ajax预习课-node.js入门

珠峰培训

课程大纲

- 什么是node.js
- 为什么要学node.js
  - node越来越 **流行**
  - node越来越 **强大**
- 如何学习node.js
- node.js快速上手
  - 第一个node程序
  - REPL
  - 异步、事件驱动模型、非I/O阻塞等特性
  - 模块和包
  - 全局对象global
  - fs文件模块

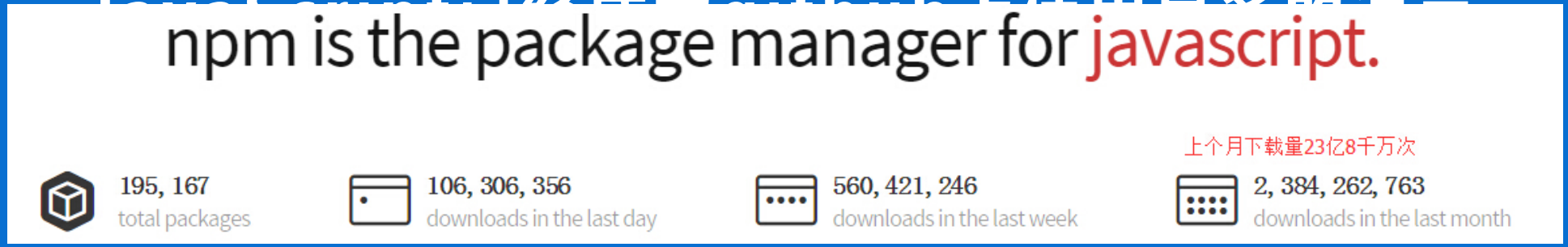
什么是node.js

- Node.js是JavaScript语言的服务器运行环境
- Node提供大量工具库，使得JavaScript可以调用操作系统级别的API
- Node内部采用Google公司的V8引擎，作为JavaScript语言解释器，速度非常快；
- Node.js是一个基于事件驱动和异步I/O的服务端JavaScript环境。

node.js的包管理系统已经成为世界上最大的开源库生态系统

npmjs官网

JavaScript已经成为github上使用最多的语言



珠峰培训
Everest training

- 项目管理：npm , grunt , gulp , bower , yeoman
- 桌面应用: node-webkit
- Web开发： express , ejs ,hexo, socket.io, restify, nodeppt , stylus, browserify , cheerio
- 数据库： mysql , mongoose , redis ,memcached
- 工具包: underscore, moment , connect ,later, log4js ,passport(oAuth),req
- 异步： async ,wind,eventProxy, bluebird
- 部署：forever, pm2 , nodemon
- 测试： jasmine , karma , protractor
- 跨平台：rio,tty
- 内核： cluster
- 模板: jade , ejs
- 博客: ghost,hexo

- 微信: wqui

学习资源

- node官网 关注Node版本更新，包括api功能及使用、bug修复、新增特性以及未来的发展趋势
- npm官网 可以搜索需要的模块，以及模块的使用说明,参考别人的源代码
- github
  在这里可以找到大量nodejs相关的项目，阅读源码，查看新技术的一手资料
- stackoverflow
  如果遇到解决不了的问题可以在此提问,会有很多热情的好友来帮你解答问题的，比如服务异常、配置什么的。

安装node

官方主页

Windows Installer	Macintosh Installer	Source Code
node-v4.2.1-x64.msi	node-v4.2.1.pkg	node-v4.2.1.tar.gz

Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.exe)	32-bit	64-bit	
Mac OS X Installer (.pkg)	64-bit		
Mac OS X Binaries (.tar.gz)	64-bit		
Linux Binaries (.tar.gz)	32-bit	64-bit	
SunOS Binaries (.tar.gz)	32-bit	64-bit	
ARM Binaries (.tar.gz)	ARMv6	ARMv7	ARMv8
Source Code	node-v4.2.1.tar.gz		

安装配置webstorm

WebStorm是开发javascript的IDE，并且支持流行的Node.js以及Angular和React等js框架。

webstorm下载

第一个node程序

- 先编写一个文件

```
console.log('zhufengpeixun');
```

- 将文件保存为 **zfpx.js**
- 打开命令行，进行 **zfpx.js** 所在的目录，执行以下命令:

```
node zfpx.js
```

- 如果一切正常，你会在命令行下面看到

```
zhufengpeixun
```

- console是node.js提供的 **控制台** 对象，其中包含了向标准输出写入的操作,跟浏览器的console功能类似。
- node是可执行程序，可以 **解释执行** 后面的脚本。

REPL

> （Read–eval–print loop，"读取-求值-输出"循环）

- 在命令行键入node命令，后面没有文件名，可以直接运行各种JavaScript命令。

```
node
5+5
```

- 特殊变量下划线（_）表示上一个命令的返回结果。

```
5+5
10
_+5
15
```

- 在REPL中，如果运行一个表达式，会直接在命令行返回结果。如果运行一条语句，就不会有任何输出，因为语句没有返回值。

```
var name='zfpx';
undefined
1+1
2
```

模块

> 每个 js文件都是一个模块，模块内部声明的变量都是 私有 变量，外部无法访问。

- 创建模块

  math.js

- 导出模块

```
exports.add = function(a,b){return a+b;}
```

- 加载模块

```
var math = require('./math');
```

- 调用模块

```
var sum = math.add(1,2);
```

模块的分类

- **核心** 模块

```
http fs path
```

- **文件** 模块

```
var math = require('./math');
```

- **第三方** 模块

```
var async = require('async');
```

包和npm

- **多个** 模块可以封装成一个 **包**
- npm是node.js默认的模块管理器,用来安装和管理node模块 网址为 `http://npmjs.org`
- 可以以包的方式通过 **npm** 安装、卸载、发布包

如何初始化一个项目

```
mkdir studynode    创建目录
cd studynode    进入目录
npm init   初始化项目描述文件
```

```
{
"name":"包的名称，必须是唯一的，由小写英文字母、数字和下划线组成，不能包含空格。",
"description": "包的简要说明。",
"version":"符合语义化八本识别规范的版本字符串。",
"keywords": "关键字数组，通常用于搜索。",
"maintainers":"维护者数组，每个元素要包含name、email（可选）、web（可选）字段。",
"contributors": "贡献者数组，格式与maintainers相同。包的作者应该是贡献者数组的第一个元素。",
"bugs": "提交bug的地址，可以是网址或者电子邮件地址。".,
"licenses":"许可证数组，每个元素要包含type(许可证的名称)和url(链接到许可证文本的地址)字段。",
"repositories": "仓库托管地址数组。每个元素要包含type(许可证的名称)和url(链接到许可证文本的地址)字段。
"dependencies":"包的依赖，一个关联数组，由包名称和版本组成。"
}
```

> 注意项目的名称不能是别人已经注册的名称

发布一个项目

```
npm adduser
Username: zhangrenyang
Password:
Email: (this IS public) zhang_renyang@126.com
npm publish
```

> 如果注册失败的话可能是因为改了镜像地址了，需要改回来 npm config set registry " http://registry.npmjs.org/ "

npm install(安装第三方模块)

- 全局安装 直接下载到Node的安装目录中， **各个** 项目都可以调用,适合工具模块，比如 `mime`

```
npm install -global [package name]
```

- 本地安装 将一个模块下载到 **当前** 目录的 **node_modules** 子目录，然后只有在 **当前** 目录和它的 **子** 目录之中，才能调用这个模块

```
npm install  [package name]
```

- 使用模块

```
var mime = require('mime');
```

全局对象global

`global` 表示Node所在的 **全局** 环境，类似于浏览器的 `window` 对象，它及其所有属性都可以在程序的 **任何** 地方访问。

console(控制台对象)

> 控制台在操作系统中的表现形式为一个操作系统中指定的字符界面，例如，在Windows操作系统中为一个命令提示窗口

- 向 **标准输出流** 打印字符并以换行符结束

```
console.log([data][, ...])
```

- 该命令的作用是返回 **信息性** 消息

```
console.info([data][, ...])
```

- 输出红色错误消息

```
console.error([data][, ...])
```

- 输出警告消息

```
console.warn([data][, ...])
```

- 输出时间，表示计时开始结束

```
console.time(label)
console.timeEnd(label)
```

fs(文件模块)

- readFileSync方法用于同步读取文件并返回一个字符串

```
var text = fs.readFileSync(fileName, "utf8");
```

- readFile方法用于异步读取文件。

```
fs.readFile(fileName, "utf8",function(err,text){});
```