```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
  <title>Cosmic Tic Tac Toe</title>
  <link href="https://
fonts.googleapis.com/css2?
family=Orbitron:wght@400;700&famil
y=Press+Start+2P&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="https://
cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
<style>
document.addEventListener('DOMCon
```

```
tentLoaded', () => {
  // DOM Elements
  const boxes =
document.querySelectorAll('.box');
  const playButton =
document.getElementById('play');
  const exitButton =
document.getElementById('exit');
  const heading =
document.getElementById('heading');
  const board =
document.getElementById('board');
  const messBox =
document.getElementById('mess-
box');
  const startScreen =
document.getElementById('start');
```

```
// Game State
let currentPlayer = 'X';
let gameActive = true;

// Winning Conditions
const winConditions = [
  [0, 1, 2], [3, 4, 5], [6, 7, 8], // rows
  [0, 3, 6], [1, 4, 7], [2, 5, 8], // columns
  [0, 4, 8], [2, 4, 6]             // diagonals
];

// Initialize Game
const initGame = () => {
  // Reset game state
  currentPlayer = 'X';
  gameActive = true;

  // Clear the board
```

```
    boxes.forEach(box => {
      box.textContent = '';
      box.removeAttribute('data-
value');
      box.classList.remove('win', 'lose');
    });

    // Clear message box
    messBox.textContent = '';
    messBox.className = 'message-
box';

    // Show game elements

document.body.classList.add('game-
active');
  };
```

```
  // Handle Player Move
  const handlePlayerMove = (e) => {
    const box = e.target;
    const boxIndex =
box.getAttribute('data-index');

    // If box is already filled or game
not active, ignore
    if (box.textContent !== '' ||
!gameActive) return;

    // Make move
    box.textContent = currentPlayer;
    box.setAttribute('data-value',
currentPlayer);

    // Check for win or draw
    if (checkWin(currentPlayer)) {
```

```
    endGame(false);
    return;
  } else if (isDraw()) {
    endGame(true);
    return;
  }

  // Switch player
  currentPlayer = currentPlayer === 'X' ? 'O' : 'X';

  // If it's computer's turn (O)
  if (currentPlayer === 'O') {
    setTimeout(computerMove, 500);
  }
};

// Computer Move
```

```
  const computerMove = () => {
    if (!gameActive) return;

    // Find all empty boxes
    const emptyBoxes =
Array.from(boxes).filter(box =>
box.textContent === '');

    if (emptyBoxes.length === 0) {
      endGame(true);
      return;
    }

    // Simple AI - random move
    const randomIndex =
Math.floor(Math.random() *
emptyBoxes.length);
    const box =
```

```
emptyBoxes[randomIndex];

  box.textContent = 'O';
  box.setAttribute('data-value', 'O');

  // Check for win or draw
  if (checkWin('O')) {
    endGame(false);
  } else if (isDraw()) {
    endGame(true);
  } else {
    currentPlayer = 'X';
  }
};

// Check for Win
const checkWin = (player) => {
  return
```

```
winConditions.some(condition => {
    return condition.every(index => {
      return
boxes[index].getAttribute('data-
value') === player;
    });
  });
};

// Check for Draw
const isDraw = () => {
  return Array.from(boxes).every(box
=> box.textContent !== '');
};

// End Game
const endGame = (draw) => {
  gameActive = false;
```

```
    if (draw) {
        messBox.textContent = "Game
Draw!";
        messBox.classList.add('draw');
    } else {
        const winner = currentPlayer ===
'X' ? 'You Win!' : 'Computer Wins!';
        messBox.textContent = winner;

messBox.classList.add(currentPlayer
=== 'X' ? 'win' : 'lose');

        // Highlight winning boxes
        const winCondition =
winConditions.find(condition => {
            return condition.every(index => {
                return
```

```
boxes[index].getAttribute('data-
value') === currentPlayer;
      });
    });

    if (winCondition) {
      winCondition.forEach(index => {

boxes[index].classList.add(currentPlay
er === 'X' ? 'win' : 'lose');
      });
    }
  }

  messBox.style.opacity = '1';
  messBox.style.visibility = 'visible';
  messBox.style.transform =
'translateY(0)';
```

```
  };

  // Reset Game
  const resetGame = () => {

document.body.classList.remove('game
-active');
    startScreen.style.display = 'flex';
  };

  // Event Listeners
  playButton.addEventListener('click',
initGame);
  exitButton.addEventListener('click',
resetGame);
  boxes.forEach(box => {
    box.addEventListener('click',
handlePlayerMove);
```

```
  });

  console.log('Game initialized
successfully');
});
<style>
</head>
<body>
  <div class="stars"></div>

  <div class="game-container">
    <button id="exit" class="btn-
exit"><i class="fas fa-times"></i></
button>

    <div id="start" class="start-
screen">
      <h1 class="title">COSMIC<br>TIC
```

```
TAC TOE</h1>
    <button id="play" class="btn-
play">START GAME</button>
  </div>

  <h1 id="heading" class="game-
title">COSMIC TIC TAC TOE</h1>

  <div id="board" class="game-
board">
    <div class="box" data-
index="0"></div>
    <div class="box" data-
index="1"></div>
    <div class="box" data-
index="2"></div>
    <div class="box" data-
index="3"></div>
```

```
    <div class="box" data-
index="4"></div>
    <div class="box" data-
index="5"></div>
    <div class="box" data-
index="6"></div>
    <div class="box" data-
index="7"></div>
    <div class="box" data-
index="8"></div>
  </div>

  <div id="mess-box" class="message-
box"></div>
 </div>


<script>document.addEventListener('
```

```
DOMContentLoaded', () => {
  // DOM Elements
  const boxes =
document.querySelectorAll('.box');
  const playButton =
document.getElementById('play');
  const exitButton =
document.getElementById('exit');
  const heading =
document.getElementById('heading');
  const board =
document.getElementById('board');
  const messBox =
document.getElementById('mess-
box');
  const startScreen =
document.getElementById('start');
```

```
// Game State
let currentPlayer = 'X';
let gameActive = true;

// Winning Conditions
const winConditions = [
  [0, 1, 2], [3, 4, 5], [6, 7, 8], // rows
  [0, 3, 6], [1, 4, 7], [2, 5, 8], // columns
  [0, 4, 8], [2, 4, 6]             // diagonals
];

// Initialize Game
const initGame = () => {
  // Reset game state
  currentPlayer = 'X';
  gameActive = true;

  // Clear the board
```

```
    boxes.forEach(box => {
      box.textContent = '';
      box.removeAttribute('data-
value');
      box.classList.remove('win', 'lose');
    });

    // Clear message box
    messBox.textContent = '';
    messBox.className = 'message-
box';

    // Show game elements

document.body.classList.add('game-
active');
  };
```

```
  // Handle Player Move
  const handlePlayerMove = (e) => {
    const box = e.target;
    const boxIndex =
box.getAttribute('data-index');

    // If box is already filled or game
not active, ignore
    if (box.textContent !== '' ||
!gameActive) return;

    // Make move
    box.textContent = currentPlayer;
    box.setAttribute('data-value',
currentPlayer);

    // Check for win or draw
    if (checkWin(currentPlayer)) {
```

```
    endGame(false);
    return;
  } else if (isDraw()) {
    endGame(true);
    return;
  }

  // Switch player
  currentPlayer = currentPlayer === 'X' ? 'O' : 'X';

  // If it's computer's turn (O)
  if (currentPlayer === 'O') {
    setTimeout(computerMove, 500);
  }
};

// Computer Move
```

```
  const computerMove = () => {
    if (!gameActive) return;

    // Find all empty boxes
    const emptyBoxes =
Array.from(boxes).filter(box =>
box.textContent === '');

    if (emptyBoxes.length === 0) {
      endGame(true);
      return;
    }

    // Simple AI - random move
    const randomIndex =
Math.floor(Math.random() *
emptyBoxes.length);
    const box =
```

```
emptyBoxes[randomIndex];

  box.textContent = 'O';
  box.setAttribute('data-value', 'O');

  // Check for win or draw
  if (checkWin('O')) {
    endGame(false);
  } else if (isDraw()) {
    endGame(true);
  } else {
    currentPlayer = 'X';
  }
};

// Check for Win
const checkWin = (player) => {
  return
```

```
winConditions.some(condition => {
    return condition.every(index => {
      return
boxes[index].getAttribute('data-
value') === player;
    });
  });
};

// Check for Draw
const isDraw = () => {
  return Array.from(boxes).every(box
=> box.textContent !== '');
};

// End Game
const endGame = (draw) => {
  gameActive = false;
```

```
    if (draw) {
        messBox.textContent = "Game
Draw!";
        messBox.classList.add('draw');
    } else {
        const winner = currentPlayer ===
'X' ? 'You Win!' : 'Computer Wins!';
        messBox.textContent = winner;

messBox.classList.add(currentPlayer
=== 'X' ? 'win' : 'lose');

        // Highlight winning boxes
        const winCondition =
winConditions.find(condition => {
            return condition.every(index => {
                return
```

```
boxes[index].getAttribute('data-
value') === currentPlayer;
      });
    });

    if (winCondition) {
      winCondition.forEach(index => {

boxes[index].classList.add(currentPlay
er === 'X' ? 'win' : 'lose');
      });
    }
  }

  messBox.style.opacity = '1';
  messBox.style.visibility = 'visible';
  messBox.style.transform =
'translateY(0)';
```

```
  };

  // Reset Game
  const resetGame = () => {

document.body.classList.remove('game
-active');
    startScreen.style.display = 'flex';
  };

  // Event Listeners
  playButton.addEventListener('click',
initGame);
  exitButton.addEventListener('click',
resetGame);
  boxes.forEach(box => {
    box.addEventListener('click',
handlePlayerMove);
```

```
  });

  console.log('Game initialized
successfully');
});

</script>
</body>
</html>
```