Web Scraping Lessons Learned & Pre-flight Checklist

Here's a list of key learnings and checkpoints derived from our HCAD scraping journey, framed as a reusable guide:

Web Scraping Project: Pre-Flight & In-Flight Checklist*(Lessons from the HCAD Project)*

I. Initial Site Reconnaissance & Understanding (The "Target Whisperer" Phase - Dr. Sharma)

1. **Manual Exploration is Non-Negotiable:**
   - **Action:** Before writing any code, *manually* go through the entire user flow you intend to automate multiple times on the target website.
   - **HCAD Lesson:** Understanding the search form, how results are displayed (list, direct to detail, "no results" page), and the structure of detail pages was crucial.
   - **Time Saver:** Could have identified the iframe earlier with more thorough initial DOM inspection.
2. **Identify the True Data Source:**
   - **Action:** Use browser DevTools (Network tab) extensively. Is data loaded with the initial HTML, fetched via XHR/Fetch (APIs), or rendered by client-side JavaScript?
   - **HCAD Lesson:** The advanced search form was within an iframe, and results were also loaded within that iframe's context, even if the main page URL didn't always reflect a full navigation. Hitting an API directly wasn't an obvious option here, making browser automation necessary.

- **Time Saver:** If an API exists, it's almost always better than UI scraping. Always hunt for APIs first.

3. **Check for iFrames EARLY:**
    - **Action:** When inspecting key elements (forms, results areas), always look "up" the DOM tree in DevTools for `<iframe>` tags.
    - **HCAD Lesson:** This was our biggest hurdle. The search form AND results were inside an iframe.
    - **Time Saver:** This should be one of the first structural things to check after confirming URL.
4. **Understand Form Submission Mechanics:**
    - **Action:** Is it a standard HTML form POST/GET? Or does JavaScript handle the submission (e.g., an `onclick` event on a button that makes an AJAX call)?
    - **HCAD Lesson:** The form within the iframe submitted and reloaded the iframe's content. Playwright's `click()` handled this, but understanding it informed *which context* to expect results in.
5. **Nature of Navigation/Content Updates:**
    - **Action:** After an action (like a search click), does the main browser URL change? Does only a part of the page update (AJAX)? Does an iframe reload?
    - **HCAD Lesson:** The main page URL (`hcad.org/...`) often stayed the same, while the iframe (`public.hcad.org/...`) handled the search and results. This dictated our `wait_for_selector` strategy instead of `expect_navigation` on the main page for results.

II. Selector Strategy & Resilience (Zero's Domain)

1. **Prioritize Stable Selectors:**
    - **Action:** Prefer `id`s, then meaningful `class` names, `data-*` attributes, or ARIA roles. Avoid relying on `xpath=/html/body/div[2]/div[1]/...` or very generic tags with indices if possible.

- **HCAD Lesson:** We used `input[name="..."]` which is good. For tables, `table.bgcolor_1` was much better than just `table`.
- **Time Saver:** Investing time in finding good, semantic selectors upfront saves countless hours of fixing broken scrapers later.

2. **Content-Based Text Selectors (with caution):**
    - **Action:** Playwright's `text=` or `:has-text()` can be powerful but use for relatively stable text.
    - **HCAD Lesson:** `p:has-text('No records match your search criteria.')` was effective.

III. Engagement & Evasion Tactics (Mac Chen's Area - though less critical for HCAD so far)

1. **User-Agent & Browser Context:**
    - **Action:** Start with Playwright's default. If issues arise, use a common, real User-Agent string. Consider setting viewport, locale, timezone.
    - **HCAD Lesson:** Default seemed okay, but we added more context details as a good practice.
2. **Rate Limiting & Delays (Professor Zhang's Politeness):**
    - **Action:** Always include delays between requests, especially in loops or when hitting multiple pages. Start conservatively.
    - **HCAD Lesson:** We have `time.sleep()` at various points. Essential for not overloading HCAD.

IV. Data Flow & Logic (Your Project-Specific Strategy)

1. **Understand Input Data Structure Thoroughly:**
    - **Action:** Know exactly what columns your input CSV will have and how they relate (e.g., multiple party rows per transaction).
    - **HCAD Lesson:** The confusion around `rp_grantee_full_names_list` being empty stemmed from the interaction between the QA sample's content and the preprocessing logic.
    - **Time Saver:** A clear data dictionary for Script 3's output would have helped anticipate preprocessing needs for Script 4.

2. **Preprocessing is Your Friend:**
    - **Action:** If input data isn't in the ideal shape for the main processing loop, preprocess it once upfront.
    - **HCAD Lesson:** Consolidating grantee names onto a primary transaction row *before* the HCAD search loop was a key refinement.
3. **Tiered Logic for Searching:**
    - **Action:** If a site's search is finicky or data quality varies, start with precise queries and broaden methodically.
    - **HCAD Lesson:** Our T1 → T4 → Fallback approach is a direct application of this.
4. **Handling Multiple Results - Disambiguation Strategy:**
    - **Action:** Plan how you'll pick the "best" match if a search returns multiple candidates. This often requires fetching more details for comparison.
    - **HCAD Lesson:** Our `choose_best_from_multiple` function with scoring based on legal components and (in future) area/ownership is key. The realization that we *need* full legal from detail pages for good scoring was important.

V. Debugging & Iteration

1. **Iterate with Small, Known Test Cases:**
    - **Action:** Don't try to boil the ocean. Test core logic with 1-3 representative input rows first.
    - **HCAD Lesson:** Focusing on `RP-2025-3548` (once the full data was used for its input) helped us validate grantee aggregation and fuzzy matching for a known good case.
    - **Time Saver:** This is *the* biggest time saver. Avoids long waits for full runs to find simple bugs.
2. **Liberal Use of `print()` and Debugging:**
    - **Action:** Print variable values, DataFrame shapes, types, and snippets of page content at critical junctures.

  - **HCAD Lesson:** Our `DEBUG:` messages were vital in tracing data flow and identifying issues like the iframe, selector failures, and CSV parsing problems.

3. **Screenshots and HTML Dumps on Error:**
   - **Action:** Automatically save these when exceptions or unexpected states occur.
   - **HCAD Lesson:** The screenshot showing the iframe, and later the HTML dump confirming the `<iframe>` tag, were pivotal.

4. **Understand Playwright's Waiting Mechanisms:**
   - **Action:** Know the difference between `wait_until="domcontentloaded"`, `"load"`, `"networkidle"`. Use `wait_for_selector` (with `state='visible'`) when you need an element to be interactable, not just in the DOM.
   - **HCAD Lesson:** The switch from `expect_navigation` to `wait_for_selector` (for result indicators after the iframe click) was key to overcoming timeouts.

If I had this checklist when we started HCAD, I would have:

- **Inspected for iFrames much earlier** (Point #3). This would have saved significant time diagnosing the "element not visible" errors.
- **Been more rigorous about the expected input data structure for Script 4 from the outset** (Point #10), which would have made the grantee consolidation logic clearer sooner.
- **Pushed for targeted single-record tests even earlier** (Point #14) before attempting broader runs.

This checklist isn't exhaustive, but it covers many of the critical points we encountered and solved in this HCAD project. It should serve as a good reminder for future scraping endeavors!