The Architecture of Retention: A Comprehensive Analysis of High-Fidelity, Offline-First Knowledge Capture Ecosystems in the Post-Pocket Era

Executive Summary

The digital knowledge landscape of late 2025 has been defined by a significant infrastructural collapse: the discontinuation of Mozilla Pocket and the acquisition-induced closure of Omnivore. These events have dismantled the default "read-later" utilities that served the global market for over a decade, forcing a fundamental re-evaluation of how digital content is captured, curated, and consumed. For the discerning knowledge worker, the primary objective has shifted from mere link aggregation—which historically led to the accumulation of "millions of random weblinks"—to "richness preservation." This concept entails the ability to capture the full semantic and visual context of an article, including high-resolution imagery, complex formatting, and typographical integrity, for reliable offline consumption.

This report provides an exhaustive analysis of the current state of the read-later market, evaluating solutions against a tripartite set of strict criteria derived from user requirements: rich content rendering, robust offline image caching, and workflow designs that mitigate the "collector's fallacy" (digital hoarding). The analysis identifies a stark bifurcation in the market between "Service-Based Ecosystems" (Readwise Reader, Raindrop.io) and "Local-First Architectures" (GoodLinks, Obsidian Web Clipper).

The investigation reveals that while subscription services offer the most frictionless capture, local-first approaches provide superior long-term data sovereignty and offline reliability. Furthermore, the report details specific technical configurations—particularly within self-hosted and markdown-based environments—that ensure images are downloaded locally rather than hotlinked, solving the historic fragility of offline reading. Ultimately, the report argues that preventing the accumulation of unread links requires a shift from passive "bookmarking" to active "triage," facilitated by specific software features like GoodLinks' automation or Readwise's "Inbox" workflow.

1. The Great Fragmentation: Market Dynamics in the Post-Pocket Era

The year 2025 will likely be recorded in the annals of internet history as the year the "Read Later" service model underwent a catastrophic, yet necessary, evolution. For nearly fifteen years, the ecosystem was dominated by a singular, monolithic utility: Pocket. Originally known as "Read It Later," Pocket defined the category, providing a ubiquitous "save button" that stripped advertisements and synchronized content across devices. Its acquisition by Mozilla in 2017 seemed to promise stability, integrating the service deeply into the Firefox browser. However, the subsequent years revealed a stagnation in innovation and a growing misalignment between the service's goal (content discovery) and the users' goal (content consumption).

1.1 The Extinction Events of 2025

The dissolution of Pocket on July 8, 2025, marked the end of an era of centralized, free-tier ubiquity.[1] Mozilla's strategic pivot, driven by a desire to focus resources on "contextual discovery" and AI-driven recommendations within the Firefox browser itself, necessitated the shutdown of the standalone Pocket applications for iOS, Android, and Kobo.[3] This decision left millions of users with a stark realization: their carefully curated libraries were tenants on a landlord's server, liable to eviction at any moment. The shutdown was not merely a service interruption; it was a destruction of workflow for users who relied on Pocket's integration with e-readers and its robust offline caching.

Compounding this disruption was the acquisition of Omnivore by ElevenLabs, which resulted in the immediate shutdown of the popular open-source platform.[4] Omnivore had risen rapidly in 2023 and 2024 as the primary refuge for users fleeing proprietary gardens. It was praised for its clean reader view, text-to-speech capabilities, and aggressive plugin development for tools like Obsidian and Logseq. Its closure demonstrated the inherent volatility of VC-backed "free forever" models. The "acquihire"—where a company is bought for its talent rather than its product—served as a harsh lesson for the self-hosted community: even open-source codebases can be abandoned if the maintainers are absorbed into closed ecosystems.

1.2 The Divergence of User Needs: Bookmarking vs. Reading

The vacuum left by these giants has clarified two distinct user needs that were previously conflated in the legacy "Read It Later" model. In the Pocket era, users dumped everything into one bucket. In the post-2025 landscape, we see a functional separation:

1. **The Archive (Bookmarking):** This is the need to store a URL and metadata for potential future reference. It is a "just in case" repository. Tools like Raindrop.io and Linkwarden excel here, managing millions of links with tagging systems and full-text search, but often failing to provide a cohesive *reading* experience.[6]
2. **The Queue (Consumption):** This is the need to strip clutter, cache content locally, and consume it in a distraction-free environment. This is the domain of Readwise Reader, GoodLinks, and Instapaper. This function requires "richness"—the preservation of the reading experience.[8]

The user's query specifically targets the latter: a workflow that ensures "richness" and a "nice article view" while offline. This distinction is critical because excellent bookmark managers often make poor offline readers (failing to cache images), while excellent readers often lack the organizational hierarchy to manage thousands of items without becoming a chaotic "hoard." The market has responded by creating specialized tools for each, forcing users to construct "stacks" rather than relying on a single app.

2. The Phenomenology of "Read Later": Hoarding vs. Reading

To address the user's specific constraint regarding "millions of random weblinks," it is necessary to analyze the psychological and design mechanisms that contribute to digital hoarding. The "Collector's Fallacy"—the belief that "collecting" information is the same as "acquiring" knowledge—is exacerbated by low-friction capture tools.

2.1 The Cognitive Load of the Infinite Queue

Legacy applications like Pocket treated every saved item as equal. A 50-page academic PDF saved for a thesis was placed alongside a 300-word recipe saved for dinner. This lack of hierarchy leads to "Queue Paralysis." When a user opens their read-later app and sees 5,000 unread items, the cognitive load required to choose what to read exceeds the motivation to read, resulting in the user closing the app and reading nothing. This is how "millions of random weblinks" accumulate: the friction of retrieval is higher than the friction of capture.

2.2 Design Patterns for Consumption

The "best way" to handle read-later workflows in 2025 involves software that actively fights this tendency through "Triage" design patterns.

- **The Inbox vs. The Library:** Modern tools like Readwise Reader differentiate between an "Inbox" (unprocessed) and a "Library" (processed). This mimics email workflows, encouraging users to touch a piece of content once—either reading it, archiving it, or deleting it—rather than letting it rot.
- **Ephemeral Storage:** Tools like GoodLinks introduce automation that can auto-delete or auto-archive content after a certain period.[9] This acknowledges that the value of news and "hot takes" decays rapidly. If an article hasn't been read in 30 days, it likely never will be.
- **Visual Richness as Motivation:** A "nice article view" is not just aesthetic; it is functional. High-fidelity typography and properly rendered images reduce the friction of reading. When an article looks like a well-typeset book chapter rather than a scraped web page, the user is more likely to engage with it.

3. Technical Analysis of "Richness" and Offline Fidelity

To recommend a superior workflow, one must understand the technical barriers to "richness." A "rich" read-later experience requires more than just scraping text; it demands the preservation of the article's *intent* and *visual context*.

3.1 The DOM Parsing Challenge

Most modern web pages are dynamic applications, not static documents. They rely on the Document Object Model (DOM) to structure content. A read-later app must utilize a parser—often a derivative of Mozilla's open-source Readability.js—to traverse the DOM and identify the "core" content (the article text) while discarding the "noise" (navbars, ads, sidebars).

- **Richness Failure Mode:** Basic scrapers flatten the DOM. They might extract the text but lose the distinction between a <blockquote> and a standard <p>, or strip the formatting from a code block (<pre><code>). This destroys the semantic richness of the content.
- **Richness Success:** Advanced parsers (like those in Readwise and GoodLinks) retain complex HTML5 elements like <figure> and <figcaption>, ensuring that images remain associated with their captions, and support LaTeX rendering for mathematical formulas.

3.2 The Challenge of Offline Image Caching

The specific requirement for a "nice article view, both when viewed online and offline (cached version)" is the most difficult technical hurdle in 2025.

- **Lazy Loading:** Most modern web pages use "lazy loading" scripts to save bandwidth. Images are only fetched when they scroll into the viewport. A basic scraper that hits the URL will often fail to trigger these scripts, resulting in saved articles with missing image placeholders or 1x1 pixel tracking gifs instead of the actual content.
- **CDN Token Expiration:** Images are often served via Content Delivery Networks (CDNs) with time-sensitive tokens in the URL. If an app merely saves the *link* to the image, that link may expire by the time the user tries to read the article offline.
- **True Offline Richness:** This requires an app to act as a crawler. It must:
  1. Parse the DOM to find image tags.
  2. Download the binary image data to local storage on the device.
  3. **Path Remapping:** Rewrite the src attributes in the HTML/Markdown to point to the local file path (e.g., file:///private/var/mobile/Containers/Data/...) rather than the remote URL.

This process is resource-intensive and difficult to synchronize across platforms, which is why many apps fail at it.

4. The Commercial Ecosystem: Service-Based

Solutions

For users prioritizing ease of use and interface polish over strict data sovereignty, commercial applications offer the most integrated experience. However, their offline reliability varies significantly.

4.1 Readwise Reader: The "Everything" Engine

Readwise Reader has emerged as the de facto successor to Pocket for "power users".[8] Unlike Pocket, which was primarily a web clipper, Reader positions itself as a comprehensive consumption engine for RSS feeds, email newsletters, EPUBs, PDFs, and Twitter threads.[10]

- **Richness Capabilities:** Reader excels at parsing. It handles complex formatting, retains code blocks, and supports YouTube transcripts and podcast transcription, highlighting the text in sync with the audio.[8] Its "Ghostreader" (AI) feature allows users to summarize or query the text, adding a layer of interactive richness.
- **Offline Performance Evolution:** Historically, Reader struggled with inconsistent offline caching. Throughout 2024, users frequently reported that images would fail to load when in airplane mode, as the app prioritized text caching over heavy media assets.[12] However, significant updates in late 2025 (specifically in September and November) introduced a completely rewritten offline engine.[15] The app now provides visual status indicators for download progress and aggressively caches document content—including images—when the app is open.[15]
- **Workflow Implication:** Reader is a subscription service (~$10/month) that unifies multiple streams. It effectively prevents "millions of random links" by separating the "Feed" (ephemeral) from the "Library" (permanent). It is the best choice for users who want to interact *actively* with text through highlighting and annotation.

4.2 GoodLinks: The Native Perfectionist

For users within the Apple ecosystem, GoodLinks represents the gold standard for "richness without bloat." It is a one-time purchase application that eschews the subscription model in favor of a robust, focused utility.[8]

- **Offline Reliability:** GoodLinks is frequently cited as having superior offline reliability compared to its competitors. Unlike apps that cache "when possible," GoodLinks explicitly saves the full rich text and images to the device immediately upon capture.[18] It utilizes iCloud to sync the *content* itself, not just the URL. This means that if the original website goes offline (link rot), the article remains fully readable on the user's iPad or iPhone in its original fidelity.[9]
- **Visual Richness:** It offers granular control over typography, margins, and line spacing, satisfying the "nice article view" requirement more effectively than almost any other app.[21] It renders articles with a focus on readability, stripping distractions while maintaining the structural integrity of the text.

- **Anti-Hoarding Automation:** Crucially, GoodLinks includes an automation engine. Users can set rules such as "Mark as Read when scrolled to bottom" or "Delete read articles after 1 week".[9] This automation is a powerful tool against the accumulation of digital clutter, keeping the reading list pruned and manageable.
- **Limitation:** It is strictly Apple-only (iOS/macOS), making it unsuitable for users with Android devices or Windows PCs.[22]

4.3 Raindrop.io: The Collector, Not the Reader

Raindrop is a phenomenal bookmark manager, arguably the best in class for organizing massive collections of links. However, functionality as a dedicated *offline* reader is secondary to its archival purpose.

- **Permanent Copy:** The Pro plan creates a "Permanent Copy" (HTML/PDF) of every bookmark.[23] This is primarily for *archival* protection against link rot.
- **Mobile Offline Issues:** Users consistently report that the mobile app requires an internet connection to navigate folders and often fails to load the "permanent copy" when in airplane mode.[24] While it caches the *list* of bookmarks effectively, it does not aggressively download and rewrite image paths for the *content* of thousands of articles for offline reading in the way GoodLinks does.
- **Verdict:** Raindrop should be used as the *archive* (the "Reference" bucket), not the *reading queue*.

5. The Self-Hosted Renaissance: Owning the Data

For users who want to ensure their library survives the next "Pocket Shutdown," self-hosted solutions offer total control. The leading contenders in 2025 are Karakeep (formerly Hoarder) and Wallabag.

5.1 Karakeep (formerly Hoarder)

Following a trademark dispute in 2025, the popular app "Hoarder" rebranded to **Karakeep**.[27] It has gained massive traction in the self-hosted community as a modern, AI-enhanced bookmark manager.

- **Richness and AI:** Karakeep differentiates itself by using AI (LLMs like OpenAI or local models via Ollama) to automatically tag and categorize content.[28] This directly addresses the "millions of random links" problem by making the hoard searchable and organized without manual effort. It parses images and text effectively, allowing users to store notes and assets alongside links.
- **Offline Reality:** As of late 2025, true offline mobile reading (cached full articles with images accessible without server connection) remains a "Planned" feature rather than a fully mature capability.[28] The mobile app acts primarily as a client for the server. If the server is unreachable (e.g., on a plane without Wi-Fi), the content retrieval can be spotty

unless specific items were recently viewed. While excellent for organization, it is not yet the robust *offline* reader that GoodLinks is.[28]

5.2 Wallabag: The Veteran Standard

Wallabag is the open-source equivalent of Pocket, having existed for over a decade. It is built on the Symfony framework and is highly stable.

- **Offline Support:** Unlike Karakeep, Wallabag has mature mobile apps (Android/iOS) that explicitly sync and cache articles for offline reading.[6] It was designed from the ground up as a "Read Later" service, not just a bookmark manager.
- **The Kobo Integration:** A unique "richness" feature of Wallabag is its ability to integrate directly with Kobo e-readers.[32] Users can push articles to their Kobo device, allowing for consumption on an E-Ink screen. This provides a superior reading experience for long-form content that LCD screens cannot match.
- **Trade-off:** The UI/UX is utilitarian and dated compared to the fluid, polished "article view" of modern apps like Reader or GoodLinks. It gets the job done but lacks the typographic elegance requested by the user.[34]

5.3 Readeck

Readeck is a newer entrant gaining favor for its clean preservation. It saves a raw copy of the article's DOM and assets, ensuring a near-perfect replica of the page as it looked when saved.[35] However, it currently lacks a native mobile app with robust offline syncing, relying instead on Progressive Web App (PWA) capabilities or third-party clients, which limits its utility for the "offline on a plane" use case compared to native apps.[36]

6. The Local-First Solution: Obsidian as the Ultimate Reader

For the user demanding "Richness," "Offline," and "No Hoarding," the most powerful (albeit manual) workflow in 2025 involves **Obsidian**. By shifting the "Read Later" queue into a local markdown vault, you eliminate the reliance on any sync service and gain absolute control over formatting and images.

6.1 The Obsidian Web Clipper (2025 Update)

Obsidian released an official Web Clipper in late 2024/2025 that revolutionized this workflow.[38]

- **Capability:** It converts web pages into Markdown, stripping ads and sidebars while retaining headers, bold text, and links. It allows for template customization, letting users define exactly what metadata (URL, Author, Date) is captured.
- **The Image Problem:** By default, web clippers often create links to remote images

(!(https://example.com/image.jpg)). This fails the "offline" test because the images require an internet connection to load.[39]

6.2 Solving the Offline Image Requirement

To ensure true offline richness in Obsidian, specific configuration is required. The research highlights two primary methods to solve this:

1. **Native Command (v1.8+):** Obsidian introduced a core command: **"Download attachments for current file"**.[41]
   - *Workflow:* The user clips the article using the official extension. Once the markdown file is in Obsidian, they trigger this command.
   - *Mechanism:* Obsidian scans the markdown file for remote image URLs, downloads the binary files to the local vault's attachment folder, and rewrites the markdown links to point to the local files (!(assets/image.jpg)).
   - *Result:* A completely self-contained, offline-readable text file with images.
2. **Plugin Automation:** For a seamless experience, users utilize plugins like **"Local Images Plus"** or **"ReadItLater"**.[43]
   - These plugins can be configured to automatically download images the moment a note is created or pasted. This removes the manual step of triggering the download command, ensuring that every clipped article is immediately "rich" and offline-ready.

6.3 Anti-Hoarding via "Inbox" Management

Obsidian encourages anti-hoarding through its folder structure. By clipping to a specific "Inbox" folder, users create a finite queue. The lack of infinite scroll and algorithmic feeds in Obsidian means the user must actively manage this folder. If the "Inbox" grows too large, the visual clutter forces the user to delete or process items, naturally curbing the accumulation of "random links."

7. Comparative Feature Matrix

The following table synthesizes the "Richness" and "Offline" capabilities of the top contenders as of late 2025.

Feature	Readwise Reader	GoodLinks	Raindrop.io (Pro)	Karakeep (Hoarder)	Obsidian (Local)	Wallabag
Primary Philosophy	Consumption Engine	Private Reader	Bookmark Manager	AI Data Hoard	Knowledge Base	Open Source Reader

Offline Text	Excellent (Cached)	Excellent (Native)	Good (Cache)	Moderate (Server)	Perfect (Local File)	Excellent (Cached)
Offline Images	Good (Improved late '25)	Excellent (Native)	Poor (Link only)	Planned / Limited	Perfect (If downloaded)	Good (Cached)
Article View	Highly Customizable	Beautiful / Clean	Functional	Modern / Clean	User Defined (CSS)	Utilitarian
Platform	Web, iOS, Android	Apple Only	All Platforms	Self-Hosted / Web	All Platforms	Web, iOS, Android, Kobo
Cost	~$10/mo (Subscription)	~$10 (One-time)	~$3/mo (Subscription)	Free (Self-Hosted)	Free	Free (Self-Hosted)
Risk of Hoarding	High (Feed focus)	Low (Auto-delete)	High (Collections)	High (Hoarding focus)	Low (Manual friction)	Moderate

8. Strategic Recommendations: Architecting the Workflow

The query asks for the "best way" to handle read-later. The analysis suggests that no single app solves the problem entirely; rather, a **workflow** is required to separate the *stream* (triage) from the *reservoir* (archive).

Recommendation A: The "Apple Native" Purist (Best Balance)

Profile: Uses iPhone/iPad/Mac. Wants zero friction, beautiful UI, and reliable offline reading. Hates subscriptions.

- **The App: GoodLinks**.[18]
- **Why:** It natively saves full article text and images to iCloud. It is strictly offline-first. It renders articles beautifully with custom typography.

- **Workflow:**
  1. **Capture:** Save to GoodLinks via Share Sheet.
  2. **Consumption:** Read offline on iPad. The rendering is superior to almost all competitors.
  3. **Triage:** Use GoodLinks' automation to **"Delete Read Articles"** or **"Delete Unread after 30 days."** This prevents the "millions of links" problem by forcing hygiene.
  4. **Archive:** If an article is worth keeping forever *after* reading, export it to Obsidian or Raindrop.
- **Richness Score:** 9/10 (Excellent rendering, but Apple exclusive).

Recommendation B: The "Power Reader" (Cross-Platform)

Profile: Uses Android/Windows/iOS. Reads significantly (PDFs, Newsletters, RSS). Willing to pay for quality.

- **The App: Readwise Reader**.[8]
- **Why:** It is the only app that successfully unifies RSS, Newsletters, and Web Articles into a single high-fidelity view. The late 2025 updates have resolved most offline image caching issues.[15]
- **Workflow:**
  1. **Capture:** Forward newsletters and save web links to Reader.
  2. **Filter:** Use "Triage" mode to swipe left/right (Archiving or Saving). Keep the "Library" distinct from the "Feed."
  3. **Consumption:** Use the "Ghostreader" AI to summarize long articles before committing to read them.
  4. **Retention:** Highlights automatically sync to Obsidian/Notion.
- **Richness Score:** 9/10 (Best-in-class parsing, high cost).

Recommendation C: The "Data Sovereign" (Local-First)

Profile: Wants absolute control. Zero chance of losing data. Willing to configure settings.

- **The App: Obsidian** with **Official Web Clipper**.[38]
- **Why:** It meets the "richness" requirement by creating a local file with downloaded images. It meets the "offline" requirement perfectly because the data lives on the device's hard drive.
- **Workflow:**
  1. **Capture:** Use the Obsidian Web Clipper extension.
  2. **Process:** Configure the clipper or use the Download attachments command to ensure images are saved locally to an assets folder.[42]
  3. **Consumption:** Read the Markdown file in Obsidian mobile.
  4. **Anti-Hoarding:** Treat the "Inbox" folder as a temporary workspace. Review it weekly. If a file isn't read, delete the markdown file.
- **Richness Score:** 10/10 (You own the raw HTML/Markdown and images).

Conclusion

The shutdown of Pocket and Omnivore has been a wake-up call for the digital ecosystem. The "best" way to handle read-later in late 2025 is no longer to rely on a passive, free cloud bucket.

For the specific requirements of **Richness**, **Nice Article View**, and **Offline Images**, **GoodLinks** is the superior technological solution for Apple users. Its architecture of downloading content locally ensures that "offline" truly means offline, and its automation tools actively fight the tendency to hoard.

For users outside the Apple ecosystem, or those requiring heavy annotation features, **Readwise Reader** is the only commercial tool that matches Pocket's legacy, provided the user accepts the subscription cost and leverages the new offline engine introduced in late 2025.

However, the only way to truly "ensure richness" regardless of platform or service availability is to adopt the **Obsidian Local-First Workflow**. By converting articles into local markdown files with downloaded images, you transform "random links" into "owned knowledge," creating the necessary friction to stop hoarding and start reading.

Works cited

1. Pocket has shut down - What you need to know | Pocket Help - Mozilla Support, accessed on December 10, 2025, https://support.mozilla.org/en-US/kb/future-of-pocket
2. Pocket (service) - Wikipedia, accessed on December 10, 2025, https://en.wikipedia.org/wiki/Pocket_(service)
3. Pocket, accessed on December 10, 2025, https://getpocket.com/pocket-and-firefox
4. Omnivore.app is joining ElevenLabs. Users have until November 15th to export their data, after which it will be deleted - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/selfhosted/comments/1geymmu/omnivoreapp_is_joining_elevenlabs_users_have/
5. Omnivore is Dead: Where to Go Next - Yury Molodtsov, accessed on December 10, 2025, https://molodtsov.me/2024/10/omnivore-is-dead-where-to-go-next/
6. 5 Best Bookmarking and Read It Later Apps - Appy Pie Automate, accessed on December 10, 2025, https://www.appypieautomate.ai/blog/best-bookmarking-and-read-it-later-apps
7. Never Lose a Link Again: How to Set Up Your Personal Web Archive with Linkwarden, accessed on December 10, 2025, https://www.enginyring.com/en/blog/never-lose-a-link-again-how-to-set-up-your-personal-web-archive-with-linkwarden
8. The 4 best read it later apps to save content in 2025 - Zapier, accessed on

December 10, 2025, https://zapier.com/blog/best-bookmaking-read-it-later-app/
9. Release Notes - GoodLinks, accessed on December 10, 2025, https://goodlinks.app/releases/
10. Need a Pocket replacement? My 6 favorite 'read it later' apps are even better | ZDNET, accessed on December 10, 2025, https://www.zdnet.com/home-and-office/work-life/need-a-pocket-replacement-my-6-favorite-read-it-later-apps-are-even-better/
11. Reader Public Beta Update #13 (Tablet Support, Improved Offline, Podcast Transcripts, Ghostreader v3, and more) : r/readwise - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1pfuwmc/reader_public_beta_update_13_tablet_support/
12. Saved Readwise articles are not available offline - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1e2p59a/saved_readwise_articles_are_not_available_offline/
13. Reader offline mode tips? : r/readwise - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1ayv8ex/reader_offline_mode_tips/
14. How have you still not solved offline? - readwise - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1l9ug2g/how_have_you_still_not_solved_offline/
15. Changelog as of November 14: Improved Offline, Layout Fixes, a Capacities Integration, & more! : r/readwise - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1p188el/changelog_as_of_november_14_improved_offline/
16. Changelog as of Sep 12: Improved Tablet & Offline Reading, Fixed Navigation & Buttons, and more! : r/readwise - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/readwise/comments/1niixhu/changelog_as_of_sep_12_improved_tablet_offline/
17. New: Offline Reading Improvements #reading #productivity - YouTube, accessed on December 10, 2025, https://www.youtube.com/shorts/8ieAeOgoBHg
18. GoodLinks – Save Articles and Read Anywhere Later - Mause Reviews, accessed on December 10, 2025, https://mausereviews.wordpress.com/2025/11/07/goodlinks-save-articles-and-read-anywhere-later/
19. GoodLinks - App Store - Apple, accessed on December 10, 2025, https://apps.apple.com/cz/app/goodlinks/id1474335294
20. GoodLinks 2.0 is out. : r/macapps - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/macapps/comments/1eew2t1/goodlinks_20_is_out/
21. Instapaper - Apps on Google Play, accessed on December 10, 2025, https://play.google.com/store/apps/details?id=com.instapaper.android&hl=en_US
22. Five Great Read-Later Apps to Replace Pocket - Lifehacker, accessed on December 10, 2025, https://lifehacker.com/tech/5-best-read-later-apps-to-replace-pocket

23. Permanent copy - Raindrop.io Help, accessed on December 10, 2025, https://help.raindrop.io/permanent-copy
24. Offline : r/raindropio - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/raindropio/comments/12usz0f/offline/
25. How to store a permanent copy for all my bookmarks? : r/raindropio - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/raindropio/comments/1nprn01/how_to_store_a_permanent_copy_for_all_my_bookmarks/
26. Offline support | Voters - Raindrop.io - Canny, accessed on December 10, 2025, https://raindropio.canny.io/feature-requests/p/offline-support
27. Hoarder is rebranding to Karakeep : r/selfhosted - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/selfhosted/comments/1js667o/hoarder_is_rebranding_to_karakeep/
28. Karakeep Docs: Introduction, accessed on December 10, 2025, https://docs.karakeep.app/
29. Karakeep, accessed on December 10, 2025, https://karakeep.app/
30. Introducing Hoarder - An open source Bookmark-Everything app with AI based tagging (mymind open source alternative) : r/selfhosted - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/selfhosted/comments/1bo9tdq/introducing_hoarder_an_open_source/
31. You can finally read your articles while being offline - Wallabag, accessed on December 10, 2025, https://wallabag.org/news/you-can-finally-read-your-articles-while-being-offline/
32. Save the web, freely | wallabag: a self hostable application for saving web pages, accessed on December 10, 2025, https://wallabag.org/
33. Choosing a new bookmarking/read later service - cyb.org.uk, accessed on December 10, 2025, https://cyb.org.uk/2025/06/05/self-hosted-bookmarking.html
34. Self-Hosting Guide to Alternatives: Pocket, Omnivore - selfh.st, accessed on December 10, 2025, https://selfh.st/alternatives/read-later/
35. What 'Read later' app is everyone using? : r/selfhosted - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/selfhosted/comments/1k04n61/what_read_later_app_is_everyone_using/
36. What are some open source alternatives? · Issue #4462 · omnivore-app/omnivore - GitHub, accessed on December 10, 2025, https://github.com/omnivore-app/omnivore/issues/4462
37. Current Favourite Read-It-Later app - Readeck : r/selfhosted - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/selfhosted/comments/1ea8w8o/current_favourite_readit later_app_readeck/
38. Highlight and capture the web in your favorite browser. The official Web Clipper extension for Obsidian. - GitHub, accessed on December 10, 2025, https://github.com/obsidianmd/obsidian-clipper

39. Clip web pages - Obsidian Help, accessed on December 10, 2025, https://help.obsidian.md/web-clipper/capture
40. Can Web Clipper save images locally? : r/ObsidianMD - Reddit, accessed on December 10, 2025, https://www.reddit.com/r/ObsidianMD/comments/1ig500j/can_web_clipper_save_images_locally/
41. Could WebClipper clip(download) the images? - Plugins ideas - Obsidian Forum, accessed on December 10, 2025, https://forum.obsidian.md/t/could-webclipper-clip-download-the-images/98532
42. Download attachments for current file should download links to images in properties, accessed on December 10, 2025, https://forum.obsidian.md/t/download-attachments-for-current-file-should-download-links-to-images-in-properties/102830
43. Obsidian Web Clipper - Basement, accessed on December 10, 2025, https://forum.obsidian.md/t/obsidian-web-clipper/53123
44. I use a plugin to download the entire website article to my local device. : r/ObsidianMD, accessed on December 10, 2025, https://www.reddit.com/r/ObsidianMD/comments/1nrjxyz/i_use_a_plugin_to_download_the_entire_website/