Programación Avanzada
Ingeniería Informática en Sistemas de Información - Curso 2020/2021
ENSEÑANZAS PRÁCTICAS Y DE DESARROLLO
Prueba de evaluación 2ª convocatoria: JavaScript y jQuery

Nombre:_____

Importante: El resultado de esta prueba consistirá en una carpeta comprimida en un archivo con formato ZIP. Dicha carpeta deberá contener exclusivamente los archivos JavaScript necesarios para la resolución del ejercicio. El nombre del fichero tendrá un formato específico dictado por el nombre de cada alumno. Por ejemplo, para un alumno llamado "José María Núñez Pérez" el fichero se nombrará como NunyezPerezJM.zip. Obsérvese que las tildes son ignoradas y las eñes sustituidas. **Las rutas de los ficheros empleados serán relativas, a fin de que las resoluciones a los ejercicios y problemas puedan ser examinadas en cualquier equipo. Cualquier entrega que no cumpla las reglas de nombrado, el formato de compresión del archivo o el contenido de los archivos del mismo, será penalizada con 2 puntos sobre 10 por cada incumplimiento. Pasado el límite de entrega se aceptará el envío del trabajo, con una penalización de 2 puntos sobre 10 de la calificación por cada minuto o fracción de retraso a partir del tercer minuto.**

Objetivos

- Demostrar los conocimientos adquiridos en el desarrollo de páginas web interactivas con JavaScript y jQuery.

Descripción del ejercicio propuesto

IMPORTANTE:
Recuerde que el uso del atributo innerHTML no se tolera y su uso invalidará el ejercicio entregado.
Los archivos HTML NO DEBEN SER MODIFICADOS DE NINGÚN MODO para implementar las funcionalidades requeridas.

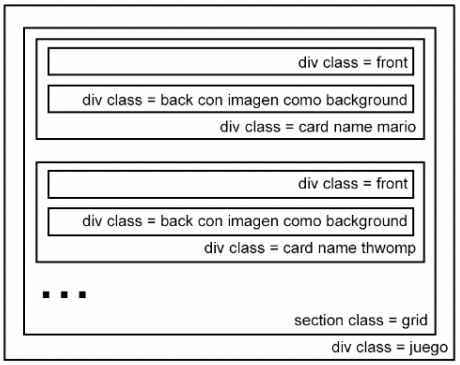
EJERCICIO 1 [**7 puntos**]. Usando exclusivamente **JavaScript** y partiendo de la página *index.html* suministrada como material adicional, incluya en un archivo llamado *juego.js* todo el código necesario para dotar a la página de las siguientes funcionalidades interactivas:

> **IMPORTANTE: Dispone en el material adicional del archivo juegoOfuscado.js para poder realizar una demostración de cómo funciona el juego y entender su funcionamiento.**

1. [**0.25 puntos**] Cree una variable llamada *vectorElementos*. Esta variable debe ser un array asociativo donde cada elemento del vector tendrá dos componentes (name e img). Este vector se deberá rellenar con la información que puede encontrar en el archivo "vectorElementos.txt" del material adicional. Como se puede observar, cada elemento de este vector representa una de las posibles cartas del juego, con su nombre y su imagen asociada (Que deberán estar en la carpeta *img*).

2. [**0.25 puntos**] Cree un vector tableroJuego que representará el tablero. Esta variable contendrá todos los elementos de la variable *vectorElementos* repetidos dos veces, ya que hay dos copias de la misma carta en el tablero. Puede usar la función *concat* para esta tarea. Una vez relleno, el contenido de este se ordenará de forma aleatoria para que las cartas estén desordenadas en cada partida.

3. [**1 puntos**] Prepare el tablero de juego en el documento HTML. Para ello debe crear una estructura similar a la siguiente:



Para conseguir esta estructura, deberá:

a) Estructurar el tablero de juego, de tal forma que debe establecer un elemento del tipo 'section' asignando la etiqueta de clase 'grid'. Este elemento deberá ubicarse como elemento hijo al contenedor del juego.
b) Debe rellenar la sección grid creada anteriormente con los pares de imágenes ordenados aleatoriamente en la variable *tableroJuego*. Para ello, debe iterar sobre la variable *tableroJuego* y realizar las siguientes operaciones:
c) Cree el elemento 'card', que deberá ser un elemento de tipo 'div'. Asigne a este elemento la clase 'card' y el atributo 'name' al que le corresponda (*name* del elemento iterado). Cada elemento de este tipo representa una carta en el tablero.
d) Cree el elemento 'front', que deberá ser un elemento de tipo 'div'. A este elemento se le deberá asignar la clase 'front'. Este elemento representa la cara visible de la carta al inicio del juego. Además, se le deberá asociar un evento al hacer clic sobre este elemento de tal forma que se ejecutará la función *procesarClick*, que se detalla en el punto 4.
e) Cree el elemento 'back', que deberá ser un elemento de tipo 'div'. A este elemento se le deberá asignar la clase 'back' y añadirle estilo de tal forma que contenga la imagen de fondo que le corresponda (img del elemento iterado). Este elemento representa la cara oculta de la carta al inicio del juego. Como se puede observar, la clase CSS provoca que no sea visible.
f) Adicionalmente, necesitará definir dos variables globales que identifiquen a los dos elementos seleccionados del tablero (al iniciar el juego no tendrán valor), y otra variable que le permita controlar cuantos elementos ha seleccionado del tablero.

Puede utilizar el inspector de elementos de su navegador con el ejemplo facilitado en el material adicional para ver esta estructura.

4. [**3.5 puntos**] Cree una función llamada *procesarClick*, de tal forma que realice las siguientes operaciones:

a) Compruebe las siguientes condiciones:
    1. Si el nodo **padre** del elemento que ha disparado el evento contiene la clase 'selected'.
    2. Si el nodo **padre** del elemento que ha disparado el evento contiene la clase 'match'.

   En caso de que se cumplan cualquiera de las dos condiciones, se deberá finalizar la ejecución de la función (return).
b) Se deberá comprobar cuántos elementos se han seleccionado (a través de una variable global *contador*), cuyo límite estará fijado en 2 (no se pueden seleccionar más de dos elementos para comprobar si hay un match) y que inicialmente tendrá el valor 0. Si no se han seleccionado dos elementos (descritos en el punto 3.f), deberá comprobar si se ha seleccionado uno o ninguno, de tal forma que:
    1. Si no tiene ningún elemento seleccionado, deberá almacenar el nombre del elemento que ha disparado el evento en una de las variables globales.
    2. Si ya tiene un elemento seleccionado previamente, deberá almacenar el nombre del elemento que ha disparado el evento en otra de las variables globales.
    3. En cualquiera de los dos casos anteriores, deberá asignar la clase "selected" al elemento padre (div con clase "card") de aquel elemento que ha disparado el evento.
c) En el caso de que haya seleccionado dos elementos, se deberá comprobar si son del mismo tipo y coinciden sus contenidos (comparación estricta), en cuyo caso deberá:
    1. Mostrar el *div* con identificador "mensajeAcierto".
    2. Invocar a la función *match* después de 1 segundo (La función *match* se definirá en el punto 5).

d) En caso contrario, deberá mostrar el div con identificador “mensajeError”.
e) En cualquier otro caso, se deberá invocar a la función *resetGuesses* después de 1 segundo, que reiniciará los elementos seleccionados al no haber encontrado *match* (La función *match* se definirá en el ejercicio 5)

5. [**1 punto**] Cree la función llamada ‘match’, que deberá marcar con una clase concreta a los elementos cuando se encuentre el par de imágenes adecuado. Para ello, obtenga todos los elementos que tengan asignado la clase “selected” y asígnele además la clase “match”. Finalmente, oculte el elemento con identificador ‘mensajeAcierto”.

6. [**1 punto**] Implemente la función *resetGuesses*, cuya función es reiniciar los elementos que se han seleccionado. Para ello deberá obtener todos los elementos que tengan asignada la clase “selected” y deberá eliminarse dicha clase. Además, debe asegurarse que aquellas variables que identifican a cada uno de los elementos y el número de elementos seleccionados son reiniciadas convenientemente. Finalmente, oculte el elemento con identificador mensajeError”.

EJERCICIO 2 [**3 puntos**]. Usando **jQuery** y partiendo de la página *formulario.html*, incluya en un archivo *validacion.js* las siguientes funciones:

a) [**0.5 puntos**] Una función llamada *marcarError($elemento, mensaje)*, que recibirá un elemento (campo *input*) y una cadena de caracteres. Esta función deberá comprobar si el elemento anterior al pasado como parámetro es una etiqueta del tipo *span* y si contiene la clase error. En caso de que no exista la etiqueta *span*:
   a) Aplicar un color de fondo rojo y un color de texto blanco.
   b) Añadir antes del elemento una etiqueta *span* con la clase “error” y el mensaje pasado como argumento. Por ejemplo, si el mensaje pasado es “Debe rellenar el campo”, se debería añadir:

```
<span class='error'>Debe rellenar el campo</span>
```

b) [**0.3 puntos**] Una función llamada *enfocado($elemento),* que recibirá como parámetro un elemento HTML. En esta función, se deberá modificar el color de fondo, estableciéndose a blanco. Por otro lado, se deberá establecer el color del texto en negro. Además, si existe el elemento *span* de error justo antes del elemento enviado como parámetro, deberá eliminarse.
c) [**0.3 puntos**] Una función *comprobarVacio($elemento)*, que recibirá como parámetro un elemento de tipo *input*. Esta función deberá comprobar si el elemento contiene algún valor. En caso de que esté vacío se deberá marcar el error, usando para ello la función desarrollada anteriormente.

Además, deberá dotar a la página de las siguientes funcionalidades una vez que el documento esté preparado:

a) [**0.1 punto**] Ocultar el bloque con clase “seleccionDosis”.
b) [**0.1 punto**] Cuando se centre el foco en algún elemento de tipo *input*, se deberá invocar a la función *enfocado*, implementada anteriormente.
c) [**0.3 puntos**] Cuando se pierda el foco de algún elemento input, se deberá invocar a la función *comprobarVacio*, implementada anteriormente.
d) [**0.3 puntos**] Cuando cambie el contenido del campo con nombre ‘apellidosPaciente’, se deberá comprobar que el texto introducido contiene al menos un espacio. En caso que no lo contenga, se deberá marcar un error informando del mismo.
e) [**0.3 puntos**] Cuando cambie el contenido del campo con nombre ‘dni’, se deberá comprobar que está compuesto por 8 dígitos y una letra. En caso que no cumpla el formato, deberá marcarse un error notificándose. Para comprobar el DNI puede basarse en la siguiente expresión regular (disponible en el material adicional):

```
/^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKET]{1}$/i
```

f) [**0.3 puntos**] Cuando se cambie el valor seleccionado del elemento con nombre ‘vacuna’, se deberá comprobar que se haya seleccionado alguna. En caso de que no se haya seleccionado, se deberá agregar el siguiente elemento después del formulario:

```
<h3 class='aviso' style='color:red;'>
      ATENCIÓN:</br>
      Debe seleccionar una vacuna
</h3>
```

En caso de que sí, se deberán eliminar todos los elementos con la clase ‘aviso’. Además, se deberá comprobar si la vacuna es distinta a “janssen”, en cuyo caso se deberá mostrar el bloque de contenido “seleccionDosis”.

g) [**0.5 puntos**] Cuando se envíe el formulario se deberá provocar la comprobación de errores disparando de forma manual la pérdida del foco de todos los componentes del formulario. Además, se deberá comprobar si se ha seleccionado un resultado o no. En caso de no haberlo seleccionado, se debe marcar un error sobre el primer elemento del *radiobutton*. Finalmente, si se encuentra alguna etiqueta *span* con clase error, se mostrará una alerta y no se podrá enviar el formulario.

Material suministrado

Como material adicional dispondrá del fichero "Material Adicional.7zip" que contendrá:

Páginas HTML base para crear las páginas webs interactivas.

Carpeta js con la librería jQuery.

Fichero de texto con la expresión regular del ejercicio 2.e.