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

Roteiro IV B - Javascript ES6

01) Um hacker invadiu o sistema ao qual você realiza o suporte, criptografou a tabela de endereço e ainda colocou ameaças (para que você pague o resgate dos dados) em vários campos, em diversas tabelas no banco de dados do sistema. Porém, a restauração do backup que você solicitou só estará pronta no dia seguinte e você precisa informar aos diversos fãs (que acessam o sistema) do famoso apresentador onde será o funeral dele. Só que o hacker não contava que você conhece o VIACEP (https://viacep.com.br/) e que pode utilizá-lo para consultar os dados do endereço do Silvio Santos internamente pelo sistema. Sabendo que o CEP da mansão Abravanel é 05650-000, preencha os dados alterados pelo hacker a partir da consulta ao VIACEP. No sistema, você precisa gerar os seguintes campos usando o operador de desestruturação: proprietario - CEP - bairro, cidade (lat, lng), ou seja, mostre no console o resultado usando o recurso Template String.

Saída esperada: Silvio Santos - 05650-000 - Morumbi, São Paulo (-23.61919020307765, -46.70793551534256)

Código:

```
const dono = {
        "proprietario": "Silvio Santos",
        "endereco": {
                "cep":'hacked, pay to recover',
                "logradouro": 'hacked, pay to recover',
                "complemento": 'hacked, pay to recover',
                "bairro": 'hacked, pay to recover',
                "localidade": 'hacked, pay to recover',
                "uf": '',
                "geo": {
                        "lat": "-23.61919020307765",
                        "lng": "-46.70793551534256"
                }
        }
}

const resultado = `Coloque seu código aqui`;

console.log(resultado);
```

02) Dado um array de frases, crie uma função *analisarTexto* que recebe uma callback function e executa diferentes operações (desenvolva as operações *contarPalavras*, *maiorFrase*) dependendo do callback fornecido.

```
const frases = [
    "JavaScript é poderoso!",
    "Callbacks são úteis.",
    "Arrow functions são mais curtas."
];


const analisarTexto = (array, callback) => {
    // TODO
};

// Callback que conta o total de palavras em todas as frases
const contarPalavras = (array) => {
    // TODO
};

// Callback que encontra a frase com mais palavras
const maiorFrase = (array) => {
    // TODO
};

console.log(analisarTexto(frases, contarPalavras));
console.log(analisarTexto(frases, maiorFrase));
```

03) Crie uma função *gerarRelatorio* que recebe um array de objetos representando funcionários e gera um relatório formatado usando template strings.

Entrada:

```
const funcionarios = [
    { nome: "Ana", cargo: "Desenvolvedora", salario: 7000 },
    { nome: "Carlos", cargo: "Gerente", salario: 12000 },
    { nome: "Beatriz", cargo: "Analista", salario: 5000 }
];

const gerarRelatorio = // TODO

console.log(gerarRelatorio(funcionarios));
```

Saída:

```
Relatório de Funcionários
-----------------------------------
Nome: Ana - Cargo: Desenvolvedora - Salário: R$ 7.000,00
Nome: Carlos - Cargo: Gerente - Salário: R$ 12.000,00
Nome: Beatriz - Cargo: Analista - Salário: R$ 5.000,00
-----------------------------------
Total de funcionários: 3
Salário médio: R$ 8.000,00
```

04) Dado um JSON contendo informações de vendas, crie uma função que:

1. Converte o JSON para um array de objetos.
2. Filtra apenas as vendas acima de um determinado valor.
3. Retorna o total de vendas após o filtro.

```
const jsonVendas = `[
    {"produto": "Notebook", "valor": 4500},
    {"produto": "Smartphone", "valor": 2500},
    {"produto": "Tablet", "valor": 1800},
    {"produto": "Monitor", "valor": 1200}
]`;

const filtrarVendas = (json, minimo) => {
    // TODO
};

console.log(filtrarVendas(jsonVendas, 2000));
```

Saída:

```
{
    totalVendas: 2,
    vendas: [
        { produto: "Notebook", valor: 4500 },
        { produto: "Smartphone", valor: 2500 }
    ]
}
```

05) Implemente um sistema de gerenciamento de clientes onde:

1. Os clientes são armazenados como JSON.
2. É possível adicionar um novo cliente.
3. É possível buscar clientes pelo nome.
4. Os resultados são formatados com template strings.

```
let clientesJSON = `[
    {"nome": "Lucas", "idade": 30, "email": "lucas@email.com"},
    {"nome": "Mariana", "idade": 25, "email": "mariana@email.com"}
]`;

const adicionarCliente = (json, nome, idade, email) => {
    //TODO
};
```

```
const buscarCliente = (json, nome) => {
    //TODO
};

// Testando...
clientesJSON = adicionarCliente(clientesJSON, "Roberto", 40, "roberto@email.com");
console.log(buscarCliente(clientesJSON, "Mariana"));
```

Saída Esperada:

```
Cliente encontrado:
Nome: Mariana
Idade: 25
Email: mariana@email.com
```

06) Desenvolva uma função que recebe uma lista de números inteiros e precisa aplicar as seguintes operações:

1. Filtrar apenas os números pares.
2. Aplicar uma função de criptografia cada número par.
3. Retornar o resultado como uma nova lista.

Você deve implementar a função processarNumeros(numeros, callbackFunction) que receberá uma lista de números como primeiro argumento e uma *callback function* como segundo argumento.

A função processarNumeros deverá:

- Filtrar os números pares usando uma *arrow function*.
- Aplicar uma função de criptografia cada número par, usando outra *arrow function*.

Regras:

1. Não utilize funções tradicionais como function() {...}. Apenas *arrow functions*.
2. Não utilize laços como for, while, etc. Use métodos como filter, map, etc.

Código para criptografia:

```
const crypto = require('crypto');

// Criptografia de dados
const criptografarMensagem = (texto, chaveSecreta) => {
    const algorithm = 'aes-256-cbc';
    const iv = crypto.randomBytes(16);
    const cipher = crypto.createCipheriv(algorithm, Buffer.from(chaveSecreta), iv);
    let encrypted = cipher.update(texto, 'utf8', 'hex');
    encrypted += cipher.final('hex');
    // Retorna o IV junto com o texto criptografado
    return `${iv.toString('hex')}:${encrypted}`;
}
```

```
// Função para descriptografar dados
const decritografar = (textoCriptografado, chaveSecreta) => {
    const algorithm = 'aes-256-cbc';
    const [ivHex, encrypted] = textoCriptografado.split(':');
    const iv = Buffer.from(ivHex, 'hex');
    const decipher = crypto.createDecipheriv(algorithm, Buffer.from(chaveSecreta), iv);
    let decrypted = decipher.update(encrypted, 'hex', 'utf8');
    decrypted += decipher.final('utf8');
    return decrypted;
}
```

Boa Atividade.