	Programación Avanzada **Ingeniería Informática en Sistemas de Información - Curso 2019/2020** **ENSEÑANZAS PRÁCTICAS Y DE DESARROLLO** **Prueba de evaluación: JavaScript y jQuery**	

Nombre:_____ **Equipo:**_____

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 de 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 [**5 puntos**]. Usando exclusivamente **JavaScript** y partiendo de la página *juego.html* suministrada como material adicional, incluya en un archivo JavaScript, llamado *tresrayas.js* todo el código necesario para dotar a la página de las siguientes funcionalidades interactivas:

1. [**0.1 punto**] Cree una variable llamada *turnoJugador* inicializada con el valor 1 y un vector llamado *tablero* de 9 posiciones inicializadas con el valor 0 que harán referencia a cada una de las posiciones del juego "tres en raya". Aunque el tablero de juego de tres en raya es bidimensional, éste se representará de forma unidimensional empleando es esquema de numeración de celdas que se desprende del código HTML en la llamada a la función *seleccionarCelda*.

2. [**1 punto**] Implemente una función llamada *marcarCelda(idCelda)*, que reciba como argumento el índice de de la celda pulsada. Esta función deberá rellenar el color de fondo de la celda seleccionada en base al valor almacenado en la posición del tablero pasado como argumento.
   a) Si el valor almacenado es 0 (no seleccionado), se deberá rellenar la celda con el color blanco.
   b) Si el valor almacenado es 1 (seleccionado por el jugador 1), se deberá rellenar la celda con el color rojo.
   c) Si el valor almacenado es 2 (seleccionado por el jugador 2), se deberá rellenar la celda con el color azul.

3. [**0.9 puntos**] Implemente una función llamada *jugarDeNuevo( )*, que muestre un mensaje de confirmación preguntando si desea jugar otra vez. En caso de que se confirme, se deberá establecer todo el vector tablero a 0 y rellenar la celda de color blanco(empleando para esto último la función *marcarCelda* implementada anteriormente).

4. [**1 punto**] Implemente una función llamada *comprobarGanador( )*. Esta función deberá comprobar si se ha dado un empate, si hay un ganador o si sigue el juego, de tal forma que devuelva:
   a) El valor 1 si ha ganado el jugador 1.
   b) El valor 2 si ha ganado el jugador 2.
   c) El valor 3 si se ha terminado en empate.
   d) El valor 0 en cualquier otro caso.

Tenga en cuenta que para que exista un ganador debe darse el caso de que tres celdas consecutivas sean del mismo color, pudiendo ser de forma horizontal, vertical o diagonal. Si no se cumplen ninguna de las condiciones necesarias para que exista un ganador, deberá comprobarse que todas las celdas están rellenas (es decir, no hay ninguna celda con valor cero), en cuyo caso será empate. Por último, si hay celdas sin rellenar, se podrá continuar con el juego.

5. [**1 punto**] Implemente una función llamada *actualizarPuntuación(ganador)*, que reciba como argumento el resultado del juego (con la misma codificación empleada en la función *comprobarGanador*). Esta función deberá mostrar una alerta indicando el jugador que ha ganado la partida o si ha quedado empate. Además, se deberá incrementar el contador correspondiente de victorias o empates indicado en el HTML.

6. [**1 punto**] Implemente una función llamada *seleccionarCelda(idCelda)*, que reciba como argumento el índice de la celda pulsada. Esta función deberá realizar las siguientes acciones:
    a) Comprobar si la celda se ha pulsado previamente, es decir, si el valor almacenado en el vector es distinto de cero. En caso de que la celda ya haya sido seleccionada previamente, se deberá mostrar una alerta indicando que la casilla ya está ocupada.
    b) Si la celda no se había seleccionado, se deberá almacenar en el vector el valor correspondiente al jugador que ha seleccionado la celda, marcarla de color (usando las funciones desarrolladas previamente), y cambiar el turno del jugador, teniendo en cuenta que debe verse reflejado el cambio en el HTML.
    c) Comprobar si el juego ha terminado, es decir, si hay un ganador, empate o si se puede continuar jugando. Puede utilizar para ello la función *comprobarGanador* desarrollada anteriormente. En el caso de que el juego haya finalizado (exista un ganador o se de un empate), se debe actualizar las puntuaciones, haciendo uso de la función *actualizarPuntuacion* desarrollada anteriormente.
    d) Por último, se debe preguntar si se desea seguir jugando, haciendo uso de la función *jugarDeNuevo*.

EJERCICIO 2 [**4 puntos**]. Usando **jQuery** y partiendo de la página *formulario.html*, incluya en un archivo *comprobacion.js* las siguiente funciones:

a) [**0,75 puntos**] Una función llamada *marcaError($elemento, mensaje)*, que recibirá un elemento (campo *input*) y una cadena de caracteres. Esta función deberá comprobar si el siguiente elemento al pasado como parámetro es una etiqueta del tipo *span* y si contiene la clase error. En caso de que no exista la etiqueta *span*:
    a) Aplicar un borde sólido de 2 px y color rojo al elemento pasado como parámetro.
    b) Añadir después del elemento una etiqueta *span* con la clase "error" y el mensaje pasado como argumento. Por ejemplo, si el mensaje pasado es "Debe rellenar el campo", se debería añadir <*span class='error'*>*"Debe rellenar el campo*</*span*>
b) [**0,75 puntos**] Una función llamada *enfocado($elemento),* que recibirá como parámetro un elemento HTML. En esta función, se deberá modificar el borde, estableciéndolo con un estilo sólido de 2 px y color verde. Además, si existe el elemento *span* de error justo después del elemento enviado como parámetro, deberá eliminarse con el efecto *fadeOut* de forma *lenta*.
c) [**0,25 puntos**] Una función comprobarVacio($elemento), que recibirá como parámetro un elemento HTML. Esta función deberá comprobar si el elemento contiene algún valor. En caso de que no esté vacío, se establecerá el borde por defecto (sin estilo). En caso contrario, 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) [**0,25 puntos**] Cuando se centre el foco en algún elemento del tipo input, se deberá invocar a la función *enfocado*, implementada anteriormente.
b) [**0,25 puntos**] Cuando se pierda el foco de algún elemento input, se deberá invocar a la función *comprobarVacio*, implementada anteriormente.
c) [1**,25 puntos**] Cuando se modifique el elemento con identificador "*numIntegrantes*", deberá:
    a) [**0,25 puntos**] Vaciar el listado, cuyo identificador corresponde a componentes.
    b) [**1 punto**] Obtener el número de integrantes del input y el tipo de agrupación que se ha seleccionado. Una vez obtenidos, se deberá comprobar:
        a) Si no se ha seleccionado ningún tipo de agrupación, se deberá marcar un error usando la función marcarError.
        b) Si se ha seleccionado "cuarteto" y el número de integrantes es mayor a 5, se ha seleccionado "chirigota" o "comparsa" y el número de integrantes es mayor a 13 o si se ha seleccionado "coro" y el número de integrantes es mayor a 30, se deberá marcar un error usando la función marcarError.
        c) En cualquier otro caso, se deberá añadir al elemento con identificador "componentes", un identificador del componente a añadir y una entrada de formulario de tipo texto con la clase "componente". Por ejemplo, si se han seleccionado 3 componentes, se deberá mostrar:
            Componente 1: <input type="text" class="componente"/>
            Componente 2: <input type="text" class="componente"/>

Componente 3: <input type="text" class="componente"/>

d) Además, se deberá establecer los eventos a cada uno de los inputs incluidos en el elemento con identificador componentes, de tal forma que cuando se tenga el foco se invoque a la función enfocado, y cuando se pierda el foco se invoque a la función comprobarVacio.

d) [**0,15 puntos**] Cuando se modifique la entrada de formulario con nombre modalidad, se deberá invocar al evento de cambio del elemento con identificador numIntegrantes.

e) [**0,15 puntos**] Comprobar si el elemento con identificador numIntegrantes tiene valor, en cuyo caso habrá que invocar al elemento de cambio a dicho elemento.

f) [**0,2 puntos**] Al enviar el formulario, se deberá provocar la comprobación de errores disparando de forma manual la pérdida del foco de todos los componentes del formulario. Finalmente, si se encuentra alguna etiqueta *span* con clase error, se mostrará una alerta y no se podrá enviar el formulario.

EJERCICIO 3 [**1 punto**]. Partiendo de la página *plugin.html* y haciendo uso del **plugin jQuery DataTables** haga que:

a) Se pueda seleccionar el número de registros que se desean mostrar entre 5, 10, 15 o "all", que mostrará todos los elementos de la tabla. [**0,5 puntos**]

b) Deshabilitar la búsqueda dentro de la tabla. [**0,5 puntos**]

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 las librerías jQuery y el código del plugin dataTables.
- Carpeta css con el CSS necesario para el funcionamiento del plugin dataTables.
- Documentación del plugin dataTables.

Datos de la prueba

Autor del documento: José F. Torres (Enero 2020).

Revisiones del documento