

## **Project Overview**

The *Tic-Tac-Toe Game* is a simple yet engaging web-based interactive application developed using HTML, CSS, and JavaScript. The purpose of this project is to recreate the classic two-player game in a digital format, emphasizing user interface design, responsive layout, and logical game flow.

This project demonstrates the core principles of front-end web development, including DOM manipulation, event-driven programming, and user interaction handling. The game allows two players to take alternate turns to mark “X” or “O” on a 3×3 grid. The winner is determined automatically based on predefined winning combinations, or the game ends in a draw when all boxes are filled without a winning pattern.

## **Objectives of the Project**

- To develop a fully functional browser-based version of the Tic-Tac-Toe game.
- To apply core concepts of HTML, CSS, and JavaScript in a practical project.
- To enhance understanding of game logic, condition handling, and user interaction in web applications.
- To create a responsive and user-friendly interface that works across different screen sizes.

## **Tools and Technologies Used**

- HTML5: Used to design the structure of the web page and create interactive elements such as buttons and containers.
- CSS3: Used for styling the layout, adding color schemes, and making the game visually appealing and responsive.
- JavaScript (ES6): Used to implement the game's logic, control flow, and interactive functionalities.

## **Detailed Description of Features**

### **1. User Interface (UI) Design**

The interface consists of a simple and minimalistic layout designed for clarity and ease of use.

- The main game board is a 3x3 grid, created using nine HTML `<button>` elements.
- Each cell of the grid serves as a clickable box for player input.
- The heading displays the game title “Tic Tac Toe,” while control buttons like “Reset Game” and “New Game” allow the user to restart or replay the game.
- A message container is displayed dynamically to announce the winner or indicate a draw.

### **2. Responsive Layout**

The CSS uses Flexbox and viewport-based units (vmin) to ensure that the game board and buttons adjust dynamically according to the screen size. This makes the game responsive and playable on both desktop and mobile devices.

### **3. Turn-Based Gameplay**

- The game alternates turns between two players — Player X and Player O.
- A boolean flag (turnO) keeps track of whose turn it is. Initially, Player X starts the game.
- When a player clicks on a box, their respective symbol (“X” or “O”) is displayed, and the button is disabled to prevent overwriting.

### **4. Dynamic Styling for Player Moves**

- Player X and Player O have distinct color schemes for better visual differentiation.
  - Player X: Red text on a grey background (.X-style).
  - Player O: Blue text on a dark background (.O-style).
- These styles are dynamically applied through JavaScript to enhance user experience and clarity during gameplay.

## 5. Winner Detection Logic

The JavaScript code defines all possible winning combinations using an array of index patterns:

```
const winPatterns = [  
  [0, 1, 2], [0, 3, 6], [0, 4, 8], [1, 4, 7], [2, 5, 8], [2, 4, 6], [3, 4, 5], [6, 7, 8],  
];
```

After each move, the program checks these patterns to determine if any player has achieved a winning combination. If three matching symbols are found in any of these patterns, the corresponding player is declared the winner.

## 6. Draw Condition Handling

A counter variable (count) keeps track of the number of moves played. If all nine boxes are filled and no winner has been declared, the game automatically displays a “Game was a Draw” message.

## 7. Game Control Buttons

- Reset Game: Clears the board, resets the turn, and allows the players to start over without refreshing the page.
- New Game: Performs the same function as reset but also hides the winner/draw message and re-enables the boxes for a fresh round.

Both buttons are linked to the resetGame() function that resets all necessary variables and visual states to their default configuration.

## 8. Message Display Functionality

The message container (msg-container) appears dynamically after a win or draw. The message (msg) is updated using JavaScript functions such as:

- showWinner(winner) – Displays the winning player’s name.
- gameDraw() – Displays a message if the game ends in a draw.

When a message is shown, the game board is disabled to prevent further moves until the player starts a new game.

## 9. Code Structure and Modularity

The JavaScript code is well-structured into reusable functions:

- `resetGame()` – Resets game state and UI.
- `disableBoxes()` and `enableBoxes()` – Control interactivity of game buttons.
- `checkWinner()` – Verifies winning patterns after each turn.
- `showWinner(winner)` – Displays winner message dynamically.
- `gameDraw()` – Handles draw condition and user feedback.

This modular approach improves readability, maintainability, and scalability of the project.

### **Working Principle**

1. The player clicks on any empty box on the grid.
2. The script places the corresponding player's symbol and switches turns.
3. After each move, the program checks for winning patterns or a draw.
4. If a winner is detected, a congratulatory message is displayed, and further inputs are disabled.
5. The players can restart the game using the reset or new game buttons.

## **Learning Outcomes**

Through this project, the following concepts and skills were gained:

- **DOM Manipulation:** Understanding how to access and modify HTML elements dynamically.
- **Event Handling:** Using event listeners to handle user interactions effectively.
- **Game Logic Implementation:** Designing algorithms for turn tracking, win detection, and draw conditions.
- **Responsive Web Design:** Building layouts that adapt to various device screen sizes.
- **Code Reusability and Modularity:** Writing maintainable and reusable functions.
- **Debugging and Testing:** Identifying and resolving logic or display errors during development.

## **Conclusion**

The *Tic-Tac-Toe Game* project successfully demonstrates the integration of HTML structure, CSS styling, and JavaScript logic to create an interactive and engaging web-based application. It effectively highlights the fundamental concepts of front-end web development and problem-solving.

This project serves as a strong foundational example of how core web technologies can be combined to build functional and visually appealing applications. It reflects the developer's understanding of logical design, user interface creation, and event-driven programming — essential skills for modern web development.

## **Future Enhancements**

- Add a scoreboard to track wins and draws across multiple rounds.
- Implement AI functionality to allow single-player mode against the computer.
- Improve animations and transitions for a more dynamic user experience.
- Add sound effects for button clicks and win/draw notifications.