Cruise Ship Inspection Scores

Architecture Design Document

SafePort Portal • Health Canada • Power Pages + Dataverse

Component	Ship Scores Browse Page
Files	shipScores.js (1,356 lines) + shipScores.css (~280 lines)
Data Sources	ethi_vessels + incidents (OData Web API)
Pattern	Two-tier disclosure tree with lazy inspection loading
A11y	WCAG 2.1 AA • native <details> semantics • WET4 focus rings
Date	2026-02-26

Table of Contents

1

System Overview

The Cruise Ship Inspection Scores page is a **public-facing, read-only browse interface** within the SafePort Portal. It displays Health Canada cruise ship inspection results in a hierarchical tree: **Cruise Lines > Vessels > Inspection History** (last 5 years). Data is loaded from Dataverse via OData Web API with lazy per-vessel inspection loading.

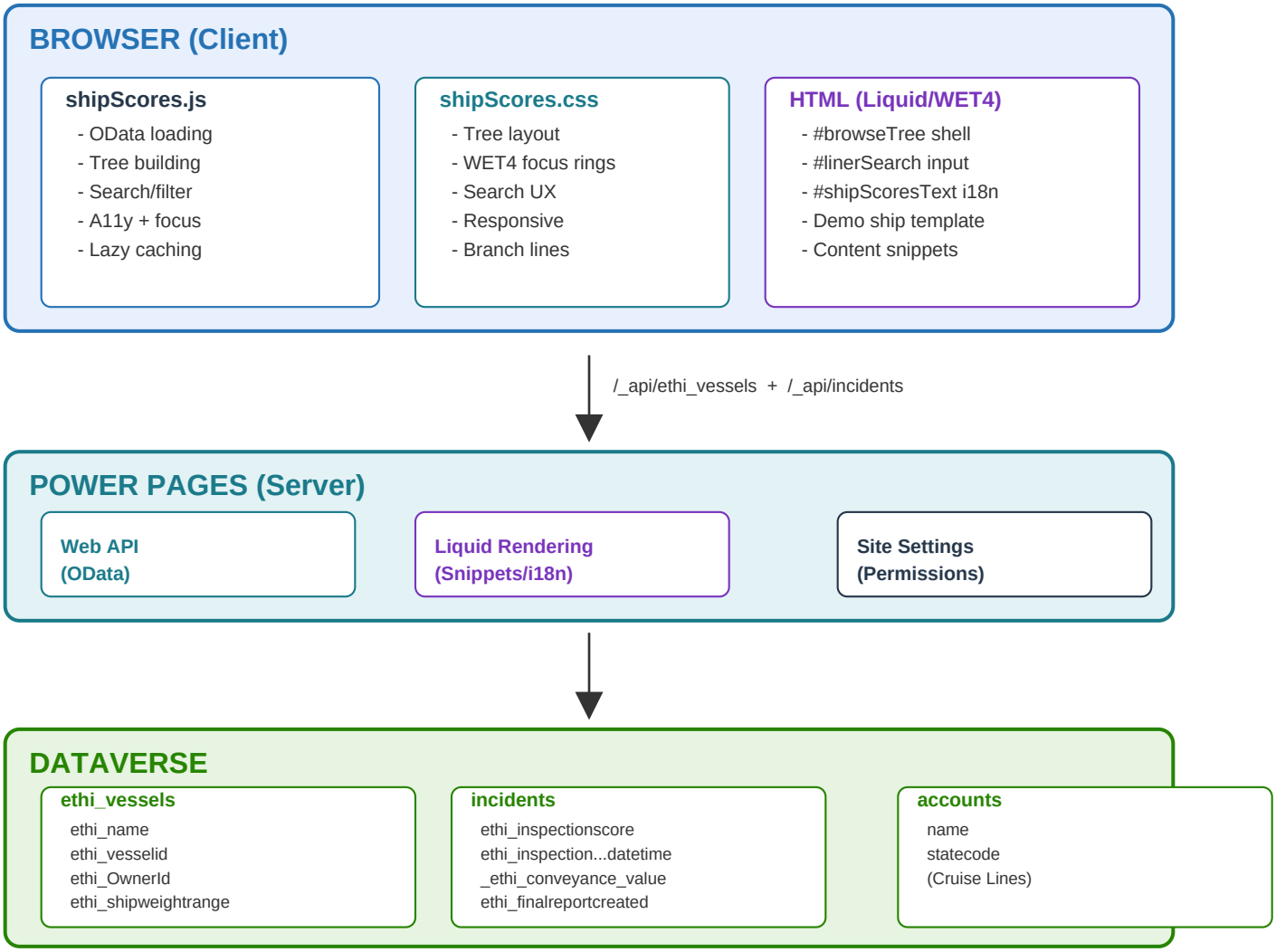

Figure 1: Three-tier system architecture — Browser, Power Pages, Dataverse

2

Component Architecture

The `shipScores.js` IIFE (1,356 lines) contains six logical modules: utilities, DOM builders, accessibility, data loading, interaction handlers, and search. All state is private to the IIFE except three functions exposed on `window.*` for cross-module communication.

Module	Functions	Responsibility
Utilities	12	Helpers: qsa, qs, textOf, isFrench, dateOnly, formatScore, GUID ops
DOM Builders	5	buildLinerNode, buildShipNode, template cloning, normalization
Accessibility	5	Tabbable summaries, region semantics, focus mgmt, title sync
Data Loading	8	Vessel OData, incident lazy load, cache, renderers
Interaction	4	Ship/liner toggle handlers, tab routing, collapse
Search	1 (complex)	Input binding, cascading filter, i18n status strings

Initialization Sequence

Step	Function	Purpose
1	`syncDocumentTitle()`	Set document.title from h1 for proper page title announcement
2	`setupDebug()`	Attach focus/tab diagnostic listeners (when DBG=true)
3	`ensureSummariesTabbable()`	Add tabindex="0" to all summary elements
4	`collapseAllOnLoad()`	Close all details elements
5	`setupLiners()`	Bind liner toggle handlers (collapse children on close)
6	`setupShips()`	Bind ship toggle + lazy load on expand
7	`setupTabRouting()`	Native tab order (no interception needed)
8	`setupSearch()`	Bind search input, Escape key, i18n status
9	`primeShipDetailsTemplateCache()`	Clone demo DOM before tree clear
10	`loadLinersAndShipsFromVessels()`	ASYNC: OData fetch, build tree, re-bind

3

Data Flow & OData Queries

Data loading happens in two phases. **Phase 1** fetches all qualifying vessels on page load (single OData call with $expand for owner and inline incident check). **Phase 2** lazily loads inspection history per vessel when the user expands a ship node. Results are cached in-memory to avoid repeat API calls.

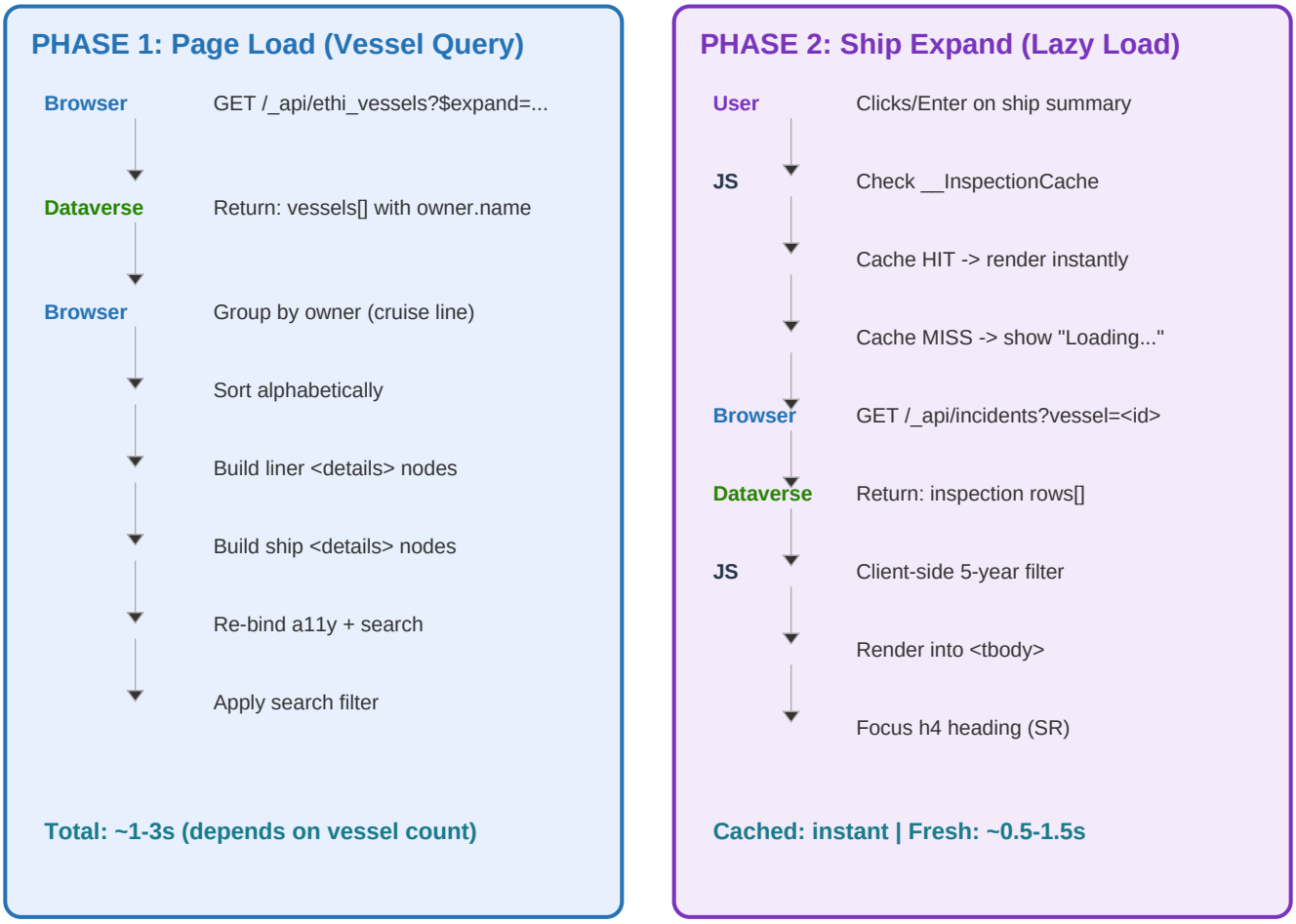

Figure 2: Two-phase data flow — Page Load (left) and Ship Expand with lazy loading (right)

OData Query Filters

Both queries filter for active records with finalized reports, qualifying inspection types (Routine Announced + Unannounced), and full inspection scope. The vessel query uses `$expand` to inline-check for qualifying incidents and fetch the owner account name. The incident query uses a GUID literal fallback: first tries `guid'...'` syntax, then retries with raw GUID on HTTP 400.

Filter	Vessel Query	Incident Query
statecode	eq 0	eq 0
statuscode	(via any())	ne 6

ethi_inspectionscore	ne null (any)	ne null
ethi_inspectionscope	786080000 (any)	786080000
ethi_finalreportcreated	ne null (any)	ne null
Inspection type GUIDs	Announced OR Unannounced	Announced OR Unannounced
Last 5 years	LastXYears server-side	Client-side isWithinLastYears()
Owner check	ethi_OwnerId/name ne null	N/A

4

Tree UI Structure

The browse tree uses native HTML `<details>` / `<summary>` elements for zero-dependency disclosure. Semantic headings (h2 liner, h3 ship, h4 section) are placed **inside** summary elements to maintain proper heading hierarchy for screen reader navigation.

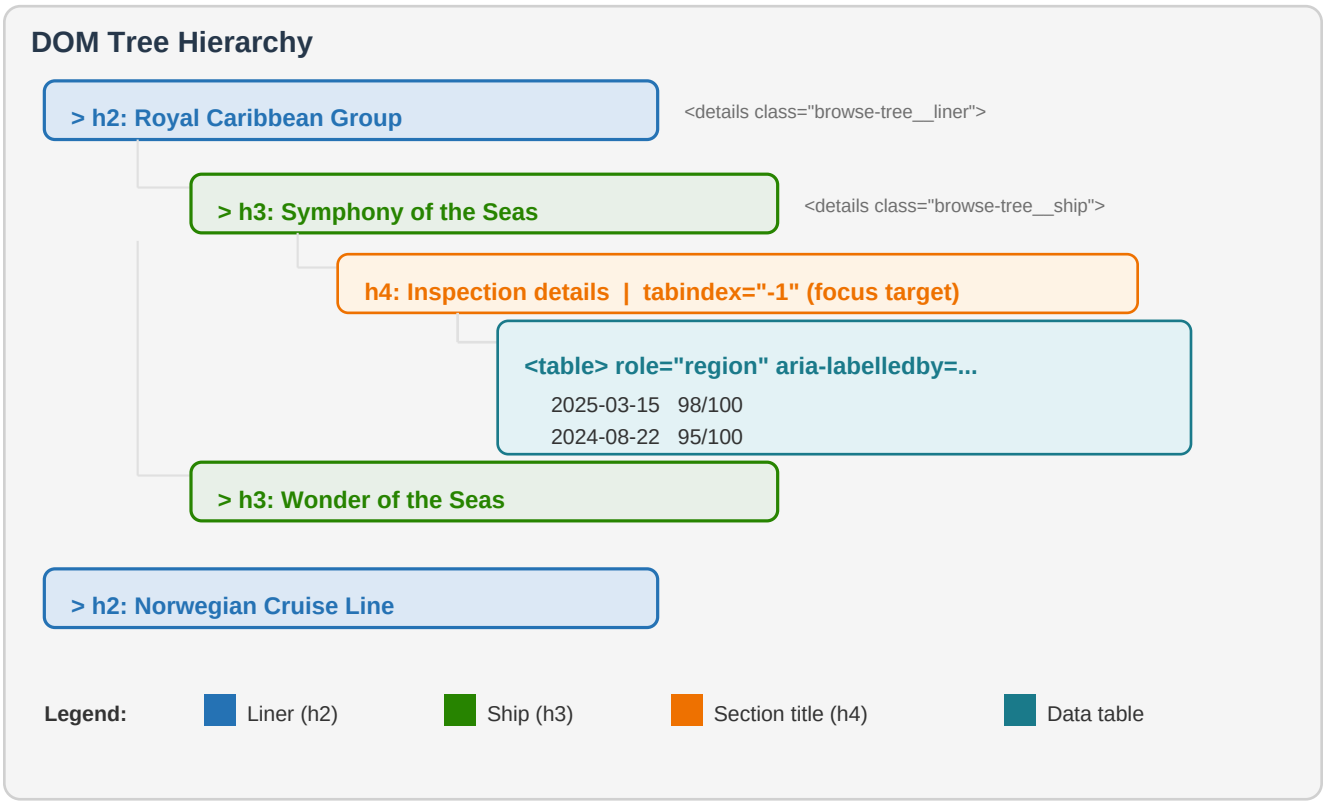

Figure 3: DOM tree hierarchy with heading levels and ARIA wiring

Template Strategy

Ship details DOM is created via **template cloning**, not string concatenation. On init, the first server-rendered `.ship-details` element is captured via `cloneNode(true)`. Each new ship gets a fresh clone with unique IDs generated for title elements and aria-labelledby re-wired. This preserves Liquid snippet text (bilingual labels) without duplicating i18n strings in JavaScript.

Step	Function	What Happens
1. Prime	`primeShipDetailsTemplateCache()`	Clone first .ship-details from server HTML
2. Clone	`getShipDetailsTemplate().cloneNode(true)`	Fresh copy per ship node
3. Fix IDs	`makeShipDomId(prefix)`	Generate unique IDs for titles
4. Wire ARIA	`setAttribute("aria-labelledby", id)`	Connect sections to titles
5. Fill data	`fillTypeAndWeight() + renderHistory()`	Populate on expand

5

Lazy Loading & Cache

Inspection history uses a per-vessel in-memory cache. On first expand, a loading row appears while the OData query runs. Subsequent expands render instantly from cache. Failed requests do NOT mark the cache as loaded, allowing retry on next expand. Concurrent requests are deduplicated via shared Promise reference.

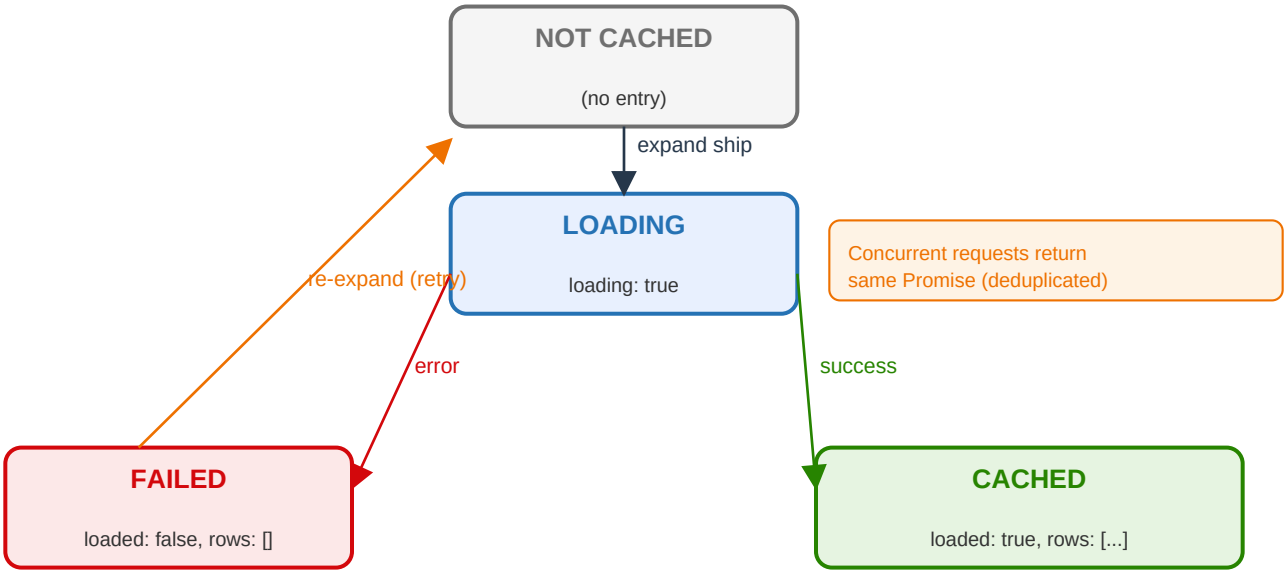

Figure 4: Inspection cache state machine — NOT CACHED > LOADING > CACHED/FAILED

Cache State	loaded	loading	rows	Behavior
Not cached	-	-	-	No entry exists; fetch on expand
Loading	false	true	[]	Fetch in progress; dedup via Promise
Cached	true	false	[...]	Render instantly; no API call
Failed	false	false	[]	Retry on next expand

6

Search & Filtering

The search input filters the tree client-side using cascading match logic: if a liner name matches, all its ships are shown; if only ship names match, the parent liner is shown with matching ships. Status messages use snippet-driven i18n templates with `{{liners}}` and `{{ships}}` placeholders.

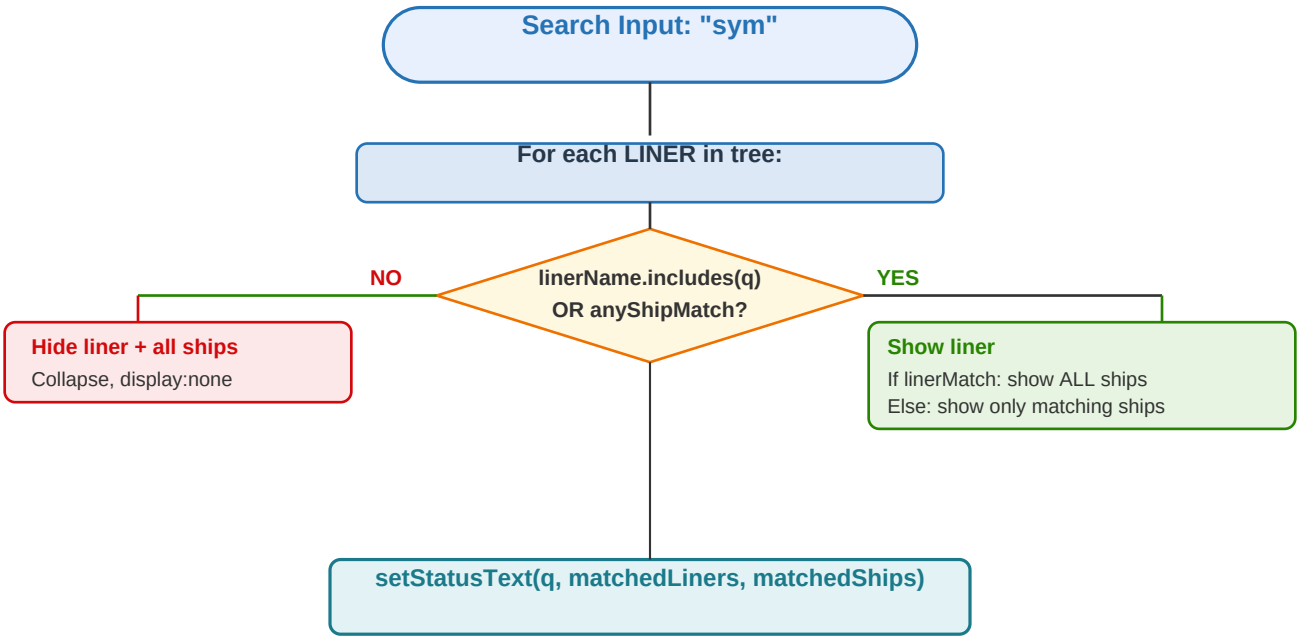

Figure 5: Search filter logic — cascading match from liner to ship level

State	data-status-* attribute	Example Text
Loading	data-status-loading	Loading cruise ship data...
Empty dataset	data-status-empty	No cruise ship data available.
Results	data-status-template	{{liners}} cruise line(s). {{ships}} ship(s).
No matches	data-status-none	No matches found.
Cleared	data-status-cleared	(empty string)

7

Accessibility Architecture

Disclosure Semantics

The tree uses native `<details>/<summary>` elements for built-in keyboard and screen reader support. **No role="button"** is added to summary elements — this would force NVDA into focus mode and prevent arrow-key browsing of expanded content.

Feature	Implementation	WCAG
Keyboard nav	tabindex="0" on summaries; Enter/Space toggles	2.1.1
Heading hierarchy	h2 (liner) > h3 (ship) > h4 (section)	2.4.6
Page title	syncDocumentTitle() from h1	2.4.2
Focus management	focusShipContent() on expand -> h4 heading	2.4.3
Region labeling	role="region" + aria-labelledby	1.3.1
Focus visible	WET4 blue ring (3px outline + 4px halo)	2.4.7
No traps	Native tab order, no custom interception	2.1.2
Orphan cleanup	Remove invalid aria-labelledby refs	4.1.2

Focus Flow on Ship Expand

1. User presses Enter on ship summary → **2.** toggle event fires → **3.** "Loading..." row shown → **4.** OData fetch completes → **5.** Table rendered into tbody → **6.** `focusShipContent()` sets focus to h4 heading (tabindex="-1", programmatically focusable) → **7.** Screen reader announces heading text, user can arrow-key through table

Tab Order Design (2026-02-25)

Info and history sections are **NOT tab stops**. They contain static data tables accessible via screen reader reading mode and table navigation commands. Tab flows naturally: `summary > summary > summary`. This prevents "tab trap" perception where users must tab through all table cells.

8

CSS Architecture

Design System Variables

Variable	Value	Purpose
`--ship-indent`	`2.5rem`	Ship node left offset (child indentation)
`--branch-x`	`1.25rem`	Vertical branch line x-position
`--inner-indent`	`1rem`	Content indent inside expanded ship
`--node-icon-gap`	`0.55rem`	Gap between triangle marker and label
`--wet-focus-blue`	`#2b6cb0`	WET4 focus ring outline color
`--wet-focus-halo`	`#bcdcff`	WET4 focus ring outer halo color

Font Size Hierarchy

Element	Size	Visual Weight
Liner name (h2)	1.55rem	Largest — primary grouping
Ship name (h3)	1.32rem	Secondary grouping
Section title (h4)	1.28rem	Content heading
Detail text (dt/dd)	1.28rem	Body content
Table header (th)	1.21rem	Slightly smaller than body
Table body (td)	0.85rem	Smallest — data density

9

Bilingual Support

Language detection uses `document.documentElement.lang` (set by Power Pages per page language), not browser locale. Content sources are distributed across multiple layers to ensure full bilingual coverage.

Content Type	Source	Mechanism
Ship/liner names	Dataverse	Data (language-neutral)
Section titles	Liquid snippets	Server-rendered HTML, cloned by template
"Cruise ship" label	JS function	cruiseShipLabel() returns EN/FR
Loading/empty/error	JS function	isFrench() ternary
Search status	HTML data attributes	#shipScores_i18n data-status-*
Table headers	Liquid snippets	getShipScoresText() reads #shipScoresText

10

Code Quality Review

Strengths

Area	Rating	Evidence
Idempotent binding	5/5	data-ship-bound="1" prevents double-binding on re-init
Error resilience	4/5	try-catch around DOM ops, GUID literal fallback on 400
Cache design	5/5	Dedup, retry on error, instant re-render on hit
A11y semantics	5/5	No role="button", orphan cleanup, heading hierarchy
Template reuse	4/5	DOM cloning preserves Liquid text; unique ID gen
Bilingual support	4/5	Snippet-driven strings, page-level lang detection
Search	4/5	Cascading match, template-based status i18n

11

Issues & Recommendations

High Priority

H1: Dead Code Cleanup — ~80 lines of commented-out code (old queries, ship pre-filtering). Remove all commented code; version control preserves history.

H2: Vessel Load Error UX — If primary vessels query fails, user sees empty tree with no explanation. Add user-visible error message with role="alert" in the tree container.

H3: Search Status ARIA Live — #linerSearchStatus may not have aria-live="polite". Screen readers won't announce search result counts without this attribute.

Medium Priority

ID	Issue	Recommendation
M1	No eTHIDiagnostics integration	Match SSI/GI logging pattern with createLogger()
M2	Info section commented out	Restore vessel info box OR remove fillTypeAndWeight()
M3	Global window.* functions	Consolidate into window.ShipScores namespace
M4	No loading visual indicator	Add skeleton/spinner to tree during fetch
M5	Table caption fully hidden	Consider visible date format hint for sighted users

Low Priority

ID	Issue	Recommendation
L1	Duplicate webapi check pattern	Use eTHIDataverse.safeAjax (preferred) via shared doOdataGet()
L2	localeCompare without locale	Specify locale for French accent handling
L3	CSS !important usage	Document justifications for GCWeb overrides

12

Testing Checklist

Functional Testing

#	Test Case	Expected Result
F1	Page loads with vessels	Liners + ships appear, all collapsed
F2	Expand liner	Child ships visible, all collapsed
F3	Expand ship	"Loading..." then inspection table renders
F4	Collapse liner	All child ships collapse
F5	Re-expand cached ship	Table renders instantly (no loading)
F6	Search "royal"	Matching liners expand, non-matching hidden
F7	Search "symphony"	Ship matches, parent liner auto-expands
F8	Clear search (Escape)	All liners visible, all collapsed
F9	Empty dataset	Status shows empty message
F10	OData vessel failure	Error message visible in tree
F11	OData incident failure	"Unable to load..." in table
F12	GUID literal 400 fallback	Retries with raw GUID, succeeds

Accessibility Testing

#	Test	AT	Expected
A1	Tab through tree	Keyboard	Summary > Summary > Summary
A2	Enter on summary	Keyboard	Toggles open/closed
A3	Page title	NVDA	Announces page title from h1
A4	Expand ship	NVDA	Focus moves to section heading
A5	Table navigation	NVDA	Ctrl+Alt+Arrow navigates cells
A6	Search type	NVDA	Status announced (aria-live)
A7	Heading nav (H key)	NVDA	h2 > h3 > h4 hierarchy
A8	VoiceOver rotor	macOS VO	Headings list shows hierarchy
A9	Swipe navigation	iOS VO	Linear: liner > ship > content
A10	Focus visible	All	Blue WET4 ring on summaries

Overall Rating: 4.5 / 5

Strong architecture with excellent caching, native disclosure semantics, and well-designed accessibility. Main improvements needed: dead code cleanup, vessel load error UX, and search status aria-live. The lazy loading pattern and template cloning strategy are particularly well-designed for the Power Pages platform.