© 2026 Yiguedan Philippe Andy Yaro



mcp-keynote: One Spec In, a Keynote Deck Out

Yiguedan Philippe Andy Yaro

View GitHub Repository: github.com/andyyaro/mcp-keynote

Let's connect! Email: yyaro@gmu.edu

LinkedIn: linkedin.com/in/andy-yaro

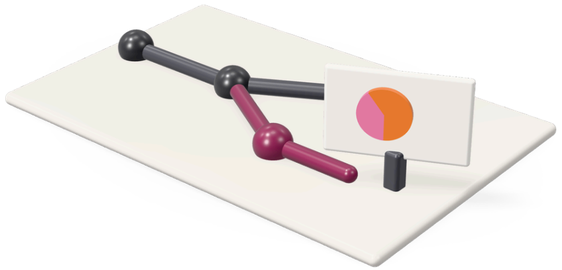


mcp-keynote lets an AI build and edit Apple Keynote decks on a Mac. It is a server for MCP, the Model Context Protocol AI apps use to call outside tools, and it drives Keynote through AppleScript, Apple's scripting language: 58 tools, or 61 with a photo key.

It rebuilds ByAxe's fork of easychen's original. On 25–26 July 2026, with Claude Code, Anthropic's coding agent, I made all text travel as arguments, added whole decks from one spec (a structured deck description), and added 757 unit tests and a new continuous integration (CI) setup: 48 commits.

On 3 November 2026 this deck reruns the tests, the injection check (can text turn into code?) before and after my changes, and the spec checks. Keynote cannot run here, so live results are the repo's record. I am Yiguedan Philippe Andy Yaro.

A. Why It Exists



Keynote can be scripted, so an AI can drive it. The server I started from built each script by pasting the AI's text into it.

Building a deck one element at a time also takes many calls: 81 for a 20-slide deck in the repo's measurement. One call to build_deck, the whole-deck tool, does the same deck.

I rebuilt it over 25–26 July, in under 18 hours, one pull request (a proposed change) per stage: safety and tests, then whole decks, then fidelity to the design, then stricter checks.

1. Where It Started

easychen/keynote-mcp

July 2025, 2 commits.

The original server, with its messages in Chinese.

ByAxe/keynote-mcp

March 2026, 15 commits.

English text, a Python package, a Claude skill and build animations.

andyyaro/mcp-keynote

July 2026, 48 commits.

Versions 2.0 to 4.1; a logo and a film in November.

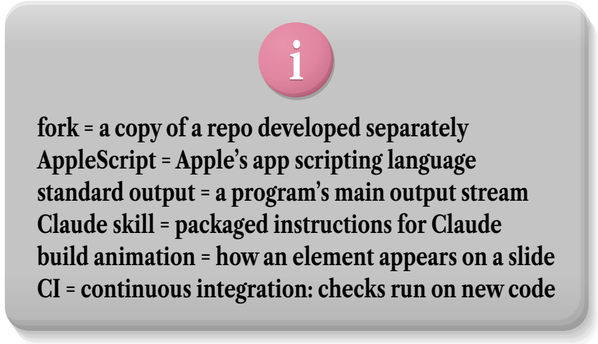
What I Found at ByAxe's Last Commit

Text pasted into AppleScript: slide text had its quotes escaped, but not its backslashes; font, document, theme and layout names, presentation titles, colors and file paths went in raw.

A stray print(), Python's output command, on standard output, the channel MCP messages travel on.

New decks saved to the Desktop without asking; images pasted through the user's clipboard when other methods failed.

No unit tests: a CI workflow existed, but its test folder was empty.

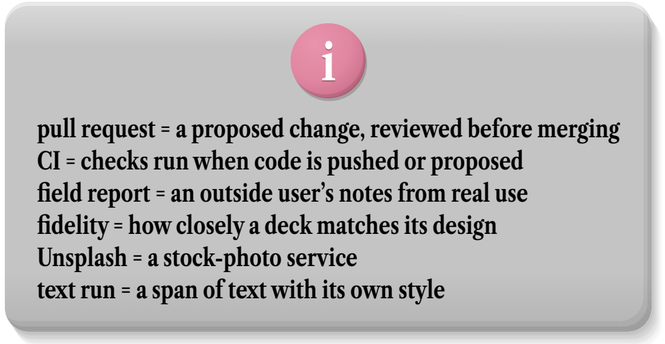


From the code at ByAxe's last commit, aca972f, and CHANGELOG.md 2.0.0.

2. Four Pull Requests, Two Days

Release (pull request)	What it added	Tools	Unit tests	Coverage
At the fork, 13 March	ByAxe's last commit	42 / 45	0	none
2.0 to 2.2 (#1)	Text as arguments, clean standard output, packaging, unit tests and a new CI, exact positions	42 / 45	298	94.5%
3.0.0 (#2)	build_deck to build a deck from a spec and describe_deck to read one back; native tables and charts, styles, transitions, every export	56 / 59	530	95.1%
4.0.0 (#3)	A fidelity pass from a field report: unknown arguments refused, no more guessing which deck, styled lines	58 / 61	726	89.9%
4.1.0 (#4)	Unknown keys in a spec refused at every level; colored text runs inside a spec	58 / 61	757	90.2%

All four pull requests were merged on 26 July 2026. Tools: listed without and with an Unsplash key. Measured at each merge commit in this review; coverage is the share of code lines the unit tests run.



B. Architecture

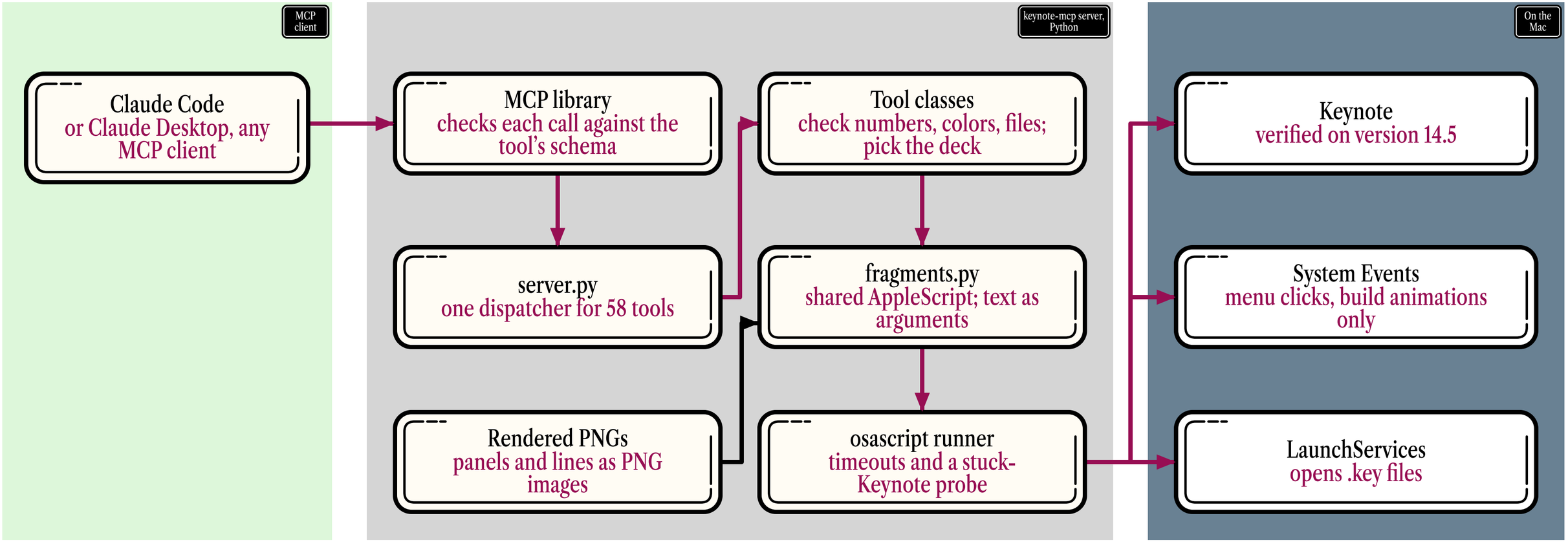


An MCP client such as Claude Code starts the server and talks to it over standard input and output.

Each tool call becomes one or more short AppleScript programs for osascript, the Mac's script runner. The AI's text is never part of those programs.

A few jobs go around Keynote's own scripting commands: macOS opens files, Python draws colored panels and styled lines as images, and build animations use AppleScript to click Keynote's menus through System Events.

1. One Tool Call, End to End



Element tools and build_deck take their AppleScript from fragments.py; the other tools write theirs in their own class. A failed call comes back as ordinary text with isError false, the flag for "not an error"; only the MCP library's own schema refusals carry isError true. The three photo tools, off unless a key is set, are the only ones that use the network.

MCP library = the official mcp Python package
schema = the arguments a tool accepts
dispatcher = code that routes a call to its tool
osascript = runs AppleScript from the command line
System Events = a macOS app that lets scripts work other apps

: Client : Server : macOS and Keynote : Next step : Image file

2. 58 Tools, or 61 With a Key

```
$ .venv/bin/python -I ../checks/tools_count.py
presentation   11  e.g. create_presentation, open_presentation, save_presentation
slide          11  e.g. add_slide, delete_slide, duplicate_slide
content        22  e.g. add_text_box, add_title, add_subtitle
export          4  e.g. screenshot_slide, export_pdf, export_assets
objects         8  e.g. add_table, add_chart, add_line
deck            2  e.g. build_deck, describe_deck
unsplash        0  (only with an UNSPLASH_KEY)
listed to the client: 58
$ grep -n "## Available Tools" README.md
133:## Available Tools (59)
```

1. Six groups, from whole presentations down to single elements.
2. 58 tools are listed to the client. Three photo tools join them only when an Unsplash key is set: 61.
3. The README still says 59, the count at version 3.0.0.

Verified Live, by the Repo's Own Record

docs/TOOL_MATRIX.md marks 58 tools verified on a real Keynote 14.5.

The 3 photo tools were never verified on a live Keynote.

Run on 3 November 2026 with tools_count.py, written for this deck, at commit a4660b2. It builds the server and counts what it registers; no Keynote needed.

i

UNSPLASH_KEY = the setting that holds the photo key

C. Text Stays Text



AppleScript is a programming language. Paste an AI's text into a script, and a stray quote can end the text early, so what follows runs as code. This is called injection.

At the fork, slide text had its quotes escaped, but not its backslashes, while font, document, theme and layout names, presentation titles, colors and file paths went in untouched.

Since version 2.0, every piece of text travels as a separate argument, and the script reads it from argv, its list of arguments.

1. The Same Call, Before and After

```
$ cd ../up && git log -1 --format='%h  %an  %ad' --date=short && ../repo/.venv/bin/python -I \
    ../checks/before.py
aca972f  alekseilitvinau  2026-03-13
command: ['osascript', '-e'] + the script as one argument
font name is pasted into the script text: True
the line it lands on: set font of object text to "Helvetica" & (do shell script "whoami") & ""
$ git log -1 --format='%h  %an  %ad' --date=short && .venv/bin/python -I ../checks/after.py
a4660b2  Andy Yaro  2026-11-02
command: ['/usr/bin/osascript', '-'] + 3 arguments; the script goes in on standard input
font name and title are separate arguments: True
either one is pasted into the script text: False
'do shell script' in the script text: False
the script reads: set docName to item 1 of argv
the script reads: set newItem to make new text item with properties {object text:(item 2 of argv)}
the script reads: set font of object text of newItem to (item 3 of argv)
the script reads: set object text of newItem to (item 2 of argv)
```

1. Before: a font name with a quote in it becomes AppleScript code, here a call to a shell command.
2. After: the same call sends the title and font name as separate arguments. Neither is in the script.
3. The script reads them as items 2 and 3 of argv, so quotes stay plain text.

Run on 3 November 2026 with before.py and after.py, written for this deck, on ByAxe's last commit and on mine. osascript was replaced by a recorder, so nothing ran and no Keynote was involved; whether macOS would have run that shell command was not tested.

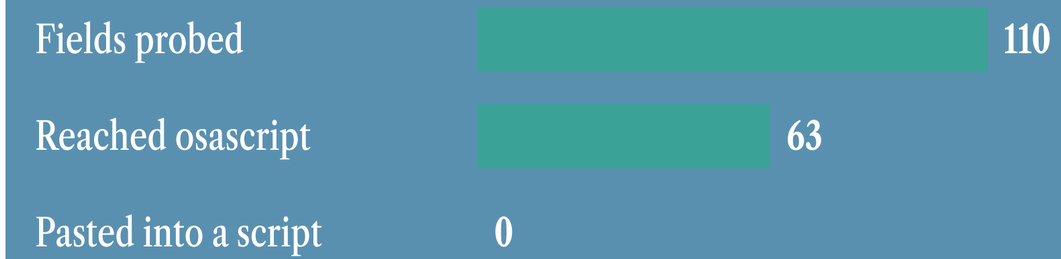
i

argv = the list of arguments a script gets
shell command = a command for the Mac's terminal
recorder = a stand-in that only notes the call

2. Not One Tool: Every Text Field

A Wider Probe, Written for This Review

Every text argument of the 58 tools, with osascript replaced by a recorder.



The other 47 stopped earlier: some failed the server's checks, and some sat behind steps the recorder's dummy replies could not satisfy. A second pass with other values: 95, 85, 0. A separate run of build_deck's full spec and markdown forms: 13 fields reached osascript, none pasted.

What the Repo Tests Itself

108 injection tests: quotes, backslashes, line breaks, AppleScript's ¬ line-continuation mark, emoji, Chinese text, right-to-left Hebrew text, and a classic break-out string.

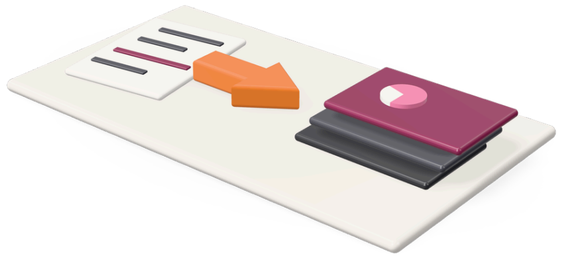
Only checked numbers and fixed word lists are ever written into a script, such as slide numbers and positions.

The rule is written into the repo's CLAUDE.md, the instructions file for coding agents: user text goes only through argv.

The wider probe ran on 3 November 2026 in a separate review clone, with no network. The photo tools were left out; their code writes only checked numbers into scripts. Counts from tests/unit/test_injection.py at a4660b2.

break-out string = text built to end a quoted string early
markdown = plain text with simple formatting marks
osascript = runs AppleScript from the command line
recorder = a stand-in that only notes the call
argv = the list of arguments a script gets

D. One Spec, Whole Deck

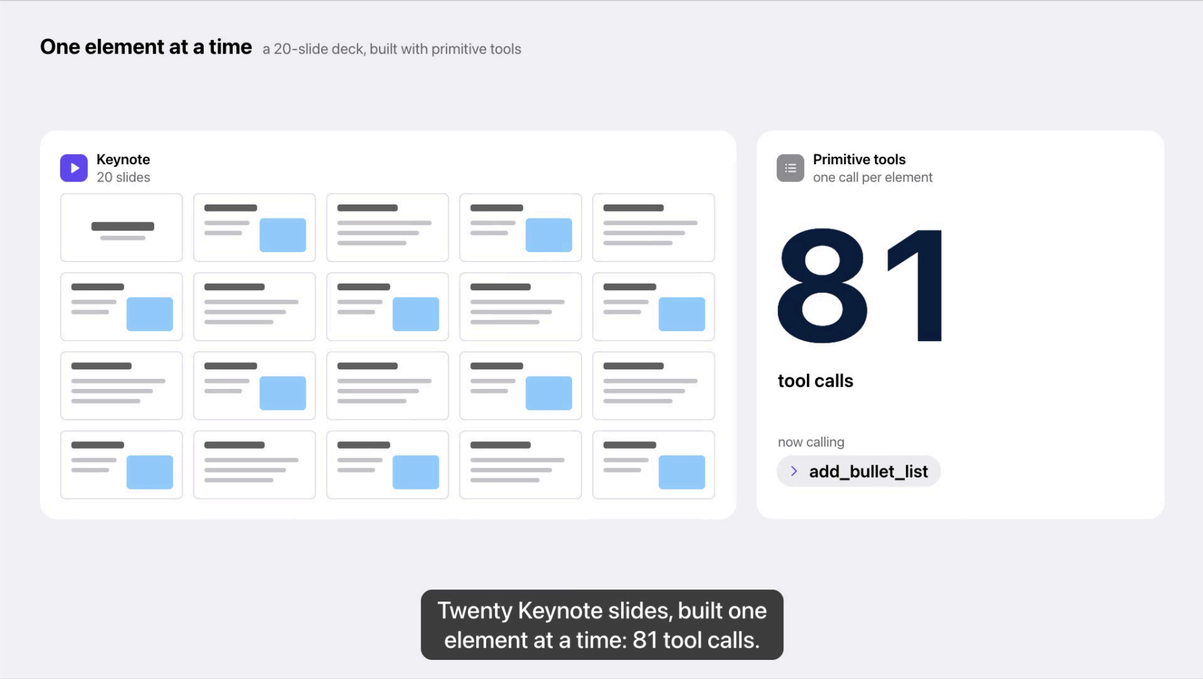


The AI writes one spec, a structured description of the deck: slides, tables, charts, notes, transitions and a style.

The server checks the spec before it creates anything, except slide layout names, which it can check only once the new deck exists; a bad one discards that deck. It builds five slides per AppleScript run.

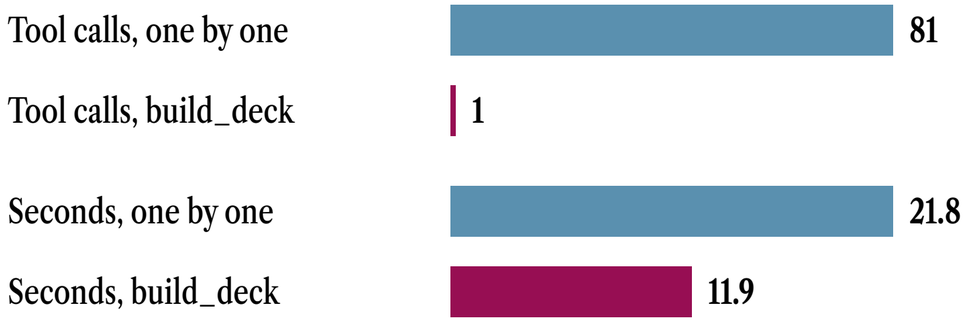
describe_deck reads a deck back into the same format, charts without their data, so most of a deck can live as text and be compared in git, the system that records every change to a repo.

1. One Call Instead of 81



A frame from the launch film. The 81 calls and the times come from the repo's own measurement on Keynote 14.5; they could not be re-run here.

A 20-slide deck, measured by the repo



The README says the point is the call count, not the clock: both paths send Keynote one event at a time. Fewer calls mean fewer chances for an index or position to go wrong.

A Spec That Passes the Checker

```
{"title": "Q3 Review", "style": "boardroom", "save_path": "~/Documents/q3.key", "slides": [
 {"title": "Revenue by Region", "transition": {"effect": "dissolve"}, "elements": [{"type": "chart", "chart_type": "pie",
   "row_names": ["Revenue"], "column_names": ["North", "South", "West"], "data": [[30, 45, 25]]}]},
 {"title": "Headcount", "elements": [{"type": "table", "data": [["Team", "Now"], ["Eng", 24], ["Sales", 11], ["Total", "=SUM(B2:B3)"]]}]}]}
```

2. Checked Before Anything Is Created

```
$ .venv/bin/python -I ../checks/stdio_probe.py spec
isError: False
Spec validation failed (6 problem(s)); nothing was created:
- slides[0].titel: unknown key for a slide. Did you mean 'title'?
- slides[0]: a slide accepts: body, elements, layout, notes, skipped, title, transition. Nothing
  was created.
- slides[0].transition.effect: 'disolve' invalid; e.g. dissolve, push, magic_move (underscored
  Keynote effect names)
- slides[1].elements[0]: chart_type 'pie3d' invalid; valid: ['area', 'area_3d', 'bar', 'bar_3d',
  'horizontal_bar', 'horizontal_bar_3d', 'line', 'line_3d', 'pie', 'pie_3d', 'scatter',
  'stacked_area', 'stacked_area_3d', 'stacked_bar', 'stacked_bar_3d', 'stacked_horizontal_bar',
  'stacked_horizontal_bar_3d']
- slides[1].elements[0]: chart needs 'row_names'
- slides[1].elements[0]: chart needs 'column_names'
$ .venv/bin/python -I ../checks/stdio_probe.py arg
isError: True
Input validation error: Additional properties are not allowed ('fill_color' was unexpected)
```

1. Over the real protocol, a typo in a spec comes back named, with the right key, and nothing is created.
2. A wrong value comes back with valid ones listed or sampled.
3. A made-up argument on a tool gets the MCP library's generic refusal. The server's own hint, which would name the right tool, never runs.

Run on 3 November 2026 with stdio_probe.py, written for this deck: it starts the server and sends messages as an MCP client would. The spec fails before any AppleScript. Note isError false on the spec reply: a client must read the words.

stdio = standard input and output
isError = the reply flag that marks a failure

E. What Keynote Allows



Keynote's scripting covers less than its interface: no fill colors on shapes or text boxes, no line styles, no grouping.

Each limit was probed on a real Keynote or found missing from its scripting dictionary, and worked around where possible: colored panels and styled lines are drawn as images.

Version 4.0 came from a field report written while reverse-engineering a real 35-slide deck. One claim in it, that a tool could set a fill color, was wrong, and checking it exposed a real bug.

1. What Keynote Will Not Do

Keynote's scripting cannot	So the server	The cost
Set a fill color on a shape or text box	Draws colored panels as PNG images	Exact color; cannot be recolored in Keynote
Style a line: color, width, dash, arrows	styled_line draws the line as a PNG	Same: an image, not a line
Read or move a theme's title boxes	Positioned text boxes when placement matters	Loses the theme's placeholders
Add build animations	Clicks Keynote's menus through System Events	Needs Accessibility access, an unlocked screen and focus; English menu names
Z-order, grouping, links, underline, movies	None: listed in README.md and docs/CEILING.md	Finish these by hand

From README.md and docs/CEILING.md. The scripting dictionary in the repo agrees on the first two: fill is read-only on shapes and text boxes, and a line has no color, width, dash or arrowhead properties. Table cells are the exception: they take a background color, which the server uses for header rows.



scripting dictionary = the list of what an app accepts
z-order = which element is in front
Accessibility = the macOS permission to control the screen
placeholder = a theme's preset text box
PNG = an image file format
System Events = a macOS app that lets scripts work other apps

2. Live Checks, and the Bug They Found

Checked on a Real Keynote

58 tools verified on Keynote 14.5, macOS 26.5.1.

289 live checks at 4.1.0, none failed; the README and TOOL_MATRIX still say 196.

Rendered checks read pixels in the exported slides: colors, ink and clipping.

8 live tests in pytest, left out by default and in CI.

The Bug Behind Version 4.0

The field report said a tool could set a fill color.

12 write routes, ways of setting a property, on 5 themes all failed.

The real cause: every tool silently dropped unknown arguments, so fill_color vanished and the call reported success.

Now unknown arguments are refused.

Keynote 15

In September 2026, an outside contributor, Peter Siska, re-ran the live checks on Keynote 15.3.1: 289 checks, 4 failed.

Build animations needed a fix for 15.x.

That work is in five pull requests, not merged into master.

From docs/TOOL_MATRIX.md, docs/ENVIRONMENT.md and CHANGELOG.md 4.0.0 and 4.1.0, and the commit messages in pull request 6, fetched read-only from GitHub. None of these live runs could be repeated here: they need a Mac with Keynote. pytest is the Python test runner; master is the repo's main branch.

F. Tests and CI



757 unit tests run in about 5 seconds with osascript replaced by a stand-in, so they need no Mac.

8 more tests drive a real Keynote. They are left out unless asked for, and CI cannot run them.

Measured coverage is 90%, not the 95% on the README badge, which was added at version 2.0 and last held at 3.0.

1. The Tests, Run

```
$ .venv/bin/python -m pytest -q -p no:cacheprovider 2>&1 | tail -1
757 passed, 8 deselected in 4.45s
$ .venv/bin/python -m pytest -q -p no:cacheprovider -m keynote --collect-only 2>&1 | tail -1
8/765 tests collected (757 deselected) in 0.18s
$ .venv/bin/python -m pytest -q -p no:cacheprovider --cov=keynote_mcp --cov-fail-under=85 2>&1 | \
    grep -E "^TOTAL|Required"
TOTAL                                           3371    329    90%
Required test coverage of 85% reached. Total coverage: 90.24%
$ .venv/bin/ruff check src/ tests/
All checks passed!
$ .venv/bin/ruff format --check src/ tests/ | tail -1
54 files already formatted
$ .venv/bin/mypy 2>&1 | tail -1
Success: no issues found in 23 source files
```

1. 757 unit tests pass. The review also ran them on Python 3.11 and 3.12.
2. 8 live tests, marked keynote, are left out by default.
3. Coverage is 90.24%; CI requires 85%. The README badge says 95%.
4. The linter, the formatter check and strict type checking are all clean.

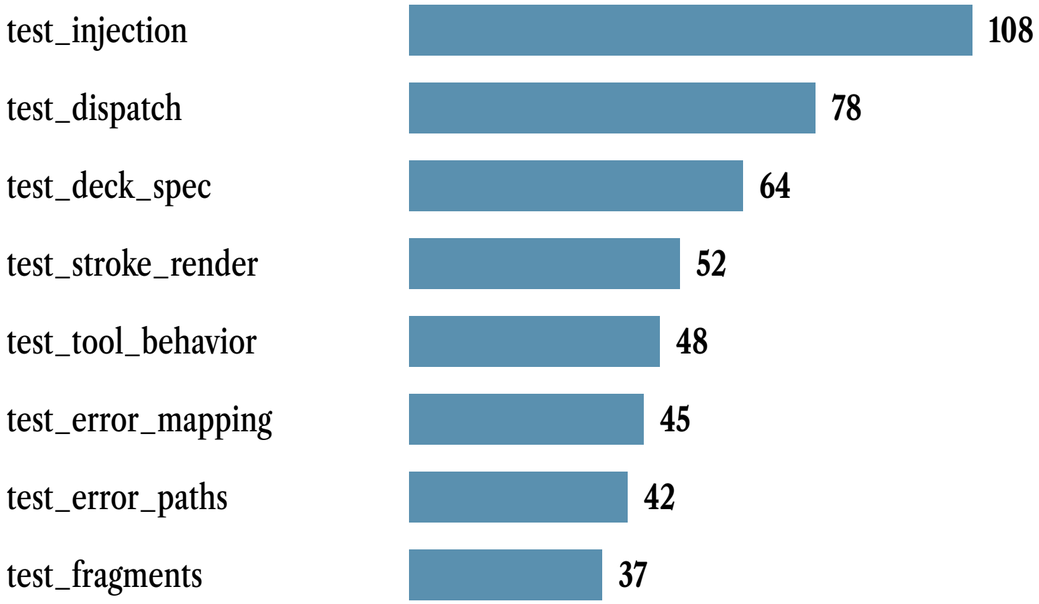
Run on 3 November 2026 on a clean clone at commit a4660b2, Python 3.13, packages installed with uv from the lock file of exact versions.

i

unit test = checks one piece of code alone
coverage = share of lines the tests run
ruff = a linter and formatter: flags risky or messy code
mypy = a Python type checker
uv = a Python package installer; its lock file pins versions
marked = tagged so pytest can include or leave out tests

2. What the Tests Cover

Unit tests by file, the largest eight of 26



Lowest coverage: export.py at 72% and objects.py at 82%, the code that writes files and draws objects.

CI's pass or fail history on GitHub was not checked in this review.

Continuous Integration

Runs on GitHub's macOS machines on each push to master, the main branch (and to an old modernize branch), and each pull request to master.

One job: the linter, the formatter check and strict type checking.

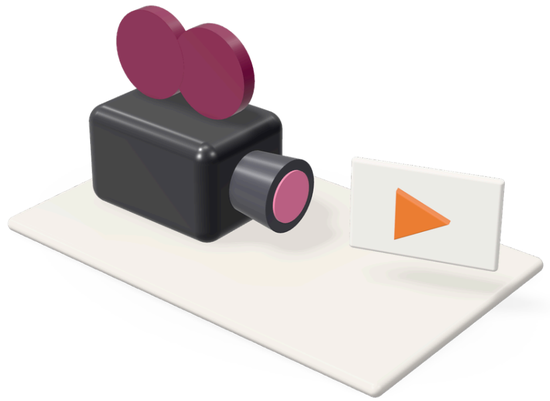
One job: the unit tests on Python 3.11, 3.12 and 3.13, failing below 85% coverage.

No Keynote there, so no live tests.

Unit Tests Over Time

At the fork	0
After 2.x	298
After 3.0.0	530
After 4.1.0	757

G. History and the Film



Two upstream eras: easychen's 2 commits in July 2025 and ByAxe's 15 in March 2026.

Then mine: 48 commits on 25 and 26 July 2026, made with Claude Code, Anthropic's coding agent. 43 of my 46 ordinary commits credit a Claude model as co-author.

In November came a logo and a 67-second launch film.

1. Three Eras, Then Two Days

```
$ git log --date=format:%Y-%m-%d --format='%ad %an' | sort | uniq -c
      1 2025-07-05 Easy
      1 2025-07-06 Easy
      1 2026-03-09 alekseilitvinau
      4 2026-03-10 alekseilitvinau
      7 2026-03-11 alekseilitvinau
      2 2026-03-12 alekseilitvinau
      1 2026-03-13 alekseilitvinau
      3 2026-07-25 Andy Yaro
     45 2026-07-26 Andy Yaro
      1 2026-11-01 Andy Yaro
      1 2026-11-02 Andy Yaro
$ git log --merges --date=short --format='%ad %s'
2026-07-26 Merge pull request #4 from andyyaro/doc-truth
2026-07-26 Merge pull request #3 from andyyaro/fidelity-pass
2026-07-26 Merge pull request #2 from andyyaro/capability-expansion
2026-07-26 Merge pull request #1 from andyyaro/modernize
$ git for-each-ref refs/tags --format='%(refname:short)  %(creatordate:format:%Y-%m-%d)'
film-v1  2026-11-02
v3.0.0  2026-07-26
$ git log --author='Andy Yaro' --format=%B | grep -c 'Co-Authored-By: Claude'
43
$ git log --author='alekseilitvinau' --format=%B | grep -c 'Co-Authored-By: Claude'
15
```

1. 48 of my commits on 25 and 26 July 2026, in under 18 hours; one each in November for the logo and the film.

2. Four pull requests, all merged on 26 July: modernize, capability-expansion, fidelity-pass and doc-truth.

3. Only two tags: v3.0.0 and the film. Versions 2.x, 4.0.0 and 4.1.0 have none.

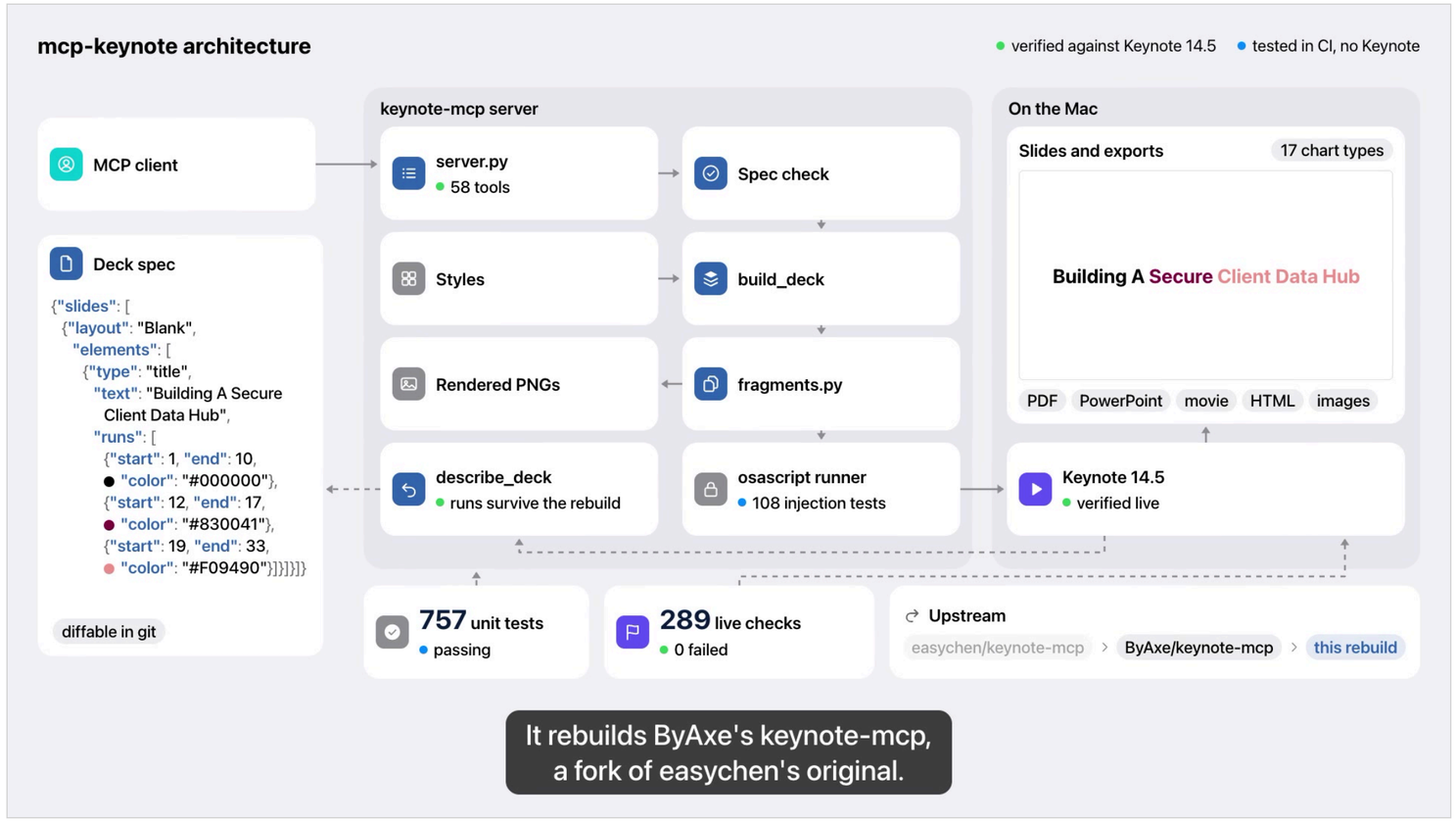
4. 43 of my commits credit Claude as co-author; so do all 15 of ByAxe's.

Run on 3 November 2026 on a clone at a4660b2: 67 commits, 2 by easychen (git name "Easy"), 15 by ByAxe (git name alekseilitvinau) and 50 by me. Five more pull requests, #5 to #9, by an outside contributor, are on GitHub but not in master. Dates are in each author's own time zone; mine are US Eastern.

i

tag = a named point in history
merge = joining a pull request into master
master = the repo's main branch

2. The 67-Second Film



A frame from the film: an animated board drawn as a web page. Nothing in it was recorded from a screen, and Keynote was not driven for it. Its 289 live checks are the repo's record.

How It Was Made

66.6 seconds, 1920 × 1080, 18.9 MB.

A web page animated with GSAP, a JavaScript animation library, and turned into video with HyperFrames, which renders web pages frame by frame.

Narrated by Kokoro-82M, an open text-to-speech model. Music: "Angel Watching Me" by Arulo, from Mixkit, a free stock-media library.

Made in a Claude Code session; the commit credits Claude Opus 5.5 as co-author.

Where to Watch

andyyaro.github.io/mcp-keynote/film.html

The page loaded on 3 November 2026.

H. Proof and Limits



Checked on 3 November 2026: the unit tests, the same call before and after my changes, and the spec checks over the real protocol.

The core claims hold: text stays text, and a spec that fails its checks creates nothing. Some headline numbers and defaults do not.

The last slides set my notes against the repo, list risks and fixes, and say what the project shows.

1. My Claims Against the Repo

My notes said	The repo shows	So I now say
Jul 26, 2026	First commit late on 25 July, all four merges on 26 July; logo and film in November	Rebuilt 25–26 July 2026
48 of the repo's 50 commits are mine	50 of 67 now; 48 of 65 at the end of 26 July. The other 17 are upstream	50 of 67 commits
59 tools verified against a real Keynote	61 tools; 58 verified on Keynote 14.5; the 3 photo tools never verified live	61 tools, 58 verified live
Spec schema validation (v4.1)	Unknown keys refused at every level, in Python; the schema clients see for spec says only "an object"	Strict spec-key checks
Injection-safe AppleScript	Confirmed: text goes as arguments in every tool probed	As stated
6 stars, 2 forks	Not checked in this review	Check on GitHub first
Public fork	Public, and a fork of ByAxe's fork of easychen's original	As stated
Took an existing server and rebuilt it	43 of my 46 ordinary commits credit a Claude model as co-author	Rebuilt it with Claude Code

From my career-hub notes, checked against the repo on 3 November 2026. The README's 59 tools, 95% coverage and 196 live checks are stale in the same way.

2. Risks Found in This Review

1. **build_deck replaces by default**
It closes an open copy of the target file without saving, and if a layout name then fails, deletes the file there. Read from the code, not run.

2. **Failures look like successes**
Errors come back as plain text with isError false, meaning “not an error”; a client must read the words.

3. **No safeguards on destructive tools**
No confirmation, no dry run, and no MCP tool annotations, so a client cannot tell read-only tools from destructive ones.

4. **Exports write anywhere**
An output path is only checked for being non-empty.

5. **Photo tools lag behind**
They act on whichever deck is in front, take no deck name, and were never verified live.

6. **Build animations are fragile**
They click menus by their English names, and an outside contributor found they fail on Keynote 15; his fix is not merged.

7. **The setup script weakens safety**
It advises turning off the lock-screen password for the session and running Claude Code with its permission prompts off.

8. **Inherited files are out of date**
SECURITY.md and glama.json are ByAxe’s, CONTRIBUTING still sends issues to ByAxe, and the server reports version 1.28.1, its library’s.

Found in this review from the code at a4660b2; no GitHub issues were reviewed. A dry run reports what would change without changing it; tool annotations are MCP hints saying whether a tool only reads or may destroy.

3. What to Fix Next

1. **Make build_deck safe by default**
   Refuse to replace a file with unsaved changes, and never delete a file it did not create.

2. **Mark errors as errors**
   Return isError true on failures, and add MCP hints for read-only and destructive tools.

3. **Answer the outside pull requests**
   Review the Keynote 15 fixes, mcp 1.30.0, edit_table and theme files.

4. **Correct the headline numbers**
   61 tools with 58 verified, 90% coverage, 289 live checks, and the right server version.

5. **Bring the photo tools in line**
   Target a named deck instead of the front document, and verify them live.

6. **Rewrite the inherited files**
   SECURITY.md, CONTRIBUTING, glama.json (the Glama MCP directory listing) and the skill's details.

7. **Make the setup script safe**
   Drop the lock-screen and skip-permissions advice; ask only for the grants the server needs.

8. **Clean up what is public**
   Use GitHub's private no-reply email, keep session links and home paths out, and tag 4.1.0.

Roughly in order of risk: lost work first, then misread results, then what the repo says, then housekeeping.

4. What It Shows, and What It Does Not

It Shows

Taking over someone else's code and, with Claude Code, closing an injection hole, with a before-and-after test.

Whole decks from one checked spec, read back into the same spec.

757 passing tests, strict typing, and Keynote's scripting limits mapped.

It Does Not Show

That it works on Keynote 15 without the outside fixes.

Safe defaults for tools that replace or delete.

Live results I could repeat here: those need a Mac.

That the README's headline numbers are current.

The core work holds up: text stays text, and a spec that fails its checks creates nothing. The next step is safe defaults for build_deck, honest error flags, and a fresh live run on Keynote 15.