Para evitar a saída NaN e ainda de um número negativo/zero: Utilizei de uma 'function' para que verificasse os valores quando inserido pelo usuário da seguinte forma. Em toda variável (externa à da function) que utilizei para pegar dados do usuário, chamei a função nessa variável externa para que pudesse verificar o valor dentro da função. Isso foi possível por meio de outra variável criada dentro da 'function' para armazenar o valor inserido e logo a seguir de um laço 'while'. Esse laço analisa o dado dessa forma: 'enquanto 'variável da function' for um NaN ou 'variável' menor ou igual a zero, alerte para inserir dados numéricos ou valores positivos: caso não esteja nessas condições, a function retorna o valor para a variável inicial que chamou a função'. Assim, em todas as variáveis que precisei identificar o valor para ver se não é um número ou é menor ou igual a 0, como, por exemplo, a variável: 'quantidadeDePostos'(Let quantidadeDePostos = nomeDaFunção('parâmetro de texto');, eu chamo a função e ela identifica isso e me retorna o valor correto para prosseguir com o código.

Para retornar um valor que seja adaptado ao usuário, um número decimal, por exemplo, utilizei o 'parseFloat' na variável que fica dentro da função para que possa aceitar os dados específicos do usuário. Como esse foi um caso que precisei para muitas variáveis, uma vez estando na função, engloba a maioria dos casos. Único caso que usei o 'parseInt' foi para a 'quantidadeDePostos' que deve ser um valor inteiro e precisei declará-lo na variável que está fora da função.

Com a função fazendo esse trabalho, reduzi mais ou menos 14 linhas de código e, além disso, caso eu queira acrescentar mais algum outra condição, dentro da function, fica muito mais fácil e acessível, principalmente para códigos que necessitem de mais linhas para serem executáveis. A function torna esse processo mais palpável para o desenvolvedor.