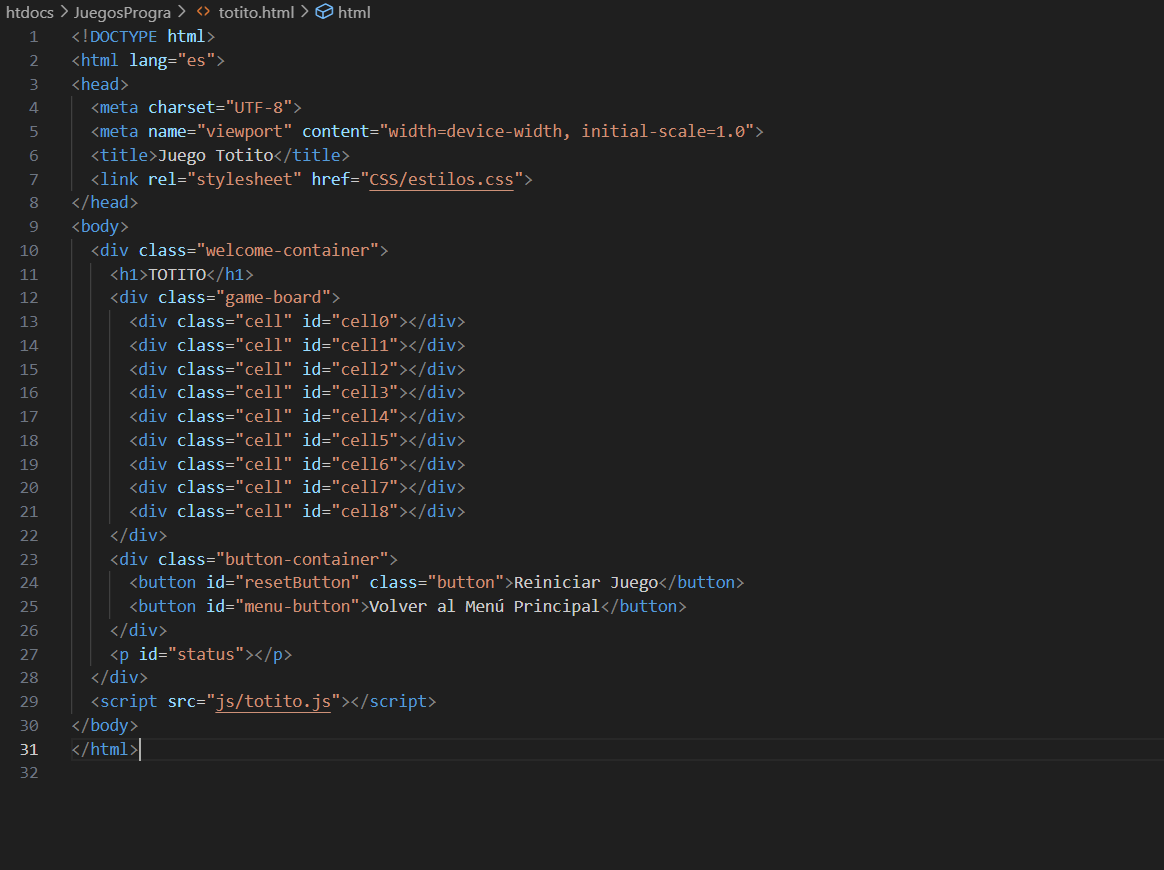
**Proyecto 1 Juego totito y memoria**

Lo primero que se realizó para el totito fue la creación del html con las funciones del juego, como la creación de las casillas y botones



Seguidamente realizamos la creación del documento “js” para poder programar la funcionalidad del juego definiendo constates que seleccionen las celdas del tablero en base a la clase

También definiendo el elemento donde se mostrara el estado del juego ( si hay un ganador o un empate y también definiendo quien juega )

También define quien empieza el juego, en este caso se inicia con “x”

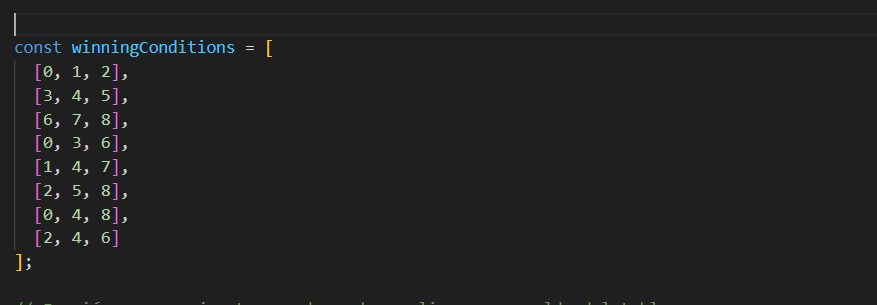
Seguimos declarando un booleano que indica si el juego esta activo, que comienza como verdadero.

El let board nos iguala las cacillas a “ “ vaciándolas para iniciar el juego



// Matriz que define las combinaciones ganadoras posibles (filas, columnas y diagonales).

En esta parte inicializamos la matriz con las combinaciones necesarias para ganar el juego en filas, columnas y diagonales.



// Función que se ejecuta cuando se hace clic en una celda del tablero.

// Obtiene la celda que fue clicada.

 // Extrae el índice de la celda a partir de su id.

// Si la celda ya está ocupada o el juego ha terminado, no hace nada.

// Actualiza el estado del tablero con el jugador actual ('X' o 'O').

// Actualiza el contenido de la celda en la interfaz con el jugador actual.

// Verifica si el juego ha terminado (ganador o empate).

La constante se inicia cuando se hace click en el tablero

La siguiente constante gracias el “e.target” obtiene la celda que fue clicada

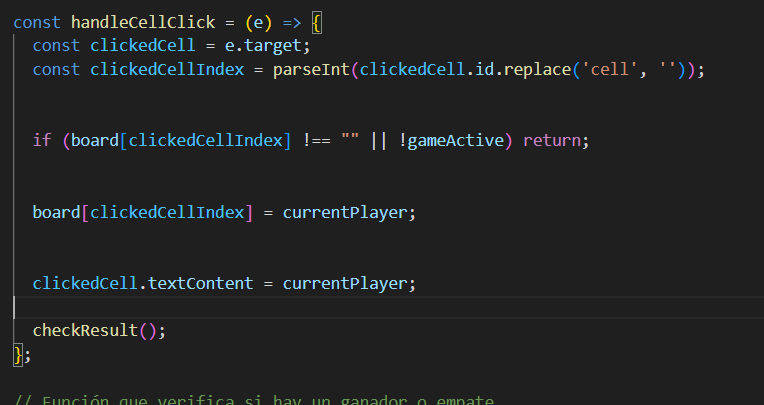
La siguiente constante obtiene la celda a partir de su ID quitando el termino “cell” para solo dejar el numero ejemplo “Cell 01” quedaría como “01” o “Cell 02” quedaría como “02”

Luego el if determina si la celda ya esta ocupada o si el juego ha terminado

El board actualiza el tablero con el jugador actual, ya sea “x” o “0”

Luego actualizamos el juego en la interfaz con el jugador actual

También tenemos la función que evalúa si el juego ha terminado con un ganador o un empate



// Si alguien ganó, muestra el mensaje de victoria y desactiva el juego.

// Muestra el jugador que ganó.

// Desactiva el juego.

 // Verifica si todas las celdas están llenas (empate).

// Si no quedan celdas vacías, es un empate.

 // Si es empate, muestra el mensaje de empate y desactiva el juego.

// Si no hay ganador ni empate, cambia el turno al otro jugador.

// Alterna entre 'X' y 'O'.

// Actualiza el texto de estado para mostrar el próximo jugador.

Primero que todo encontramos la función que no verifica si hay un ganador, gracias al let verificamos si alguien gano o no

Gracias al for recorremos todas la combinaciones ganadoras

Luego con la constante obtenemos una combinación ganadora

Luego pasamos con los let para obtener el primer, el segunto y el tercer valor de la combinación

Con el primer “if” Si alguna celda de esta combinación esta vacia sigue a la siguiente

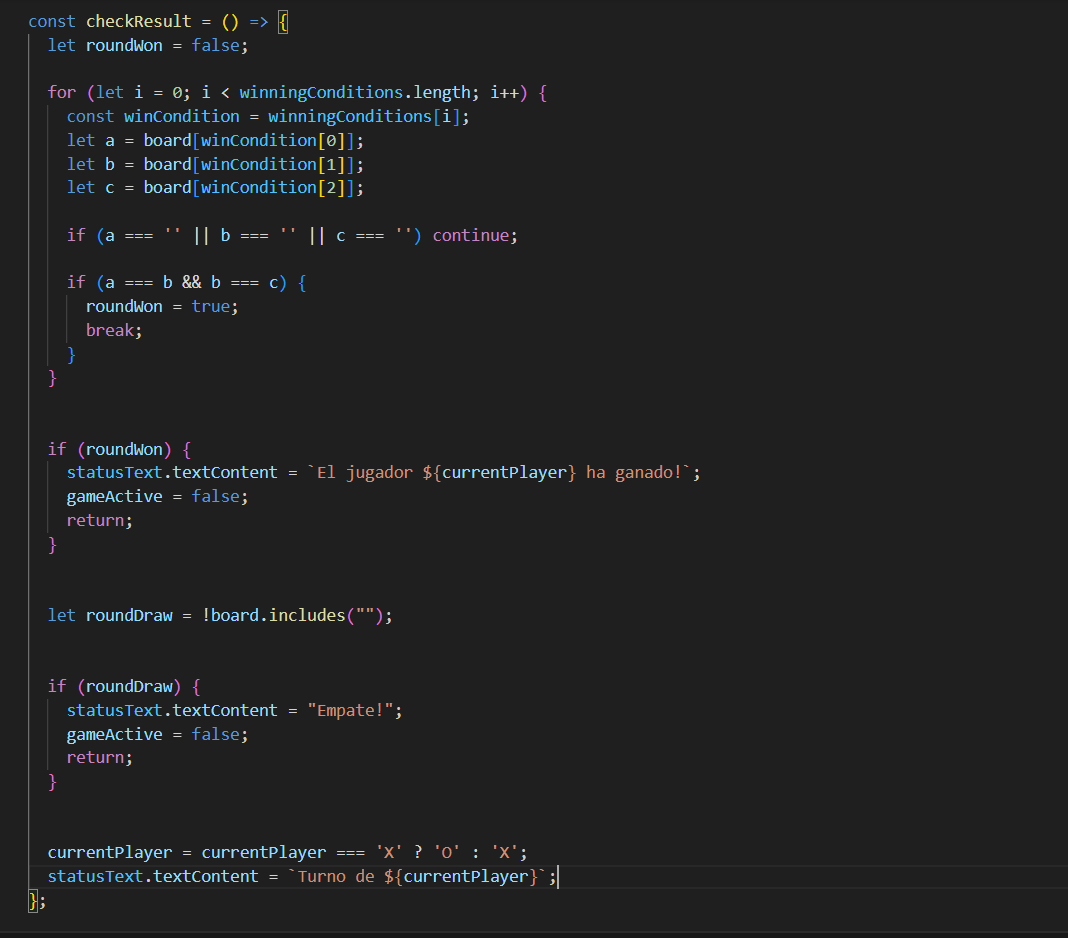
Con el segundo “if” vemos si todos los valores de la combinación sin iguales entonces hay un ganador y se establece la victoria y si no se rompe el bucle

El tercer “if” en caso de que haya ganado un jugador se muestra el mensaje en pantalla y se desactiva el juego cambiando el booleano a falso

Verídica si todas las casillas están llenas (empate) en caso no queden celdas vacías, se declara empate

Si es empate el cuarto “if” muestra el mensaje de empate y desactiva el juego

Después se evalua si no hay un ganador y tampoco un empate cambia el turno de jugador alternando entre “X” o “0” y actualiza el texto del juego



Con la constante reiniciamos el juego

Luego el tablero se resetea aunque este vacio

Luego iniciamos el juego e inicializamos el jugador “x” y mostramos el texto con el turno del jugador y actualizamos cada celda para vaciarlas todas

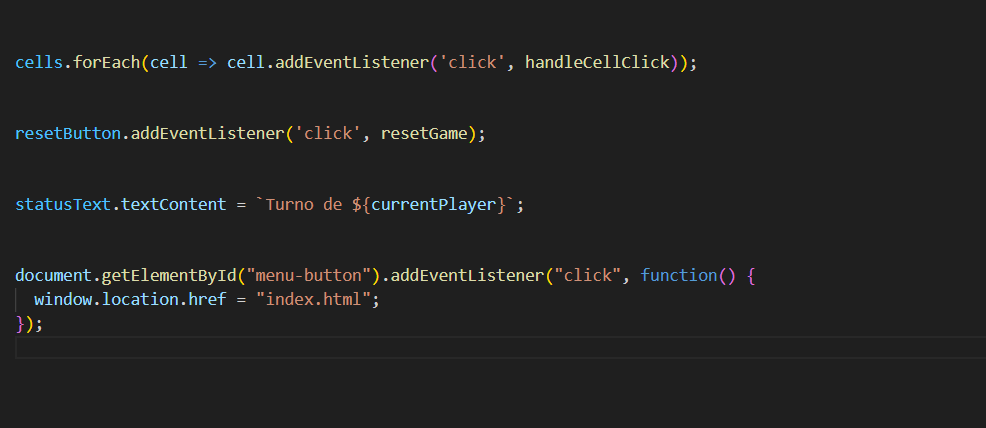


En esta parte del código añadimos un Listener a cada celda para que se detecte cuando se hace click en una celda

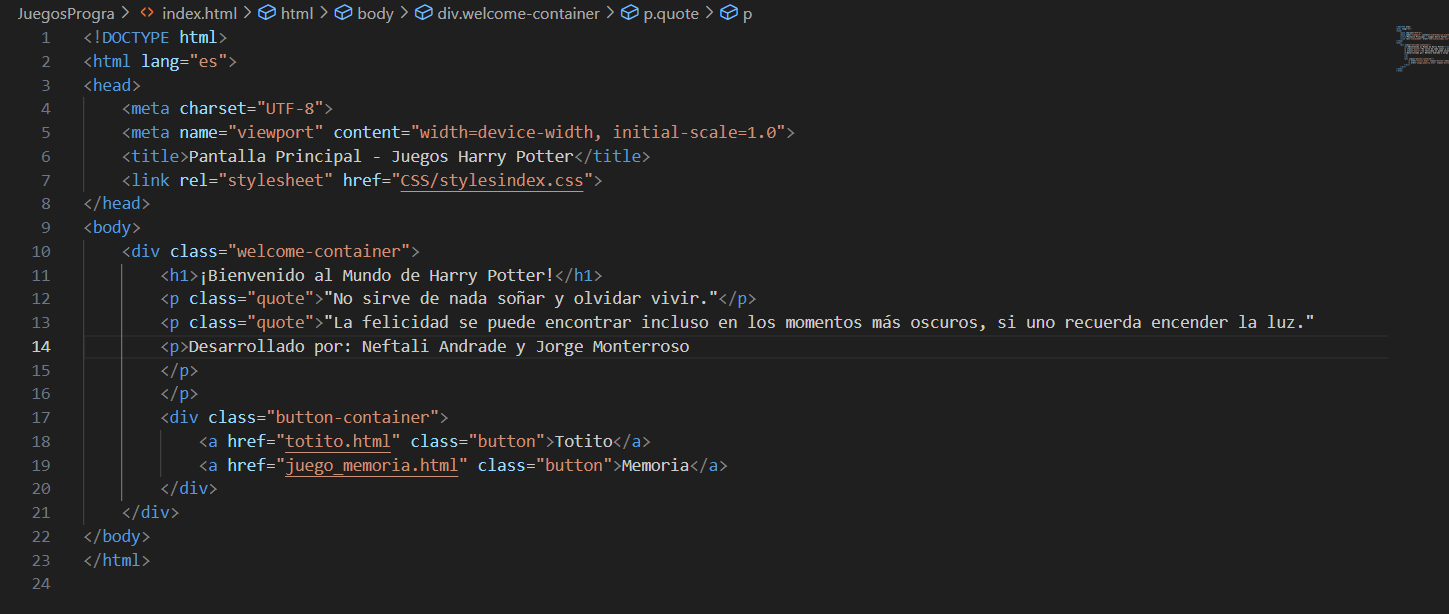
Luego hacemos lo mismo, pero para el botón de reiniciar juego

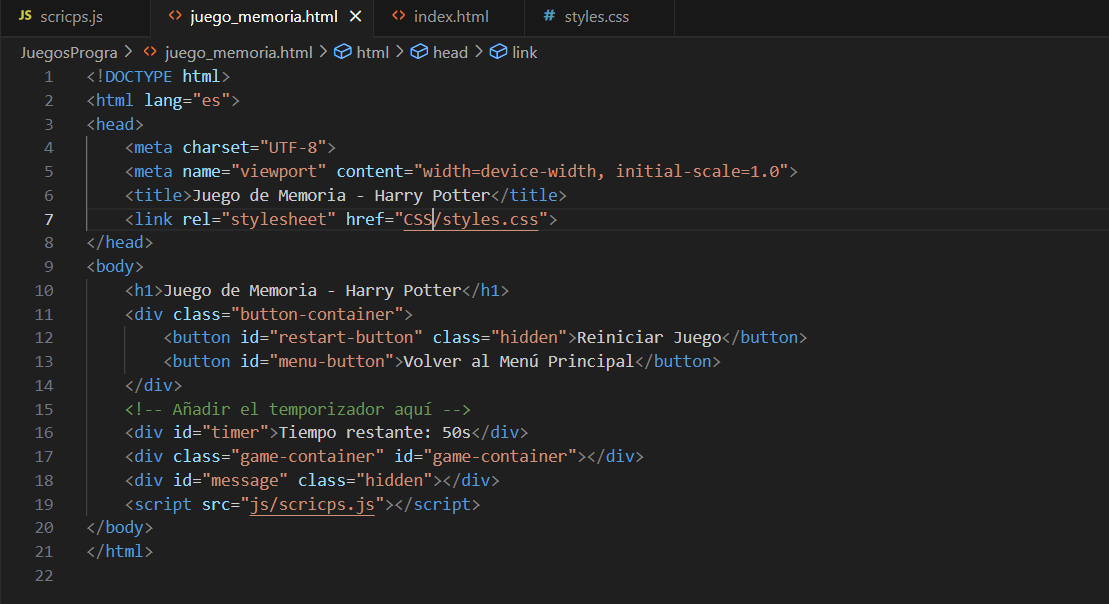
Luego establecemos texto del jugador inicial para que muestre el turno de “x”

Luego añadimos otro listener para que botón que redirige al menú en la pagina principal cambiando al index



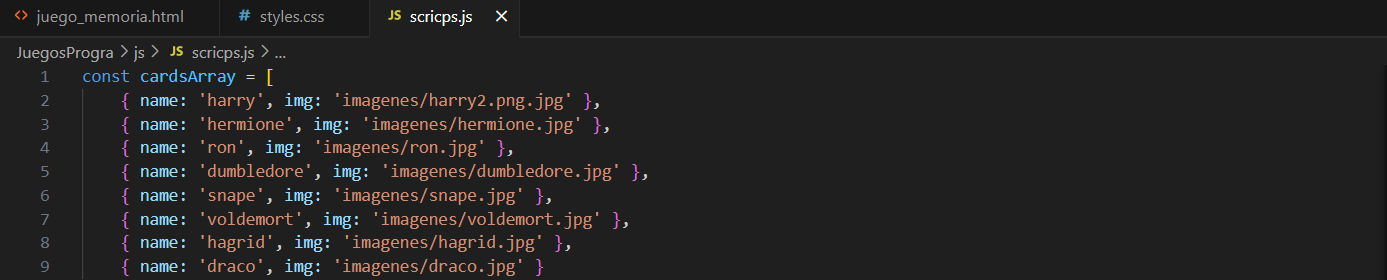
Lo primero que se realizó para el juego de memoria y el index fue la creación del html con las funciones del juego, como la creación de las casillas y botones



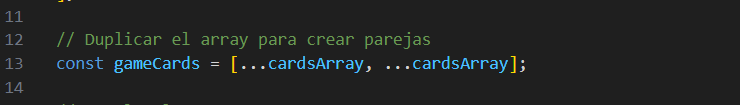


Seguidamente la primera parte del código consiste en crear un **array** (arreglo) que contiene los datos de las cartas que se mostrarán en el juego. Cada objeto en el array cardsArray tiene dos propiedades:

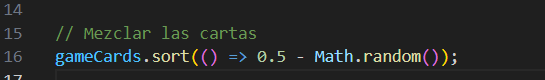
* name: El nombre del personaje que representa la carta.
* img: La ruta de la imagen correspondiente al personaje.



Después, el array cardsArray se duplica para crear las parejas de cartas. El operador de propagación (...) se usa para duplicar las cartas en un nuevo array llamado gameCards.



Una vez que tenemos las parejas de cartas, se mezclan al azar para que el jugador no sepa dónde está cada carta. Se utiliza el método sort() con una función que devuelve un valor aleatorio.



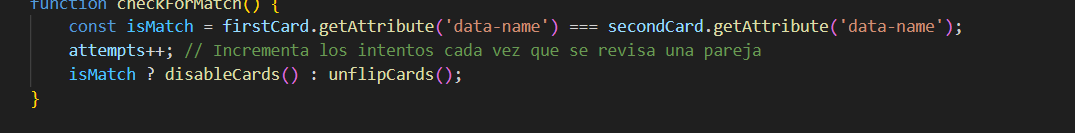
La función createCards() se encarga de generar las cartas y mostrarlas en el contenedor del juego. Se recorre el array gameCards, creando para cada carta un elemento div que contendrá la imagen correspondiente, y luego se agrega al contenedor de cartas (gameContainer).



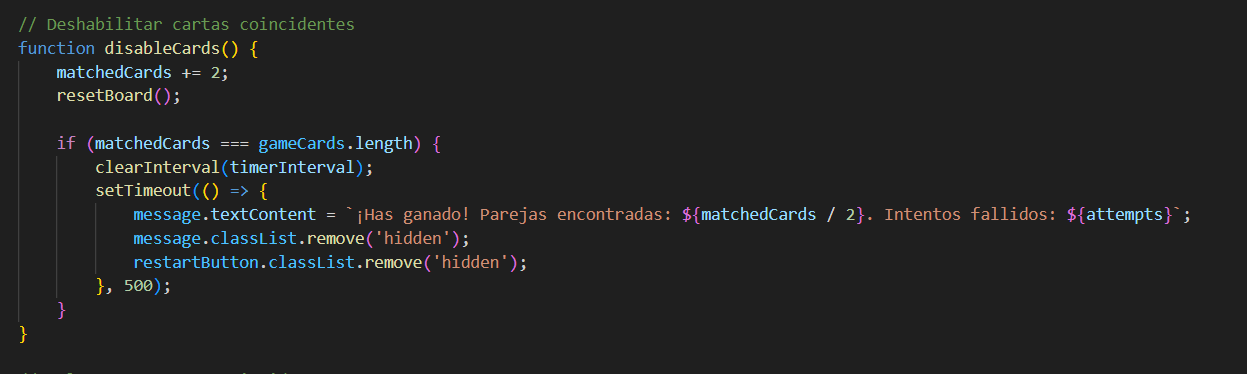
La función flipCard() es activada cada vez que el jugador hace clic en una carta. Esta función revisa si ya hay una carta volteada y, si es así, se voltea la segunda carta para comprobar si ambas coinciden.



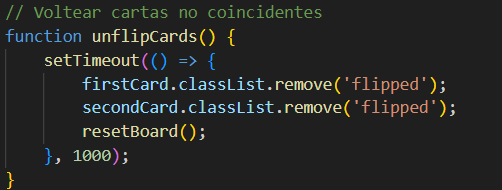
Después de que se volteen dos cartas, la función checkForMatch() comprueba si los nombres de las dos cartas seleccionadas coinciden. Si es así, se desactivan esas cartas. Si no, se vuelven a voltear.



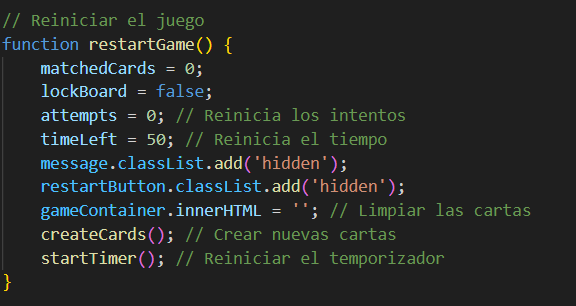
Si las dos cartas coinciden, la función disableCards() deshabilita esas cartas para que no puedan ser volteadas nuevamente.



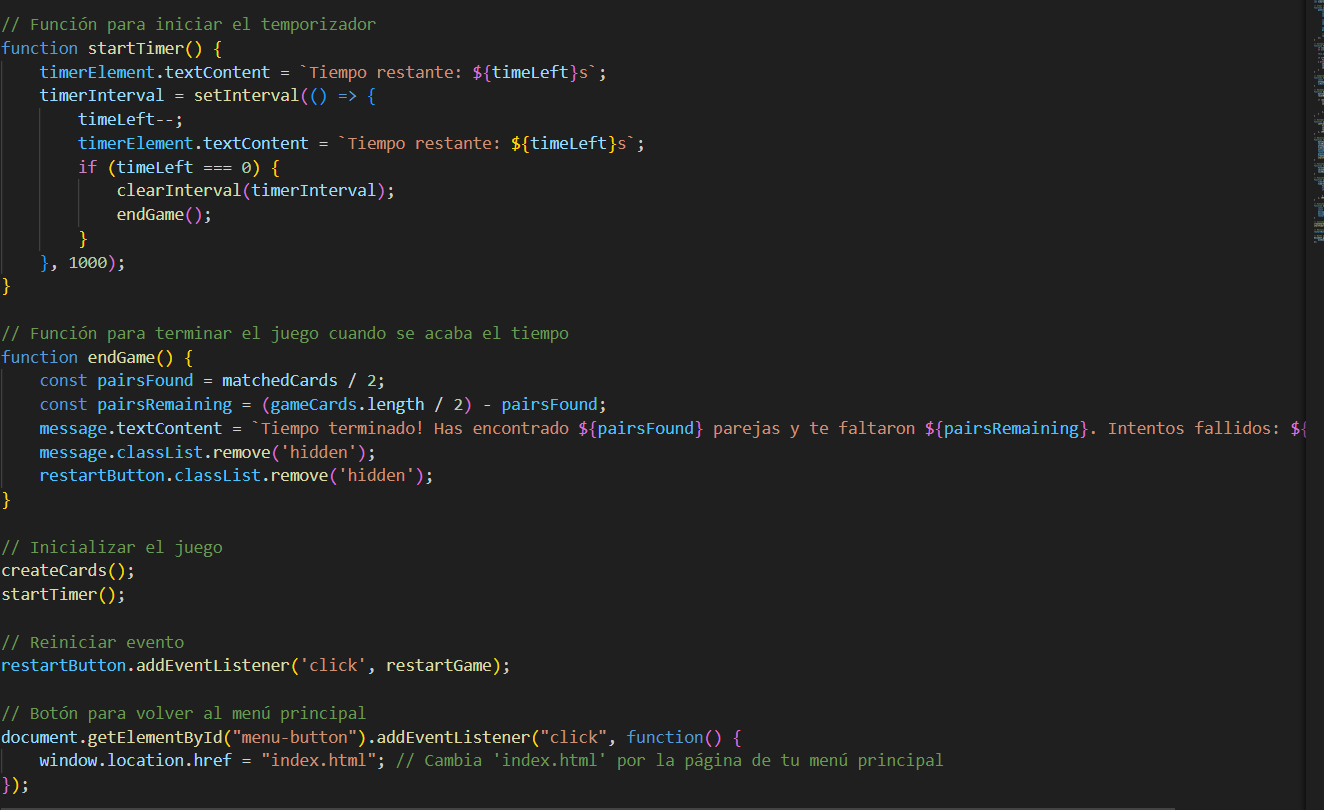
Si las cartas no coinciden, la función unflipCards() vuelve a ocultar las imágenes de las cartas tras un pequeño retraso.



Si el jugador desea reiniciar el juego, la función restartGame() resetea el tablero, las cartas, los intentos y el temporizador.



El temporizador se implementa para que el jugador tenga un límite de tiempo para encontrar todas las parejas. La función startTimer() actualiza el tiempo restante en pantalla y finaliza el juego si el tiempo se acaba.



***Uso de inteligencia articial***

La inteligencia artificial fue una gran herramienta que nos ayudo a la hora de resolución de dudas y recomendaciones para el código, por ejemplo: en la realización del juego de totito fue de gran ayuda a la hora de la realización del “for” que buscaba un posible ganador ya que este mis mo no funcionaba porque no existía el if que verificara que las variables sean iguales y tengan el mismo signo para dar la victoria o no y en si ala inteligencia artificial fue de gran ayuda en otras partes del código dando recomendaciones y proporcionando ideas.