## ANÁLISIS DE LA CLASE VISTA

La clase View actúa como el puente entre el código de TypeScript y el HTML. Su función principal es capturar los elementos del DOM y proporcionar métodos para actualizar la pantalla sin que el resto de la aplicación tenga que manipular el HTML directamente.

### 1. Propiedades de la Clase (Atributos)

En la primera sección, se definen los elementos que la calculadora necesita controlar. Se utiliza el tipado fuerte de TypeScript para asegurar que cada propiedad corresponda a un elemento HTML específico:

- **HTMLParagraphElement / HTMLSpanElement**: Se usan para etiquetas de texto (como el historial de la operación).
- **HTMLInputElement**: Se usa para el cuadro de texto del resultado, permitiendo leer valores.
- **NodeListOf<HTMLButtonElement>**: Representa una colección (lista) de botones, ideal para aplicar una misma lógica a todos los números o a todos los operadores a la vez.
- **HTMLButtonElement**: Referencias a botones únicos con funciones específicas (igual, limpiar, borrar último).

### 2. El Constructor: Vinculación con el DOM

El constructor es el encargado de inicializar la vista en cuanto se crea la instancia.

- Utiliza document.querySelector y querySelectorAll para buscar las clases CSS definidas en tu HTML.
- **El operador ! (Non-null assertion)**: Le indica a TypeScript que estamos seguros de que ese elemento existe en el HTML, evitando errores de "posiblemente nulo".
- **Conversión de tipos (as HTMLInputElement)**: Asegura que el compilador trate al elemento genérico como un campo de entrada para acceder a su propiedad .value.

### 3. Métodos de Actualización (Interfaz Dinámica)

*showResult(valor: string)*

Es el método principal para mostrar el número actual o el resultado final.

- **Control de desbordamiento**: Si el número es muy largo (más de 10 caracteres), lo recorta y añade puntos suspensivos para no romper el diseño visual.
- **Manejo de errores**: Incluye una validación amigable para el usuario: si el resultado matemático es Infinity (típico de una división por cero), muestra el mensaje: *"No se puede dividir ÷ 0"*.
- **Estado de la UI**: Bloquea automáticamente los botones numéricos si se alcanza el límite de caracteres permitido.

*showOperation(valor: string)*

Actualiza la parte superior de la calculadora (la operación en curso), permitiendo que el usuario vea qué números y operadores ha presionado antes del resultado final.

*habilitarNumeros()*

Un método de utilidad que restablece el estado de los botones. Es crucial para cuando el usuario limpia la pantalla o borra caracteres y necesita volver a escribir.

excedeLimite(valor: string)

Una función auxiliar (helper) que devuelve un valor booleano (true/false). Sirve para que el **Controlador** pueda decidir si permite seguir agregando dígitos o no, manteniendo la lógica de validación centralizada.

---

**Importancia en el Patrón MVC**

En este diseño, la clase Vista es **pasiva**. No sabe qué es una "suma" o una "multiplicación"; simplemente recibe strings y los imprime, o detecta clicks y espera a que el **Controlador** le diga qué hacer. Esto facilita el mantenimiento: si mañana decides cambiar el diseño HTML, solo tendrías que modificar esta clase.