
בוקר טוב

What’s on for today?

A workshop on setting up a web developer environment

What you should have already setup

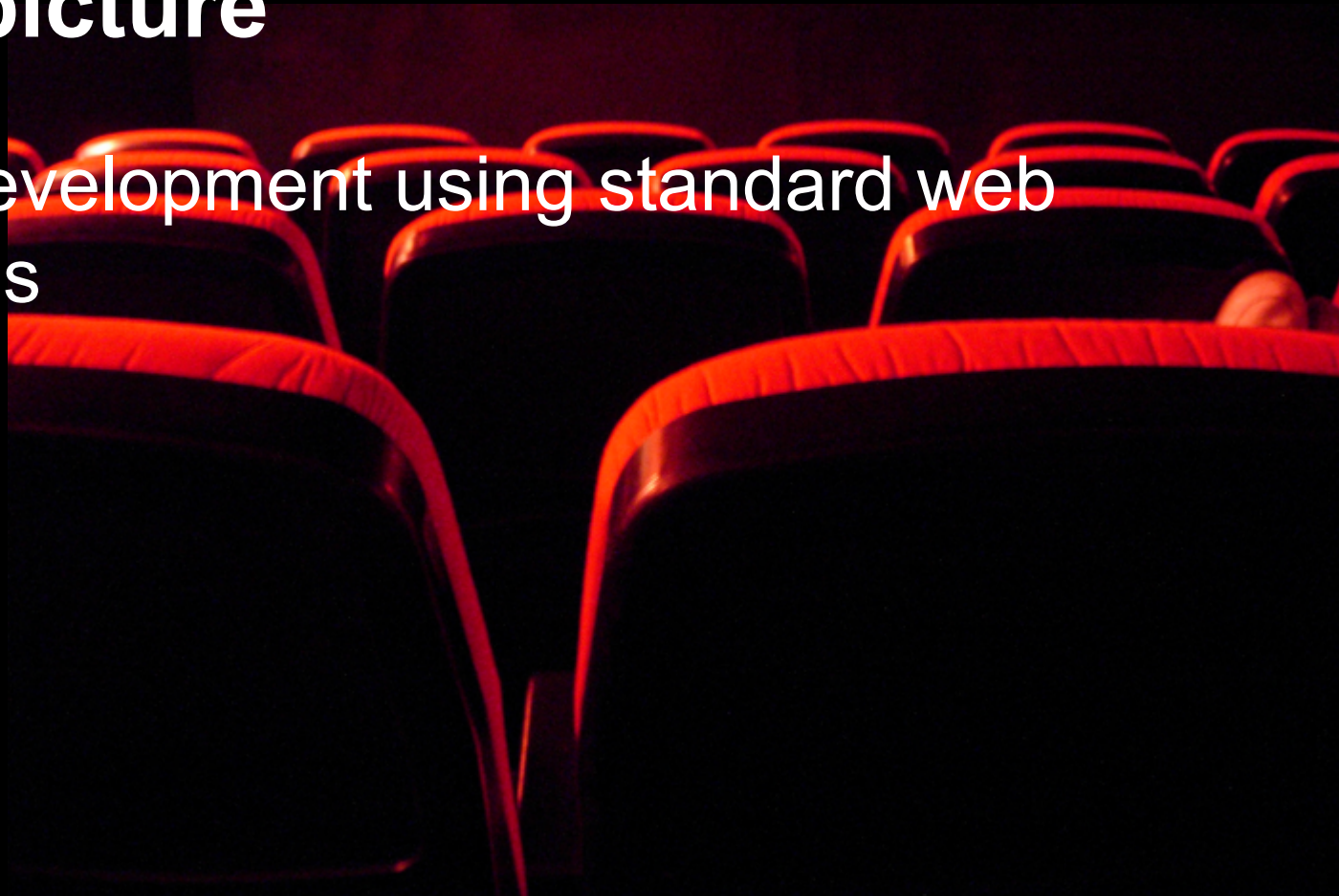
- Installed:

  node; npm; git; ruby+sass+compass; IDEA/Sublime.

- GitHub Account that you can push to.

The big picture

Fullstack development using standard web technologies



This is a blog post with ID 2

by Author B

Posted on 7-22-2012

POst two ake sweet visualizations. http://jsantell.github.com/dancer.js _v0.2.1 (6/15/2012)_ Features --- * Use real-time audio waveform and frequency data and map it to any arbitrary visualization * Leverage beat detection into your visualizations * Simple API to time callbacks and events to any section of a song * Supports Web Audio (webkit), Audio Data (mozilla) and flash fallback (v9+) * Extensible framework supporting plugins and custom behaviours

Read More ›

This is a blog post with ID 1

Blog Search

Popular Blog Categories

Dinosaurs
Spaceships
Fried Foods
Wild Animals
Alien Abductions
Business Casual
Robots
Fireworks

Side Widget Well

Bootstrap's default wells work great for side widgets! What is a widget anyways...?

What’s the stack?

Liveperson stack

Just before we start

About me

storyboard

GIT

storyboard

SPA Server using
Node & NPM

storyboard

Grunt

storyboard

Bower

storyboard

web developer toolings

storyboard - BONUS

requirejs and it’s optimizer rjs

Timetable

10:00 - 11:00 Some Gitting **code along
11:00 - 12:00 SPA Server
12:00 - 13:00 LUNCH
13:00 - 14:00 Grunting
14:00 - 15:30 EXERCISE
15:30 - 16:00 Solution
16:00 - 16:30 Bower **code along

Timetable

16:30 - 17:00 Tooling
17:00 - 18:00 EXERCISE

If we'll have time we'll go into requirejs.

Let’s Start

Why bother Git

it makes developers

HAPPY

Git is a decentralized vc

no need for server installation just fs

Git is much faster than Subversion

Branch Handling

Access Control

in SVN versioning features depend on commit access

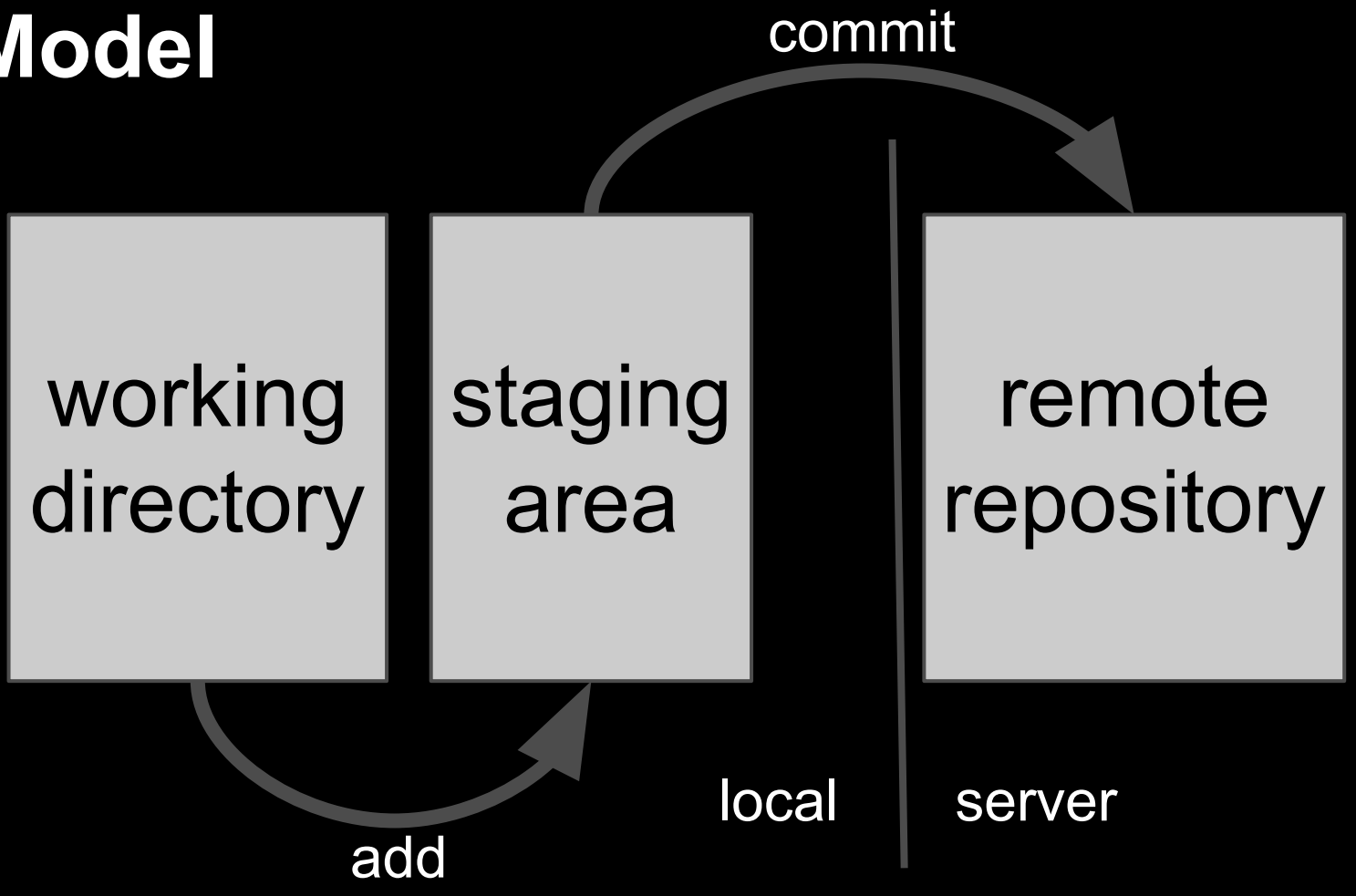
SVN is Single Repository

git can reference different remote repositories all at the same time

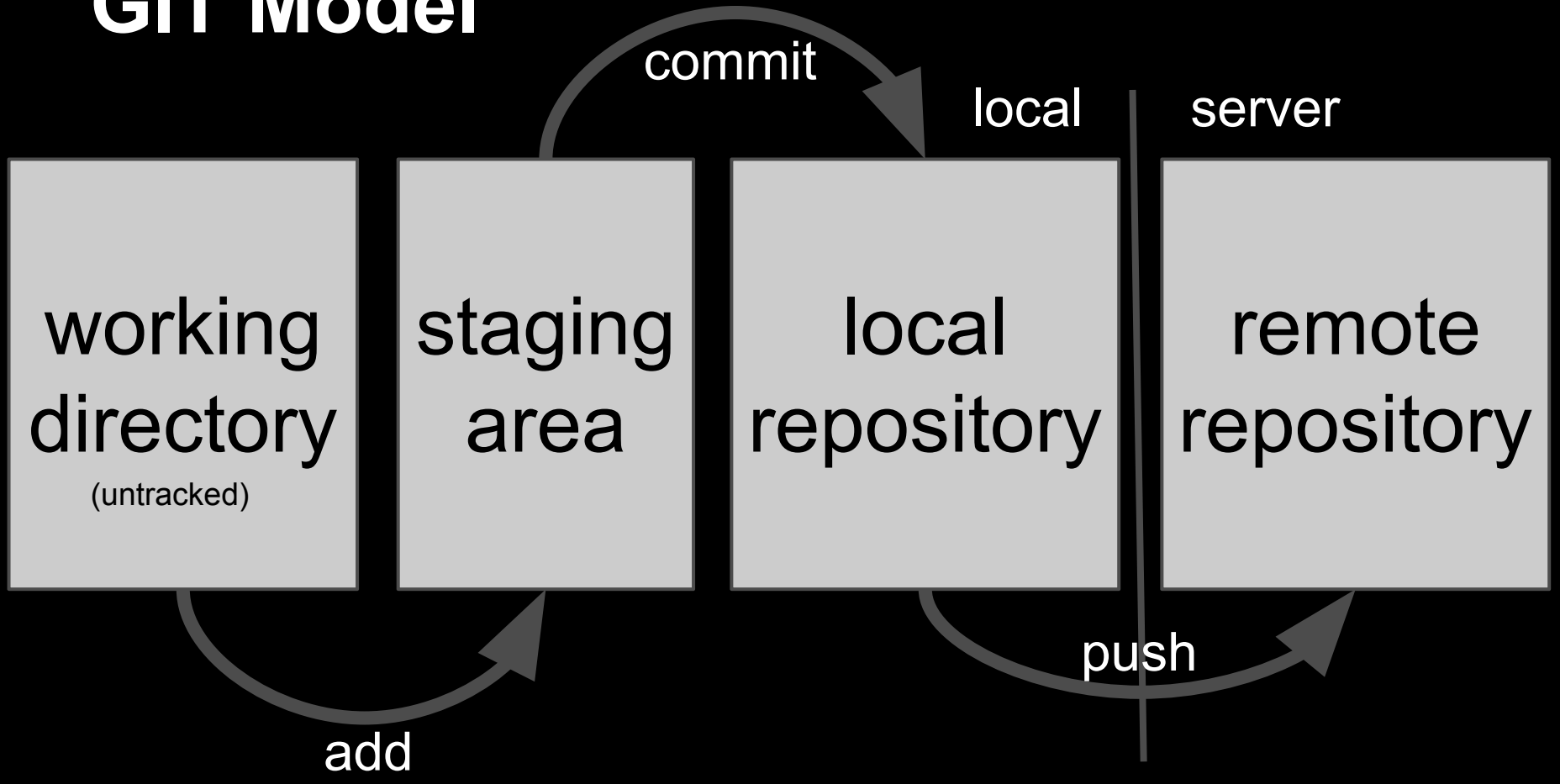
Sub Repositories

full local control

SVN Model


commit
working directory
staging area
remote repository
add
local
server

GIT Model


commit
local
server
working directory
(untracked)
staging area
local repository
remote repository
push
add

Git references

git-scm

git wiki

Git Cheat-Sheet

CREATE

Clone an existing repository

```
$ git clone ssh://user@domain.com/repo.git
```

Create a new local repository

```
$ git init
```

LOCAL CHANGES

Changed files in your working directory

```
$ git status
```

Changes to tracked files

```
$ git diff
```

Add all current changes to the next commit

```
$ git add .
```

Add some changes in <file> to the next commit

```
$ git add -p <file>
```

Commit all local changes in tracked files

```
$ git commit -a
```

BRANCHES & TAGS

List all existing branches

```
$ git branch
```

Switch HEAD branch

```
$ git checkout <branch>
```

Create a new branch based on your current HEAD

```
$ git branch <new-branch>
```

Create a new tracking branch based on a remote branch

```
$ git branch --track <new-branch>
  <remote-branch>
```

Delete a local branch

```
$ git branch -d <branch>
```

Mark the current commit with a tag

```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes

```
$ git remote -v
```

MERGE & REBASE

Merge <branch> into your current HEAD

```
$ git merge <branch>
```

Rebase your current HEAD onto <branch>

Don't rebase published commits!

```
$ git rebase <branch>
```

Abort a rebase

```
$ git rebase --abort
```

Continue a rebase after resolving conflicts

```
$ git rebase --continue
```

Use your configured merge tool to solve conflicts

```
$ git mergetool
```

Use your editor to manually solve conflicts and (after resolving) mark file as resolved

```
$ git add <resolved-file>
```

```
$ git rm <resolved-file>
```

UNDO

Add some changes in <file> to the next commit

```
$ git add -p <file>
```

Commit all local changes in tracked files

```
$ git commit -a
```

Commit previously staged changes

```
$ git commit
```

Change the last commit

Don't amend published commits!

```
$ git commit --amend
```

COMMIT HISTORY

Show all commits, starting with newest

```
$ git log
```

Show changes over time for a specific file

```
$ git log -p <file>
```

Who changed what and when in <file>

```
$ git blame <file>
```

```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes

```
$ git remote -v
```

Show information about a remote

```
$ git remote show <remote>
```

Add new remote repository, named <remote>

```
$ git remote add <remote> <url>
```

Download all changes from <remote>, but don't integrate into HEAD

```
$ git fetch <remote>
```

Download changes and directly merge/ integrate into HEAD

```
$ git pull <remote> <branch>
```

Publish local changes on a remote

```
$ git push <remote> <branch>
```

Delete a branch on the remote

```
$ git push <remote> :<branch>
```

Publish your tags

```
$ git push --tags
```

resolved

```
$ git add <resolved-file>
```

```
$ git rm <resolved-file>
```

UNDO

Discard all local changes in your working directory

```
$ git reset --hard HEAD
```

Discard local changes in a specific file

```
$ git checkout HEAD <file>
```

Revert a commit (by producing a new commit with contrary changes)

```
$ git revert <commit>
```

Reset your HEAD pointer to a previous commit

...and discard all changes since then

```
$ git reset --hard <commit>
```

...and preserve all changes as unstaged changes

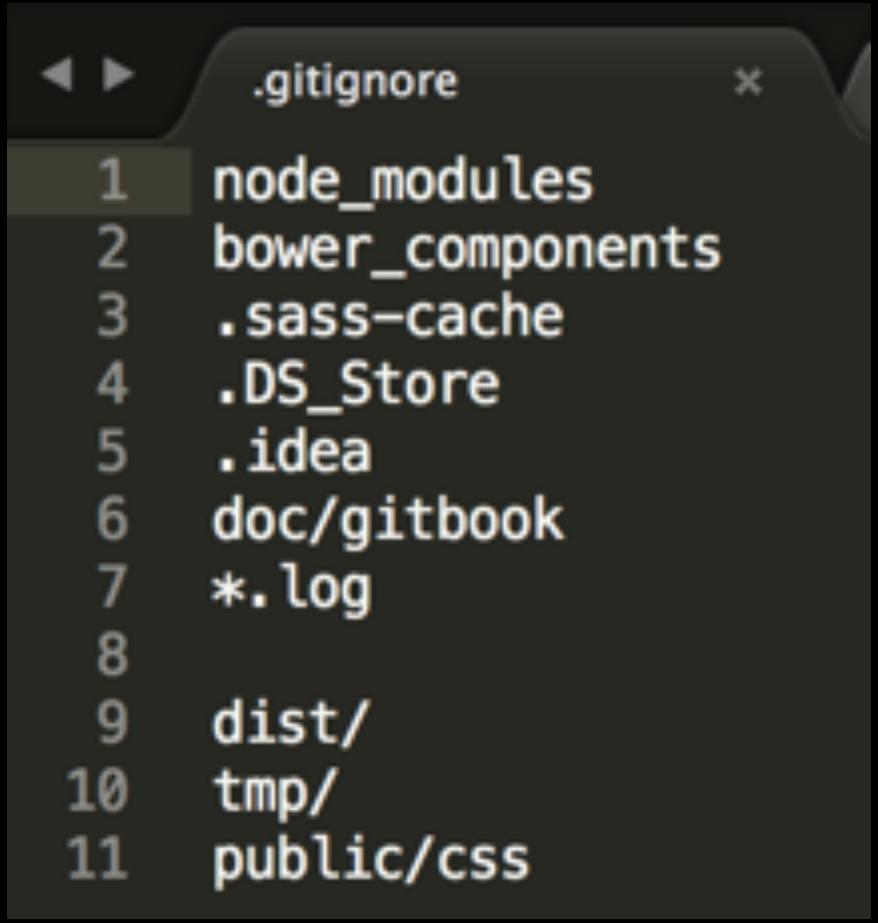
```
$ git reset <commit>
```

...and preserve uncommitted local changes

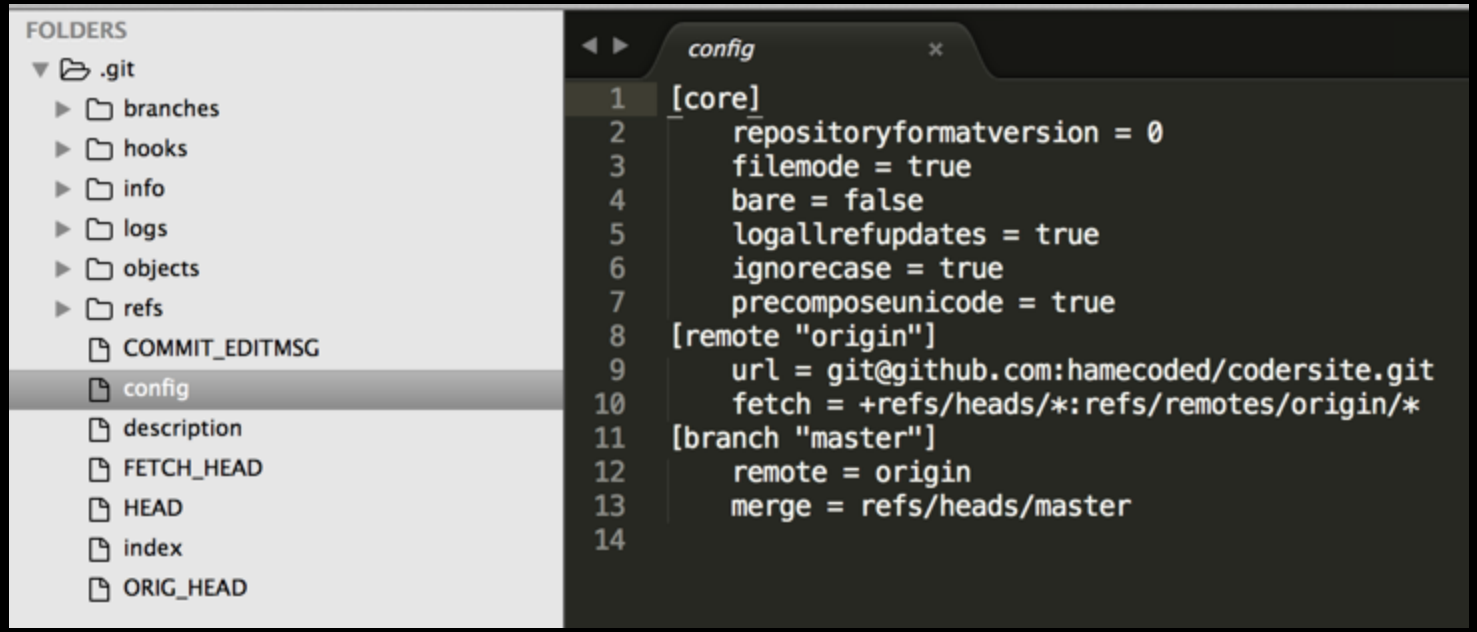
```
$ git reset --keep <commit>
```

git files you should be familiar with

.gitignore


.gitignore
1 node_modules
2 bower_components
3 .sass-cache
4 .DS_Store
5 .idea
6 doc/gitbook
7 *.log
8
9 dist/
10 tmp/
11 public/css

.git directory



Create flow

...or create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:hamecoded/test.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:hamecoded/test.git
git push -u origin master
```

one time config

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```

Enables helpful colorization of command line output

Update

Fetch latest changes from origin

```
git fetch
```

(but this does not merge them).

Pull latest changes from origin

```
git pull
```

(does a fetch followed by a merge)

Apply a patch that some sent you

```
git am -3 patch.mbox
```

(in case of a conflict, resolve and use `git am --resolved`)

Branch

Switch to the $id branch

```
git checkout $id
```

Merge branch1 into branch2

```
git checkout $branch2
git merge branch1
```

Create branch named $branch based on the HEAD

```
git branch $branch
```

Create branch $new_branch based on branch $other and switch to it

```
git checkout -b $new_branch $other
```

Delete branch $branch

```
git branch -d $branch
```

Exercise

Setup a local git repository
configure it
add a README file
commit
push to GitHub

20min

node & NPM

Node Package Manager

with node we'll run things
with npm we'll install things

What will we use npm for?

system tools

```
➜  node_modules git:(master) ✗ ls
bower                grunt-init           node-inspector     season
connect              jade                 npm                sentiment
grunt-cli            jshint               npm-check-updates  weinre
➜  node_modules git:(master) ✗ pwd
/usr/local/lib/node_modules
```

project libraries

powers grunt and bower

which themselves power many packages
and that we'll shortly use

make sure you have it set in your path

```
export PATH=$HOME/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/local/sbin
```

This way cli modules could be used in terminal

npm init

let’s try it

Let’s setup our SPA server

if fnf return index

The hash#

skips the network

History API

modifies history urls

But still a url path will always turn to the server

SEO and SPA

the downside

Google crawlers are now doing better with SPA

but still not 100%

To SPA or not to SPA?

Blog vs App

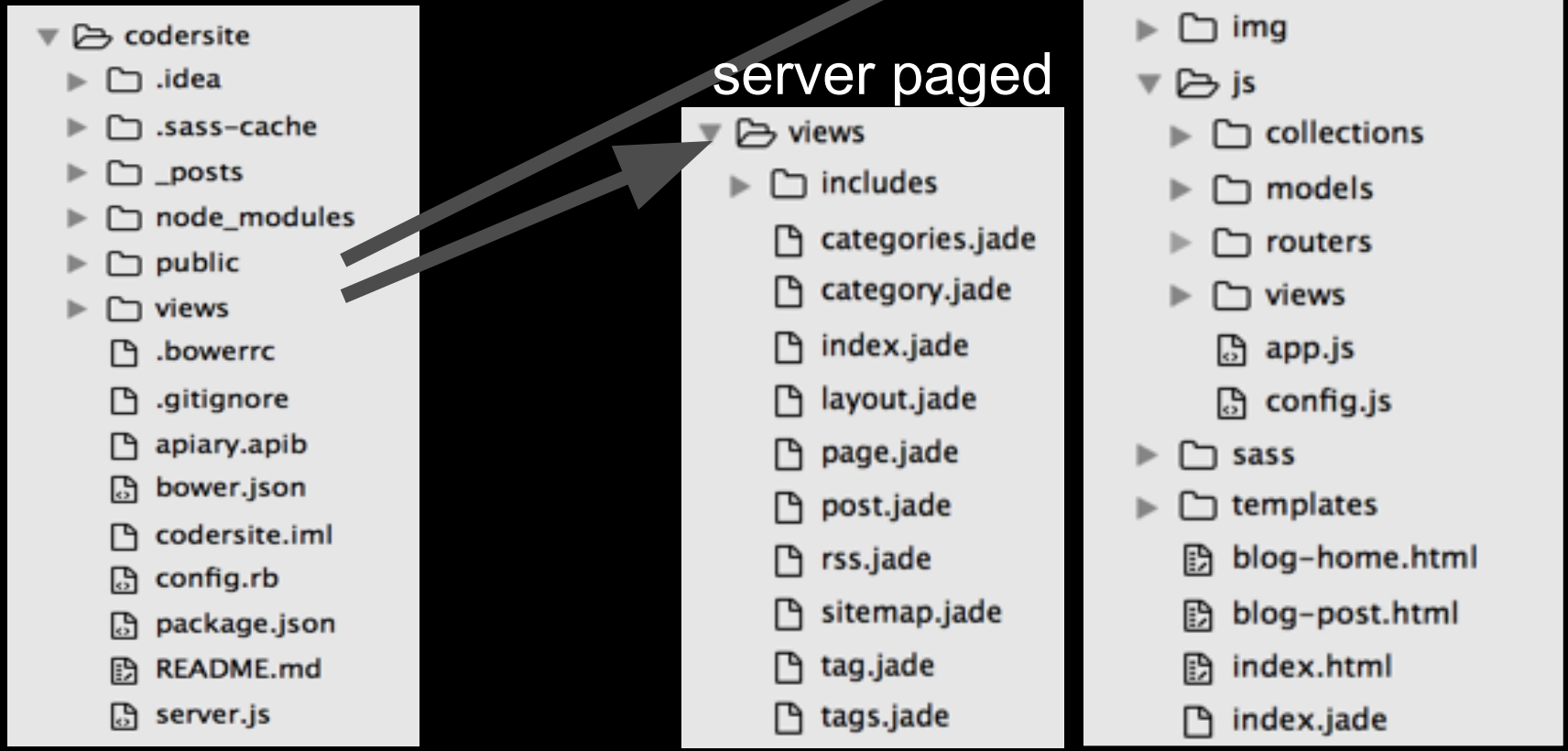
kudo to JavaScript on the server

cs-fw can run on the server

Dualism

both server-pages and single-paged

Organize project fs



Express

web application framework for node

npm install express --save

our first module install

serving static content

production vs development

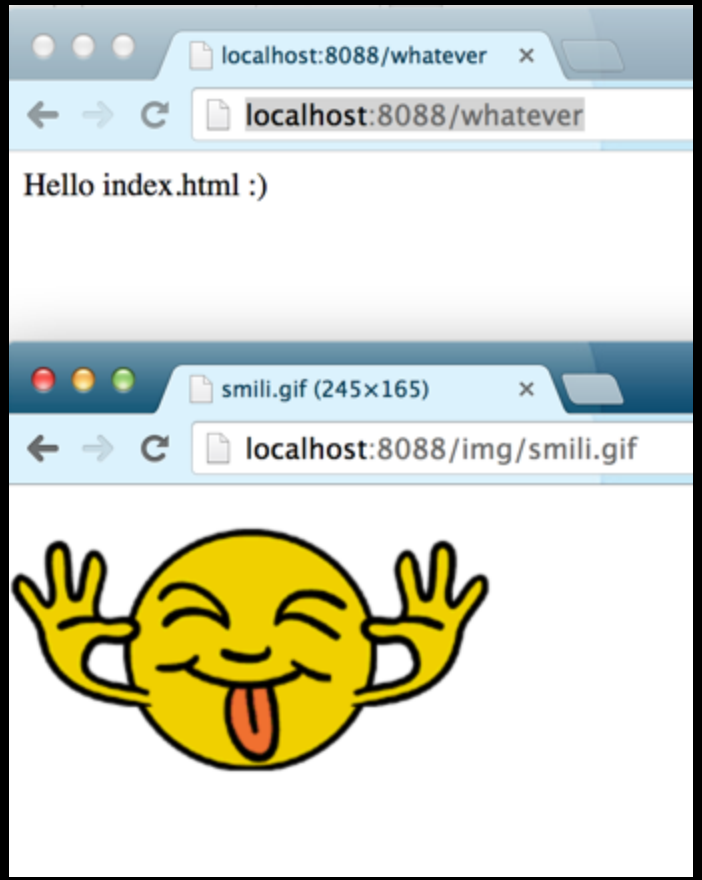
distinguishing between environments

serving the same file

```
/**
 * An express server to serve a SPA app
 * we use express to simplify handling of static files
 * whilst directing all paths to main index.html
 * for handling browser refresh
 * a more complicate solution using standard modules can be found here:
 * https://coderwall.com/p/b2kmkg
 * an interesting discussion:
 * http://stackoverflow.com/questions/7268033/basic-static-file-server-in-nodejs
 */
var express = require('express'),
        app = express(),
        staticDir= (process.env.NODE_ENV || "development") === "production" ? "dist" : "public";

app.use(express.static(__dirname + '/' + staticDir));

app.get('*', function (req, res) {
    res.sendFile(__dirname + '/' + staticDir + '/index.html');
});

var server = app.listen(process.env.PORT || 8088, function() {
    console.log('Listening on port %d', server.address().port);
});
```

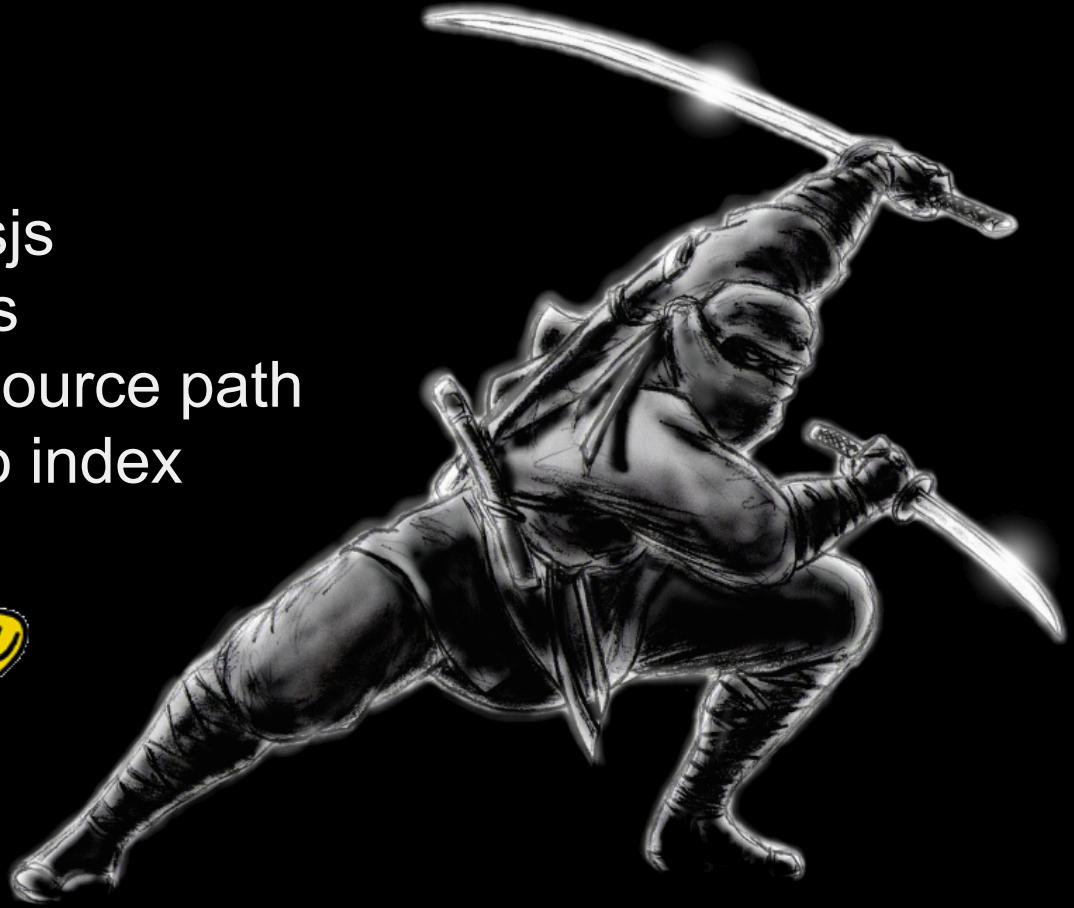
run server

$ node server.js


localhost:8088/whatever
localhost:8088/whatever
Hello index.html :)
smili.gif (245×165)
localhost:8088/img/smili.gif

Exercise

install expressjs
setup server.js
show both resource path
and redirect to index



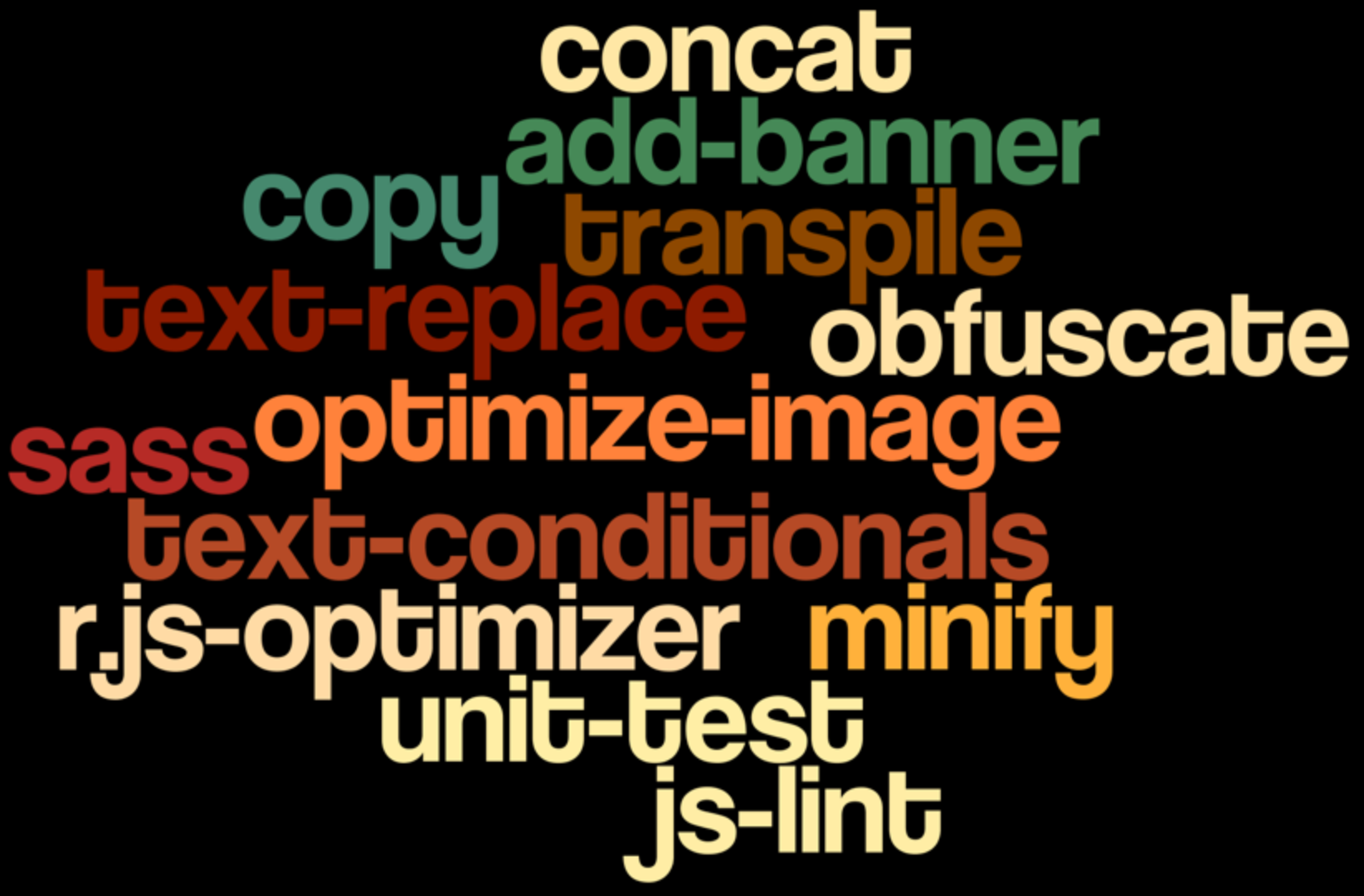
20min

GRUNT

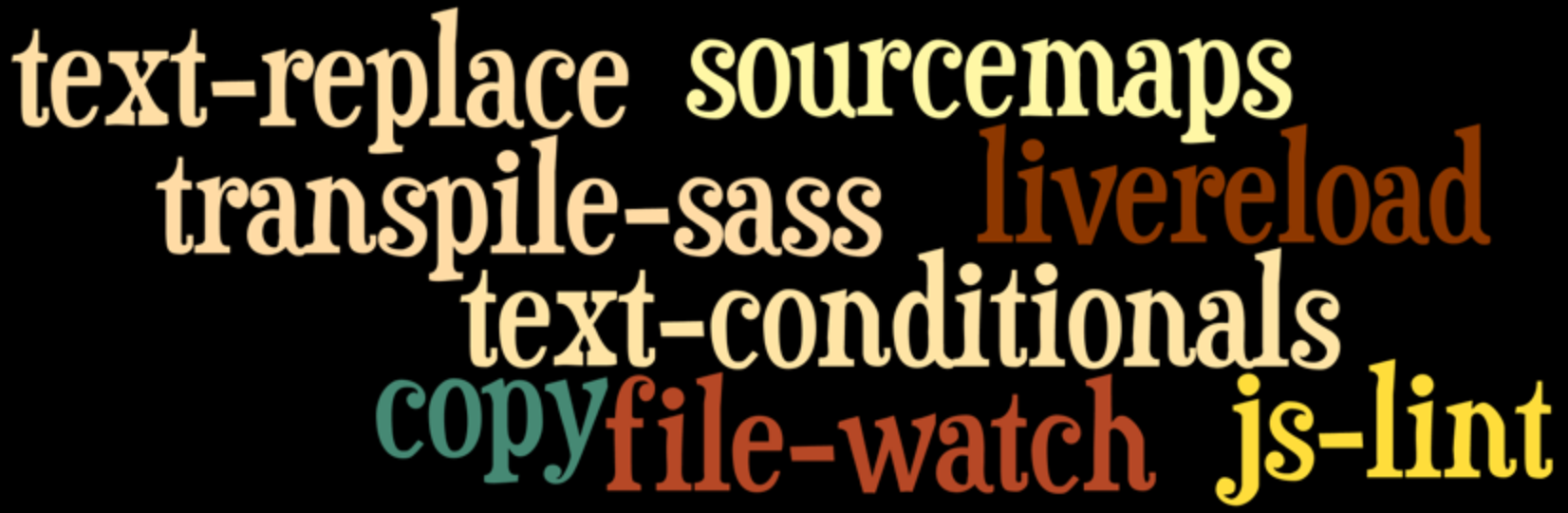
The JavaScript Task Runner

what will it be used for?

packaging a release dist


concat
add-banner
copy
transpile
text-replace
obfuscate
sass
optimize-image
text-conditionals
r.js-optimizer
minify
unit-test
js-lint

aiding development


text-replace
sourcemaps
transpile-sass
livereload
text-conditionals
copy
file-watch
js-lint

over 3,453 plugins

and counting

you'll need the cli

npm install -g grunt-cli

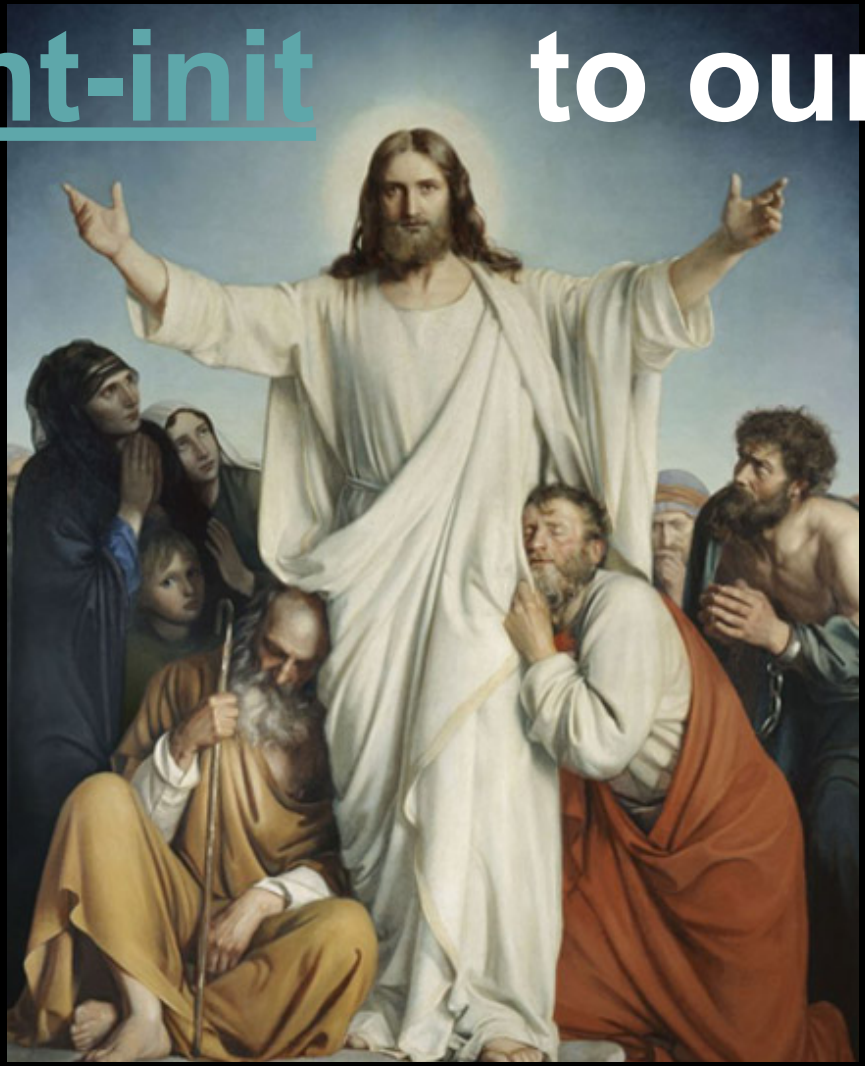
you'll also need it in your project

npm install grunt --save-dev

all we need is Gruntfile.js

we need to write it ourselves?

grunt-init to our aid



$ grunt-init gruntfile

```
➜ test2  grunt-init gruntfile
Running "init:gruntfile" (init) task
This task will create one or more files in the current directory, based on the
environment and the answers to a few questions. Note that answering "?" to any
question will show question-specific help and answering "none" to most questions
will leave its value blank.

"gruntfile" template notes:
This template tries to guess file and directory paths, but you will most likely
need to edit the generated Gruntfile.js file before running grunt. If you run
grunt after generating the Gruntfile, and it exits with errors, edit the file!

Please answer the following:
[?] Is the DOM involved in ANY way? (Y/n)
[?] Will files be concatenated or minified? (Y/n)
[?] Will you have a package.json file? (Y/n)
[?] Do you need to make any changes to the above before continuing? (y/N)

Writing Gruntfile.js...OK
Writing package.json...OK

Initialized from template "gruntfile".

Done, without errors.
```

```
/*global module:false*/
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    // Metadata.
    pkg: grunt.file.readJSON('package.json'),
    banner: '/*! <%= pkg.title || pkg.name %> - v<%= pkg.version %> - ' +
      '<%= grunt.template.today("yyyy-mm-dd") %>\n' +
      '<%= pkg.homepage ? "* " + pkg.homepage + "\\n" : "" %>' +
      '* Copyright (c) <%= grunt.template.today("yyyy") %> <%= pkg.author.name %>;' +
      ' Licensed <%= _.pluck(pkg.licenses, "type").join(", ") %> */\n',
    // Task configuration.
    concat: {
      options: {
        banner: '<%= banner %>',
        stripBanners: true
      },
      dist: {
        src: ['lib/<%= pkg.name %>.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },
    ...
  });

  // These plugins provide necessary tasks.
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-qunit');
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  // Default task.
  grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

};
```

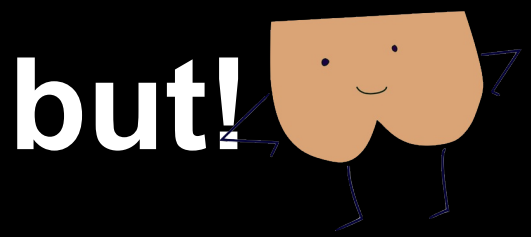
config

load

register

File Structure

```
/*global module:false*/
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    // Metadata.
    pkg: grunt.file.readJSON('package.json'),
    banner: '/*! <%= pkg.title || pkg.name %> - v<%= pkg.version %> - ' +
      '<%= grunt.template.today("yyyy-mm-dd") %>\n' +
      '<%= pkg.homepage ? "* " + pkg.homepage + "\\n" : "" %>' +
      '* Copyright (c) <%= grunt.template.today("yyyy") %> <%= pkg.author.name %>;' +
      ' Licensed <%= _.pluck(pkg.licenses, "type").join(", ") %> */\n',
    // Task configuration.
    concat: {
      options: {
        banner: '<%= banner %>',
        stripBanners: true
      },
      dist: {
        src: ['lib/<%= pkg.name %>.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },
    ...
  });

  // These plugins provide necessary tasks.
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  // Default task.
  grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

};
```



load-grunt-tasks
module
can save us
having to explicitly
load the npm task

```
/*global module:false*/
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    // Metadata.
    pkg: grunt.file.readJSON('package.json'),
    banner: '/*! <%= pkg.title || pkg.name %> - v<%= pkg.version %> - ' +
      '<%= grunt.template.today("yyyy-mm-dd") %>\n' +
      '<%= pkg.homepage ? "* " + pkg.homepage + "\\n" : "" %>' +
      '* Copyright (c) <%= grunt.template.today("yyyy") %> <%= pkg.author.name %>;' +
      ' Licensed <%= _.pluck(pkg.licenses, "type").join(", ") %> */\n',
    // Task configuration.
    concat: {
      options: {
        banner: '<%= banner %>',
        stripBanners: true
      },
      dist: {
        src: ['lib/<%= pkg.name %>.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },
    ...
  });

  // These plugins provide necessary tasks.
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-qunit');
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  // Default task.
  grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

};
```

Data from within

pkg.title
concat.dist.dest

Module Version Management

quiz

major.minor.patch

^(major) ~(minor) 1.0.x(patch)
*/latest(latest)

More

"<1.0.0 || >=2.3.1 <2.4.5 || >=2.5.2 <3.0.0"

"git://github.com/flatiron/winston#master"

npm-check-updates

Another general tool that can come handy

```
# updates package.json to latest version
$ npm install -g npm-check-updates
$ npm-check-updates -u
$ npm install
```

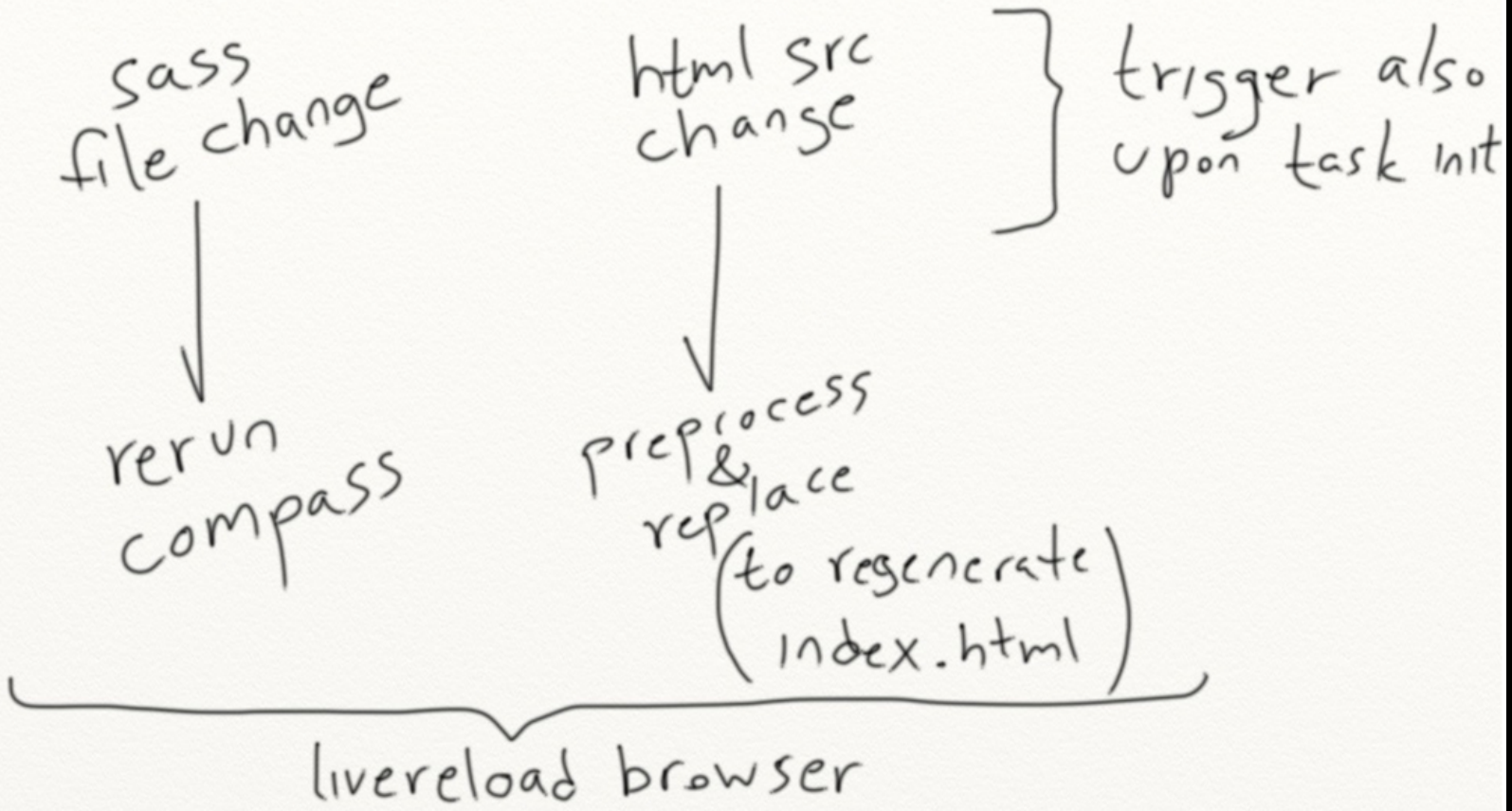
```
➜  myBlog git:(master) ✗ npm-check-updates -u
npm http GET https://registry.npmjs.org/grunt
npm http GET https://registry.npmjs.org/grunt-contrib-jshint
npm http GET https://registry.npmjs.org/grunt-contrib-uglify
npm http GET https://registry.npmjs.org/grunt-contrib-qunit
npm http GET https://registry.npmjs.org/grunt-contrib-concat
npm http GET https://registry.npmjs.org/grunt-contrib-watch
npm http 304 https://registry.npmjs.org/grunt-contrib-concat
npm http 304 https://registry.npmjs.org/grunt-contrib-jshint
npm http 200 https://registry.npmjs.org/grunt-contrib-uglify
npm http 200 https://registry.npmjs.org/grunt-contrib-qunit
npm http 200 https://registry.npmjs.org/grunt
npm http 200 https://registry.npmjs.org/grunt-contrib-watch

"grunt" can be updated from ~0.4.2 to ~0.4.5 (Installed: 0.4.5, Latest: 0.4.5)
"grunt-contrib-jshint" can be updated from ~0.7.2 to ~0.10.0 (Installed: 0.10.0, Latest: 0.10.0)
"grunt-contrib-watch" can be updated from ~0.5.3 to ~0.6.1 (Installed: 0.6.1, Latest: 0.6.1)
"grunt-contrib-qunit" can be updated from ~0.3.0 to ~0.5.1 (Installed: none, Latest: 0.5.1)
"grunt-contrib-concat" can be updated from ~0.3.0 to ~0.4.0 (Installed: none, Latest:
"grunt-contrib-uglify" can be updated from ~0.2.7 to ~0.5.0 (Installed: 0.5.0, La

package.json upgraded
➜  myBlog git:(master) ✗ npm install
```

Version Management cli plugin

Building a Release

clean → compass → css minify → lint js → require consolidate js → html replace vars → cache bust → omit html fragments → obfuscate js → add banner to srcs → package → clean

Grunt Watch


sass file change
rerun compass
html src change
preprocess & replace
(to regenerate index.html)
trigger also upon task init
livereload browser

The Grunt Recipe

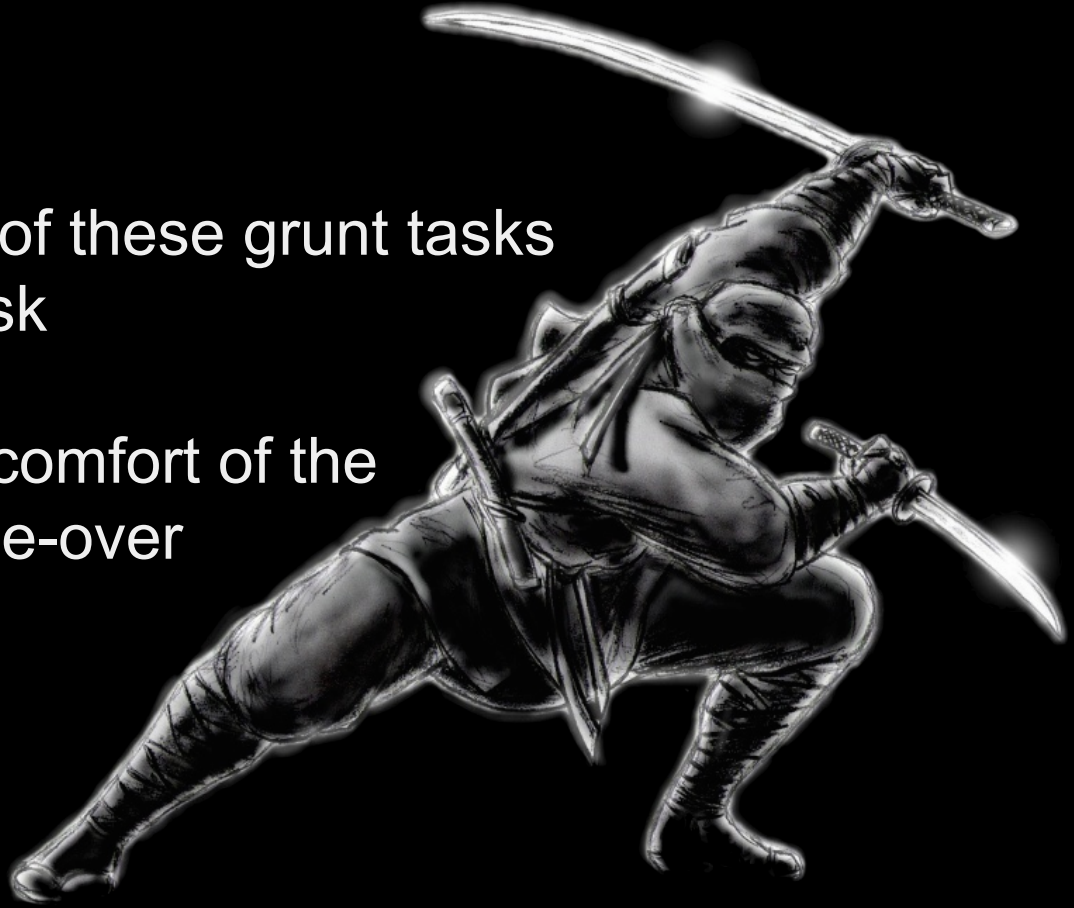
```
"devDependencies": {
  "grunt": "^0.4.5",
  "grunt-contrib-clean": "~0.6.0",
  "grunt-contrib-compass": "^1.0.1",
  "grunt-contrib-copy": "^0.5.0",
  "grunt-contrib-cssmin": "^0.10.0",
  "grunt-contrib-jshint": "~0.10.0",
  "grunt-contrib-requirejs": "^0.4.4",
  "grunt-contrib-uglify": "~0.5.1",
  "grunt-contrib-watch": "^0.6.1",
  "grunt-preprocess": "^4.0.0",
  "grunt-replace": "~0.7.9",
  "load-grunt-tasks": "^0.6.0"
},
```

Exercise

wire as many of these grunt tasks into a build task

*use with the comfort of the clis we've gone-over

ex
tips
ref



1.5h

Solution

Bower

A package manager for the web

“and peace shall come upon us”

it’s just another node package

npm install -g bower

resembles npm syntax

bower_components is to bower
as
node_modules is to npm

bower init

generates bower.json

search for components over 17,583 of them

bower search <package>

Do not install project devDependencies

bower install -p

.bowerrc

configure bower

let’s move cs libs to server root

```
{
    "directory" : "public/bower_components"
}
```

What we'll need:

```
"devDependencies": {
  "bootstrap": "latest",
  "marionette": "latest",
  "requirejs": "latest",
  "requirejs-text": "latest",
  "mustache": "latest"
}
```

Exercise

setup bower and install the mentioned libraries

ex

20min

Developers Toolbox

Sublime IDE

why not IDEA?

Good enough reasons

1. biggest plugin ecosystem (Package Control)
2. multiple cursors
3. shortcuts like move-lines actually work
4. lightweight and fast

All this for just

1 license: USD $70

10+ licenses: USD $60 / license

25+ licenses: USD $55 / license

50+ licenses: USD $50 / license

emmet

cheatsheet

livereload

no full page refresh on edit

taglet or browser extension

css sourcemaps

edit sass directly through devtool

compass alpha version

gem install compass --pre

don’t forget to update sass gem

gem update sass

browser support

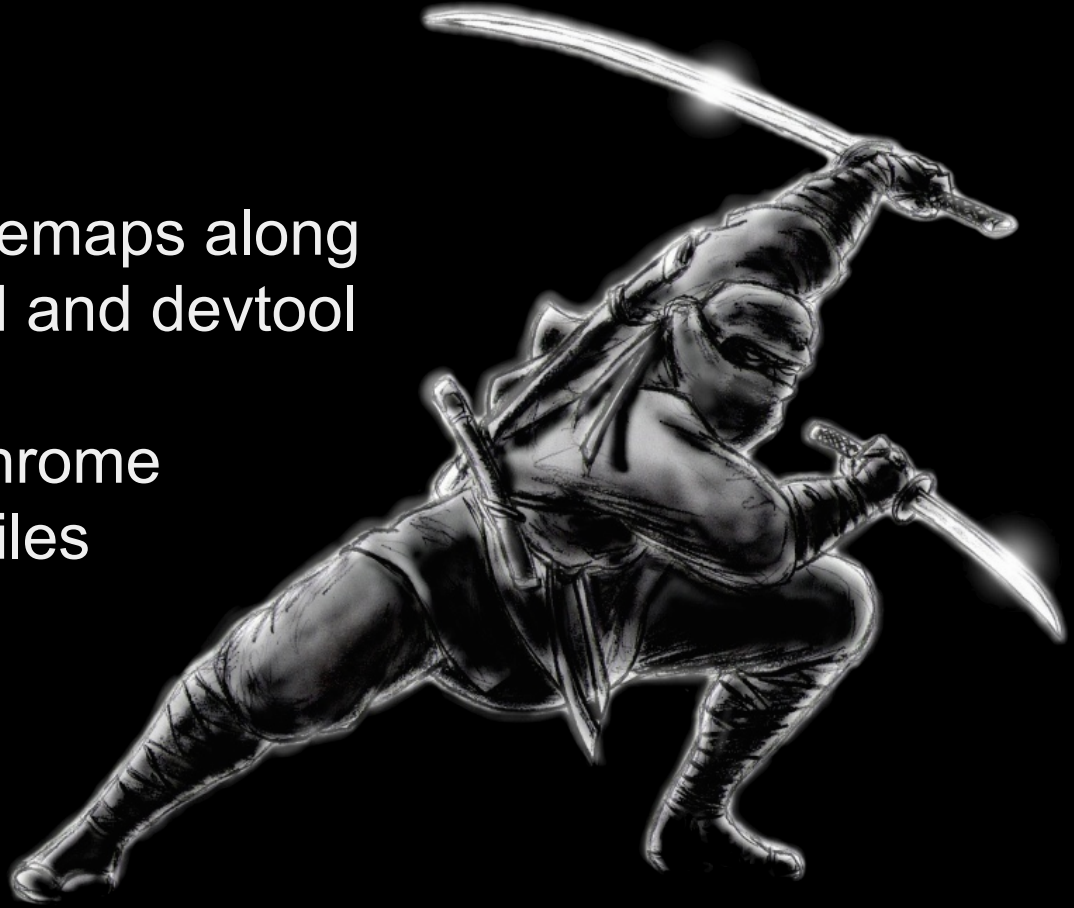
Chrome DevTool filesystem support

livestyle

diff your devtool styling session

Exercise

use css sourcemaps along
with livereload and devtool
fs support
to modify in chrome
devtool sass files



ex

1h

requirejs and r.js

how they differ?

delivered through Grunt

grunt-requirejs or grunt-contrib-requirejs

require config

path, shim and all the others

require config exports

demo

include
vs
findNestedDependencies

require vs define

link

Exercise

configure require
create a single js file for release

ex

1h