whitch function I need if O need a memmory game in js?

To create a memory game in JavaScript, you will need several functions for different parts of the game. Here are some key functions you might want to implement:

**Initialization Function**: Set up the game board, shuffle the cards, and reset scores.

`javascript

function initializeGame() {

// Logic to create, shuffle and display cards

}

`

**Card Flip Function**: Handle the logic for flipping a card when clicked.

`javascript

function flipCard(card) {

// Logic to flip the card and check for a match

}

`

**Match Check Function**: Compare two flipped cards to see if they match.

`javascript

function checkForMatch(card1, card2) {

// Logic to check if the two cards are the same

}

`

**Score Update Function**: Update the player's score or moves.

`javascript

function updateScore() {

// Logic to update score based on matches found

}

`

**End Game Function**: Determine when the game is over and display results.

`javascript

function endGame() {

// Logic to display win/lose message and reset game

}

`

These functions will work together to create the interactive experience of a memory game. You may also want to include event listeners for user interactions.

dont I need a functcion thet will delet the cards that I mutch yet from the bord?

Yes, you will need a function to remove matched cards from the board. This function can help in updating the game state when a match is found. Here’s an example of how you can implement it:

function removeMatchedCards(card1, card2) {

// Logic to remove matched cards from the board

card1.classList.add('matched');

card2.classList.add('matched');

// Optionally, you can set a timeout to hide these cards after a short delay

setTimeout(() => {

card1.style.display = 'none';

card2.style.display = 'none';

}, 1000); // Hides matched cards after 1 second

}

This function can be called within your match check logic after determining that two cards match.

give me the code for the game on one page of html,css and js

Below is a simple implementation of a memory game in one HTML file with embedded CSS and JavaScript.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

}

.matched {

background-color: #4caf50;

pointer-events: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

const cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.classList.add('matched');

card2.classList.add('matched');

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

createBoard();

</script>

</body>

</html>

This code creates a simple memory game where you can click on cards to reveal symbols. If two cards match, they remain visible; if not, they flip back after a short delay.

add a game over function and function that will delete the has mutched cards

Here's an updated version of the memory game that includes a game over function and a function to delete matched cards from the board once all pairs are found:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

}

.matched {

background-color: #4caf50;

pointer-events: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.classList.add('matched');

card2.classList.add('matched');

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

setTimeout(() => {

alert('Congratulations! You found all pairs!');

resetGame();

}, 500);

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

createBoard();

}

createBoard();

</script>

</body>

</html>

In this implementation, the gameOver function displays a congratulatory message when all pairs are found, and the resetGame function can be used to start a new game after the game is over, ensuring the matched cards are cleared from the board.

I dont wont that the game over function will use alert

Certainly! Here’s an updated version of the memory game where the game over function updates the DOM instead of using an alert. It will display a message on the page when the game is over:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

}

.matched {

background-color: #4caf50;

pointer-events: none;

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.classList.add('matched');

card2.classList.add('matched');

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

}

createBoard();

</script>

</body>

</html>

In this version, when the game is over, a message is displayed on the page itself instead of using an alert. This message will inform the player that they have found all the pairs.

add a bottomn to start a new game after the last game finished

Here’s the updated memory game code that includes a button to start a new game after the last game is finished:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

}

.matched {

background-color: #4caf50;

pointer-events: none;

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.classList.add('matched');

card2.classList.add('matched');

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

In this version, a button labeled "New Game" appears when the game is over. When clicked, it resets the game state and starts a new game. The button is initially hidden and will only display after all pairs have been found.

disain it nicer tha bottomn is not nice

Sure! Below is the updated memory game code with an improved design for the "New Game" button to make it look nicer, including some styling enhancements:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

.matched {

background-color: #4caf50;

pointer-events: none;

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.classList.add('matched');

card2.classList.add('matched');

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

In this version, the "New Game" button has a nicer design with improved styling. It has a background color, rounded corners, and hover effects to enhance its appearance and interactivity.

I wont that each peair that the user found will delet from the game bord

Here’s the updated memory game code that completely removes each matched pair from the game board instead of simply marking them as matched. When pairs are found, they will be deleted from the board.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

removeMatchedCards(firstCard, secondCard);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

Changes Made:

When a pair of cards is matched, both cards are removed from the DOM using the remove() method, effectively deleting them from the game board.

The game logic remains the same, ensuring that players can still play the game until all pairs are found and removed.

This implementation keeps the board clean by removing matched cards instead of just marking them visually, enhancing the game's playability.

add a delay beafor the delet

Here’s the updated memory game code that adds a delay before the matched cards are deleted from the board. This allows players to see the matched cards for a brief moment before they disappear:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '🥝'];

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

setTimeout(() => {

removeMatchedCards(firstCard, secondCard);

}, 1000); // Delay before deleting matched cards

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

Changes Made:

A setTimeout is used in the checkForMatch function with a delay (1000 milliseconds) before calling the removeMatchedCards function, allowing the player to see the matched cards for a moment before they disappear.

This setup ensures a smoother user experience by providing a visual confirmation before the matched cards are removed from the board.

I wont the image will be imojis

The current implementation already uses emojis as the symbols for the memory game. If you would like to customize the emoji selections or change them, you can modify the symbols array at the beginning of the script. Here’s an example of the game code, retaining the use of emojis:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['🍎', '🍌', '🍒', '🍇', '🍏', '🍍', '🍓', '']; // Emojis as symbols

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

setTimeout(() => {

removeMatchedCards(firstCard, secondCard);

}, 1000);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

This code uses emojis as the values displayed on each card. You can change the emojis in the symbols array to any other emoji of your choosing if you want to customize the game further.

I wont smille images and mor cards

To use smiley images and increase the number of cards in the memory game, you can modify the symbols array with emoji representations of smiley faces or replace them with image URLs. Below is an example of modifying the code to use more smiley emojis and bumping up the number of pairs from 8 to 16.

Here's the updated code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 100px);

gap: 10px;

}

.card {

width: 100px;

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['😀', '😁', '😂', '😍', '😜', '😎', '🤪', '😇',

'😃', '😄', '😅', '😻', '🤩', '😏', '😋', '']; // 16 smiley emojis

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

setTimeout(() => {

removeMatchedCards(firstCard, secondCard);

}, 1000);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

Changes Made:

The symbols array now contains 16 different smiley emojis, doubling the number of cards in the game to provide more pairs to match.

Each emoji is represented in Unicode format in the array, which will be displayed on the cards.

With this setup, you'll have a memory game that features a variety of smiley emojis as requested.

I wont the card will be on the with of the screen

To make the cards occupy the full width of the screen while maintaining the memory game layout, you can adjust the CSS grid properties. Here's the updated code that makes the cards align across the full width of the available space:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(4, 1fr); /\* Adjusted to use 1fr for equal widths \*/

width: 100%; /\* Set the width of the board to 100% \*/

max-width: 800px; /\* You can set a max width to avoid stretching too wide \*/

gap: 10px;

padding: 10px; /\* Added padding for better spacing \*/

}

.card {

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['😀', '😁', '😂', '😍', '😜', '😎', '🤪', '😇',

'😃', '😄', '😅', '😻', '🤩', '😏', '😋', '']; // 16 smiley emojis

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

setTimeout(() => {

removeMatchedCards(firstCard, secondCard);

}, 1000);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

Changes Made:

The .board class now uses grid-template-columns: repeat(4, 1fr); to ensure that each card takes an equal fraction of the available space. This makes the cards responsive across different screen sizes.

The width of the board is set to 100% with a max-width: 800px; to control how wide it can get, ensuring proper visualization without stretching too much on larger displays.

Added padding to the board to provide some spacing around the cards.

With this setup, the cards will occupy the full width of the screen while maintaining their number and appearance.

no i mean that tthat the cards will stend in longers rows no that the cards will be widers

To arrange the cards in longer rows, you can modify the CSS to change the number of columns and add additional emoji pairs to fill those rows. Below is an updated version of the memory game code with more rows of cards:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Memory Game</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

flex-direction: column;

background-color: #f0f0f0;

}

h1 {

margin-bottom: 20px;

}

.board {

display: grid;

grid-template-columns: repeat(8, 1fr); /\* Increased to 8 columns for longer rows \*/

width: 100%; /\* Set the width of the board to 100% \*/

max-width: 1200px; /\* You can set a max width for appearance \*/

gap: 10px;

padding: 10px; /\* Added padding for better spacing \*/

}

.card {

height: 100px;

background-color: #fff;

border: 1px solid #ddd;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

cursor: pointer;

user-select: none;

transition: background-color 0.3s, transform 0.3s;

}

.card:hover {

transform: scale(1.05);

}

#message {

margin-top: 20px;

font-size: 24px;

color: #333;

}

#newGameButton {

display: none;

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

background-color: #008cba;

color: white;

border: none;

border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

#newGameButton:hover {

background-color: #005f75;

transform: scale(1.05);

}

#newGameButton:focus {

outline: none;

}

</style>

</head>

<body>

<h1>Memory Game</h1>

<div class="board" id="board"></div>

<div id="message"></div>

<button id="newGameButton">New Game</button>

<script>

const symbols = ['😀', '😁', '😂', '😍', '😜', '😎', '🤪', '😇',

'😃', '😄', '😅', '😻', '🤩', '😏', '😋', '',

'😊', '😣', '😆', '']; // 20 smiley emojis

let cardValues = [...symbols, ...symbols];

let firstCard = null;

let secondCard = null;

let lockBoard = false;

let matchedCardsCount = 0;

function shuffle(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]];

}

return array;

}

function createBoard() {

const board = document.getElementById('board');

shuffle(cardValues).forEach(value => {

const card = document.createElement('div');

card.classList.add('card');

card.setAttribute('data-value', value);

card.addEventListener('click', flipCard);

board.appendChild(card);

});

}

function flipCard() {

if (lockBoard || this === firstCard) return;

this.textContent = this.getAttribute('data-value');

this.classList.add('flipped');

if (!firstCard) {

firstCard = this;

return;

}

secondCard = this;

lockBoard = true;

checkForMatch();

}

function checkForMatch() {

if (firstCard.getAttribute('data-value') === secondCard.getAttribute('data-value')) {

setTimeout(() => {

removeMatchedCards(firstCard, secondCard);

}, 1000);

} else {

setTimeout(() => {

unflipCards(firstCard, secondCard);

}, 1000);

}

}

function removeMatchedCards(card1, card2) {

card1.remove();

card2.remove();

matchedCardsCount += 2;

if (matchedCardsCount === cardValues.length) {

gameOver();

}

resetBoard();

}

function unflipCards(card1, card2) {

card1.textContent = '';

card2.textContent = '';

resetBoard();

}

function resetBoard() {

[firstCard, secondCard, lockBoard] = [null, null, false];

}

function gameOver() {

const message = document.getElementById('message');

message.textContent = 'Congratulations! You found all pairs!';

document.getElementById('newGameButton').style.display = 'block';

}

function resetGame() {

matchedCardsCount = 0;

cardValues = [...symbols, ...symbols];

document.getElementById('board').innerHTML = '';

document.getElementById('message').textContent = '';

document.getElementById('newGameButton').style.display = 'none';

createBoard();

}

document.getElementById('newGameButton').addEventListener('click', resetGame);

createBoard();

</script>

</body>

</html>

Changes Made:

Increased the number of columns in the .board class to repeat(8, 1fr); to allow for longer rows of cards.

Increased the symbols array to contain 20 different smiley emojis, providing a larger set of cards.

With these changes, the game will display more cards arranged in longer rows across the screen.