Tic-Tac-Toe Game JS Code Explanation

The provided code is an implementation of a simple Tic-Tac-Toe game using JavaScript. Here's an explanation of the code:

statusDisplay: It represents the element that displays the game status, such as the current player's turn or the game result. It is selected using the querySelector method and the CSS class name "game--status".

gameActive, currentPlayer, gameState: These variables are used to track the game state.

* gameActive: It determines if the game is active or paused. If it's false, further moves are ignored.
* currentPlayer: It stores the current player's symbol ("X" or "O").
* gameState: It represents the current state of the game board, with each element corresponding to a cell. An empty cell is represented by an empty string.

winningMessage, drawMessage, currentPlayerTurn: These are functions that generate dynamic messages to be displayed during the game.

* winningMessage: It returns a message indicating which player has won.
* drawMessage: It returns a message indicating a draw game.
* currentPlayerTurn: It returns a message indicating whose turn it is.

statusDisplay.innerHTML = currentPlayerTurn(): Sets the initial message on the game status element to show the current player's turn.

handleCellClick: This function is called when a cell is clicked by the player.

* It receives the clickedCellEvent parameter representing the event object.
* It extracts the clicked cell and its index using the target property and dataset.cellIndex.
* It checks if the clicked cell is already played or if the game is paused. If true, it returns without further action.
* Otherwise, it proceeds to handle the cell played and validate the game result.

handleCellPlayed: Updates the game state and the cell display when a cell is played.

* It sets the current player's symbol in the clicked cell.
* It updates the gameState array with the current player's symbol at the corresponding index.

winningConditions: It is a two-dimensional array that represents the winning combinations in Tic-Tac-Toe.

* Each inner array represents a winning condition, consisting of three cell indices that need to match.
* For example, [0, 1, 2] represents winning when the cells at indices 0, 1, and 2 are all occupied by the same player.

handleResultValidation: This function checks if the game has reached a winning state or a draw.

* It iterates through the winningConditions array to check each winning condition.
* It compares the game state at the indices of each winning condition and checks if they match the current player's symbol.
* If a winning condition is met, it sets the game result message, updates the game status display, and sets gameActive to false.
* If no winning conditions are met and there are no empty cells left, it indicates a draw.
* In that case, it sets the draw message, updates the game status display, and sets gameActive to false.
* If neither winning nor draw conditions are met, it proceeds to handle the player change.

handlePlayerChange: It toggles the current player between "X" and "O" and updates the game status display accordingly.

handleRestartGame: Resets the game state and UI when the restart button is clicked.

* It sets gameActive to true, sets currentPlayer back to "X", and resets gameState to an array of empty strings.
* It updates the game status display with the current player's turn.
* It clears the cell display by setting the textContent of each cell to an empty string.

Event Listeners: The code attaches event listeners to the game cells and restart button using querySelectorAll and addEventListener.

* For the game cells, each cell element triggers the handleCellClick function when clicked.
* The restart button triggers the handleRestartGame function when clicked.

This code provides the basic logic for playing a Tic-Tac-Toe game, handling player turns, validating game results, and allowing the game to be restarted.