Avaliação:

Questão 01

Listar as principais tags de uma estrutura base para construção de um documento HTML.

Questão 02

Relembrar “tag ancora”, enfatizando a ancora entre páginas (hiperkink)

Questão 03

Encontrar novas tags html

Exemplo a tag Table

A tag <table> em HTML é usada para criar uma tabela. Tabelas são usadas para exibir dados em formato de grade, com linhas e colunas. A estrutura básica de uma tabela em HTML inclui as seguintes tags:

<table>: A tag de abertura da tabela.

<tr>: Representa uma linha na tabela.

<th>: Representa um cabeçalho de coluna (opcional, usado nas primeiras linhas).

<td>: Representa uma célula de dados na tabela.

</table>: A tag de fechamento da tabela.

Aqui está um exemplo simples de uma tabela HTML:

html

Copy code

<table>

<tr>

<th>Nome</th>

<th>Idade</th>

</tr>

<tr>

<td>João</td>

<td>30</td>

</tr>

<tr>

<td>Maria</td>

<td>25</td>

</tr>

</table>

Neste exemplo, temos uma tabela com duas colunas: "Nome" e "Idade". A primeira linha (dentro das tags <tr>) é geralmente usada como cabeçalho da tabela (usando as tags <th>), e as linhas subsequentes (também dentro das tags <tr>) contêm os dados da tabela nas células <td>.

Esta é uma tabela HTML muito básica. Você pode personalizar a aparência e o comportamento da tabela usando CSS e JavaScript, se necessário.

Questão 04

Explique sobre sua pesquisa de vetores e matrizes na internet

let blooms = [];

Esta linha cria uma nova variável chamada blooms usando a palavra-chave let. Em JavaScript, a palavra-chave let é usada para declarar variáveis que podem ser alteradas (ou seja, que podem receber novos valores) posteriormente no código.

Neste caso, blooms é inicializada como um array vazio []. Um array é uma estrutura de dados em JavaScript que pode armazenar uma coleção de valores, como números, strings, objetos e assim por diante.

Portanto, a linha de código let blooms = []; cria uma variável chamada blooms que é um array vazio, pronto para armazenar elementos. Você pode adicionar elementos a esse array posteriormente no seu código usando métodos como push, splice, ou atribuindo valores diretamente a índices específicos. Por exemplo:

blooms.push("rosa"); // Adiciona "rosa" ao array blooms

blooms.push("lírio"); // Adiciona "lírio" ao array blooms

Agora, o array blooms conteria dois elementos: "rosa" e "lírio". Este é um exemplo simples de como você pode usar arrays em JavaScript para armazenar dados de maneira organizada e acessá-los quando necessário.

Questão 05

Considere o código abaixo:

// Função para cadastrar Bloom

document.getElementById('cadastroForm').addEventListener('submit', function(e) {

e.preventDefault();

const nome = document.getElementById('nome').value;

const id = blooms.length + 1;

blooms.push({ id, nome });

exibirBloom();

document.getElementById('nome').value = '';

});

Agora comente linha a linha do código:

O código é um trecho de JavaScript que lida com a submissão de um formulário com o id 'cadastroForm'. Ele parece estar relacionado ao cadastro de elementos chamados "Bloom" e utiliza manipulação do DOM para interagir com a página web. Vou explicar linha por linha:

1. document.getElementById('cadastroForm').addEventListener('submit', function(e) {
   * Aqui, estamos selecionando um elemento do DOM com o id 'cadastroForm'. O addEventListener é usado para adicionar um ouvinte de eventos a esse elemento. O evento que estamos ouvindo é o 'submit', que ocorre quando o formulário é enviado. Quando o formulário é submetido, a função definida como segundo argumento será executada.
2. e.preventDefault();
   * Dentro da função de evento, a primeira coisa que fazemos é chamar e.preventDefault(). Isso impede o comportamento padrão de submissão do formulário, que seria recarregar a página. Em vez disso, estamos assumindo o controle da submissão do formulário por meio do JavaScript.
3. const nome = document.getElementById('nome').value;
   * Esta linha obtém o valor do elemento de input com o id 'nome' e armazena-o na variável nome. Isso presumivelmente corresponde ao nome do "Bloom" que está sendo cadastrado.
4. const id = blooms.length + 1;
   * Aqui, estamos calculando o novo ID para o "Bloom" que está sendo cadastrado. Parece que blooms é um array que contém "Blooms" previamente cadastrados. Estamos usando o comprimento desse array mais 1 como o novo ID.
5. blooms.push({ id, nome });
   * Esta linha cria um objeto com as propriedades id e nome e o adiciona ao array blooms. Isso efetivamente cadastra um novo "Bloom" no array com o ID e nome fornecidos.
6. exibirBloom();
   * Essa linha chama uma função chamada exibirBloom(). Presumivelmente, essa função é responsável por atualizar a exibição na página web para mostrar o novo "Bloom" cadastrado.
7. document.getElementById('nome').value = '';
   * Finalmente, após o cadastro do "Bloom", o valor do elemento de input com o id 'nome' é definido como uma string vazia, limpando o campo de entrada para que o usuário possa inserir o próximo nome facilmente.

Em resumo, este código lida com a submissão de um formulário para cadastrar "Blooms". Ele extrai o nome do "Bloom" do formulário, gera um novo ID com base no comprimento do array existente, cadastra o "Bloom" no array, atualiza a exibição e limpa o campo de entrada do nome.

Questão 06

Considere o código abaixo:

// Função para exibir Bloom na tabela

function exibirBloom() {

const tableBody = document.getElementById('bloomTableBody');

tableBody.innerHTML = '';

blooms.forEach(bloom => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${bloom.id}</td>

<td>${bloom.nome}</td>

<td>

<button onclick="editarBloom(${bloom.id})">Editar</button>

<button onclick="excluirBloom(${bloom.id})">Excluir</button>

</td>

`;

tableBody.appendChild(row);

});

}

Explique linha a linha do código demonstrado acima:

// Função para exibir Bloom na tabela

function exibirBloom() {

// Obtém uma referência ao elemento do corpo da tabela com o id 'bloomTableBody'

const tableBody = document.getElementById('bloomTableBody');

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

tableBody.innerHTML = '';

// Itera sobre cada elemento no array 'blooms'

blooms.forEach(bloom => {

// Cria um novo elemento 'tr' (linha) para a tabela

const row = document.createElement('tr');

// Define o conteúdo HTML da linha com dados do 'bloom'

row.innerHTML = `

<td>${bloom.id}</td>

<td>${bloom.nome}</td>

<td>

<button onclick="editarBloom(${bloom.id})">Editar</button>

<button onclick="excluirBloom(${bloom.id})">Excluir</button>

</td>

`;

// Adiciona a linha criada à tabela

tableBody.appendChild(row);

});

}

Resumo:

* A função exibirBloom atualiza a tabela na página HTML com informações dos "Blooms".
* Ela começa obtendo uma referência ao elemento do corpo da tabela com o id 'bloomTableBody'.
* Em seguida, ela limpa o conteúdo atual desse elemento para preparar a tabela para uma nova exibição.
* Em um loop forEach, ela itera sobre cada "Bloom" no array blooms.
* Para cada "Bloom", ela cria uma nova linha <tr> na tabela e define o conteúdo da linha com as informações do "Bloom", incluindo ID, nome e botões "Editar" e "Excluir".
* Por fim, a linha é adicionada ao corpo da tabela usando tableBody.appendChild(row). Isso constrói visualmente a tabela com os dados atualizados dos "Blooms".

Questão 07

Considere o código:

// Função para editar Bloom

function editarBloom(id) {

// Pede ao usuário para editar o nome do "Bloom" através de um prompt e armazena o resultado em 'nome'

const nome = prompt('Editar Nome:', blooms.find(bloom => bloom.id === id).nome);

// Verifica se o usuário pressionou "Cancelar" ou deixou o campo em branco

if (nome !== null) {

// Encontra o índice do "Bloom" com o ID fornecido no array 'blooms'

const index = blooms.findIndex(bloom => bloom.id === id);

// Atualiza o nome do "Bloom" no array 'blooms' com o novo nome

blooms[index].nome = nome;

// Chama a função 'exibirBloom' para atualizar a exibição da tabela com os dados atualizados

exibirBloom();

}

}

Aqui está um resumo do que a função faz:

* A função editarBloom é chamada com um argumento id, que representa o ID do "Bloom" a ser editado.
* Ela usa a função prompt para exibir uma caixa de diálogo que permite ao usuário editar o nome do "Bloom". O valor atual do nome é recuperado usando blooms.find(bloom => bloom.id === id).nome e é exibido como valor padrão no prompt.
* A função verifica se o resultado da caixa de diálogo não é nulo (ou seja, o usuário não pressionou "Cancelar" ou deixou o campo em branco). Se o usuário forneceu um novo nome, o código continua.
* Em seguida, a função procura o índice do "Bloom" com o ID fornecido no array blooms usando blooms.findIndex(bloom => bloom.id === id).
* O nome do "Bloom" no array blooms é atualizado com o novo nome usando blooms[index].nome = nome.
* Finalmente, a função exibirBloom é chamada para atualizar a exibição da tabela com os dados atualizados após a edição do "Bloom".

Questão 08

Considere os comentários abaixo:

// Função para excluir Bloom

function excluirBloom(id) {

// Exibe uma caixa de diálogo de confirmação e armazena a resposta em 'confirmacao'

const confirmacao = confirm('Tem certeza que deseja excluir este Bloom?');

// Verifica se o usuário confirmou a exclusão

if (confirmacao) {

// Usa a função 'filter' para criar um novo array 'blooms' excluindo o "Bloom" com o ID fornecido

blooms = blooms.filter(bloom => bloom.id !== id);

// Chama a função 'exibirBloom' para atualizar a exibição da tabela após a exclusão

exibirBloom();

}

}

Aqui está um resumo do que a função faz:

* A função excluirBloom é chamada com um argumento id, que representa o ID do "Bloom" a ser excluído.
* Ela usa a função confirm para exibir uma caixa de diálogo de confirmação ao usuário, perguntando se eles têm certeza de que desejam excluir o "Bloom".
* A resposta do usuário (verdadeiro ou falso) é armazenada na variável confirmacao.
* A função verifica se o valor de confirmacao é verdadeiro (ou seja, o usuário confirmou a exclusão).
* Se o usuário confirmou a exclusão, a função usa a função filter para criar um novo array blooms que exclui o "Bloom" com o ID fornecido. Isso remove efetivamente o "Bloom" do array.
* Por fim, a função exibirBloom é chamada para atualizar a exibição da tabela, refletindo a exclusão do "Bloom". A tabela agora mostrará os "Blooms" restantes após a exclusão.

Questão 09

Faça uma pesquisa sobre os métodos:

Splice

Shift

Pop

Aplique-os no código da prova:

Segue o texto abaixo:

A função excluirBloom que você criou na prova faz a exclusão de um "Bloom" do array blooms usando a função filter. A função filter cria um novo array contendo todos os elementos que atendem a uma determinada condição. Neste caso, a condição é que o id do "Bloom" não seja igual ao id fornecido como argumento.

Aqui está a linha relevante que realiza a exclusão:

javascript

blooms = blooms.filter(bloom => bloom.id !== id);

Isso efetivamente remove o "Bloom" com o id correspondente do array blooms. O resultado da função filter é um novo array que não contém mais o "Bloom" com o id especificado.

Se você está procurando por uma função que remove um elemento de um array sem criar um novo array, você pode usar o método splice. O splice permite remover um elemento de um array existente sem criar um novo array. Aqui está como você poderia usá-lo para remover um "Bloom" com base no id:

javascript

function excluirBloom(id) {

const index = blooms.findIndex(bloom => bloom.id === id);

if (index !== -1) {

blooms.splice(index, 1);

exibirBloom();

}

}

Neste código, findIndex é usado para encontrar o índice do "Bloom" com o id especificado e, em seguida, splice é usado para remover o elemento no índice encontrado. Note que verificamos se o índice não é igual a -1 para garantir que o "Bloom" foi encontrado antes de tentar removê-lo.

Outra abordagem que você pode usar para remover um elemento de um array é o método pop(). O método pop() remove e retorna o último elemento de um array. No entanto, essa abordagem só é apropriada se você desejar remover o "Bloom" mais recentemente adicionado, que está na parte final do array. Aqui está como você pode usá-lo:

javascript

function excluirUltimoBloom() {

if (blooms.length > 0) {

blooms.pop();

exibirBloom();

}

}

Nesse código, verificamos se o array blooms não está vazio antes de tentar remover o último elemento. Se estiver vazio, a função não faz nada. Caso contrário, ela remove o último "Bloom" usando pop() e depois chama exibirBloom() para atualizar a exibição da tabela.

Lembre-se de que a escolha do método de remoção depende dos requisitos específicos do seu aplicativo e de como os "Blooms" são gerenciados no array. Você pode escolher a abordagem que melhor se adapta ao seu caso de uso.

Outra abordagem semelhante que você pode usar é o método shift(). O método shift() remove e retorna o primeiro elemento de um array. Isso só seria apropriado se você desejar remover o "Bloom" mais antigo, que está na parte inicial do array. Aqui está como você pode usá-lo:

javascript

function excluirPrimeiroBloom() {

if (blooms.length > 0) {

blooms.shift();

exibirBloom();

}

}

Nesse código, verificamos se o array blooms não está vazio antes de tentar remover o primeiro elemento. Se estiver vazio, a função não faz nada. Caso contrário, ela remove o primeiro "Bloom" usando shift() e depois chama exibirBloom() para atualizar a exibição da tabela.

Assim como a abordagem anterior com pop(), a escolha entre shift() e pop() depende dos requisitos específicos do seu aplicativo e de como os "Blooms" são gerenciados no array. Você pode escolher a abordagem que melhor se adapta ao seu caso de uso.

Além do pop() e shift() mencionados anteriormente, outro método que você pode usar para remover um elemento específico de um array é o splice(), mas com uma sintaxe diferente da que foi mencionada anteriormente. O splice() pode ser usado para remover um elemento em qualquer posição do array. Aqui está como você pode usá-lo para remover um "Bloom" com base no id:

javascript

function excluirBloom(id) {

const index = blooms.findIndex(bloom => bloom.id === id);

if (index !== -1) {

blooms.splice(index, 1);

exibirBloom();

}

}

Neste código, findIndex é usado para encontrar o índice do "Bloom" com o id especificado e, em seguida, splice é usado para remover o elemento no índice encontrado. Assim como nas abordagens anteriores, verificamos se o índice não é igual a -1 para garantir que o "Bloom" foi encontrado antes de tentar removê-lo.

Portanto, esta é outra maneira de remover um "Bloom" específico com base no id de um array. A escolha entre esses métodos depende dos requisitos específicos do seu aplicativo e da estrutura dos dados no array.

A escolha da melhor forma de excluir elementos de um array depende do contexto específico do seu aplicativo e dos requisitos. No entanto, se você está procurando uma abordagem prática e simples para iniciantes, a função splice() é uma opção sólida, pois é relativamente fácil de entender e usar. Além disso, ela pode ser usada para remover elementos em qualquer posição do array.

Aqui está um exemplo simples do uso do splice() para remover um elemento com base no índice:

javascript

function excluirBloom(index) {

blooms.splice(index, 1);

exibirBloom();

}

Neste exemplo, index representa o índice do elemento a ser removido. O segundo argumento 1 indica que apenas um elemento deve ser removido.

No entanto, se você deseja remover um elemento com base em alguma condição, como o id, você pode primeiro encontrar o índice desse elemento usando findIndex() e, em seguida, usar splice() para removê-lo, como mostrado em respostas anteriores.