Este código implementa um jogo da velha (Tic-Tac-Toe) com dois jogadores: um jogador humano e uma IA (inteligência artificial) que usa o algoritmo Minimax para tomar decisões de jogada. Vou explicar cada parte detalhadamente:

**Variáveis e Elementos HTML**

* **board**: Seleciona o elemento com o ID 'game-board', que contém as células do jogo.
* **cells**: Seleciona todas as células do jogo (elementos com a classe 'cell').
* **resetButton**: Botão que permite resetar o jogo.
* **currentPlayer**: Variável que armazena qual jogador está atualmente jogando, começando com 'X'.
* **gameActive**: Indica se o jogo está ativo (não há vencedor ou empate).
* **boardState**: Um array que guarda o estado atual do tabuleiro (posições ocupadas por 'X', 'O' ou vazias).

**Condições de Vitória**

* **winningConditions**: Um array que define as combinações de células que resultam em vitória (linhas, colunas e diagonais).

**Função handleCellClick**

Esta função é chamada quando o jogador humano clica em uma célula. A lógica dela é:

1. **Selecionar a célula clicada**.
2. **Verificar se a célula já está ocupada ou se o jogo terminou**. Se uma dessas condições for verdadeira, o clique é ignorado.
3. **Atualizar o estado da célula** com o símbolo do jogador atual (X ou O).
4. **Verificar se há um vencedor ou empate** chamando a função checkResult().
5. **Mudar de jogador** usando switchPlayer().
6. **Se for a vez da IA** (currentPlayer === 'O'), a IA faz uma jogada após uma pequena pausa de 500ms, chamando makeAIMove().

**Função updateCell**

Atualiza o conteúdo visual de uma célula no tabuleiro e altera o estado correspondente no array boardState.

**Função switchPlayer**

Alterna entre os jogadores 'X' (humano) e 'O' (IA).

**Função checkResult**

Verifica se há um vencedor ou empate:

1. **Itera pelas condições de vitória** (linhas, colunas e diagonais).
2. **Verifica se todas as três posições da condição estão preenchidas com o mesmo símbolo**.
3. Se houver um vencedor, o jogo termina (define gameActive = false).
4. Se todas as células estiverem preenchidas e não houver vencedor, é um empate.

**Função makeAIMove**

Esta função faz a jogada da IA usando o algoritmo **Minimax**:

1. Calcula a melhor jogada com minimax().
2. Atualiza o tabuleiro com a jogada escolhida.
3. Verifica se a jogada resultou em vitória ou empate.

**Algoritmo Minimax**

O algoritmo **Minimax** é utilizado pela IA para determinar a melhor jogada:

1. **emptyIndexes()** retorna os índices das células vazias no tabuleiro.
2. **Condições de vitória**: Se a IA ('O') venceu, retorna um score de 10; se o jogador humano ('X') venceu, retorna -10; se houver empate, o score é 0.
3. **Simulação de jogadas**: A IA simula cada jogada possível e avalia o resultado de cada uma.
4. **Recursão**: O Minimax chama a si mesmo para simular as jogadas futuras até encontrar o resultado final de cada simulação.
5. **Escolha da melhor jogada**: A IA escolhe a jogada com o maior score (para maximizar sua chance de ganhar) quando é a vez dela, e a jogada com o menor score quando é a vez do oponente (para minimizar a chance do humano ganhar).

**Função emptyIndexes**

Retorna os índices das células que ainda estão vazias no tabuleiro.

**Função winning**

Verifica se há uma condição de vitória específica no tabuleiro para o jogador ('X' ou 'O').

**Função resetGame**

Restaura o estado inicial do jogo:

1. Reseta o tabuleiro visualmente e o array boardState.
2. Define o jogador atual como 'X' e o jogo como ativo novamente.

**Eventos**

* **Cada célula** do tabuleiro escuta o evento de clique e chama a função handleCellClick ao ser clicada.
* **Botão de reset** escuta o clique para reiniciar o jogo, chamando resetGame.