Ionic

1. npm install cordova ionic -g
2. ionic start ProjectName TemplateName

Yo

See Yeoman workflow.

Yo is a tool that helps start projects right away and is used together with bower, grunt, etc.

npm install yo -g
npm install generator-angular -g (in case of working in an angular application)

1. Create a project folder.
2. Go to it and type yo angular (In case of wanting an angular application)

Angular

For single page applications:
bower install angular-route -S
or bower install angular-ui-router -S

For json server:
npm install json-server -g
json-server --watch db.json

For REST support:
bower install angular-resource -S
and add: <script src="../bower_components/angular-resource/angular-resource.min.js"></script> (for example)

For unit testing, with Jasmine and Karma:
npm install jasmine-core --save-dev
npm install karma-cli -g
npm install karma-jasmine --save-dev
npm install phantomjs karma-phantomjs-launcher karma-chrome-launcher --save-dev
bower install angular-mocks -S

For end to end testing, with protractor:
npm install protractor -g
webdriver-manager update

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

3. heroku login
4. heroku create app_name
5. 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
5. For rails console -> rails c, then something like Car.column_names, Car.primary_key, …
6. To generate a model: rails g model person first_name last_name (remember to rake!)
7. To manipulate the plurals of models: config -> initializers -> inflections

CRUD Operations with Scaffold:

C:

- p1 = Person.new; p1.first_name = "Joe"; p1.last_name = "Smith"; p1.save
- p2 = Person.create(first_name: "Jane", last_name: "Doe")

R:

- Person.all.order(first_name: :desc)
- Person.all.order(first_name: :desc).to_a
- Person.first
- Person.all.first
- Person.all[0] (avoid using this, better use the one above)
- Person.take
- Person.take 2
- Person.all.map { |person| person.first_name }
- Person.pluck(:first_name)
- Person.where(last_name: "Doe")
- Person.where(last_name: "Doe").first
- Person.where(last_name: "Doe")[0] (avoid using this)
- Person.where(last_name: "Doe").pluck(:first_name)
- Person.find_by!(last_name: "Doe")
- Person.count

- Person.all.map { |person| “#{person.first_name} #{person.last_name}” }
- Person.offset(1).limit(1).map { |person| “#{person.first_name} #{person.last_name}” }

U:

- jane = Person.find_by first_name: “Jane”; jane.last_name = “Smithie”; jane.save
- Person.find_by(last_name: “Smith”).update(last_name: “Smithson”)

D:

- jane = Person.find_by first_name: “Jane”; jane.destroy
- joe = Person.find_by first_name: “Joe”; Person.delete(joe.id)

CRUD with SQL fragments (VULNERABLE TO SQL INJECTION):

- Person.where(“age BETWEEN 30 and 33”).to_a
- Person.find_by(“first_name like ‘%man’”)

The default scope enables to specify certain things about queries such as the order of sorting; to change it in, for example, hobby.rb:

```
default_scope { order :name }
```

- Hobby.pluck :name
- Hobby.unscoped.pluck :name

In person.rb

```
scope :ordered_by_age, -> { order age: :desc }
scope :starts_with, -> (starting_string){ where(“first_name LIKE ?”, “#{starting_string}%”)}
```

- Person.ordered_by_age.pluck :age
- Person.ordered_by_age.starts_with(“Jo”).pluck :age, :first_name
- Person.ordered_by_age.limit(2).starts_with(“Jo”).pluck :age, :first_name

Validators enforces certain behavior in tables, for example so that a record is unique in the column. In job.rb:

```
validates: title, :company, presence: true
```

- rails c
- job = Job.new
- job.errors
- job.save
- job.errors
- job.errors.full_messages

Or writing your own validator, in salaryrange.rb:

```
validate :min_is_less_than_max
def min_is_less_than_max
    if min salary > max_salary
        errors.add(:min_salary, “cannot be grater than max salary!”)
    end
end
```

- sr = SalaryRange.create min_salary: 30000.00, max_salary: 10000.00

- sr.errors
- sr.errors.full_messages
- sr.save! (this will throw an exception, without ! just will result in “false”)

The N + 1 Problem happens when querying a lot with queries like Person.all.each { |p| puts p.personal_info_weight }, this will query a lot the personal_infos table. To solve:

- Person.includes(:personal_info).all.each {.....

To make transactions:

```
ActiveRecord::Base.transaction do
    david.withdraw(100)
    mary.deposit(100)
end
```

To prevent SQL injection:

Array Condition Syntax:

- Person.where(“age BETWEEN ? AND ?”, 28, 34).to_a
- Person.where(“first_name LIKE ? OR last_name LIKE ?”, ‘%J%’, ‘%J%’).to_a

Hash Condition Syntax:

- Person.where(“age BETWEEN :min_age AND :max_age”, min_age: 28, max_age: 32).to_a
- Person.where(“first_name LIKE :pattern OR last_name LIKE :pattern”, pattern: ‘%J%’).to_a

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

Full DB example:

- rails g new advanced_ar
- cd advanced_ar
- rails g model person first_name age:integer last_name
- rake db:migrate
- In db/seeds.rb: (if using create instead of create!, it will not tell if it failed.)

```
Person.destroy_all
Person.create ! [
    { first_name: “Kalman”, last_name: “Smith”, age: 33 } ,
    { ….
]
```

- rake db:seed
- rails g migration add_login_pass_to_people login pass
- rake db:migrate
- (Update the seeds.rb file) (pass as plain text because this is an example)
- rails db
- .headers on

- .mode columns
- select * from people;
  Relationships:
- rails g model personal_info height:float weight:float person:references
- rake db:migrate
- bill = Person.find_by first_name: “Bill”
- bill.personal_info (it will be nil)
- pi1 = PersonalInfo.create height: 6.5, weight: 220
- bill.personal_info = pi1 (the first one will lose it’s reference)
- bill.build_personal_info height: 6.0, weight: 180
- bill.save
- rails g model job title company position_id person:references
- rake db:migrate
- person.rb:
  class Person < ActiveRecord::Base
  has_one :personal_info
  has_many :jobs
  end
- job.rb:
  class Job < ActiveRecord:Base
  belongs_to :person
  end
- rails c
- ActiveRecord::Base.logger = nil
- Job.create company: “MS”, title: “Developer”, position_id: “#1234”
- p1 = Person.first
- p1.jobs (will be nil)
- p1.jobs << Job.first
- Job.first.person
  Many to Many relationship:
- rails g model hobby name
- rails g migration create_hobbies_people person:references hobby:references
- In ######_create_hobbies_people.rb, add:
  create_table :hobbies_people, id: false do |t|...
- rake db:migrate
- In person.rb:
  has_and_belongs_to_many: hobbies
- In hobby.rb:
  has_and_belongs_to_many :people
- josh = Person.find_by first_name: “Josh”
- lebron = Person.find_by first_name: “LeBron”
- programming = Hobby.create name: “Programming”
- josh.hobbies << programming; lebron.hobbies << programming
- programming.people
  Rich Many to Many Relationships
- rails g model salary_range min_salary:float max_salary:float job:references

- rake db:migrate
- job.rb and salary_range.rb:
  belongs_to :person
  has_one :salary_range

  belongs_to: job
- person.rb:
  has_many :jobs
  has_many :approx_salaries, through: :jobs, source: :salary_range
- lebron = Person.find_by(first_name: “LeBron”)
- lebron.jobs.count
- lebron.jobs.pluck(:id) (if result is [12, 13]
- Job.find(12).create_salary_range(min_salary: 10000.00, max_salary: 20000.00)
- Job.find(13).create….
- lebron.approx_salaries
- To make it better: in person.rb
  def max_salary
  approx_salaries.maximum(:max_salary)
  end
- lebron = Person.find_by last_name: “James”
- lebron.max_salary

Other methods: person.jobs = jobs (replaces existing jobs)
person.jobs.clear (Disassociates jobs from person, sets FK to NULL)

- In seeds.rb you can do something like:
  Person.first.jobs.create! [….
- Person.first.jobs.where(company: “MS”).count
- In person.rb, optional you can:
  has_many :jobs
  has_many :my_jobs, class_name: “Job” (it’s just an alias)
- The option :dependent allows specify :delete, :destroy or :nullify
  In person.rb
  has_one :personal_info, dependent: :destroy
- Then, mike.destroy (before, mike = Person….)

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
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

2) Create package.json in project folder with the following:

```
{
  "name": "conFusion",
  "private": true,
  "devDependencies": {},

  "engines": {
    "node": ">=0.10.0"
  }
}
```

3) Install grunt into project folder:

npm install grunt --save-dev

4) 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).

5) 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 -->
```

6) 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 -->
```

7) 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
```

8) 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
  }
}
```

9) You can test it with “grunt” command.

10) Install copy and clean modules:

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

11) 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
12) Install watch and connect modules:
npm install grunt-contrib-watch --save-dev
npm install grunt-contrib-connect --save-dev
13) 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"
  }
}
```

2) npm install -g gulp

3) Then, in the project folder: npm install gulp --save-dev

4) 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

5) Create gulpfile.js (Code at the end)

6) Run default task (clean, jshint, usemin, imagemin, copyfont) with the command: gulp

7) To start browser watch: gulp watch

8) When working with $scope, uglify task causes some issues, so we need to install: npm install gulp-ng-annotate --save-dev

```
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'),
         ngannotate = require('gulp-ng-annotate'),
        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: [ngannotate(),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);
            });
```