*How Angular,* node*.js, npm and other ui stuff is working.*

If divide processes into maven phases it could be slightly clearer:

First of all angular is just a js framework like a Spring in java.

Node.js it is like another browser (actually chrome browser contains the same engine like node js) which could perform java script code and contains web server (like apache/jbos) for communicating through the network.

Often Node.js is used like middleware backend for routing requests to other web services written in java. Or just working like regular web server with connection to java.

Browser it is an engine (could compare it to JVM but actually it a program which performs a lot more – debug, shows console, network – real desktop app) which is pulling inside js files or HTML pages with java script. (If it is JSP old architecture we are sending pages html with js) and start performing it. In our application browser sends requests to Server Apache and apache responds only once with js files. Then browser load them into and start working with those files. One of these files is huge app.bundle.js file which contains whole our application written in JavaScript (JS). That file was compiled from typescript to JavaScript file. To reload that file we should close browser or clean the cash. This is why it is often when changes made but we still see the error because application (browser) is still working with the same file. Then when we put new url or click buttons there is an Angular router (there is a huge documentation made by google team for understanding how router works) which is directing our request to specific peace of code (components) into that huge js file. And what we see when new page after click is opened it only means we have been routed to new component which we loaded by browser. And if new component needs new info from DB in our application it (browser) prepares request using angular controllers and sends it to Apache server which redirect it to Jbos servers with FCRM(Asset) applications and controllers of those web server apps extracting model data and do some operations if it is necessary.

Phases:

- Dependencies

For that npm is responsible. That program like a maven but written in JavaScript pulling all dependencies (from public/private repo) and putting them in local folder *node\_modules*. Command *npm install*.

- Compile

In our application it is webpack. That program like a maven but written in JavaScript. Compiling our Typescript code into poor JavaScript code (js and js.map files) and combine all it into one giant JavaScript file (app.bundle.js). There are implementations (some other programs) where could be many files for different modules. Js.map files it is mapping for debugging purposes - to allow engine(browser) to know which string in js file corresponds to string into initially written by us Typescript file.

In local env we are doing only one compilation when browser is loading application. In dev and higher we are doing 2 compilations one in some Angular JavaScript (in advance) what save us a lot of time during first loading of page. And then doing some faster compilation into pure JavaScript. This why we are not able to debug as 2 files of mapping created and it just doesn't understand how to map it.

- Test

Karma is used. Which actually also is using webpack to compile tests written in Typescript into pure js. And karma has its own server (like apache, jbos) which is wrapper for node.js web server which allows to upgrade js file into engine (browser) each time we made some changes.

- Package

Again webpack which could convert (pack) it into any package (tar, gz and so on). Idea exactly the same like maven. We could also convert them into npm libraries and publish them (npm publish) and be able to load those libraries and use them in our code. Exactly like we are doing with jars and maven dependencies. For instance we have MC header and footer which are some libraries written by guys from MC and we are using them in our UI code. Those libraries are into local MC repo.

- Install

- Deploy

Those phases are performed by Array. Probably some package just gets installed into some place and so on.

When we are doing local *npm run local*  we actually launch node.js but not for performing our application it is just for webpack as it is written in js and need platform to perform it (like java for java applications), also we run node.js for server locally (in dev and higher we use apache) but locally we start some server written in java and working in node.js. That is why we have seen that process into process tasks – it was actually server not our code working into node.js.

Sometimes indeed angular is used in middleware when node.js is launched and performing those code in exact way as regular back end. There is also such stuff like Web workers – it is basically helpers in building html pages and sending those pages to browser to boost performance at client side.

There are some tools which also help us to work with angular projects. For instance Angular CLI it is wrapper of webpack which helps to interact with it using command line interface. Basically when we type something like this ng new my-app then it starts using webpack perform some actions. That one builds new project structure with karma config, when Mitch started our project he didn’t have that CLI so he created everything manually by copping files from public open source repos into our project. CLI also helps run tests and other operations described above – as I mentioned it actually webpack (wrapper of it).