

# Options and Commands

---



**Paul O'Fallon**

@paulofallon



# Overview



Options in a CLI

The command pattern

Authorizing with Twitter

Calling the Twitter API



# CLI Options Parameters

LS(1)

User Commands

LS(1)

## NAME

ls - list directory contents

## SYNOPSIS

ls [OPTION]... [FILE]...

## DESCRIPTION

List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all  
do not ignore entries starting with .

-A, --almost-all  
do not list implied . and ..

--author  
with -l, print the author of each file

-b, --escape  
print C-style escapes for nongraphic characters

--block-size=SIZE  
scale sizes by SIZE before printing them; e.g., '--block-size=M' prints sizes in units of 1,048,576 bytes; see SIZE format below

Manual page ls(1) line 1 (press h for help or q to quit)



# CLI Options Parameters

✓ `ls -l -a`

✓ `ls -la`

✓ `ls -l --all`

✓ `ls -l --all --block-size=K`

✓ `ls -l --all --block-size K`

# CLI Command Pattern

→ ~ npm

Usage: npm <command>

where <command> is one of:

access, adduser, bin, bugs, c, cache, completion, config,  
ddp, dedupe, deprecate, dist-tag, docs, doctor, edit,  
explore, get, help, help-search, i, init, install,  
install-test, it, link, list, ln, login, logout, ls,  
outdated, owner, pack, ping, prefix, profile, prune,  
publish, rb, rebuild, repo, restart, root, run, run-script,  
s, se, search, set, shrinkwrap, star, stars, start, stop, t,  
team, test, token, tst, un, uninstall, unpublish, unstar,  
up, update, v, version, view, whoami

npm <command> -h      quick help on <command>  
npm -l                display full usage info  
npm help <term>      search for help on <term>  
npm help npm          involved overview



# CLI Command Pattern

→ ~ npm

Usage: npm <command>

where <command> is one of:

access, adduser, bin, bugs, c, cache, completion, config,  
ddp, dedupe, deprecate, dist-tag, docs, doctor, edit,  
explore, get, help, help-search, i, init, install,  
install-test, it, link, list, ln, login, logout, ls,  
outdated, owner, pack, paste, ping, prefix, prefix-git,  
publish, rb, re, rm, rm-repo, run-script, search, set,  
s, se, search, set-script, set-script-sh, set-script-sh-  
team, test, tok, token, token, token, token, token,  
up, update, v, version, whoami

npm <command> -h

npm -l

npm help <term>

npm help npm

→ ~ git

usage: git [--version] [--help] [-C <path>] [-c name=value]  
[--exec-path=<path>] [--html-path] [--man-path] [--info-path]  
[-p | --paginate | --no-pager] [--no-replace-objects] [--bare]  
[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]  
<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone	Clone a repository into a new directory
init	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add	Add file contents to the index
mv	Move or rename a file, a directory, or a symlink
reset	Reset current HEAD to the specified state
rm	Remove files from the working tree and from the index



# CLI Command Pattern

→ ~ npm

Usage: npm <command>

where <command> is one of:

access, adduser, bin, bugs, c, cache, comp  
ddp, dedupe, deprecate, dist-tag, docs, do  
explore, get, help, help-search, i, init,  
install-test, it, link, list, ln, login, l  
outdated, owner → ~ git  
publish, rb, re usage: git [--version] [--  
s, se, search, [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]  
team, test, tok [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]  
up, update, v, [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]  
<command> [<args>]

npm <command> -h

npm -l

npm help <term>

npm help npm

di These are common Git commands used in various situations:

in start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

reset Reset current HEAD to the specified state

rm Remove files from the working tree and from the index

→ ~ aws

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]

To see help text, you can run:

aws help

aws <command> help

aws <command> <subcommand> help

aws: error: too few arguments



# commander

public

build passing

downloads

30M/month

gitter

join chat

The complete solution for **node.js** command-line interfaces, inspired by Ruby's **commander**.

[API documentation](#)

## Installation


```
$ npm install commander --save
```

## Option parsing


Options with commander are defined with the `.option()` method, also serving as documentation for the options. The example below parses args and options from `process.argv`, leaving remaining args as the `program.args` array which were not consumed by options.

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */
```

 npm install commander

[how? learn more](#)

 abetomo published 5 days ago

2.13.0 is the latest of 51 releases

[github.com/tj/commander.js](https://github.com/tj/commander.js)

MIT

Collaborators [list](#)



## Stats

1,496,508 downloads in the last day

8,984,609 downloads in the last week

29,957,992 downloads in the last month

178 open issues on GitHub

44 open pull requests on GitHub





```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var program = require('commander');

program
  .version('0.1.0')
  .option('-p, --peppers', 'Add peppers')
  .option('-P, --pineapple', 'Add pineapple')
  .option('-b, --bbq-sauce', 'Add bbq sauce')
  .option('-c, --cheese [type]', 'Add the specified type of cheese [marble]', 'marble')
  .parse(process.argv);

console.log('you ordered a pizza with:');
if (program.peppers) console.log('  - peppers');
if (program.pineapple) console.log('  - pineapple');
if (program.bbqSauce) console.log('  - bbq');
console.log('  - %s cheese', program.cheese);
```



# Commander and commands

`/bin/something.js` → `program.command('foo', 'A command')`

`/bin/something-foo.js` → `program.command('bar', 'A sub-command')`

`/bin/something-foo-bar.js` → `program`  
`.command('baz')`  
`.description('A sub-sub-command')`  
`.action(...)`



# Commands in Our CLI

twine configure consumer



a top-level command



a sub-command of “configure”



# Commands in Our CLI

## PSTwine

[Details](#)[Settings](#)[Keys and Access Tokens](#)[Permissions](#)

### Application Settings

*Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.*

Consumer Key (API Key) 0tqjdi4rwt99ZfxgtuM8Spd4y

Consumer Secret (API Secret) O5MGLxI7VznrXXXWOp6Rx7C6xdNFnKWet3nVspRLgDUXTAdcBL

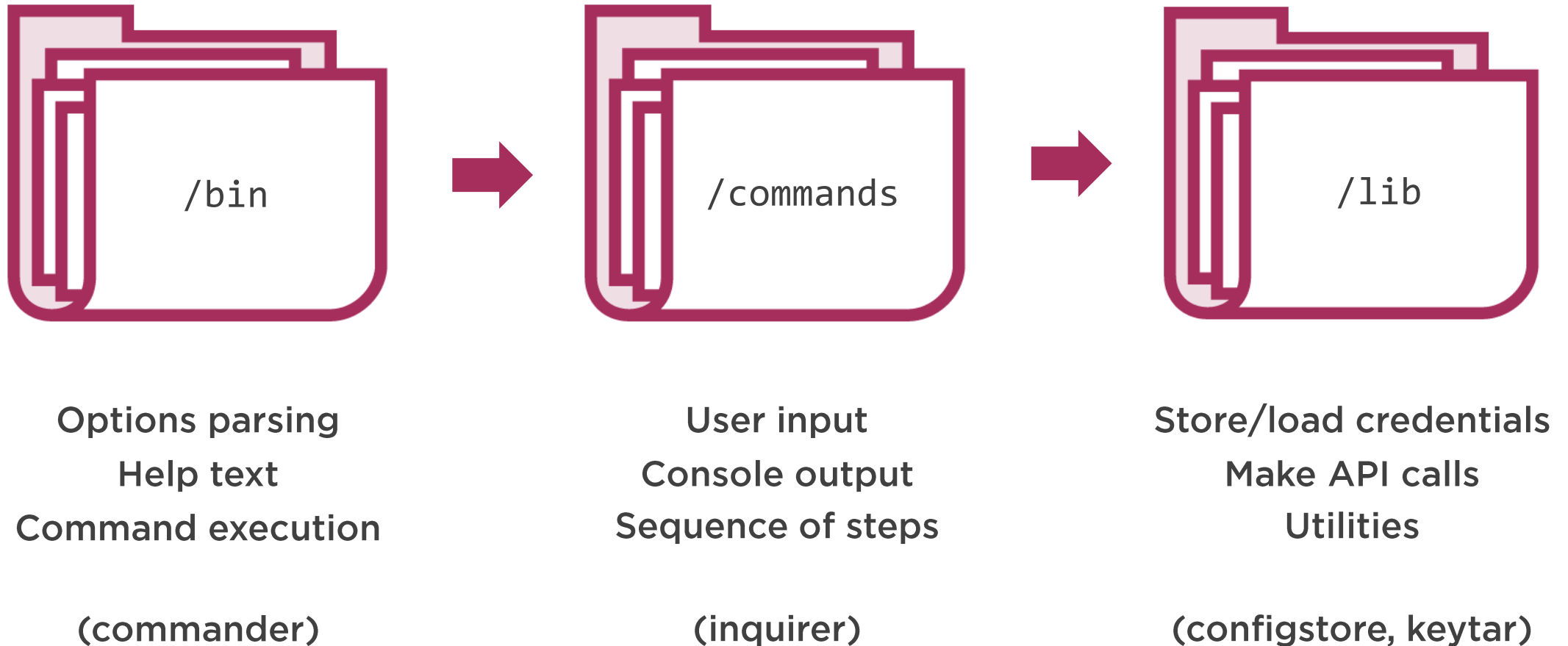
Access Level Read and write ([modify app permissions](#))

Owner paulofallon

Owner ID 242127204



# Project Directory Structure



# Demo



Add Commander module

Refactor our first command

Add code coverage



# Twine in Three Steps

Previous Module

**Consumer  
API Key and  
Secret**

App developer  
(at deploy time)

This Module

**Twitter  
Account Token  
and Secret**

App user  
(once, or seldom)

Subsequent Module

**Specific  
Twitter API Call**





 Search all documentation...

## Basics

[Getting started](#)[Things every developer should know](#)[Authentication](#)[Rate limits](#)[Rate Limiting](#)[Response Codes](#)[Cursoring](#)[Security](#)[Twitter IDs \(snowflake\)](#)[Counting characters](#)[t.co links](#)[Adding international support to apps](#)

## Accounts and users

# Authentication

[Overview](#)[Guides](#)[API Reference](#)

Overview contents ^

[OAuth with the Twitter API](#)[Using OAuth](#)[Application-only authentication](#)[Basic authentication](#)[3-legged OAuth](#)[PIN-Based OAuth](#)[OAuth echo](#)[Authentication and authorization](#)[Application permission model](#)

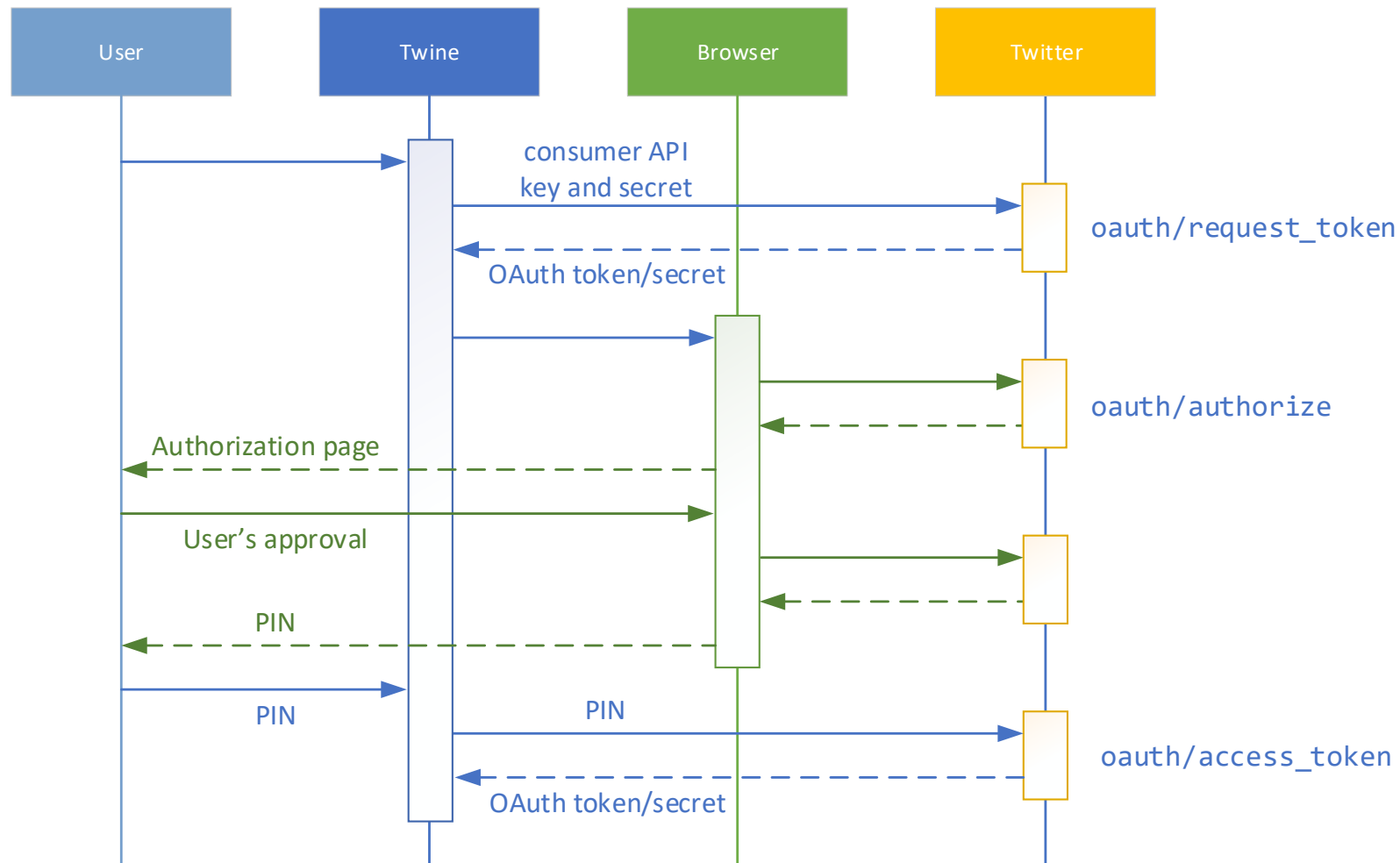
## PIN-based authorization

The PIN-based OAuth flow is intended for applications which cannot access or embed a web browser in order to redirect the user to the authorization endpoint. Examples of such applications would be command-line applications, embedded systems, game consoles, and certain types of mobile apps.





# Twitter's PIN-based Authentication



opn public

A better **node-open**. Opens stuff like websites, files, executables. Cross-platform.

## Why?

- Actively maintained
- Supports app arguments
- Safer as it uses `spawn` instead of `exec`
- Fixes most of the open `node-open` issues
- Includes the latest **xdg-open script** for Linux

## Install


```
$ npm install opn
```

## Usage


```
const opn = require('opn');

// Opens the image in the default image viewer
opn('unicorn.png').then(() => {
  // image viewer closed
});

// Opens the url in the default browser
opn('http://sindresorhus.com');
```

 npm install opn

[how? learn more](#)

 [sindresorhus](#) published 3 weeks ago

**5.2.0** is the latest of 18 releases

[github.com/sindresorhus/opn](https://github.com/sindresorhus/opn)

MIT

Collaborators [list](#)



## Stats

**400,194** downloads in the last day

**2,309,480** downloads in the last week

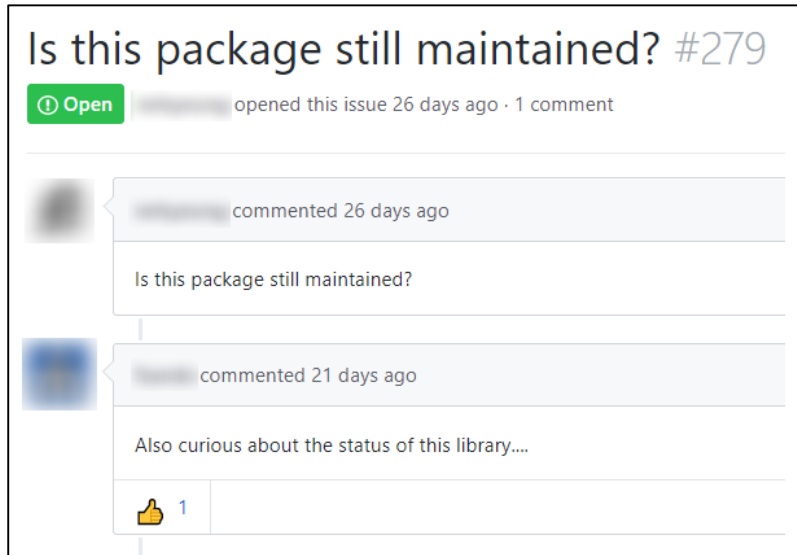
**9,200,671** downloads in the last month

**13 open issues** on GitHub

**One open pull request** on GitHub



# Finding a Twitter Library



**No PIN-based authentication**

**Lots of open issues**

**No longer maintained**



# Demo



Implement a Twitter API client

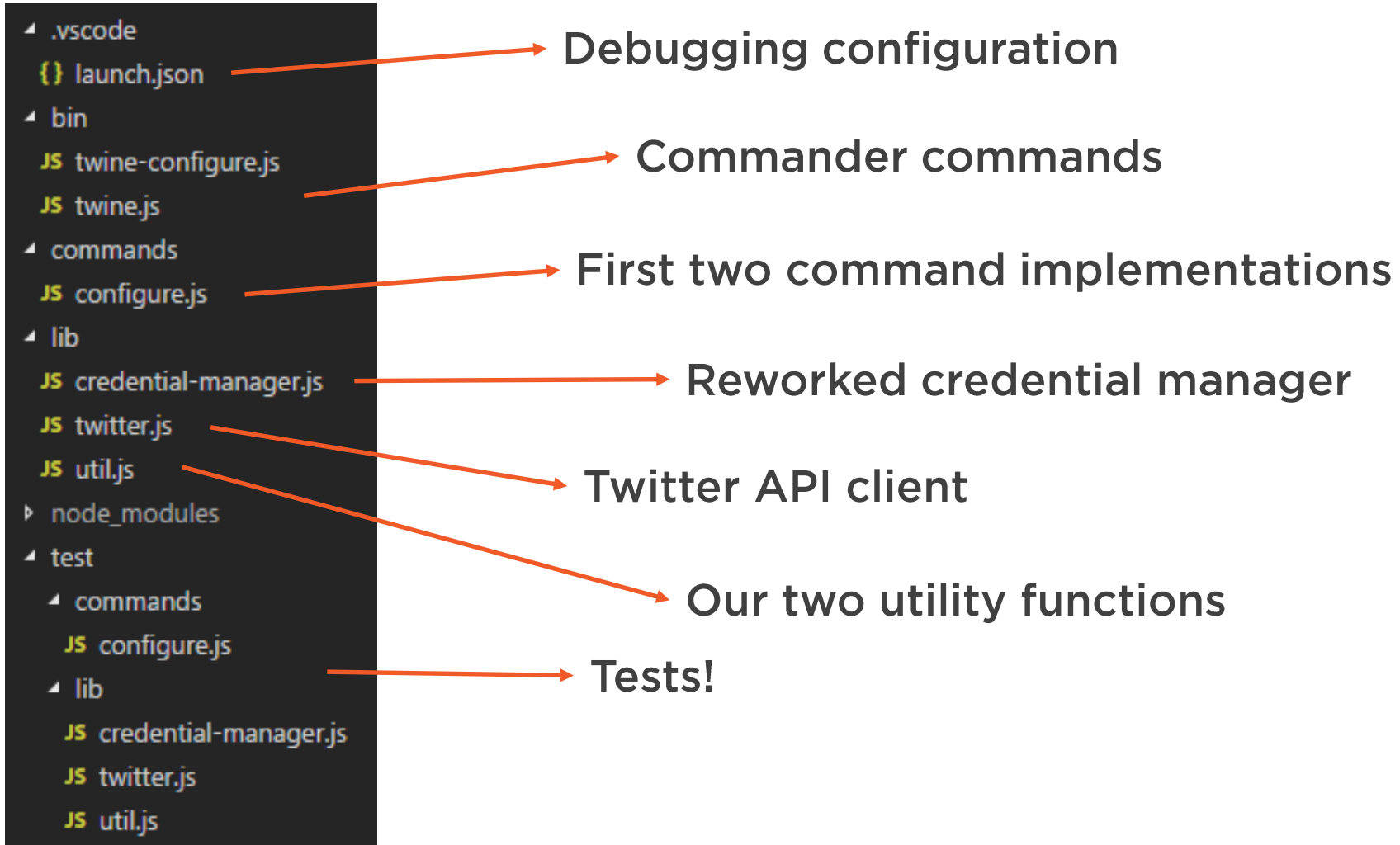
Add our second command

Implement PIN-based authentication

Run twine in the node debugger



# Where We Are Now



# Summary



Common option and command patterns

Restructured our project

Introduced some new modules

Implemented a minimal Twitter client

Finished step two of our twine CLI

