Frontend Machine Coding Interview: Cheat Sheet

by Geeky Frontend

Machine coding interviews are a must in all tech companies hiring for frontend/UI roles at any level. There are plenty of machine coding questions out there that can become part of the frontend interview, but the underlying concepts are quite common. As part of this Frontend Machine Coding Cheat-Sheet, we aim to capture common concepts in a single blog post, so that anyone can revise all the important concepts from a single place.

Search Logic

Related Questions: Auto Complete Search or Type Ahead search

Search for a text in the Array of objects using `Array.includes()` or a Regex expression

```
// 🔍 Using Includes
const data = [
  { name: "Alice Johnson", role: "Developer" },
  { name: "Bob Smith", role: "Designer" },
  { name: "Charlie Brown", role: "Project Manager" },
];
const searchTerm = "Bob";
const resultIncludes = data.filter(item =>
  item.name.includes(searchTerm) || item.role.includes(searchTerm)
);
console.log("Search using includes():", resultIncludes);

// 🔍 Using Regex
```

```
const regex = new RegExp(searchTerm, "i"); // 'i' for case-insensitive
const resultRegex = data.filter(item =>
  regex.test(item.name) || regex.test(item.role)
);
console.log("Search using RegExp:", resultRegex);
```

Accessibility Considerations for Search Boxes

1️⃣ *Search Box with Auto-suggestion*

HTML

```
<label for="search">Search:</label>
<input
  type="text"
  id="search"
  role="combobox"
  aria-autocomplete="list"
  aria-expanded="false"
  aria-controls="suggestions"
  autocomplete="off"
/>
<ul id="suggestions" role="listbox" hidden></ul>
```

JavaScript (with Keyboard Support)

```
const input = document.getElementById("search");
const list = document.getElementById("suggestions");
const suggestions = ["Apple", "Banana", "Blueberry", "Cherry", "Mango"
let activeIndex = -1;

input.addEventListener("input", () => {
  const value = input.value.toLowerCase();
  const matched = suggestions.filter(s => s.toLowerCase().includes(val

  list.innerHTML = "";
  activeIndex = -1;

  if (matched.length > 0 && value) {
    list.hidden = false;
    input.setAttribute("aria-expanded", "true");

    matched.forEach((item, index) => {
```

```
      const li = document.createElement("li");
      li.id = `suggestion-${index}`;
      li.setAttribute("role", "option");
      li.textContent = item;

      li.addEventListener("click", () => {
        input.value = item;
        list.hidden = true;
        input.setAttribute("aria-expanded", "false");
      });

      list.appendChild(li);
    });
  } else {
    list.hidden = true;
    input.setAttribute("aria-expanded", "false");
  }
});

// Keyboard support
input.addEventListener("keydown", (e) => {
  const items = list.querySelectorAll("li");

  if (items.length === 0) return;

  if (e.key === "ArrowDown") {
    activeIndex = (activeIndex + 1) % items.length;
  } else if (e.key === "ArrowUp") {
    activeIndex = (activeIndex - 1 + items.length) % items.length;
  } else if (e.key === "Enter" && activeIndex >= 0) {
    e.preventDefault(); // Prevent form submission
    input.value = items[activeIndex].textContent;
    list.hidden = true;
    input.setAttribute("aria-expanded", "false");
  } else if (e.key === "Escape") {
      list.hidden = true;
      input.setAttribute("aria-expanded", "false");
  }

  items.forEach((item, index) => {
    if (index === activeIndex) {
        item.style.backgroundColor = '#f0f0f0'; // Example active styl
        input.setAttribute("aria-activedescendant", item.id);
    } else {
        item.style.backgroundColor = '';
    }
```

```
  });
});
```

2️⃣ A general search box without Auto-suggestion

```
<input
  type="search"
  role="searchbox"
  aria-label="Search site"
  placeholder="Search..."
/>
```

Listing Logic

Related Questions: Auto Complete Search or Type Ahead search, News feed / Infinite scroll, Pagination

Responsive listing using flexbox/grid

HTML

```
<h2>Flexbox Listing</h2>
<div class="flex-container" id="flexList"></div>
```

CSS

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  justify-content: center;
}

.flex-card {
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  padding: 1rem;
```

```
  width: 200px;
  text-align: center;
  border: 1px solid #eee;
}
```

JavaScript

```
const items = ["HTML", "CSS", "JavaScript", "React", "Vue", "Angular",
const flexList = document.getElementById("flexList");

items.forEach((name) => {
  const card = document.createElement("div");
  card.className = "flex-card";
  card.innerHTML = `<h4>${name}</h4>`;
  flexList.appendChild(card);
});
```

Product Catalog listing with images

HTML

```
<h1>🛍️ Product Catalog</h1>
<div class="catalog" id="catalog"></div>
```

CSS

```
.catalog {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
  gap: 1.5rem;
}

.product-card {
  background: #fff;
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.08);
  transition: transform 0.2s ease;
}

.product-card:hover {
    transform: translateY(-5px);
```

```
}

.product-card img {
  width: 100%;
  height: 160px;
  object-fit: cover;
}

.product-info {
  padding: 1rem;
}

.product-info h3 {
  font-size: 1.1rem;
  margin: 0 0 0.5rem;
}

.product-info .price {
  font-weight: bold;
  color: #333;
  margin-top: 0.5rem;
}
```

JavaScript

```
const products = [
  { name: "Classic Sneakers", price: "$59.99", image: "https://images.
  { name: "Leather Backpack", price: "$89.00", image: "https://images.
  { name: "Smart Watch", price: "$199.99", image: "https://images.unsp
  { name: "Sunglasses", price: "$39.99", image: "https://images.unspla
];

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

products.forEach(product => {
  const card = document.createElement("div");
  card.className = "product-card";
  card.innerHTML = `
    <img src="${product.image}" alt="${product.name}">
    <div class="product-info">
      <h3>${product.name}</h3>
      <p class="price">${product.price}</p>
    </div>
  `;
  catalog.appendChild(card);
});
```

Infinite scroll using `IntersectionObserver`

HTML

```
<h2>📜 Infinite Scroll Example</h2>
<div id="item-container"></div>
<div class="loader" id="loader">Loading more items...</div>
```

CSS

```
.item {
  background: #fff;
  border: 1px solid #eee;
  margin-bottom: 1rem;
  padding: 1rem;
  border-radius: 6px;
}

.loader {
  text-align: center;
  padding: 1rem;
  font-size: 1rem;
  color: #777;
}
```

JavaScript

```
const container = document.getElementById("item-container");
const loader = document.getElementById("loader");
let itemCount = 0;
const batchSize = 10;

function loadItems(count) {
  for (let i = 0; i < count; i++) {
    const item = document.createElement("div");
    item.className = "item";
    item.textContent = `Product ${++itemCount}`;
    container.appendChild(item);
  }
}

const observer = new IntersectionObserver((entries) => {
  const entry = entries[0];
```

```
  if (entry.isIntersecting) {
    // Simulate loading delay
    setTimeout(() => {
      loadItems(batchSize);
    }, 500);
  }
});

// Initial load
loadItems(batchSize);
observer.observe(loader);
```

API Calls

1️⃣ API calls using fetch - GET, POST, PATCH

👉 *GET API call example using fetch()*

```
const container = document.getElementById("posts-container");

async function fetchPosts() {
  try {
    const response = await fetch(
      "https://jsonplaceholder.typicode.com/posts?_limit=5"
    );
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const posts = await response.json();
    // container.innerHTML = ""; // Clear loading text

    posts.forEach((post) => {
      const div = document.createElement("div");
      div.className = "post";
      div.innerHTML = `
        <h3>${post.title}</h3>
        <p>${post.body}</p>
      `;
      // container.appendChild(div);
    });
```

```
  } catch (error) {
    // container.innerHTML = `<p style="color:red;">❌ Error: ${error.
  }
}

fetchPosts();
```

👉 *POST API call example*

```
const form = document.getElementById("postForm");

form.addEventListener("submit", async (e) => {
  e.preventDefault();

  const title = document.getElementById("title").value;
  const body = document.getElementById("body").value;

  try {
    const response = await fetch("https://jsonplaceholder.typicode.com
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({
        title,
        body,
        userId: 1
      })
    });

    if (!response.ok) {
        throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const data = await response.json();

    alert(`✅ Post submitted successfully! ID: ${data.id}`);
  } catch (error) {
    alert(`❌ API failed, Error: ${error.message}`);
  }
});
```

👉 *Cancel the fetch API call using `AbortController`*

```
let controller;

document.getElementById("startBtn").addEventListener("click", async ()
  const log = document.getElementById("log");
  controller = new AbortController();
  const signal = controller.signal;

  log.textContent = "Fetching data...";

  try {
    const response = await fetch("https://jsonplaceholder.typicode.com
      signal // Pass signal to the fetch call
    });

    const data = await response.json();
    log.textContent = `Fetched ${data.length} posts`;
  } catch (error) {
    if (error.name === "AbortError") {
      log.textContent = "Fetch aborted by user.";
    } else {
      log.textContent = `Error: ${error.message}`;
    }
  }
});

document.getElementById("cancelBtn").addEventListener("click", () => {
  if (controller) {
    controller.abort(); // Abort the fetch request
  }
});
```

2️⃣ Parallel API calls

👉 *Parallel API call using `Promise.all()`*

```
async function fetchDataWithAll() {
  try {
    const [posts, users] = await Promise.all([
      fetch("https://jsonplaceholder.typicode.com/posts").then((res) =
      fetch("https://jsonplaceholder.typicode.com/users").then((res) =
    ]);

    alert(`✅ Posts fetched: ${posts.length}\n✅ Users fetched: ${user
  } catch (error) {
    // Fails if any one of the promises reject
```

```
    alert(`❌ Error: ${error.message}`);
  }
}
```

👉 *Parallel API call using `Promise.allSettled()`*

```
async function fetchDataWithAllSettled() {
  const results = await Promise.allSettled([
    fetch("https://jsonplaceholder.typicode.com/posts").then((res) =>
    fetch("https://jsonplaceholder.typicode.com/users").then((res) =>
    fetch("https://jsonplaceholder.typicode.com/invalid-endpoint").the
        if (!res.ok) throw new Error('Invalid endpoint');
        return res.json();
    })
  ]);

  let output = "";
  results.forEach((result, index) => {
    if (result.status === "fulfilled") {
      output += `✅ Request ${index + 1} succeeded.\n`;
    } else {
      output += `❌ Request ${index + 1} failed: ${result.reason}\n`;
    }
  });

  alert(output);
}
```

DOM - Event handling & Delegation

1️⃣ Event handling

```
function appendLog(message) {
  // Logic to display log messages...
  console.log(message);
}

// Click
document.getElementById("clickBtn").addEventListener("click", () => {
  appendLog("Button Clicked");
```

```
});

// Double Click
document.getElementById("dblClickBtn").addEventListener("dblclick", ()
  appendLog("Button Double Clicked");
});

// Key Press
document.addEventListener("keydown", (e) => {
  appendLog(`Key Pressed: "${e.key}"`);
});

// Mouse Move
document.addEventListener("mousemove", (e) => {
  // This is very noisy, consider throttling it
  // appendLog(`Mouse at X: ${e.clientX}, Y: ${e.clientY}`);
});
```

2️⃣ Event Delegation

HTML

```
<h2>Click a card</h2>
<div class="grid-container" id="grid">
  <div class="card" data-id="1">Item 1</div>
  <div class="card" data-id="2">Item 2</div>
  <!-- ... more cards ... -->
</div>
```

CSS

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(120px, 1fr));
  gap: 1rem;
}
.card {
  background-color: #f0f0f0;
  padding: 1rem;
  text-align: center;
  cursor: pointer;
  border-radius: 8px;
}
```

JavaScript

```
const grid = document.getElementById("grid");

// Attach one listener to the parent
grid.addEventListener("click", function (event) {
  // Find the closest ancestor that is a .card
  const card = event.target.closest(".card");

  // If a card was clicked (and not the gap between cards)
  if (card) {
    const id = card.dataset.id;
    alert(`✅ You clicked on Card ${id}`);
  }
});
```

Debouncing & Throttling

1️⃣ Debouncing

```
function debounce(fn, delay) {
  let timeoutId;
  return function (...args) {
    clearTimeout(timeoutId);
    timeoutId = setTimeout(() => {
        fn.apply(this, args);
    }, delay);
  };
}

function handleSearch(event) {
  console.log(`🔎 Searching for: ${event.target.value}`);
}

const debouncedSearch = debounce(handleSearch, 500);
document.getElementById("searchInput").addEventListener("input", debou
```

2️⃣ Throttling

```
function throttle(fn, limit) {
  let inThrottle;
  return function (...args) {
    if (!inThrottle) {
      fn.apply(this, args);
      inThrottle = true;
      setTimeout(() => inThrottle = false, limit);
    }
  };
}

function handleScroll() {
  console.log(`📍 Scroll Y: ${window.scrollY}`);
}

const throttledScroll = throttle(handleScroll, 300);
window.addEventListener("scroll", throttledScroll);
```

Media Queries - Responsive Design

CSS

```
.box {
  padding: 2rem;
  text-align: center;
  color: white;
  font-size: 1.5rem;
  border-radius: 8px;
}

/* Default styles (Mobile First approach) */
.box {
    background-color: blue; /* Blue for mobile */
    font-size: 1rem;
}

/* Tablet: 501px and up */
@media (min-width: 501px) {
  .box {
    background-color: green;
    font-size: 1.2rem;
  }
```

```
}

/* Desktop: 701px and up */
@media (min-width: 701px) {
  .box {
    background-color: red;
    font-size: 1.5rem;
  }
}
```

Modal/Menu Implementation - Absolute positioning

HTML

```
<button id="openModalBtn">Open Modal</button>
<div id="modalOverlay" class="modal-overlay">
  <div class="modal">
    <button class="close-btn" id="closeModalBtn">×</button>
    <h3>Hello From Modal</h3>
    <p>This is an example modal dialog box.</p>
  </div>
</div>
```

CSS

```
.modal-overlay {
  display: none; /* Hidden by default */
  position: fixed;
  inset: 0; /* top, right, bottom, left = 0 */
  background: rgba(0, 0, 0, 0.6);
  display: flex; /* Use flexbox to center */
  justify-content: center;
  align-items: center;
  opacity: 0;
  transition: opacity 0.3s ease;
  pointer-events: none;
}

.modal-overlay.show {
```

```
  opacity: 1;
  pointer-events: auto;
}

.modal {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  max-width: 400px;
  width: 90%;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
  position: relative;
}

.close-btn {
  position: absolute;
  top: 0.5rem;
  right: 0.75rem;
  background: transparent;
  border: none;
  font-size: 1.5rem;
  cursor: pointer;
}
```

JavaScript

```
const openBtn = document.getElementById("openModalBtn");
const closeBtn = document.getElementById("closeModalBtn");
const overlay = document.getElementById("modalOverlay");

function openModal() {
    overlay.classList.add("show");
}

function closeModal() {
    overlay.classList.remove("show");
}

openBtn.addEventListener("click", openModal);
closeBtn.addEventListener("click", closeModal);

// Close when clicking outside the modal
overlay.addEventListener("click", (e) => {
  if (e.target === overlay) {
    closeModal();
  }
```

```
});

// Close with Escape key
document.addEventListener('keydown', (e) => {
    if (e.key === 'Escape' && overlay.classList.contains('show')) {
        closeModal();
    }
});
```

API Data Polling

```
let pollingId = null;

function startPolling(url, interval = 5000) {
  // Stop any existing polling
  if (pollingId) {
    clearInterval(pollingId);
  }

  const poll = async () => {
    try {
      console.log('Polling for new data...');
      const response = await fetch(url);
      const data = await response.json();
      console.log("Polled Data:", data.title);
    } catch (error) {
      console.error("Polling error:", error);
      // Optional: stop polling on error
      // clearInterval(pollingId);
    }
  };

  // Call immediately, then set interval
  poll();
  pollingId = setInterval(poll, interval);
  console.log(`Polling started with ID: ${pollingId}`);
}

function stopPolling() {
    if(pollingId) {
        clearInterval(pollingId);
        console.log(`Polling stopped (ID: ${pollingId})`);
        pollingId = null;
```

```
    }
}

// Example Usage:
// startPolling("https://jsonplaceholder.typicode.com/posts/1", 3000);
// To stop: stopPolling();
```

Random Number Generation

Generate a random number between a range of integers

```
function getRandomInteger(min, max) {
  // Ensure min and max are integers
  min = Math.ceil(min);
  max = Math.floor(max);
  // The maximum is inclusive and the minimum is inclusive
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
console.log(getRandomInteger(1, 100)); // ex: 35
```

Generate a floating random number

```
function getRandomFloat(min, max) {
  return Math.random() * (max - min) + min;
}
console.log(getRandomFloat(1, 10)); // ex: 4.6345...
```

Date handling in JavaScript

Difference between 2 dates

```
function getDateDifference(date1, date2) {
  const diffMs = Math.abs(date2 - date1); // Difference in millisecond
```

```
  const days = Math.floor(diffMs / (1000 * 60 * 60 * 24));
  const hours = Math.floor((diffMs / (1000 * 60 * 60)) % 24);
  const minutes = Math.floor((diffMs / (1000 * 60)) % 60);

  return { days, hours, minutes };
}

const date1 = new Date('2025-06-10T10:00:00');
const date2 = new Date('2025-06-13T12:30:00');
console.log(getDateDifference(date1, date2));
// Output: { days: 3, hours: 2, minutes: 30 }
```

Display date format

```
const now = new Date();

// Simple, locale-sensitive formats
console.log(now.toLocaleDateString());     // e.g., "10/3/2025" (US)
console.log(now.toLocaleTimeString());     // e.g., "2:34:59 AM" (US)
console.log(now.toLocaleString());       // e.g., "10/3/2025, 2:34:59

// More control with Intl.DateTimeFormat
const options = {
    weekday: 'long',
    year: 'numeric',
    month: 'long',
    day: 'numeric'
};
console.log(new Intl.DateTimeFormat('en-US', options).format(now));
// e.g., "Friday, October 3, 2025"
```

Happy Coding :)