Positive Feedback on Your Code

Your Advanced Search Hub is a well-structured, self-contained HTML/JS/CSS application that's already quite modularized in the JavaScript section (e.g., separated concerns like rendering functions, state management, and event handlers). It provides a clean UI for building complex queries (dorks) across engines, with thoughtful features like templates, history, favorites, and safe mode. The use of localStorage for persistence is efficient, and the CSP meta tag shows good security awareness. It's responsive and user-friendly, especially for OSINT or research tasks.

Suggested Improvements

Here are targeted improvements to enhance code quality, performance, usability, and maintainability. I've prioritized them based on impact.

1. **UI/UX Enhancements**:
   - **Debounce Input Events**: The `renderPreview()` function triggers on every keystroke in the search query or regex fields. Add a debounce (e.g., 300ms delay) to reduce unnecessary re-renders:

     javascript Inklappen Terugloop Uitvoeren Kopiëren

     ```
     function debounce(fn, delay) {
       let timer;
       return (...args) => { clearTimeout(timer); timer = setTimeout(() => fn(...args), delay); };
     }
     dom.searchQueryInput.addEventListener('input', debounce(() => { checkSearchButtonState(); render
     ```

     This improves performance on slower devices.
   - **Accessibility (ARIA and Focus)**: Add more ARIA labels/roles (e.g., for engine chips: `aria-label="Select ${engine.name}"`). Ensure keyboard navigation works fully in sub-options. Test with screen readers.
   - **Error Handling**: In `buildQueryString()`, handle invalid dates or empty required fields more gracefully (e.g., highlight inputs). Expand `showToast` for specific errors like "Invalid domain format".
   - **Mobile Optimization**: The grid layouts are good, but test on small screens—e.g., make category buttons stack vertically if needed. Add touch-friendly tap targets (increase padding on buttons).
   - **Theme Toggle**: Add a dark/light mode switch (store in localStorage). Use CSS variables for colors to make it easy to swap.

2. **Code Quality and Maintainability**:

- **Further Modularization**: While JS is organized, split into logical modules if expanding (e.g., separate files for `renderers.js`, `state.js`, `handlers.js`). Use ES modules with `<script type="module">` for future-proofing.
- **Comments and Documentation**: Add more inline comments, especially in complex functions like `buildQueryString()`. Consider JSDoc for functions.
- **Performance for Large Data**: History/favorites are capped at 50, but for templates, add a limit (e.g., 20). Use virtual scrolling if lists grow.
- **Security Review**: CSP is solid, but test for XSS (e.g., sanitize user inputs in templates). Avoid `innerHTML` where possible—use `textContent` or DOM methods.
- **Bug Fixes/Noted Issues**:
  - In `renderSubOptions()`, ensure sub-options persist on category re-render.
  - `validateDates()` could check if dates are in the future/past logically.
  - DuckDuckGo/Yandex date handling is skipped—add fallback warnings.

3. **Functionality Tweaks**:
   - **Query Building**: In `decorateQueryWithTerms()`, handle edge cases like leading/trailing quotes or escaped characters better.
   - **Engine-Specific Adaptations**: Expand `operatorCheats` with more operators (e.g., Bing's `feed:` for RSS). Dynamically adjust query syntax per engine (e.g., filetype vs. ext).
   - **Testing**: Add simple unit tests (e.g., with Jest) for `buildQueryString()` to ensure consistency across engines.

Suggested New Features

Based on trends in similar tools (e.g., Google Dorking for OSINT, GitHub secret scanners like GitDorker, and advanced search apps), here are prioritized feature ideas. I've drawn from open-source projects (e.g., GitHub dork tools) and web trends (e.g., AI assistance, precision filters). Aim for 5-10 citations minimum, but focus on high-value additions.

Core Features

- **AI-Assisted Query Suggestions**: Integrate a simple AI prompt (e.g., via free APIs like Grok or OpenAI if allowed) to

suggest refined dorks based on user input. For example, for "find leaked keys", suggest `"API_KEY" filetype:env site:github.com`. This aligns with AI advancements in search tools (e.g., LinkedIn's AI-Assisted Search).

- **Dork Library/Cheatsheet Expansion**: Add a searchable library of pre-built dorks (e.g., from GitHub collections). Users can import/export dork packs. Include categories like phone numbers or CVEs.
- **GitHub-Specific Search Integration**: Add a "GitHub Dorks" category with direct API calls to GitHub search (rate-limited). Scan for leaks like secrets or vulnerabilities.
- **Result Previews/Snippets**: After searching, fetch and display snippets from results (using a proxy API to avoid CORS). This provides quick insights without opening tabs.
- **Advanced Filters and Autocomplete**: Add autocomplete for common operators/dorks in the query input. Include natural language processing (e.g., convert "find PDFs after 2020" to operators).

Additional Features

- **More Search Engines**: Add privacy-focused ones like Brave Search, Startpage, or Ecosia. Support custom engine URLs via user input.
- **Batch/Multi-Dork Execution**: Allow running multiple dorks in sequence and aggregating results (inspired by automation tools).
- **Export to CSV/JSON**: Beyond current export, add query/history export as CSV for analysis in tools like Excel.
- **PWA Support**: Add a manifest.json and service worker for offline use (templates/history work offline).
- **Community Sharing**: Allow sharing templates via URL params or a public gallery (anonymized).
- **Analytics Integration**: Track query success (e.g., via optional Google Analytics) for personal insights.
- **Voice Input**: Use Web Speech API for dictating queries.
- **Custom Categories**: Let users add/edit categories via a form, stored in localStorage.
- **SEO/Research Tools**: Add integrations for keyword volume or backlink checks (via free APIs).
- **Dark Web/OSINT Expansion**: (Ethically) Add Tor/onion search dorks or Shodan integration for devices.

These suggestions build on your tool's strengths while addressing gaps seen in similar projects. Start with low-effort ones like debounce and dork library for quick wins. If you implement any, test thoroughly!

Citations:

- Advanced Search & Filter Configurations: Enhancing Content

- Advanced Search
- Google Advanced Searching Tools
- Advanced Search Engine - Toolbox b2b
- Best Practices for Integrating Advanced Search Functions on Websites
- 19 advanced Google search operators you need to know
- Search hub for Coveo Hive search interfaces
- Unlocking the Power of Advanced Document Search Capabilities
- What Are Advanced Search Options?
- Exploring the Features of Advanced SEO Tools
- Big Adv Advancements in Search to Know
- Advanced AI-Assisted Search
- The-Osint-Toolbox/OSINT-Advanced-Searching
- techgaun/github-dorks
- google-dorks · GitHub Topics
- The Easiest Way to Find CVEs at the Moment? GitHub Dorks!
- DorkScraper
- How I Earned $200 Using GitHub Dorking
- What are the most useful dorks you use for your everyday...
- A collection of Awesome Google Dorks.