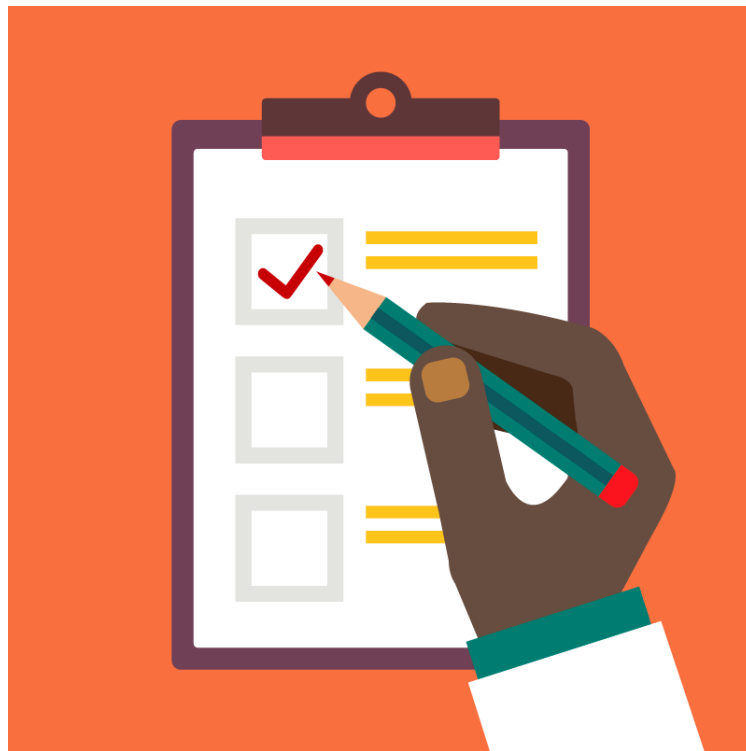




NodeJS Intro, NPM, standard libraries

Agenda

- Node.js intro;
- Installation;
- Modules in Node;
- Local modules;
- NPM;
- Built-in modules;
- HTTP;
- FS;
- OS, PROCESS, PATH, URL;



NODE.JS INTRO

Node.js intro

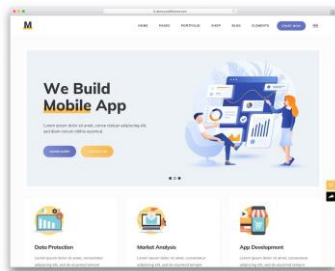
Node.js is an open-source, cross-platform, JavaScript run-time environment that executes JavaScript code outside of a browser.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Consequently, **Node.js** represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.



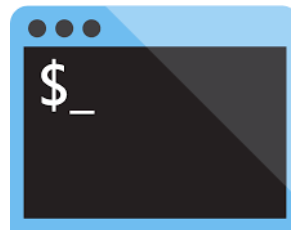
What can you do with Node.js



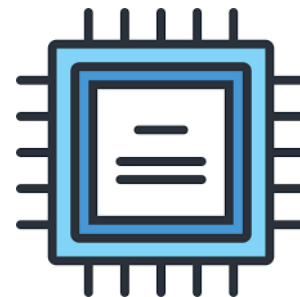
SPA
(Single page application)



RPA
(Real time application)



CLI
(Command-line interface)



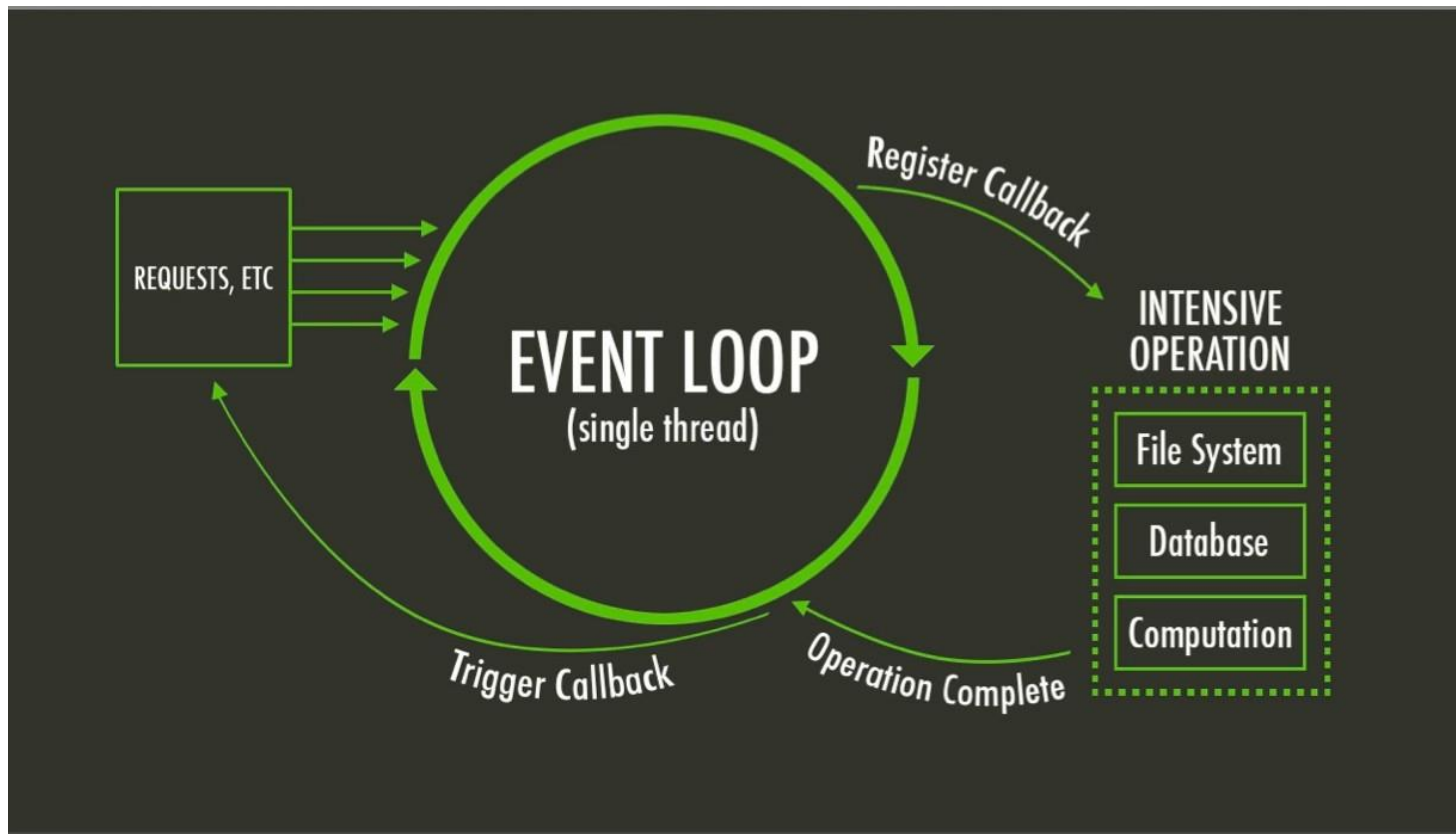
Hardware

Why you need to learn it?

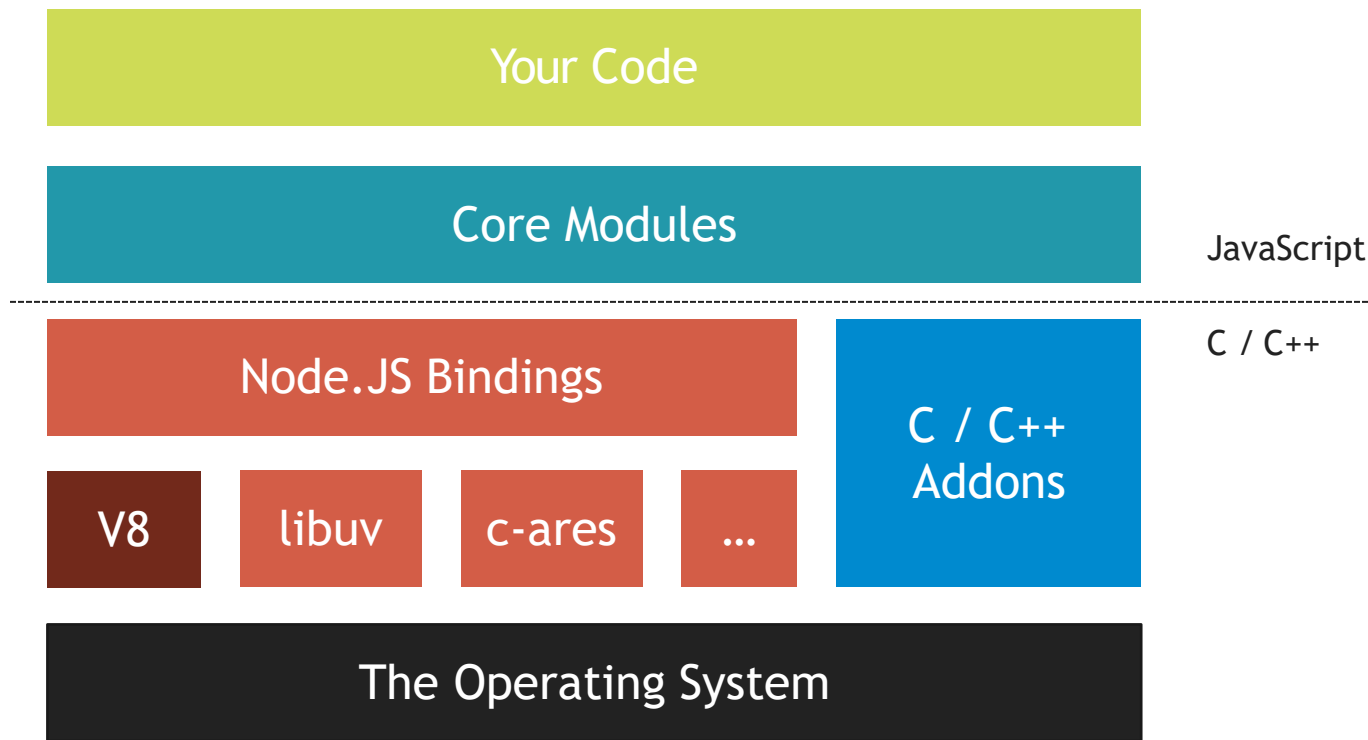


Who use Node.js

How it works?



Architecture



INSTALLATION

Node.js Environment

1. Download Node.js archive

Download latest version of Node.js installable archive file from nodejs.org;

2. Install Node.js

Follow the instructions from official site to install Node.js in your OS;

3. Run command to check that you have everything working

```
sh-3.2$ node -v
v10.13.0
sh-3.2$
```

Note: Make sure you have added nodejs bins to the PATH environment variable.



Now you have downloaded and installed Node.js on your computer, **Let's try to launch our first app!**

1. Create file

Create a JS file named "test.js", and add the following code:

```
1  
2 console.log("Hello World!");  
3
```

2. Command Line Interface

Start your command line interface, write "node test.js» and hit enter, result should look like this:

```
sh-3.2$ node test.js  
Hello World!  
sh-3.2$ █
```

NVM

NVM - Simple script to manage multiple active Node.js versions

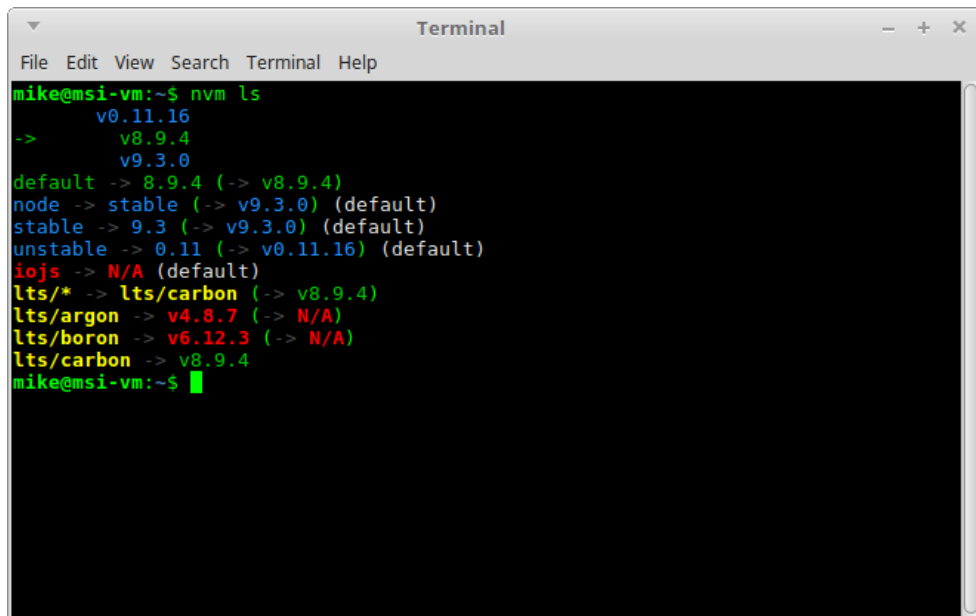
```
$ nvm ls-remote
# lists all of the available versions of Node.js

$ nvm ls
# list locally installed version

$ nvm install 0.12.3
# install the version 0.12.3

$ nvm use 0.12.3
# switch to and use the installed 0.12.3 version

$ nvm --help # the help documents
```

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "mike@msi-vm:~\$". The command "nvm ls" has been executed, showing the following output: v0.11.16, v8.9.4, v9.3.0, default -> 8.9.4 (-> v8.9.4), node -> stable (-> v9.3.0) (default), stable -> 9.3 (-> v9.3.0) (default), unstable -> 0.11 (-> v0.11.16) (default), iojs -> N/A (default), lts/* -> lts/carbon (-> v8.9.4), lts/argon -> v4.8.7 (-> N/A), lts/boron -> v6.12.3 (-> N/A), lts/carbon -> v8.9.4. The prompt is now "mike@msi-vm:~\$".

```
mike@msi-vm:~$ nvm ls
v0.11.16
-> v8.9.4
v9.3.0
default -> 8.9.4 (-> v8.9.4)
node -> stable (-> v9.3.0) (default)
stable -> 9.3 (-> v9.3.0) (default)
unstable -> 0.11 (-> v0.11.16) (default)
iojs -> N/A (default)
lts/* -> lts/carbon (-> v8.9.4)
lts/argon -> v4.8.7 (-> N/A)
lts/boron -> v6.12.3 (-> N/A)
lts/carbon -> v8.9.4
mike@msi-vm:~$
```

<https://github.com/creationix/nvm>

MODULES

Modules

What is a Module in **Node.js**? Consider modules to be the same as JavaScript libraries. A set of functions you want to include in your application.

Node.js has 3 types of modules

3rd party Modules

NPM Registry contains a huge list of 3rd party modules for all your needs

Your Own(Local) Modules

You can create your own modules, and easily include them in your applications.

Built-in(Core) Modules

Node.js has a set of built-in modules which you can use without any further installation.

Check Built-in modules reference for a complete list of modules.

Your Own Modules

You can create your own modules, and easily include them in your applications.

Example

Create a module that returns the current date and time:

```
3
4  exports.myDateTime = function () {
5    return Date();
6  };
7
```

Use the **exports** keyword to make properties and methods available outside the module file.

Now you can include and use the module in any of your Node.js files:

```
8
9  const dateUtils = require('./dateUtils');
10 const timestamp = dateUtils.myDateTime();
11 console.log(timestamp);
12
```

NPM

What is NPM?

NPM stands for **N**ode **P**ackage **M**anager. It allows for seamless node.js package management. You can install, share and manage node.js packages.

NPM consists of three components:

- **Website**
- **Registry**
- **CLI**

Website

npm official website is <https://www.npmjs.com/>. Using this website you can find packages, view documentation, share and publish packages.

Registry

npm registry is a large database consisting of more than half a million packages. Developers download packages from the npm registry and publish their packages to the registry.

CLI(Command Line Interface)

The is the command line that helps in interacting with the npm for installing, updating and uninstalling packages and managing dependencies.



The **package.json** is the project manifest file.
Using **package.json** you can manage dependencies and write scripts.
It has all the metadata about the project.

```
{  
  "name": "project",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC",  
  "devDependencies": {  
    "gulp": "^3.9.0"  
  }  
}
```

BUI LT-IN MODULES

Built-in Modules

Node.js has a set of built-in modules which you can use without any further installation.

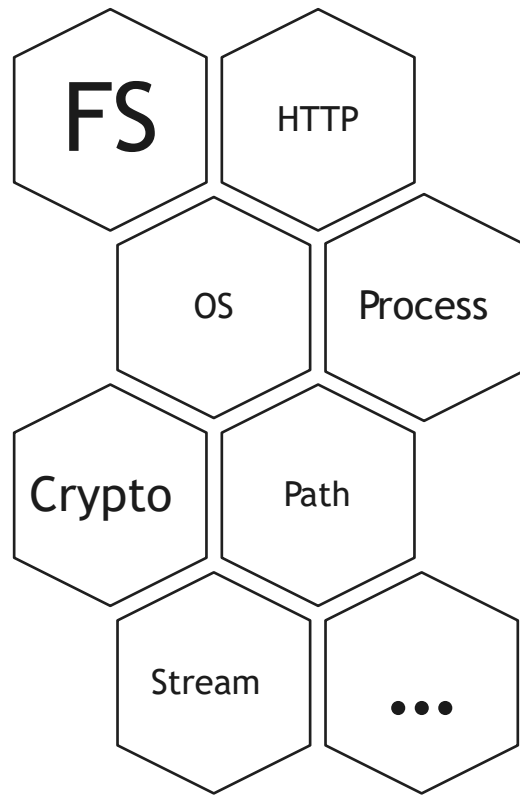
The most popular modules: http for web, os for operation system, fs for file system, path for managing path

To include a module, use the `require()` function with the name of the module:

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

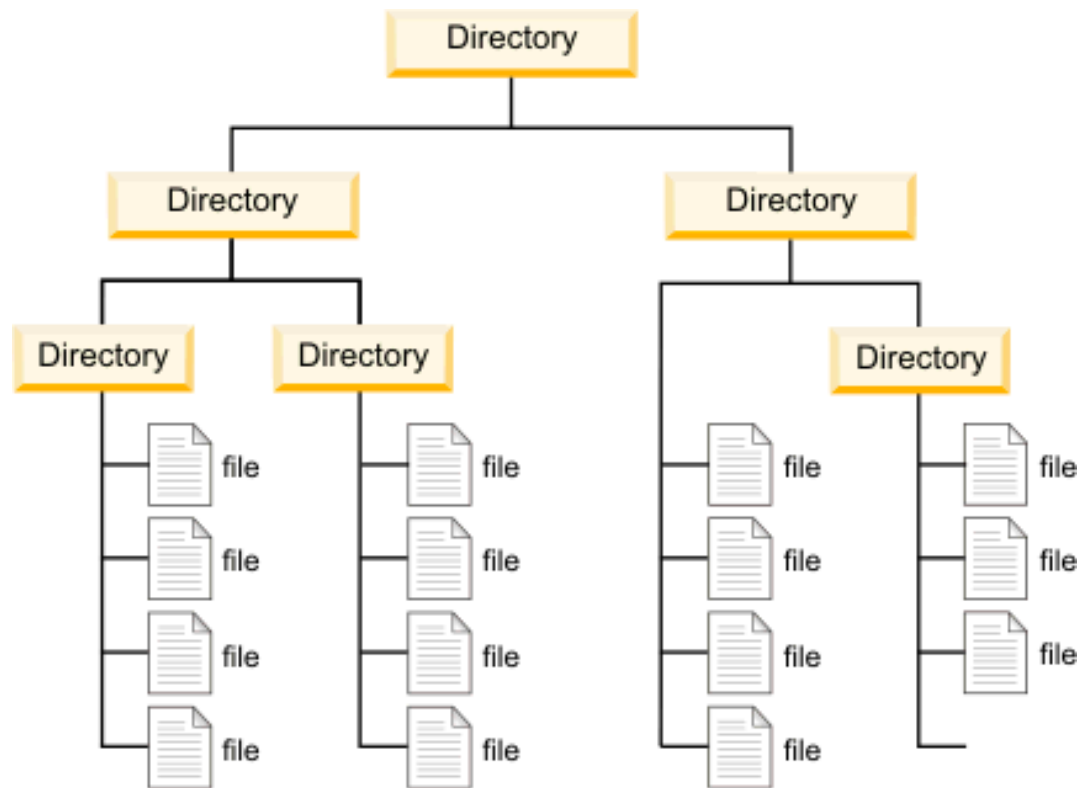
Now your application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello World!');  
}).listen(8080);
```



File system

A file system is a process that manages how and where data on a storage disk, typically a hard disk drive (HDD), is stored, accessed and managed. It is a logical disk component that manages a disk's internal operations as it relates to a computer and is abstract to a human user.



- `fs.readFileSync(<path>, <encoding>)`
- `fs.writeFileSync(<path>, <content>, <encoding>)`
- `fs.renameSync(<oldPath>, <newPath>)`
- `fs.readdirSync(<path>)`
- `fs.unlinkSync(<path>)`

HTTP, FS, OS

OS Module

```
1  /*
2   import the os module & destructure the desired properties/functions
3   similar to:
4   const os = require('os');
5   const { platform, arch, release, totalmem, freemem } = os;
6  */
7  const { platform, arch, release, totalmem, freemem } = require('os');
8
9  // log some information about the operating system
10 console.log(`Your Operating System: ${release()} ${platform()} ${arch()}`);
11
12 // log some information about the memory (ram) (number is rounded to two decimals)
13 console.log(`${((freemem() / totalmem()) * 100).toFixed(2)} % of your RAM is free.`);
14
```

- `os.arch()`
- `os.cpus()`
- `os.endianness()`
- `os.freemem()`
- `os.homedir()`
- `os.hostname()`
- `os.loadavg()`
- `os.networkInterfaces()`
- `os.platform()`
- `os.release()`
- `os.tmpdir()`
- `os.totalmem()`
- `os.type()`
- `os.uptime()`
- `os.userInfo()`

Path module

```
1 var path = require("path");
2
3 // Normalization
4 console.log('normalization : ' + path.normalize('/test/test1//2slashes/1slash/tab/..'));
5
6 // Join
7 console.log('joint path : ' + path.join('/test', 'test1', '2slashes/1slash', 'tab', '..'));
8
9 // Resolve
10 console.log('resolve : ' + path.resolve('main.js'));
11
12 // extName
13 console.log('ext name : ' + path.extname('main.js'));
```


QA