ITSE412– Week 4

Node.JS Introduction

What is node.js?

"Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. "

http://nodejs.org/

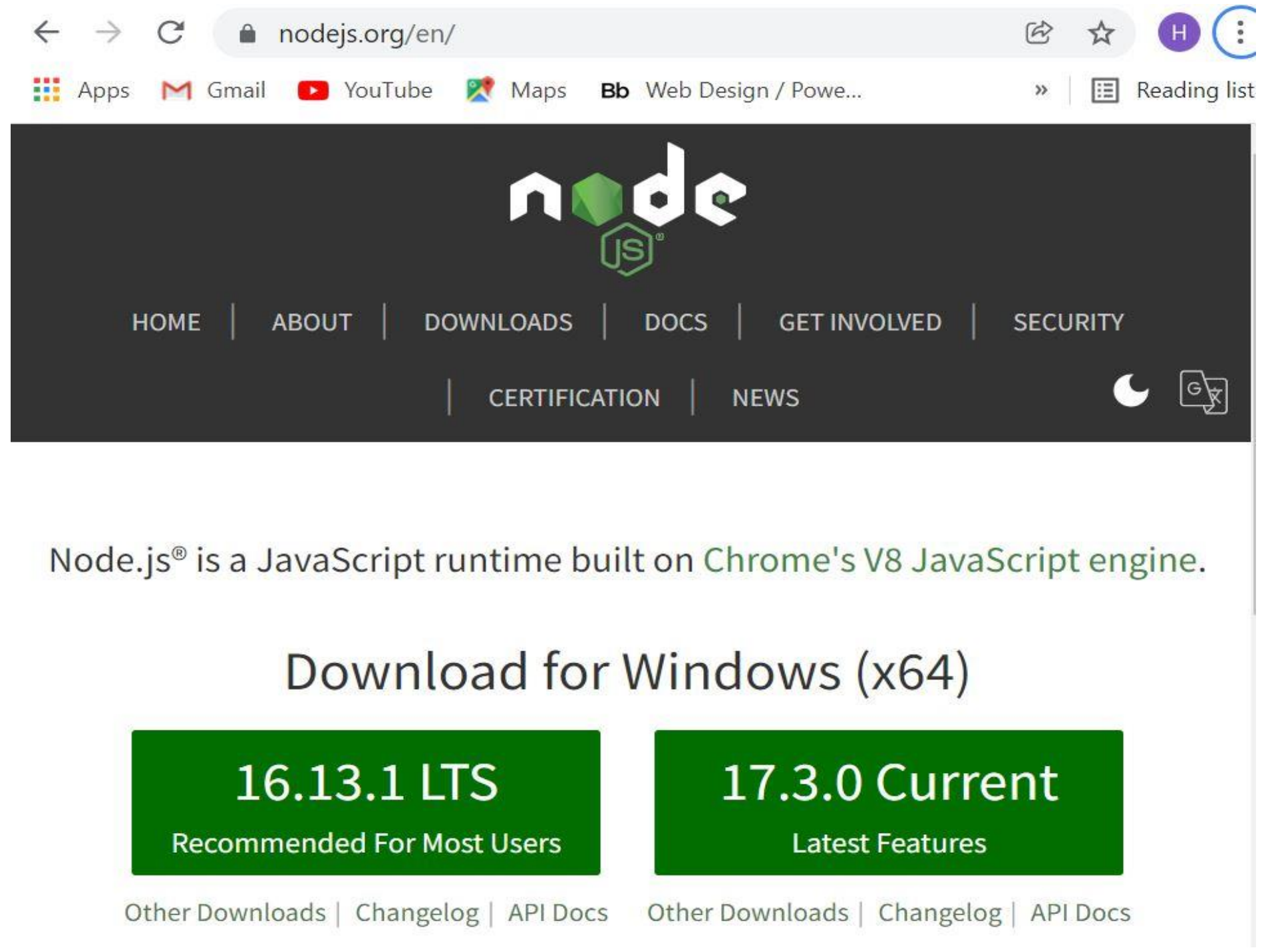
- A platform not a language.
- It's NOT a web framework.
- Built in C++ on top of Chrome's v8 Engine
- Used for building fast, scalable network applications.
- Can handle thousands of Concurrent connections with Minimal overhead (cpu/memory) on a single process
- IT IS A RUNTIME ENVIRONEMNT TO RUN JAVASCRIPT CODE OUTSIDE THE BROWSER (SERVER SIDE).

Node.js main goal

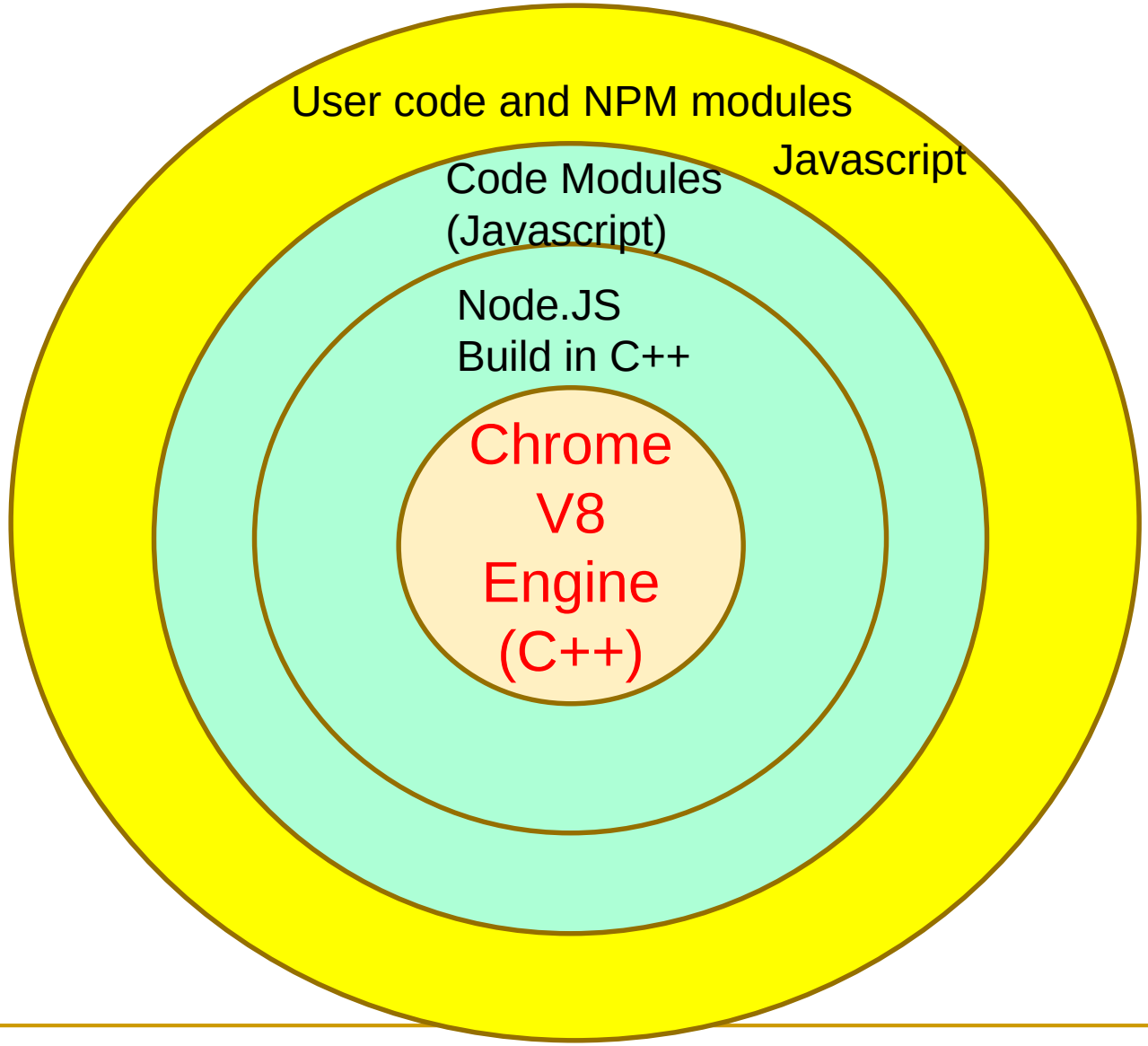
"Node's goal is to provide an easy way to build scalable Network and Web applications"

About Node.js……

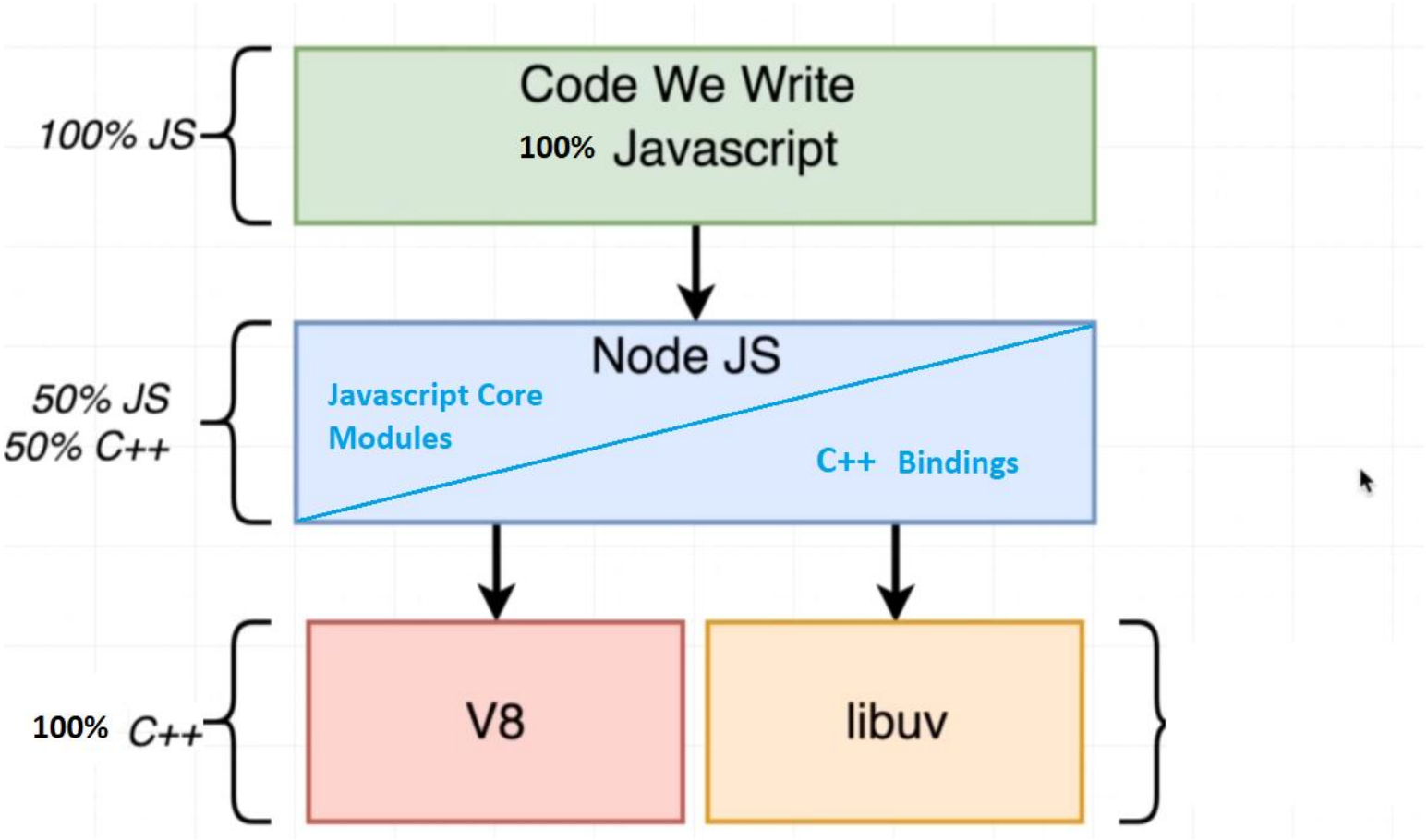
- Created in 2009 by Ryan Dahl, a software engineer working at Google
- Development && maintenance sponsored by Joyent (www.joyent.com/)
- Licence MIT
- Last release : 16.3.1
- Based on Google V8 Engine
- 1.3+ M packages


nodejs.org/en/
Apps
Gmail
YouTube
Maps
Web Design / Powe...
Reading list
HOME
ABOUT
DOWNLOADS
DOCS
GET INVOLVED
SECURITY
CERTIFICATION
NEWS
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
Download for Windows (x64)
16.13.1 LTS
Recommended For Most Users
17.3.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

Node.js Architecture



NodeJS Architecture



Used by…..

Rails to Node
« Servers were cut to 3 from 30 »
« Running up to 20x faster in some scenarios »
« Frontend and backend mobile teams could be combined [...] »

Java to Node
« Built almost twice as fast with fewer people »
« Double the requests per second »
« 35% decrease in the average response time »

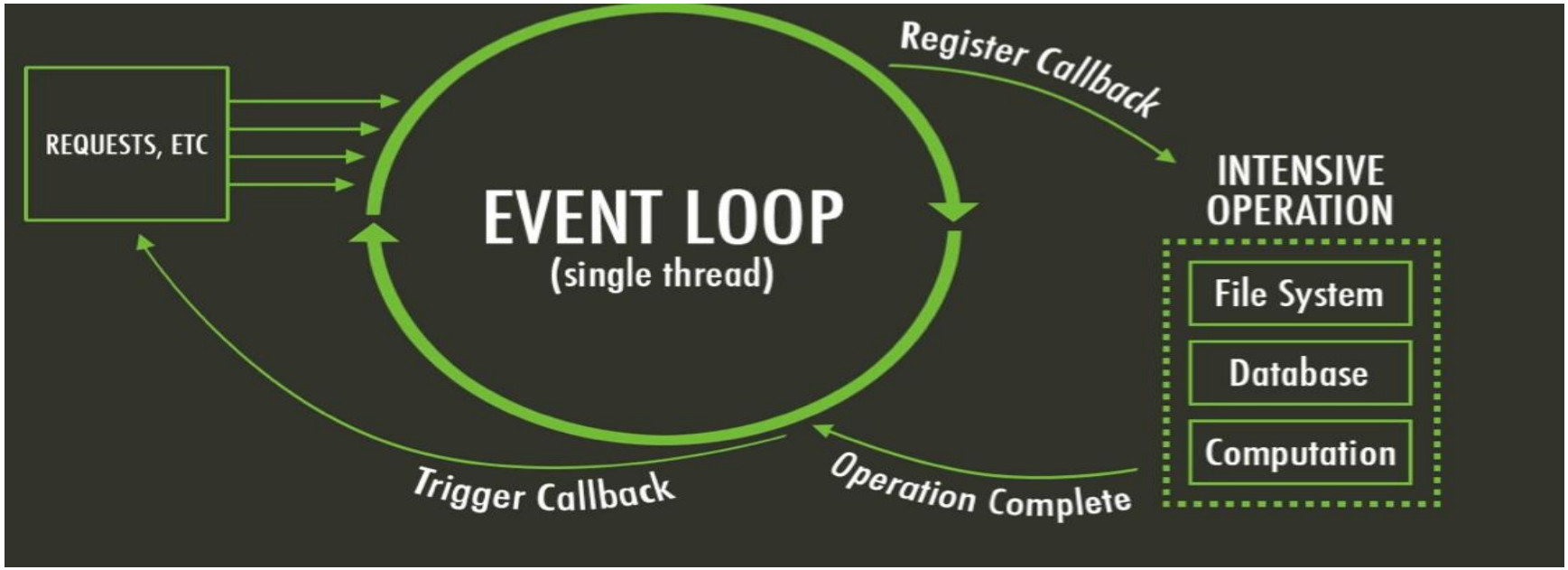
Why use node.js ?

- Non Blocking I/O
- Huge library (>1M modules)
- Easy to write and use modules
- Uses Single Thread with Event Loop
- Fast and reliable (based on V8 Javascript Engine)
- One Language for Frontend and Backend
- Cross platform (Windows, Linux, Mac)
- Active community

The idea behind node.js….

- Perform asynchronous processing on single thread instead of classical multithread processing, minimize overhead & latency, maximize scalability
- Ideal for applications that serve big number of requests, e.g. web apps
- Not so ideal for heavy calculations, e.g. massive parallel computing

Node.js Event Loop

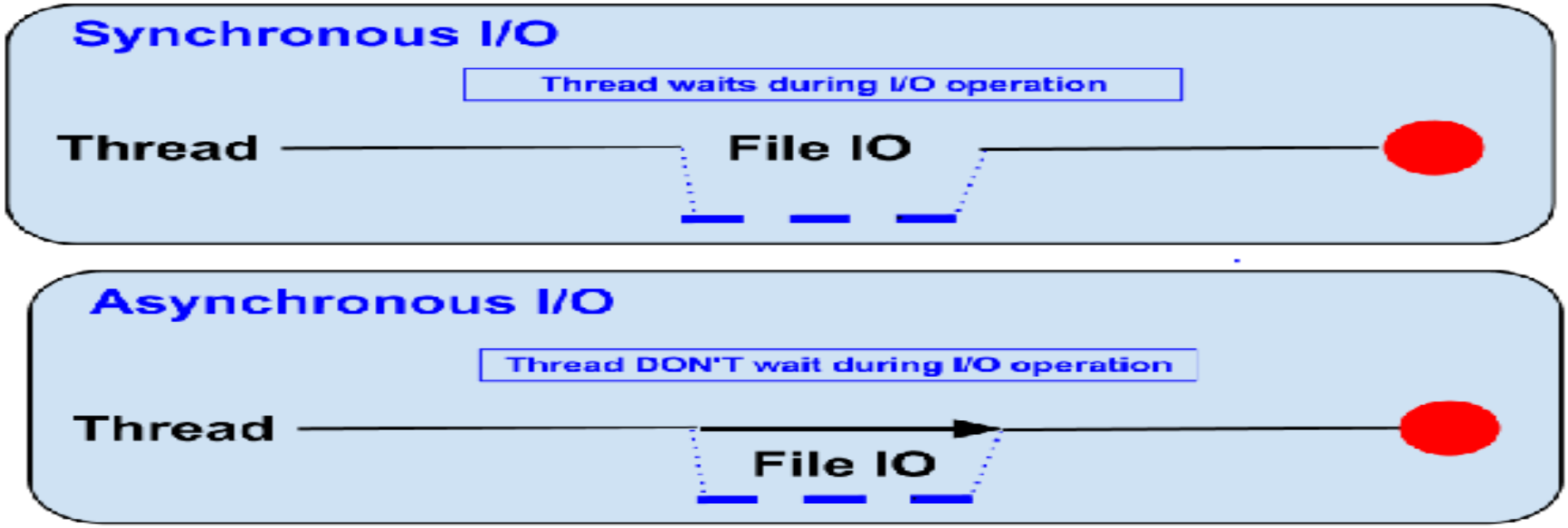


This is a very simple and basic model. You must:

1. Avoid synchronous code because it blocks the event loop
2. Use: callbacks, callbacks, and more callbacks

Blocking vs Non-Blocking……

Example :: Read data from file and show data



Blocking…..

1. Read data from file
2. Show data
3. Do other tasks

Ex: blocking.js

```
const fs = require('fs');
var data = fs.readFileSync("test.txt" );
console.log( data );
console.log( "Do other tasks" );
```

Non-Blocking……

- Read data from file
- When reading data completed, show data
- Do other tasks


```
const fs = require('fs');
fs.readFile( "test.txt", function( err,
data ) {
console.log(data);
});
console.log( "Do other tasks" );
```

Application areas include

- Basic web Applications (MEAN framework)
- Real-time Applications (Chat-Messaging – Vedio streaming – e-commerce…..)
- Scalable API Proxies
- High Concurrency Applications (payment systems, ….)
- Single-Page Applications (SPAs)

Node.js is good for….

- ✓ Web application
- ✓ Websocket server
- ✓ Proxy server
- ✓ Streaming server
- ✓ Fast file upload client
- ✓ Any Real-time data apps
- ✓ Anything with high I/O

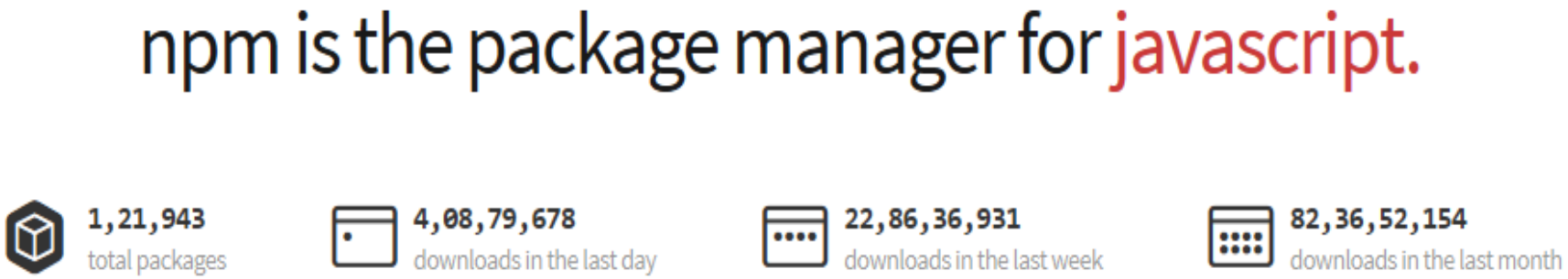
But not for applications that require intensive computing power

Getting Started…..

- http://nodejs.org/ and Download the installer
- Install it

Node.js comes with npm

- https://npmjs.org/
- # of modules = 1,21,943



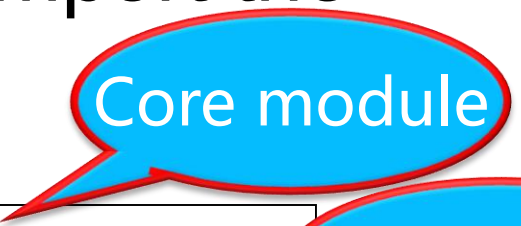
Install and use modules

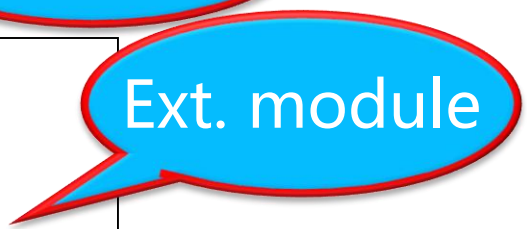
- First use npm to install the modeule:

  C:\proj1> npm install express

- Second use require() to import the modeule in node.js code:

```
s   const http = require('http');
    const fs = require('fs');
    const express = require('express');
```





Development for the Web….

- Web pages and Web apps use the following standard technologies:
  - HTML for markup
  - CSS for style
  - JavaScript for dynamic content
- Data is generated on the server, transferred to the client, and displayed by the browser
- Server Side technologies include PHP, JSP, ASP.net, Djanog, and now Node.js

Development for REST APIs……

- REST = REpresentational State Transfer
- Architectural style for client-server
- Node.js is greate for developing REST API's
- Easy to implement the CRUD model:
  - POST => Create
  - GET => Read
  - PUT => Update
  - DELETE => Delete