#node-lesson5

## 反馈

- 下课时间太长

- node 调试不太懂

- buffer 不清楚

## 1.复习

### 1.1node的调试

- 前两样调试用于面试

- node自带的debug调试命令 n下一步 s步入 o步出

- node-debug foo.js node-inspector

- vsc 推荐使用

- vsc launch.json

- 抽风式的好使

- ws

### 1.2 es6

- 用es新特性之前加上严格模式

- 什么是es？

- 为什么用es的新标准？

- 为什么前端不用，node要用？

es6定义了一个标准---->let命令---->js代码里面写了let---->v8引擎解析，能解析说明支持了es6里的let命令---->node升级了

- 严格模式

- let

- const

- 块级作用域{}

let foo =456;

{

let foo =123;

}

console.log(foo);

- 字符串的扩展 includes(str) startsWith(str) endsWith(str) repeat(num) `${变量名}`

var bar='暗示的反思'

var foo='123'+'阿斯顿发斯蒂芬'+bar+'是短发飞';

var foo2=`阿斯顿发斯蒂芬${bar}是短发飞`

- 箭头函数、

- let const 箭头函数 用来看api使用

### 1.3 buffer

- 创建Buffer

- Buffer的一些属性和方法

- Buffer的一些属性和方法

+ buf[index] 通过下标访问 buffer 的某个字节的数据

+ buf.indexOf(value,[byteOffset],[encoding]) 查找某个字符在 buffer 内存中的字节下标

+ buf.includes(value,[byteOffset],[encoding])

+ buf.length

+ buf.slice([start,[end]])

+ buf.toString([encoding],[start], [end])

+ buf.write(string,[offset],[length],[encoding])

### 1.4 文件流

- 流对象 stream

- pipe（）

## 2.网络编程

我们处于互联网时代，我们可以随时随地通过 Internet 上网、浏览新闻、玩LOL、上淘宝购物等等。

这些过程都发生了网络数据的交互。

bs browser server

cs client server

网络编程 是指\*\*编写程序使两台联网的计算机可以完成网络数据交互，完成网络通信。\*\*

注意：这里的计算机泛指可以上网的设备，比如PC、手机、服务器、智能电视等等。

强调：网络编程重在思想，node只是一个可以帮助我们网络编程的一个工具而已。

使用其他编程语言或者操作系统进行网络编程，思想都是一样的。

### 什么是服务器？

```

服务器就是一个台电脑，是一台性能比较好的大电脑，它需要支持高扩展性，提供服务的

服务器一旦部署好服务后，一般动的比较少，linux系统比较常用，不需要比较复杂的可视化操作界面，因为复杂的可视化操作系统比较耗资源，如win10，和windows相比linux比较安全

服务器用linux的centos，ubuntu是有桌面版的

服务器多用centos

服务器不要用图形化界面，太耗费资源

```

### 什么是应用服务器？

```

作为服务器执行共享业务应用程序的底层的系统软件

```

### 什么是web服务器？

```

web服务器是一种应用服务器，提供了web服务，对内提供web应用程序的的运行环境

还有邮箱服务器，数据库服务器。

```

Apache、Nginx、IIS、tomcat

### Node 没有 Web 容器

如果我用js写了一个web应用程序，那么node就是web服务器

.net平台的 ASP或者ASP.net 需要 IIS 作为服务器容器，

PHP需要搭载 Apache 或者 Nginx 作为服务器容器，

Java 的 JSP 需要 tomcat 作为服务器容器，

ruby 的 ruby on rails 需要 搭配 Apache 等作为自己的服务器容器。。。

Node，不需要服务器容器。

Node，不需要服务器容器。

Node，不需要服务器容器。

### 为什么叫 Node

(node是js开发服务器的开发环境，运行环境，服务器软件(让文件可以通过网络请求来访问))

Node是一个面向网络而生的平台。

Ryan Dahl 在创建Node项目的时候给它起了一个名字叫做 web.js ，就是一个Web服务器。

类似于 Apache、tomcat、IIS 等服务器软件。

web.js 的发展超出了作者的最初想法，变成了构建网络应用的一个基础平台。

然后就可以在这个基础平台之上构建很多东西，比如服务器、客户端、各种各样的命令行工具等。

Node的目标就是成为一个构建快速、可伸缩的网络应用平台。

每一个Node进程构成网络应用中的一个节点。这就是 Node 的含义。

### ip

ip唯一标识 一台设备

端口号标识一个设备上的应用

### 跨域的场景

1.域名不同 （www.wuyou.com 和www.liuxi.com 即为不同的域名）

2.二级域名相同，子域名不同 （www.wuyou.wu.com www.liuxi.wu.com 为子域不同）

3.端口不同，协议不同 （ http://www.wuyou.com 和https://www.wuyou.com属于跨域www.wuyou.con:8888和www.wuyou.con:8080)

前端需要用到跨域的地方: (加载图片，文件资源很少会跨域，你用别人的网站图片是另一码事……，一般，一般来讲，ajax请求的时候，才会用到跨域)

1. 使用ctn节点，进行动静资源分离时，也就是说，我们的前端文件，与需要的后段文件，就没在一个项目里

2. 我们后段需要请求别的端口号，或者说是别的服务器项目中的后台文件，这个时候，按理说不应该是前端的任务，正儿八经的，应该是前端文件所在项目中的后台文件，帮我们把数据请求回来，因为他们是不存在跨域的，跨域只是浏览器限制的

但是，如果后台的人不给我们做这一步(它是个半瓶水，或者是个棒槌)

或者

说这个接口做了，用几天就不用了(比如双十一这种暂时的活动接口)

那就需要我们进行jsonp跨域请求了,

由上面可知，用到跨域的情形还是挺少的，而且我们前面讲过如何解决跨域

jsonp

自己这个php后台文件，请求数据做中转

如果只是调试阶段有跨域，我们有浏览器插件

### 前后分离

### 协议

- 双方或多方同意，并且达成共识，遵守这样的约束

### url

### 端口号

ip能够定位电脑，端口号可以定位一个应用程序

node 默认是3000 多了往上加

Apache服务器，访问资源:

部署:

丢到服务器软件配置的根目录下

访问:

ip+刚开始配的服务器根目录下的要访问的文件的路径

node服务器，访问资源:

部署:

不需要配置根目录，只需要一个js文件

(这个js文件中必须配置了端口号，集成了http核心模块的功能)， 如果做了下面这些操作，node+文件名，命令行中会一直卡在这里，进程一直在开启着，别人可以访问

如果只是一个普通的js文件，node+文件名，一下子就执行完了。做不到一直开启的状态。

注意：我们所有对这个端口的访问，都是这个js文件，接收，并处理的。与其他的服务器不一样，比如：Apache，就是我配置了根目录，你想访问什么文件，拿到那个文件的路径就可以，到时候，你的访问是这个文件进行处理。具体代码见笔记中两个js文件

但nodejs不一样，比如：你要通过这个端口号访问 :3000/public/index.html。这个时候，是js文件接收到这个请求，然后将public/index.html文件的内容readFild读出来，然后返给前端。

注意: 一个js文件一个端口号，我们可以同时node + 文件名，同时开启好几个node项目，但是注意端口号要区分开

访问:

访问的时候: ip+端口号+此项目中文件名

浏览器中访问，肯定是上面要齐全

如果是html中写url的时候，如果是访问的本项目中的文件(不管是后台文件 还是前端文件)，可以直接写相对路径，比如url: './app.js'(是相对该项目根目录的？？看看index.html中怎么写的)

端口号

默认3000，如果你开了好几个node项目，那就需要注意，端口号要一致。

文件名

如果不写

默认是会主动运行这个项目中的index.html文件，一般服务器都会做这个设置，到了nodejs中，因为所有请求都是这个js文件接收的，所以一般都会写这么个代码。

if (pathname == '/' && req.method == 'GET') {

fs.readFile(path.join(\_\_dirname, 'index.html'), (err, data) => {

if (err) {

throw err;

}

res.end(data);

});

}

当项目启动以后，前端文件改了，可以不用重启服务器，后台文件改了，需要重启。app.js肯定是后台文件，但是不知道前端页面的js文件属于前端还是后台文件？应该是前端文件吧……

## 2 http协议

通过报文传输

### 2.1在地址栏输入网址后页面是如何呈现的？

- DNS 把域名转化成ip DNS服务器来做这个事情 运营商提供的dns服务器

- CDN 内容分发网络

- 输入 URL：http://www.baidu.com

- DNS 域名解析

+ 计算机无法识别域名，计算机与计算机之间要想进行通信，必须通过ip地址用来定位该计算机所在的位置

+ 在浏览器中，输入的ip地址或者域名，默认给你加了一个80端口号（对方的服务器监听的就是80端口）

+ 158.12.25.652 域名就是为了好记

+ 为了好记，所以我们的 万维网提供了 一个 域名这样的概念

+ 当你输入了 ip 地址后，浏览器会自动去 找DNS域名解析服务器，

- 将用户输入的地址封装成 HTTP Request 请求报文 发送到服务器

+ 浏览器将用户输入的 URL 地址根据HTTP协议 封装成了 http 请求报文（请求头+请求行+请求体）

+ 该报文说白了也就是字符串而已，最终也要被转成了二进制数据再发送到服务器

- 后台服务器接收到用户HTTP Request 请求报文

+ 后台服务器接收到 客户端发送给自己的数据（二进制数据）

\* 首先把二进制数据按照编码解析成字符，（人类可以识别的）

\* 解析成字符之后，再按照 HTTP 协议规范中定义的格式解析出来

- 后台服务器处理用户请求信息

+ 当得到用户请求报文之后，根据请求报文中的 get、port或者 URL、或者URL中的查询字符串或者 请求体中的数据

+ 根据用户的特定的请求数据做特定的处理

- 后台服务器将相应结果封装到 HTTP Response 响应报文中 发送给客户端

+ 当我们解析和处理完用户请求报文消息之后

+ 服务器开始将具体的 要发送给客户端的数据 根据 HTTP 协议规范 封装成 HTTP协议响应报文

+ 响应头、响应字段、响应体

+ 该数据说白了也是具有特定格式的字符串而已，最终这个字符串也要转换成二进制数据发送到客户端

+ 发送到客户端也是通过 Socket（ip地址、端口号） 发送到了该客户单

- 用户浏览器接收到响应后开始渲染html、css，解析和执行 JavaScript 代码

+ 当客户端解析到 服务器发送过来的 二进制数据

+ 客户端浏览器也会将 二进制数据 根据编码类型解析成 字符串

+ 然后根据 HTTP 协议，解析服务器发送过来的 响应报文

+ 然后根据响应报文中的报文内容（报文头、报文体）做具体的解析

- 当浏览器在解析的过程中遇到 一些静态资源时，会再次重复上面的步骤

### 2.2HTTP协议

HTTP协议就是 浏览器 和 服务器 之间通信的一个数据格式规范

在HTTP协议中，始终是以一种 一问一答 的形式在进行沟通和交流（数据交换）

服务器如果没有收到浏览器的请求消息，服务器永远不会主动的发送响应消息

浏览器不发出请求，服务器不会主动的发送响应

- 浏览器发送请求数据到服务器

- 服务器解析浏览器发送的请求数据

- 服务器响应数据到客户端浏览器

### 2.3cookie&session

- http是一个无状态的协议，每次去请求服务器的时候，服务器是不能记住客户端的

- cookie 信物，客户端请求服务器了以后，服务器给客户端一个信物，下次客户端再请求服务器的时候，给服务器这个信物

服务器就知道是哪个客户端了

缺点是，不能存储太多内容、不安全

- session 存储在服务器端的，保存着用户信息的，和cookie搭配来使用，可以用cookie里面的key对应上session里面的值

session可以存在内存或硬盘上都可以。

### 2.4前后端分离

### 2.5

- loaclhost

- 127.0.0.1

- ip地址

### Nginx

代理，委托一个人帮我去做事情，上网代理，就是我上网的时候如果上网被墙了，我可以用代理帮我们去上网。

```

nginx除了是个web服务器还能够做反向代理服务器，

反向代理服务器的作用，可以用来做负载均衡

代理，委托一个人帮我去做事情，上网代理，就是我上网的时候如果上网被墙了，我可以用代理帮我们去上网。

```

- 可以用nginx来做负载均衡