

```

<!DOCTYPE html>
<!-- saved from
url=(0091)file:///H:/My%20Drive/Classroom/CS/AP%20Performance%20Task/Code/
Ultimate%20Tic-Tac-Toe.html -->
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">

    <meta name="viewport" content="width=device-width">
    <title>Ultimate Tic-Tac-Toe</title>
    <style>html{
height: 100%;
width: 100%;
font-family: Verdana;
}
body{
height: calc(100% - 16px);
width: calc(100% - 16px);
}
h1{
margin: 0;
text-align: center;
line-height: 6vh;
}
table{
table-layout: fixed;
}
#bigBoard{
width: 75vh;
height: 75vh;
position: absolute;
top: 12.5vh;
left: calc((100vw - 75vh) / 2)
}
.miniBoard{
border: 2px solid green;
width: 100%;
height: 100%;
}
/* .miniBoard:hover:not([done]){
background-color: green;
} */
.miniBoard.allowed:not([done]){
background-color: lightgreen;
}
.square{
outline: 2px solid black;
font-size: 5vh;

```

```

        text-align: center;
        height: calc((75vh / 9) - 8px);
        overflow: none;
    }
    [p="x"]{
        color: red;
    }
    [p="o"]{
        color: blue;
    }
    [p="tie"]{
        color: transparent;
        background: linear-gradient(to right, red, blue);
        -webkit-background-clip: text;
    }
    [turn="x"]{
        border: 2px solid red;
    }
    [turn="o"]{
        border: 2px solid blue;
    }
    .allowed .square:hover:not([p]){
        background-color: darkgray;
    }
    [p]:not(#blurb), [done], .miniBoard:hover:not(.allowed){
        cursor: not-allowed;
    }
    [done="x"]{
        background-color: pink;
    }
    [done="o"]{
        background-color: lightblue;
    }
    [done="tie"]{
        background: linear-gradient(to right, pink, lightblue);
    }
    #blurbBox{
        margin: 0 3vh;
    }
    button{
        border: 2px solid;
        background-color: lightgreen;
    }
    button:hover{
        background-color: lightgray;
    }
    button:active{

```

```

        background-color: darkgray;
    }
    #gameInfo{
        display: flex;
        justify-content: center;
        line-height: 4vh;
    }
    #rules {
        width: calc(((100vw - 75vh) / 2) - 10px);
        word-break: break-word;
    }</style>
</head>
<body>
    <h1 id="pageTitle">Ultimate Tic-Tac-Toe</h1>
    <table id="bigBoard" class="allowed" turn="x">
        <tbody><tr>
            <td>
                <table class="miniBoard allowed">
                    <tbody><tr>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                    </tr>
                    <tr>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                    </tr>
                    <tr>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
                    </tr>
                </tbody></table>
            </td>
            <td>
                <table class="miniBoard allowed">
                    <tbody><tr>

```

```

        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
    </tr>
    <tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
    </tr>
    <tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
    </tr>
</tbody></table>
</td>
<td>
    <table class="miniBoard allowed">
        <tbody><tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>

```

```

        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
    </tr>
</tbody></table>
</td>
</tr>
<tr>
<td>
    <table class="miniBoard allowed">
        <tbody><tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>
<td>
    <table class="miniBoard allowed">
        <tbody><tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>

```

```

        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>
<td>
    <table class="miniBoard allowed">
        <tbody><tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>
</tr>

```

```

<tr>
  <td>
    <table class="miniBoard allowed">
      <tbody><tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
      </tr>
      <tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
      </tr>
      <tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
      </tr>
    </tbody></table>
  </td>
  <td>
    <table class="miniBoard allowed">
      <tbody><tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
      </tr>
      <tr>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
      </tr>
    </tbody></table>
  </td>

```

```

        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>
<td>
    <table class="miniBoard allowed">
        <tbody><tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
        <tr>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
            <td class="square"
onclick="this.tile.board.takeTurn(this.tile)"></td>
        </tr>
    </tbody></table>
</td>
</tr>
</tbody></table>
    <div id="gameInfo">
        <h4 id="blurbBox"><span id="blurb" p="x">X's
turn</span></h4>
        <button id="newGameButton" onclick="board.reset()">Start
a new Game!</button>
    </div>

```



`<p id="rules">Rules: Ultimate Tic-Tac-Toe is like a Tic-Tac-Toe game of Tic-Tac-Toe games. The goal is to get 3 boards in a row, but a player only gets a board by winning the Tic-Tac-Toe game within it by getting 3 tiles in a row. Also, when a player plays in a board, the next player has to play in the board corresponding to the first player's position in the board they played in. For example, if X plays in the top right tile of the top left board, O then has to play in the top right board. If a player's move corresponds to a board that is already completed, then the next player may play in any of the open boards. Click on an open tile to play, or click the button below to see a randomized game! </p>`

`<button id="randGameButton" onclick="randomGame(350)">See an example game</button>`

```
<script>class Tile{
  constructor(value, elem, board, coords){
    this.value = value
    this.elem = elem
    this.elem.tile = this
    this.board = board
  this.coords = coords
    this.localCoords = coords.slice(-2)
    this.elem.removeAttribute("done")
    if(this.coords.length==4){
      this.elem.removeAttribute("p")
      this.elem.innerHTML = ""
    }else{
      this.elem.removeAttribute("done")
      this.elem.classList.add("allowed")
    }
  }
  get allowed(){
    return this.elem.classList.contains("allowed")
  }
  set allowed(allowed){
    if(allowed){
      this.elem.classList.add("allowed")
    }else{
      this.elem.classList.remove("allowed")
    }
  }
  get p(){
    return this.elem.getAttribute("p") ||
this.elem.getAttribute("done")
  }
  get done(){
    return this.p
  }
}
```

```

set p(p){
    if(!p){
        return
    }
    if(this.coords.length==4){
        this.elem.innerHTML = p;
        this.elem.setAttribute("p", p);
        this.value = p
    }else{
        this.elem.setAttribute("done", p)
    }
}
set done(done){
    this.p = done
}
get tiles(){
    if(this.coords.length==4){
        return
    }
    return Array.prototype.concat.apply([], this.value)
}
update(value=this.value){
    if(this.coords.length==4){
        this.value = value
        this.p = value
    }else{
        for(var i of this.value){
            for(var j of i){
                // console.log([i, j, this,
this.coords.length])
                j.checkThree()
                j.update()
            }
        }
        this.checkThree()
    }
}
checkThree(){
    if(this.coords.length==4){
        return
    }
    // for(var i of this.tiles.filter(e=>e.p)){
    //     for(var j of this.tiles.filter(e=>e.p==i.p&&e!=i)){
    //         for(var k of
this.tiles.filter(e=>e.p==j.p&&e!=i&&e!=j)){
    //             // console.log([i, j, k])

```

```

        //
        if((i.localCoords[0]==j.localCoords[0]&&j.localCoords[0]==k.localCoords[0]
        )||(i.localCoords[1]==j.localCoords[1]&&j.localCoords[1]==k.localCoords[1]
        )){
            //
            // console.log([i, j, k])
            // this.board.showThree(i, j, k)
            // return [i, j, k]
            // }
            // }
            // }
            if(!this.tiles.some(e=>!e.p)){
                // console.log(this.tiles, this.value[1][1])
                // console.log(this)
                this.board.showThree(this.value[1][1])
            }
            if(this.value[1][1].p){
                if(
                    this.value[0][0].p==this.value[1][1].p &&
this.value[1][1].p==this.value[2][2].p ||
                    this.value[2][0].p==this.value[1][1].p &&
this.value[1][1].p==this.value[0][2].p
                ){
                    this.board.showThree(this.value[1][1],
this.value[1][1].p)
                    return this.value[1][1]
                }
            }
            for(var i of
this.tiles.filter(e=>e.localCoords[0]==e.localCoords[1] && e.p &&
e.p!="tie")){
                if(
                    this.value[i.localCoords[0]].every(e=>e.p==i.p) ||
                    this.value.every(e=>e[i.localCoords[1]].p==i.p)
                ){
                    this.board.showThree(i, i.p)
                    return i
                }
            }
        }
    }
}

class BigBoard extends Tile{
    constructor(elem, blurb, dontReset=false){
        super([], elem, false, []);
        this.blurb = blurb
        if(!dontReset){
            this.reset()
        }
    }
}

```

```

        }
        this.board = this
    }
    reset(){
        this.turn = "x"
        this.win = false
        this.value = []
        for(var [ii, i] of
[...this.elem.firstChild.children].entries()){
            var tempRow = []
            // console.log(tempRow)
            for(var [jj, j] of [...i.children].entries()){
                var tempBoard = []
                // console.log(tempBoard)
                for(var [aa, a] of
[...j.firstChild.firstChild.children].entries()){
                    var tempBoardRow = []
                    // console.log(tempBoardRow)
                    for(var [bb, b] of [...a.children].entries()){
                        // console.log(a, b, i, j)
                        tempBoardRow.push(new Tile("", b, this, [ii,
jj, aa, bb]))
                    }
                    // console.log(tempBoardRow)
                }
                tempBoard.push(tempBoardRow)
            }
            tempRow.push(new Tile(tempBoard, j.firstChild,
this, [ii, jj]))
        }
        this.value.push(tempRow)
    }
    return this
}
get turn(){
    return this.elem.getAttribute("turn")
}
set turn(turn){
    this.elem.setAttribute("turn", turn)
    this.blurb.setAttribute("p", turn)
    this.blurb.innerHTML = turn.toUpperCase()+"'s turn"
}
takeTurn(tile){
    if(tile.value || !this.value[tile.coords[0]][tile.coords[1]].allowed
|| this.win){
        return
    }
    tile.update(this.turn)
}

```

```

        this.update()
        if(this.win){

document.querySelectorAll(".allowed").forEach(e=>e.classList.remove("allow
ed"))

            return
        }
        this.turn = this.turn=="x"?"o":"x"
        // if(!this.value[tile.coords[0]][tile.coords[1]].done &&
this.value[tile.coords[0]][tile.coords[1]].elem.querySelectorAll(".square:
not([p])").length==0){
            // this.value[tile.coords[0]][tile.coords[1]].done = true
            // }
            if(!this.value[tile.coords[2]][tile.coords[3]].done){

document.querySelectorAll(".allowed").forEach(e=>e.classList.remove("allow
ed"))

                this.value[tile.coords[2]][tile.coords[3]].allowed = true
            }else{
                //
console.log(document.querySelectorAll(".miniBoard:not([done])"))

document.querySelectorAll(".miniBoard:not([done])").forEach(e=>e.classList
.add("allowed"))
            }
            return tile
        }
        showThree(tile, winner="tie"){
            if(tile.coords.length==4){
                this.value[tile.coords[0]][tile.coords[1]].p = winner;
                this.value[tile.coords[0]][tile.coords[1]].allowed =
false;
            }else if(tile.coords.length==2){
                // this.done = tile.p
                // this.blurb.setAttribute("done", tile.p)
                this.turn = winner
                this.blurb.innerHTML =
winner!="tie"?winner.toUpperCase()+" wins!":"It's a tie!"
                this.win = winner
                // console.log(this.win)
            }
        }
    }
}
var board = new BigBoard(document.getElementById("bigBoard"),
document.getElementById("blurb"))
async function randomGame(time=100){
    function randomTurn(){

```

```
        var openBoards = board.tiles.filter(e=>e.allowed)
        var openTiles =
openBoards.map(e=>e.tiles.filter(a=>!a.p)).flat()

board.takeTurn(openTiles[Math.floor(Math.random()*openTiles.length)])
    }
    return new Promise((resolve, reject) => {
        var botGame = setInterval(() => {
            if(!board.win){
                randomTurn()
            }else{
                clearInterval(botGame)
                resolve(board.win)
            }
        }, time)
    })
}</script>

</body></html>
```