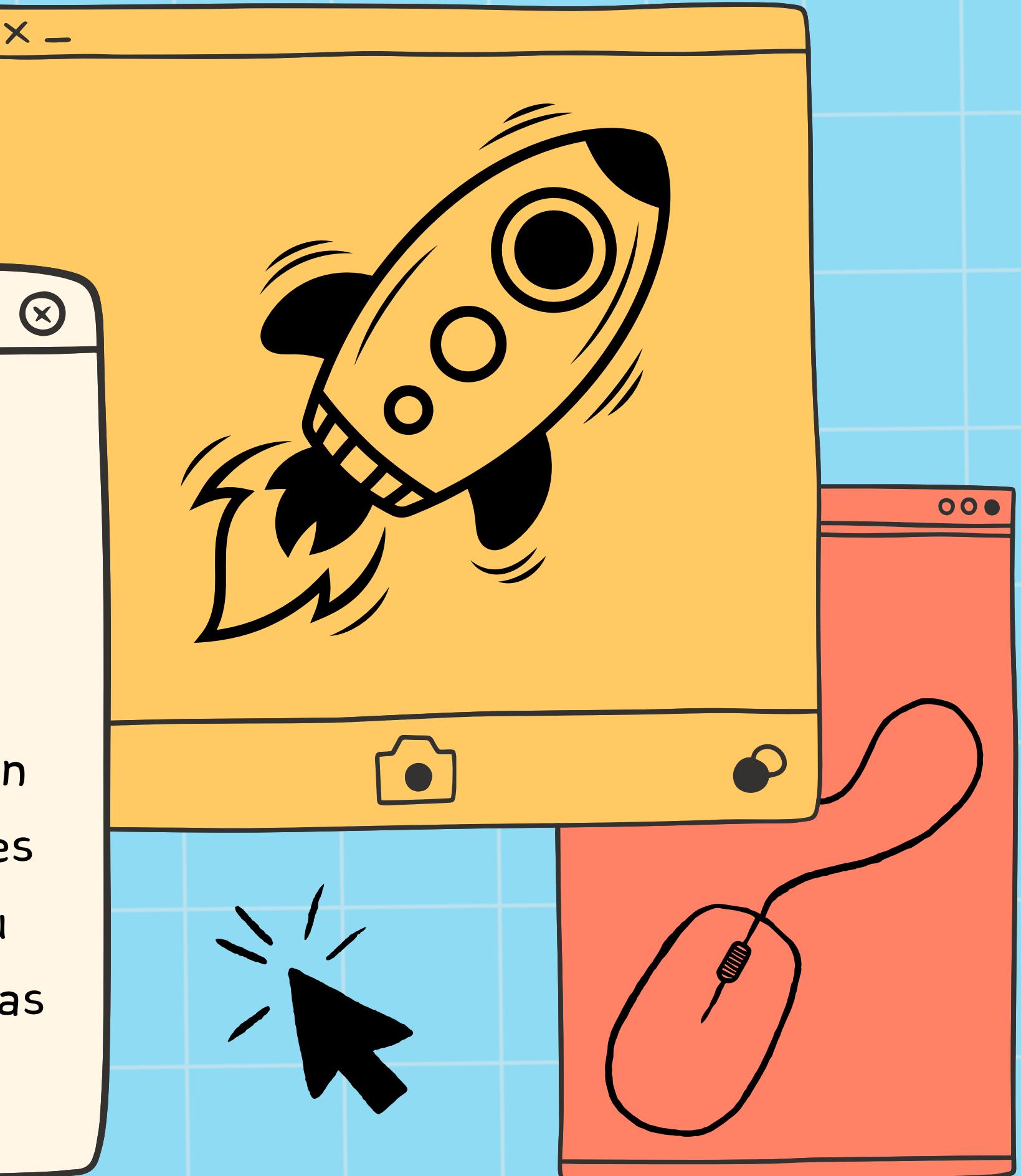
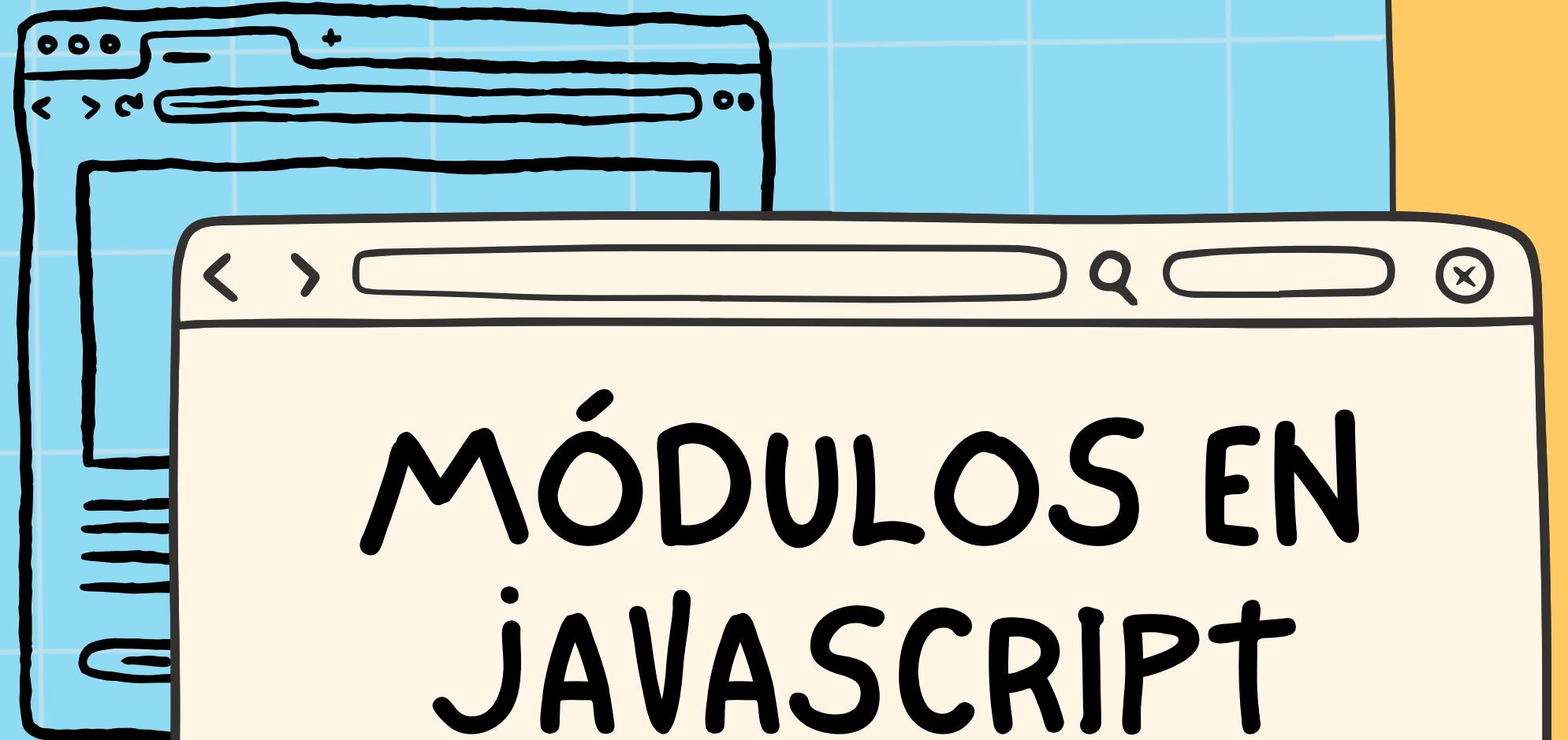


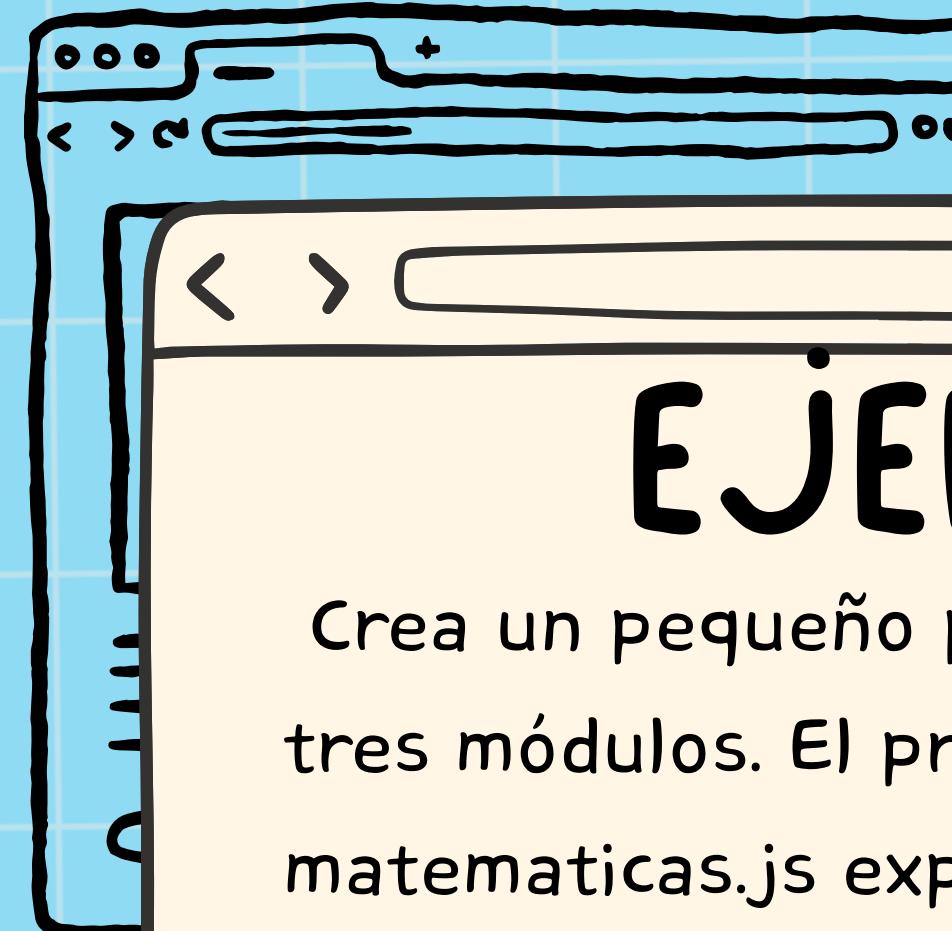
PROGRAMACIÓN ORIENTADA A OBJETOS (POO) EN JAVASCRIPT

La POO es un paradigma de programación basado en el uso de objetos y clases. En JavaScript, puedes crear clases utilizando la sintaxis `class`, definir propiedades y métodos, y usar la palabra clave `this` para referirse al objeto actual.

EJERCICIO...

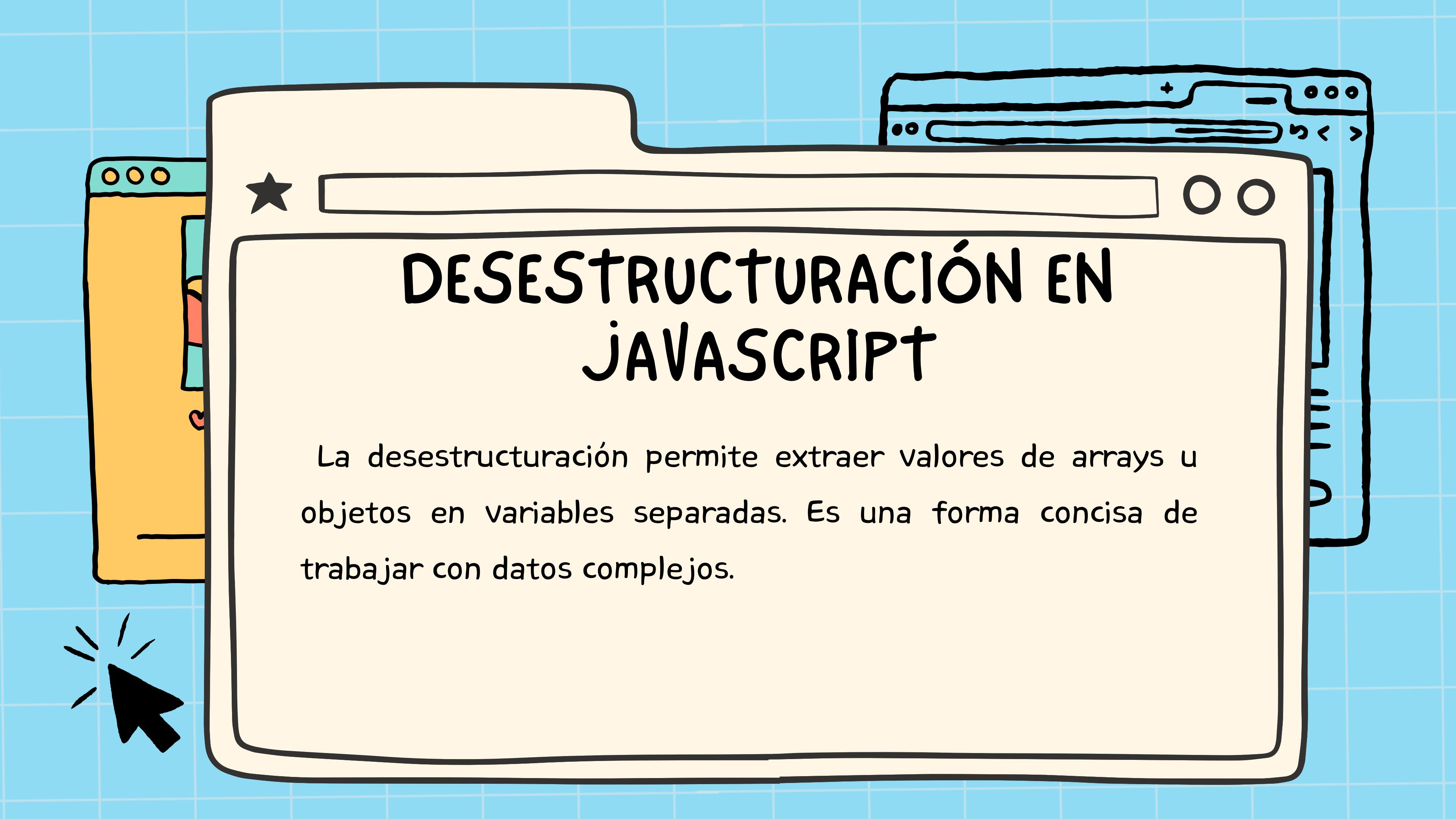
Crea una clase Persona con propiedades como nombre, edad, y genero. Añade métodos saludar() y cumplirAños(). Luego, crea una subclase Estudiante que herede de Persona y agregue la propiedad curso. Implementa un método estudiar() en la subclase que imprima un mensaje personalizado. Crea instancias de ambas clases y usa todos los métodos.





Crea un pequeño proyecto que consista en tres módulos. El primer módulo matematicas.js exporta funciones para operaciones básicas (suma, resta, multiplicacion, division). El segundo módulo utilidades.js exporta una función mostrarResultado. El tercer módulo app.js importa las funciones de los otros dos módulos y realiza operaciones con ellas, mostrando los resultados.





DESESTRUCTURACIÓN EN JAVASCRIPT

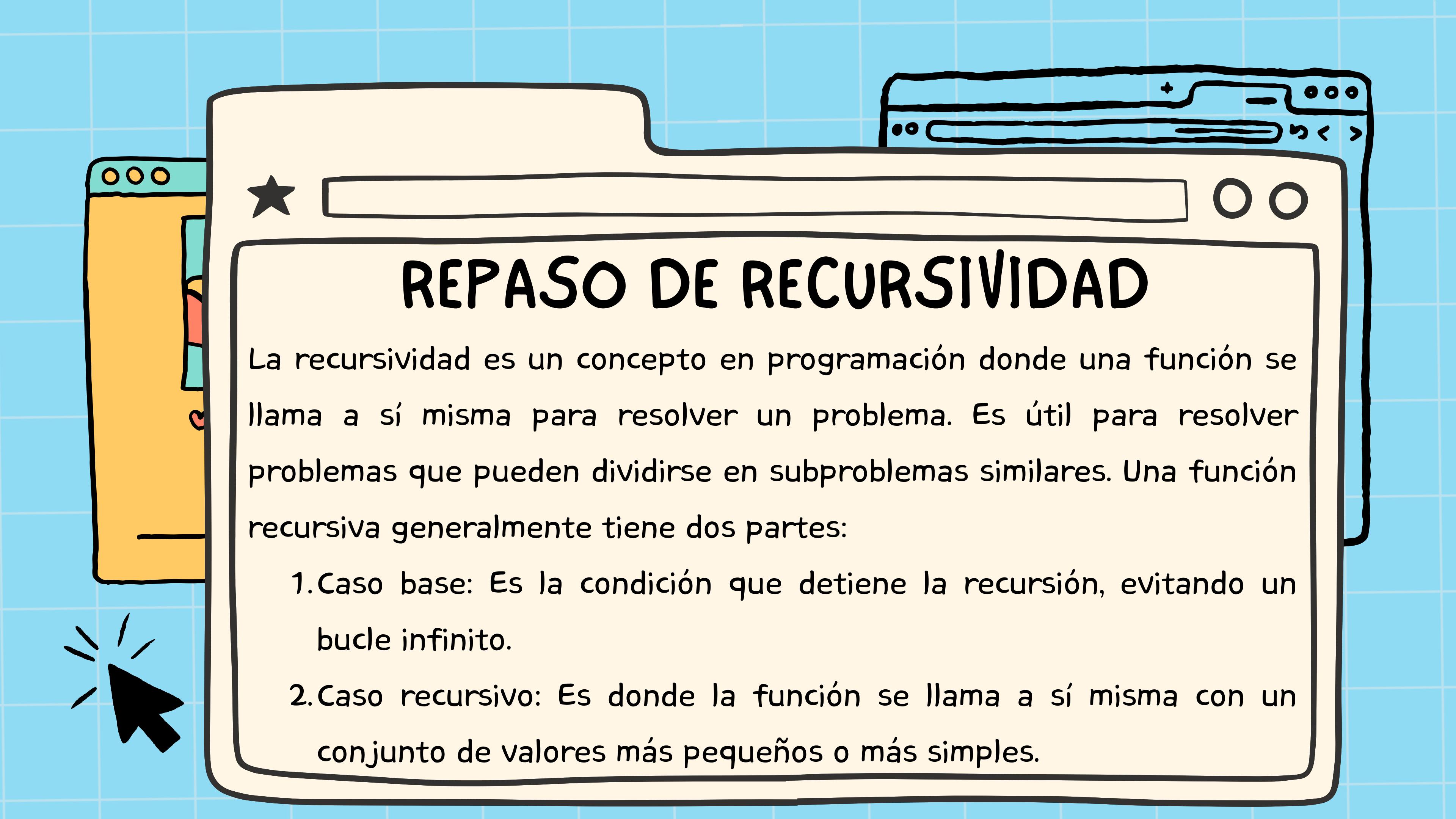
La desestructuración permite extraer valores de arrays u objetos en variables separadas. Es una forma concisa de trabajar con datos complejos.

EJERCICIO

Tienes un objeto empresa con múltiples niveles de información. Usa la desestructuración para extraer y renombrar propiedades, y utiliza valores por defecto para algunas propiedades no definidas. Luego, imprime toda la información extraída.

```
const empresa = { nombre: 'Tech Solutions', ubicacion: { ciudad: 'Bogotá', pais: 'Colombia' }, empleados: [ { nombre: 'Juan', cargo: 'Desarrollador' }, { nombre: 'Ana', cargo: 'Diseñadora' } ], anioFundacion: 2010 };
```

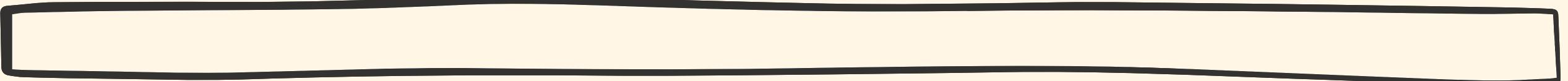




REPASO DE RECUSIVIDAD

La recursividad es un concepto en programación donde una función se llama a sí misma para resolver un problema. Es útil para resolver problemas que pueden dividirse en subproblemas similares. Una función recursiva generalmente tiene dos partes:

1. Caso base: Es la condición que detiene la recursión, evitando un bucle infinito.
2. Caso recursivo: Es donde la función se llama a sí misma con un conjunto de valores más pequeños o más simples.



RETO DE RECURSIVIDAD: SUMA DE NÚMEROS EN UN ARRAY

Tienes un array de números, y tu tarea es escribir una función recursiva que calcule la suma de todos los números en el array.

Instrucciones:

1. Define una función recursiva llamada `sumaArray` que tome dos parámetros: el array de números y un índice que empieza en 0.
2. El caso base será cuando el índice sea igual a la longitud del array, en cuyo caso la función debe devolver 0 (ya que no hay más números que sumar).
3. En el caso recursivo, la función debe devolver el valor del elemento actual del array sumado al resultado de la llamada recursiva para el siguiente índice.

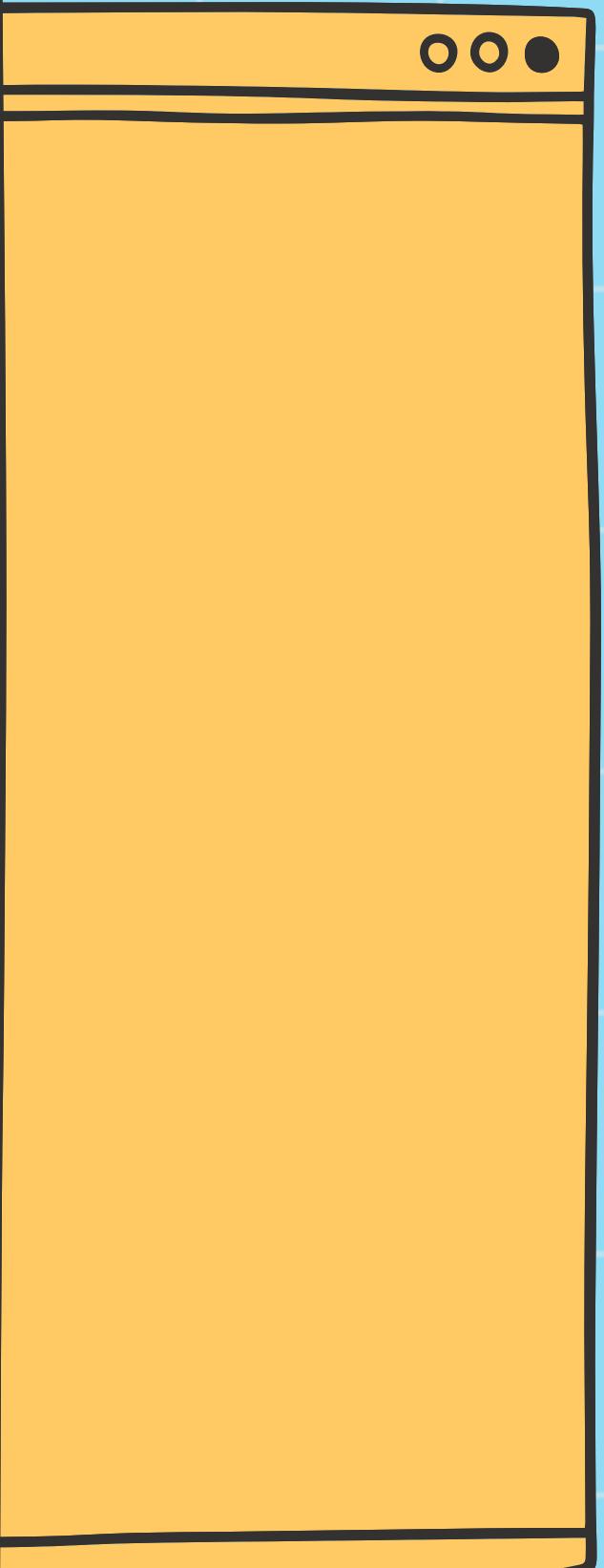


Repaso de callbacks en JavaScript

En JavaScript, un callback es una función que se pasa como argumento a otra función y se ejecuta después de que la primera función ha terminado su trabajo. Los callbacks se usan comúnmente para manejar operaciones asincrónicas, como llamadas a APIs, temporizadores, o tareas que toman tiempo.

Conceptos Clave:

- Función de orden superior: Una función que acepta otra función como argumento o devuelve una función como resultado.
- Callback: La función que se pasa como argumento a otra función.



Ejemplo Básico:

```
function obtenerDatos(callback) {  
    setTimeout(() => {  
        const datos = { nombre: "Luis", edad: 28 };  
        callback(datos); // Llamada al callback con los datos obtenidos  
    }, 2000);  
  
}  
  
function mostrarDatos(datos) {  
    console.log("Datos obtenidos:", datos);  
}  
  
obtenerDatos(mostrarDatos); // Pasamos 'mostrarDatos' como callback a 'obtenerDatos'
```





Reto de callbacks: Encuentra el Elemento Único

Descripción del Reto:

Imagina que tienes un array de números donde todos los números están repetidos excepto uno. Tu tarea es encontrar ese número único utilizando callbacks.

Instrucciones:

1. Crea una función `encontrarUnico` que reciba un array de números y un callback.
2. Dentro de `encontrarUnico`, utiliza un algoritmo que pueda identificar el número que no se repite en el array.
3. Una vez que encuentres el número único, llama al callback y pásale el número encontrado.
4. Crea otra función que se pase como callback a `encontrarUnico` y muestre el número único en la consola.

ooo

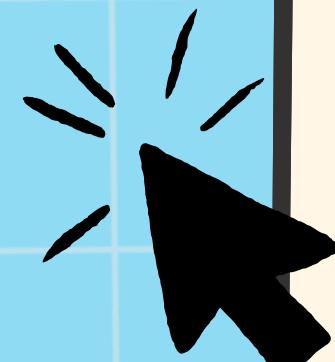




RETO: CONTADOR DE ARCHIVOS EN UN SISTEMA DE CARPETAS

Descripción del Reto:

Imagina que tienes un sistema de carpetas, donde cada carpeta puede contener archivos o más carpetas dentro de ella. Tu tarea es contar el número total de archivos dentro de una estructura de carpetas usando Programación Orientada a Objetos, Recursividad, Módulos, y Desestructuración.





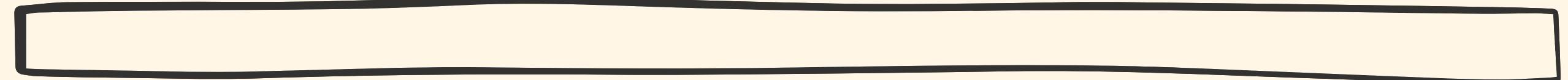
Instrucciones

1. Crear Clases con POO

- Archivo: Representa un archivo dentro de una carpeta.
- Carpeta: Representa una carpeta que puede contener archivos y otras carpetas.
- SistemaDeCarpetas: Administra la estructura de carpetas y contiene un método para contar todos los archivos.

2. Implementar la Recursividad

- La clase SistemaDeCarpetas debe tener un método contarArchivos que recorra las carpetas de manera recursiva para contar todos los archivos.

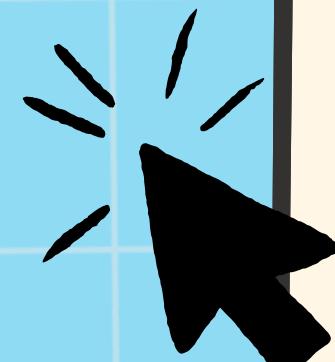


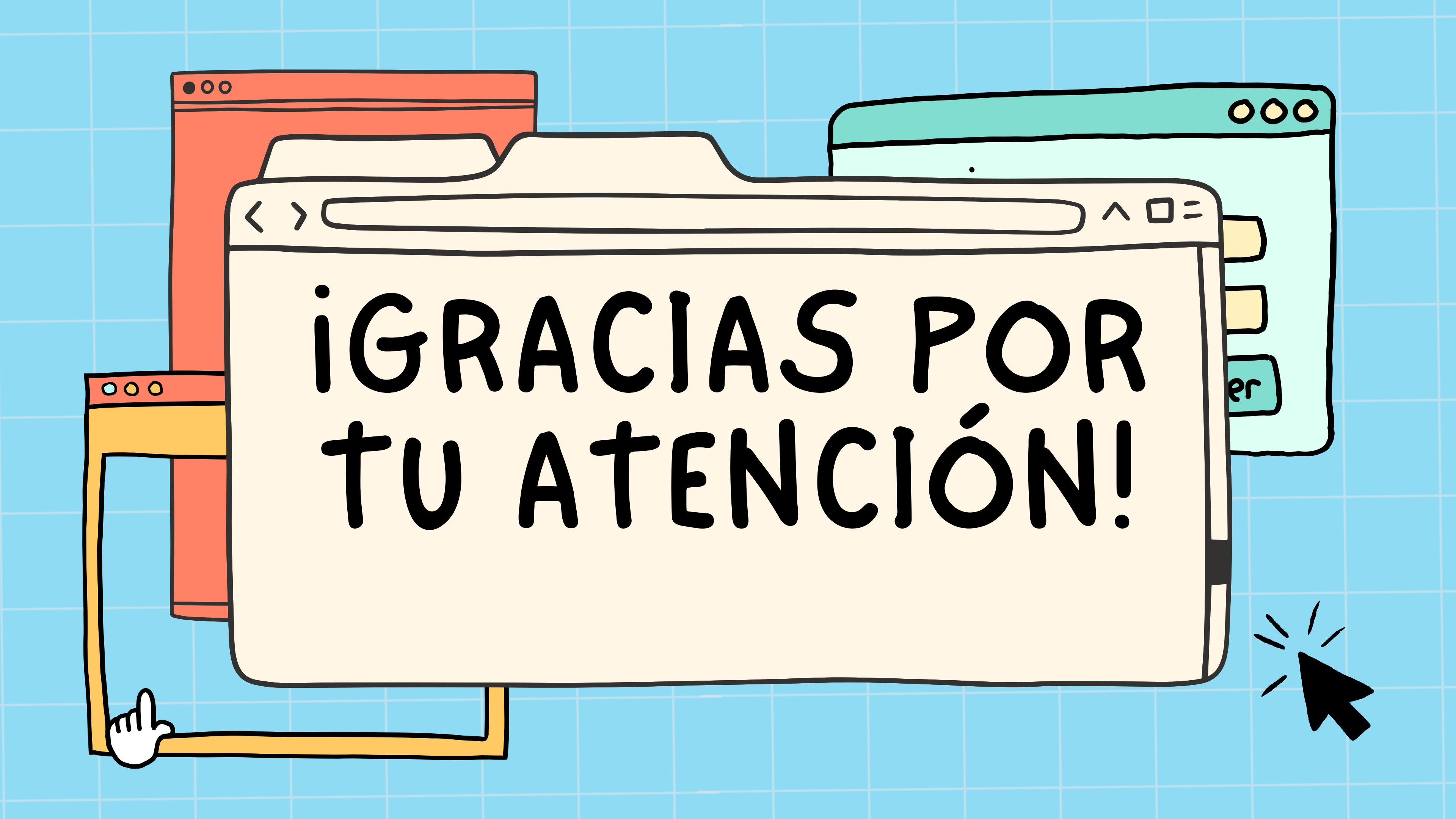
3. Usar Módulos

- Divide el código en dos módulos: uno para las clases (`clases.js`) y otro para la lógica principal del juego (`juego.js`).

4. Aplicar Desestructuración

- Usa desestructuración para extraer propiedades de los objetos Carpeta al contar los archivos.





**¡GRACIAS POR
TU ATENCIÓN!**

