DAY 1

NODE.JS

Belkacem Alidra

June 25, 2018

nomades.ch

WHOAMI():

- dev@b-alidra.com
- 10 years of Web & Mobile development
- Working daily with Node.js since 2015

TODAY

- Node.js
  - The event loop
  - CLI and debugging
  - HTTP Server
  - Release schedule
- npm
  - package.json
- Express.js
- nodemon
- Let’s code

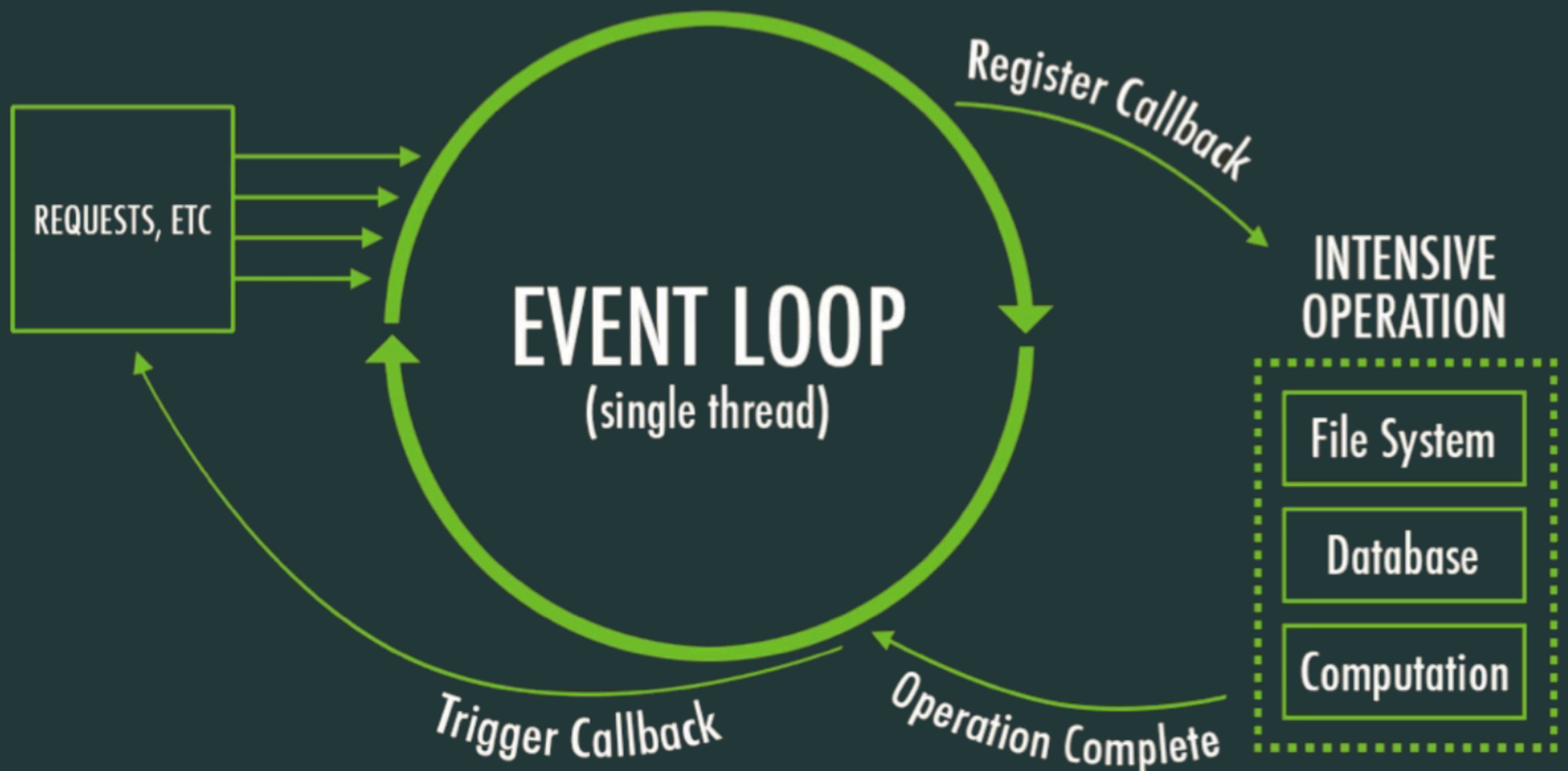
NODE.JS

WHAT IS NODE.JS ?

A server-side Javascript run-time environment

WHAT IS NODE.JS ?

- created in 2009 by Ryan Dahl
- is based on Google's V8 Javascript Engine
- has an event-driven architecture
- handles I/O asynchronously


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

THE EVENT LOOP

You already know how the event loop works in the browser:

```
// executed immediately by the main thread.
console.log("zero seconds");

// register a callback that will be executed once the operation is completed.
// In this case the "operation" is waiting two seconds.
setTimeout(() => console.log("two seconds."), 2000);

// the same as previously, although this operation will be completed faster.
// Thus, this callback will be executed before the previous one by the main
// thread.
setTimeout(() => console.log("one second."), 1000);

// As the first console.log(), this will be executed immediately by the main
// thread.
console.log("some seconds");
```

THE EVENT LOOP

Node.js script example

```
// always start with this for clarity and speed.
"use strict";

// this is how modules are "imported"
const fs = require("fs");

// console.log() works in Node.js too
console.log("sync ready");

// NOTE: readFileSync() I/O is executed in the main thread,
// could throw an Error.
let buf = fs.readFileSync("/etc/hosts");
// You can use console.log(buf.toString()) to output file content
console.log("done reading.");

console.log("async ready");

// NOTE: readFile() I/O is executed in a worker thread,
// could callback an Error.
fs.readFile("/etc/hosts", function callback(err, buf) {
    console.log("done async reading.");
});
console.log("done reading ?");
```

COMMAND LINE INTERFACE (CLI)

REPL

```
iMac-de-Belkacem:~ cas$ node
> console.log("Hello from the REPL");
Hello from the REPL
undefined
> w = "Welcome";
'Welcome'
> w
'Welcome'
> _
'Welcome'
> .editor
// Entering editor mode (^D to finish, ^C to cancel)
function welcome(name) {
  return `Hello ${name}!`;
}

welcome('Node.js User');

// ^D

'Hello Node.js User!'
> .exit
iMac-de-Belkacem:~ cas$
```

COMMAND LINE INTERFACE (CLI)

Debug

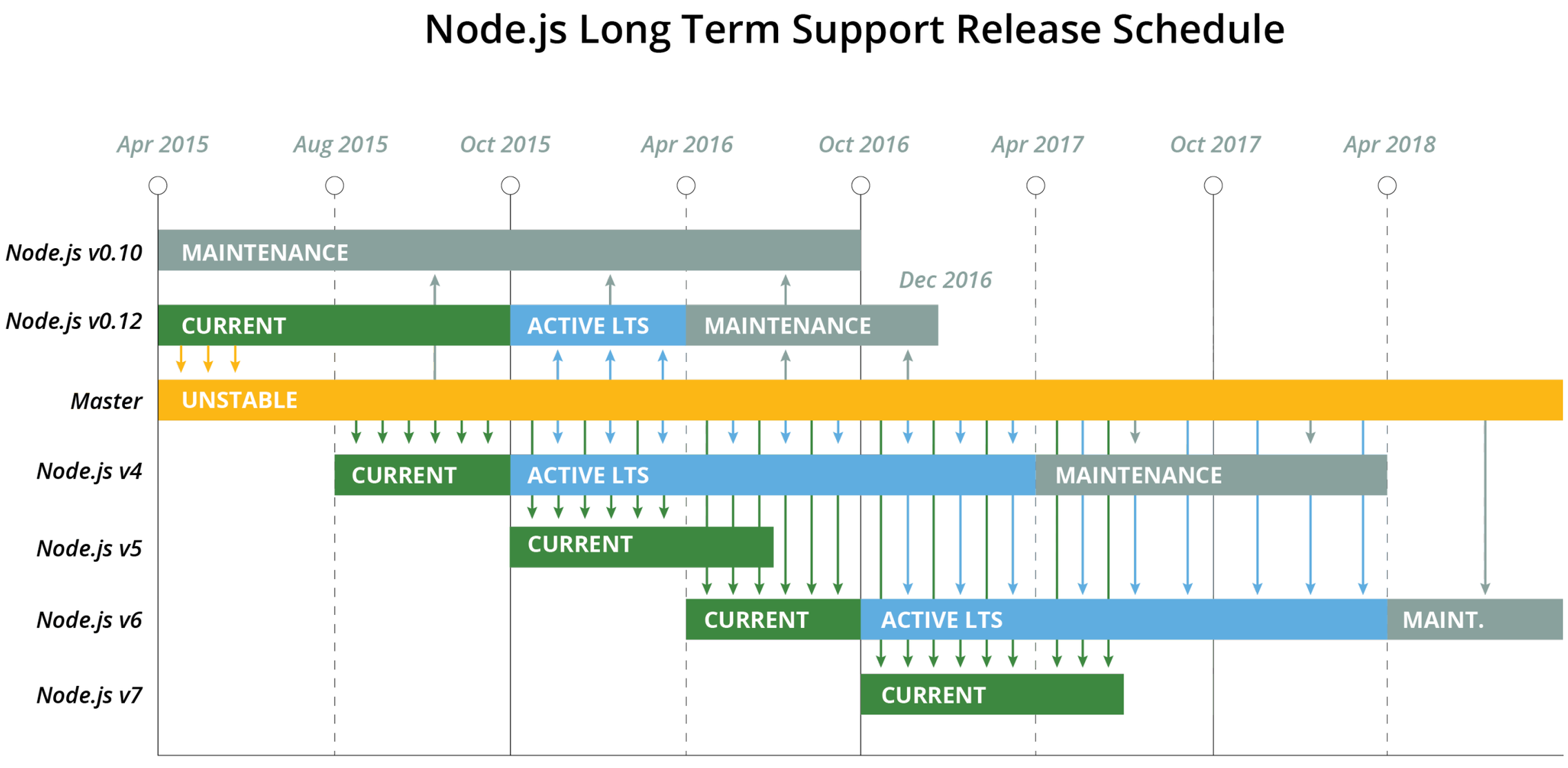
```
iMac-de-Belkacem:Day 1 cas$ node inspect event-loop.js
< Debugger listening on ws://127.0.0.1:9229/21165db4-7c48-493c-ab5e-4440acc3aeb7
< For help see https://nodejs.org/en/docs/inspector
< Debugger attached.
Break on start in event-loop.js:1
> 1 (function (exports, require, module, __filename, __dirname) { // always start with this for clarity and speed.
  2 "use strict";
  3
debug> n
break in event-loop.js:5
  3
  4 // this is how modules are "imported"
> 5 const fs = require("fs");
  6
  7 // console.log() works in Node.js too
debug> n
break in event-loop.js:8
  6
  7 // console.log() works in Node.js too
> 8 console.log("sync ready");
  9
 10 // NOTE: readFileSync() I/O is executed in the main thread,
debug> sb
[Function: setBreakpoint]
debug> c
< sync ready
< done reading.
< async ready
< done reading ?
< done async reading.
< Waiting for the debugger to disconnect...
debug>
```

HTTP SERVER

The classic Hello World

```
"use strict"; // always 2
// "http" and "util", like "fs", are standard Node.js module.
// see https://nodejs.org/dist/latest-v8.x/docs/api/
const http = require("http");
const util = require("util");

// the port on which our server will listen on.
const port = 8080;

// create a http server to handle requests.
const server = http.createServer((req, res) => {
    // util.inspect() is helpful for debugging. See also console.dir().
    console.log(util.inspect(req.socket.address(), {colors:true}));
    res.end(`Welcome to ${req.url}\n`); // ES6 Template literals
});

// listen() will register into the event loop.
server.listen(port, () => console.log("listening on port " + port));

// just like "done reading?" in our previous example, this will be printed
// *before* the server is listening.
console.log("EOF");
```

LTS RELEASE SCHEDULE



COPYRIGHT © 2015 NODESOURCE, LICENSED UNDER CC-BY 4.0

NPM

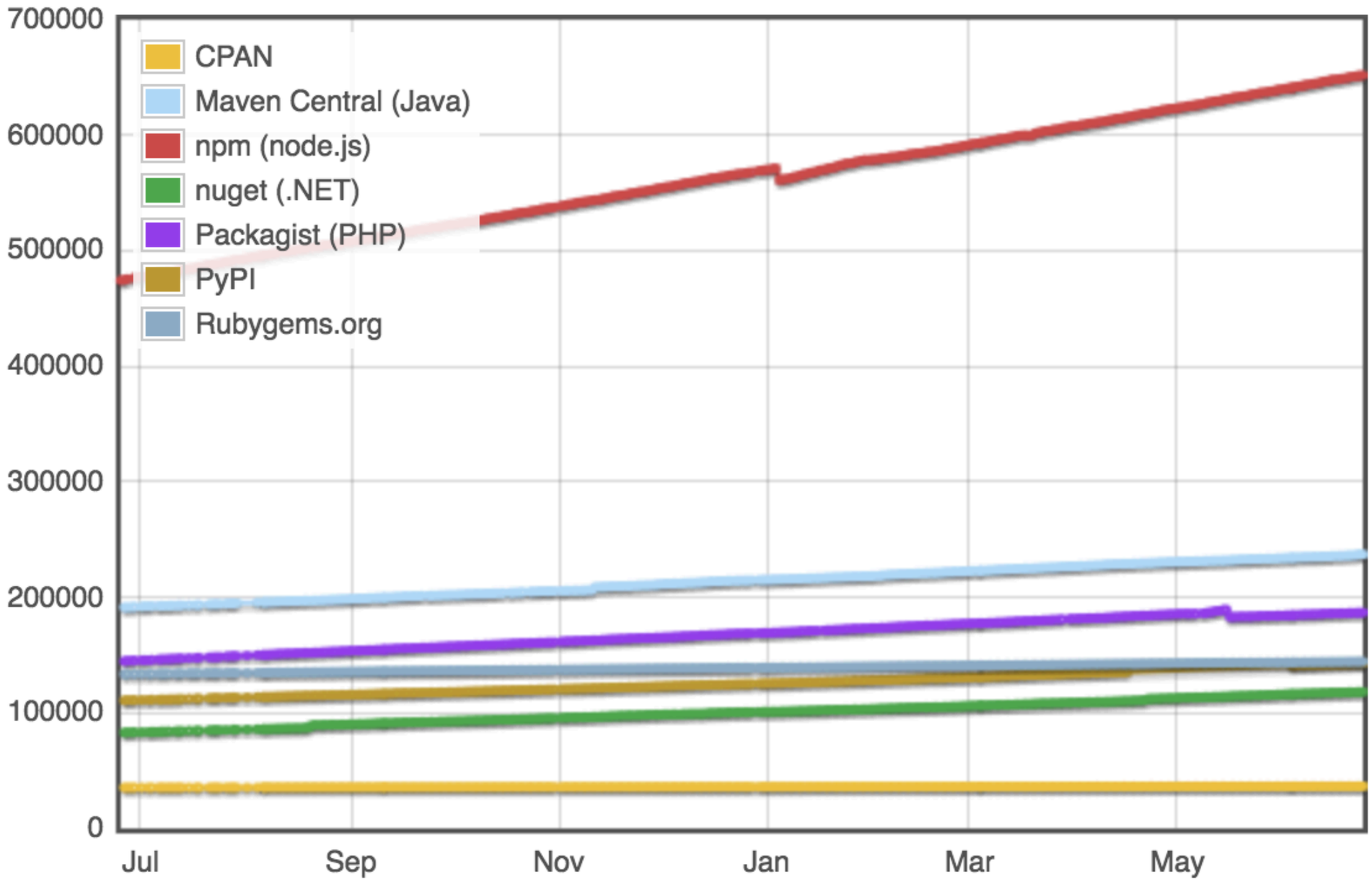
WHAT IS NPM ?

npm is the Node.js package manager.

Similar to Composer (PHP), Maven (Java) or Bower (Front JS).

It is a command line tool that is primarily used to interact with the *npm registry* (for example to install modules).

THE NPM REGISTRY



modulecounts.com

HOW TO USE NPM ?

% npm init

will help you to create the **package.json** file

% npm i[nstall] [-g] express

will install the “express” module from the registry into the node_modules/ directory. Once installed, you may *require* it:

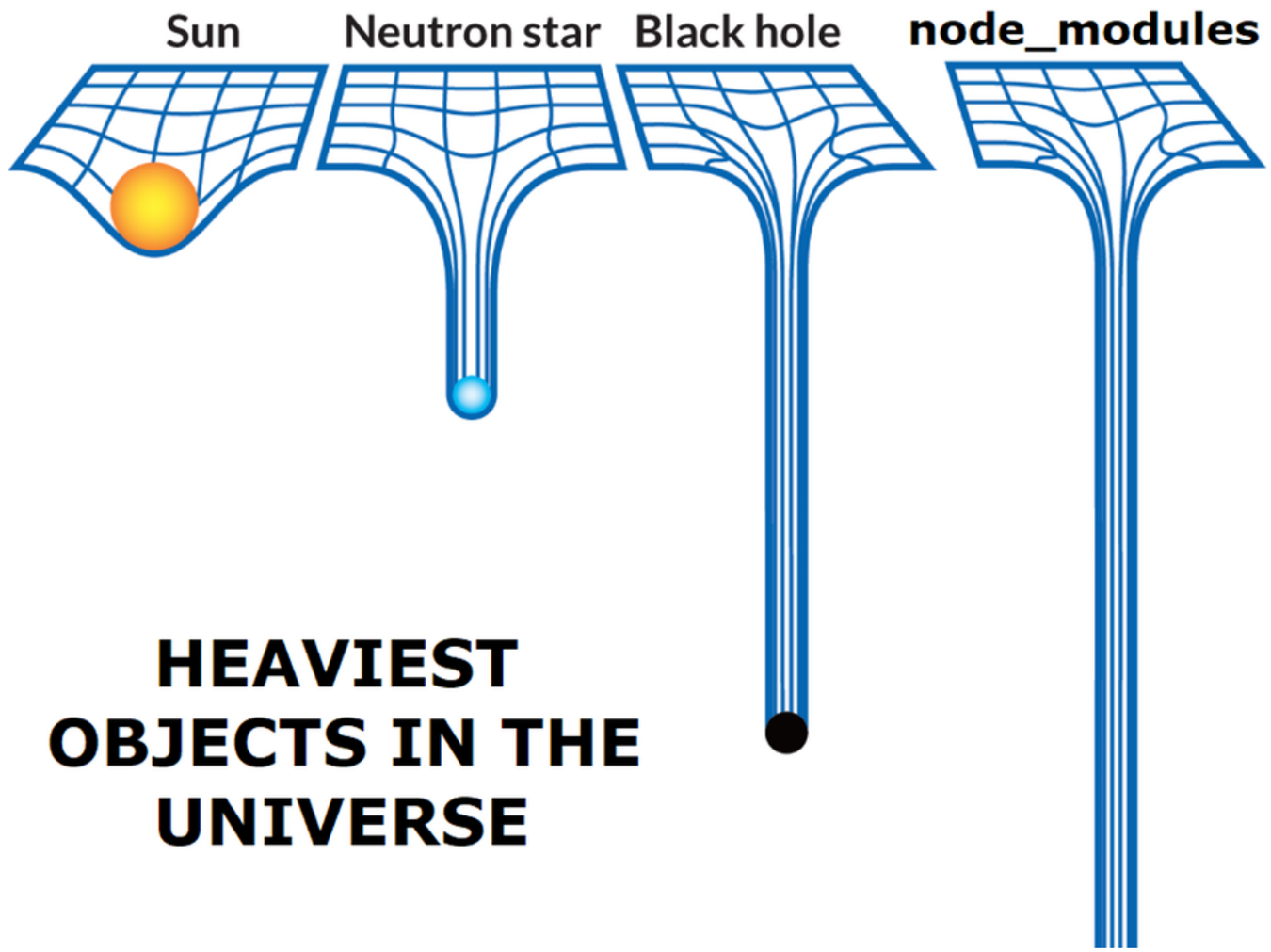
```
const express = require("express");
```

PACKAGE[-LOCK].JSON

- package.json
  - contains the dependencies of your project
  - contains also some metadata
  - must be added to Git index
- package-lock.json
  - similar to composer.lock
  - contains the list of installed dependencies
  - must be added to Git index

NODE_MODULES.JSON

- contains the installed dependencies of your project
- must be added to Git ignore file



EXPRESS.JS

WHAT IS EXPRESS ?

Express is a very popular minimalist web server framework for Node.js inspired by Sinatra (Ruby on Rails).

HELLO EXPRESS

Basic Hello World

```
1 /* see http://expressjs.com/en/starter/hello-world.html */
2 const express = require('express')
3 const app = express()
4
5 app.get('/', (req, res) => {
6     res.send('Hello World!')
7 })
8
9 app.listen(3000, () => {
10     console.log('Example app listening on port 3000!')
11 })
```

HTTP methods examples

```
app.get("/login",         (req, res) => ... ); // HTTP GET
app.post("/login",        (req, res) => ... ); // HTTP POST
app.put("/users/:id",     (req, res) => ... ); // HTTP PUT
app.patch("/users/:id",   (req, res) => ... ); // HTTP PATCH
app.delete("/users/:id", (req, res) => ... ); // HTTP DELETE
```

Full documentation and examples at expressjs.com

NODEMON

WHAT IS NODEMON ?

nodemon is a tool to automatically restart your application when it crash or you make changes.

Local installation:

```
% npm install --save-dev nodemon
```

Usage:

```
% ./node_modules/.bin/nodemon --verbose index.js
```

More about nodemon at https://nodemon.io

LET'S CODE !

YET ANOTHER BLOG ENGINE

Write a simple blogging server with express.
For now we'll use a static hardcoded array of posts items:

```
app.locals.posts = [
    {_id: 1, title: "Bacon Avocado Salad", body: "Place bacon in a large..."},
    {_id: 2, title: "Crispy Orange Beef", body: "Lay beef strips out in..."},
    {_id: 3, title: "Simple BBQ Ribs", body: "Place ribs in a large..."},
];
// create
app.post("/api/posts", (req, res) => ...);
// read
app.get("/api/posts", (req, res) => ...);
app.get("/api/posts/:id", (req, res) => ...);
// update
app.put("/api/posts/:id", (req, res) => ...);
// destroy
app.delete("/api/posts/:id", (req, res) => ...);
```

Questions ?

READ ON LATER

Understanding the Node.js Event loop by Trevor Norris