This is a JavaScript game where we would have a bunch of blank cards with images under them. The goal is to pick two cards that have the same image. If you do that, your score increases.

To start with our html is pretty simple. We just have a single div element and in our css, we define width, height and display as flex. You can see the full repository <a href=”https://github.com/thedumebi/memory-game”>here</a>.

In our app.js file, we are going to add an event listener to our document so that once it is loaded, all that we want would happen. First we are going to make a card array which would contain twelve (12) duplicate objects. After doing that, we are going to sort the array so that the objects are scattered.

document.addEventListener("DOMContentLoaded", () => {

    // card options

    const cardArray = [

        {

            name: "fries",

            img: "images/fries.png"

        },

        {

            name: "fries",

            img: "images/fries.png"

        },

        {

            name: "cheeseburger",

            img: "images/cheeseburger.png"

        },

        {

            name: "cheeseburger",

            img: "images/cheeseburger.png"

        },

        {

            name: "hotdog",

            img: "images/hotdog.png"

        },

        {

            name: "hotdog",

            img: "images/hotdog.png"

        },

        {

            name: "ice-cream",

            img: "images/ice-cream.png"

        },

        {

            name: "ice-cream",

            img: "images/ice-cream.png"

        },

        {

            name: "milkshake",

            img: "images/milkshake.png"

        },

        {

            name: "milkshake",

            img: "images/milkshake.png"

        },

        {

            name: "pizza",

            img: "images/pizza.png"

        },

        {

            name: "pizza",

            img: "images/pizza.png"

        },

    ]

    cardArray.sort(() => 0.5 -Math.random())

})

Next, we get our div which has a class of grid and our result using querySelector then we create three empty lists which would hold our chosen cards, the id of the cards and then the cards we won.

const grid = document.querySelector(".grid")

    const resultDisplay = document.querySelector("#result")

    var cardsChosen = []

    var cardsChosenId = []

    var cardsWon = []

Next, we create our board. We would use a for loop to do this (each selected image in my case has the same width and height) and in the loop, we create an image element and add a click event listener which would call a flipcard function and then we append the image to our grid. each image element created has the same src attribute so all of them look the same. However, each of them has a different data-id attribute. This would be useful when we want to check if two cards are the same.

 function createBoard() {

        for (let i = 0; i < cardArray.length; i++) {

            var card = document.createElement("img");

            card.setAttribute("src", "images/blank.png");

            card.setAttribute("data-id", i)

            card.addEventListener("click", flipCard)

            grid.appendChild(card)

        }

    }

In our <code>flipCard()</code> function, when we click, we would get the data-id attribute and then push the card name based on the id to our chosen cards array. We would also push the id to our cards chosen id. Next we set the <code>src</code> image attribute to the one that matches the card array. If the length of our chosen cards array is equal to two (2) because we want to check if two cards match, we call a function that checks for match.

 function flipCard() {

        var cardId = this.getAttribute("data-id")

        cardsChosen.push(cardArray[cardId].name)

        cardsChosenId.push(cardId)

        this.setAttribute("src", cardArray[cardId].img)

        if (cardsChosen.length ===2) {

            setTimeout(checkForMatch, 500)

        }

    }

In our function that checks for match, we are going to do three things. If a player clicks on the same card twice, in our <code>cardsChosenId</code>, we would have the same id twice, so if they match, we send them an alert that they clicked the same image and we reset the image back to the blank image.<br />

Else, if the two names in the chosen cards array match, we send an alert that they have found a match and then we change the images to the white images and remove the event listeners so the player won’t be able to pick it again and lastly we push it to the cards won array. <br />

The last thing to do here is to address if the names of the two chosen cards do not match. If they do not, then we reset the <code>src</code> of the images back to the blank image and then we send an alert that the player should try again.

function checkForMatch() {

        var cards = document.querySelectorAll("img")

        const optionOneId = cardsChosenId[0]

        const optionTwoId = cardsChosenId[1]

        if (optionOneId == optionTwoId) {

            cards[optionOneId].setAttribute("src", "images/blank.png")

            cards[optionTwoId].setAttribute("src", "images/blank.png")

            alert("You have clicked the same image")

        }

        else if (cardsChosen[0] === cardsChosen[1]) {

            alert("You found a match")

            cards[optionOneId].setAttribute("src", "images/white.png")

            cards[optionTwoId].setAttribute("src", "images/white.png")

            cards[optionOneId].removeEventListener("click", flipCard)

            cards[optionTwoId].removeEventListener("click", flipCard)

            cardsWon.push(cardsChosen)

        } else {

            cards[optionOneId].setAttribute("src", "images/blank.png")

            cards[optionTwoId].setAttribute("src", "images/blank.png")

            alert("Sorry, try again")

        }

        cardsChosen=[]

        cardsChosenId=[]

        resultDisplay.textContent = cardsWon.length

        if (cardsWon.length === cardArray.length/2) {

            resultDisplay.textContent = "Congratulations! You found them all!"

        }

    }

After we reset the <code>cardsChosen</code> and the <code>cardsChosenId</code> back to empty arrays. We also change our <code>resultDisplay</code> text content to match the length of the <code>cardsWon</code> array and if the length of our <code>cardsWon</code>array is equal to half the length of our <code>cardArray</code>, we show the user that they have found it all.