Desafio da Sprint - Parte 2

Problemas no valor de input

Os inputs e/ou prompts, recebem valores literais, em string, para a maioria dos cálculos foi necessário convertê-los em numéricos. Entretanto, eu também queria receber valores com *vírgula* para facilitar o usário em digitar.

- Problema em verificar com NaN

Ao verificar se o valor é `NaN`, qualquer coisa que o *JavaScript* não conseguir transformar em número, será um `NaN`. Portanto, valores com vírgulas seriam um problema.

- Resolução

Verifiquei, primeiramente, a existência de vírgulas para tentar fazer uma conversão para um número real, como: `4,52 -> 4.52`.
Após isso, este novo valor **(que veio dessa função de formatação numérica)** é verificado por outra função, tal qual que verifica o valor recebido.

- Funções criadas

> Transformar Números:

```
const floatTransform = (value) => {
  let novoValor = 0;
  if (value.search(",") >= 0) {
    novoValor = value.replace(".", "")
    novoValor = novoValor.replace(",", ".")
    if (Number(novoValor))
      novoValor = parseFloat(Number.parseFloat(novoValor).toFixed(2))
  } else {
    novoValor = parseFloat(Number.parseFloat(value).toFixed(2))
  }
  return novoValor
}
```

> Verificar o valor (NaN)

```
const validInput = (value) => {
  const padrao = /[\d,\.]+$/;
```

```
  is (isNaN(value) || !pattern.test(value)) {
    alert("Valor inválido.")
    return false;
  }

  if (value.length === 0) {
    alert("Preencha o campo")
    return false;
  }

  if (Number(value) && value < 0>) {
    alert("Valores negativos proibidos")
    return false;
  }
  return true;
}
```

Após isso, consegui aplicar para todos os outros inputs e fazer os cálculos corretos para cada situação.

Sobre os links de referência

Notei nos comentários, um outro método para verificar a validade numérica de um valor, o `isNumeric()`, pelo que entendi, funciona para o mesmo propósito e do mesmo jeito, aparentemente, um usuário explicou como funciona este método de forma que destrichanva a função:

```
function IsNumeric(input) {
    return (input - 0) == input && ('' +input).trim().length > 0
}
```

Tambem notei a utilizacao do próprio `IsNaN` em várias situações de dados.