	Programación Avanzada **Ingeniería Informática en Sistemas de Información - Curso 2020/2021** **ENSEÑANZAS PRÁCTICAS Y DE DESARROLLO** **Prueba de evaluación: JavaScript & 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 de los ejercicios. No incluya ningún otro archivo. 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. No se aceptará ningún envío que no cumpla con las anteriores especificaciones de formato y nombrado. **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. No es necesario que incluya el script de la base de datos en el envío.**

Objetivos

- Demostrar los conocimientos adquiridos en el desarrollo de aplicaciones empleando 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.**

EJ. 1 [8 puntos]. Se va a implementar el juego del ahorcado, cuyo objetivo es adivinar una palabra a base de ir seleccionando posibles letras que la formarán. Usando exclusivamente código JavaScript y partiendo de la página Ahorcado.html, implemente en un archivo llamado *ahorcado.js* todo el código necesario para dotar a la página de las siguientes funcionalidades interactivas:

1. [**0.5 puntos**] Cargue en una variable global el vector que puede encontrar en el archivo “vector.js”, cuyos elementos están compuesto por pares palabra-pista. Implemente una función llamada *generaPalabra* que seleccione de forma aleatoria un elemento de ese vector. El primer elemento (clave) de la palabra seleccionada en mayúscula se deberá asignar como valor a una variable global llamada “palabra”.

2. [**0.5 puntos**] Implemente una función llamada *pintarGuiones*, que reciba como argumento un valor entero *N* que identificará la longitud de la palabra que se debe adivinar. La función deberá rellenar un vector global llamado “oculta” de *N* posiciones con guiones bajos (_). Asimismo, se deberá cargar en el elemento HTML con identificador “palabra” el contenido del vector en forma de cadena. Para ello, puede concatenar todos sus elementos.

3. [**2 puntos**] Implemente una función llamada *generaABC* que incluya en el elemento HTML, cuyo identificador corresponde a “abcdario”, un botón por cada letra del abecedario en mayúsculas (de la A a la Z). Asigne a todos los botones la clase “letra” y como identificador del botón la misma letra. Por ejemplo, el botón correspondiente a la letra “A” deberá tener la clase CSS “letra” y el valor de identificador “A”. Además, se debe dotar a cada botón de un evento de tal forma que cuando se haga clic ellos se invoque a la función *intento* (que se desarrollará más adelante), pasando como argumento el identificador de dicho botón. Finalmente, añada cada uno de los botones creados a un vector global llamado “buttons”, que deberá definir previamente.

   Puede iterar sobre los diferentes caracteres y utilizar el método *charCodeAt para encontrar el código Unicode (equivalente a los ASCII a nivel del alfabeto) de partida y finalización del bucle* y *fromCharCode* (puede encontrar documentación en el material adicional) para tratar con ellos. Tenga en cuenta que, por simplicidad, no se considerará la creación de un botón para la letra “Ñ”.

4. [**2 puntos**] Implemente la función *intento*, que reciba como argumento una letra y compruebe si dicha letra está en la palabra a descubrir, de tal forma que:

   a. Si la letra está contenida en la palabra, se deberá descubrir la/las posiciones de la variable global llamada “oculta”, desvelando los guiones que correspondan a la letra pasada como argumento (modificando dicho vector) y actualizar el contenido del elemento HTML con identificador “palabra”, mostrando las letras acertadas y

guiones restantes. Además, con el fin de mostrar los mensajes de interacción, se debe modificar el contenido del elemento con identificador "msg" mostrando un mensaje que diga "ACIERTO!". Asigne a dicho elemento la clase "msg acierto".

b. Si la letra no está contenida en la palabra, se deberá tener en cuenta que se ha agotado un intento de los que había disponible. Este intento fallido debe verse reflejado en el elemento HTML con identificador "intentos", siendo éste el valor de una variable global (descrita en el punto 7) que lleve el conteo de los intentos restantes. Además, se deberá mostrar un mensaje que diga "FALLO", utilizando el mismo razonamiento que en el punto anterior, con la salvedad de que en este caso la clase será "msg fallo".

   Por último, se deberá desvelar una imagen en función del número de intentos consumidos. Para ello, hay definidas un total de 6 imágenes que estarán inicialmente ocultas, de tal forma que cada vez que se cometa un fallo se deberá desvelar una. La imagen a desvelar dependerá del número de intentos que se hayan errado. Para mostrar la imagen, deberá seleccionarla en función de su identificador ("imageX"), donde X hace referencia al número de intento, y asignarle la clase "fade-in", además de las que ya tenía.

Finalmente, se deberá comprobar si se ha finalizado el juego, llamando para ello a la función *compruebaFin* (definida más adelante) y limpiar la zona donde se muestran los mensajes de acierto o fallo después de 2 segundos.

5. [**0.25 puntos**] Implemente la función pista, que deberá añadir al elemento con identificador "hueco-pista" la pista correspondiente a la palabra que ha sido seleccionada.

6. [**2 puntos**] Implemente la función *compruebaFin*, que deberá verificar si se ha descubierto la palabra al completo (no existen guiones), o si se han agotado el número máximo de intentos. Para el primer caso, se deberá mostrar un mensaje de felicitación, y para el segundo, un mensaje de "Game Over". Utilice para ello el elemento cuyo identificador es "msg-final". Dote a este elemento de la clase "zoom-in".

   Además, en cualquiera de los casos se deberán deshabilitar los botones correspondientes al abecedario (puede iterar sobre la variable buttons) y modificar el texto del botón con identificador "reset", así como el código que tiene asociado su evento clic, de tal forma que, en lugar de que se llame a la función *inicio*, se recargue la página.

7. [**0.75 puntos**] Implemente la función *inicio*, que deberá inicializar el juego. Para ello, genere la palabra aleatoria, pinte los guiones en el documento HTML, establezca los botones con las letras del abecedario y establezca el número máximo de intentos a 6, quedando registrado en una variable contador y mostrándose el valor de la misma en el elemento HTML con identificador "intentos".

EJ. 2 [2 puntos]. Implemente la validación de un formulario utilizando jQuery. Se generará un *plugin* que permita ser reutilizable. Para ello, implemente las siguientes funcionalidades:

1. [**1 punto**] Desarrolle un *plugin* de jQuery en un archivo llamado "*examen.js*" que permita realizar validaciones sobre campos de formulario:

   a. [**0.25 puntos**] Implemente la función *comprobarRelleno*, que recibirá como argumento el nombre de un campo determinado. Si el valor del elemento que invoca a la función está vacío, deberá añadir después del mismo un elemento con la clases "error" y "msg_XXX", donde XXX corresponderá al nombre del campo. El mensaje contenido en la etiqueta *span* deberá ser "El XXX debe estar relleno".

   b. [**0.25 puntos**] Implemente la función *limpiarMensajes*, que recibirá como argumento el nombre de un campo determinado. Esta función deberá eliminar el elemento *span* con clase "error" que haya justo después de dicho campo, en caso de que la hubiera.

   c. [**0.25 puntos**] Implemente la función *validarEmail*, que deberá comprobar si el valor del elemento que invoca a la función es un *email* o no. Para ello puede utilizar la expresión regular facilitada en el material adicional. En caso de que el elemento no sea un *email*, se deberá añadir un elemento con la clase "error" que indique el mensaje "El email no tiene el formato correcto".

d. [**0.25 puntos**] Implemente la función *hayErrores*, que devolverá un valor booleano en función de si hay o no algún elemento con clase "error" en el documento.

2. [**1 punto**] Implemente en un archivo llamado "*miscript.js*" el código jQuery necesario para crear el siguiente comportamiento. Puede emplear para ello las funciones implementadas en el plugin.:

   a. [**0.25 puntos**] Cuando se ponga el foco en algún elemento del tipo texto o en un *textarea*, se deberán limpiar todos los errores asociados a dicho elemento, en caso de que los hubiera.

   b. [**0.25 puntos**] Cuando se pierda el foco de algún elemento del tipo texto o *textarea*, se deberá comprobar que el campo esté relleno. Además, si el campo es del tipo *email*, se deberá asegurar que cumple el formato.

   c. [**0.5 puntos**] Cuando se intente enviar el formulario, se deberá, en primer lugar, limpiar todos los mensajes de error. Posteriormente, volver a comprobar que todos los elementos cumplen con las condiciones (todos rellenos y que además el *email* tenga el formato correcto). El formulario sólo se deberá enviar en caso de que no haya errores en los campos.

Material suministrado

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

- Carpeta ahorcado:
  - Carpeta img que contiene las imágenes necesarias para visualizar el juego.
  - Hoja de estilo para formatear el juego.
  - Documentación en formato PDF del método *fromCharCode*.
  - Fichero HTML de partida.
  - Vector necesario para la implementación del juego.
  - Carpeta *demo*, donde podrá tener una demostración de cómo funciona el juego.
- Carpeta validación:
  - Hoja de estilo.
  - Fichero HTML de partida.
  - Librería jQuery.

Datos de la prueba

Autor del documento: José Francisco Torres Maldonado (Enero 2021).

Revisiones del documento

1. Carlos D, Barranco González (Enero 2021): Mejoras de texto y formato.