http协议

浏览器和web服务器之间的通信协议

1. 头信息 Headers
2. 通用头信息 General

Request URL：请求的URL，对应浏览器地址栏，向服务器获取哪些内容

Request Method：请求的方法，获取内容的方法：get/post

Status Code：响应的状态码

1\*\* 正在响应，还没结束

2\*\* 响应成功

3\*\* 响应的重定向，会跳转到另一个URL

4\*\* 客户端请求错误

5\*\* 服务器端错误

Remote Address：请求的服务器IP地址及端口号

1. 响应头信息 Response Header

Connection: 连接方式 ，keep-alive 持久连接

Content-Encoding: 内容压缩形式，gzip

Content-Type： 响应的文件类型 text/html/image（影像画像）...

Location：跳转的URL，常配合着状态码3\*\*使用

1. 请求头信息 Request Header

Accept： 浏览器接受的文件类型

Accept-Encoding：接受的压缩形式有哪些

User-Agent： 客户端使用的浏览器版本

1. 请求主体

可有可无web服务器发出请求，浏览器向服务器发送请求传递的数据

http模块

1. 模拟浏览器

模拟浏览器向web服务器发送请求，还可以 创建web服务器

1. 模拟浏览器

http.get(url,callback) 发送请求

url 请求的URL

callback 回调函数，获取服务器端的响应

res 响应的对象

res.statusCode 获取响应的状态码

res.on('data',function(chunk){ })

事件：监听服务器端是否有数据传输过来

chunk 传输的数据，格式为buffer

例：

//引入http模块

const http = require('http');

//向web服务器发送请求

http.get('http://www.tmooc.cn',function(res){

//参数1.请求URL 2.获取响应 res:响应的对象

//响应的状态码

console.log(res.statusCode);

//获取响应的内容

res.on('data',function(chunk){

console.log(chunk.toString());

}) //事件：监听是否有数据流响应，chunk是buffer形式

});

1. 创建web服务器

|  |
| --- |
| const http = require('http'); //引入模块  var app = http.createServer(); // 创建web服务器  app.listen(8080); //设置监听的端口，监听浏览器的请求  //收到请求给予回复，固定死的参数  app.on('request',function(req,res){  req 请求的对象 （在cmd中可查看）  req.url 请求的url，获取**端口号后边的部分 /login**  req.method 请求的方法 获取...  req.headers 请求的头信息 获取...  res 响应的对象 （在服务器响应）  res.write() 设置响应的内容  res.writeHead(状态码，头信息 对象) 跳转  **res.end() 结束并发送响应到浏览器（必须写）**  }); |

//引入模块

const http = require('http');

//创建web服务器

var app = http.createServer();

// http://127.0.0.1:8080

//设置监听的端口号

app.listen(8080);

//接受请求，根据请求作出响应

app.on('request',function(req,res){

//req：请求的对象，获取所有请求

//请求的URL 请求的方法

console.log(req.url,req.method);

//请求的头信息

console.log(req.hearders);

//res：响应的对象，作出具体响应

//设置响应内容

//res.write('this is homepage');

//设置响应的头信息和状态码，跳转

res.writeHead(302,{ //后面的参数是对象

Location:'http://www.tmooc.cn'

});

//结束并发送响应内容到浏览器

res.end()

}); //一旦有请求，自动触发事件 服务器响应反映在回调函数中

//PS：每次更改内容要重启服务器

使用zlib模块对服务器端响应压缩

|  |
| --- |
| //1.1引入zlib模块  const zlib = require('zlib');  //1.2设置内容的压缩形式  'Content-Encoding':'gzip'  //1.3创建gzip压缩  var gzip = zlib.createGzip();  //1.4使用读取流读取文件并 响应到浏览器  fs.createReadStream('tmooc.html').pipe(gzip).pipe(res) |