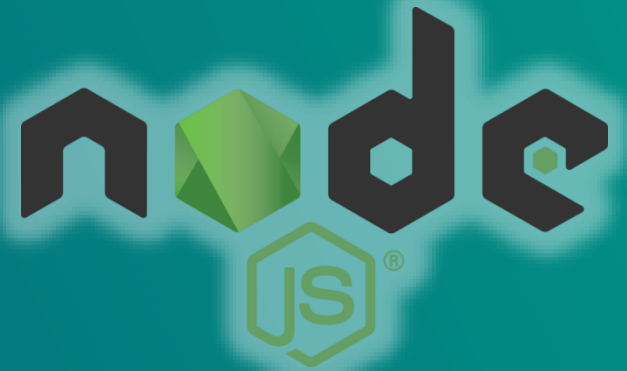


Back-End Development

Exploring

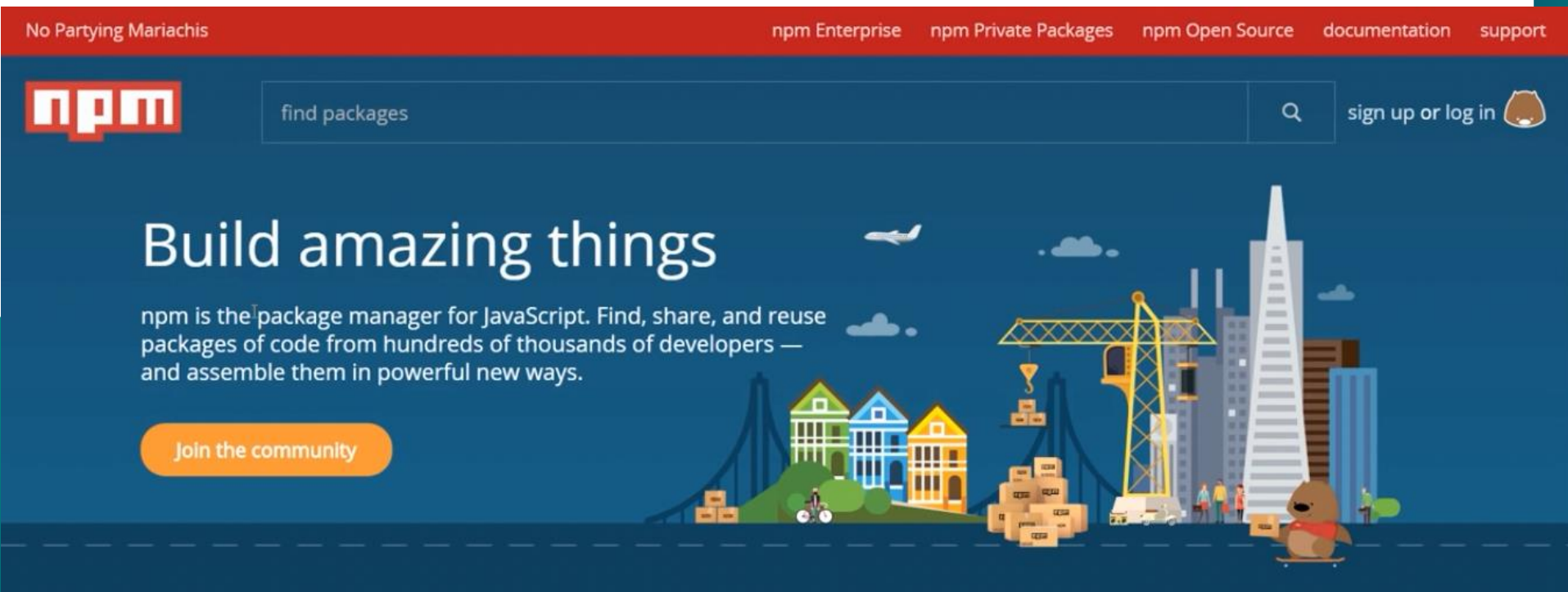
#4 Node Package Manager





Node Package Manager

NPM is the package manager for JavaScript. Basically it's just a bunch of tools which will help the developers to use 3rd party packages/modules, for building their application easily.



npmjs.com

Package

Package is a file/directory that is described by a *package.json*, including:

1. A folder containing a program described by a *package.json* file,
2. A gzipped tarball containing (1.),
3. A url that resolves to (2.),
4. A *<name>@<version>* that is published on the registry with (3.),
5. A *<name>@<tag>* that points to (4.),
6. A *<name>* that has latest tag satisfying (5.),
7. A *git* url that, when cloned, results in (1.).

Modules

Module is anything that can be loaded with *require()* in an NodeJS program, including:

- A folder with a *package.json* file containing a main field,
- A folder with an *index.js* file in it, or
- Just a JavaScript file.

Note: Most NPM packages are modules!

NPM Packages/Modules

NPM has hundreds of packages/modules which you can use easily.

- angular
- bluebird
- body-parser
- chalk
- commander
- config
- express
- ejs
- jest
- jshint
- lodash
- lru-cache
- meddleware
- nodemon
- node-static
- nodeUtils
- path
- pluralize
- request
- serve-favicon
- table
- typescript
- ***etc...***



Getting Started

On terminal (at your dir project), type:

- **npm -v**

check the version of installed npm.

- **npm init**

*set package, make a **package.json** file.*

package.json is a JSON file that contains lot of informations about our app & keeps track which package/modules we used.

** similar to our medical records, right? :)*

- **npm install namaPackage --save**

install an NPM package/module (npm i).

- **npm install**

*install packages that written in **package.json***

```
1  {
2    "name": "Lin",
3    "version": "1.0.0",
4    "description": "Uji coba",
5    "main": "dua.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [
10     "Lintang"
11   ],
12   "author": "Lintang Wisesa",
13   "license": "ISC"
14 }
15
```

package.json

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

1: cmd ▼

```
D:\zzz>npm -v
3.10.10
```

```
D:\zzz>npm init
```

```
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.
```



Chalk

- Install Chalk

npm install chalk

- Install chalk & update package.json

npm install chalk --save

More info: github.com/chalk/chalk



Chalk

#1 Base Color

```
const chalk = require('chalk');  
  
console.log(chalk.red('Ini Merah'));  
console.log(chalk.yellow('Ini Kuning'));  
console.log(chalk.green('Ini Hijau'));  
console.log(chalk.blue('Ini Biru'));
```

```
D:\zzz\lin_backend>node 0  
Ini Merah  
Ini Kuning  
Ini Hijau  
Ini Biru
```



Chalk

#2 Keyword

```
const chalk = require('chalk');  
  
console.log(chalk.keyword('orange')('Hai 1'));  
console.log(chalk.keyword('purple')('Hai 2'));  
console.log(chalk.keyword('cyan')('Hai 3'));  
console.log(chalk.keyword('gold')('Hai 4'));  
console.log(chalk.keyword('magenta')('Hai 5'));
```

```
D:\zzz\lin_backend>node 0  
Hai 1  
Hai 2  
Hai 3  
Hai 4  
Hai 5
```



Chalk

#3 RGB & HEX

```
const chalk = require('chalk');
console.log(chalk.rgb(255,0,0)('RGB 1'));
console.log(chalk.rgb(0,255,0)('RGB 2'));
console.log(chalk.rgb(255,0,255)('RGB 3'));
console.log(chalk.hex('#0000FF')('HEX 1'));
console.log(chalk.hex('#FF0000')('HEX 2'));
console.log(chalk.hex('#808000')('HEX 3'));
```

```
D:\zzz\lin_backend>node 0
RGB 1
RGB 2
RGB 3
HEX 1
HEX 2
HEX 3
```



Chalk

#4 *bgColor*

```
const chalk = require('chalk');  
  
console.log(chalk.bgRed('Halo'));  
console.log(chalk.bgYellow('Halo'));  
console.log(chalk.bgGreen('Halo'));  
console.log(chalk.bgBlue('Halo'));
```

```
D:\zzz\lin_backend>node 0
```

```
Halo  
Halo  
Halo  
Halo
```

```
bold dim italic underline inverse strikethrough black
red green yellow blue magenta cyan white gray bgBlack
bgRed bgGreen bgYellow bgBlue bgMagenta bgCyan bgWhite
```

Colors

■ Install Colors

npm install colors

■ Install colors & update package.json

npm install colors --save

More info: npmjs.com/package/colors

Colors

#1 Base Color

```
const colors = require('colors');  
console.log(colors.red('Hello'));  
console.log(colors.yellow('Hello'));  
console.log(colors.green('Hello'));  
console.log(colors.blue('Hello'));  
console.log(colors.magenta('Hello'));  
//black, cyan, white, gray, grey
```

```
D:\zzz\lin_backend>node 0  
Hello  
Hello  
Hello  
Hello  
Hello
```

Colors

#2 *bgColor*

```
const colors = require('colors');
console.log(colors.bgRed('Hello'));
console.log(colors.bgYellow('Hello'));
console.log(colors.bgGreen('Hello'));
console.log(colors.bgBlue('Hello'));
console.log(colors.bgMagenta('Hello'));
//bgBlack, bgCyan, bgWhite
```

```
D:\zzz\lin_backend>node 0
```

```
Hello
Hello
Hello
Hello
Hello
```

Colors

#3 Styles

```
const colors = require('colors');  
console.log(colors.bold('Hello'));  
console.log(colors.rainbow('Hello'));  
console.log(colors.trap('Hello'));  
console.log(colors.inverse('Hello'));  
console.log(colors.america('Hello'));  
console.log(colors.zebra('Hello'));
```

```
D:\zzz\lin_backend>node 0  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello
```


Colors

#4 New Way

```
const colors = require('colors');  
console.log('Hello'.red);  
console.log('Hello'.rainbow);  
console.log('Hello'.trap);  
console.log('Hello'.bgYellow.black);  
console.log('Hello'.america);
```

```
D:\zzz\lin_backend>node 0
```

```
Hello
```

```
Hello
```

```
HĖŁŁŎ
```

```
Hello
```

```
Hello
```

Slug

It slugifies every string, even when it contains unicode!
For short: make strings url-safe.

- Install slug

npm install slug

- Install it & update package.json

npm install slug --save

Slug

#1 Slug

```
var slug = require('slug');
```

```
var satu = slug('NodeJS ♥ is ☢');  
var dua = slug('I <3 NodeJS');
```

```
console.log(satu)  
console.log(dua)
```

Slug

#2 Edit

```
var slug = require('slug');

slug.charmap[' ♥ '] = 'am crazy of'

var tiga = slug('i ♥ nodejs', '_');
var empat = slug('I ♥ NODEJS',
{lowercase: false})

console.log(tiga)
console.log(empat)
```



Moment

Moment is a NPM module to parse, validate, manipulate, and display dates and times in JavaScript.
(see momentjs.com)

- Install moment

npm install moment

- Install moment & update package.json

npm install moment --save



Moment

#1 Now

```
const moment = require('moment');

var now1 = moment();
var now2 = moment().format();
var now3 = moment().format("ddd, hA");
var now4 = moment().format(
  "dddd, MMMM Do YYYY, h:mm:ss a");

console.log(now1);
console.log(now2);
console.log(now3);
console.log(now4);
```



Moment

#2 *fromNow* & *toNow*

```
const moment = require('moment');  
  
var w = moment([2007]).fromNow();  
var x = moment([2007, 0, 29]).fromNow(true);  
var y = moment([2007]).toNow();  
var z = moment([2007, 0, 29]).toNow(true);  
  
console.log(w);  
console.log(x);  
console.log(y);  
console.log(z);
```



Moment

#3 a-to-z

```
const moment = require('moment');  
  
var a = moment([2007, 2, 27]);  
var b = moment([2007, 0, 29]);  
  
console.log(a.to(b));  
console.log(a.to([2007, 1, 27]));  
console.log(a.to(new Date(2007, 3, 6)));  
console.log(a.to("2017-11-29"));
```


Lo Lodash

Lodash makes JavaScript easier by taking the hassle out of working with arrays, numbers, objects, strings, etc. (see <https://lodash.com>)

Lodash's modular methods are great for (1) iterating arrays, objects, or strings, (2) testing values, and (3) creating composite functions.

- Install lodash

npm install lodash

- Install lodash & update package.json

npm install lodash --save or

npm i lodash --save



Lodash

#1 String

```
const _ = require('lodash');  
  
console.log(_.isString(135));  
console.log(_.isString('Startup'));  
  
console.log(_.capitalize('GOOGLE'));  
console.log(_.upperFirst('facebook'));  
console.log(_.upperCase('alibaba'));  
console.log(_.lowerFirst('TWITTER'));  
console.log(_.lowerCase('YAHOO'));
```



Lodash

#2 Number

```
const _ = require('lodash');  
  
console.log(_.isNumber(24));  
console.log(_.isNumber('Andi'));  
  
console.log(_.add(100, 2));  
console.log(_.subtract(48, 5));  
console.log(_.multiply(2, 9));  
console.log(_.divide(75, 3));  
  
console.log(_.ceil(99.3));  
console.log(_.floor(99.3));
```

```
const _ = require('lodash');  
var x = [1,3,2,4,3,5,4,6];  
var y = ['Andi', 1, 'Budi', 2];
```

```
console.log(_.isArray(x));  
console.log(_.uniq(x));  
console.log(_.max(x));  
console.log(_.min(y));  
console.log(_.sum(x));  
console.log(_.reverse(y));
```

UNDERSCORE.JS

Underscore

- Install Underscore

npm install underscore

- Install it & update package.json

npm install underscore --save

More info: npmjs.com/

```
const _ = require('underscore');  
var arr = [1, 2, 3]  
  
var x = _.map(arr, function(num){  
    return num * 3;  
});  
  
console.log(x)
```

```
const _ = require('underscore');  
var x = [1,2,3]  
var y = [4,5,6]  
var z = [7,8,9]  
var arr = [x, y, z]  
  
var ok = _.map(arr, _.first);  
  
console.log(ok)
```

```
const _ = require('underscore');  
var obj = {a: 1, b: 2, c: 3}  
  
var y = _.map(obj, function(num, key){  
    return num * 3;  
});  
  
console.log(y)
```



```
const _ = require('underscore');  
var arr = [1,2,3,4,5,6]
```

```
var nilaiMax = _.max(arr);  
var nilaiMin = _.min(arr);
```

```
console.log(nilaiMax)  
console.log(nilaiMin)
```

```
const _ = require('underscore');  
var pns = [  
  {nama: 'Andi', usia: 34},  
  {nama: 'Budi', usia: 40},  
  {nama: 'Caca', usia: 46}  
]  
  
var tertua = _.max(pns, (x) => x.usia);  
var termuda = _.min(pns, (x) => x.usia);  
  
console.log(tertua)  
console.log(termuda)
```

```
const _ = require('underscore');
```

```
var arr = [1,2,3,4,5,6]
```

```
function genap(x) {  
  return x % 2 == 0  
}
```

```
var findGenap = _.find(arr, genap);  
var filterGenap = _.filter(arr, genap);
```

```
console.log(findGenap)  
console.log(filterGenap)
```



Yargs

Yargs helps you build interactive command line tools, by parsing arguments and generating an elegant user interface (See <http://yargs.js.org>).

It gives you:

- (1) commands and (grouped) options,
- (2) a dynamically generated help menu based on your arguments &
- (3) bash-completion shortcuts for commands and options.

■ Installation

`npm install yargs --save`



satu.js

```
var argv = require('yargs').argv;  
console.log(argv.x , argv.y);
```

```
// Run script!  
// node satu  
// node satu --x=4 --y=4  
// node satu -x 'Halo' -y 'kawan'
```



satu.js

```
const argv = require('yargs').argv
console.log(argv.usia)
if (argv.usia >= 25) {
  console.log('Ingat umur, kak!')
} else {
  console.log('Enjoy your Life!')
}
```

```
// Run script!
// node satu
// node satu --usia=24
// node satu -usia 31
```



satu.js

```
const argv = require('yargs').argv  
var perintah = process.argv;
```

```
console.log(perintah);  
console.log(perintah[3]);  
console.log(argv.judul);
```

// Run script!

// node satu

// node satu --judul=Iron Man

// node satu daftar --judul=Iron Man



Nodemon

Nodemon is a tool/utility that will monitor for any changes in your source and automatically restart your server. Just use **nodemon** instead of **node** to run your code, and now your process will automatically restart when your code changes.

See <https://nodemon.io>.

■ Installation

```
npm install -g nodemon
```

■ Open & run source with Nodemon

```
nodemon namaFile.js
```

■ Restart

```
rs
```


#1 Buatlah sebuah program yang saat dijalankan akan menampilkan data CPU user di console, dengan warna latar obj 1 kuning & obj 2 merah!

Gunakan module OS & Chalk!

```
D:\zzz coding\tes2>node soal1
```

```
{ "model": "Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz", "speed": 2594, "times": { "user": 10870375, "nice": 0, "sys": 6904437, "idle": 201087484, "irq": 2472781 } }
```

```
{ "model": "Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz", "speed": 2594, "times": { "user": 6988437, "nice": 0, "sys": 1802890, "idle": 210070859, "irq": 71796 } }
```

#1 Solved!

```
const os = require('os');
const chalk = require('chalk')

var dataCPU0 = JSON.stringify(os.cpus()[0]);
var dataCPU1 = JSON.stringify(os.cpus()[1]);

console.log(
chalk.bgYellow.rgb(0,0,0)(dataCPU0));
console.log(
chalk.bgRed.rgb(255,255,255)(dataCPU1));
```

```
D:\zzz coding\tes2>node soal1
```

```
{ "model": "Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz", "speed": 2594,
  "times": { "user": 10870375, "nice": 0, "sys": 6904437, "idle": 201087484,
  "irq": 2472781 } }
```

```
{ "model": "Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz", "speed": 2594,
  "times": { "user": 6988437, "nice": 0, "sys": 1802890, "idle": 210070859,
  "irq": 71796 } }
```

#2 Buatlah sebuah program yang saat dijalankan akan menampilkan data CPU user di console, dengan style obj 1 rainbow & obj 2 trap!

Gunakan module OS & Colors!

```
D:\zzz coding\tes2>node soal2
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":10879734,"nice":0,"sys":6917937,"idle":201466609,
"irq":2479843}}
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":6995859,"nice":0,"sys":1804484,"idle":210463812,
"irq":71890}}
```

#2 Solved!

```
const os = require('os');
const colors = require('colors')

var dataCPU0 = JSON.stringify(os.cpus()[0]);
var dataCPU1 = JSON.stringify(os.cpus()[1]);

console.log(colors.rainbow(dataCPU0));
console.log(colors.trap(dataCPU1));
```

```
D:\zzz coding\tes2>node soal2
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":10879734,"nice":0,"sys":6917937,"idle":201466609,
"irq":2479843}}
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":6995859,"nice":0,"sys":1804484,"idle":210463812,
"irq":71890}}
```

#3 Buatlah sebuah program yang saat dijalankan akan menampilkan data CPU user yang memiliki user times tertinggi & terendah!

Gunakan module OS & Underscore!

```
D:\zzz coding\tes2>node soal4
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":11513875,"nice":0,"sys":3339375,"idle":204615265,
"irq":36250}}
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":6115734,"nice":0,"sys":1956796,"idle":211395984,
"irq":34750}}
```

#3 Solved!

```
const colors = require('colors')
const os = require('os')
const _ = require('underscore');

var data = os.cpus();
var tertinggi = _.max(data, (x) => x.times.user);
var terendah = _.min(data, (x) => x.times.user);

console.log(colors.bgGreen(JSON.stringify(tertinggi)));
console.log(colors.bgRed(JSON.stringify(terendah)));
```

```
D:\zzz coding\tes2>node soal4
```

```
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":11513875,"nice":0,"sys":3339375,"idle":204615265,
"irq":36250}}
```

```
{"model":"Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz","speed":2594,
"times":{"user":6115734,"nice":0,"sys":1956796,"idle":211395984,
"irq":34750}}
```

#4 Buatlah sebuah program yang saat dijalankan di terminal dengan parameter sebuah angka, dapat menentukan angka yang diinput user tersebut tergolong genap atau ganjil!

Gunakan module Yargs!

```
D:\zzz coding\tes2>node soal3 --x=26  
Angka 26 itu GENAP!
```

```
D:\zzz coding\tes2>node soal3 --x=11  
Angka 11 itu GANJIL!
```

#4 Solved!

```
const colors = require('colors')
const argv = require('yargs').argv

if (argv.x % 2 == 0) {
  console.log
  (colors.bgGreen(`Angka ${argv.x} itu GENAP!`))
}
else {
  console.log
  (colors.bgMagenta(`Angka ${argv.x} itu GANJIL!`))
}
```

```
D:\zzz coding\tes2>node soal3 --x=26
Angka 26 itu GENAP!
```

```
D:\zzz coding\tes2>node soal3 --x=11
Angka 11 itu GANJIL!
```


Back-End Development

Exploring

#4 Node Package Manager

