COMPUTAÇÃO UFCG	Bacharelado em Ciência de Computação Disciplina: **Programação para Web I** Turma: **001** Professor: Demetrio Gomes Mestre

Roteiro IV - Javascript ES6: Template Strings, Arrow Functions e Tipagem Dinâmica

01) Você foi contratado para desenvolver um sistema de **relatórios dinâmicos** para uma empresa. Esse relatório deve exibir informações sobre as vendas realizadas, formatadas de maneira clara e organizada. A empresa fornece um **array de objetos** com os seguintes dados de cada venda:

```
const vendas = [
    { produto: "Notebook", preco: 4500, quantidade: 3, vendedor: "Sara" },
    { produto: "Smartphone", preco: 2300, quantidade: 5, vendedor: "Matheus" },
    { produto: "Monitor", preco: 1200, quantidade: 2, vendedor: "Gabriel" },
    { produto: "Teclado Mecânico", preco: 350, quantidade: 4, vendedor: "Sara" },
    { produto: "Notebook", preco: 4500, quantidade: 6, vendedor: "Gabriel" },
    { produto: "Monitor", preco: 1200, quantidade: 3, vendedor: "Matheus" }
];
```

Objetivo:

1. **Crie uma arrow function chamada gerarRelatorio(vendas)** que recebe o array de vendas e retorna uma **string formatada** usando a representação **template strings**.
2. O relatório deve seguir este modelo:

```
Relatório de Vendas:

- Produto: Notebook
  Quantidade: 3
  Preço Unitário: R$ 4.500,00
  Total: R$ 13.500,00
  Vendedor: Sara

- Produto: Smartphone
  Quantidade: 5
  Preço Unitário: R$ 2.300,00
  Total: R$ 11.500,00
  Vendedor: Matheus
…
Total Geral: R$ 25.000,00
```

```
Total de comissão (5%):
Sara: R$ 675,00
Matheus: R$ 575,00
```

Regras adicionais:

- O preço deve ser formatado como **moeda brasileira (R$ X.XXX,XX)**.
- O relatório deve exibir **um total geral** da soma das vendas no final.
- Utilize apenas **template strings** para estruturar a saída.

Dicas:

- Utilize a função `Intl.NumberFormat()` para formatar os valores monetários:

```
const formatarMoeda = (valor) => new Intl.NumberFormat('pt-BR', { style: 'currency', currency: 'BRL' }).format(valor);
```

02) Você também foi contratado por uma plataforma de streaming que precisa enviar **e-mails automáticos** para seus clientes. Para isso, você deve criar uma função que **gera mensagens personalizadas** utilizando **template strings**. O sistema fornece um **array de objetos** com os seguintes dados de clientes:

```
const clientes = [
    { nome: "Davi", email: "davi@email.com", plano: "Premium", ativo: true },
    { nome: "Mariana", email: "mariana@email.com", plano: "Básico", ativo: false },
    { nome: "Gabriel", email: "gabriel@email.com", plano: "Padrão", ativo: true },
    { nome: "Ana", email: "ana@email.com", plano: "Premium", ativo: false },
    { nome: "Huandrey", email: "huandrey@email.com", plano: "Padrão", ativo: true }
];
```

Objetivo:
Crie uma arrow function chamada gerarEmail(cliente) que recebe um objeto representando um cliente e retorna um **e-mail formatado** usando **template strings**.
O conteúdo do e-mail deve variar conforme o status do cliente:

- Se o cliente **está ativo**, a mensagem deve agradecer pela assinatura e informar o plano atual.
- Se o cliente **está inativo**, a mensagem deve incentivá-lo a reativar a assinatura.

Os e-mails devem seguir este formato:
Para clientes ativos:

```
Para: davi@email.com
Olá, Davi!

Obrigado por ser um assinante do nosso plano Premium! Estamos felizes em tê-lo conosco.

Caso precise de suporte, estamos à disposição.
```

```
Atenciosamente,
Equipe StreamingWeb.
```

Para clientes inativos:

```
Para: mariana@email.com
Olá, Mariana!

Notamos que sua assinatura do plano Básico está inativa. Que tal voltar e aproveitar nossos conteúdos exclusivos?

Reative agora e continue sua experiência conosco!

Atenciosamente,
Equipe StreamingWeb.
```

03) A StreamingWeb solicitou que você desenvolvesse um sistema para validar e processar **dados de usuários cadastrados** em sua plataforma. No entanto, os dados vêm de **diversas fontes** (formulários, APIs externas, arquivos CSV), o que pode resultar em **tipos inconsistentes** (strings numéricas, valores null, números mal formatados, etc.).

A entrada de dados pode ser um **array de objetos**, mas alguns valores podem estar com tipos errados:

```
const usuarios = [
    { nome: "Cleciana", idade: "25", ativo: "true", saldo: "1234.56" },
    { nome: "Gustavo", idade: 30, ativo: true, saldo: 980 },
    { nome: "Rayane", idade: null, ativo: "false", saldo: "1500.90" },
    { nome: "Igor", idade: "NaN", ativo: 1, saldo: undefined },
    { nome: "Samuel", idade: "22 anos", ativo: false, saldo: "0" }
];
```

Objetivo:
Crie uma **arrow function** chamada **normalizarUsuario(usuario)** que recebe um objeto representando um usuário e retorna um novo objeto **com os tipos corrigidos**:

1. **`idade`** deve ser sempre um número inteiro (`number`) ou `null` se não puder ser convertido.
2. **`ativo`** deve ser sempre um booleano (`true` ou `false`).
3. **`saldo`** deve ser sempre um número (`number`) com **duas casas decimais** (`float`).

Ajuste os valores conforme necessário:

4. Se `idade` for uma string numérica (`"25"`), converta para `25`.
5. Se `idade` for `"NaN"` ou contiver caracteres não numéricos (`"22 anos"`), transforme em `null`.
6. Se `ativo` for `"true"`, `"false"`, `1`, `0`, converta corretamente para `true` ou `false`.
7. Se `saldo` for `undefined` ou inválido, defina como `0.00`.

Crie uma **arrow function** chamada **processarUsuario(lista)** que recebe um array de usuários, normaliza cada um deles e imprime os resultados corrigidos no console.

Saída Esperada:

```
[
    { nome: "Cleciana", idade: 25, ativo: true, saldo: 1234.56 },
    { nome: "Gustavo", idade: 30, ativo: true, saldo: 980.00 },
    { nome: "Rayane", idade: null, ativo: false, saldo: 1500.90 },
    { nome: "Igor", idade: null, ativo: true, saldo: 0.00 },
    { nome: "Samuel", idade: null, ativo: false, saldo: 0.00 }
]
```

Dicas

1. Use `parseInt()` para converter a idade, mas valide se o resultado é um número real.
2. Para garantir **duas casas decimais** no saldo, utilize `.toFixed(2)`

04) Escreva uma função arrow que recebe um valor de X e imprima a soma dos 5 pares consecutivos a partir de X, inclusive o X, se for par. Se o valor de entrada for 4, por exemplo, a saída deve ser 40, que é o resultado da operação: 4+6+8+10+12, enquanto que se o valor de entrada for 11, por exemplo, a saída deve ser 80, que é a soma de 12+14+16+18+20. Atenção: Você deve usar uma função de callback para realizar o somatório.

Entrada
O valor de X.

Saída
Imprima a saida conforme a explicação acima e o exemplo abaixo.
Entrada: 4 => Saída: 40
Entrada: 11 => Saída: 80

05) O governo dos Estados Unidos anunciou a retomada de tarifas sobre produtos brasileiros como aço e alumínio, justificando a medida com a alegação de que o Brasil estaria desvalorizando artificialmente sua moeda, afetando a competitividade norte-americana.

Com base nesse contexto, considere o seguinte objeto que representa os dados de uma exportação brasileira para os EUA:

```
const exportacao = {
  paisDestino: "Estados Unidos",
  produto: {
    nome: "aço",
    valorEmDolares: 100000,
    taxaImposta: 0.25
  },
  empresa: "Siderúrgica Brasil Ltda"
};
```

Objetivo:
Utilize desestruturação para extrair:

- O nome do produto;
- O valor em dólares;
- A taxa imposta;
- O nome da empresa.

Calcule o valor final com a taxa aplicada e imprima no seguinte formato:

Produto: aço
Empresa: Siderúrgica Brasil Ltda
Valor original: US$ 100000
Taxa: 25%
Valor com tarifa: US$ 125000

`Boa atividade!`