

# NativeQDA

## How to Run Locally

- Download and install Node v6 - [Node.js Download](#)
- Download and install MongoDB - [MongoDB Download](#)
- Clone repo from Github (The development repo will have the most recent commits)
- Setup PATH Variables for Git and MongoDB

Development repo deploys to [nativeqda-dev](#) every push to master

```
git clone git@github.com:Aidan275/nativeqda-dev.git
```

Production repo deploys to [nativeqda](#) every push to master

```
git clone git@github.com:Aidan275/nativeqda.git
```

- Change to the root dir and download dependencies using NPM

```
cd nativeqda-dev
npm install
```

This will download all the dependencies for the app into the node\_modules folder, it may take a minute (~150 MB including dev dependencies).

- Install [gulp](#) globally to build and serve the development and production environments

```
npm install -g gulp
```

Gulp is used to automate deployment to production and provides additional development features.

Some of the commands that exist at the moment include:

```
gulp serve          /* Inject the paths into index.html then serves the files for development
                     located in /src and watches these files and reloads the app/browser
                     when changes are made */

gulp build          /* Builds the optimised app for production. Copies optimised code
                     to the dist folder */

gulp serve-build    /* This will watch the AngularJS JS files, vendor JS, vendor CSS,
                     scripts, styles, images, fonts, and AngularJS HTML and run the
                     appropriate tasks to bundle/minify/copy/etc. the modified files
                     when changes are detected and reload the app/browser.    */
```

The scripts for these commands are located in gulpfile.js.

**Note:** The app must be built before deploying as this creates the files in the dist directory which are used in production.

- Install [Browsersync](#) globally to automatically reload the browser when file changes are detected

```
npm install -g browser-sync
```

- Removed the .env file from gitignore so it's now included in pushes/pulls to/from github.

**Note:** This may break others login authentication due to a different JWT\_SECRET key.

- Start MongoDB in a separate command line/terminal with

```
mongod
```

- Try gulp serve to run the app in its development environment
- Not working? Try going to localhost:3000 instead.

## Documentation

Documentation is generated from the inline comments stored in the source code.

This means that all the docs are kept in sync as the code changes.

For the front-end we use [ngDoc](#), a form of jsDoc, with the task runner Grunt to generate the documentation.

For the back-end we use [apiDoc](#), which generates RESTful web API documentation using the inline comments, also using the task runner Grunt.

To generate the documentation, use the following command

```
grunt docs
```

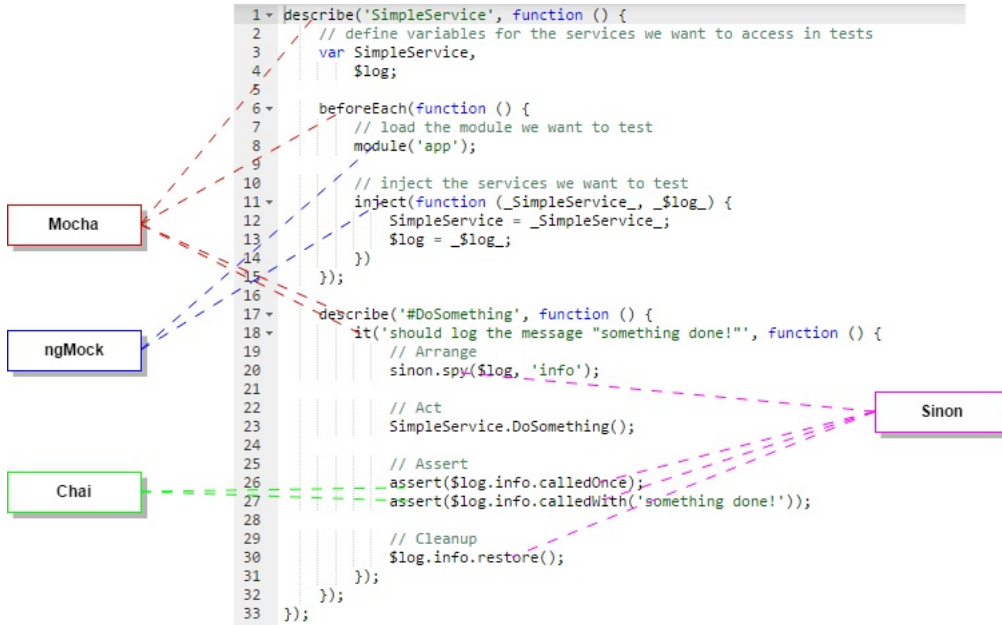
This should start a server at <http://localhost:8000> where the documentation can be viewed.

The documentation is also hosted at <http://docs.nativeqda.xyz>, which we will update periodically.

# Testing

Unit tests are run using the following set of tools:

- [Karma](#) - test runner used to run the tests against code
- [Mocha](#) - testing framework used to define our overall unit test with describe, beforeEach and it functions
- [Chai](#) - assertion library used to verify the results of our unit tests
- [Sinon](#) - library used for creating test spies, stubs and mocks in javascript



## Running Unit Tests

To run the unit tests use the following karma command

```
karma start
```

This script will start the Karma test runner to execute the unit tests. Moreover, Karma will sit and watch the source and test files for changes and then re-run the tests whenever any of them change. This is the recommended strategy; if your unit tests are being run every time you save a file then you receive instant feedback on any changes that break the expected code functionality.

- the configuration is found at 'karma.conf.js'
- the unit tests are found next to the code they are testing and are named as '....spec.js'.