\ THE CODER'S KNIFE

Code & Development Toolkit

User Manual & Onboarding Guide

Field: Code & Development · Knife 2 of 50 · \ Offline.Ltd

8	3	100%	0
TOOLS	MODES (B64)	OFFLINE	DEPENDENCIES

This manual covers every tool, input field, output format, and keyboard shortcut in The Coder's Knife. It serves as both an onboarding guide for new users and a complete reference for experienced developers. The same content is available as a Markdown file for upload to any AI assistant.

Version 2.0 · For use with The_Coders_Knife_v2.html

Table of Contents

1 Getting Started

The Coder's Knife is a single HTML file containing eight professional coding utilities. It runs entirely in your browser with no external resources, no internet connection, and no installation. Open the file, start working.

System Requirements

Browser	Status	Notes
Chrome 90+	Recommended	Full support, fastest hashing
Firefox 88+	Full support	All features work
Safari 15+	Full support	macOS and iOS
Edge 90+	Full support	Chromium-based
Any modern browser	Works	Any device from the last 10 years

How to Open the File

Double-click `The_Coders_Knife_v2.html` in your file manager. It opens in your default browser. No server, no build step, no configuration. The file is self-contained: HTML, CSS, and JavaScript are all inside.

PRO TIP

Email the HTML file to your webmail account with a memorable subject line like "CODER KNIFE". It will always be waiting for you on any computer in the world, as long as you have access to your email. The knife travels as HTML. Your work travels as JSON.

The Welcome Screen

The first time you open the knife, a welcome modal appears with a quick overview. It only shows once. After that, you go straight to the first tool. You can always access the same information from the **? Help** tab.

2 Interface Overview

The interface is divided into four zones: header, tab bar, content area, and disclaimer bar.

The Header

The header shows the red backslash (\) brand mark on the left, followed by the product name. On the right, the current version number and the Offline.Ltd label appear. The left border is accented in the knife's field colour (steel grey).

The Tab Bar

The tab bar contains one button per tool, in order: Hash Generator, JSON Formatter, Regex Tester, Base Converter, Diff Viewer, JWT Inspector, Base64, and Timestamp. Two system tabs appear at the right end: ■ **Files** (with an unsaved-changes indicator dot) and **? Help**.

The Content Area

Each tool has its own panel that appears when its tab is selected. Every panel includes a title, a divider, input fields, action buttons, and output fields. Most panels also have a "? Help" link that jumps to the relevant help section.

Tooltips

Fields marked with an ■ icon have tooltips. Hover (or tap on mobile) to see a brief explanation of what the field expects and how it works.

The Disclaimer Bar

At the bottom of every page, a disclaimer bar reminds you that by using these tools you assume responsibility for your own professional use, and that Offline.Ltd's liability shall not exceed the cost of the software. A link to the full disclaimer is provided.

3 Saving & File Management

Your work is saved automatically in your browser's local storage every time you make a change. To carry your work to another device or browser, use the file export/import system.

Auto-Save

Every keystroke and selection triggers an auto-save to local storage. If you close the browser and reopen the file, your last state is restored. The amber dot on the ■ Files tab indicates unsaved changes that have not yet been exported.

Exporting Your Full State

1. Click the ■ **Files** tab. 2. Click **Export All** ↓. 3. A JSON file named `coder.knife.state_YYYY-MM-DD.json` downloads to your default folder. 4. Store this file safely — it contains the complete state of every tool.

Importing a State File

1. Click the ■ **Files** tab. 2. Click **Import All** ↑ and select your JSON state file. 3. The knife validates the file, then restores all tool states.

> **CAUTION**
>
> Importing a state file overwrites all current tool data. Export first if you have unsaved work.

Per-Tool Export & Import

Each tool has its own Export and Import buttons on the Files tab. This is useful for sharing one tool's data without affecting others, or for restoring just one tool from a backup.

Clearing All Data

The **Clear All Data** button on the Files tab erases everything from local storage. A confirmation dialog protects against accidental deletion. This action cannot be undone — export first.

> **PRO TIP**
>
> Name your exported state files by project or context: `coder.knife.state_projectX.json`. This makes it easy to switch between projects by importing the right file.

4 Hash Generator

Tab: Hash Generator · Shortcut: Ctrl + 1

Converts any text input into a fixed-length cryptographic fingerprint. The same input always produces the same hash. Even one changed character produces a completely different result. Useful for verifying data integrity, generating unique identifiers, and checking files after transfer.

Inputs

Field	Description
Algorithm	Select SHA-256 (recommended), SHA-512 (maximum strength), or SHA-1 (legacy, avoid for security).
Input	Any text to hash. Paste or type freely.

Outputs

Field	Description
Output	The hex-encoded hash string. Click Copy to copy to clipboard.

Algorithm Reference

Algorithm	Bits	Speed	Status
SHA-256	256	Fast	Secure
SHA-512	512	Moderate	Secure
SHA-1	160	Fast	Legacy — avoid for security

PRO TIP

Hash your exported JSON state files after important work. Store the hash somewhere safe. If you ever need to verify the file is untouched, run it through again and compare.

5 JSON Formatter

Tab: JSON Formatter · Shortcut: Ctrl + 2

Paste any JSON — minified, malformed, or hard to read — and the formatter validates it, reports errors with line numbers, and pretty-prints it with proper indentation. Also supports minification for production use.

Inputs

Field	Description
Input JSON	Raw or minified JSON to validate and format.

Actions

Button	Description
Format & Validate	Pretty-prints with 2-space indentation, reports errors.
Minify	Removes all whitespace for production use.
Clear	Resets input and output.

Use Cases

Reading API responses, debugging config files, preparing JSON for documentation, and minifying payloads for production deployment.

6 Regex Tester

Tab: Regex Tester · Shortcut: Ctrl + 3

Write, test, and understand regular expressions in real time. Enter a pattern and a test string; all matches are found and displayed instantly as you type.

Inputs

Field	Description
Pattern	A regular expression without surrounding slashes.
Flags	g (global), gi (global + case-insensitive), gm (global + multiline), i (case-insensitive), or none.
Test String	The text to match the pattern against.

Quick Reference

Pattern	Meaning	Example
.	Any character	a.c matches "abc"
*	Zero or more	ab* matches "a", "abb"
+	One or more	ab+ matches "ab", "abb"
?	Zero or one	ab? matches "a", "ab"
^	Start of line	^Hello
$	End of line	world$
[abc]	Character class	[aeiou]
\d	Digit [0-9]	\d{4}
\w	Word char	\w+
\s	Whitespace	\s+
(x|y)	Alternation	(cat|dog)
{n,m}	Repetition range	\d{2,4}

7 Base Converter

Tab: Base Converter · Shortcut: Ctrl + 4

Enter a number in any supported base and see it converted to all others in real time. Supports decimal (base 10), binary (base 2), octal (base 8), and hexadecimal (base 16).

Inputs

Field	Description
Input Value	A number in the selected base.
Input Base	Decimal (10), Binary (2), Octal (8), or Hexadecimal (16).

Outputs

Field	Description
Decimal	Base-10 representation.
Binary	Base-2 representation.
Octal	Base-8 representation.
Hexadecimal	Base-16 representation (uppercase).

Use Cases

Working with file permissions (octal), memory addresses (hex), bitwise operations (binary), and colour codes (hex to decimal). Each output has a Copy button.

8 Diff Viewer

Tab: Diff Viewer · Shortcut: Ctrl + 5

Paste original and modified text side by side. Click Compare and the viewer highlights every added and removed line with colour coding. Ideal for spotting changes in code, configs, or documents.

Inputs

Field	Description
Original	The baseline text (left pane).
Modified	The changed text (right pane).

Outputs

Field	Description
Differences	Colour-coded diff output: green for additions, red for removals.

Use Cases

Comparing code revisions, reviewing document edits, spotting accidental changes in configuration files, and verifying that a find-and-replace only changed what you expected.

9 JWT Inspector

Tab: JWT Inspector · Shortcut: Ctrl + 6

Paste any JSON Web Token and instantly see its header, payload, and signature decoded into readable JSON. Standard claims like exp, iat, sub, and iss are highlighted with human-readable labels and live expiry countdowns. Everything stays in your browser.

Inputs

Field	Description
Token	A JWT string, with or without the "Bearer " prefix (stripped automatically).

Outputs

Field	Description
Structure	Colour-coded view of the three JWT segments (header.payload.signature).
Header (JOSE)	Decoded JSON header showing algorithm and token type.
Payload (Claims)	Decoded JSON payload with all claims.
Standard Claims	Human-readable table of recognised claims (sub, iss, exp, iat, etc.) with expiry info.
Signature	Raw signature string. Cannot be verified offline without the signing key.

CAUTION

JWTs are decoded here, not verified. Signature verification requires the signing secret or public key, which this tool intentionally does not ask for. Use your backend for verification.

PRO TIP

Paste a token with or without the "Bearer " prefix — the inspector strips it automatically. Use "Copy as curl header" to get a ready-to-paste Authorization header.

10 Base64 Encoder / Decoder

Tab: Base64 · Shortcut: Ctrl + 7

Three modes in one tool: encode text to Base64, decode Base64 back to text, and convert any file to a Base64 string or data URL ready for embedding in HTML and CSS.

Modes

Mode	Description
Text → Base64	Encodes plain text into a Base64 string.
Base64 → Text	Decodes a Base64 string back to plain text. Also offers a download button for binary content.
File → Base64 / Data URL	Drag-and-drop or browse for any file. Produces raw Base64 and a complete data: URL with correct MIME type. Images get a live preview.

Use Cases

Embedding small images as data URIs in HTML or CSS, encoding API payloads or credentials, decoding Base64 strings from API responses, and converting favicons, SVGs, or small assets for inline use.

PRO TIP

The Data URL output is ready to paste directly into an <img src="..."> tag or a CSS url(...) value. No escaping needed.

11 Timestamp & Date Converter

Tab: Timestamp · Shortcut: Ctrl + 8

Enter any timestamp or date string and see it converted into every common format simultaneously: Unix seconds, milliseconds, ISO 8601 (UTC and local), human-readable, relative time, and day of week. Click any value to copy it instantly.

Inputs

Field	Description
Input	Unix timestamp (seconds or milliseconds), ISO 8601, RFC 2822, or any format the browser understands. Press NOW to snapshot the current time.

Output Formats

Format	Description
UNIX SECONDS	Integer seconds since epoch (10 digits).
UNIX MILLISECONDS	Integer milliseconds since epoch (13 digits).
ISO 8601 (UTC)	Standard UTC format: YYYY-MM-DDTHH:MM:SSZ.
ISO 8601 (LOCAL)	Same format in local timezone.
HUMAN READABLE (UTC)	Full date and time in UTC.
HUMAN READABLE (LOCAL)	Full date and time in local timezone.
RELATIVE	Time ago or time until (e.g. "3 hours ago").
DAY OF WEEK	Monday, Tuesday, etc.

Date Arithmetic

Add or subtract minutes, hours, days, weeks, or months from the current input. The result updates in all formats simultaneously.

Code Snippets

Ready-to-paste timestamp code in JavaScript, Python, Go, Ruby, and PHP. Click the value to copy.

PRO TIP

Click any format value to copy it. Click NOW, then immediately grab the code snippet you need — it takes two seconds to get a timestamp in any language.

12 Keyboard Shortcuts

All shortcuts use Ctrl on Windows/Linux and Cmd on macOS.

Global

`Ctrl + S`	Export full knife state
`Ctrl + H`	Open Help tab
`Escape`	Close any modal

Tool Switching

`Ctrl + 1`	Hash Generator
`Ctrl + 2`	JSON Formatter
`Ctrl + 3`	Regex Tester
`Ctrl + 4`	Base Converter
`Ctrl + 5`	Diff Viewer
`Ctrl + 6`	JWT Inspector
`Ctrl + 7`	Base64
`Ctrl + 8`	Timestamp

13 Troubleshooting

JavaScript Is Disabled

The knife requires JavaScript. If you see a blank page, check your browser settings and ensure JavaScript is enabled. No extensions or plugins are needed.

Local Storage Was Cleared

If your data disappeared, your browser's local storage may have been cleared (by a privacy tool, manual action, or browser update). This is why regular exports are important. Import your last state file from the Files tab to restore everything.

Hash Output Looks Wrong

Hashes are case-sensitive and encode-sensitive. Trailing whitespace, different line endings, or encoding differences will produce different hashes. Ensure your input matches exactly.

JSON Reports an Error But Looks Correct

Common causes: trailing commas (not valid JSON), single quotes instead of double quotes, unquoted keys, or comments (JSON does not support comments). The error message includes a position indicator to help locate the problem.

Regex Matches Nothing

Check your flags. Without the "g" flag, only the first match is found. Without "m", ^ and $ match the start/end of the entire string, not individual lines.

Import Shows "Wrong Knife"

The state file's "knife" field must be exactly "coder". State files from other knives are not compatible. The error message will show which knife the file belongs to.

Tabs Overflow on Mobile

On narrow screens, the tab bar scrolls horizontally. Swipe left and right to access all tabs. The system tabs (Files, Help) are always at the right end.

14 Accuracy & Limitations

Tool	Accuracy	Notes
Hash Generator	Exact	Uses the browser's native Web Crypto API. Output matches openssl and sha256sum.
JSON Formatter	Exact	Uses native JSON.parse/stringify. Follows ECMA-404 strictly.
Regex Tester	Exact	Uses the browser's native RegExp engine. Behaviour matches JavaScript regex.
Base Converter	Exact	Limited to JavaScript's safe integer range (up to 2■³ – 1 for integers).
Diff Viewer	Line-level	Compares line by line. Does not do word-level or character-level diffing.
JWT Inspector	Decodes only	Decodes header and payload. Cannot verify signatures without the signing key.
Base64	Exact	Uses native btoa/atob. File mode uses FileReader API.
Timestamp	Millisecond	Depends on browser's Date implementation. Handles seconds and millisecond inputs.

CAUTION

These tools are provided as utilities and reference aids. They are not certified for safety-critical systems, medical diagnosis, legal proceedings, financial auditing, or any context where errors could cause harm. Cryptographic tools are for educational and utility purposes and are not a substitute for professional security review. Offline.Ltd's total liability shall not exceed the purchase price paid for this software.

15 Glossary

Base64	An encoding scheme that represents binary data as ASCII text using 64 printable characters. Used to embed files in HTML, encode credentials, and transmit binary over text-only channels.
Claim (JWT)	A key-value pair in a JWT payload. Standard claims include sub (subject), iss (issuer), exp (expiration), and iat (issued at).
Data URL	A URI scheme (data:) that embeds file content directly in a document. Format: data:[mediatype];base64,[data].
Diff	A comparison showing the differences between two texts. Added lines are typically shown in green, removed lines in red.
Epoch	The Unix epoch: January 1, 1970, 00:00:00 UTC. Unix timestamps count seconds or milliseconds from this moment.
Flag (Regex)	A modifier that changes how a regex pattern is applied. Common flags: g (global), i (case-insensitive), m (multiline).
Hash	A fixed-length string produced by a cryptographic function. Deterministic: the same input always produces the same output. Any change to the input produces a completely different hash.
Hex / Hexadecimal	Base-16 number system using digits 0–9 and letters A–F. Used for memory addresses, colour codes, and compact binary representation.
ISO 8601	International standard for date and time representation. Example: 2025-06-15T14:30:00Z (the Z indicates UTC).
JSON	JavaScript Object Notation. A lightweight data-interchange format using key-value pairs, arrays, strings, numbers, booleans, and null.
JOSE	JSON Object Signing and Encryption. The family of standards that includes JWT. The JWT header is a JOSE header.
JWT	JSON Web Token. A compact, URL-safe token format with three Base64URL-encoded segments: header, payload, and signature.
Local Storage	A browser API that stores key-value pairs persistently on the user's device. Data survives page reloads but not browser data clearance.
MIME Type	A label identifying a file's content type (e.g. image/png, application/json). Used in data URLs and HTTP headers.
Minify	Remove all unnecessary whitespace, formatting, and comments from code or data to reduce file size.

Octal	Base-8 number system using digits 0–7. Commonly used for Unix file permissions (e.g. 755).
Regex	Regular Expression. A pattern language for matching text. Used for search, validation, and text processing.
SHA-256	Secure Hash Algorithm producing a 256-bit (32-byte) hash. Part of the SHA-2 family. Currently recommended for most use cases.
SHA-512	Secure Hash Algorithm producing a 512-bit (64-byte) hash. Maximum strength in the SHA-2 family.
State File	A JSON file containing the complete state of all tools in the knife. Used for backup, transfer between devices, and project switching.
Tofu	A rendering artifact (white rectangle) that appears when a font cannot render a character. Named after the shape of the replacement glyph.
Unix Timestamp	The number of seconds (or milliseconds) since the Unix epoch (January 1, 1970 UTC). A universal way to represent a moment in time.
UTF-8	A character encoding capable of representing all Unicode characters. The dominant encoding on the web.

\ THE CODER'S KNIFE

Knife 2 of 50 · Code & Development

Ship it.

Version 2.0 · Offline.Ltd