The Stateless Myth vs. The Stateful Reality

The popular mental model assumes a straight line. Real coding agents navigate a path-dependent query loop.

The Straight Line Myth

1. Build system prompt
2. Give tool list
3. Execute & Return Answer

The Maze / Reality

- Prepare & shrink context
- Stream partial outputs
- Update internal state
- Retry, compact, or recover

Silent Error Recovery

Errors are intentionally withheld from the user. The runtime optimizes for keeping the agent alive rather than failing at the first hurdle.

- **Context Collapse:** Micro-compacting the transcript.
- **Reactive Compaction:** Shrinking memory on the fly.
- **Truncation Retry:** Handling max-output-token failures silently.

Tombstoning the Poisoned Context

If a streaming path falls back, the runtime must explicitly emit a tombstone to invalidate the orphaned partial message. A half-streamed assistant message is not harmless.

Poisoned Transcripts
Confuses the model's memory.

Invalid Tool-Use
Breaks schema structures mid-execution.

Broken API History
Causes fatal errors in future LLM calls.

The Burden of Mutable State

Stateless APIs are elegant. Long-running coding work is not.
The runtime absorbs the mechanical infrastructure problems.

The Model's Job

- Choosing actions
- Synthesizing info
- Generating code

The Runtime's Job

- Compressing history
- Mediating approvals
- Tracking pending tools
- Managing memory state

The Prompt is Just a Mask

Most real agent bugs are runtime bugs wearing prompt-shaped masks. When your agent feels unreliable, look at the loop.

Check carried state:
Are internal turns remembered?

Check recovery classifications:
Are failures caught or crashed?

Check retry logic: Are partial outputs polluting the transcript?