# node-lesson6

## 反馈

好 老师说的点很多

老师http协议还不太懂

## 复习

### 1.1复习网络编程

- 什么是网络编程

- 使用 node 进行网络编程需要使用其它web服务器作为容器吗

- 什么是协议？什么是http协议？

- 在网络编程中，为什么要有协议？制定协议有什么好处？

### 1.2 http协议

### 1.3session&cookie

###网站的优化----->思想

1.能压缩的压缩能合并的合并

少一个script也好，少一次请求

2.少用js去遍历对象或者写for循环

3.能提前运算好的提前运算好，使用枚举法

var a=1024\*1024;

4.CDN节点，从最近的CDN节点请求数据

5.流式(响应式)图片 🡪 图片压缩



### 1.4 工具的使用

### 通过 nodemon 实现 保存文件实时重启

1. 安装nodemon ` npm install -g nodemon `

2. 基本使用 `nodemon server.js`

只要执行了上面的命令，那么当你修改了 server.js 那么nodemon会帮你自动重启 server

###2.1 使用npm命令执行代码

1.package.json里面配置scripts属性里面写npm的命令

2.npm的命令在控制台直接npm打印出来可以查看

3.npm命令后面接的是具体的命令，打npm命令调用具体的命令

###2.2编写node脚本

1.把js代码变成node脚本

在要执行的js文件当中加入下面这句话，是固定写法

#!/usr/bin/env node

2.package.json里面配置bin属性里面写命令,这个命令的名称是可以自定义的

3.必须打npm link这个命令，把当前包安装到全局去

###2.3通过命令调用脚本

## http 模块

### 使用 http 模块进行web开发

localhost:3000 不走网卡

127.0.0.1:3000 走网卡

192.168.141.XXX 通过网卡发请求给路由或交换机，在请求回来你的电脑

- 创建 HTTP 服务器

`http.createServer([requestListener])`

- 通过监听Event: 'request'来进行

server.on('request',(request,response) => {});

- request、response对象

request对象，用于接收请求报文内的信息

response对象，用于返回响应报文内的信息

- response.write()

write方法用于向响应体里写内容，用于向客户端发送信息，可以被多次使用

- response.end()

在一次请求和响应过程当中，只要调用了 end 就不能再继续 write 了

就说明你是在一次请求和响应处理过程中已经结束了响应，但是又去 向客户端发送数据

这个时候就会报错

- server.listen(port[, hostname][, backlog][, callback])

监听端口，执行这句，服务正式被启动使用

server.listen(3000,'127.0.0.1',() => {

console.log('server is listening at port 3000');

});

### request

- request.headers 请求头

- request.method 请求方法

- request.url 请求路径

- request.httpVersion 请求HTTP协议版本

### response

- res.writeHead(200,{'Content-Type': 'text/html; charset=utf-8'});

设置响应报文头

- res.write('<h1>hello world</h1>');

- res.end();

### Content-Type

内容类型，一般是指网页中存在的Content-Type，用于定义网络文件的类型和网页的编码，

决定文件接收方将以什么形式、什么编码读取这个文

### 通过 req.url和req.method的不同给出不同的响应

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

res.writeHead(200, {

'Content-Type': 'text/html; charset=utf-8'

});

res.write('<h1>首页</h1>');

res.end();

}

### request通过data事件和end事件来接收post的数据

let data = '';

req.on('data', (chunk) => {

data += chunk;

});

req.on('end', () => {

console.log(data);

res.writeHead(200, {

'Content-Type': 'text/html; charset=utf-8'

});

res.write('<h1>收到数据了</h1>');

res.end();

});

### 通过 querystring解析post的数据

querystring.parse('foo=bar&baz=qux&baz=quux&corge')

// returns { foo: 'bar', baz: ['qux', 'quux'], corge: '' }

### url

- url.parse(urlStr, [parseQueryString], [slashesDenoteHost])

urlStr->url字符串

parseQueryString->为true时将使用queryString分析查询字符串，默认为false

slashesDenoteHost

默认为false，//foo/bar 形式的字符串将被解释成 { pathname: ‘//foo/bar' }

如果设置成true，//foo/bar 形式的字符串将被解释成 { host: ‘foo', pathname: ‘/bar' }

服务器相关操作事件和API

- Event: 'close'

- Event: 'connection'

- Event: 'request'

- server.close([callback])

- server.listen(port[, hostname][, backlog][, callback])

请求对象

- Event: 'data'

- message.headers

- message.httpVersion

- message.method

- message.url

+ require('url').parse(request.url)

+ require('url').parse(request.url, true)

响应对象

- response.end([data][, encoding][, callback])

- response.setHeader(name, value)

- response.statusCode

- response.statusMessage

- response.write(chunk[, encoding][, callback])

- response.writeHead(statusCode[, statusMessage][, headers])

## gulp

[官网](<http://www.gulpjs.com>)

[中文网](<http://www.gulpjs.com.cn>)

- 前端自动化构建工具

js压缩,var x,xname，混淆

合并.

css压缩

html压压缩

- grunt ,webpack...

### 核心就5个方法

- task,gulp中是一个个任务的形式来实现功能。

+ task('任务名',function(){

.....

});

- src

+ src('./\*.js')

- dest('./minjs/')// 指定处理后的文件的输出路径.

- watch('./\*.js',['任务名1','任务名2']);

- run('任务名');//执行指定的任务.

### gulp的安装

- 使用npm 进行安装

- `npm install gulp-cli -g`;

- `npm i -g gulp-cli`;

### gulp 使用

#### 使用时还需要在项目中通过npm非全局安装gulp

- `npm install gulp --save-dev`

#### 还需要在当前项目根目录添加一个gulpfile.js文件来写具体的任务代码.

### gulp的一些插件

- 也是使用npm安装

- 对js代码进行压缩 gulp-uglify

.pipe(uglify())

- 对代码进行合并 gulp-concat

.pipe(concat('all.js'))

.pipe(concat('all.css'))

- 对css进行压缩 gulp-cssnano

.pipe(cssnano())

- 对html进行压缩 gulp-htmlmin

.pipe(htmlmin({collapseWhitespace:true,minifyJS:false,minifyCSS:true}))