EJERCICIO COMENTADO. CALCULADORA.

Crear un programa 'Calculadora', con las 4 operaciones básicas, sumar, restar, multiplicar y dividir. ¿Cómo se ha de 'subdividir' el programa?

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"/>
<script>

function sumar() {
        let num1, num2, res;
        num1 = parseInt(document.calculadora.valor1.value);
        num2 = parseInt(document.calculadora.valor2.value);
        res = num1 + num2;

        document.calculadora.resultado.value = res;
    }

function restar() {
        let num1, num2, res;
        num1 = parseInt(document.calculadora.valor1.value);
        num2 = parseInt(document.calculadora.valor2.value);
        res = num1 - num2;

        document.calculadora.resultado.value = res;
    }

function multiplicar() {
        let num1, num2, res;
        num1 = parseInt(document.calculadora.valor1.value);
        num2 = parseInt(document.calculadora.valor2.value);
        res = num1 * num2;
        document.calculadora.resultado.value = res;
    }

function dividir() {
      // codigo similar a los anteriores
    }
</script>
</head>

<body>
<form name="calculadora">
Primer número: <input type="text" name="valor1"><br>
Segundo número: <input type="text" name="valor2"><br>

<input type="button" name="button1" value="sumar" onclick="sumar()">
<input type="button" name="button2" value="restar" onclick="restar()">
<input type="button" name="button3" value="multiplicar" onclick="multiplicar()">
<input type="button" name="button4" value="dividir" onclick="dividir()"><br>
Resultado: <input type="text" name="resultado">

</form>
</body>
```

Esta línea **muestra el resultado de la operación matemática en el campo de texto llamado `resultado` del formulario** que se llama `calculadora`.

Desglose paso a paso:

Funciones, notaciones importantes:

1. Se usa la función **`parseInt()`**, que es una función integrada en JavaScript para **convertir texto (strings) a números enteros**.

***¿Qué hace exactamente `parseInt()`?**

`parseInt(valor)`:

- Toma un **string** (texto) y trata de **convertirlo en un número entero**.
- Si el valor no es un número válido, devuelve `NaN` (Not a Number).
- Ignora espacios en blanco al principio.
- Ignora cualquier contenido después del primer carácter no numérico.

Ejemplos básicos:

Entrada (`valor1.value`)	Resultado de `parseInt()`
`"10"`	`10`
`" 20"`	`20`
`"15abc"`	`15` (ignora el "abc")
`"abc"`	`NaN`
`""` (vacío)	`NaN`

---

document.calculadora.resultado.value = res;

Esta línea **muestra el resultado de la operación matemática en el campo de texto llamado `resultado` del formulario** que se llama 'calculadora':

1. **`document`**: hace referencia al documento HTML completo.
2. **`document.calculadora`**: accede al formulario que tiene el atributo `name="calculadora"`.

3. **`document.calculadora.resultado`**: accede al campo de entrada (`input`) que tiene el atributo `name="resultado"`, dentro del formulario.
4. **`.value`**: es el valor que se muestra en el campo de texto.
5. **`= res`**: asigna a ese campo el resultado de la operación almacenado en la variable `res`.

   * Aunque usar `document.formName.inputName` es válido, hoy en día se prefiere usar `querySelector` por ser más flexible:

   document.querySelector('input[name="resultado"]').value = res;

2. Las funciones `restar()` y `multiplicar()` hacen lo mismo pero con sus respectivas operaciones:

¡Ojo! Debería agregar validaciones para división por cero. ¿Cómo mejoraría el código?

Solución:

A) **Completar la función `dividir()`**, añadiendo manejo de división por cero:

```
function dividir() {
 let num1 = parseInt(document.calculadora.valor1.value);
 let num2 = parseInt(document.calculadora.valor2.value);
 let res;

 if (num2 === 0) {
  res = "Error: división por cero";
 } else {
  res = num1 / num2;
 }

 document.calculadora.resultado.value = res;
```

}

B) **Validar entradas**:
   a. Verificar que los valores sean realmente números antes de hacer la operación.
   b. Usar `Number.isNaN(num1)` para comprobar si la conversión fue válida.
   c.

C) **Usar `Number()` en lugar de `parseInt()`**:
   a. `parseInt()` puede dar resultados inesperados con decimales.
   b. Mejor usar `Number()` para admitir decimales.

Este código es un buen ejemplo básico de cómo crear una calculadora con JavaScript. Usa formularios, funciones y eventos `onclick` para interactuar con el usuario. Solo falta implementar y mejorar la función de división y agregar validaciones para hacerlo más robusto y funcional.

Formulario HTML

```
<form name="calculadora">
  Primer número: <input type="text" name="valor1"><br>
  Segundo número: <input type="text" name="valor2"><br>
```

- Se pide al usuario que introduzca dos números.

```
 <input type="button" name="button1" value="sumar"
onclick="sumar()">
  <input type="button" name="button2" value="restar"
onclick="restar()">
  <input type="button" name="button3" value="multiplicar"
onclick="multiplicar()">
  <input type="button" name="button4" value="dividir"
onclick="dividir()"><br>
```

- Cada botón ejecuta una función distinta al hacer clic.

```
 Resultado: <input type="text" name="resultado">
</form>
```

- Muestra el resultado de la operación.

Observaciones generales:

¿Por qué el <script> está en la cabecera (<head>) del HTML?

Colocar el <script> en la cabecera es una práctica tradicional. Se hace para que:

1. **El navegador cargue el JavaScript antes de que empiece a renderizar la página.**
2. El código esté preparado y definido antes de que el usuario interactúe con la página.

Pero esto tiene una desventaja importante:

Cuando el script está en el <head> y **se ejecuta inmediatamente**, puede ocurrir que:

El script intente acceder a elementos del DOM (como el formulario o los inputs) **antes de que se hayan cargado en el documento**, lo que causaría errores o que no funcione como se espera.

¿Cómo evitar ese problema?

Opción 1: Mover el <script> al final del <body>

```
<body>
  <!-- Tu formulario aquí -->
```

```
  <script>
    // Funciones JavaScript aquí
  </script>
</body>
```

- Esta es la opción más común hoy en día.
- El script se ejecuta **solo después de que el HTML ya se haya cargado**.

Opción 2: Usar `defer` en el `<script src="">` (si es externo)

Si usas un archivo externo (`.js`), puedes hacer esto:

```
<head>
  <script src="script.js" defer></script>
</head>
```

- `defer` hace que el script **espere a que todo el HTML se cargue antes de ejecutarse**.
- Funciona **solo con archivos externos**, no con scripts dentro del HTML.

Opción 3: Ejecutar el código solo después de que la página se haya cargado

Dentro del `<head>`, puedes usar un evento como este:

```
window.onload = function() {
  // Todo tu código aquí
}
```

Conclusión

- El script está en el `<head>` por tradición y para estar "listo desde el inicio".
- **Pero es mejor colocarlo al final del `<body>`** o usar `defer` si es externo, para asegurarte de que el DOM esté listo cuando el script intente usarlo.

Solución alternativa:

```
<head>
<script type="text/javascript">

    function calcula(operacio) {
        var num1;
        var num2;
        var res;
        num1 = parseInt(document.calculadora.valor1.value);
        num2 = parseInt(document.calculadora.valor2.value);

        if (operacio == 1) res = num1 + num2;
        if (operacio == 2) res = num1 - num2;
        if (operacio == 3) res = num1 * num2;
        if (operacio == 4) res = num1 / num2;

        document.calculadora.resultado.value = res;
    }


</script>
</head>

<body>
<form name="calculadora">

Primer número: <input type="text" name="valor1"><br>
Segundo número: <input type="text" name="valor2"><br>

<input type="button" name="button1" value="sumar" onclick="calcula(1)">
<input type="button" name="button2" value="restar" onclick="calcula(2)">
<input type="button" name="button3" value="multiplicar" onclick="calcula(3)">
<input type="button" name="button4" value="dividir" onclick="calcula(4)"><br>
Resultado: <input type="text" name="resultado">

</form>
</body>
```