# 第四天上课笔记

在很久很久以前， 2009年，有一个小伙子 叫 Cyan；使用C++做一个项目，需要服务器提供相关数据；

他嫌麻烦：需要服务器软件，apache，iis，nginx，Tomcat；

服务器端开发语言：JAVA，.net，C#，PHP，ruby等；

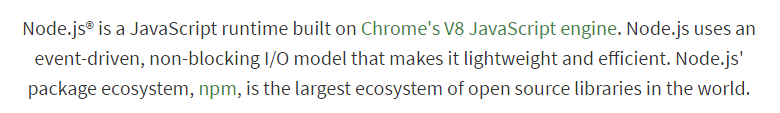
于是自己用C++写了个web.js; 这个是nodejs的前身；

刚开始：只可以运行在 linux 操作系统；提供访问服务和运算能力；

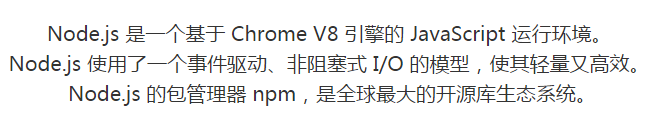
从0.6版本，Window提供支持；

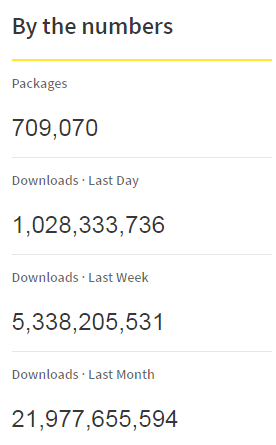
Nodejs是什么

官方网站https://nodejs.org/en/如此说：



翻译成人话<http://nodejs.cn/>：



数据说明一切：<https://www.npmjs.com/>

说直白点，~~Nodejs是运行在服务器端的JavaScript~~；

准确的说，Nodejs是一个基于V8引擎的JavaScript运行环境；JavaScript天生是运行在客户端的，也就是浏览器的解释器里面；为了让JavaScript运行在服务器端，那么他需要一个运行环境，也就是我们的nodejs了；

思考ECMAScript、BOM、DOM的关系；

C#

.Net Framework

JavaScript

JAVA

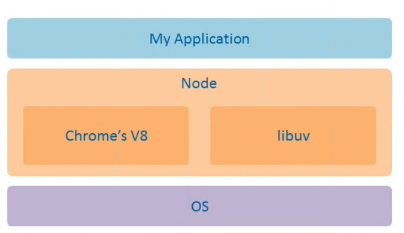
Nodejs

Java vm

同时，node还提供了大量工具库，让我们可以去完成客户端JavaScript所不能完成的事情：读取文件、创建HTTP等，也就是传说中的API；

所以，是Nodejs选择和成就了JavaScript，而不是JavaScript发展出来一个Nodejs；当然，之所以选择JavaScript也是因为JavaScript本身也比较火。

所以，严格讲nodejs并不是一门语言，也不是JavaScript框架，是JavaScript运行环境，不过我们在交流时并不会严格去区分；



其中，chrome v8负责解释ES；

Libuv是作者开发好的一些系统操作，比如文件读取、http协议实现；

Nodejs可以拿来做web开发：可以直接做后台服务，或者只做服务分发；还可以拿来做游戏，处理高并发的鼠标操作和键盘操作；

首先执行：**javascrip---》核心模块---》C++扩展的模块---》libuv**

Windows常用命令行

切换磁盘符：D:

切换目录：cd 跟上路径 change directory

创建目录：mkdir make directory

目录文件列表：dir directory

清空控制台：cls clear screen

删除文件：del delete

查看系统路径：path 得到path的配置值

**如何运行nodejs:**

1,DOS窗口下，输入node，就进入node编程；REPL模式：直接在命令行下执行；

2,使用node执行js文件：

Node 文件名 后缀可以省略；

注意文件路径，在文件所在的目录打开cmd窗口，就可以不要路径，如果执行其他地方的nodejs，需要指定相对路径或绝对路径；

Node命令基本用法

1. 进入REPL环境：node

-REPL：Read，Eval，Print，Loop，也就是接收指令并执行，打印结果到控制台，然后在循环继续操作；

1. CTRL+C连着摁两次 或者 .exit退出node的REPL；

Nodejs全局对象

1. global：

-类似客户端js的window；可以在REPL打印看下；

-可以用来挂载全局变量；

Nodejs异步编程

这是阻塞的：

var a = 1;

console.log(a);

console.time(‘main’);

for (var i = 0; i <50000000000; i++) {

// console.log(i);

}

console.timeEnd(‘main’);

console.log('终于干完了');

怎么不阻塞执行:

var a = 1;

console.log(a);

console.time('main');

setTimeout(()=>{

for (var i = 0; i <5000000000; i++) { }

console.log('我后面才执行');

}, 0);

console.timeEnd('main');

console.log('终于干完了');

我们还见过的异步：$.ajax

Nodejs是非阻塞I/O操作

Node.js 使用了一个事件驱动、非阻塞式 I/O的模型，使其轻量又高效。

I/O:input/out,表示输入输出操作；

比如：鼠标操作、键盘操作等就是I/O操作；

**事件驱动**：

到饭店吃饭；

点餐：找服务员点餐，告诉厨师；

服务员是站在你旁边等着上菜吗？？？

服务员不会站在这里，会继续服务下一个客人；

乡村基：点餐区、取餐区

**异步和同步、非阻塞式、阻塞**：

老王要用水壶烧水：

水壶：哑壶，响壶；

老王使用哑壶在厨房看着烧水，负（sha）责（bu）的老（la）王（ji）；

对老王来讲，阻塞的；

对水烧开这个通知来讲：同步；

老王使用哑壶烧水，自己去客厅看电视；

不断的到厨房去瞅一眼；

对老王来讲，非阻塞；

对水烧开这个通知来讲：同步；

老王使用响壶烧水，自己去阳台思考人生；

对老王来讲：非阻塞；

对水烧开这个通知来讲：异步；

老王使用响壶烧水，自己在厨房看着；

对老王来讲：阻塞；

对水烧开这个通知来讲：异步；

异步是指消息通知；

非阻塞：CPU的线程；

异步的场景：

$.ajax({

//异步 消息通知

Success:function(){

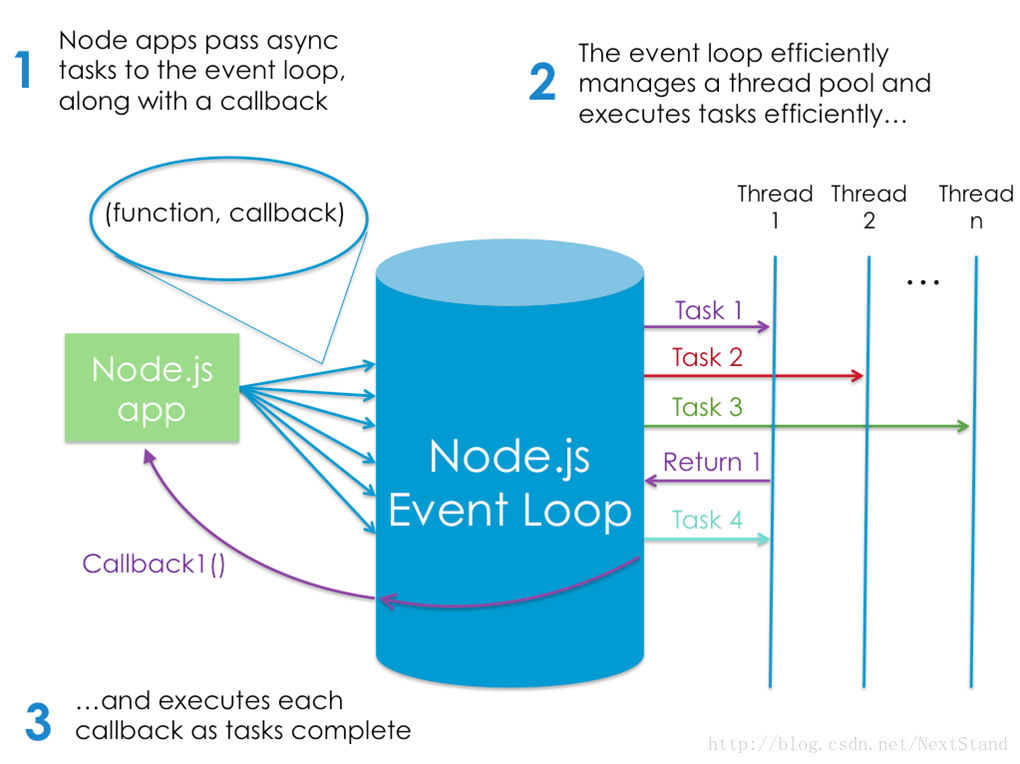
},

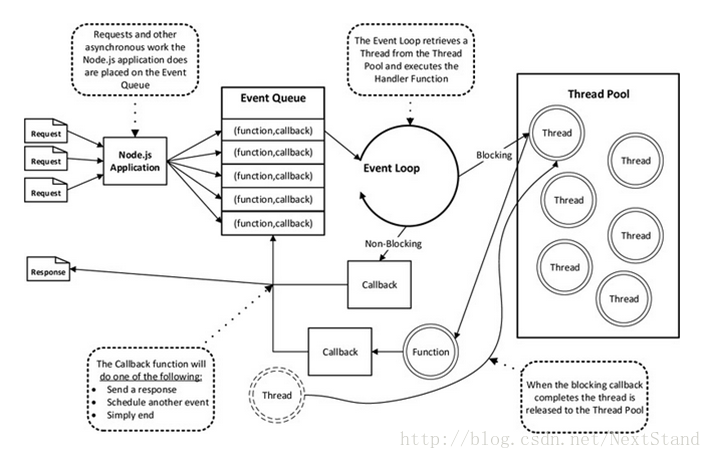
Fail:function(){

}

})

看图理解nodejs的事件驱动和非阻塞驱动：





Nodejs把所有阻塞操作交给一个线程池；

主线程负责不断的往返调度；

非阻塞的优势

1. 提高代码的响应效率；
2. V8的特点是单线程运行，一次只能完成一个任务；Node采用异步操作，把任务放到一个队列中，等到前面的任务执行完成后执行后面的操作；
3. 很好的兼容IO操作异常；

在进行I/O操作的时候，CPU做自己的事情，网络传送自己的数据；

I/O操作执行完成后，告诉CPU可以进行相关的数据计算处理；

Nodejs做web服务的时候，性能是PHP的8倍；

做了一个平台：

PHP 200台 2万 == 400万

Nodejs 25台 2万 == 50万

回调函数的设计

对于一个操作如果需要回调函数，那么：

1. 回调函数肯定是该操作函数的最后一个参数，至于前面有多少参数不定；
2. 如果不需要回调函数，那么是可以省略的；
3. 所有的回调函数的第一个参数默认接收错误信息，第二个参数才是接收到的回调数据；
4. 这些都是node的统一约定；

认识进程和线程

启动一个应用程序就是启动一个进程；

一个进程至少有一个线程；

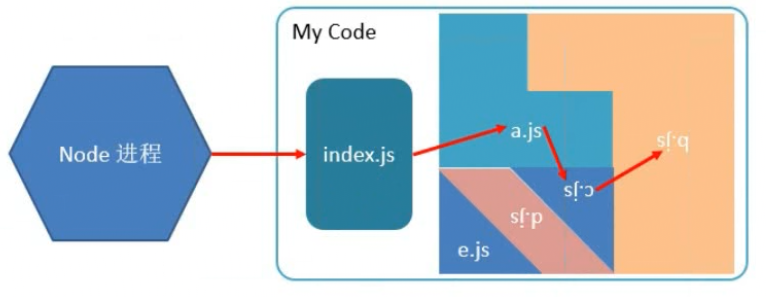
线程的创建是比较消耗时间的；

模块化编程

Nodejs做项目的时候，我们应该对整个项目进行 模块化；

调用其它模块的模块称为主模块；

模块和文件是一一对应；



模块分类：

1. 系统模块：我们可以直接使用的，nodejs提供的模块；
2. 自定义模块：根据实际需求自己定义的一些模块；
3. 第三方模块：更多的时候我们使用别人写好的模块来完成相应的功能；

模块使用流程：

创建—》导出成员—》载入模块—》使用；

模块内的变量是局部变量，不是全局变量；

**操作模块：模块的导入导出**

nodejs预留的三个操作模块的方法

1,**require**：引用模块，可以是任何模块，参数是模块的路径；首先找系统模块，然后在node\_modules目录里面找，最后才找自定义模块；

如果自定义模块放置在node\_modules里面，前面不需要相对路径；

支持绝对路径；

2,**exports**：导出自定义的属性或者方法；

3,**module**：一次性导出全部属性或者方法；

全局变量global：

Global就是nodejs里面的Window

\_\_dirname：当前文件所在的文件夹，磁盘物理路径；

\_\_filename：当前文件的完整路径和文件名；