PRÁCTICA 5.01 Asincronismo básico

Normas de entrega

- En cuanto al **código**:
  - en la **presentación interna**, importan los **comentarios**, la claridad del código, la significación de los nombres elegidos; todo esto debe permitir considerar al programa como **autodocumentado**. No será necesario explicar que es un `if` un `for` pero sí su funcionalidad. Hay que comentar las cosas destacadas y, sobre todo, las **funciones** y **clases** empleadas. La ausencia de comentarios será penalizada,
  - en la **presentación externa**, importan las leyendas aclaratorias, información en pantalla y avisos de ejecución que favorezcan el uso de la aplicación,
  - todo el código debe estar situado dentro del evento `window.onload = () => {};` o a través del evento `document.addEventListener("DOMContentLoaded", () => {});` ,
  - si no se especifica lo contrario, la información resultante del programa debe aparecer en la consola del navegador `console.log(información)`,
  - los ejercicios deben realizarse usando ***JavaScript* ES6** y usar el modo estricto (`use strict`) No se podrá utilizar *jQuery* ni cualquier otra biblioteca (si no se especifica lo contrario en el enunciado),
  - para el nombre de **variables**, **constantes** y **funciones** se utilizará *lowerCamelCase,*
  - para la asignación de eventos se utilizará `addEventListener()` indicando sus tres parámetros en su definición,
  - debes dividir tu código en **bibliotecas temáticas** de funciones y/o clases a partir de este ejercicio,
  - se usarán las funcionalidades `import` y `export` para crear bibliotecas de funciones temáticas a partir de esta práctica,
  - todo el código que sea susceptible de retrasar su ejecución debe escribirse de forma asíncrona utilizando asincronismo,
  - todas las funciones asíncronas deben tener control de errores.
- En cuanto a la **entrega** de los archivos que componen los ejercicios:
  - todos los ejercicios en **una carpeta** (creando las **subcarpetas** necesarias para documentación anexa como imágenes o estilos) cuyo nombre queda a discreción del discente,
  - el nombre de los ficheros necesarios para resolver el ejercicio será el número de ejercicio que contenga,
  - el código contendrá ejemplos de ejecución, si procede, y
  - la carpeta será **comprimida** en formato `ZIP` y será subida a **Aules** de forma puntual.

NOTA: todas estas acciones se realizarán al cargar la página, es decir, tras el evento `window.load();` .

Ejercicio 1 - Promesa

Crea una `promise` simple que se resuelva en un tiempo aleatorio de 1 a tres segundos de lanzarla mostrando el resultado por consola (debidamente formateado). Esto simula un retraso en el acceso a los datos. La promesa generará un numero aleatorio entre 1 y 100 y se resolverá si ese valor es par y se rechazará si el valor es impar.

Ejercicio 2 - Feos

Muestra el contenido del fichero `feos.json` en un `<div>` ordenado por nombre alfabéticamente de forma ascendente. Simula un retraso de 3 segundos en la carga. El listado debe estar, como no podría ser de otra manera, debidamente formateado. Utiliza fetch para la carga del fichero.

Ejercicio 3 - Gentuza

Utiliza la **API** de *Star Wars* (**https://swapi.dev/api/people**) para mostrar un listado de los personajes de todas las películas. La información que debe aparecer en el listado será el nombre, su peso (**en kilogramos**), su género, su color de pelo y su año de nacimiento debidamente formateada, eso sí. Usa fetch y catch con el fin de gestionar posibles errores en la conexión.