

teste 1555

Criado em 09/09/2025, 16:15

## Observações

ou criar um jogo da velha completo com interface amigável, verifica

## Conversação

🤖 IA - Mensagem 1

■

# Jogo da Velha Completo

Vou criar um jogo da velha completo com interface amigável, verificação automática de vitória/empate e opção de reinício. O código será bem comentado para facilitar o entendimento.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <title>Jogo da Velha</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif;
    }

    body {
      background: linear-gradient(135deg,
#1a2a6c, #b21f1f, #fdbb2d);
      min-height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      padding: 20px;
    }
  </style>
</head>
<body>
```

```
.container {
  background-color: rgba(255, 255, 255,
0.9);
  border-radius: 15px;
  box-shadow: 0 10px 30px rgba(0, 0, 0,
0.3);
  padding: 30px;
  text-align: center;
  max-width: 500px;
  width: 100%;
}

h1 {
  color: #2c3e50;
  margin-bottom: 20px;
  font-size: 2.5rem;
  text-shadow: 2px 2px 4px rgba(0, 0,
0, 0.1);
}

.status {
  font-size: 1.4rem;
  margin: 20px 0;
  padding: 10px;
  border-radius: 8px;
  background-color: #f8f9fa;
  box-shadow: 0 4px 6px rgba(0, 0, 0,
0.1);
  transition: all 0.3s ease;
}

.board {
  display: grid;
  grid-template-columns: repeat(3,
1fr);
  grid-gap: 10px;
  margin: 20px auto;
  max-width: 300px;
}

.cell {
  background-color: #3498db;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0,
0.2);
  cursor: pointer;
  aspect-ratio: 1/1;
```

```
display: flex;
    justify-content: center;
    align-items: center;
    font-size: 3rem;
    font-weight: bold;
    color: white;
    transition: all 0.3s ease;
}

.cell:hover {
    background-color: #2980b9;
    transform: translateY(-5px);
}

.cell.x {
    background-color: #e74c3c;
}

.cell.o {
    background-color: #2ecc71;
}

.reset-btn {
    background: linear-gradient(to right,
#3498db, #2c3e50);
    border: none;
    border-radius: 8px;
    color: white;
    cursor: pointer;
    font-size: 1.2rem;
    font-weight: bold;
    padding: 12px 25px;
    margin-top: 20px;
    transition: all 0.3s ease;
    box-shadow: 0 4px 8px rgba(0, 0, 0,
0.2);
}

.reset-btn:hover {
    transform: translateY(-3px);
    box-shadow: 0 6px 12px rgba(0, 0, 0,
0.3);
}

.reset-btn:active {
    transform: translateY(0);
}
```

```

        .win {
            background-color: #f1c40f;
            color: #2c3e50;
            font-weight: bold;
        }

        footer {
            margin-top: 30px;
            color: #7f8c8d;
            font-size: 0.9rem;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Jogo da Velha</h1>

        <div class="status" id="status">Vez do
Jogador X</div>

        <div class="board" id="board">
            <!-- As células serão geradas via
JavaScript -->
            </div>

            <button class="reset-btn"
id="reset">Reiniciar Jogo</button>

            <footer>
                <p>Desenvolvido em JavaScript - Dois
jogadores</p>
            </footer>
        </div>

        <script>
            // Constantes e variáveis globais
            const statusDisplay =
document.getElementById('status');
            const resetButton =
document.getElementById('reset');
            let gameActive = true;
            let currentPlayer = "X";
            let gameState = ["", "", "", "", "", "",
"", "", ""];

            // Mensagens do jogo
            const winningMessage = () => `Jogador
${currentPlayer} venceu!`;

```

```

const drawMessage = () => `Empate!`;
const currentPlayerTurn = () => `Vez do
Jogador ${currentPlayer}`;

// Condições de vitória (índices das
células que formam uma linha)
const winningConditions = [
  [0, 1, 2], // linha superior
  [3, 4, 5], // linha do meio
  [6, 7, 8], // linha inferior
  [0, 3, 6], // coluna esquerda
  [1, 4, 7], // coluna do meio
  [2, 5, 8], // coluna direita
  [0, 4, 8], // diagonal \
  [2, 4, 6] // diagonal /
];

// Inicialização do jogo
function initGame() {
  const board =
document.getElementById('board');
  board.innerHTML = '';

  // Cria as 9 células do tabuleiro
  for (let i = 0; i < 9; i++) {
    const cell =
document.createElement('div');
    cell.classList.add('cell');
    cell.setAttribute('data-cell-
index', i);
    cell.addEventListener('click',
handleCellClick);
    board.appendChild(cell);
  }

  // Atualiza o status do jogo
  statusDisplay.innerHTML =
currentPlayerTurn();

  statusDisplay.classList.remove('win');

  // Reinicia o estado do jogo
  gameActive = true;
  currentPlayer = "X";
  gameState = ["", "", "", "", "", "",
"", "", ""];
}

```

```

// Manipula o clique em uma célula
function
handleCellClick(clickedCellEvent) {
    // Obtém a célula clicada
    const clickedCell =
clickedCellEvent.target;
    const clickedCellIndex =
parseInt(clickedCell.getAttribute('data-cell-
index'));

    // Verifica se a célula já foi
preenchida ou se o jogo acabou
    if (gameState[clickedCellIndex] !==
"" || !gameActive) {
        return;
    }

    // Processa a jogada
    handleCellPlayed(clickedCell,
clickedCellIndex);
    handleResultValidation();
}

// Processa a jogada em uma célula
function handleCellPlayed(clickedCell,
clickedCellIndex) {
    // Atualiza o estado interno do jogo
e a interface
    gameState[clickedCellIndex] =
currentPlayer;
    clickedCell.innerHTML =
currentPlayer;

    clickedCell.classList.add(currentPlayer.toLowerCa
se());
}

// Valida o resultado após cada jogada
function handleResultValidation() {
    let roundWon = false;
    let winningLine = [];

    // Verifica todas as condições de
vitória
    for (let i = 0; i <
winningConditions.length; i++) {
        const winCondition =
winningConditions[i];

```



```

        let a =
gameState[winCondition[0]];
        let b =
gameState[winCondition[1]];
        let c =
gameState[winCondition[2]];

        // Se alguma célula estiver
vazia, continua
        if (a === '' || b === '' || c ===
'') {
            continue;
        }

        // Se três células iguais
formarem uma linha
        if (a === b && b === c) {
            roundWon = true;
            winningLine = winCondition;
            break;
        }
    }

    // Se houver vitória, finaliza o jogo
    if (roundWon) {
        gameActive = false;
        statusDisplay.innerHTML =
winningMessage();

statusDisplay.classList.add('win');

        // Destaca as células vencedoras
        for (let index of winningLine) {

document.querySelector(`[data-cell-
index="${index}"]`).classList.add('win');
        }

        return;
    }

    // Verifica se houve empate (todas as
células preenchidas sem vencedor)
    let roundDraw
=gameState.includes("");
    if (roundDraw) {
        gameActive = false;
        statusDisplay.innerHTML =
drawMessage();
    }

```

```

drawMessage(),

statusDisplay.classList.add('win');
    return;
}

// Se não houve vitória nem empate,
muda o jogador
    handlePlayerChange();
}

// Alterna entre os jogadores
function handlePlayerChange() {
    currentPlayer = currentPlayer === "X"
? "O" : "X";
    statusDisplay.innerHTML =
currentPlayerTurn();
}

// Reinicia o jogo
resetButton.addEventListener('click',
initGame);

// Inicializa o jogo quando a página
carrega
    window.onload = initGame;
</script>
</body>
</html>

```

## Explicação do Código

### HTML

- Define a estrutura básica da página com título, área de status, tabuleiro e botão de reinício.

### CSS

- Estiliza a página com um fundo gradiente e design moderno.
- Cria um visual atraente para as células do jogo e botões.
- Fornece feedback visual para os jogadores (cores diferentes para X e O, destaque para vitória).

### JavaScript



- **Variáveis globais:** Controlam o estado do jogo (jogador atual, estado do tabuleiro, etc.).
- **initGame():** Inicializa o jogo, criando as células e resetando o estado.
- **handleCellClick():** Gerencia o clique do jogador em uma célula.
- **handleCellPlayed():** Atualiza a interface e o estado interno do jogo.
- **handleResultValidation():** Verifica se houve vitória ou empate após cada jogada.
- **handlePlayerChange():** Alterna entre os jogadores X e O.

O jogo está completo e pronto para uso. Basta copiar o código e abrir em um navegador. Dois jogadores podem alternar clicando nas células, e o jogo detecta automaticamente vitórias, empates e permite reiniciar a partida.

