Interacting with the Environment

Paul O'Fallon

@paulofallon

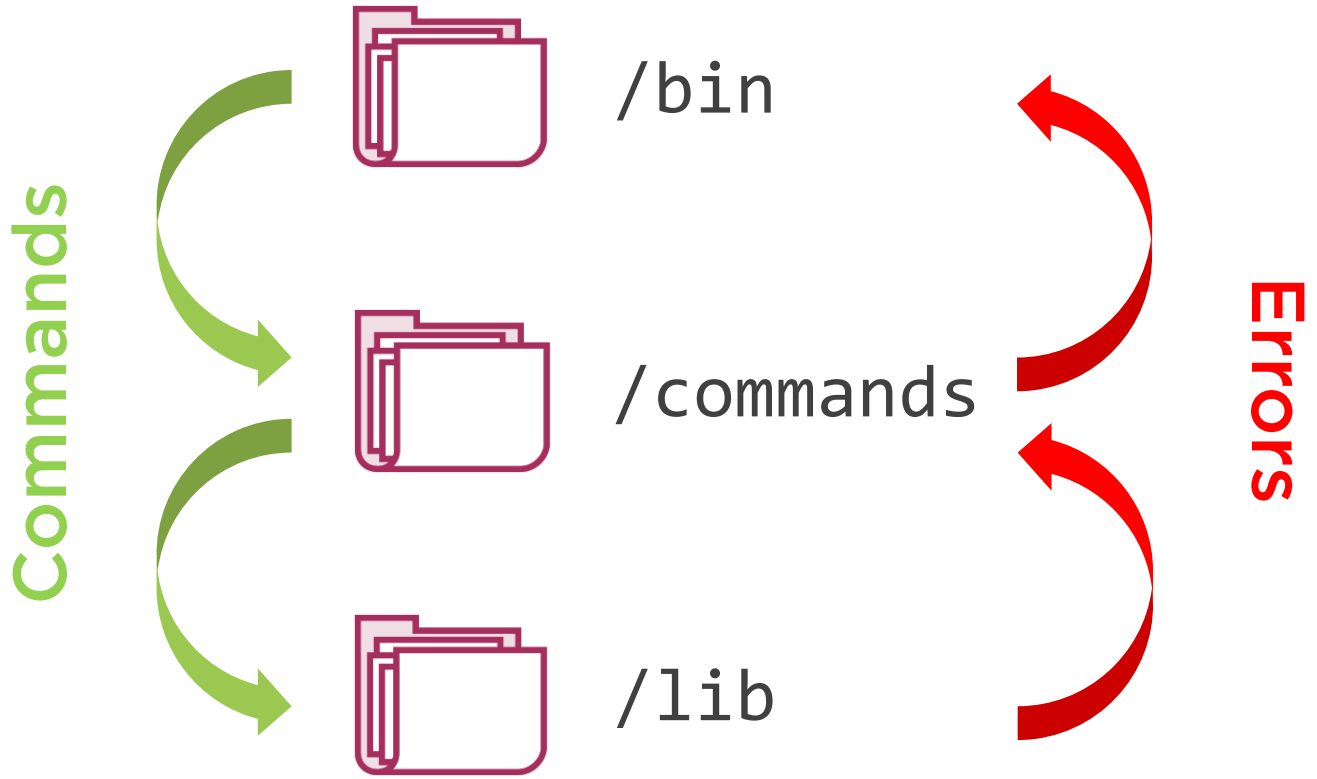
Overview

Handling errors with proper exit statuses

Pattern for using environment variables

Leveraging standard input and output

Handling Errors in Twine



Returning a Proper Exit Status

```
C:\Users\paul>type test.txt
"This is a test"

C:\Users\paul>echo %errorlevel%
0

C:\Users\paul>type foo.txt
The system cannot find the file specified.

C:\Users\paul>echo %errorlevel%
1

C:\Users\paul>
```

```
≡

➜  ~ cat test.txt
This is a test
➜  ~ echo $?
0
➜  ~ cat foo.txt
cat: foo.txt: No such file or directory
➜  ~ echo $?
1
➜  ~
```

Exiting Node.js Correctly on Error

```
// This is an example of what *not* to do:
if (someConditionNotMet()) {
  printUsageToStdout();
  process.exit(1);
}
```

```
// How to properly set the exit code while letting
// the process exit gracefully.
if (someConditionNotMet()) {
  printUsageToStdout();
  process.exitCode = 1;
}
```

https://nodejs.org/dist/latest-v8.x/docs/api/process.html#process_process_exit_code

Demo

Raise errors with the proper messages

Set the correct exit status

Refactor our Credential Manager

Environment Variables

```
C:\>set
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\paul\AppData\Roaming
asl.log=Destination=file
ChocolateyInstall=C:\ProgramData\chocolatey
ChocolateyLastPathUpdate=Tue Nov  7 10:03:39 2017
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=FLEX-LAPTOP
ComSpec=C:\WINDOWS\system32\cmd.exe
configsetroot=C:\WINDOWS\ConfigSetRoot
```

```
$ env
SHELL=/bin/bash
TERM=xterm-256color
USER=paul
NAME=flex-laptop
HOSTTYPE=x86_64
PAGER=less
PWD=/home/paul
LANG=en_US.UTF-8
HOME=/home/paul
LOGNAME=paul
```

Environment Variables as Config Overrides

AWS Documentation » AWS Command Line Interface » User Guide » Configuring the AWS CLI » Environment Variables

Environment Variables

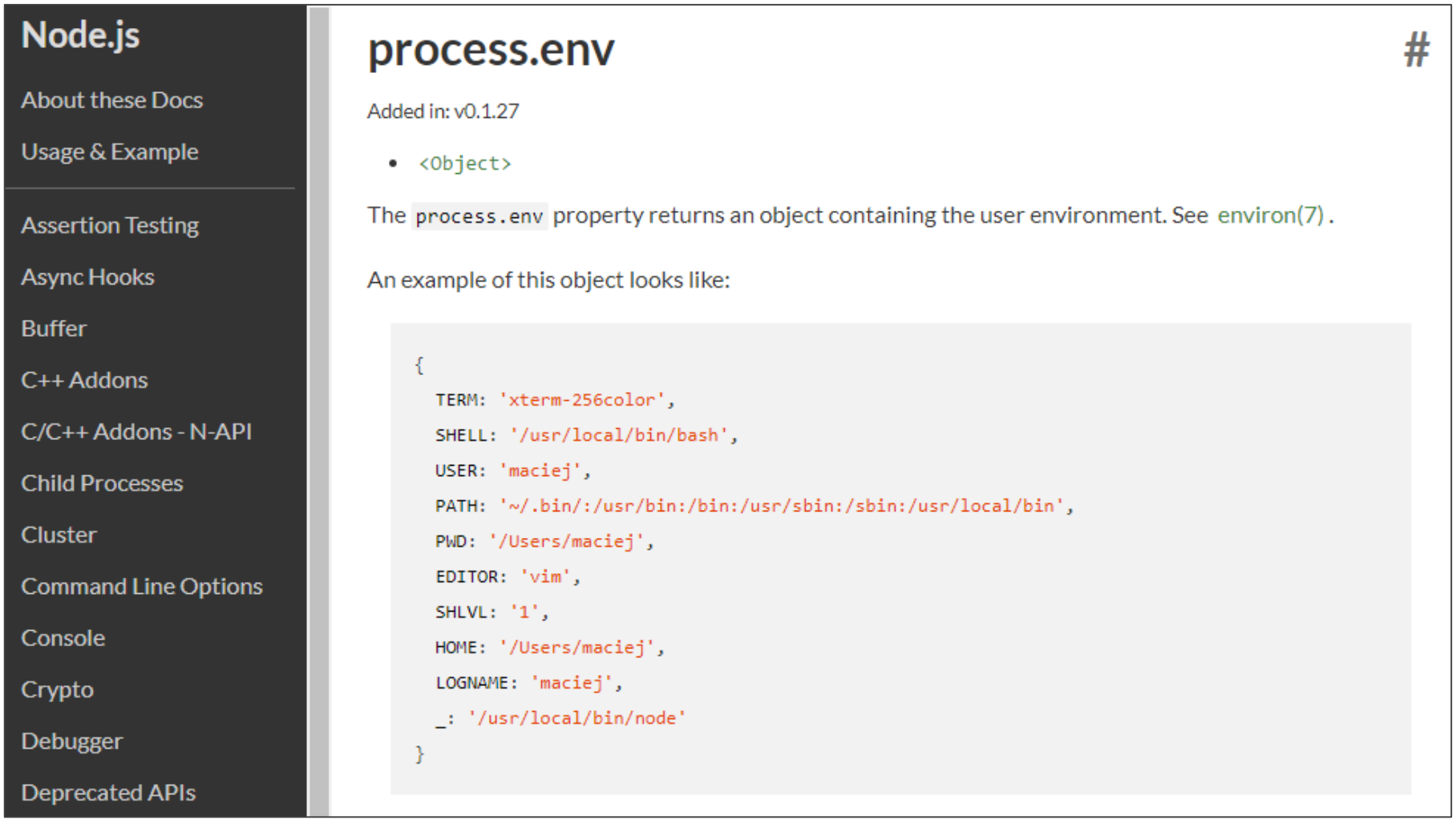
Environment variables override configuration and credential files and can be useful for scripting or temporarily setting a named profile as the default.

The AWS CLI supports the following environment variables.

- `AWS_ACCESS_KEY_ID` – AWS access key.
- `AWS_SECRET_ACCESS_KEY` – AWS secret key. Access and secret key variables override credentials stored in credential and config files.
- `AWS_SESSION_TOKEN` – Specify a session token if you are using temporary security credentials.
- `AWS_DEFAULT_REGION` – AWS region. This variable overrides the default region of the in-use profile, if set.

https://docs.aws.amazon.com/cli/latest/userguide/cli-environment.html

Accessing Environment Variables in Node.js



https://nodejs.org/dist/latest-v8.x/docs/api/process.html#process_process_env

Twine in Three Steps

Consumer

API Key and

Secret

Twitter

Account Token

and Secret

Specific

Twitter API Call

Twine in Three Steps

`TWINE_CONSUMER_KEY`

`TWINE_CONSUMER_SECRET`

Twitter

Account Token

and Secret

Specific

Twitter API Call

Twine in Three Steps

`TWINE_CONSUMER_KEY`

`TWINE_CONSUMER_SECRET`

`TWINE_ACCOUNT_KEY`

`TWINE_ACCOUNT_SECRET`

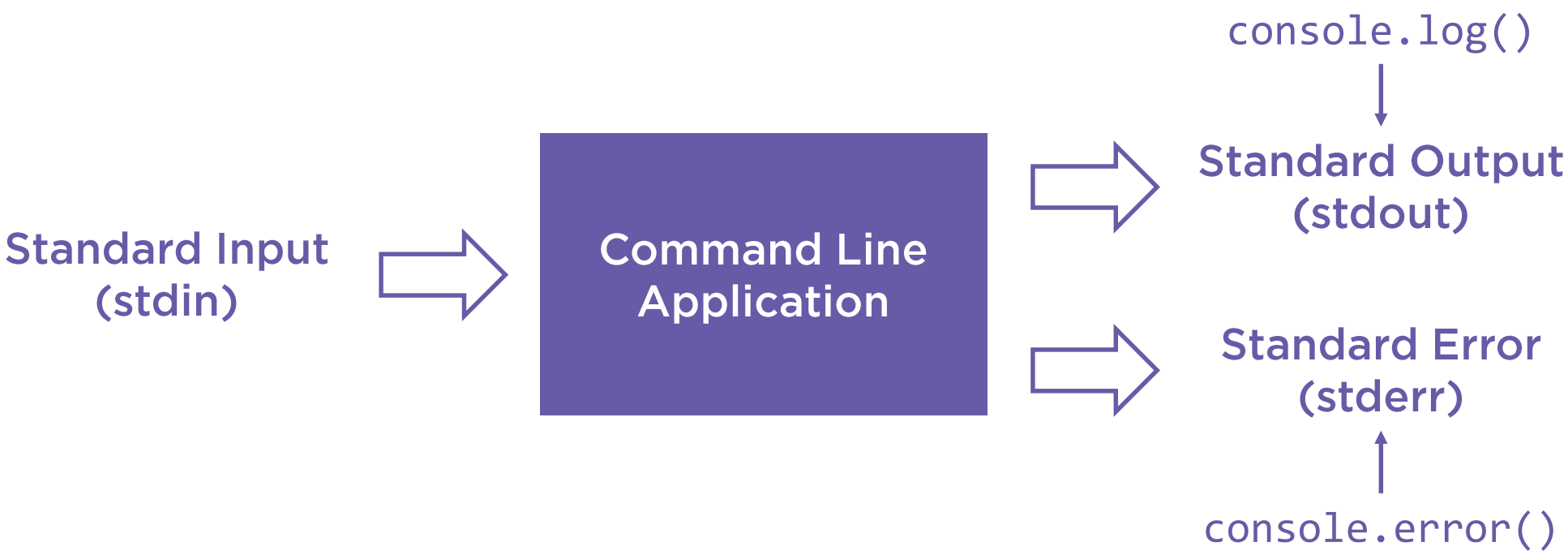
Specific

Twitter API Call

Demo

Support Environment Variable Overrides

Input and Output File Descriptors



Accessing These in Node.js

Standard In: `process.stdin`

Standard Out: `process.stdout`

Standard Error: `process.stderr`

They are streams!

```
process.stdin.pipe(process.stdout)
```

How Can We Leverage This?

Support params both on the command line and read from stdin

A single parameter

```
twine [some command] param
```

```
echo “param” | twine [some command]
```

Multiple parameters

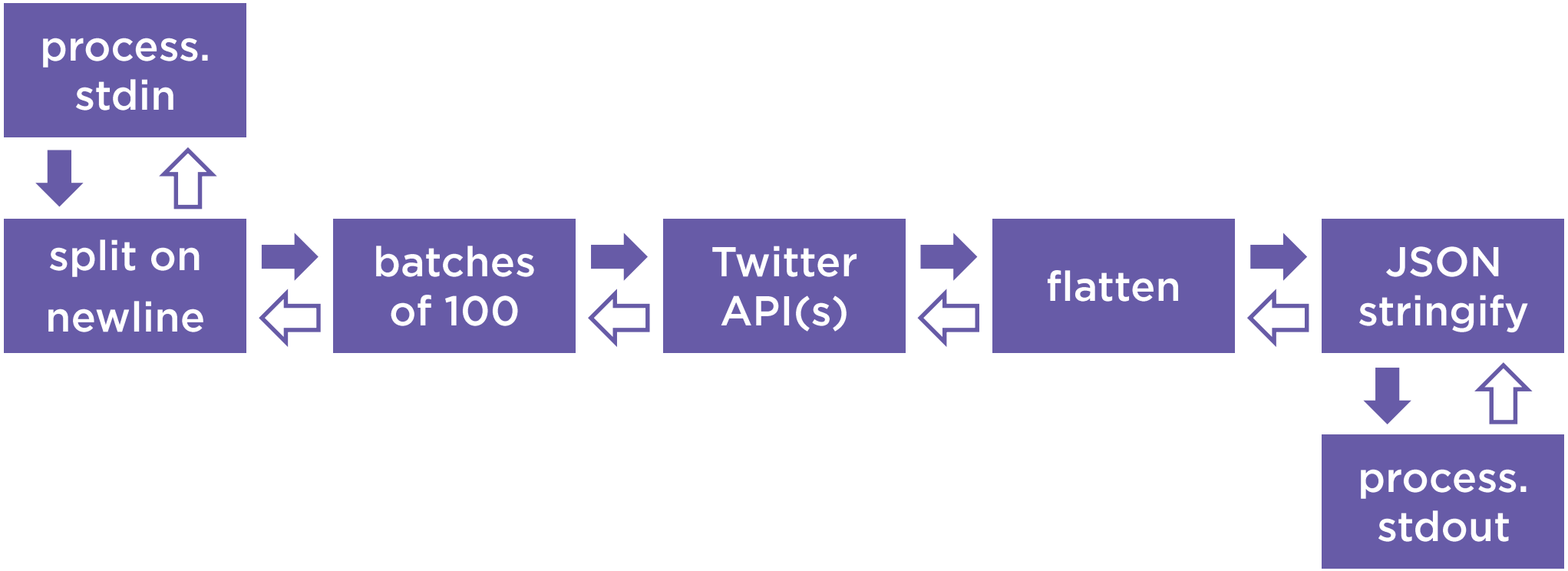
```
twine [some command] param1,param2
```

```
twine [some command] < param-file.txt
```

```
param-generator | twine [some command]
```

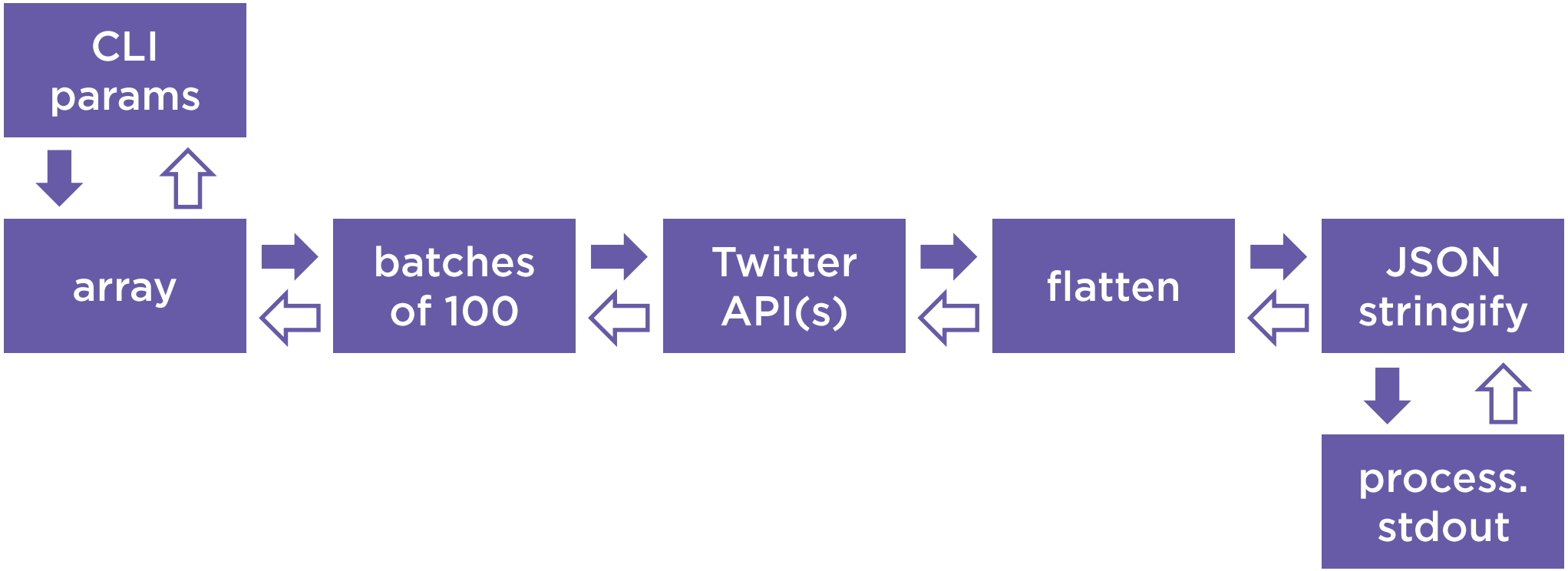
Handling Streaming Input/Output in Twine

```
param-generator | twine lookup users
```



Handling Command Line Input as Well

```
twine lookup users user1,user2
```



Several New Stream-related Modules

split2	https://github.com/mcollina/split2
parallel-transform	https://github.com/mafintosh/parallel-transform
through2	https://github.com/rvagg/through2
JSONStream	https://github.com/dominictarr/JSONStream
from2-array	https://github.com/binocarlos/from2-array
promise-streams	https://github.com/spion/promise-streams

Demo

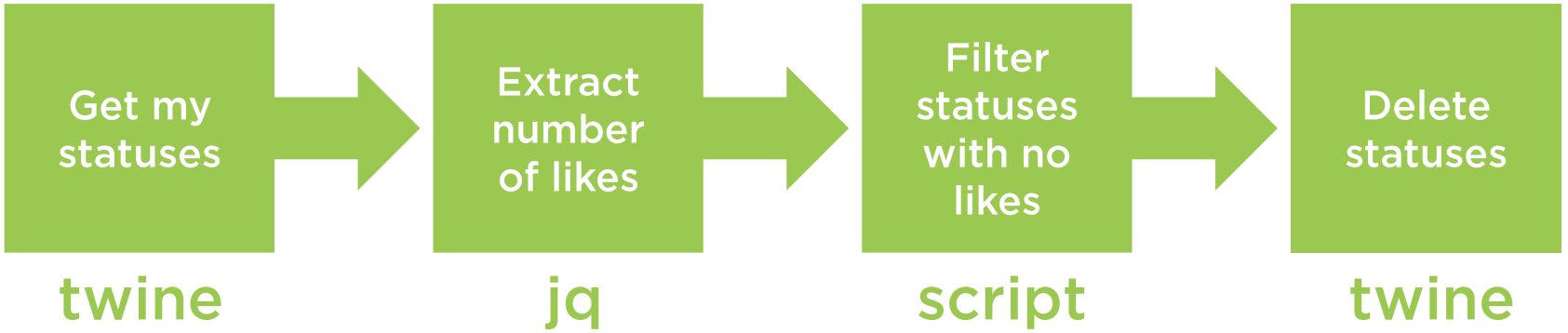
Add “lookup” commands

Support parameters on standard in

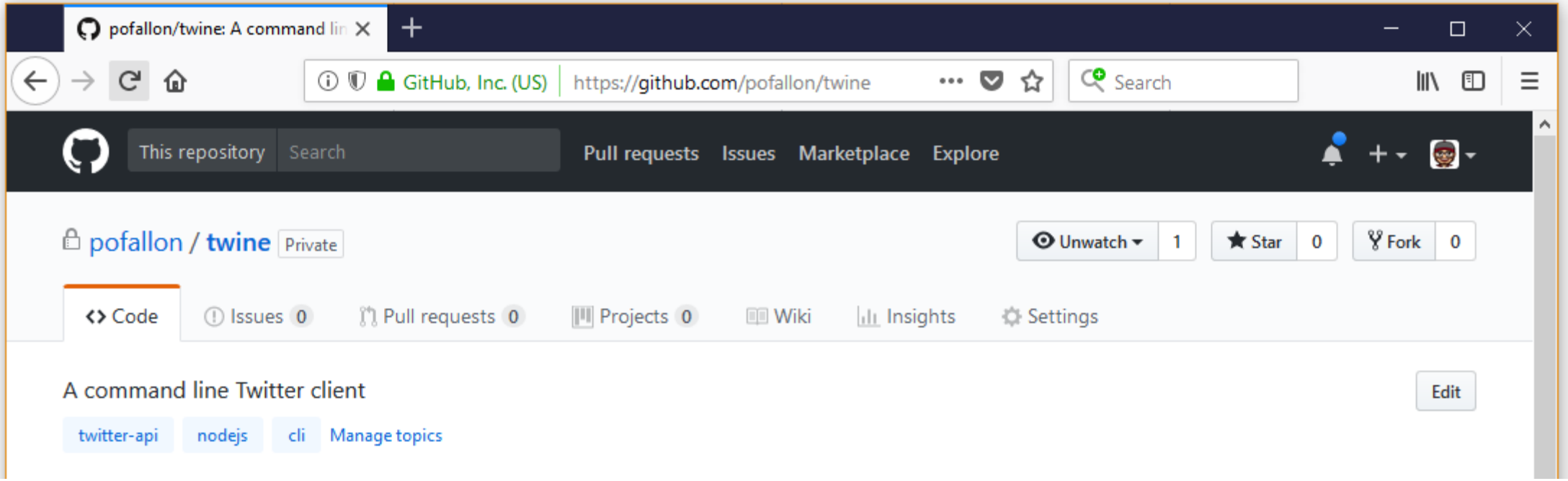
Parse our JSON output

What Did We Just Do?

What Else Could We Do?



Add Some Commands of Your Own!



Summary

Capture and handle errors correctly

Environment variable overrides

Commands that accept input from stdin