

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="style.css">
8      <title>Debounce (Search) Example</title>
9  </head>
10
11 <body>
12
13     <h2>Search Users</h2>
14
15     <input type="text" id="search" placeholder="Search user name..." autocomplete="off" required />
16
17     <ul id="userList"></ul>
18
19     <p class="no-results">No results found</p>
20
21     <script src="script.js"></script>
22
23 </body>
24
25 </html>
```

```
1 // -----
2
3 // Array of user names (our sample database for searching)
4
5 // -----
6
7 const users = [
8
9     "Alice Johnson",
10    "Bob Smith",
11    "Charlie Brown",
12    "David Williams",
13    "Evelyn Taylor",
14    "Frank Miller",
15    "Grace Lee",
16    "Hannah Davis",
17    "Ian Thomas"
18
19 ];
20
21 // -----
22
23 // Getting references to HTML elements
24
25 // -----
26
27 // Input box where the user types the search query
28
29 const searchInput = document.getElementById("search");
30
31 // UL element where the list of users will be displayed
32
33 const userList = document.getElementById("userList");
34
35 // Element that shows "No Results Found"
36
37 const noResults = document.querySelector(".no-results");
```

```
38
39 // -----
40
41 // Function: displayUsers
42
43 // Purpose: Displays a list of users in the UI
44
45 // -----
46
47 const displayUsers = (filteredUsers) => {
48
49     // Clear previous results before showing new ones
50
51     userList.innerHTML = "";
52
53     // If no matched users found, show "No Results" message
54
55     if (filteredUsers.length === 0) {
56
57         noResults.style.display = "block";
58
59         return; // Stop the function here
60     }
61
62     // If results exist, hide the "No Results" message
63
64     noResults.style.display = "none";
65
66     // Loop through each user and create a <li> for it
67
68     filteredUsers.forEach((user) => {
69
70         const li = document.createElement("li"); // create list item
71
72         li.textContent = user; // set user name inside <li>
73
74         userList.appendChild(li); // add <li> to the list
75     });
```

```
76 };
77
78 // -----
79
80 // Function: debounce
81
82 // Purpose: Delays execution of a function until user stops typing
83
84 // -----
85
86 // Example: delay = 1000 means "run the function only after 1 second of no typing"
87
88 // Why debounce?
89
90 // - Prevents function from running on every keystroke.
91
92 // - Improves performance, especially for search or API calls.
93
94 const debounce = (fn, delay) => {
95
96   let timeoutId; // stores the timer ID
97
98   return function (...args) { // rest operator collects all arguments
99
100     // Clear previously set timer (if user is still typing)
101
102     clearTimeout(timeoutId);
103
104     // Start a new timer – this will run only if user stops typing
105
106     timeoutId = setTimeout(() => {
107
108       fn(...args); // call the original function with its arguments
109
110       console.log("✅ Debounce fired after", delay / 1000, "seconds");
111
112     }, delay);
113   };
114 }
```

```
114 };
115
116 // -----
117
118 // Function: searchUsers
119
120 // Purpose: Filter users based on typed text
121
122 // -----
123
124 const searchUsers = (searchTerm) => {
125
126     // Convert both user name & search term to lowercase
127
128     // to make search case-insensitive
129
130     const filteredUsers = users.filter((user) => user.toLowerCase().includes(searchTerm.toLowerCase()));
131
132     // Display matching results
133
134     displayUsers(filteredUsers);
135 };
136
137 // -----
138
139 // Show all users when the page first loads
140
141 // -----
142
143 displayUsers(users);
144
145 // -----
146
147 // Create a debounced version of searchUsers
148
149 // delay = 1000 ms (1 second)
150
151 // -----
```

```
152
153 const debouncedSearch = debounce(searchUsers, 1000);
154
155 // -----
156
157 // Add event listener to input box. This triggers on every keystroke
158
159 // -----
160
161 searchInput.addEventListener("input", (event) => {
162
163     // Send the typed value to the debounced search function
164
165     debouncedSearch(event.target.value);
166 });
```