



Node VIP课程

讲师: 星云

QQ: 2146457496

课程目标

2

- ▶ 理解Node平台的组成及模块系统
- ▶ 理解Node异步编程模型
- ▶ 掌握Node核心API
- ▶ 熟悉常用的Node模块
- ▶ 掌握使用ORM框架访问数据库
- ▶ 掌握使用Express开发Web应用程序
- ▶ 了解Node分布式开发
- ▶ 了解Node的部署方式

准备知识

3

- ▶ 掌握JavaScript基本语法
- ▶ 熟悉JavaScript事件操作和异步编程方式
- ▶ 了解SQL语言基本语法

课程内容

4

- ▶ 第一部分：Node 介绍
- ▶ 第二部分：Node 模块系统
- ▶ 第三部分：Node 核心API
- ▶ 第四部分：Node 异步编程
- ▶ 第五部分：Node 数据库操作
- ▶ 第六部分：Node Web开发
- ▶ 第七部分：Node 部署

第一部分：Node 介绍

5

- ▶ Node的由来
- ▶ Node的整体结构
- ▶ Node的特点
- ▶ Node的应用场景
- ▶ Node的获取和安装
- ▶ Node学习资源
- ▶ Node开发环境介绍
- ▶ REPL交互式命令行

Node的由来

- ▶ Ryan Dahl为了开发一个高性能的Web服务器
- ▶ 基于事件驱动、非阻塞I/O
- ▶ JavaScript当选理由
 - ▶ 图灵完备
 - ▶ 闭包特性天生支持异步编程
 - ▶ 开发门槛低
 - ▶ 构建I/O库没有历史包袱
 - ▶ V8引擎的高性能
- ▶ Node就是网络中的一个节点



Node的整体结构

7



Node的特点

8

- ▶ 单线程
 - ▶ 只有一个线程
 - ▶ 无限循环
 - ▶ 异步工作方式
 - ▶ 不会死锁，没有线程切换的开销
 - ▶ 任何一个未处理的异常都会导致整个程序崩溃

Node的特点

9

► 异步编程

► 异步I/O

- 先发起I/O请求，不等待返回数据
- 当系统准备好数据，得到系统的通知再处理数据
- 代码不是顺序执行的

► 事件与回调函数

- 注册一个事件回调函数
- 当事件发生时，得到通知
- 得到通知时真正执行回调函数
- 代码不是顺序执行的

```
4
5  var fs = require("fs");
6
7  console.log("Step 1");
8  var file = fs.readFileSync('D:/test.txt');
9  console.log(file.toString());
10 console.log("Step 2");
11
```

```
4
5  var fs = require("fs");
6
7  console.log("Step 1");
8  fs.readFile('D:/test.txt', function (err, file) {
9    if (err) {
10     console.log(err);
11     throw err;
12    }
13    console.log(file.toString());
14  });
15  console.log("Step 2");
16
```

Node的特点

10

- ▶ 跨平台
 - ▶ libuv是跨平台的
 - ▶ Windows
 - ▶ Unix/Linux(Like)

Node的应用场景

11

- ▶ 高并发应用
- ▶ Web应用
- ▶ 分布式应用
- ▶ 实时应用
- ▶ 游戏
- ▶ 工具

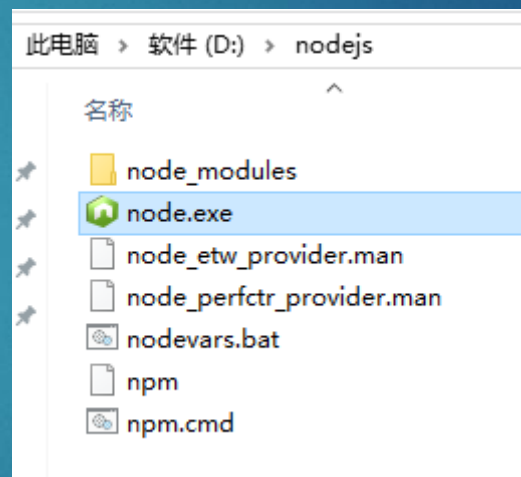
Node的获取和安装

12

- ▶ 官网下载<https://nodejs.org/en/download/>
- ▶ 中文网下载<http://nodejs.cn/download/>
- ▶ 默认安装
- ▶ 检查安装结果

```
C:\Users\>node --version  
v5.6.0
```

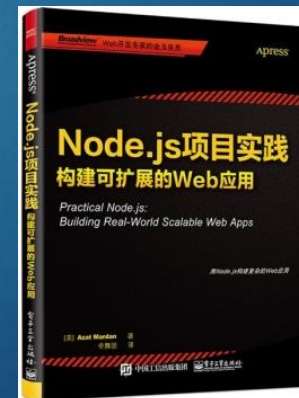
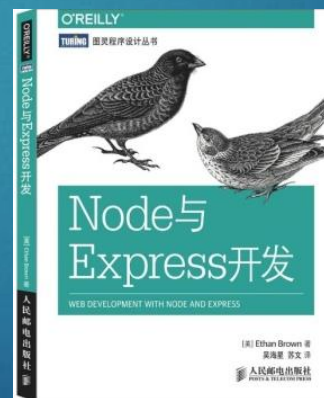
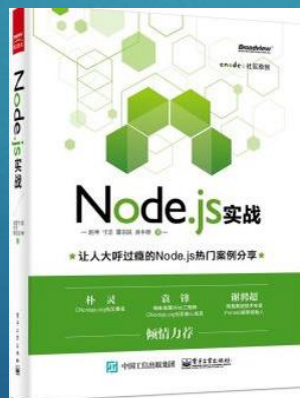
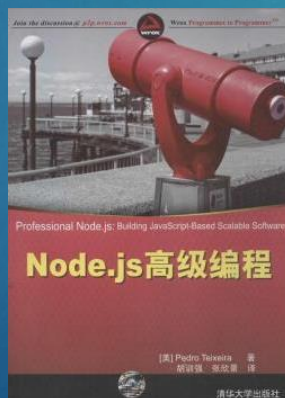
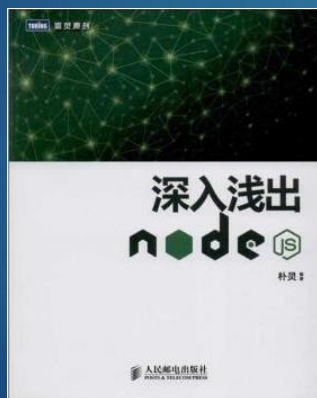
node-v5.6.0-x64.msi



Node学习资源

13

- ▶ Node官网<https://nodejs.org>
- ▶ Node中文网<http://nodejs.cn>
- ▶ Node核心API文档 <http://nodejs.cn/doc/node/>
- ▶ NPM官网<https://www.npmjs.com/>
- ▶ 相关书籍



Node开发环境介绍

14

- ▶ WebStorm
- ▶ Visual Studio + NTVS
- ▶ EditPlus
- ▶ VIM
- ▶ EMACS
- ▶ XCode
- ▶ Eclipse

Node开发环境介绍

15

► WebStorm 11

- <http://www.jetbrains.com/webstorm/download/>

webstorm 11的官方下载地址是
<http://www.jetbrains.com/webstorm/>
最好从官方地址下载安装。

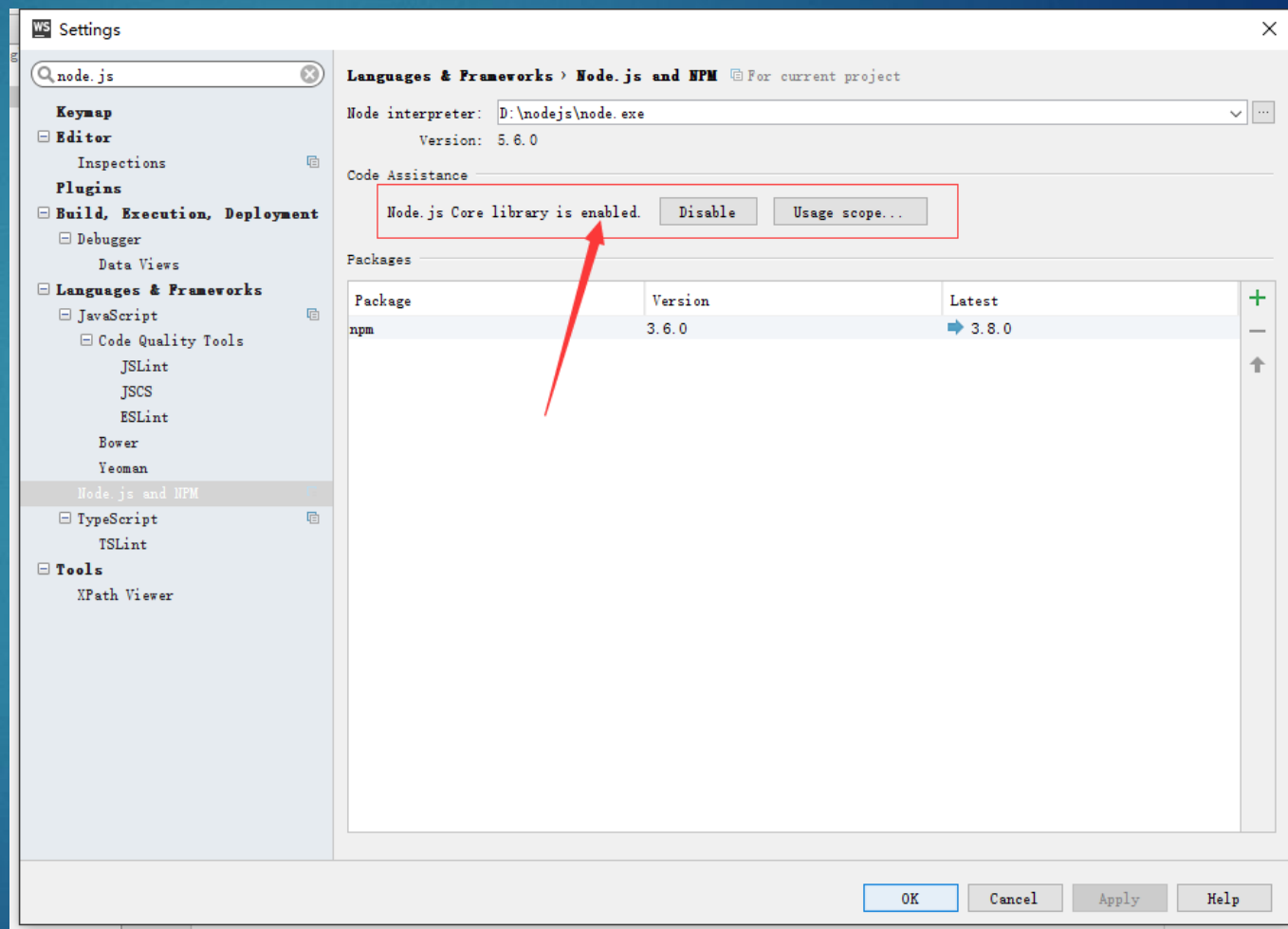
安装完成之后, 请使用license server的注册方式。
<http://0.idea.lanyus.com>

如果此地址不行, 请使用
<http://1.idea.lanyus.com>
<http://2.idea.lanyus.com>

Node开发环境介绍

16

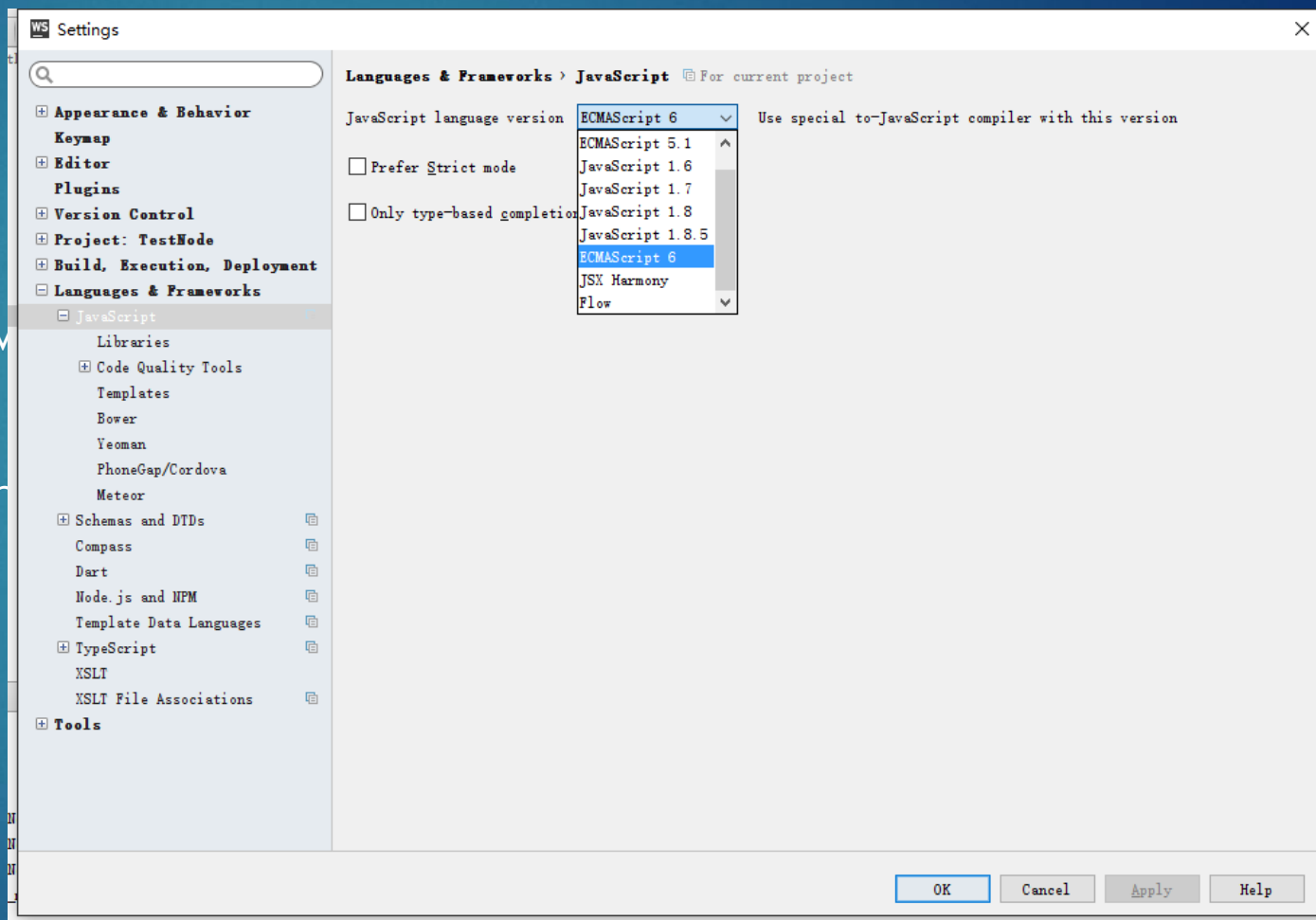
- ▶ WebStorm 11 配置
 - ▶ File->Default Settings
 - ▶ 左上角搜索node.js
 - ▶ 左侧Node.js and NPM
 - ▶ 右侧Core Assistance
 - ▶ Enable Core Lib



Node开发环境介绍

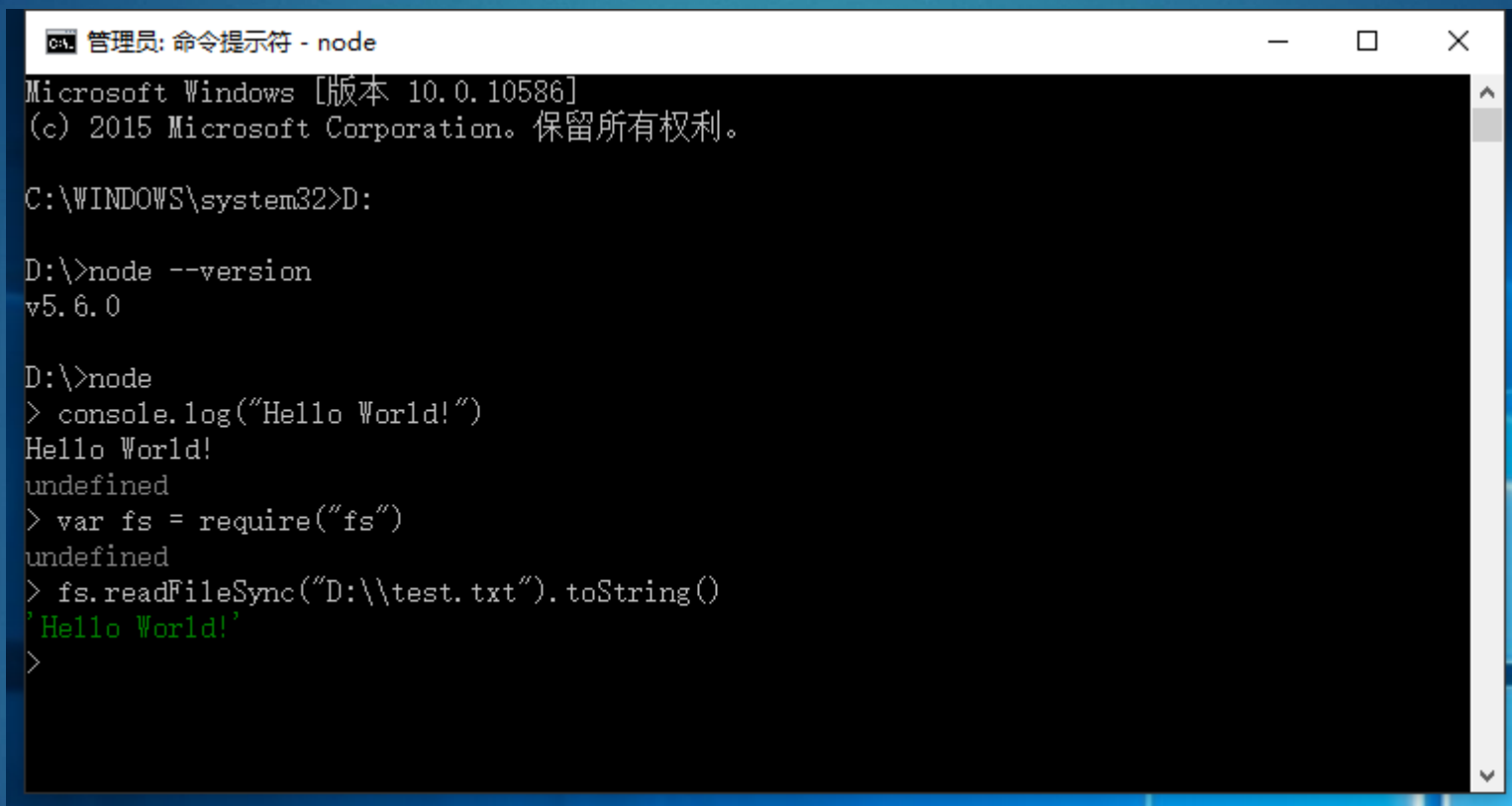
17

- ▶ WebStore 11 配置ES6
 - ▶ File -> Settings
 - ▶ Languages & Frameworks
 - ▶ 选择JavaScript
 - ▶ 右侧JavaScript version
 - ▶ 选择ECMAScript 6



REPL交互式命令行

18

A screenshot of a Windows Command Prompt window titled "管理员: 命令提示符 - node". The window shows the Node.js REPL environment. The user has navigated to the D: drive and entered several commands to test Node.js functionality. The output shows the Node.js version (v5.6.0), the result of a console.log statement ("Hello World!"), and the result of reading a file ("Hello World!").

```
Microsoft Windows [版本 10.0.10586]
(c) 2015 Microsoft Corporation. 保留所有权利。

C:\WINDOWS\system32>D:

D:\>node --version
v5.6.0

D:\>node
> console.log("Hello World!")
Hello World!
undefined
> var fs = require("fs")
undefined
> fs.readFileSync("D:\\test.txt").toString()
'Hello World!'
>
```


第二部分：Node模块系统

19

- ▶ JavaScript代码组织问题
- ▶ CommonJS规范介绍
- ▶ Node模块规范
- ▶ Node模块加载
- ▶ Node核心模块
- ▶ Node自定义模块
- ▶ Node包规范
- ▶ NPM介绍

JavaScript代码组织问题

20

- ▶ JavaScript语言特性没有明确定义代码的组织方式
 - ▶ 以前的JavaScript主要用于浏览器端
 - ▶ 页面端加载JavaScript的方式是使用<script></script>标签
 - ▶ 代码组织没有统一标准
 - ▶ 大部分人编码不考虑函数的可见性和作用域
 - ▶ 不能体现很好的封装性

CommonJS规范介绍



21

- ▶ CommonJS的目标是为JavaScript提供一个标准库
- ▶ 用于除浏览器端以外的一些通用用途
- ▶ 定义了模块和包的规范，以及一组API
- ▶ 目前知识一个建议，并不是标准
- ▶ Node借鉴了CommonJS规范，实现了自己的一套
- ▶ 官网: <http://www.commonjs.org/>

Node模块规范

22

- ▶ <http://www.commonjs.org/specs/modules/1.0/>
- ▶ Node模块规范跟CommonJS模块规范基本相同
 - ▶ 使用require函数导入一个模块
 - ▶ require函数返回模块export的API
 - ▶ 可以循环依赖
 - ▶ 如果require导入模块失败，throw一个异常
 - ▶ 在模块里有一个exports变量，用来导出对外公布的API
 - ▶ 模块只能使用exports变量来导出API，不能使用其他方式

```
4
5 function Module(id, parent){
6     this.id = id;
7     this.exports = {};
8     this.parent = parent;
9     if(parent && parent.children){
10         parent.children.push(this);
11     }
12     this.filename = null;
13     this.loaded = false;
14     this.children = [];
15 }
```


Node模块加载

- ▶ require函数
 - ▶ require(模块名)
 - ▶ 从当前路径开始，逐级向上查找node_modules目录
 - ▶ require(相对路径) 路径不需要带文件扩展名
 - ▶ require(绝对路径) 路径不需要带文件扩展名
- ▶ 模块加载一次以后会被缓存起来
 - ▶ require一个模块多次，得到的是同一个对象
 - ▶ 模块是根据模块的文件路径进行缓存的
- ▶ 模块间循环require
 - ▶ 会出现partial的模块对象

```
[ 'd:\\WebStormProjects\\TestNode\\module\\ex2\\node_modules',  
  'd:\\WebStormProjects\\TestNode\\module\\node_modules',  
  'd:\\WebStormProjects\\TestNode\\node_modules',  
  'd:\\WebStormProjects\\node_modules',  
  'd:\\node_modules' ]
```


Node核心模块

24

▶ Node核心模块

- ▶ 参考CommonJS规范的定义
- ▶ 底层由C/C++实现
- ▶ `require(模块名)`
- ▶ Node启动时预先加载到内存中
- ▶ 不需要通过路径搜索
- ▶ API文档: <http://nodejs.cn/doc/node/index.html>

Node自定义模块

25

- ▶ Node核心模块以外的模块，都是自定义模块
- ▶ 通过路径搜索动态加载
 - ▶ require(模块名)
 - ▶ require(相对路径)
 - ▶ require(绝对路径)
- ▶ 用JavaScript实现
 - ▶ exports
 - ▶ module
- ▶ 用C/C++扩展实现
 - ▶ 大部分是其他语言（主要是C/C++）实现的现有库的一个适配

Node包规范

26

- ▶ 参考CommonJS的包规范: <http://wiki.commonjs.org/wiki/Packages/1.0>
- ▶ 包描述文件
 - ▶ package.json
- ▶ 包格式
 - ▶ 压缩包格式
 - ▶ 文件夹形式
- ▶ 包目录结构
 - ▶ /package.json 包描述文件
 - ▶ /bin 二进制文件dll、so文件等
 - ▶ /文件名.js JavaScript源码, 主入口
 - ▶ /lib JavaScript源码
 - ▶ /doc 文档
 - ▶ /test 单元测试
 - ▶ /node_modules 依赖的自定义模块目录

电脑 > 本地磁盘 (D:) > nodejs > node_modules > npm > node_modules > tar >				
名称	修改日期	类型	大小	
examples	2016/2/28 1:48	文件夹		
lib	2016/2/28 1:48	文件夹		
node_modules	2016/2/28 1:48	文件夹		
test	2016/2/28 1:48	文件夹		
.npmignore	2016/1/16 7:25	NPMIGNORE 文件	1 KB	
.travis.yml	2016/2/9 13:03	YML 文件	1 KB	
LICENSE	2016/2/9 13:03	文件	1 KB	
package.json	2016/2/9 13:31	JSON File	2 KB	
README.md	2016/2/9 13:03	MD 文件	2 KB	
tar.js	2016/1/16 7:25	JavaScript 文件	5 KB	

Node包规范

27

► 包描述文件

```
16 {
17   "author": {"name": "Isaac Z. Schlueter"...},
23   "name": "tar",
24   "description": "tar for node",
25   "version": "2.2.1",
26   "repository": {"type": "git"...},
30   "main": "tar.js",
31   "scripts": {"test": "tap test/*.js"...},
34   "dependencies": {
35     "block-stream": "*",
36     "fstream": "~1.0.2",
37     "inherits": "2"
38   },
39   "devDependencies": {
40     "graceful-fs": "~4.1.2",
41     "rimraf": "1.x",
42     "tap": "0.x",
43     "mkdirp": "~0.5.0"
44   },
45   "license": "ISC",
46   "gitHead": "52237e39d2eb68d22a32d9a98f1d762189fe6a3d",
47   "bugs": {"url": "https://github.com/isaacs/node-tar/issues"...},
50   "homepage": "https://github.com/isaacs/node-tar#readme",
51   "_id": "tar@2.2.1",
52   "_shasum": "8e4d2a256c0e2185c6b18ad694aec968b83cb1d1",
53   "_from": "tar@>=2.2.1 <2.3.0",
54   "_npmVersion": "2.14.3",
55   "_nodeVersion": "2.2.2",
56   "_npmUser": {"name": "zkat"...},
60   "dist": {"shasum": "8e4d2a256c0e2185c6b18ad694aec968b83cb1d1"...},
64   "maintainers": [...],
82   "directories": {},
83   "_resolved": "https://registry.npmjs.org/tar/-/tar-2.2.1.tgz",
84   "readme": "ERROR: No README data found!"
85 }
```

NPM介绍

28

npm is the package manager for javascript.

- ▶ NPM官网: <https://www.npmjs.com/>
- ▶ NPM文档: <https://docs.npmjs.com/>
- ▶ NPM是JavaScript的包管理平台（不仅是Node！）
- ▶ NPM是一个开放的平台
- ▶ 采用CommonJS包规范
- ▶ NPM命令行工具
 - ▶ <https://docs.npmjs.com/cli/access>
 - ▶ 安装一个包: `npm install 包名`
 - ▶ 查看帮助: `npm -h`
- ▶ 不要信任所有的NPM包



248,809
total packages



64,420,957
downloads in the last day



907,978,345
downloads in the last week



3,636,573,548
downloads in the last month

习题

29

- ▶ 1. 创建一个模块
- ▶ 2. 将模块打包成一个包 *
- ▶ 3. 通过不同的require方式使用创建的模块

第三部分：Node核心API

30

- ▶ Console 控制台
- ▶ Buffer 缓冲区
- ▶ Events 事件处理
- ▶ File System 文件系统
- ▶ Net/Http 网络/Http
- ▶ Stream 流处理
- ▶ Child Processes/Process 子进程/进程
- ▶ Cluster 集群
- ▶ Errors 错误处理

