Group Lab #6. Create a Game with ChatGPT

Contents

0. Intro

We will develop a several simple Games in JavaScript with the focus on asynchronous programming.
To start, create a folder game in VSC and 3 files inside:

- game1.html for HTML
- game1.css for CSS and
- game1.js for JS

1. Tic Tac Toe

1.1. Create an HTML for it

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tic Tac Toe</title>
    <link rel="stylesheet" href="game1.css">
```

```
</head>
<body>
    <h1>Tic Tac Toe</h1>
    <div id="game-board" class="game-board">
        <div class="cell" onclick="makeMove(this, 0)"></div>
        <div class="cell" onclick="makeMove(this, 1)"></div>
        <div class="cell" onclick="makeMove(this, 2)"></div>
        <div class="cell" onclick="makeMove(this, 3)"></div>
        <div class="cell" onclick="makeMove(this, 4)"></div>
        <div class="cell" onclick="makeMove(this, 5)"></div>
        <div class="cell" onclick="makeMove(this, 6)"></div>
        <div class="cell" onclick="makeMove(this, 7)"></div>
        <div class="cell" onclick="makeMove(this, 8)"></div>
    </div>
    <h2 id="status"></h2>
    <button onclick="resetGame()">Reset Game</button>
    <script src="game1.js"></script>
</body>
</html>
```

Task 1.1. Run on Live Server. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does in 1-3 sentences (what elements of HTML, CSS, JS you see and recognize, etc.).

1.2. Create a CSS for it

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 50px;
}

h1 {
    font-size: 2em;
}

.game-board {
    display: grid;
    grid-template-columns: repeat(3, 100px);
    grid-gap: 5px;
    justify-content: center;
    margin-bottom: 20px;
}

.cell {
    width: 100px;
    height: 100px;
    background-color: #f0f0f0;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 2em;
    cursor: pointer;
}

.cell.taken {
    pointer-events: none;
}

button {
    font-size: 16px;
    padding: 10px 20px;
}
```

Task 1.2. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does (if anything) in 1-3 sentences. How did the CSS affect the game layout?

1.3. Create a JS for it

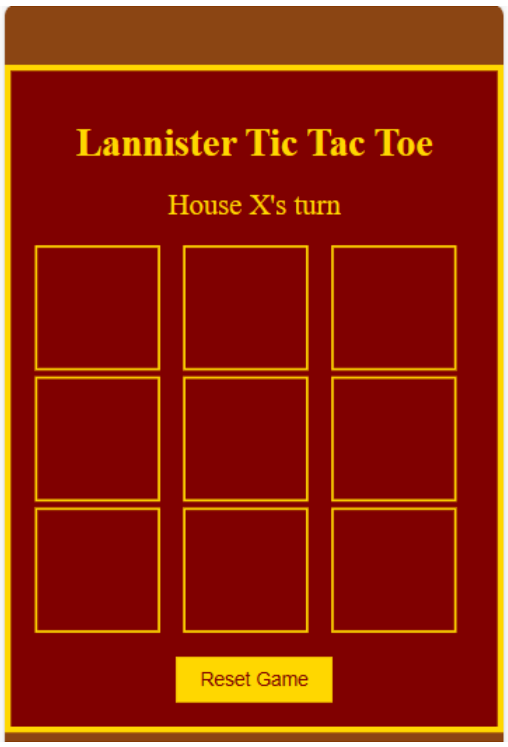
```
let board = ['', '', '', '', '', '', '', '', ''];
let currentPlayer = 'X';
let gameActive = true;

function makeMove(cell, index) {
    if (board[index] !== '' || !gameActive) {
        return;
    }

    board[index] = currentPlayer;
    cell.innerText = currentPlayer;
    cell.classList.add('taken');

    checkWinner();
}

function checkWinner() {
    const winningCombinations = [
        [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
    ];
    for (const combination of winningCombinations) {
        const [a, b, c] = combination;
        if (board[a] && board[a] === board[b] && board[a] === board[c]) {
            document.getElementById('status').innerText = `Player ${currentPlayer} Wins!`;
            gameActive = false;
            return;
        }
    }

    if (!board.includes('')) {
        document.getElementById('status').innerText = 'Draw!';
        gameActive = false;
        return;
    }

    currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
}

function resetGame() {
    board = ['', '', '', '', '', '', '', '', ''];
    currentPlayer = 'X';
    gameActive = true;
    document.getElementById('status').innerText = '';
    document.querySelectorAll('.cell').forEach(cell => {
        cell.innerText = '';
        cell.classList.remove('taken');
    });
}
```

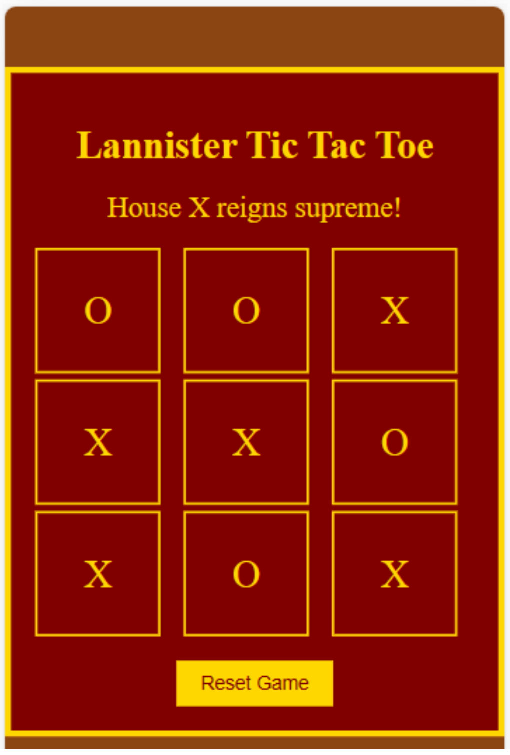
Task 1.3. Run on Live server, test the game several times (different test cases), provide your results.

Task 2.1. Convert all functions to arrow functions. Provide screenshot of your code and output

Task 2.2. Customize the layout and / or functionality of the game so that the game looks like a brand-new game just created by you and your team. It should have some unique ideas like Barbie style, Christmas Style, Halloween style, Horror movie style, be 4 by 4, Etc. Hint: you can use ChatGPT if you want to. Provide a screenshot of your code and output. Deploy on obi2 and provide a link to it.

Sample run:





2. Rock, Paper, or Scissors

2.1. Create an HTML for it

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="game2.css">
    <title>Rock, Paper, Scissors</title>
</head>
<body>
    <div class="game-container">
        <h1>Rock, Paper, Scissors</h1>
        <div class="choices">
            <div class="choice" id="rock" onclick="playRound('rock')">Rock</div>
            <div class="choice" id="paper" onclick="playRound('paper')">Paper</div>
            <div class="choice" id="scissors" onclick="playRound('scissors')">Scissors</div>
        </div>
        <div class="result">
            <h2 id="result-text">Choose an option to start the game</h2>
        </div>
    </div>
    <script src="game2.js"></script>
</body>
</html>
```

Task 3.1. Run on Live Server. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does in 1-3 sentences.

2.2. Create a CSS for it

```
body {
    font-family: 'Arial', sans-serif;
    background-color: #eee;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.game-container {
    text-align: center;
}

.choices {
    display: flex;
    justify-content: center;
    margin-top: 20px;
}

.choice {
    background-color: #fff;
    border: 2px solid #333;
    padding: 10px 20px;
    margin: 0 10px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.choice:hover {
    background-color: #f4f4f4;
}

.result {
    margin-top: 30px;
}
```

Task 3.2. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does in 1-3 sentences. How did the CSS affect the game?

2.3. Create a JS for it

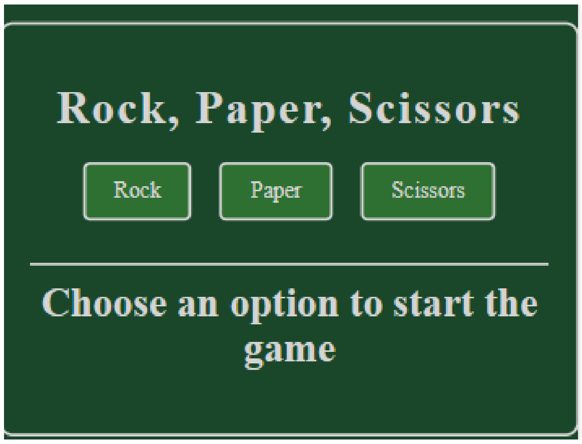
```
const resultText = document.getElementById('result-text');
const choices = ['rock', 'paper', 'scissors'];
const playRound = (playerChoice) => {
    const computerChoice = choices[Math.floor(Math.random() * 3)];
    if (playerChoice === computerChoice) {
        resultText.textContent = `It's a draw! Both chose ${playerChoice}.`;
    } else if (
        (playerChoice === 'rock' && computerChoice === 'scissors') ||
        (playerChoice === 'paper' && computerChoice === 'rock') ||
        (playerChoice === 'scissors' && computerChoice === 'paper')
    ) {
        resultText.textContent = `You win! ${playerChoice} beats ${computerChoice}.`;
    } else {
        resultText.textContent = `You lose! ${computerChoice} beats ${playerChoice}.`;
    }
};
```

Task 3.3. Run on Live Server, test the game several times, provide your results.

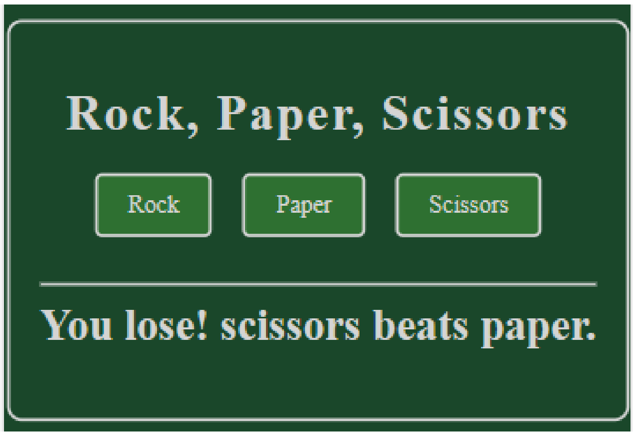
Task 4.1. Implement a Score System: Keep track of the player's and computer's scores. Update the scores on the webpage after each round. Display a message when a player reaches a certain score, declaring them as the winner.

Task 4.2. Customize the layout and / or functionality of the game so the game looks like a brand-new game just created by you and your team. It should have some ideas like having unusual style or images for Rock, Paper, and Scissors, Etc. Hint: you can use ChatGPT if you want to. Provide screenshot of your code and output, deploy on obi2 and provide a link to it.

Sample run:



Choose an option to start the game



You lose! scissors beats paper.

3. Matching Pairs

3.1. Create an HTML for it

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="game3.css">
    <title>Barbie Matching Game</title>
</head>
<body>
    <div class="game-container">
        <h1>Barbie Matching Game</h1>
        <div id="grid" class="grid"></div>
        <button id="restart" class="restart">Restart</button>
    </div>
    <script src="game3.js"></script>
</body>
</html>
```

Task 5.1. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does in 1-3 sentences.

3.2. Create a CSS for it

```
body {
    font-family: 'Arial', sans-serif;
    background-color: #FFC0CB; /* Light Pink Background */
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #fff;
}

.game-container {
    text-align: center;
}

h1 {
    font-size: 2em;
    color: #FF69B4; /* Hot Pink Color for Barbie Theme */
    margin-bottom: 20px;
```

```
}

.grid {
    display: grid;
    grid-template-columns: repeat(4, 100px);
    grid-gap: 10px;
    margin: 0 auto;
}

.card {
    width: 100px;
    height: 100px;
    position: relative;
    perspective: 1000px;
    cursor: pointer;
}

.card .front,
.card .back {
    width: 100%;
    height: 100%;
    position: absolute;
    backface-visibility: hidden;
    border-radius: 10px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 2em;
    font-weight: bold;
}

.card .front {
    background-color: #FF69B4; /* Hot Pink */
}

.card .back {
    background-color: #fff;
    color: #FF69B4; /* Hot Pink */
    transform: rotateY(180deg);
}

.card.flipped .front {
    transform: rotateY(180deg);
}

.card.flipped .back {
    transform: rotateY(360deg);
```

```
}

.restart {
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 1em;
    border: none;
    border-radius: 5px;
    background-color: #FF69B4; /* Hot Pink */
    color: #fff;
    cursor: pointer;
}

.restart:hover {
    background-color: #FF1493; /* Deep Pink */
}
```

Task 5.2. Observe the code and output in the browser, check Console for errors. Describe what the code has in it and what it does in 1-3 sentences. How did the CSS affect the game?

3.3. Create a JS for it

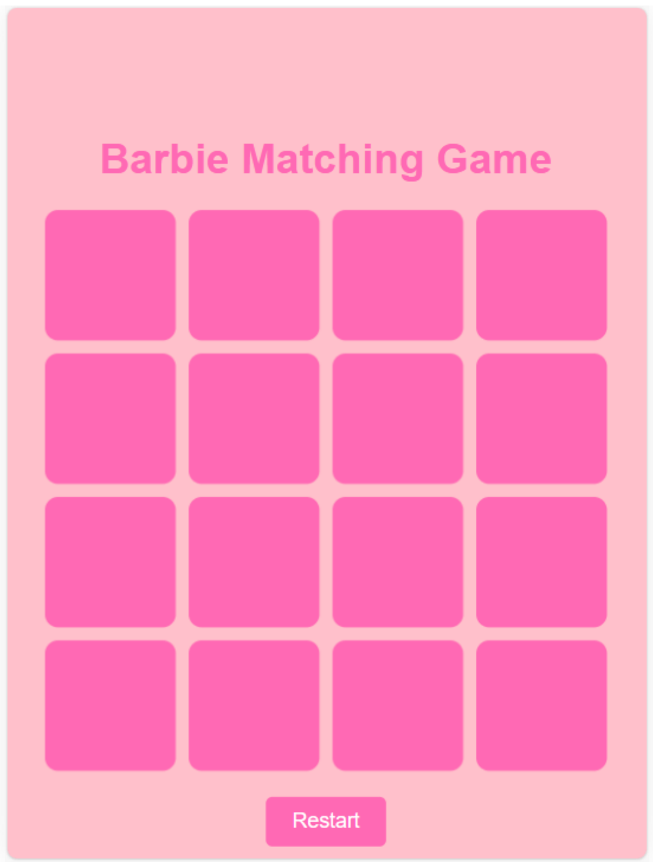
```
document.addEventListener('DOMContentLoaded', () => {
    const grid = document.getElementById('grid');
    const restartButton = document.getElementById('restart');
    const words = ['hat', 'sat', 'rat', 'that', 'mat', 'fat', 'bat', 'cat', 'hat', 'sat', 'rat', 'that',
'mat', 'fat', 'bat', 'cat'];
    let flippedCards = [];
    let matchedPairs = 0;

    const shuffleArray = array => {
        for (let i = array.length - 1; i > 0; i--) {
            const j = Math.floor(Math.random() * (i + 1));
            [array[i], array[j]] = [array[j], array[i]];
        }
    };

    const createBoard = () => {
        shuffleArray(words);
        for (let i = 0; i < words.length; i++) {
            const card = document.createElement('div');
            card.classList.add('card');
            card.dataset.name = words[i];
            card.innerHTML = `<div class="front"></div><div class="back">${words[i]}</div>`;
            grid.appendChild(card);
        }
    };
    const flipCard = e => {
        const clickedCard = e.target.closest('.card');
        if (clickedCard && !clickedCard.classList.contains('flipped') && flippedCards.length < 2) {
            clickedCard.classList.add('flipped');
            flippedCards.push(clickedCard);
            if (flippedCards.length === 2) {
                setTimeout(checkForMatch, 500);
            }
        }
    };

    const checkForMatch = () => {
        const [card1, card2] = flippedCards;
        if (card1.dataset.name === card2.dataset.name) {
            matchedPairs++;
            if (matchedPairs === words.length / 2) {
                setTimeout(() => alert('Congratulations! You found all the pairs!'), 500);
            }
        } else {
            card1.classList.remove('flipped');
            card2.classList.remove('flipped');
        }
        flippedCards = [];
    };

    const restartGame = () => {
        grid.innerHTML = '';
        flippedCards = [];
        matchedPairs = 0;
        createBoard();
    };

    grid.addEventListener('click', flipCard);
    restartButton.addEventListener('click', restartGame);

    createBoard();
});
```

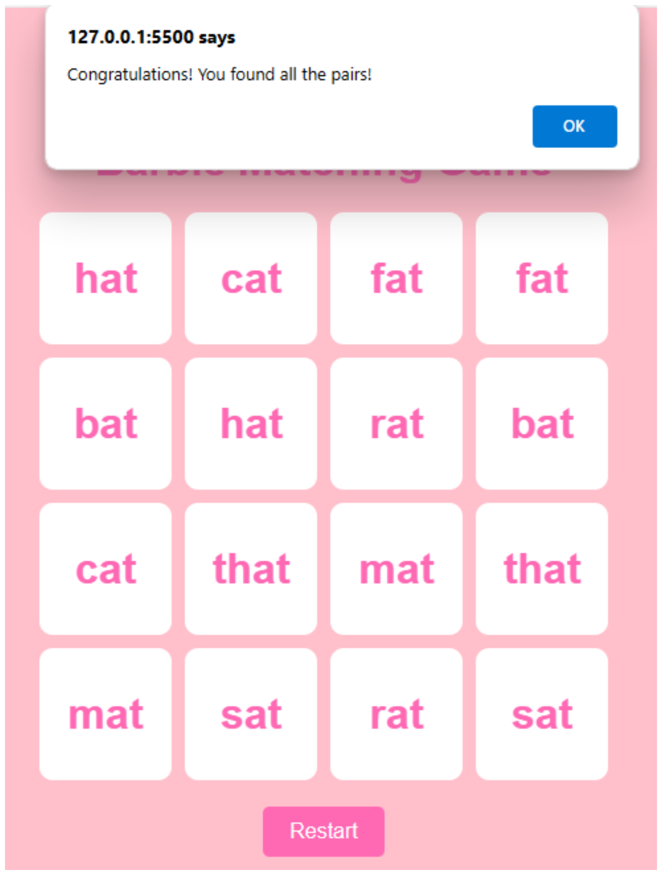
Task 5.3. Run on Live Server, test the game several times, provide your results.

Task 6.1. Implement a Time: Create a start button that starts the game, the game should not start before the start button is pressed. Display the time it took to complete the game at.

Task 6.2. Customize the layout and / or functionality of the game so the game looks like a brand-new game just created by you and your team. It should have some ideas like having unusual style or images for pairs, Etc. on obi2 and provide a link to it.Hint: you can use ChatGPT if you want to. Provide screenshot of your code and output.

Sample run:





127.0.0.1:5500 says

Congratulations! You found all the pairs!

4. Custom Game

Task 7. Create your own game with ChatGPT. Try to be creative. The game can be complex or simple but it should work. Provide a zip file of all your games in the same folder and a PDF with screenshots of source code and output.