an already occupied cell.


[JavaScript Application]

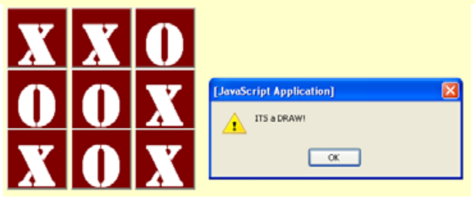
PLAYER X WINS!

OK

[JavaScript Application]

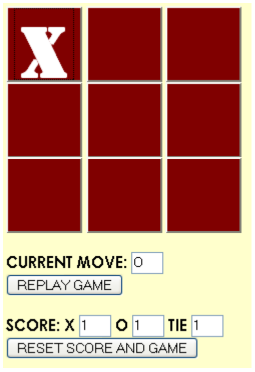
PLAYER O WINS!

OK


[JavaScript Application]

ITS a DRAW!

OK

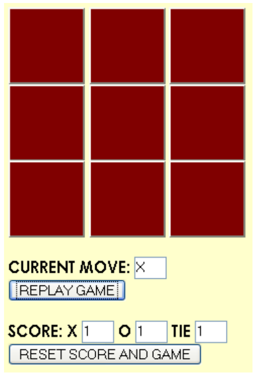

CURRENT MOVE: O

REPLAY GAME

SCORE: X 1 O 1 TIE 1

RESET SCORE AND GAME

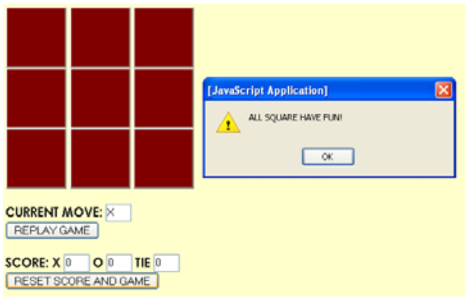
Note on the left that the current move is "O", the score is X–1, O–1 and Tie-1. If I click Replay Game, the scores should STAY the SAME as it appeared to the right of here. Note that the current move is also set to X now, since X always makes the first move.


CURRENT MOVE: X

REPLAY GAME

SCORE: X 1 O 1 TIE 1

RESET SCORE AND GAME


[JavaScript Application]

ALL SQUARE HAVE FUN!

OK

CURRENT MOVE: X

REPLAY GAME

SCORE: X 0 O 0 TIE 0

RESET SCORE AND GAME

Therefore, the scores are only reset if you click the Reset Score and Game button. Note that I alerted the players "All Square Have Fun!"

There you have it. This may not be the most efficient way of writing code, but it is a way to attack this problem. Building small parts incrementally and not do the whole thing at once. Here is the summary outline of the strategy. Make sure each step works before you move to the next one. When you have *if* or *if-else* statements, try and indent your code so that you know where the braces match.

- Create 9 simple buttons.
- Change it to pretty buttons using style sheets.
- Try to see if you can click any button and have it appear with an X by writing a simple *changeBox(cell)* function.
- Now increase the complexity by being able to click the button with X and O appearing alternatively by modifying your simple *changeBox(cell)* function and also writing a new *changeSymbol()* function.
- Now check to see if the box is empty before you display an X or an O by modifying your *changeBox(cell)* function by introducing a few *if – else* statements.
- Write the *checkWin()* function to see if a move by X or O created a WIN for the player. In the function, try checking the first row, and test it on the first row. If it is done correctly, you can mimic the code for the rest of the rows, columns and two diagonals. Modify your *changeBox(cell)* function to make sure that you perform a *checkWin()* after every legal move only. Again, you have to introduce a few *if – else* statements.
- Now check if the move created a tie. Using *numMoves* variable to count the number of moves for each legal move. If it reaches 9, and no one wins, then it is a tie. Again, modify your *changeBox(cell)* by introducing an *if* statement.
- Now when a move created a WIN for a player, identify the player ie. X or O and increase the number of times the player wins ie. *xCount* or *oCount* for X and O player respectively. Display the new count after each WIN.
- Write a *replayGame()* function that resets the buttons, *numMoves*, but not *xCount* nor *oCount*.
- Write a *resetScore()* function that resets all the counts, scores, buttons etc. This function calls *replayGame()* to help in resetting the game.

Note that this is one assignment where I will provide minimal to no help as you have to incorporate all the things you learn in the class exercises in this assignment. I am confident you can accomplish this so should you.

Good Luck

You deserve a well timed Spring Break