LUO JIN 2020-05-13 (WED)

1. Node概述

1.1 为什么要学习服务器开发基础

- 能够和后端更加紧密的配合
- 网站业务逻辑前置，学习前端技术需要后端技术的支撑（Ajax）
- 扩展视野，站在更高的角度上看东西

1.2 服务器开发要做的事情

- 实现网站的业务逻辑：常见的登陆，输入完用户名和密码登陆之后，服务器要看当前在网站有没有注册过，如果注册过了，输入的信息完全正确的话，就告诉他登陆成功了
- 基于数据的增删改查

1.3 为什么学习Node

- 使用js语法开发后端应用
- 一些公司要求前端工程师掌握Node开发
- 生态系统活越，有大量开源库可以使用，做文件上传功能，直接拿来用，不用自己做
- 前端开发工具大多是基于Node

1.4 Node是什么？

- 基于谷歌v8引擎的JavaScript代码运行环境
- 浏览器可以运行JS代码，浏览器事JS的运行环境
- Node也可以，因为他包含 chrome的V8引擎

2. Node 运行环境安装

2.1 版本

10.13.0 LTS long term Support 长期支持的版本 稳定版

11.1.0Current 最新版，实验版，不稳定

去官网下载安装

2.2 node安装失败的话

1）. 错误代号2505/2503

原因：系统权限不足

解决： 1）管理员身份运行powershell

2）输入安装命令 **msiexec /package node**安装包位置+安装包的名字

2）执行命令报错

原因：Node安装目录写入环境变量中失败

解决：将node安装目录添加到环境变量中

.

命令行： 一般先在命令行当前目录中找，找不到回去系统变量Path中的找

3. Nodejs.快速入门

3.1 组成

JS组成： ECMAScript，Dom，Bom

Node.js：由 ECMAScript和node环境提供的一些附加API，包括文件，网路，路径等等一些更加强大的API

3.2 nodejs基础语法

ECMAScript,的左右语法都可以使用

1. node.js的模块化开发

1.1 JS开发弊端

文件依赖关系不明确：文件引入有先后顺序

命名冲突：本身不严格，同名变量不会报错会覆盖掉前边的

1.2 模块化开发

生活中：电脑的组装，某一个模块坏了，只需换坏了的就可以

软件中：一个功能就是一个模块，多个谋爱组成完整应用，抽离某一个模块不回影响其他功能的运行

1.3 模块化开发规范

一个js文件就是一个模块，模块内部定义的变量和函数，在默认情况外无法得到

模块内可以使用exports对象进行成员到此处，使用require方法导入其他模块

导出；exports.函数方法/变量=函数方法/变量；　exports.add=add；　模块的后缀名可以省略

引入：const a = (A模块 =) require（'./A模块.js'）；A.函数方法 a.add(10,20);

1.4 模块化开发导出的另一种方式

① **module.eports.**version= version

② module.eports. / eports 默认执行同一个对象，指向同一地址空间，eports是 module.eports.的别名（引用地址空间）；

③ module.eports = { name: 'luojin'} module.eports.重新指向一个对象时候，他就会指向别的，不会指向同一个了。**所以都是以 module.eports.为准的。给.eports重新指定一个对象是不对的**。当两个不指向同一对象的时候，以 module.eports.为准

3. 系统模块

3.1 什么是系统模块

Nodejs运行环境提供的API：文件模块(fs)→读取文件，写入文件，创建文件夹

3.2 文件操作

fs = file system

Const fs = require('fs');

3.2.1 文件读取

Fs.readFile('文件路径/文件名'【，文件编码】，callback);

回调函数：硬盘读取文件内容，需要花时间，当文件读完之后，硬盘会说读完了，获取文件读取的结果

读取文件语法实例

```
// 读取上一级css目录下中的base.css
fs.readFile('../css/base.css', 'utf-8' (err, doc) => {
  // 如果文件读取发生错误 参数err的值为错误对象 否则err的值为null
  // doc参数为文件内容
  if (err == null) {
    // 在控制台中输出文件内容
     console.log(doc);
   }
});
```

// 1，通过模块的名字fs对模块进行引用

const fs = require('fs');

// 2，通过模块内部的readFile读取文件内容

fs.readFile('./01.hello.js', 'utf-8', (error, doc) => { .....})

3.2.2 写入文件内容

Fs.writeFile('文件路径/文件名'，'数据'，callback)

```
const content = '<h3>正在使用fs.writeFile写入文件内容</h3>';
fs.writeFile('../index.html', content, err => {
  if (err != null) {
      console.log(err);
      return;
  }
  console.log('文件写入成功');
});
```

3.3 path 路径操作

为什么要进行路径拼接：

- 不同操作系统的路径分隔符不统一
- /public/uploads/avatar
- Windows上是 ₩ /

- Linux 上是 / （linux一般被用作网站服务器）

3.4 路径拼接：

```
path.join('路径', '路径', ...)
```

```
//导入path模块
const path = require('path');
//路径拼接
let finialPath = path.join('itcast', 'a', 'b', 'c.css')
//输出结果 itcast\a\b\c.css
console.log(finialPath);
```

3.5 相对路径 vs 绝对路径

1）大多情况使用绝对路径，因为相对路径相对的都是**命令行**的当前目录

2）在文件读取或者设置文件路径是**都会选择绝对路径**

如果命令行工具的当前目录和要执行的文件在同一目录，执行成功；如果不在同一目录，则会执行失败

3）使用_dirname获取当前文件所在的绝对路径

Path.join(_dirname,"01.hello.js")

4）require 方法相对的就是当前文件，这个可以也相对路径

4. 第三方模块

4.1 什么是第三方模块

别人写好的，具有特定功能的，可以直接拿来用=包=第三方模块，由于很多文件放在统一个文件中。

1）以js文件形式存在，提供实现项目具体功能的API接口

2）以命令行工具形式存在，辅助项目开发

4.2 获取第三方模块

npmjs.com 第三方模块的存储和分发仓库

npm = node package manager：node第三方模块管理工具

下载：npm install 想安装的模块名->npm install formidable 文件上传的模块

默认下载到了命令行的当前目录：node_modules下边就有，package.json也会下载出来

删除：npn uninstall package 想要卸载的模块名 -> npn uninstall formiable

全局安装和本地安装：

- 命令行工具：全局安装：让左右项目都能使用
- 库文件：本地安装

4.3 第三方模块 nodemon

Nodemon是一个命令行工具，辅助项目开发

每次修改文件都要在命令行工具执行该文件，非常繁琐，每次保存都要在命令行重新执行

1）下载：npm install nodemon **-g**

2）在命令行中使用nodemon命令代替node 执行文件

3）命令行工具别挂起，会监视该文件的修改，保存，然后自动执行

4) ctrl + C 终止操作

4.4 第三方模块 nrm

Nrm(npm registry manager)：下载地址切换工具

Npm 默认的下载地址在国外，国内下载速度慢

国外：npmjs.com –> 国内：npm.taobao.org –>开发者

1） 下载：npn install nrm -g

2） 查询可用的下载地址列表 nrm ls

3） 切换npm下载地址 nrm use Taobao

4） Nrm ls

5） Npm install gulp