ChatGPT

Converting Selected Web Content to Markdown – Tools & Methods

Converting a highlighted section of a webpage into clean Markdown can be achieved through a variety of tools. Below is a comprehensive look at the best solutions, organized by category, with details on how they work and their pros/cons. We focus on browser-based tools that let you **visually select** content and convert it to well-structured Markdown, preserving elements like code blocks, formatting, links, and images.

Summary of Solutions

Tool / Method	Type	Key Features
MarkDownload	Browser Extension (Chrome, Firefox, Edge, Safari)	Clip entire page or selected content to Markdown; shows a preview for editing or copying; can download as `.md` file [1]. Preserves headings, formatting, links, lists, and images; Obsidian integration available.
Copy as Markdown	Browser Extension (Chrome, Firefox, Edge)	Copies selected text as Markdown to clipboard, with support for **links, images, italic/bold/strike**, lists, tables, and fenced code blocks (with language hints) [2]. Lightweight and open-source.
Markdownizr	Browser Extension (Chrome)	Copies selected content (or whole page) as **clean Markdown**. Allows custom rules to strip or remove specific HTML elements [3] [4] for cleaner output. Open-source extension with context-menu and toolbar button.
Copycat	Browser Extension (Chrome)	Advanced right-click menu for copying content in various formats. Can copy a selection as Markdown (or plain text, HTML, Org-mode, etc.) [5]. Highly configurable with keyboard shortcuts; open-source.
Joplin Web Clipper	Browser Extension (Chrome, Firefox)	Saves selected content (or full pages) directly into Joplin notes as Markdown [6]. Preserves formatting, code, and images in a note-taking workflow. Requires Joplin app; also offers a “clip selection” option.
Turndown Bookmarklet	Bookmarklet (Any Browser)	A simple **bookmarklet** that converts the currently highlighted webpage portion to Markdown using the Turndown JS library [7]. Copies the Markdown to clipboard instantly, no extension installation needed.

Tool / Method	Type	Key Features
LLM (ReaderLM via Ollama)	AI Model / Workflow	Utilizes a local Large Language Model (e.g. Jina AI's **ReaderLM-v2** via Ollama) to parse HTML and output Markdown. ReaderLM is optimized for high-quality Markdown conversion – it excels at preserving **code fences, nested lists, tables** and other complex elements [8]. Can be integrated into custom scripts or workflows for improved handling of messy HTML.
Web2Markdown (Answer.dev)	Browser Extension + Cloud API	An open-source Chrome extension that sends page content to a web API for conversion. For example, the *web2md* extension copies the page and uses an online service (web2md.answer.ai) to return Markdown [9] [10]. Content is then copied to your clipboard. (Requires internet connection.)

Below we dive into each solution, including usage instructions, and discuss their advantages and drawbacks.

Browser Extensions for Markdown Clipping

Several browser add-ons make it one-click simple to turn a webpage selection into Markdown. These extensions run entirely in the browser (often offline) and preserve most formatting.

MarkDownload – Markdown Web Clipper

Link: Available on Chrome Web Store (also Firefox Add-ons, Edge, Safari); source on GitHub.
Overview: MarkDownload is a popular open-source extension that can **clip an entire page or just the highlighted portion** into a Markdown file [1]. When you click the extension button, it opens a popup showing the generated Markdown, which you can copy to clipboard or download as an `.md` file [1]. If you have text selected on the page, it will capture *only* that selection [1].

Features: MarkDownload preserves the structure and formatting of content well. Headers, **bold/italic text**, links, lists, and blockquotes are all retained as Markdown syntax. It also captures **code blocks** (using triple backticks) and can handle images by inserting `![](URL)` links. By default, images are referenced by their web URL, but the extension has options to download images alongside the markdown (e.g. in a zip or to an Obsidian vault). MarkDownload even supports direct integration with Obsidian – you can configure it to send the clipped Markdown straight to your Obsidian vault for note-taking convenience.

Pros:
- **Visual selection support:** Just highlight content and click; no need to manually copy HTML.
- **Live preview/edit:** You get a rendered Markdown preview in the popup, allowing quick fixes before saving [11].
- **Cross-browser and open-source:** Works on Chrome, Firefox, Edge, Safari; code is on GitHub.
- **Obsidian integration:** Can streamline clipping web content into Obsidian MD notes (via a URI handler).
- **Preserves formatting** of most elements (including lists and inline styles) accurately.

Cons:
- **Dynamic or complex sites:** Not guaranteed to work on every website – pages with heavy dynamic content or script-generated sections might clip poorly [12]. The developer notes it "is not guaranteed to work on all websites" [12].
- **Images handling:** By default it links images (which is usually fine), but if you want local copies, you must use the download option or Obsidian integration (which may require additional setup). Some users report multiple download dialogs if a page has many images (one per image file).
- **Minor formatting quirks:** In some cases, extra line breaks or spacing might need small manual cleanup, though the output is generally clean.

Copy as Markdown (by Laxman / notlmn)

Link: Available on Chrome Web Store and Firefox Add-ons; source on GitHub (`notlmn/copy-as-markdown`).
Overview: *Copy as Markdown* is a lightweight extension focused on copying the **selected text** on a page as Markdown (to your clipboard). It supports an array of rich content: **hyperlinks, images, italic/bold/strikethrough text, inline code, lists (bulleted/numbered), checklists, tables, and fenced code blocks** [2]. You can initiate it via a right-click context menu or keyboard shortcut. Rather than saving to a file, it's meant to quickly copy Markdown for use in your documents or notes.

Features: When you select text and activate the extension, it converts the HTML of that selection to Markdown behind the scenes. For example, images become `![](image_url)` syntax, links become `[anchor text](href)`, tables are formatted in GitHub-Flavored Markdown, and `<pre><code>` blocks turn into ``` fences (with language tags if detectable) [2]. It specifically tries to detect the programming language of code snippets and adds the info string after the triple backticks for syntax highlighting [13].

One nice detail is that if you use the extension on a page without selecting anything, some versions will default to copying the page title and URL in Markdown (i.e., `[Title](URL)`) – handy for quickly grabbing a reference link.

Pros:
- **Rich content support:** Handles nearly all common formatting. As one reviewer noted, it covers *italic, bold, strikethrough, inline/code, code blocks, lists, headings, links, images,* and even tables [14]. This means minimal manual fixing after pasting.
- **Quick clipboard copy:** Directly puts Markdown into your clipboard for easy pasting. Great for gathering notes or documentation.
- **Multi-browser & open-source:** Supports Chrome, Firefox, and Edge. The extension is open-source and has an active maintainer; last updated in 2024.
- **Privacy-friendly:** Runs offline entirely; no data collection. (On Chrome it requires minimal permissions like activeTab and contextMenus.)
- **Lightweight:** Only ~30KB in size [15], so it doesn't bloat your browser.

Cons:
- **Limitations on Chrome for alt-text:** Due to Chrome API limitations, it cannot retrieve the alt text of images or the actual text of a hyperlink if it's not in the selection. In those cases it falls back to using the raw URL as the link text or image alt [16]. (Firefox's version doesn't have this issue and can capture alt/title text normally.)

- **Tables edge cases:** The developer notes table support is included, but some users have found complex tables don't always convert perfectly [17] (perhaps needing some touch-up).
- **No file saving or preview:** This extension is purely "copy to clipboard." It doesn't show you the Markdown before copying, and if you want a file, you'll need to paste it into an editor and save manually (unlike MarkDownload which has a preview/download UI).

Markdownizr

Link: Chrome Web Store (also see markdownizr.com for info and source).
Overview: Markdownizr is an extension dedicated to extracting **clean Markdown** from web content. You can either highlight a portion of the page or leave nothing selected (to grab the entire page content), then click the "M" button. The selected content is converted to Markdown and **automatically copied** to your clipboard [18] (a notification or modal can confirm the copy). It also provides a context menu option "Get Markdown" for convenience [18].

Features: Markdownizr emphasizes producing well-formed Markdown without extraneous clutter. It achieves this through customizable **filters**. In the extension's options, you can specify certain HTML tags to *strip* (meaning keep their inner text but remove the tag itself) and others to *delete entirely* (remove the tag and its content) [4]. For instance, you might strip out `<div>` wrappers or remove `<script>` and `<style>` sections that you never want in the output. This level of control helps in cleaning up those bits of HTML that a generic converter might include. The extension comes with sensible defaults, but you can tweak as needed [3] [4].

Like others, Markdownizr uses the Turndown library under the hood. It preserves formatting similar to the above tools (links, basic styling, lists, code blocks, images, etc.), but the real appeal is the *flexibility* in tuning the output to your liking.

Pros:
- **Visual selection or full page:** Works with either a highlight or the whole page if nothing is selected [19]. Good for both focused clipping and full-page saving.
- **Customizable output:** The ability to define HTML tags to strip or remove is powerful for getting "just the content" you want [4]. For example, you could remove nav bars or sidebars that might otherwise be included.
- **Clipboard-ready:** It copies the Markdown for you immediately [18] – no extra click to copy after conversion.
- **Open-source and secure:** The site touts that it's free, open-source, and contains no trackers or malware [20].
- **Future-proofing:** The developers highlight that unlike some extensions that are hard-coded, Markdownizr lets you adapt to different sites via settings [3]. This means if a certain website's HTML is messy, you can add its unwanted tags to your filters.

Cons:
- **Chrome only:** As of now, it's primarily a Chrome/Chromium extension. There isn't an official Firefox version (though the source could potentially be adapted).
- **No in-browser preview:** It immediately copies to clipboard, so you don't see the formatted Markdown unless you paste it somewhere. (However, this keeps the workflow quick.)
- **Learning curve for filters:** To take full advantage of the custom filters, users might need to experiment

with the settings. The default works well in many cases, but adjusting it requires some understanding of the source HTML.
- **Turndown limitations:** It relies on the Turndown library, so any conversion quirks of Turndown (e.g., how it handles nested lists or certain HTML) will apply. Advanced content like some HTML tables or embedded media might not convert perfectly without tweaks.

Copycat – Multi-Format Content Copier

Link: Chrome Web Store (open-source on GitHub under *BlackGlory/Copycat*).
Overview: Copycat isn't exclusively a Markdown tool – it's more of an "ultimate copier" extension that can copy browser content in many formats (Plain text, HTML, JSON, etc.). Notably, it supports copying **selections, links, images, or entire tabs** as Markdown among its options [5]. After installing, it adds a **right-click context menu** with various submenus (and you can configure which options appear). For example, you can right-click on a page and see options to copy the page title or URL in different formats, or select text, right-click and choose "Copy selection as → Markdown". There are also keyboard shortcuts available for power users.

Features: The strength of Copycat is its versatility. Relevant to our topic, if you highlight part of a page, **Copycat can copy that selection as Markdown** [21]. It will do a conversion similar to the above tools, retaining links, formatting, etc. If you right-click an image, you can copy it as a Markdown image code (`![](url)`), or right-click a hyperlink and copy as Markdown link, etc. It essentially combines the functionality of multiple single-purpose extensions into one. You can also configure custom template rules (for example, "HTML without attributes" or user-defined cleaning of HTML before copying [22]). The extension's options allow enabling/disabling certain menu items and binding commands to hotkeys.

Pros:
- **All-in-one tool:** Instead of having separate extensions for copying links, images, or selections in Markdown, Copycat handles them all in one package [5]. This can streamline your browser toolbar.
- **Highly configurable:** You can choose which context menu items you want, set keyboard shortcuts for specific copy actions, and even define custom copy formats. For tech users, this is extremely flexible.
- **Supports multiple markup formats:** In addition to Markdown, it can do Org-mode, AsciiDoc, BBCode, etc. [23]. If you ever need those, it's a bonus.
- **Open source:** Source code is available, and it has active development. The extension is a bit larger (1.25 MiB) [24] due to its breadth of features, but still respects privacy (no data collection [25]).
- **Continual updates:** As of mid-2024 it was updated, indicating good maintenance [26].

Cons:
- **Chrome only (currently):** There isn't an official Firefox version listed. Firefox users have other alternatives for Markdown copying, but they'd miss out on Copycat's combined features unless a port is made.
- **Complexity:** With many options, the context menu can become quite long or confusing until you customize it. There may be a slight learning curve to configure it to your liking, compared to simpler single-purpose extensions.
- **No GUI preview:** Like most clipboard-focused tools, you don't get to see the Markdown formatted before it's copied. You have to trust the conversion (or paste it out to verify).
- **Potential overlap:** If you only need Markdown selection clipping, Copycat might be overkill. It shines best if you appreciate its multi-format copy features beyond Markdown.

Joplin Web Clipper

Link: Chrome Web Store / Firefox Add-ons (comes with Joplin installation instructions).
Overview: Joplin is an open-source note-taking app that uses Markdown for its notes. Its Web Clipper extension allows you to **send content from your browser into Joplin directly**. Among the clipping options are: *Clip simplified page*, *Clip complete page (Markdown)*, *Clip complete page (HTML)*, *Clip selection*, *Clip screenshot*, or *Clip URL* [6]. For our purposes, using *Clip selection* or *Clip complete page (Markdown)* will convert the content to Markdown and create a new note in Joplin with that content.

Features: When clipping to Joplin in Markdown mode, the extension uses Joplin's HTML-to-Markdown converter (which is quite robust, since it had to handle Evernote HTML in the past). It preserves formatting, images, and attachments by downloading them into Joplin's resource folder. **Code blocks, headings, lists, tables** etc., are generally well-maintained in the resulting note. The Web Clipper panel that appears allows you to choose the target notebook in Joplin and give the note a title before you hit save [27] [28]. Because content is saved into your Joplin app, it's available for search, offline access, and editing in Joplin immediately.

Additionally, in recent versions Joplin's clipper has an option to copy content to clipboard as Markdown (with the source URL appended) instead of creating a note [29] – effectively giving you a direct copy-paste capability similar to other extensions, but integrated with Joplin for attribution.

Pros:
- **Excellent output quality:** Joplin's conversion engine is known to handle messy HTML well (it was designed to import from web/Evernote), so the Markdown you get is usually clean and well-structured.
- **Embedded images and files:** When clipping to the app, it actually saves images into the note, so you have local copies (no broken image links later).
- **Selection or full page:** You have the choice to grab just a selection or the whole page. If you pick "simplified page", it will use a readability mode to strip ads/extraneous parts, then convert to Markdown – useful for article text.
- **Secure and offline:** All data stays in your Joplin, which can be end-to-end encrypted. You don't rely on third-party servers.
- **Free and open-source:** Both Joplin and its clipper are FOSS.

Cons:
- **Requires Joplin app running:** The extension communicates with a Joplin instance on your computer. You must have Joplin open (and the clipper service enabled) to use it [30] [31]. This ties the solution to those who use Joplin for note management.
- **Not a general clipboard tool by default:** Its primary use case is saving into Joplin's database, not outputting a markdown file or clipboard text (though the clipboard option exists in config, it's a bit hidden). If you don't use Joplin, this is probably overkill just for Markdown conversion.
- **Limited to notes:** The clipped Markdown is stored in Joplin; if you want to use it elsewhere, you'd have to open the note and copy the text out. This extra step means it's less direct for, say, quickly pasting into a document.
- **Minor formatting adjustments:** As with any converter, you might occasionally want to tweak the result in the note (e.g., Joplin might turn some complex web layouts into a series of images or nested lists that you later refine). But generally, it's solid.

Simple Scripts & Bookmarklets

If you prefer not to install a full extension, or want a more customizable approach, scripts and bookmarklets can do the job of converting selected HTML to Markdown.

Turndown-based Bookmarklet

One elegant solution is using a **JavaScript bookmarklet** that injects a converter library on the fly. A popular choice is a snippet that loads the Turndown library (HTML-to-Markdown converter) and processes the selected HTML. For example, the following bookmarklet (by user *chrillek* on a forum) does exactly that:

> **Bookmarklet code:** *(JavaScript snippet that loads turndown.js and converts selection)*

This bookmarklet will take **whatever text you've highlighted** on the page, convert it to Markdown using Turndown, and then copy the Markdown text to your clipboard [7]. If nothing is selected, it won't do anything (or could be modified to grab the whole page as a fallback). You add it by creating a new bookmark in your browser with the `javascript:(()=>{ ... })()` code as the URL.

Pros:
- **No extension needed:** Works in any modern browser (Chrome, Firefox, Safari, etc.) by virtue of being a bookmark. It's also easy to share or tweak since it's just code.
- **Uses Turndown:** Turndown is a well-tested library, so the output quality is similar to what extensions like MarkDownload/Markdownizr would give (they also use Turndown or similar). It preserves links, formatting, lists, etc., properly.
- **Clipboard automation:** The script programmatically copies the result to the clipboard [32], so you don't have to manually select and copy the output – it's instantaneous.
- **Selective or full page:** The bookmarklet can be adapted – e.g., one version includes a "smart fallback" that if no text is highlighted, it tries to find the main article content on the page automatically [33]. This gives flexibility similar to extensions.

Cons:
- **Injects external script:** The typical implementation fetches the Turndown JS from a CDN (like unpkg.com) each time [34]. This means a slight delay on first use and reliance on that CDN. Also, some very strict sites might block such script injection. (One can mitigate by self-hosting the script or using an offline version with a userscript manager.)
- **Basic error handling:** If something doesn't convert well (or nothing is selected), you may not get feedback. It's not as user-friendly as extensions with UI.
- **No GUI or settings:** Any customization (like skipping certain elements) would require editing the JS code. It's not difficult (Turndown can be extended with rules), but less convenient than an extension's options panel.
- **Site restrictions:** A few websites (especially big ones like GitHub) may block bookmarklet scripts for security. For those pages, this method won't work [35]. But for the vast majority of sites it should work fine [7].

Usage: To use this method, simply create the bookmarklet and click it whenever you want to copy Markdown. For instance, one can use the snippet from the DEVONtech forum which was:

```
javascript:(() => {
  const script = document.createElement("script");
  script.src = "https://unpkg.com/turndown/dist/turndown.js";
  script.onload = () => {
    const selection = window.getSelection().getRangeAt(0);
    const div = document.createElement('div');
    div.appendChild(selection.cloneContents());
    const turndownService = new TurndownService();
    const markdown = turndownService.turndown(div.innerHTML);
    navigator.clipboard.writeText(markdown);
  };
  document.head.appendChild(script);
})();
```

(You can find variations of this code online or in the Turndown docs. Always be cautious when copying code from the web – ensure you understand it, for security.)

Custom Scripts or Bookmarklets with Readability

Another script-based approach is combining Mozilla's Readability library to isolate main content, then converting to Markdown. This goes beyond the question's scope slightly, but worth noting: you could have a bookmarklet that first grabs `document.article` content (like Mercury/Readability does) and then feeds that to Turndown. This would be similar to a "simplified page to Markdown" function.

Pros: Cleaner output for cluttered news sites. **Cons:** More complex script and potential to miss content outside the main article.

(There are no off-the-shelf bookmarklets doing exactly this with Markdown that we know of, but it's a doable DIY for advanced users.)

AI-Powered Solutions (LLM Integration)

For especially complex or poorly structured HTML content, an AI (LLM) approach might yield superior results in Markdown conversion. Large Language Models can infer structure where HTML is broken, and ensure the Markdown is semantically meaningful. Here we consider using local LLMs (to avoid sending data to external APIs) like **Ollama**-hosted models.

Jina AI's ReaderLM (via Ollama or API)

One of the state-of-the-art efforts in this space is **ReaderLM-v2** by Jina AI – a 1.5B parameter model specifically tuned for HTML-to-Markdown (and HTML-to-JSON) conversion [36]. This model was trained on lots of examples to learn how to output well-formatted Markdown from raw web HTML. According to Jina, *ReaderLM-v2 produces high-quality Markdown, excelling at complex elements such as code blocks, nested lists, tables, and even LaTeX equations* [8]. In other words, it's designed to preserve the kinds of rich content elements the user is interested in.

How to use locally: The ReaderLM model can be run locally using the **Ollama** system or other MLC frameworks. For instance, Jina provides an Ollama-compatible model checkpoint. You would run an Ollama server (which allows local API calls to the model) with `ollama run readerlm-v2` (after importing the model). Then, a script or extension could send the selected HTML to this local API and receive the Markdown response.

There are also open-source projects that showcase how to integrate this: - The `docs-to-md` tool by ackness uses ReaderLM via Ollama as a backend to convert documentation pages to Markdown [37] [38]. - Jina's own **Reader API** (cloud service) can be tried out to convert a webpage by URL to Markdown using ReaderLM (with a simple curl command) [39].

In our context, one could imagine a workflow: you highlight content, click a "Convert with AI" button (perhaps a custom extension or a manual step), which copies the HTML and sends it to a local LLM. The LLM then returns structured Markdown that might be cleaner than rule-based methods for certain cases (like content with inconsistent tags, or requiring minor reformatting).

Pros:
- **Intelligent parsing:** The LLM can handle imperfect HTML. It might close unclosed tags, infer list structures even if the HTML was just paragraphs, or rearrange content logically. This can result in more **human-like, clean Markdown** output when the source is not straightforward.
- **Preserves complex structures:** ReaderLM in particular is designed to output things like tables and fenced code blocks correctly [8], where simpler converters might falter or output as plain text.
- **Local processing:** Using something like Ollama means everything stays on your machine (no external API calls), preserving privacy while harnessing AI.
- **Customizable via prompts:** You could instruct a general LLM (like GPT-4 via API, or a local LLaMA model) with a prompt like: *"Convert the following HTML to Markdown, preserving all headings, links, lists, and code blocks:"* followed by the HTML snippet. A good model will follow the instruction and produce Markdown. This opens possibilities to tweak style (for example, you could ask it to enforce a certain line wrap or add footnote links for references, etc.).

Cons:
- **Resource and speed:** Running a 1.5B model or larger locally requires a decent machine (ReaderLM-v2 can handle long inputs but is optimized for GPU use [40]). Converting a lengthy page via LLM might be slower (several seconds) compared to instant output from a JavaScript library.
- **Possible errors or hallucinations:** If the model is not perfectly aligned, it might *hallucinate* or mistakenly alter content. For instance, an AI might rephrase text or fill in missing pieces that weren't actually there – which could be undesirable if you need exact fidelity. However, specialized models like ReaderLM aim to minimize this and stick to converting format, not content. (Jina addressed stability to reduce off-track generation [41].)
- **Setup complexity:** There's no off-the-shelf browser extension that plugs into Ollama (as of writing). You'd need to do some custom wiring – perhaps using a userscript or a small local web server that the browser can query. This is more involved than installing a typical extension.
- **LLM size limits:** Very large pages might hit token limits, though ReaderLM-v2 supports up to 512k tokens context [8] which is huge. More commonly, using GPT-4 or others might have smaller limits (8k-32k tokens), meaning extremely long HTML selections could be problematic.

When to use: An AI-based solution is likely overkill for well-structured pages (the extensions above handle those fine). But if you frequently clip content that ends up needing heavy manual cleaning (maybe pages with inconsistent HTML, or converting parts of pages that include dynamic content), an LLM might save time by outputting a cleaner version. It's also an exciting area if you want to integrate with other AI workflows (like summarization after clipping, etc.).

Example: You could run a local instance of ReaderLM-v2 and use a small script to feed it a page's HTML: it will return a Markdown string. Early versions of ReaderLM noted that some implementations actually used a combination of regex and Turndown internally [42], but v2 is more end-to-end learned. Users have reported high accuracy of the Markdown output, though one must watch out for minor AI artifacts.

In summary, **LLM integration** is an optional layer for power users who demand the *absolute best* Markdown from chaotic sources, and are willing to trade a bit of complexity and compute for it.

Other Noteworthy Mentions

- **Pandoc (CLI tool):** Pandoc is a universal document converter that can turn HTML into Markdown as well. While not a browser tool, one could copy the page HTML (e.g., using the browser's "Save as HTML" or developer tools to grab a portion) and then run `pandoc -f html -t markdown` on it. Pandoc's Markdown output is quite good and it handles images, links, etc. This is more of a manual or batch process, but for completeness, pandoc is a reliable option. Pros: very accurate, highly customizable via flags (e.g., `--wrap=none` to control line wrapping, `--extract-media` to download images). Cons: not interactive in-browser, and doesn't understand what part you highlighted unless you feed it that exact snippet. It's best for full-page conversions or scripts.

- **DEVONthink/DT Clipper (Mac):** The DEVONthink app on Mac has a web clipper that can capture selection as Markdown. If you are in that ecosystem, it's similar to Joplin's idea: the bookmarklet we mentioned was actually an inspiration for some DT users [7]. This is more niche, but if one uses DEVONthink, you might already have this capability.

- **Obsidian Official Web Clipper:** Recently, the Obsidian community introduced an official web clipper (as a separate app called *Obsidian Importer* for browsers [43]). It can clip pages into Obsidian in Markdown. It's similar in concept to MarkDownload (which was community-made). If you use Obsidian and prefer an official tool, keep an eye on that. At the moment, MarkDownload remains a popular choice due to its flexibility.

- **Mobile Solutions:** If you need this on mobile browsers, extensions may not be available. But bookmarklets (as described above) do work on mobile Safari or Firefox (with some fiddling). Additionally, mobile apps like iA Writer or Drafts offer share extensions to convert HTML to Markdown when sharing a webpage – useful on iOS/Android, though outside desktop browser scope.

Comparison Table of Key Features

To wrap up, here's a quick comparison of the key solutions:

Solution	Selection-Based?	Preserves Code Blocks & Tables	Images Handling	Requires Internet?	Open Source?
MarkDownload	Yes (partial or full page) [1]	Yes (code fenced, lists, etc.)	References images by URL (option to download)	No (offline)	Yes
Copy as Markdown	Yes (selection only) [2]	Yes (incl. fenced code, *tables*)	Inlines image URLs; no auto-download	No (offline)	Yes
Markdownizr	Yes (selection or full) [19]	Yes (uses Turndown standard)	Inlines image URLs; user can filter out elements	No (offline)	Yes
Copycat	Yes (selection, link, image, or tab) [5]	Yes (via conversion for selection)	Inlines image URLs (or copy image data)	No (offline)	Yes
Joplin Web Clipper	Yes (selection or full) [6]	Yes (converts with high fidelity)	Downloads images into note storage	No (local app)	Yes
Turndown Bookmarklet	Yes (selection; whole page fallback) [7]	Yes (via Turndown library)	Inlines image URLs by default	No (offline once cached)	Yes (script)
LLM (ReaderLM via Ollama)	*N/A (needs custom integration)*	**Yes (specialized for code, tables)** [8]	Can output image links or even base64 if prompted	No (runs locally)	Partially (model weights available)
Web2MD (Answer.dev)	Full page by default (selection not specified)	Yes (likely uses Turndown or AI)	Inlines image URLs	**Yes (cloud API)** [10]	Client OSS (server not sure)

(Note: "Preserves tables" means it attempts to format HTML tables into Markdown table syntax. Not all tools handle complex tables perfectly. AI solutions and Copy as Markdown explicitly attempt it, others rely on Turndown's basic table support or may skip tables.)

Conclusion

For most users, a browser extension like **MarkDownload** or **Copy as Markdown** will provide the quickest way to highlight content and grab Markdown, preserving the essentials of formatting, links, and code. These tools work offline and integrate directly into your browsing workflow. If you have specific needs —

such as filtering out junk — **Markdownizr** offers more control, while **Copycat** offers a Swiss-army-knife approach for various formats including Markdown.

Power users interested in the cutting edge can explore **AI-based conversion** via local LLMs like ReaderLM to handle those edge cases where traditional parsers struggle. This adds complexity but can yield impressively clean results on hard-to-parse pages [8].

In summary, the ecosystem for visually selecting web content and converting to Markdown is rich:

- **Quick Clipboard Copy:** *Copy as Markdown* or *Copycat* (fast, convenient).
- **Full-Page and File Save:** *MarkDownload* (with preview and file export, plus selection support).
- **Customized Cleanup:** *Markdownizr* or custom bookmarklets (tweak output to your liking).
- **Note-taking Integration:** *Joplin Web Clipper* (direct to notes with offline storage).
- **AI Enhancement:** *ReaderLM/LLM workflows* (for the adventurous, when you need that extra level of parsing intelligence).

Each approach has its pros and cons, but all of the above will help you avoid the tedium of manual copy-paste and formatting. You can now highlight what you need on a webpage, click a button, and get well-structured Markdown ready for your documentation, notes, or static site generator.

Sources: The information and quotes above were drawn from official documentation and user reports of the tools. For example, MarkDownload's usage and selection feature is documented on its GitHub/Obsidian forum post [1], Copy as Markdown's capabilities are described on its Chrome Web Store listing [2] and a user blog [14], and Markdownizr's website details its filtering options [3] [4]. The Turndown bookmarklet code and explanation were provided on a DEVONtechnologies forum [7]. For AI, Jina AI's announcement highlights ReaderLM's Markdown prowess [8]. These and other references are cited throughout the text for further exploration.

---

[1] [11] [12] MarkDownload - Markdown Web Clipper - Share & showcase - Obsidian Forum
https://forum.obsidian.md/t/markdownload-markdown-web-clipper/173

[2] [13] [15] [16] Copy as Markdown - Chrome Web Store
https://chromewebstore.google.com/detail/copy-as-markdown/nlaionblcaejecbkcillglodmmfhjhfi?hl=en-US

[3] [4] [18] [19] [20] Markdownizr :: Chrome Extension
https://markdownizr.com/

[5] [21] [22] [23] [24] [25] [26] Copycat - Chrome Web Store
https://chromewebstore.google.com/detail/copycat/jdjbiojkklnaeoanimopafmnmhldejbg?hl=en

[6] [27] [28] [30] [31] How Joplin Web Clipper can organize your web content and keep your data secure | We Love Open Source - All Things Open
https://allthingsopen.org/articles/how-joplin-web-clipper-organize-web-content

[7] [32] [34] [35] Bookmarklet to convert HTML selection to Markdown - Tips - DEVONtechnologies Community
https://discourse.devontechnologies.com/t/bookmarklet-to-convert-html-selection-to-markdown/62170

[8] [39] [40] [41] jinaai/ReaderLM-v2 · Hugging Face
https://huggingface.co/jinaai/ReaderLM-v2

[9] [10] GitHub - AnswerDotAI/web2md-ext: Get a markdown version of any webpage with a keyboard shortcut.
https://github.com/AnswerDotAI/web2md-ext

[14] [17] Using the ‘Copy as Markdown’ Browser Extension | Jason N. Gaylord
https://www.jasongaylord.com/blog/2020/05/20/copy-as-markdown-browser-extension

[29] Combine Clip selection and URL in a new web clipper option - Support - Joplin Forum
https://discourse.joplinapp.org/t/combine-clip-selection-and-url-in-a-new-web-clipper-option/20384

[33] Turn Any Webpage to Markdown with this Bookmarklet | by AI Rabbit | Medium
https://medium.com/@airabbitX/turn-any-webpage-to-markdown-with-this-ridiculously-simple-bookmarklet-850cf28f8f00

[36] Jina AI - X
https://x.com/JinaAI_/status/1879551743748706487

[37] [38] GitHub - ackness/docs-to-md: readthedocs to markdown
https://github.com/ackness/docs-to-md/

[42] Jina AI Releases Reader-LM 0.5b and 1.5b for converting HTML to ...
https://www.reddit.com/r/LocalLLaMA/comments/1feiip0/jina_ai_releases_readerlm_05b_and_15b_for/

[43] Obsidian's New Web Clipper - You'll Want to Try It - Stephan Miller
https://www.stephanmiller.com/new-obsidian-web-clipper-is-a-game-changer/