第14期课堂笔记(第七周第二天)

NODE

是一个以webkit(V8)引擎来渲染和解析JS的平台

> - 在没有NODE之前，JS是一门基于客户端浏览器运行的脚本编程语言；JS是前端编程语言；
> - 在有了NODE之后，JS也可以运行在NODE环境中了，我们可以把NODE安装在服务器端，这样我们就可以在服务器端执行JS，JS也可以处理服务器端的业务逻辑了；JS是后台编程语言；
>
> JS是全栈编程语言

JS运行在浏览器中，浏览器提供了很多内置的属性和方法(window全局对象)；JS如果运行在NODE环境下，它也提供了很多的属性和方法(global全局对象)；

如何在NODE环境下执行JS

- 在WB的指定JS文件中，通过鼠标右键 Run xxx.js 来运行JS代码(这样是把JS在NODE环境中执行了)；这种方式有些时候会出现缓存,尤其是把执行的文件换目录后，缓存出现的几率特别大。

- 找到当前需要执行的JS所在文件目录，在这个目录中打开DOS窗口，在窗口中输入 node xxx.js 的命令把对应JS执行；

传统后台语言和NODE的对比

传统后台语言：JAVA、PHP、C#、.NET…
NODE作为后台运行的环境,拿JS编程的优势：

- 快：基于V8引擎渲染
- 无阻塞的I/O操作(异步对文件进行增删改查)
- 基于事件驱动的单线程异步编程

NODE中的模块

node把所有的js按照模块进行划分，分为：

- 内置模块：node平台天生提供的模块
- 自定义模块：自己写的模块
- 第三方模块：别人写的我们调取使用

第三方模块

1、下载安装第三方模块

> 所有的第三方模块都在npmjs.com上
> 在命令窗口中执行 “npm install 模块名” 来安装，把需要使用的第三方模块安装到项目目录下
> npm就是NODE中用来管理模块的命令，npm install 是安装，npm uninstall 是卸载…

2、导入

> require
> 例如：var less = require(‘less’);

3、使用

> less.render()

4、如果我们想把LESS等第三方模块添加到命令行中，我们需要把LESS安装在全局NODE环境下；

> npm install less -g
> 安装成功后执行 lessc -v ，可以查看到版本号

自定义模块

在NODE环境下，我们创建的每一个JS都可以理解为一个单独的模块，模块和模块之间没有冲突。我们经常需要实现模块和模块之间的相互调用(例如B调取A中的fn方法)：

- 首先在B模块中导入A模块：var A=require('./A'); 即使在同一级目录下，我们也需要加./，如果不加的话默认找的是node_modules下的模块，而不是我们自定义的模块
- 其次还需要在A模块中把供外面使用的方法暴露出来：module.exports={fn:fn};
- 最后在B模块中就可以通过使用：A.fn() 执行对应的方法了

A.js

```
function fn() {
    console.log(1);
}

module.exports = {
    fn: fn
};
```

B.js

```
function fn() {
    console.log(2);
}

var A = require('./A');
A.fn();
```

思考题：创建三个自定义模块，A/B/C，A模块中有一个方法”任意数求和sum”，B模块中有一个方法“求平均数avg”，在B中需要调取A中的sum方法，在C模块中点去B模块中的avg方法，实现获取：98 95 92 96 95 94 92 98 93 90 这十个分数的平均数…

A.js

```
function sum() {
    var total = null;
    arguments.__proto__ = Array.prototype;
    arguments.forEach(function (item, index) {
        item = Number(item);
        !isNaN(item) ? total += item : null;
    });
    return total;
}
module.exports = {
    sum: sum
};
```

B.js

```
var a = require('./A');
function avg() {
    arguments.__proto__ = Array.prototype;
    arguments.sort(function (cur, next) {
        return cur - next;
    });
    arguments.pop();
    arguments.shift();
    //a.sum(arguments) -> a.sum([98,95,96...]) 目标:a.sum(98,95,96...)
    return (a.sum.apply(null, arguments) / arguments.length).toFixed(2);
}
module.exports = {
    avg: avg
};
```

C.js

```
var b = require('./B');
console.log(b.avg(98, 95, 92, 96, 95, 94, 92, 98, 93, 90));
```

内置模块

node天生提供好的模块，node主要应用于服务器端开发，理解服务器端需要做的事情，我们也就理解node内置模块了。

> 服务器端需要做的事情
>
> - 创建服务，监听端口号
> - 接收和解析客户端的请求
> - 在服务器上把客户端需要的内容找到(文件内容的读取)
> - 把找到的内容返回
>
> 常用的内置模块：http、url、fs…

1、http
创建服务、监听端口、接收信息、返回内容…

> var server1=http.createServer([callback]); 创建一个服务，这块的回调函数不是在服务创建成功就会执行，而是需要客户端向当前的服务器发送请求，它才会执行（客户端只要发送一个请求，它就会被触发一次执行）。
>
> server1.listen([port],[callback]); 给创建的服务监听端口号(0-65535之间)，端口号不能重复，当服务创建成功端口号也监听成功后就会执行对应的回调函数；

```
"D:\WebStorm 10.0.3\bin\runnerw.exe" "C:\Program Files\nodejs\node.exe" server.js
events.js:160
      throw er; // Unhandled 'error' event
      ^

Error: listen EACCES 0.0.0.0:80
    at Object.exports._errnoException (util.js:1022:11)
    at exports._exceptionWithHostPort (util.js:1045:20)
    at Server._listen2 (net.js:1249:19)
    at listen (net.js:1298:10)
    at Server.listen (net.js:1376:9)
    at Object.<anonymous> (e:\前端课程\js正式\7.第七周\2.第二天\js\server.js:8:9)
    at Module._compile (module.js:573:32)
    at Object.Module._extensions..js (module.js:582:10)
    at Module.load (module.js:490:32)
    at tryModuleLoad (module.js:449:12)

Process finished with exit code 1
```

这个报错说明当前80端口被电脑上的其他应用给占用了，我们需要换一个端口号。
启动服务后，服务在不出现意外或者错误的情况下是不会停止的，随时监听客户端的请求，完成我们的响应操作。

2、服务创建成功后如何的向当前服务发送请求

- 如果服务在本地，我们可以在浏览器地址栏中输入：http://localhost:80/….来访问
- 通过主机的IP地址或者域名来访问，例如：http://192.168.1.107:80/….来访问

3、url模块

> var url=require(‘url’);
> url.parse([string],[boolean]) : 解析一个URL地址，可以把地址中的每一部分分别获取到
>
> - [string] 要解析的URL地址字符串
> - [boolean] 设定是否把问号传参的值解析为对象，默认是false不解析，写true则为解析

案例：

```
var url = require('url');
var str = 'http://www.zhufengpeixun.com:80/student/index.html?name=zxt&age=30#haha';
var result = url.parse(str);
console.log(result);

//->结果如下
{
  protocol: 'http:',   //->协议
  slashes: true, //->是否有斜线
  auth: null, //->作者
  host: 'www.zhufengpeixun.com:80', //->域名+端口
  port: '80', //->端口
  hostname: 'www.zhufengpeixun.com', //->域名
  hash: '#haha', //->哈希值
  search: '?name=zxt&age=30', //->问号传参
  query: 'name=zxt&age=30', //->问号传参,不带问号
  pathname: '/student/index.html', //->路径名称
  path: '/student/index.html?name=zxt&age=30', //->路径名称+问号传参
  href: 'http://www.zhufengpeixun.com:80/student/index.html?name=zxt&ag
```

```
    e=30#haha' //->原始字符串
20. }
```

如果第二个参数设为true的话：

```
1. {
2.     ...
3.     query: { name: 'zxt', age: '30' },
4.     pathname: '/student/index.html'
5.     ...
6. }
7. //->query是以对象键值对的方式存储的
```

4、fs模块

实现对服务器上的文件进行I/O操作的

var fs=require(‘fs’);

- fs.readFileSync 同步读取文件中的内容
  var con=fs.readFileSync(‘./index.html’);
  第一个参数是读取文件的路径地址
  第二个参数是读取文件内容的编码格式
  读取出来的内容是字符串格式的
- fs.readFile 异步读取文件中的内容
  fs.readFile(‘./index.html’,function(){
  读取成功后触发会发回调函数执行
  });

除了读取内容以外还可以向文件中写入内容

fs.writeFileSync([pathname],[content],[encode])

- 第一个参数是路径名称
- 第二个参数是需要写入的内容(注意只能是字符串或者buffer格式的数据)
- 第三个参数是编码格式，一般常用’utf-8’

我们当前的写入为覆盖式写入，新写入的内容会覆盖原来的