```
// Hello.
//
// This is JSHint, a tool that helps to detect errors and potential
// problems in your JavaScript code.
//
// To start, simply enter some JavaScript anywhere on this page. Your
// report will appear on the right side.
//
// Additionally, you can toggle specific options in the Configure
// menu.

function main() {
  return 'Hello, World!';
}

main();
let maximumTime = 60,
    flipped_cards = 0,
    timeCounter = 0,
    counterFunc,
    first_card,
    second_card,
    first_value,
    clicks = 0,
    second_value,
    matched = 0,
    gameIsRunning = false,
    firstRun = true,
    disabled_cards = [],
    moves = document.querySelector(".moves"),
    btn = document.querySelector(".btn"),
    time = document.querySelector(".time"),
    gameover_cont = document.querySelector(".game-over-container"),
    gameover_header = document.querySelector(".header"),
    gameover_details = document.querySelector(".details"),
    gameover_play_again = document.querySelector(".play_again"),
    cards = document.querySelectorAll(".card");


    // refresh the page when the game is over after pressing on "PLAY AGAIN"
function reset(){

   window.location.href = window.location.href;

}

function play(){

   if((!gameIsRunning) && (!firstRun)){

        gameIsRunning = true;
        return resume();

    } else if(!firstRun){

        gameIsRunning = false;
        return pause();

    }

    firstRun = false;
    gameIsRunning = true;
    this.innerHTML = "PAUSE";

  //shuffle the cards
  shuffle();

  // add click event listeners
  addEvent()

  // start the time counter
  count()
}


// shuffling function
function shuffle(){
   cards.forEach((card)=>{card.style.order = Math.floor(Math.random()*10)});
   return;
}


// function which listens for click events on cards
function addEvent(){

    // add click event listeners to the cards and attach the checkMatch function to it
    cards.forEach((card)=>{card.addEventListener("click",checkMatch)});

    // remove click event listener to the already matched cards
    if(disabled_cards.length > 0){
        disabled_cards.forEach((card)=>{
            card.removeEventListener("click",checkMatch);
        })
    }

    return;

}

// lock the cards to wait for the flipped cards to flip back
function lockCards(){
    cards.forEach((card)=>{card.removeEventListener("click",checkMatch)});
    return;
}



// function which checks for match
function checkMatch(){

    clicks++;

    moves.innerHTML ="Moves: <span>"+clicks+"</span>";

    // flip the card
    this.classList.add("flip");

    // show the image that was hidden under the clicked card
    document.querySelector(".card-"+this.dataset.id).classList.add("show");

    if(flipped_cards == 0){
        first_card = this;
        flipped_cards++;
        return;
    }

    // lock the cards and wait until the cards flip back
    lockCards();


    flipped_cards = 0;
    second_card = this;

    // get the value of first clicked card and second clicked card
    first_value = first_card.dataset.win;
    second_value = second_card.dataset.win;

    if(first_value == second_value){

        disabled_cards.push(first_card);
        disabled_cards.push(second_card);

        if(matched == 5) return game_over();
        matched++;
        first_card.removeEventListener("click",checkMatch);
        second_card.removeEventListener("click",checkMatch);

        // unlock the cards after 1 seconds if its a match
        setTimeout(addEvent, 1000);

    } else {

        // flip back the two cards after 1 seconds if its not a match
        setTimeout(() => {

            first_card.classList.remove("flip");
            second_card.classList.remove("flip");

            // hide the images under the flipped cards
            document.querySelector(".card-"+first_card.dataset.id).classList.remove("show"
            document.querySelector(".card-"+second_card.dataset.id).classList.remove("show

            //unlock cards after flipping back the cards
            addEvent();

        }, 1000);

    }

    return;

}

// pause the game when playing
function pause(){

    btn.innerHTML = "RESUME";
    clearInterval(counterFunc);
    cards.forEach((card)=>{card.removeEventListener("click",checkMatch)});
    return;
}


// resume the game if it was paused
function resume(){
    btn.innerHTML = "PAUSE";
    count();
    addEvent();
    return;
}


function game_over(){

    clearInterval(counterFunc);

    gameover_cont.style.left = 0;
    gameover_cont.style.top = 0;

    gameover_header.innerHTML = "GAME OVER !!";

    if(timeCounter < 60){
        gameover_details.innerHTML = timeCounter+" SEC |  "+clicks+" MOVES";
      } else {
        gameover_details.innerHTML = (Math.floor(timeCounter/60))+" MIN, "+(timeCounter%60
    }

    btn.innerHTML = "PLAY";
    cards.forEach((card)=>{card.removeEventListener("click",checkMatch)});

    gameover_play_again.onclick = reset;

    return;
}



function count(){

 counterFunc = setInterval(()=>{

   timeCounter++;

    if(timeCounter < 60){
        time.innerHTML = "Time: <span> "+timeCounter+" sec </span>";
    } else {
        time.innerHTML = "Time: <span> "+(Math.floor(timeCounter/60))+" min, "+(timeCounte
    }

    if((timeCounter == maximumTime)) return game_over();


 },1000);

}


btn.onclick = play;
```

CONFIGURE

Metrics

There are 19 functions in this file.

Function with the largest signature take 1 arguments, while the median is 0.

Largest function has 23 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 4 while the median is 1.

17 warnings

- 17 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
- 70 Missing semicolon.
- 73 Missing semicolon.
- 79 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 79 Missing semicolon.
- 88 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 88 Missing semicolon.
- 92 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 94 Missing semicolon.
- 104 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 104 Missing semicolon.
- 158 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 184 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 184 Missing semicolon.
- 214 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
- 214 Missing semicolon.
- 225 'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').

JSHint
version 2.11.1

About
Documentation
Install
Contribute
Blog