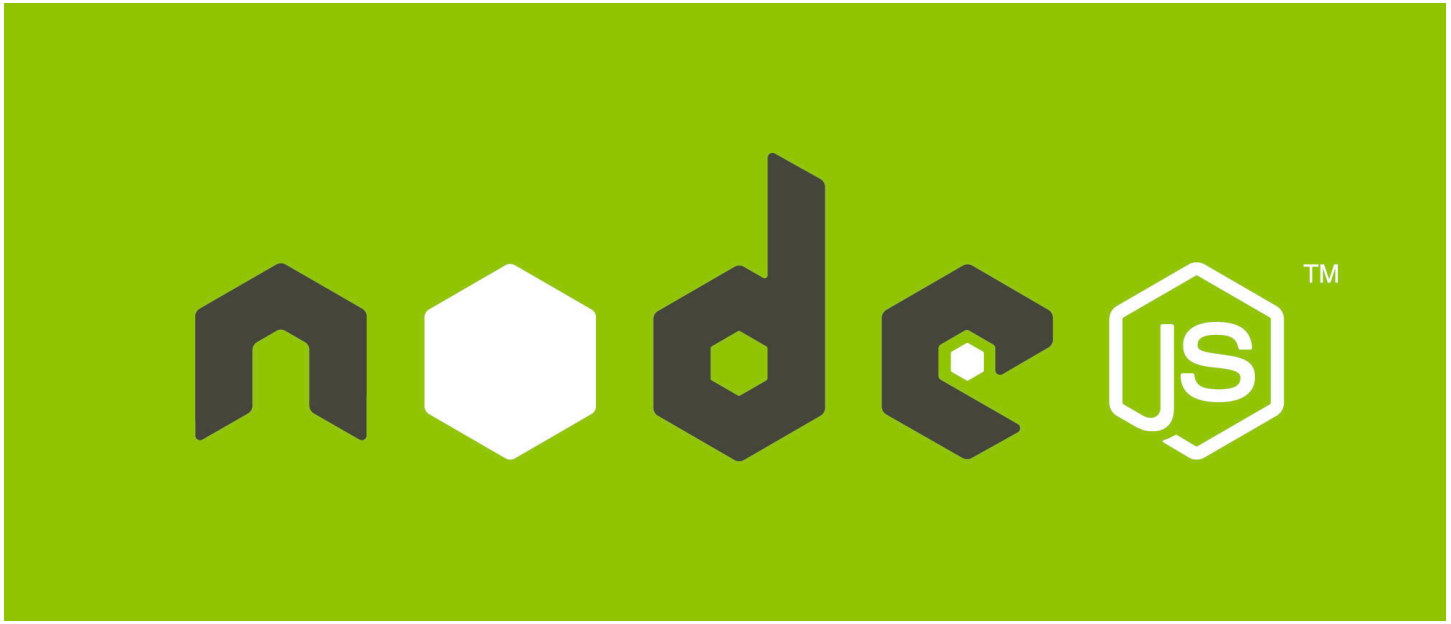
NodeJS

04

Presented by
Subhash EP

Subhash


node
JS
TM

3 Kinds of Modules

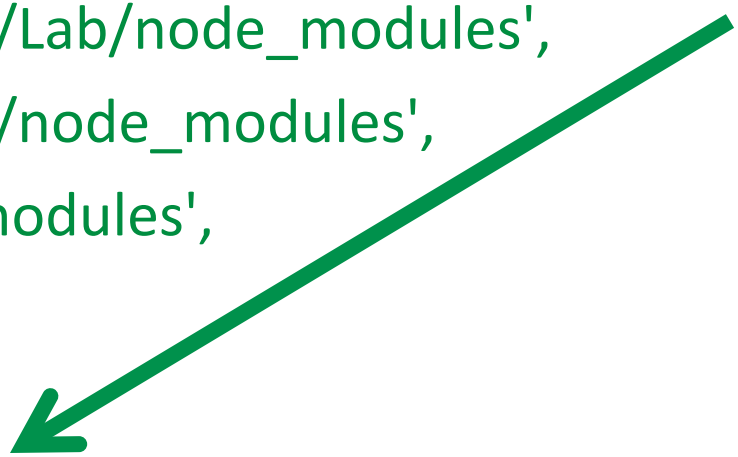
- file-based modules – already discussed
- core modules – already discussed
- external node_modules – let's discuss now

require function scanning

- if module name prefixed with ‘./’ or ‘../’ or ‘/’, then it is assumed to be a file-based
- Otherwise, we look for core modules with the same name
  - require('bar’)
- If no matching core module, we look for a node_module called 'bar'.

require function scanning order

```
paths:
  [
   '/Users/subhash/Node.js/Lab/L03A/node_modules',
   '/Users/subhash/Node.js/Lab/node_modules',
   '/Users/subhash/Node.js/node_modules',
   '/Users/subhash/node_modules',
   '/Users/node_modules',
   '/node_modules' ] }
```



node_modules everywhere

- Node.js looks for 'node_modules/bar.js' in the current folder
- followed by every parent folder
- until it reaches the root of the file system tree for the current file or
- until a bar.js is found

Advantages of node_modules

- Simplify Long File Relative Paths
- Increasing Reusability
- Decreasing Side Effects
- Overcoming Module Incompatibility

JSON

- a standard format to transfer data
- a subset of JavaScript object literals
- Node.JS NPM uses JSON files for configuring modules

NPM

- First, reusable modules using node_modules.
- Next, get modules shared publically by the Node.js community
  - Answer: Node Package Manger (npm)
  - a command line tool that integrates with the online NPM registry (www.npmjs.org/)

npm version

```
my:~ subhash$ npm --version
2.7.4
my:~ subhash$
```

package.json

- part of NPM is a JSON file - package.json.
  - to share a module with world
- To create a package.json in current folder
  - $ npm init
  - This will ask you a few questions
  - Answer them or simply press enter
  - To create a boilerplate package.json

Installing an NPM package

- A sample app needs underscore module
- Let us install from global repository npmjs.org
  - npm install underscore
- This will put it into node_modules/ underscore in the current folder.
- To load this module,
  - require('underscore’);

Saving Dependencies

- npm install has an optional command line flag available
  - --save
  - tells NPM to write the information about what you installed into package.json

Refresh the node_modules Folder

- To refresh the node_modules folder from your package.json
  - $ npm install
- This simply downloads a fresh copy of the dependencies specified in your package.json.

Listing All Dependencies

- To see which packages you have installed, you can run
  - $ npm ls

Removing a Dependency

- Remove a dependency using
  - $ npm rm
- For example,
  - npm rm underscore --save
  - deletes the underscore folder from node_modules locally and
  - modifies the dependencies section of your package.json

Semantic Versioning

- Node.js follows a three-digit versioning scheme X.Y.Z where all X, Y, and Z are non-negative integers.
- X is the major version, Y is the minor, and Z is the patch version.

Semantic Versioning

- Patch versions must be incremented if backward compatible fixes are introduced.
- Minor versions must be incremented if backward compatible new features are introduced.
- Major versions must be incremented if backward incompatible fixes/features/ changes are introduced.

Semantic Versioning in NPM / package.json

- NPM and package.json support semantic versioning.
- For example, to install the exact version:
  - $ npm install underscore@1.0.3
- if okay with all patch versions of 1.0:
  - $ npm install underscore@"~1.0.0"
- if okay with any minor version changes:
  - $ npm install underscore@"^1.0.0”

Global Node.js Packages

- It is really simple to make command line utilities in Node.js.
- The objective of global Node.js packages is to provide command line utilities
  - npm install -g browserify
  - This put browserify on the command line,

Other features

- update global packages
  - $ npm update -g package-name
- list global packages
  - $ npm ls –g
- uninstall packages
  - npm rm -g package-name
  - For example, to uninstall Browserify,
    - $ npm rm -g browserify

Package.json and require

- package.json for NPM managing dependencies and putting them in node_modules.
- It looks for a JavaScript file/folder in node_modules that matches what we asked require to load,
  - for example, foo in require('foo')
  - if it resolves to a folder, Node.js tries to load index.js from that folder

- Project Structure for Demo mainproperty

```
|-- app.js
|-- node_modules
        |-- foo
            |-- package.json
            |-- lib
                |-- main.js
```

- main.js is a simple file that logs to the console to indicate it was loaded

Modules Summary

- Assume you require('something'). Then:
  - If something is a core module, return it.
  - If something is a relative path return that file OR folder.
  - If not, look for node_modules/filename or node_modules/foldername each level up until you find a file OR folder that matches something.

When matching a file OR folder

- If it matched a file name, return it.
- If it matched a folder name and it has package.json with main, return that file.
- If it matched a folder name and it has an index file, return it.

Some Commonly Used Packages

- Underscore
- optimist
- Date / Time – moment
- colors

That's all

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep