Forth in Jupyter Notebooks

August 23, 2026

1 Using Forth in a Jupyter Notebook

Jupyter is an application that lets us combine documentation and executable code in a *Notebook* document. Using a custom Jupyter Kernel we can run Forth code and bring the output back into the notebook.

A notebook is list of cells that contain either documentation or code. We can think of these cells like Forth blocks: they can be individually executed and we develop our Forth program one cell at a time. Unlike blocks, they can be of arbitrary size.

This gives us the interactivity we have at the Forth prompt, plus the session is a file you can save, diff, and hand to someone else.

Andrew McKewan, August 2026

2 Notebook Demo

[4]:
```
.( hello )

: test 12 0 do i . loop ;
test
```

```
hello 0 1 2 3 4 5 6 7 8 9 10 11
```

[4]:
```
<0> ok
```

3 Jupyter Architecture

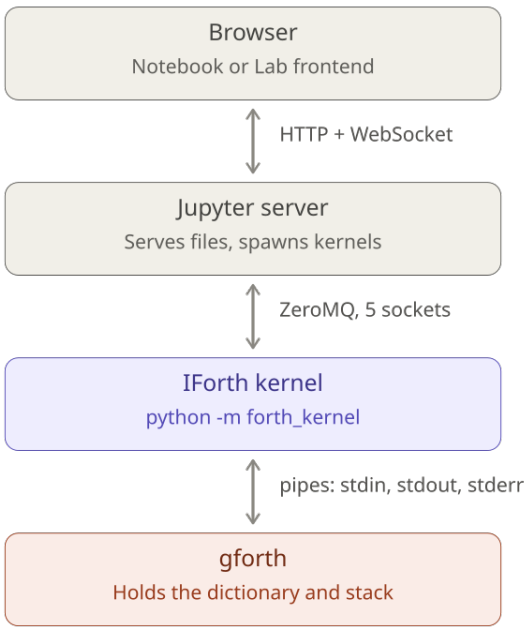
Jupyter is a small stack of cooperating processes:

- **Browser** — renders cells, displays output. Executes nothing.
- **Server** — owns the notebook file, launches kernels. Language-agnostic.
- **Kernel** — a separate process that actually runs your code.
- **Protocol** — JSON messages over ZeroMQ. The server doesn't know what language it's talking to.

Where Forth fits in:

- The IForth kernel is a few hundred lines of Python.
- Speaks Jupyter's protocol upward, drives gforth downward through plain pipes.
- Writes a line to gforth's stdin, reads what comes back, returns it as cell output.

- One gforth session per kernel — the dictionary persists for the whole notebook.



Purple and coral are the Forth-specific layers

4 IForth Kernel

Here is the IForth repo on github: https://github.com/sohang3112/iforth

How it actually works

- One gforth subprocess per kernel, started on first use, killed on restart.
- A cell is split into lines; each is written to stdin and the output read back.
- gforth echoes what you send it, so the kernel skips the echo before reading real output.
- After the cell, the kernel sends .s and shows the stack as the cell's result.

Where it gets interesting

- No terminal, so nothing signals "output is finished" — gforth just goes quiet.
- Original approach: wait a fixed time and assume it's done. Slow when nothing's happening, truncated when something is.
- Better: watch for gforth's own ok / compiled prompt as the end marker.
- But QUIT re-enters the interpreter without a prompt — so ttester.fs failing a test hangs the cell.
- Fix: send your own marker after each line and read until it comes back.

What still doesn't work

- Words that read input (KEY, ACCEPT) — there's no way to type into a running word.
- Anything using AT-XY or PAGE — no terminal to address.
- Output that happens to end in ok can be mistaken for the prompt