复习：

Node.js适合于IO密集型应用，不适合于CPU密集型应用。

JS和Node.js区别：

JS运行于客户端浏览器中，存在兼容性问题；数据类型：值类型+引用类型(ES+DOM/BOM+自定义)

Node.js运行于服务器端(V8引擎)，不存在兼容性问题；数据类型：值类型+引用类型(ES+扩展对象+自定义)。

var s1 = {age:10};

var s2 = {age:20};

s2.age++; //s1.age:10 s2.age:21

var s3 = s1; //让s3也指向对象{age:10}

s3.age++; //s1.age:11 s3.age:11

s3 = {age:30} //s1.age:11 s3.age:30

Node.js模块系统：

Node.js中每个.js文件都是一个“Module”，每个模块都可以引入其它模块；也可以导出自己的成员供其它模块来使用。

(function(exports, require, module, \_\_filename, \_\_dirname){

exports = module.exports; //{ }

var age=20;

var fn = function(){}

//exports.age = 20;

//exports.f = fn;

//module.exports.age = 20;

//module.exports.f = fn;

//exports = { age: age, f: fn }; //无效语句

module.exports = {age: age, f: fn} //有效导出语句

})

Node.js中模块的类型：

(1)官方提供的原生模块

global、util、url、fs、http

(2)第三方模块

mysql、oracle、express...

(3)自定义模块

文件模块和目录模块

今日目标：

(1)模块和NPM —— 有些乱但了解即可

(2)常用的原生模块 —— 重点&难点

1.自定义模块的两种形式

**(1)文件模块**

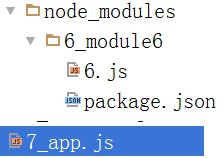
创建一个js文件，如m3.js，导出需要公开的数据。其它模块可以require('./m3')模块

**(2)目录模块**

方式1：创建一个目录，假设名为m4，其中创建名为index.js的文件，导出需要公开的数据。其它模块可以require('./m4')模块

方式2：创建一个目录，假设名为m5，其中创建package.json文件，其中声明main属性指定默认执行的启动JS文件，如5.js，其中导出需要公开的数据。其它模块可以require('./m5')模块

方式3：创建一个目录，必须名为**node\_modules**，其中再创建目录模块，假设名为m6，其中创建package.json文件，其中声明main属性指定默认执行的启动JS文件，如6.js，其中导出需要公开的数据。其它模块可以require('m6')模块



练习：使用方式3，创建两个目录模块，circle、rectangle，都对外公开两个方法size()、perimeter()，返回指定图形的面积和周长。

最后在最外层的主模块中引入上述两个模块：

circle.size( r ) circle.perimeter( r )

rectange.size( w, h ) rectange.perimeter( w, h )

|  |
| --- |
| 自己尝试：若有个文件名为m7.js，还有个目录m7/index.js，还有个node\_modules/m7/index.js；最后app.js中 ：  require('./m7') 引入的是哪个模块？  require('m7') 引入的是哪个模块？ |

2.NPM包管理器

Node Package Manager：Node.js的第三方模块/包管理器，可用于下载、更新、删除、维护包依赖关系的工具。

npm工具默认到www.npmjs.org网站下载所需的第三方模块包。

使用NPM工具下载一个新的软件包：

npm install 包名

npm uninstall 包名

更多的NPM命令参数可以使用 npm -h 进行查看。

3.Node.js官方提供的原生模块 —— querystring

querystring模块用于处理HTTP请求URL中的查询字符串

var obj = qs.parse(str) 把查询字符串解析为JS对象

var str = qs.stringify(obj) 把JS对象转换为查询字符串

4.Node.js官方提供的原生模块 —— url

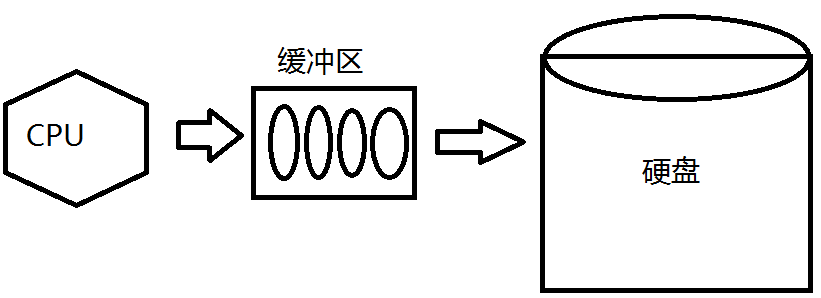
url模块用于解析一个HTTP请求地址，获取其中各个不同的部分

var obj = url.parse( str ) 把一个URL字符串解析为一个对象

var obj = url.parse( str, true ) 把一个URL字符串解析为一个对象，并把其中的查询字符串也解析为对象

5. Node.js官方提供的原生模块 —— Buffer

Buffer：缓冲区，本质是一块内存区域，用于暂存以后要用到的数据（可能是数字、字符串、二进制图片/音视频等），该区域就称为“缓存”。



//分配一个指定大小的缓冲区

var buf1 = Buffer.alloc( 1024 );

//使用一个数字数组创建一个缓冲区

var buf2 = Buffer.from( [1, 3, 5] );

//使用一个字符串创建一个缓冲区

var buf3 = Buffer.from( 'abcdefg' );

//把一个缓冲区中的数据转换为字符串

var str = buf3.toString( );

6. Node.js官方提供的原生模块 —— fs —— 重点

fs模块提供了对文件系统中的文件/目录进行增删改查、读写的功能。

//同步读取文件中的内容

var data = fs.readFileSync( file );

//同步向文件中写出内容（删除已有内容）

fs.writeFileSync( file, str/buf )

//同步向文件中追加写出内容（不删除已有内容）

fs.appendFileSync( file, str/buf )

练习：使用上述方法，实现把 4.css 文件复制为 4.backup.css

//异步读取文件中的内容

fs.readFile( file, function(err, data){ } );

//异步向文件中写出内容（删除已有内容）

fs.writeFile( file, str/buf, function(err){ } )

练习：使用异步方法，实现把 4.css 文件复制为 44.backup.css

提示：后续的项目中，文件读写可以使用同步或异步，但推荐使用异步方法——最大限度的发挥Node.js的优势。

7. Node.js官方提供的原生模块 —— http —— 重点

HTTP模块可用于编写基于HTTP协议的客户端程序（即浏览器）；也可以编写基于HTTP协议的服务器端程序（即Web服务器）

用http模块编写一个Web服务器：

(1)接收客户端的HTTP请求消息

(2)解析客户端请求消息

(3)读取客户端请求的文件

(4)向客户端发送HTTP响应消息，主体就是客户端请求的文件

var server = http.createServer( );

server.listen(80);

server.on('request', function(req, res){

//解析请求消息

//向客户端写出响应消息

})

今日练习：

使用Node.js创建一个Web 服务器，根据客户端请求的地址的不同，输出不同的HTML页面内容，如

http://127.0.0.1/login.html

服务器应该返回./public/login.html中的内容；

http://127.0.0.1/news.html

服务器应该返回./public/news.html中的内容；

若请求资源在public目录下不存在，则返回404的响应消息。