T2Actividad7: Calculadora extensible con interacción

Ejercicio: Crear una calculadora extensible con `alert`

Enunciado

El objetivo del ejercicio es crear una calculadora extensible con las siguientes características:

1. **Método** `calculate`:
   - Solicita al usuario una operación en formato `"NUMERO operador NUMERO"` usando `prompt`.
   - Calcula el resultado y lo muestra con un `alert`.
   - Detecta y maneja errores, como:
     - Números no válidos.
     - Operadores no soportados.
2. **Método** `addMethod`:
   - Permite al usuario agregar nuevos operadores interactivamente.
   - Solicita:
     - El símbolo del operador (`+`, `-`, `/`, etc.).
     - El código de la operación como una función.
   - Agrega el operador y permite probarlo con una nueva operación.

Análisis del Problema

1. **Solicitud inicial de operación**:
   - Pedir una operación básica al usuario.
   - Calcular y mostrar el resultado con `alert`.
2. **Extensibilidad**:
   - Permitir al usuario agregar un nuevo operador interactivamente.
   - Solicitar la operación que utiliza el nuevo operador.
3. **Validación**:
   - Manejar errores como entradas no numéricas, operadores inválidos o funciones mal definidas.

Solución Propuesta

Se implementó una solución en JavaScript que utiliza:

- Un objeto `Calculator` con métodos:
  - `calculate` para realizar operaciones básicas.
  - `addMethod` para agregar nuevos operadores.
- Interactividad con el usuario mediante `prompt`, `alert` y `confirm`.

Código Fuente

```
function Calculator() {
    this.methods = {
```

```
        "+": (a, b) => a + b,
        "-": (a, b) => a - b,
    };

    this.calculate = function (str) {
        const [num1, operator, num2] = str.split(" ");

        const a = parseFloat(num1);
        const b = parseFloat(num2);

        if (isNaN(a) || isNaN(b)) {
            alert("Error: Los números no son válidos.");
            return;
        }

        const func = this.methods[operator];

        if (!func) {
            alert(`Error: Operador no válido "${operator}".`);
            return;
        }

        return func(a, b);
    };

    this.addMethod = function (name, func) {
        this.methods[name] = func;
    };
}

function ejecutarCalculadoraExtensible() {
    const calc = new Calculator();

    const operacion = prompt(
        'Introduce la operación en formato "NUMERO operador NUMERO" (e.g., "3 + 5"):'
    );
    const resultado = calc.calculate(operacion);

    if (resultado !== undefined) {
        alert(`El resultado de la operación "${operacion}" es: ${resultado}`);
    }

    const agregar = confirm("¿Quieres agregar una nueva operación a la calculadora?");
    if (agregar) {
        const nuevoOperador = prompt("Introduce el símbolo del nuevo operador (e.g.,
'*'):");
        const nuevoCodigo = prompt(
            "Introduce el código de la función en formato (a, b) => {...}:\nPor ejempl
o: (a, b) => a * b"
        );

        try {
            const nuevaFuncion = eval(nuevoCodigo);
            calc.addMethod(nuevoOperador, nuevaFuncion);
            alert(`El operador "${nuevoOperador}" se ha agregado correctamente.`);
```

```
            const nuevaOperacion = prompt(
                `Introduce una operación usando el nuevo operador "${nuevoOperador}":`
            );
            const nuevoResultado = calc.calculate(nuevaOperacion);

            if (nuevoResultado !== undefined) {
                alert(
                    `El resultado de la operación "${nuevaOperacion}" es: ${nuevoResul
tado}`
                );
            }
        } catch (error) {
            alert("Error: No se pudo agregar la nueva operación. Verifica el códig
o.");
        }
    }
}

ejecutarCalculadoraExtensible();
```

Pruebas Realizadas

1. **Operación básica**:
   - Entrada: `"3 + 5"`.
   - Salida esperada: `El resultado de la operación "3 + 5" es: 8`.
2. **Agregar multiplicación**:
   - Entrada: `"*"` como operador, `(a, b) => a * b` como función.
   - Operación nueva: `"2 * 3"`.
   - Salida esperada: `El resultado de la operación "2 * 3" es: 6`.
3. **Error en operador o función**:
   - Entrada de operador inválido o función mal definida.
   - Salida esperada: `Error: No se pudo agregar la nueva operación. Verifica el código.`

Conclusión

El programa permite realizar operaciones básicas, extender la funcionalidad con nuevos operadores y manejar errores adecuadamente. Toda la interacción se realiza mediante `alert`, `prompt` y `confirm`.