Contents

package.json

npm start defaults to run node server.js

Unless otherwise specified in *package.json - scripts - start*, `npm start` runs as `node server.js`, as long as the current directory has already run npm init, i.e. *package.json* is present.

conditionally chain commands

- `{command1} && {command2}` : only run command2 after the running of command1 results in success
  - `node test.js && zuul -ui tape --local 5325 -- test.js` : only run zuul after local test is passed, i.e. give me the good ui showing me all tests are passed in order to make me feel good

dependencies

- `npm i some_package --save` will add that package into package.json.dependencies
- `npm i some_package --save_dev` will add that package into package.json.devDependencies
- `npm install --registry http://registry.cnpmjs.org` will install node modules as specified by package.json but from the registry url, as opposed to the default registry url https://registry.npmjs.org/

The difference is that, dependencies is what the project needs to run in production, while devDependencies is used only in development but not in production.

> If someone is planning on downloading and using your module in their program, then they probably don't want or need to download and build the external test or documentation framework that you use.
> In this case, it's best to map these additional items in a devDependencies object.
> These things will be installed when doing npm link or npm install from the root of a package, and can be managed like any other npm configuration param. See npm help 7 npm-config for more on the topic.
> - `npm help package.json`

npm commands

- `npm run uncommonScriptName` : run script specified in package.json/scripts.uncommonScriptName. These uncommonScriptName are script names that aren't start, test, i, install, and several others.
- `npm run start:watch` : run script specified in package.json/scripts.start:watch (notice that the colon : here acts as a normal character)

npm modules

Try not to install node-sass

It takes forever.

It is not hard to get around *sass*. We can use javascript inline styles for variable sharing.