HTML:

<!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" />

  </head>

  <body>

    <div class="screen">

      <div class="overlay">

        <!-- TTT=Tic Tac Toe -->

        <section class="TTT">

          <!-- Top  row line 1,2,3 = trl

      middle row line, 4,5,6 = mrl

      bottom row line, 7,8,9 = brl

      Left column line 1,4,7 = lcl

      middle column line, 2,5,8 = mcl

      right column line, 3,6,9 = rcl

      cross line from left top corner to

      right bottom corner, 1,5,9 =ltoprbottoml

      cross line from right top corner

      to left buttom corner, 3,5,7 = rtopbottomll-->

          <div class="row1">

            <div class="gameBox">

              <button class="boxBtn box1 trl lcl ltoprbottoml">1</button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box2 trl mcl">2</button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box3 trl rcl rtopbottomll">3</button>

            </div>

          </div>

          <div class="row2">

            <div class="gameBox">

              <button class="boxBtn box4 mrl lcl">4</button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box5 mrl mcl ltoprbottoml rtopbottomll">

                5

              </button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box6 mrl rcl">6</button>

            </div>

          </div>

          <div class="row3">

            <div class="gameBox">

              <button class="boxBtn box7 brl lcl rtopbottomll">7</button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box8 brl mcl">8</button>

            </div>

            <div class="gameBox">

              <button class="boxBtn box9 brl rcl ltoprbottoml">9</button>

            </div>

          </div>

        </section>

        <section class="gameTxt">

          <div class="gameWonMsg"><h1 class="winMsg">You won!</h1></div>

          <div class="gameStartMsg">

            <button>

              <h1 class="startMsg">Start Playing Tic Tac Toe!</h1>

            </button>

          </div>

          <div class="gameStartMsg replayTxt">

            <button><h1 class="startMsg replayMsg">Play Again!</h1></button>

          </div>

          <div class="gameStartMsg replayTxt">

            <button>

              <h1 class="startMsg drawMsg">It`s a DRAW! Click to replay</h1>

            </button>

          </div>

        </section>

      </div>

      <div class="leftSide">

        <div><h1 class="turn0">O`S TURN</h1></div>

        <div><h1 class="leftTimeOut">TIME IS OUT! X WON</h1></div>

        <div><h1 class="leftTimer">00:00:00</h1></div>

      </div>

      <div class="rightSide">

        <div><h1 class="turn1">X`S TURN</h1></div>

        <div><h1 class="rightTimeOut">TIME IS OUT! O WON</h1></div>

        <div><h1 class="rightTimer">00:00:00</h1></div>

      </div>

    </div>

    <script src="scirpt.js"></script>

  </body>

</html>

CSS:

.screen {

  position: relative;

  height: 100vh;

  width: 100vw;

}

.overlay {

  position: absolute;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  z-index: 2;

}

.leftSide,

.leftSide,

.rightSide {

  position: absolute;

  top: 0;

  bottom: 0;

  width: 50%;

}

.leftSide {

  left: 0;

  background: rgb(255, 255, 255);

}

.turn0,

.turn1 {

  display: flex;

  font-size: 100px;

  align-items: center;

  justify-content: center;

}

.leftTimer,

.rightTimer {

  font-size: 150px;

  display: flex;

  align-items: center;

  justify-content: center;

}

.leftTimeOut,

.rightTimeOut {

  font-size: 100px;

  display: flex;

  align-items: center;

  justify-content: center;

}

.rightSide {

  right: 0;

  background: rgb(255, 255, 255);

}

.center {

  flex: 2; /\* The center section will take twice as much space as the left and right sections \*/

  display: flex;

  flex-direction: column;

  align-items: center;

}

/\* class names for win lines

<!-- Top  row line 1,2,3 = trl

        middle row line, 4,5,6 = mrl

        bottom row line, 7,8,9 = brl

        Left column line 1,4,7 = lcl

        middle column line, 2,5,8 = mcl

        right column line, 3,6,9 = rcl

        cross line from left top corner to

        right bottom corner, 1,5,9 =ltoprbottoml

        cross line from right top corner

        to left buttom corner, 3,5,7 = rtopbottomll--> \*/

.TTT {

  background-color: aqua;

  width: 367px;

  height: 355px;

  position: absolute;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  z-index: 1; /\* To place it above the leftSide and rightSide \*/

}

.boxBtn {

  font-size: 125.1111111px;

  margin: 5px 20px;

  border-radius: 5%;

  height: 90px;

  /\* Make the numbers invisible \*/

  color: transparent;

}

/\* Makes them side by side \*/

.row1,

.row2,

.row3 {

  display: flex;

}

.gameBox {

}

.gameStartMsg {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 50vh;

}

.gameStartMsg button {

  background-color: transparent;

  border: none;

}

.startMsg {

  color: rgb(0, 0, 0);

  font-size: 50px;

  z-index: 999;

}

.drawMsg {

  z-index: 1;

  margin-top: 75%;

}

.startMsg:hover {

  scale: 1.5;

}

.winMsg {

  font-size: 150px;

  color: rgb(0, 0, 0);

}

.gameWonMsg {

  display: flex;

  justify-content: center;

  align-items: center;

}

.replayTxt {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 1vh;

}

.hidden {

  display: none;

}

.boxBtn.blue {

  background-image: url("circle.png");

  background-repeat: no-repeat;

  background-size: contain; /\* adjust as needed \*/

}

.boxBtn.red {

  background-image: url("x.png");

  background-repeat: no-repeat;

  background-size: contain; /\* adjust as needed \*/

}

JS:

/\* class names for win lines

<!-- Top  row line 1,2,3 = trl

        middle row line, 4,5,6 = mrl

        bottom row line, 7,8,9 = brl

        Left column line 1,4,7 = lcl

        middle column line, 2,5,8 = mcl

        right column line, 3,6,9 = rcl

        cross line from left top corner to

        right bottom corner, 1,5,9 =ltoprbottoml

        cross line from right top corner

        to left buttom corner, 3,5,7 = rtopbottomll--> \*/

/\* Boxes of the game \*/

const numb1 = document.querySelector(`.box1`);

const numb2 = document.querySelector(`.box2`);

const numb3 = document.querySelector(`.box3`);

const numb4 = document.querySelector(`.box4`);

const numb5 = document.querySelector(`.box5`);

const numb6 = document.querySelector(`.box6`);

const numb7 = document.querySelector(`.box7`);

const numb8 = document.querySelector(`.box8`);

const numb9 = document.querySelector(`.box9`);

//The game window

const tTT = document.querySelector(`.TTT`);

// game text

const winMsg = document.querySelector(`.winMsg`);

const startMsg = document.querySelector(`.startMsg`);

const replayMsg = document.querySelector(`.replayMsg`);

const drawMsg = document.querySelector(`.drawMsg`);

//Player representation

const player0 = `X`; // Red

const player1 = `O`; // Blue

//Player screen side

const leftP1 = document.querySelector(`.leftSide`);

const rightP0 = document.querySelector(`.rightSide`);

//Switcher for player 1 and 2.

let playerTurn = true;

//boolean to avoid further selection of boxes

let gameOver = false;

/\* put the boxes intop array => Allows to

lopp trough the options. The plan was to use for loop like this:

Not possible because JS dosnt allow that.

let numbAll;

// Player 0 and 1, in red and blue

const selected0 = function () {

  for (let i = 0; i >= 9; i++)

    if (`numb${i}` === ``) {

      `numb${i}`.style.backgroundColor = `red`;

    }

};

const selected1 = function () {

  for (let i = 0; i >= 9; i++)

    if (`numb${i}` === ``) {

      `numb${i}`.style.backgroundColor = `blue`;

    }

}; \*/

let numbAll = [

  document.querySelector(`.box1`),

  document.querySelector(`.box2`),

  document.querySelector(`.box3`),

  document.querySelector(`.box4`),

  document.querySelector(`.box5`),

  document.querySelector(`.box6`),

  document.querySelector(`.box7`),

  document.querySelector(`.box8`),

  document.querySelector(`.box9`),

];

//Timer and its functions

const leftTimer = document.querySelector(`.leftTimer`);

const rightTimer = document.querySelector(`.rightTimer`);

let leftTimerId, rightTimerId;

let leftStartTime, rightStartTime;

// Initialize elapsed time for each player

let leftElapsed = 0;

let rightElapsed = 0;

const leftTimeOut = document.querySelector(`.leftTimeOut`);

const rightTimeOut = document.querySelector(`.rightTimeOut`);

leftTimeOut.classList.add(`hidden`);

rightTimeOut.classList.add(`hidden`);

const timerForLeft = function () {

  let timePassed = Date.now() - leftStartTime + leftElapsed;

  let seconds = Math.floor(timePassed / 1000);

  let milliseconds = timePassed % 1000;

  leftTimer.textContent = `${seconds}:${milliseconds}`;

  if (timePassed >= 10000) {

    leftTimer.textContent = `10:000`;

    leftTimeOut.classList.remove(`hidden`);

    player1TurnMsg.classList.add(`hidden`);

    player0TurnMsg.classList.add(`hidden`);

    clearInterval(leftTimerId);

    gameOver = true;

    replayMsg.classList.remove(`hidden`);

  }

};

const timerForRight = function () {

  let timePassed = Date.now() - rightStartTime + rightElapsed;

  let seconds = Math.floor(timePassed / 1000);

  let milliseconds = timePassed % 1000;

  rightTimer.textContent = `${seconds}:${milliseconds}`;

  if (timePassed >= 10000) {

    rightTimer.textContent = `10:000`;

    rightTimeOut.classList.remove(`hidden`);

    player1TurnMsg.classList.add(`hidden`);

    player0TurnMsg.classList.add(`hidden`);

    clearInterval(rightTimerId);

    gameOver = true;

    replayMsg.classList.remove(`hidden`);

  }

};

const btnFunction = function () {

  if (playerTurn) {

    if (leftTimerId) {

      clearInterval(leftTimerId);

      leftElapsed += Date.now() - leftStartTime; // Store elapsed time

      leftTimerId = null;

    }

    if (!rightTimerId) {

      rightStartTime = Date.now();

      rightTimerId = setInterval(timerForRight, 1);

    }

  } else {

    if (rightTimerId) {

      clearInterval(rightTimerId);

      rightElapsed += Date.now() - rightStartTime; // Store elapsed time

      rightTimerId = null;

    }

    if (!leftTimerId) {

      leftStartTime = Date.now();

      leftTimerId = setInterval(timerForLeft, 1);

    }

  }

};

// Hide turn msg and timer msg

const player1TurnMsg = document.querySelector(`.turn0`);

const player0TurnMsg = document.querySelector(`.turn1`);

const player1TurnTimer = document.querySelector(`.leftTimer`);

const player0TurnTimer = document.querySelector(`.rightTimer`);

player1TurnMsg.classList.add(`hidden`);

player0TurnMsg.classList.add(`hidden`);

player1TurnTimer.classList.add(`hidden`);

player0TurnTimer.classList.add(`hidden`);

//Function to switch color when is players turn

//Note that "leftP1.style.background = `blue`;" is added to playGameMsg function

//So that colour blue is already shown when play button is pressed.

const turnColor = function () {

  if (!playerTurn) {

    player0TurnTimer.classList.add(`hidden`);

    player0TurnMsg.classList.add(`hidden`);

    player1TurnMsg.classList.remove(`hidden`);

    player1TurnTimer.classList.remove(`hidden`);

    leftP1.style.background = `blue`;

    rightP0.style.background = `white`;

  } else if (playerTurn) {

    player1TurnTimer.classList.add(`hidden`);

    player1TurnMsg.classList.add(`hidden`);

    player0TurnMsg.classList.remove(`hidden`);

    player0TurnTimer.classList.remove(`hidden`);

    leftP1.style.background = `white`;

    rightP0.style.background = `red`;

  }

};

const clearTurnColor = function () {

  playerTurn = false;

  leftP1.style.background = `white`;

  rightP0.style.background = `white`;

  player0TurnMsg.classList.add(`hidden`);

  player1TurnMsg.classList.add(`hidden`);

  player0TurnTimer.classList.add(`hidden`);

  player1TurnTimer.classList.add(`hidden`);

};

/\* .. \*/

// Function to start the timers

const startTimers = function () {

  // Start the left timer for player1

  if (!leftStartTime) leftStartTime = Date.now();

  if (!leftTimerId) leftTimerId = setInterval(timerForLeft, 1);

};

/\* -- \*/

//Function to play the game text.

startMsg.classList.remove(`hidden`);

tTT.classList.add(`hidden`);

const playGameMsg = function () {

  player1TurnTimer.classList.remove(`hidden`);

  player1TurnMsg.classList.remove(`hidden`);

  leftP1.style.background = `blue`;

  startMsg.classList.add(`hidden`);

  tTT.classList.remove(`hidden`);

  replayMsg.classList.add(`hidden`);

  startTimers();

  ticTacToeGame();

};

startMsg.addEventListener(`click`, playGameMsg);

//function to replay the game

const replayGame = function () {

  // Reset and stop the timers

  if (leftTimerId) {

    clearInterval(leftTimerId);

    leftTimerId = null;

  }

  if (rightTimerId) {

    clearInterval(rightTimerId);

    rightTimerId = null;

  }

  leftElapsed = 0;

  rightElapsed = 0;

  leftStartTime = null;

  rightStartTime = null;

  // Reset the displayed time

  leftTimer.textContent = "0:0";

  rightTimer.textContent = "0:0";

  // Start the timer for the first player

  leftStartTime = Date.now();

  leftTimerId = setInterval(timerForLeft, 1);

  clearTurnColor();

  turnColor();

  //This removes all the red and blue classes.

  numbAll.forEach((box) => {

    box.classList.remove("red");

    box.classList.remove("blue");

  });

  //hides the pop messages

  winMsg.classList.add(`hidden`);

  replayMsg.classList.add(`hidden`);

  player0TurnMsg.classList.add(`hidden`);

  player0TurnTimer.classList.add(`hidden`);

  rightTimeOut.classList.add(`hidden`);

  leftTimeOut.classList.add(`hidden`);

  //Resets the game state

  playerTurn = true;

  gameOver = false;

  //starts the game

  ticTacToeGame();

};

replayMsg.addEventListener(`click`, replayGame);

// Draw msg:

drawMsg.classList.add(`hidden`);

//funciton to draw replay

const replayDrawGame = function () {

  // Reset and stop the timers

  if (leftTimerId) {

    clearInterval(leftTimerId);

    leftTimerId = null;

  }

  if (rightTimerId) {

    clearInterval(rightTimerId);

    rightTimerId = null;

  }

  leftElapsed = 0;

  rightElapsed = 0;

  leftStartTime = null;

  rightStartTime = null;

  // Reset the displayed time

  leftTimer.textContent = "0:0";

  rightTimer.textContent = "0:0";

  // Start the timer for the first player

  leftStartTime = Date.now();

  leftTimerId = setInterval(timerForLeft, 1);

  clearTurnColor();

  turnColor();

  //This removes all the red and blue classes.

  numbAll.forEach((box) => {

    box.classList.remove("red");

    box.classList.remove("blue");

  });

  //hides the pop messages

  drawMsg.classList.add(`hidden`);

  //Resets the game state

  playerTurn = true;

  gameOver = false;

  //starts the game

  ticTacToeGame();

};

drawMsg.addEventListener(`click`, replayDrawGame);

// funciton to stop the game and show the player who won.

winMsg.classList.add(`hidden`);

replayMsg.classList.add(`hidden`);

const gameWinnerMsg = function () {

  // Stop the timers

  if (leftTimerId) {

    clearInterval(leftTimerId);

    leftTimerId = null;

  }

  if (rightTimerId) {

    clearInterval(rightTimerId);

    rightTimerId = null;

  }

  if (!playerTurn) {

    winMsg.textContent = `${player0} Won!`;

    winMsg.classList.remove(`hidden`);

    replayMsg.classList.remove(`hidden`);

    playerTurn = true;

  } else if (playerTurn) {

    winMsg.textContent = `${player1} Won!`;

    winMsg.classList.remove(`hidden`);

    replayMsg.classList.remove(`hidden`);

    playerTurn = false;

  }

  gameOver = true;

};

//The whole game in a callable function

const ticTacToeGame = function () {

  const changePlayer = function () {

    if (

      gameOver ||

      this.classList.contains("blue") ||

      this.classList.contains("red")

    ) {

      return;

    }

    if (playerTurn) {

      btnFunction();

      this.classList.add("blue");

      turnColor();

      if (

        numb1.classList.contains("blue") &&

        numb2.classList.contains("blue") &&

        numb3.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb1.classList.contains("blue") &&

        numb5.classList.contains("blue") &&

        numb9.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb3.classList.contains("blue") &&

        numb5.classList.contains("blue") &&

        numb7.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb1.classList.contains("blue") &&

        numb4.classList.contains("blue") &&

        numb7.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb2.classList.contains("blue") &&

        numb5.classList.contains("blue") &&

        numb8.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb3.classList.contains("blue") &&

        numb6.classList.contains("blue") &&

        numb9.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb4.classList.contains("blue") &&

        numb5.classList.contains("blue") &&

        numb6.classList.contains("blue")

      ) {

        gameWinnerMsg();

      } else if (

        numb7.classList.contains("blue") &&

        numb8.classList.contains("blue") &&

        numb9.classList.contains("blue")

      ) {

        gameWinnerMsg();

      }

      playerTurn = false;

    } else if (!playerTurn) {

      btnFunction();

      this.classList.add("red");

      turnColor();

      if (

        numb1.classList.contains("red") &&

        numb2.classList.contains("red") &&

        numb3.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb1.classList.contains("red") &&

        numb5.classList.contains("red") &&

        numb9.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb3.classList.contains("red") &&

        numb5.classList.contains("red") &&

        numb7.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb1.classList.contains("red") &&

        numb4.classList.contains("red") &&

        numb7.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb2.classList.contains("red") &&

        numb5.classList.contains("red") &&

        numb8.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb3.classList.contains("red") &&

        numb6.classList.contains("red") &&

        numb9.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb4.classList.contains("red") &&

        numb5.classList.contains("red") &&

        numb6.classList.contains("red")

      ) {

        gameWinnerMsg();

      } else if (

        numb7.classList.contains("red") &&

        numb8.classList.contains("red") &&

        numb9.classList.contains("red")

      ) {

        gameWinnerMsg();

      }

      playerTurn = true;

    }

    //Function for a draw

    const allBoxesFilled = function () {

      let boxes = [

        numb1,

        numb2,

        numb3,

        numb4,

        numb5,

        numb6,

        numb7,

        numb8,

        numb9,

      ];

      return boxes.every(

        (box) => box.classList.contains("blue") || box.classList.contains("red")

      );

    };

    // After a player makes a move

    if (!gameOver) {

      // Delay the draw check by a small amount of time

      setTimeout(() => {

        if (allBoxesFilled()) {

          // Reset and stop the timers

          if (leftTimerId) {

            clearInterval(leftTimerId);

            leftTimerId = null;

          }

          if (rightTimerId) {

            clearInterval(rightTimerId);

            rightTimerId = null;

          }

          leftElapsed = 0;

          rightElapsed = 0;

          leftStartTime = null;

          rightStartTime = null;

          btnFunction();

          drawMsg.classList.remove(`hidden`);

          console.log("replayMsg classList:", replayMsg.classList);

          gameOver = true;

        }

      }, 100); // 100 ms delay, adjust as needed

    }

  };

  /\* Applies the switch \*/

  numbAll.forEach((box) => {

    box.addEventListener(`click`, changePlayer);

  });

};