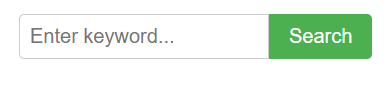
**Q1: Can you create a simple HTML search input with a button to submit a keyword?**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Simple Search</title>

<style>

.search-container {

margin: 20px;

display: flex;

}

.search-input {

padding: 8px;

font-size: 16px;

border: 1px solid #ccc;

border-radius: 4px 0 0 4px;

width: 200px;

}

.search-button {

padding: 8px 16px;

font-size: 16px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 0 4px 4px 0;

cursor: pointer;

}

.search-button:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form class="search-container" action="/search" method="GET">

<input type="text" class="search-input" name="q" placeholder="Enter keyword...">

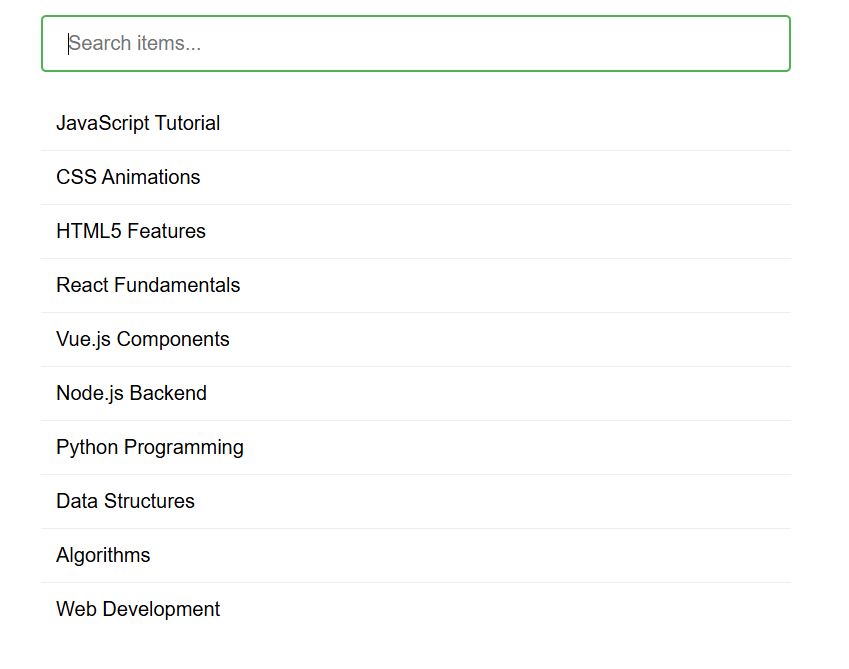
<button type="submit" class="search-button">Search</button>

</form>

</body>

</html>

**Q2: Build an HTML search bar styled with CSS that filters a list of items in real time using JavaScript as the user types.**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Real-Time Search Filter</title>

<style>

body {

font-family: Arial, sans-serif;

max-width: 600px;

margin: 0 auto;

padding: 20px;

}

.search-container {

margin-bottom: 20px;

}

#search-input {

width: 100%;

padding: 12px 20px;

font-size: 16px;

border: 2px solid #ddd;

border-radius: 4px;

box-sizing: border-box;

outline: none;

transition: border-color 0.3s;

}

#search-input:focus {

border-color: #4CAF50;

}

#items-list {

list-style-type: none;

padding: 0;

margin: 0;

}

#items-list li {

padding: 12px;

border-bottom: 1px solid #eee;

transition: all 0.3s;

}

#items-list li:last-child {

border-bottom: none;

}

#items-list li:hover {

background-color: #f9f9f9;

}

.no-results {

padding: 20px;

text-align: center;

color: #888;

font-style: italic;

}

.highlight {

background-color: #ffeb3b;

padding: 2px 0;

}

</style>

</head>

<body>

<div class="search-container">

<input

type="text"

id="search-input"

placeholder="Search items..."

aria-label="Search items"

>

</div>

<ul id="items-list">

<li>JavaScript Tutorial</li>

<li>CSS Animations</li>

<li>HTML5 Features</li>

<li>React Fundamentals</li>

<li>Vue.js Components</li>

<li>Node.js Backend</li>

<li>Python Programming</li>

<li>Data Structures</li>

<li>Algorithms</li>

<li>Web Development</li>

</ul>

<script>

document.addEventListener('DOMContentLoaded', function() {

const searchInput = document.getElementById('search-input');

const itemsList = document.getElementById('items-list');

const originalItems = Array.from(itemsList.getElementsByTagName('li'));

// Store original items and their text content

const itemsData = originalItems.map(item => ({

element: item,

originalHTML: item.innerHTML,

text: item.textContent.toLowerCase()

}));

searchInput.addEventListener('input', function() {

const searchTerm = this.value.trim().toLowerCase();

let hasResults = false;

itemsData.forEach(item => {

// Reset to original HTML before applying new highlights

item.element.innerHTML = item.originalHTML;

if (searchTerm === '') {

item.element.style.display = '';

hasResults = true;

return;

}

if (item.text.includes(searchTerm)) {

item.element.style.display = '';

hasResults = true;

// Highlight matching text

if (searchTerm.length > 0) {

const regex = new RegExp(searchTerm, 'gi');

item.element.innerHTML = item.originalHTML.replace(

regex,

match => `<span class="highlight">${match}</span>`

);

}

} else {

item.element.style.display = 'none';

}

});

// Show "no results" message if needed

const noResultsElement = document.querySelector('.no-results');

if (!hasResults) {

if (!noResultsElement) {

const li = document.createElement('li');

li.className = 'no-results';

li.textContent = 'No items found matching your search.';

itemsList.appendChild(li);

}

} else if (noResultsElement) {

noResultsElement.remove();

}

});

// Focus the search input on page load

searchInput.focus();

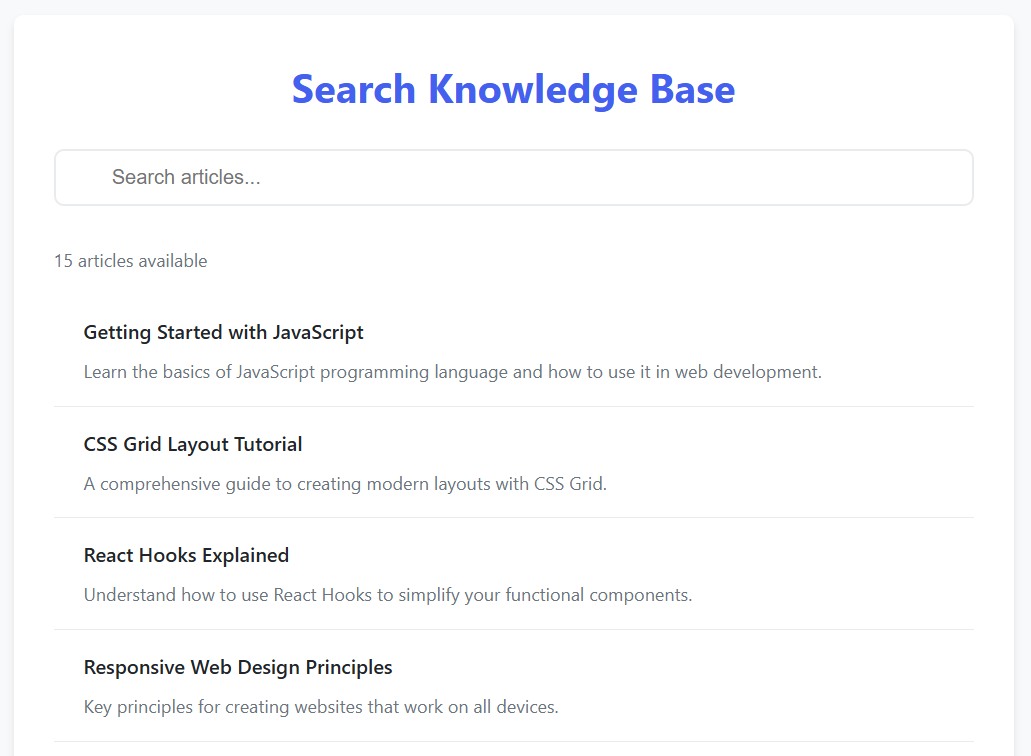
});

</script>

</body>

</html>

**Q3: Create a fully styled HTML, CSS, and JavaScript search interface that allows users to filter a dynamic list of items using keywords and highlights the matching results.**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dynamic Search Interface</title>

<style>

:root {

--primary-color: #4361ee;

--secondary-color: #3f37c9;

--accent-color: #4895ef;

--light-color: #f8f9fa;

--dark-color: #212529;

--success-color: #4cc9f0;

--text-color: #2b2d42;

--border-radius: 8px;

--box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

--transition: all 0.3s ease;

}

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

line-height: 1.6;

color: var(--text-color);

background-color: var(--light-color);

padding: 2rem;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 2rem;

background: white;

border-radius: var(--border-radius);

box-shadow: var(--box-shadow);

}

h1 {

text-align: center;

margin-bottom: 1.5rem;

color: var(--primary-color);

}

.search-container {

position: relative;

margin-bottom: 2rem;

}

.search-icon {

position: absolute;

left: 15px;

top: 50%;

transform: translateY(-50%);

color: var(--accent-color);

}

#search-input {

width: 100%;

padding: 12px 20px 12px 45px;

font-size: 1rem;

border: 2px solid #e9ecef;

border-radius: var(--border-radius);

outline: none;

transition: var(--transition);

}

#search-input:focus {

border-color: var(--accent-color);

box-shadow: 0 0 0 3px rgba(72, 149, 239, 0.2);

}

.results-info {

display: flex;

justify-content: space-between;

margin-bottom: 1rem;

font-size: 0.9rem;

color: #6c757d;

}

.results-list {

list-style: none;

border-radius: var(--border-radius);

overflow: hidden;

}

.result-item {

padding: 1rem 1.5rem;

border-bottom: 1px solid #e9ecef;

transition: var(--transition);

}

.result-item:last-child {

border-bottom: none;

}

.result-item:hover {

background-color: #f8f9fa;

transform: translateX(5px);

}

.result-title {

font-weight: 600;

margin-bottom: 0.5rem;

color: var(--dark-color);

}

.result-description {

font-size: 0.9rem;

color: #6c757d;

}

.highlight {

background-color: rgba(255, 235, 59, 0.5);

padding: 0 2px;

border-radius: 3px;

}

.no-results {

text-align: center;

padding: 2rem;

color: #6c757d;

}

.no-results-icon {

font-size: 2rem;

margin-bottom: 1rem;

color: #adb5bd;

}

@media (max-width: 600px) {

body {

padding: 1rem;

}

.container {

padding: 1.5rem;

}

}

</style>

</head>

<body>

<div class="container">

<h1>Search Knowledge Base</h1>

<div class="search-container">

<svg class="search-icon" width="18" height="18" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

<circle cx="11" cy="11" r="8"></circle>

<line x1="21" y1="21" x2="16.65" y2="16.65"></line>

</svg>

<input

type="text"

id="search-input"

placeholder="Search articles..."

aria-label="Search articles"

autocomplete="off"

>

</div>

<div class="results-info">

<span id="results-count">15 articles available</span>

<span id="results-filtered"></span>

</div>

<ul class="results-list" id="results-list">

<!-- Items will be populated by JavaScript -->

</ul>

</div>

<script>

document.addEventListener('DOMContentLoaded', function() {

// Sample data - in a real app this would come from an API

const items = [

{

id: 1,

title: "Getting Started with JavaScript",

description: "Learn the basics of JavaScript programming language and how to use it in web development."

},

{

id: 2,

title: "CSS Grid Layout Tutorial",

description: "A comprehensive guide to creating modern layouts with CSS Grid."

},

{

id: 3,

title: "React Hooks Explained",

description: "Understand how to use React Hooks to simplify your functional components."

},

{

id: 4,

title: "Responsive Web Design Principles",

description: "Key principles for creating websites that work on all devices."

},

{

id: 5,

title: "Node.js Backend Development",

description: "Building server-side applications with Node.js and Express."

},

{

id: 6,

title: "JavaScript Async/Await Patterns",

description: "Master asynchronous programming in JavaScript using async/await."

},

{

id: 7,

title: "CSS Variables and Custom Properties",

description: "How to use CSS variables to create maintainable stylesheets."

},

{

id: 8,

title: "Vue.js Component Architecture",

description: "Building reusable components in Vue.js applications."

},

{

id: 9,

title: "Web Accessibility Guidelines",

description: "Making your website accessible to all users following WCAG standards."

},

{

id: 10,

title: "TypeScript Fundamentals",

description: "Introduction to TypeScript and its benefits for large-scale applications."

},

{

id: 11,

title: "JavaScript Performance Optimization",

description: "Techniques to improve the performance of your JavaScript code."

},

{

id: 12,

title: "CSS Flexbox Complete Guide",

description: "Everything you need to know about CSS Flexbox layout."

},

{

id: 13,

title: "REST API Best Practices",

description: "Designing clean and efficient RESTful APIs for your applications."

},

{

id: 14,

title: "JavaScript Design Patterns",

description: "Common design patterns and their implementations in JavaScript."

},

{

id: 15,

title: "Progressive Web Apps (PWA)",

description: "Building web applications that work offline and feel like native apps."

}

];

const searchInput = document.getElementById('search-input');

const resultsList = document.getElementById('results-list');

const resultsCount = document.getElementById('results-count');

const resultsFiltered = document.getElementById('results-filtered');

// Initialize the list with all items

renderItems(items);

// Focus the search input on page load

searchInput.focus();

// Add event listener for search input

searchInput.addEventListener('input', function() {

const searchTerm = this.value.trim().toLowerCase();

if (searchTerm === '') {

renderItems(items);

resultsFiltered.textContent = '';

return;

}

const filteredItems = items.filter(item => {

return item.title.toLowerCase().includes(searchTerm) ||

item.description.toLowerCase().includes(searchTerm);

});

renderItems(filteredItems, searchTerm);

// Update filtered count

resultsFiltered.textContent = filteredItems.length > 0 ?

`${filteredItems.length} matches found` : '';

});

function renderItems(itemsToRender, searchTerm = '') {

resultsList.innerHTML = '';

if (itemsToRender.length === 0) {

resultsList.innerHTML = `

<div class="no-results">

<div class="no-results-icon">🔍</div>

<p>No articles found matching your search</p>

<p>Try different keywords or check your spelling</p>

</div>

`;

return;

}

itemsToRender.forEach(item => {

const li = document.createElement('li');

li.className = 'result-item';

let title = item.title;

let description = item.description;

// Highlight matches if search term exists

if (searchTerm) {

const regex = new RegExp(searchTerm, 'gi');

title = title.replace(regex, match => `<span class="highlight">${match}</span>`);

description = description.replace(regex, match => `<span class="highlight">${match}</span>`);

}

li.innerHTML = `

<div class="result-title">${title}</div>

<div class="result-description">${description}</div>

`;

resultsList.appendChild(li);

});

}

});

</script>

</body>

</html>