**1. Video Data Array**

const videos = [

{ title: "Exploring the Mountains", details: "1.5M views • 2 weeks ago", thumbnail: "https://via.placeholder.com/300x150", likes: 0, shares: 0 },

...

];

* This array holds objects representing video data.
* Each object contains:
  + title: The name of the video.
  + details: Additional info like views and time.
  + thumbnail: A placeholder URL representing the video image.
  + likes and shares: Counters starting at 0 for interactivity.

**2. Element References**

const searchInput = document.getElementById("searchInput");

const videoContainer = document.getElementById("videoContainer");

* searchInput: Refers to the search bar element where users input text to filter videos.
* videoContainer: Refers to the container where video cards are dynamically displayed.

**3. Rendering Videos**

function renderVideos(filter = "") {

videoContainer.innerHTML = ""; // Clear current videos

videos

.filter(video => video.title.toLowerCase().includes(filter.toLowerCase()))

.forEach((video, index) => {

const videoCard = document.createElement("div");

videoCard.className = "video-card";

videoCard.innerHTML = `

<img src="${video.thumbnail}" alt="Video thumbnail">

<div class="video-info">

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

<div class="video-details">${video.details}</div>

<div class="video-actions">

<button class="action-btn like-btn" data-index="${index}">

👍 Like (<span>${video.likes}</span>)

</button>

<button class="action-btn share-btn" data-index="${index}">

🔄 Share (<span>${video.shares}</span>)

</button>

</div>

</div>

`;

videoContainer.appendChild(videoCard);

});

attachButtonListeners(); // Reattach event listeners

}

* Clears videoContainer before rendering new content.
* Filters videos based on the filter parameter (user input in the search bar).
* Dynamically creates and adds video-card elements:
  + Includes video data (thumbnail, title, details).
  + Adds **Like** and **Share** buttons with counters and unique data-index attributes for tracking which video is being interacted with.

**4. Search Input Listener**

searchInput.addEventListener("input", event => {

renderVideos(event.target.value);

});

* Listens for user input in the search bar.
* Passes the search query (event.target.value) to the renderVideos() function to update displayed videos dynamically.

**5. Attaching Button Listeners**

function attachButtonListeners() {

document.querySelectorAll(".like-btn").forEach(button => {

button.addEventListener("click", event => {

const index = event.target.getAttribute("data-index");

videos[index].likes++;

renderVideos(searchInput.value); // Update UI with new like count

});

});

document.querySelectorAll(".share-btn").forEach(button => {

button.addEventListener("click", event => {

const index = event.target.getAttribute("data-index");

videos[index].shares++;

alert(`You shared: ${videos[index].title}`);

renderVideos(searchInput.value); // Update UI with new share count

});

});

}

* Finds all buttons with .like-btn or .share-btn classes and adds click event listeners.
* On **Like Button Click**:
  + Increases the likes count of the corresponding video.
  + Updates the UI with renderVideos(searchInput.value) to reflect the new count.
* On **Share Button Click**:
  + Increases the shares count of the corresponding video.
  + Displays an alert message showing the shared video's title.
  + Updates the UI.

**6. Initial Render**

renderVideos();

* Renders all videos when the page loads, without any search filters applied.

**Summary of Features:**

1. **Dynamic Content**: Videos are created and displayed based on a data array.
2. **Search Functionality**: Filters videos in real time using the input text.
3. **Interactivity**:
   * **Like Button**: Increments the like count for a specific video.
   * **Share Button**: Increments the share count and shows an alert with the video title.
4. **Responsive Updates**: The page dynamically updates when users interact with buttons or search.