

Volo.js:

Command-line love for Web Developers

Scott Davis, ThirstyHead.com



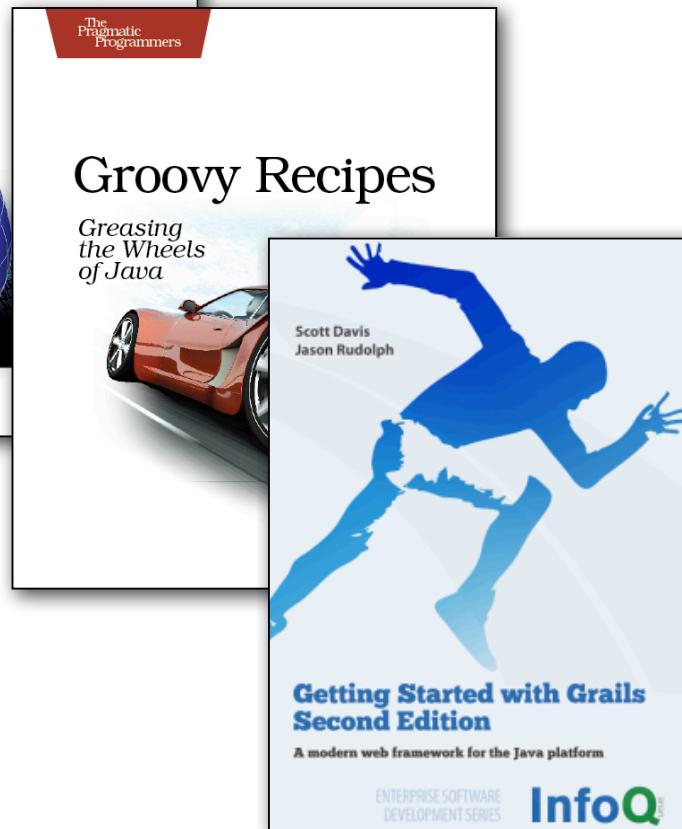
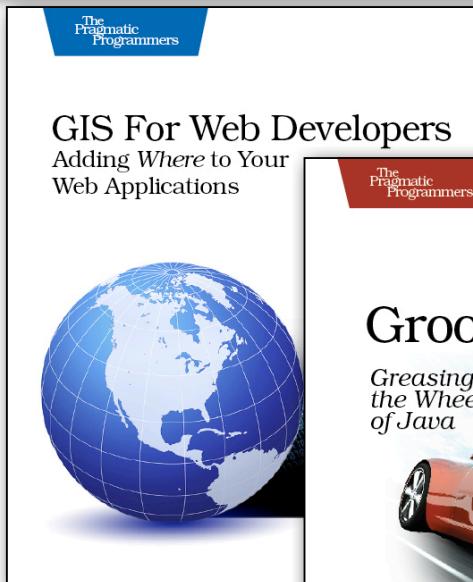
ThirstyHead.com

training done right.



ThirstyHead.com

training done right.



HTML



<bootstrap>



GRAILS

To create the structure for your first Grails application, type `grails create-app racetrack`.

```
$ grails create-app racetrack
Welcome to Grails 1.2 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: /opt/grails
...
Created Grails Application at /web/racetrack
```

You should see a flurry of activity as Grails creates the basic directory structure for your new application.

A screenshot of a web browser window. The title bar says "Ruby on Rails Guides: Getting Started". The address bar shows the URL "guides.rubyonrails.org/getting_started.html#creating-the-blog-application". The main content area displays the following text:

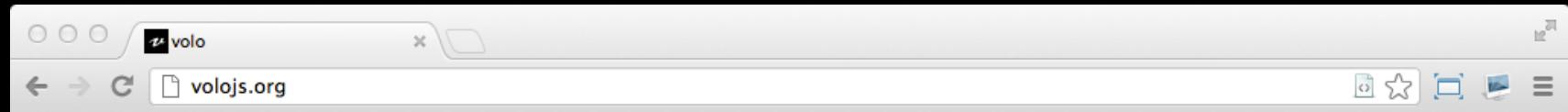
3.2 Creating the Blog Application

To begin, open a terminal, navigate to a folder where you have rights to create files, and type:

```
$ rails new blog
```

This will create a Rails application called Blog in a directory called blog.

Client-side ?



Get started now:

```
npm install -g volo
```

The screenshot shows a web browser window with the title bar "volo" and the address bar "volojs.org". The main content area is titled "Create" and contains instructions for creating a new project. It provides three examples:

- Create 'foo' project with volo's default template `volo create foo`
- Create 'foo' from h5bp/html5-boilerplate v4.0.0 on GitHub `volo create foo h5bp/html5-boilerplate/4.0.0`
- Create 'foo' from zipball URL `volo create foo http://mozilla.github.com/mortar/builds/app-stub.zip`

At the bottom right, there is a link "more about create".

```
bash                                bash                                bash                                bash
volo — bash — 82x24
```

```
~/code/local/volo$ volo create bookstore
Downloading: https://nodeload.github.com/volojs/create-template/zipball/master
https://github.com/volojs/create-template/zipball/master used to create bookstore
~/code/local/volo$ tree bookstore/
bookstore/
├── README.md
├── package.json
├── tools
│   └── build.js
│       └── r.js
└── volofile
    └── www
        ├── index.html
        └── js
            ├── app
            │   └── main.js
            ├── app.js
            └── lib
                └── require.js

5 directories, 9 files
~/code/local/volo$ 
```

RequireJS

requirejs.org



A JAVASCRIPT MODULE LOADER

Home 

Start 

Download 

API 

Optimization 

Use with jQuery 

Use with Node 

Use with Dojo 

CommonJS Notes 

FAQs 

Common Errors 

Writing Plugins 

Why Web Modules 

Why AMD 

Requirements 

History 

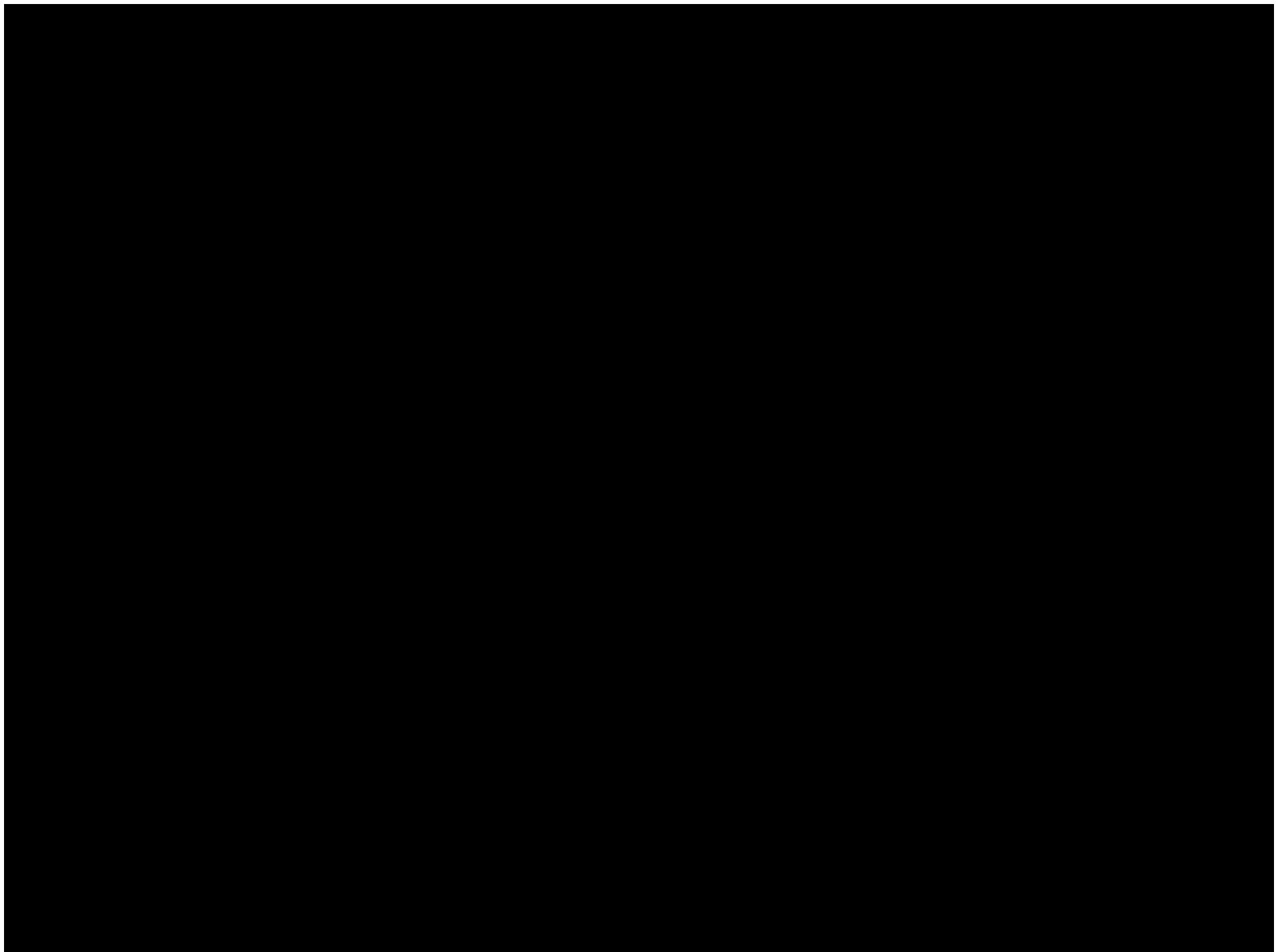
Get Help 

Blog 

```
/* ---  
  
RequireJS is a JavaScript file and module  
loader. It is optimized for in-browser use, but  
it can be used in other JavaScript environments,  
like Rhino and Node. Using a modular script  
loader like RequireJS will improve the speed and  
quality of your code.  
  
IE 6+ ..... compatible ✓  
Firefox 2+ ..... compatible ✓  
Safari 3.2+ ..... compatible ✓  
Chrome 3+ ..... compatible ✓  
Opera 10+ ..... compatible ✓  
  
Get started then check out the API.  
--- */
```



Latest Release: [2.1.1](#)
Open source: [new BSD or MIT licensed](#)
web design by [Andy Chung](#) © 2011



<dependencies>



GRAILS

You could roll up your sleeves and try to write comprehensive search facilities yourself, or you could type `grails install-plugin searchable` and take advantage of someone else's hard work. Since laziness is a virtue among software developers – we call it ***code reuse*** to make it sound better to non-programmers – let's take the latter approach.

There's a lot of console activity when you install a new plugin. Here's a guided tour of what is going on.

```
$ grails install-plugin searchable
...
Reading remote plugin list ...
Plugin list out-of-date, retrieving..
[get] Getting: http://svn.codehaus.org/grails/trunk/grails-
plugins/.plugin-meta/plugins-list.xml
[get] To: /Users/sdavis/.grails/1.2.0/plugins-list-core.xml

Reading remote plugin list ...
[get] Getting: http://plugins.grails.org/.plugin-meta/plugins-
list.xml
[get] To: /Users/sdavis/.grails/1.2.0/plugins-list-default.xml
```



Ruby on Rails: Download rubyonrails.org/download

RubyGems

RubyGems is the standard Ruby package manager. It's similar to apt-get, emerge, and other OS package managers. [Download](#)



Rails

With RubyGems loaded, you can install all of Rails and its dependencies through the command line:

gem install rails

New versions of Rails can be installed the same way.



Home - Chai

chais.com

Chai Assertion Library

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

Download Chai v1.3.0

for Node Another platform? [Browser](#) [Rails](#)

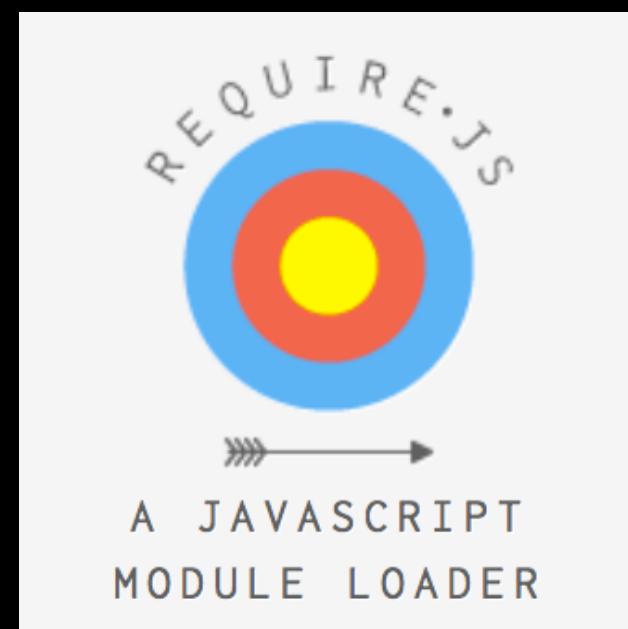
The `chai` package is available on npm.

```
$ npm install chai
```

[View Node Guide](#)

[Issues](#) | [Fork on GitHub](#) | [Changelog](#) | [Google Group](#) | [Build Status](#)

remember ?



The screenshot shows a web browser window with the title bar "volo" and the address bar "volojs.org". The main content area is titled "Add" and contains instructions for adding JavaScript libraries. It lists four examples with their corresponding command-line syntax:

- Queries github for 'jquery', then downloads latest version `volo add jquery`
- Adds backbone as well as underscore and jquery `volo add backbone`
- Uses [semver](#) to add latest 2.x version of requirejs `volo add requirejs/~2`
- Add the amdjs github group's version of backbone `volo add amdjs/backbone`

At the bottom right of the content area is a link: "more about add".

```
bookstore — bash — 82x24
bash                                bash                                bash                                bash
~/code/local/volo/bookstore$ volo add jquery
Using github repo "jquery/jquery" for "jquery"...
Downloading: http://code.jquery.com/jquery-1.8.2.js
Installed github:jquery/jquery/1.8.2 at www/js/lib/jquery.js
AMD dependency name: jquery
~/code/local/volo/bookstore$ tree
.
├── README.md
├── package.json
├── tools
│   └── build.js
│       └── r.js
└── volofile
    └── www
        ├── index.html
        └── js
            ├── app
            │   └── main.js
            ├── app.js
            └── lib
                ├── jquery.js
                └── require.js

5 directories, 10 files
```

package.json

```
{  
  "amd": { } ,  
  "volo": {  
    "baseUrl": "www/js/lib",  
    "dependencies": {  
      "jquery": "github:jquery/jquery/1.8.2"  
    }  
  }  
}
```

```
jquery/package.json at ee96 · GitHub, Inc. [US] https://github.com/jquery/jquery/blob/master/package.json
```

```
1  {
2      "name": "jquery",
3      "title": "jQuery",
4      "description": "JavaScript library for DOM operations",
5      "version": "1.8.3pre",
6      "homepage": "http://jquery.com",
7      "author": {
8          "name": "jQuery Foundation and other contributors",
9          "url": "https://github.com/jquery/jquery/blob/master/AUTHORS.txt"
10     },
11     "repository": {
12         "type": "git",
13         "url": "https://github.com/jquery/jquery.git"
14     },
15     "bugs": {
16         "url": "http://bugs.jquery.com"
17     },
18     "licenses": [
19         {
20             "type": "MIT",
21             "url": "https://github.com/jquery/jquery/blob/master/MIT-LICENSE.txt"
22         }
23     ],
24     "dependencies": {},
25     "devDependencies": {
26         "grunt-compare-size": "~0.2.0",
27         "grunt-git-authors": ">=1.0.0",
28         "grunt-update-submodules": ">=0.1.2",
29         "grunt": "~0.3.17",
30         "testswarm": "0.2.2"
31     },
32     "keywords": []
33 }
```



jQuery Blog » jQuery 1.7 Released

blog.jquery.com/2011/11/03/jquery-1-7-released/

Asynchronous Module Definition (AMD)

jQuery [now supports](#) the [AMD API](#). Note that jQuery 1.7 is *not* a script loader itself; it cooperates with AMD-compliant loaders such as RequireJS or curl.js so it can be loaded dynamically and the `ready` event can be controlled by the loader. Now an AMD-compliant loader can load an unmodified version of jQuery 1.7 from a CDN such as Google's or Microsoft's. Many thanks to James Burke (@jrburke) for submitting the patch and unit tests, then waiting patiently for us to incorporate it.

The screenshot shows a web browser window with the following details:

- Title Bar:** AMD · amdjs/amdjs-api Wiki
- Address Bar:** GitHub, Inc. [US] <https://github.com/amdjs/amdjs-api/wiki/AMD>
- Toolbar Buttons:** New Page, Edit Page, Page History

The main content area displays the following text and sections:

AMD

The Asynchronous Module Definition (**AMD**) API specifies a mechanism for defining modules such that the module and its dependencies can be asynchronously loaded. This is particularly well suited for the browser environment where synchronous loading of modules incurs performance, usability, debugging, and cross-domain access problems.

It is unrelated to the technology company [AMD](#) and the processors it makes.

- [Tests](#)
- [Discussion](#)

API Specification

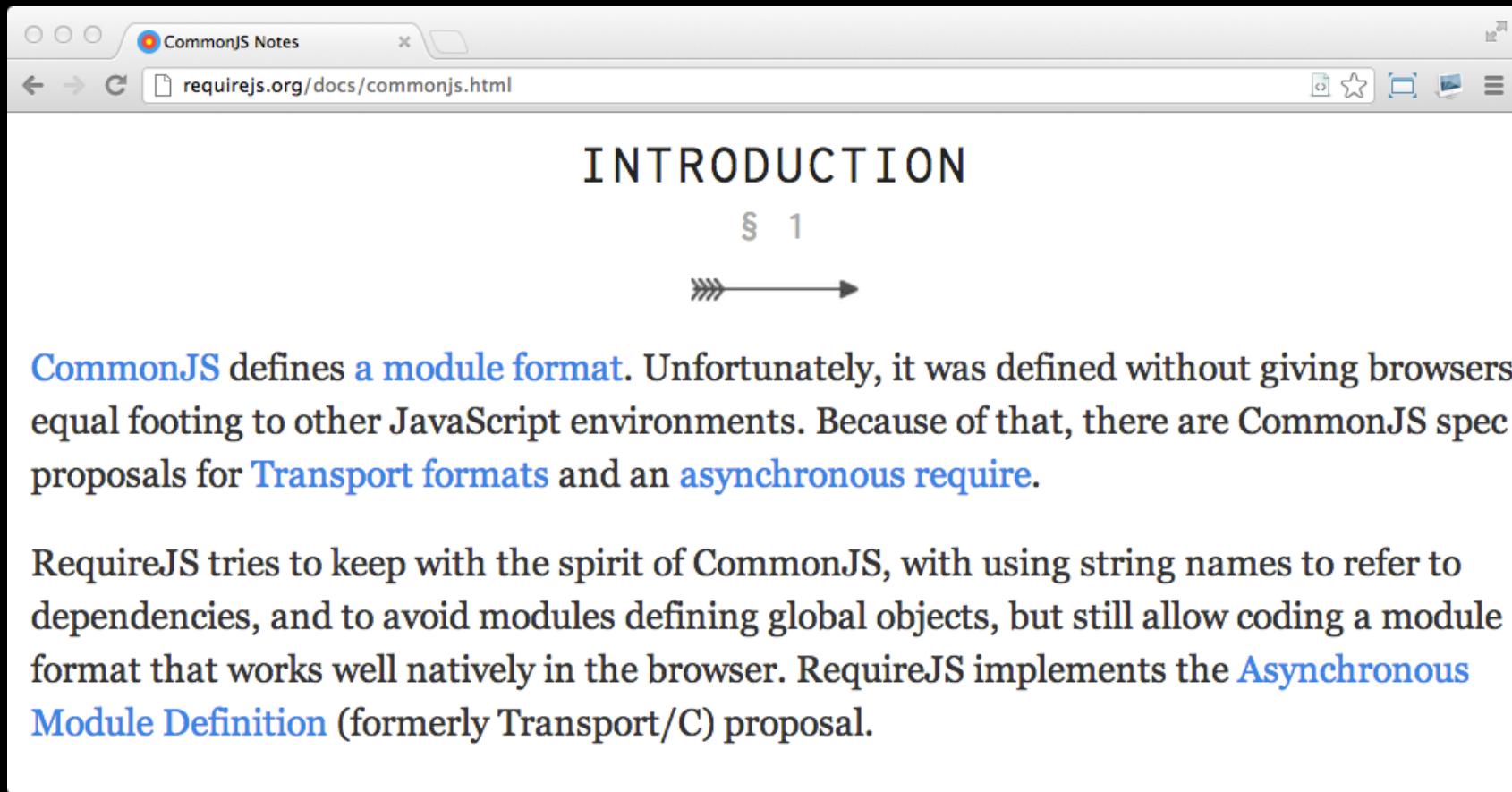
Using require and exports

Sets up the module with ID of "alpha", that uses require, exports and the module with ID of "beta":

```
define("alpha", ["require", "exports", "beta"], function (require, exports, beta) {
    exports.verb = function() {
        return beta.verb();
        //Or:
        return require("beta").verb();
    }
});
```

CommonJS == Synchronous, Server-Side (Node.js)

AMD == Asynchronous, Client-Side (Browser)



The screenshot shows a web browser window titled "CommonJS Notes". The address bar displays "requirejs.org/docs/commonjs.html". The main content area features a large, bold title "INTRODUCTION" centered at the top. Below it is a subtitle "§ 1" followed by a horizontal navigation bar with three arrows pointing right. The text below the title discusses the history and evolution of CommonJS, mentioning its synchronous nature and the lack of equal footing for browsers. It also highlights proposals for transport formats and asynchronous require statements. The text concludes by noting that RequireJS aims to maintain the spirit of CommonJS while adapting it for the browser environment.

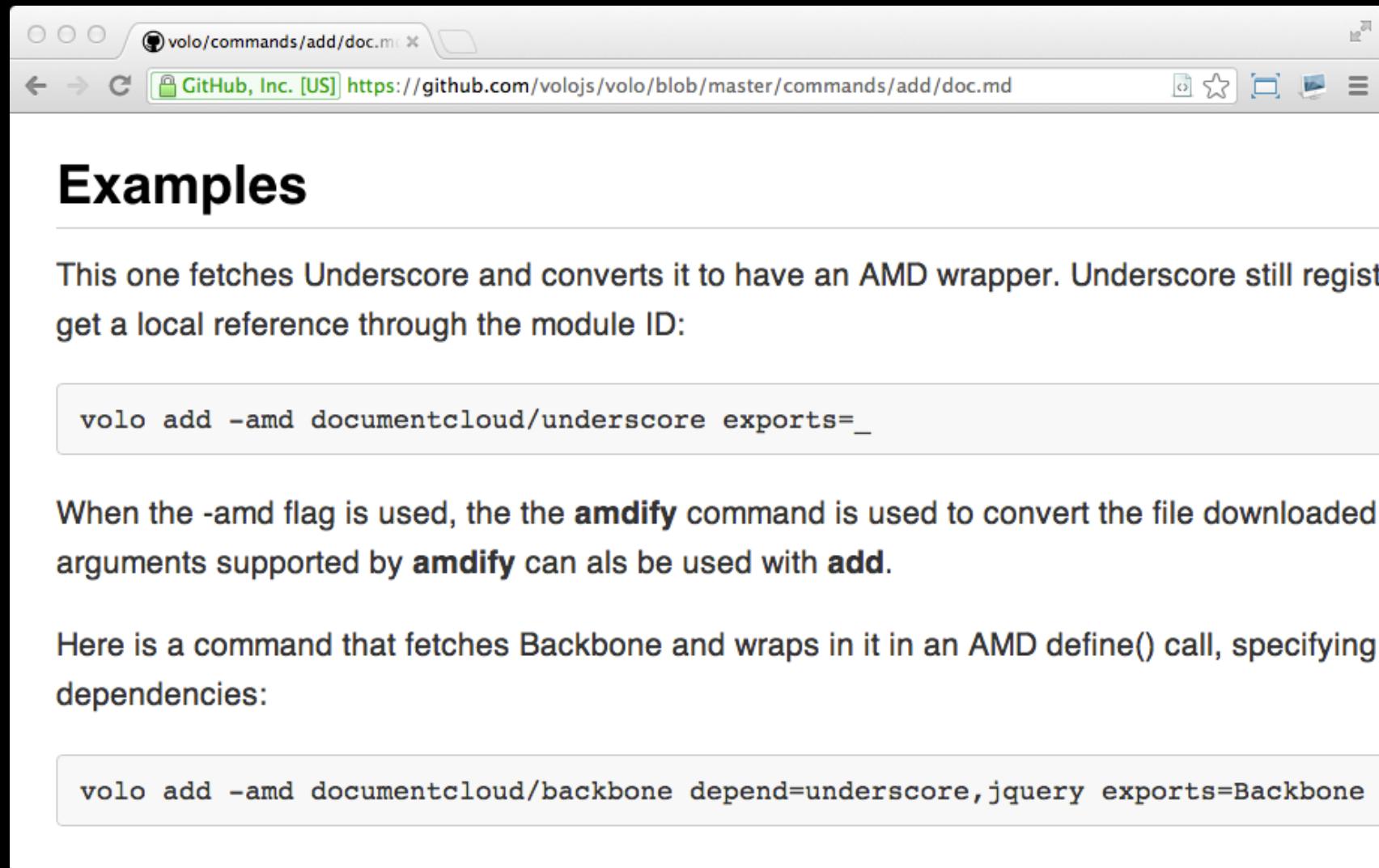
CommonJS defines a [module format](#). Unfortunately, it was defined without giving browsers equal footing to other JavaScript environments. Because of that, there are CommonJS spec proposals for [Transport formats](#) and an [asynchronous require](#).

RequireJS tries to keep with the spirit of CommonJS, with using string names to refer to dependencies, and to avoid modules defining global objects, but still allow coding a module format that works well natively in the browser. RequireJS implements the [Asynchronous Module Definition](#) (formerly Transport/C) proposal.

The screenshot shows a web browser window with the following details:

- Title Bar:** "Tagneto: AMD support for Underscore and Backbone" is displayed in the title bar.
- Address Bar:** The URL "tagneto.blogspot.com/2012/01/amd-support-for-underscore-and-backbone.html" is shown.
- Toolbar:** Standard browser toolbar icons for back, forward, search, and refresh are visible.
- Content Area:**
 - Section Header:** "Tagneto" is the main title in a large, bold, black serif font.
 - Text:** "Notes on JavaScript, RequireJS, and Mozilla Labs." is written below the main title.
 - Date:** "Sunday, January 22, 2012" is displayed in a greenish-grey font.
 - Section Header:** "AMD support for Underscore and Backbone" is the main heading for the article, written in a large, bold, red font.
 - Text:** "As mentioned previously, at the moment there is no support for AMD in Underscore, a decision that also affects Backbone." is the first paragraph of the article, written in black text.
 - Text:** "However, there have been enough developers interested in using Underscore and Backbone with AMD that forks were set up under the amdjs organization that have optional define() calls in them:" is the second paragraph of the article, written in black text.
 - Links:** Two links are shown in red:
 - "<https://github.com/amdjs/underscore>
 - "<https://github.com/amdjs/backbone>

Volo.js can “amd-ify” libs...



The screenshot shows a web browser window with the URL <https://github.com/volojs/volo/blob/master/commands/add/doc.md>. The page content is as follows:

Examples

This one fetches Underscore and converts it to have an AMD wrapper. Underscore still registers itself as a global variable, but you can also get a local reference through the module ID:

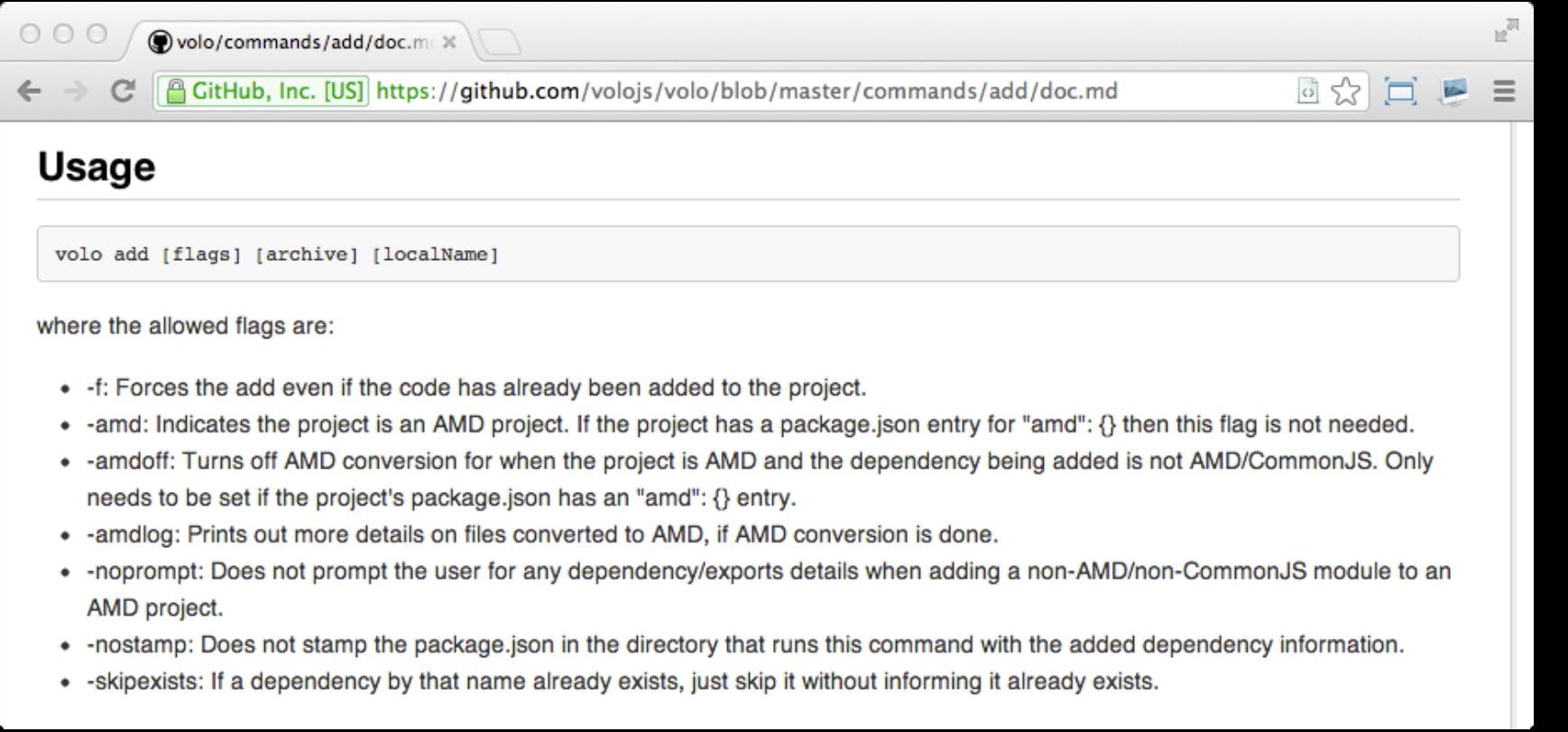
```
volo add -amd documentcloud/underscore exports=_
```

When the `-amd` flag is used, the `amdify` command is used to convert the file downloaded. Most arguments supported by `amdify` can also be used with `add`.

Here is a command that fetches Backbone and wraps in it in an AMD `define()` call, specifying dependencies:

```
volo add -amd documentcloud/backbone depend=underscore,jquery exports=Backbone
```

...or it can ignore AMD
with “ -amloff ”
and just download the lib



The screenshot shows a web browser window with the URL <https://github.com/volojs/volo/blob/master/commands/add/doc.md>. The page content is as follows:

Usage

```
volo add [flags] [archive] [localName]
```

where the allowed flags are:

- `-f`: Forces the add even if the code has already been added to the project.
- `-amd`: Indicates the project is an AMD project. If the project has a package.json entry for "amd": {} then this flag is not needed.
- `-amloff`: Turns off AMD conversion for when the project is AMD and the dependency being added is not AMD/CommonJS. Only needs to be set if the project's package.json has an "amd": {} entry.
- `-amldlog`: Prints out more details on files converted to AMD, if AMD conversion is done.
- `-noprompt`: Does not prompt the user for any dependency/exports details when adding a non-AMD/non-CommonJS module to an AMD project.
- `-nostamp`: Does not stamp the package.json in the directory that runs this command with the added dependency information.
- `-skipexists`: If a dependency by that name already exists, just skip it without informing it already exists.

The screenshot shows a GitHub wiki page for the repository 'volojs/volo'. The page title is 'Library best practices'. The top navigation bar includes links for 'Code', 'Network', 'Pull Requests (0)', 'Issues (20)', 'Wiki' (which is active), and 'Graphs'. Below the navigation bar are links for 'Home', 'Pages', 'Wiki History', and 'Git Access'. On the right side of the page are buttons for 'New Page', 'Edit Page', and 'Page History'. The main content area contains the following text:

Systems work better when there are good default conventions. Configuration should be possible, but for the 80% case, the conventions should be used.

volo uses the following conventions. The more you adhere to them, the easier it will be for others to consume your code.

There are two basic types of libraries: libraries that are a single JS file (jquery), and libraries that are a collection of modular scripts (dojo), and they have slightly different guidelines.

- Single JS file library
 - Example
- Module collection
 - Main module example
 - Independent module example
- package.json
 - amd
 - volo.baseUrl
 - volo.url
 - volo.dependencies
 - volo.type
- Minimizing GitHub
 - Just publish a ZIP file
 - Just publish a JS file
 - package.json in GitHub

The screenshot shows a web browser window with the following details:

- Title Bar:** Library best practices · volojs · GitHub
- Address Bar:** GitHub, Inc. [US] https://github.com/volojs/volo/wiki/Library-best-practices
- Content Area:**
 - ## Minimizing GitHub
 - If you prefer to not use GitHub to develop your code, there are few options that can still be used by default with volo.
 - ### Just publish a ZIP file

Let everyone know where to find a .zip file of your dependency. It can be added to volo via:

```
volo add http://your.domain.com/path/to/your/lib.zip
```

This approach does not make it easy to find the .zip file via `volo search` or when `volo add query` is used.
 - ### Just publish a JS file

Let everyone know the URL to your library file. It can be added to a project via this volo command:

```
volo add http://your.domain.com/path/to/your/lib.js
```

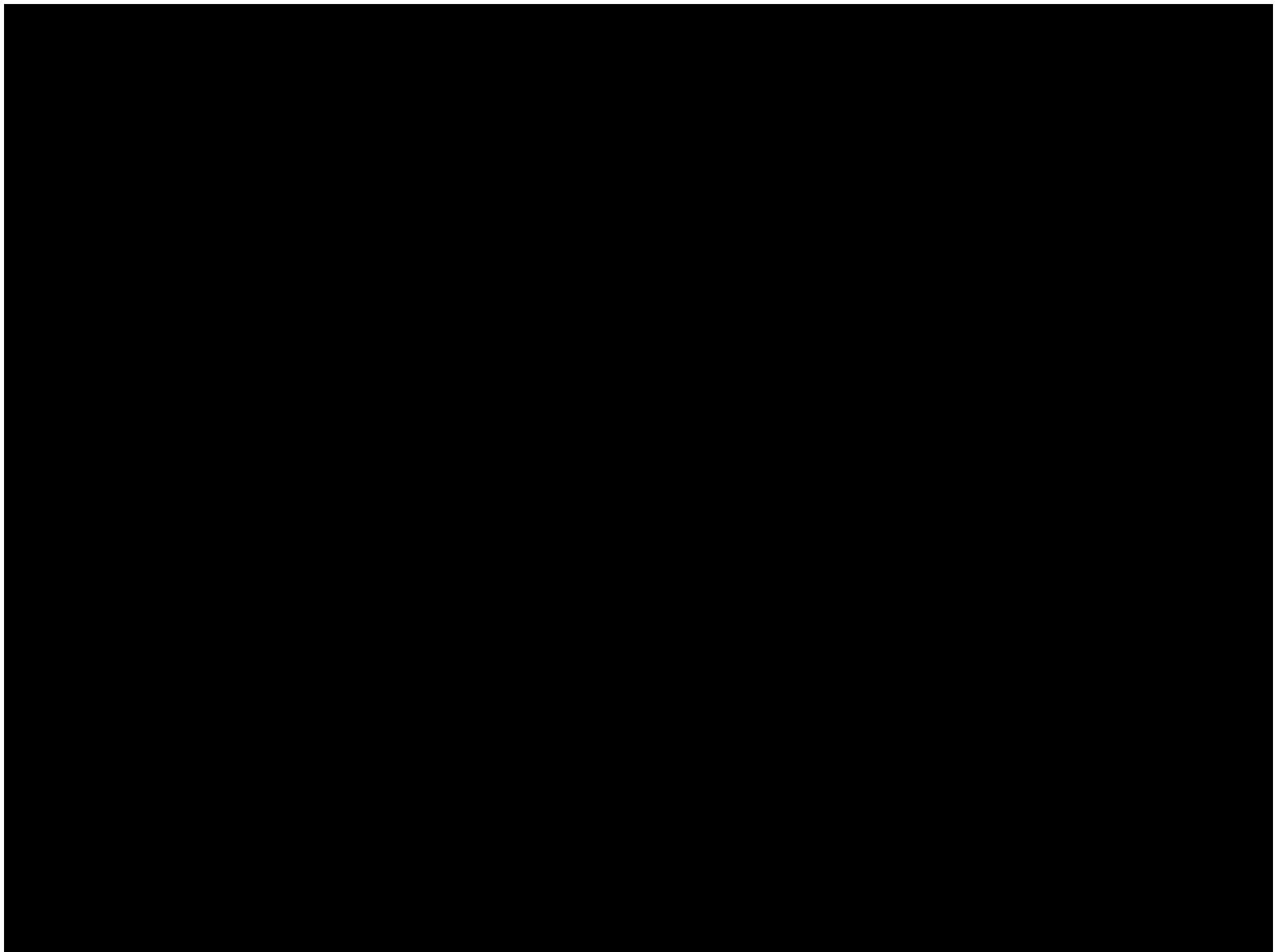
This approach does not make it easy to find the .zip file via `volo search` or when `volo add query` is used.
 - ### package.json in GitHub

Only place a package.json file in GitHub. Set `volo.url` to indicate where to find the file. Example:

```
{  
  "volo": {  
    "url": "http://your.domain.com/path/to/1.2.3/lib.js"  
  }  
}
```

Update the the package.json whenever you have a release.

It is also helpful to put in a README.md file that describes where your source is hosted and how to interact with your project.



<extras>

The screenshot shows a web browser window with the URL `volojs.org` in the address bar. The page content is as follows:

Automate

Quickly code up project automation using a volofile and JavaScript. Reuse automation commands by publishing/installing from npm.

Simple automation: Create a `volofile` in your project's top-level directory. This file is just a node module that defines a property for each automation task. npm install reusable command line functions.

```
npm install local tools to use in volofile      npm install jshint uglify-js
```

Simple volofile that cleans, lints and uglifies

```
module.exports = {  
  clean: {  
    summary: 'removes a.min.js',  
    run: 'v.rm a.min.js'  
  },  
  lint: {  
    summary: 'runs jshint on a.js',  
    run: 'n.jshint a.js'  
  },  
  uglify: {  
    summary: 'minifies a.js to a.min.js',  
    depends: ['clean', 'lint'],  
    run: 'n.uglifyjs -o a.min.js a.js'  
  }  
};
```

[more about volofiles](#)

The screenshot shows a web browser window with the title bar "volo" and the address bar "volojs.org". The main content area displays a section about managing appcache. It includes instructions for installing the command-line tool, adding it to a volofile, and generating the appcache with a single command. A code snippet for a volofile entry is shown, along with the command to run it.

Manage the appcache: Make or update an appcache with one simple command. Never worry about adding files or changing the appcache version again.

Install the reusable volofile command, `volo-appcache`

```
npm install volo-appcache
```

Add an entry to your project's volofile to use it

```
module.exports = {
  appcache: require('volo-appcache')({
    dir: 'www-built'
  })
}
```

Now generating the appcache is a single command

```
volo appcache
```

[more about volo-appcache](#)

The screenshot shows a web browser window with the URL `volojs.org` in the address bar. The page content is as follows:

Deploy straight to github: If your site is all client-side, you can publish to github with one command. Your site is live within seconds.

Install the reusable volofile command, `volo-ghdeploy`

```
npm install volo-ghdeploy
```

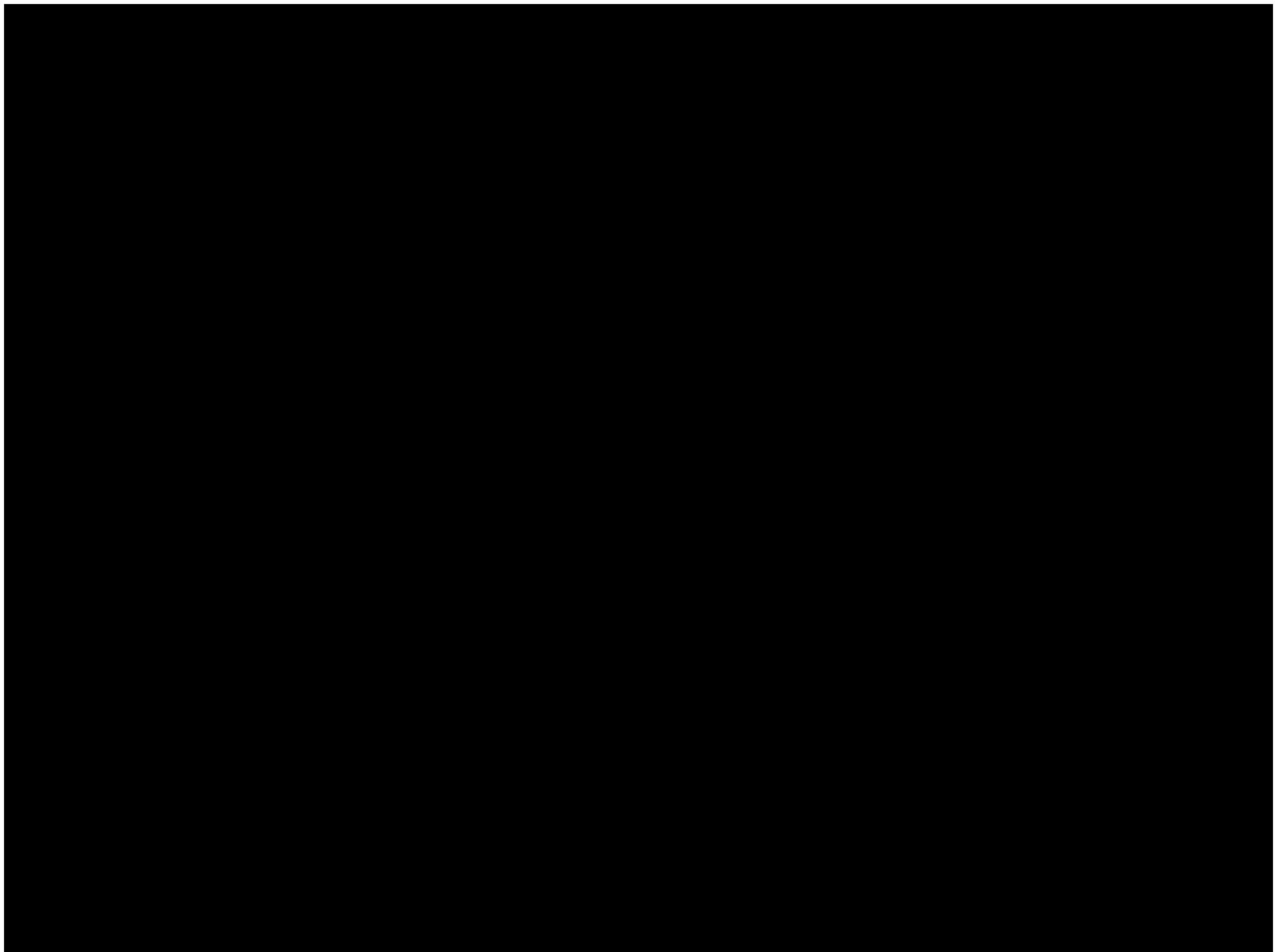
Add an entry to your project's volofile to use it

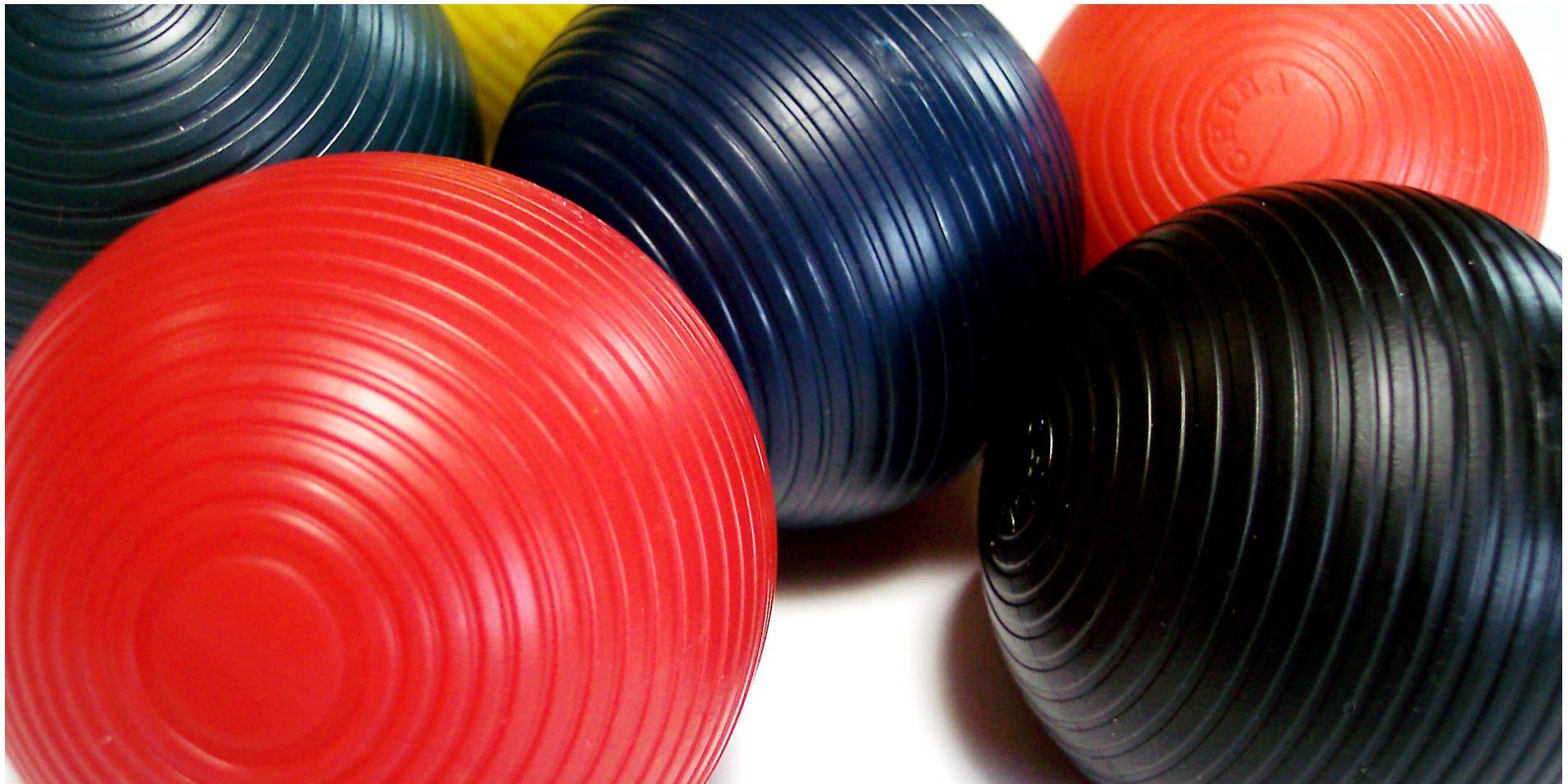
```
module.exports = {
  ghdeploy: require('volo-ghdeploy')('built',
'deployed')
}
```

Now deploying to GitHub is a single command

```
volo ghdeploy
```

[more about volo-ghdeploy](#)





Volo.js:

Command-line love for Web Developers

Scott Davis, ThirstyHead.com



scott@thirstyhead.com



Scott Davis
@scottdavis99

Questions?
Thanks for your time.

source and slides:

<http://scottdavis99.github.com/>

Image Credits

All images courtesy of <http://morgueFile.com>