ibmec

Nome do Aluno: _____		**Matrícula:** _____	**Nota:** _____
Disciplina: Desenvolvimento Web	**Período:**	**Prova:**	
Professor: Jonh Carvalho			**Ass. Professor**
Curso:		**Data:**	**2025.1**

ATENÇÃO!

1. Se a prova for escrita a lápis, total ou parcialmente, o aluno não terá direito a solicitar revisão. As folhas de rosto e de perguntas deverão ser devolvidas juntamente com a prova.
2. Apenas os alunos cujos nomes constem na lista de freqüência (ou seja, que estejam regularmente matriculados na disciplina) podem realizar esta prova.
3. Quando solicitado, o aluno deverá apresentar documento de identificação (carteira de identidade ou de habilitação) para que possa realizar a prova.
4. Não é permitido sair de sala nos primeiros 30 minutos do horário da prova. Após esse tempo, o aluno que sair de sala estará necessariamente concluindo a sua prova.
5. Ao terminar a prova, o aluno deve permanecer sentado e solicitar ao professor que a recolha.
6. A lista de freqüência deve ser assinada pelo aluno no momento em que entregar a prova ao professor.
7. Após a correção, apenas o próprio aluno poderá examinar ou receber a prova. No ato da entrega da prova pelo professor, o aluno deverá assinar a lista de freqüência.

Secretaria Acadêmica

Declaro estar ciente dos procedimentos acima

_____

Assinatura do aluno

1. Um desenvolvedor está criando um dashboard responsivo para um sistema de analytics. O layout deve incluir:

a) Um **cabeçalho** fixo com logo e menu de usuário

b) Uma **barra lateral** (sidebar) com menus de navegação

c) Uma **área principal** com:

- Um painel de resumo (4 cards em grid)
- Um gráfico de linha
- Uma tabela de dados

d) Um **rodapé** com informações de copyright

Requisitos:

- A área principal deve usar **CSS Grid** para organização macro (definição de áreas)
- Os cards no painel de resumo devem usar **Flexbox** para alinhamento interno
- O layout deve ser responsivo, colapsando a sidebar em mobile

Questão: Ao implementar o CSS, qual abordagem atende **MELHOR** aos requisitos de organização macro/micro e responsividade?

```
<div class="dashboard">
  <header>...</header>
  <nav>...</nav>
  <main>
    <div class="resumo">
      <div class="card">...</div> <!-- 4 cards -->
    </div>
    <div class="grafico">...</div>
    <div class="tabela">...</div>
  </main>
  <footer>...</footer>
</div>
```

A)

```
.dashboard {
  display: flex;
}
.resumo {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
}
.card {
  display: block;
}
```

B)

```
.dashboard {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 250px 1fr;
}
main {
  grid-area: main;
  display: flex;
  flex-direction: column;
}
.resumo {
  display: flex;
  gap: 20px;
}
.card {
  flex: 1;
}
@media (max-width: 768px) {
  .dashboard {
    grid-template-areas:
      "header"
      "main"
      "footer";
    grid-template-columns: 1fr;
  }
}
```

C)

```
.dashboard {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
}
.resumo {
  grid-column: span 12;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
}
```

D)

```
.dashboard {
  display: flex;
  flex-wrap: wrap;
}
main {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
}
.resumo {
  display: flex;
}
```

E)

```
.dashboard {
  position: relative;
}
.resumo {
  display: flex;
  justify-content: space-between;
}
.card {
  float: left;
  width: 23%;
}
```

2. Um desenvolvedor está criando um sistema de gerenciamento de produtos para um e commerce. Cada produto é representado como um objeto com as propriedades:

- `id` (número)
- `nome` (string)
- `preco` (número)
- `categoria` (string)
- `estoque` (número)

O sistema possui um array `produtos` com 100 itens no seguinte formato:

```
const produtos = [
  { id: 1, nome: "Teclado Mecânico", preco: 250, categoria: "periféricos", estoque:
 50 },
  { id: 2, nome: "Monitor 4K", preco: 1200, categoria: "monitores", estoque: 20 },
  // ... outros produtos
];
```

Requisitos:

1. Calcular o preço médio dos produtos de uma categoria específica
2. Atualizar o estoque de um produto pelo ID
3. Encontrar o produto mais caro por categoria

Questão: Qual das seguintes alternativas implementa **corretamente e de forma mais eficiente** a função `atualizarEstoque` que recebe um `id` e `quantidade`, e atualiza o estoque do produto correspondente?

A)

```
function atualizarEstoque(id, quantidade) {
  produtos.forEach(produto => {
    if (produto.id === id) {
      produto.estoque += quantidade;
    }
  });
}
```

B)

```
function atualizarEstoque(id, quantidade) {
  const produto = produtos.find(p => p.id === id);
  produto.estoque = quantidade;
}
```

C)

```
function atualizarEstoque(id, quantidade) {
  const index = produtos.findIndex(p => p.id === id);
  if (index !== -1) {
    produtos[index].estoque += quantidade;
  }
}
```

D)

```
function atualizarEstoque(id, quantidade) {
  produtos.map(produto => {
    if (produto.id === id) {
      return { ...produto, estoque: quantidade };
    }
    return produto;
  });
}
```

E)

```
function atualizarEstoque(id, quantidade) {
  for (let i = 0; i < produtos.length; i++) {
    produtos[i].estoque = produtos[i].id === id
      ? quantidade
      : produtos[i].estoque;
  }
}
```

3. Um desenvolvedor está criando uma aplicação web para gestão de tarefas. O HTML inicial contém:

```
<ul id="lista-tarefas">
  <li class="tarefa" data-id="1">
    <span>Revisar documento</span>
    <button class="concluir">Concluir</button>
  </li>
  <li class="tarefa" data-id="2">
    <span>Enviar relatório</span>
    <button class="concluir">Concluir</button>
  </li>
</ul>
<input type="text" id="nova-tarefa">
<button id="adicionar">Adicionar Tarefa</button>
```

Requisitos funcionais:

1. Adicionar novas tarefas ao digitar no input e clicar no botão
2. Marcar tarefas como concluídas ao clicar no botão "Concluir"
3. Validar se o input não está vazio antes de adicionar
4. Remover visualmente a tarefa concluída após 2 segundos

Questão: Qual implementação JavaScript **atende completamente** aos requisitos?

A)

```
document.getElementById('adicionar').addEventListener('click', () => {
  const input = document.getElementById('nova-tarefa');
  if (input.value.trim() === '') return;

  const ul = document.getElementById('lista-tarefas');
  ul.innerHTML += `
    <li class="tarefa">
      <span>${input.value}</span>
      <button class="concluir">Concluir</button>
    </li>
  `;
  input.value = '';
});
```

B)

```
// Adicionar tarefa
document.getElementById('adicionar').addEventListener('click', () => {
  const input = document.getElementById('nova-tarefa');
  if (!input.value.trim()) return;
  const li = document.createElement('li');
  li.className = 'tarefa';
  li.innerHTML = `
    <span>${input.value}</span>
    <button class="concluir">Concluir</button>
```

```
  `;

  document.getElementById('lista-tarefas').appendChild(li);
  input.value = '';
  // Concluir tarefa
  li.querySelector('.concluir').addEventListener('click', () => {
    li.style.textDecoration = 'line-through';
    setTimeout(() => li.remove(), 2000);
  });
});
```

C)

```
// Adicionar tarefa
document.querySelector('#adicionar').onclick = function() {
  const texto = document.querySelector('#nova-tarefa').value;
  document.querySelector('#lista-tarefas').insertAdjacentHTML(
    'beforeend',
    `<li><span>${texto}</span><button>Concluir</button></li>`
  );
};
// Concluir tarefa
document.addEventListener('click', (e) => {
  if (e.target.classList.contains('concluir')) {
    e.target.parentElement.remove();
  }
});
```

D)

```
function adicionarTarefa() {
  const input = document.getElementById('nova-tarefa');
  if (input.value) {
    const novoItem = document.createElement('li');
    novoItem.textContent = input.value;
    document.getElementById('lista-tarefas').appendChild(novoItem);
  }
}
document.querySelectorAll('.concluir').forEach(btn => {
  btn.addEventListener('click', (e) => {
    e.target.parentElement.style.opacity = '0.5';
  });
});
```

E)

```
// Adicionar tarefa
const adicionar = document.getElementById('adicionar');
adicionar.onclick = () => {
  const input = document.getElementById('nova-tarefa');
  const ul = document.getElementById('lista-tarefas');

  if (input.value) {
    const li = document.createElement('li');
    li.dataset.id = Date.now();
    li.innerHTML = `
```

```
      <span>${input.value}</span>
      <button class="concluir">Concluir</button>
    `;
    ul.appendChild(li);
    input.value = '';
  }
};
// Delegation para concluir
ul.addEventListener('click', (e) => {
  if (e.target.classList.contains('concluir')) {
    const li = e.target.closest('li');
    li.classList.add('concluida');
    setTimeout(() => li.style.display = 'none', 2000);
  }
});
```

4. Um desenvolvedor está criando um sistema de monitoramento de criptomoedas. A aplicação deve consumir a API pública CoinGecko e exibir os dados em uma tabela HTML. Os dados retornados incluem:

(https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd)

```
[
  {
    "id": "bitcoin",
    "symbol": "btc",
    "name": "Bitcoin",
    "current_price": 55000,
    "price_change_percentage_24h": 2.5,
    "market_cap": 1000000000000
  },
  // ... outros registros
]
```

Requisitos:

1. Consumir dados da API ao carregar a página

2. Exibir os dados em uma tabela com colunas: Nome, Símbolo, Preço (USD), Variação 24h e Market Cap

3. Formatar os valores numéricos:

- Preço: USD 55,000.00

- Variação: +2.50% (verde para positivo, vermelho para negativo)

- Market Cap: $1.00T

4. Implementar tratamento de erros para falhas na requisição

5. Adicionar um indicador de carregamento durante a requisição

Questão: Qual implementação JavaScript **atende corretamente** a todos os requisitos?

```
<!-- Estrutura HTML -->
<table id="crypto-table">
  <thead>
    <tr>
      <th>Nome</th>
      <th>Símbolo</th>
      <th>Preço (USD)</th>
      <th>Variação 24h</th>
      <th>Market Cap</th>
    </tr>
  </thead>
  <tbody id="crypto-body"></tbody>
</table>
<div id="loading">Carregando...</div>
<div id="error-msg" style="color:red; display:none">Erro ao carregar dados</div>
```

A)

```
async function loadData() {
  const response = await fetch('https://api.coingecko.com/api/v3/coins/markets?vs_c
urrency=usd');
  const data = await response.text();
  document.getElementById('crypto-body').innerHTML = data.map(item => `
    <tr>
      <td>${item.name}</td>
      <td>${item.symbol}</td>
      <td>$${item.current_price}</td>
      <td>${item.price_change_percentage_24h}%</td>
      <td>$${item.market_cap}</td>
    </tr>
  `).join('');
}
loadData();
```

B)

```
document.addEventListener('DOMContentLoaded', () => {
  const loading = document.getElementById('loading');
  const errorMsg = document.getElementById('error-msg');

  loading.style.display = 'block';

  fetch('https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd')
    .then(response => response.ok ? response.json() : Promise.reject())
    .then(data => {
      const tbody = document.getElementById('crypto-body');
      tbody.innerHTML = data.map(coin => `
        <tr>
          <td>${coin.name}</td>
          <td>${coin.symbol.toUpperCase()}</td>
          <td>USD ${coin.current_price.toLocaleString('en-
US', {minimumFractionDigits: 2})}</td>
          <td style="color: ${coin.price_change_percentage_24h >= 0 ? 'green' : 're
d'}">
            ${coin.price_change_percentage_24h.toFixed(2)}%
          </td>
          <td>$${(coin.market_cap / 1e12).toFixed(2)}T</td>
        </tr>
      `).join('');
    })
    .catch(() => {
      errorMsg.style.display = 'block';
    })
    .finally(() => {
      loading.style.display = 'none';
    });
});
```

C)

```
const loadCryptoData = async () => {
  try {
```

```
    const res = await fetch('https://api.coingecko.com/api/v3/coins/markets?vs_curr
ency=usd');
    const coins = await res.json();

    coins.forEach(coin => {
      const row = document.createElement('tr');
      row.innerHTML = `
        <td>${coin.id}</td>
        <td>${coin.symbol}</td>
        <td>${coin.current_price}</td>
        <td>${coin.price_change_percentage_24h}</td>
        <td>${coin.market_cap}</td>
      `;
      document.getElementById('crypto-body').appendChild(row);
    });
  } catch (e) {
    console.error(e);
  }
};
window.onload = loadCryptoData;
```

D)

```
function renderCryptoTable() {
  fetch('https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd')
    .then(response => response.json())
    .then(data => {
      const table = document.getElementById('crypto-table');
      data.forEach(item => {
        const newRow = table.insertRow();
        newRow.innerHTML = `
          <td>${item.name}</td>
          <td>${item.symbol}</td>
          <td>${item.current_price.toFixed(2)}</td>
          <td>${item.price_change_percentage_24h}</td>
          <td>${item.market_cap}</td>
        `;
      });
    });
}
renderCryptoTable();
```

E)

```
const loadData = () => {
  document.getElementById('loading').style.visibility = 'visible';

  const xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd')
;
  xhr.onload = function() {
    if (xhr.status === 200) {
      const data = JSON.parse(xhr.response);
      // ... processamento dos dados
    }
  };
  xhr.send();
```

```
};
document.addEventListener('DOMContentLoaded', loadData);
```

Bom teste!