Week 11: Coding Assignment

URL to GitHub Repository: https://github.com/ValeriaMontano/TicTacToe

URL to Your Coding Assignment Video:

https://www.youtube.com/watch?v=9qB280f6w5k&t=22s

Instructions:

- In Visual Studio Code, write the code that accomplishes the objectives listed below and ensures that the code compiles and runs as directed.
- Create a new repository on GitHub for this week's assignments and push this document, with your project code, to the repository.
- Include the URLs for this week's repository and video where instructed.
- Submit this document as a .PDF file in the LMS.

Coding Steps:

- Using any of the tools you've worked with so far, create a game of Tic-Tac-Toe.
    - Create a Tic-Tac-Toe game grid using your HTML element of choice.
    - When a cell in the grid is clicked, an **X** or **O** should appear in that spot depending on whose turn it is.
    - A heading should say whether it is X's or O's turn and change with each move made.
    - A button should be available to clear the grid and restart the game.
    - When a player has won, or the board is full and the game results in a draw, a Bootstrap alert or similar Bootstrap component should appear across the screen announcing the winner.

Week 11: Coding Assignment

Video Steps:

- Create a video, up to five minutes max, showing and explaining how your project works with an emphasis on the portions you contributed.
- This video should be done using screen share and voice over.
- This can easily be done using Zoom, although you don't have to use Zoom, it's just what we recommend.
    - You can create a new meeting, start screen sharing, and start recording.
    - This will create a video recording on your computer.
- This should then be uploaded to a publicly accessible site, such as YouTube.
    - Ensure the link you share is **PUBLIC** or **UNLISTED**!
    - If it is not accessible by your grader, your project will be graded based on what they can access.
-

```
//declare the variables

const cells = document.querySelectorAll(".cell");
const statusText = document.querySelector("#statusText");
const playAgain = document.querySelector("#playAgain");
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],
];

//array of placeholders - nine empty strings one for each cell

let options = ["", "", "", "", "", "", "", "", ""];
//keep track of the current player

let currentPlayer = "X";
// keep track of the game status

let running = false;

initializeGame();
function initializeGame() {
  cells.forEach(cell) => cell.addEventListener("click", cellClicked));
  playAgain.addEventListener("click", restartGame);
  statusText.textContent = `${currentPlayer}'s turn`;
  running = true;
}

function cellClicked() {
  const cellIndex = this.getAttribute("cellIndex");
```

Week 11: Coding Assignment

```
  if (options[cellIndex] != "" || !running) {
    return;
  }
  updateCell(this, cellIndex);

  checkWinner();
}

function updateCell(cell, index) {
  //update the placeholders
  options[index] = currentPlayer;
  cell.textContent = currentPlayer;
}
function changePlayer() {
  currentPlayer = currentPlayer === "X" ? "O" : "X";
  statusText.textContent = `${currentPlayer}'s turn.`;
}
function checkWinner() {
  let roundWon = false;
  for (let i = 0; i < winConditions.length; i++) {
    const condition = winConditions[i];
    const cellA = options[condition[0]];
    const cellB = options[condition[1]];
    const cellC = options[condition[2]];

    //check for empty spaces
    if (cellA == "" || cellB == "" || cellC == "") {
      continue;
    }
    //check for same characters/winner
    if (cellA == cellB && cellB == cellC) {
      roundWon = true;
      break;
    }
  }
  }
  if (roundWon) {
    statusText.textContent = `${currentPlayer} wins!`;
    running = false;
  } else if (!options.includes("")) {
    statusText.textContent = `Draw!`;
    running = false;
  } else {
    changePlayer();
  }
}
function restartGame() {
  currentPlayer = "X";
  options = ["", "", "", "", "", "", "", "", ""];
  statusText.textContent = `${currentPlayer}'s turn.`;
  cells.forEach((cell) => (cell.textContent = ""));
  running = true;
}

```

Week 11: Coding Assignment

index.html > html > head > link

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, Collapse Folders in Explorer " />
    <title>TicTacToe</title>
    <link rel="stylesheet" href="style.css" />
    <link
      rel="stylesheet"
      href="node_module/bootstrap/dist/css/bootstrap.css"
    />
  </head>
  <body>
    <div id="Container">
      <h1>Tic Tac Toe</h1>
      <div id="cellContainer">
        <div cellIndex="0" class="cell"></div>
        <div cellIndex="1" class="cell"></div>
        <div cellIndex="2" class="cell"></div>
        <div cellIndex="3" class="cell"></div>
        <div cellIndex="4" class="cell"></div>
        <div cellIndex="5" class="cell"></div>
        <div cellIndex="6" class="cell"></div>
        <div cellIndex="7" class="cell"></div>
        <div cellIndex="8" class="cell"></div>
      </div>
      <h2 id="statusText"></h2>
      <button id="playAgain">Play Again</button>
    </div>
    <script src="script.js"></script>
    <script src="node_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>
  </body>
</html>
```

style.css

```css
* {
  background-color: beige;
}
.cell {
  width: 75px;
  height: 75px;
  border: 2px solid;
  box-shadow: 0 0 0 2px;
  line-height: 75px;
  font-size: 50px;
  cursor: pointer;
}

#Container {
  font-family: "Permanent Marker", cursive;
  text-align: center;
}

h1 {
  color: yellowgreen;
}

#cellContainer {
  display: grid;
  grid-template-columns: repeat(3, auto);
  width: 225px;
  margin: auto;
  color: rgb(67, 111, 67);
}

#playAgain {
  width: 100px;
  height: 50px;
  color: white;
  font-family: "Permanent Marker", cursive;
  font-size: 15px;
  background-color: yellowgreen;
}
```