

```
// === Live Search for Microsite Pages ===
fetch("data/pages.json")
  .then((res) => {
    if (!res.ok) throw new Error("Failed to load pages.json");
    return res.json();
})
  .then((pages) => {
    const container = document.getElementById("pageList");
    const input = document.getElementById("searchInput");

    function renderCards(query = "") {
      const q = query.toLowerCase();
      container.innerHTML = "";

      pages.forEach((page) => {
        const match =
          page.title.toLowerCase().includes(q) ||
          page.description.toLowerCase().includes(q);

        if (match) {
          container.innerHTML += `
<div class="col-md-6 page-card">
  <div class="shout-card">
    <strong>📄 ${page.title}</strong>
    <br />
    <small>${page.description}</small>
    <br />
    <a href="${page.url}" class="btn btn-primary w-100 mb-2">Open Page</a>
  </div>
</div>`;
        }
      });
    }

    if (container.innerHTML === "") {
      container.innerHTML = `
        <div class="alert alert-warning">
          ⚠️ No matching pages found. Try a different keyword or check your spelling.
        </div>`;
    }
  }

  renderCards();

  input.addEventListener("input", () => {
    renderCards(input.value);
  });
}) .catch((err) => {
```

```
document.getElementById("pageList").innerHTML = `<div class="alert alert-danger">  
    X Could not load page data. Check your <code>data/pages.json</code> file.  
    </div>`;  
    console.error("Search error:", err);  
});  
  
// Open a popup window with provided URL  
function openPopup(url) {  
    window.open(url, "popupWindow",  
    "width=auto,height=auto,scrollbars=yes");  
}  
  
// ===== TOGGLE SECTION VISIBILITY =====//  
function toggleSection(id) {  
    var section = document.getElementById(id);  
    section.style.display = section.style.display === "none" ?  
    "block" : "none";  
}  
  
document.querySelectorAll(".toggle-submenu").forEach((button) => {  
    button.addEventListener("click", () => {  
        const submenu = button.nextElementSibling;  
        const isOpen = submenu.classList.contains("open");  
  
        // Toggle submenu  
        submenu.classList.toggle("open");  
  
        // Update arrow  
        button.textContent = button.textContent.replace(  
            isOpen ? "▲" : "▼",  
            isOpen ? "▼" : "▲"  
        );  
    });  
});  
  
// ===== Instant Collapse on Scroll Start =====  
  
const banner = document.querySelector(".banner-wrapper");  
const toolbar = document.getElementById("desktop-nav-toolbar");  
const header = document.querySelector(".nhs-header");  
  
let isCollapsed = false;  
  
window.addEventListener("scroll", () => {  
    const scrollY = window.scrollY;
```

```
// Buffer zone: collapse if scrollY > 30, expand only if
scrollY < 10
  if (scrollY > 1 && !isCollapsed) {
    banner?.classList.add("shrink");
    header?.classList.add("shrink");
    toolbar?.classList.add("hide-on-scroll");
    isCollapsed = true;
  } else if (scrollY < 1 && isCollapsed) {
    banner?.classList.remove("shrink");
    header?.classList.remove("shrink");
    toolbar?.classList.remove("hide-on-scroll");
    isCollapsed = false;
}
});
```

//Copy Snippet Code

```
function copyVisibleSnippet() {
  // Find the button that was clicked
  const btn = event.target;

  // Find the nearest .snippet block in the same container
  const snippet =
  btn.closest(".container")?.querySelector(".snippet");

  if (snippet) {
    navigator.clipboard.writeText(snippet.innerText).then(() => {
      btn.textContent = "Copied!";
      setTimeout(() => (btn.textContent = "Copy"), 1500);
    });
  }
}

// === Load Shared Partials ===
async function loadPartial(id, filePath, wrapper = null) {
  const res = await fetch(filePath);
  if (!res.ok) throw new Error(`Failed to load ${filePath}`);
  const html = await res.text();
  const container = document.getElementById(id);
  if (container) {
    container.innerHTML = wrapper
      ? `${wrapper.start}${html}${wrapper.end}`
      : html;
  }
}

// === Helpers to create menu items ===
function createMenuItems(items, isDropdown = false) {
  const fragment = document.createDocumentFragment();
```

```

items.forEach((item) => {
  const li = document.createElement("li");
  li.className = item.children ? "nav-item dropdown" : "nav-item";

  const a = document.createElement("a");
  a.textContent = item.label || "";
  a.href = item.href || "#";

  if (item.children) {
    // Parent item (dropdown trigger)
    a.className = isDropdown
      ? "dropdown-item submenu-toggle"
      : "nav-link dropdown-toggle";
    if (!isDropdown) a.setAttribute("data-bs-toggle", "dropdown");

    li.appendChild(a);

    // Child dropdown menu
    const childUL = document.createElement("ul");
    childUL.className = "dropdown-menu";

    // Children rendered as dropdown items (these are inside
    // a dropdown)
    item.children.forEach((child) => {
      if (child.children) {
        // Nested submenu (Bootstrap doesn't officially
        // support nested dropdowns;
        // if you already style ".dropdown-submenu", we
        // replicate structure)
        const subLi = document.createElement("li");
        subLi.className = "dropdown-submenu";

        const subA = document.createElement("a");
        subA.className =
          "dropdown-item btn btn-block btn-primary submenu-
          toggle";
        subA.href = "#";
        subA.textContent = child.label || "";
        subLi.appendChild(subA);

        const subMenu = document.createElement("ul");
        subMenu.className = "dropdown-menu";

        child.children.forEach((leaf) => {
          const leafLi = document.createElement("li");
          const leafA = document.createElement("a");
          leafA.className = "dropdown-item btn btn-block"

```

```

    btn-primary";
        leafA.href = leaf.href || "#";
        leafA.textContent = leaf.label || "";
        leafLi.appendChild(leafA);
        subMenu.appendChild(leafLi);
    });

    subLi.appendChild(subMenu);
    childUL.appendChild(subLi);
} else {
    const childLi = document.createElement("li");
    const childA = document.createElement("a");
    childA.className = "dropdown-item btn btn-block btn-primary";
    childA.href = child.href || "#";
    childA.textContent = child.label || "";
    childLi.appendChild(childA);
    childUL.appendChild(childLi);
}
});

li.appendChild(childUL);
} else {
    // Simple link
    a.className = "nav-link";
    li.appendChild(a);
}

fragment.appendChild(li);
});

return fragment;
}

// === Build menus from JSON, inject into existing ULs ===
async function buildMenusFromJson(jsonPath) {
    const res = await fetch(jsonPath);
    if (!res.ok) throw new Error(`Failed to load JSON: ${jsonPath}`);
    const menuData = await res.json();

    // Header UL: .navbar-nav mx-auto
    const headerUl = document.querySelector("#desktop-nav-toolbar .navbar-nav");
    // Sidebar UL: .nav.navbar-nav.flex-column
    const sidebarUl = document.querySelector(
        "#sidebarMenu .nav.navbar-nav.flex-column"
    );

    if (headerUl) {

```

```

        headerUl.innerHTML = ""; // clear any server/partial
        leftovers[...]
        headerUl.appendChild(createMenuItems(menuData, false));
    }
    if (sidebarUl) {
        sidebarUl.innerHTML = ""; // clear any server/partial
        leftovers[...]
        sidebarUl.appendChild(createMenuItems(menuData, false));
    }
}

markActiveLinks();
}

// === Fallback to legacy partial if JSON fails ===
async function buildMenusWithFallback() {
    try {
        await buildMenusFromJson("data/Menu Manifest.json");
    } catch (e) {
        // Fallback: load the old partial if JSON not available
        console.warn("Menu JSON failed, falling back to partials/menu.html", e);
    }

    // Inject into header UL
    const headerUl = document.querySelector("#desktop-nav-
toolbar .navbar-nav");
    if (headerUl) {
        const res = await fetch("partials/menu.html");
        if (res.ok) {
            headerUl.innerHTML = await res.text();
        }
    }
}

// Inject into sidebar UL
const sidebarUl = document.querySelector(
    "#sidebarMenu .nav.navbar-nav.flex-column"
);
if (sidebarUl) {
    const res2 = await fetch("partials/menu.html");
    if (res2.ok) {
        sidebarUl.innerHTML = await res2.text();
    }
}
}

// === Active page marking ===
function markActiveLinks() {
    const current =
        window.location.pathname.split("/").pop().toLowerCase();
    const allLinks = document.querySelectorAll('a');

```

```

        "#desktop-nav-toolbar a.nav-link, #sidebarMenu a"
    );

    allLinks.forEach((a) => {
        // Strip any query/hash, compare file names
        const hrefFile = (a.getAttribute("href") || "")  

            .split("/")
            .pop()
            .split("#")[0]
            .split("?")[0]
            .toLowerCase();
        if (hrefFile && current && hrefFile === current) {
            a.classList.add("active");
            // If this link is inside a dropdown, you can also add
            'show' to its parent menu for context
            const parentDropdown = a.closest(".dropdown");
            if (parentDropdown) {
                const trigger =
parentDropdown.querySelector(".dropdown-toggle");
                if (trigger) trigger.classList.add("active");
            }
        } else {
            a.classList.remove("active");
        }
    });
}

// === Boot sequence ===
loadPartial("shared-header", "partials/header.html").then(()  

=> {
    // IMPORTANT: remove any existing calls that load 'partials/
    menu.html'
    // e.g., DO NOT call:
    // loadPartial("main-nav", "partials/menu.html", {...});
    // loadPartial("sidebar-nav", "partials/menu.html");
    // Call this after your header partial has been injected
    initHeaderSearch();
    // Build menus from JSON (with fallback if JSON fails)
    buildMenusWithFallback();

    // Footer and theme logic as before
    loadPartial("shared-footer", "partials/footer.html");

    // === Theme Selector Logic ===
    const themeSelectors = document.querySelectorAll(".theme-
    selector");
    themeSelectors.forEach(selector) => {
        selector.addEventListener("change", function () {
            const selectedTheme = this.value;
        });
    });
}
);

```

```
document.body.className = document.body.className
  .split(" ")
  .filter((cls) => !cls.startsWith("theme-"))
  .join(" ")
  .trim();

document.body.classList.add("theme-" + selectedTheme);

if (localStorage.getItem("darkMode") === "true") {
  document.body.classList.add("dark-mode");
}

localStorage.setItem("theme", selectedTheme);
applyMermaidTheme();
});;
});

// === Dark Mode Toggle Logic ===
const toggleButtons = document.querySelectorAll(".toggle-dark");
const body = document.body;

function updateDarkMode() {
  const isDark = body.classList.toggle("dark-mode");
  toggleButtons.forEach((btn) => {
    btn.textContent = isDark ? "🌙 Dark Mode" : "☀️ Light
Mode";
  });
  localStorage.setItem("darkMode", isDark ? "true" :
"false");
  applyMermaidTheme();
}

toggleButtons.forEach((btn) => {
  btn.addEventListener("click", updateDarkMode);
});

// === Initial Theme Load ===
const savedTheme = localStorage.getItem("theme");
if (savedTheme) {
  document.body.classList.add("theme-" + savedTheme);
}
if (localStorage.getItem("darkMode") === "true") {
  document.body.classList.add("dark-mode");
}

applyMermaidTheme();
};

async function initHeaderSearch() {
```

```

try {
  const res = await fetch("data/pages.json");
  if (!res.ok) throw new Error("pages.json not found");
  const pages = await res.json();

  // Configure Fuse
  const fuse = new Fuse(pages, {
    keys: ["title", "description"],
    threshold: 0.3,
  });

  const input = document.getElementById("headerSearch");
  const resultsList = document.getElementById("headerSearchResults");

  if (!input || !resultsList) return; // header not loaded yet

  input.addEventListener("input", () => {
    const query = input.value.trim();
    resultsList.innerHTML = "";

    if (query.length > 1) {
      const results = fuse.search(query).slice(0, 5); // top 5
      results.forEach((r) => {
        const li = document.createElement("li");
        li.className = "list-group-item";
        li.innerHTML = `
          <a href="${r.item.url}" class="fw-bold">${r.item.title}</a>
          <div class="small text-muted">${r.item.description}</div>
        `;
        resultsList.appendChild(li);
      });
    }
  });
} catch (err) {
  console.error("Search init failed:", err);
}

input.addEventListener("input", () => {
  const q = input.value.trim();
  resultsList.innerHTML = "";
  resultsList.classList.remove("show");

  if (q.length > 1) {
    const results = fuse.search(q).slice(0, 7);
  }
}

```

```
results.forEach((r) => {
  const li = document.createElement("li");
  li.className = "list-group-item";
  li.innerHTML = `
    <a href="${r.item.url}" class="fw-bold d-block">${r.item.title}</a>
    <div class="small text-muted">${r.item.description}</div>
  `;
  resultsList.appendChild(li);
});
if (results.length) resultsList.classList.add("show");
});

// hide on blur / outside click
document.addEventListener("click", (e) => {
  const form = document.querySelector('[role="search"]');
  if (!form?.contains(e.target)) {
    resultsList.classList.remove("show");
  }
});
```