# CODE EXPLANATION

VIDEO LINK: <https://www.youtube.com/watch?v=Y-GkMjUZsmM> (WEBDEV SIMPLIFIED)

1. Selecting all the cells const cellElements = document.querySelectorAll('[data-cell]');
2. Loop through all the cells cellElements.forEach(cell =>

        cell.addEventListener('click', handleClick, {once:true})

    })

1. Creating function handleClick()

function handleClick(){

//checking for placeMarks

placeMarks(cell, currentClass);

//for win, for draw, switch turns

swapTurns();

setBoardHoverClass();

}

1. Creating variables for turns x or circle. To check if its x’s or circle’s turn

const X\_CLASS = 'x';

const CIRCLE\_CLASS = 'circle'

let circleTurn

1. function placeMarks(cell,currentClass){

     cell.classList.add(currentClass);

}

It’s simple all we need is to pass current cell and class and inside it just add the class.

1. Switching Turns according to class. It will take circleTurn and set it to opposite of it. In that way we can switch our turns.

function swapTurns(){

     circleTurn = !circleTurn;

}

1. Applying hover states b/c currently we can’t see what we created through our CSS. Make sure to call it after swap turn so that you know which turn it is

const board = document.getElementById('board');

getting board through its id.

function setBoardHoverClass(){

    board.classList.remove(X\_CLASS); //removing existing class

    board.classList.remove(CIRCLE\_CLASS);

    if(circleTurn){

        board.classList.add(CIRCLE\_CLASS);  //checking the turn and adding class according to it

    }

    else{

        board.classList.add(X\_CLASS);

    }

}

1. We are not getting hover state for first turn how to do that is by creating a function

startGame()

function startGame(){

    circleTurn = false;

    cellElements.forEach(cell => {

        cell.addEventListener('click', handleClick, {once:true})

    })

    setBoardHoverClass();

}

1. Check for wins and draw

Creating an array of winning combinations and will use this inside a function to check for wins.

const WINNING\_COMBINATIONS = [

    [0, 1, 2],

    [3, 4, 5],

    [6, 7, 8],

    [0, 3, 6],

    [1, 4, 7],

    [2, 5, 8],

    [0, 4, 8],

    [2 ,4, 6]

]

if(checkWin(currentClass)){

     }

Now create that checkWin function, what we will check here is if all the winning combination is met with some of the combination

function checkWin(currentClass){

    return WINNING\_COMBINATIONS.some(combination =>{

        return combination.every(index =>{

            return cellElements[index].classList.contains(currentClass)

        })

    })

}

If the current class is in all of these three elements inside of the combination then we are a winner.

1. Now whenever X or O wins we have to display a message

const winningMessageElement = document.getElementById('winningMessage');

const winningMessageTextElement = document.querySelector('[data-winning-message-text]');

if(checkWin(currentClass)){

      endGame(false)

    }

function endGame(draw){

    if(draw){

    }else{

        winningMessageTextElement.innerText = `${circleTurn ? "O's" : "X's"} Wins!`

    }

    winningMessageElement.classList.add('show');

}

1. Figure it if it draws

function handleClick(e){

    // place marks

    //e.target  whatever we clicked on

    const cell = e.target;

    //current class. If its circleTurn than circle class otherwise x class

    const currentClass = circleTurn ? CIRCLE\_CLASS : X\_CLASS;

    placeMarks(cell, currentClass);

    // check for win

    if(checkWin(currentClass)){

      endGame(false)

    }else if(isDraw()){ //if its draw then end the game.

        endGame(true)

    }else{ // and if its neither of those we swap turns

        swapTurns();

        setBoardHoverClass();

    }

    // check for draw

    // switch turns

}

\*above the inside the endgame function add this

if(draw){

        winningMessageTextElement.innerText = 'Draw'

    }

Creating the draw function. Checking if all the cells are filled.

function isDraw(){

    return [...cellElements].every(cell =>{

        return cell.classList.contains(X\_CLASS) || cell.classList.contains(CIRCLE\_CLASS)

    })

}

cellElements doesn’t have every method so we deconstruct it into an array.

1. Restart button

const restartButton = document.getElementById('restartButton')

startGame()

restartButton.addEventListener('click', startGame)

Add an event listener with a function startGame, but to make it work we need to do some modifications in our startGame function because at the moment the function is not reversing the state of everything that happens. What it mean is that first we set and then reset it.

function startGame(){

    circleTurn = false;

    cellElements.forEach(cell => {

        cell.classList.remove(X\_CLASS);

        cell.classList.remove(CIRCLE\_CLASS);

        cell.removeEventListener('click', handleClick)

        cell.addEventListener('click', handleClick, {once:true})

    })

    setBoardHoverClass();

    winningMessageElement.classList.remove('show')

}

Highlighted lines shows that these classes and event listener is removed when one won or the game is draw. Also removing the show class once restart button is clicked.

Deconstruct array study yourself.