Para este bloque de ejercicios, es obligatorio un fichero externo con el script al margen del HTML.

1. **(Objetos: creación, recorrido y propiedades)** Vamos a simular un objeto llamado "alumno" con las siguientes propiedades:
   - edad: por defecto 18.
   - admin: por defecto false.
   - stats: un objeto con las propiedades:
     - posts: número de publicaciones (por defecto 0)
     - followers: número de seguidores (por defecto 0)

Ahora vamos a probar el objeto. Todos los apartados que pidan mostrar un resultado será en un párrafo de la página con id "output".

- Incrementa el número de posts en 2 y de followers en 3.
- Cambia el permiso de administrador a "true" y muestra el resultado en el párrafo.
- Quita la propiedad de administrador.
- Crea una función fuera del objeto que recibe un objeto como parámetro y recorre todas sus propiedades del objeto y muestra el resultado en el párrafo. Si se encuentra alguna propiedad que sea un objeto como "stats", recórrelo y muestra todas sus propiedades. Si alguna propiedad es del tipo "función" no se mostrará.

2. **(Objetos: clonación)** Partiendo del enunciado anterior, clona el objeto incluyendo las propiedades anidadas si fuese posible. Incrementa el número de posts en 1 y muestra los dos objetos en el párrafo para observar los cambios.

3. **(Objetos: referencia this y operador ?)** Seguimos con el objeto "alumno" de los ejercicios anteriores. El objeto tendrá el siguiente método:
   - incrementarStats(tipo, n): Suma "n" veces el número de posts o followers según el valor primer parámetro "tipo". Se realizarán las siguientes validaciones:
     - Se comprobará con el operador "?" si el objeto "stats" existe y en este caso intentaremos acceder a la propiedad recibida en el parámetro "tipo". Solo se realizará la suma si la variable "tipo" tiene los valores "posts" o "followers".
     - Se comprobará si el parámetro "n" es numérico.
     - Si no se cumplen las condiciones anteriores, se ignorará la suma sin mostrar mensaje de error.

Ahora el código de prueba que incrementa el objeto "stats" empleará el método anterior en lugar de modificar directamente la propiedad.

4. **(Objetos: función con new)** El objeto "alumno" en este punto contiene 3 propiedades y un método. Convierte el objeto a una función llamada "classAlumno" que se pueda instanciar con new. La variable con el objeto alumno ahora instanciará "classAlumno". Comprueba si el resto del código sigue funcionando.

5. **(Tipo Symbol con objetos)** Volvemos al ejercicio 3 con el objeto creado sin emplear funciones. Vamos a añadir dos propiedades de tipo “Symbol” en la definición del objeto “alumno” al principio del script:
   - Un Symbol específico con descripción “id”. La propiedad del objeto “alumno” asociada a este Symbol tendrá como valor un ID generado con un número aleatorio del 0 al 9999.
   - Un Symbol global con descripción “estado”. El valor de esta propiedad asociada al Symbol en el objeto “alumno” será inicialmente el String “activo”.

Tras clonar el objeto como se hacía en los objetos anteriores, se borrará la propiedad del “Symbol” específico con el ID en el nuevo objeto. Se creará otra variable “Symbol” que reemplaza la anterior con un nuevo ID autogenerado. Así garantizamos que la propiedad es única y que también se genera otro valor (no hace falta comprobar si genera un número distinto).

Finalmente, vamos a modificar nuestra función que muestra el objeto para que también se visualicen las propiedades con “Symbol” (no se hace en el for.. in por defecto). Se mostrará la descripción del símbolo y el valor de la propiedad del objeto asociado (por ejemplo, id: 2425).

6. **(Arrays: métodos)** Vamos a crear un array que gestione una lista con las frutas manzana, pera y plátano. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id “output”. Se realizarán las siguientes modificaciones en el array.
   - Reemplaza el segundo elemento por "sandía". Muestra el array.
   - Crea una copia de las frutas desde la posición 1 hasta la 3 (sin incluir 3). Muestra el array con la copia.
   - Crea un nuevo array concatenando la lista actual con los elementos uvas y mango. Muestra el nuevo array.

Continuamos en el array original ignorando las copias creadas:
- Busca el índice de "sandía" en el array original y muestra el resultado.
- Muestra si el array contiene "pera".
- Muestra la primera fruta que tenga más de 5 letras.
- Crea un array con todas las frutas que empiecen con la letra "m" y muestra el resultado.
- Muestra el índice de la primera fruta que tenga la letra "a".

7. **(Arrays: métodos)** Partimos de la cadena de texto "rojo,verde,azul,amarillo" con varios colores separados entre comas. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id “output”. Se realizarán los siguientes apartados.
   - Convierte la cadena en un array. Comprobamos si es un array con el método correspondiente. Si es un array se muestra por pantalla en nuestro párrafo, en caso contrario un mensaje de error y el script no continuará. Si está todo correcto continuamos de la siguiente manera:
     - Recorre el array de la forma más eficiente posible que permita mostrar por pantalla cada color en mayúsculas.

- Crea un nuevo array con objetos que incluya el nombre de cada color y la longitud de la palabra (por ejemplo "color {nombre: rojo, longitud: 4}"). Muestra el resultado por pantalla
- Volvemos al array inicial con el texto de cada color. Ordenar los colores alfabéticamente. Muestra el resultado por pantalla.
- Ahora ordena los valores en orden alfabético inverso. Muestra el resultado.
- Calcula la suma total de letras de todos los colores del array inicial. Muestra el resultado.
- Convierte el array de nuevo en una cadena, ahora separada por " | ". Muestra la cadena por pantalla.

8. **(Arrays: recorrer y conversiones)** Vamos a crear un script de gestión de calificaciones. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id “output”. Los pasos a seguir serán los siguientes:
   - Crea dos arrays con 4 calificaciones numéricas cada uno. Se crearán con Array.from a partir de números del 1 al 10 generados al azar.
   - Ahora creamos una función con parámetros variables. Le pasamos el array con el operador “…”. La función recorre todos los parámetros y los muestra por pantalla. Vamos a llamar la función dos veces con los arrays creados anteriormente para mostrarlos por pantalla.
   - Ahora combinamos los dos arrays con el operador “…”. Recorreremos el array y se calcula el promedio de todas las notas combinadas. Muéstralo por pantalla.
   - En este ejercicio hay que usar “for of" al menos una vez.

9. **(Map, Set, WeakMap y WeakSet).** Vamos a gestionar un catálogo de películas. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id “output”. El script tendrá la siguiente funcionalidad.
   - Crea un Map llamado “películas”, donde la clave sea un Symbol con descripción de título y el valor sea un objeto con: título, año, director, genero. Se realizarán las siguientes acciones.
     - Crea 3 objetos con películas de prueba y añádelas al Map. Puedes crearlas manualmente con el script (normalmente los objetos se reciben en formato JSON que se estudiará posteriormente).
     - Muestra el listado completo recorriendo los valores del Map con for...of.
   - Crea un Set con los “géneros”. Todos los géneros del Map se exportarán a un array que se empleará para crear el “Set”. Muestra el “Set” por pantalla.
   - Crea un WeakMap con información privada. Las claves serán los objetos con películas de prueba que se crearon anteriormente y el valor una cadena de texto con información confidencial, por ejemplo copias disponibles.
   - Crea un WeakSet para gestionar las películas en cartelera. Añade alguno de los objetos anteriores.
   - Asigna null a uno de los objetos con películas que pertenezcan al Map, WeakMap y WeakSet. Muestra por pantalla los valores del Map y también comprueba si existe la película en el WeakMap y WeakSet.
   - Ahora elimina la película con la clave a la que asignaste “null” del Map. Muestra los valores del Map y observa las diferencias.

10. **(Conversiones entre objetos y asignación desestructurante).** Seguimos con el mismo Map de "películas" que el ejercicio anterior. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id "output". El script tendrá la siguiente funcionalidad:
    - Vamos a convertir el Map de la siguiente manera. La clave ya no será un "Symbol" y se empleará la propiedad con el título. El valor será un objeto con el resto de propiedades excepto el título. Debes emplear asignación desestructurante para separar la clave y el resto de valores del objeto.
    - Muestra el Map anterior por pantalla. Para recorrerlo se empleará for..of directamente a partir del método entries del mapa y asignación desestructurante.
    - Crea un array con las películas más antiguas que el año 2000. El array solo contiene el objeto con el valor sin incluir la clave.
    - Si el array obtenido anteriormente no está vacío, escoge el primer resultado. Crea variables con el mismo nombre que la descripción de cada valor del objeto (título, año, director, genero). Asocia las variables al objeto mediante asignación desestructurante y muestra los resultados por pantalla.

11. **(Conversión de objetos a datos primitivos y con JSON).** Se estudiará en Unidad 7.

12. **(Clases en JavaScript).** Crear una clase "Vehiculo" con algunas propiedades, un constructor y métodos de instancia. Cada vez que se pida mostrar un resultado será en un párrafo de la página con id "output". Se pedirán los siguientes apartados progresivamente:
    a) Define una clase "Vehiculo" con:
        - Propiedades: marca, modelo, año.
        - Un método mostrarInfo() que devuelva una cadena con las propiedades definidas anteriormente.
        - Un método calcularAntiguedad() que indica cuantos años tiene el vehículo teniendo en cuenta la fecha actual y la propiedad correspondiente.
        - El script creará tres objetos "Vehiculo" y se mostrará por pantalla el resultado de los métodos "mostrarInfo" y "calcularAntiguedad" por cada uno de ellos.
    b) Define una clase "Coche" que herede de "Vehiculo" con la siguiente información:
        - Añade propiedades nuevas: puertas y combustible.
        - Sobrescribe el método mostrarInfo() para que incluya las nuevas propiedades.
        - Crea una instancia de Coche y muestra toda su información y antiguedad.
        - Elige un vehículo del apartado anterior y compara por separado si ambos pertenecen a la clase "Vehiculo" y también a la clase "Coche".
    c) Ahora vamos a incorporar nuevas características a la clase "Vehiculo":
        - Añade una propiedad privada kilometros en Vehiculo. Crea métodos getter y setter para modificarla. Esta propiedad no estará en el constructor ni en el método mostrarInfo().
        - Añade una propiedad estática totalVehiculos para contar cuántos vehículos se han creado. Cada vez que se crea un vehículo se sumará desde el constructor.
        - Añade un método estático mostrarTotalVehiculos que devuelva ese número.

- Hasta ahora en el script se está mostrando la información y antigüedad. Añade kilómetros a alguno de los vehículos que has creado y muestra también los kilómetros de cada coche.
- Al final de script se indicarán cuántos vehículos se han creado, ya sean vehículos con la clase base o coches.

13. **(Prototipos y descriptores de propiedad).** Partimos del ejercicio anterior. Vamos a añadir propiedades a objetos de la clase una vez creada. Los resultados se seguirán mostrando por pantalla en un párrafo de la página con id "output".
    a) Vamos a modificar algunos objetos de la siguiente forma:
        - Elige uno de los tres objetos vehículo y añade una propiedad color con un valor con la notación `objeto.color="valor"`. Elige otro objeto vehículo y añade esta misma propiedad con el método "defineProperty" y otro valor. Muestra por pantalla la propiedad "color" en los tres vehículos y observa las diferencias.
        - Ahora añade la propiedad color con un valor por defecto al prototipo de la clase "Vehiculo". IMPORTANTE: Haz que esta propiedad sea "configurable". Muestra por pantalla los tres vehículos.
    b) Ahora vamos a hacer algunas comprobaciones para entender mejor cómo funcionan los objetos y prototipos. Hemos añadido la propiedad "color" al objeto de dos maneras diferentes y luego al prototipo.
        - Muestra si la propiedad "color" existe en los tres objetos de tipo "Vehiculo" con getOwnPropertyDescriptor. En caso afirmativo muestra si es enumerable y el valor de la propiedad. Si no se encuentra la propiedad "enumerable" muestra directamente el texto "No se encuentra".
        - En este punto, la propiedad solo debería ser enumerable en el objeto que se definió explícitamente con `objeto.color="valor"`.
        - Modifica el prototipo y redefine la propiedad "color" para que sea enumerable y con el mismo valor por defecto que había anteriormente. Si da error, explica el motivo en el código y modifica la propiedad enumerable en el objeto en el que no se definió la propiedad color "explícitamente".
        - Recorre los tres objetos con "for..in" y muestra la propiedad color en aquellos casos que exista. Si no existe indica "No es enumerable".
        - Finalmente, convierte la propiedad "color" para que sea solo lectura en el prototipo. Intenta modificar esta propiedad en el objeto que creaste para leer la propiedad "color" del prototipo. Cambia a "modo strict" y observa las diferencias.

14. **(Herencia múltiple y mixins).** Partimos del ejercicio anterior. Vamos a crear un mixin con el método "calcularCombustible". Recibe una cantidad litros. Si es "gasolina" devuelve la cantidad multiplicada por 1.5€ y si es "diesel" por 2€. En caso contrario muestra un alert con un mensaje de error. Añade este método a la clase "Coche" y prueba el resultado.

15. **(Módulos).** Partimos del ejercicio anterior. Vamos a organizar el código en módulos. Se crearán dos módulos:
    - "Vehículo". Solo exporta la clase vehículo con "default".

- “Coche”. Exporta la clase “Coche” con “default” y el mixin con otra exportación separada.

El código de prueba será el mismo aunque importando las clases y el mixin de los módulos que se han creado. El mixin se importará asignando el nombre “nuevoMetodo”.