01

初识Nodejs

- JavaScript是什么?
- JavaScript可以运行在哪里?
    - JavaScript一种在浏览器中解释运行的脚本语言，它的解释器被称为JavaScript引擎，为浏览器的一部分，是广泛用于

  客户端的脚本语言，最早是在HTML网页上使用，用来给HTML（HTML5）网页增加动态功能

浏览器	内核
IE	Trident
FireFox	Gecko
Chrome	WebKit\Bink
Safari	WebKit
Opera	Presto
Edge	Chakra

Node.js的诞生

- 作者Ryan Dahl 瑞恩·达尔
    - 2004 纽约 读数学博士
    - 2006 退学到智利 转向开发
    - 2009.5对外宣布node项目，年底js大会发表演讲
    - 2010 加入Joyent云计算公司
    - 2012 退居幕后

> 1.简单的说 Node.js 就是运行在服务端的 JavaScript。
>
> 2.Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。
>
> 3.Node.js是一个事件驱动I/O的服务端JavaScript环境（由C++编写），基于Google的V8引擎设计，V8引擎执行Javascript的速度非常快，性能非常好。

事件驱动

事件驱动模型主要包含3个对象：事件源、事件和事件处理程序。

· 事件源：产生事件的地方(html元素)

· 事件：点击/鼠标操作/键盘操作等等

· 事件对象：当某个事件发生时，可能会产生一个事件对象，该时间对象会封装好该时间的信息，传递给事件处理程序

· 事件处理程序：响应用户事件的代码
我们使用的window系统也算得上是事件驱动了。简单的事例：监听鼠标点击事件，并能够显示鼠标点击的位置x,y。

Node.js的工作原理

nodejs是单线程，异步I/O，事件驱动

1.node.js的单线程并不是真正的单线程，只是开启了单个线程进行业务处理（cpu的运算），同时开启了其他线程专门处理I/O。当一个指令到达主线程，主线程发现有I/O之后，直接把这个事件传给I/O线程，不会等待I/O结束后，再去处理下面的业务，而是拿到一个状态后立即往下走，这就是"单线程"、"异步I/O"。

2.Node.js的I/O 处理完之后会有一个回调事件，这个事件会放在一个事件处理队列里头，在进程启动时node会创建一个类似于While(true)的循环，它的每一次轮询都会去查看是否有事件需要处理，是否有事件关联的回调函数需要处理，如果有就处理，然后加入下一个轮询，如果没有就退出进程，这就是所谓的"事件驱动"。

3.在node.js中，事件主要来源于网络请求，文件I/O等，根据事件的不同对观察者进行了分类，有文件I/O观察者，网络I/O观察者。事件驱动是一个典型的生产者/消费者模型，请求到达观察者那里，事件循环从观察者进行消费，主线程就可以马不停蹄的只关注业务不用再去进行I/O等待。

Node.js可以用来做什么?

- 具有复杂逻辑的动态网站
- WebSocket服务器
- 命令行工具
- 带有图形界面的本地应用程序
- ……

Node.js开发环境准备

1. 普通安装方式官方网站
2. 配置环境变量

   （1）node.js的msi包是一路next就可以了

   （2）安装完后，可以在命令行中输入node -v 来查看安装版本和是否安装成功，再输入npm-v查看npm模块是否正常

   （3）配置npm的全局模块（新建文件夹node_global、node_cache）

   npm config set prefix "D:\nodejs\node_global" 》模块配置位置
   npm config set cache "D:\nodejs\node_cache" 》缓存文件

   （4）配置环境变量

   进入环境变量对话框，在【系统变量】下新建【NODE_PATH】，输入 D:\nodejs\node_global\node_modules

   将【用户变量】下的【Path】修改为 D:\nodejs\node_global

   (6) 配置完后，安装个module测试下，我们就安装最常用的express模块，打开cmd窗口输入命令，进行模块的全局安装

   npm install express -g　　# -g是全局安装的意思

服务器端模块化

- 服务器端模块化规范CommonJS与实现Node.js
- 模块导出与引入

  模块导出：exports、module

  模块引入：require()
- 模块导出机制分析

  exports导出调用时要调用导出方法

  module导出调用时直接使用方法
- 模块加载规则
  - 模块查找 不加扩展名的时候会按照如下后缀顺序进行查找 .js .json
- 模块分类
  - 自定义模块
  - 系统核心模块
    - fs 文件操作
    - http 网络操作
    - path 路径操作
    - querystring 查询参数解析
    - url url解析
    - ……

02

Buffer基本操作

> Buffer对象是Node处理二进制数据的一个接口。它是Node原生提供的全局对象，可以直接使用，不需要require('buffer')。

- 实例化
  - Buffer.from(array) 数组
  - Buffer.from(string) 字符串
  - Buffer.alloc(size) 字节长度
- 功能方法
  - Buffer.isEncoding() 判断是否支持该编码
  - Buffer.isBuffer() 判断是否为Buffer
  - Buffer.byteLength() 返回指定编码的字节长度，默认utf8
  - Buffer.concat() 将一组Buffer对象合并为一个Buffer对象
- 实例方法
  - write() 向buffer对象中写入内容
  - slice() 截取新的buffer对象
  - toString() 把buf对象转成字符串
  - toJson() 把buf对象转成json形式的字符串

核心模块API

路径操作

- 路径基本操作API

  path

文件操作(fs)

- 文件信息获取
- 读文件操作
- 写文件操作
- 目录操作

文件操作案例

包

> 多个模块可以形成包，不过要满足特定的规则才能形成规范的包

NPM （node.js package management）

> 全球最大的模块生态系统，里面所有的模块都是开源免费的；也是Node.js的包管理工具。

- 官方网站

npm包安装方式

- 本地安装
- 全局安装

解决npm安装包被墙的问题

- --registry
  - npm config set registry=https//registry.npm.taobao.org
- cnpm
  - 淘宝NPM镜像,与官方NPM的同步频率目前为10分钟一次
  - 官网: http://npm.taobao.org/
  - npm install -g cnpm –registry=https//registry.npm.taobao.org
  - 使用cnpm安装包: cnpm install 包名

npm常用命令

- 安装包
- 更新包
- 卸载包

yarn基本使用

- 类比npm基本使用

自定义包

包的规范

- package.json必须在包的顶层目录下
- 二进制文件应该在bin目录下
- JavaScript代码应该在lib目录下
- 文档应该在doc目录下
- 单元测试应该在test目录下

package.json字段分析

- 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：符合语义化版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，可以是网站或者电子邮件地址
- 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（仓库类型，如git）、url（仓- 库的地址）和path（相对于仓库的路径，可选）字段
- dependencies：生产环境包的依赖，一个关联数组，由包的名称和版本号组成
- devDependencies：开发环境包的依赖，一个关联数组，由包的名称和版本号组成

03

Web开发概述

Node.js实现静态网站功能

- 使用http模块初步实现服务器功能
- 实现静态服务器功能

参数传递与获取

- get参数获取
- post参数获取

动态网站开发

- 创建服务器实现动态网站效果

模板引擎

- 理解模板引擎本质
- 引擎基本使用

Express基本使用

- 静态服务器
- 路由
- 中间件　　next() 作用就是把请求传递到下一个 中间件（函数）
- 模板引擎整合 art-template 官网：http://aui.github.io/art-template/

- 常用API基本使用

04

Express基本使用

- 中间件
- 参数处理
- 模板引擎整合

Express图书管理案例

Node.js操作数据库

- Mysql环境准备

mysql第三方包基本使用

- 操作数据库基本步骤
- 实现增删改成基本操作

基于数据库实现登录功能

Express整合数据库实现增删改查业务

后台API开发

- 基于数据库的json接口开发
- 基于数据库的jsonp接口开发