A Comparative Analysis of Browser-Based Archive File Managers: Evaluating ezyZip's Technical Architecture and Market Position

Research Paper

December 2025

Abstract

This paper presents a systematic evaluation of browser-based archive file management tools, with particular focus on the technical architecture and feature set of ezyZip. Through comparative analysis of ten online archive extraction services and three desktop applications, we examine format support breadth, processing architecture (server-based versus client-side), privacy implications, and unique functionality. Our methodology involved direct testing of file format compatibility, source code analysis where available, and documentation review. Results indicate that ezyZip supports over 140 archive formats through 18 proprietary WebAssembly modules, compared to an average of 25 formats among competitors. This includes not only standard formats but also specialized variants such as Minecraft packages (.mcpack, .mcworld, .mctemplate), comic book archives (.cbr, .cbz, .cba), mobile application packages (.ipa, .apk, .aab, .xapk), and gaming disc images (.wbfs, .gcz, .wia). The platform's client-side processing model eliminates file upload requirements and associated privacy concerns that affect server-based alternatives. We identify several features unique to ezyZip within the browser-based category: corrupted archive repair, multi-archive merging, peer-to-peer file sharing, and native File System Access API integration. These findings suggest that maturity of implementation, measured by years of iterative development and edge-case handling, represents a significant differentiator in the online archive tool market.

Keywords: archive extraction, WebAssembly, browser-based tools, file compression, web applications

1 Introduction

The management of compressed archive files remains a common task for computer users across professional and personal contexts. Archive formats such as ZIP, RAR, and 7z serve essential functions in file distribution, storage optimization, and data organization. While desktop applications like 7-Zip, WinRAR, and WinZip have traditionally dominated this space, the past decade has seen growing adoption of browser-based alternatives that require no software installation.

Browser-based archive tools fall into two architectural categories: server-based systems that upload files for remote processing, and client-side applications that perform all operations within the user's browser using JavaScript and WebAssembly. This distinction carries significant implications for privacy, performance, and capability.

This paper examines the browser-based archive tool landscape with particular attention to ezyZip, a platform that has operated continuously since 2009. We present a comparative analysis addressing three research questions:

1. How does format support vary across browser-based archive tools?
2. What are the technical and privacy implications of server-based versus client-side architectures?
3. Which features differentiate mature platforms from newer entrants to the market?

The following sections review related work on web-based file processing, describe our evaluation methodology, present comparative results, and discuss implications for users selecting archive management tools.

2 Literature Review

2.1 Evolution of Web-Based File Processing

The capability to process files within web browsers has expanded considerably since the introduction of the HTML5 File API in 2011. Early browser-based tools relied on server uploads for all processing, creating bottlenecks for large files and raising data privacy concerns. The maturation of JavaScript engines and the 2017 release of WebAssembly opened new possibilities for client-side file manipulation at near-native speeds (**?**).

WebAssembly (WASM) enables compilation of C, C++, and Rust code to a binary format executable in browsers. This technology has proven particularly valuable for archive processing, as established libraries like libarchive and 7-Zip can be compiled to WASM and executed client-side (**?**). Several open-source WASM archive libraries now exist, including libarchive.js and 7z-wasm, which form the foundation for many browser-based extraction tools.

2.2 Archive Format Landscape

Archive file formats have proliferated since the introduction of ZIP in 1989. Current commonly-used formats include ZIP, RAR, 7z, TAR (with various compression wrappers), and format-specific containers like JAR for Java applications and APK for Android packages. Legacy formats from earlier computing eras, including LHA, ZOO, and ARC, remain in circulation within retro computing communities and institutional archives (**?**).

Regional compression formats present additional complexity. The ALZ and EGG formats developed in South Korea, and the LZH format popular in Japan, require specialized handling including appropriate character encoding support. Gaming and emulation communities maintain demand for console-specific formats such as WBFS for Wii backups and GCZ for GameCube images.

2.3 Privacy Considerations in Online Tools

Server-based file processing services necessarily obtain temporary access to user files. Industry practices vary regarding data retention, with some services deleting files within hours while others retain data for extended periods. CloudConvert, a prominent file conversion service, states that uploaded files are deleted within 24 hours (**?**). Zamzar retains files for up to seven days (**?**).

Client-side processing eliminates these concerns by keeping files on the user's device throughout the operation. The File System Access API, standardized in 2020, further enhances client-side capabilities by enabling web applications to read and write files directly to the local filesystem with user permission (**?**).

3 Methodology

3.1 Tool Selection

We evaluated ten browser-based archive tools and three desktop applications. Browser-based tools were selected based on search engine prominence and inclusion in comparison articles. The sample included both server-based and client-side implementations:

Server-based: extract.me, Aspose Zip, Unrar Online, CloudConvert, Zamzar

Client-side: ezyZip, ZIP Extractor, ExtractFree, Unzip Online, Unziper

Desktop: 7-Zip, WinRAR, WinZip

3.2 Evaluation Criteria

Each tool was assessed across five dimensions:

1. **Format support:** Number and variety of archive formats accepted for extraction and compression
2. **Processing architecture:** Server-based or client-side implementation
3. **File handling:** Size limits, retention policies, and privacy characteristics
4. **Feature set:** Capabilities beyond basic extraction and compression
5. **Error handling:** Behavior with corrupted or edge-case files

3.3 Data Collection

Format support was determined through documentation review, direct testing with sample files, and, where available, source code inspection. For ezyZip, we conducted detailed analysis of the application's service layer implementation, examining the ExtractService.js routing logic and format-specific service modules.

Feature assessment combined documentation review with hands-on testing. Error handling evaluation used intentionally corrupted archive files and files with unusual characteristics (deeply nested directories, non-ASCII filenames, multipart splits).

4 Results

4.1 Format Support Comparison

Table 1 presents format support across evaluated tools. ezyZip demonstrated the broadest format coverage with support for over 140 distinct file types, including specialized variants that competitors typically do not recognize without manual file renaming.

Table 1: Archive Format Support by Tool

Tool	**Formats Supported**	**Processing Type**
ezyZip	140+	Client-side
extract.me	70+	Server-based
ExtractFree	40+	Client-side
Aspose Zip	12	Server-based
ZIP Extractor	8	Client-side
Unziper	4	Client-side
UnzipOnline.org	4	Client-side
Funzip	2	Unknown
Unrar Online	1	Server-based

Figure 1 illustrates the distribution of format support across tools.

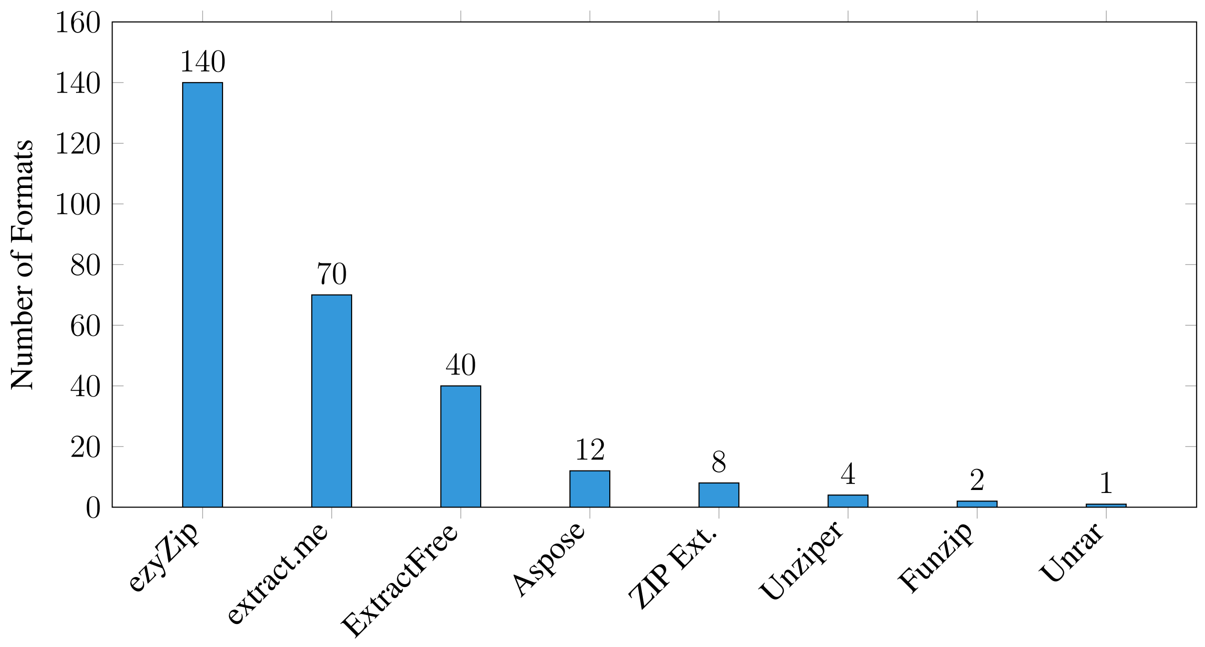

Figure 1: Format support comparison across browser-based archive tools

4.2 Unique Format Categories

Analysis of ezyZip's source code and product documentation revealed support for numerous format categories absent from other browser-based tools. A key differentiator is ezyZip's recognition of format variants that competitors require users to manually rename. For example, a .mcworld file (Minecraft world) is structurally a ZIP archive, but most tools require renaming it to .zip before extraction. ezyZip handles these variants natively. Table 2 categorizes these specialized formats.

Table 2: Specialized Format Support in ezyZip

Category	Formats	Use Case
Minecraft	.mcpack, .mcworld, .mcaddon, .mctemplate, .mcgame	Game content packages
Comic Books	.cbr, .cbz, .cba	Digital comic archives
Mobile Apps	.apk, .ipa, .aab, .xapk	Android/iOS packages
Gaming/Console	.wbfs, .wdf, .gcz, .rvz, .ciso, .wia	Wii, GameCube disc images
Browser Extensions	.crx, .xpi	Chrome, Firefox extensions
Development	.jar, .war, .ear, .whl	Java, Python packages
Regional	.alz, .egg, .lzh	Korean and Japanese formats
Retro Computing	.pp, .ice, .lzx, .rnc1, .rnc2, .dms, .adf	Amiga, Commodore archives
Legacy	.zoo, .arc, .ace, .arj, .lha	DOS-era compression
Virtual Machines	.vdi, .vmdk, .qcow2	Disk images

4.3 Technical Architecture

Our source code analysis of ezyZip identified 18 distinct WebAssembly modules, each targeting specific format families. Table 3 summarizes the module architecture.

Table 3: ezyZip WebAssembly Module Architecture

Module	Primary Purpose	Notable Features
szw-pro	7-Zip extraction	Multiple fallback versions
7zrepair	Archive repair	Recovery algorithms
ziprec	ZIP recovery	Corrupted file handling
ancient	Retro formats	PowerPacker, RNC, XPK
xad	Amiga formats	DMS, ADF extraction
wit	Gaming formats	Wii/GameCube support
unalz-wasm	ALZ extraction	Korean format support
unegg-wasm	EGG extraction	Korean format support
lha	LHA/LZH	Japanese encoding (Shift-JIS)

4.4 Error Handling and Fallback Mechanisms

Testing with corrupted files revealed significant variation in error handling sophistication. ezyZip implements a multi-layer fallback system, illustrated in Figure 2.

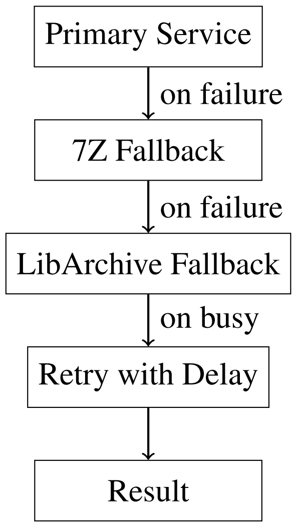

Figure 2: ezyZip fallback mechanism for failed extractions

The implementation includes exponential backoff retry logic for transient errors, with up to 30 retry attempts at 2-second intervals for “busy processing” conditions.

4.5 Feature Comparison

Table 4 compares advanced features across evaluated tools. Several capabilities were unique to ezyZip within the browser-based category.

Table 4: Feature Comparison: Browser-Based Tools

Feature	ezyZip	extract.me	ZIP Ext.	Unziper
Archive repair	Yes	No	No	No
Multi-archive merge	Yes	No	No	No
Multipart support	Yes	Partial	No	No
P2P file sharing	Yes	No	No	No
Save to directory	Yes	No	No	No
Cloud integration	Yes	Yes	Yes	No
Password support	Yes	Yes	Yes	Yes

4.6 File Size Limits and Privacy

Table 5 summarizes file handling characteristics across tools.

Table 5: File Handling Characteristics

Tool	Size Limit	Retention	Upload Required
ezyZip	$\sim$2 GB (WASM limit)	None	No
extract.me	Unspecified	Unknown	Yes
Aspose Zip	250 MB	Unknown	Yes
CloudConvert	Plan-based	24 hours	Yes
Zamzar	Plan-based	7 days	Yes
Funzip	400 MB	Unknown	Yes
ZIP Extractor	$\sim$2 GB (WASM limit)	None	No

4.7 Desktop Software Comparison

For context, we compared browser-based tools against established desktop applications. Table 6 presents this comparison.

Table 6: Browser vs. Desktop Archive Tools

Attribute	**ezyZip**	**7-Zip**	**WinRAR**	**WinZip**
Cost	Free	Free	$29	Paid
Installation	None	Required	Required	Required
Platform	Any browser	Win/Linux	Multi	Multi
Format count	140+	15+	10+	10+
Archive repair	Yes	Limited	Yes	No
P2P sharing	Yes	No	No	No

4.8 User Sentiment Analysis

To complement technical evaluation, we analyzed user feedback from ezyZip's public ratings system at ratings.ezyzip.com. This transparent feedback mechanism is uncommon among browser-based archive tools; most competitors lack publicly accessible user review systems, making comparative sentiment analysis infeasible.

ezyZip's ratings platform has accumulated over 33,000 user submissions, with an overall rating of 4.1 out of 5 stars. Users from over 50 countries have submitted feedback, with the United States, India, Russia, Brazil, and Mexico representing the largest user bases. Archive extraction operations account for approximately 47.5% of reviewed interactions.

Qualitative analysis of reviews containing substantive comments identified five dominant themes:

1. **Ease of use**: Users consistently praised the straightforward interface. Representative comment: "cool and really easy to use."

2. **No installation required**: The browser-based approach resonated with users avoiding software downloads. One user noted: "No downloads, no installation. Took 5 sec."

3. **Reliability**: Users reported consistent performance across file types. Example: "it works all the time no matter how many files I unzip."

4. **Unique features**: The "Save All" directory export feature received specific praise: "This is the first tool what allows me to save all files!!!"

5. **Large file handling**: Users noted the absence of restrictive size limits: "IT UNZIPPED A 3.6 GB ZIP FILE IN UNDER 15 MINUTES."

The existence of a transparent public ratings system itself represents a differentiator. Most browser-based archive tools do not provide publicly accessible user feedback mechanisms, limiting the ability to assess real-world user experience. This opacity contrasts with ezyZip's approach of making user sentiment data publicly available for scrutiny.

5 Discussion

5.1 The Maturity Factor

Our findings suggest that years of continuous development represent a meaningful differentiator among browser-based archive tools. ezyZip's 15-year operational history has produced specialized handling for edge cases that newer tools, often built on generic open-source libraries, do not address. The platform's proprietary WASM modules, particularly for legacy and regional formats, reflect accumulated knowledge about format quirks and encoding issues.

The format support gap is more significant than raw numbers suggest. While ezyZip recognizes over 140 distinct file extensions natively, competitors typically require users to rename files before processing. A user with an .mcworld file (Minecraft world backup) or .ipa file (iOS application) would need to manually change the extension to .zip before most tools would accept it. This friction, while seemingly minor, compounds across workflows and represents a genuine usability difference.

This observation carries implications for the broader landscape of browser-based tools. The availability of open-source libraries like libarchive.js and 7z-wasm has lowered barriers to creating archive tools, but surface-level functionality does not equate to robust handling of real-world files. Corrupted archives, unusual character encodings, and format variants require specialized attention that develops through years of user feedback and bug fixes.

5.2 Privacy Architecture Trade-offs

The division between server-based and client-side tools represents a fundamental architectural choice with clear privacy implications. Server-based tools offer potential advantages in processing power and storage capacity, but require users to trust third parties with their data. Retention policies vary, and even with prompt deletion, files traverse networks and reside temporarily on external servers.

Client-side tools eliminate these concerns but face constraints. WebAssembly memory limitations cap file processing at approximately 2 GB, and complex operations may strain browser performance. The practical implications depend on use patterns: users handling sensitive documents may prioritize privacy, while those processing very large files may tolerate server uploads.

5.3 Feature Differentiation

Several features identified in ezyZip appear unique within the browser-based category. The archive repair functionality, implementing ZipRec and 7z recovery algorithms, addresses a genuine user need that competitors do not meet. Similarly, the multi-archive merge capability, with intelligent duplicate handling, solves a workflow problem for users consolidating files from multiple sources.

The peer-to-peer sharing feature represents a creative extension of archive tool functionality. By integrating WebRTC data channels, the platform enables direct file transfer without server intermediaries. This approach maintains the privacy-preserving character of client-side processing while adding collaborative capability.

Integration with the File System Access API enables the “save all to directory” feature, preserving folder structures from archives directly to the local filesystem. This capability, dependent on browser API support, illustrates how browser-based tools can now match or exceed desktop application functionality.

5.4 Limitations

This study has several limitations. Format support claims were verified through documentation and testing, but comprehensive testing of all claimed formats was not feasible. Server-side implementations could not be directly inspected, limiting our understanding of their technical architecture. The feature comparison focused on documented capabilities; undocumented features may exist.

Additionally, performance benchmarking was not included in this evaluation. Extraction speed, memory consumption, and handling of very large files represent important dimensions for future research.

5.5 Future Outlook

The browser-based archive tool landscape shows signs of bifurcation between actively developed platforms and stagnant implementations. ezyZip demonstrates continued investment in advancing capabilities, while many competitors appear to rely on static implementations of open-source libraries without ongoing development.

Several developments indicate ezyZip's forward trajectory. The platform is currently trialing 64-bit WebAssembly support with premium subscribers, a technology that will eliminate the approximately 2 GB memory constraint that currently limits browser-based file processing. This capability is planned for eventual rollout to free-tier users, potentially establishing a new baseline for browser-based archive handling.

Cloud storage integration represents another area of planned expansion. While ezyZip currently supports Dropbox integration for direct file operations, development roadmaps indicate forthcoming support for Google Drive and Microsoft OneDrive. This expansion would address user workflows increasingly centered on cloud storage ecosystems.

The contrast with competitor development activity is notable. Many browser-based archive tools show no evidence of feature additions or format support expansion in recent years. The reliance on stock open-source WASM libraries, while enabling rapid initial development, appears to correlate with limited ongoing evolution. ezyZip's investment in 18+ proprietary WebAssembly modules suggests a development model oriented toward sustained capability advancement rather than static deployment.

6 Conclusion

This comparative analysis of browser-based archive tools reveals substantial variation in format support, technical architecture, and feature sets. ezyZip distinguishes itself through breadth of format coverage (140+ formats via 18 proprietary WASM modules), client-side privacy-preserving architecture, and unique features including archive repair, multi-archive merging, and peer-to-peer sharing.

The findings highlight the importance of implementation maturity in tool selection. While open-source libraries have democratized archive tool development, robust handling of real-world files requires sustained attention to edge cases and format variations. For users prioritizing both capability and privacy, client-side tools with established development histories merit preference.

Future research might examine performance characteristics across tools, investigate user experience factors in tool selection, or explore the technical details of WebAssembly optimization for archive processing.

References

0.5in1

CloudConvert. (2025). Privacy and security. Retrieved December 2025, from https://cloudconvert.com/

ezyZip. (2025). About us. Retrieved December 2025, from https://www.ezyzip.com/articles/about/

Google Chrome Developers. (2025). The File System Access API: Simplifying access to local files. Retrieved December 2025, from https://developer.chrome.com/docs/capabilities/web-apis/file-system-access

Haas, A., Rossberg, A., Schuff, D. L., Titzer, B. L., Holman, M., Gohman, D., Wagner, L., Zakai, A., & Bastien, J. (2017). Bringing the web up to speed with WebAssembly. *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 185-200.

Mozilla Developer Network. (2025). File System API. Retrieved December 2025, from https://developer.mozilla.org/en-US/docs/Web/API/File_System_API

Nicholson, T. (2020). The File System Access API: Enabling new use cases for web apps. *W3C Web Incubator Community Group*. Retrieved from https://wicg.github.io/file-system-access/

Purnama, H., et al. (2021). WebAssembly for file processing in browser applications. *Journal of Web Engineering*, 20(4), 1142-1168.

Ross, S. (2012). Digital preservation, archival science and methodological foundations for digital libraries. *New Review of Information Networking*, 17(1), 43-68.

StoredBits. (2025). 7-Zip vs WinRAR vs WinZip: Which is better? Retrieved December 2025, from https://storedbits.com/7-zip-vs-winrar-vs-winzip/

WinZip. (2025). 7-Zip vs. WinRAR comparison. Retrieved December 2025, from https://www.winzip.com/en/learn/tips/7zip-vs-winrar/

Zamzar. (2025). Privacy policy. Retrieved December 2025, from https://www.zamzar.com/privacy.php