Relación 11 - Introducción a JavaScript

Ejercicio 1. Cree una nueva web, con un esqueleto básico generado mediante Emmet, e incorpore en la cabecera de la página (no en su cuerpo) un código dinámico en el cliente que genere un mensaje emergente al usuario, dándole la bienvenida a la página. Cree una segunda versión de la página que repita el proceso pero lo realice al final del cuerpo de la página.

Ejercicio 2. Repita el ejercicio anterior, pero en este caso haciendo uso de un archivo externo con la dinámica del cliente.

Ejercicio 3. Añada al cuerpo de la página anterior una cabecera de primer nivel, y un par de párrafos con texto autogenerado. A continuación, deberá generar mediante código dinámico en el cliente un nuevo párrafo, detrás de los anteriores, indicando que ese texto se ha generado dinámicamente.

Ejercicio 4. Incorpore, en la página anterior o en cualquier otra, un botón tal que, al hacer *click* sobre él, muestre un mensaje emergente al usuario. Cree una segunda versión que, en lugar de mostrar un mensaje al usuario, lo muestre en el cuerpo de la página. En una tercera versión, haga que el mensaje se imprima por consola.

Ejercicio 5. En la página del ejercicio anterior, incorpore un manejador de evento a todo párrafo, de modo que, al hacer *click* sobre él, muestre un mensaje emergente al usuario.

Ejercicio 6. En la página del ejercicio anterior, incluya un nuevo control de texto, un elemento desplegable y un nuevo botón. Al hacer *click* sobre el botón, deberá mostrar en mensaje emergente el valor actual del control de texto. Además, al cambiar la selección del desplegable, deberá actualizar el valor del texto, reflejando el valor de la opción seleccionada en el desplegable. Por último, al escribir sobre el control de texto, se deberá cambiar el valor del botón, para que este botón pase a mostrar el mismo texto el control de texto.

Ejercicio 7. Genere mediante JavaScript, directamente al arrancar la web, una página con una lista numerada, conteniendo cada elemento de la lista el cuadrado del elemento de lista correspondiente (primer elemento 1, segundo 4, tercero 9, etc. hasta llegar a 100.)

Ejercicio 8. Repita el ejercicio anterior pero para que no se lance al cargar la página sino en respuesta a la pulsación de un botón, devolviendo el resultado en una nueva página en blanco. Si no lo ha hecho en el ejercicio anterior, emplee en éste el mecanismo de interpolación de cadenas que vimos en clase.

Ejercicio 9. Adapte el ejercicio anterior para que en lugar de aparecer una lista con 100 elementos, primero pregunte al usuario cuántos quiere generar (entre 1 y 1000) y después proceda a generarlos.

Ejercicio 10. Modifique el ejemplo anterior para que en lugar de en una hoja aparte, el resultado aparezca en un contenedor debajo del botón.

Ejercicio 11. Defina en Javascript una función que reciba como parámetro de entrada de la función un número, y dependiendo de si el valor del mismo es 1, 2 o 3, haga las

acciones saludar en mensaje emergente, imprimir por consola o mostrar un mensaje en un determinado elemento a partir de su id, respectivamente. En caso de que el valor recibido no sea ninguno de los tres, deberá imprimir a la salida, en una nueva página. Haga varias versiones de esta función, empleando distintas estructuras selectivas (por ejemplo, if-else anidado o switch). Pruebe las funciones en una página html, asociando un manejador al evento de pulsar en un botón, de modo que se pida al usuario que proporcione una opción y que se llame a la función creada anteriormente pasándole el valor recibido del usuario.

Ejercicio 12. Cree un nuevo archivo JavaScript en el que almacenar la primera función solicitada en el ejercicio anterior. Crear una nueva página que importe el archivo JS anterior. En este nueva página, cree un control de formulario con un desplegable que muestre las opciones 1, 2, 3 y 4 y tal que, cada vez que modifiquemos la opción seleccionada, lance la acción correspondiente, haciendo uso de la función creada anteriormente. Además, salvo que la opción seleccionada sea la 4, deberá mostrarse junto al desplegable, un texto que indique la opción que se ha seleccionado. Por último, asegúrese de que la primera carga de la página tenga seleccionada la opción 1 y que llame inicialmente a esa acción.

Ejercicio 13. Cree una nueva página similar a la anterior pero que, en lugar de tener un desplegable de opciones, tenga 4 elementos div dentro de un contenedor con grid, cada elemento conteniendo un número del 1 al 4, y de modo que, cuando pasemos el ratón por encima del div correspondiente, se lance la acción correspondiente a su número. Nota: si pasamos a un manejador de evento la palabra reservada this, estaremos pasando el elemento HTML sobre el que se está disparando el evento.

Ejercicio 14. Empleando los mismos mecanismos de los ejercicios anteriores y los tipos de datos vistos en la clase anterior, y empleando controles de formulario y botones, desarrolle una calculadora que mantenga en todo momento el total calculado y vaya realizando sobre dicho total operaciones de distintos tipos (al menos suma, resta, multiplicación y división). Puede intentar el ejercicio proporcionando los números como botones, como controles de entrada de texto o mediante algún otro mecanismo que se le ocurra.

Ejercicio 15. Cree una página con un control numérico que acepte valores del 0 al 9 y tal que, cada vez que cambie el número introducido, genere en la misma página una tabla HTML con la tabla de multiplicar que corresponda al número introducido.

Ejercicio 16. En una nueva página, cree con JavaScript un array con una lista de tareas pendientes, dadas como cadenas de texto (por ejemplo, `"Hacer la compra"`, `"Preparar clase"` y `"Escribir artículo"`). Posteriormente, recorra el array, e imprima por consola una línea con cada tarea. Pruebe las siguientes alternativas que hemos visto en clase: bucle `for` "clásico", `for-in` y `for-of`.

Ejercicio 17. Cree una versión de la página anterior que, en lugar de imprimir por consola, genere la lista de tareas dentro de una lista HTML ordenada. Además, pruebe una variante, aparte de los bucles anteriores, empleando el método `forEach` para arrays. Para la función a pasar a `forEach`, pruebe a definir dicha función aparte, con su nombre, a continuación pruebe con función anónima con sintaxis `function` y finalmente empleando la notación de flecha.

Ejercicio 18. Adapte en una nueva página el ejercicio anterior para que el procesamiento no se produzca sin más durante la carga del script sino en una nueva función, `procesaArray`, que reciba como parámetro un array cualquiera, y recorra el mismo

generando un párrafo para cada elemento, conteniendo la posición y el contenido del elemento. Por ejemplo, para un array `["Hola", "buenas", "tardes"]` debería tener un primer párrafo conteniendo 0 - Hola, un segundo con 1 - buenas, etc. Asegúrese de probar distintas versiones de la función empleando estructuras de programación imperativa (bucles de los distintos tipos vistos en clase, o métodos de orden superior para arrays, como `forEach`, etc.) Haga pruebas con distintos tipos de arrays recibidos, pudiendo contener cadenas, números, objetos, etc.

Ejercicio 19. Defina en Javascript una nueva función que reciba como parámetro de entrada un array y devuelva también un array. Seleccione solamente aquellos elementos del array original que cumplan una cierta condición, y realice alguna operación sobre dichos elementos. Haga varias versiones de la función, al menos una empleando instrucciones de programación imperativa (instrucciones selectivas y repetitivas clásicas) y otra con programación funcional (funciones como `map`, `filter`, etc.)

Ejercicio 20. Retomando el ejemplo sobre la lista de tareas, incluya en una nueva página un campo de texto y un botón para incorporar nuevas tareas en el array de tareas y posteriormente refrescar la lista HTML ordenada de tareas, para que incluya en cada momento todas las tareas incluidas en el array.

Ejercicio 21. En el ejemplo anterior, incorpore tres nuevos botones, uno para añadir al principio, otro para eliminar la primera tarea del array y otro para eliminar la última, de forma que para estos dos últimos botones se devuelva en un mensaje emergente el elemento eliminado, y en los tres casos se actualice la lista ordenada para reflejar cómo ha quedado el array.

Ejercicio 22. Cree una nueva copia de la página anterior, y aplique los cambios que se van proponiendo a continuación:

- Cambie el array de tareas para que, en lugar de un array de cadenas, almacene ahora un array de objetos, cada uno de ellos conteniendo los campos "id", "nombre", "seleccionada" y "terminada". Contendría algo del tipo `[{id:1, nombre:"Hacer la compra", seleccionada:true, terminada:false}]`. Adapte el código que actualiza la lista ordenada de tareas para que, en lugar de cada objeto tarea, se imprima únicamente el nombre de cada una, y no los 4 campos de la misma. Por simplificar el ejercicio, que ya tiene muchos otros aspectos en los que centrarnos, vamos a dejar atrás las funciones que añaden o eliminan elementos del array de tareas, centrémonos en lo que se pide en este ejercicio.

- Cree de forma estática (en el HTML), debajo de la lista de tareas, un nuevo campo de entrada de texto para poder buscar sobre la lista de tareas anterior. Cree además un elemento de cabecera de segundo nivel que indique algo así como "Tareas seleccionadas", y tras ella una tabla HTML que contenga una sección de cabecera con su fila con celdas resaltadas "id", "nombre" y "terminada", y un cuerpo de la tabla con un "id" asociado a la etiqueta, para poder acceder posteriormente a ella.

- Cree una nueva función JavaScript que tome el valor del campo anterior, recorra el array de tareas y actualice el campo "seleccionada" de cada tarea cuyo nombre incluya el texto del campo de entrada anterior.

- Cree otra función que reciba un array de objetos tarea y devuelva aquellas tareas del array que hayan sido seleccionadas (haciendo uso del campo correspondiente). Mo-

difique la función del apartado anterior para que llame a esta nueva función creada, obteniendo así el array filtrado. Posteriormente, deberá recorrer este nuevo array y, para cada objeto del mismo, deberá añadir una fila a la tabla creada anteriormente. Antes de empezar a recorrer las tareas seleccionadas, recuerde vaciar las filas de la tabla (en caso contrario, si lanzamos dos veces seguidas esta funcionalidad, estaremos añadiendo varias veces las filas pero no eliminando nunca.)

- Observará que hemos creado una función que actualiza las tareas para indicar las seleccionadas, y que luego llama a la otra función para filtrar y actualizar la tabla de tareas. Sin embargo, no hay ningún evento que esté llamando a la función anterior. Añada en este apartado un manejador al control de texto creado para filtrar de modo que, al soltar una tecla presionada del teclado (evento denominado "keyup") se lance la función correspondiente, de modo que la tabla siempre refleje las tareas seleccionadas.

Ejercicio 23. Realice los siguientes ejercicios introductorios sobre manejo del DOM (prescindiendo de `document.getElementById` y de la propiedad `innerHTML`, que seguramente ya se han empleado en ejercicios anteriores):

- Cree una página sencilla que, en el momento de cargarla, solicite al usuario en elemento emergente su nombre y haga lo propio con otro elemento emergente para sus apellidos, y a partir de estos cree los nodos del DOM necesarios para generar una cabecera de primer nivel con el nombre recibido y otra de segundo nivel para los apellidos.

- Añada a la anterior, o a cualquier otra página, de forma estática, dos párrafos con un texto de 3 palabras, generados automáticamente (con Emmet). Incluye estos párrafos dentro de un contenedor con un id. A continuación, cree un botón tal que, al pulsar sobre él, cambie la propiedad `display` de los párrafos contenidos en ese contenedor para que sean elementos de línea en lugar de bloque, empleando la propiedad `style` de los elementos afectados. Asigne el manejador del evento en Javascript, no de forma estática en la etiqueta HTML. Nota: recuerde que algunas funciones devuelven una lista de nodos en lugar de simplemente un nodo, y que se puede facilitar el acceso sistemático a cada nodo pasando esta lista a un array, con la función `Array.from(miListaDeNodos);`. No obstante, a día de hoy una lista de nodos ya es iterable, así como accesible a través de indexado posicional, no requiriendo la conversión a un array.

- Cree una variante más simple del apartado anterior, basada en asignar o desasignar una cierta clase, con el nombre que quiera e inicialmente ausente del HTML, al contenedor de los párrafos creado. Defina en CSS una regla que asigne a todo párrafo hijo de la clase elegida la propiedad `display` al valor elemento de línea. Cree un botón "Alternar display" tal que, al pulsar sobre él, asigne la clase anterior al contenedor de párrafos en caso de que no la tenga y que, en caso de tenerla, la elimine del elemento.

Ejercicio 24. Adapte el ejercicio final sobre el array de tareas para emplear los nuevos mecanismos que hemos visto en clase para acceder a elementos del DOM y para reemplazar la asignación explícita de manejadores de evento en la propia etiqueta HTML por

la asignación en JavaScript, probando las opciones de asignación directa de función a propiedad y de añadir *listener* de evento.

Ejercicio 25. En una nueva página, crear en html un esqueleto básico con dos columnas (ayudándose de CSS Grid o de Bootstrap) del mismo tamaño (la mitad del espacio cada una). Incluya en la columna de la izquierda una cabecera de primer nivel, seguido de una serie de párrafos enmarcados dentro de un contenedor de bloque. En la columna de la derecha, cree un formulario HTML con controles que permitan actuar sobre el DOM con la funcionalidad que se describe a continuación. Posteriormente, empleando el mismo tipo de elementos comentados en el ejercicio anterior (acceso a DOM y asignación de manejadores de eventos), agregar funcionalidades que permitan lo siguiente:

- Un control de texto en el formulario tal que, cada vez que dejemos de presionar una tecla del teclado, se actualice el texto contenido en el elemento `<h1>` con el texto indicado en el control de texto.
- Un control de número para indicar el número de nuevos párrafos a mostrar en la columna izquierda de la página, de forma que, cada vez que se pulse en un nuevo botón, se añadan a los párrafos existentes tantos párrafos como indique este número, siendo todos ellos copia del primer párrafo existente. Incorporar unos botones de radio con opciones "inicio" y "fin", de modo que cada nuevo párrafo se añada al principio o al final del contenedor según si se ha marcado una opción de radio o la otra. Asegúrese de que la opción marcada por defecto es "fin";
- Un control desplegable con opciones de "inicio" y "fin", y un nuevo botón "Eliminar párrafo", que elimine el primer o el último párrafo del contenedor. Por defecto debe estar seleccionada la opción "inicio";

Ejercicio 26. En una página existente con controles de formulario, o en una nueva página que cree con algunos de estos, desarrolle código JavaScript que haga lo siguiente al cargar la página:

- Devolver en un mensaje emergente el número total de elementos de tipo input.
- Devolver por consola el número total de elementos de tipo input, select y textarea.
- Devolver debajo del formulario (fuera del mismo, en el cuerpo del documento) una lista con los nombres de todos los tipos de input encontrados (atributo `type` de los inputs).
- Empleando el tipo de datos `Set` que existe en JavaScript (ver este enlace para más info.), adaptar el apartado anterior para que el listado anterior no saque repetidos. Además, añada algo en su tratamiento para que, en caso de estar mostrando un tipo nulo, en lugar de aparecer un item vacío se muestre el texto "list".
- Empleando propiedades relacionadas con la navegación a través de los elementos HTML del DOM, asignar al formulario un manejador de evento para que, cada vez que pasemos por encima del mismo, se recorran todos los elementos hijos del mismo y se imprima por consola el nombre del nodo y su id en caso de tenerlo. Además, se deberá mostrar en un mensaje emergente el nombre del nodo padre del formulario, junto con el nombre de nodo de su hermano previo y posterior, en caso de que existan.

Ejercicios de repaso

Ejercicio 27. [*Asociación dinámica de manejadores de eventos*]
Cree una pequeña web que contenga lo siguiente mediante código estático:

- Un formulario con los siguientes controles:
  - Un *input* de tipo botón.
  - Un *select* con dos opciones, con valores 1 y 2 y textos "Primero" y "Segundo".
- Un párrafo con un texto generado automáticamente.

Genere lo siguiente mediante código dinámico en cliente:

- Un manejador de evento, generado dinámicamente, asociado a la pulsación del botón, que muestre en un mensaje emergente el texto del párrafo.
- Otro manejador asociado al cambio en la opción seleccionada en el desplegable anterior, de modo que, si la opción actual del mismo es la 1, imprima por consola el texto asociado a la opción, y si la opción es la 2, lo muestre en mensaje emergente.
- Un manejador tal que, al pasar por encima del párrafo, indique en la consola que estamos sobre el párrafo.

Ejercicio 28. [*Acceso al elemento que disparó un evento*]
Cree una copia de la web del ejercicio anterior, excluyendo la dinámica. A continuación, genere lo siguiente mediante código dinámico en cliente:

- Cree una función JS que reciba como parámetro un evento y muestre un mensaje emergente con el código completo de la etiqueta del elemento sobre el que se ha lanzado el evento.
- Asocie como manejador a la pulsación del botón la función creada en el apartado anterior.
- Repita el mismo proceso del apartado anterior para manejar los eventos de cambio del desplegable y de paso sobre el párrafo.

Ejercicio 29. [*Selección múltiple de elementos y recorrido*]
Cree una web con una serie de párrafos generados automáticamente. Ponga a varios de ellos una cierta clase. A continuación, añada lo siguiente:

- Copie en su código JS la misma función del primer apartado del ejercicio anterior.
- Seleccione mediante JS todos los párrafos que tengan la clase creada.
- Recorra con algún tipo de bucle los elementos anteriores. Dentro del bucle, asocie a cada uno de los elementos anteriores, como manejador del evento de pulsar sobre el párrafo en cuestión, la función creada en el primer apartado.
- Repita el mismo proceso del apartado anterior mediante una función de orden superior.

Ejercicio 30. [*Acceso a elementos de formulario*]
Cree una web con un formulario, asígnele un *name*, e incluya en el formulario elementos de entrada de texto, número, casillas de verificación sobre distintas aficiones, grupo de botones de radio, área de texto y elementos desplegables de selección simple y múltiple. Añada un botón tal que, al pulsar sobre él, haga lo siguiente:

- Seleccionar cada elemento de entrada de texto o de número e imprimir por consola su nombre y su valor.
- Seleccionar cada casilla de verificación, comprobar las que están marcadas y devolverlas por mensaje emergente. Indicar por consola cuáles no están marcadas.
- Seleccionar las opciones del elemento de selección desplegable e imprimir por consola el texto mostrado en cada una (no su valor).

Ejercicio 31. [*Acceso a elementos de formulario*]
Dado el mismo ejemplo del ejercicio anterior, añada lo siguiente a la funcionalidad del botón:

- Pedir al usuario un texto en un mensaje emergente que indique lo que quiere mostrar en el área de texto. Modificar dicha área para que muestre el texto indicado por el usuario. Pedir confirmación al usuario antes de modificar el texto.
- Indicar por consola qué valor del grupo de botones de radio está seleccionada.
- Mostrar en mensaje emergente la opción seleccionada en el elemento de selección simple.

Ejercicio 32. [*Modificación de elementos de estilo*]
Cree una web con párrafos de texto o tómela de otro ejercicio. Desarrolle el siguiente código dinámico:

- Cree una función "ocultarParrafo", que reciba como parámetro un párrafo y lo oculte empleando alguna propiedad de su estilo.
- Asocie a algún párrafo de la web un manejador de evento de modo que, al pulsar sobre él, llame a la función anterior pasándole como argumento el elemento sobre el que se ha pulsado.
- Cree otra función que seleccione el último párrafo de la página y llame a la función anterior pasándole el párrafo seleccionado.
- Mediante un nuevo botón en su web, llame a la función creada en el apartado anterior.

Ejercicio 33. [*Modificación de elementos de estilo*]
Partiendo de la web del ejercicio anterior, añada un botón que haga lo siguiente, al pulsar sobre él:

- Acceda al primer párrafo de su página.

- Si el mismo tiene un borde rojo, debe cambiarlo por uno verde. Si tiene un borde verde, debe eliminarlo.
- Si no tiene borde, debe preguntar al usuario si quiere un borde, y si es así deberá crear un borde rojo, de la forma que desee.

Ejercicio 34. [*Modificación de estilo mediante clases*]
Repita el ejercicio anterior, pero trabajando con funciones de manipulación de clases en lugar de modificar directamente elementos de estilo.

Ejercicio 35. [*Comprobación de estilo mediante clases*]
Cree una página con diversos párrafos con las clases "a", "b", ambas o ninguna. Crear una función JS que accede a todos los párrafos de su página y los vaya recorriendo para obtener:

- El número de párrafos con la clase "a".
- El número de párrafos con la clase "b".
- El número de párrafos con ambas clases.

Llame a la función al cargar su página, y muestre por consola los números obtenidos por la función.

Ejercicio 36. [*Comprobación de estilo y actualización del DOM*]
Partiendo del resultado del ejercicio anterior, añada lo siguiente a la función creada:

- Inserción de tres etiquetas de salto de línea al final del cuerpo de la página, empleando una sola llamada a función para actuar sobre el DOM y sin necesidad de crear nodos.
- Creación de un nuevo nodo del DOM que sea de tipo párrafo.
- Creación de un nuevo nodo de texto que indique la información sobre la cantidad de párrafos obtenidos por la función.
- Adición del nodo de texto al nodo párrafo anterior.
- Inserción del nodo párrafo creado después del último salto de línea de la página.

Ejercicio 37. [*Atributos del DOM*]
Cree una página web con una lista no ordenada de enlaces, algunos de ellos que lleven a alguna dirección web y otros que no lleven a ningún lado. Añadir tras la lista un botón. Realizar mediante JS:

- Declarar una constante global **URL**, que indique la dirección que desee.
- Crear una función que recorra cada enlace de la página y haga lo siguiente:
  - Si no tiene el atributo **href**, se le asignará una url determinada la URL declarada anteriormente.
  - Si lo tiene, imprimirá por consola su valor.

- Asignar al botón un manejador que haga lo siguiente:
  - Pedir al usuario un número entre 0 y el número de enlaces que tenga menos 1.
  - Eliminar el atributo href del elemento indicado.

Ejercicio 38. [*Navegación por el DOM*]
Empleando la página HTML del ejercicio anterior como base, desarrollar mediante código JS:

- Una función que haga lo siguiente:
  - Acceder al primer elemento de tipo lista de la página.
  - Desde dicho elemento, acceder a sus hijos y recorrerlos. Para cada hijo (**li**), deberá:
    - Imprimir por consola el código del primer hijo de dicho hijo (deberá ser un enlace **a**) y el de su padre (deberá ser el **ul**).
    - Reemplazar el código del elemento por un texto que indique el número de hijo, comenzando por "Hijo 1" (deberá desaparecer el enlace).
- Un manejador de evento asociado a un nuevo botón en la página.

Ejercicio 39. [*Navegación por el DOM*]
Sobre alguna página con una serie de párrafos, incorporar un manejador de evento para que al pasar por encima de cada uno de ellos:

- Si el párrafo tiene un hermano previo, deberá imprimir por consola el contenido de dicho párrafo hermano.
- Si no lo tiene, deberá imprimir en su lugar el contenido del hermano posterior.