The Complete Node.JS Developer

March 2018

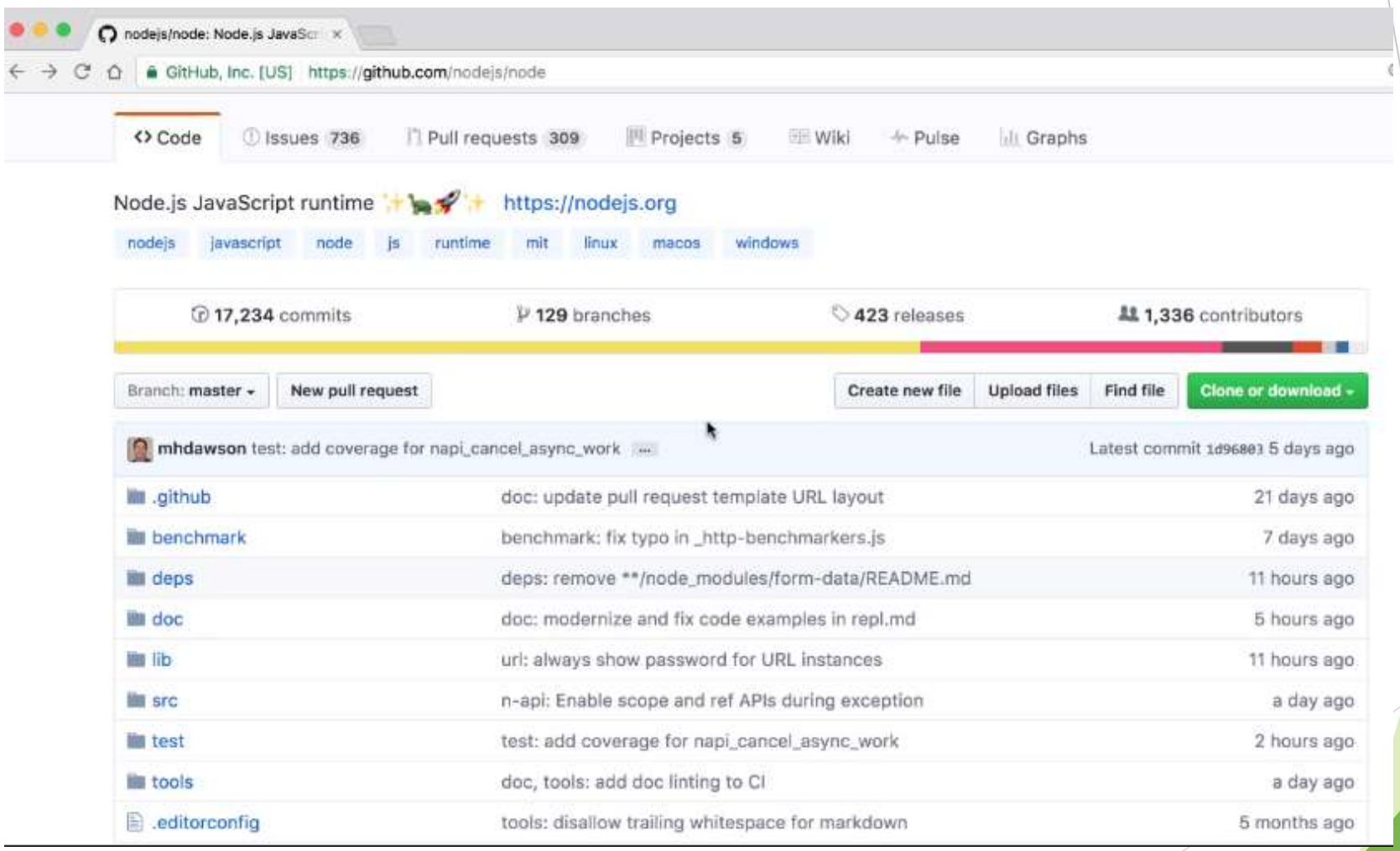
What is Node.js

- Its an open source, server side, runtime environment.
- Node is a JS runtime that runs on V8 engine.
- The V8 engine is an open source JS engine written in C++, that takes JS code and compiles it to machine code.
- The V8 engine is used both inside node and the chrome browser.
- Truly cross platform
- Uses JS as its language

What Node.js Is

Server Side	Cross Platform
JavaScript is awesome!	Open Source

- Node Community



- What is Node

- With Node we can create JS applications outside the context of the browser.
- With node we have a feature set that is similar to JAVA, PHP,RUBY, .NET etc.
- With node we can write JS applications that can access the file system, query databases, create webservers.
- Both node and the JS running inside of the browser are running on the exact same engine –V8 JS runtime engine

What is Node.js

- The V8 engine is an open source engine that takes JavaScript code and compiles it into much faster machine code and this is the reason why node.js is so fast.
- The V8 engine is written in C++
- We are not going to write any code in C++ because we are not extending node.js but we are going to use node.js for which JS is used.
- By writing “node ” on the command prompt we are creating a new node process
- D:/nodeDemo>node
- >

```
> console.log('Hello world!');
Hello world!
undefined
>
```

- When the above command is written in the node prompt what node does is, it takes the JS code and compiles it into machine code and executes it.
- The V8 engine is running behind when the JS command is being executed.

- **The Browser's V8 and Node's V8**

Browser's V8	Node's V8
The browser has feature like manipulating what is shown inside the browser window.	Node has features like the file system and their manipulation
Inside the browser "window" is the global object.	In node the global object is "global"
Window stores objects like Array, CSS manipulation and HTML objects	Most of the objects are similar to window except it does not have CSS, HTML etc.
Browser has an object called as "document" which stores the reference of the DOM of the current browser window.	"document" object is not available in node but something similar to document , called "process" is part of node. "process" gives specific information about what is being processed. "process.exit(0)" will shut down the current process in node or press **ctrl+c** twice to exit from the current process.

- **Why Node.js**
- So far we have seen that node is JS runtime on chromes V8 JS engine.
- The 2nd statement in the nodejs.org home page states that "Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient"
- Let's explore the above underlined statement with an example.
- **What is I/O**: I/O is what any application does all the time.

  Like reading/ writing from/to the database is I/O

  Changing the files from the file system

  Making an HTTP request to a separate webserver

  I/O is the communication between the node application inside the IOT.
- I/O takes time to execute.

Why Node.js

- **Non Blocking I/O :** When one user is requesting some data from the server, the other user is requesting some manipulated files without stopping the other user's process is non blocking.
- Node is single threaded.
- Because of Node being single threaded and non blocking it executes the code in half the time compared to blocking

Why Node.js

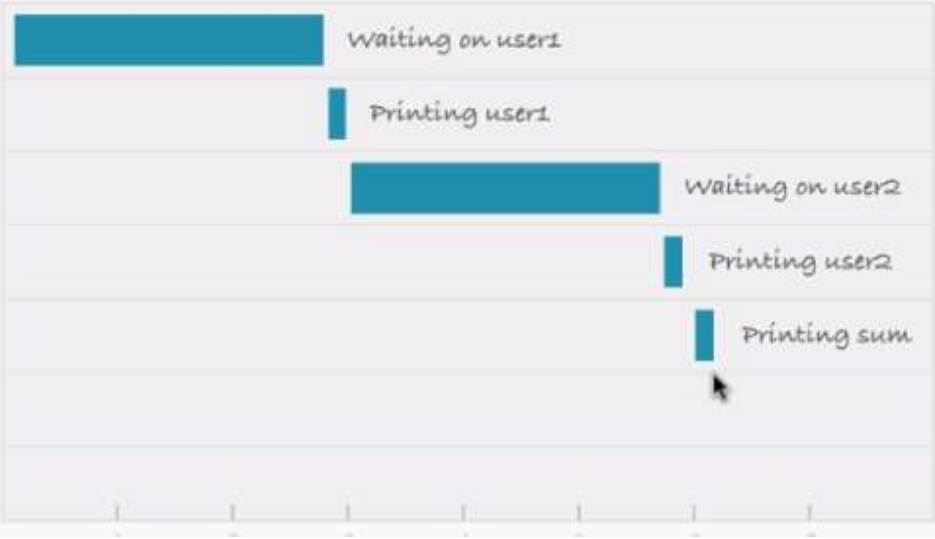
Blocking

Non-blocking

blocking.js — /Users/Andrew/Desktop/blocking-demo

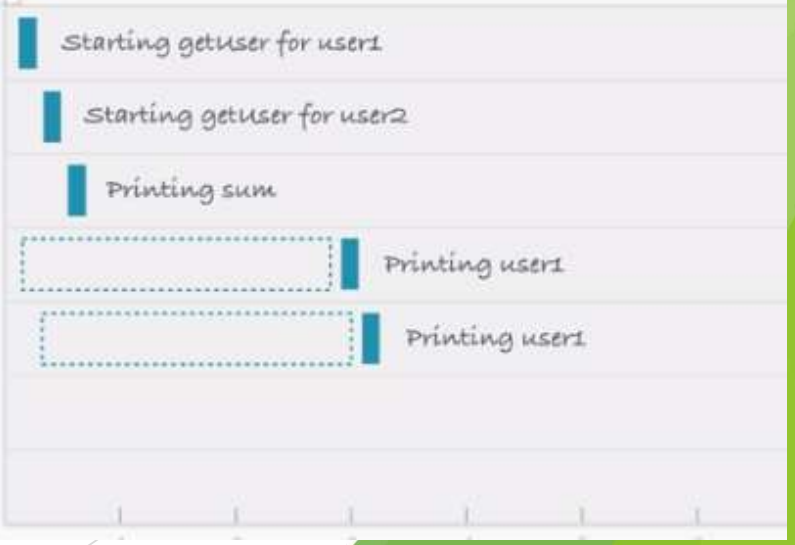
blocking.js

```
var getUserSync = require('./getUserSync');

var user1 = getUserSync('123');
console.log('user1', user1);

var user2 = getUserSync('321');
console.log('user2', user2);

var sum = 1 + 2;
console.log('The sum is ' + sum);
```



non-blocking.js

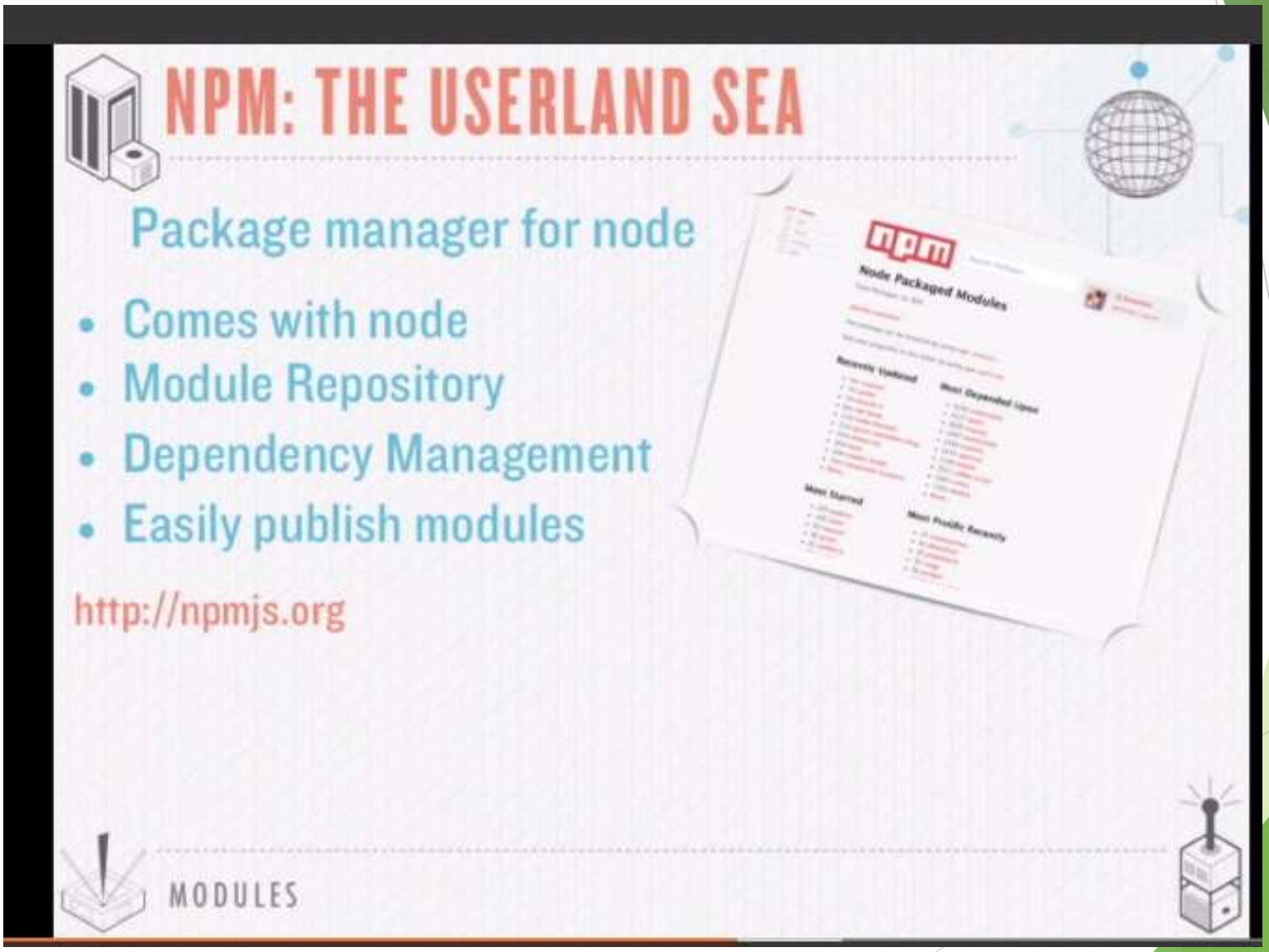
```
var getUser = require('./getUser');

getUser('123', function (user1) {
  console.log('user1', user1);
});

getUser('321', function (user2) {
  console.log('user2', user2);
});

var sum = 1 + 2;
console.log('The sum is ' + sum);
```



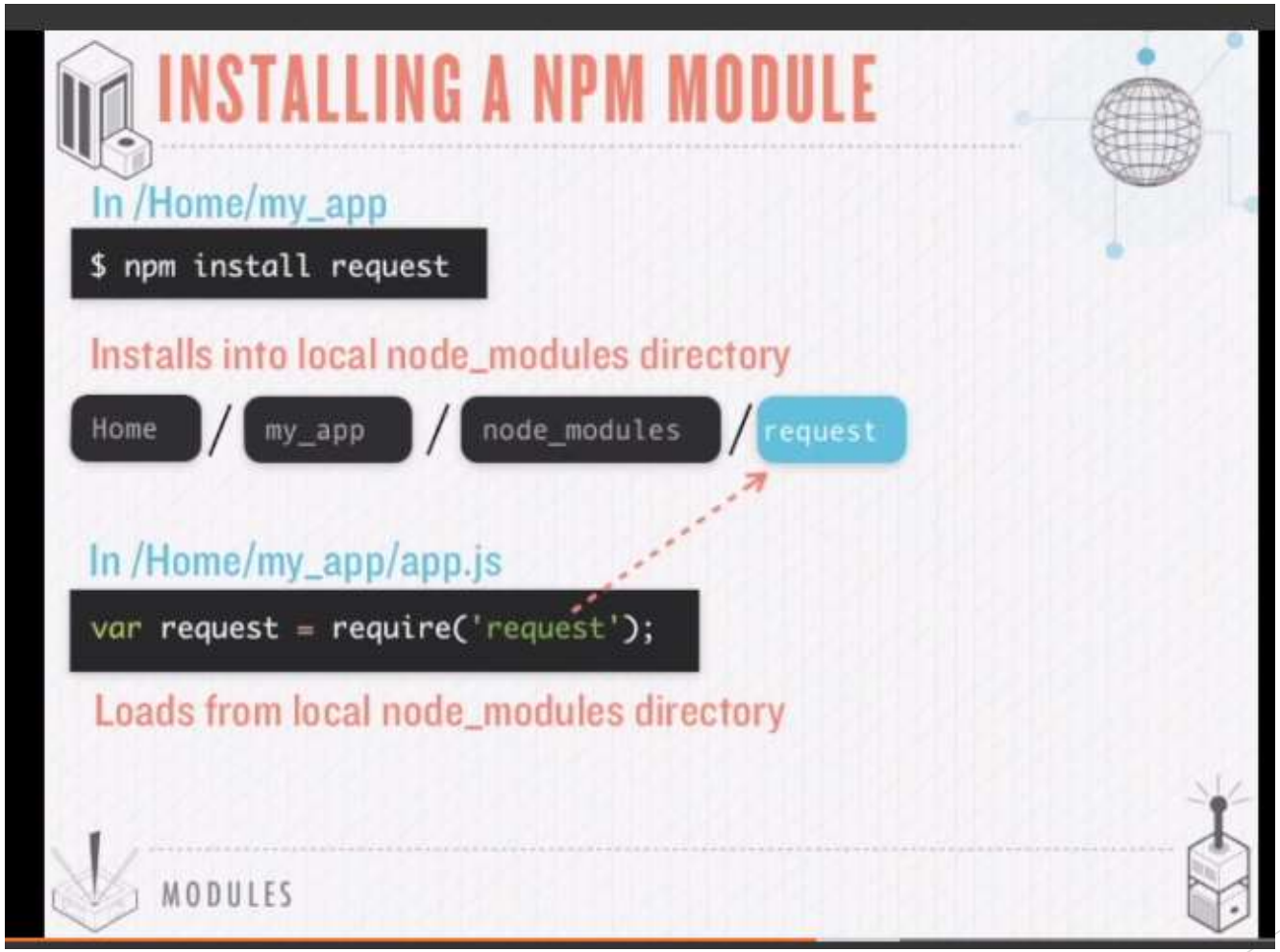
- **Why Node.js**
- The 3rd statement on the home page of nodejs.org is that the "Node.js ecosystem, npm, is the largest ecosystem of open source libraries in the world"
- This means the node community is very large and people are developing many libraries to solve common issues or problems.

- **<u>Using Require</u>**
- Require is a function in node.js
- Its helps us do 3things
  - It let’s us load modules that come bundled with up with node.js
  - Require is used to load built in modules like http, fs etc

- **NPM**


NPM: THE USERLAND SEA
Package manager for node
Comes with node
Module Repository
Dependency Management
Easily publish modules
http://npmjs.org
npm
Node Packaged Modules
MODULES

NPM


INSTALLING A NPM MODULE
In /Home/my_app
$ npm install request
Installs into local node_modules directory
Home / my_app / node_modules / request
In /Home/my_app/app.js
var request = require('request');
Loads from local node_modules directory
MODULES

NPM

LOCAL VS GLOBAL

Install modules with executables globally

```
$ npm install coffee-script -g
```

global

```
$ coffee app.coffee
```

Global npm modules can't be required

```
var coffee = require('coffee-script');
```

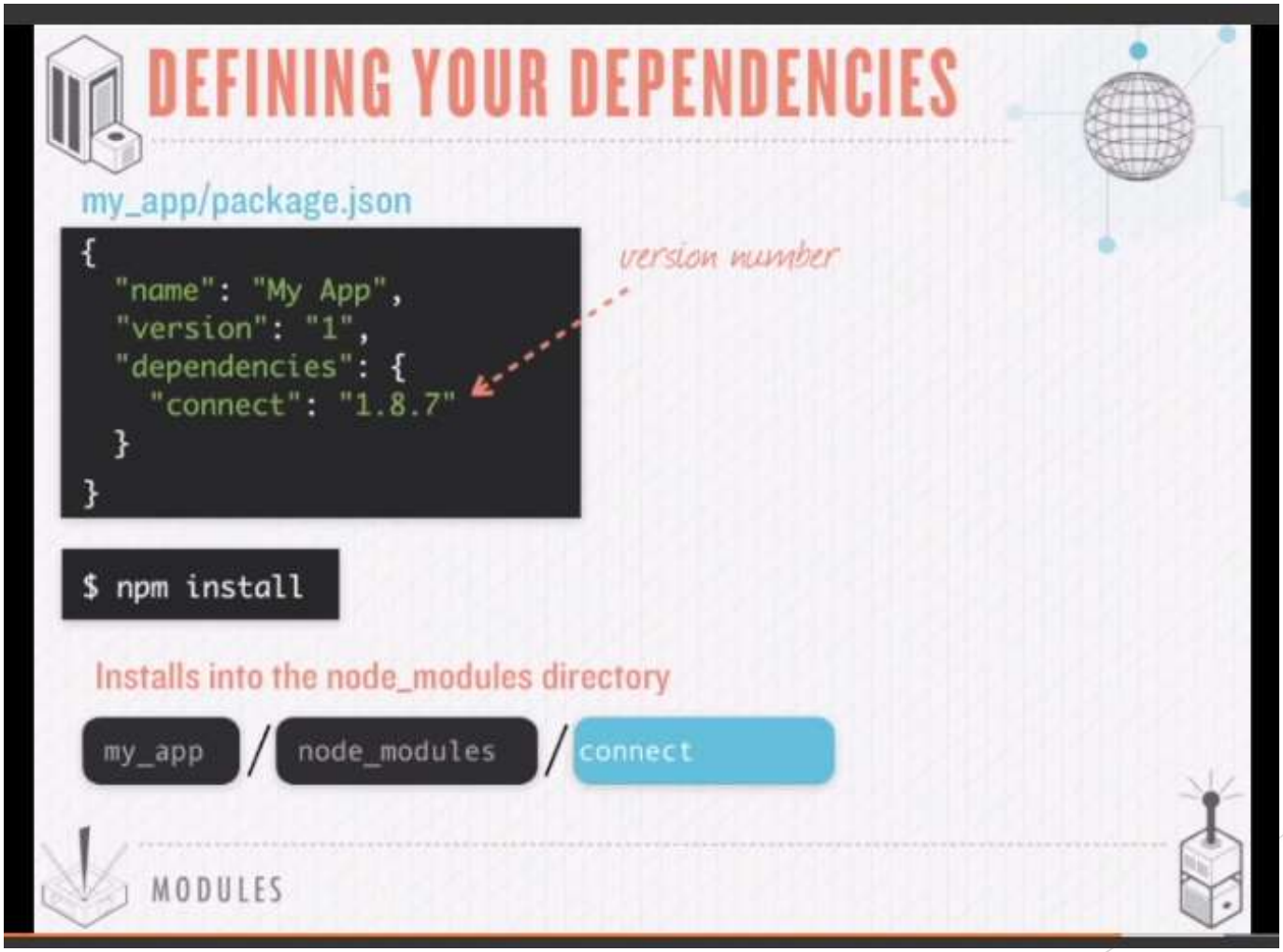
```
$ npm install coffee-script
```

Install them locally

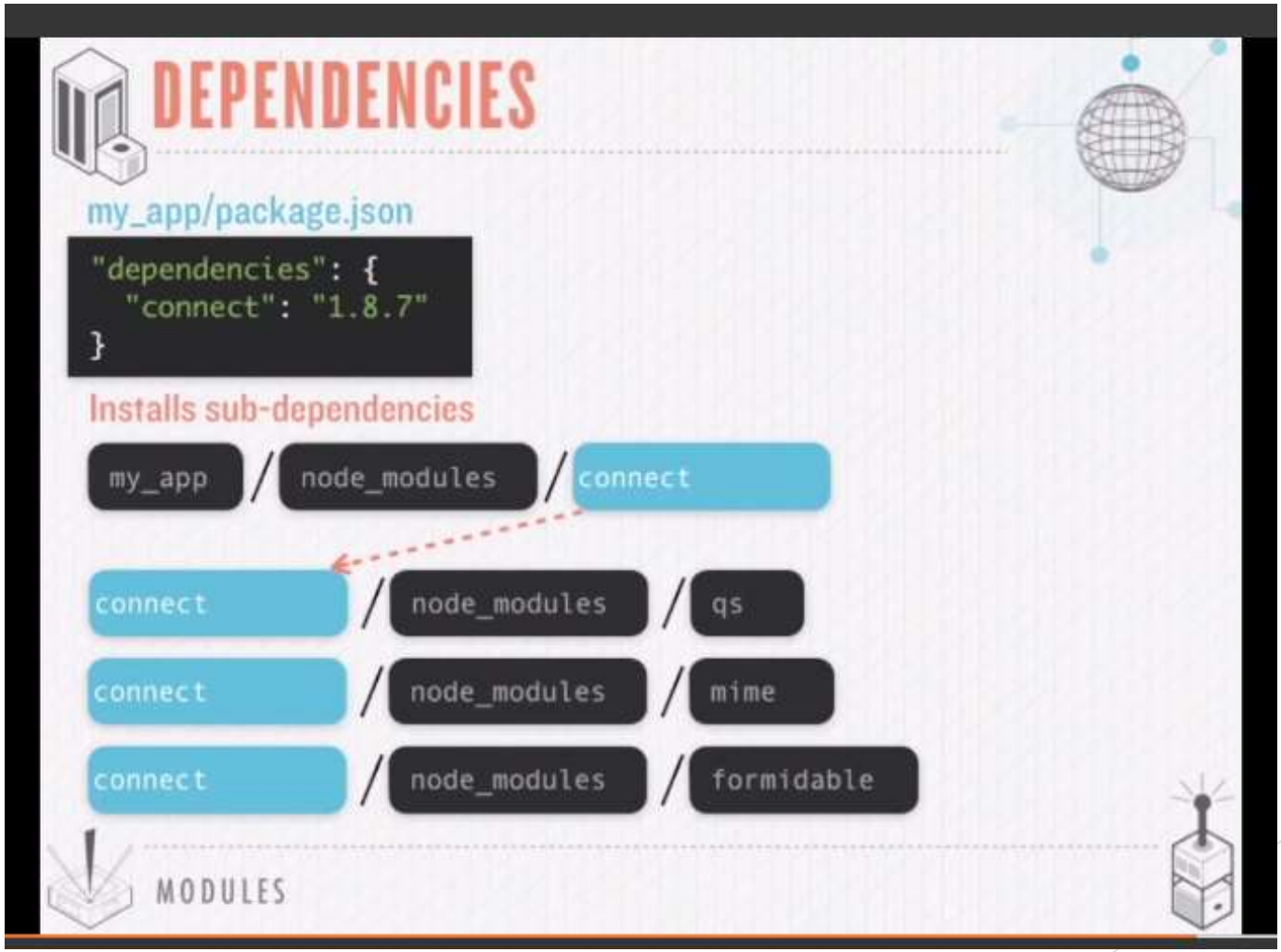
```
var coffee = require('coffee-script');
```

MODULES

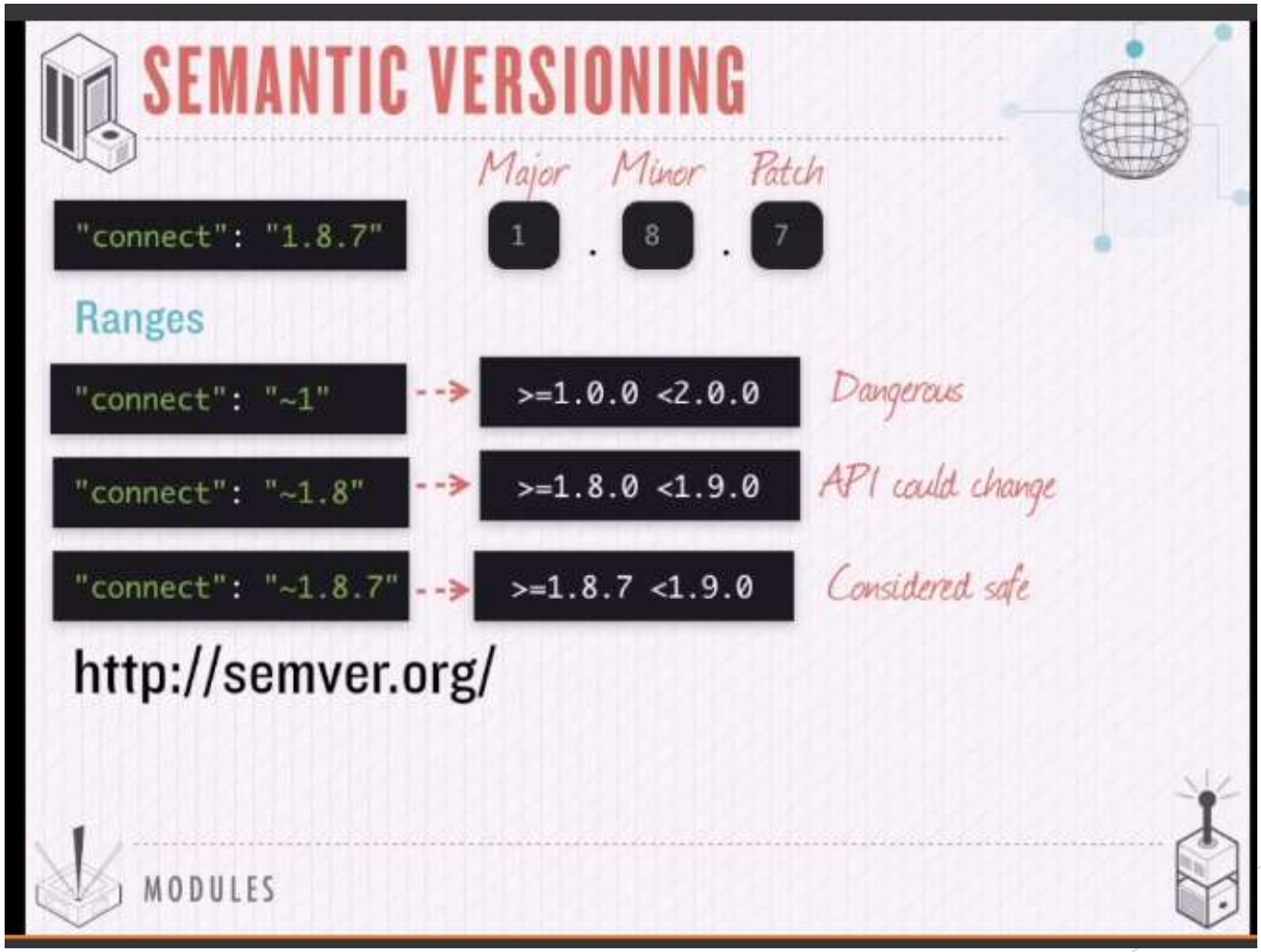
NPM


DEFINING YOUR DEPENDENCIES
my_app/package.json
{
"name": "My App",
"version": "1",
"dependencies": {
"connect": "1.8.7"
}
}
version number
$ npm install
Installs into the node_modules directory
my_app / node_modules / connect
MODULES

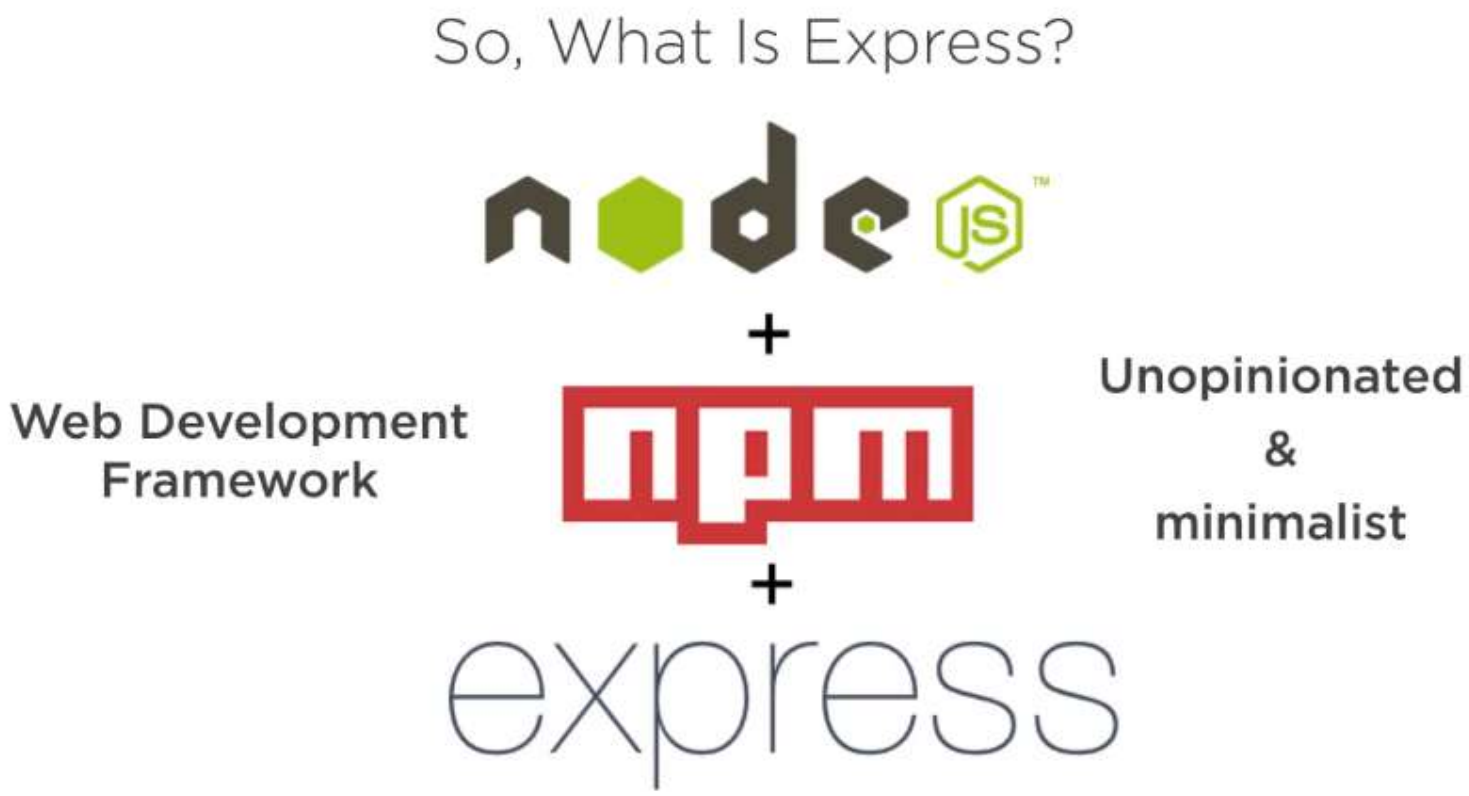
NPM


DEPENDENCIES
my_app/package.json
"dependencies": {
"connect": "1.8.7"
}
Installs sub-dependencies
my_app / node_modules / connect
connect / node_modules / qs
connect / node_modules / mime
connect / node_modules / formidable
MODULES

NPM

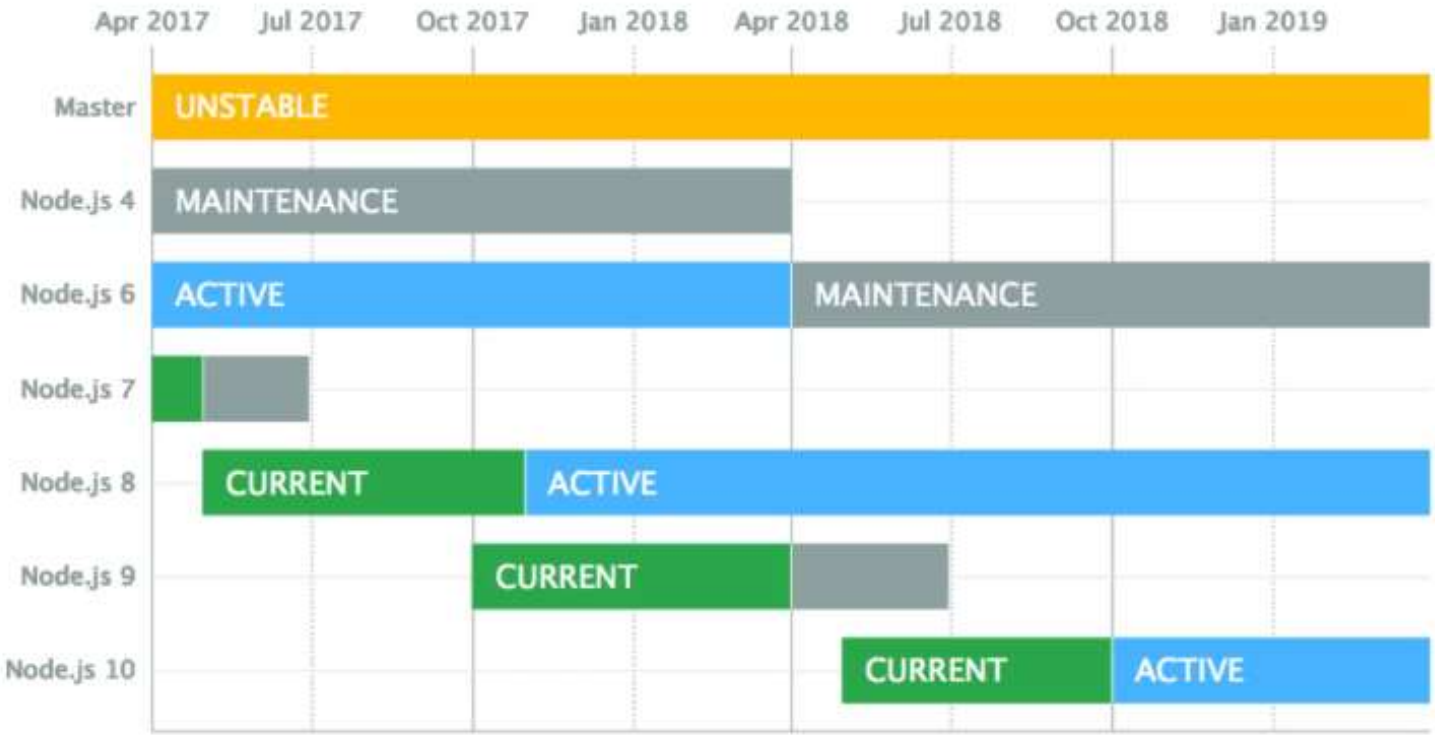

SEMANTIC VERSIONING
Major
Minor
Patch
"connect": "1.8.7"
1 . 8 . 7
Ranges
"connect": "~1"
>=1.0.0 <2.0.0
Dangerous
"connect": "~1.8"
>=1.8.0 <1.9.0
API could change
"connect": "~1.8.7"
>=1.8.7 <1.9.0
Considered safe
http://semver.org/
MODULES

Express


So, What Is Express?
node
+
Web Development Framework
npm
+
express
Unopinionated & minimalist

Dealing With Node Versions

Current Version Status Of Node



- **Dealing With Node Versions**
- It is quite possible that one is working on Node version 6
- Some other project could be Node version 8
- One may have Node 10 and so on..
- To deal with version issues, there is a tool called
  - **NVM – Node Version Manager**
- NVM allows to use or change the node version that is being used

NVM

creationix/nvm: Node Version

GitHub, Inc. [US] | https://github.com/creationix/nvm

Node Version Manager

build passing | version v0.33.8 | cii best practices passing

Table of Contents

Choosing the IDE

Which IDE?

Sublime Text

WebStorm

VS Code

.npmrc

npmrc | npm Documentation

Secure https://docs.npmjs.com/files/npmrc

npmjs.com

npm

Getting Started

How npm works

Private Modules

Troubleshooting

Using npm

CLI Commands

Configuring npm

folders

npmrc

package.json

npm policy documents

npm, the Company

view all on one page

npmrc

The npm config files

Description

npm gets its config settings from the command line, environment variables, and `npmrc` files.

The `npm config` command can be used to update and edit the contents of the user and global npmrc files.

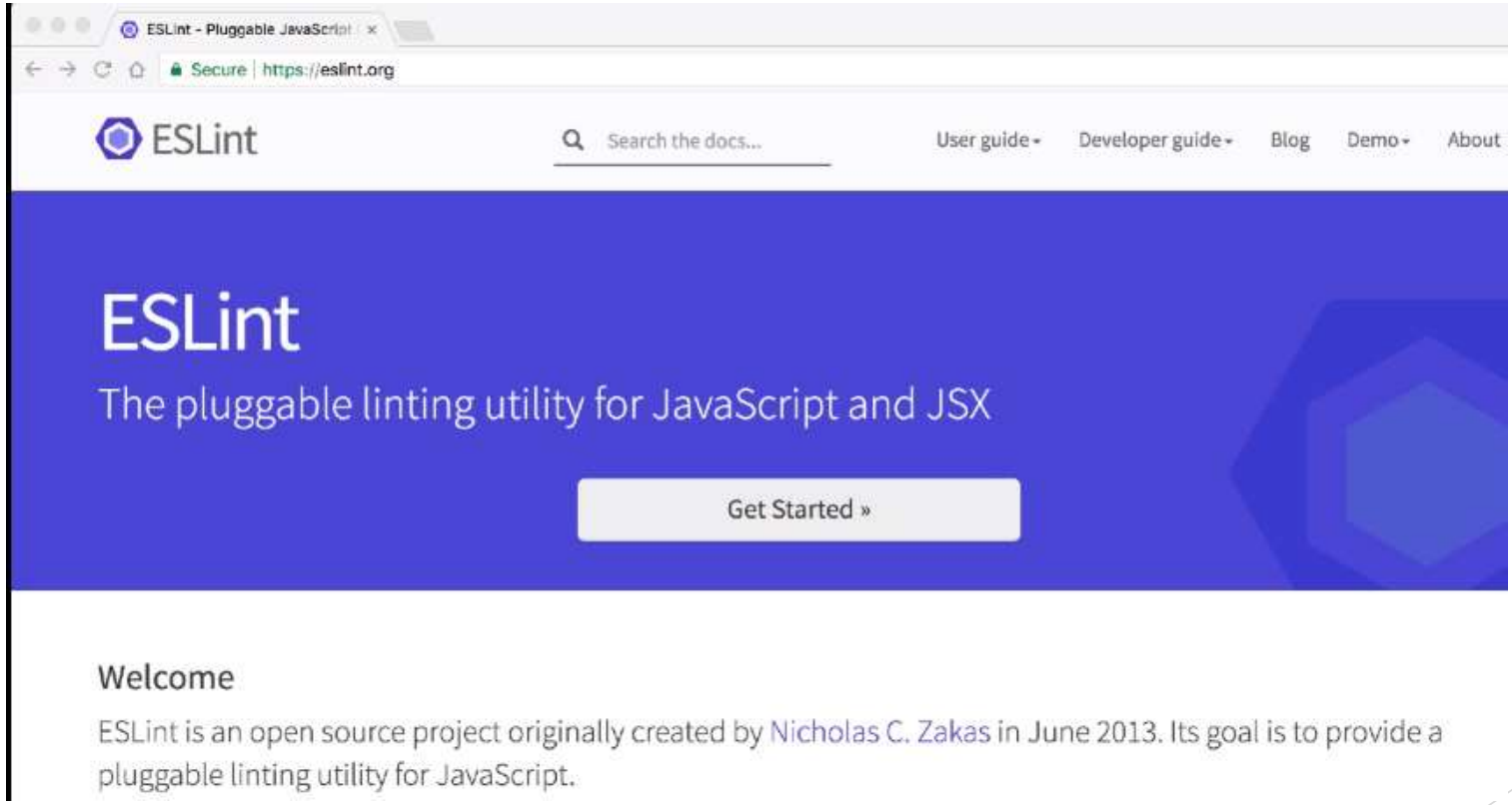
For a list of available configuration options, see npm-config.

FILES

The four relevant files are:

- per-project config file (/path/to/my/project/.npmrc)
- per-user config file (~/.npmrc)
- global config file ($PREFIX/etc/npmrc)

- **ESLint Installation**
- ES lint is a static code analysis



Node.green

Secure | https://node.green

Apps | Yahoo India | Variable Scope (Java | AngularJS Tutorial | | Bookmarks | Hello World!! | Make custom, reusab | Deep Dive into Custo | AngularJS AJAX | tut | Other bookmarks

Node.js ES2015 Support — kangax's compat-table applied only to Node.js — Learn more

requires harmony flag — Created by William Kapke

	Nightly! 11.0.0 (99% complete)	10.9.0 (99% complete)	10.8.0 (99% complete)	10.3.0 (99% complete)	9.11.2 (99% complete)	8.9.4 (99% complete)	8.6.0 (99% complete)	8.2.1 (99% complete)	7.10.1 (99% complete)	7.5.0 (99% complete)	6.14.4 (99% complete)	6.4.0 (95% complete)	5.12 (57% com)
temporal dead zone (strict mode)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
for/for-in loop iteration scope (strict mode)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
block-level function declaration	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
functions													
arrow functions													
0 parameters	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1 parameter, no brackets	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
multiple parameters	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
lexical "this" binding	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
"this" unchanged by call or apply	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
can't be bound, can be curried	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
lexical "arguments" binding	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
no line break between params and =>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
correct precedence	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
no "prototype" property	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
lexical "super" binding in constructors	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Fla
lexical "super" binding in methods	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Fla

```
function(){
  return (() => 5)() === 5;
}
```

Node.green

Secure | https://node.green

Apps | Yahoo India | Variable Scope (Java | AngularJS Tutorial | | Bookmarks | Hello World!! | Make custom, reusab | Deep | Other bookmar

asyn 1/9

Node.js ES2017 Support

kangax's compat-table applied only to Node.js

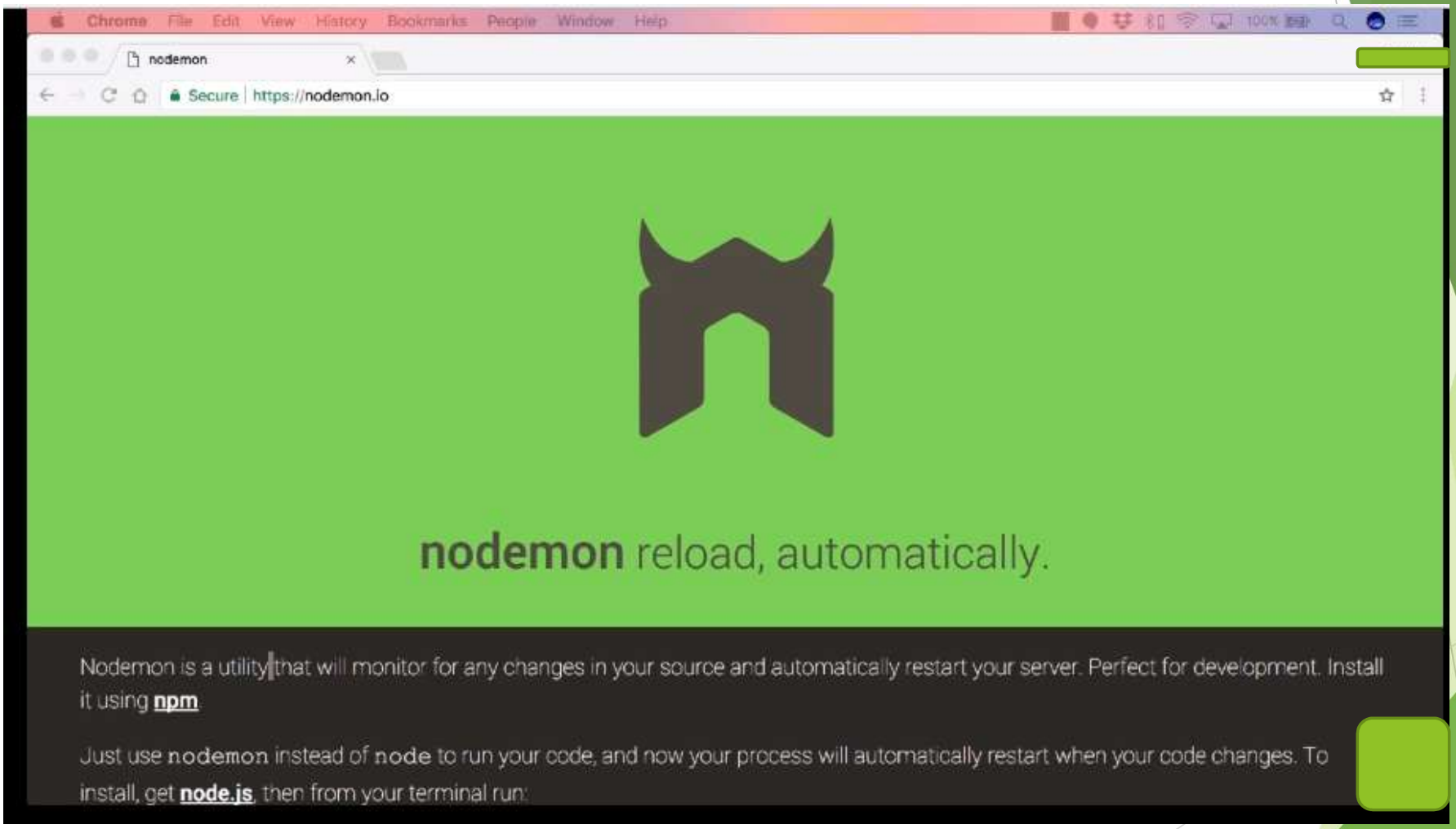
Learn more

Requires harmony flag | Created by William Kapke

	Nightly!	11.0.0 100% complete	10.9.0 100% complete	10.8.0 100% complete	10.3.0 100% complete	9.11.2 100% complete	8.9.4 73% complete	8.6.0 73% complete	8.2.1 65% complete	7.10.1 54% complete	7.5.0 54% complete	6.14.4 23% complete	6.4.0 18% complete	5.1 17% co
Object.getOwnPropertyDescriptors		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Flag	Error	Er
Object.getOwnPropertyDescriptors doesn't provide undefined descriptors		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Error	Er
String padding														
String.prototype.padStart		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Flag	Flag	Error	Error	Er
String.prototype.padEnd		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Flag	Flag	Error	Error	Er
trailing commas in function syntax														
in parameter lists		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error	Er
in argument lists		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error	Er
async functions														
return		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
throw		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
no line break between async and function		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
no "prototype" property		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
await		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
await, rejection		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er
must await a value		s				Yes	Yes	Yes	Yes	Yes	Flag	Error	Error	Er

```
function(){
  async function a(){
    return "foo";
  }
  var p = a();
  if (!(p instanceof Promise)) {
    return false;
  }
  p.then(function(result) {
```

Nodemon


Chrome File Edit View History Bookmarks People Window Help
nodemon
Secure https://nodemon.io
nodemon reload, automatically.
Nodemon is a utility that will monitor for any changes in your source and automatically restart your server. Perfect for development. Install it using npm.
Just use nodemon instead of node to run your code, and now your process will automatically restart when your code changes. To install, get node.js, then from your terminal run:

- **Templating Engines**
- Pug (changed the name from jade)

```html
<html>
    <head>
        <title>MyApp</title>
    </head>
    <body>
        <h1>My Title</h1>
        <p>
            <h3>My Sub</h3>
        </p>
    </body>
</html>
```

Normal HTML

```
html
    head
        title MyApp
    body
        h1 My Title
        p
            h3 My Sub
```

Html with pug templating engine

Pug and JavaScript

```
<html>
    <head>
        <title>MyApp</title>
    </head>
    <body class="MyClass">
        <h1>My Title</h1>
        <p>
            <h3>My Sub</h3>
        </p>
    </body>
</html>
```

```
html
    head
        title MyApp
    body(class=["MyClass"])
        h1 My Title
        p
            h3 My Sub
```

- **Why Node.js**

What is Express?

- Express is a web development framework for node.js
- Express is the web piece that will be built inside Node.js
- Express is lightweight, un-opinionated and minimalist web framework used to build the website
- Express handles routing , page rendering etc.
- Its an npm package within node and code will be written using express objects

- What is NodeJS
- It is V8 JavaScript runtime
- Event Driven
- Non-blocking standard libraries
- Most API speak streams
- Extensible via C/C++ and-ons
- Provides a package manager and module system for JS/native extensions.
- Node is just JavaScript without browser.

Installing Node JS

- https://www.windowsazure.com/en-us/develop/nodejs/
- Click on widows installer link after going to nodejs.org/#download
- Choose the Windows Installer, Open it and run it

- Getting Started:

Node Js Event Loop

- Node js is a single threaded application but it support concurrency via concept of event and callbacks.
- As every API of Node js are asynchronous and being a single thread, it uses **async** function calls to maintain the concurrency.
- Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.
- Node.js uses events heavily and it is also one of the reasons why Node.js is pretty fast compared to other similar technologies.

Node JS event loop

- Events seems similar to what callbacks are.
- The difference lies in the fact that callback functions are called when an asynchronous function returns its result where as event handling works on the observer pattern.
- Node.js has multiple in-built events available through **events** module and **EventEmitter** class which is used to bind events and event listeners as follows:

```
// Import events module
var events = require('events');
// Create an eventEmitter object
 var eventEmitter = new events.EventEmitter();
```

Following is the syntax to bind event handler with an event:

```
// Bind event and even handler as follows
 eventEmitter.on('eventName', eventHandler);
```

We can fire an event programatically as follows:

```
// Fire an event
eventEmitter.emit('eventName');
```

For those familiar with client-side JavaScript development, think of all the `.on*()` methods, such as `element.onclick()`, that are used in conjunction with DOM.

Elements to convey user interaction.

This pattern works well when a single item can emit many possible events.

Node uses this pattern in the form of the `EventEmitter`, and is located in places such as `Server`, `Socket` and the `'http'` module.

It's useful when we need to emit more than one type of state change from a single instance.

Event Emitter and the Event Loop

To simplify interaction with the event loop the `EventEmitter` was created.

It is a generic wrapper that more easily allows creating event-based APIs.

The `'fs'` module mostly uses the error back callback style.
It would technically be possible to emit additional events for some calls, such as `fs.readFile()`, but the API was made to only alert user if the desired operation succeeded or if something failed.

Another common pattern is succeed or fail.
There are two common implementations around today.
First is the "error back" callback style, where the error of the call is the first argument passed to the callback.
The second has emerged with ES6, using `Promises`.

- How Node Applications Work?
- In Node Application, any async function accepts a callback as a last parameter and the callback function accepts error as a first parameter.

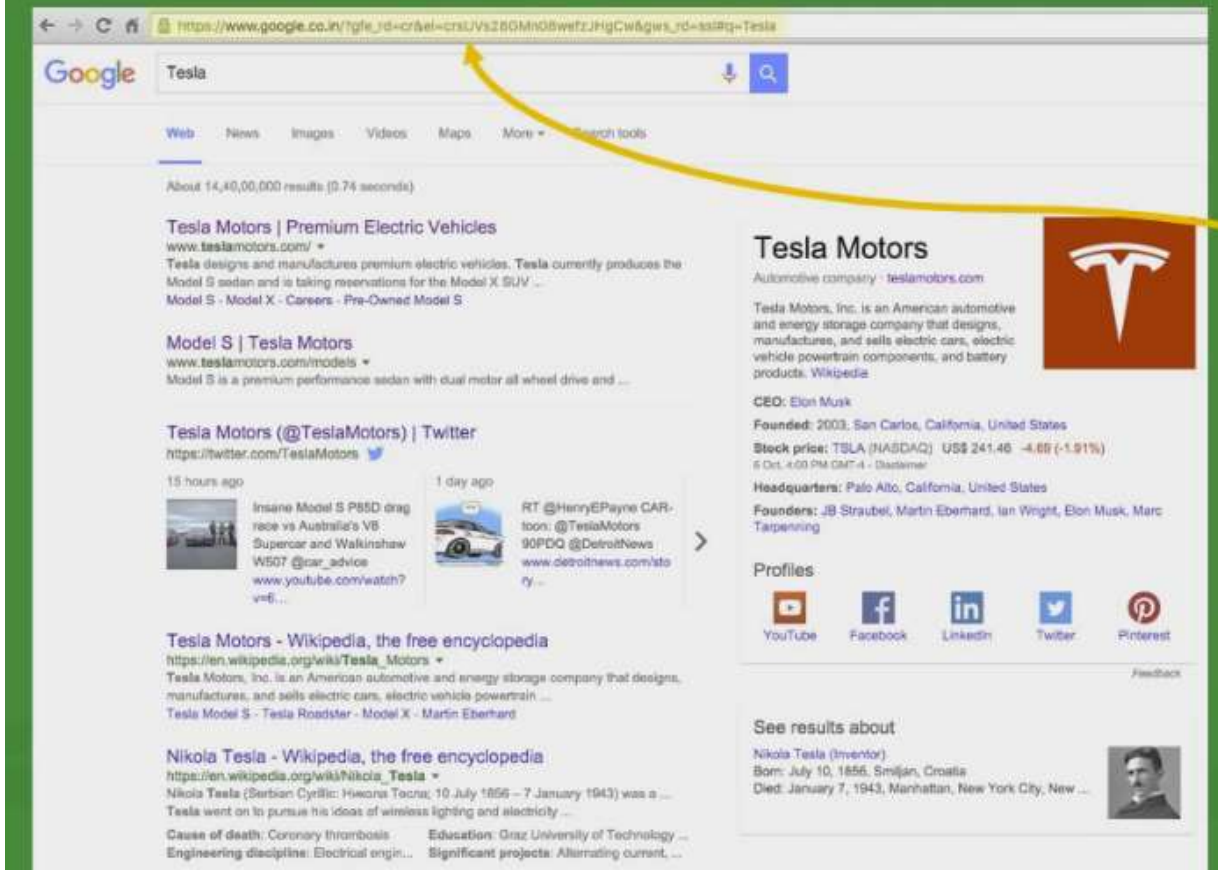
HTTP Requests and Methods

Read	GET	• Requesting read only data from an API or a resource. • URL exposes the request data e.g. /api/getData/aerolatte. • Should not be used to modify data on the server.
Create	POST	• Data is not exposed and travels in the body of the request. • Request has the potential of modifying data on the server. • Used in login/signup forms, APIs that accept & store data in a db
Update	PUT	• Used for updating existing data on the server.
Delete	DELETE	• Used for deleting / marking data for deletion on the server.

HTTP Methods & Idempotency

Read	➡	GET	• Is Idempotent :: You will always get the same data back.
Create	➡	POST	• Is Not Idempotent :: Every request may result in a new set of data created on the server.
Update	➡	PUT	• Is Idempotent :: Every request should have the same result.
Delete	➡	DELETE	• Depends upon the implementation.

HTTP Methods & Cacheability



A GET request is cacheable.

POST/PUT/DELETE are not cacheable.

Routes

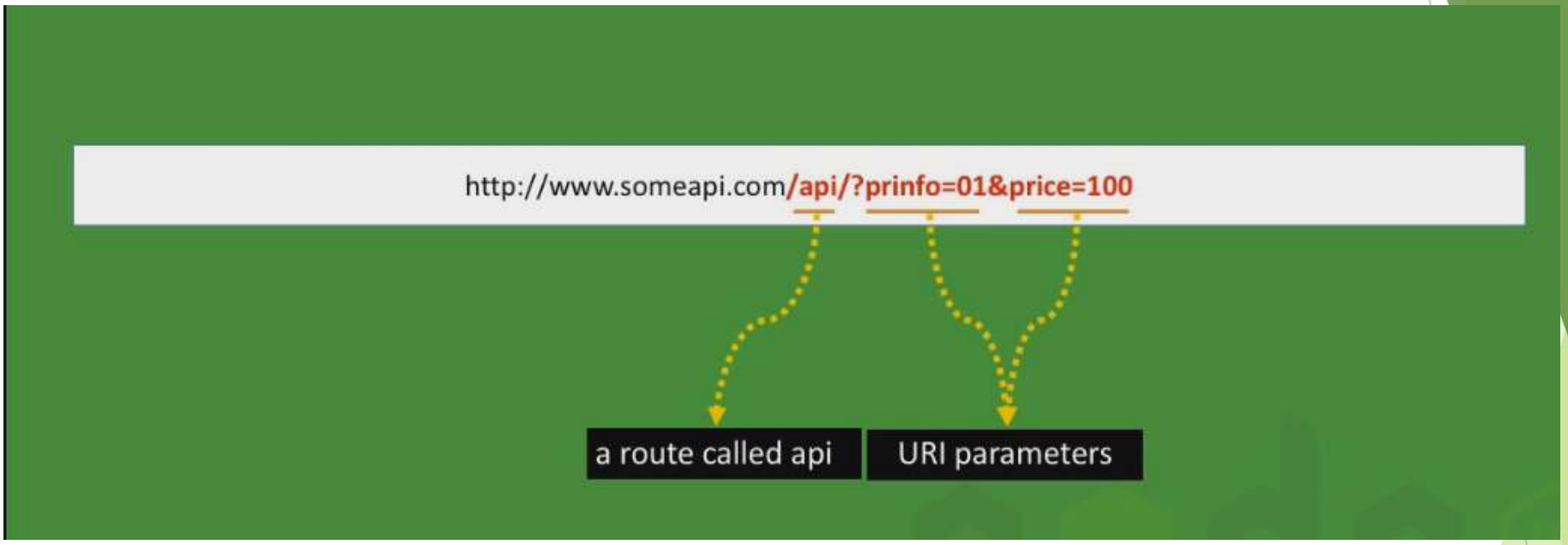
www.somesite.com/**about** → GET : We're just fetching data here.

www.somesite.com/**login** → POST : Since we're sending username and password which should not travel as a URI parameter.

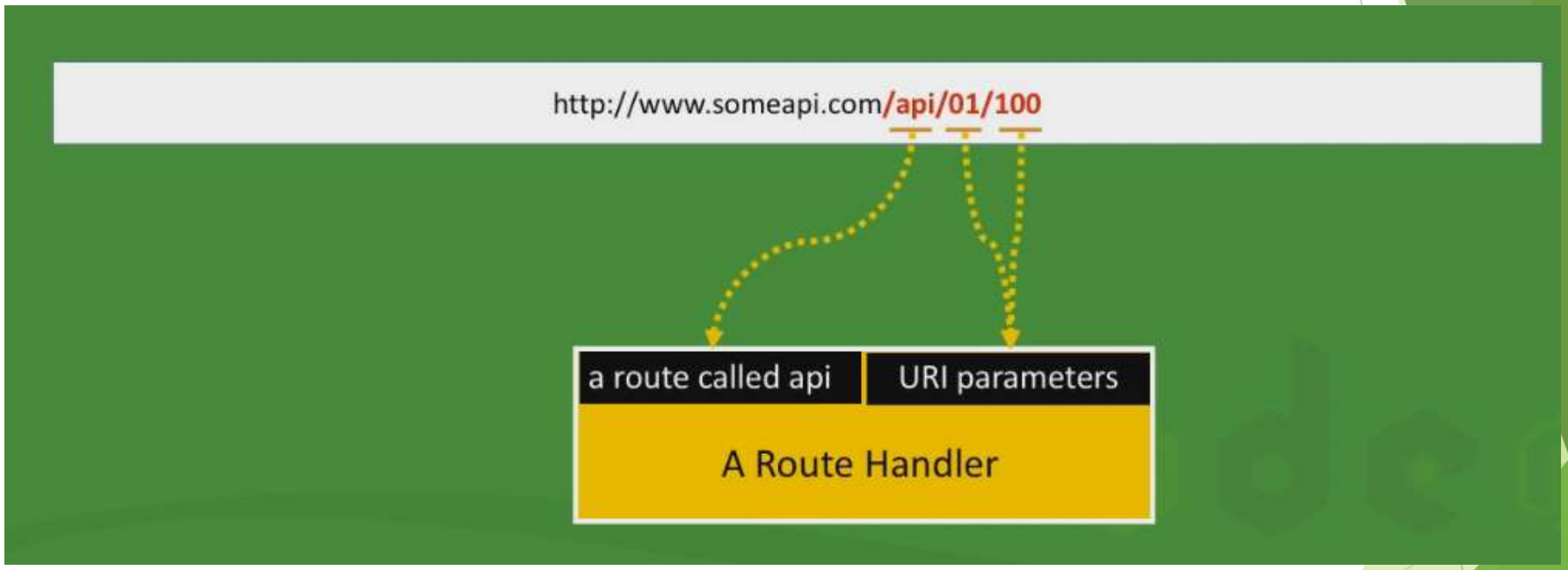
www.somesite.com/**signup** → POST : A new record would be created on the server.

www.someapi.com/**products/?id=01** → GET: Reading data about a product from an API.

- Routes


http://www.someapi.com/api/?prinfo=01&price=100
a route called api
URI parameters

- Routes


http://www.someapi.com/api/01/100
a route called api
URI parameters
A Route Handler

The **req** object

http://localhost:3000/api/products?**id=021**&**color=Red**&**color=Orange**&**sortBy=price**

req.url : /api/products?id=021&color=Red&color=Orange&sortBy=price

req.method : GET

req.headers : {

```
host: 'localhost:3000'
connection: 'keep-alive',
'cache-control': 'max-age=0',
accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*,
'upgrade-insecure-requests': '1'
}
```

- What is Express?
- Express is the web development framework for node.js
- Express is the web piece that will be built inside nodeJS
- It is lightweight and minimalist web framework that can be used to build a website
- It is an npm object that is going to sit inside node.