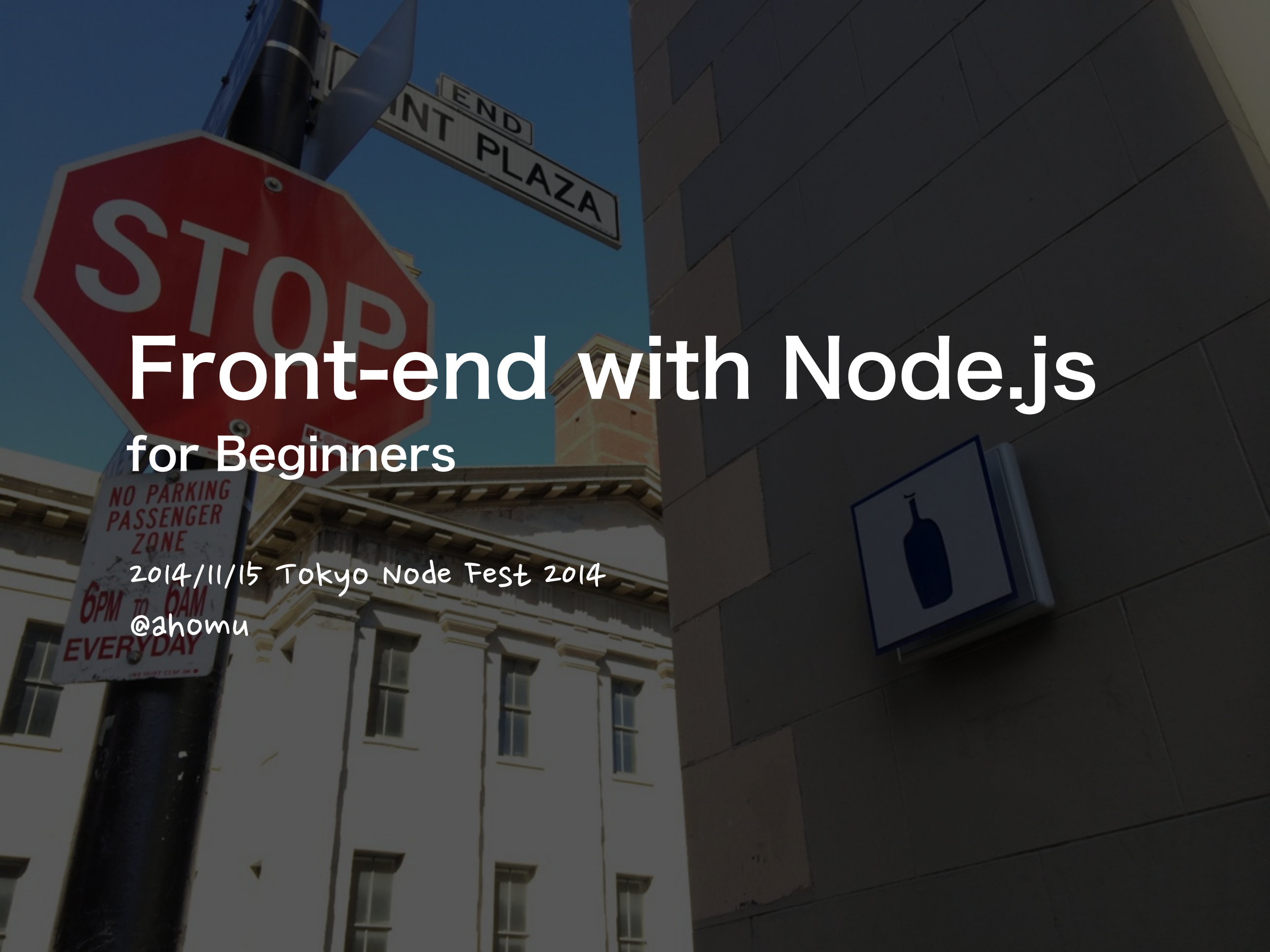


Front-end with Node.js

for Beginners

2014/11/15 Tokyo Node Fest 2014

@ahomu

Front-end with Node.js

for Beginners ← 重要!!

2014/11/15 Tokyo Node Fest 2014

@ahomu

さとう　あゆむ

佐藤　歩

本名

@ahomu

年間維持費　￥8,480

http://aho.mu


Software Design plus
HTML・CSS・JavaScriptの基本から現場で役立つ技術まで満載！
10年力つくる
フロントエンドエンジニア養成読本
Front-End Engineer
サービス品質価値を左右する
パフォーマンス入門
実践CSS レイアウトパターン、設計、プリプロセッサ
Web環境の変化に耐える
JavaScript
サイバーエージェントの現役エンジニアが必須スキルを伝授
●Webブラウザの基礎知識 ●マークアップテクニックレシピ ●テスト
●UI/UXデザイン入門 ●バージョン管理 ●モバイル・マルチデバイス対応 ●セキュリティ


Webアプリケーション開発のためのプログラミング技術情報誌
FOR ALL WEB APPLICATION DEVELOPERS ウェブDBプレス
WEB+DB PRESS
モバイルアプリからもWebアプリからも使いやすい!
vol.82 2014
Web API
デザインの鉄則
▶RESTに沿った設計 ▶明確なデータ構造 ▶適切なエラー
はじめてのGo
シンプルな言語仕様、型システム、並行処理
クラウド会計ソフトfreee
開発ノウハウ大公開
Hubot／Web Components
Java 8を活かすJVMオプション
Perlの構文解析器の作り方


Webアプリケーション開発のためのプログラミング技術情報誌
FOR ALL WEB APPLICATION DEVELOPERS ウェブDBプレス
WEB+DB PRESS
メンバーの多様性を活かし、未知の問題に対処する
vol.83 2014
強いチームの作り方
意見の衝突を恐れない!!
実践 写真があれば検索できる!
画像認識
多機能シェルで日々の作業を省力化
zsh大活用
もっと知りたい! Markdown
Google BigQueryでビッグデータ解析
データベース設計アンチパターン

Front-end Tooling

かつてのフロントエンド開発

いいかい？
仕事をするには
たくさんの事を覚えな
てはいけない

公社に連れて来られた子供は
体を改造されると同時に
〝条件付け〟をほどこされる

彼女にとって幸いだったのは
たった一つ条件付けの結果
〝記憶が消されること〟だ

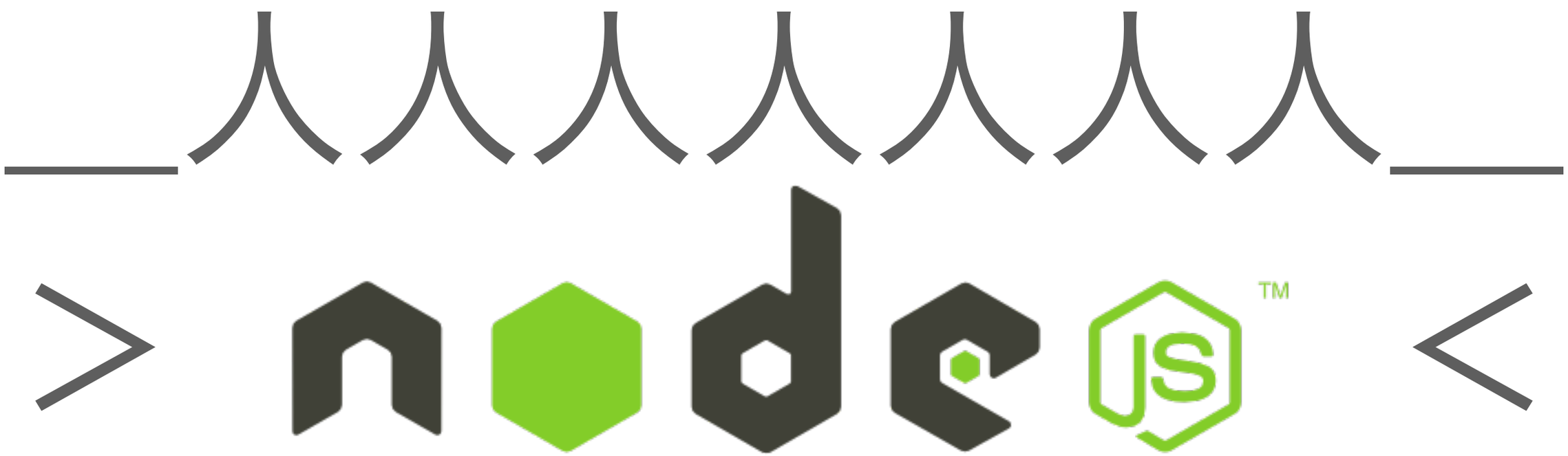
良い仕事は全て単純な
作業の堅実な積み重ねだ

メモを作るから
毎日勉強するように

良い仕事は全て単純な作業の堅実な積み重ね

地道な手作業の連続

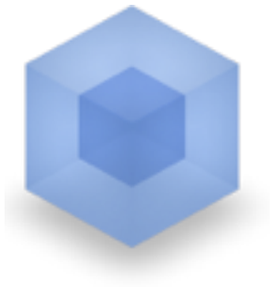
自動化・効率化する手段


node
JS
TM

1.パッケージマネージャー

2. タスクランナー

3. モジュールシステム



Package Manager

パッケージマネージャー

Before

配布サイトを開く

ダウンロードする

作業ディレクトリにコピー

After

コマンド叩いてインストール

```
% npm install normalize.css
```

```
% bower install jquery
```

Node land

express

CoffeeScript

GRUNT

koa

Browser land

jQuery
write less, do more.

ANGULARJS
by Google

BACKBONE.JS

Node land

Browser land

npm

Node.js標準の
パッケージマネージャ

Bower

フロントエンド向けの
パッケージマネージャ

```
% npm install -g bower
```

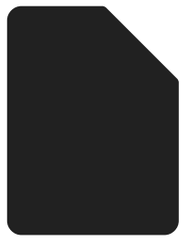
package.json

```
{
  "name": "best-practices",
  "version": "1.0.0",
  "description": "package using versioning practices",
  "author": "Charlie Robbins <charlie@nodejitsu.com>",
  "main": 'index.js',
  "dependencies": {
    "colors": "0.x.x",
    "express": "2.3.x",
    "optimist": "0.2.x"
  },
  "devDependencies": {
    "vows": "0.5.x"
  },
  "engine": "node >= 0.4.1"
}
```

bower.json

```
{
  "name": "my-project",
  "version": "1.0.0",
  "main": "path/to/main.css",
  "ignore": [
    ".jshintrc",
    "**/*.txt"
  ],
  "dependencies": {
    "<name>": "<version>",
    "<name>": "<folder>",
    "<name>": "<package>"
  },
  "devDependencies": {
    "<test-framework-name>": "<version>"
  }
}
```

2つのリポジトリと設定ファイル



package.json

```
% npm install
```

bower.json

```
% bower install
```

The npm Blog

Blog about npm things.

npm and front-end packaging

We've known for a while that front-end asset and dependency management is a huge use-case for npm and a big driver of Node.js adoption in general. But how big, exactly? It's a hard question to answer. The list of most-downloaded packages on npm is not very helpful: packages like async, minimist and request are the bread-and-butter packages that are depended upon by thousands of other packages, so of course they get installed and downloaded all the time as part of the installations of those packages.

A more interesting and revealing question is: what packages do people *explicitly* install? By which we mean, how many times did somebody (or some robot) actually run the command `npm install thispackage`? We recently started plugging our log data into Jut, which has made it easy and quick to answer these questions for the first time. The resulting list of the top 50 explicitly-installed npm packages is very different and very interesting. 32% of the packages in the top 50 (and 50% of the actual downloads) are front-end tools or frameworks, with Grunt, Bower and Gulp leading the pack (mobile is also a huge use-case, and we'll be talking about it in a later blog post). Plus, usage of all these packages is growing steadily:

Node land

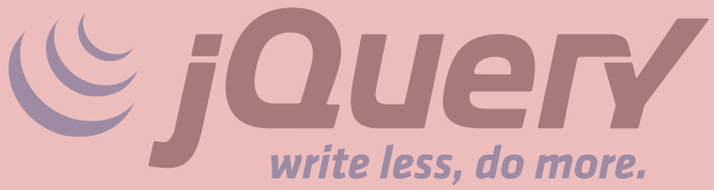
express

CoffeeScript

GRUNT

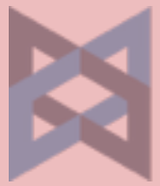
koa

Browser land



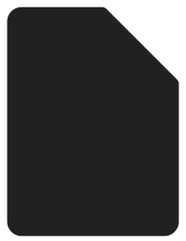
npm

NGULARJS
by Google



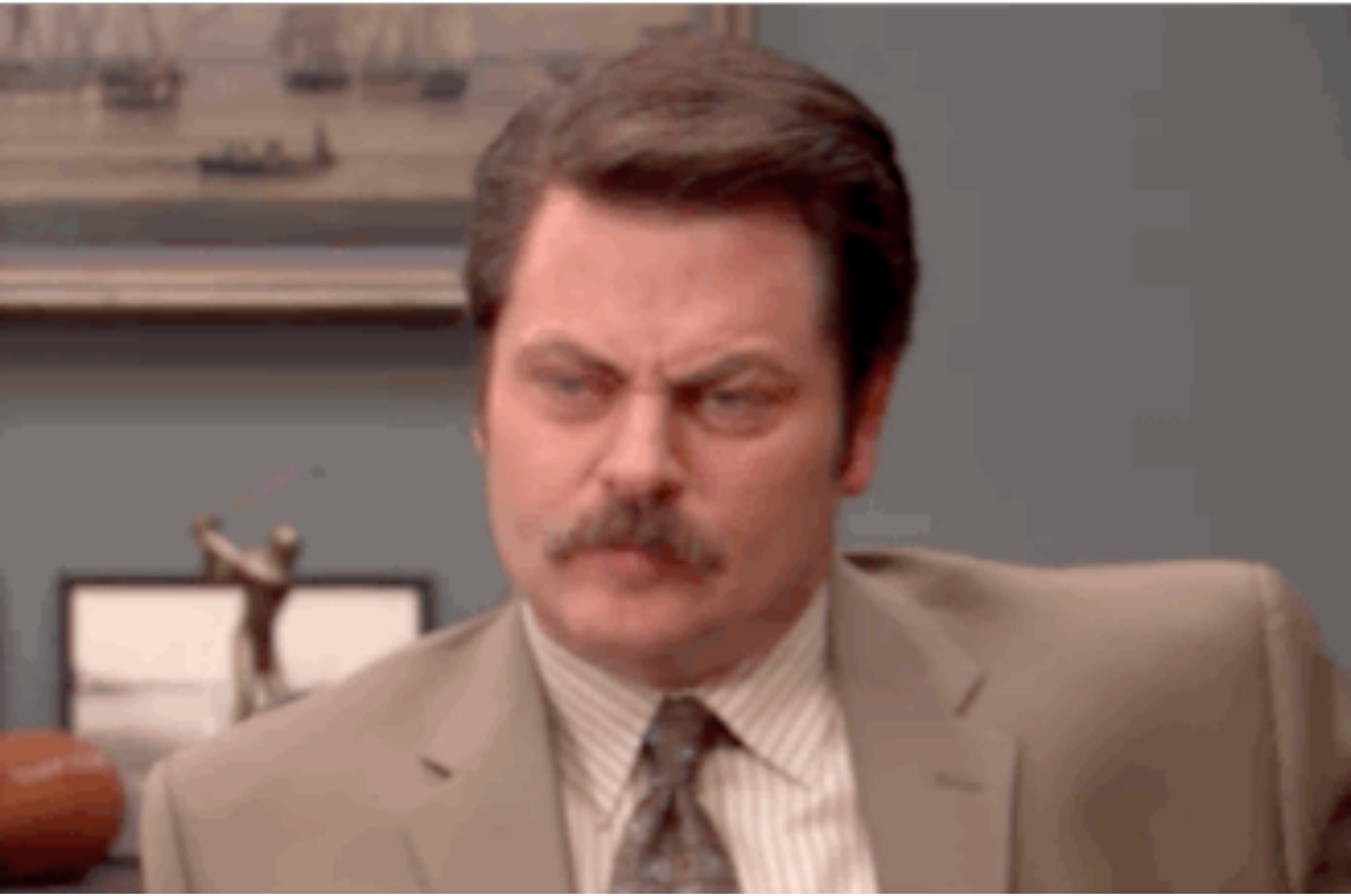
BACKBONE.JS

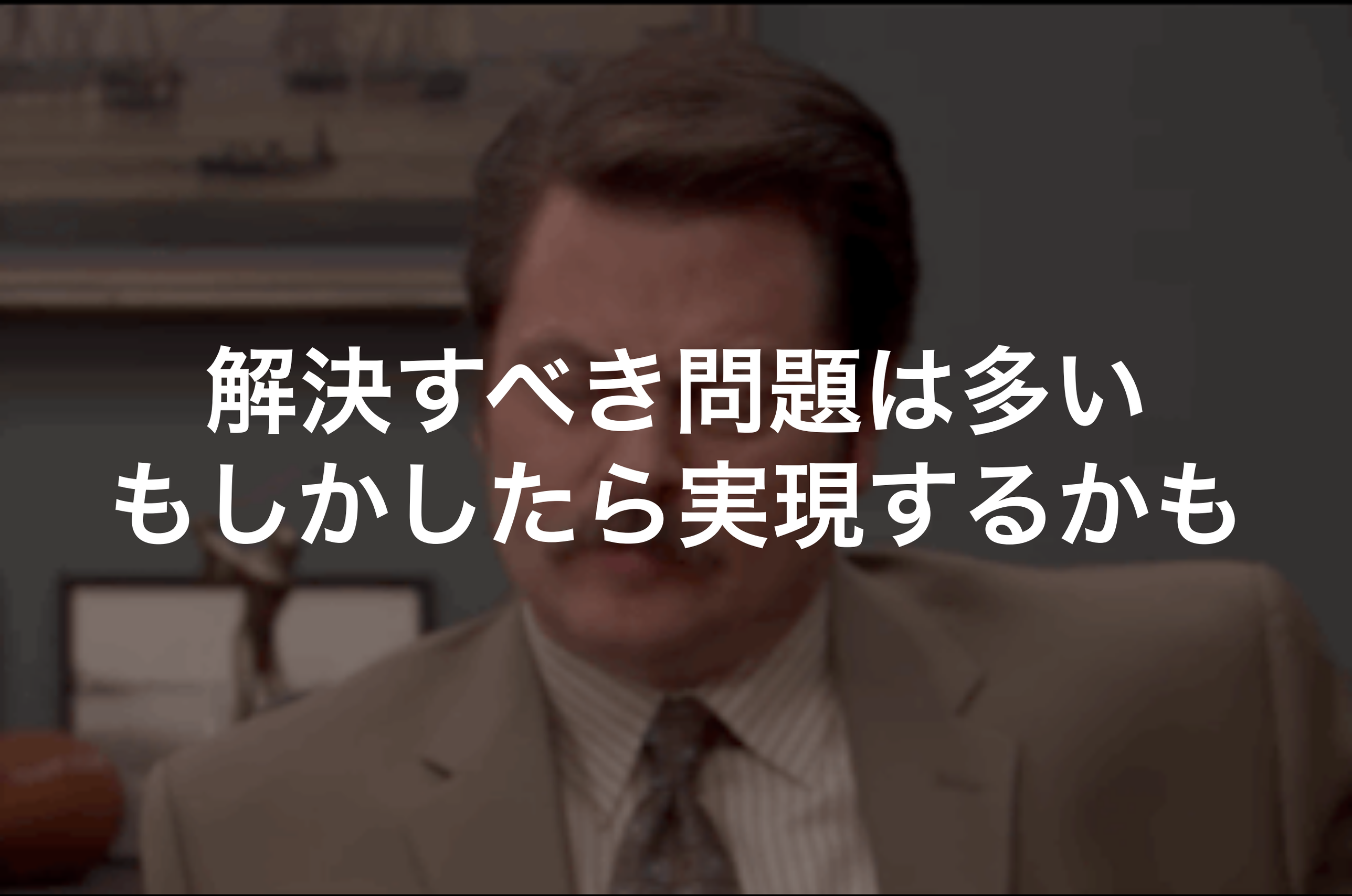
すべてのエコシステムに対応する？



package.json

```
% npm install express underscore # server
% npm install jquery backbone    # client
% npm install grunt-browserify   # build task
```




解決すべき問題は多い
もしかしたら実現するかも

npm とフロントエンドのパッケージ管理の未来

2014-11-13

JavaScript 系パッケージマネージャの重複問題

npm は言わずもがな Node.js のパッケージマネージャだが、フロントエンド開発においては Bower も利用するのが一般的になっている。この現状の問題点は、`package.jon` と `bower.json` という似たような管理ファイルを二重で管理しなければならないということだ。

現状の使い分けをおさらいをしておくと、次のような感じになる。

npm

タスクランナー（Grunt/gulp）・モジュールシステム（browserify/webpack）・テストスイート（karma/testem）などの開発環境系の管理が npm の主なお仕事。インストールされたパッケージは `node_modules` 内に展開されて、CommonJS スタイルのモジュール管理から利用する。

※ 本題につながる話としては、ブラウザで動くライブラリの一部は npm にも publish されている。

Bower

Tips: shortcuts

- いつものコマンドを短縮してみる（bowerも同じ）

```
% npm install --save <package-name>
% npm i -S <package-name>
```

```
% npm remove --save-dev <package-name>
% npm rm -D <package-name>
```

Tips: npm-shrinkwrap

- 現在の node_modules 内のバージョンを記録して固定

```
% npm shrinkwrap
wrote npm-shrinkwrap.json
```

```
"name": "A",
"version": "0.1.0",
"dependencies": {
  "B": {
    "version": "0.0.1",
    "dependencies": {
      "C": {
        "version": "0.1.0" ...
```

Tips: bower.json resolutions

- バージョンの競合で依存解決できないとき

```
Unable to find a suitable version for angular
please choose one:
```

```
"resolutions": {
    "angular": "1.3.0"
}
```

Task Runner

タスクランナー

Before

JSHint で文法チェックする

```
% jshint src1.js src2.js
```

画像のファイルサイズを最適化する

```
% imageoptim --directory ~/images
```

JavaScript をコードミニファイする

```
% uglifyjs src1.js src2.js
```

After

まとめて自動で実行

```
% grunt some-task
```

```
% gulp some-task
```

多彩なタスクを実行してくれる

- CSS Sprites の生成
- 画像のファイルサイズ最適化
- 開発用ローカルサーバの起動
- スタイルガイドの生成
- テストスイートの実行
- JSHint / CSSLint の実行
- JavaScript / CSS のミニファイ
- ファイルの結合・モジュールシステムのビルド　etc...

Grunt

Node.js タスクランナー

```
% npm install -g grunt-cli
% npm install —-save-dev grunt
```

Gulp

Streaming ビルドシステム

```
% npm install -g gulp
% npm install —-save-dev gulp
```

GRUNT

Plugins

This plugin listing is automatically generated from the npm module database. Officially maintained "contrib" plugins are marked with a star ★ icon.

In order for a Grunt plugin to be listed here, it must be published on npm with the gruntplugin keyword. Additionally, we recom
that you use the gruntplugin grunt-init template when creating a Grunt plugin.

3,827 plugins

Search:

← 1 2 3 4 5 →

Plugin	Updated	Grunt Version	Downlo (last 30 da
★ **contrib-watch** by Grunt Team Run predefined tasks whenever watched file patterns are added, changed or deleted.	4 months ago	~0.4.0	58716
★ **contrib-jshint** by Grunt Team Validate files with JSHint.	7 months ago	~0.4.0	57037
★ **contrib-uglify** by Grunt Team Minify files with UglifyJS.	2 months ago	~0.4.0	50382

gulp plugins

search plugins

1254 plugins

gulp-inject 1.0.2

repo klei/gulp-inject

A javascript, stylesheet and webcomponent injection plugin for Gulp, i.e. inject file references into your index.html

gulpplugin

gulp-coffee 2.2.0

repo wearefractal/gulp-coffee

Compile CoffeeScript files

gulpplugin

1,254 plugins

gulp-mocha 1.1.1

repo sindresorhus/gulp-mocha

Run Mocha tests

gulpplugin

gulp-uglify 1.0.1

repo terinjokes/gulp-uglify

Minify files with UglifyJS.

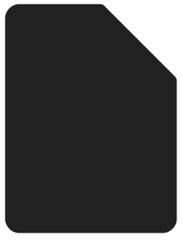
gulpplugin

gulp-header 1.2.2

repo tracker1/gulp-header

Gulp extension to add header to file(s) in the pipeline.

タスクの設定とプラグインの管理



Gruntfile.js
Gulpfile.js

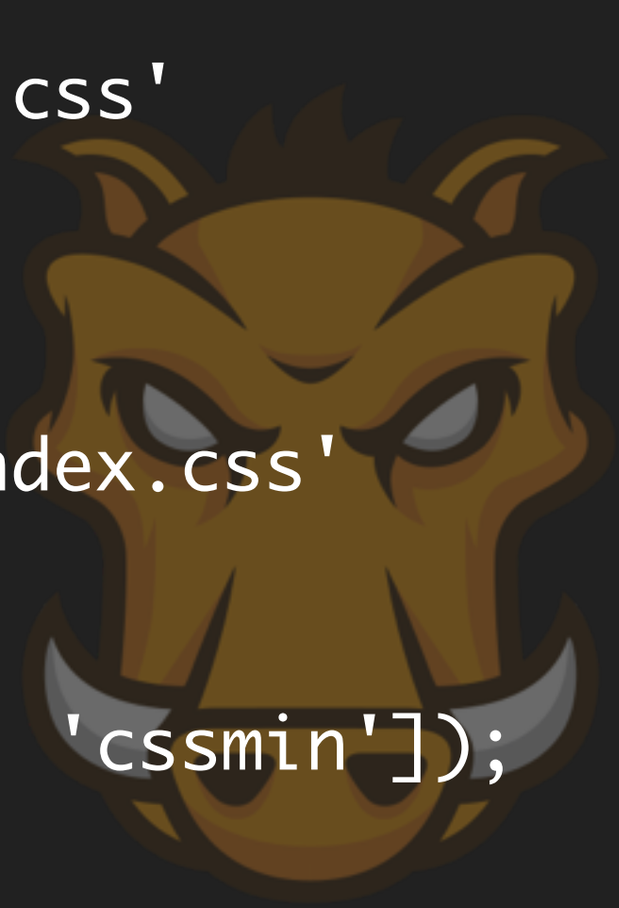
```
grunt.loadNpmTasks()
require('gulp-***')
```

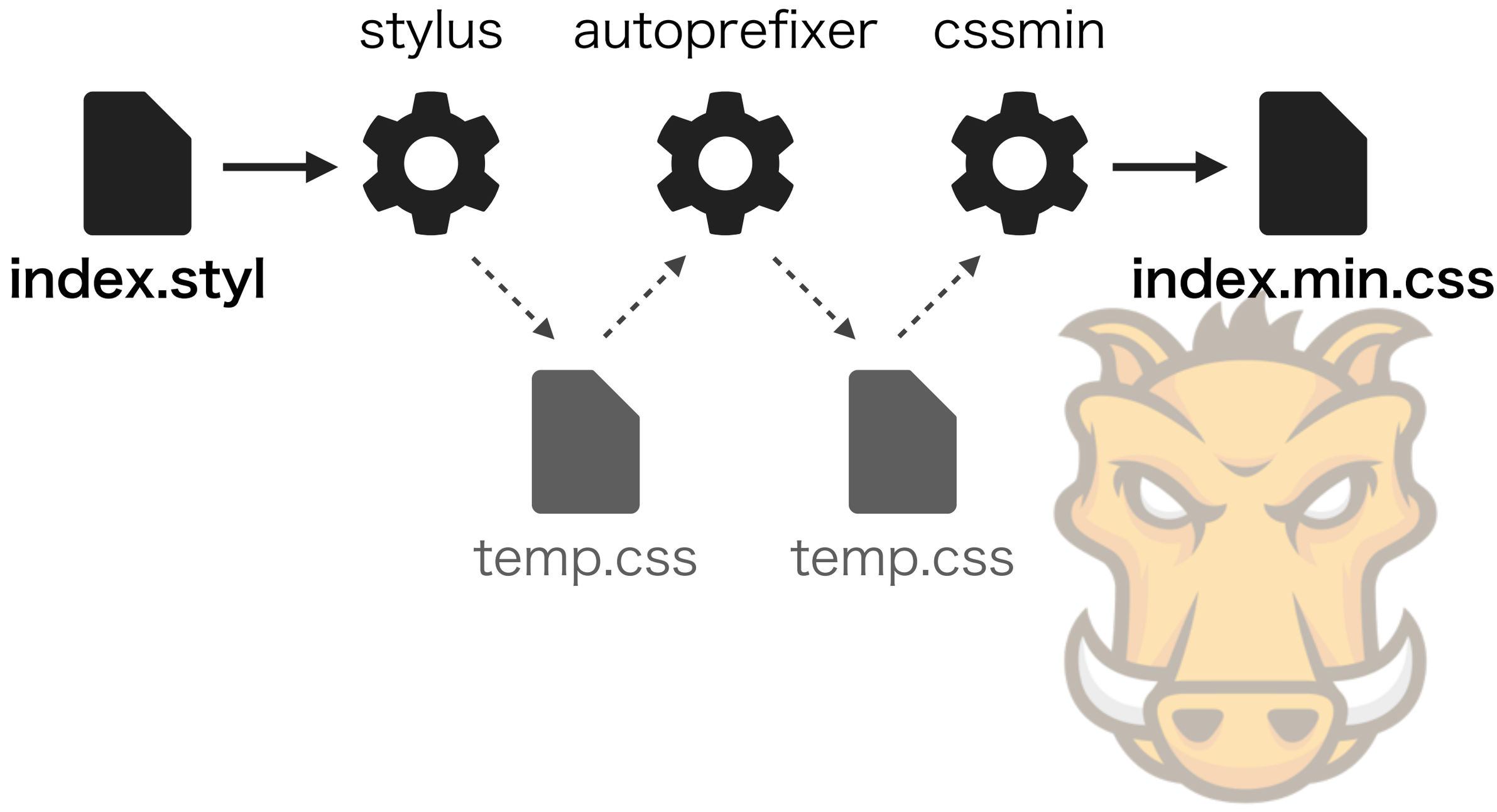
package.json

```
% npm i --save-dev
```

Configuration over code

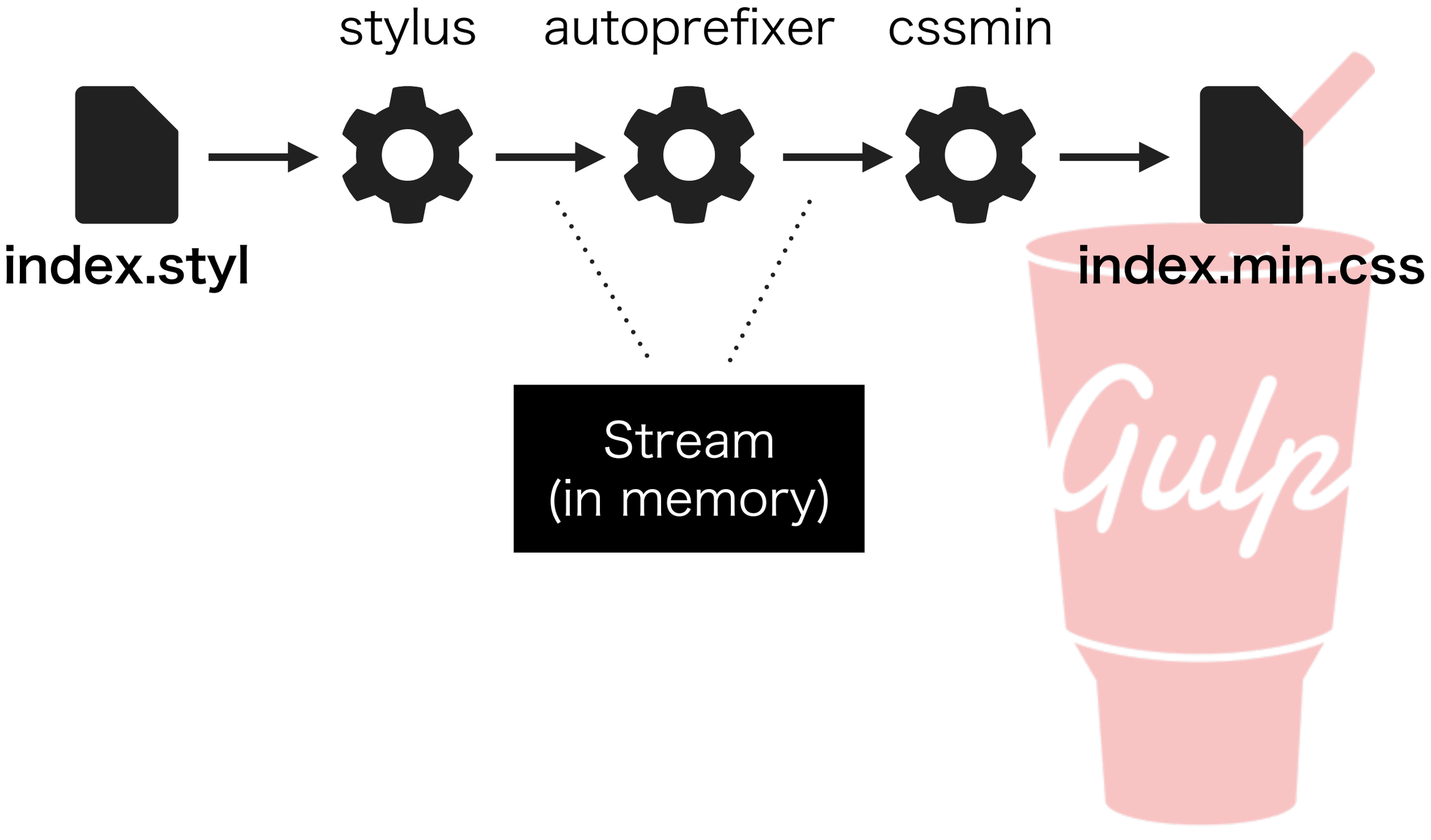
```
grunt.initConfig
  stylus:
    dist:
      files:
        'temp/index.css': 'src/index.styl'
  autoprefixer:
    dist:
      files:
        'temp/index.css': 'temp/index.css'
  cssmin:
    dist:
      files:
        'dist/index.min.css': 'temp/index.css'

grunt.registerTask('build-css',
                   ['stylus, autoprefixer', 'cssmin']);
```




stylus
autoprefixer
cssmin
index.styl
temp.css
temp.css
index.min.css

Code over configuration

```
var gulp         = require('gulp');
var sass         = require('gulp-stylus')
var autoprefixer = require('gulp-autoprefixer')
var cssmin       = require('gulp-cssmin')

gulp.task('cssbuild', function () {
    gulp.src('src/index.styl')
        .pipe(stylus())
        .pipe(autoprefixer())
        .pipe(cssmin())
        .pipe(gulp.dest('dist'))
});
```


stylus
autoprefixer
cssmin
index.styl
index.min.css
Stream
(in memory)
Gulp

Gruntfile.coffee

```
grunt.initConfig
  jsduck:
    options:
      'builtin-classes': false
      'warnings'       : ['-dup_member', '-type_name']
      'external'       : ['XMLHttpRequest']
    dist:
      src  : ['dist/phalanx.debug.js']
      dest : 'docs'

  plato:
    options:
      jshint : grunt.file.readJSON('.jshintrc')
    dist:
      src  : ['src/**/*.js']
      dest : 'reports'

grunt.loadNpmTasks 'grunt-jsduck'
grunt.loadNpmTasks 'grunt-plato'
```

The State of Grunt

August 2014

http://cowboy.github.io/state-of-grunt-fe-summit-2014-talk/

"Cowboy" Ben Alman
Bocoup

Tips: Grunt plugins

- **jit-grunt**
  - タスク実行時までロードを遅延させる
- **grunt-conccurrent**
  - タスクの実行を並列化して高速化
- **grunt-parallelize**
  - タスクの入力ファイル水平分割して高速化
- **load-grunt-config**
  - 設定ファイルの分割とタスクのオートロード


設定が肥大化して
辛くなってしまったら

FYI: npm run <command>

- npm@2.0.0 で引数も渡せるようになった

```
% npm run lint
```

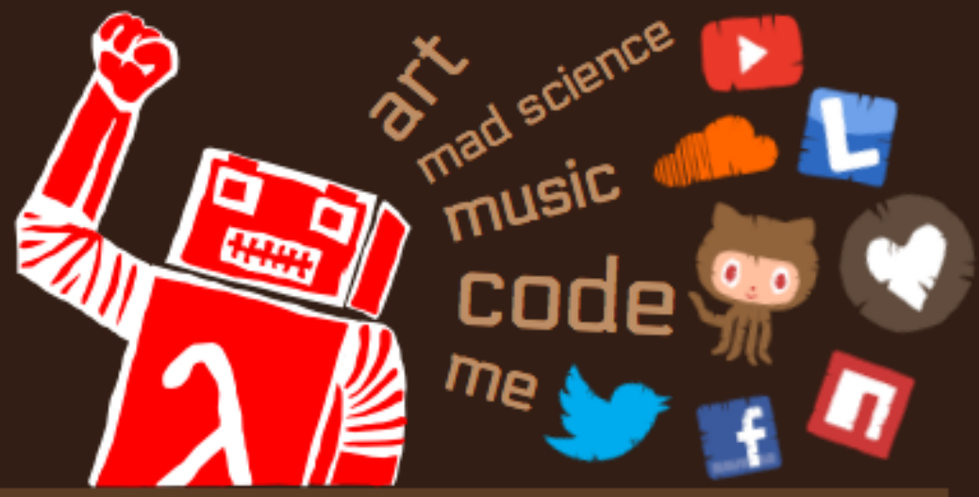
```
"scripts": {
  "lint": "jshint **.js",
  "lint:checkstyle": "npm run lint --
        --reporter checkstyle > checkstyle.xml"
}
```


Goooood!!

gulpjs/gulp: package.json

```
"scripts": {

    "prepublish": "marked-man --name gulp docs/CLI.md
> gulp.1",

    "lint": "jshint lib bin index.js --reporter
node_modules/jshint-stylish/stylish.js --exclude
node_modules",

    "test": "npm run-script lint && mocha --reporter
spec",

    "coveralls": "istanbul cover _mocha --report
lcovonly -- -R spec && cat ./coverage/lcov.info |
coveralls && rm -rf ./coverage"

},
```

https://github.com/gulpjs/gulp/blob/master/package.json



task automation with npm run

commit c13668f2b5f4f515e97723fa3322aa009181629c
Author: James Halliday
Date: Mon Nov 18 01:52:10 2013 +0800

There are some fancy tools for doing build automation on javascript projects that I've never felt the appeal of because the lesser-known `npm run` command has been perfectly adequate for everything I've needed to do while maintaining a very tiny configuration footprint.

Here are some tricks I use to get the most out of `npm run` and the `package.json` "scripts" field.

the scripts field

If you haven't seen it before, npm looks at a field called `scripts` in the package.json of a project in order to make things like `npm test` from the `scripts.test` field and `npm start` from the `scripts.start` field work.

`npm test` and `npm start` are just shortcuts for `npm run test` and `npm run start` and you can use `npm run` to run whichever entries in the `scripts` field you want!

Another thing that makes `npm run` really great is that npm will automatically set up `$PATH` to look in `node_modules/.bin`, so you can just run commands supplied by dependencies and devDependencies directly

Task centric configuration - JSON設定ライク
統合タスクランナーとしてバランスが良い

Target centric configuration - JSコードライク
1ファイルに複数の処理をするビルドプロセス向き

`npm run` は意外に使える
Node.js 環境なら確実に利用できて手軽＆シンプル

Module System

モジュールシステム

Before

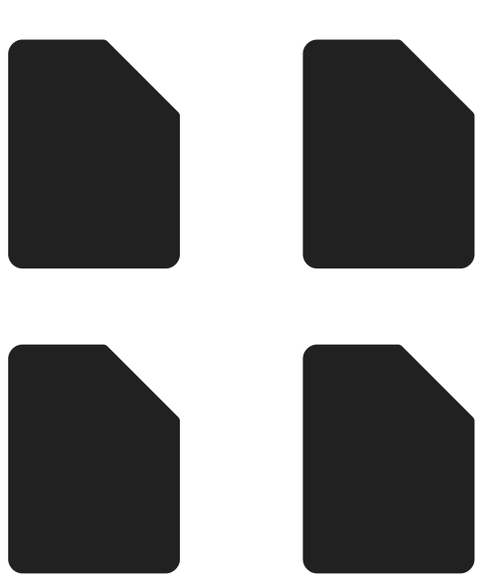
- 再利用性が低い長大なコード
- ページ単位？くらいで分割してるかも
- <script> 要素を順番に並べよう

After

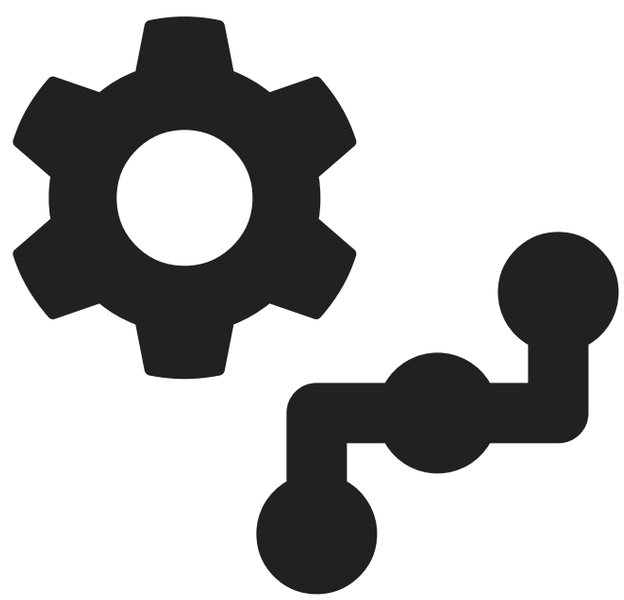
- クライアントサイドMVC / Flux（ｷﾘｯ
- モジュール管理（ｷﾘｯ
- 依存解決（ｷﾘｯ

モジュールシステムに求めるもの

- 分割されたコード（ファイル）をモジュールとして管理
- ビルドプロセス or ランタイム に依存解決



Module management



Resolve dependencies

ES6!

AMD!

Browserify!

なんでもいいよ！！

Browserify

CommonJS をブラウザで実行

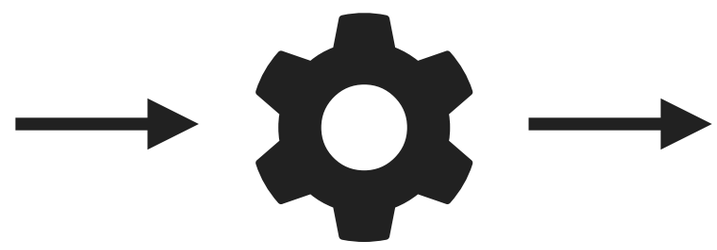
```
% npm install --g browserify
```

node_modules

node built-ins

Modules
(CommonJS)

Browserify



bundle.js

example:CommonJS

```
# module.js
module.exports = function (n) {
    return n * 111
};

# index.js
var module = require('./module');
console.log(module(5));

# cmd
$ browserify index.js > bundle.js
$ node bundle.js # => 555
```

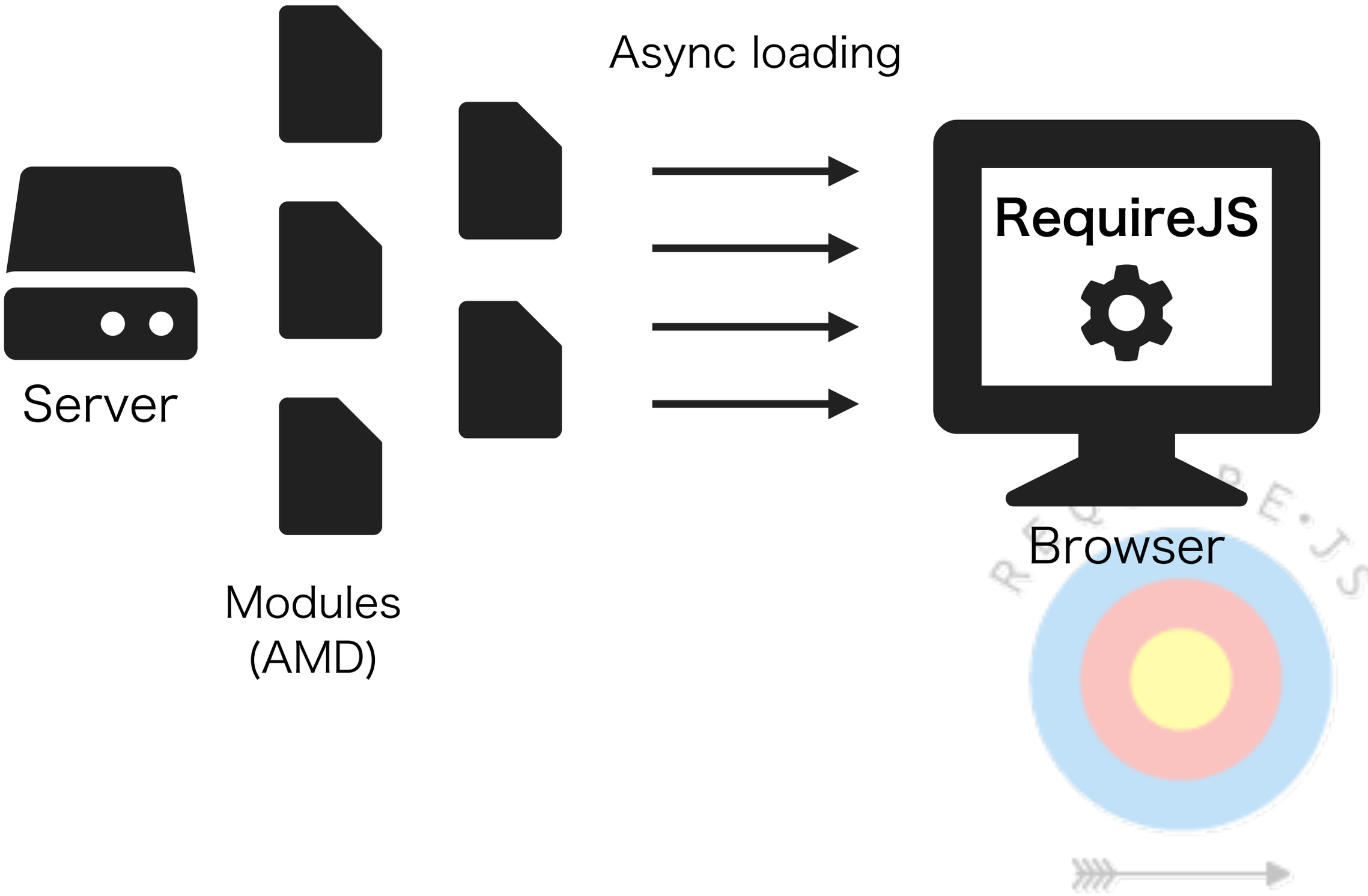
package.json

```
{
  "name": "mypkg",
  "version": "1.2.3",
  "main": "main.js",
  "browser": {
    "foo": "./vendor/foo.js"
  },
  "browserify": {
    "transform": "browserify-shim"
  },
  "browserify-shim": {
    "foo": "FOO"
  }
}
```

Require.js

AMD と呼ばれる一連の仕組み

```
% bower install requirejs
% npm install -g r.js
```

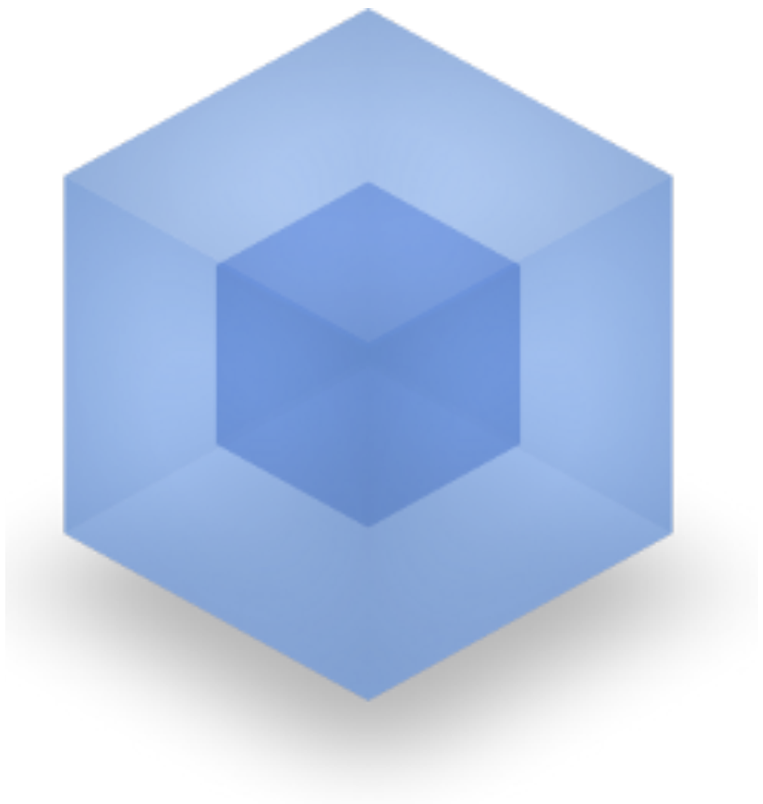

Async loading
RequireJS
Server
Modules
(AMD)
Browser

example:AMD

```
# html
<script src="require.js"
        data-main="main.js" async></script>

# main.js
define([‘module’], function(module) {
  alert(module.foo); // ‘bar’
});

# module.js
define(function() {
  return {foo: ‘bar’}
});
```

require.config

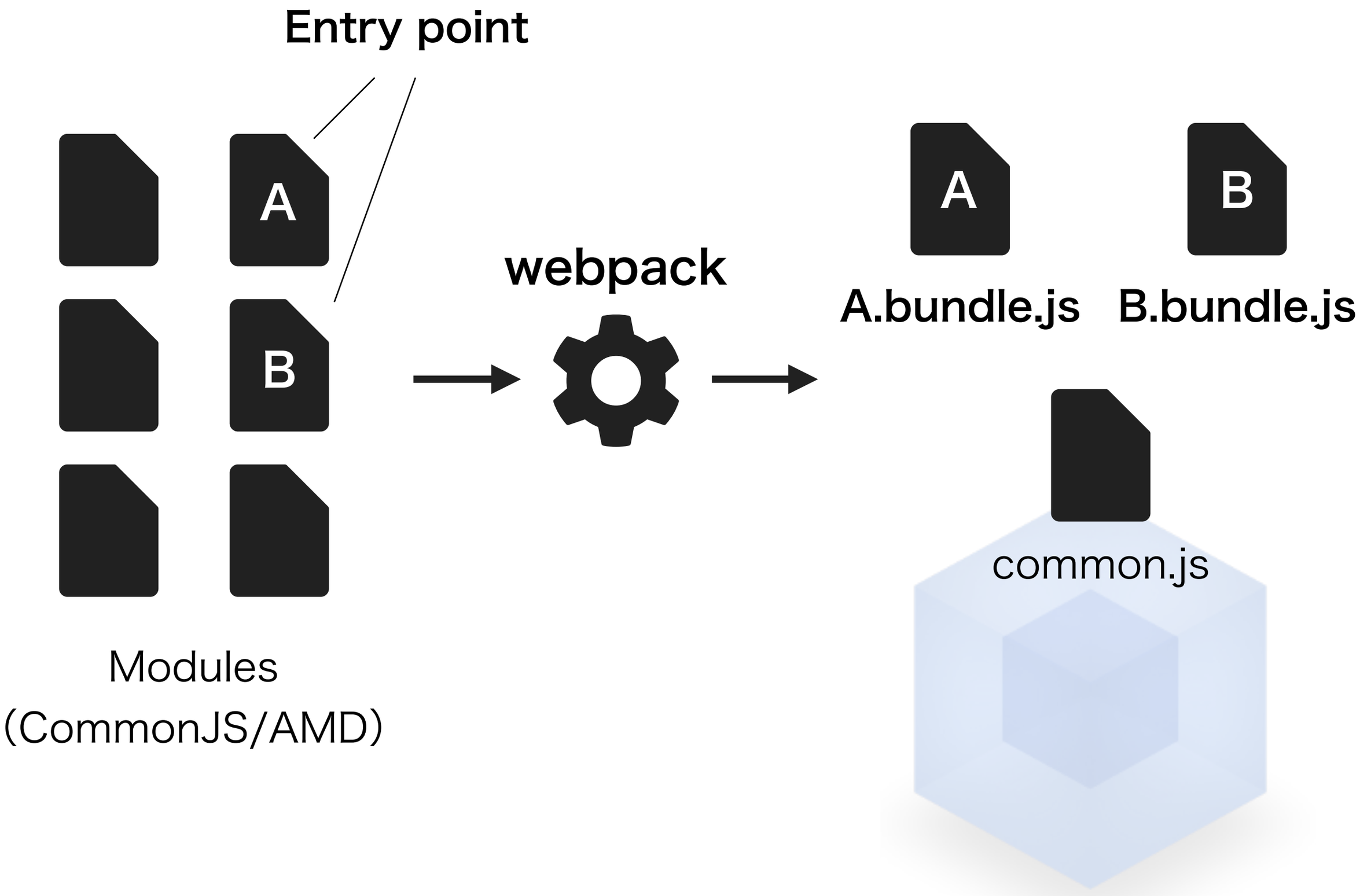
```
require.config({
  paths: {
    jquery     : "libs/jquery.min",
    backbone   : "libs/backbone.min",
    underscore : "libs/underscore.min",
  },
  shim : {
    underscore : {
      exports : "_"
    },
    backbone : {
      deps    : [ "jquery", "underscore" ],
      exports : "Backbone"
    }
  }
});
```

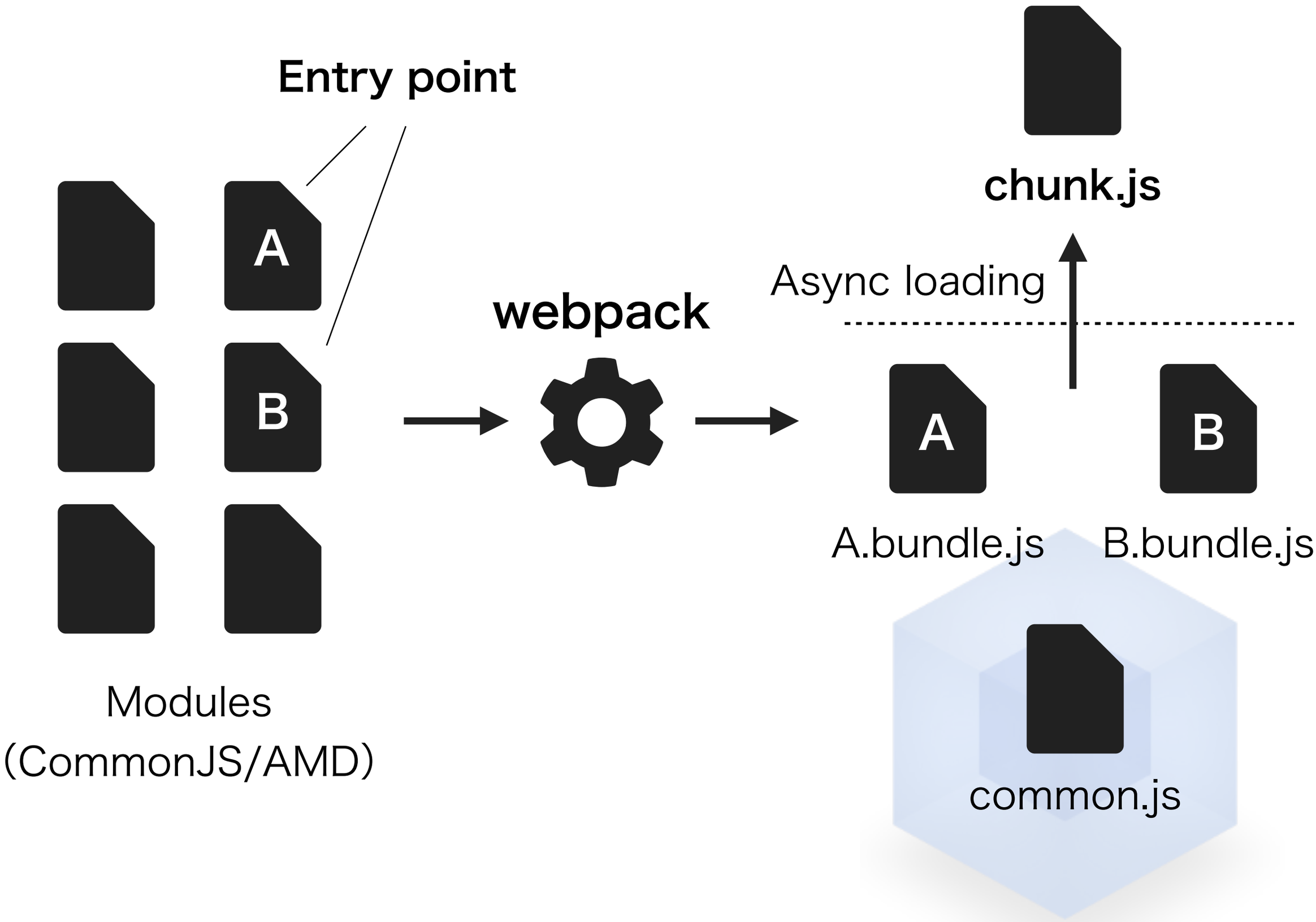


webpack

AMD, CommonJS に両対応+α

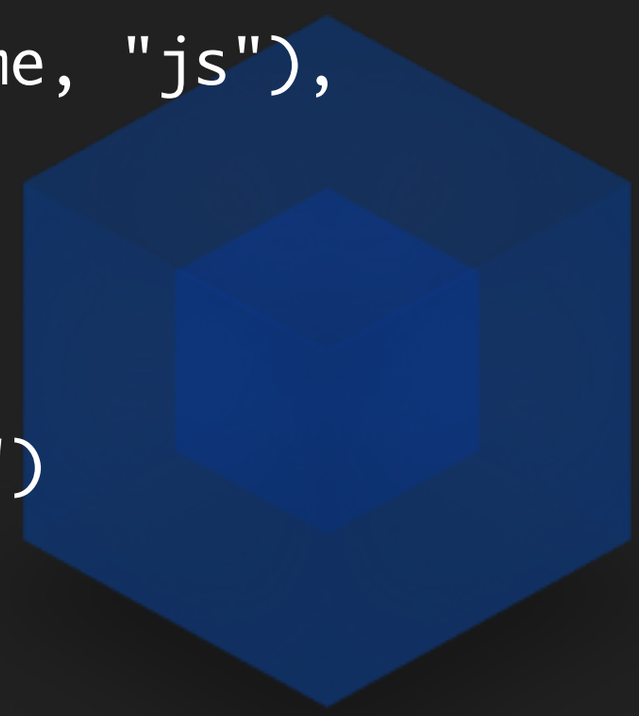
```
% npm install --g webpack
```


Entry point
A
B
webpack
A
B
A.bundle.js
B.bundle.js
common.js
Modules
(CommonJS/AMD)


Entry point
A
B
webpack
chunk.js
Async loading
A
B
A.bundle.js
B.bundle.js
common.js
Modules
(CommonJS/AMD)

webpack.config.js

```
var path = require("path");
var CommonsChunkPlugin = require("../../lib/optimize/
CommonsChunkPlugin");
module.exports = {
    entry: {
        pageA : "./pageA",
        pageB : "./pageB"
    },
    output: {
        path          :  path.join(__dirname, "js"),
        filename      : "[name].bundle.js",
        chunkFilename : "[id].chunk.js"
    },
    plugins: [
        new CommonsChunkPlugin("commons.js")
    ]
}
```



webpack

MODULE BUNDLER

2014-07-04 **Help to improve webpack by filling out this little survey. Thanks!**

Edit

COMPARISON

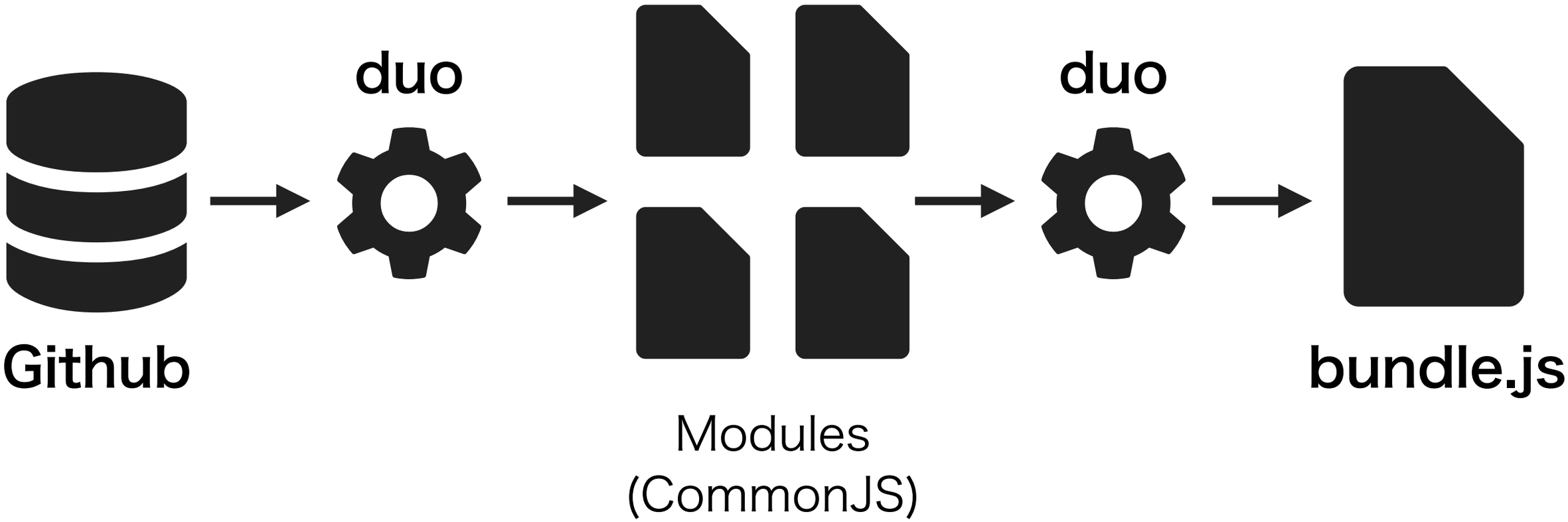
Feature	webpack/webpack	jrburke/requirejs	substack/node-browserify
CommonJs `require`	**yes**	only wrapping in `define`	**yes**
CommonJs `require.resolve`	**yes**	no	no
CommonJs `exports`	**yes**	only wrapping in `define`	**yes**
AMD `require`	**yes**	**yes**	no
AMD `require` loads on demand	**yes**	with manual configuration	no
generate a single bundle	**yes**	yes♦	yes
load each file separate	no	no♦	no
multiple bundles	**yes**	with manual configuration	with manual configuration
additional chunks are loaded on demand	**yes**	**yes**	no
multi pages build with common bundle	with manual configuration	**yes**	with manual configuration

http://webpack.github.io/docs/comparison.html

duo

パッケージ管理+ビルドシステム

```
% npm install --g duo
```


duo
duo
Github
bundle.js
Modules
(CommonJS)

example:CommonJS & CSS

```
# JavaScript
var uid = require('matthewmueller/uid');
var fmt = require('yields/fmt');

var msg = fmt('Your unique ID is %s!', uid());
window.alert(msg);
```

```
# CSS
@import 'necolas/normalize.css';
@import './layout/layout.css';

body {
  color: teal;
  background: url('./background-image.jpg');
}
```

かんたんパッケージマネージャDuo

2014年11月14日(金) NEW！
テーマ：サービス・技術

みなさん、こんにちは。Ameba事業本部ゲーム部門の平木(@Layzie) と申します。

最近はSteamで安いゲームを漁ってばかりの毎日です。このエンジニアブログでは初めて執筆になります。

さて、今回エンジニアブログで何を書こうか悩んだのですが、悩んだ結果Duoというパッケージマネージャの紹介をしようということになりました。

このDuo、GitHubのStar数は結構多いんですが、(2014/11現在2618Star)あまり紹介されてる記事とかが無いので紹介してみようと思った次第です。

プロフィール

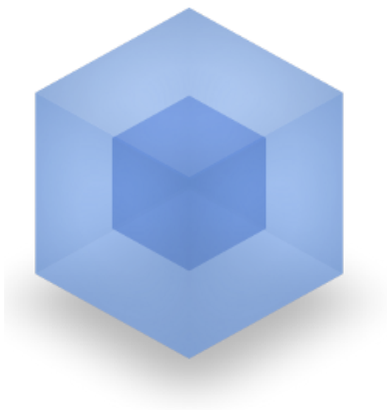
サイバーエージェント公式エンジニアブログ
プロフィール | ピグの部屋
なう | グルっぽ

自己紹介：不定期更新。サイバーエージェントのエンジニアが、持ち回りで技術、環境、職場を語ります。 続きを見る

※個人の感想です

CommonJSスタイルさえ理解していれば一番ラク
設定ファイルもほぼ必要ないので導入も容易

PCならAMDも悪くないが、モバイルではどうか
パス解決の設定が分厚くなりがち＆独自記法△



大艦巨砲主義
設定はもちろん分厚くなるが、分割ビルドは魅力的

GitHubからのインストール＆ビルドがシームレス◎
現状だと npm + Browserify と比べて優位性が低い

Conclusion

まとめ

- パッケージリポジトリは npm に
  集約されるかもしれないし、されないかもしれない
- Gruntタスクランナー と Gulpビルドシステム は
  好きに使い分けたら良いのでは
- npm run 便利だよ（個人的な推し）
- 今は CommonJS スタイルがフロントでもオススメかも
- モジュール管理は WebComponents とか ES6 とか
  色々あるから色々変化しやすそう

Thanks!

- http://aho.mu
- @ahomu
- github.com/ahomu