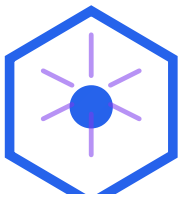


WebWeaveX

UNIVERSAL RUNTIME COGNITION INFRASTRUCTURE

Deterministic runtime cognition for humans and AI agents. Understand, continue, reconstruct, replay, and reason about authenticated operational software systems — with bit-for-bit identical results across five language SDKs.

Technical Whitepaper & Complete Reference

Version 3.0.0 · Python SDK 3.0.1

Author: Piyush Mishra · ni_sh_a.char

github.com/ni-sh-a-char/WebWeaveX

Registry availability verified 17 August 2026

Contents

1. Executive Summary

WebWeaveX is deterministic runtime cognition infrastructure. It captures what a running software system *is doing* — its DOM event surfaces, network envelopes, state machines and execution history — and turns that into a canonical, graph-structured model with a stable cryptographic identity.

The core claim is narrow and testable: **the same input produces the same bytes, and therefore the same SHA-256 digest, in every supported language**. That property is what makes the rest possible. Once runtime state has a stable identity you can diff it, prove two runs equivalent, persist and resume it, merge histories across sessions, and hand a compact representation to an LLM instead of a half-megabyte of HTML.

What ships today

SDK	DISTRIBUTION	VERSION	STATUS
Python	PyPI — `webweavex`	3.0.1	PUBLISHED
JavaScript / TypeScript	npm — `webweavex`	3.0.0	PUBLISHED
Dart / Flutter	pub.dev — `webweavex`	3.0.0	PUBLISHED
Java	Maven Central — `io.github.piyush-mishra-00:webweavex`	3.0.0	PUBLISHED
Kotlin	Prebuilt JAR in `kotlin` branch	3.0.0	DIRECT JAR

Who it is for

Human engineers

Inspect complex authenticated web apps, preserve login workflows across runs, audit runtime behaviour between releases, and build integration tooling against a stable model instead of brittle selectors.

AI agents

Hold long-running session continuity without re-authenticating, reason over compact IR graphs that fit a context window, and verify every tool call against a deterministic hash rather than trusting a screenshot.

2. The Problem: Why Runtime State Is Hard

Modern software is dynamic, stateful, authenticated and distributed. The tools most teams reach for were designed for none of those properties.

CHALLENGE	TRADITIONAL SCRAPERS & LLM WRAPPERS	WEBWEAVEX
Surface capture	Returns a static HTML string; structure is implicit and re-derived every time	Captures a layered runtime graph with stable node identities
Auth continuity	Session collapses the moment the process exits	Persists authenticated session state as an encrypted, resumable envelope
Operational context	No memory of prior state; every run starts cold	Tick-indexed memory fabric merges histories across runs
Replay verification	Breaks on hashed CSS class names and generated IDs	Proves equivalence over normalized graph hashes
Reconstruction	Requires hand-written mocks and fixtures	Rebuilds topology from a unified intermediate representation
Determinism	Output varies run to run; diffs are noise	Byte-stable canonical serialization, SHA-256 identity
AI integration	Overflows the context window with raw markup	Compact structured IR sized for prompting

The determinism problem, concretely

Two runs of the same page rarely produce identical bytes. Object key order varies by language and runtime. Floating-point values serialize differently in Python, JavaScript and the JVM. Timestamps, nonces, session IDs, memory addresses and generated class names all leak into output. Whitespace differs by parser.

Any one of these makes a hash useless. WebWeaveX addresses them as a single normalization layer applied before any hashing or encryption occurs — and then holds every SDK to the same layer.

3. What WebWeaveX Is

WebWeaveX sits between raw operational software (browsers, SPAs, desktop apps, microservices, repositories) and downstream consumers (engineering tools, auditing suites, AI agents).

CONCEPT	OPERATIONAL MEANING
Runtime cognition infrastructure	Captures live behaviour — graphs, events, execution state — rather than transient markup snapshots.
Operational runtime substrate	Provides stable node identities, structural fingerprints, and tick-indexed execution history for a live session.
Authenticated session continuation	Resumes an authenticated session from a user-authorized token, cookie jar or credential envelope.
Deterministic extraction engine	Standardizes DOM trees, network envelopes and state payloads into canonical UTF-8 JSON before hashing or encryption.
Replay & reconstruction	Proves topological equivalence between two runs, and rebuilds state from the intermediate representation.
Federated memory fabric	Merges multi-turn execution histories into a deterministic key-value / graph memory layer.
Cross-language SDK parity	A shared mathematical spec so Python, JavaScript, Dart, Java and Kotlin compute identical digests.

> **What it is not.** WebWeaveX is not a scraper, not a browser-automation framework, and not an LLM wrapper. It does not decide *what* to extract or *why*. It gives whatever does make those decisions a stable, verifiable substrate to work against.

4. Core Concepts & Architecture

The canonical pipeline

Every input, regardless of source type, flows through the same six phases.

```
Universal Input  (web | repository | native | connector | document)
      |
      v
[1] INGEST       source-type router selects the extraction engine
      |
      v
[2] NORMALIZE    volatile keys stripped, keys sorted, floats
                 canonicalized, whitespace collapsed, UTF-8
      |
      v
[3] COGNIZE      semantic / causal / workflow engines annotate
                 the raw capture with operational meaning
      |
      v
[4] GRAPH        runtime graph built: stable node ids + edges
      |
      v
[5] HASH         SHA-256 over canonical bytes -> pipeline_hash
      |
      v
[6] ENCRYPT      Kaalka v5 envelope for session state at rest
      |
      v
Pipeline Output  graph + pipeline_hash + encrypted_session
```

Subsystems

SUBSYSTEM	RESPONSIBILITY
Extraction engines	Web/SPA, repository, native runtime, multimodal, and document extraction. Each produces the same IR shape.
Determinism layer	Canonical JSON, normalization, stable serialization, Python-compatible float representation. The foundation everything else rests on.
Runtime graph	Builds and queries the node/edge model; computes graph fingerprints and lineage.
Replay engine	Validates equivalence between runs at the state, graph, memory, DOM and fingerprint levels independently.
Reconstruction engine	Rebuilds runtime, graph, browser and memory state from a stored IR or envelope.
Memory fabric	Builds, merges, queries, replicates and persists runtime memory with a stable memory hash and verifiable lineage.
Adaptive layer	Selector healing, semantic anchors, modal recovery, infinite-scroll recovery, layout resilience — the parts that absorb real-world drift.
Kaalka v5 crypto	Encrypted persistence for session and memory state, portable across all five SDKs.
Workflow & synchronization	Objective planning, workflow replay, runtime deltas, synchronized multi-source runtimes.

5. The Determinism Contract

Determinism is the product. Everything else — replay, reconstruction, memory merging, cross-language parity — is a consequence of it. The contract has four layers, applied in order.

#	LAYER	GUARANTEE
1	**Normalization**	Volatile runtime keys (timestamps, nonces, addresses, generated ids) are stripped against a shared `VOLATILE_RUNTIME_KEYS` set. Whitespace is collapsed. Structure is made order-independent.
2	**Canonical JSON**	Keys sorted with a stable comparator, no insignificant whitespace, UTF-8 encoding, and Python-compatible float `repr` so `0.1 + 0.2` serializes identically on every runtime.
3	**Stable serialization**	A single `StableSerialize` implementation per SDK, byte-certified against the Python reference, producing the exact input to the hash function.
4	**Fingerprint**	SHA-256 over those canonical bytes. Identical inputs yield identical digests in Python, JavaScript, Dart, Java and Kotlin.

> **Why float representation matters.** JavaScript's `JSON.stringify(0.1+0.2)` and Python's `repr(0.1+0.2)` agree, but the JVM's default `Double.toString` does not. The Kotlin and Java SDKs therefore ship a dedicated `PyFloat` port rather than using platform defaults. This kind of detail is where cross-language determinism is actually won or lost.

Replay equivalence

Equivalence is checked at several independent levels, so a failure localizes to a subsystem:

- `validate_replay_equivalence` — overall run equivalence
- `validateGraphReplayEquivalence` — graph topology only
- `validateMemoryReplayEquivalence` — memory fabric only
- `validateDomReplayEquivalence` — DOM surface only
- `validateFingerprintReplayEquivalence` — digest only

6. Kaalka v5 Security Model

Kaalka v5 is the encryption contract for state at rest. Its job is to make a captured session portable and resumable without leaving credentials sitting in plaintext on disk, and to do so in a format every SDK can read.

What is protected

ARTIFACT	PROTECTION
Session state (tokens, cookies, auth envelopes)	Encrypted before persistence via `encrypt_session_state` / `save_encrypted_session`
Runtime memory	Encrypted envelope via `saveRuntimeMemory`, verifiable lineage on load
Individual values	`encrypt_value` / `decrypt_value` for field-level protection
Integrity	`compute_kaalka_hash` binds the envelope to a digest

Operating principles

- **Authorization is the caller's responsibility.** WebWeaveX continues sessions the operator already legitimately holds. It does not acquire credentials, defeat authentication, or bypass access control.
- **Encryption applies to state at rest.** Transport security remains the responsibility of the surrounding application.
- **Envelopes are portable.** A session encrypted by the Python SDK is readable by the Kotlin SDK, because the key-derivation and cipher spec are shared, not reimplemented per language.
- **Normalization precedes encryption.** Volatile fields are stripped first, so an envelope does not accidentally pin a run to a wall-clock timestamp.

> **Responsible use.** Session continuation is a powerful capability. Use it only against systems you own or are explicitly authorized to test. Persisted envelopes contain live credentials in encrypted form — treat the key material and the envelope with the same care you would treat the credentials themselves, and keep both out of version control.

Reporting vulnerabilities

Security issues should be reported privately through the process in `SECURITY.md` in the repository rather than opened as public issues.

7. Platform Availability — Verified Matrix

Every row below was checked directly against the live package registry API on 17 August 2026. This section supersedes any badge or table that disagrees with it.

SDK	REGISTRY	COORDINATES	LATEST	STATUS
Python	PyPI	`webweavex`	3.0.1	PUBLISHED
JavaScript / TS	npm	`webweavex`	3.0.0	PUBLISHED
Dart / Flutter	pub.dev	`webweavex`	3.0.0	PUBLISHED
Java	Maven Central	`io.github.piyush-mishra-00:webweavex`	3.0.0	PUBLISHED
Kotlin	GitHub (`kotlin` branch)	`webweavex-kotlin-3.0.0.jar`	3.0.0	DIRECT JAR

Published version history

REGISTRY	RELEASED VERSIONS
PyPI	0.1.0, 1.0.1, 1.0.2, 1.0.3, 2.0.0, 2.0.1, 2.1.0, **3.0.1**
npm	0.1.0, 2.0.0, 2.0.1, 2.1.0, **3.0.0**
pub.dev	0.1.0, 0.1.1, 2.1.0, **3.0.0**
Maven Central	0.1.0, **3.0.0** (groupId `io.github.piyush-mishra-00`)

> **Correction to previously published claims.** Earlier README badges advertised `io.webweavex:webweavex:3.0.0` and `io.webweavex:webweavex-kotlin:3.0.0` . Both of those coordinates are wrong — the `io.webweavex` groupId returns 404 on Maven Central.
>
> The Java SDK *is* published, under the groupId declared in the project's own `pom.xml` : `io.github.piyush-mishra-00:webweavex:3.0.0` , released 24 July 2026 with jar, sources, javadoc and GPG signatures. Note that the `search.maven.org` Solr index does not return this artifact; the authoritative check is the repository itself at `repo1.maven.org/maven2/io/github/piyush-mishra-00/webweavex/` .
>
> The Kotlin SDK is *not* published — no `webweavex-kotlin` artifact exists under any groupId on Central. It ships as a prebuilt JAR from the `kotlin` branch.
>
> Additionally, the bundled `kotlin/dist/README.md` documents `WebWeaveX.extract("https://example.com")` . No class `io.webweavex.WebWeaveX` exists in the shipped JAR; that snippet does not compile. The correct entry points are documented in section 8e.

8. Installation & Usage by Platform

8a. Python — PyPI PUBLISHED

Reference implementation. Requires Python 3.10+. Latest published version 3.0.1.

```
pip install webweavex

# with browser automation
pip install "webweavex[browser]"
python -m playwright install chromium

# everything
pip install "webweavex[full]"
```

Canonical pipeline

```
from webweavex import UniversalInput, run_canonical_pipeline

spec = UniversalInput(
    source="https://example.com",
    source_type="web",
)

result = run_canonical_pipeline(spec)
print(result.pipeline_hash)   # deterministic SHA-256 digest
```

Extraction

```
from webweavex import (
    extract, extract_async, extract_repo, extract_docs,
    extract_web, extract_repository, extract_multimodal,
)

doc   = extract("https://example.com")
repo  = extract_repository("./my-project")   # codebase knowledge graph
media = extract_multimodal("./report.pdf")
```

Graph, fingerprint and replay

```
from webweavex import (
    build_runtime_graph, query_runtime_graph,
    compute_global_runtime_fingerprint,
    validate_replay_equivalence,
)

graph = build_runtime_graph([ir_a, ir_b])
nodes = query_runtime_graph(graph, {"type": "element"})
fp    = compute_global_runtime_fingerprint(graph)

assert validate_replay_equivalence(run_one, run_two)
```

Session persistence with Kaalka v5

```
from webweavex import (
    encrypt_session_state, decrypt_session_state,
    save_encrypted_session, load_encrypted_session,
)

blob = encrypt_session_state({"auth_token": "..."})
save_encrypted_session("session.enc", blob)

restored = decrypt_session_state(load_encrypted_session("session.enc"))
```

Optional extras

EXTRA	PULLS IN
`browser`	playwright ≥ 1.40
`parsers`	tree-sitter-languages
`ocr`	pytesseract, Pillow
`ingestion`	python-docx, pytesseract, Pillow
`llm`	groq
`full`	all of the above
`dev`	pytest, pytest-cov, build

Base dependencies, always installed: `requests`, `httpx`, `beautifulsoup4`, `lxml`, `markdownify`, `pypdf`. The top-level package re-exports **173 names**; import from `webweavex` only — `core.*` is internal.

8b. JavaScript / TypeScript — npm PUBLISHED

Dual ESM + CommonJS build with bundled type declarations. Requires Node.js 18+.

```
npm install webweavex     # or: pnpm add webweavex / yarn add webweavex
```

Canonical pipeline

```
import { UniversalInput, runCanonicalPipeline, VERSION } from "webweavex";

const input = new UniversalInput({
  source: "https://example.com",
  sourceType: "web",
});

const result = await runCanonicalPipeline(input);
console.log(VERSION, result.pipelineHash);
```

Authenticated session continuity

```
import {
  createRuntimeSession, persistRuntimeSession,
  restoreRuntimeSession, extractWithSession,
} from "webweavex";

const session = await createRuntimeSession({ source: "https://app.example.com" });
await persistRuntimeSession(session, "session.enc");

// later — resume without re-authenticating
const resumed = await restoreRuntimeSession("session.enc");
const page    = await extractWithSession("https://app.example.com/dashboard", resumed);
```

Determinism, replay, reconstruction

```
import {
  computeStableDomHash, computeSpaFingerprint,
  validateReplayEquivalence, reconstructRuntime,
} from "webweavex";

const domHash = computeStableDomHash(html);
const equal   = validateReplayEquivalence(runA, runB);
const rebuilt = reconstructRuntime(intermediateRepresentation);
```

Memory fabric

```
import {
  buildRuntimeMemory, mergeRuntimeMemories,
  queryRuntimeMemory, saveRuntimeMemory,
} from "webweavex";

const mem    = buildRuntimeMemory(extractions);
const merged = mergeRuntimeMemories([mem, priorMemory]);
await saveRuntimeMemory(merged, "memory.enc");
```

8c. Dart / Flutter — pub.dev PUBLISHED

Pure Dart, including a BeautifulSoup-parity HTML engine, so it runs on Flutter mobile, Dart server and Dart CLI with no native bridge.

```
dart pub add webweavex        # or: flutter pub add webweavex
```

```
import 'package:webweavex/webweavex.dart';

void main() {
  // Python-aligned canonical entry points
  final fp      = computeGlobalRuntimeFingerprint(graph);
  final ok      = validateReplayEquivalence(runA, runB);
  final rebuilt = reconstructRuntime(ir);

  // Bundled Soup engine
  final soup     = Soup(htmlString);
  final semantic = extractSemanticHtml(htmlString);

  // Adaptive layer
  final healed = healSelector(brokenSelector, dom);
  final anchor = buildSemanticAnchor(element);
}
```

8d. Java — Maven Central PUBLISHED

Published to Maven Central on 24 July 2026: jar, sources, javadoc and GPG signatures. The artifact is 374 KB with 102 classes across 32 packages.

> **Mind the groupId.** It is `io.github.piyush-mishra-00`, not `io.webweavex`. The latter appears in older documentation and returns 404. If your build cannot resolve the dependency, this is almost certainly the cause.

```
<dependency>
  <groupId>io.github.piyush-mishra-00</groupId>
  <artifactId>webweavex</artifactId>
  <version>3.0.0</version>
</dependency>
```

```
// build.gradle.kts
dependencies {
    implementation("io.github.piyush-mishra-00:webweavex:3.0.0")
}
```

Usage — the example that ships inside the artifact

```
import io.webweavex.WebWeaveX;
import io.webweavex.crypto.Hashing;
import io.webweavex.determinism.GlobalRuntimeFingerprint;
import io.webweavex.determinism.StableSerialize;
import io.webweavex.graph.RuntimeGraph;
import io.webweavex.replay.ReplayEquivalence;
import java.util.*;

public class App {
    public static void main(String[] args) {
        System.out.println("WebWeaveX Java SDK v" + WebWeaveX.VERSION);

        Map<String, Object> data = new LinkedHashMap<>();
        data.put("b", 2);
        data.put("a", 1);
        String canonical = StableSerialize.stableSerialize(data);
        String hash      = Hashing.computeDeterministicHash(data);

        String fp = GlobalRuntimeFingerprint.compute(new HashMap<>());

        Map<String, Object> graph = RuntimeGraph.normalizeRuntimeGraph(
            Map.of("nodes", List.of(Map.of("id", "1")), "edges", List.of()));

        Map<String, Object> env = Map.of("browser_ir", Map.of("runtime_identity", "test"));
        Map<String, Object> r   = ReplayEquivalence.validate(env, new LinkedHashMap<>(env));
        System.out.println("equivalent=" + r.get("equivalent"));
    }
}
```

Runtime kernel pipeline

```
import io.webweavex.kernel.RuntimeKernel;
import io.webweavex.kernel.UniversalInput;

UniversalInput input = UniversalInput.of("https://example.com")
        .sourceType("web")
        .tick(0L)
        .build();

RuntimeKernel kernel = RuntimeKernel.getRuntimeKernel("web");
Map<String, Object> result = kernel.runPipeline(input.toDict(), 0L);
```

Building from source instead

```
git clone -b java https://github.com/ni-sh-a-char/WebWeaveX.git
cd WebWeaveX/java
mvn clean install
```

The Java tree mirrors the Python engine package-for-package under `io.webweavex.*` — `adaptive`, `application`, `ast`, `auth`, `causality`, `distributed` and others. Parity against the Python golden vectors is enforced in CI by the `java-parity` and `parity-regression` workflows.

8e. Kotlin — Prebuilt JAR DIRECT JAR

A compiled JAR ships at `kotlin/dist/webweavex-kotlin-3.0.0.jar` on the `kotlin` branch: 220 KB, 76 public classes under `io.webweavex.*`.

```
mkdir -p libs
curl -L -o libs/webweavex-kotlin-3.0.0.jar \
  https://github.com/ni-sh-a-char/WebWeaveX/raw/kotlin/kotlin/dist/webweavex-kotlin-3.0.0.jar
```

```
// build.gradle.kts
dependencies {
    implementation(files("libs/webweavex-kotlin-3.0.0.jar"))
}
```

Correct usage — verified against the shipped JAR

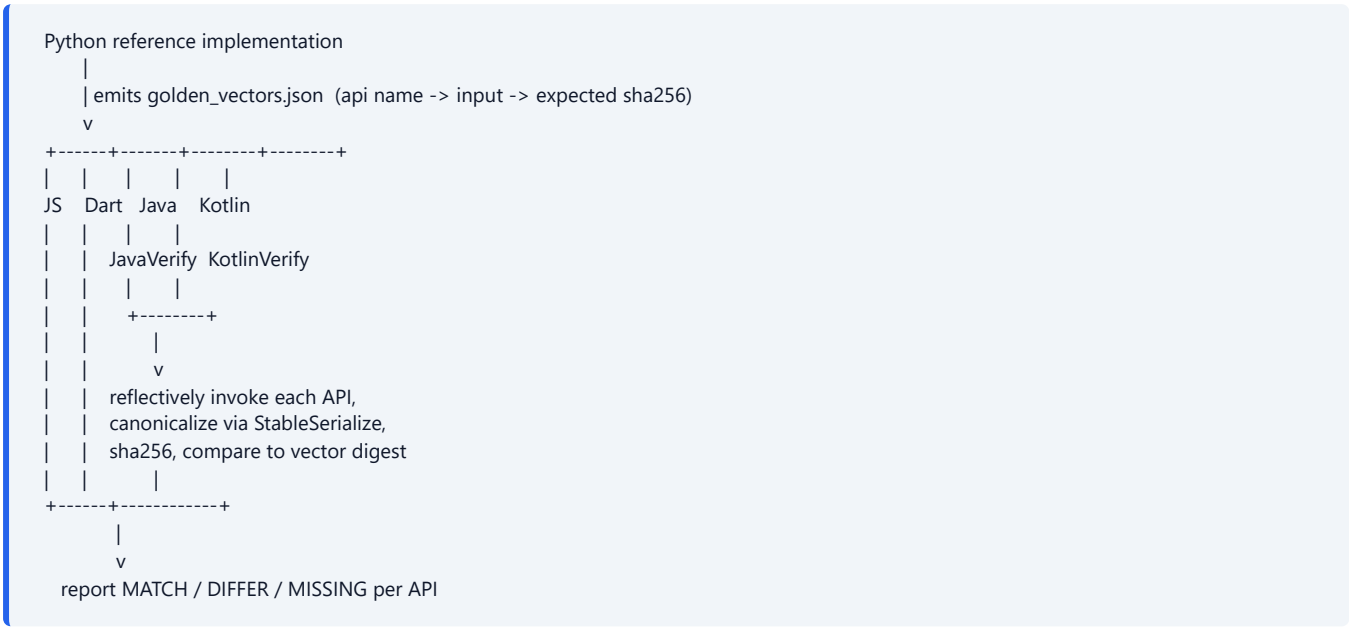
```
import io.webweavex.runtime.RuntimeKernel
import io.webweavex.runtime.UniversalInput
import io.webweavex.extract.ExtractionPipeline
import io.webweavex.fingerprint.Fingerprint

fun main() {
    val kernel = RuntimeKernel()
    val input  = UniversalInput("https://example.com")
    val output = kernel.extract(input)

    println("version:      ${kernel.version}")
    println("capabilities: ${kernel.capabilities}")
    println("fingerprint:  ${Fingerprint.compute(input.toMap())}")

    val result = ExtractionPipeline.extractText(htmlString, "html")
}
```

Shipped packages

PACKAGE	KEY TYPES
`io.webweavex.runtime`	RuntimeKernel, UniversalInput, UniversalOutput, RuntimeNode, RuntimeEdge, DeterministicClock, ReplayClock, LogicalClock
`io.webweavex.extract`	ExtractionPipeline, HtmlExtractor, JsonExtractor, MarkdownExtractor
`io.webweavex.determinism`	CanonicalJson, Normalization, StableSerialize, PyFloat, PyJson
`io.webweavex.crypto`	KaalkaV5
`io.webweavex.replay`	ReplayEngine, ReplayEquivalence, ReplaySnapshot, ReplayResult
`io.webweavex.memory`	MemoryEngine, MemoryStore, MemorySnapshot, MemoryEntry
`io.webweavex.repository`	RepositoryAnalyzer, KnowledgeGraph, QueryEngine, SearchIndex, LanguageDetector
`io.webweavex.graph`	RuntimeGraph
`io.webweavex.workflow`	WorkflowEngine, WorkflowStep, WorkflowResult
`io.webweavex.fetch`	Crawler, HttpTransport, JavaNetTransport

9. Cross-Language Parity Verification

Parity is not asserted, it is tested. Python generates golden vectors; each other SDK replays them through a reflective harness and compares digests.



The JVM harnesses resolve each API by fully-qualified class name from a shared index. Kotlin ports live at the same `io.webweavex.*` FQCN as their verified Java counterparts, so the same index file drives both.

BRANCH	CI WORKFLOWS
`python`	`ci.yml`
`javascript`	`ci` , `benchmark` , `nightly` , `provenance` , `publish` , `release` , `security`
`dart`	`dart.yml`
`java`	`java-build` , `java-parity` , `parity-regression`
`kotlin`	`ci.yml`

10. Use Cases

Agent session continuity

An autonomous agent authenticates once, persists an encrypted envelope, and resumes across process restarts — without re-running a login flow or storing raw credentials.

Regression detection on live apps

Capture a fingerprint per release. A changed digest localizes to graph, DOM, memory or state via the per-level equivalence checks, instead of a screenshot diff nobody trusts.

Context compression for LLMs

Replace a large raw HTML payload with a structured IR graph that carries the operational meaning and fits comfortably in a prompt.

Repository cognition

`extract_repository` builds a queryable knowledge graph of a codebase — language detection, symbol lookup, planned queries — for code-aware agents.

Deterministic test fixtures

Record a runtime once, reconstruct it from IR in CI. No live network, no flake, byte-identical every run.

Cross-stack verification

A Python backend and a Flutter client can independently compute the same fingerprint for the same state and compare, without shipping a shared binary.

Selector drift — a concrete example

A deploy renames a hashed CSS class and a recorded selector breaks. The adaptive layer resolves the element by semantic anchor rather than the literal selector, and records the healed mapping in adaptive memory so the next run starts from the repaired state.

```
from webweavex import heal_selector, save_adaptive_memory

healed = heal_selector(broken_selector, dom)
save_adaptive_memory("adaptive.enc", {"selector": healed})
```

11. Repository Layout & Contributing

The repository uses long-lived per-language branches. `main` carries documentation and the public site only — a clone of `main` contains *no* SDK source, which surprises most first-time contributors.

BRANCH	CONTENTS	BUILD ENTRY POINT
`main`	Docs portal, README, public site	—
`python`	`webweavex/` package + `core/` engines	`pyproject.toml`
`javascript`	`src/` TypeScript sources	`package.json` (tsup)
`dart`	`lib/webweavex.dart` + `lib/src/`	`pubspec.yaml`
`java`	`java/src/main/java/io/webweavex/`	`java/pom.xml`
`kotlin`	`kotlin/src/` + `kotlin/dist/*.jar`	`kotlin/build.gradle.kts`

```
# work on a specific SDK
git clone -b python https://github.com/ni-sh-a-char/WebWeaveX.git
```

Contribution rules that actually matter here

- **Never break determinism.** A change that alters canonical bytes changes every published fingerprint. If output must change, it is a major version.
- **Parity is a cross-branch concern.** Adding an API to one SDK without porting and vector-certifying it in the others creates silent divergence.
- **No wall-clock or randomness in canonical paths.** Use the injected clock providers (`DeterministicClock`, `ReplayClock`) so replay stays reproducible.
- Standard project documents apply: `CONTRIBUTING.md`, `CODE_OF_CONDUCT.md`, `SECURITY.md`, `ROADMAP.md`.

12. Known Gaps & Roadmap

An honest list of what is not done. These are the highest-value contribution targets.

GAP	IMPACT	SUGGESTED RESOLUTION
Kotlin is not on Maven Central	Kotlin adoption requires a manual JAR download, unlike the other four SDKs	Publish `webweavex-kotlin` via Sonatype under `io.github.piyush-mishra-00` — the same path already used successfully for the Java artifact
README badges used the wrong Maven groupId (`io.webweavex`)	Users hit unresolvable dependencies for an artifact that does exist — under a different coordinate	Corrected to `io.github.piyush-mishra-00` ; use live shields.io Maven queries so the badge tracks reality
`kotlin/dist/README.md` documents a non-existent `WebWeaveX` class	Copy-pasted quick-start does not compile	Update to `RuntimeKernel` / `ExtractionPipeline` (corrected in section 8e)
Python is 3.0.1 while other SDKs are 3.0.0	Minor version-reporting confusion; determinism is unaffected	Align patch versions at the next coordinated release
Published benchmark figures are labelled v2.0.0	Performance claims lag the current release	Re-run the benchmark suite against 3.0.x and republish
Coverage percentages are asserted in docs, not emitted by CI	Unverifiable quality claim	Wire a coverage reporter and use its live badge
No repository description or topics on GitHub	Poor discoverability in GitHub search	Set the description and add topics (deterministic, ai-agents, replay, runtime, web-extraction)

Direction

- Maven Central publication for the Kotlin SDK (Java is already published)
- Live, registry-backed badges across all documentation
- Refreshed 3.0.x benchmark and coverage reporting
- Expanded connector fabric and distributed extraction coverage

13. License, Citation & Contact

License

WebWeaveX is released under the **Apache License 2.0**. You may use, modify and distribute it, including commercially, provided you retain the license and notice files and state significant changes. The license includes an express patent grant.

Citation

The repository ships a `CITATION.cff`. For academic use:

```
Mishra, Piyush. WebWeaveX: Universal Runtime Cognition Infrastructure.
Version 3.0.0, 2026. https://github.com/ni-sh-a-char/WebWeaveX
```

Links

Documentation	https://ni-sh-a-char.github.io/WebWeaveX/
Repository	https://github.com/ni-sh-a-char/WebWeaveX
PyPI	https://pypi.org/project/webweavex/
npm	https://www.npmjs.com/package/webweavex
pub.dev	https://pub.dev/packages/webweavex
Maven Central	https://central.sonatype.com/artifact/io.github.piyush-mishra-00/webweavex
Issues	https://github.com/ni-sh-a-char/WebWeaveX/issues
Security	See `SECURITY.md` — report privately, not as a public issue
Support the project	https://buymeacoffee.com/piyushmishra00