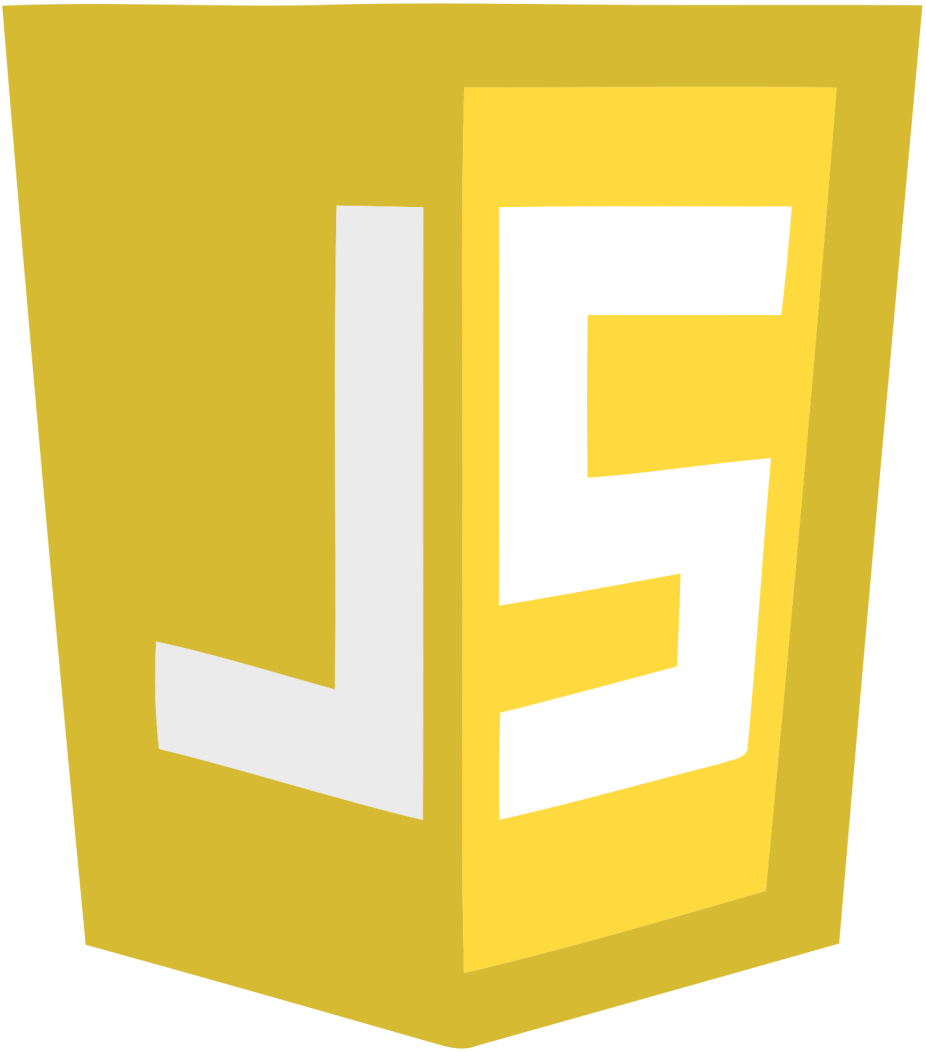
JavaScript



Trabajo del solitario

Víctor Heredia

Sergio Gomez

Índice

1.Introducción

Para este trabajo hemos utilizado Javascript, Html, CSS, además de eso hemos utilizado las librerías proporcionadas de Jquery y Bootstrap y hemos trabajado todo desde Visual Studio Code

2.Desarrollo

```
                de elementos span -->
            <div id="mesa">
                <div class="tapetes_superiores">
                    <div id="inicial" class="tapete"><span id="contador_inicial" class="contador"></span></div><!--
                    --><div id="sobrantes" ondrop="drop(event)" ondragover="allowDrop(event)" class="tapete"><span id="contador_sobrantes" class="contador"></span></div>
                </div>
                <div id="receptor1" ondrop="drop(event)" ondragover="allowDrop(event)" class="tapete receptor"><span id="contador_receptor1" class="contador"></span></div><!--
                --><div id="receptor2" ondrop="drop(event)" ondragover="allowDrop(event)" class="tapete receptor"><span id="contador_receptor2" class="contador"></span></div><!--
                --><div id="receptor3" ondrop="drop(event)" ondragover="allowDrop(event)" class="tapete receptor"><span id="contador_receptor3" class="contador"></span></div><!--
                --><div id="receptor4" ondrop="drop(event)" ondragover="allowDrop(event)" class="tapete receptor"><span id="contador_receptor4" class="contador"></span></div>
            </div>

            <!-- Botón de reset que reinicia todo: partida, contadores de tiempo, de movimientos, ... -->
            <button id="reset" onclick="comenzar_juego()" class="btn btn-danger">Reiniciar</button>
        </div> <!-- class="central" -->
```

Hemos añadido los atributos ondrop y ondragover para implementar la funcionalidad drag and drop sobre los receptores y el tapete de sobrantes, en el mazo inicial o tapete inicial esto está hecho directamente sobre el JS.
Respecto a la declaración de variables en el JS no voy a comentar nada ya que hemos usado las que nos dio usted con su respectiva explicación en el documento.

```
// Desarrollo del comienzo del juego
function comenzar_juego() {
    cont_inicial = document.getElementById("contador_inicial");
    cont_sobrantes = document.getElementById("contador_sobrantes");
    cont_receptor1 = document.getElementById("contador_receptor1");
    cont_receptor2 = document.getElementById("contador_receptor2");
    cont_receptor3 = document.getElementById("contador_receptor3");
    cont_receptor4 = document.getElementById("contador_receptor4");
    cont_movimientos = document.getElementById("contador_movimientos");
    /* Crear baraja, es decir crear el mazo_inicial. Este será un array cuyos
    elementos serán elementos HTML <img>, siendo cada uno de ellos una carta.
    Sugerencia: en dos bucles "for", bárranse los "palos" y los "numeros", formando
    oportunamente el nombre del fichero "png" que contiene a la carta (recuérdese poner
    el "path" correcto en la URL asociada al atributo "src" de <img>). Una vez creado
    el elemento <img>, inclúyase como elemento del array "mazo_inicial".
    */

    /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! **/
    mazo_inicial = [
        './imagenes/baraja/1-cir.png', './imagenes/baraja/1-cua.png', './imagenes/baraja/1-hex.png', './imagenes/baraja/1-ova.png',
        './imagenes/baraja/2-cir.png', './imagenes/baraja/2-cua.png', './imagenes/baraja/2-hex.png', './imagenes/baraja/2-ova.png',
        './imagenes/baraja/3-cir.png', './imagenes/baraja/3-cua.png', './imagenes/baraja/3-hex.png', './imagenes/baraja/3-ova.png',
        './imagenes/baraja/4-cir.png', './imagenes/baraja/4-cua.png', './imagenes/baraja/4-hex.png', './imagenes/baraja/4-ova.png',
        './imagenes/baraja/5-cir.png', './imagenes/baraja/5-cua.png', './imagenes/baraja/5-hex.png', './imagenes/baraja/5-ova.png',
        './imagenes/baraja/6-cir.png', './imagenes/baraja/6-cua.png', './imagenes/baraja/6-hex.png', './imagenes/baraja/6-ova.png',
        './imagenes/baraja/7-cir.png', './imagenes/baraja/7-cua.png', './imagenes/baraja/7-hex.png', './imagenes/baraja/7-ova.png',
        './imagenes/baraja/8-cir.png', './imagenes/baraja/8-cua.png', './imagenes/baraja/8-hex.png', './imagenes/baraja/8-ova.png',
        './imagenes/baraja/9-cir.png', './imagenes/baraja/9-cua.png', './imagenes/baraja/9-hex.png', './imagenes/baraja/9-ova.png',
        './imagenes/baraja/10-cir.png', './imagenes/baraja/10-cua.png', './imagenes/baraja/10-hex.png', './imagenes/baraja/10-ova.png',
        './imagenes/baraja/11-cir.png', './imagenes/baraja/11-cua.png', './imagenes/baraja/11-hex.png', './imagenes/baraja/11-ova.png',
        './imagenes/baraja/12-cir.png', './imagenes/baraja/12-cua.png', './imagenes/baraja/12-hex.png', './imagenes/baraja/12-ova.png'];
    // Barajar
    barajar(mazo_inicial);
    // Dejar mazo_inicial en tapete inicial
    cargar_tapete_inicial(mazo_inicial);

    // Puesta a cero de contadores de mazos
    set_contador(cont_sobrantes, 0);
    set_contador(cont_receptor1, 0);
    set_contador(cont_receptor2, 0);
    set_contador(cont_receptor3, 0);
    set_contador(cont_receptor4, 0);
    set_contador(cont_movimientos, 0);
    set_contador(cont_inicial, mazo_inicial.length);

    // Arrancar el conteo de tiempo
    arrancar_tiempo();
} // comenzar_juego
```

1. La función obtiene referencias a varios elementos HTML en la página, incluyendo los contadores para el mazo inicial, el mazo restante y las cuatro pilas receptoras, así como el contador para el número de movimientos realizados.
2. La función crea un array llamado "mazo_inicial" que contiene las rutas de archivo para las imágenes de las cartas en el mazo. Crea este array usando dos bucles for para generar las rutas de archivo para cada carta y agregarlas al array.
3. La función llama a una función "barajar" para mezclar las cartas en el array "mazo_inicial".
4. La función llama a una función "cargar_tapete_inicial" para mostrar las cartas del mazo inicial en el tablero de juego.
5. La función establece los valores de los contadores para el mazo restante y las cuatro pilas receptoras en cero, y también establece el valor del contador para el número de movimientos realizados en cero.
6. La función inicia el temporizador del juego.

```
*/
function barajar(mazo) {
    /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! **/
    for (let i = 0; i <= mazo.length - 1; i++) {
        j = Math.floor(Math.random() * mazo.length);
        [mazo[i], mazo[j]] = [
            mazo[j], mazo[i]];
    };
} // barajar
```

La función utiliza un bucle "for" para recorrer todas las cartas del mazo. En cada iteración, se genera un número aleatorio "j" entre 0 y la longitud del mazo. Luego, se intercambia la carta en la posición actual del bucle con la carta en la posición "j".

```
function cargar_tapete_inicial(mazo) {
    /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! **/
    paso = 0
    let carta;
    for (let i = 0; i <= mazo.length - 1; i++) {
        carta = document.createElement("img");
        carta.setAttribute("id", i);
        carta.setAttribute("src", mazo[i]);
        carta.setAttribute("width", "50");
        carta.setAttribute("height", "50");
        carta.setAttribute("style", "top:" + paso + "px; left:" + paso + "px;");
        carta.setAttribute("draggable", "false");
        carta.setAttribute("ondragstart", "drag(event)");
        tapete_inicial = document.getElementById("inicial");
        tapete_inicial.appendChild(carta);
        paso = paso + 11
    }

    carta.draggable = true;


} // cargar_tapete_inicial
```

- La función recibe como argumento el mazo, que es un array de strings que representan las rutas de las imágenes de las cartas.
- Dentro de la función se inicializa la variable "paso" a 0, que se utiliza para calcular la posición en píxeles de cada carta en el eje vertical.
- A continuación, se inicia un bucle "for" que recorre el mazo de cartas con una variable "i".
- En cada iteración del bucle, se crea un elemento "img" con el método "document.createElement".
- Se establecen varios atributos en el elemento creado con el método "setAttribute". Estos atributos son: el id de la carta, la ruta de la imagen de la carta, el ancho y el alto de la imagen, la posición en píxeles de la carta en el eje vertical y horizontal, el atributo "draggable" y la función "ondragstart".
- Finalmente, se inserta la carta creada en el elemento HTML con id "inicial" utilizando el método "appendChild".
- La variable "paso" se incrementa en cada iteración del bucle para que las cartas se muestren desplazadas verticalmente.
- Al final, se establece que la variable "carta" es arrastrable ("carta.draggable = true;").

```
/**
    Esta función debe incrementar el número correspondiente al contenido textual
        del elemento que actúa de contador
*/
function inc_contador(contador) {
    contador.innerHTML = +contador.innerHTML + 1;
} // inc_contador

/**
    Idem que anterior, pero decrementando
*/
function dec_contador(contador) {
    /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! ***/
    contador.innerHTML = +contador.innerHTML - 1;
} // dec_contador

/**
    Similar a las anteriores, pero ajustando la cuenta al valor especificado
*/
function set_contador(contador, valor) {
    //contador  = document.getElementById("cont_tiempo");
    contador.innerHTML = valor;
} // set_contador
```

Esto no tiene mucha complicación simplemente es ir añadiendo 1 cada vez que se mueva una carta de sitio y el set es para enseñar ese valor.

Esta función allowDrop define el comportamiento de la zona de soltar cuando se arrastra una carta y se sitúa encima de ella. Si el objetivo es uno de los elementos receptor1, receptor2, receptor3 o receptor4, o si el objetivo es el elemento sobrantes pero la procedencia de la carta no es sobrantes, entonces se permite soltar la carta en esa zona.
La función drag se ejecuta cuando se empieza a arrastrar una carta. La información de la carta arrastrada se guarda en el evento dataTransfer para que pueda ser recuperada posteriormente. Además, se guarda el ID del elemento que emite el evento para poder ser utilizado en la función allowDrop.

La función drop es muy larga así que vamos paso a paso

```
function drop(event) {
    let procedencia = document.getElementById(emisor).parentElement.id;
    let data = event.dataTransfer.getData("text");
    let objetivo = event.target.id;
    cont_movimientos = document.getElementById("contador_movimientos");
    event.preventDefault();

    let carta_nombre = document.getElementById(data).src.substr(-10, 5);
    carta_nombre = carta_nombre.replace("/", "");
    let numero_simbolo = carta_nombre.split("-")
    let color_emisor;
    let color_receptor;
    let numero_siguiente;
    let mover = false;
    if (numero_simbolo[1] == "ci" || numero_simbolo[1] == "he") {
        color_emisor = "gris"
    } else {
        color_emisor = "naranja"
    }
    let numero = numero_simbolo[0];
```

Esta parte es la encargada de almacenar en variables todos los datos necesarios a la hora de hacer drop en una carta y comprobar que tipo de carta es.

```
switch (objetivo) {
    case "receptor1":

        carta_receptor = mazo_receptor1[mazo_receptor1.length - 1];

        if (carta_receptor) {
            numero_simbolo = mazo_receptor1[mazo_receptor1.length - 1].split("-")
            if (numero_simbolo[1] == "ci" || numero_simbolo[1] == "he") {
                color_receptor = "gris"
            } else {
                color_receptor = "naranja"
            }
        }

        numero_siguiente = numero;
        numero_siguiente++;
        if (mazo_receptor1.length == 0 && numero == 12 || mazo_receptor1.length != 0 && color_emisor
            != color_receptor && numero_siguiente == numero_simbolo[0]) {
            mazo_receptor1.push(carta_nombre);
            inc_contador(cont_receptor1)
            document.getElementById(data).draggable = false;
            inc_contador(cont_movimientos)
            event.target.appendChild(document.getElementById(data));
            mover = true;

        }

        break;
```

```
        case "receptor2":

            carta_receptor = mazo_receptor2[mazo_receptor2.length - 1];

            if (carta_receptor) {
                numero_simbolo = mazo_receptor2[mazo_receptor2.length - 1].split("-")
                if (numero_simbolo[1] == "ci" || numero_simbolo[1] == "he") {
                    color_receptor = "gris"
                } else {
                    color_receptor = "naranja"
                }
            }
            numero_siguiente = numero;
            numero_siguiente++;
            if (mazo_receptor2.length == 0 && numero == 12 || mazo_receptor2.length != 0 && color_emisor
                != color_receptor && numero_siguiente == numero_simbolo[0]) {
                mazo_receptor2.push(carta_nombre);
                inc_contador(cont_receptor2)
                document.getElementById(data).draggable = false;
                inc_contador(cont_movimientos)
                event.target.appendChild(document.getElementById(data));
                mover = true;
            }
            break;
```

```
        case "receptor3":

            carta_receptor = mazo_receptor3[mazo_receptor3.length - 1];

            if (carta_receptor) {
                numero_simbolo = mazo_receptor3[mazo_receptor3.length - 1].split("-")
                if (numero_simbolo[1] == "ci" || numero_simbolo[1] == "he") {
                    color_receptor = "gris"
                } else {
                    color_receptor = "naranja"
                }
            }
            numero_siguiente = numero;
            numero_siguiente++;
            if (mazo_receptor3.length == 0 && numero == 12 || mazo_receptor3.length != 0 && color_emisor
                != color_receptor && numero_siguiente == numero_simbolo[0]) {
                mazo_receptor3.push(carta_nombre);
                inc_contador(cont_receptor3)
                document.getElementById(data).draggable = false;
                inc_contador(cont_movimientos)
                event.target.appendChild(document.getElementById(data));
                mover = true;
            }
            break;

        case "receptor4":

            carta_receptor = mazo_receptor4[mazo_receptor4.length - 1];

            if (carta_receptor) {
                numero_simbolo = mazo_receptor4[mazo_receptor4.length - 1].split("-")
                if (numero_simbolo[1] == "ci" || numero_simbolo[1] == "he") {
                    color_receptor = "gris"
                } else {
                    color_receptor = "naranja"
                }
            }
            numero_siguiente = numero;
            numero_siguiente++;
            if (mazo_receptor4.length == 0 && numero == 12 || mazo_receptor4.length != 0 && color_emisor
                != color_receptor && numero_siguiente == numero_simbolo[0]) {
                mazo_receptor4.push(carta_nombre);
                inc_contador(cont_receptor4)
                document.getElementById(data).draggable = false;
                inc_contador(cont_movimientos)
                event.target.appendChild(document.getElementById(data));
                mover = true;
            }
            break;
```

```
            }
            break;

        case "sobrantes":

            mazo_sobrantes.push(data);
            inc_contador(cont_sobrantes)
            inc_contador(cont_movimientos)
            event.target.appendChild(document.getElementById(data));
            mazo_sobrantes.push(carta_nombre);
            mazo_aux.push(mazo_inicial[mazo_inicial.length - 1]);
            mover = true;
            break;

    }
```

Aquí básicamente lo que estamos haciendo es aumentar los contadores y asignar las cartas enviadas a los distintos tapetes, dentro hacemos comprobaciones para que no se haga ningún movimiento ilegal.

```
if (procedencia == "sobrantes" && mover) {
    mazo_sobrantes.pop();
    mazo_aux.pop();
    dec_contador(cont_sobrantes)
}

if (objetivo != "sobrantes" && mover) {
    document.getElementById(data).draggable = false;

}
if (procedencia == "inicial" && mover) {
    mazo_inicial.pop();
    document.getElementById(procedencia).lastElementChild.draggable = true;
    dec_contador(cont_inicial)
    if (mazo_inicial.length == 0 && mazo_sobrantes.length != 0) {
        set_contador(cont_sobrantes, 0);
        mazo_inicial = mazo_aux;
        mazo_sobrantes = [];
        mazo_aux = [];
        set_contador(cont_inicial, mazo_inicial.length);
        barajar(mazo_inicial);
        cargar_tapete_inicial(mazo_inicial);
        var e = document.getElementById("sobrantes");

        var child = e.lastElementChild;
        while (child) {
            child = e.lastElementChild;
            if (child.id != "contador_sobrantes" && child) {
                e.removeChild(child);
            } else if (child.id == "contador_sobrantes") {
                child = null;
            }

        }
    }
}
```

En estas comprobaciones solo se encargan de hacer “controles” el primer if comprueba que se haya movido alguna carta es una especie de rollback, el segundo es para que no se pueda sacar una carta de los 4 tapetes, y el tercero se encarga de reiniciar el mazo inicial barajando el mazo sobrantes.

```
if (mazo_inicial.length == 0 && mazo_sobrantes.length == 0) {
    alert("Ha finalizado la partida, has realizado " + cont_movimientos.innerHTML + " durante " + document.getElementById("cont_tiempo").innerHTML);
    var marcadores = document.getElementById("marcadores");
    var carta= document.createElement("video");
    carta.autoplay="true"
    carta.loop="true"
    carta.setAttribute("src", "./imagenes/Mono.mp4");
    tapete_inicial = document.getElementById("inicial");
    tapete_inicial.appendChild(carta);
    new Audio('./imagenes/Audio.mp3').play()
    if (mazo_inicial.length === 0 && mazo_sobrantes.length === 0) {
      marcadores.style.display = "none";
    } else {
      marcadores.style.display = "block";
```

Este último if es para hacer la comprobación de que el juego ha terminado y proporcionar feedback al usuario

3.Conclusiones

El código en si no era un gran problema ya que es un juego con un mecanismo bastante sencillo en realidad es mucho más sencillo que un solitario tradicional, lo que hemos tenido grandes problemas en entender el enunciado ya que no es para nada claro e incluso está bastante mal redactado y es pesado, las fuentes proporcionadas tampoco han servido de mucha ayuda y hemos tenido que recurrir a fuentes externas de google para poder realizar el trabajo ya que dejaban muchas cosas de por medio que no se explicaban en los documentos.

Hemos tenido problemas a la hora de avanzar ya que a medida que avanzábamos surgían nuevos problemas de funcionamiento, por lo tanto también en ese sentido ha sido un poco frustrante, aburrido y tedioso.
Lo bueno es que al final lo hemos conseguido, porque el código lo hemos optimizado al 99% y además después de probarlo 20 veces ha funcionado todo perfectamente.

4.Recomendaciones

La redacción del trabajo es mejorable, en el código hay muchas variables que su significado no es demasiado claro y al haber tantas variables hace que sea muy complicado saber para qué sirve todo, por lo que otra recomendación es dar nombres más específicos a las variables, más simples.

5.Bibliografía

carloseduardoreinacardenas (2015). Como implementar Drag and Drop con HTML5
HTML Drag and Drop API
https://platzi.com/blog/drag-and-drop-html5/
w3 schools HTML Drag and Drop API
https://www.w3schools.com/html/html5_draganddrop.asp
Rachel Andrew(31 agosto 2021) Como se usa la API de drag and drop
https://web.dev/drag-and-drop/
Curso de JAVASCRIPT desde Cero. (20 de octubre de 2020
YouTube.https://www.youtube.com/watch?v=z95mZVUcJ-E&ab_channel=SoyDalto
Algunos documentos ofrecidos en la página de moodle + dados en el proyecto