Lecture NODEJS

Play

Dynamic Vs. Static

- Static Page
  - Client/Consumer's Viewpoint: an url refers to an identical html file
  - Server/Producer's Viewpoint: a file stored within or sub-within the root folder of a Web Server
  - it is a html ...
  - Can be display directly at a browser

- Dynamic Page
  - Client/Consumer's Viewpoint: an url refers to a dynamic html (maybe vary each time requested)
  - Server/Producer's Viewpoint: a program/script produces html
  - it is NOT a html, but a program producing html(s)
  - Can't be display directly at a browser

- Dynamic Web Page, Dynamic HTML(DHTML), what's the difference?

Server-Side Web Programming

- server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl
- the web server contains software that allows it to run those programs and send back their output as responses to web requests
- each language/framework has its pros and cons

PHP vs. Node.js

- JavaScript everywhere
- https://www.infoworld.com/article/3166109/application-development/php-vs-nodejs-an-epic-battle-for-developer-mind-share.html

Where PHP wins: Mixing code with content

- You're typing along, pouring thoughts into text for your website, and you want to add a branch to the process, a little if-then statement to make it look pretty, say, depending on some parameter in the URL. Or maybe you want to mix in text or data from a database. With PHP, you open up the magic PHP tags and start writing code within seconds. No need for templates—everything is a template! No need for extra files or elaborate architectures, only programmable logistical power at your fingertips.

Where Node wins: Separating concerns

- Mixing code with content is a crutch that can end up crippling you. Sure, it's fun to mix code in with HTML the first two or three times you do it. But soon your code base becomes a tangled mess of logic. Real programmers add structure and separate the cosmetic layer from the logical layer. It's cleaner for new programmers to understand and easier to maintain. The frameworks running on Node.js are built by programmers who know that life is better when the model, view, and controller are separate.

Where PHP wins: Deep code base

- The web is filled with PHP code. The most popular platforms for building websites (WordPress, Drupal, Joomla) are written in PHP. Not only are the platforms open source, but so are most of their plugins. There's PHP code everywhere, and it's waiting for you to download, modify, and use for your needs.

Where Node wins: Newer code means more modern features

- Sure, there are thousands of great open source PHP files, but some are 8-year-old WordPress plugins hoping and praying that someone will download them. Who wants to spend hours, days, or weeks monkeying with code that hasn’t been updated in years? Node.js plugins are not only newer, they were built with full knowledge of the latest architectural approaches. They were built by programmers who understand that modern web apps should push most of the intelligence to the client.

Where PHP wins: Simplicity (sort of)

- There's not much to PHP: a few variables and basic functions for juggling strings and numbers. It's a thin layer that doesn't do much except move the data from port 80 to the database and back. That's what it's supposed to do. A modern database is a magical tool, and it makes sense to leave the heavy lifting to it. PHP is the right amount of complexity for a job that's not supposed to be complex.

Where Node wins: Complexity of closures and more

- JavaScript may have many little idiosyncrasies that drive some mad, but for the most part it is a modern language that sports a modern syntax and a few useful features like closures. You can reconfigure and extend it easily, making powerful libraries like jQuery possible. You can pass functions around like objects. Why limit yourself?

Where PHP wins: New code is helping it catch up

- If you're a programmer who wants to do more than interact with a database and format the results, you can now do more with PHP without holding your nose. Facebook's HHVM adds support for Hack, a complete language filled with modern features like type annotations, generics, and lambda expressions. Using this limits your code to running only on the HHVM, but that's not the worst thing in the world. It's very fast.

Where Node wins: Dozens of language options

- If PHP users are happy to get access to Hack, they should consider moving to the world of Node.js because many major languages can be cross-compiled to run in JavaScript. There are well-known options like Java, C#, or Lisp and dozens of others like Scala, OCaml, and Haskell. There are even gifts for nostalgic lovers of BASIC or Pascal.

Where PHP wins: No client app needed

- All of the talk about using the same language in the browser and on the server is nice, but what if you don't need to use any language on the browser? What if you ship the data in HTML form? The browser pops it up, and there are no headaches or glitches caused by misfiring JavaScript threads that try to create a page on the browser from two dozen web service calls. Pure HTML works more often than anything else, and PHP is optimized to create that. Why bother with JavaScript on the browser? Build up everything on the server and avoid overloading that little browser on the little phone.

Where Node wins: Service calls are thinner than HTML-fat PHP calls

- While AJAX-crazy HTML5 web apps can have too many moving parts, they are cool—and very efficient. Once the JavaScript code is in the browser cache, the only thing that moves along the wires is the new data. There's not a ton of HTML markup, and there are no repeated trips to download the entire page. Only the data has changed. If you're willing to put in the time to create a slick browser-side web app, there's a big payoff. Node.js is optimized to deliver the data and only the data through web services. If your app is complex and data-rich, it's a good foundation for efficient delivery.

Where PHP wins: SQL

- PHP was built to co-exist with MySQL and its many variants, like MariaDB. If MySQL isn't exactly right, there are other great SQL databases from Oracle and Microsoft. Your code can switch with a few changes to your queries. The vast SQL world doesn’t end at its borders. Some of the most stable, well-developed code will interface with an SQL database, meaning all that power can also be easily integrated into a PHP project. It may not be one perfect, happy family, but it's a big one.

Where Node.js wins: JSON

- If you must have access to SQL, Node.js has libraries to do that. But Node.js also speaks JSON, the lingua franca for interacting with many of the latest NoSQL databases. That’s not to say you can’t get JSON libraries for your PHP stack, but there's something fluid about the simplicity of working with JSON when using JavaScript. It's one syntax from browser to web server to database. The colons and the curly brackets work the same way everywhere. That alone will save you from hours of frustration.

Where PHP wins: Speed of coding

- For most developers, writing PHP for web apps feels faster: no compilers, no deployment, no JAR files or preprocessors—simply your favorite editor and some PHP files in a directory. Your mileage will vary, but when it comes to banging a project together quickly, PHP is a good tool to use.

Where Node.js wins: Raw speed

- Writing JavaScript code is a bit harder when you're counting curly brackets and parentheses, but when it's done, your Node.js code can fly. The callback mechanism is brilliant because it saves you from juggling the threads. The core is well-built and designed to do all that for you. Isn't that what everyone wants?

Where PHP wins: Competition

- The battle for the hearts and minds of PHP developers is still unfolding. The HHVM team and the Zend team are working hard to deliver fast code for everyone. Independent benchmarks are appearing, and everyone is pushing the code bases to the limit. This only means better performance.

Where Node.js wins: Solidarity

- Do you really want two different code bases? Sure, competition helps, but fragmentation soon follows. What happens when your code runs on only one of the two? Competition doesn't do any good if you have to spend weeks or months rewriting your code. While Node.js experienced its own splintering a few years back, with the launch of io.js, the Node.js universe has since reunited, giving it the kind of language solidarity that PHP developers may soon long for.

About Node.js……

- Created by Ryan Dahl in 2009
- Development && maintenance sponsored by Joyent
- Licence MIT
- Based on Google V8 Engine
- +99 000 packages

Success Stories.....

Linked in

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 [...] »

PayPal™

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

GROUPON

Walmart

YAHOO!

UBER

Development for the Web....

- Traditional desktop applications have a user interface wired up with background logic
  - user interfaces are based on Windows Forms, Jswing, WPF, Gtk, Qt, etc. and dependant on operating system
- On the web user interfaces are standardized
  - HTML for markup
  - CSS for style
  - JavaScript for dynamic content
- In the past proprietary technologies existed on the web, e.g. Adobe Flash, ActiveX
  - They are slowly dying
- Data is generated on the server, transferred to the client and displayed by the browser
- Server Side technologies include PHP, JSP, ASP.net, Rails, Djanog, and yes, also node.js

Why node.js ?

- Non Blocking I/O
- V8 Javascript Engine
- Single Thread with Event Loop
- 40,025 modules
- Windows, Linux, Mac
- 1 Language for Frontend and Backend
- Active community

What is node.js?

- Asynchronous I/O framework
- Core in c++ on top of v8
- Rest of it in javascript
- Swiss army knife for network Related stuffs
- Can handle thousands of Concurrent connections with Minimal overhead (cpu/memory) on a single process
- It’s NOT a web framework, and it’s also NOT a language

> Node.js is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
>
> — from nodejs.org

The idea behind node.js....

- Perform asynchronous processing on single thread instead of classical multithread processing, minimize overhead & latency, maximize scalability

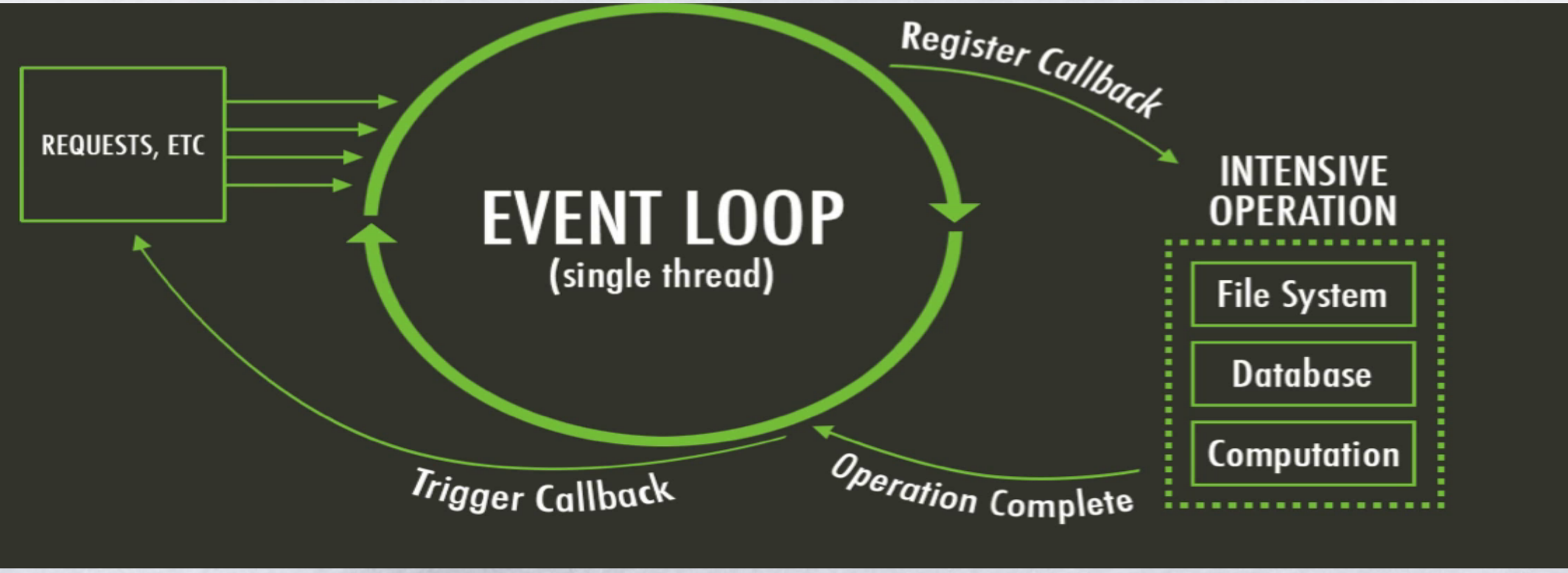
- Scale horizontally instead of vertically

- Ideal for applications that serve a lot of requests but don't use/need lots of computational power per request

- Not so ideal for heavy calculations, e.g. massive parallel computing

- Also: Less problems with concurrency

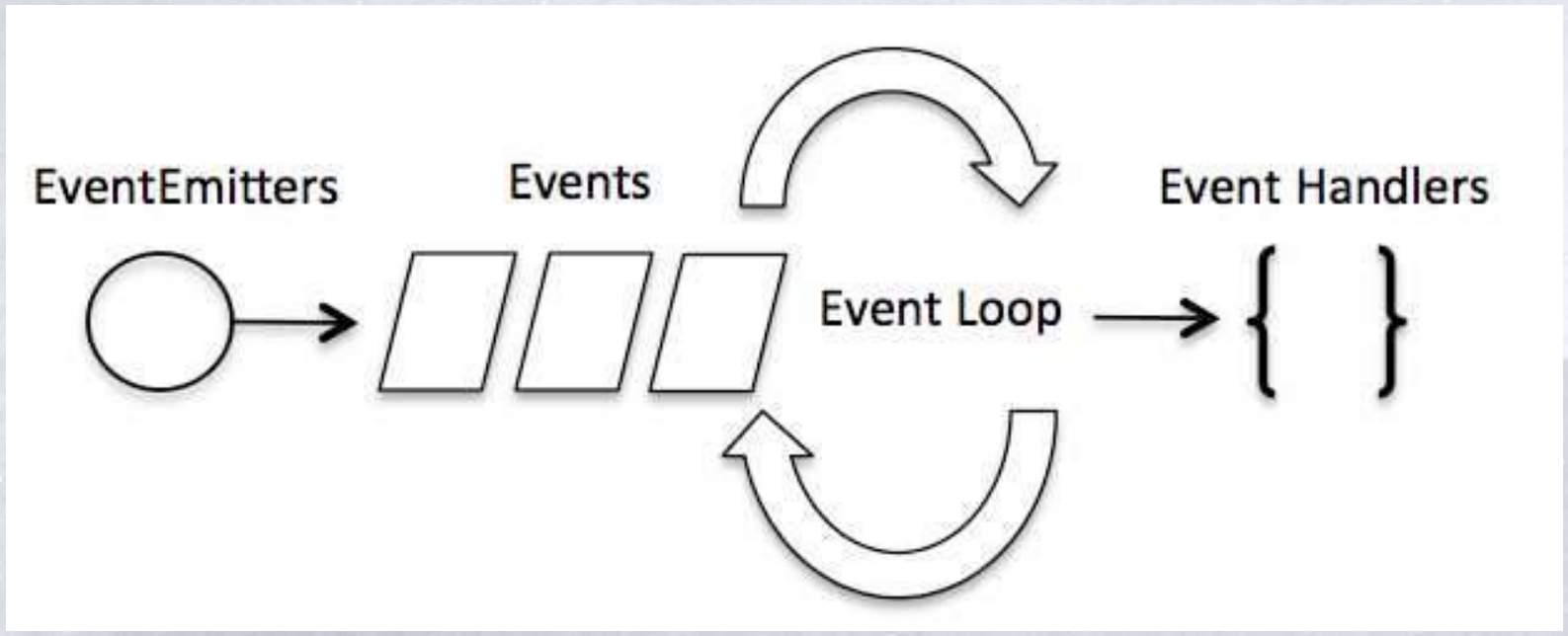
Node.js Event Loop



There are a couple of implications of this apparently very simple and basic model

- Avoid synchronous code at all costs because it blocks the event loop
- Which means: callbacks, callbacks, and more callbacks

事件驱动模型

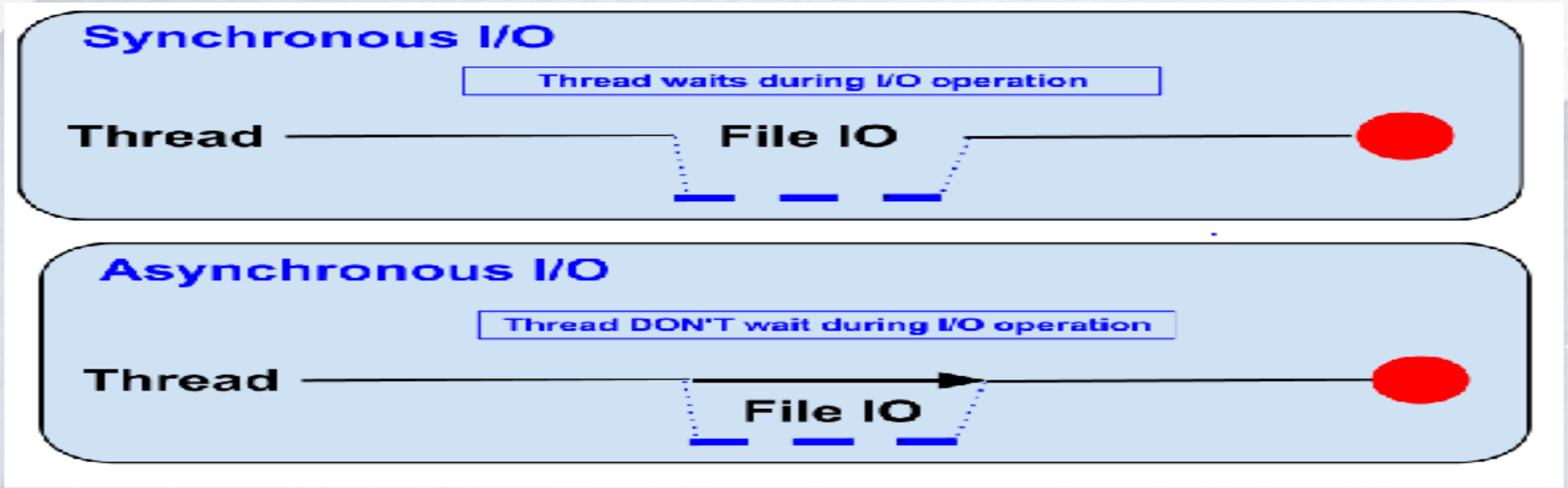


Synchronous vs Asynchronous

Synchronous	Asynchronous
Waits for each operation to complete, after that executes next operation.	Never waits each operation to complete, executes all at one time. Results handled as available.
Step by Step execution	Callbacks used to handle results

Blocking vs Non-Blocking……

- Example :: Read data from file and show data



同步式 I/O 和异步式 I/O 的特点

同步式 I/O(阻塞式)	异步式 I/O(非阻塞式)
利用多线程提供吞吐量	单线程即可实现高吞吐量
通过事件片分割和线程调度利用多核CPU	通过功能划分利用多核CPU
需要由操作系统调度多线程使用多核 CPU	可以将单进程绑定到单核 CPU
难以充分利用 CPU 资源	可以充分利用 CPU 资源
内存轨迹大 ， 数据局部性弱	内存轨迹小，数据局部性强
符合线性的编程思维	不符合传统编程思维

Blocking.....

- Read data from file
- Show data
- Do other tasks

```
var data = fs.readFileSync( “test.txt” );
console.log( data );
console.log( “Do other tasks” );
```

Non-Blocking……

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

```
fs.readFile( “test.txt”, function( err, data ) {
console.log(data);
});
```

Callback

When to use it ?

- Chat/Messaging
- Real-time Applications
- Intelligent Proxies
- High Concurrency Applications
- Communication Hubs

Node.js for....

- 网站（如express/koa等）
- im即时聊天(socket.io)
- api（移动端，pc，h5）
- HTTP Proxy（淘宝、Qunar、腾讯、百度都有）
- 前端构建工具(grunt/gulp/bower/webpack/fis3...)
- 写操作系统（NodeOS）
- 跨平台打包工具（PC端的electron、nw.js，比如钉钉PC客户端、微信小程序IDE、微信客户端，移动的cordova，即老的Phonegap，还有更加有名的一站式开发框架ionicframework）
- 命令行工具（比如cordova、shell.js）
- 反向代理（比如anyproxy，node-http-proxy）

When to NOT use Node.js

- When you are doing heavy and CPU intensive calculations on server side
- Node.js with a relational DB behind. Relational DB tools for Node.js are still in their early stages.

Getting Started.....

- https://nodejs.org/en/download/

LTS
Recommended For Most Users

Current
Latest Features

Windows Installer	Macintosh Installer	Source Code
node-v8.9.1-x86.msi	node-v8.9.1.pkg	node-v8.9.1.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binaries (.tar.gz)	64-bit	
Linux Binaries (x86/x64)	32-bit	64-bit
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v8.9.1.tar.gz	

Additional Platforms

SunOS Binaries	32-bit	64-bit
Docker Image	Official Node.js Docker Image	
Linux on Power Systems	64-bit le	
Linux on System z	64-bit	
AIX on Power Systems	64-bit	

Node Package Manager (NPM)

- www.npmjs.org
- 450.000+ modules
- package.json file (npm init)
- npm install <module.name> --save
-

File package.json.....

- Project informations
  - Name
  - Version
  - Dependencies
  - Licence
  - Main file
  - Etc...

Install module.....

- $npm install <module name>

Popular Node.js modules

- Express - Express.js, a simple web development framework for Node.js, and the de-facto standard for the majority of Node.js applications out there today.
- connect - Connect is an extensible HTTP server framework for Node.js, providing a collection of high performance middleware components.
- socket.io - Server-side component of the two most common websockets components out there today.

In NodeJS

Standard JavaScript with

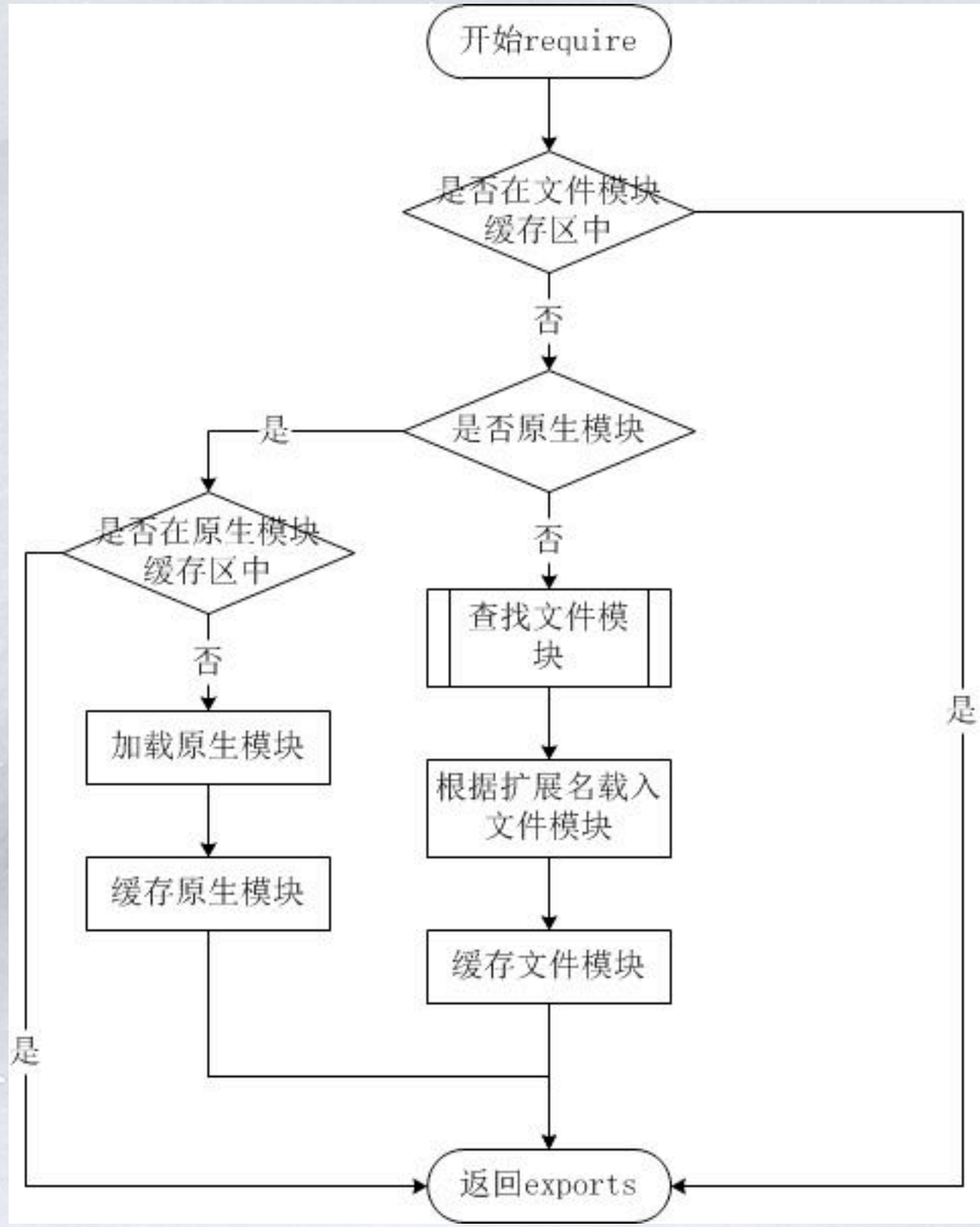
- Buffer
- C/C++ Addons
- Child Processes
- Cluster
- Console
- Crypto
- Debugger
- DNS
- Domain
- Events
- File System
- Globals
- HTTP
- HTTPS
- Modules
- Net
- OS
- Path
- Process
- Punycode
- Query Strings
- Readline
- REPL
- Stream
- String Decoder
- Timers
- TLS/SSL
- TTY
- UDP/Datagram
- URL
- Utilities
- VM
- ZLIB

... but without DOM manipulation

Using module.....

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

require方法查找策略



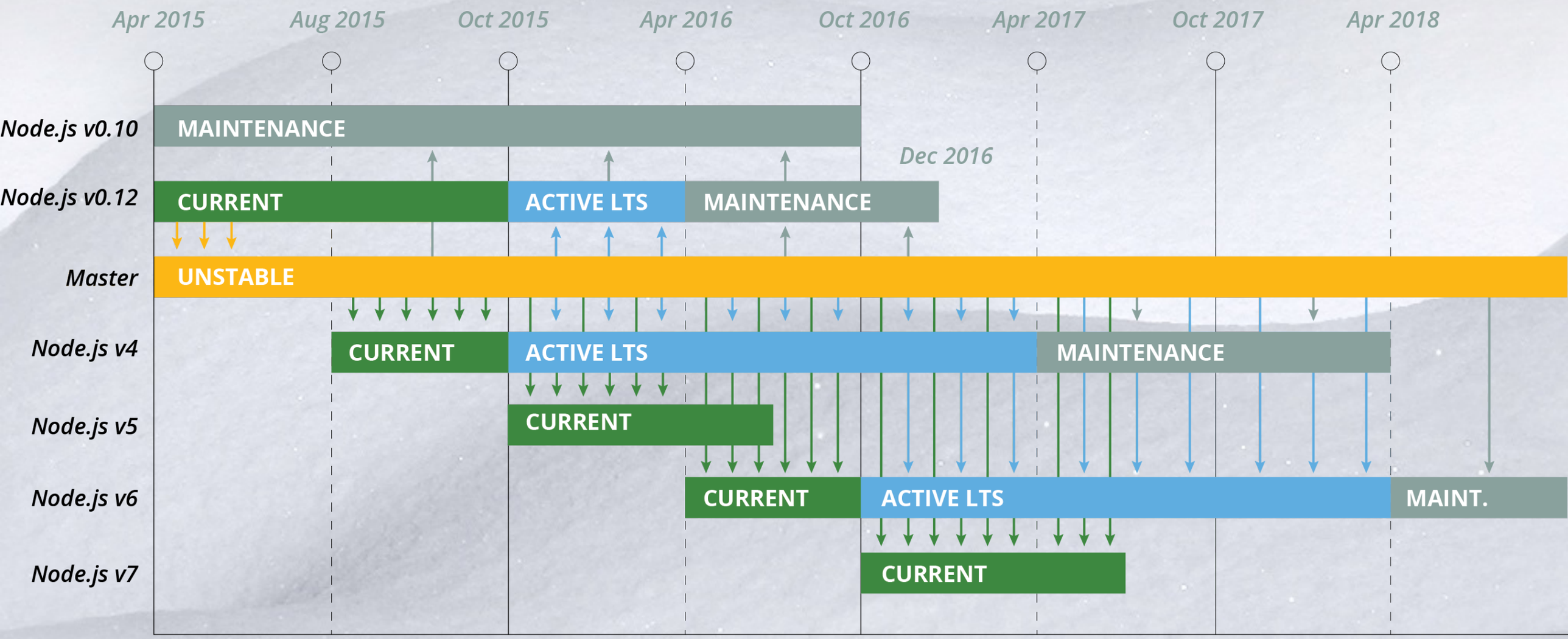
A simple file server

```
var http = require(“http”);

var server = http.createServer(function(req, res){
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n')
});

server.listen(3000);

console.log('Server running at http://localhost:3000/');
```

```
node server.js
```

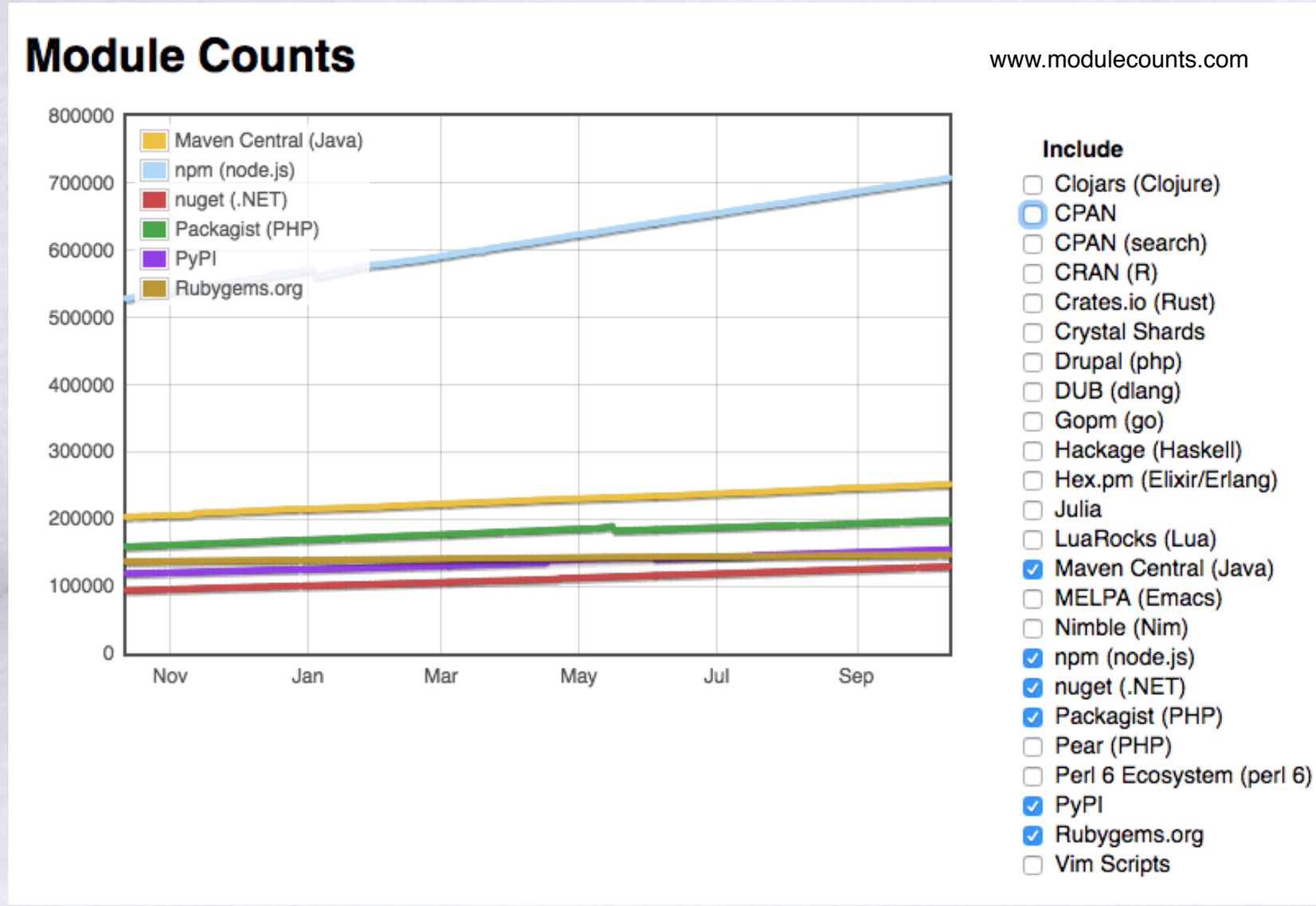
版本帝?

Node.js Long Term Support Release Schedule



已无性能优势?

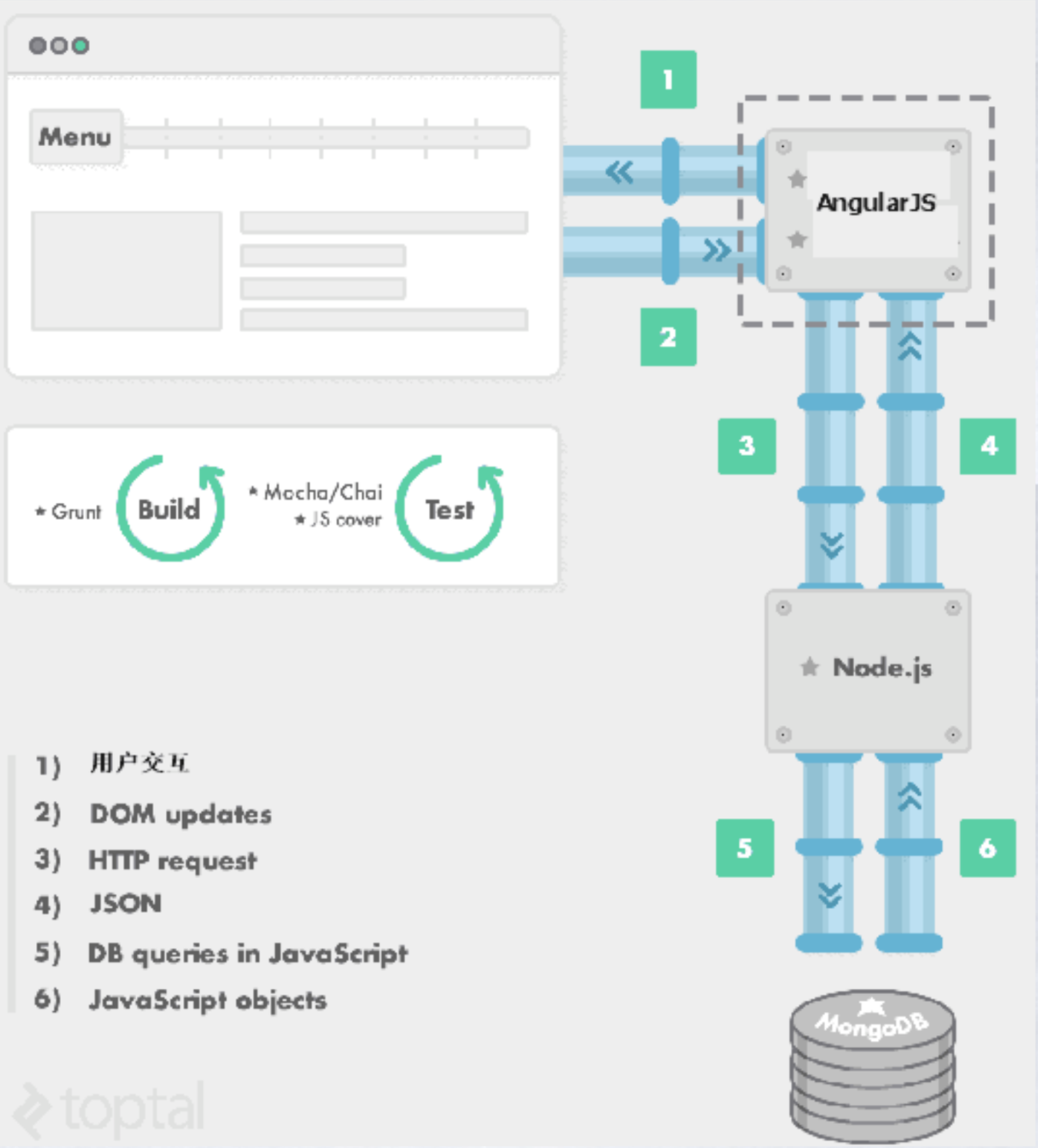
- 实现成本
- 调优成本
- 学习成本
-



MEAN

- MEAN是一个Javascript平台的现代Web开发框架总称
  - MongoDB是一个使用JSON风格存储的数据库，非常适合javascript。(JSON是JS数据格式)
  - ExpressJS是一个Web应用框架，提供有帮助的组件和模块帮助建立一个网站应用。
  - AngularJS是一个前端MVC框架。
  - Node.js是一个并发 异步 事件驱动的Javascript服务器后端开发平台。

MEAN架构原理



1) 用户交互
2) DOM updates
3) HTTP request
4) JSON
5) DB queries in JavaScript
6) JavaScript objects

toptal

Web框架

框架名称	特性	点评
Express	简单、实用，路由中间件等五脏俱全	最著名的Web框架
Derby.js && Meteor	同构	前后端都放到一起，模糊了开发便捷，看上去更简单，实际上上对开发来说要求更高
Sails、Total	面向其他语言，Ruby、PHP等	借鉴业界优秀实现，也是Node.js成熟的一个标志
MEAN.js	面向架构	类似于脚手架，又期望同构，结果只是蹭了热点
Hapi和Restfy	面向Api && 微服务	移动互联网时代Api的作用被放大，故而独立分类。尤其是对于微服务开发更是利器
ThinkJS	面向新特性	借鉴ThinkPHP，并慢慢走出自己的一条路，对于Async函数等新特性支持，无出其右
Koa	专注于异步流程改进	下一代Web框架

框架选型

- 业务场景、特点
- 自身团队能力、喜好，有时候技术选型决定团队氛围的，需要平衡激进与稳定
- 熟悉程度
- 个人学习求新，企业架构求稳，无非喜好与场景而已

技术栈演进

分类	2015年	2016年	选型原因
Web框架	express 4.x	koa 1.0 && 2.0	主要在流程控制上的便利
数据库	mongoose（mongodb）	mongoose（mongodb）	对mongodb和mysql支持都一样，不过是mongodb更简单，足以应付绝大部分场景
异步流程控制	bluebird（Promise/A+实现）	bluebird（Promise/A+实现）1）Koa 1.0 使用co + generator 2）Koa 2.0 使用async函数	流程控制演进路线，从promise到async函数，无论如何，promise都是基石，必要掌握的
模板引擎（视图层）	ejs && jade	jade && nunjucks	给出了2种，一种可读性好，另一种简洁高效，都是非常好的
测试	mocha	ava	mocha是Node.js里著名的测试框架，但对新特性的支持没有ava那么好，而ava基于babel安装也要大好多
调试	node-inspector	VSCode	在Node 6和7出来之后，node-inspector支持的不是那么好，相反VSCode可视化，简单，文件多时也不卡，特别好用

预处理器

名称	实现	描述
模板引擎	art\mustache\ejs\hbs\jade …	上百种之多，自定义默认，编译成html，继而完成更多操作
css预处理器	less\sass\scss\rework\postcss	自定义语法规则，编译成css
js友好语言	coffeescript、typescript	自定义语法规则、编译成js

跨平台

平台	实现	点评
Web/H5	纯前端	
PC客户端	nw.js和electron	尤其是atom和vscode编辑器最为著名，像钉钉PC端，微信客户端、微信小程序IDE等都是这样的，通过web技术来打包成PC客户端
移动端	cordova（旧称PhoneGap），基于cordova的ionicframework	这种采用h5开发，打包成ipa或apk的应用，称为Hybrid开发（混搭），通过webview实现所谓的跨平台，应用的还是非常广泛的

其他

用途	说明	前景
爬虫	抢了不少Python的份额，整体来说简单，实用	看涨
命令行工具	写工具、提高效率，node+npm	看涨
微服务与RPC	Node做纯后端不好做，但在新项目和微服务架构下，必有一席之地	看涨
微信公众号开发	已经火了2年多了，尤其是付费阅读领域，还会继续火下去，gitchat就是使用Node.js做的	看涨
反向代理	Node.js可以作为nginx这样的反向代理，比如node-http-proxy和anyproxy等，其实使用Node.js做这种请求转发是非常简单的	看涨

Ref

- cnodejs.org
- https://github.com/ElemeFE/node-interview/tree/master/sections/zh-cn

Thanks!!!