

PDF HUB

Complete Website & 33 PDF Tools Technical Documentation Manual

Authoritative System Specifications, Local WebAssembly Architecture, Complete Tool Walkthroughs, Privacy Models, Formats Matrix, and Deployment Reference.

33 Verified Utilities**100% Client-Side RAM Model****Zero Server Uploads****PWA & Offline Ready****WCAG 2.1 AA Compliant****Primary Canonical Domain:** <https://pdfshub.netlify.app/>**Alternative Deployment:** <https://pdfshub.smartstores.in/>**Project Repository:** github.com/officialhazim42/pdfshub**Author / Lead Engineer:** Hazim Bashir (@officialhazim42)**Documentation Version:** 1.0 (Production Release)**Publication Date:** October 2026

This technical manual was compiled directly from the active production codebase. All technical engines, memory limits, and privacy classifications are verified from source implementations.

Table of Contents

Part I: Executive Summary & Architecture Overview	1
1.1 Project Mission & System Overview	1
1.2 Architecture Stack & Runtime Model	1
1.3 Progressive Web App (PWA) Capabilities	2
Part II: Privacy & Data-Processing Architecture	3
2.1 Local WebAssembly Execution Model	3
2.2 31 Local vs. 2 Network-Assisted Tools Classification	3
2.3 Local Storage & Zero Persistent Files Guarantee	4
Part III: Supported Formats & Hardware Memory Limits	5
3.1 Master File Formats Matrix	5
3.2 RAM Bounds & Mobile Constraints	5
Part IV: Universal Troubleshooting & FAQ	6
4.1 Diagnostic Resolution Matrix (OOM, Passwords, Scanned Files)	6
4.2 Master Frequently Asked Questions	7
Part V: Authoritative 33 PDF Tools Technical Reference	8
5.1 Compare PDF (Organize PDF)	8
5.2 Compress PDF (Optimize PDF)	9
5.3 Crop PDF (Edit PDF)	10
5.4 Edit PDF (Edit PDF)	11
5.5 Excel to PDF (Convert to PDF)	12
5.6 Extract Pages (Organize PDF)	13
5.7 PDF Forms (Edit PDF)	14
5.8 HTML to PDF (Convert to PDF)	15
5.9 JPG to PDF (Convert to PDF)	16
5.10 Merge PDF (Organize PDF)	17
5.11 OCR PDF (Extract Content)	18
5.12 Organize PDF (Organize PDF)	19

5.13 Add Page Numbers (Edit PDF)	20
5.14 PDF to Excel (Convert from PDF)	21
5.15 PDF to JPG (Convert from PDF)	22
5.16 PDF to Markdown (Extract Content)	23
5.17 PDF to PDF/A (Optimize PDF)	24
5.18 PDF to PowerPoint (Convert from PDF)	25
5.19 PDF to Word (Convert from PDF)	26
5.20 PowerPoint to PDF (Convert to PDF)	27
5.21 Protect PDF (Secure PDF)	28
5.22 Redact PDF (Secure PDF)	29
5.23 Remove Pages (Organize PDF)	30
5.24 Repair PDF (Optimize PDF)	31
5.25 Rotate PDF (Edit PDF)	32
5.26 Scan to PDF (Convert to PDF)	33
5.27 Sign PDF (Edit PDF)	34
5.28 Split PDF (Organize PDF)	35
5.29 AI Summarizer (Extract Content)	36
5.30 Translate PDF (Extract Content)	37
5.31 Unlock PDF (Secure PDF)	38
5.32 Add Watermark (Edit PDF)	39
5.33 Word to PDF (Convert to PDF)	40

Part VI: Developer Setup, Build Pipelines & QA **41**

6.1 Local Development & NPM Commands	41
6.2 Netlify & Cloudflare Production Pipelines	41
6.3 Automated Test Suites & Verification Procedures	42

Part I: Executive Summary & Architecture Overview

1.1 Project Mission & Overview

PDFHub is a high-performance, browser-first digital document suite offering **33 free, full-featured utilities** for organizing, editing, converting, compressing, repairing, and securing PDF documents.

Unlike traditional web-based PDF converters that transmit confidential user documents to multi-tenant cloud storage buckets, PDFHub compiles document transformation pipelines into WebAssembly and client-side JavaScript. Processing executes in the user's browser memory, guaranteeing absolute data confidentiality, immediate zero-latency execution, and complete independence from server infrastructure.

Core Architectural Tenet: No account required, no registration walls, no subscription fees, no artificial file watermarks, and zero backend document storage.

1.2 Technology Stack

Layer	Technology	Implementation Role
Runtime	HTML5, Vanilla CSS3, Modern ES Modules	Zero-build static web application guaranteeing maximum compatibility and zero runtime bloat.
PDF Stream Processing	<code>pdf-lib</code> (v1.17.1)	Low-level PDF catalog parsing, page tree assembly, xref restructuring, form flattening, and AES encryption.
PDF Rendering & Extraction	<code>pdf.js</code> (v3.11.174)	Pixel-accurate canvas rasterization, text coordinate clustering, and visual difference detection.
OCR Recognition	<code>Tesseract.js</code> (v5.0.5)	WebAssembly-compiled optical character recognition executing inside dedicated background Web Workers.
Office Document Processing	<code>SheetJS</code> , <code>Mammoth.js</code> , <code>docx.js</code> , <code>PptxGenJS</code>	Local parsing and conversion of DOCX, XLSX, and PPTX office document formats in browser RAM.
Packaging	<code>JSZip</code> (v3.10.2)	Client-side binary archive packaging for batch split and image extraction operations.

1.3 Progressive Web App (PWA) Architecture

PDFHub is engineered as a fully compliant Progressive Web App (PWA) under modern web app standards:

- Web App Manifest:** Dual specification (`manifest.webmanifest` and `manifest.json`) with `display: standalone` , custom brand colors, and multi-resolution maskable icons (192px and 512px).
- Service Worker:** Implemented in `sw.js` using a robust cache-first and stale-while-revalidate strategy for application shells, styles, and vendor scripts.
- Offline Shell:** Dedicated `offline.html` fallback for network disconnection states, allowing local tools to operate completely offline.
- Cross-Platform Installation:** One-click installation on Windows, macOS, Linux (Chrome/Edge/Brave), Android, and iOS Safari.

Part II: Privacy & Data-Processing Architecture

2.1 Local WebAssembly Execution Model

Traditional document conversion services require users to upload confidential files over HTTP to third-party web servers. This architecture introduces severe risks including data breaches, accidental retention, unauthorized training on sensitive user documents, and regulatory non-compliance with GDPR, HIPAA, and CCPA standards.

PDFHub reverses this paradigm by transmitting the *processing logic* to the user's browser once, rather than transmitting the user's *data* to the cloud. When a document is loaded into any PDFHub utility:

1. The file is selected via native HTML5 `<input type="file">` or Drag & Drop.
2. The browser runtime reads the binary data into an `ArrayBuffer` using `FileReader.readAsArrayBuffer()`.
3. Client-side WebAssembly engines parse the binary streams in RAM without executing network calls.
4. The resulting document binary is packaged into a temporary memory `Blob` and triggered for instant download via `URL.createObjectURL()`.

2.2 Data Processing Classification

Classification	Count	Tools Included	Network & Transmission Details
100% Client-Side Local	31 Tools	Merge, Split, Remove, Extract, Organize, Scan, Compress, Repair, JPG to PDF, Word to PDF, PPT to PDF, Excel to PDF, HTML to PDF, PDF to JPG, PDF to Word, PDF to PPT, PDF to Excel, PDF to PDF/A, Rotate, Page Numbers, Watermark, Crop, Edit, Forms, Protect, Unlock, Sign, Redact, Compare, Summarize, PDF to Markdown	Zero Network Calls. Files never leave client device memory. Fully operational offline once cached.
CDN Model-Assisted	1 Tool	OCR PDF (<code>ocr.html</code>)	Tesseract.js executes in a local WASM Web Worker. On initial run, it requests trained weights (<code>eng.traineddata.gz</code>) from CDN. Document images remain 100% local.
Hybrid External API	1 Tool	Translate PDF (<code>translate.html</code>)	Text is extracted locally via <code>pdf.js</code> . Selected text chunks (<500 characters) are transmitted via HTTPS to MyMemory Public Translation API.

2.3 Browser Storage & Privacy Verification

PDFHub strictly avoids persisting document binaries in `localStorage`, `IndexedDB`, or session cookies. Browser storage is used solely for benign client preferences:

- `pdfhub-theme` : Stores user color preference (`"dark"` or `"light"`).
- `pdfhub-favorites` : JSON array of tool URL strings bookmarked by the user.
- `pdfhub-recent-history` : Timestamped tool navigation history (tool names only; no file metadata or contents).

Part III: Supported Formats & Hardware Memory Limits

3.1 Master Supported Formats Matrix

Operation Category	Supported Input Formats	Supported Output Formats	Processing Engine
PDF Assembly & Manipulation	<code>.pdf</code> (v1.0 – v2.0)	<code>.pdf</code> , <code>.zip</code>	<code>pdf-lib</code> + <code>JSZip</code>
Office to PDF Conversion	<code>.docx</code> , <code>.pptx</code> , <code>.xlsx</code> , <code>.xls</code> , <code>.csv</code>	Formatted PDF (Browser Print Engine)	<code>Mammoth.js</code> , <code>JSZip</code> , <code>SheetJS</code>
Image & Camera Capture	<code>.jpg</code> , <code>.jpeg</code> , <code>.png</code> , <code>.webp</code> , Camera Stream	<code>.pdf</code> , <code>.jpg</code> , <code>.png</code>	HTML5 Canvas 2D + <code>pdf-lib</code>
PDF to Office Conversion	Text-based <code>.pdf</code>	<code>.docx</code> , <code>.xlsx</code> , <code>.pptx</code> , <code>.txt</code> , <code>.rtf</code>	<code>pdf.js</code> + <code>docx.js</code> / <code>SheetJS</code> / <code>PptxGenJS</code>
Markup & Content Extraction	<code>.html</code> , HTML String, Web URL, <code>.pdf</code>	<code>.pdf</code> , <code>.md</code> , <code>.txt</code>	DOM Parser + <code>pdf.js</code>

3.2 Hardware Memory Limits & Recommendations

Client Memory Bounds: Because all transformations occur in browser RAM, file capacities are bounded by the device's available heap memory rather than cloud server quotas.

Platform / Device	Recommended File Size	Recommended Page Count	Behavior on Exceeded Limits
Modern Desktop (16+ GB RAM)	Up to 150 – 200 MB	Up to 1,000 pages	Flawless execution; large merges may take 3–5 seconds to serialize.
Standard Laptop (8 GB RAM)	Up to 100 MB	Up to 500 pages	Smooth processing; canvas-heavy operations may use ~300 MB browser tab heap.
Mobile / Tablet (iOS / Android)	Up to 40 – 50 MB	Up to 150 pages	Exceeding tab allocation may cause mobile browser to reload tab automatically.

Part IV: Universal Troubleshooting & FAQ

4.1 Diagnostic Resolution Matrix

Observed Symptom	Underlying Root Cause	Verified Technical Resolution
Browser tab crashes or reloads during merge or conversion	Browser ran out of JavaScript heap memory on an exceptionally large PDF (> 150 MB).	Close other browser tabs. For massive documents, use Split PDF to divide files into batches before applying heavy operations.
PasswordRequired or Decryption Error	Source document is encrypted with owner or user password security.	Unlock the document first using the authorized password in Unlock PDF before merging or editing.
Converted Word or Excel file contains no text	Source PDF consists of scanned raster images rather than selectable vector text.	Run the document through OCR PDF to extract text characters from images before converting.
Corrupted or damaged PDF stream error	Damaged cross-reference (xref) table or truncated EOF marker in source file.	Run the damaged file through Repair PDF to regenerate the document catalog and xref table.

4.2 Master Frequently Asked Questions

Q: Are my documents ever uploaded to remote servers?

A: No. 31 of PDFHub's 33 tools run 100% locally in browser memory. OCR PDF runs locally via WebAssembly, fetching language models once from a CDN. Translate PDF transmits short text chunks (<500 chars) to MyMemory API. Your files never touch a PDFHub server.

Q: Can I use PDFHub without an internet connection?

A: Yes. Once loaded or installed as a Progressive Web App (PWA), the service worker caches application assets, enabling full offline operation for all 31 local tools.

Q: Can PDFHub recover a lost password for an encrypted PDF?

A: No. Unlock PDF removes encryption when the user supplies the correct authorized password. PDFHub does not perform cryptographic brute-forcing.

Part V: Authoritative 33 PDF Tools Technical Specifications

The following section provides the complete, source-code-verified technical specification, workflow instructions, execution pipeline, common errors, and FAQ for every single one of the 33 utilities implemented in PDFHub.

1. Compare PDF

Review two PDF versions side by side and highlight visual differences.

PRODUCTION ROUTE <code>/tools/compare.html</code>	UNDERLYING ENGINE <code>pdf.js (dual rendering) + Canvas Pixel-by-Pixel Diff</code>
SUPPORTED INPUTS Two PDF documents (Original PDF and Modified PDF)	SUPPORTED OUTPUTS Interactive side-by-side on-screen visual difference audit
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Compare PDF renders matching pages from an original and modified document, then highlights differing pixels so you can review revisions without uploading either file.

2. Step-by-Step Instructions

- Step 1:** Open Compare PDF and choose the Original PDF file.
- Step 2:** Choose the Modified PDF file and wait for both previews to load.
- Step 3:** Select Compare Pages, then move through pages with the arrows.
- Step 4:** Review the highlighted differences and page counts before acting on the result.

3. Execution Pipeline (What Happens on Execute)

Renders corresponding pages from Document 1 and Document 2 onto dual canvases, extracts pixel image data arrays, compares RGBA values per pixel, and highlights visual discrepancies on an overlay canvas.

Primary Action Trigger: `Compare Pages (#goBtn)`

4. Practical Example Use Case

Auditing contract revisions or engineering blueprints to spot subtle changes or unauthorized wording alterations.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use matching versions with similar page sizes for the clearest comparison. - Check more than the first page when a change may be later in the document.	⚠ Common Mistakes - Selecting the same version twice hides meaningful changes. - Expecting a downloadable redline file; this tool provides an on-screen comparison.
---	--

6. Troubleshooting & Common Errors

Comparing documents with completely different page counts or resolutions. Solution: Use matching document versions.

If a file will not load, confirm it is a readable PDF and try a fresh copy.

If pages have different sizes, compare the content visually rather than relying on pixel counts.

7. Privacy & Data Security

100% Client-Side. Both files remain in local browser memory.

8. Tool FAQ

Q: Does this edit either PDF?

A: No. It only renders and compares pages.

Q: Can I compare more than two files?

A: No. Use two files per comparison.

2. Compress PDF

Reduce PDF structure and metadata overhead in the browser.

PRODUCTION ROUTE
`/tools/compress.html`

SUPPORTED INPUTS
Single PDF document (.pdf)

MEMORY BOUND
Client RAM bound (up to 100 MB)

UNDERLYING ENGINE
`pdf.js (canvas page rasterization) + canvas JPEG re-compression + pdf-lib`

SUPPORTED OUTPUTS
Reduced-size PDF document (compressed.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

Compress PDF rewrites the document with object streams and clears document metadata. It is most useful for structure-heavy PDFs; image-heavy files may not shrink much.

2. Step-by-Step Instructions

- Step 1:** Open Compress PDF and select one PDF.
- Step 2:** Choose Low, Medium, or High compression.
- Step 3:** Select Compress PDF and wait for the generated file.
- Step 4:** Compare the displayed output size with the original and review the PDF before sharing it.

3. Execution Pipeline (What Happens on Execute)

Renders pages onto HTML5 canvas with scaled resolution based on level (Low = 1.5x scale/0.85 quality, Medium = 1.2x scale/0.65 quality, High = 0.9x scale/0.45 quality), encodes canvas frames as compressed JPEGs, and constructs a fresh PDF.

Primary Action Trigger: `Compress PDF (#goBtn)`

4. Practical Example Use Case

Shrinking a 25 MB scanned government filing down to under 5 MB to pass portal upload constraints.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Try Low first when preserving compatibility matters.
- Keep the original because compression does not always reduce image-heavy files.

⚠ Common Mistakes

- Assuming High compression changes embedded image quality; this implementation mainly changes PDF structure.
- Deleting the original before checking the output.

6. Troubleshooting & Common Errors

Rasterization eliminates selectable text layers. Solution: For searchable text preservation, use Low compression or OCR afterwards.
If parsing fails, the PDF may be malformed or encrypted. Try Repair PDF or an unprotected copy.
If the output is larger, keep the original; the tool cannot guarantee a smaller file.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Does compression guarantee a smaller file?

A: No. Results depend on the source structure.

Q: Does it upload my PDF?

A: The conversion runs in the browser.

3. Crop PDF

Change the visible crop box using page margins.

PRODUCTION ROUTE <code>/tools/crop.html</code>	UNDERLYING ENGINE <code>pdf-lib (page.setCropBox, page.setMediaBox)</code>
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Cropped PDF document (cropped.pdf)
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Crop PDF adjusts the visible page bounds with top, bottom, left, and right margins. It can apply the change to all, odd, or even pages.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF to Crop PDF.
- Step 2:** Enter top, bottom, left, and right margins in points.
- Step 3:** Choose whether to apply the margins to all, odd, or even pages.
- Step 4:** Select Crop PDF and inspect the resulting page edges.

3. Execution Pipeline (What Happens on Execute)

Reads MediaBox dimensions, subtracts user-specified margin offsets (top, bottom, left, right in points), sets new CropBox rectangle boundaries on selected pages, and outputs trimmed PDF.

Primary Action Trigger: `Crop PDF (#goBtn)`

4. Practical Example Use Case

Trimming printer bleed marks, header stamps, or excessive white margins from eBooks or academic articles.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use the preview dimensions shown on the page to estimate points. - Start with small margins and verify that headers and footers remain visible.	⚠ Common Mistakes - Entering margins larger than the page dimensions. - Confusing cropping with deleting content; the original content may still exist outside the crop box.
---	---

6. Troubleshooting & Common Errors

Entering margin offsets larger than page dimensions causing inverted crop boxes. Solution: Use modest values (20–50 pt).
If the output is blank or clipped, retry with smaller margins.
If the file cannot be read, use a valid, unencrypted PDF.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I crop individual pages?

A: This page applies the setting by page parity, not individually.

Q: Does cropping remove content permanently?

A: It changes visible bounds rather than redrawing content.

4. Edit PDF

Annotate PDF pages with pen, text, highlights, shapes, and lines.

PRODUCTION ROUTE

/tools/edit.html

SUPPORTED INPUTS

Single PDF document (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

pdf.js (canvas rendering) + HTML5 Drawing Layer + pdf-lib

SUPPORTED OUTPUTS

Annotated and edited PDF document (edited.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Edit PDF renders a page for annotation and lets you draw, type text, highlight, add rectangles or lines, and erase drawn marks before saving.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and wait for the first page to render.
- Step 2:** Choose Pen, Text, Highlight, Rectangle, Line, or Eraser.
- Step 3:** Click or drag on the page to add annotations; move through pages with the page controls.
- Step 4:** Save Edited PDF and review the flattened annotations in the output.

3. Execution Pipeline (What Happens on Execute)

Renders PDF pages to canvas, captures user freehand pen strokes, typed text stamps, semi-transparent highlight rectangles, and shapes, scales coordinates to PDF point space, draws annotations onto PDF streams, and saves.

Primary Action Trigger: Save Edited PDF (#goBtn)

4. Practical Example Use Case

Freehand sketching corrections, adding notes, or highlighting key paragraphs on drafts and student submissions.

5. Best Practices & Mistakes to Avoid

✓ **Practical Tips**

- Zoom your browser before placing small annotations.
- Use Undo while working and save only after checking each page.

⚠ **Common Mistakes**

- Leaving a text field unfinished before changing tools.
- Assuming erasing an annotation removes original PDF text.

6. Troubleshooting & Common Errors

Attempting to edit original pre-existing vector text like a word processor. Note: Edit PDF applies an annotation overlay.
If the page is hard to use on mobile, rotate the device or work at a larger viewport.
If a mark is misplaced, use Undo or Clear Page before exporting.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I edit existing PDF text?

A: The tool adds annotations; it does not rewrite original text.

Q: Are annotations saved in the output?

A: Yes, they are flattened into the exported PDF.

5. Excel to PDF

Turn a selected workbook sheet into a printable PDF using the browser print dialog.

PRODUCTION ROUTE /tools/excel-to-pdf.html	UNDERLYING ENGINE SheetJS (xlsx.full.min.js) + Browser Print Engine
SUPPORTED INPUTS Microsoft Excel spreadsheet (.xlsx, .xls, .csv)	SUPPORTED OUTPUTS Formatted print view triggering browser "Save as PDF" destination
MEMORY BOUND Client RAM bound (~50 MB)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Excel to PDF reads a workbook with SheetJS, lets you choose a sheet, orientation, and page size, then opens a styled print view.

2. Step-by-Step Instructions

- Step 1:** Choose an XLSX, XLS, or CSV file.
- Step 2:** Select the sheet to export and review the table preview.
- Step 3:** Choose portrait or landscape and A4 or Letter.
- Step 4:** Select Convert to PDF, then choose Save as PDF in the print dialog.

3. Execution Pipeline (What Happens on Execute)

Parses spreadsheet workbooks using SheetJS, renders chosen sheet as an HTML table with styled borders, applies orientation (Portrait/Landscape) and page size, and triggers native browser print to PDF.

Primary Action Trigger: [Convert to PDF \(#goBtn\)](#)

4. Practical Example Use Case

Converting financial balance sheets, monthly budget spreadsheets, or employee rosters into clean PDF documents.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use landscape for wide tables. - Check the selected sheet and print preview before saving.	⚠ Common Mistakes - Closing the print dialog without saving. - Expecting formulas or workbook formatting to behave exactly like desktop Excel.
---	--

6. Troubleshooting & Common Errors

Very wide spreadsheets getting cut off horizontally. Solution: Choose Landscape orientation and select A4 or Letter. If the workbook will not load, verify its extension and try opening it in a spreadsheet app first. If columns are clipped, choose landscape or adjust the source sheet.

7. Privacy & Data Security

100% Client-Side. SheetJS runs entirely in browser memory.

8. Tool FAQ

Q: Does this export every sheet?

A: No. Select one sheet at a time.

Q: Where is the file saved?

A: Your browser print dialog controls the destination.

6. Extract Pages

Copy selected page numbers or ranges into a new PDF.

PRODUCTION ROUTE <code>/tools/extract-pages.html</code>	UNDERLYING ENGINE <code>pdf-lib (PDFDocument.create, copyPages, addPage)</code>
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Isolated PDF document (extracted-pages.pdf)
MEMORY BOUND Client RAM bound (~100 MB recommended)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Extract Pages copies the page numbers you specify into a new PDF while preserving their sorted order.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and note the page count shown.
- Step 2:** Enter the pages or ranges you want to keep.
- Step 3:** Select Extract Pages.
- Step 4:** Open the generated PDF and confirm the selected pages.

3. Execution Pipeline (What Happens on Execute)

Reads source PDF, maps comma-separated target page numbers, initializes a target PDFDocument, copies only the requested page indices, and outputs a clean standalone PDF.

Primary Action Trigger: `Extract Pages (#goBtn)`

4. Practical Example Use Case

Extracting just the signature page and contract addendum from a long legal mortgage document.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Use commas between separate selections and hyphens for ranges.
- Keep a copy of the original document.

⚠ Common Mistakes

- Entering zero-based page numbers; the interface uses page 1 as the first page.
- Assuming excluded pages remain in the output.

6. Troubleshooting & Common Errors

Leaving the page input field empty. Solution: Provide page numbers or ranges (e.g. 1-3, 5).

Targeting non-numeric characters. Solution: Use digits, hyphens, and commas only.

If no pages are extracted, use valid numbers within the displayed range.

If parsing fails, try a fresh readable PDF.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I preserve page order?

A: Yes, selected pages are written in ascending order.

Q: Does extraction modify the original?

A: No.

7. PDF Forms

Fill supported interactive PDF form fields and flatten the result.

PRODUCTION ROUTE

`/tools/forms.html`

SUPPORTED INPUTS

Interactive fillable AcroForm PDF (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

`pdf-lib (PDFDocument.getForm, flatten)`

SUPPORTED OUTPUTS

Flattened completed PDF document (filled-form.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PDF Forms reads AcroForm fields, presents editable controls for text, checkboxes, and dropdowns, then flattens the entered values into a new file.

2. Step-by-Step Instructions

- Step 1:** Upload a fillable PDF.
- Step 2:** Wait for the fields list to appear.
- Step 3:** Enter text, select dropdown values, and set checkboxes.
- Step 4:** Select Save Filled PDF and review every field in the downloaded file.

3. Execution Pipeline (What Happens on Execute)

Inspects document catalog for interactive form fields, renders HTML input controls for text fields, checkboxes, and dropdowns, updates AcroForm field values upon submission, flattens fields permanently, and saves.

Primary Action Trigger: `Save Filled PDF (#goBtn)`

4. Practical Example Use Case

Filling out official government tax forms, job applications, or medical registration forms in the browser.

5. Best Practices & Mistakes to Avoid

✓ **Practical Tips**

- Use the field names and types to confirm you are entering data in the intended place.
- Keep the original unfilled form for reuse.

⚠ **Common Mistakes**

- Uploading a scan with no interactive fields.
- Leaving required business information unchecked or blank.

6. Troubleshooting & Common Errors

Loading a flat, scanned non-fillable PDF. Solution: Forms tool requires interactive AcroForm fields.
If no fields are found, use a form-enabled PDF or Edit PDF for visual annotations.
If one field fails, try a simpler value and save again.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Does it support scanned forms?

A: No, scans do not contain interactive fields.

Q: Are fields still editable after saving?

A: The tool flattens them, so keep the original if needed.

8. HTML to PDF

Print pasted HTML or an opened URL to a PDF from the browser.

PRODUCTION ROUTE <code>/tools/html-to-pdf.html</code>	UNDERLYING ENGINE <code>Native Browser Print Engine (window.print)</code>
SUPPORTED INPUTS HTML markup string or website URL	SUPPORTED OUTPUTS PDF document printed via native browser engine
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

HTML to PDF supports pasted markup and URL mode. It opens a print-ready page or the requested URL and relies on the browser print engine.

2. Step-by-Step Instructions

- Step 1:** Choose HTML Code or URL.
- Step 2:** Paste HTML or enter a complete website URL.
- Step 3:** Choose A4 or Letter when using HTML code.
- Step 4:** Select Convert to PDF and choose Save as PDF in the print dialog.

3. Execution Pipeline (What Happens on Execute)

Creates an isolated print document containing user HTML markup and print stylesheets, sets page size (A4, Letter), and triggers window.print(). In URL mode, opens target URL in new tab.

Primary Action Trigger: `Convert to PDF (#goBtn)`

4. Practical Example Use Case

Converting an HTML invoice template, receipt code, or article into a formatted PDF.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use print-friendly CSS and responsive images in pasted HTML. - Check external fonts and images before printing.	⚠ Common Mistakes - Expecting URL mode to bypass a site login or cross-origin restrictions. - Closing the print tab before the browser finishes rendering.
---	---

6. Troubleshooting & Common Errors

External stylesheets or images in raw HTML failing due to CORS when printed. Solution: Use inline CSS or base64 images.

If a URL does not open, verify the address and your network connection.
If images are missing, ensure their URLs are reachable and allowed by the site.

7. Privacy & Data Security

100% Client-Side for HTML markup. URL mode opens public target in new tab.

8. Tool FAQ

Q: Does this create a file automatically?

A: The browser print dialog handles saving.

Q: Can I paste arbitrary HTML?

A: Yes, but external resources may not load.

9. JPG to PDF

Combine JPG, PNG, GIF, or WebP images into a single PDF.

PRODUCTION ROUTE <code>/tools/jpg-to-pdf.html</code>	UNDERLYING ENGINE <code>pdf-lib (embedJpg, embedPng)</code>
SUPPORTED INPUTS One or more JPG, JPEG, PNG, or WebP images	SUPPORTED OUTPUTS Combined PDF document (images.pdf)
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

JPG to PDF embeds selected images as PDF pages, with fit, A4, or Letter sizing and a configurable margin.

2. Step-by-Step Instructions

- Step 1:** Select or drop one or more images.
- Step 2:** Review the image preview grid and remove unwanted images.
- Step 3:** Choose Fit image, A4, or Letter and set the margin.
- Step 4:** Select Convert to PDF and review the page order.

3. Execution Pipeline (What Happens on Execute)

Reads image ArrayBuffers, determines dimensions, creates PDF pages according to chosen page size (Fit to image, A4, Letter), scales image to fit margins, embeds binary pixel streams, and saves PDF.

Primary Action Trigger: `Convert to PDF (#goBtn)`

4. Practical Example Use Case

Bundling smartphone photos of IDs, driver licenses, or expense receipts into a single professional PDF document.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Select images in the order you want them to appear.
- Use a consistent orientation for a cleaner document.

⚠ Common Mistakes

- Uploading a non-image file.
- Assuming the preview order will sort itself; check it before export.

6. Troubleshooting & Common Errors

Attempting to upload unsupported formats like TIFF or BMP. Solution: Save or export images to JPG or PNG first. If an image is not embedded, try PNG or JPG and confirm the browser can display it. If pages are clipped, choose Fit image or reduce the margin.

7. Privacy & Data Security

100% Client-Side. Processed in local browser memory.

8. Tool FAQ

Q: Can I create a multi-page PDF?

A: Yes, each selected image becomes a page.

Q: Does it accept WebP?

A: The interface advertises browser-supported image types including WebP.

10. Merge PDF

Combine multiple PDFs into one document in the selected file order.

PRODUCTION ROUTE

`/tools/merge.html`

SUPPORTED INPUTS

Two or more PDF files (.pdf)

MEMORY BOUND

Client RAM bound (~150-200 MB combined input recommended)

UNDERLYING ENGINE

`pdf-lib` (`PDFDocument.create`, `copyPages`, `addPage`)

SUPPORTED OUTPUTS

Single merged PDF file (merged.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Merge PDF copies every page from two or more selected PDFs into one document, following the order shown in the file list.

2. Step-by-Step Instructions

- Step 1:** Select or drop two or more PDFs.
- Step 2:** Review the file list and remove any incorrect file.
- Step 3:** Confirm the list order is the desired page order.
- Step 4:** Select Merge PDFs and review the combined download.

3. Execution Pipeline (What Happens on Execute)

Reads each file into an `ArrayBuffer`, parses PDF catalogs via `PDFDocument.load`, creates a new blank `PDFDocument`, iterates through each source document copying page references into the target document, and serializes the binary structure with `doc.save()`.

Primary Action Trigger: `Merge PDFs` (`#mergeBtn`)

4. Practical Example Use Case

Combining separated quarterly tax reports, receipts, and an executive summary cover page into one cohesive dossier for filing.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Choose files in the order you want them merged.
- Keep the originals until page order and bookmarks are checked.

⚠ Common Mistakes

- Selecting fewer than two files.
- Assuming filenames are automatically sorted alphabetically.

6. Troubleshooting & Common Errors

Encrypted or password-protected PDF loaded without permissions (fails with `PasswordRequired` error). Solution: Unlock the PDF first with `Unlock PDF`.

Selecting a single file (Merge button remains disabled). Solution: Add at least two PDF documents.

Corrupt PDF stream in one file. Solution: Run the corrupt file through `Repair PDF` before merging.

If one file fails, remove it and test that PDF separately in `Repair PDF`.

If the browser blocks multiple downloads elsewhere, allow downloads for the site.

7. Privacy & Data Security

100% Client-Side. No network calls or server transmissions.

8. Tool FAQ

Q: Can I reorder files after selecting them?

A: The page currently merges in the displayed selection order.

Q: Are PDFs uploaded?

A: The merge happens in the browser.

11. OCR PDF

Recognize text in scanned PDF pages with browser-based Tesseract OCR.

PRODUCTION ROUTE <code>/tools/ocr.html</code>	UNDERLYING ENGINE <code>pdf.js (canvas rendering) + Tesseract.js (Optical Character Recognition)</code>
SUPPORTED INPUTS Scanned or image-only PDF document (.pdf)	SUPPORTED OUTPUTS Searchable extracted text (.txt) and on-screen copyable text
MEMORY BOUND Client RAM bound (~20 pages per batch recommended)	PRIVACY STATUS CDN Language Model Fetch (1x) + Local WASM Worker

1. What the Tool Does

OCR PDF renders selected pages and sends the images through the locally loaded Tesseract.js worker. It returns recognized text rather than a searchable PDF.

2. Step-by-Step Instructions

- Step 1:** Upload a scanned PDF.
- Step 2:** Choose the recognition language and optional pages or ranges.
- Step 3:** Select Extract Text and wait while each page is recognized.
- Step 4:** Review the text, then copy it or download the TXT file.

3. Execution Pipeline (What Happens on Execute)

Renders PDF pages to high-resolution canvas images, feeds pixel matrices to Tesseract.js worker, performs character recognition against selected language traineddata, and compiles output text.

Primary Action Trigger: `Extract Text with OCR (#goBtn)`

4. Practical Example Use Case

Extracting text and tables from flat image-only PDF scans of historical books or paper invoices.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Clear, high-resolution scans and the correct language improve recognition. - Run a few pages first to check accuracy before processing a long document.	⚠ Common Mistakes - Expecting perfect recognition from skewed or low-resolution scans. - Choosing the wrong language for the document.
---	---

6. Troubleshooting & Common Errors

Low resolution scans producing poor OCR accuracy. Solution: Ensure scans are at least 150–300 DPI.
Selecting the wrong language dictionary. Solution: Choose the matching language from the dropdown menu.
If language data cannot load, check connectivity and retry.
If the browser becomes slow, process fewer pages at a time.

7. Privacy & Data Security

Hybrid: Text extraction is 100% local in WebAssembly Worker. Network access is required once to download language trained data from CDN.

8. Tool FAQ

Q: Does OCR create a searchable PDF?

A: No. It outputs recognized text and a text download.

Q: Is OCR fully offline?

A: Required libraries and language data may be loaded from CDNs.

12. Organize PDF

Reorder, rotate, and delete PDF pages using thumbnails.

PRODUCTION ROUTE
`/tools/organize.html`

SUPPORTED INPUTS
Single PDF document (.pdf)

MEMORY BOUND
Client RAM bound (documents up to 80 pages recommended for smooth drag-and-drop)

UNDERLYING ENGINE
`pdf.js (thumbnail rendering) + pdf-lib (page reordering)`

SUPPORTED OUTPUTS
Reordered and rotated PDF document (organized.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

Organize PDF renders page thumbnails so you can drag pages into a new order, rotate individual pages, delete pages, or rotate all pages.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and wait for the thumbnail grid.
- Step 2:** Drag thumbnails to change their order.
- Step 3:** Use the rotate or delete controls on individual pages, or Rotate All.
- Step 4:** Select Save Organized PDF and verify the page sequence.

3. Execution Pipeline (What Happens on Execute)

Renders thumbnail canvases for every page using pdf.js, allows drag-and-drop visual reordering and per-page rotation (+90°), maps the final visual order to page indices, and compiles a reordered PDF using pdf-lib.

Primary Action Trigger: `Save PDF (#goBtn)`

4. Practical Example Use Case

Fixing upside-down scanned invoices and rearranging mixed-up pages before archiving financial records.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Use the page labels after every reorder to confirm the final sequence.
- Keep the source file because deletion and rotation are applied to the exported copy.

⚠ Common Mistakes

- Dragging from the wrong thumbnail after pages have shifted.
- Deleting a page without checking the resulting count.

6. Troubleshooting & Common Errors

Reordering very large documents (100+ pages) causing browser canvas memory pressure. Solution: Split into smaller batches first.

If thumbnails are slow, close other heavy tabs and work in smaller documents.

If a page appears rotated incorrectly, use the individual rotate control again.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I undo a deletion?

A: Not within the current page; reload the original to start over.

Q: Does it change the original?

A: No, it creates a new PDF.

13. Add Page Numbers

Stamp configurable page numbers onto every page of a PDF.

PRODUCTION ROUTE

/tools/page-numbers.html

SUPPORTED INPUTS

Single PDF document (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

pdf-lib (page.drawText, StandardFonts.Helvetica)

SUPPORTED OUTPUTS

Numbered PDF document (numbered.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Add Page Numbers embeds a number label on each page with position, format, starting number, font size, and color controls.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and wait for its page count.
- Step 2:** Choose a position, format, starting number, font size, and color.
- Step 3:** Select Add Numbers.
- Step 4:** Review the first and last pages of the generated PDF.

3. Execution Pipeline (What Happens on Execute)

Iterates through pages, calculates coordinates based on position (Bottom/Top, Center/Left/Right), formats numbering string ("Page X of Y", "X", etc.), embeds font, draws text at exact offsets, and saves PDF.

Primary Action Trigger: Add Page Numbers (#goBtn)

4. Practical Example Use Case

Adding formal pagination to bound manuscripts, legal briefs, reports, or printed thesis documents.

5. Best Practices & Mistakes to Avoid

✓ **Practical Tips**

- Use a smaller font on narrow pages.
- Leave a clear margin at the selected edge.

⚠ **Common Mistakes**

- Choosing white text on a light page.
- Forgetting that the start number affects every subsequent page.

6. Troubleshooting & Common Errors

Page numbers overlapping existing footer text. Solution: Adjust position to Top Right or adjust margin.
If a number overlaps content, choose another position or smaller size.
If the PDF cannot load, repair or export a readable copy first.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I number selected pages only?

A: This page applies numbering to every page.

Q: Does it modify existing text?

A: It adds a new text stamp.

14. PDF to Excel

Extract aligned PDF text into XLSX or CSV rows and columns.

PRODUCTION ROUTE

/tools/pdf-to-excel.html

SUPPORTED INPUTS

Text-based PDF containing structured tables (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

pdf.js (coordinate clustering) + SheetJS (XLSX.write)

SUPPORTED OUTPUTS

Spreadsheet workbook (.xlsx) or comma-separated values (.csv)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PDF to Excel reads text positions, detects rows and columns, and exports the extracted table data. It works best when the source has selectable, aligned text.

2. Step-by-Step Instructions

- Step 1:** Upload the PDF containing a table or aligned text.
- Step 2:** Choose pages, column detection, and output format.
- Step 3:** Use Preview Table to inspect the extracted rows.
- Step 4:** Select Export to Excel and verify totals and column alignment.

3. Execution Pipeline (What Happens on Execute)

Extracts text items and their X/Y coordinates, clusters horizontally aligned items into columns and vertically aligned items into rows, builds a 2D worksheet matrix, and writes an XLSX workbook.

Primary Action Trigger: [Convert to Excel \(#goBtn\)](#)

4. Practical Example Use Case

Extracting bank statement transaction tables, invoices, or financial audits from PDF into Excel for analysis.

5. Best Practices & Mistakes to Avoid

✓ **Practical Tips**

- Try Auto first, then Tab/space or Each line if the table structure is unusual.
- Always compare important totals with the source PDF.

⚠ **Common Mistakes**

- Using a scanned PDF without OCR.
- Treating visual spacing as guaranteed spreadsheet columns.

6. Troubleshooting & Common Errors

Irregularly spaced text items causing offset columns. Solution: Check table layout in the preview modal.
If no data appears, run OCR first or choose a text-based source.
If columns shift, change the detection mode and preview again.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can it extract scanned tables?

A: Not directly; run OCR first.

Q: Does preview export the file?

A: No, use Export to Excel.

15. PDF to JPG

Render selected PDF pages into JPG or PNG images.

PRODUCTION ROUTE

`/tools/pdf-to-jpg.html`

SUPPORTED INPUTS

Single PDF document (.pdf)

MEMORY BOUND

Client RAM bound (~100 MB)

UNDERLYING ENGINE

`pdf.js (page canvas rendering) + canvas.toBlob + JSZip`

SUPPORTED OUTPUTS

Individual JPG/PNG image files or a packaged .zip archive

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PDF to JPG renders selected pages with PDF.js at the chosen quality and downloads each image separately.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose quality and JPG or PNG format.
- Step 2:** Enter page numbers or leave the range blank for all pages.
- Step 3:** Select Convert to Images.
- Step 4:** Open the downloaded images and verify resolution and page order.

3. Execution Pipeline (What Happens on Execute)

Iterates through selected PDF pages, renders each page onto an HTML5 canvas via pdf.js at chosen scale (1x, 1.5x, 2x), extracts image blobs (JPEG/PNG), and either downloads directly or bundles all into a ZIP.

Primary Action Trigger: `Convert to Images (#goBtn)`

4. Practical Example Use Case

Extracting PDF brochure pages for posting on Instagram, LinkedIn, or incorporating into social media marketing.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Use Standard quality for quick previews and High for printed output.
- Convert only the pages you need to reduce memory use.

⚠ Common Mistakes

- Selecting a page outside the document range.
- Expecting a single multi-page image file.

6. Troubleshooting & Common Errors

Selecting high resolution on a 200-page PDF exhausting canvas memory. Solution: Render specific page ranges. If rendering fails, retry with a smaller page range or lower quality. If downloads are blocked, allow multiple downloads for PDFHub.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Does each page become a separate file?

A: Yes.

Q: Can I choose PNG?

A: Yes, the format selector offers JPG and PNG.

16. PDF to Markdown

Extract selectable PDF text into editable Markdown with optional headings.

PRODUCTION ROUTE

/tools/pdf-to-markdown.html

SUPPORTED INPUTS

Text-based PDF document (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

pdf.js (text and font styling inspection)

SUPPORTED OUTPUTS

Clean Markdown document (.md) and formatted preview

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PDF to Markdown extracts text with PDF.js, optionally detects headings and lists, and provides an editable output before downloading.

2. Step-by-Step Instructions

- Step 1:** Upload a text-based PDF.
- Step 2:** Choose whether to add page separators and detect headings and lists.
- Step 3:** Select Convert to Markdown and review the editable output.
- Step 4:** Copy the Markdown or select Download .md.

3. Execution Pipeline (What Happens on Execute)

Extracts text fragments, detects heading levels based on font size differentials, identifies bullet lists and indentation, formats clean Markdown syntax, and provides copy-to-clipboard and .md file download.

Primary Action Trigger: Convert to Markdown (#goBtn)

4. Practical Example Use Case

Converting PDF technical documentation or blog articles into Markdown for GitHub repositories or static site generators.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Review heading detection because visual layout cannot always be inferred from PDF text order.
- Clean up tables and complex layouts manually.

⚠ Common Mistakes

- Assuming scanned pages contain selectable text.
- Downloading without reviewing line order.

6. Troubleshooting & Common Errors

Scanned PDFs without text will produce empty Markdown files. Solution: Use OCR PDF first.

If the result is empty, run OCR or use a PDF with selectable text.

If paragraphs are out of order, the source layout may not expose reading order.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I edit the generated Markdown?

A: Yes, the output is an editable text area.

Q: Does it preserve images?

A: It focuses on extracted text and Markdown structure.

17. PDF to PDF/A

Rewrite a PDF with selectable archival metadata and PDF/A options.

PRODUCTION ROUTE
`/tools/pdf-to-pdfa.html`

SUPPORTED INPUTS
Standard PDF document (.pdf)

MEMORY BOUND
Client RAM bound

UNDERLYING ENGINE
`pdf-lib (metadata injection & OutputIntent dictionaries)`

SUPPORTED OUTPUTS
Archival-compliant PDF/A document (archival-pdfa.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

PDF to PDF/A sets metadata and archival catalog entries for the selected PDF/A level. The page itself warns that certified ISO compliance needs a dedicated validator.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF.
- Step 2:** Choose PDF/A-1b, 2b, or 3b and enter optional title and author.
- Step 3:** Select Convert to PDF/A.
- Step 4:** Validate the downloaded result with an archival validator before relying on it.

3. Execution Pipeline (What Happens on Execute)

Loads PDF byte stream, injects ISO 19005-1 / 19005-2 PDF/A conformance tags, builds OutputIntents dictionary with RGB color profile identifier, writes Dublin Core XMP metadata, and saves document.

Primary Action Trigger: `Convert to PDF/A (#goBtn)`

4. Practical Example Use Case

Preparing legal court filings, academic dissertations, or official records requiring archival PDF/A preservation.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Keep the source PDF for future edits.
- Use descriptive title and author metadata for retrieval.

⚠ Common Mistakes

- Treating the output label as proof of ISO compliance.
- Using unsupported interactive features in an archival workflow.

6. Troubleshooting & Common Errors

Non-standard embedded fonts in original PDF may fail strict external ISO validators. Verify before regulatory submission.
If conversion fails, try a readable unencrypted PDF.
If a validator reports errors, use a certified PDF/A converter.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Does PDFHub guarantee PDF/A compliance?

A: No; the tool explicitly recommends certified validation.

Q: Can I attach files with PDF/A-3b?

A: The UI exposes the level but does not add attachments.

18. PDF to PowerPoint

Render up to 20 PDF pages as slide images in a PPTX.

PRODUCTION ROUTE /tools/pdf-to-ppt.html	UNDERLYING ENGINE pdf.js (canvas rendering) + PptxGenJS (pptxgen.bundle.js)
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Microsoft PowerPoint presentation (.pptx)
MEMORY BOUND Client RAM bound (~50 MB)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

PDF to PowerPoint renders selected PDF pages to images and places each page on a presentation slide. The result is image-based rather than editable text.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose pages, slide quality, and slide size.
- Step 2:** Review the generated page thumbnails.
- Step 3:** Select Export as PPTX.
- Step 4:** Open the presentation and check image scale and slide order.

3. Execution Pipeline (What Happens on Execute)

Renders each PDF page into a high-resolution image blob via pdf.js, initializes a PptxGenJS presentation instance with selected aspect ratio (16:9 or 4:3), adds a slide for each page, and writes the .pptx binary.

Primary Action Trigger: `Convert to PowerPoint (#goBtn)`

4. Practical Example Use Case

Converting exported PDF presentations back into a slide deck to present in PowerPoint or Google Slides.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use High quality for presentation screens and Standard for faster processing. - Select only needed pages to keep the presentation manageable.	⚠ Common Mistakes - Expecting the PDF text to become editable slide text. - Selecting more than the supported 20-page range.
--	--

6. Troubleshooting & Common Errors

Text on slides is rendered as raster images rather than vector editable text. Slides serve as backdrop slides. If export fails, lower the quality or reduce the page selection. If slides appear soft, rerun with High quality.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Are slides editable?

A: Each PDF page is placed as an image.

Q: Can I export all pages?

A: Only up to the page limit exposed by the tool.

19. PDF to Word

Extract PDF text into DOCX, RTF, or plain text output.

PRODUCTION ROUTE

</tools/pdf-to-word.html>

SUPPORTED INPUTS

Text-based PDF document (.pdf)

MEMORY BOUND

Client RAM bound (~50 MB)

UNDERLYING ENGINE

pdf.js (text line extraction) + docx.js
(index.umd.js) + fallback RTF/TXT

SUPPORTED OUTPUTS

Editable Microsoft Word document (.docx), Rich Text (.rtf), or Plain Text (.txt)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PDF to Word groups extracted PDF text into lines and produces an editable document or text file. Complex layouts and scans may need cleanup or OCR.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and select the pages to convert.
- Step 2:** Choose DOCX, RTF, or plain text.
- Step 3:** Select Convert to Word and wait for text extraction.
- Step 4:** Open the output and compare important formatting with the PDF.

3. Execution Pipeline (What Happens on Execute)

Extracts text content fragments and bounding coordinates with pdf.js, groups text items into vertical line structures, constructs a Document AST with Paragraph and Heading objects in docx.js, and packs into a .docx blob.

Primary Action Trigger: [Convert to Word \(#goBtn\)](#)

4. Practical Example Use Case

Converting agreements or reports back into editable Microsoft Word documents to revise clauses or wording.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Run OCR first for scanned documents.
- Use TXT or RTF when a simple, portable result is more important than layout.

⚠ Common Mistakes

- Expecting exact columns, images, or typography from text extraction.
- Skipping the output review.

6. Troubleshooting & Common Errors

Scanned PDFs containing images of text produce empty output. Solution: Run OCR PDF first to generate a text layer. If no text is extracted, use OCR PDF.

If DOCX generation fails, the page may fall back to RTF; review the status message.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can it convert scanned PDFs?

A: OCR is recommended first.

Q: Does it preserve the original layout?

A: It preserves text order approximately, not full visual layout.

20. PowerPoint to PDF

Read PPTX slide text and create a browser print layout for PDF.

PRODUCTION ROUTE

</tools/ppt-to-pdf.html>

SUPPORTED INPUTS

Microsoft PowerPoint presentation (.pptx)

MEMORY BOUND

Client RAM bound (~50 MB PPTX)

UNDERLYING ENGINE

JSZip (PPTX XML & media extraction) + Browser Print Engine

SUPPORTED OUTPUTS

Formatted print view triggering browser "Save as PDF" destination

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

PowerPoint to PDF reads slide XML from a PPTX archive, previews extracted slide text, and opens a printable layout with slide and handout options.

2. Step-by-Step Instructions

- Step 1:** Select a PPTX presentation.
- Step 2:** Wait for slide text previews, then choose a slide layout and background.
- Step 3:** Select Convert to PDF.
- Step 4:** Choose Save as PDF in the print dialog and inspect the result.

3. Execution Pipeline (What Happens on Execute)

Unzips PPTX package in memory, extracts slide XML, geometries, text runs, and embedded media, formats slides into print CSS pages (1 slide/page or 2 slides/page handouts), and opens browser print dialog.

Primary Action Trigger: [Convert to PDF \(#goBtn\)](#)

4. Practical Example Use Case

Preparing lecture slide decks or business presentations for distribution and printing as PDF handouts.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Check the slide preview for missing text before printing.
- Use one slide per page when readability matters.

⚠ Common Mistakes

- Expecting animations and all slide artwork to be preserved.
- Closing the print dialog without choosing Save as PDF.

6. Troubleshooting & Common Errors

Uploading legacy binary .ppt files. Solution: Save as .pptx first.

Complex PowerPoint animation effects are omitted in static PDF export.

If the PPTX will not open, confirm it is a valid Office Open XML presentation.

If slide images or formatting are missing, export from PowerPoint for a faithful result.

7. Privacy & Data Security

100% Client-Side. Zip extraction occurs in local memory.

8. Tool FAQ

Q: Does it preserve animations?

A: No, PDF is static.

Q: Can I create handouts?

A: The page offers two- and four-per-page layouts.

21. Protect PDF

Add password encryption and permission settings to a PDF.

PRODUCTION ROUTE
`/tools/protect.html`

SUPPORTED INPUTS
Single PDF document (.pdf)

MEMORY BOUND
Client RAM bound

UNDERLYING ENGINE
`pdf-lib (userPassword, ownerPassword, permissions)`

SUPPORTED OUTPUTS
Password-encrypted PDF document (protected.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

Protect PDF uses pdf-lib password options to create an encrypted copy with a chosen password and permissions.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF.
- Step 2:** Enter a password twice and choose the available permission preset.
- Step 3:** Select Protect PDF.
- Step 4:** Open the downloaded file and test the password before deleting the source.

3. Execution Pipeline (What Happens on Execute)

Loads PDF with ignoreEncryption flag, invokes doc.save({ userPassword, ownerPassword, permissions }), generates standard AES encryption dictionaries, and outputs a protected document.

Primary Action Trigger: `Protect PDF (#goBtn)`

4. Practical Example Use Case

Securing confidential payroll sheets, medical records, or tax returns before emailing them.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Use a unique, long password and store it securely.
- Share the password through a separate channel from the document.

⚠ Common Mistakes

- Forgetting the password.
- Assuming viewer permissions are enforced equally by every PDF reader.

6. Troubleshooting & Common Errors

Forgetting the password. Solution: PDFHub cannot recover lost passwords; retain a secure record.
If passwords do not match, re-enter both fields.
If encryption fails, retry with a readable PDF and a strong browser session.

7. Privacy & Data Security

100% Client-Side. Password and file never leave browser memory.

8. Tool FAQ

Q: Can PDFHub recover a forgotten password?

A: No.

Q: Does this upload the password?

A: The operation is performed in the browser.

22. Redact PDF

Draw black rectangles over sensitive areas and export a flattened PDF.

PRODUCTION ROUTE

`/tools/redact.html`

SUPPORTED INPUTS

Single PDF document (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

`pdf.js (preview) + pdf-lib (page.drawRectangle with black fill)`

SUPPORTED OUTPUTS

Permanently redacted PDF document (redacted.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Redact PDF renders pages for marking, then draws black rectangles into the output PDF. Inspect the exported copy carefully before sharing.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose a page.
- Step 2:** Drag over every sensitive region to create a black box.
- Step 3:** Move through pages and repeat as needed; use Clear Page to start a page again.
- Step 4:** Select Save Redacted PDF and verify the result in a separate viewer.

3. Execution Pipeline (What Happens on Execute)

Renders page previews, captures user-drawn bounding boxes over sensitive regions, translates coordinates to PDF coordinate space, draws opaque black rectangles permanently onto the page content stream, and saves.

Primary Action Trigger: `Save Redacted PDF (#goBtn)`

4. Practical Example Use Case

Blacking out Social Security Numbers, credit card details, or confidential client names before public disclosure.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Cover complete lines and nearby identifying context, not just the visible word.
- Search or select text in the exported PDF to confirm sensitive content is no longer exposed.

⚠ Common Mistakes

- Redacting only a screenshot layer or forgetting another page.
- Sharing the source PDF instead of the redacted output.

6. Troubleshooting & Common Errors

Assuming transparent highlights redact text. Redact tool draws solid 100% opaque black rectangles.

If a box is misplaced, clear that page and redraw.

If text can still be selected outside a box, expand the redaction area and export again.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Is redaction reversible?

A: Treat the exported result as permanent, but retain the source privately.

Q: Does it detect sensitive text automatically?

A: No, marking is manual.

23. Remove Pages

Delete selected page numbers or ranges from a PDF.

PRODUCTION ROUTE
`/tools/remove-pages.html`

SUPPORTED INPUTS
Single PDF document (.pdf)

MEMORY BOUND
Client RAM bound (~100 MB recommended)

UNDERLYING ENGINE
`pdf-lib (PDFDocument.load, removePage)`

SUPPORTED OUTPUTS
Cleaned PDF document (removed-pages.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

Remove Pages copies every page except the page numbers or ranges you enter. It prevents exporting an empty document.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and note the page count.
- Step 2:** Enter the page numbers or ranges to remove.
- Step 3:** Select Remove Pages and confirm the result count in the status message.
- Step 4:** Open the output and check the first, last, and adjacent pages.

3. Execution Pipeline (What Happens on Execute)

Parses the PDF structure, converts comma-separated page input into a 0-indexed set, sorts page indices in descending order, invokes `doc.removePage(idx)` sequentially to avoid index shift, and re-serializes the pruned PDF.

Primary Action Trigger: `Remove Pages (#goBtn)`

4. Practical Example Use Case

Deleting blank scanner separation sheets and confidential pricing tables before sharing a proposal with a client.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Write down the pages to keep before removing several sections.
- Keep the original as a backup.

⚠ Common Mistakes

- Entering an invalid range or accidentally targeting every page.
- Forgetting that later page numbers shift after removal.

6. Troubleshooting & Common Errors

Removing all pages in the document. Solution: Ensure at least one page remains in the target document.
Typing out-of-bounds page numbers. Solution: Check total pages displayed in the file information badge.
If no pages are removed, enter at least one valid page number.
If all pages are selected, the tool stops instead of creating an empty PDF.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I remove all pages?

A: No, the tool prevents an empty result.

Q: Can I undo removal?

A: Re-upload the original and select a different range.

24. Repair PDF

Try to parse and reserialize a malformed PDF structure.

PRODUCTION ROUTE
`/tools/repair.html`

SUPPORTED INPUTS
Corrupted or unreadable PDF document (.pdf)

MEMORY BOUND
Client RAM bound

UNDERLYING ENGINE
`pdf-lib (PDFDocument.load with ignoreEncryption & xref recovery)`

SUPPORTED OUTPUTS
Repaired and normalized PDF document (repaired.pdf)

PRIVACY STATUS
100% Client-Side Local Memory

1. What the Tool Does

Repair PDF uses lenient pdf-lib loading and reserialization to rebuild a document that can still be parsed. It cannot restore missing data.

2. Step-by-Step Instructions

- Step 1:** Upload the damaged PDF.
- Step 2:** Select Repair PDF to start lenient parsing.
- Step 3:** Wait while the document is reserialized.
- Step 4:** Open the repaired download and compare its pages with the original.

3. Execution Pipeline (What Happens on Execute)

Attempts tolerant parsing of damaged byte streams, rebuilds broken cross-reference (xref) tables, recovers orphaned object dictionaries, and serializes a clean standards-compliant PDF file.

Primary Action Trigger: `Repair PDF (#goBtn)`

4. Practical Example Use Case

Fixing older PDFs downloaded from legacy portals that throw syntax errors in Acrobat or preview tools.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Keep the damaged source and make a copy before experimenting.
- Try a PDF exported again by the original application if repair cannot recover it.

⚠ Common Mistakes

- Treating repair as guaranteed recovery.
- Overwriting the only copy of the source.

6. Troubleshooting & Common Errors

Severely truncated or zero-byte files where stream data is completely missing cannot be magically reconstructed. If parsing fails, obtain a fresh export or use a dedicated desktop repair tool. If pages are missing, the source may not contain recoverable data.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can it repair every PDF?

A: No. It only helps when the structure remains parseable.

Q: Does repair upload the file?

A: The page processes it in the browser.

25. Rotate PDF

Rotate all, odd, even, or custom PDF pages by a selected angle.

PRODUCTION ROUTE <code>/tools/rotate.html</code>	UNDERLYING ENGINE <code>pdf-lib (page.setRotation)</code>
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Rotated PDF document (rotated.pdf)
MEMORY BOUND Client RAM bound (~150 MB)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Rotate PDF updates page rotation metadata for the selected pages. It supports all, odd, even, or custom page selections.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF.
- Step 2:** Choose 90, 180, or 270 degrees.
- Step 3:** Choose All pages, Even pages, Odd pages, or Custom pages.
- Step 4:** Select Rotate PDF and inspect the page orientation.

3. Execution Pipeline (What Happens on Execute)

Reads PDF pages, computes target orientation by adding selected angle (+90°, +180°, +270°) modulo 360 to existing rotation attribute, applies transformation to selected pages (all, odd, even, custom), and saves binary.

Primary Action Trigger: `Rotate PDF (#goBtn)`

4. Practical Example Use Case

Correcting sideways or upside-down orientation on scanned tax returns, contracts, or architectural schematics.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use Preview or a PDF viewer to confirm orientation before sharing. - Use custom pages for mixed portrait and landscape scans.	⚠ Common Mistakes - Confusing odd/even page parity with zero-based indexes. - Rotating all pages when only scanned pages need correction.
---	--

6. Troubleshooting & Common Errors

Applying rotation to all pages when only page 1 was sideways. Solution: Use the custom page range filter.
If custom rotation does nothing, check that page numbers are valid.
If a viewer caches the old display, close and reopen the output.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Does rotation change text?

A: No, it changes page orientation metadata.

Q: Can I rotate by 45 degrees?

A: No, the controls provide 90-degree increments.

26. Scan to PDF

Capture camera images or upload photos and assemble them into a PDF.

PRODUCTION ROUTE

`/tools/scan-to-pdf.html`

SUPPORTED INPUTS

Direct webcam / camera stream or uploaded images (.jpg, .png, .webp)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

`navigator.mediaDevices.getUserMedia` + `HTML5 Canvas`
+ `pdf-lib`

SUPPORTED OUTPUTS

Multi-page scanned PDF document (scanned-document.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Scan to PDF uses the camera when permitted or accepts uploaded images, optionally applies grayscale or high contrast, and embeds each image on a page.

2. Step-by-Step Instructions

- Step 1:** Choose Start Camera and grant permission, or upload document images.
- Step 2:** Capture or add all pages and remove any incorrect image from the grid.
- Step 3:** Choose Fit image, A4, or Letter and an enhancement.
- Step 4:** Select Create PDF and review every scanned page.

3. Execution Pipeline (What Happens on Execute)

Connects to camera stream, captures snapshots to an in-memory canvas, optionally applies B&W contrast threshold or grayscale filters, converts frames to JPEG blobs, and embeds them as pages into a new PDF.

Primary Action Trigger: `Generate PDF (#goBtn)`

4. Practical Example Use Case

Photographing paper receipts, signed paper contracts, or student homework directly on a phone or laptop webcam into a PDF.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Use even lighting and keep the camera parallel to the page.
- Preview enhancement choices so small text remains readable.

⚠ Common Mistakes

- Leaving the camera stream running after finishing.
- Adding pages in the wrong order.

6. Troubleshooting & Common Errors

Camera permission denied by browser. Solution: Enable camera permissions in the browser address bar icon.
Generating PDF without capturing any snaps. Solution: Click "Capture Page" at least once before generating.
If camera access is denied, use image upload or change the browser permission.
If the output is rotated, retake the image or rotate it before scanning.

7. Privacy & Data Security

100% Client-Side. Camera frames are processed directly in local canvas memory.

8. Tool FAQ

Q: Can I use this without a camera?

A: Yes, upload JPG, PNG, or WebP images.

Q: Does it perform OCR?

A: No, use OCR PDF for text recognition.

27. Sign PDF

Draw or type a signature and place it on a rendered PDF page.

PRODUCTION ROUTE /tools/sign.html	UNDERLYING ENGINE HTML5 Signature Pad Canvas + pdf-lib (embedPng, drawImage)
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Signed PDF document (signed.pdf)
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Sign PDF creates a drawn or typed signature, lets you click to place it on pages, and embeds the signature image in the exported document.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose Draw or Type.
- Step 2:** Create your signature, then navigate to the target page.
- Step 3:** Click the page to place the signature and drag it into position if needed.
- Step 4:** Select Save Signed PDF and review placement before sending it.

3. Execution Pipeline (What Happens on Execute)

Allows user to draw signature on touch/mouse canvas or generate typed signature with cursive script font, places signature stamp over selected page and coordinates, embeds signature PNG into PDF, and saves.

Primary Action Trigger: Save Signed PDF (#goBtn)

4. Practical Example Use Case

Signing employment offer letters, NDAs, rental lease agreements, or sales contracts electronically.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Keep the signature size consistent and place it away from existing text. - Use a copy of the document when testing placement.	⚠ Common Mistakes - Saving without placing a signature. - Placing the signature on the wrong page after navigating.
---	--

6. Troubleshooting & Common Errors

Placing signature stamp outside visible page boundary. Solution: Click directly onto the document preview canvas. If drawing is difficult, use Type mode or a stylus/touch screen. If the signature looks offset, adjust its position before saving.

7. Privacy & Data Security

100% Client-Side. Signature drawing and document remain strictly in browser.

8. Tool FAQ

Q: Is this a certificate-based digital signature?

A: No, it embeds a drawn or typed image.

Q: Can I place signatures on multiple pages?

A: Yes, add one on each page as needed.

28. Split PDF

Create separate PDFs from ranges, every page, or groups of N pages.

PRODUCTION ROUTE <code>/tools/split.html</code>	UNDERLYING ENGINE <code>pdf-lib + JSZip</code>
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Individual .pdf files or a bundled .zip archive
MEMORY BOUND Client RAM bound (documents up to 100 MB or ~500 pages recommended)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Split PDF copies selected page groups into separate files. It supports explicit ranges, one file per page, and every-N-pages grouping.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose a split mode.
- Step 2:** Enter ranges or a group size when the chosen mode needs it.
- Step 3:** Select Split PDF and wait for each group to be created.
- Step 4:** Open the resulting files and verify their names and page ranges.

3. Execution Pipeline (What Happens on Execute)

Loads the PDF into an ArrayBuffer, resolves total page count, parses the chosen split mode (ranges, all pages, or every N pages), extracts page subsets into fresh PDFDocument instances, serializes them, and bundles multiple outputs into a ZIP via JSZip.

Primary Action Trigger: `Split PDF (#splitBtn)`

4. Practical Example Use Case

Extracting a 5-page confidential section from a 100-page employee handbook to email to department heads.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use explicit ranges when the output groups need different sizes. - Allow multiple downloads in the browser for multi-file exports.	⚠ Common Mistakes - Leaving ranges empty in range mode. - Choosing Every page for a very large document without expecting many downloads.
--	--

6. Troubleshooting & Common Errors

Invalid page range format (e.g. typing "5-2" instead of "2-5"). Solution: Use ascending numbers separated by commas or hyphens.

Requesting page numbers higher than total document page count. Solution: Review page numbers in the loaded file card.

If no files are created, correct the ranges or N value.

If downloads are blocked, update the browser permission for this site.

7. Privacy & Data Security

100% Client-Side. Processed in local browser memory.

8. Tool FAQ

Q: Can I split every page separately?

A: Yes, choose Every page as separate PDF.

Q: Does the original remain intact?

A: Yes.

29. AI Summarizer

Extract PDF text and request a summary through the page's configured AI service.

PRODUCTION ROUTE <code>/tools/summarize.html</code>	UNDERLYING ENGINE <code>pdf.js (text extraction) + Client-Side Heuristic Sentence Scoring (localSummarize)</code>
SUPPORTED INPUTS Text-based PDF document (.pdf)	SUPPORTED OUTPUTS Formatted text summary on-screen, copy-to-clipboard, or .txt download
MEMORY BOUND Client RAM bound (up to 50 pages)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

AI Summarizer extracts up to the configured page limit and builds a summary request for the selected style. Its final result depends on the configured external AI service and network availability.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose a summary style.
- Step 2:** Set the maximum number of pages to process.
- Step 3:** Select Summarize PDF and wait for extraction and the service response.
- Step 4:** Review the summary against the original, then copy or download it.

3. Execution Pipeline (What Happens on Execute)

Extracts live text runs from first N pages, cleans and tokenizes sentences, computes word frequency against a stop-word dictionary, scores sentences by keyword density, formats summary based on style (Concise, Detailed, Bullets, Executive), and displays result.

Primary Action Trigger: `Summarize PDF (#goBtn)`

4. Practical Example Use Case

Quickly understanding key takeaways and findings from 40-page academic papers, annual reports, or industry whitepapers.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Verify important claims against the source document. - Use concise style for quick orientation and detailed style for a structured review.	⚠ Common Mistakes - Treating an AI summary as a substitute for the original. - Sending sensitive documents without checking the service behavior.
--	--

6. Troubleshooting & Common Errors

Attempting to summarize a scanned image-only PDF. Solution: Run OCR PDF first to generate a text layer. If the request fails, check connectivity and retry with fewer pages. If extraction is empty, run OCR first for a scanned PDF.

7. Privacy & Data Security

100% Client-Side. PDF.js extracts text locally; local keyword-density NLP analyzes sentences in browser RAM.

8. Tool FAQ

Q: Is the summary guaranteed accurate?

A: No; AI output can omit or misstate details.

Q: Does this work fully offline?

A: The configured AI request may require network access.

30. Translate PDF

Extract selected PDF text and translate it through the configured free translation API.

PRODUCTION ROUTE <code>/tools/translate.html</code>	UNDERLYING ENGINE <code>pdf.js (text extraction) + MyMemory Public Translation API + pdf-lib</code>
SUPPORTED INPUTS Text-based PDF document (.pdf)	SUPPORTED OUTPUTS Translated text view, .txt download, or translated PDF document
MEMORY BOUND Client RAM bound (up to 20 pages)	PRIVACY STATUS Hybrid: Local Extraction + MyMemory Public API

1. What the Tool Does

Translate PDF extracts selected pages and sends short text chunks to the MyMemory translation API, then lets you review, copy, download, or save the result.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and choose source and target languages.
- Step 2:** Enter pages or ranges, or leave the field blank for the allowed page limit.
- Step 3:** Select Translate PDF and wait for each page’s text to be processed.
- Step 4:** Review names, numbers, and terminology before copying or saving the translation.

3. Execution Pipeline (What Happens on Execute)

Extracts text lines from selected pages, partitions text into chunks, sends translation requests across chosen language pairs, displays translated results, and optionally writes translated text to a new PDF.

Primary Action Trigger: `Translate PDF (#goBtn)`

4. Practical Example Use Case

Translating foreign language user manuals, academic abstracts, or product specifications into English or Spanish.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use the correct source language when auto-detection is uncertain. - Review legal, medical, and technical translations with a qualified person.	⚠ Common Mistakes - Assuming a translation API can preserve the original layout. - Ignoring the 20-page limit.
--	---

6. Troubleshooting & Common Errors

Exceeding free translation API daily rate limits. Solution: Translate essential pages or short sections.
If translation fails, check network access and retry a shorter selection.
If text is empty, run OCR first for a scanned PDF.

7. Privacy & Data Security

Hybrid: Text extraction is local. Text chunks (up to 500 characters) are transmitted via HTTPS to MyMemory Translation API.

8. Tool FAQ

Q: Does it translate images?

A: No, it extracts text.

Q: Is the translation guaranteed?

A: No; machine translation needs review.

31. Unlock PDF

Remove supported PDF encryption when you provide the document password.

PRODUCTION ROUTE

`/tools/unlock.html`

SUPPORTED INPUTS

Password-protected PDF document (.pdf)

MEMORY BOUND

Client RAM bound

UNDERLYING ENGINE

`pdf-lib (PDFDocument.load with password)`

SUPPORTED OUTPUTS

Unencrypted PDF document (unlocked.pdf)

PRIVACY STATUS

100% Client-Side Local Memory

1. What the Tool Does

Unlock PDF loads a protected PDF with the entered password and saves a new copy without the original encryption when the browser library supports it.

2. Step-by-Step Instructions

- Step 1:** Upload the protected PDF.
- Step 2:** Enter its password if the document requests one.
- Step 3:** Select Unlock PDF.
- Step 4:** Open the output and confirm it no longer prompts for the password.

3. Execution Pipeline (What Happens on Execute)

Attempts to decrypt and load PDF using provided password, strips internal encryption handler dictionaries, and re-saves unencrypted document.

Primary Action Trigger: `Unlock PDF (#goBtn)`

4. Practical Example Use Case

Removing the password from your own bank statements or tax documents so you do not need to re-type it every time.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips

- Only unlock documents you own or are authorized to process.
- Keep the protected original in a secure location.

⚠ Common Mistakes

- Trying to bypass a password you do not know.
- Assuming every encryption scheme is supported.

6. Troubleshooting & Common Errors

Entering an incorrect password fails decryption. PDFHub decrypts authorized files; it does not brute-force passwords. If the password is rejected, verify it and check capitalization. If the encryption is unsupported, use the original authoring application or a compatible tool.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can it guess a password?

A: No.

Q: Does it remove permissions-only restrictions?

A: Results depend on the PDF library and encryption type.

32. Add Watermark

Stamp custom text across every page with opacity, color, angle, and size controls.

PRODUCTION ROUTE <code>/tools/watermark.html</code>	UNDERLYING ENGINE <code>pdf-lib (page.drawText with rotation & opacity)</code>
SUPPORTED INPUTS Single PDF document (.pdf)	SUPPORTED OUTPUTS Watermarked PDF document (watermarked.pdf)
MEMORY BOUND Client RAM bound	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Add Watermark draws the text you provide across every page using the selected style settings.

2. Step-by-Step Instructions

- Step 1:** Upload a PDF and enter the text to stamp.
- Step 2:** Choose font size, opacity, color, and angle.
- Step 3:** Select Add Watermark.
- Step 4:** Review every page for readability and placement before sharing.

3. Execution Pipeline (What Happens on Execute)

Calculates page center coordinates, computes bounding box of watermark text string, applies rotation angle (45° diagonal or 0° horizontal), sets opacity transparency (0.1 to 0.8), stamps text across every page, and saves PDF.

Primary Action Trigger: `Add Watermark (#goBtn)`

4. Practical Example Use Case

Stamping "CONFIDENTIAL", "DRAFT", or company copyright text across sensitive business documents before circulation.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Use low opacity when the underlying content must remain readable. - Test on a copy before applying a mark to an important source.	⚠ Common Mistakes - Using text that covers critical content. - Forgetting that the mark is applied to every page.
---	--

6. Troubleshooting & Common Errors

Setting opacity too high (1.0) obscuring text underneath. Solution: Use 0.15 to 0.3 for subtle background watermark. If the text is hard to see, increase opacity or choose a contrasting color. If the output fails, retry with a readable unencrypted PDF.

7. Privacy & Data Security

100% Client-Side. Zero server uploads.

8. Tool FAQ

Q: Can I watermark only selected pages?

A: The current tool applies it to every page.

Q: Does PDFHub add developer branding as a watermark?

A: No. User-entered watermark text is the only watermark feature.

33. Word to PDF

Convert DOCX text and basic formatting into a browser print layout.

PRODUCTION ROUTE <code>/tools/word-to-pdf.html</code>	UNDERLYING ENGINE <code>mammoth.js (DOCX to HTML) + Browser Print Engine (window.print)</code>
SUPPORTED INPUTS Microsoft Word document (.docx)	SUPPORTED OUTPUTS Formatted print view triggering browser "Save as PDF" destination
MEMORY BOUND Client RAM bound (~50 MB DOCX)	PRIVACY STATUS 100% Client-Side Local Memory

1. What the Tool Does

Word to PDF converts DOCX content to HTML with Mammoth, applies page and font settings, and opens the browser print dialog for PDF output.

2. Step-by-Step Instructions

- Step 1:** Select a DOCX file.
- Step 2:** Choose page size and font size.
- Step 3:** Select Convert to PDF and wait for the print-ready page.
- Step 4:** Choose Save as PDF in the print dialog and inspect the result.

3. Execution Pipeline (What Happens on Execute)

Extracts DOCX XML and styles using Mammoth, generates semantic HTML with embedded styles for selected page size (A4, Letter) and font size, launches sandboxed print window, and triggers native browser PDF printer.

Primary Action Trigger: `Convert to PDF (#goBtn)`

4. Practical Example Use Case

Converting resumes, academic essays, or business proposals created in Word into non-editable PDF documents.

5. Best Practices & Mistakes to Avoid

✓ Practical Tips - Review headings, tables, and images after conversion. - Use the source Word file for complex layout adjustments.	⚠ Common Mistakes - Expecting exact page breaks from Word. - Closing the print dialog instead of saving.
---	--

6. Troubleshooting & Common Errors

Uploading legacy binary .doc files (Word 97-2003). Solution: Open and save as .docx in Word or Google Docs before converting.

Complex shapes or WordArt not rendering identically. Solution: Verify layout in the print preview before saving.

If the file will not load, confirm it is a valid DOCX rather than legacy DOC.

If layout changes, simplify the source or export directly from Word.

7. Privacy & Data Security

100% Client-Side. Mammoth runs entirely in client JavaScript.

8. Tool FAQ

Q: Does it accept .doc files?

A: The interface accepts DOCX.

Q: Where does conversion happen?

A: DOCX parsing happens in the browser; printing is handled by the browser.

Part VI: Developer Setup, Build Pipelines & QA

6.1 Local Development Environment

PDFHub requires Node.js (version 18 or higher) and uses standard npm scripts for asset validation, SEO audits, and production distribution bundling.

Command	Description
<code>npm run build</code>	Compiles clean production distribution bundle into <code>dist/</code> directory.
<code>npm run validate:assets</code>	Validates all deployment assets, PWA manifest, service worker, 33 tools, and 33 guides.
<code>npm run audit:seo</code>	Runs automated SEO audit across all 73 canonical public routes (1,310 checks).
<code>npm run build:docs</code>	Re-generates all 33 tool markdown docs, core documents, and web portal.
<code>npm run test:docs</code>	Executes Playwright automated runtime verification on the documentation portal.

6.2 Netlify & Cloudflare Production Pipelines

- Netlify Configuration:** Configured via `netlify.toml` with `publish = "dist"` and `command = "node scripts/build-dist.mjs"` . Exact cache-control headers are enforced for `sw.js` (no-cache) and `manifest.webmanifest` .
- Cloudflare Workers:** Configured via `wrangler.json` with `assets.directory = "dist"` . All production assets are strictly checked to ensure zero files exceed the Cloudflare 25 MiB asset limit.
- Canonical Domain:** All sitemaps, Open Graph metadata, canonical links, and structured data use `https://pdfshub.netlify.app/` as primary origin.

QA Signoff: PDFHub has undergone comprehensive end-to-end testing across 73 public routes, 33 PDF utilities, PWA installation flows, and visual lightbox guides with 100% test pass rates and zero console errors.