COMP 2406

Here we look at modularizing javascript with modules including using imported npm modules.

Javascript Modules in Node.js

Node.js provides three kinds of modules

- **core modules –ship with node.js**
- **file based modules –loaded from local file system**
- **imported from npm registry**

Server-side

- **Here we are talking about modularizing server-side javascript for execution in node.js environment.**
- **Here we don't talk about modularizing client-side javascript shipped to the browser.**

- **References: Beginning Node.js Copyright © 2014 by Basarat Ali Syed**

- **Syed, Basarat Ali (2014-11-28). Beginning Node.js . Apress. Chapter 4**

Core Modules and Imported Node Modules

- `require('bar');`
- look for core modules with the same name, for example, `bar`
- If no core module matching this name is found, we look for an imported node_module called `'bar'`.

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 65). Apress Chapter 4

File based modules (our own modules)

- `require('./bar);`
- `require('../bar/bar');`
- `require('/full/path/to/a/node/module/file');`

- File-based module is loaded from the specified path

Scanning order for Imported Node Modules

```
/home/ryo/project/node_modules/bar.js
/home/ryo/node_modules/bar.js
/home/node_modules/bar.js
/node_modules/bar.js
```

- **In other words, 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.**

[Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 66). Apress. Kindle Edition.]

Simulating Imported Node Modules

```
// home/foo.js
var bar = require('bar');
bar(); // hello node_modules!

// home/node_modules/bar.js
module.exports = function () {
    console.log(' hello node_modules!');
}
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 66). Apress.

As local file Modules

```
// home/foo.js
var bar = require('./bar');
bar(); // hello node_modules!

// home/bar.js
module.exports = function () {
    console.log(' hello node_modules!');
}
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 66). Apress.

index.js

implicit file: index.js

It is common to have a several files working toward a singular goal.

Node.js has explicit support for this mechanism.

If a path to the module resolves to a folder (instead of a file), Node.js will look for an `index.js` file in that folder and return that as the module file.

[Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 66). Apress. Kindle Edition. chap 4]

implicit loading of `index.js`

```
//bar/bar1.js
module.exports = function () {
    console.log('bar1 was called'); }

//bar/bar2.js
module.exports = function () {
     console.log('bar2 was called'); }

//bar/index.js
exports.bar1 = require('./bar1');
exports.bar2 = require('./bar2');

//foo.js
var bar = require('./bar');
bar.bar1();
bar.bar2();
```

[Syed, Basarat Ali (2014-11-28). Beginning Node.js (pp. 66-67). Apress. Kindle Edition. Chapter 4]

implicit loading of `index.js` from `node_modules`

```
//node_modules/bar/bar1.js
module.exports = function () {
  console.log(' bar1 was called'); }

//node_modules/bar/bar2.js
module.exports = function () {
  console.log(' bar2 was called'); }

//node_modules/bar/index.js
exports.bar1 = require('./bar1');
exports.bar2 = require('./bar2');

//foo.js
var bar = require('bar'); //node_modules module bar
bar.bar1();
bar.bar2();
```

[Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 67). Apress. Kindle Edition. Chap. 4]

using implicit loading of `index.js` from `node_modules`

```
foo/foo.js //provides utilities you

bar/bar.js
require('../foo/foo.js')

bar/sub1/sub2/file.js
require('../../../foo/foo.js')

//better idea:
/node_modules/foo/index.js

//in each file:
require('foo')
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 68). Apress. Kindle Edition. Chap 4

Module Incompatibility

In Node.js, each module can have its own `node_modules` folder and different versions of `moduleZ` can coexist.

Modules do not need to be global in Node.js

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap 4

Node will Cache Modules

```
projectroot
|--
node_modules/foo/index.js
   |-- moduleA/ a.js
   |-- moduleB/ b.js
```

If `a.js` and `b.js` both `require('foo')` they will get the same cached module export result.

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap. 4

Own versions of Modules

```
projectroot
  |--
node_modules/foo/index.js
  |-- moduleA/a.js
  |-- moduleB
      |--
      node_modules/foo/index.js
      |-- b.js
```

If `a.js` and `b.js` both `require('foo')` this time they will get the different versions of module foo.

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap. 4

Implicit Loading of JSON objects

In Node.js, if a foo.js is not found as a result of a require('foo'), Node.js will look for a foo.json

foo.json will be treated as a JSON string and converted to a javascript object and returned as the export of the module.

Popular mechanism for configuration data.

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap 4

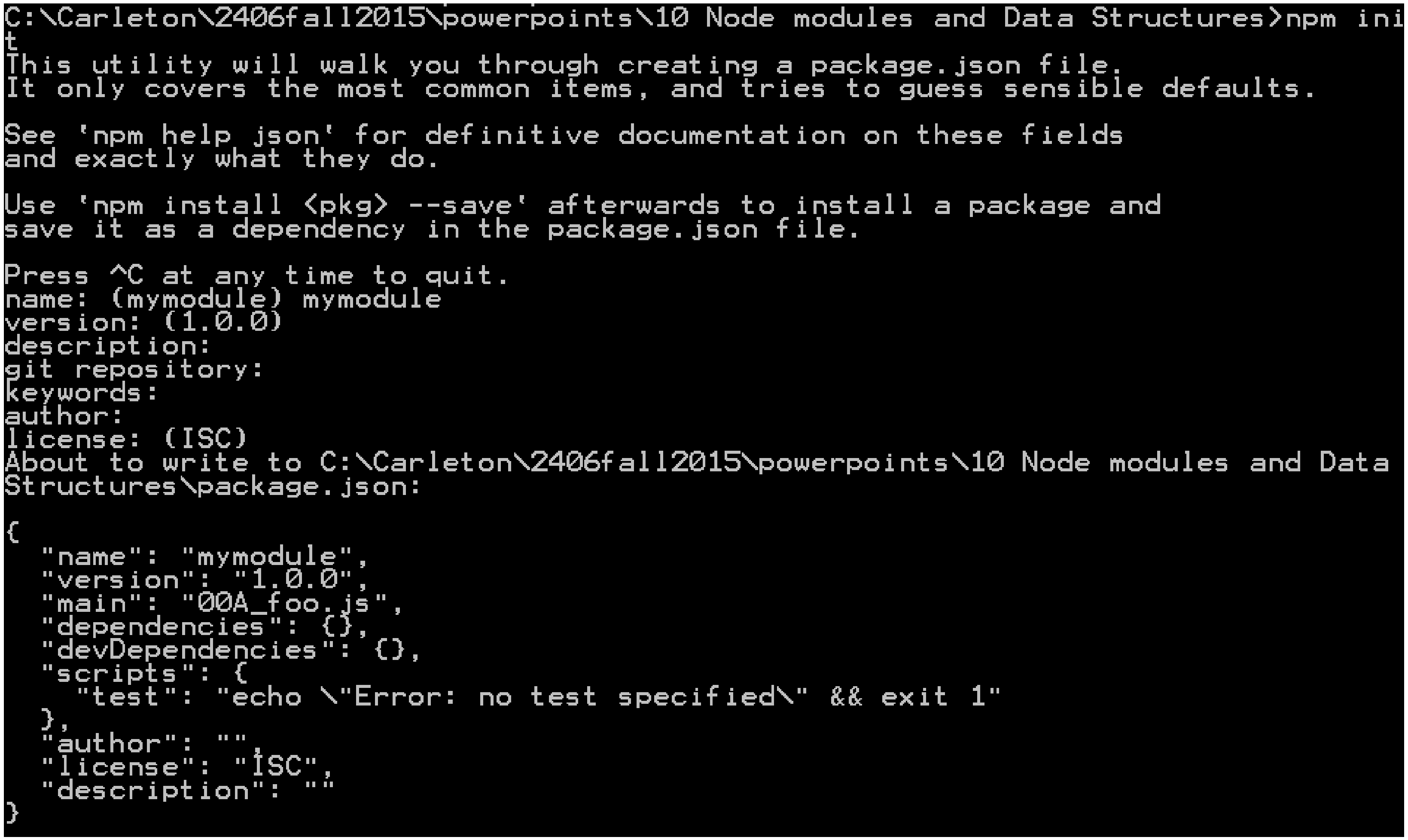
Implicit Loading of JSON objects

```
//config.json
{"foo": "this is the value for foo"}

//app.js
var config = require('./config');
console.log(config.foo); //this is the value for foo
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap 4

Creating a package.json depedencies file

`npm init`



```
C:\Carleton\2406fall2015\powerpoints\10 Node modules and Data Structures>npm ini
t
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg> --save' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (mymodule) mymodule
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Carleton\2406fall2015\powerpoints\10 Node modules and Data
Structures\package.json:

{
  "name": "mymodule",
  "version": "1.0.0",
  "main": "00A_foo.js",
  "dependencies": {},
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

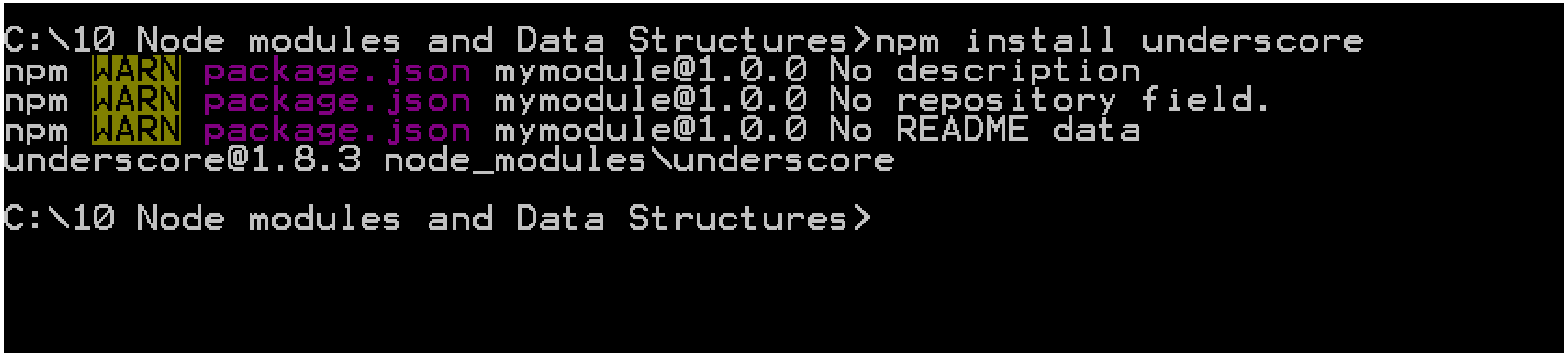
Creating a package.json depedencies file

npm init

```
{
  "name": "mymodule",
  "version": "1.0.0",
  "description": "",
  "main": "00A_foo.js",
  "dependencies": {},
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Installing an npm package

`npm install underscore`



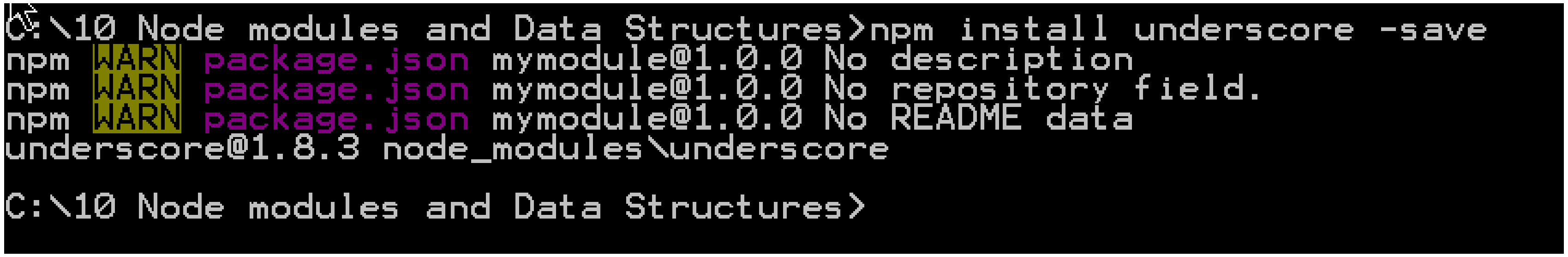
```
C:\10 Node modules and Data Structures>npm install underscore
npm WARN package.json mymodule@1.0.0 No description
npm WARN package.json mymodule@1.0.0 No repository field.
npm WARN package.json mymodule@1.0.0 No README data
underscore@1.8.3 node_modules\underscore

C:\10 Node modules and Data Structures>
```

```
{
  "name": "mymodule",
  "version": "1.0.0",
  "main": "00A_foo.js",
  "dependencies": {},
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

No effect on `package.json`

Installing an npm package (with -save or --save option)

```
npm install underscore -save   //or --save
```



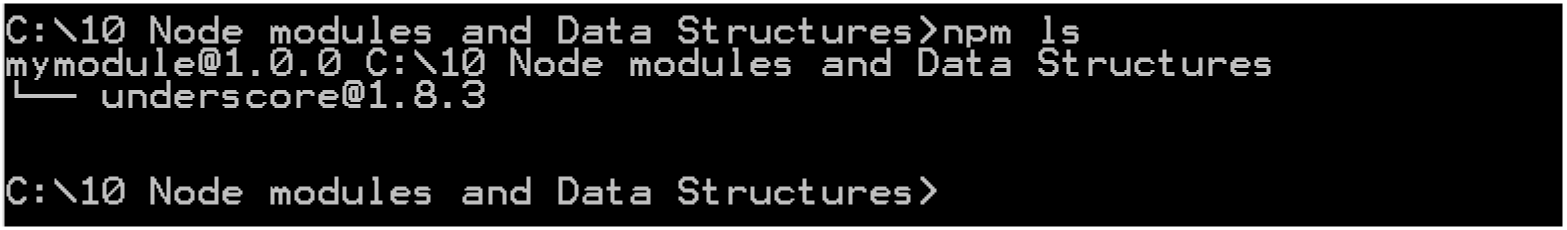
```
C:\10 Node modules and Data Structures>npm install underscore -save
npm WARN package.json mymodule@1.0.0 No description
npm WARN package.json mymodule@1.0.0 No repository field.
npm WARN package.json mymodule@1.0.0 No README data
underscore@1.8.3 node_modules\underscore

C:\10 Node modules and Data Structures>
```

```
{
  "name": "mymodule",
  "version": "1.0.0",
  "main": "00A_foo.js",
  "dependencies": {
    "underscore": "^1.8.3"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Updates `package.json`

Listing installed dependencies

`npm ls`



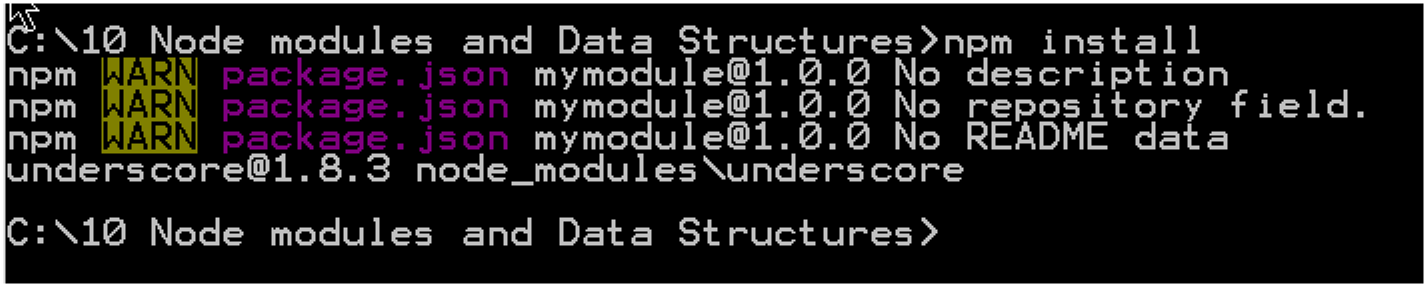
```
C:\10 Node modules and Data Structures>npm ls
mymodule@1.0.0 C:\10 Node modules and Data Structures
└── underscore@1.8.3

C:\10 Node modules and Data Structures>
```

Removing depedencies

```
npm rm underscore
//or
npm rm underscore -save //to update package.json
```

Installing all depencies in your package.json

```
npm install //installs from package.json
```



```
C:\10 Node modules and Data Structures>npm install
npm WARN package.json mymodule@1.0.0 No description
npm WARN package.json mymodule@1.0.0 No repository field.
npm WARN package.json mymodule@1.0.0 No README data
underscore@1.8.3 node_modules\underscore

C:\10 Node modules and Data Structures>
```

```
{
  "name": "mymodule",
  "version": "1.0.0",
  "main": "00A_foo.js",
  "dependencies": {
    "underscore": "^1.8.3"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Will install all dependencies listed in `package.json`

npm Semantic Versioning

Npm uses the following Semantic (meaningful) version numbering scheme

```
1.8.3
```

Major version number
Must be incremented when a non-backwards compatible feature is added or a non-backwards compatible bug fix is made

Minor version number
Must be incremented when a backwards compatible feature is added

Patch version must be incremented when a backwards compatible bug fix is made

npm Semantic Versioning

Npm uses the following Semantic (meaningful) version numbering scheme

```
~1.8.3
```

~ means latest patch compatible version

e.g. 1.8.5
But not
1.9.0 or 2.0.1

npm Semantic Versioning

Npm uses the following Semantic (meaningful) version numbering scheme

```
^1.8.3
```

^ means latest minor version compatible version

e.g. 1.9.5
but not 2.0.1

npm Semantic Versioning –other options

Npm uses the following Semantic (meaningful) version numbering scheme

```
1.8.*
>=1.8.9
<2.0.3
* //latest version
```

Installing specific versions

```
> npm install underscore@1.0.3

> npm install underscore@"~1.0.0"

> npm install underscore@"^1.0.0"

> npm install underscore  //latest version
```

Installing specific versions –with `package.json`

```
//in package.json
"dependencies": {
     "underscore": "^1.6.0"
}


npm install

//Or to update package.json
npm install underscore@"^1.0.0" -save

Or to update package.json
$ npm update -save
```

npm includes installing global modules with –g option

```
npm install -g browserify
```

```
Intended for installing modules that provide
command line tools

Not intended for modules you require() with your code
```

package.json has a special main property for loading files

`package.json` does provide one special `main` property for loading file system modules. (There can only be one `main` however.)

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

//package.json

```
{ "main" : "./lib/main.js" }
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 79). Apress. Kindle Edition.

package.json has a special main property for loading files

If and code were to

`require('foo')`,

**Node.js would look at `package.json`, see the `main` property, and run
`'./lib/main.js'`.**

`Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 79). Apress. Kindle Edition.`

package.json has a special main property for loading files

Example from `rimraf` npm module

```
//package.json from the rimraf npm
//Module Showing the Main Property

{ "name": "rimraf",
  "version": "2.2.7",
  "main": "rimraf.js",
  ... truncated...

}
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 79). Apress. Kindle Edition.

Some popular NPM modules

`underscore` –probably the most popular npm module. Provides function programming utilities.

```
//without  underscore
var foo = [1,10,50,200,900,90,40];
var rawResults = []
for (i = 0; i < foo.length; i + +) {
   if (foo[ i] > 100) { rawResults.push( foo[ i]); }
 }
 console.log( rawResults);

//with underscore
var foo = [1, 10,50,200,900,90,40];
var _ = require('underscore');
var results = _.filter(foo, function(item){
    return item > 100 });
console.log( results);
```

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap 4

Some popular NPM modules

`optimist` –for handling command line arguments

`moment` –provides man time and date utilities beyond javascript's built in `Date`

`colors` –provides custom string colors for console output

Resouces:
NPM online registry:
`http://npmjs.org/`

Semantic versioning the official guide:
`http://semver.org/`

Semantic versioning parser in NPM:
`https://github.com/isaacs/node-semver`

Syed, Basarat Ali (2014-11-28). Beginning Node.js (p. 69). Apress. Kindle Edition. Chap 4

Resources

Resouces:
NPM online registry:
`http://npmjs.org/`

Semantic versioning the official guide:
`http://semver.org/`

Semantic versioning parser in NPM:
`https://github.com/isaacs/node-semver`