**Server Side Render Vue Apps with Nuxt.js**

<https://www.youtube.com/watch?v=ZYUWsjUxxUQ&index=9&list=PLl-K7zZEsYLkbvTj8AUUCfBO7DoEHJ-ME>

Vue comes with a server-side rendering library called Nuxt. Simple and easy to set up compared to other JS frameworks as Nuxt is just something you install and configure and away you go…

Nuxt not only helps you to set up your server-side render our assets but also help to separate our static assets from our server-side assets.

**Getting started**

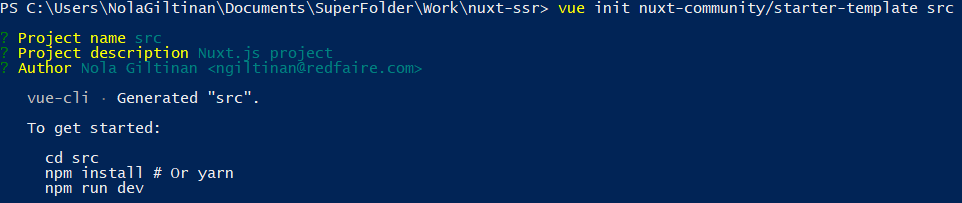
Before anything install the latest stable version of Node from here <https://nodejs.org/en/>. Nuxt requires a Node Version > 8.

**Then in command shell (problems with bash / git / vue-cli combo)**

> mkdir nuxt-ssr && cd $\_

> npm i -g vue-cli

> vue init nuxt-community/starter-template src



> mkdir public && mkdir functions

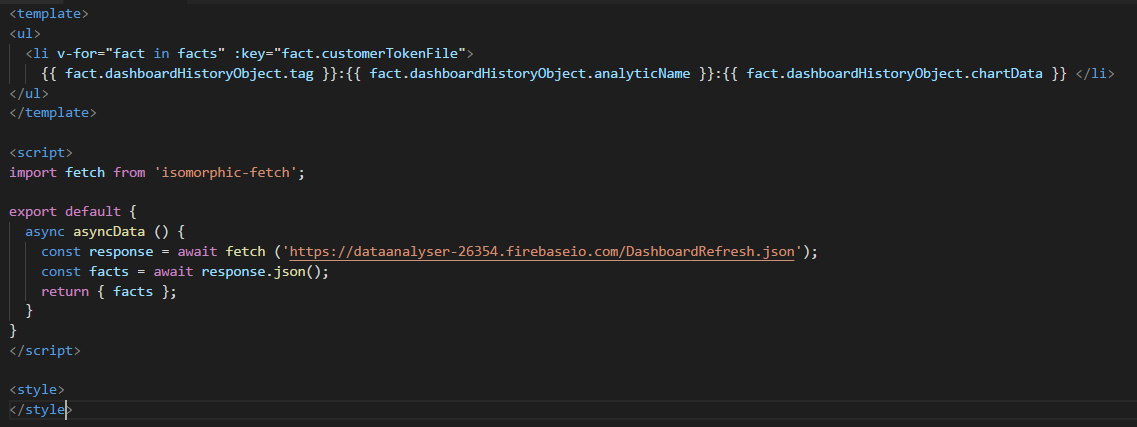
> cd src

> npm install

> npm add isomporphic-fetch

Open up nuxt-ssr folder in IDE to view project layout

Modify index.vue as follows:



> npm run dev

To ensure we can run the app on old browsers add the following to the devDependencies of the package.json:

"babel-plugin-module-resolver":"^2.7.1",

"babel-plugin-transform-runtime":"^6.23.0",

"babel-preset-es2015":"^6.24.1",

"babel-preset-stage-0":"^6.24.1",

> npm install

Then into nuxt.config.js i.e. Nuxt’s configuration file:

Head is the head tag for the App

Build is where you configure how the App gets built, where it gets built to etc..



Add the following to the build section

publicPath: '/public/',

vendor: ['isomorphic-fetch'],

extractCSS: true,

babel: {

presets: ['es2015','stage-0'],

plugins: [

["transform-runtime", {

"polyfill": true,

"regenerator": true

}]

]

},

Also add the following above the build section, to make sure the built app goes into the functions directory

buildDir: '../functions/nuxt',

Create a package.json in the functions folder (otherwise we will get an error on the build in relation to the transform-runtime plugin).



> npm install (in the functions folder)

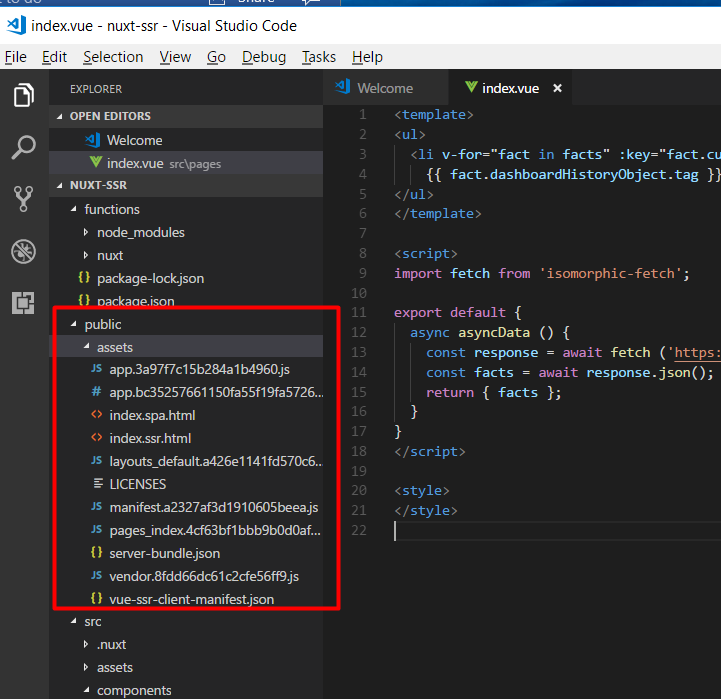
Now build the app using npm (back in the src folder)

> npm run build

We should now have a nuxt folder in our project root with the server dependencies. We also have all our static assests, but we don’t want to deliver our static assets from our function server. We want to deliver our static assets (images, css, client-side js) from our Firebase Hosting CDN. So let’s take the static assets and ensure they go to the public folder and are served from Firebase hosting CDN.

To move the static assets from the nuxt/dist folder to the public folder (in the project root)

> cp -R functions/nuxt/dist public/assets



Now we can set up our server code. In the functions folder we will create an index.js file as follows:



This will server side render the html. It will also set up CDN caching for the dynamic server-side rendered response (with the res.set Cache-Control command).

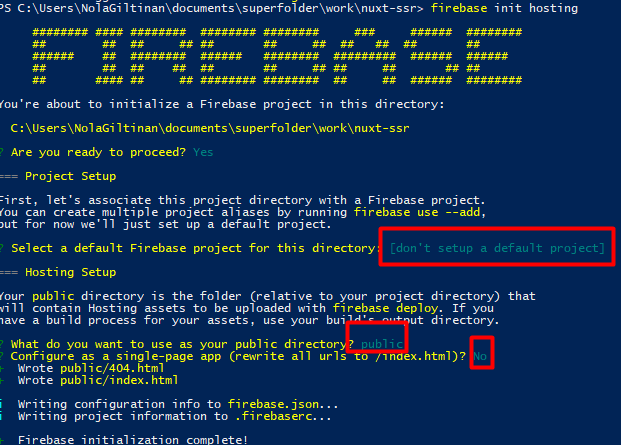


max-age=600 => this will be cached in the user’s local browser cache for 600 seconds

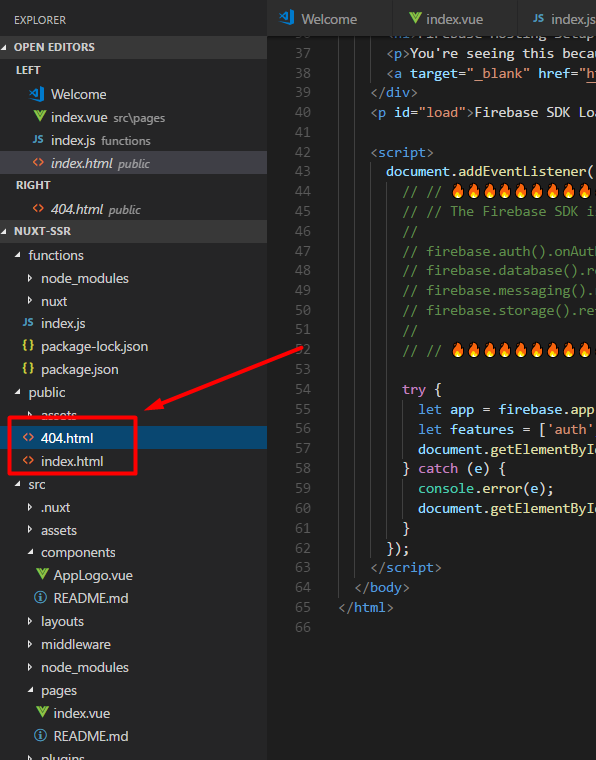
s-maxage=1200 => this will be cached on CDN level for 1200 seconds

So, all set up now. Lets run this locally using firebase init using the firebase cli (see notes below on Firebase cli, if you don’t already have it installed). We do this by

> firebase init hosting



This automatically creates an index.html and 404.html in the public folder which we delete:



Then in firebase.json (in the project root) add a ‘rewrites’ object. The rewrite for the server side rendered app will tell cloud functions in firebase hosting how to communicate with each other.

For all routes, call function ssrapp (as defined in index.js)-.

"rewrites": [{

"source": "\*\*",

"function": "ssrapp"

}]

Now we will need to redefine our publicPath in the build object of both nuxt.config.js and index.js

publicPath: '/',

> cd src

> npm install

> npm run build

Copy assets again

> cd ..

> cp -R functions/nuxt/dist public

> firebase deploy

<https://medium.com/codingthesmartway-com-blog/introduction-to-server-rendered-vue-js-apps-with-nuxt-91dbfd80795a>

**Firebase Hosting**

If Firebase cli was not set up already set up globally using npm i.e.

> npm i -g firebase-tools

> firebase login

This will automatically bring up a Google Login in a browser window, followed by a security prompt.

