**What is Node’s REPL?**

**Ans**: Node.js comes with virtual environment called REPL (aka Node shell). REPL stands for Read-Eval-Print-Loop. It is a quick and easy way to test simple Node.js/JavaScript code. To launch the REPL (Node shell), open command prompt (in Windows) or terminal (in Mac or UNIX/Linux) and type node as shown below.

**Let’s see where node was installed and check the version:**

$ which node

**To verify that your installation was successful let’s give Node’s REPL a try:**

$ node

**Now let’s write a “Hello world” string display program:**

> console.log('Hello World');

**For getting help:**

> .help

**we can now focus our attention on npm, which was included in the install:**

$ which npm

**Node Packaged Modules/third-party dependencies:**

npm can install packages in local or global mode. In local mode it installs the package in a node\_modules folder in your parent working directory. This location is owned by the current user. Global packages are installed in {prefix}/lib/node\_modules/which is owned by root (where {prefix} is usually /usr/ or /usr/local).

**N.B**: sudo to install packages globally

**Changing the Location of Global Packages:**

**$** npm config

$ npm config list

$ npm config get prefix

**Installing Packages in Global Mode:**

$ npm install uglify-js --global

**N.B:** use the --global flag, but this can be abbreviated to –g

**Listing Global Packages:**

$ npm list --global

**We can change that with the --depth=0 option:**

$ npm list -g --depth=0

Uglify package to minify example.js into example.min.js:

$ uglifyjs example.js -o example.min.js

**N.B:** Any packages installed globally will become available from the command line

**Installing Packages in Local Mode:**

When you install packages locally, you normally do so using a package.json file. Let’s go ahead and create one.

$ npm init

package name: (project)

version: (1.0.0)

description: Demo of package.json

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

Press Enter to accept the defaults, then type yes to confirm. This will create a package.json file at the root of the project.

i.e,

{

"name": "project",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

**Tip**: If you want a quicker way to generate a package.json file use npm init –y

“The main field is the primary entry point to your program and the scriptsfield lets you specify script commands that are run at various times in the lifecycle of your package”

**let’s try and install Underscore JS:**

$ npm install underscore

**N.B:** Note that a lockfile is created

If we have a look in package.json we will see that a dependencies field has been added:

{

...

"dependencies": {

"underscore": "^1.8.3"

}

}

**Managing Dependencies with package.json:**