**Actividad Módulos en Javascript**

**Presentado a:** Instructor César Marino Cuéllar Chacón

**Por Aprendiz: Laura Sofia Agredo**

**Ficha:** 3064975

**Competencia:** Diseñar la solución de software de acuerdo con procedimientos y requisitos técnicos

**Resultado de Aprendizaje:** Verificar los entregables de la fase de diseño del software de acuerdo con lo establecido en el informe de análisis

Tecnólogo en Análisis y Desarrollo de Software

Servicio Nacional de Aprendizaje SENA

Centro de Teleinformática y Producción Industrial

Regional Cauca

Popayán, día **5** de **09** del año **2025**

**Tabla de Contenido**

[1. Actividad o Ejercicio1 3](#_Toc197272968)

[1.1 Enunciado 3](#_Toc197272969)

[1.2 Solución 3](#_Toc197272970)

[2. Actividad o Ejercicio2 4](#_Toc197272971)

[2.1 Enunciado 4](#_Toc197272972)

[2.2 Solución 4](#_Toc197272973)

[3. Bibliografía 5](#_Toc197272974)

# **Actividad o Ejercicio1**

# **Enunciado**

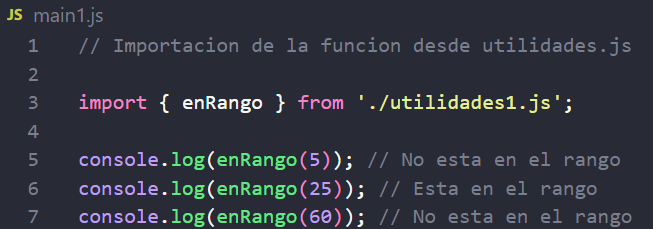
Crear una función llamada **enRango** que reciba un número y determine si está en el rango entre 10 y 50 (inclusive). La función debe retornar:

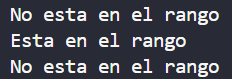
* "Está en el rango" si el número está entre 10 y 50 (inclusive).
* "Fuera del rango" si el número no está en ese rango.

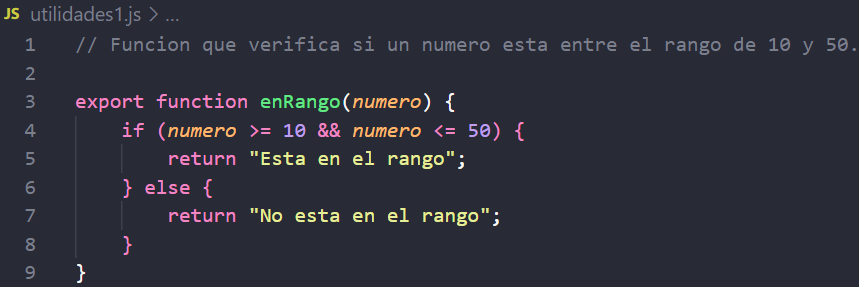
**Uso de módulos:**

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

# **Solución**





****

# **Actividad o Ejercicio2**

# **Enunciado**

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

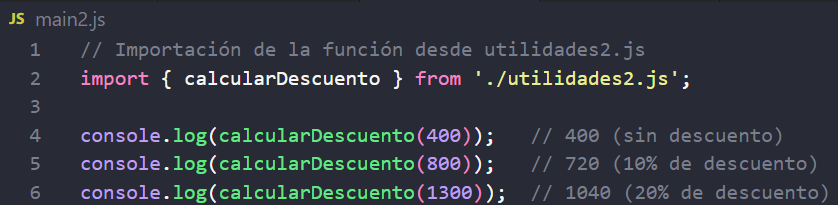
* Si el precio es mayor a 1000, el descuento es del 20%.
* Si el precio es entre 500 y 1000, el descuento es del 10%.
* Si el precio es menor a 500, no se aplica descuento.

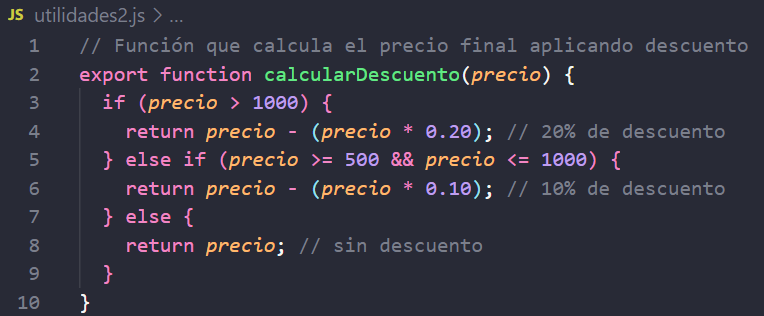
La función debe retornar el precio final después de aplicar el descuento.

**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.

# **Solución**





# **Actividad o Ejercicio3**

# **Enunciado**

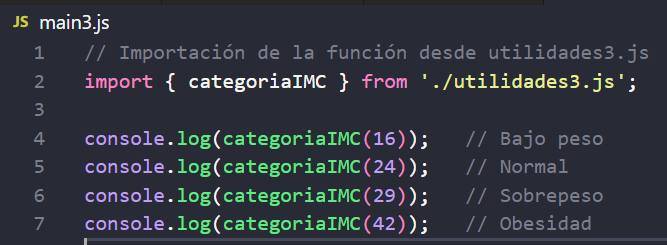
Escriba una función llamada **categoriaIMC** que reciba el índice de masa corporal (IMC) de una persona y determine su categoría:

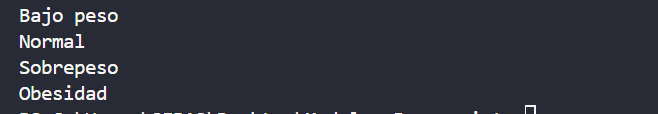
* "Bajo peso" si el IMC es menor a 18.5.
* "Normal" si el IMC está entre 18.5 y 24.9.
* "Sobrepeso" si el IMC está entre 25 y 29.9.
* "Obesidad" si el IMC es 30 o mayor.

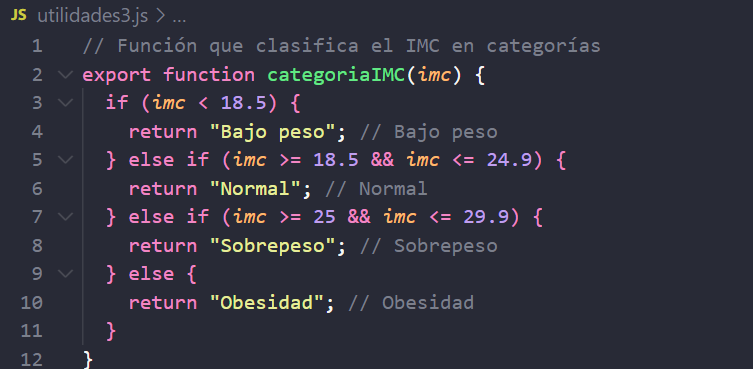
**Uso de módulos:**

* Exporta la función categoriaIMC en un archivo **utilidades3.js**.
* En un archivo **main3.js**, importa la función categoriaIMC y pruébala con diferentes valores de IMC.

# **Solución**







# **Actividad o Ejercicio4**

# **Enunciado**

Escriba una función llamada **nivelRiesgo** que reciba dos parámetros:

* la **edad** de una persona
* un valor **booleano** (true o false) que indique si la persona tiene enfermedades previas.

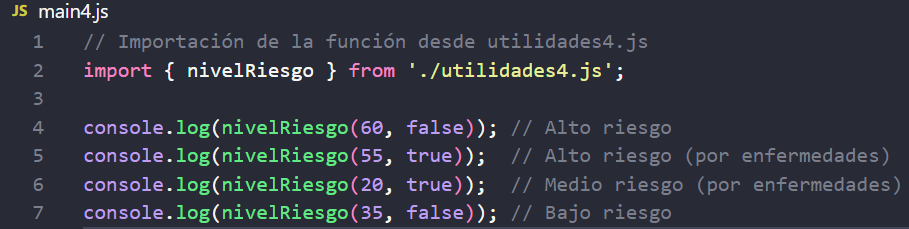
La función debe retornar:

* "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.

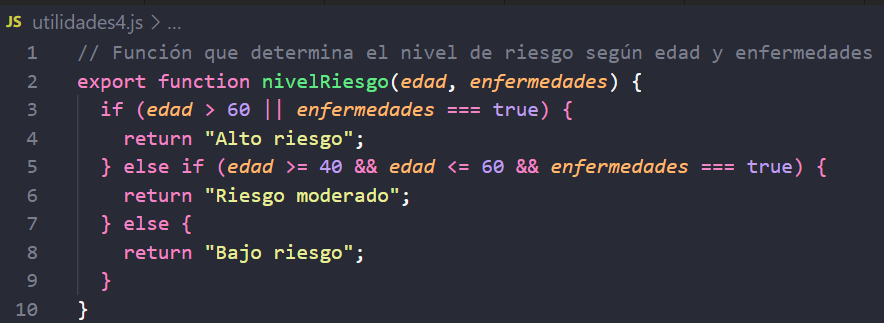
**Uso de módulos:**

* Exporta la función nivelRiesgo en un archivo **utilidades4.js**.
* En un archivo **main4.js**, importa la función nivelRiesgo y pruébala con diferentes combinaciones de edad y enfermedades.

# **Solución**







# **Actividad o Ejercicio5**

# **Enunciado**

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.

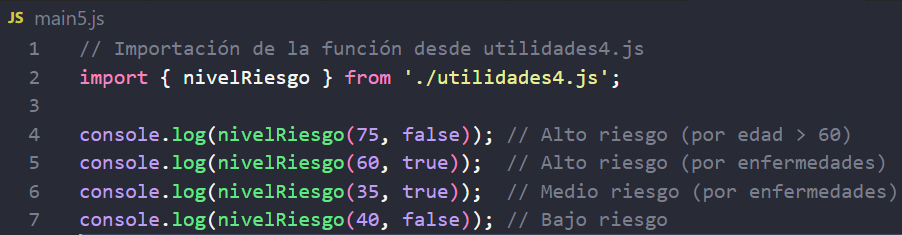
La función debe retornar:

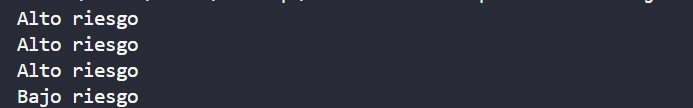
* "Es bisiesto" si el año es bisiesto.
* "No es bisiesto" si el año no lo es.

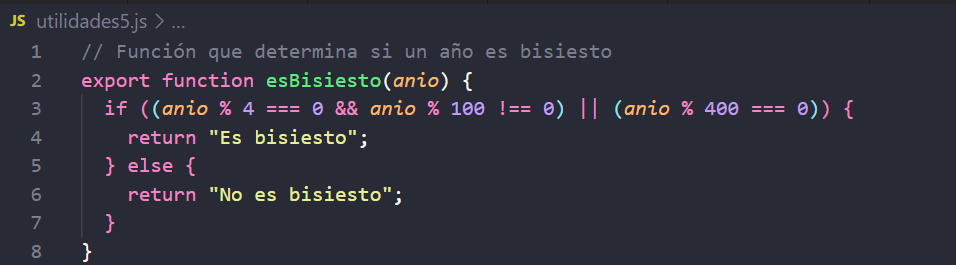
**Uso de módulos:**

* Exporta la función esBisiesto en un archivo **utilidades5.js**.
* En un archivo **main5.js**, importa la función esBisiesto y pruébala con diferentes años.

# **Solución**







# **Actividad o Ejercicio6**

# **Enunciado**

Escribe una función llamada **esElegibleParaPrestamo** que reciba dos parámetros:

* el **salario mensual** de una persona
* su **puntaje de crédito**

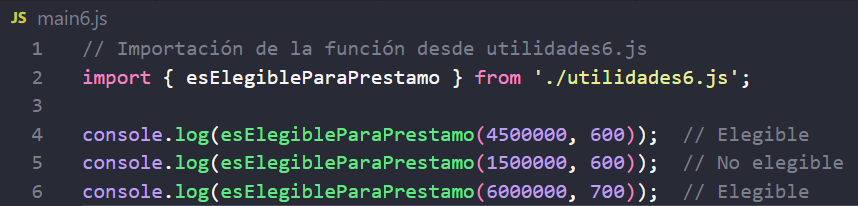
La función debe retornar:

* "Elegible para préstamo" si el salario es mayor a 3,000,000 y el puntaje de crédito es mayor a 650.
* "No elegible para préstamo" en cualquier otro caso.

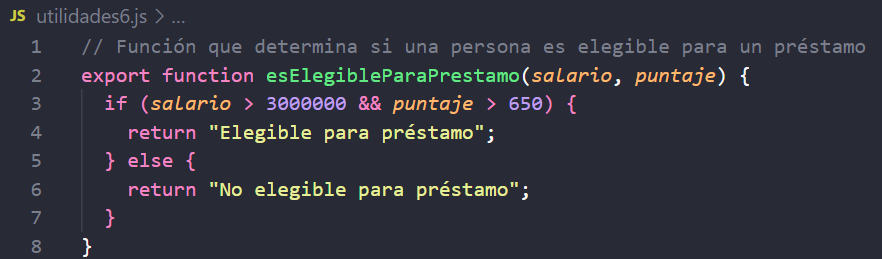
**Uso de módulos:**

* Exporta la función esElegibleParaPrestamo en un archivo **utilidades6.js**.
* En un archivo **main6.js**, importa la función esElegibleParaPrestamo y pruébala con diferentes valores de salario y puntaje de crédito.

# **Solución**







# **Actividad o Ejercicio7**

# **Enunciado**

Crear un archivo en formato **JSON** llamado libros.json que contenga objetos de tipo libro con los siguientes atributos:

* Título
* Autor
* Número de páginas
* Editorial
* Idioma

Hacer una función llamada **consultarLibrosPorPalabraClaveTitulo** que reciba como parámetro una palabra.

* La función debe retornar un arreglo con todos los libros cuyo título contenga esa palabra.

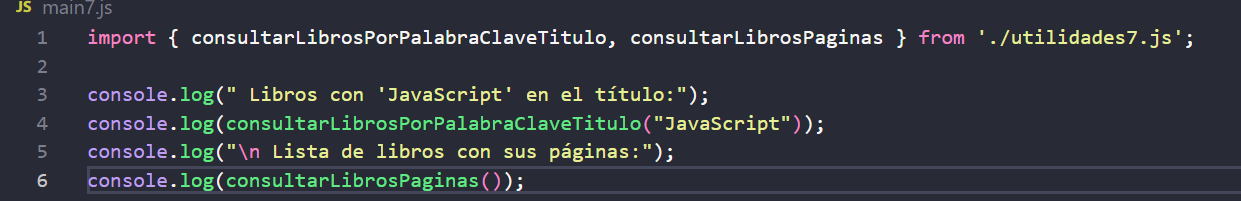
Hacer una función llamada **consultarLibrosPaginas** que cree un arreglo de objetos que contengan:

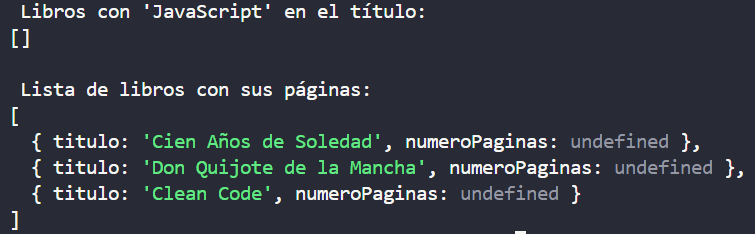
* título del libro
* número de páginas de todos los libros existentes.

**Uso de módulos:**

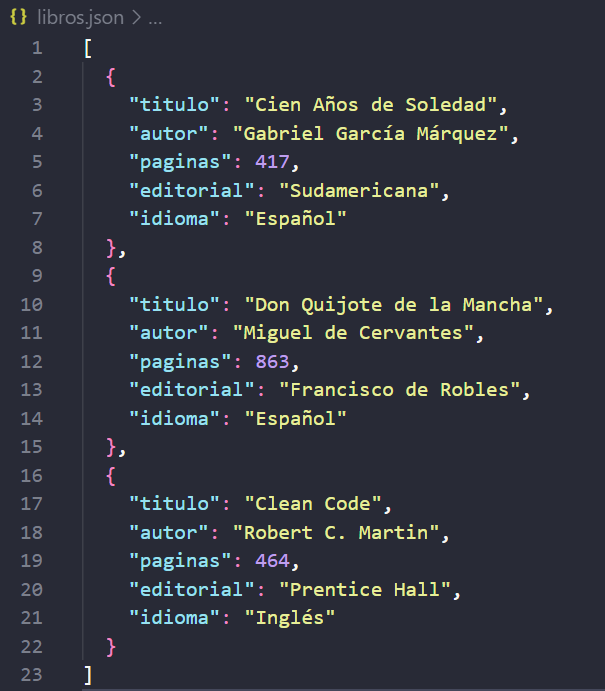
* En un archivo utilidades7.js importa el archivo libros.json.
* Exporta las funciones consultarLibrosPorPalabraClaveTitulo y consultarLibrosPaginas.
* En un archivo main7.js, importa las funciones y pruébalas con diferentes valores.

# **Solución**









# **Actividad o Ejercicio8**

# **Enunciado**

En la alcancía es posible guardar monedas de las siguientes denominaciones: $200, $500 y $1000.  
No se guardan ni billetes ni otras monedas.  
El dueño de la alcancía quiere conocer:

* cuántas monedas tiene de cada denominación,
* cuánto dinero total tiene,
* y poder vaciarla (“romper la alcancía”).

La aplicación debe permitir:

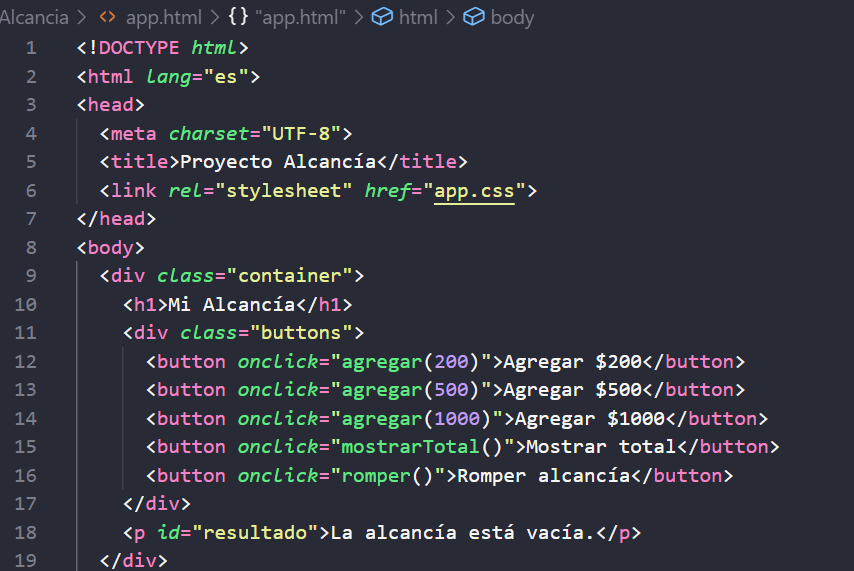
* Agregar una moneda de las denominaciones válidas.
* Contar cuántas monedas tiene de cada denominación.
* Calcular el total de dinero ahorrado.
* Romper la alcancía (vaciar contenido).
* Comenzar una nueva alcancía.

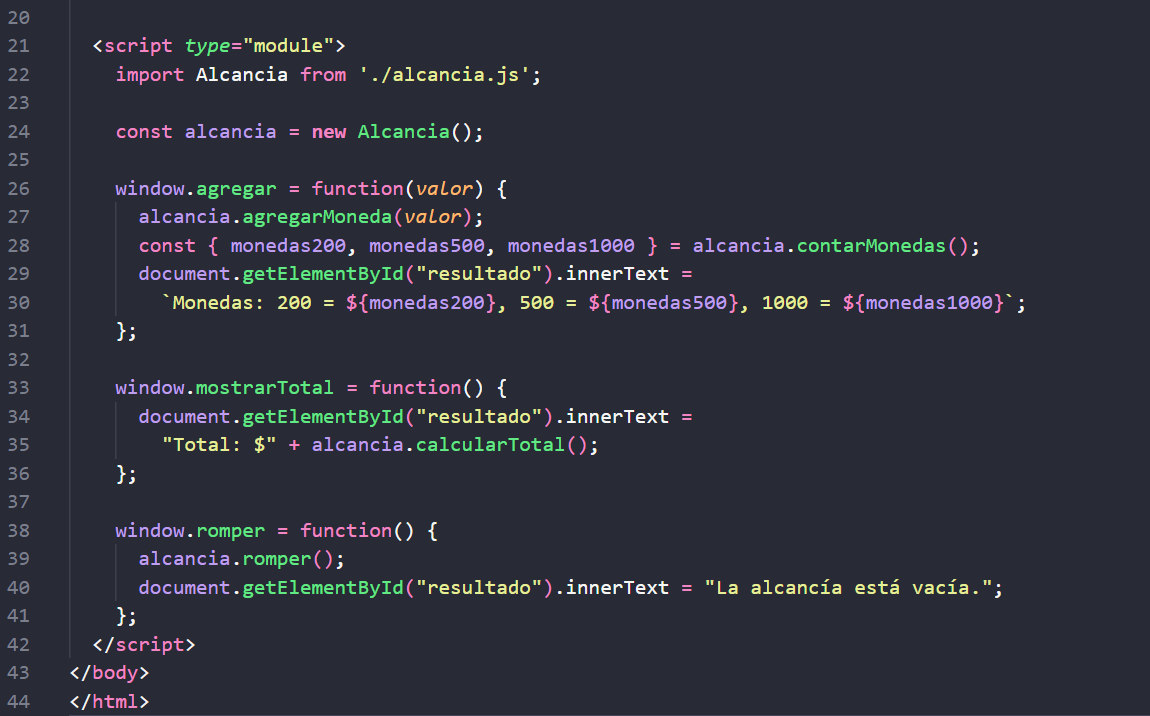
**Uso de módulos:**

* Hacer un export default de la clase **Alcancia** en el archivo alcancia.js.
* En un archivo main8.js, importar la clase **Alcancia**.
* Crear un archivo app8.html para interactuar con el usuario, importando main8.js como módulo.

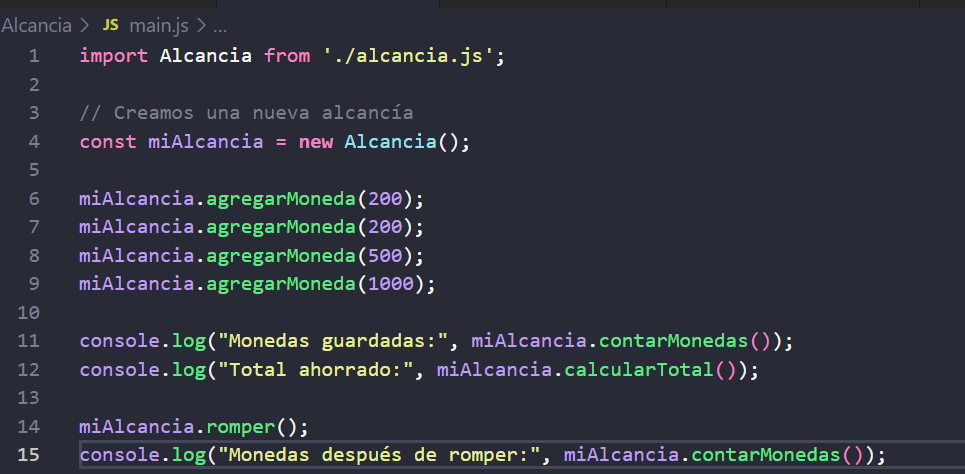
# **Solución**

HTML:





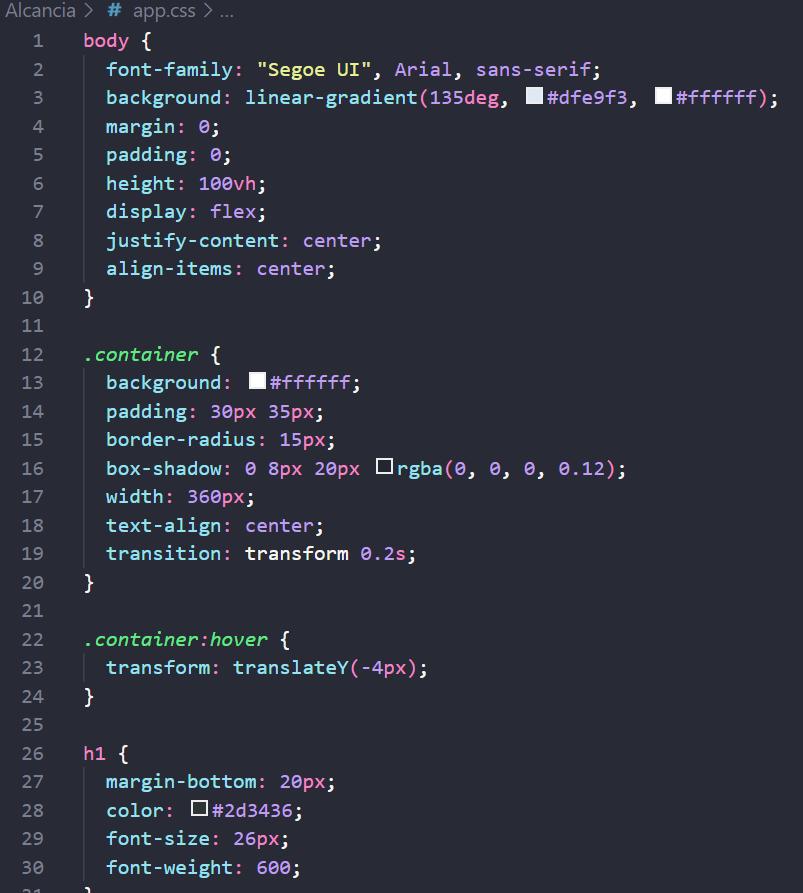
MAIN:



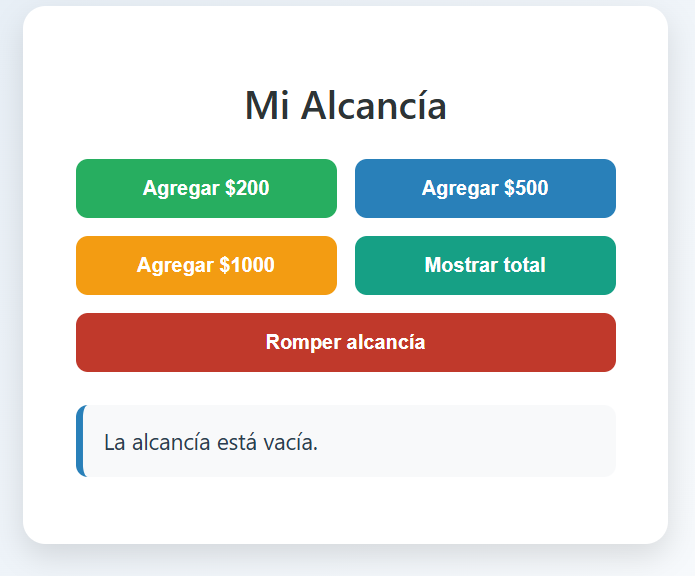
ALCANCIA.JS:



CSS:



**LINK PAGINA WEB ALCANCIA:** [**http://127.0.0.1:5500/Alcancia/app.html**](http://127.0.0.1:5500/Alcancia/app.html)

****

# **Actividad o Ejercicio9**

# **Enunciado**

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:

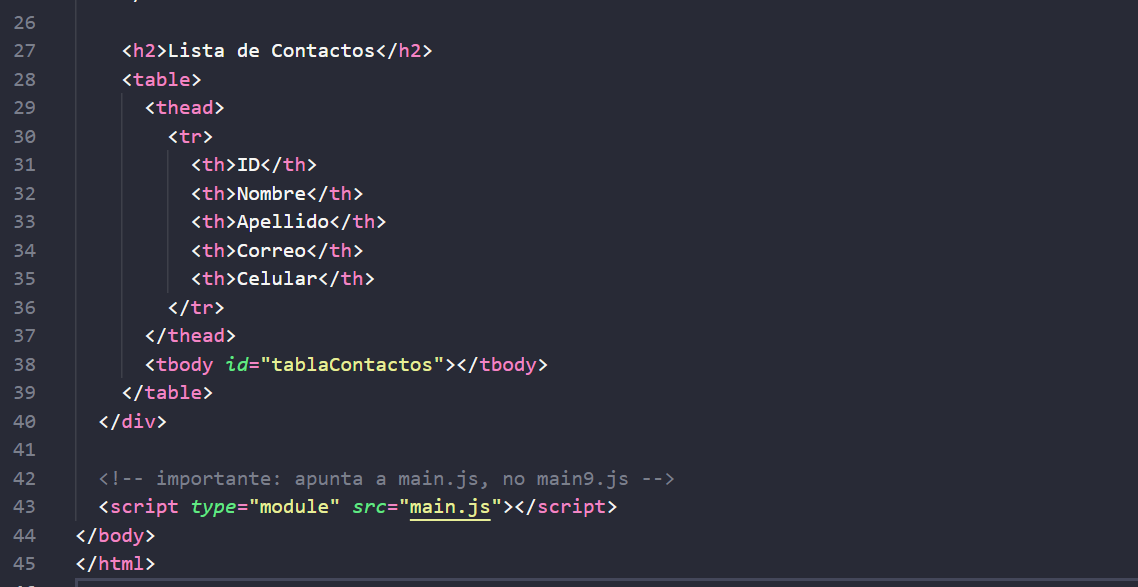
* Identificación
* Nombre
* Apellido
* Correo
* Celular

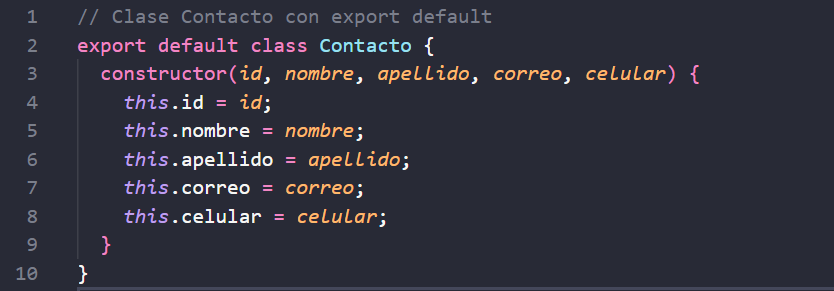
**Uso de módulos:**

* 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> con type="module" llamando al archivo **main.js**.
* En el archivo **main.js** se deben crear las funciones que le permitan 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.

# **Solución HTML:**



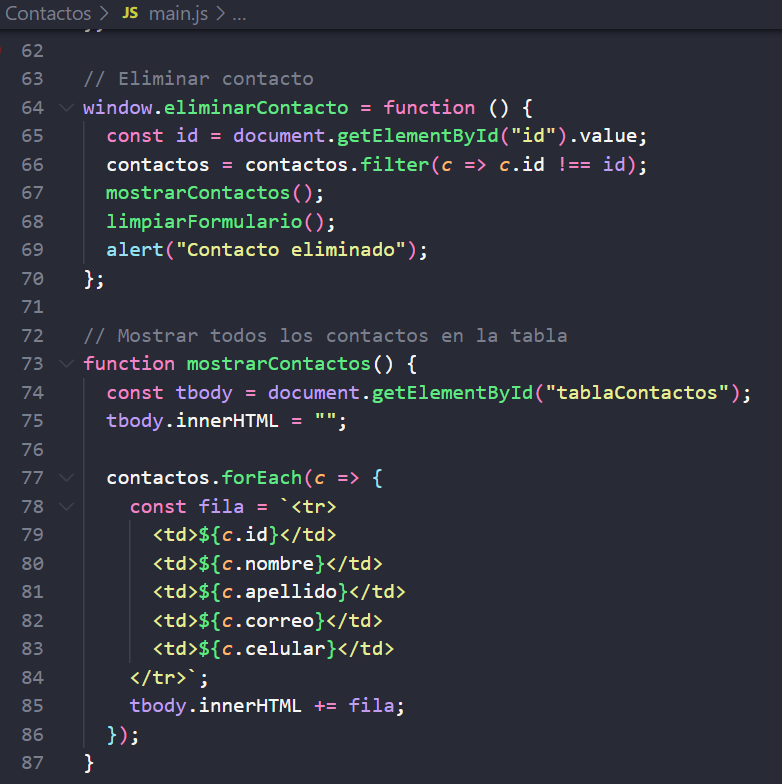


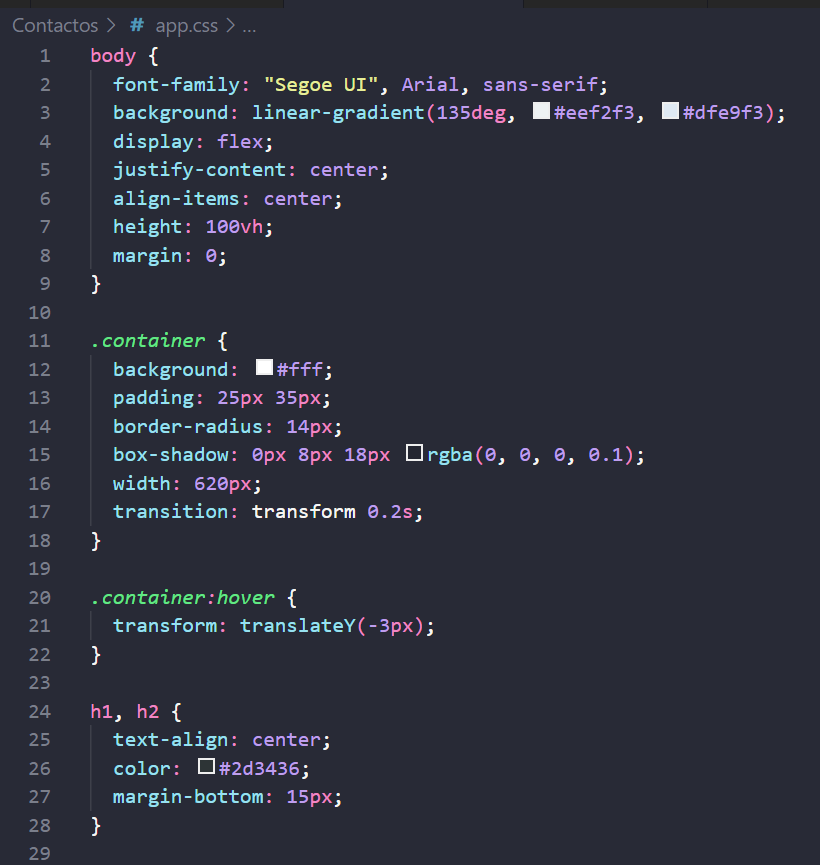
CONTACTO.JS: 

CONSULTAR:







CSS: 

**LINK:** [**http://127.0.0.1:5500/Contactos/app.html**](http://127.0.0.1:5500/Contactos/app.html)

****

# **Bibliografía**

* Piñeiro Gómez, José Manuel. **Diseño de bases de datos relacionales**. Editorial Paraninfo, 2014. 164 páginas.
* Sánchez, José Juan. **Unidad 4 – Teoría de Bases de Datos**. Disponible en: https://josejuansanchez.org/bd/unidad-04-teoria/index.html
* Tutorial de JavaScript Moderno. Disponible en: <https://es.javascript.info/>
* W3Schools. **JavaScript Tutorial**. Disponible en: <https://www.w3schools.com/js/>
* LenguajeJS. **Curso de JavaScript en Español**. Disponible en: <https://lenguajejs.com/javascript/>
* SweetAlert2. **Documentación oficial**. Disponible en: <https://sweetalert2.github.io/>
* Bootstrap. **Documentación oficial**. Disponible en: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
* Curso en YouTube. *Manejo de módulos en JavaScript*. Disponible en: <https://www.youtube.com/watch?v=Z34BF9PCfYg>