**1.命令行窗口**

**又叫(小黑屏)、CMD窗口、终端、shell**

- 开始菜单 --> 运行 --> CMD --> 回车

- 常用的指令：

dir 列出当前目录下的所有文件

cd 目录名 进入到指定的目录

md 目录名 创建一个文件夹

rd 目录名 删除一个文件夹

- 目录

. 表示当前目录

.. 表示上一级目录

- 环境变量（windows系统中变量）

path

C:\work\jdk\jdk1.7.0\_75/bin;

%CATALINA\_HOME%/bin;

C:\work\soft\tools\AppServ\Apache24\bin;

C:\work\soft\tools\AppServ\php5;

C:\Users\lilichao\AppData\Local\Programs\Fiddler;

C:\work\environment\Egret\Egret Wing 3\bin;

C:\Users\lilichao\AppData\Roaming\npm;

C:\Program Files\MongoDB\Server\3.2\bin;

C:\Users\lilichao\Desktop\hello

- 当我们在命令行窗口打开一个文件，或调用一个程序时，

系统会首先在当前目录下寻找文件程序，如果找到了则直接打开

如果没有找到则会依次到环境变量path的路径中寻找，直到找到为止

如果没找到则报错

- 所以我们可以将一些经常需要访问的程序和文件的路径添加到path中，

这样我们就可以在任意位置来访问这些文件和程序了

**2.I/O (Input/Output)**

- I/O操作指的是对磁盘的读写操作

**3.Node**

- Node是对ES标准一个实现，Node也是一个JS引擎

- 通过Node可以使js代码在服务器端执行

- Node仅仅对ES标准进行了实现，所以在Node中不包含DOM 和 BOM

- Node中可以使用所有的内建对象

String Number Boolean Math Date RegExp Function Object Array

而BOM和DOM都不能使用

但是可以使用 console 也可以使用定时器（setTimeout() setInterval()）

- Node可以在后台来编写服务器

Node编写服务器都是单线程的服务器

- 进程

- 进程就是一个一个的工作计划（工厂中的车间）

- 线程

- 线程是计算机最小的运算单位（工厂中的工人）

线程是干活的

- 传统的服务器都是多线程的

- 每进来一个请求，就创建一个线程去处理请求

- Node的服务器单线程的

- Node处理请求时是单线程，但是在后台拥有一个I/O线程池

**4.模块化**

- 在Node中，一个js文件就是一个模块

- 在Node中，每一个js文件中的js代码都是独立运行在一个函数中

而不是全局作用域，所以一个模块的中的变量和函数在其他模块中无法访问

- 我们可以通过 exports 来向外部暴露变量和方法

- 只需要将需要暴露给外部的变量或方法设置为exports的属性即可

**5.引入其他的模块**

在node中，通过require()函数来引入外部的模块

require()可以传递一个文件的路径作为参数，node将会自动根据该路径来引入外部模块

这里路径，如果使用相对路径，必须以.或..开头

使用require()引入模块以后，该函数会返回一个对象，这个对象代表的是引入的模块

我们使用require()引入外部模块时，使用的就是模块标识，我们可以通过模块标识来找到指定的模块

- 模块分成两大类

核心模块

- 由node引擎提供的模块

- 核心模块的标识就是，模块的名字

文件模块

- 由用户自己创建的模块

- 文件模块的标识就是文件的路径（绝对路径，相对路径）

相对路径使用.或..开头

**6.Node运行js文件机制**

在node中有一个全局对象 global，它的作用和网页中window类似

在全局中创建的变量都会作为global的属性保存

在全局中创建的函数都会作为global的方法保存

当node在执行模块中的代码时，它会首先在代码的最顶部，添加如下代码

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

在代码的最底部，添加如下代码

}

实际上模块中的代码都是包装在一个函数中执行的，并且在函数执行时，同时传递进了5个实参

**exports**

- 该对象用来将变量或函数暴露到外部

**require**

- 函数，用来引入外部的模块

**module**

- module代表的是当前模块本身

- exports就是module的属性

- 既可以使用 exports 导出，也可以使用module.exports导出

**\_\_filename**

C:\Users\lilichao\WebstormProjects\class0705\01.node\04.module.js

- 当前模块的完整路径

**\_\_dirname**

C:\Users\lilichao\WebstormProjects\class0705\01.node

- 当前模块所在文件夹的完整路径

**7.暴露变量和方法**

exports 和 module.exports

- 通过exports只能使用.的方式来向外暴露内部变量

exports.xxx = xxx

- 而module.exports既可以通过.的形式，也可以直接赋值

module.exports.xxx = xxxx

module.exports = {}

**8.包**

CommonJS的包规范允许我们将一组相关的模块组合到一起，形成一组完整的工具。

CommonJS的包规范由包结构和包描述文件两个部分组成。

包结构

–用于组织包中的各种文件

包描述文件

–描述包的相关信息，以供外部读取分析

**9.包结构**

包实际上就是一个压缩文件，解压以后还原为目录。符合规范的目录，应该包含如下文件：

–package.json 描述文件

–bin 可执行二进制文件

–lib js代码

–doc文档

–test单元测试包结构

**10.包描述文件**

包描述文件用于表达非代码相关的信息，它是一个JSON格式的文件–package.json，位于包的根目录下，是包的重要组成部分。

**package.json中的字段**

–name名字、description描述、version版本、keywords关键字、maintainers主要作者、contributors贡献者、bugs提交bugs的地址、licenses协议版权、repositories仓库、dependencies依赖、homepage主页、os系统、cpu、engine引擎、builtin、directories、implements、scripts、author、bin、main、devDependencies。

任何的json文件中不能写注释