# Ship a Node CLI in 4 files

4 steps · swipe →

@inkwell.notes

# 1. Point "bin" at your built entry file

package.json

```
{
  "name": "@inkwell/notes",
  "version": "0.1.0",
  "type": "module",
  "bin": { "notes": "./dist/cli.js" },
  "files": ["dist"],
  "scripts": { "build": "tsc", "test": "node --test" }
}
```

the command name

# 2. Parse arguments with node:util, no dependency

src/cli.ts

```ts
#!/usr/bin/env node
import { parseArgs } from "node:util";
import { addNote, listNotes } from "./store.js";

const { values, positionals } = parseArgs({
  allowPositionals: true,
  options: { tag: { type: "string", short: "t" } },
});

const [command, ...words] = positionals;
if (command === "add") await addNote(words.join(" "), values.tag);
else if (command === "ls") console.table(await listNotes(values.tag));
else console.error("usage: notes add <text> [-t tag] | notes ls [-t tag]");
```

# 3. Test the real binary, not just functions

test/cli.test.ts

```ts
import { test } from "node:test";
import assert from "node:assert/strict";
import { execFileSync } from
  "node:child_process";

test("adds and lists a note", () => {
  const run = (...args: string[]) =>
    execFileSync("node", ["dist/cli.js",
      ...args], { encoding: "utf8" });
  run("add", "buy oat milk", "-t", "home");
  assert.match(run("ls", "-t", "home"), /buy
    oat milk/);
});
```

# 4. Build, test, bump, publish

terminal

```
npm run build && npm test
npm version patch
npm publish --access public
npx @inkwell/notes add "it works" -t demo
```