npm node.js

dgreed iTransform

→ npm is the package manager for JavaScript

→ files containing reusable code are called modules.

→ Package is nothing but a directory with one & more modules & along a special file called Package.JSON, this contains metadata about the package.

→ npm is a way to share your code with other developers & reuse that code from other developers & easily manage the different versions of your code

Package.json (npm init command to create P.json)
- it lets you manage the dependencies of your project
- it lets you add scripts that help with initial build of your project

we can use

--yes to create package.json file with default's

(npm init --yes) same file as package.json

To set default author & license name:

author
→ npm config set init-author-name "mahesh"

license
→ npm set init-license "MIT"

then delete previous Package.json file & create again by using --yes command

To know default's of

author
→ npm config get init-author-name

→ npm get init-license

To delete

→ npm config delete init-license

init-author-name

then delete Package.json & create by --yes

command =

install & uninstall local packages

To add dependencies in Package.json

& to install local Packages we use

① → npm install moment --save

② → npm install moment Package name } do save

③ if you want install packages for development

→ npm install lodash --save-dev

④ To uninstall

local

→ npm uninstall moment --save

→ npm uninst lodash --save-dev

⑤ To install of (global packages)

npm install moment -g

⑥ To uninstall

npm uninstall moment -g

or

npm rm/remove moment -g

Listing Packages

npm list

gives all packages in a tree structure

npm list --depth 1 (gives dependencies like above)

--depth 0 (list's all packages & no dependency packages)

To list global packages

npm list --global true --depth 0

Listing Packages

npm list
gives all packages in a tree structure

npm list --depth 1 (gives dependencies like express)

--depth 0 (list's all packages & no dependency packages)

To list global packages

npm list --global true --depth 0

npm list

NPM versioning: ("lodash": "^3.3.0")
major minor patch version
package version

To install specific package version

npm install lodash@3.3.0 --save

To install latest patch version

npm install lodash@4.14 --save

To install latest version with major

npm install lodash@4 --save

"4.14.1"

"^4.14.1" (Package.json)

↳ caret symbol

(1) Caret ^

this specifies to stick to the given major version & but when you run npm retrieve the latest new mind & Patch

---

(2) "~4.14.1" after giving ~ run npm install

↓

tilt

leave the major & mind version as it is but retrieve the latest version & patch number, when I run npm install command

(3) "*"

when specify that it going to retrieve the latest version & lodash irrespective of major mind of patch versions

Updating Packages

To update particular package

use

→ npm update lodash --save

(gets updated to latest version)

(dev packages)

→ npm update --dev --save-dev

To update package globally use

npm update -g

To install npm latest version

npm install npm@latest -g

npm Prune:

It used to remove unwanted additional packages.

~~of unwanted~~ from our project

npm Prune

Shortcuts

→ npm init -y (initialized y --yes)

→ npm i? lodash (to install local package)
i = install

→ npm i? lodash -S (to install package in package.json)

→ npm i moment -D (to save package as dev-dependency)

→ -g = --global

→ npm -v = npm --version

npm Scripting:

create app.js

console.log("NPM Shortcuts test");

node app.js

o/p: NPM Shortcuts test

Introduction to Node.js

Node.js is a platform which allows us to run javascript on a computer / server. It gives us the ability to read, delete & update files and easily communicate with a database

What we will learn

1. The inner workings of Node.js
   - V8 engine (Converts Javascript in machine code (0,1))
   - modules
   - Event emitter
   - The file system
2. Creating a web server
   - Routing
   - Express
   - Templating
3. Make a Node.js application (todo app)

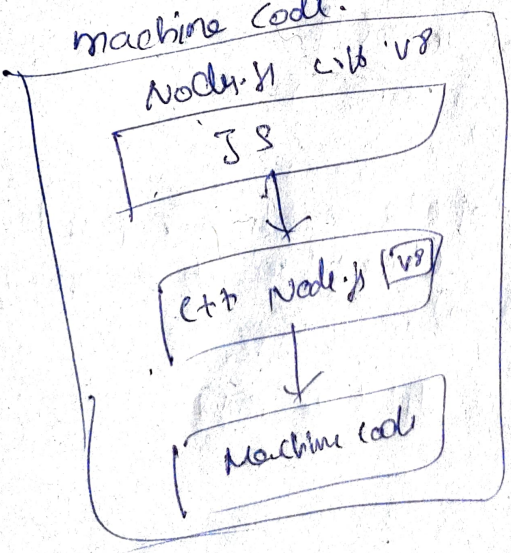
Java V8 Engine

Java script Engines

- completely do not understand Javascript
- A JS engin takes JS & converts it into something it does understand - machine code.

Actually

- Node.js is written in c++
- At the heart of Node.js is the V8 engine
- The V8 engine converts our JS into machine code.



Type here to search

Global object: (Same as window object)

It gives us access to some methods we can use straight out of the box in node js.

window object

```
setTimeout(function(){
    console.log('3 seconds have passed');
}, 3000);
```

(3 points after 3 sec's)

O/P: 3 seconds have passed

Set

```
var time = 0;
```

window object

```
setInterval(function(){
    time += 2;
    console.log('3 seconds have passed');
    (time + ' seconds have passed');
}, 2000);
```

O/P: 2 seconds have passed

4

6

8

to stop (Ctrl + C)

Node tells us where we are in a particular directory & what file we are in by:

→ console.log(__dirname);

gives Path in o/p.

⟹ (To know file)

→ console.log(__filename);

o/p: give app.js file

Function Expressions

```
var sayBye = function(){
    console.log('bye');
};

sayBye();
```

anonymous function

This is Function expression

variable name

call we invoke with () parenthesis;

We can keep one function in another function

```
function callFunction(fun){
    fun();
}
```

one fun

anonymous

```
var sayBye = function () {
    console.log('bye');
};
callFunction(sayBye);
```

o/p: bye

what's going to happen is sayBye variable is going to be passed through function then we call sayBye() function which has anonymous function.

after running code o/p is bye

(4) module & require()

require is a global object that is used to use another

1. `require('./count')`

'./' defines current direct
(: count is a file n

↓

this string is a path to the module th
we require in this file.

"require() function used to use one module in another module."

`module.exports = counter;` used to make available only one part / function

this indicates defines what part of the module we want to make available to all the files which require this module

count.js

```
var counter = function(arr) {
    return 'There are ' + arr.length + ' element in the array';
};

module.exports = counter;
```

app.js

```
var counter = require('./count')

console.log(counter(['shaun', 'crystal', 'ryu']));
```

O/P: There are 3 elements in the array.

module pattern

it can use to make modules from other modules

Stuff.js

```
module.exports.counter = function(arr){
    return 'There are' + arr.length + 'elem in this array';
};

module.exports.adder = function(a,b){
    return `The sum of the 2 numbers is ${a+b}`;
};

module.exports.pi = 3.142;
```

app.js:

```
var stuff = require('./Stuff')
console.log(stuff.counter(['shn','ryn','ryn']));
// console.log(stuff.adder(5,6));
// console.log(stuff.adder(stuff.pi,5));
```

O/P: There are 3 elem in the array
The sum of the 2 numbers is
is 8.142