Disciplina: Desenvolvimento Web Avaliação: AS	Matrícula	Nota
Professor: Jonh Carvalho Nome do aluno:		Ass. Professor

Questão 1: **(1 ponto)**

FGV - 2024 - AL-SC - Relacione as tags semânticas incorporadas à linguagem HTML5 com suas respectivas aplicações.

1. <article>

2. <aside>

3. <header>

4. <nav>

( ) Utilizado quando precisamos representar um agrupamento de links de navegação.

( ) Utilizado quando precisamos declarar um conteúdo que não precisa de outro para fazer sentido em um documento HTML.

( ) Utilizado para representar o cabeçalho de um documento ou seção declarado no HTML.

( ) Utilizado quando precisamos criar um conteúdo de apoio/adicional ao conteúdo principal.

Assinale a opção que indica a relação correta, na ordem apresentada.

A) 4 – 3 – 2 – 1.

B) 1 – 3 – 2 – 4.

C) 4 – 1 – 3 – 2.

D) 3 – 2 – 4 – 1.

E) 2 – 4 – 1 – 3.

Questão 2: **(1 ponto)**

Qual das seguintes propriedades CSS é usada para centralizar horizontalmente um elemento dentro de um contêiner flexível?

A) align-items: center;

B) justify-content: center;

C) text-align: center;

D) margin: auto;

Questão 3: **(1 ponto)**

Qual das seguintes propriedades CSS pode ser usada para criar uma transição suave entre diferentes estados de um elemento, e como você especificaria uma transição de 2 segundos para a propriedade background-color?

A) transition: background-color 2s;

B) animation: background-color 2s;

C) transform: background-color 2s;

D) transition: 2s background-color;

Questão 4: **(1 ponto)**

CS-UFG – 2024 - Analise o código a seguir.

Isto é um exemplo

Qual é a sintaxe correta, em JavaScript, para mudar o conteúdo do elemento HTML especificado no código?

A) document.getElementById(“exemplo”).innerHTML = “Isto é outro exemplo”;

B) @demo.innerHTML = “Isto é outro exemplo”;

C) document.getElementByName(“exemplo”).innerHTML = “Isto é outro exemplo”;

D) document.getElement(“p”).innerHTML = “Isto é outro exemplo”;

E) document.getElementByTag(“exemplo”).innerHTML = “Isto é outro exemplo”;

Questão 5: **(1 ponto)**

Qual das seguintes opções descreve corretamente o uso do método addEventListener em JavaScript?

A) addEventListener é usado para adicionar um novo elemento ao DOM.

B) addEventListener é usado para remover um elemento do DOM.

C) addEventListener é usado para anexar um manipulador de eventos a um elemento específico.

D) addEventListener é usado para modificar o estilo de um elemento.

Questão 6: (1 ponto)

Com base nos requisitos da AP2 para "Busca na API" e "Construção de Interfaces via Javascript", qual é a abordagem correta para consumir uma API externa e exibir os dados dinamicamente em uma tabela HTML?

A) Usar apenas innerHTML para inserir dados diretamente no HTML, sem validação ou tratamento de erros.

B) Utilizar fetch() para consumir a API, tratar erros adequadamente, e usar createElement() junto com appendChild() para criar elementos DOM de forma dinâmica e organizada.

C) Fazer requisições síncronas com XMLHttpRequest e inserir os dados usando document.write().

D) Consumir a API apenas no carregamento da página, sem possibilidade de atualização dinâmica dos dados.

Questão 7: (4 pontos)

Cenário:

Na página "Investimentos" (ou "Dados Econômicos"), você já está consumindo uma API para buscar e exibir uma lista de diferentes tipos de investimentos (ou indicadores) em uma tabela HTML, que é construída dinamicamente com JavaScript. Para melhorar a experiência do usuário, o cliente solicitou a adição de uma funcionalidade de filtro que permita aos usuários encontrar rapidamente os itens de seu interesse.

Requisitos da Funcionalidade:

1. **Elemento de Filtro:**
   - Acima da tabela de investimentos, adicione um campo de entrada de texto (<*input type="text"*>).
   - Este campo deve ter um id claro (ex: filtro-investimentos) e um placeholder instrutivo (ex: "Buscar por nome...").

2. **Lógica de Filtragem em Tempo Real:**

   - Utilizando JavaScript, adicione um event listener ao campo de entrada que seja acionado a cada vez que o usuário digitar (sugestão: evento input).
   - A função associada ao evento deve obter o valor digitado pelo usuário.
   - Filtre o array de objetos (os dados originais recebidos da API) para encontrar os itens cujo nome (ou outra propriedade relevante) contenha o texto digitado. A busca não deve diferenciar maiúsculas de minúsculas (case-insensitive).

3. **Atualização da Tabela (Manipulação do DOM):**

- A tabela na interface do usuário deve ser limpa e recriada dinamicamente para refletir apenas os resultados do filtro.
- A filtragem não deve realizar uma nova requisição à API a cada tecla pressionada; ela deve operar sobre os dados já carregados na primeira vez.
- Se o campo de filtro estiver vazio, a tabela deve voltar a exibir todos os investimentos originais.
- Se a busca não retornar nenhum resultado, a tabela deve exibir uma única linha com uma mensagem amigável, como "Nenhum resultado encontrado.".

Critérios de Avaliação:

- Estrutura HTML: Inclusão correta do elemento <input> com os atributos necessários. **(0,5 pontos)**
- Manipulação de Eventos: Uso correto do addEventListener para capturar a entrada do usuário em tempo real. **(1,0 pontos)**
- Lógica de Programação: Implementação correta e eficiente da lógica de filtro no array de dados (usando filter, toLowerCase, includes, etc.). **(1,5 pontos)**
- Manipulação do DOM: Capacidade de limpar e renderizar novamente o corpo da tabela (<tbody>) de forma eficiente para exibir os resultados filtrados e as mensagens de estado (vazio ou sem resultados). **(1,0 pontos)**

Exemplo de Estrutura

No seu arquivo HTML:

```
<!-- ... outros elementos da página ... -->

<div class="container-filtro">
    <label for="filtro-investimentos">Buscar Investimento:</label>
    <input type="text" id="filtro-investimentos" placeholder="Digite o nome do
investimento...">
</div>

<table id="tabela-investimentos">
    <thead>
        <tr>
            <th>Nome</th>
            <th>Tipo</th>
            <th>Rentabilidade Anual</th>
        </tr>
    </thead>
    <tbody>
        <!-- Linhas da tabela serão inseridas aqui via JavaScript -->
    </tbody>
</table>

<!-- ... outros elementos da página ... -->
```

AS/
├── filtro.html

```
├── filtro.js
└── style.css
```

No seu arquivo JavaScript, a lógica seria adicionada para interagir com esses novos elementos.

```
    // Aguarda o carregamento completo do DOM antes de executar o script

    // --- DADOS (Resposta de uma API) ---
    // Em um cenário real, esta variável seria preenchida com os dados de um `fetch`.

    // --- SELEÇÃO DE ELEMENTOS DO DOM ---


    // Função responsável por renderizar as linhas da tabela na interface.
    // O array de objetos de investimento a ser exibido.

    // 1. Limpa o conteúdo atual do corpo da tabela para evitar duplicatas

    // 2. Verifica se o array de dados está vazio

    // 3. Itera sobre os dados e cria uma linha (<tr>) para cada item

    // Adiciona as células à linha

    // Adiciona a linha completa ao corpo da tabela

    // --- LÓGICA DO FILTRO ---

    // Adiciona um 'event listener' que dispara a cada vez que o usuário digita no campo

    // Obtém o termo de busca, remove espaços em branco e converte para minúsculas

    // Filtra o array original de investimentos

    // Retorna true se o nome do investimento (em minúsculas) incluir o termo de busca

    // Chama a função para redesenhar a tabela com os dados filtrados

    // --- RENDERIZAÇÃO INICIAL ---           // Exibe todos os investimentos na tabela
quando a página é carregada pela primeira vez
```