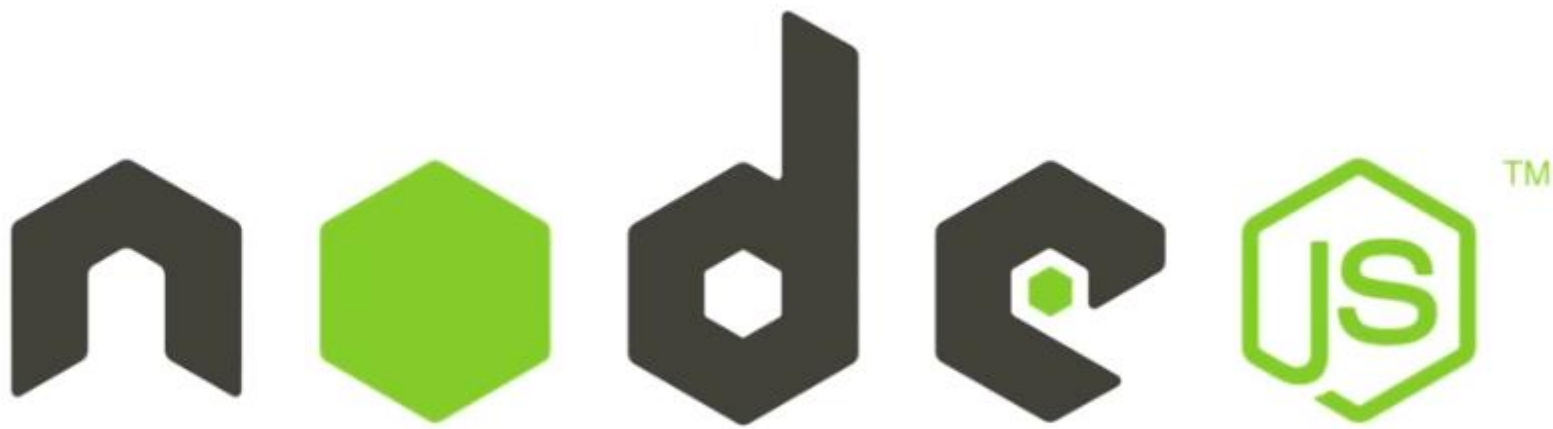
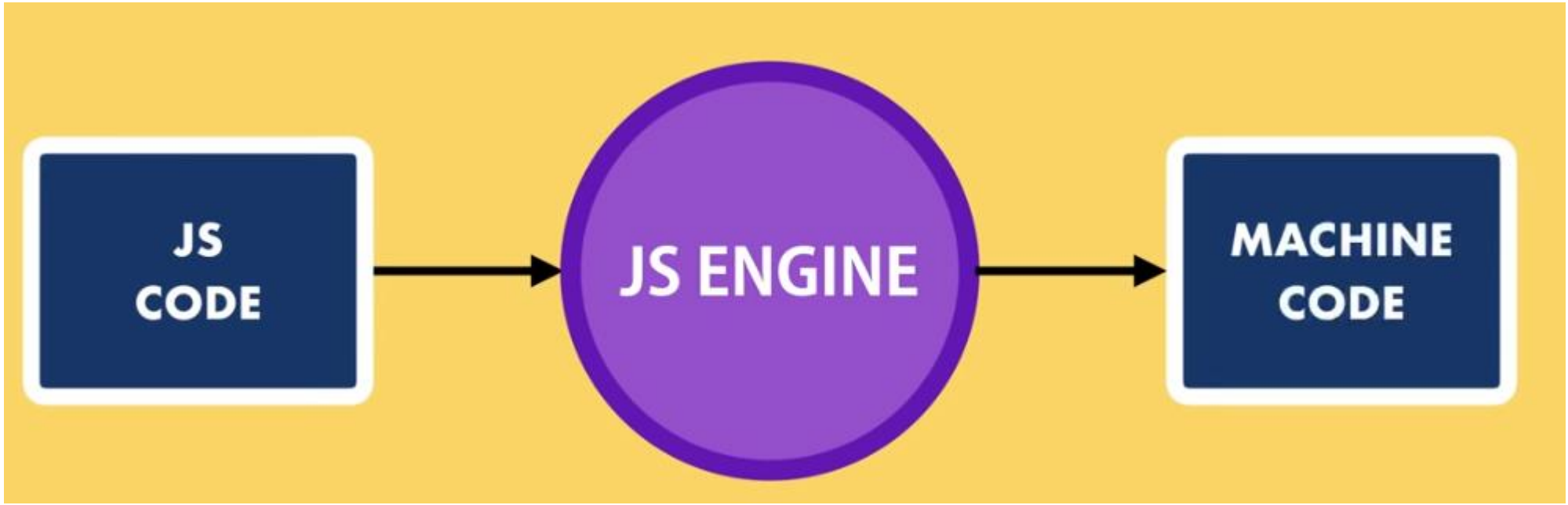

node
JS
TM

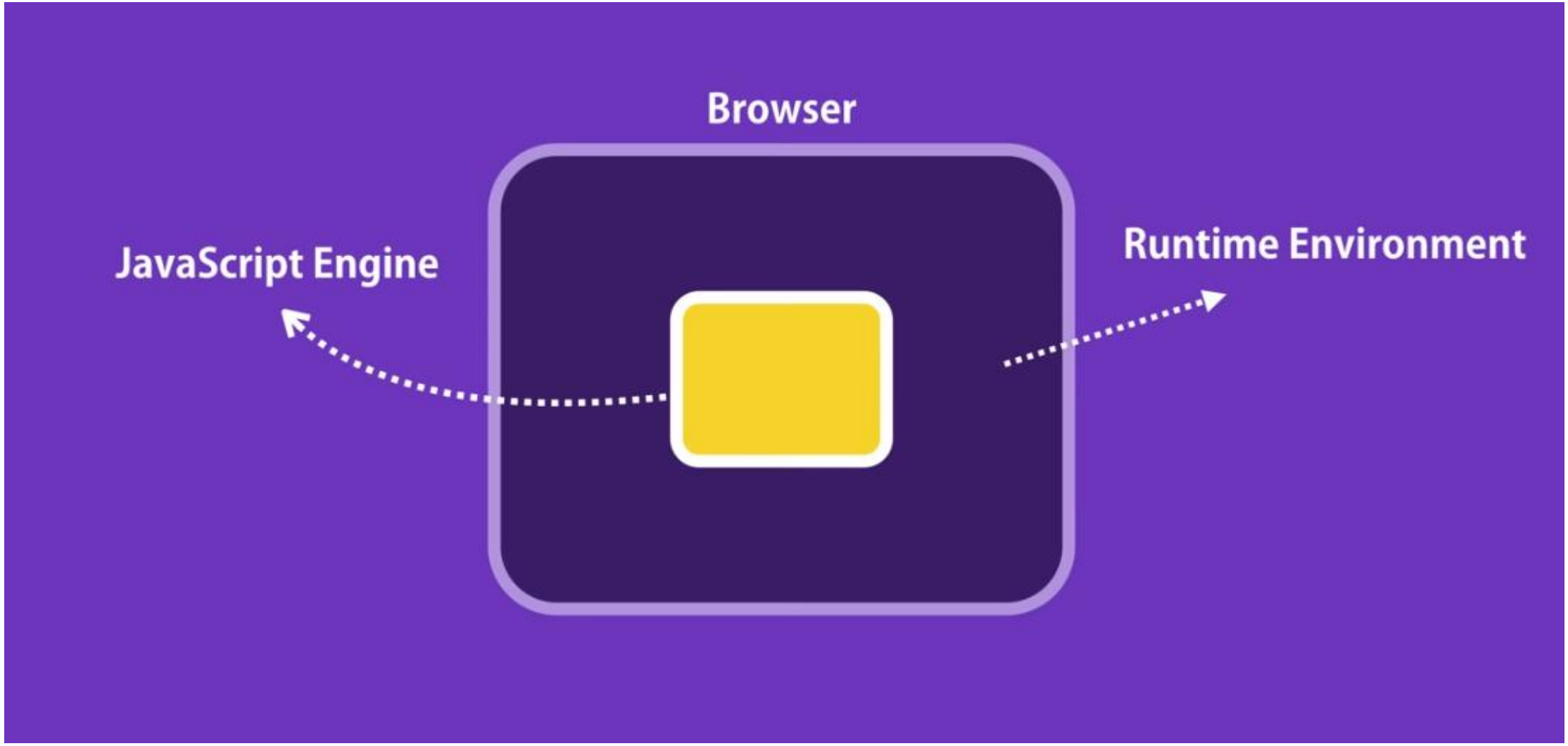
A **runtime environment** for executing JavaScript code

Cross platform… *Open source…* *Outside of a browser…*

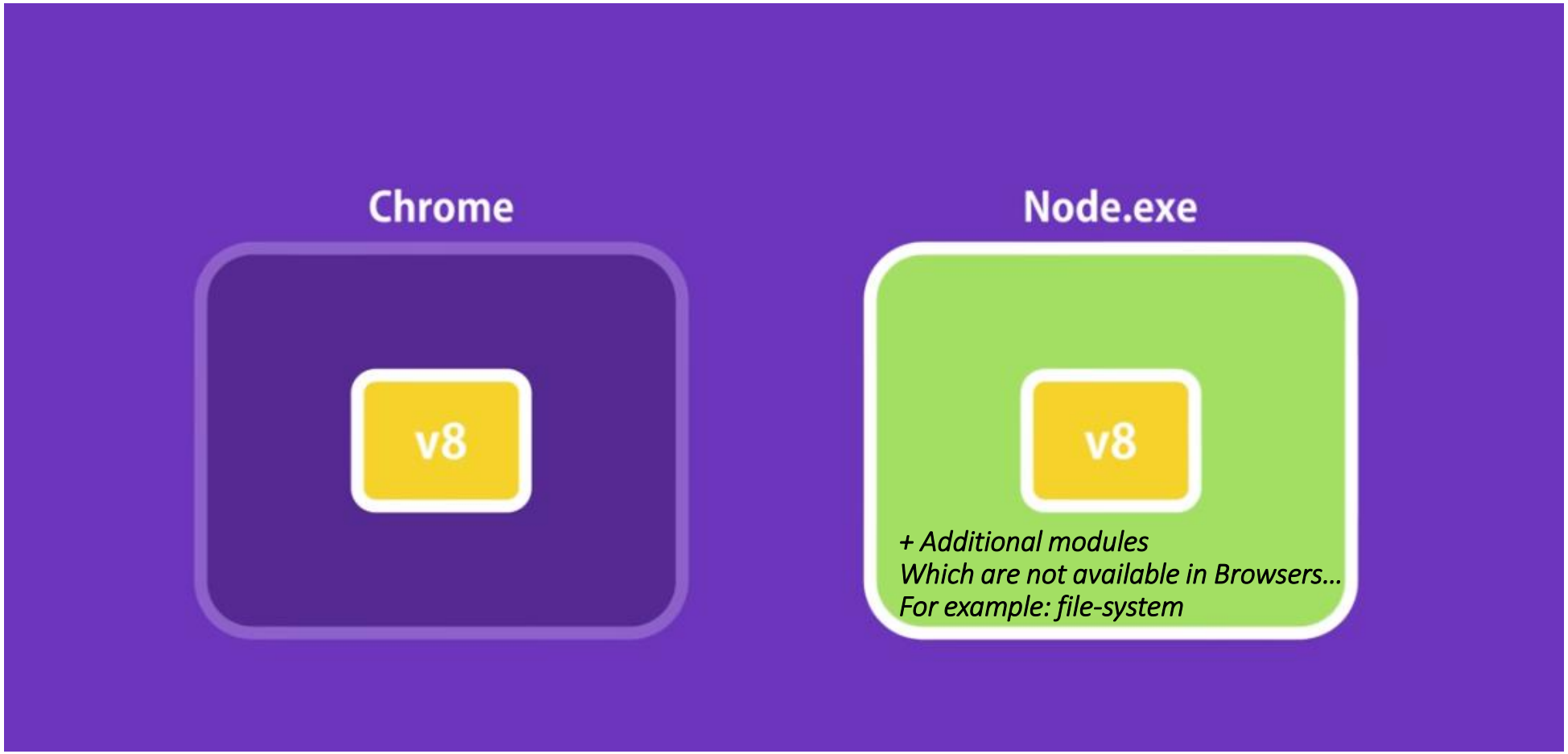
What is runtime environment?



Browser provide runtime env. For JS code



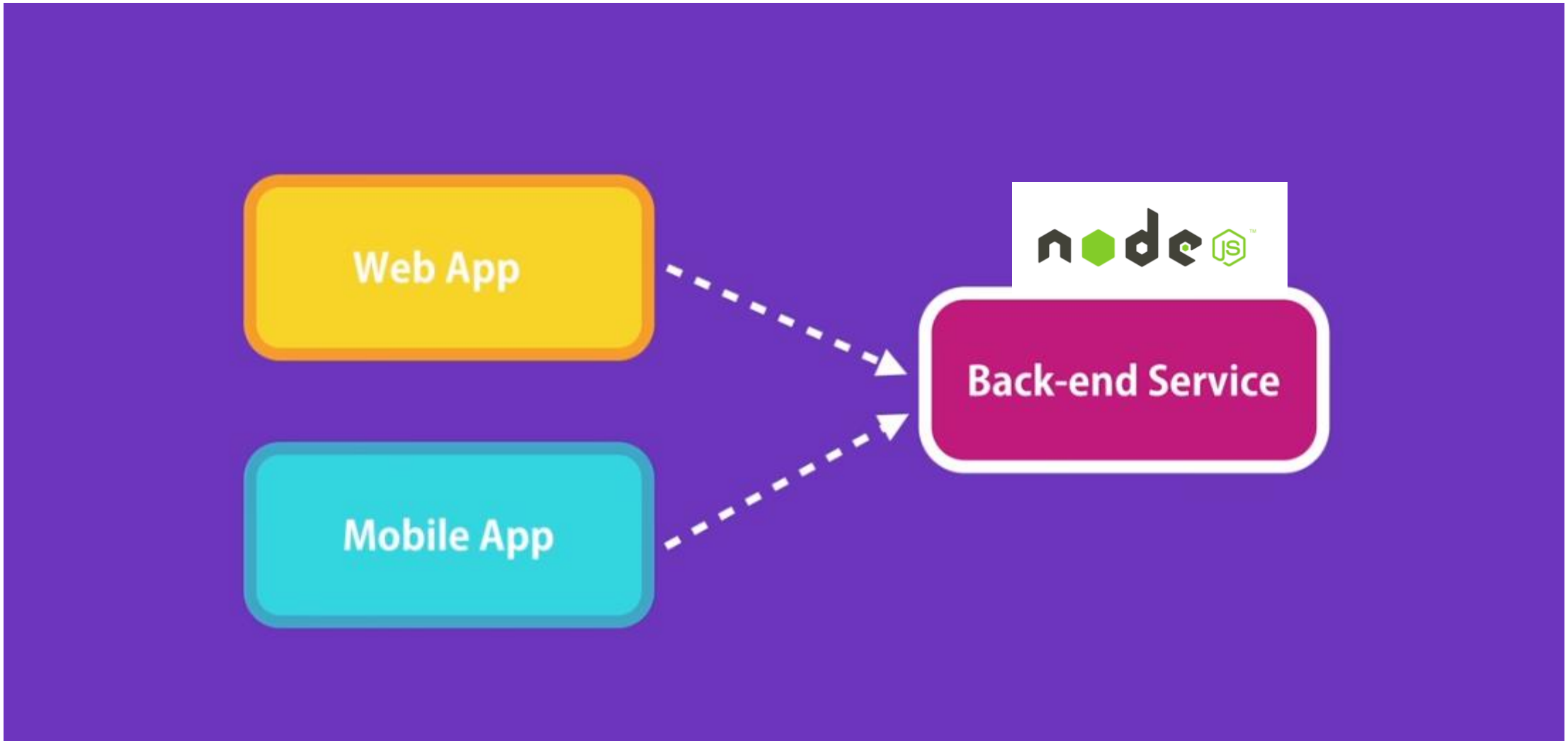
Node also provide runtime env. For JS code



We often use Node to build back-end services

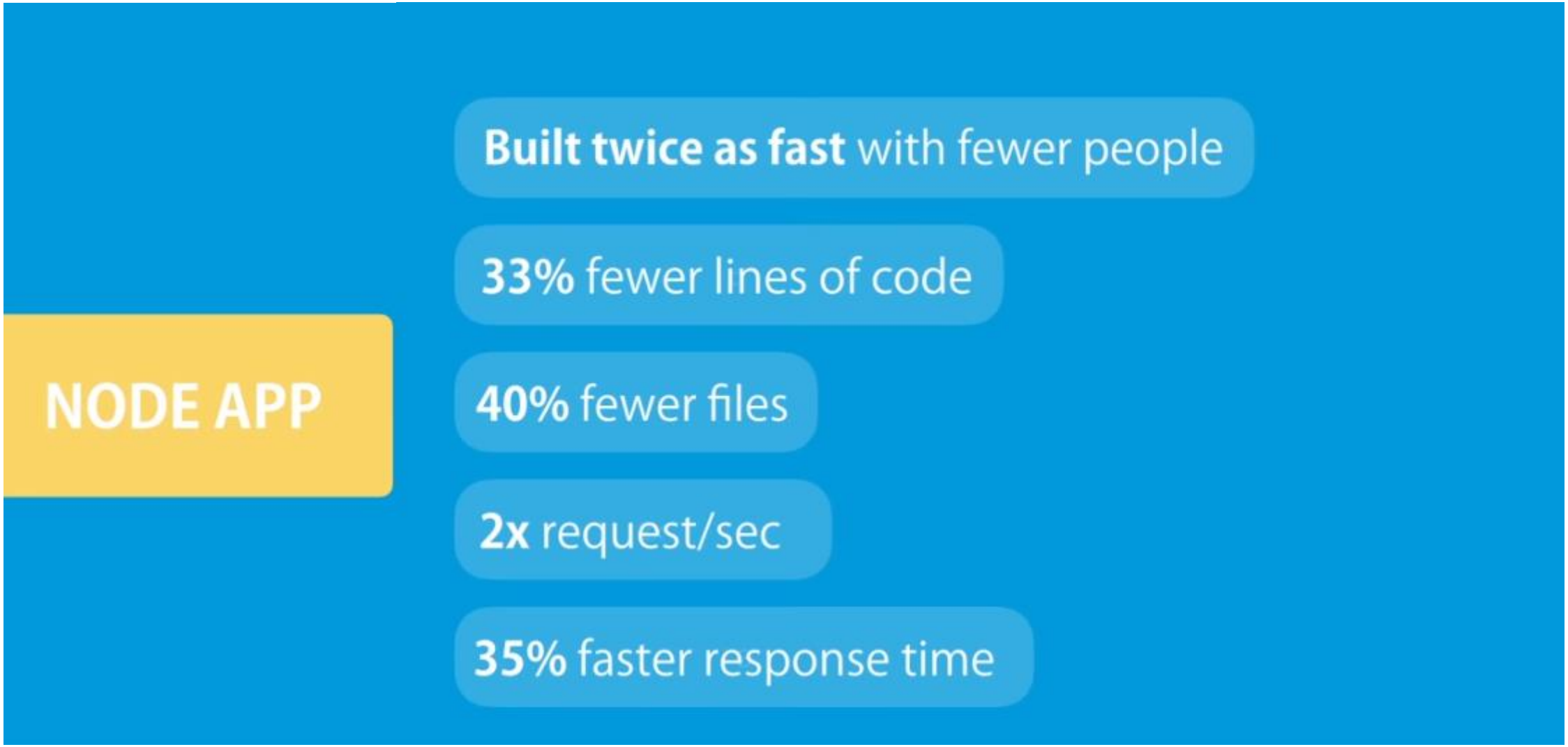
Web hosting …

REST services…


Web App
Mobile App
node
Back-end Service

Ideal for:

Highly-scalable, data-intensive and real-time apps


PayPal
node JS
VS
spring by Pivotal
NODE APP
Built twice as fast with fewer people
33% fewer lines of code
40% fewer files
2x request/sec
35% faster response time

Great for prototyping and agile development

Easy to get start...

Superfast and highly scalable

Used by PayPal, Uber, Netflix...

JavaScript everywhere

Backend & Frontend JS

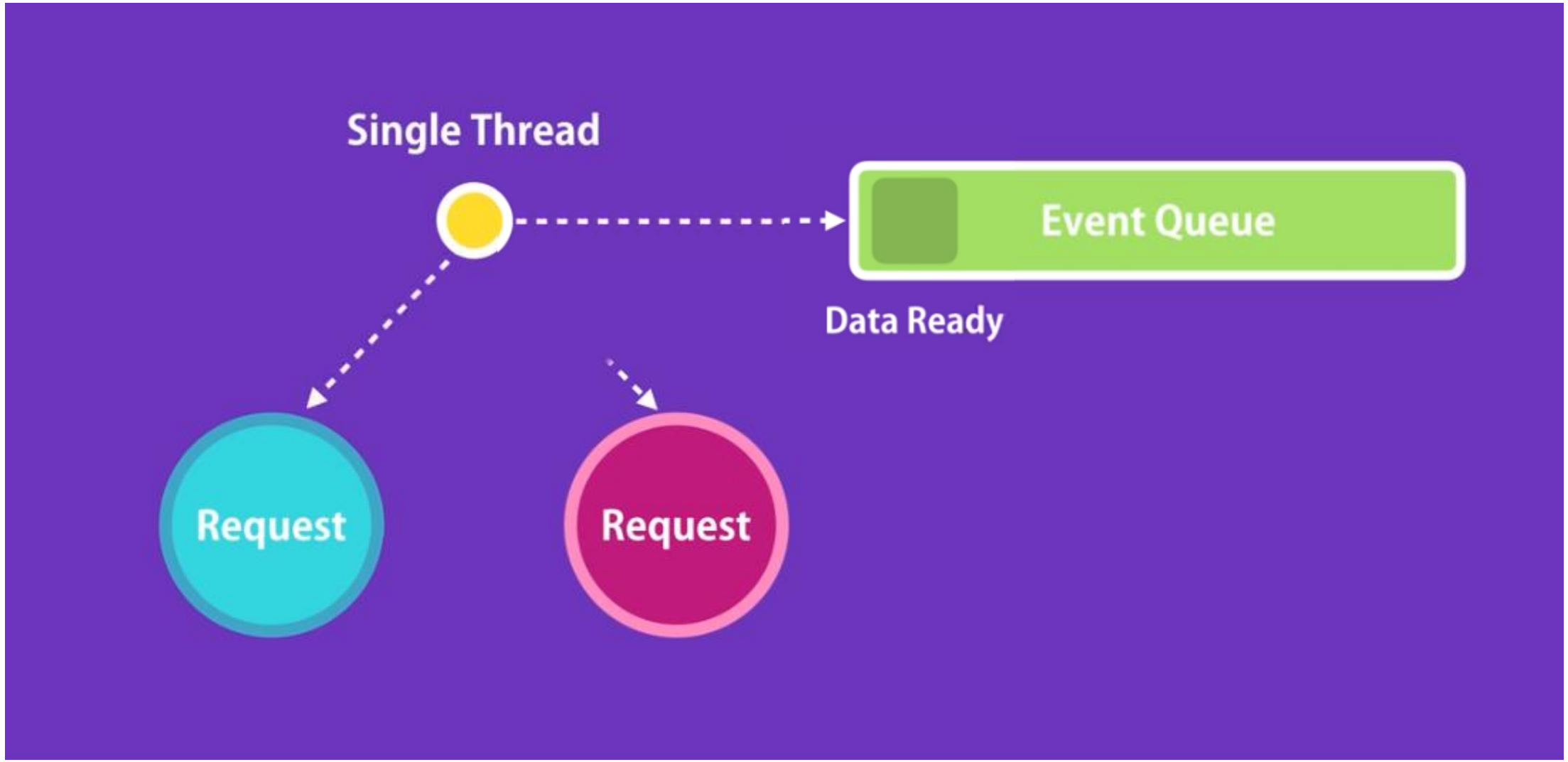
Cleaner and more consistent codebase

Large ecosystem of open-source libs

Free open source library for almost everything

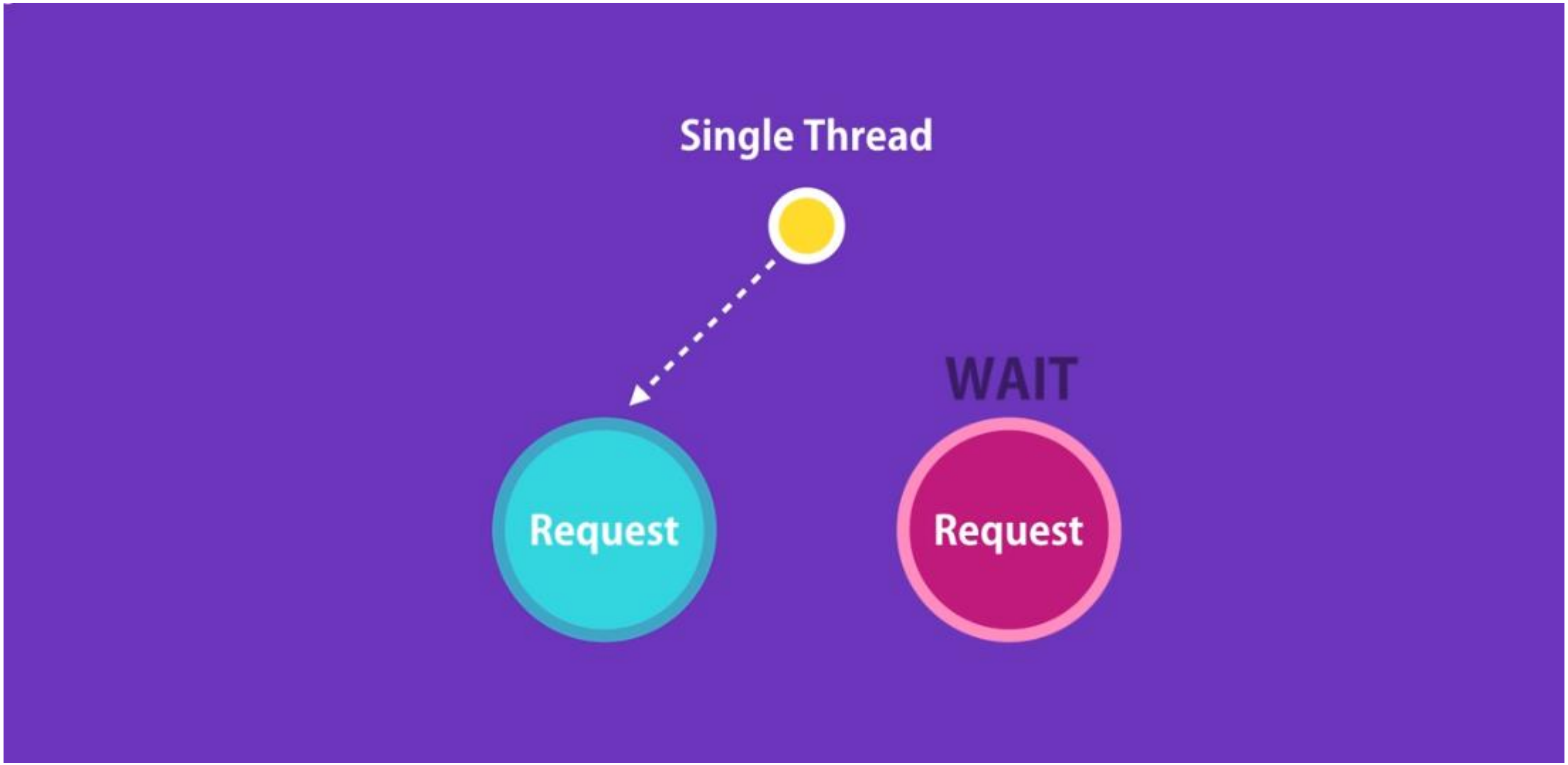
Node applications are

asynchronous by default


Single Thread
Event Queue
Data Ready
Request
Request

Node is ideal for

I/O-intensive apps


Single Thread
WAIT
Request
Request

Do **not** use Node for

CPU-intensive apps

Software Package Manager

The name **npm** (Node Package Manager) stems from when npm first was created as a package manager for Node.js.

All **npm** packages are defined in files called **package.json**.

The content of package.json must be written in **JSON**.

At least two fields must be present in the definition file: **name** and **version**.

Example

```
{
"name" : "foo",
"version" : "1.2.3",
"description" : "A package for fooing things",
"main" : "foo.js",
"keywords" : ["foo", "fool", "foolish"],
"author" : "John Doe",
"licence" : "ISC"
}
```

Managing Dependencies

npm can manage **dependencies**.

npm can (in one command line) install all the dependencies of a project.

Dependencies are also defined in **package.json**.

Software Package Manager

The name **npm** (Node Package Manager) stems from when npm first was created as a package manager for Node.js.

All **npm** packages are defined in files called **package.json**.

The content of package.json must be written in **JSON**.

At least two fields must be present in the definition file: **name** and **version**.

Example

```
{
"name" : "foo",
"version" : "1.2.3",
"description" : "A package for fooing things",
"main" : "foo.js",
"keywords" : ["foo", "fool", "foolish"],
"author" : "John Doe",
"licence" : "ISC"
}
```

Managing Dependencies

npm can manage **dependencies**.

npm can (in one command line) install all the dependencies of a project.

Dependencies are also defined in **package.json**.

Sharing Your Software

If you want to share your own software in the **npm registry**, you can sign in at:

https://www.npmjs.com

Publishing a Package

You can publish **any directory** from your computer as long as the directory has a **package.json file**.

Check if npm is installed:

```
C:\>npm
```

Check if you are logged in:

```
C:\>npm whoami
```

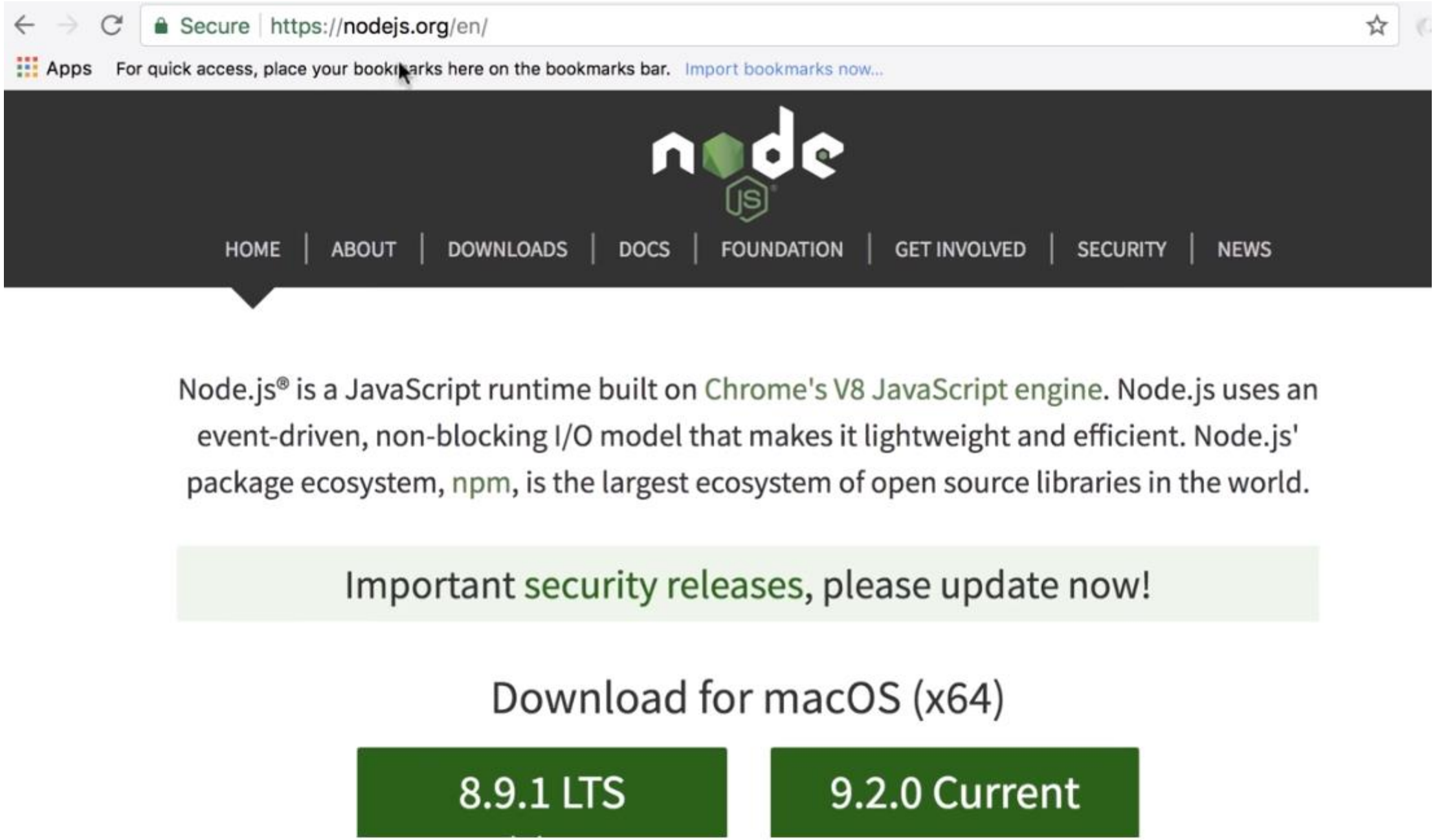
If not, log in:

```
C:\>npm login
Username: <your username>
Password: <your password>
```

Navigate to your project and publish your project:

```
C:\Users\myuser>cd myproject
C:\Users\myuser\myproject>npm publish
```

Let’s install Node and write some code!


Secure
https://nodejs.org/en/
Apps
For quick access, place your bookmarks here on the bookmarks bar.
Import bookmarks now...
node
JS
HOME
ABOUT
DOWNLOADS
DOCS
FOUNDATION
GET INVOLVED
SECURITY
NEWS
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.
Important security releases, please update now!
Download for macOS (x64)
8.9.1 LTS
9.2.0 Current

Visual Studio Code - Code Editin

https://code.visualstudio.com

Visual Studio Code | Docs | Updates | Blog | API | Extensions | FAQ | Search Docs | Download

Version 1.33 is now available! Read about the new features and fixes from March.

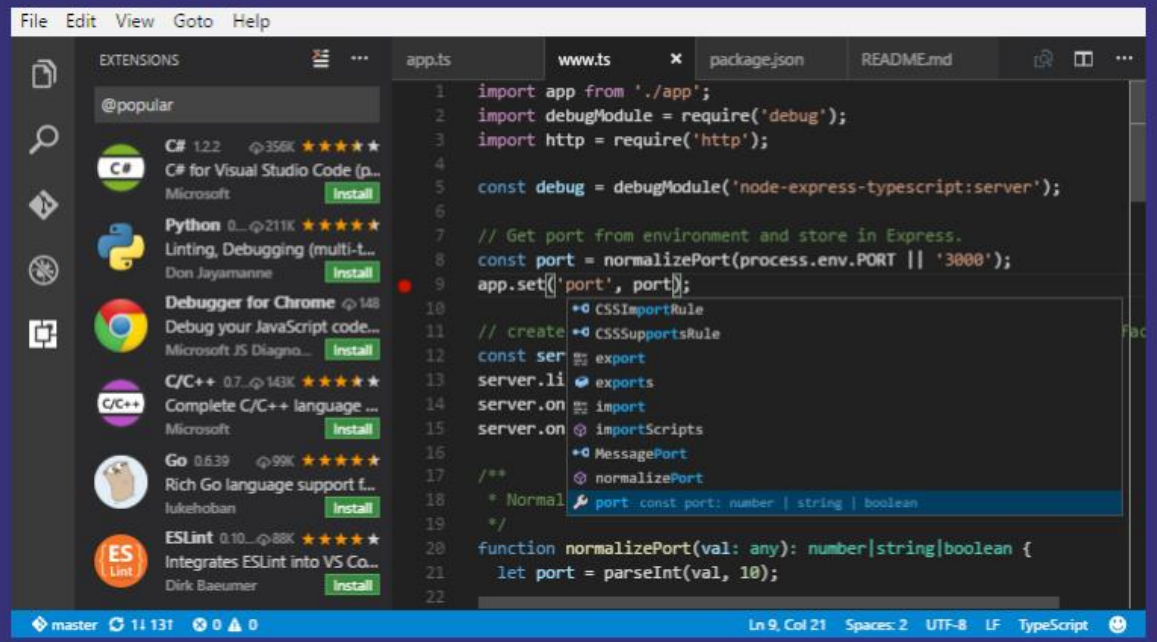
Code editing. Redefined.

Free. Open source. Runs everywhere.

Download for Windows
Stable Build

Other platforms and Insiders Edition

By using VS Code, you agree to its license and privacy statement.



IntelliSense

Debugging

Built-in Git

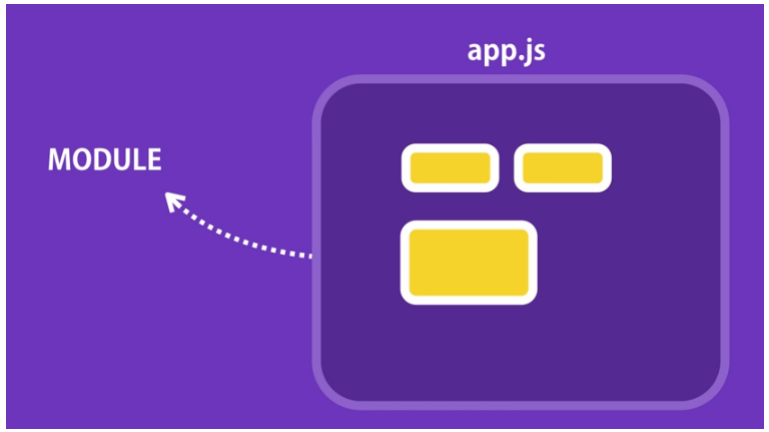
Extensions

Few concepts:

- Console
- Window
- Global

Modules

- The variables and functions defined in that modules are scoped in this file
  - Variables/function are private to the module
  - You can export (make public)/import modules
- Every node application has at least one file (or one module) which is called main module
- Every file is a module. Every variable/function defined in this files are scoped to this module



Models:

- **Create (export)**
- **Load (require)**
- **Jshint**
- **Module Wrapper function**

```
Module {
  id: '.',
  exports: {},
  parent: null,
  filename: 'E:\\work\\bottomline\\18.04\\nodejs\\start\\app.js',
  loaded: false,
  children: [],
  paths:
   [ 'E:\\work\\bottomline\\18.04\\nodejs\\start\\node_modules',
     'E:\\work\\bottomline\\18.04\\nodejs\\node_modules',
     'E:\\work\\bottomline\\18.04\\node_modules',
     'E:\\work\\bottomline\\node_modules',
     'E:\\work\\node_modules',
     'E:\\node_modules' ] }
```

Node.js – built in modules & objects

- Node.js built-in models and objects:
  - https://www.w3schools.com/nodejs/ref_modules.asp
  - https://nodejs.org/docs/latest-v9.x/api/
- Open path > parse

Node.js – built in modules & objects

- Node.js built-in models and objects:
  - https://www.w3schools.com/nodejs/ref_modules.asp
  - https://nodejs.org/docs/latest-v9.x/api/
- Open path > parse
- Open OS … i.e. freemem
- Open FileSystem
- Open Events > EventEmitter
- Open HTTP > Server

Install Express using NPM

- NPM package page
  - ❖ https://www.npmjs.com/
- Type Express in the search bar
- Follow the instructions:
  - npm install express
  - copy the example demo code

What Is NPM?

- ✓ **Node Package Manager**
- ✓ Pre-installed with Node.js
- ✓ Easily install modules/packages on your system
- ✓ Modules are basically JavaScript libraries
- ✓ Makes it easy for developers to share & reuse code

npm is the world's largest **Software Library** (Registry)

npm is also a software **Package Manager** and **Installer**

The World's Largest Software Registry (Library)

npm is the world's largest **Software Registry**.

The registry contains over 800,000 **code packages**.

Open-source developers use **npm** to **share** software.

Many organizations also use npm to manage private development.

Using npm is Free

npm is free to use.

You can download all npm public software packages without any registration or logon.

Command Line Client

npm includes a **CLI** (Command Line Client) that can be used to download and install software:

Windows Example

```
C:\>npm install <package>
```

```
npm install angular
```

```
$ npm install -g vue-cli
```

```
npm install --save-dev webpack
```

```
$ npm install bootstrap@4.0.0-alpha.6
```

NPM

Package.json File

- ✓ Manifest file with app info
- ✓ Lists dependencies (name & version)
- ✓ Specify if versions should be updated
- ✓ Create NPM scripts
- ✓ Easily create with **“npm init”**

NPM

- ❑ npm init
- ❑ npm init --yes
- ❑ npm set init-author-name “itay”
- ❑ npm get init-author-name
- ❑ npm config delete init-author-name
- ❑ explore package.json

npm-package-lock.json

A manifestation of the manifest

DESCRIPTION

`package-lock.json` is automatically generated for any operations where npm modifies either the `node_modules` tree, or `package.json`. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

This file is intended to be committed into source repositories, and serves various purposes:

- Describe a single representation of a dependency tree such that teammates, deployments, and continuous integration are guaranteed to install exactly the same dependencies.
- Provide a facility for users to "time-travel" to previous states of `node_modules` without having to commit the directory itself.
- To facilitate greater visibility of tree changes through readable source control diffs.
- And optimize the installation process by allowing npm to skip repeated metadata resolutions for previously-installed packages.

One key detail about `package-lock.json` is that it cannot be published, and it will be ignored if found in any place other than the toplevel package. It shares a format with npm-shrinkwrap.json, which is essentially the same file, but allows publication. This is not recommended unless deploying a CLI tool or otherwise using the publication process for producing production packages.

If both `package-lock.json` and `npm-shrinkwrap.json` are present in the root of a package, `package-lock.json` will be completely ignored.

npm --save

- Installs into folder
- Saves the dependency into package.json
- So… we can only copy package.json and install it later using npm install
- Try: **npm install lodash --save**
- explore package.json

npm --save

- Installs into folder
- Saves the dependency into package.json
- So… we can only copy package.json and install it later using npm install
- Try: **npm install lodash --save**
- explore package.json

npm install

- ❑ Delete node_modules
- ❑ npm install
- ❑ node app.js
- ❑ It works! thanks to:

```
"dependencies": {
        "express": "^4.16.4",
        "lodash": "^4.17.11"
}
```

npm install --save-dev

- ❑ npm install gulp gulp-sass --save-dev
- ❑ explore node_modules its huge! Due to gulp dependancies
- ❑ explore package.json

```
"devDependencies": {
        "gulp": "^4.0.0",
        "gulp-sass": "^4.0.2"
}
```

- ❑copy app.js + package.json to a new folder,
- ❑ npm install --production

    “**devDependencies**” were not installed

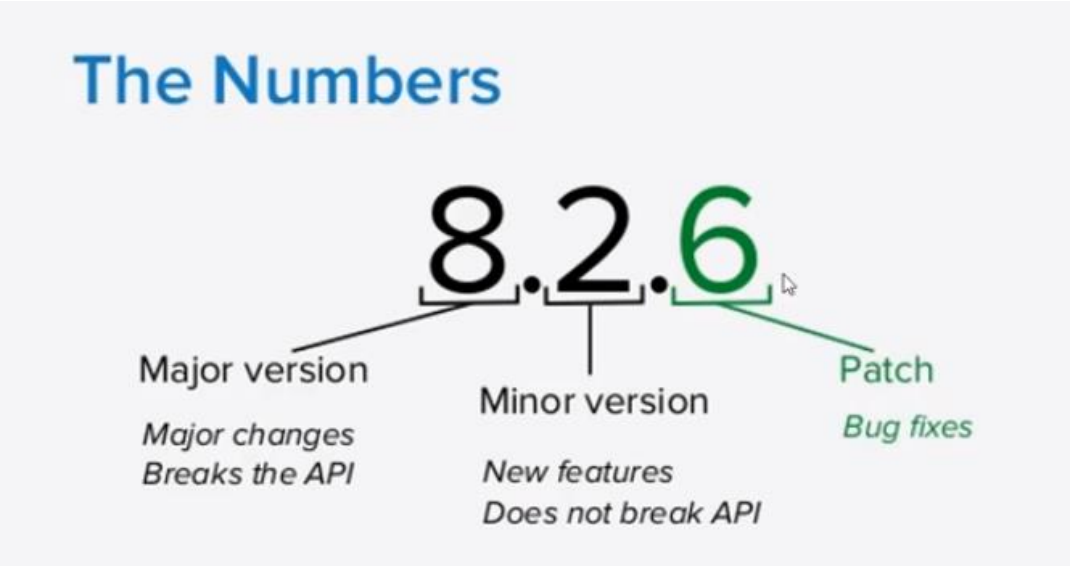
npm uninstall

- ❑ npm uninstall gulp gulp-sass --save-dev
- ❑ explore node_modules
- ❑ explore package.json

  gone -- "devDependencies": {
  "gulp": "^4.0.0",
  "gulp-sass": "^4.0.2"
  }
- ❑ npm rm lodash --save

npm install version

- ❑ npm install lodash@4.17.3 --save
- ❑ explore package.json
- ❑ npm update lodash –save
- ❑ explore package.json



npm prefix ^

- explore package.json

```
"dependencies": {
        "express": "^4.16.4",
        "lodash": "^4.17.11"
}
```

- What does ^ mean?
  - What does ^4.17.11?
  - we push it into a repo and someone runs: npm install
  - It will install the latest minor version (17), i.e: 4.18.1
  - If there is version 5.1.1 it will still take 4.X.X

npm prefix ~

- What does ~ mean?
  - What does ~4.17.11?
  - we push it into a repo and someone runs: npm install
  - It will install the latest patch version (11), i.e: 4.17.25
  - If there is version 4.18.1 it will still take 4.17.X

npm no-prefix

- What does no prefix mean?
  - What does 4.17.11?
  - we push it into a repo and someone runs: npm install
  - It will install the exact version 4.17.11

npm *

- What does * prefix mean?
  - What does “*” mean?
  - we push it into a repo and someone runs: npm install
  - It will install the latest version
  - Not a good idea since api may change in major version

npm best recommended

- “module” : “^X.X.X”
  most recommended!

npm global modules

- ❑ npm install –g nodemon
- ❑ nodemon continuously watch your application. Every time you save it will start it
- ❑ Where is it installed?... npm root –g
  - explore this folder
- ❑ nodemon
- ❑ Now modify app.js and save it
- ❑ Good with Express server (loads the server)

npm global modules

- ❑ npm install –g live-server
- ❑ create index.html
- ❑ live-server

More commands

- ❑ npm remove –g live-server
- ❑ npm list
- ❑ npm list –depth 0 (shows into the depth level)

npm scripts

- explore package.json

```
"scripts": {
        "start": "node app.js"
}
```

- npm start
- why?

1. skip the search for the js file
2. deploy to cloud platform (i.e. Heroku) .
   The server will execute the start
3. starting server locally (i.e. live-server)
   add this script: "server " : "live-server"
   now from terminal: *npm run server*