CODE FOR TIC TAC TOE GAME

HTML Code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Tic-Tac-Toe</title>

<link rel="stylesheet" href="style.css" type="text/css">

<link href="https://fonts.googleapis.com/css?family=Fredoka+One|Press+Start+2P&display=swap" rel="stylesheet">

</head>

<body>

<header>

<h1>Tic-Tac-Toe</h1>

</header>

<div class='setup'>

<div class="chooseplayertype">

<h2>Player 1 (X):</h2>

<div class="selecttype">

<div class='selection human one'>Human</div>

<div class='selection ai one'>AI</div>

</div>

</div>

<div class="chooseplayertype">

<h2>Player 2 (O):</h2>

<div class="selecttype">

<div class='selection human two'>Human</div>

<div class='selection ai two'>AI</div>

</div>

</div>

<div class="startbuttondiv">

<button class="startgame">Start Game</button>

</div>

</div>

<div class="turn"></div>

<div class="winner"></div>

<div class='gameboard'></div>

<script src='script.js'></script>

</body>

</html>

JavaScript code

// DOM grabbing module

const DOM = (() => {

return {

//Human or AI

selection: document.querySelectorAll('.selection'),

//Select Human/AI

makeActive: function(e) {

e.target.parentNode.querySelectorAll('.selection').forEach(child => {

child.classList.remove('active')

})

e.target.classList.add('active')

if (e.target.classList.contains('human')) {

if (e.target.classList.contains('one')) {

Controller.player1.type = 'human'

} else {

Controller.player2.type = 'human'

}

} else if (e.target.classList.contains('one')) {

Controller.player1.type = 'ai'

} else {

Controller.player2.type = 'ai'

}

},

boardContainer: document.querySelector('.gameboard'),

//fetch current squares

getSquares: function() {

return this.boardContainer.querySelectorAll('.square')

},

newSquare: function(html) {

const square = document.createElement('div')

square.className = 'square'

square.innerHTML = html;

return square;

},

newSquareInner: function(mark) {

return `<span>${mark}</span>`

},

clearBoard: function() {

DOM.getSquares().forEach(square => {

this.boardContainer.removeChild(square)

})

},

render: function(board) {

this.clearBoard()

board.forEach(square => {

this.boardContainer.appendChild(DOM.newSquare(DOM.newSquareInner(square.mark)))

});

},

setupWindow: document.querySelector('.setup'),

startButton: document.querySelector('.startgame'),

turnIndicator: document.querySelector('.turn'),

winnerBanner: document.querySelector('.winner'),

winDisplay: function(winner) {

if (winner) {

this.winnerBanner.textContent = `${winner} wins!`;

} else {

this.winnerBanner.textContent = "It's a tie!"

}

const playAgainContainer = document.createElement('div')

playAgainContainer.className = 'playagaincontainer'

const playAgain = document.createElement('button')

playAgain.textContent = 'Play Again'

playAgain.className = 'playagain'

playAgainContainer.appendChild(playAgain)

this.winnerBanner.appendChild(playAgainContainer);

playAgain.addEventListener('click', () => {

location.reload();

return false;

})

}

}

})()

//Gameboard module - argument to where it will render

const Gameboard = ((container) => {

//each square saved as an object with a mark property, which can either be an empty string, 'X', or 'O'

const square = {

mark: ''

}

//board saved as array

const board = [];

//board is inaccessible to other modules, but they can fetch it

const getBoard = () => {

return board;

}

//controlled board manipulation sent to the game controller

const newMarker = (mark, index) => {

board[index] = {mark}

DOM.render(board)

}

// game board load

const init = () => {

for (let count = 1; count <= 9; count++) {

board.push(square)

}

DOM.render(getBoard())

}

// make these available to other modules

return {

getBoard,

init,

newMarker

};

//where the board will render IIFE

})(document.querySelector('.gameboard'))

const Controller = (() => {

//type is decided in start screen

const player1 = {

name: 'Player 1',

marker: 'X',

type: ''

}

const player2 = {

name: 'Player 2',

marker: 'O',

type: ''

}

//turn counter

let player1turn = true;

//page load selection screen

const init = () => {

DOM.selection.forEach(element => {

element.addEventListener('click', DOM.makeActive)

}),

DOM.startButton.addEventListener('click', () => {

if(selectionCheck()) {

startGame();

} else {

alert('Please select a player type for each player')

}

})

}

//toggle turn

playerToggle = () => {

player1turn = !player1turn;

}

//before starting, make sure the type of player is selected for each

const selectionCheck = () => {

return Boolean(player1.type && player2.type)

}

const startGame = () => {

DOM.setupWindow.style.display = 'none'

Gameboard.init()

takeTurn()

}

//check the board for every win condition

const checkWinner = () => {

const board = Gameboard.getBoard()

const winConditions = [

[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 any of the conditions are true, stop the game and announce the winner in the console

if (winConditions.some(array => {

let winCheck = [];

array.forEach(box => {

if (board[box].mark !== '') {

winCheck.push(board[box])

}

})

if (winCheck.length == 3) {

if (winCheck.every(square => {

return square.mark == 'X';

})){

DOM.winDisplay(player1.name)

return true;

} else if (winCheck.every(square => {

return square.mark == 'O';

})){

DOM.winDisplay(player2.name)

return true;

} else {

return false;

}}

})) {

return true;

//if the board has 9 marks without a winner, it's a tie

} else if (board.filter(square => {

return square.mark !== ''

}).length == 9) {

DOM.winDisplay()

return true;

} else return false;

}

// AI...sort of...

const computerPlay = (marker) => {

let choices = Gameboard.getBoard().map((square, index) => {

if (square.mark !== '') {

return false;

} else {

return index;

}

})

choices = choices.filter(item => {

return item !== false;

})

const selection = Math.floor(Math.random() \* choices.length);

Gameboard.newMarker(marker, choices[selection]);

playerToggle()

takeTurn()

}

const humanPlay = (marker) => {

DOM.getSquares().forEach(square => {

square.addEventListener('click', e => {

if (e.currentTarget.textContent == '') {

const index = Array.from(e.currentTarget.parentNode.children).indexOf(e.currentTarget)

Gameboard.newMarker(marker, index)

playerToggle()

takeTurn()

return;

}

})

})

}

//flow controller, checks winner, swaps the player turn indicator

const takeTurn = () => {

if (!checkWinner()){

let player;

if (player1turn) {

player = player1

} else {

player = player2

}

DOM.turnIndicator.textContent = `${player.name}'s turn:`

if (player.type == 'ai' ) {

computerPlay(player.marker)

} else {

humanPlay(player.marker)

}

} else console.log('Winner found, stopping game');

}

//load the game

init()

//export so the DOM can set their types

return {

player1,

player2,

}

})()

CSS Code

html {

font-family: 'Press Start 2P', cursive;

-webkit-user-select: none;

-moz-user-select: none;

user-select: none;

background-color: #FCF7E8;

color: #CE4D41;

}

header {

text-align: center;

font-size: 1.2rem;

margin-top: 3rem;

}

.setup {

display: flex;

flex-wrap: wrap;

width: 50%;

margin: 3rem auto;

}

.chooseplayertype {

min-width: 50%;

text-align: center;

}

.selection {

padding: 0.5rem;

margin: 0 1rem;

width: 5rem;

border: 1px solid #519391;

border-radius: 3px;

margin: 1rem auto;

}

.selecttype {

margin: 3rem 0 1rem;

}

.selection:hover {

transform: scale(1.05);

}

.active {

background-color: #86CBC0;

}

.startbuttondiv {

margin: 1rem auto;

}

.startgame {

padding: 1rem;

}

.gameboard {

display: grid;

height: 40vh;

width: 20vw;

grid-template: repeat(3, 1fr) / repeat(3, 1fr);

margin: 15vh auto;

justify-items: center;

outline: 3px solid #FCF7E8;

outline-offset: -1px;

}

.turn {

text-align: center;

font-size: 1rem;

margin: 3rem 0;

}

.square {

display: flex;

text-align: center;

font-family: 'Fredoka One', cursive;

font-size: 3rem;

height: 100%;

width: 100%;

border: 1px solid black;

cursor: pointer;

}

.square:hover {

background-color: rgba(220, 220, 220, 0.5);

}

span {

margin: auto;

}

button {

font-size: 0.6rem;

font-family: 'Press Start 2P', cursive;

}

button, .selection {

cursor: pointer;

}

.winner {

display: block;

text-align: center;

font-size: 1.6rem;

margin: 1rem 0 -5vh;

}

.playagaincontainer {

display: block;

text-align: center;

}

.playagain {

font-size: 1rem;

margin: 2rem 0 -2rem;

padding: 0.9rem;

}

@media screen and (max-width: 800px) {

html {

font-size: 0.7rem;

}

h2 {

font-size: 0.9rem;

}

.setup {

width: 100%;

}

.selection {

font-size: 0.7rem;

}

button {

font-size: 0.8rem;

}

.gameboard {

height: 30vh;

width: 30vh;

outline: 5px solid #FCF7E8;

outline-offset: -2px;

}

}