CPIFP Los Enlaces

Febrero 2023

Proyecto: Juego del Solitario

Desarrollo Web en Entorno Cliente

Aaron Bresser, Jorge de Mingo, Alejandro Sanchez

ÍNDICE

1. Introducción

Nuestro objetivo era desarrollar el juego del solitario utilizando Javascript, también hemos utilizado css para darle estilos a nuestra página del juego.

Con respecto al IDE o entorno de desarrollo integrado hemos optado por usar Visual Studio Code, ya que es bastante sencillo de utilizar y podemos instalar únicamente las herramientas de desarrollo requeridas y personalizarlo de acuerdo a nuestras necesidades.

Las reglas del juego son de sobra conocidas por todos, pero, por si quedase alguna duda las pasamos a explicar aquí:

- La baraja de cartas consiste en:
    - **12 números:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 y 12.
    - **4 palos:** óvalos, cuadrados, hexágonos y círculos.

Existirán 6 tapetes donde se almacenarán los mazos de cartas; el tapete inicial, el tapete de sobrantes y cuatro tapetes receptores.

El juego consiste en coger la carta de arriba del tapete inicial o bien del tapete de sobrantes, e intentar introducirla en alguno de los mazos de los cuatro tapetes receptores.

El funcionamiento de los tapetes es el siguiente:

- Tapete inicial: Al comienzo, el mazo completo se ha mezclado de forma aleatoria y las cartas se tomarán una por una desde la parte superior. El propósito es tratar de colocar la carta que se haya tomado en uno de los tapetes receptores siempre y cuando se satisfagan determinadas condiciones.

- Tapete de sobrantes: Se utiliza para colocar temporalmente las cartas que no pueden ser colocadas en ninguno de los tapetes receptores. La última carta depositada en este tapete se convierte en una fuente alternativa de cartas para el mazo del tapete inicial, y se puede intentar colocarla en uno de los tapetes receptores en cualquier momento, siempre y cuando se cumplan las reglas correspondientes, al igual que si la carta hubiera venido del mazo del tapete inicial.

- Tapetes receptores: En este lugar, se colocarán las cartas en orden descendente (empezando por el 12 como la primera carta y terminando con la última) en los mazos correspondientes, alternando los colores (naranja y gris) para cada carta que se coloque en cada mazo.

Cuando se hayan agotado todas las cartas del mazo del tapete inicial, las cartas restantes en el tapete de sobrantes serán automáticamente barajadas y devueltas al tapete inicial para empezar de nuevo con esas cartas.

El juego termina cuando no quedan cartas en ninguno de los dos tapetes. En ese momento, se mostrará una ventana de aviso que informará sobre el tiempo de juego y la cantidad de movimientos realizados durante la partida.

2. Desarrollo

2.1 Declaración de variables

Empezamos a desarrollar nuestro archivo de javascript declarando las variables globales.

```
/***** INICIO DECLARACIÓN DE VARIABLES GLOBALES *****/

// Array de palos
let palos = ["ova", "cua", "hex", "cir"];
// Array de número de cartas
//let numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]; modo largo
// En las pruebas iniciales solo se trabajará con cuatro cartas por palo:

let numeros = [9, 10, 11, 12];

// paso (top y left) en pixeles de una carta a la siguiente en un mazo
let paso = 5;

// Tapetes
let tapete_inicial = document.getElementById("inicial");
let tapete_sobrantes = document.getElementById("sobrantes");
let tapete_receptor1 = document.getElementById("receptor1");
let tapete_receptor2 = document.getElementById("receptor2");
let tapete_receptor3 = document.getElementById("receptor3");
let tapete_receptor4 = document.getElementById("receptor4");

// Mazos
let mazo_inicial = [];
let mazo_sobrantes = [];
let mazo_receptor1 = [];
let mazo_receptor2 = [];
let mazo_receptor3 = [];
let mazo_receptor4 = [];

// Contadores de cartas
let cont_inicial = document.getElementById("cont_inicial");
let cont_sobrantes = document.getElementById("cont_sobrantes");
let cont_receptor1 = document.getElementById("cont_receptor1");
let cont_receptor2 = document.getElementById("cont_receptor2");
let cont_receptor3 = document.getElementById("cont_receptor3");
let cont_receptor4 = document.getElementById("cont_receptor4");
let cont_movimientos = document.getElementById("cont_movimientos");

// Tiempo
let cont_tiempo = document.getElementById("cont_tiempo"); // span cuenta tiempo
let segundos = 0;    // cuenta de segundos
let temporizador = null; // manejador del temporizador

let tap = true; // variable para saber tapete de origen

/***** FIN DECLARACIÓN DE VARIABLES GLOBALES *****/
```

Después de eso asociamos el botón de reiniciar al comienzo de juego, para que si se pulsa se reinicie el juego completo.

```
// Rutina asociada a boton reset: comenzar_juego
document.getElementById("reset").onclick = comenzar_juego;
```

2.2 Función comenzar juego

Lo primero que hacemos es vaciar el mazo inicial, luego creamos variables por cada tapete para obtener las cartas que tenemos, después vaciamos todos los tapetes. Después llamamos a dos métodos que se explicarán más adelante.

```
// vaciar el mazo inicial
mazo_inicial = [];
```

```
var cartasIniciales = tapete_inicial.getElementsByTagName("img");
var cartasSobrantes = tapete_sobrantes.getElementsByTagName("img");
var cartasReceptorUno = tapete_receptor1.getElementsByTagName("img");
var cartasReceptorDos = tapete_receptor2.getElementsByTagName("img");
var cartasReceptorTres = tapete_receptor3.getElementsByTagName("img");
var cartasReceptorCuatro = tapete_receptor4.getElementsByTagName("img");


//vaciar todos los tapetes
while (cartasIniciales.length > 0) {
  tapete_inicial.removeChild(cartasIniciales[0]);
}

while (cartasSobrantes.length > 0) {
  tapete_sobrantes.removeChild(cartasSobrantes[0]);
}

while (cartasReceptorUno.length > 0) {
  tapete_receptor1.removeChild(cartasReceptorUno[0]);
}

while (cartasReceptorDos.length > 0) {
  tapete_receptor2.removeChild(cartasReceptorDos[0]);
}

while (cartasReceptorTres.length > 0) {
  tapete_receptor3.removeChild(cartasReceptorTres[0]);
}

while (cartasReceptorCuatro.length > 0) {
  tapete_receptor4.removeChild(cartasReceptorCuatro[0]);
}

actualizar_inicial();
actualizar_sobrante();
```

Ahora creamos la baraja, la barajamos y la cargamos en el tapete inicial. Esas dos últimas funciones se explican más adelante.

```
//crerar baraja
let cartas;
for (let i = 0; i < palos.length; i++) {
  for (let j = 0; j < numeros.length; j++) {
    cartas = numeros[j] + "-" + palos[i];
    mazo_inicial.push(cartas);
  }
}

barajar(mazo_inicial);  // Barajar

cargar_tapete_inicial(mazo_inicial); // Dejar mazo_inicial en tapete inicial
```

Iniciamos el tiempo y ponemos todos los contadores a 0. Con esto ya finaliza la función comenzar juego.

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


// 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); // iniciar contador de cartas
```

2.3 Función cargar mazo inicial

Después de eso creamos la función para cargar el mazo inicial en el tapete inicial. En esta función lo primero que hacemos es crear una variable paso, después recorremos todas las cartas que tiene el mazo inicial y hacemos una serie de cosas:

- Primero creamos una variable carta, la cual la creamos como un elemento imagen.

```
let carta = document.createElement("img");
```

- Luego creamos otra variable llamada numcarta en la cual vamos a ir sacando el número de cada carta con la propiedad split.

```
let numcarta = cartas[i].split("-")[0];
```

- Después a la variable carta le añadimos un src con la imagen de cada carta.

```
carta.src = "imagenes/baraja/" + cartas[i] + ".png";
```

- A continuación realizamos una serie de comprobaciones para saber el palo de la carta y le añadimos un atributo con el palo.

```
if (cartas[i].includes("cir")) { // dar clases a las cartas
  carta.setAttribute('color', 'cir')
  carta.setAttribute('colores', 'negro')
} else if (cartas[i].includes("hex")) {
  carta.setAttribute('color', 'hex')
  carta.setAttribute('colores', 'negro')
} else if (cartas[i].includes("cua")) {
  carta.setAttribute('color', 'cua')
  carta.setAttribute('colores', 'rojo')
} else if (cartas[i].includes("ova")) {
  carta.setAttribute('color', 'ova')
  carta.setAttribute('colores', 'rojo')
}
```

- Luego le damos a la carta un atributo con el número, tambíen le damos una serie de atributos style para que las cartas aparezcan una debajo de otro en diagonal, le ponemos a todas las cartas que no se puedan arrastrar y le damos un atributo “ondragstart” y “dragStart(event)”.

```
carta.setAttribute('numero', numcarta);
carta.style.position = "absolute";
carta.style.top = paso + "px";
carta.style.left = paso + 3 + "px";
carta.style.width = "60px";
carta.draggable = false;
carta.setAttribute("ondragstart", "dragStart(event)")
tapete_inicial.appendChild(carta);
```

- Y por último cargamos la carta en el tapete inicial, al paso le añadimos 5 y comprobamos que si la carta es la última del mazo le ponemos la propiedad draggable true para que se pueda arrastrar.

```
tapete_inicial.appendChild(carta);
paso += 5;
if (i + 1 == cartas.length) {
  carta.setAttribute('draggable', 'true')
} else {
  carta.setAttribute('draggable', 'false')
}
```

2.4 Función barajar

Creamos la función barajar, que lo único que hace es sacar cartas aleatorias con la función math.random.

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

2.5 Función arrancar_tiempo

Hacemos la función arrancar_tiempo, que inicia un temporizador que cuenta el tiempo en segundos y lo muestra en formato "HH:MM:SS" en un contador.

```
function arrancar_tiempo() {
  /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! **/
  if (temporizador) clearInterval(temporizador);
  let hms = function () {
    let seg = Math.trunc(segundos % 60);
    let min = Math.trunc((segundos % 3600) / 60);
    let hor = Math.trunc((segundos % 86400) / 3600);
    let tiempo = ((hor < 10) ? "0" + hor : "" + hor)
      + ":" + ((min < 10) ? "0" + min : "" + min)
      + ":" + ((seg < 10) ? "0" + seg : "" + seg);
    set_contador(cont_tiempo, tiempo);
    segundos++;
  }
  segundos = 0;
  hms(); // Primera visualización 00:00:00
  temporizador = setInterval(hms, 1000);

} // arrancar_tiempo
```

2.6 Funciones de contadores

Realizamos distintas funciones para los contadores.

```
function inc_contador(contador) {
  contador.innerHTML = +contador.innerHTML + 1;
} // inc_contador

function dec_contador(contador) {
  /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! ***/
  contador.innerHTML = +contador.innerHTML - 1;
} // dec_contador

function set_contador(contador, valor) {
  /*** !!!!!!!!!!!!!!!!!! CÓDIGO !!!!!!!!!!!!!!!!!!! **/
  contador.innerHTML = valor;
} // set_contador
```

2.7 Función actualizar inicial

Creamos la función actualizar inicial, lo que hace es coger todas las cartas del mazo inicial y las actualiza dándole a la última que se pueda arrastrar y dándole la id última y las demás no dejamos arrastrar.

```
function actualizar_inicial() {
  //actualizar baraja
  let cartas = document.querySelectorAll("#inicial img");// cogemos todas las cartas del tapete inicial
  for (let i = 0; i < cartas.length; i++) {
    if (i + 1 == cartas.length) {
      cartas[i].setAttribute('draggable', 'true') //le damos draggable a la ultima carta
      cartas[i].setAttribute('id', 'ultima') // le damos id a la ultima carta
    } else {
      cartas[i].setAttribute('draggable', 'false') // le quitamos draggable a las demas cartas
    }

  }
} // actualizar_baraja
```

2.8 Función actualizar sobrante

Hacemos lo mismo con los sobrantes, pero dándole a todas las cartas una id.

```
function actualizar_sobrante() {
  //actualizar baraja
  let cartas = document.querySelectorAll("#sobrantes img"); // cogemos todas las cartas del tapete sobrante
  for (let i = 0; i < cartas.length; i++) {
    if (i + 1 == cartas.length) {
      cartas[i].setAttribute('draggable', 'true') //le damos draggable a la ultima carta
      cartas[i].setAttribute('id', 'ultimaSobrante') // le damos id a la ultima carta
    } else {
      cartas[i].setAttribute('draggable', 'false') // le quitamos draggable a las demas cartas
      cartas[i].setAttribute('id', 'sobrante') // le damos id a las demas cartas
    }

  }
} // actualizar_baraja
```

2.9 Función dragstart

Creamos la función dragStart para declarar el inicio del drag y mediante una variable miramos si viene del tapete inicial o del sobrante.

```
function dragStart(event) {

  event.dataTransfer.setData("text", event.target.id);
  if (event.target.parentNode.id == "inicial") {
    tap = true;
  } else {
    tap = false;
  }
}
```

2.10 Función allowdrop

Hacemos la función allowDrop con distintos parámetros:

- Para cada receptor si está vacío, que tendrá que ser un rey y si viene del inicial.

```
if (event.target.id == "receptor1" && mazo_receptor1.length == 0 && tap) {//comprobamos si el tapete esta vacio y si viene del inicial
  if (mazo_inicial[mazo_inicial.length - 1].split("-")[0] == 12) { //comprobamos si la carta es un rey
    event.preventDefault();

  }
```

- Para cada receptor si no está vacío, que tendremos que mirar si el número es menor y si el color es distinto y si viene del inicial.

```
} else if (event.target.id == "receptor1" && mazo_receptor1.length > 0 && tap && cartaroja(mazo_inicial) != cartaroja(mazo_receptor1)) {//comprobamos
  if (mazo_inicial[mazo_inicial.length - 1].split("-")[0] == mazo_receptor1[mazo_receptor1.length - 1].split("-")[0] - 1) { //comprobamos si la carta
    event.preventDefault();

  }
```

- Para cada receptor si está vacío, que tendrá que ser un rey y si viene del sobrantes.

```
} else if (event.target.id == "receptor1" && mazo_receptor1.length == 0 && !tap) {//comprobamos si el tapete esta vacio y si viene del sobrantes
  if (mazo_sobrantes[mazo_sobrantes.length - 1].split("-")[0] == 12) { //comprobamos si la carta es un rey
    event.preventDefault();

  }
```

- Para cada receptor si no está vacío, tendremos que mirar si el número es menor y si el color es distinto y si viene del sobrantes.

```
} else if (event.target.id == "receptor1" && mazo_receptor1.length > 0 && !tap && cartaroja(mazo_sobrantes) != cartaroja(mazo_receptor1)) {//comprobam
  if (mazo_sobrantes[mazo_sobrantes.length - 1].split("-")[0] == mazo_receptor1[mazo_receptor1.length - 1].split("-")[0] - 1) { //comprobamos si la ca
    event.preventDefault();

  }
```

- Si va al receptor de sobrantes siempre va a sobrantes y actualizamos el tapete inicial y sobrantes.

```
} else if (event.target.id == "sobrantes") { //comprobamos si va a sobrantes
  event.preventDefault();
}
actualizar_inicial();
actualizar_sobrante();
```

- Por último hacemos la función catarroja, que lo que hace es que si la carta es roja devuelve un true

```
function cartaroja(mazo) { //hacemos una funcion para que si la carta es roja devuelva true y si no false
  if (mazo[mazo.length - 1].split("-")[1] == "ova" || mazo[mazo.length - 1].split("-")[1] == "cua") {
    return true;
  } else {
    return false;
  }
}
```

2.11 Función drop

Creamos la función drop, en la cual vamos a tener en cuenta distintas opciones:

- Primero creamos la variable data, con la cual obtendremos los datos de la carta

```
var data = event.dataTransfer.getData("text");
```

- Vamos a mirar a qué tapete va, si viene del mazo inicial y si el mazo inicial no está vacío, luego lo que vamos a hacer es pushear la carta al mazo que va, luego metemos la carta al tapete, le damos estilos y por último actualizamos tapete, incrementamos el contador de movimientos y actualizamos los contadores de los tapetes (este lo vamos a hacer por cada tapete).

```
if (event.target.id == "receptor1" && tap && mazo_inicial.length > 0) { //controlo el tapete y si viene del inicial

  mazo_receptor1.push(mazo_inicial[mazo_inicial.length - 1]); //añado la carta al tapete
  mazo_inicial.pop(); //quito la carta del inicial

  tapete_receptor1.appendChild(document.getElementById(data));
  tapete_receptor1.lastChild.style.top = "20%";
  tapete_receptor1.lastChild.style.left = "20%";
  actualizar_inicial();
  inc_contador(cont_movimientos); //aumentamos el contador de movimientos
  actualizar_contadores(); //actualizamos los contadores
```

- Hacemos lo mismo pero cuando viene del tapete de sobrantes.

```
} else if (event.target.id == "receptor1" && !tap && mazo_sobrantes.length) {  //controlo el tapete y si viene del
  actualizar_sobrante();
  mazo_receptor1.push(mazo_sobrantes[mazo_sobrantes.length - 1]);
  mazo_sobrantes.pop();


  tapete_receptor1.appendChild(document.getElementById(data));
  console.log(data);
  tapete_receptor1.lastChild.style.top = "20%";
  tapete_receptor1.lastChild.style.left = "20%";
  actualizar_sobrante();
  inc_contador(cont_movimientos);
  actualizar_contadores();
```

Dentro del drop llamamos a la función vaciar sobrantes.

```
vaciar_sobrantes(tapete_sobrantes);
```

2.11.1 Función vaciar_sobrantes

Está función lo que hace es comprobar si el mazo inicial no tiene cartas, entonces lo que hacemos es coger una variable sobrantes que coje todas las cartas del tapete sobrantes, y lo que hace por cada carta es coger su número y su palo y lo carga en el mazo inicial y lo borra del mazo de sobrantes. Después de esto elimino todas las cartas del tapete sobrantes con un delay de 100 milisegundos. Para finalizar barajeo otra vez el mazo inicial, cargo el mazo al tapete inicial y actualizo el mazo inicial y el de sobrantes.

```
function vaciar_sobrantes(tapete_sobrantes) {

  if (mazo_inicial.length == 0) { // Condicional que activa la función cuando el mazo inicial se acaba.
    var sobrantes = tapete_sobrantes.getElementsByTagName("img"); // Obtengo las cartas del tapete de sobrantes.

    for (i = 0; i < sobrantes.length; i++) { // Bucle que recorre los sobrantes, obtiene su número y su palo y los vuelve a añadir a mazo_inicial.
      var num = sobrantes[i].getAttribute("numero");
      var palo = sobrantes[i].getAttribute("color");

      mazo_inicial.push(num + "-" + palo); // Incluye la carta al mazo inicial de nuevo.
      mazo_sobrantes.pop(); // Elimina la carta del mazo sobrantes.
    }

    setTimeout(function () {
      while (sobrantes.length > 0) tapete_sobrantes.removeChild(sobrantes[0]); // Bucle que elimina las cartas del anterior tapete (sobrantes).
    }, 100);


    barajar(mazo_inicial); // Baraja el mazo inicial.
    cargar_tapete_inicial(mazo_inicial); // Carga el mazo inicial en el tapete inicial.
    actualizar_inicial();
    actualizar_contadores();
  }
}
```

2.11.2 Función actualizar_contadores

Creamos la función para actualizar todos los contadores, que lo único que hace es poner los contadores con el tamaño de los mazos.

```
function actualizar_contadores() { // Actualiza los contadores de cartas en cada tapete.
  cont_inicial.innerHTML = mazo_inicial.length;
  cont_sobrantes.innerHTML = mazo_sobrantes.length;
  cont_receptor1.innerHTML = mazo_receptor1.length;
  cont_receptor2.innerHTML = mazo_receptor2.length;
  cont_receptor3.innerHTML = mazo_receptor3.length;
  cont_receptor4.innerHTML = mazo_receptor4.length;

}
```

Para finalizar la función drop, utilizamos la función comprobarfinjuegocorto y sonido.

```
comprobarfinjuegocorto(); //comprueba si se ha ganado el juego en el modo corto
//comprobarfinjuegolargo(); comprueba si se ha ganado el juego en el modo largo, descomentar si se quiere jugar en modo largo
sonido(); // reproduce el sonido cada vez q se mueve una carta
```

2.12 Funciones finales

Para concluir con el javascript hacemos 3 funciones:

- Comprobarfinjuegocorto, lo que hace es comprobar la longitud de los mazos jugando con 4 números para saber si se ha acabado el juego y muestra un alert con un delay de 100 milisegundos.

```
function comprobarfinjuegocorto() {
  if (mazo_inicial.length == 0 && mazo_sobrantes.length == 0 && mazo_receptor1.length == 4 && mazo_receptor2.length == 4 && mazo_receptor3.length == 4 && mazo_receptor4.length == 4) {
    setTimeout(function () {
      alert("Has ganado en " + cont_movimientos.innerHTML + " movimientos y has tardado " + cont_tiempo.innerHTML);
      comenzar_juego();
    }, 100);
  }
```

- Lo mismo pero jugando con 12 números.

```
function comprobarfinjuegolargo() {
  if (mazo_inicial.length == 0 && mazo_sobrantes.length == 0 && mazo_receptor1.length == 12 && mazo_receptor2.length == 12 && mazo_receptor3.length == 12 && mazo_receptor4.length == 12
    setTimeout(function () {
      alert("Has ganado en " + cont_movimientos.innerHTML + " movimientos y has tardado " + cont_tiempo.innerHTML);
      comenzar_juego();
    }, 100);
  }
```

- Por último hacemos la función de sonido, que pone un sonido de burbuja.

```
function sonido() {
  var sonido = new Audio("../burbuja.mp3");
  sonido.play();
  sonido.onended = function () {
    sonido.pause();
    sonido.currentTime = 0;
  };
}
```

3. Conclusiones

Programar un solitario en JavaScript no es fácil si no tienes experiencia, pero es una excelente manera de practicar la resolución de problemas y el pensamiento lógico, ya que se deben tomar en cuenta todas las posibles jugadas y movimientos del jugador para que el juego funcione correctamente.

Puede ser una tarea desafiante y gratificante para aquellos que desean mejorar sus habilidades de programación y crear un proyecto divertido y funcional.

4. Bibliografía

W3Schools. (s.f). *JavaScript Tutorial*. W3Schools. Recuperado el 18 de febrero de 2023 de: https://www.w3schools.com/

ChatGPT. Recuperado el 18 de febrero de 2023 de:

https://chat.openai.com/chat

Francesc Ricart. Recuperado el 18 de febrero de 2023 de:

https://francescricart.com/insertar-y-eliminar-sonido-con-javascript-en-una-pagina-web/

JavaScript Tutorial. Recuperado el 18 de febrero de 2023 de:

https://www.javascripttutorial.net/web-apis/javascript-drag-and-drop/