**Ruby on Rails**

Railsinstaller.org

Phantomjs.org

Then add the bin directory to PATH.

1. rails new appname
2. rails server (or rails s)
3. rails generate controller controller\_name [action]
4. To embed ruby into the html -> <%= … %>
5. Remember to configure routes (can see them with rake routes)
6. gem install httparty (for requests)
7. When changing the Gemfile –> bundle install or bundle update

Note: To deploy to Heroku:

1. Install rails\_12factor gem
2. Put the next ones (change them if already there in Gemfile)

gem ‘sqlite3’, group: :development

gem ‘rails\_12factor’, group: :production

1. heroku login
2. heroku create app\_name
3. git push heroku master

Note 2: For Scaffold:

1. rails g scaffold car make color year:integer (String is assumed for types) (make, color and year are the columns)
2. Then, to apply it: rake db:migrate
3. To see it, go to localhost:3000/cars, or localhost:3000/cars.json
4. To rollback: rake db:rollback

More examples:

rails g migration add\_price\_to\_cars ‘price:decimal{10,2}’ (then rake db:migrate)

rails g migration rename\_make\_to\_company (This will generate an empty method, you must put manually, in this case, rename\_column :cars, :make, :company

Note: There is a file, schema.rb, where you can load the previous snapshot with rake db:schema:load

Note 3: To configure database:

1. Go to config->database.yaml and specify development: database: db/development.sqlite3
2. To go to the db console -> rails db
3. Then .tables, .headers on, .mode columns, select \* from cars, etc.
4. To get out, .exit
5. Optional: Use something like DB SQLite browser

**NodeJS**

1. Go to  [https://nodejs.org](https://nodejs.org/)
2. Check version with node –v and npm –v

**Less**

1. npm install -g less
2. To compile the file: lessc mystyles.less > mystyles.css

**Bower**

1. npm install -g bower
2. bower init
3. Installing bower components: bower install bootstrap –S, bower install font-awesome –S, …
4. Update links in files, i.e. <link href="bower\_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

**Grunt**

1. install grunt cli:

npm install -g grunt-cli

1. Create package.json in project folder with the following:

{

"name": "conFusion",

"private": true,

"devDependencies": {},

"engines": {

"node": ">=0.10.0"

}

}

1. Install grunt into project folder:

npm install grunt --save-dev

1. Create Gruntfile.js. The JSHint task is set to examine all the JavaScript files in the app/scripts folder, and the Gruntfile.js and generate any reports of JS errors or warnings. We have set up a Grunt task named build, that runs the jshint task and the build task is also registered as the default task (Code at the end).
2. Into html surround css imports with:

<!-- build:css styles/main.css -->

<link href="../bower\_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="../bower\_components/bootstrap/dist/css/bootstrap-theme.min.css" rel="stylesheet">

<link href="../bower\_components/font-awesome/css/font-awesome.min.css" rel="stylesheet">

<link href="styles/bootstrap-social.css" rel="stylesheet">

<link href="styles/mystyles.css" rel="stylesheet">

<!-- endbuild -->

1. Also for js files:

<!-- build:js scripts/main.js -->

<script src="../bower\_components/angular/angular.min.js"></script>

<script src="scripts/app.js"></script>

<!-- endbuild -->

1. Install JSHint:

npm install grunt-contrib-jshint --save-dev

npm install jshint-stylish --save-dev

npm install time-grunt --save-dev

npm install jit-grunt --save-dev

1. Create .jshintrc file:

{

"bitwise": true,

"browser": true,

"curly": true,

"eqeqeq": true,

"esnext": true,

"latedef": true,

"noarg": true,

"node": true,

"strict": true,

"undef": true,

"unused": true,

"globals": {

"angular": false

}

}

1. You can test it with “grunt” command.
2. Install copy and clean modules:

npm install grunt-contrib-copy --save-dev

npm install grunt-contrib-clean --save-dev

1. Install modules to prepare dist folder:

npm install grunt-contrib-concat --save-dev

npm install grunt-contrib-cssmin --save-dev

npm install grunt-contrib-uglify --save-dev

npm install grunt-filerev --save-dev

npm install grunt-usemin --save-dev

1. Install watch and connect modules:

npm install grunt-contrib-watch --save-dev

npm install grunt-contrib-connect --save-dev

1. Run “grunt serve” to start browser.

Example of how the Gruntfile should look:

'use strict';

module.exports = function (grunt) {

// Time how long tasks take. Can help when optimizing build times

require('time-grunt')(grunt);

// Automatically load required Grunt tasks

require('jit-grunt')(grunt, {

useminPrepare: 'grunt-usemin'

});

// Define the configuration for all the tasks

grunt.initConfig(

{

pkg: grunt.file.readJSON('package.json'),

// Make sure code styles are up to par and there are no obvious mistakes

jshint:

{

options:

{

jshintrc: '.jshintrc',

reporter: require('jshint-stylish')

},

all:

{

src:

[

'Gruntfile.js',

'app/scripts/{,\*/}\*.js'

]

}

},

useminPrepare:

{

html: 'app/menu.html',

options:

{

dest: 'dist'

}

},

// Concat

concat:

{

options:

{

separator: ';'

},

// dist configuration is provided by useminPrepare

dist: {}

},

// Uglify

uglify:

{

// dist configuration is provided by useminPrepare

dist: {}

},

cssmin:

{

dist: {}

},

// Filerev

filerev:

{

options:

{

encoding: 'utf8',

algorithm: 'md5',

length: 20

},

release:

{

// filerev:release hashes(md5) all assets (images, js and css )

// in dist directory

files:

[{

src:

[

'dist/scripts/\*.js',

'dist/styles/\*.css',

]

}]

}

},

// Usemin

// Replaces all assets with their revved version in html and css files.

// options.assetDirs contains the directories for finding the assets

// according to their relative paths

usemin:

{

html: ['dist/\*.html'],

css: ['dist/styles/\*.css'],

options:

{

assetsDirs: ['dist', 'dist/styles']

}

},

copy:

{

dist:

{

cwd: 'app',

src: [ '\*\*','!styles/\*\*/\*.css','!scripts/\*\*/\*.js' ],

dest: 'dist',

expand: true

},

fonts:

{

files:

[

{

//for bootstrap fonts

expand: true,

dot: true,

cwd: 'bower\_components/bootstrap/dist',

src: ['fonts/\*.\*'],

dest: 'dist'

},

{

//for font-awesome

expand: true,

dot: true,

cwd: 'bower\_components/font-awesome',

src: ['fonts/\*.\*'],

dest: 'dist'

}

]

}

},

watch:

{

copy:

{

files: [ 'app/\*\*', '!app/\*\*/\*.css', '!app/\*\*/\*.js'],

tasks: [ 'build' ]

},

scripts:

{

files: ['app/scripts/app.js'],

tasks:[ 'build']

},

styles:

{

files: ['app/styles/mystyles.css'],

tasks:['build']

},

livereload:

{

options:

{

livereload: '<%= connect.options.livereload %>'

},

files:

[

'app/{,\*/}\*.html',

'.tmp/styles/{,\*/}\*.css',

'app/images/{,\*/}\*.{png,jpg,jpeg,gif,webp,svg}'

]

}

},

connect:

{

options:

{

port: 9000,

// Change this to '0.0.0.0' to access the server from outside.

hostname: 'localhost',

livereload: 35729

},

dist:

{

options:

{

open: true,

base:

{

path: 'dist',

options:

{

index: 'menu.html',

maxAge: 300000

}

}

}

}

},

clean:

{

build:

{

src: [ 'dist/']

}

}

});

grunt.registerTask('build', [

'clean',

'jshint',

'useminPrepare',

'concat',

'cssmin',

'uglify',

'copy',

'filerev',

'usemin'

]);

grunt.registerTask('serve',['build','connect:dist','watch']);

grunt.registerTask('default',['build']);

};

**Gulp**

1. package.json with the following:

{

"name": "conFusion",

"private": true,

"devDependencies": {

},

"engines": {

"node": ">=0.10.0"

}

}

1. npm install -g gulp
2. Then, in the project folder: npm install gulp --save-dev
3. Install all modules with: npm install jshint gulp-jshint jshint-stylish gulp-imagemin gulp-concat gulp-uglify gulp-minify-css gulp-usemin gulp-cache gulp-changed gulp-rev gulp-rename gulp-notify browser-sync del --save-dev
4. Create gulpfile.js (Code at the end)
5. Run default task (clean, jshint, usemin, imagemin, copyfont) with the command: gulp
6. To start browser watch: gulp watch

var gulp = require('gulp'),

minifycss = require('gulp-minify-css'),

jshint = require('gulp-jshint'),

stylish = require('jshint-stylish'),

uglify = require('gulp-uglify'),

usemin = require('gulp-usemin'),

imagemin = require('gulp-imagemin'),

rename = require('gulp-rename'),

concat = require('gulp-concat'),

notify = require('gulp-notify'),

cache = require('gulp-cache'),

changed = require('gulp-changed'),

rev = require('gulp-rev'),

browserSync = require('browser-sync'),

del = require('del');

gulp.task('jshint', function() {

return gulp.src('app/scripts/\*\*/\*.js')

.pipe(jshint())

.pipe(jshint.reporter(stylish));

});

// Clean

gulp.task('clean', function() {

return del(['dist']);

});

// Default task

gulp.task('default', ['clean'], function() {

gulp.start('usemin', 'imagemin','copyfonts');

});

gulp.task('usemin',['jshint'], function () {

return gulp.src('./app/menu.html')

.pipe(usemin({

css:[minifycss(),rev()],

js: [uglify(),rev()]

}))

.pipe(gulp.dest('dist/'));

});

// Images

gulp.task('imagemin', function() {

return del(['dist/images']), gulp.src('app/images/\*\*/\*')

.pipe(cache(imagemin({ optimizationLevel: 3, progressive: true, interlaced: true })))

.pipe(gulp.dest('dist/images'))

.pipe(notify({ message: 'Images task complete' }));

});

gulp.task('copyfonts', ['clean'], function() {

gulp.src('./bower\_components/font-awesome/fonts/\*\*/\*.{ttf,woff,eof,svg}\*')

.pipe(gulp.dest('./dist/fonts'));

gulp.src('./bower\_components/bootstrap/dist/fonts/\*\*/\*.{ttf,woff,eof,svg}\*')

.pipe(gulp.dest('./dist/fonts'));

});

// Watch

gulp.task('watch', ['browser-sync'], function() {

// Watch .js files

gulp.watch('{app/scripts/\*\*/\*.js,app/styles/\*\*/\*.css,app/\*\*/\*.html}', ['usemin']);

// Watch image files

gulp.watch('app/images/\*\*/\*', ['imagemin']);

});

gulp.task('browser-sync', ['default'], function () {

var files = [

'app/\*\*/\*.html',

'app/styles/\*\*/\*.css',

'app/images/\*\*/\*.png',

'app/scripts/\*\*/\*.js',

'dist/\*\*/\*'

];

browserSync.init(files, {

server: {

baseDir: "dist",

index: "menu.html"

}

});

// Watch any files in dist/, reload on change

gulp.watch(['dist/\*\*']).on('change', browserSync.reload);

});