**Manual de uso: Calculadora en JavaScript**

El programa se trata de una calculadora que permite realizar operaciones básicas (suma, resta, multiplicación, división y raíz cuadrada) a través de un programa que interactúa con el usuario mediante el uso de la función “prompt”. El programa también guarda un historial de operaciones válidas que se pueden consultar al final.

**Requisitos previos**

* Tener un entorno que soporte JavaScript (como un navegador web con la consola abierta o un editor de código compatible con Node.js).
* Conocimientos básicos de JavaScript (variables, funciones, control de flujo).

**Funciones principales**

1. **calculadora()**
   * Es la función principal que controla el flujo del programa.
   * Llama a otras funciones como validarInput(), operacion(), y mostrarHistorial().
2. **validarInput()**
   * Pide al usuario el tipo de operación y los números a utilizar.
   * Verifica que los datos introducidos sean válidos (números y tipo de operación).
3. **operacion(num1, num2, tipOperacion)**
   * Realiza la operación solicitada usando un switch que llama a las funciones correspondientes: suma(), resta(), multiplicacion(), division(), y raizCuadrada(). Se pasa por parámetros los números y el tipo de operación.
4. **Funciones de operaciones matemáticas**
   * **suma(x, y)**: Realiza la suma de x y y.
   * **resta(x, y)**: Realiza la resta de x y y.
   * **multiplicacion(x, y)**: Realiza la multiplicación de x y y.
   * **division(x, y)**: Realiza la división de x y y, verifica que y no sea cero, simplemente para no almacenar en el historial.
   * **raizCuadrada(x)**: Calcula la raíz cuadrada de x.
5. **mostrarHistorial()**
   * Permite al usuario ver el historial de operaciones realizadas.
   * Muestra las operaciones válidas registradas en el array historial.

**Ejecución del programa**

El programa sigue el siguiente flujo:

1. **Inicio:** Se ejecuta la función calculadora().
2. **Solicitud de operación:** Se pide al usuario que ingrese el tipo de operación.
3. **Ingreso de números:** Se solicitan los números a operar (uno o dos, según la operación).
4. **Validación de entrada:** Se verifica que los datos introducidos sean válidos.
5. **Ejecución de operación:** Si los datos son válidos, se realiza la operación.
6. **Almacenamiento en el historial:** La operación se guarda en el historial si es válida.
7. **Repetición o salida:** El usuario puede continuar realizando operaciones o salir ingresando "-1".
8. **Mostrar historial:** Al finalizar, se ofrece la opción de ver el historial de operaciones.

**Casos de uso**

* **Caso 1: Suma**
  + Usuario selecciona "suma", introduce 5 y 3.
  + Resultado mostrado: 5 + 3 = 8.
  + Operación almacenada en el historial.
* **Caso 2: División por cero**
  + Usuario selecciona "division", introduce 8 y 0.
  + Resultado: Se muestra un mensaje de error y la operación no se guarda en el historial.
* **Caso 3: Raíz cuadrada**
  + Usuario selecciona "raiz cuadrada", introduce 9.
  + Resultado mostrado: Raiz cudrada de 9 = 3.
  + Operación almacenada en el historial.

**Errores comunes**

* **Introducción de valores no numéricos:** Se muestra un mensaje de error indicando que se deben ingresar números.
* **Intento de división por cero:** Se muestra un mensaje indicando que no es posible dividir por cero.
* **Mala introducción del tipo de operación:** Se muestra un mensaje indicando que debe escribir bien la operación a realizar.

**Salida del programa**

* Para salir del programa, el usuario debe introducir "-1" cuando se le pide el tipo de operación.

**Desafíos encontrados en la tarea**

* Al introducir la **pedida de datos** en la función calculadora, no conseguí implementar un bucle que permitiese al usuario realizar operaciones sin refrescar la página, por lo que decidí introducir las pedidas de datos en la función validarInput. Una posible solución sería que validarInput() devolviese un booleano, y mediante un ciclo while, solicitar al usuario datos hasta que el booleano sea true. Al final deje la pedida de datos en validarInput() sin ningún ciclo, y el ciclo lo realizo llamando a las funciones hasta que se introduzca “-1”.
* **Gestión del historial:** Cuando el usuario introducía -1 para finalizar el programa, seguía solicitando los dos números, por lo que al no ser introducidos, le saltaba al usuario la alerta de introducirlos, y si los introducía, se guardaban en el historial como operación “-1” y la guardaba en el historial. La solución fue si el usuario introduce “-1”, realizar una condición la cual no le solicite números al usuario.
* **Gestión del historial:** Realice un if para cada operación, para así poder poner cada símbolo de la operación correspondiente. Por ejemplo, imprimía 4 suma 4 = 8, por lo que implemente un if para añadir los símbolos y así quedará mas visual, 4 + 4 = 8.