

CHAT GPT

 **Juan:**

Tengo que hacer un trabajo que se llama *Análisis de código con diferentes enfoques* y necesito tu ayuda porque me han dado varias versiones de una misma calculadora hechas en JavaScript y tengo que entenderlas y compararlas entre sí ¿puedes ayudarme con eso?

ChatGPT:

¡Hola Juan claro que sí estaré encantado de ayudarte con ese trabajo tan interesante el objetivo principal es analizar cómo una misma funcionalidad puede estar construida de distintas formas usando JavaScript lo primero que vamos a hacer es verificar si todas las versiones de la calculadora hacen lo mismo y efectivamente todas tienen una interfaz similar con dos cajas para introducir números y un botón que al hacer clic ejecuta tres operaciones fundamentales que son sumar multiplicar y determinar cuál de los dos números es mayor esto es importante porque nos permite comparar el código desde su estructura sin preocuparnos por la funcionalidad en sí ya que esa parte se mantiene constante en todos los ejemplos 😊**

 **Juan:**

Vale perfecto entonces ¿puedes explicarme cómo está hecha cada una? Empieza por la calculadora clásica y coméntame el código paso a paso para que lo entienda

ChatGPT:

Claro Juan la versión clásica de la calculadora utiliza un enfoque muy tradicional en JavaScript se define una función llamada `calcular` que se activa cuando el usuario hace clic en el botón de calcular lo primero que hace esta función es acceder a los valores que ha introducido el usuario en los campos de texto usando `document.getElementById` luego transforma esos valores en números con la función `Number` ya que por defecto todo lo que viene del input es una cadena de texto después de eso calcula la suma y el producto y finalmente determina cuál de los dos números es mayor usando una estructura condicional `if else` si ambos son iguales devuelve un mensaje indicándolo lo interesante aquí es que todo se hace dentro de una sola función y aunque es sencilla puede ser difícil de mantener en proyectos más grandes ya que todo el código está concentrado en un único bloque no hay separación de responsabilidades ni modularidad y eso es algo que debemos tener en cuenta a medida que nuestros programas se hacen más complejos 📁**



Juan:

Háblame de la moderna

□ **ChatGPT:**

La calculadora moderna representa un paso adelante en cuanto a estilo y claridad del código en lugar de usar una función tradicional se implementa una función flecha que está directamente asociada al evento click del botón esto se hace con `addEventListener` que es una manera más limpia de capturar eventos al hacer clic la función flecha recoge los valores de los inputs los transforma en números y luego hace las operaciones necesarias lo interesante aquí es que el código es más compacto más expresivo y se hace uso de una técnica moderna conocida como operador ternario para mostrar si los números son iguales o cuál es mayor esto hace que el código se lea mejor y que sea más fácil de mantener y entender además evita contaminar el espacio global con funciones sueltas ya que todo está contenido dentro del manejador del evento de forma elegante ✨**



Juan:

Y la funcional ¿qué tiene de diferente?

□ **ChatGPT:**

Buena pregunta Juan la calculadora funcional introduce una forma distinta de pensar la programación aquí el objetivo es dividir el código en pequeñas funciones puras cada una con una única responsabilidad por ejemplo `getNumber` se encarga solo de obtener el número a partir del id del input y `setText` se ocupa únicamente de colocar el resultado en el HTML luego tenemos un objeto llamado `operaciones` que contiene funciones también puras que realizan la suma el producto y la comparación este enfoque ayuda a que el código sea más predecible más fácil de probar y más reutilizable además favorece el pensamiento matemático y lógico ya que trabajamos con funciones que no dependen del contexto externo por último la función `calcular` orquesta todo llamando a las funciones en orden y se asocia al botón de calcular mediante `addEventListener` lo que tenemos aquí es un ejemplo de programación funcional en estado puro que aunque puede parecer más abstracta al principio es muy poderosa para escribir código claro y seguro 💡**



Juan:

Y la de clases ¿cómo es?

ChatGPT:

La versión de clases Juan aplica el paradigma de la programación orientada a objetos en este caso se define una clase llamada `Calculadora` que contiene dentro de ella todos los elementos necesarios para hacer funcionar la app dentro del constructor se capturan y almacenan los elementos HTML donde se introducen los números y donde se muestran

los resultados luego hay un método `obtenerNumero` que convierte los valores de los inputs a número otro método `calcular` que recoge esos números y llama al método `mostrarResultados` que se encarga de mostrar la suma el producto y el número mayor esto se hace dentro del objeto lo que encapsula la lógica en una estructura que puede reutilizarse muchas veces finalmente se crea una instancia de esta clase y se le asocia el evento del botón lo interesante aquí es que el código está muy bien organizado con responsabilidades separadas dentro de la clase y aunque es más complejo de entender sobre todo si no estás familiarizado con objetos y clases es ideal para escalar a proyectos grandes y mantener el código ordenado a lo largo del tiempo □**

👤 **Juan:**

¿Puedes compararlas todas en una misma respuesta?

□ **ChatGPT:**

Claro Juan si comparamos las cuatro versiones podemos ver que todas hacen lo mismo pero con enfoques muy distintos la versión clásica es sencilla y directa fácil para empezar pero poco mantenible si el proyecto crece la versión moderna es probablemente la mejor para quien está aprendiendo hoy día ya que usa funciones flecha y eventos de una manera clara elegante y actual la versión funcional da un paso hacia la modularidad total dividiendo el código en funciones puras lo que la hace ideal para entender cómo se estructuran procesos complejos y la versión de clases es la que mejor representa un enfoque profesional orientado a objetos ideal si vas a trabajar en proyectos más amplios que requieren escalabilidad y reutilización de código en cuanto a facilidad de comprensión probablemente la moderna y la funcional sean las más accesibles para un estudiante que empieza mientras que la de clases puede parecer más desafiante pero también es la más potente a largo plazo 🔍

👤 **Juan:**

¿Hay otras formas de hacer esta calculadora?

□ **ChatGPT:**

Sí Juan existen muchas otras formas de implementar una calculadora usando JavaScript dependiendo del objetivo y el contexto por ejemplo podrías usar un framework moderno como React js que permite construir componentes reutilizables y manejar los estados de forma muy eficiente o usar Vue js si prefieres una curva de aprendizaje más suave también podrías utilizar Angular si estás trabajando en aplicaciones empresariales complejas con estructura modular incluso podrías integrar Web Workers si quisieras que los cálculos se hicieran en segundo plano y no bloquearan la interfaz lo cual sería útil si estuvieras procesando datos en tiempo real o con grandes volúmenes todas estas opciones muestran lo amplio y flexible que es JavaScript y cómo se puede adaptar a distintos estilos de programación desde el más sencillo al más avanzado 🌐

👤 **Juan:**

Chat necesito que me muestres una tabla comparativa entre las distintas versiones de la calculadora y que me la expliques paso a paso de forma detallada quiero comprender por qué unas son más fáciles de entender que otras cuál tiene el código más limpio y cuál sería más fácil de mantener o ampliar en el futuro ¿puedes explicármelo todo? □

ChatGPT:

Claro que sí Juan vamos a desglosar esa comparativa con calma para que entiendas cada aspecto en profundidad la tabla que estás mencionando analiza tres características importantes de cada versión de la calculadora que son la facilidad de entender la limpieza del código y la facilidad para mantenerlo o expandirlo en el futuro empecemos con la **facilidad de entender** en este caso la calculadora clásica se evalúa como de dificultad moderada esto es porque aunque tiene una estructura sencilla toda la lógica está concentrada en una sola función larga sin divisiones ni separaciones lo que puede confundir a medida que crece el código en cambio las otras tres versiones la moderna la funcional y la de clases están marcadas con alta facilidad de comprensión porque organizan mejor el código por ejemplo la moderna utiliza funciones flecha y eventos separados lo que permite que cada parte tenga su propio espacio la funcional divide las tareas en funciones pequeñas específicas como obtener datos y mostrar resultados y la de clases encapsula todo dentro de una estructura orientada a objetos que agrupa funciones relacionadas haciendo más fácil entender qué hace cada parte del código en su contexto □

En cuanto al **código limpio** aquí la calculadora moderna y la funcional destacan como las más limpias esto se debe a que utilizan estilos modernos más compactos y elegantes en el caso de la moderna se aprovechan funciones flecha y el uso claro de `addEventListener` lo que evita mezclar la lógica con el HTML mientras que la funcional promueve el uso de funciones puras que no dependen de variables externas y pueden reutilizarse fácilmente todo esto da como resultado un código mucho más claro y legible por otro lado la clásica al tener toda la lógica en un solo bloque presenta un código más denso difícil de mantener visualmente y la de clases aunque estructurada se puede ver más cargada porque utiliza muchos métodos propiedades y referencias a `this` lo que puede ser complicado para alguien que aún no domina la programación orientada a objetos 📖

Por último analizamos la **facilidad de mantenimiento o expansión futura** y aquí vemos que mientras la calculadora clásica se queda en un nivel moderado las otras tres destacan como más preparadas para crecer con el tiempo la moderna al tener una estructura modular permite añadir nuevas funciones fácilmente sin afectar el resto del código la funcional gracias a sus funciones puras puede ampliarse cambiando pequeñas piezas sin que eso afecte el comportamiento general y la de clases al estar orientada a objetos es perfecta para escalar y organizar aplicaciones más complejas ya que puedes añadir métodos crear nuevas clases o reutilizar estructuras existentes sin problema ✂

Pero además Juan la IA también sugiere que se pueden explorar **otros enfoques de implementación** más avanzados por ejemplo podrías probar la **programación reactiva** usando una biblioteca como **React js** que está pensada para manejar de forma eficiente

las interfaces de usuario en este caso los cambios en los datos provocan automáticamente actualizaciones en la interfaz sin tener que tocar el DOM directamente esto permite construir aplicaciones dinámicas de manera muy clara y organizada también puedes utilizar **frameworks como Vue js o Angular** que ofrecen herramientas muy potentes para estructurar mejor el código estos frameworks te permiten trabajar por componentes separar la lógica de la vista y mantener el código escalable ordenado y reutilizable además si tu proyecto requiere hacer muchos cálculos al mismo tiempo y no quieres que eso ralentice la experiencia del usuario podrías usar **Web Workers** que permiten trasladar los cálculos a un hilo separado del principal así mientras los números se procesan la interfaz sigue siendo fluida y no se bloquea todo esto demuestra que JavaScript es un lenguaje extremadamente versátil que permite resolver un mismo problema de múltiples formas y adaptarse a todo tipo de proyectos desde los más simples hasta los más complejos 🚀🌟

👤 **Juan:**

Realizame una tabla comparativa breve y explícala en profundidad

❑ **ChatGPT:**

Por supuesto Juan vamos a ir parte por parte para que lo entiendas todo sin dejarte nada fuera la tabla comparativa que mencionas analiza tres aspectos clave que son **la facilidad de entender el código la limpieza del código y su facilidad de mantenimiento o expansión futura** y lo hace comparando las cuatro versiones que has estado estudiando la calculadora clásica la moderna la funcional y la de clases empecemos por la **facilidad de entender** aquí se valora qué tan accesible es cada enfoque para una persona que esté leyendo o trabajando con el código la **calculadora clásica** se evalúa como de dificultad **moderada** porque aunque tiene una estructura sencilla la lógica está toda dentro de una única función sin modularidad lo que puede resultar confuso cuando se quiere modificar algo o añadir nuevas funcionalidades en cambio las otras tres versiones están marcadas como de **alta facilidad de comprensión** la **moderna** por usar funciones flecha y separar el evento del botón del resto del código la **funcional** porque divide todo en pequeñas funciones que se entienden muy bien por separado y la **de clases** porque organiza los elementos del programa dentro de un objeto que agrupa todo el comportamiento de forma ordenada esto ayuda a ver de forma clara qué hace cada parte del programa en su contexto 🧠

El segundo aspecto es el **código limpio** es decir qué tan legible y organizado está el código aquí las versiones **moderna y funcional** destacan con un nivel **alto** porque ambas se apoyan en prácticas de programación actuales que favorecen la claridad y la estructura por ejemplo la moderna separa el comportamiento del evento y evita mezclarlo con el HTML y la funcional promueve el uso de funciones puras que no dependen del entorno externo lo que permite que el código sea más fácil de leer y de mantener en comparación la **clásica** queda como **moderada** porque aunque funciona todo está aglutinado en una sola función lo que puede generar confusión y la **de clases** queda en un nivel **medio** ya que aunque está bien estructurada puede verse más densa debido a los métodos internos

las propiedades y el uso continuo de `this` que puede resultar difícil de entender al principio si no se domina la programación orientada a objetos 📖

En cuanto al tercer aspecto que es el **mantenimiento o expansión futura** se valora qué tan fácil sería actualizar ese código más adelante o añadirle más funciones aquí la clásica vuelve a quedar como **moderada** porque no está pensada para crecer fácilmente al estar todo en una misma función cualquier cambio puede afectar el resto del programa sin embargo las otras tres versiones reciben una valoración **alta** en este sentido porque están diseñadas de forma que permiten añadir o modificar partes sin que se rompa el resto del código la moderna con su estructura modular es fácil de ampliar la funcional gracias a sus funciones pequeñas puede cambiarse sin problemas y la de clases está preparada para escalar mediante herencia o la creación de nuevas clases derivadas ✂

Y además Juan como bien has visto la IA también te sugiere que explores **otros enfoques más avanzados** que te podrían ayudar a llevar el concepto de la calculadora a otro nivel por ejemplo podrías implementar la lógica de tu calculadora usando **programación reactiva con React js** que es una biblioteca muy potente que permite que los elementos de la pantalla se actualicen automáticamente en función de los datos es decir si el usuario cambia algo el resultado se actualiza sin que tengas que escribir código extra para actualizar el DOM eso hace que todo sea más fluido y ordenado también puedes explorar el **uso de frameworks como Vue js o Angular** que ofrecen una estructura de trabajo más sólida y organizada para aplicaciones grandes estos frameworks te permiten trabajar con componentes reutilizables dividir mejor la lógica y mantener el código limpio incluso si tu aplicación crece mucho y si algún día estás trabajando con muchos cálculos que podrían ralentizar tu aplicación la IA recomienda usar **Web Workers** que son una forma de mover los cálculos a un segundo plano sin que la interfaz se quede bloqueada lo cual es ideal si quieres que la experiencia del usuario siga siendo rápida y fluida incluso mientras se hacen operaciones complejas todo esto muestra que JavaScript no es solo un lenguaje simple sino una herramienta muy poderosa y flexible que se puede adaptar a muchos estilos y necesidades diferentes 🚀🌐

📊 Tabla Comparativa de las Versiones de la Calculadora

Característica	Calculadora Clásica	Calculadora Moderna	Calculadora Funcional	Calculadora con Clases
Facilidad de entender	Moderada	Alta	Alta	Alta
Código limpio	Moderado	Alto	Alto	Medio
Mantenimiento/Futuro	Moderado	Alto	Alto	Alto

👤 **Juan:**

He visto que al final del análisis se sugiere que se podrían usar otros enfoques distintos para implementar la calculadora. Me gustaría que me hables de eso en profundidad.

☐ **ChatGPT:**

Claro Juan te lo explico todo desde cero para que lo entiendas perfectamente y puedas incluirlo en tu trabajo con toda seguridad cuando hablamos de **otros enfoques de implementación** nos referimos a formas más avanzadas de construir aplicaciones que van más allá del uso básico de JavaScript tradicional y están pensadas para facilitar el desarrollo cuando el proyecto crece o se vuelve más dinámico por ejemplo una de las sugerencias que te da la IA es usar **programación reactiva** esto se refiere a un estilo de programación donde los elementos de la interfaz de usuario reaccionan automáticamente cuando los datos cambian una de las bibliotecas más conocidas para esto es **React js** con React no tienes que estar actualizando el DOM manualmente como haces con `getElementById` o `textContent` en lugar de eso defines componentes que se renderizan automáticamente cada vez que cambian los datos eso significa que el código es más limpio más fácil de mantener y mucho más escalable especialmente útil si tu aplicación tiene muchos elementos que cambian en tiempo real o si necesitas tener una estructura moderna y profesional ✨

Luego tenemos la opción de usar **frameworks como Vue js o Angular** estos frameworks son herramientas más completas que no solo te ayudan a construir la interfaz sino también a organizar todo tu proyecto con una estructura clara por ejemplo en **Vue js** puedes separar la lógica del HTML y dividir tu aplicación en **componentes reutilizables** lo que te permite trabajar mejor en equipo o mantener el código ordenado si tu aplicación se hace más grande lo mismo pasa con **Angular** aunque es un poco más complejo está diseñado para proyectos grandes y corporativos y te ofrece muchas herramientas integradas para que no tengas que preocuparte por detalles como el enrutamiento la gestión del estado o las peticiones al servidor en resumen estos frameworks están hechos para que puedas enfocarte en la lógica de tu aplicación sin complicarte con la estructura interna ☐

Y por último te hablo de los **Web Workers** que son una opción súper interesante cuando tienes que hacer **muchas operaciones pesadas** por ejemplo si tu calculadora tuviera que procesar cientos de números o trabajar con cálculos complejos como estadísticas grandes funciones o simulaciones físicas podrías usar un Web Worker para que todos esos cálculos se hagan en segundo plano sin bloquear la interfaz eso significa que mientras el Worker hace su trabajo tu página sigue funcionando y el usuario puede seguir escribiendo haciendo clic o viendo los resultados sin que se quede congelado esto es ideal para que la experiencia del usuario sea fluida y profesional especialmente si estás pensando en trabajar con gráficos datos en tiempo real o tareas que requieren mucho procesamiento

⚙️☐

Parte 1: Verificación de que todas hagan lo mismo

Cada una de las calculadoras realiza las mismas operaciones (suma, multiplicación y encontrar el número mayor) y tiene la misma interfaz con dos entradas de números y un botón para realizar el cálculo. Los resultados de las operaciones se muestran en el mismo formato.

- **Calculadora Clásica**: Usa una función tradicional en JavaScript para realizar las operaciones y actualizar los resultados en el DOM con `getElementById`.
- **Calculadora Moderna**: Utiliza un enfoque más moderno con funciones de flecha y el uso de `addEventListener` para manejar el evento del botón.
- **Calculadora Funcional**: Se enfoca en un estilo más funcional con funciones puras para obtener el número, establecer el texto y realizar las operaciones.
- **Calculadora con Clases**: Implementa una clase `Calculadora` para estructurar el código y manejar las operaciones.

En resumen, todas las calculadoras realizan las mismas funciones de suma, multiplicación y encontrar el mayor número, aunque usan diferentes enfoques de programación.

Parte 2: Análisis del código con IA

A continuación se presenta el código comentado de cada una de las versiones de la calculadora.

Calculadora Clásica

```
// Esta función se activa cuando se hace clic en el botón 'Calcular'
function calcular() {
    // Se obtienen los valores de los números ingresados
    const numero1 = Number(document.getElementById("numero1").value);
    const numero2 = Number(document.getElementById("numero2").value);

    // Se calculan la suma y el producto
    const suma = numero1 + numero2;
    const producto = numero1 * numero2;
    let mayor;

    // Se determina cuál es el número mayor
    if (numero1 > numero2) {
        mayor = numero1;
    } else if (numero2 > numero1) {
        mayor = numero2;
    } else {
        mayor = "Los números son iguales";
    }

    // Se muestran los resultados en el HTML
    document.getElementById("suma").textContent = suma;
    document.getElementById("producto").textContent = producto;
    document.getElementById("mayor").textContent = mayor;
}
```

Calculadora Moderna

```
// Se agrega un event listener al botón para ejecutar el cálculo cuando se hace clic
document.getElementById("calcularBtn").addEventListener("click", () => {
  // Se obtienen los números ingresados como valores numéricos
  const numero1 = Number(document.getElementById("numero1").value);
  const numero2 = Number(document.getElementById("numero2").value);

  // Se calculan la suma, el producto y el número mayor
  const suma = numero1 + numero2;
  const producto = numero1 * numero2;
  const mayor = numero1 === numero2 ? "Los números son iguales" :
Math.max(numero1, numero2);

  // Se muestran los resultados en el HTML
  document.getElementById("suma").textContent = suma;
  document.getElementById("producto").textContent = producto;
  document.getElementById("mayor").textContent = mayor;
});
```

Calculadora Funcional

```
// Función para obtener el valor numérico de un input por su id
const getNumber = (id) => Number(document.getElementById(id).value);

// Función para establecer el texto de un elemento por su id
const setText = (id, value) => document.getElementById(id).textContent = value;

// Función principal para realizar los cálculos
const calcular = () => {
  // Se obtienen los dos números ingresados
  const [numero1, numero2] = [getNumber("numero1"), getNumber("numero2")];

  // Se definen las operaciones de suma, producto y mayor usando funciones puras
  const operaciones = {
    suma: (a, b) => a + b,
    producto: (a, b) => a * b,
    mayor: (a, b) => (a === b ? "Los números son iguales" : Math.max(a, b))
  };

  // Se establecen los resultados en el HTML
  setText("suma", operaciones.suma(numero1, numero2));
  setText("producto", operaciones.producto(numero1, numero2));
  setText("mayor", operaciones.mayor(numero1, numero2));
};

// Se asocia la función calcular al botón de clic
document.getElementById("calcularBtn").addEventListener("click", calcular);
```

Calculadora con Clases

// Definición de la clase Calculadora

```
class Calculadora {
```

```
    // Constructor que recibe los ids de los elementos HTML
```

```
    constructor(numero1Id, numero2Id, sumald, productold, mayorId) {
```

```
        this.numero1Input = document.getElementById(numero1Id);
```

```
        this.numero2Input = document.getElementById(numero2Id);
```

```
        this.sumaOutput = document.getElementById(sumald);
```

```
        this.productoOutput = document.getElementById(productold);
```

```
        this.mayorOutput = document.getElementById(mayorId);
```

```
    }
```

```
    // Método para obtener el número ingresado
```

```
    obtenerNumero(input) {
```

```
        return Number(input.value);
```

```
    }
```

```
    // Método principal que ejecuta los cálculos
```

```
    calcular() {
```

```
        const numero1 = this.obtenerNumero(this.numero1Input);
```

```
        const numero2 = this.obtenerNumero(this.numero2Input);
```

```
        // Se llaman a los métodos para mostrar los resultados
```

```
        this.mostrarResultados(numero1, numero2);
```

```
    }
```

```
    // Método para mostrar los resultados de las operaciones
```

```
    mostrarResultados(numero1, numero2) {
```

```
        this.sumaOutput.textContent = numero1 + numero2;
```

```
        this.productoOutput.textContent = numero1 * numero2;
```

```
        this.mayorOutput.textContent = numero1 === numero2 ? "Los números son
```

```
iguales" : Math.max(numero1, numero2);
```

```
    }
```

```
}
```

```
// Se crea una instancia de la clase y se asocia el evento al botón
const calculadora = new Calculadora("numero1", "numero2", "suma", "producto",
"mayor");
document.getElementById("calcularBtn").addEventListener("click", () =>
calculadora.calcular());
```

Comparativa de las Versiones

Aquí se presenta una tabla comparativa entre las diferentes versiones de la calculadora en términos de facilidad de entender, limpieza del código y facilidad de mantenimiento/expansión.

Característica	Calculadora Clásica	Calculadora Moderna	Calculadora Funcional	Calculadora con Clases
Facilidad de entender	Moderada	Alta	Alta	Alta
Código limpio	Moderado	Alto	Alto	Medio
Mantenimiento/Futuro	Moderado	Alto	Alto	Alto

Sugerencia de Otros Enfoques

La IA sugiere que se podrían explorar más enfoques de implementación como:

- **Programación reactiva** usando bibliotecas como React.js, que permite un manejo más eficiente de las interfaces de usuario.
- **Uso de frameworks** como Vue.js o Angular para estructurar más fácilmente el código.
- **Uso de Web Workers** para hacer los cálculos en segundo plano si se tiene una gran cantidad de operaciones.

Parte 3: Conclusiones

1. ¿Qué estilo te parece más fácil y más difícil?

Para mí, la *Calculadora Moderna* es la más fácil. El código es más sencillo y se entiende mejor, sobre todo porque no tengo mucha experiencia con programación. Es más directo y fácil de seguir.

En cambio, la *Calculadora con Clases* me resulta más difícil. Aunque tiene sus ventajas, el código es más complicado y no me resulta tan fácil de entender, especialmente porque no tengo mucha idea de cómo funcionan las clases y los objetos en programación.

2. ¿Es todo JavaScript? ¿Por qué tantas formas?

Sí, todas las versiones están escritas en JavaScript, pero cada una emplea un enfoque diferente. JavaScript por lo que he podido investigar es muy flexible y permite varios estilos para resolver un mismo problema. Por ejemplo:

- *Calculadora Clásica*: Usa funciones globales, un enfoque tradicional.
- *Calculadora Moderna*: Emplea funciones flecha y eventos, haciendo el código más limpio y modular.
- *Calculadora Funcional*: Se enfoca en funciones puras, promoviendo un estilo matemático.
- *Calculadora con Clases*: Usa programación orientada a objetos, ideal para proyectos más grandes y estructurados.

3. ¿Qué versión de la calculadora es más adecuada para nosotros?

Para proyectos simples, la Calculadora Moderna es la mejor opción: es fácil de entender, eficiente y suficiente para tareas básicas. Si el objetivo es aprender programación orientada a objetos o desarrollar aplicaciones más complejas, la Calculadora con Clases es la mejor elección.

4. ¿Qué versión de la calculadora es más adecuada para aprender?

Después de investigar las cuatro versiones, creo que la Calculadora Moderna es la mejor para aprender. Es fácil de entender y permite experimentar con el código mientras aprendes. Introduce conceptos modernos de forma clara, lo que me parece ideal para alguien que está empezando y quiere practicar sin complicarse demasiado.

5. ¿Qué ventajas y desventajas encuentras en cada enfoque?

- **Calculadora Clásica:**
 - Ventajas: Fácil de entender para principiantes, código sencillo.
 - Desventajas: Difícil de mantener en proyectos grandes, ya que las funciones globales pueden generar conflictos.
- **Calculadora Moderna:**
 - Ventajas: Código limpio, modular y fácil de mantener.
 - Desventajas: Requiere algo de práctica con funciones flecha y eventos.

- **Calculadora Funcional:**
 - Ventajas: Promueve código con funciones puras.
 - Desventajas: Puede ser difícil de entender sin experiencia en programación.
- **Calculadora con Clases:**
 - Ventajas: Facilita el trabajo con aplicaciones grandes gracias a la programación orientada a objetos.
 - Desventajas: Requiere comprensión de clases y objetos, lo que puede ser complicado para principiantes.

6. ¿Qué versión o propuesta te parece más interesante?

Desde mi punto de vista, la Calculadora Moderna es la más interesante. Como no tengo experiencia, me parece que el planteamiento lógico es más sencillo y fácil de entender. Es directo y claro, lo que me ayuda a seguir el código sin complicaciones. Creo que es una buena forma de empezar a familiarizarme con la programación sin que sea complicado para mí.