* require引入模块

var http = require("http");

* 创建服务器

http.createServer() 方法创建服务器

listen 方法绑定 8888 端口

函数通过 request, response 参数来接收和响应数据

var http = require('http');

var fs = require('fs');

var url = require('url');

// 创建服务器

http.createServer( function (request, response) {

//获得请求文件.html的文件名

var pathname = url.parse(request.url).pathname;

console.log("Request for " + pathname + " received.");

// 从文件系统中读取请求的文件内容

fs.readFile(pathname.substr(1), function (err, data) {

if (err) {

console.log(err);

// HTTP 状态码: 404 : NOT FOUND

// Content Type: text/plain

response.writeHead(404, {'Content-Type': 'text/html'});

}else{

// HTTP 状态码: 200 : OK

// Content Type: text/plain

response.writeHead(200, {'Content-Type': 'text/html'});

// 响应文件内容

response.write(data.toString());

}

// 发送给前端

response.end();

});

}).listen(8080);

// 控制台会输出以下信息

console.log('Server running at <http://127.0.0.1:8080/>');

node server.js

显示Server running at http://127.0.0.1:8888/

node命令

客户端

var http = require('http');

// 用于请求的选项

var options = {

host: 'localhost',

port: '8080',

path: '/index.html'

};

// 处理响应的回调函数

var callback = function(response){

// 不断更新数据

var body = '';

response.on('data', function(data) {

body += data;

});

response.on('end', function() {

// 数据接收完成

console.log(body);

});

}

// 向服务端发送请求

var req = http.request(options, callback);

req.end();

使用npm安装模块

$ npm install <Module Name>

全局安装与本地安装

npm install express # 本地安装

npm install express -g # 全局安装

* 本地安装

1. 将安装包放在 ./node\_modules 下（运行 npm 命令时所在的目录），如果没有 node\_modules 目录，会在当前执行 npm 命令的目录下生成 node\_modules 目录。

2. 通过 require() 来引入本地安装的包。

就是你哪个项目用这个包，就install在哪儿

var express = require('express');

* 全局安装

1. 将安装包放在 /usr/local 下或者你 node 的安装目录。

找了半个小时发现在"C:\Users\as\AppData\Roaming\npm\node\_modules\express"

2. 可以直接在命令行里使用。

删除全局模块

npm uninstall -g

删除本地模块

npm uninstall 模块

node

启动Node终端

然后变为REPL 的命令行窗口

>

REPL(Read Eval Print Loop:交互式解释器)

用起来感觉就是console，能输入表达式，返回结果，也能输入一串代码

ctrl+double click 'c'退出

回调函数

执行异步操作的函数将回调函数作为最后一个参数，回调函数接收错误对象作为第一个参数。

var fs = require("fs");

//一个异步的函数，最后一个参数是回调函数

fs.readFile('input.txt', function (err, data) {

if (err){

// 回调函数接收错误对象作为第一个参数，如果error了，则进入if，然后return

console.log(err.stack);

return;

}

console.log(data.toString());//如果接受数据正常，则执行后边的

});

引入 events 模块

var events = require('events');

创建 eventEmitter 对象

var eventEmitter = new events.EventEmitter();

events模块只有一个EventEmitter类

封装了事件触发与事件监听器功能

eventEmitter.on('some\_event', function(){

console.log('event trigger!');

})

eventEmitter.on()绑定一个事件和事件处理函数，相当于注册了一个事件监听器

创建事件也不知不觉完成了

eventEmitter.emit('some\_event')

eventEmitter.emit()触发some\_event事件

对于每个事件，EventEmitter 支持若干个事件监听器。

当事件触发时，注册到这个事件的事件监听器被依次调用

emit可以传多个参数，后边的参数作为回调函数参数（事件处理函数）传递。

emitter.on('some\_event2', function(arg1, arg2) {

console.log('listener2', arg1, arg2);

});

emitter.emit('someEvent', '66666', '7777777');