INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUÍ	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ** **Curso: Análise e Desenvolvimento de Sistemas** **Disciplina: Programação para a Internet I** **Professor: Ely**

Exercício DOM 02

1) Considere os seguintes trechos de código para exibir uma mensagem de erro por 3 segundos e use uma variação do exemplo para os erros emitidos nas próximas questões:

HTML:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src = "scripts.js"></script>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <button id="botaoErro">Mostrar Erro</button>
    <div id="mensagemErro" class="oculto">O campo deve ser preenchido</div>

</body>
</html>
```

CSS:

```css
.oculto {
    display: none;
}

#mensagemErro {
    color: red;
    margin-top: 10px;
}
```

JS:

```
document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('botaoErro').addEventListener('click', function() {
        var errorMessage = document.getElementById('mensagemErro');
        errorMessage.classList.remove('oculto');

        setTimeout(function() {
            errorMessage.classList.add('oculto');
        }, 3000);
    });
});
```

https://www.w3schools.com/howto/howto_js_snackbar.asp

Refatore o código de forma que seja possível passar uma mensagem de erro específica e que possa ser passado o id do componente HTML que irá receber a mensagem de erro.

2) Considere o seguinte trecho de código onde obtemos o valor de um campo de texto e o exibimos em uma div:

HTML

```
<body>
    <label for="caixaDeTexto">Digite o texto:</label>
    <input type="text" id="caixaDeTexto">
    <button id="botaoExibir">Exibir</button>
    <div id="conteudo"></div>
</body>
```

JS:

```
document.addEventListener('DOMContentLoaded', function () {
    var botaoExibir = document.getElementById('botaoExibir');
    botaoExibir.addEventListener('click', exibirConteudo);
});

function exibirConteudo() {
    var conteudo = document.getElementById('caixaDeTexto').value;
```

```
    document.getElementById('conteudo').innerHTML = conteudo;
}
```

Altere o código anterior validando se o campo foi preenchido:

a) Retire os espaços usando a função trim() das strings e faça um if testando se a string resultante é nula/vazia;

b) Sinalize que o conteúdo do campo não pode ser vazio usando a função de exibir mensagens de erro da questão anterior.

2) Crie uma página que permita aos usuários calcular a taxa de engajamento em uma postagem de rede social. A página deve conter os seguintes elementos:

a) Dois campos de entrada:

i) Número de Interações (Curtidas, Comentários, Compartilhamentos): O número total de interações na postagem;

ii) Número de Visualizações: O número total de visualizações da postagem.

b) Um botão para calcular a taxa de engajamento;

c) Um componente para exibir mensagens de erro;

d) Uma div para exibir o resultado da taxa de engajamento em porcentagem.

Implemente também a validação dos campos de entrada para garantir que apenas números sejam aceitos, utilizando a função isNaN(). Se o usuário inserir valores inválidos, exiba uma mensagem de erro solicitando valores numéricos válidos.

Fórmula da Taxa de Engajamento:

$$\text{Taxa de Engajamento} = \left(\frac{\text{Número de Interações}}{\text{Número de Visualizações}}\right) \times 100$$

3) Crie uma página web que permita aos usuários carregarem uma imagem utilizando um componente de upload de arquivo. A página deve conter os seguintes elementos:

a) Um componente de upload de arquivo (input type="file") onde o usuário pode selecionar a imagem.

b) Uma div com o id "resultado" onde a imagem será exibida.

c) Um botão que, ao ser clicado, carrega a imagem.

Para implementar essa funcionalidade, considere os seguintes pontos:

a) Utilize a função document.createElement() para criar a tag img.

b) b) Altere o atributo src da imagem usando a propriedade img.src = URL.createObjectURL(file), onde file é o arquivo selecionado pelo usuário.

c) c) Adicione a imagem à div com id="resultado" utilizando resultado.appendChild(img).

https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement

4) Crie um outro exemplo de carregamento de imagens em que o endereço de 3 imagens estejam em elementos de um select previamente configurado. Ao mudar o conteúdo do componente, a imagem específica deve ser carregada usando um evento chamado on change.

https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/change_event

5) Considere o trecho HTML abaixo:

```
<div>
    <input type="checkbox" id="facebook" name="redesSociais" value="Facebook">
    <label for="facebook">Facebook</label>
</div>
<div>
    <input type="checkbox" id="twitter" name="redesSociais" value="Twitter">
    <label for="twitter">Twitter</label>
</div>
<div>
    <input type="checkbox" id="instagram" name="redesSociais" value="Instagram">
    <label for="instagram">Instagram</label>
</div>
<div>
    <input type="checkbox" id="linkedin" name="redesSociais" value="LinkedIn">
    <label for="linkedin">LinkedIn</label>
```

```
</div>
<div>
    <input type="checkbox" id="tiktok" name="redesSociais" value="TikTok">
    <label for="tiktok">TikTok</label>
</div>
<div class = "oculto" id="mensagemErro"></div>
<div id="redesSelecionadas"></div>
<button type="button" id="enviarBtn">Enviar Seleção</button>
```

Implemente uma funcionalidade que permita aos usuários selecionar suas redes sociais favoritas. Ao clicar no botão, deve ser validado se pelo menos um checkbox foi marcado. Caso contrário, uma mensagem com as redes selecionadas deve ser exibida

Para implementar essa funcionalidade, utilize a função document.getElementsByName(nome) e percorra o array de componentes retornado testando a propriedade checked.

6) Implemente uma página web onde o usuário pode digitar uma hashtag em uma caixa de texto e clicar em um botão para adicioná-la a uma lista de hashtags populares. A lista deve ser exibida em um componente select com o atributo size=5, permitindo que o usuário visualize até cinco hashtags de uma vez.

   Para implementar essa funcionalidade, utilize a função document.createElement() para criar tags option e a função appendChild para adicionar a opção criada ao select.

7) Altere a questão anterior para não permitir:
   a) hashtags repetidas;
   b) hashtags vazias;
   c) hashtags com comprimento menor que 2 caracteres;
   d) mais que de 5 hashtags.

8) Ainda na questão anterior, crie um botão que remova uma hashtag selecionada do select. Para isso, utilize a propriedade selectedOptions e a função removeChild do componente select

   https://developer.mozilla.org/en-US/docs/Web/API/HTMLSelectElement/selectedOptions
   https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild

9) Considere o código HTML abaixo:

```
<h1>Mover Ativos Financeiros entre Carteiras</h1>
<div class="container">
    <div>
        <label for="ativosDisponiveis">Ativos Disponíveis:</label>
        <select id="ativosDisponiveis" size="10" multiple>
            <option value="AAPL">Apple (AAPL)</option>
            <option value="GOOGL">Google (GOOGL)</option>
            <option value="AMZN">Amazon (AMZN)</option>
            <option value="MSFT">Microsoft (MSFT)</option>
            <option value="TSLA">Tesla (TSLA)</option>
            <option value="BRK.B">Berkshire Hathaway (BRK.B)</option>
            <option value="FB">Meta (FB)</option>
            <option value="V">Visa (V)</option>
            <option value="JNJ">Johnson & Johnson (JNJ)</option>
            <option value="WMT">Walmart (WMT)</option>
        </select>
    </div>
    <div class="buttons">
        <button id="moverParaDireitaBtn">></button>
        <button id="moverParaEsquerdaBtn"><</button>
    </div>
    <div>
        <label for="carteiraInvestimentos">Carteira de
Investimentos:</label>
        <select id="carteiraInvestimentos" size="10" multiple></select>
    </div>
</div>
```

Implemente uma página web onde o usuário pode mover ativos financeiros entre duas carteiras: "Ativos Disponíveis" e "Carteira de Investimentos". A página deve conter dois componentes select e dois botões para mover os ativos selecionados de um select para o outro. Observe que ambos os componentes possuem seleção múltipla.

10) Modifique a questão anterior para que:

a) Realize uma validação para garantir que pelo menos um item esteja selecionado antes de tentar movê-lo.

b) Habilite/desabilite botões de forma lógica quando um select estiver cheio/vazio;

c) Crie um leiaute que coloque os botões entre os selects.

11) Considere o código abaixo representando o esboço de uma página de gerenciamento de tarefas:

HTML:

```
<h1>Gerenciamento de Tarefas</h1>
    <div>
        <label for="descricaoTarefa">Descrição da Tarefa:</label>
        <input type="text" id="descricaoTarefa" placeholder="Ex: Concluir relatório">
    </div>
    <button id="adicionarBtn">Adicionar Tarefa</button>
    <table id="tabelaTarefas">
        <thead>
            <tr>
                <th>ID</th>
                <th>Descrição</th>
                <th>Data de Início</th>
                <th>Data de Conclusão</th>
                <th>Ações</th>
            </tr>
        </thead>
        <tbody>
            <!-- Linhas da tabela serão adicionadas aqui -->
        </tbody>
    </table>
```

CSS:

```
body {
    font-family: Arial, sans-serif;
    margin: 20px;
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
}

div {
    margin-bottom: 10px;
}

input {
    padding: 10px;
```

```
    font-size: 16px;
    width: calc(100% - 22px);
    box-sizing: border-box;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-top: 10px;
}

button:hover {
    background-color: #0056b3;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

th, td {
    padding: 10px;
    border: 1px solid #ddd;
    text-align: left;
}

th {
    background-color: #f4f4f4;
}

button.concluirBtn {
    padding: 5px 10px;
    font-size: 14px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-right: 5px;
}

button.concluirBtn:hover {
    background-color: #218838;
```

```
}

button.excluirBtn {
    padding: 5px 10px;
    font-size: 14px;
    background-color: #FF4136;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button.excluirBtn:hover {
    background-color: #E00000;
}
```

Implemente uma página onde o usuário pode adicionar tarefas a uma lista de tarefas. A página deve conter um campo de entrada para a descrição da tarefa, um botão para adicionar uma nova linha à tabela e um campo de data de início que será a data de inclusão da tarefa. Cada linha na tabela deve ter um botão de concluir para registrar a data de conclusão e um botão de excluir para remover a linha correspondente.

Orientações e fluxo básico do script que manipula o D.O.M da página:

a) Configuração Inicial:

- Espera pelo carregamento completo do DOM com DOMContentLoaded.
- Criar uma estrutura minimamente aceitável para representar uma tarefa. Ex:

```
var tarefa = {
        id: idCounter++,
        descricao: descricaoTarefa,
        dataInicio: dataInicio,
        dataConclusao: ""
    };
```

- Inicializa variáveis para armazenar referências aos elementos do DOM, um array para tarefas e um contador de ID.

b) Adiciona Evento de Clique no Botão de Adicionar:

- Obtém e valida a descrição da tarefa do campo de entrada.
- Obtém a data de início como a data atual.

- Cria um objeto de tarefa com ID, descrição, data de início e data de conclusão vazia.
- Adiciona a tarefa ao array de tarefas.
- Chama a função para adicionar a linha correspondente na tabela.

c) Função para Adicionar Linhas na Tabela:

- Cria uma nova linha na tabela e define um atributo chamado data-id com o ID da nova tarefa:

```
novaLinha.setAttribute('data-id', tarefa.id);
```

- Adiciona células na linha para o ID, descrição, data de início, data de conclusão e ações.
- Preenche as células com os dados da tarefa.
- Adiciona botões de concluir e excluir na célula de ações.
- Adiciona eventos de clique aos botões de concluir e excluir.

d) Função para Concluir Tarefas:

- Obtém a tarefa correspondente pelo ID:

```
var linha = tabelaTarefas.querySelector(`[data-id='${id}']`);
```

- Define a data de conclusão como a data atual.
- Atualiza a célula da data de conclusão na tabela.

e) Função para Excluir Tarefas:

- Remove a tarefa do array pelo ID, usando o querySelector como anteriormente.
- Remove a linha correspondente da tabela.

12) Adicione as seguintes funcionalidades/validações:

a) Adicionar a funcionalidade reabrir tarefa;
b) Não permitir excluir tarefas finalizadas;
c) Não permitir incluir tarefas sem descrição;
d) Exibir uma caixa de confirmação antes de excluir uma tarefa.