**TIC TAC TOE:**

**HTML:**

So basically we always link CSS and JS in the html.

Now lets understand about the body of the html, we have started with startGame() which has been declared in the JS. Now lets know what’s onload, whenever we open a web page of html the file page gets opened so the first thing that has to be loaded is start game() i.e., it’s a event handling the function on the web page that has been mentioned in the onload in which that has been loaded first, then other functions will be loaded.

Css and html works together but startgame() has been first mentioned so that get’s executed first.

In h1 tag we mention the color of the table and we have taken class as text then mention the name(Tic tac toe).

For result we have created a division and then we have passed message in id and class as text1. Message is that where we tell which person has won and whose turn is gonna, is the game has been tie or not i.e., of either X or O that will be shown in the result.

Then we create a table and name class shift. So we know that we need 3 rows and 3 columns for the tic tac toe game.

So in html we have td for column i.e., table data and tr for rows i.e., table rows

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Then we create a button for startgame() in onclick that’s whenever we click on that button new game will be started i.e., on the button we have written start new game in which we can start from first

**CSS:**

In CSS part in body we have mentioned the font family what type of font we needed and background we have linear gradient towards right we have combined 4 colors in it.

In table we have used border collapse in which it collapses all the spaces in html and combines it into a proper rows and columns, then spacing of border is set to 0 as we don’t need any spaces in between the rows and columns. Padding is also set to 0 so that there’s no spacing in between. Then Margin, background color & border radius is set.

In table tr & td we set the width and height is set, border color is set to light grey color where(0.05 is the strength/opacity of that border), it’s radius, size of the font and alignment of the text is set . Then comes table hover that’s how we need to move the mouse the background color cursor pointer and border radius is set.

In class but(button) we set the position, width, height, bg color, color, border radius, cursor pointer, font size, font weight, transform translate setting the location of the button, alignment i.e., left and top

In class text (tic tac toe) we have done same as button i.e., the alignment, position, top, left, transform, font weight, color, font size.

In class shift that’s mentioned in the table in which we set the location of the table (top, left and transform is set translate coordinates is not always same as -50,+50 for tr td it’ll be different 1,100).

In class text1 i.e., of result is same as class text only difference is the size of the font is lesser than the text of tic tac toe i.e., heading. In this translate coordinates are not same again its difficult/complex to calculate.

In button hover that’s setting the button color to green i.e., we can notice when the cursor is pointing the light glows.

**JavaScript:**

1. Create a function **startGame()** that’s set in the onload in this we run a for loop from 1 to 9 i.e., from start to end and increment it after each iteration and we mention clearbox of i i.e., 1,2,3… so on. It clears all the box. Then document.turn chooses whose turn it’s so we use inbuilt function math.random which is <0.5 if it’s then it’s gives chance to O else to X. now if the winner is declared then we again restart the game so in order to start new game we set all the boxes to null i.e., document.winner=null. We set the message whose turn is about to start.

(method) Document.getElementById(elementId: string): HTMLElement

Returns a reference to the first object with the specified value of the ID attribute.

@param elementId — String that specifies the ID value.

Style-(property) ElementCSSInlineStyle.style: CSSStyleDeclaration

Color-(property) CSSStyleDeclaration.color: string

2. Create a function **set message** where we pass msg as a parameter. Then I call message by using doc.getEleId in which we pass messages what has to be printed (property) CSSStyleDeclaration.color: string i.e., who’s turn, who has won and so on using textcontent of msg getting displayed in result.

3. Create a function **nextMove** and pass square as a parameter that has been declared as a class name in html and also nextMove in onclick by passing this as a keyword that is written in the JS. If winer is not equal to null i.e., winner has been already declared and again if we forcefully click on any box it pops up a message telling that X/O that’s stored in doc.winner has already won the game. If not then we check if any square is null i.e., telling turn first X then O in a sequential manner i.e., by switching the turn one after the other and making sure that same person is not playing again and again. If we try to overwrite the square/box that has already been filled/occupied then it tells that this particular square is already used. Runs until the winner is not declared.

4. Create a function **switch turn**. In this we check for winner that’s stored in the doc.turn, if winner is found then we set message congratulations you win and store the winner in doc.winner else if winner is not found we check for tie if it’s then we set message it’s a tie so play again. Else if X has played and if it’s O turn then it sets doc.turn to O ad sets message that it’s O turn. Similarly, if O has already played and it’s X turn so doc.turn is set to X and tells that it’s X turn.

5. Create function **checkForWinner(move)**. Here move is which one’s moving , how it’s moving and how it’s decided that has been done. So we first assign var result as false. Using if condition we check for all the rows by giving all the possibilities that matches in order to declare the winner. If any one of the possibility matches then the result is set to true and then we return the result.

6. Create a function **checkrow** by passing a,b,c and move as parameters. In this also we do same as we did in check for winner & move is deciding whose turn it is. So we first assign var result as false. Using if condition we check if getbox(a,b,c) is equal to move cz here a,b,c are variables and 1,2,3 are parameters corresponding to that.

Eg: getbox(a) == move && getbox(b) == move && getbox(c) == move

(1) (2) (3) == move

Then it returns result as true and prints winner by congratulating

7. Create a function **getbox(number)**. In thi we return the value of the box with a particular number i.e., 1,2,3/4,5,6/ …. So on. Where a,b,c is a string and 1,2,3 is a number. Textcontent is the one that checks which condition is passed into it.

8. Create a function **clearbox(number).** In this we set all the boxes to null that’s to empty in which it clears all the squares that’s why we run a for loop in startgame function

9. Create a function **checkForTie().** In this we run for loop from 1-9 as we have only 9 boxes in tic tac toe. In this using if condition we check if getbox is equal to null i.e., we check whether all the squares are empty as the person has not played so even the winner has not been declared so if yes then we return false else if return true then check for tie function is called and the message is printed i.e., it’s a tie play again.