CENTRO UNIVERSITÁRIO SENAI CIMATEC

PROGRAMA FORD <ENTER>

CURSO: PROGRAMADOR FRONT-END

DESAFIO 1 PARTE 2: ALTERNATIVAS PARA EVITAR A SAÍDA DE *NaN* (NOT A NUMBER)

GABRIEL MESSIAS DO ESPÍRITO SANTO VELOSO DA GLÓRIA

Salvador
2025

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Neste desafio, o objetivo foi criar uma aplicação que calcula o gasto diário com combustível a partir de informações fornecidas pelo usuário, como a distância até o trabalho, o consumo do carro e os preços dos combustíveis em diferentes postos. No entanto, uma parte essencial do desafio era garantir que os dados inseridos fossem válidos, evitando o erro NaN (Not a Number), que pode ocorrer quando o usuário digita texto, deixa a resposta vazia ou insere valores incompatíveis com os tipos esperados. Para contornar esse problema, implementei uma série de validações e mensagens de orientação, garantindo uma melhor experiência e segurança na execução do código.

2. DESENVOLVIMENTO

2.1 TRATAMENTO DE ENTRADAS INVÁLIDAS E PREVENÇÃO DE NAN (NOT A NUMBER)

A primeira etapa do script exibe um alert() com orientações claras ao usuário sobre como preencher os campos:

Figura 1: Alerta inicial para o usuário

```
alert("Você deve utilizar apenas números e pontuação para responder as perguntas, nenhuma
letra ou simbolo deve ser utilizado e não deixe nenhum espaço vazio/sem resposta.
\n\nUtilize o ponto (.) para separar as casas decimais, e não a virgula (,) !");
```

Fonte: Elaboração própria a partir do Visual Studio Code.

Essa etapa foi pensada para reduzir a chance de erro humano, como inserir letras, símbolos ou usar vírgulas para decimais (o que geraria NaN com parseFloat).

Mesmo com a orientação inicial, é importante validar cada entrada no código, já que nem sempre o usuário segue as instruções. Para isso, utiliza-se a função parseFloat() para converter o valor do prompt() em número decimal e, logo em seguida, uma estrutura condicional foi utilizada para verificar se o valor digitado é um número válido:

Figura 2: Validação com isNaN()

```
var preco = parseFloat(prompt("Qual o valor do combustível no posto " + i + "?"));

    if (isNaN(preco)) {
        alert("Valor inválido. Tente novamente.");
        i--; // decrementa o índice para refazer a pergunta ao mesmo posto
        continue; // pula para a próxima iteração
    }
```

Fonte: Elaboração própria a partir do Visual Studio Code.

Além disso, adicionei uma segunda verificação para garantir que o número seja maior que zero, já que valores negativos ou nulos também são inválidos nesse contexto:

Figura 3: Verificação para saber se var preco <= 0

```
if (preco <= 0) {
    alert("Valor inválido. Tente novamente.");
    i--;
    continue;
}
```

Fonte: Elaboração própria a partir do Visual Studio Code

Com essas duas validações combinadas com o uso de i-- e continue, garantem que a pergunta será repetida caso o usuário digite algo incorreto e o loop não avança até que um valor numérico e válido seja fornecido. A partir daí o programa evita qualquer chance de exibir ou calcular com NaN.

2.2 PARSEFLOAT() SEM RESULTAR *NaN* RESUMO DOS COMENTÁRIOS E EXEMPLOS

A função parseFloat() em JavaScript é utilizada para converter uma string em um número decimal. Quando a string começa com um caractere que não pode ser interpretado como número, o resultado da conversão é NaN (Not a Number). Para evitar esse retorno indesejado, é recomendável combinar parseFloat() com a verificação isNaN().

Exemplos de uso correto da função incluem:

- parseFloat("25.7") → retorna 25.7
- parseFloat(" 10.5") → retorna 10.5 (ignora espaços)
- parseFloat("30 reais") → retorna 30

Já exemplos como parseFloat("reais 30") ou parseFloat("abc") resultam em NaN, pois o primeiro caractere não é válido. Nesses casos, a função isNaN() pode ser utilizada para validar o valor antes de utilizá-lo.

Figura 4: isNaN() para validação na utilização do parseFloat

```
let valor = parseFloat(entrada);
if (!isNaN(valor)) {
    // valor é válido
}
```

Fonte: Elaboração própria a partir do Visual Studio Code

3. CONSIDERAÇÕES FINAIS

A implementação de verificações para evitar o retorno de valores inválidos, como NaN, é essencial para garantir a confiabilidade e estabilidade de aplicações que dependem de entrada manual de dados. Ao utilizar a função parseFloat() em conjunto com isNaN(), é possível tratar adequadamente os dados inseridos pelo usuário, impedindo que valores não numéricos ou mal formatados comprometam os cálculos do programa. Além disso, a orientação clara ao usuário, aliada a essas validações, contribui significativamente para uma melhor experiência de uso. Com essa abordagem, o código torna-se mais robusto, prevenindo erros comuns e assegurando que apenas informações válidas sejam processadas.

REFERÊNCIAS

W3SCHOOLS. **JavaScript parseFloat() Method.** W3Schools. Disponível em: https://www.w3schools.com/jsref/jsref_parsefloat.asp. Acesso em: 11 abr. 2025.

STACK OVERFLOW. **How do you check that a number is NaN in JavaScript?** Stack Overflow. Disponível em: https://stackoverflow.com/questions/2652319/how-do-you-check-that-a-number-is-nan-in-javascript. Acesso em: 11 abr. 2025.