ACTIVIDAD EVALUACIÓN

JUEGO DE CARTAS

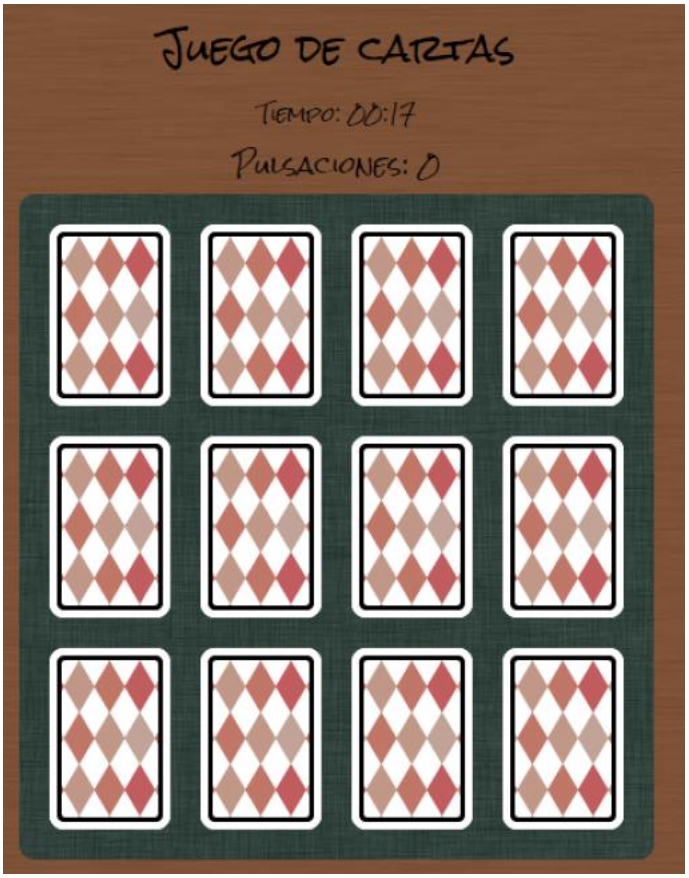
Queremos implementar un juego de cartas usando JavaScript. El juego consta en un tablero de N x M cartas donde se despliegan (NxM)/2 parejas de cartas de manera aleatoria y bocabajo.

- *Por ejemplo, podemos tener un tablero de 2 x 2 cartas donde se despliegan 2 parejas de cartas, uno de 2x3 donde se despliegan 3 parejas o un tablero de 4 x 4 cartas donde se despliegan 8 parejas de cartas.*

El usuario irá seleccionando parejas de dos en dos, y si coinciden se mostrarán boca arriba. El juego finaliza cuando el usuario ha acertado todas las parejas.

Aquí se puede ver el juego en funcionamiento: http://mypuzzle.org/find-the-pair.

En esta fase sólo vamos a implementar la lógica necesaria para el juego, sin entrar al aspecto visual, que finalizaremos en la unidad 4, cuando hablemos del DOM. Pero ver una captura del mismo puede ayudarte:



Para ello, vamos a necesitar las siguientes clases:

1. Una clase llamada **`Carta`** que representa una única carta. La clase tendrá:
    i. Dos propiedades privadas:
        - **`palo`**. Palo de la carta.
        - **`nombre`**: Nombre de la carta.
    ii. Al menos los siguientes métodos:
        - Constructor, getters y setters.
        - El método **`toString():String`** que devuelve una representación como cadena de la carta (Ejemplo: 2-PICAS)

b. Una clase llamada **`Baraja`** que nos permite representar las cartas de una baraja completa y que consta de:
    i. Una propiedad llamada **`cartas`** que consiste en un array bidimensional de objetos **`Carta`**. Las filas consisten en los palos, y las columnas el nombre de la carta.
    ii. Los siguientes métodos:
        - Constructor sin parámetros. Inicializa el array de cartas para crear los correspondientes objetos de tipo **`Carta`**.
        - **`generaCarta(): Carta`** escoge aleatoriamente una carta (nombre y palo) y la devuelve.

Dentro del archivo donde se define la clase Baraja, definiremos un array constante con los palos (podrán ser PICAS, CORAZONES, TRÉBOLES o DIAMANTES) y los nombres posibles (pueden ser A, 2, 3, 4, 5, 6, 7, 8, 9, J, Q, K)

c. Una clase llamada **`Partida`** que nos permite gestionar los datos de una partida. Consta de:
    i. Dos propiedades **`filas`** y **`columnas`** que almacenarán el tamaño del tablero en el que se repartirán las cartas en esa partida. Estos valores los escoge el usuario. Será necesario asignar valores por defecto en el caso de que el usuario no dé números válidos o la combinación escogida no sea un número par de cartas (necesario para hacer parejas).
    ii. Una propiedad llamada **`baraja`** que es un objeto de tipo **`Baraja`**.
    iii. Una propiedad **`cartasSeleccionadas`** que son las parejas necesarias para comenzar la partida, y que se separan de la baraja completa el principio de la partida.
    iv. Una propiedad llamado **`mazo`** consistente en un array bidimensional del número de filas y columnas establecido para esa partida. Contendrá objetos **`Carta`**, que son los que tenemos en la partida actual.
    v. Una propiedad llamada **`cartaVolteada`** que representa la última carta que hemos volteado.
    vi. Una propiedad llamada **`aciertos`** de tipo **`entero`** que contará las cartas que hemos acertado.

vii. Una propiedad **`numeroIntentos`** de tipo entero que almacena el número de intentos realizados.

viii. Los siguientes métodos:

- **`constructor (filas, columnas)`**. Inicializa las propiedades.
- **`selecciona().`** Extrae cartas de la baraja y las inserta por duplicado en el array de cartas seleccionadas.
- **`baraja().`** Reordena las cartas seleccionadas de manera aleatoria. Para ello, se recomienda usar la función `sort()` pasándole una función de ordenación.
- **`reparte().`** Inicializa el tablero a filas x columnas. Extrae las cartas seleccionadas y las inserta secuencialmente en el mazo.
- **`voltea(fila, columna).`** Voltea la carta cuya posición le pasamos y guarda estas posiciones. El número de intentos se incrementa en 1.
- **`compruebaAcierto(fila, columna): boolean.`** Comprueba si la carta cuya posición le pasamos coincide con la que está volteada. En caso afirmativo, devolvemos true e incrementamos el número de aciertos
- **`haFinalizado(): boolean`**. Devuelve verdadero si la partida ha finalizado (hemos acertado todas las cartas).
- **`_cartaEnMazo()`**: Método privado para comprobar si ya hemos seleccionado una determinada carta en el método selecciona(). Este método podría evitarse si eliminamos la carta de la baraja una vez que la hemos seleccionado.

Crearemos un fichero con el código principal, donde seleccionaremos las cartas de la partida, las barajaremos y las repartiremos en forma de tabla de acuerdo con el tamaño seleccionado por el usuario.

A continuación, se mostrará una tabla con borde en el fichero html, que mostrará el contenido de las cartas de la partida (en el juego real estas cartas están boca a bajo y no podremos verlas.

Juego de cartas

9-TREBOLES	A-DIAMANTES	4-DIAMANTES
A-DIAMANTES	4-DIAMANTES	9-TREBOLES

La función para mostrar la tabla sería algo así:

```
function mostrarTabla(){
var codigoHTML="<table border=1>"
for (var i = 0; i < partida._mazo.length; i++) {
    codigoHTML+="<tr>"
    for (var j=0;j<partida._mazo[i].length; j++){
        if (partida._mazo[i][j]==null)
            codigoHTML="<td></td>"
        else
            codigoHTML+="<td><br>"+partida._mazo[i][j]+"<br></td>";
    }
        codigoHTML+="</tr>"
    }
    codigoHTML+="</table>"
document.getElementById("mazo").innerHTML=codigoHTML;
}
```

Cada 5 segundos le pediremos al usuario dos veces que introduzca una posición en formato "fila-columna" que simulará el clic sobre las dos cartas cuando tengamos la interfaz gráfica. Voltearemos la primera y comprobaremos el acierto con la segunda. El código de partida (incompleto) sería algo así:

```
function pedirCartas(){
    // Pedir carta 1
    // Voltear carta 1
    // Pedir carta 2
    // Comprobar acierto
    if(partida.haFinalizado()){
        console.log("PARTIDA FINALIZADA!!");
    }
    else
        setTimeout(pedirCartas(), 5000)
}
```

Mientras tanto en la consola simularemos el juego real, mostrando las cartas de forma genérica y rellenando las posiciones acertadas con **null**:

```
▼Array(2) i
  ▶0: (3) [null, Carta, Carta]
  ▶1: (3) [Carta, Carta, null]
```