learn.js

```
document.addEventListener('DOMContentLoaded', function() {
    // === Stacked CWE Severity Bar Chart (Learn View) ===
    // Only run if we have data for the chart and the proper container exists.
    if (
        window.learnCweSeverityData &&
        Array.isArray(window.learnCweSeverityData.labels) &&
        window.learnCweSeverityData.labels.length > 0 &&
        document.getElementById('learnCweSeverityChart')
    ) {
        const ctx =
document.getElementById('learnCweSeverityChart').getContext('2d');
        const data = window.learnCweSeverityData.data || {};

        new Chart(ctx, {
            type: 'bar',
            data: {
                labels: window.learnCweSeverityData.labels, //CWE names/codes
for x-axis
                datasets: [
                    { label: 'Critical', data: data.CRITICAL || [],
backgroundColor: '#f55855' },
                    { label: 'High', data: data.HIGH || [], backgroundColor:
'#f8a541' },
                    { label: 'Medium', data: data.MEDIUM || [],
backgroundColor: '#3b8ded' },
                    { label: 'Low', data: data.LOW || [], backgroundColor:
'#42d392' }
                ]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    title: { display: true, text: 'CWEs by Severity (Key Set)'
},
                    tooltip: { mode: 'index', intersect: false }
                },
                scales: { x: { stacked: true }, y: { stacked: true } }
            }
        });
    }
```

```
    // === Utility: HTML Encode (avoid XSS in dynamic content) ===
    function htmlEncode(text) {
        var el = document.createElement('div');
        el.innerText = text || '';
        return el.innerHTML;
    }

    // === Show "Loading..." Panel for CWE Detail ===
    function showLoadingState(code) {
        document.getElementById('cweDetailPanel').innerHTML = `
            <div style="text-align: center; padding: 40px; color: #a9adc1;">
                <div style="font-size: 2rem; margin-bottom: 16px;">⏳</div>
                <div>Loading ${code} details from MITRE...</div>
                <div style="margin-top: 8px; font-size: 0.9em;">This may take
a few seconds</div>
            </div>
        `;
    }

    // === Show Error/Failure Panel for CWE Detail ===
    function showErrorState(code, error) {
        document.getElementById('cweDetailPanel').innerHTML = `
            <div style="text-align: center; padding: 40px; color: #f47174;">
                <div style="font-size: 2rem; margin-bottom: 16px;">⚠️</div>
                <div>Failed to load ${code} details</div>
                <div style="margin-top: 8px; font-size: 0.9em; color:
#a9adc1;">
                    Error: ${htmlEncode(error)}
                </div>
                <div style="margin-top: 16px;">
                    <a
href="https://cwe.mitre.org/data/definitions/${code.replace('CWE-','')}.html"
                       target="_blank"
                       style="color: #63a4ff; text-decoration: underline;">
                        View on MITRE website ↗
                    </a>
                </div>
            </div>
        `;
    }

    // === Fetch CWE data from backend API ===
    async function fetchCweData(code) {
        try {
            const response = await fetch(`/api/cwe/${code}`);
```

```
            const result = await response.json();

            if (result.success && result.data) {
                return result.data;
            } else {
                throw new Error(result.error || 'Failed to fetch CWE data');
            }
        } catch (error) {
            console.error(`Error fetching CWE ${code}:`, error);
            throw error;
        }
    }

    // === Render the CWE Detail View (description, mitigation, examples, etc)
===
    async function renderDetail(code) {
        // Show loading state immediately
        showLoadingState(code);

        try {
            // First check if we have local data
            const cweDict = window.learnCweDict || {};
            let entry = cweDict[code];

            // If no local data or it's placeholder data, fetch from API
            if (!entry ||
                entry.description?.includes('Click to fetch detailed
information') ||
                entry.mitigations?.includes('Loading detailed information...')
||
                !entry.description ||
                entry.description.length < 50) {

                console.log(`Fetching fresh data for ${code}...`);
                entry = await fetchCweData(code);

                // Update local cache
                if (window.learnCweDict) {
                    window.learnCweDict[code] = entry;
                }
            }

            if (!entry) {
                throw new Error('No CWE data available');
            }
```

```
            // Extract relationships
            let relParent = entry.relationships ?
                entry.relationships.filter(r => r[1] &&
r[1].toLowerCase().includes('parent')).map(r => r[0]) : [];
            let relChild = entry.relationships ?
                entry.relationships.filter(r => r[1] &&
r[1].toLowerCase().includes('child')).map(r => r[0]) : [];
            let relRelated = entry.relationships ?
                entry.relationships.filter(r => r[1] &&
r[1].toLowerCase().includes('related')).map(r => r[0]) : [];

            // Build the detailed view
            let relationshipsHtml = '';
            if (relParent.length || relChild.length || relRelated.length) {
                relationshipsHtml = '<div style="margin-bottom:16px;">
<strong>Relationships:</strong><ul style="margin-left:15px;">';

                if (relParent.length) {
                    relationshipsHtml += `<li><span
style="color:#3b8ded;">Parent: </span>`;
                    relParent.forEach((parent, index) => {
                        relationshipsHtml += `<a target="_blank"
href="https://cwe.mitre.org/data/definitions/${parent.replace('CWE-
','')}.html" style="color:#63a4ff;text-decoration:underline;">${parent}</a>`;
                        if (index < relParent.length - 1) relationshipsHtml +=
', ';
                    });
                    relationshipsHtml += '</li>';
                }

                if (relChild.length) {
                    relationshipsHtml += `<li><span
style="color:#3b8ded;">Children: </span>`;
                    relChild.slice(0, 3).forEach((child, index) => {
                        relationshipsHtml += `<a target="_blank"
href="https://cwe.mitre.org/data/definitions/${child.replace('CWE-','')}.html"
style="color:#63a4ff;text-decoration:underline;">${child}</a>`;
                        if (index < Math.min(relChild.length, 3) - 1)
relationshipsHtml += ', ';
                    });
                    if (relChild.length > 3) relationshipsHtml += '...';
                    relationshipsHtml += '</li>';
                }

                if (relRelated.length) {
                    relationshipsHtml += `<li><span
```

```
style="color:#3b8ded;">Related: </span>`;
                    relRelated.slice(0, 2).forEach((related, index) => {
                        relationshipsHtml += `<a target="_blank"
href="https://cwe.mitre.org/data/definitions/${related.replace('CWE-
','')}.html" style="color:#63a4ff;text-decoration:underline;">${related}</a>`;
                        if (index < Math.min(relRelated.length, 2) - 1)
relationshipsHtml += ', ';
                    });
                    if (relRelated.length > 2) relationshipsHtml += '...';
                    relationshipsHtml += '</li>';
                }

                relationshipsHtml += '</ul></div>';
            }

            // Build mitigations HTML
            let mitigationsHtml = '';
            if (entry.mitigations && entry.mitigations.length > 0) {
                mitigationsHtml = `<div style="margin-bottom:16px;">
<strong>Key Mitigations:</strong><ul style="margin-
left:15px;color:#42d392;">`;
                entry.mitigations.slice(0, 3).forEach(mitigation => {
                    mitigationsHtml += `<li>💡 ${htmlEncode(mitigation)}
</li>`;
                });
                mitigationsHtml += '</ul></div>';
            }

            // Build examples HTML
            let examplesHtml = '';
            if (entry.examples && entry.examples.length > 0) {
                examplesHtml = `<div style="margin-bottom:16px;">
<strong>Examples:</strong><ul style="margin-left:15px;color:#fca311;">`;
                entry.examples.slice(0, 2).forEach(example => {
                    // Truncate long examples
                    const truncatedExample = example.length > 100 ?
example.substring(0, 100) + '...' : example;
                    examplesHtml += `<li>📝 <code
style="background:#263159;padding:2px 6px;border-
radius:3px;">${htmlEncode(truncatedExample)}</code></li>`;
                });
                examplesHtml += '</ul></div>';
            }

            // Source indicator
            let sourceHtml = '';
```

```
            if (entry.source === 'scraped') {
                sourceHtml = '<div style="margin-top:12px;padding:8px
12px;background:#263159;border-radius:6px;font-size:0.9em;color:#94a3b8;">
<span style="color:#42d392;">✓</span> Live data from MITRE</div>';
            }

            document.getElementById('cweDetailPanel').innerHTML = `
                <div>
                    <h2 style="color:#63a4ff;margin-top:0;margin-
bottom:12px;">${code}: ${htmlEncode(entry.name)}</h2>

                    <div style="margin-bottom: 18px; color:#e2e8f2;">
                        <strong>Definition:</strong>
                        <div style="margin-top:8px;line-
height:1.5;">${htmlEncode(entry.description || 'No definition available.')}
</div>
                    </div>

                    ${mitigationsHtml}
                    ${examplesHtml}
                    ${relationshipsHtml}

                    <div style="margin-top:18px;color:#a9adc1;font-
size:0.97em;">
                        <a
href="https://cwe.mitre.org/data/definitions/${code.replace('CWE-','')}.html"
                           target="_blank"
                           style="color:#63a4ff;text-decoration:underline;">
                            View official CWE documentation ↗
                        </a>
                    </div>

                    ${sourceHtml}
                </div>
            `;

        } catch (error) {
            console.error(`Failed to render CWE ${code}:`, error);
            showErrorState(code, error.message);
        }
    }

    // === Click-binding for CWE list items: show details on select ===
    function hookList(id) {
        const list = document.getElementById(id);
        if (!list) return;
```

```
        Array.from(list.children).forEach(li => {
            li.onclick = async function() {
                // Update visual selection
                Array.from(list.children).forEach(lii => {
                    lii.style.background = '';
                    lii.style.borderLeft = '4px solid transparent';
                    lii.style.color = '#e3e7f2';
                });

                this.style.background = '#212d46';
                this.style.borderLeft = '4px solid #63a4ff';
                this.style.color = '#63a4ff';

                // Render the detail (this will handle loading)
                await renderDetail(this.dataset.cwe);
            };
        });

        // Auto-select first item if available
        if (list.children.length) {
            list.children[0].click();
        }
    }

    // Hook up both lists
    hookList('cweListColumn');
    hookList('cweListFull');

    // Toggle between key CWEs and full list
    document.getElementById('toggleShowAllCwe').onclick = function() {
        let showAll = this.innerText.includes('Show All');

        document.getElementById('cweListColumn').style.display = showAll ?
'none' : '';
        document.getElementById('cweListFull').style.display = showAll ? '' :
'none';
        document.getElementById('cweListModeTip').innerText = showAll ? '(Full
Catalog)' : '(Your Key Set)';
        this.innerText = showAll ? "Show Only Key CWEs" : "Show All CWEs";

        // Auto-select first item in the newly visible list
        let list = showAll ? document.getElementById('cweListFull') :
document.getElementById('cweListColumn');
        if (list && list.children.length) {
            list.children[0].click();
```

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