Actividad: Módulos en Javascript

Temario:

1. Módulos
    a. Qué son los Módulos?
    b. Exportar Módulos
    c. Importar Módulos

Ejercicios

1. Crear una función llamada **enRango** que reciba un número y determine si está en el rango entre 10 y 50 (inclusive). La función debe retornar:

- "Está en el rango" si el número está entre 10 y 50 (inclusive).
- "Fuera del rango" si el número no está en ese rango.

Uso de módulos:

- Exporta la función **enRango** en un archivo **utilidades.js.**
- En un archivo **main.js**, importa la función **enRango** y pruébala con diferentes números.

2. Escribir una función llamada **calcularDescuento** que reciba el precio de un producto y calcule el descuento aplicable de acuerdo a la siguiente lógica:

- Si el precio es mayor a 1000, el descuento es del 20%.
- Si el precio es entre 500 y 1000, el descuento es del 10%.
- Si el precio es menor a 500, no se aplica descuento.

La función debe retornar el precio final después de aplicar el descuento.

Uso de módulos:

- Exporta la función **calcularDescuento** en un archivo **utilidades.js**.
- En un archivo **main.js**, importa la función **calcularDescuento** y pruébala con diferentes precios.

3. Escriba una función llamada **categorialMC** que reciba el índice de masa corporal (IMC) de una persona y determine su categoría:
   - "Bajo peso" si el IMC es menor a 18.5.
   - "Normal" si el IMC está entre 18.5 y 24.9.
   - "Sobrepeso" si el IMC está entre 25 y 29.9.
   - "Obesidad" si el IMC es 30 o mayor.

Uso de módulos:

- Exporta la función **categorialMC** en un archivo **utilidades.js.**
- En un archivo **main.js**, importa la función **categorialMC** y pruébala con diferentes valores de IMC.

4. Escriba una función llamada **nivelRiesgo** que reciba dos parámetros: la edad de una persona y un valor booleano (true o false) que indique si la persona tiene enfermedades previas. La función debe retornar:
   - "Alto riesgo" si la persona tiene más de 60 años o tiene enfermedades previas.
   - "Riesgo moderado" si la persona tiene entre 40 y 60 años y tiene enfermedades previas.
   - "Bajo riesgo" en cualquier otro caso.

Uso de módulos:

- Exporta la función **nivelRiesgo** en un archivo **utilidades.js**.
- En un archivo **main.js**, importa la función **nivelRiesgo** y pruébala con diferentes combinaciones de edad y enfermedades.

5. Escriba una función llamada **esBisiesto** que reciba un año y determine si es bisiesto. Un año es bisiesto si:
   - Es divisible por 4, pero no es divisible por 100, a menos que también sea divisible por 400.

La función debe retornar:

- "Es bisiesto" si el año es bisiesto.
- "No es bisiesto" si el año no lo es.

Uso de módulos:

- Exporta la función **esBisiesto** en un archivo **utilidades.js**.
- En un archivo **main.js**, importa la función **esBisiesto** y pruébala con diferentes años.

6. Escribe una función llamada **esElegibleParaPrestamo** que reciba dos parámetros: el salario mensual de una persona y su puntaje de crédito. La función debe retornar:
   - "Elegible para préstamo" si el salario es mayor a 3000000 y el puntaje de crédito es mayor a 650.
   - "No elegible para préstamo" en cualquier otro caso.

Uso de módulos:

- Exporta la función **esElegibleParaPrestamo** en un archivo **utilidades.js.**
- En un archivo **main.js**, importa la función **esElegibleParaPrestamo** y pruébala con diferentes valores de salario y puntaje de crédito.

7. Crear un archivo en formato json llamado **libros.json** que contenga objetos de tipo libro que tengan los siguientes atributos:
   a. Titulo
   b. Autor
   c. Número páginas
   d. Editorial
   e. Idioma

   Hacer una función llamada **consultarLibrosPorPalabraClaveTitulo** que reciba como parámetro una palabra. La función debe retornar un arreglo con todos los libros que en su titulo contenga la palabra que recibe como parámetro.

   Hacer una función llamada **consultarLibrosPaginas** que crea un arreglo de objetos que contienen dos atributos así: titulo del libro y número de páginas de todos los libros existentes.

Uso de módulos:

- En un archivo **utilidades.js** importa el archivo **libros.json**
- Exporta la función **consultarLibrosPorPalabraClaveTitulo** en un archivo **utilidades.js.**
- Exporta la función **consultarLibrosPaginas** en un archivo **utilidades.js.**
- En un archivo **main.js**, importa las funciones **consultarLibrosPorPalabraClaveTitulo**, **consultarLibrosPaginas** pruébala con diferentes valores de salario y puntaje de crédito.

Nota: El archivo libros.json lo puede crear con la ayuda de chatgpt.

8. Proyecto Alcancia. En la alcancía es posible guardar monedas de las siguientes denominaciones: $200, $500 y de $1000. No se guardan ni billetes ni monedas de otras denominaciones. Al dueño de la alcancía le parece muy útil conocer cuánto tiene en la alcancía sin necesidad de romperla, es más, él quiere conocer cuántas monedas tiene en cada denominación para así romper la alcancía sólo cuando quiera disponer de todo su dinero ahorrado. Por lo anterior se necesita crear una aplicación web que le permita simular el comportamiento de la alcancía.

La aplicación debe permitir:

- Agregar una moneda de una de las denominaciones indicadas.
- Contar cuántas monedas tiene de cada denominación.
- Calcular el total de dinero ahorrado.
- Romper la alcancía vaciando su contenido.
- Comenzar una nueva alcancía.

La aplicación debe hacerse orientada a objetos. Para ello usted debe crear una clase llamada **Alcancia**, la cual tiene 3 atributos que representan la cantidad de monedas de cada una de las denominaciones. Adicionalmente la alcancía tiene que tener unos métodos o funciones que permiten realizar las tareas mencionadas anteriormente.

Uso de módulos

- Hacer un **export default** a la clase **Alcancia** en el archivo **alcancia.js**
- En un archivo **main.js** importar la clase **Alcancia**
- En el archivo **main.js** realizar todas las operaciones.
- Crear un archivo html para interactuar con el usuario el cual debe crear una etiqueta script para que incorpore el archivo **main.js** como **type=module**

9. Hacer una aplicación que permita gestionar sus contactos. Para ello debe crear una clase **Contacto** en un archivo llamado **contacto.js**. La clase debe tener los siguientes atributos:
    a. Identificación
    b. Nombre
    c. Apellido
    d. Correo
    e. Celular

Uso de módulos

- Hacer un **export default** a la clase **Contacto** en el archivo **contacto.js**
- En un archivo **main.js** importar la clase **Contacto**
- Crear un archivo llamado **app.html** con el código html de la interfaz anexa. Aquí se debe crear una etiqueta **script** de **type=module** llamando al **archivo main.js**
- En el archivo **main.js** se deben crear las funciones que le permita hacer las siguientes operaciones:
    - Agregar un contacto a la agenda. Se debe mostrar en la tabla inferior. No puede haber más de un contacto con la misma identificación
    - Consultar contacto por identificación.
    - Actualizar Contacto
    - Eliminar Contacto

Referencias:

1. Tutorial de Javascript Modernohttps://es.javascript.info/
2. Tutorial de w3schools: https://www.w3schools.com/js/
3. Curso en youtube: https://www.youtube.com/watch?v=Z34BF9PCfYg&t=106s
4. Curso internet: https://lenguajejs.com/javascript/

Fecha de Entrega: Junio 6 de 2025

Forma de Entrega: Solución de cada uno de los ejercicios en repositorio en github. Adicionalmente hacer un documento en formato pdf con la prueba de cada uno de los ejercicios.

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	29-05-2025