Node的模块

为了让Node.js的文件可以相互调用，Node.js提供了一个简单的模块系统。 模块是Node.js 应用程序的基本组成部分，文件和模块是一一对应的。换言之，一个 Node.js 文件就是一个模块，这个文件可能是JavaScript 代码、JSON 或者编译过的C/C++ 扩展。

创建模块

在 Node.js 中，创建一个模块非常简单，如下我们创建一个 'main.js' 文件，代码如下:

```
var hello = require('./hello');
hello.world();
```

以上实例中，代码 require('./hello') 引入了当前目录下的hello.js文件（./ 为当前目录，node.js默认后缀为js）。 Node.js 提供了exports 和 require 两个对象，其中 exports 是模块公开的接口，require 用于从外部获取一个模块的接口，即所获取模块的 exports 对象。 接下来我们就来创建hello.js文件，代码如下:

```
exports.world = function() {
  console.log('Hello World');
}
```

在以上示例中，hello.js 通过 exports 对象把 world 作为模块的访问接口，在 main.js 中通过 require('./hello') 加载这个模块，然后就可以直接访 问 hello.js 中 exports 对象的成员函数了。有时候我们只是想把一个对象封装到模块中，格式如下:

```
module.exports = function() {
  // ...
}
```

文件夹模块

除了通过文件定义模块外，也可以通过文件夹定义模块来包含一组程序和库。

文件夹模块的模块定义通过下面方法:

- 文件夹下面定义一个package.jon，来指定入口文件
- 文件夹下面的index.js

---

npm

NPM是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种:

- 允许用户从NPM服务器下载别人编写的第三方包到本地使用。
- 允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。
- 允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。 过输入 "npm -v" 来测试是否成功安装。

使用 npm 命令安装模块

npm 安装 Node.js 模块语法格式如下:

```
npm install supervisor -g
```

我们使用 npm 命令安装常用的 Node.js web框架模块 express:

```
$ npm install express
```

安装好之后，express 包就放在了工程目录下的 node_modules 目录中，因此在代码中只需要通过 require('express') 的方式就好，无需指定第三方包路径。

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

全局安装与本地安装

npm 的包安装分为本地安装（local）、全局安装（global）两种

```
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. 可以通过 require() 来引入本地安装的包。

全局安装

1. 将安装包放在 /usr/local 下或者你 node 的安装目录。

2. 可以直接在命令行里使用。

你可以使用以下命令来查看所有全局安装的模块：

```
$ npm ls -g
```

Package.json 属性说明

- name - 包名。
- version - 包的版本号。
- description - 包的描述。
- homepage - 包的官网 url 。
- author - 包的作者姓名。
- contributors - 包的其他贡献者姓名。
- dependencies - 依赖包列表。如果依赖包没有安装，npm 会自动将依赖包安装在 node_module 目录下。
- repository - 包代码存放的地方的类型，可以是 git 或 svn，git 可在 Github 上。
- main - main 字段是一个模块ID，它是一个指向你程序的主要项目。就是说，如果你包的名字叫 express，然后用户安装它，然后require("express")。
- keywords - 关键字

卸载模块

我们可以使用以下命令来卸载 Node.js 模块。

```
$ npm uninstall express
```

卸载后，你可以到 /node_modules/ 目录下查看包是否还存在，或者使用以下命令查看：

```
$ npm ls
```

更新模块

我们可以使用以下命令更新模块：

```
$ npm update express
```

创建模块

```
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (node_modules) runoob                    # 模块名
version: (1.0.0)
description: Node.js 测试模块(www.runoob.com)  # 描述
entry point: (index.js)
test command: make test
git repository: https://github.com/runoob/runoob.git  # Github 地址
keywords:
author:
license: (ISC)
About to write to ……/node_modules/package.json:      # 生成地址

{
  "name": "runoob",
  "version": "1.0.0",
  "description": "Node.js 测试模块(www.runoob.com)",
  ……
}


Is this ok? (yes) yes
```

以上的信息，你需要根据你自己的情况输入。在最后输入 "yes" 后会生成 package.json 文件。 接下来我们可以使用以下命令在 npm 资源库中注册用户（使用邮箱注册）：

```
$ npm adduser
Username: mcmohd
Password:
Email: (this IS public) mcmohd@gmail.com
```

接下来我们就用以下命令来发布模块:

```
$ npm publish
```

如果你以上的步骤都操作正确，你就可以跟其他模块一样使用 npm 来安装。

其他一些NPM命令

- npm view moduleNames--查看node模块的package.json文件夹
- npm list --查看当前目录下已安装的node包
- npm search packageName--发布一个npm包的时候，需要检验某个包名是否已存在
- npm root：查看当前包的安装路径
- npm root -g：查看全局的包的安装路径

官方文档

node－inspector