实验一：HTTP前端测试分析

## 一、实验概述

Web系统分为客户端和服务器端，用户是与客户端浏览器打交道，测试人员可以站在用户的角度从前端进行分析，辅助找些系统问题。本实验中使用浏览器自带的开发者工具，进行监控。

## 二、实验目的

1. 从前端角度来发现Web系统的问题。

2. 熟悉Web前端测试方法。

3. 掌握应用层协议HTTP的结构和工作过程。

## 三、实验要求

### 3.1 对百度首页进行前端分析

使用Chrome 或者IE或Firefox，打开百度首页，打开开发者工具模式进行监控首页的加载过程，对加载过程进行分析。根据监控结果，写出分析总结：

* 发送的总请求数。
* 整个页面的加载时间,页面渲染时间。
* 消耗时间最大的请求是哪一个，所花的时间为多少？
* 响应是否有异常。

### 3.2 对新浪首页进行前端分析

* 网址输入：<https://www.sina.com.cn/>
* 对比分析新浪共发送总请求数是多少？
* 整体页面加载时间是多少，页面渲染时间多少？
* 排名前五的请求有哪些？各消耗多长时间？文件大小是多少？
* 如果文件不是很大，但耗时多，说明了什么呢？
* 页面响应总时间长，但是用户并没有感到很慢，用户体验还可以，这说明了什么?

### 3.3 熟悉HTTP GET请求 和POST请求，对比GET请求和POST请求的区别

1. 比如当用户输入 <http://www.baidu.com> 这是一个发送了GET请求给服务器，请求服务将网站首页的HTML返回。理解下列重要关键字段的意义，并截图说明：

General：Request URL， Request Method， Status Code，

Request Headers： Accept，Accept-Encoding， Accept-Language，Cache-Control, Cookie, Host, User-Agent

Response Headers: Cache-Control, Connection, Content-Encoding, Content-Type, Expires, Set-Cookies。

1. 点击登录，输入用户名密码点击登录按钮，实际上是发送了一个POST请求给服务器，理解POST的General，Request Headers， Response Headers 中字段的意义，并截图说明。
2. 对比分析GET 和POST请求，说明GET请求和POST请求的区别。

## 四、实验过程

使用**edge**浏览器进行实验，F12打开浏览器自带的开发者工具，输入网址打开百度首页，监控首页的加载过程。监控结果总览如图1所示。

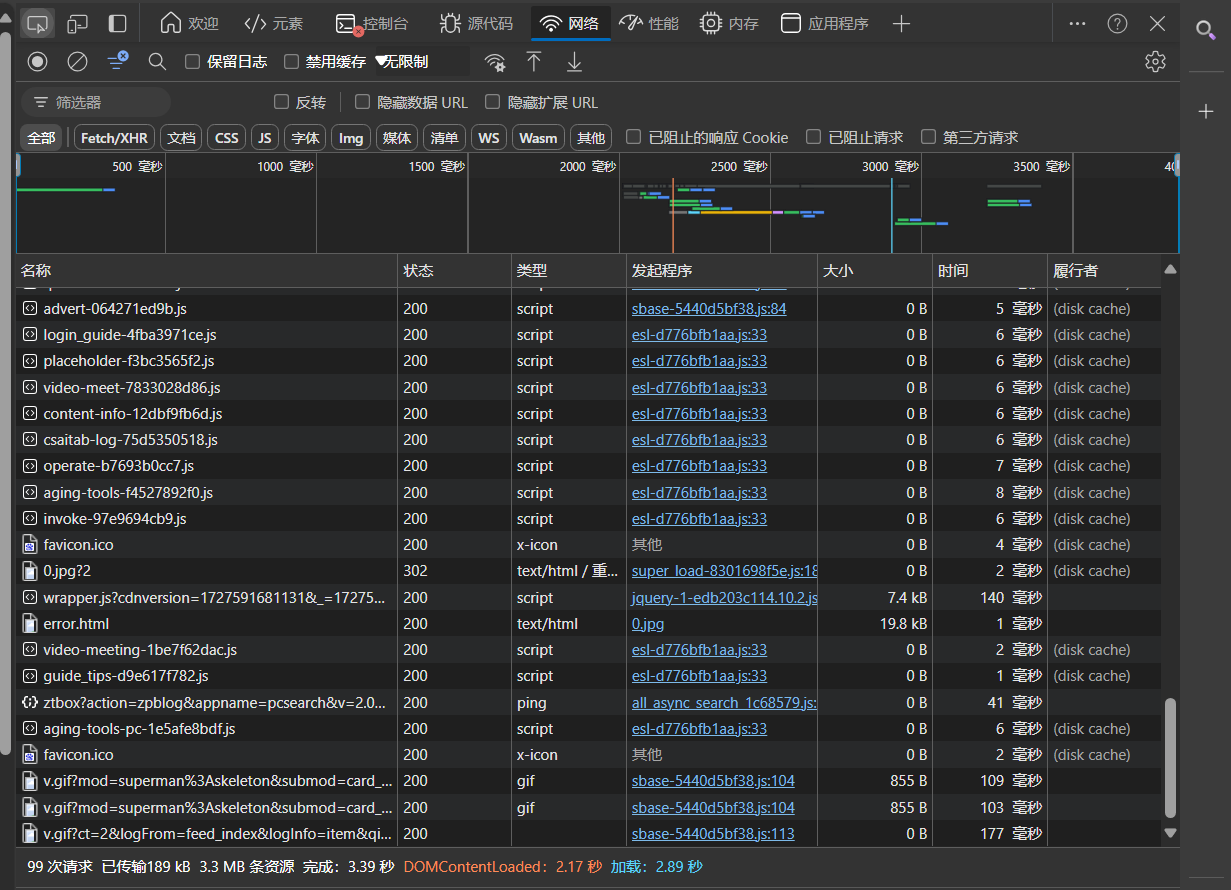


图1  百度首页监控结果总览

从图2中我们可以看到，发送的总请求数为99次（64requests）；DOM树构建完成，即HTML页面由上向下解析HTML结构到末尾封闭标签</html>的时间为2.17ms（DOMContentLoaded: 2.17ms）；整个页面的加载时间为 2.89s加载:（Load: 2.89s）；页面上所有HTTP请求从发送到响应完成的时间为:3.39s完成:（Finish: 3.39s）。



图2  底部信息

再由图3所示，对比各请求的时间信息后可以发现，消耗时间最大的请求是一个ping类型的POST请求。其具体的请求行为

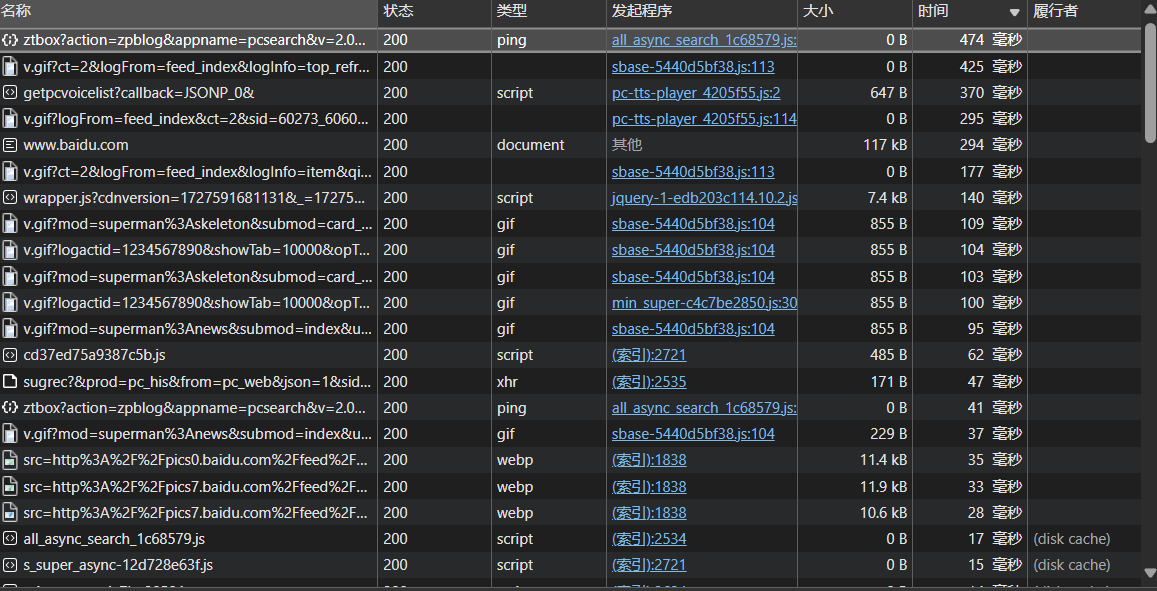
BAIDUID=894E4DD57E8ECFF263CEA4299A5A0389:FG=1; BAIDUID\_BFESS=894E4DD57E8ECFF263CEA4299A5A0389:FG=1; BAIDU\_WISE\_UID=wapp\_1726547109790\_324; BIDUPSID=894E4DD57E8ECFF263CEA4299A5A0389; PSTM=1726562043; \_\_bid\_n=19203dc4da180647ea133e; BDUSS=XRIYkhvbG9Cd09vRkgwQXU0NWppNVhxUkR6RDZ0UnM3QWhpfnhyY2xFMFNkUk5uRUFBQUFBJCQAAAAAAQAAAAEAAAB2SE1CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABLo62YS6OtmT; BDUSS\_BFESS=XRIYkhvbG9Cd09vRkgwQXU0NWppNVhxUkR6RDZ0UnM3QWhpfnhyY2xFMFNkUk5uRUFBQUFBJCQAAAAAAQAAAAEAAAB2SE1CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABLo62YS6OtmT; ZFY=1lcaAZyd9u:Anos:Bih:BFxI56jC8Y9cvIA9:BoFqUOM:Bps:C; newlogin=1; RT="z=1&dm=baidu.com&si=41b473ff-0d3e-4b5c-9e23-a14cea38092f&ss=m1ks96ke&sl=0&tt=0&bcn=https%3A%2F%2Ffclog.baidu.com%2Flog%2Fweirwood%3Ftype%3Dperf&ld=n1&nu=481emiv1&cl=wfy&ul=y0l2i&hd=y0l36"; H\_PS\_PSSID=60273\_60601\_60772\_60816\_60840; BA\_HECTOR=ahak2l8h0501258h8g8kak2035n8ek1jfhsr31u  


图3  消耗时间最大的请求



图4  具体的时间消耗信息

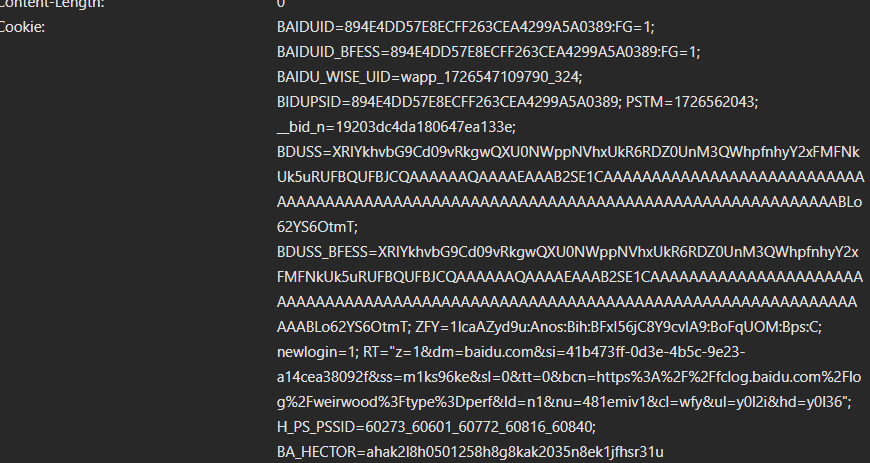


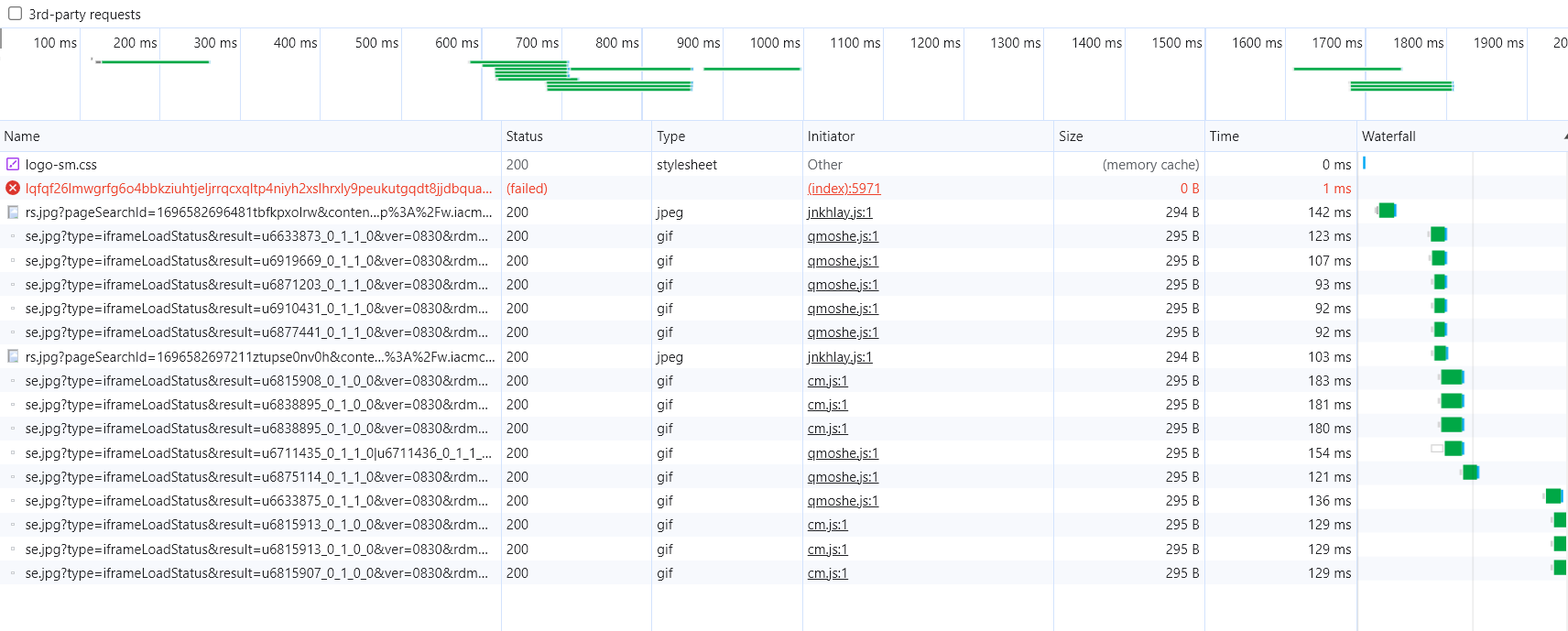
图5  部分请求信息

请求响应结果方面，从图1的Status列中可以看到，所有的请求Status Code返回值均为200，因此响应无异常。

### 4.2 对新浪首页进行前端分析

使用edge浏览器的开发者工具，输入网址<https://www.sina.com.cn/>对新浪首页的加载进行监控。监控结果总览如图6所示。

同理，从图7中可以得知新浪发送的请求数为640次；DOM树构建完成的时间为6.30s；页面上所有HTTP请求从发送到响应完成的时间为18.50 s。



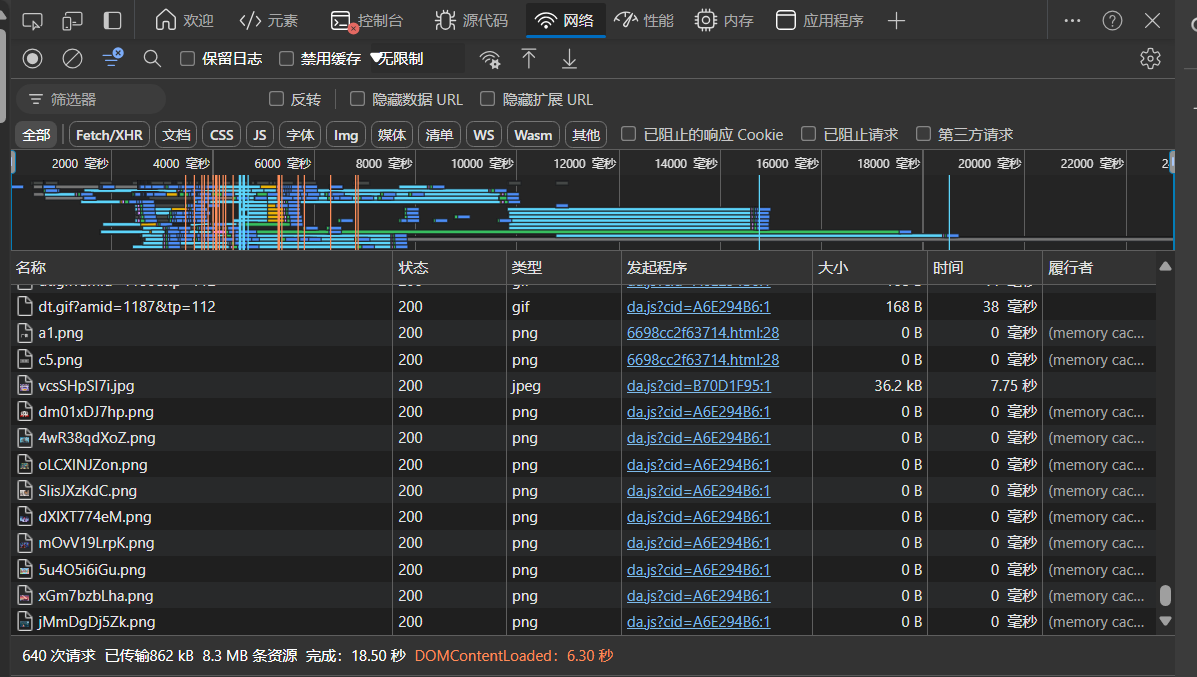


图6  新浪首页监控结果总览



图7  底部信息

排名前五的请求及其消耗时间和文件大小依次为如图8所示。

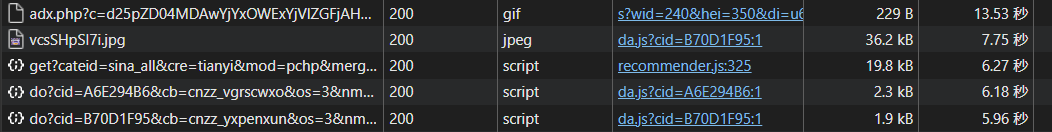


图8  排名前五的请求

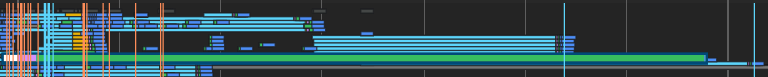


图9  请求的时间消耗详细情况

文件大小并不是很大，但下载时间却很长，这可能是因为在等待响应和实际下载文件的时间上并没有花费太多时间。相反，耗时较多的环节可能是在阻塞、域名解析或初始化连接等方面。换句话说，下载过程中出现了阻塞、域名解析或初始化连接的问题，导致下载时间延长。

当页面响应总时间长时，但用户并不感觉很慢，这通常是因为页面的加载顺序是优化过的。根据用户需求，将需要迅速呈现的信息进行优先加载，以此提高用户体验，用户对于后续内容的等待也更加耐心。

在页面加载时，应优先加载必要的资源（如页面结构、CSS、JavaScript等），以便网页迅速呈现，并提高可用性。耗时较长、不需要及时展示的内容（如图片、视频等）则可以后续自行加载，以免影响页面的初始加载速度。

### 4.3 熟悉HTTP GET请求 和POST请求，对比GET请求和POST请求的区别

#### 4.3.1 用edge浏览器打开http://www.baidu.com，这是一个发送了GET请求给服务器，请求服务将网站首页的HTML返回。理解下列重要关键字段的意义，并截图说明

General: Request URL, Request Method, Status Code，

Request Headers: Accept, Accept-Encoding, Accept-Language, Cache-Control, Cookie, Host, User-Agent

Response Headers: Cache-Control, Connection, Content-Encoding, Content-Type, Expires, Set-Cookies

GET请求截图如图10、11、12、13、14所示。



图10  GET

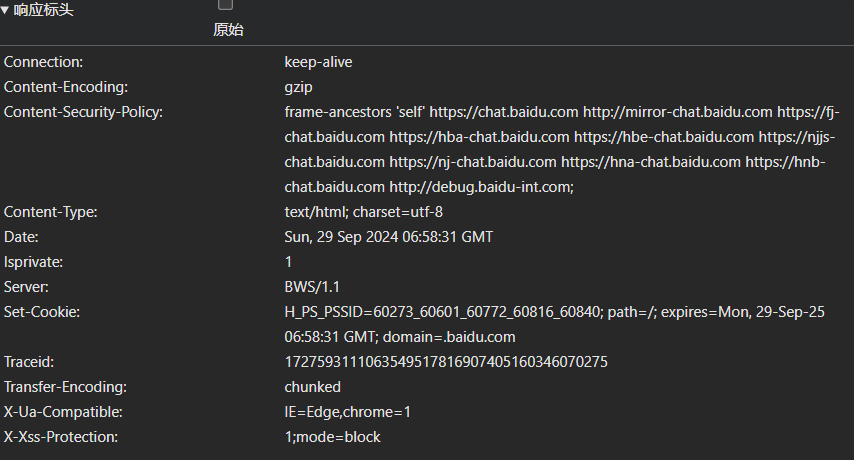
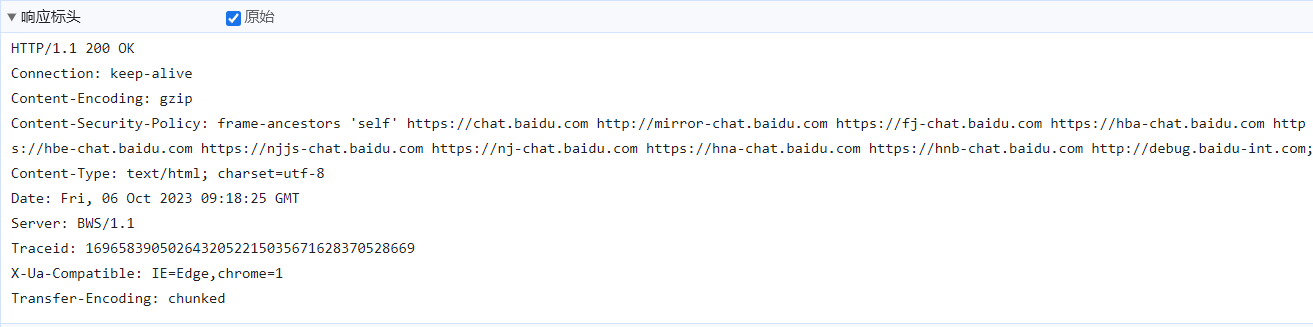


图11  解析响应头



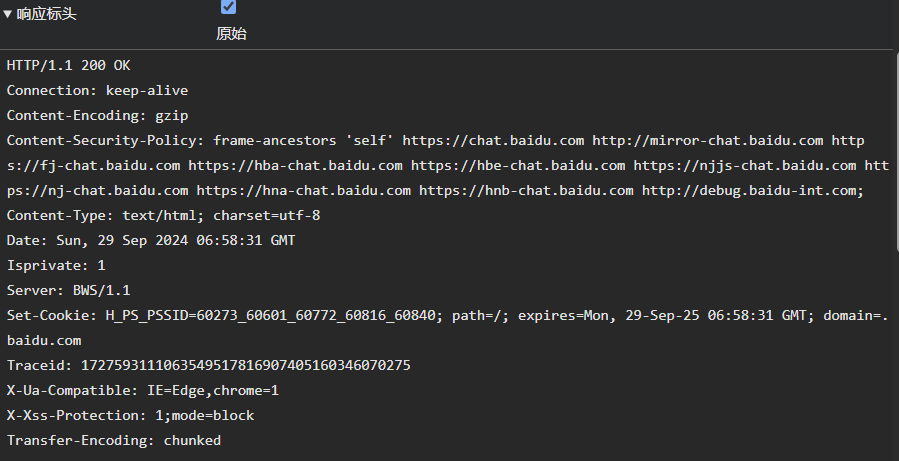
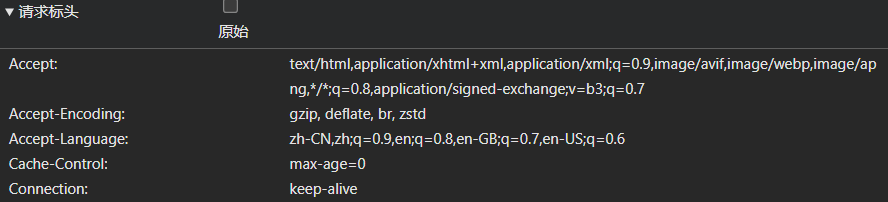


图12  原始响应头



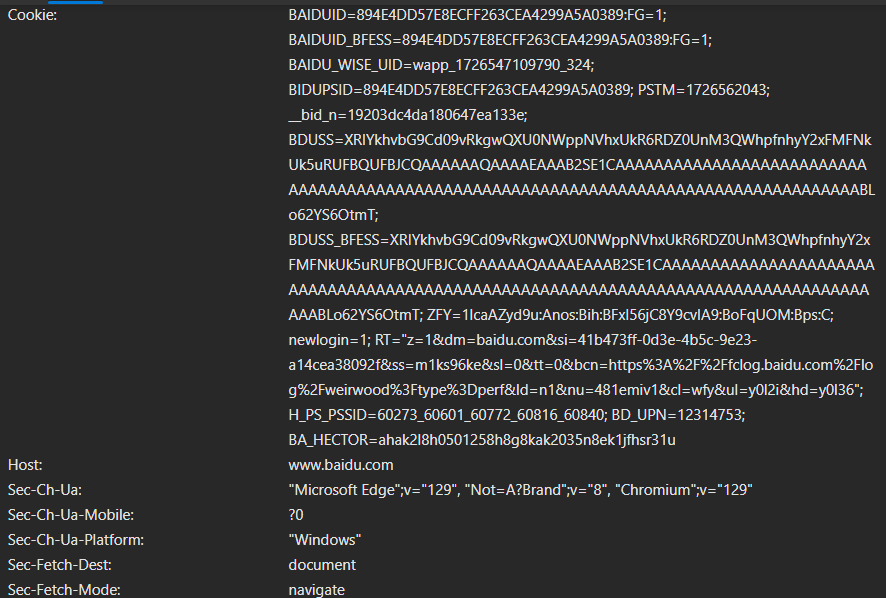
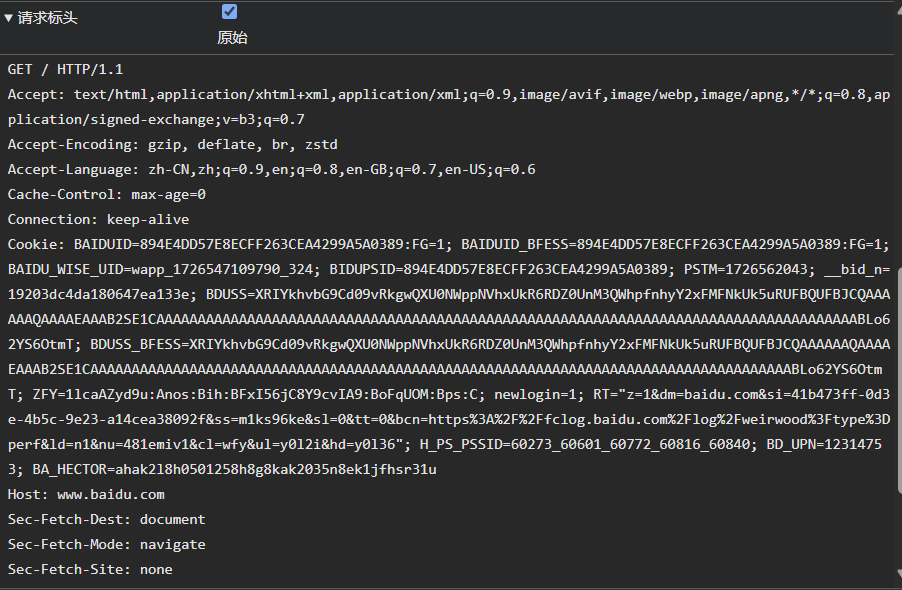


图13 解析请求头



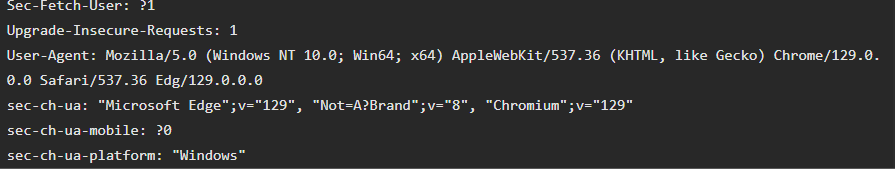


图14  原始请求头

【General】：

* Request URL：请求的URL，指定要访问的网络资源。
* Request Method：请求方法，表示客户端使用的是哪种方法来获取资源。
* Status Code：状态码，表示服务器对请求的处理结果。

【Request Headers】：

* Accept：指示浏览器可以接受的媒体类型。
* Accept-Encoding：指定浏览器支持的压缩方法。
* Accept-Language：浏览器声明自己接受的语言类型。
* Cache-Control：指定请求和响应遵循的缓存机制。
* Cookie：将Cookie的值发送给HTTP服务器。
* Host：指定被请求资源的主机名。
* User-Agent：浏览器通知服务器使用的操作系统、浏览器名称和版本号。

【Response Headers】：

* Cache-Control：指定客户端的缓存机制。
* Connection：表示是否需要持久连接。
* Content-Encoding：服务器表明文档的编码方式。
* Content-Type：服务器告知浏览器返回的对象类型和字符集。
* Expires：服务器通知浏览器在何时认为文档已过期，不再缓存。
* Set-Cookies：用于把Cookie发送到客户端浏览器，每一个写入Cookie都会生成一个Set-Cookie

#### 4.3.2 点击登录，输入用户名密码点击登录按钮，实际上是发送了一个POST请求给服务器，理解POST的General，Request Headers，Response Headers中字段的意义，并截图说明

POST请求截图如图15、16、17、18、19所示。

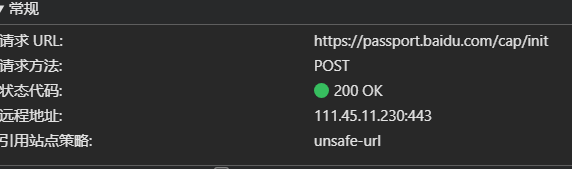


图15 POST

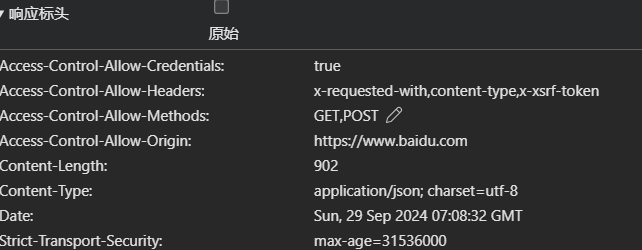


图16  解析响应头

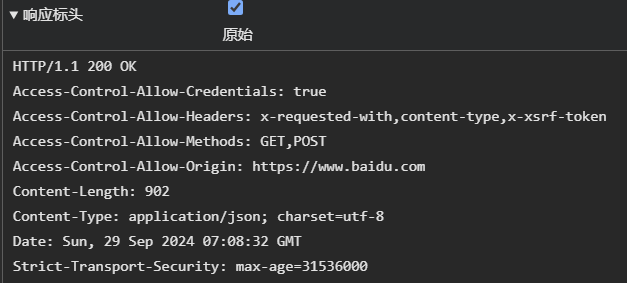
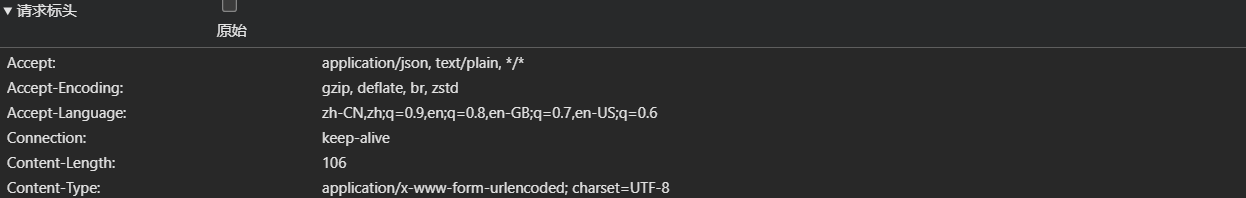


图17 原始响应头





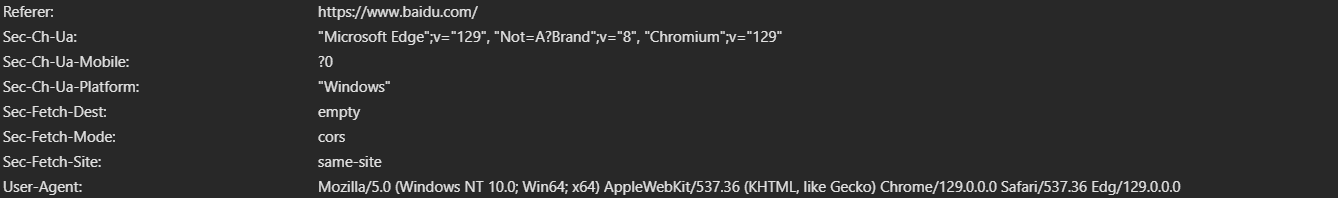


图18  解析请求头

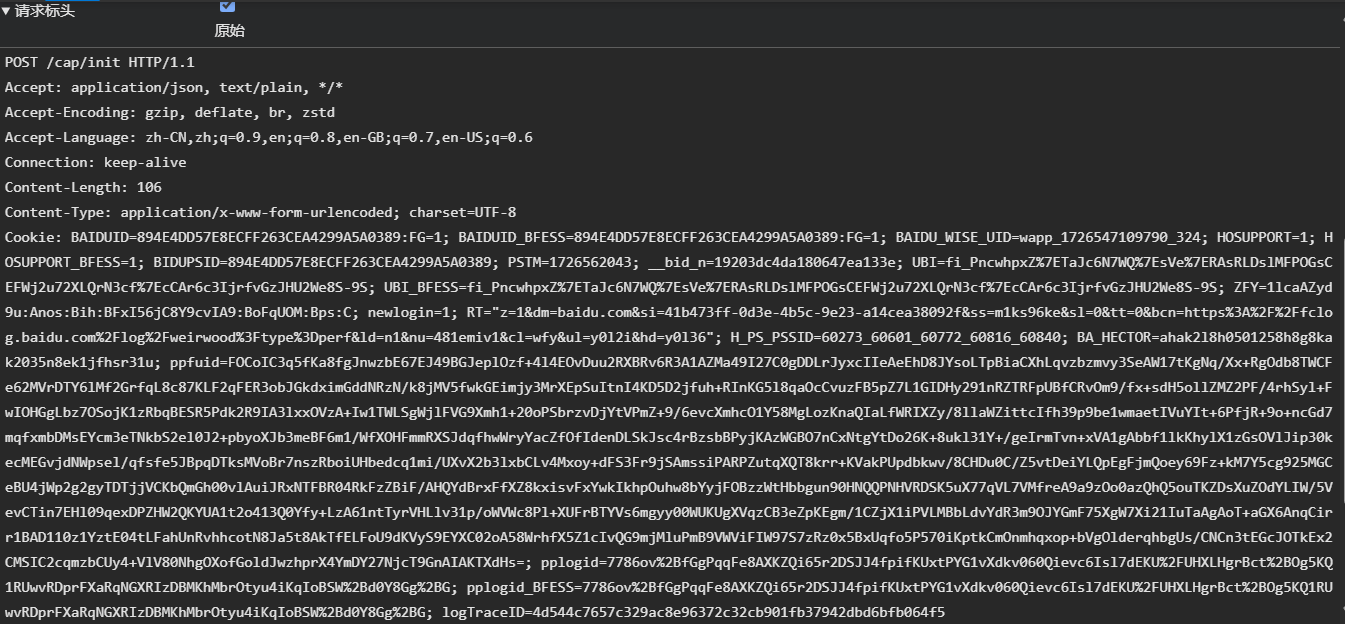




图19  原始请求头

* General:

Request URL: https://passport.baidu.com/cap/init

Request Method: POST

Status Code: 200 OK

* Request Headers:

Accept: \*/\*

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Cookie：

Host: passport.baidu.com

User-Agent:Mozilla/5.0 (Windows NT10.0;Win64; x64; rv:87.0)Gecko/20100101 Firefox/87.0

以上字段含义与GET方法一致。

Content-Length: 4102

浏览器发送给服务器数据的长度

Content-Type: application/x-www-form-urlencoded

用于定义网络文件的类型和网页的编码，决定浏览器将以什么形式，什么编码读取这个文件。application/x-www-form-urlencoded是POST提交数据的方式：通过页面表单方式提交。

* Response Headers:

Cache-Control: public

Connection: keep-alive

Content-Encoding: gzip

Content-Type: text/html

Expires: 0

Set-Cookies

以上字段含义与GET方法一致。

#### 4.3.3 对比分析GET和POST请求，说明GET请求和POST请求的区别

1. 参数传递位置：
   * GET请求：参数通过URL的查询字符串（Query String）进行传递，在URL末尾添加?key1=value1&key2=value2的形式。
   * POST请求：参数在请求的消息体（Body）中进行传递，不会直接暴露在URL中。
2. 参数长度限制：
   * GET请求：由于参数直接暴露在URL中，有长度限制。不同浏览器和服务器对URL长度的限制有所不同，通常约为2048个字符。
   * POST请求：没有明确的参数长度限制，理论上可以传输更大的数据量。
3. 参数安全性：
   * GET请求：由于参数暴露在URL中，可能被保存在浏览器历史记录、服务器日志等地方，存在安全性风险。因此，不应将敏感信息放在URL中传递。
   * POST请求：参数不会直接显示在URL中，安全性相对较高，适合传输敏感信息。
4. 缓存与书签：
   * GET请求：由于参数在URL中，可以被浏览器缓存和添加到书签中，方便用户重复访问相同资源。
   * POST请求：由于参数在请求的消息体中，不会被缓存或添加到书签中。
5. 幂等性：
   * GET请求：GET请求是幂等的，即多次请求相同的URL和参数，结果应该保持一致，不会产生副作用。
   * POST请求：POST请求不是幂等的，多次请求相同URL和参数，可能会导致服务器上的资源状态发生改变或副作用。
6. 使用场景：
   * GET请求：适合获取资源、查询操作，不涉及对服务器数据的修改。常用于浏览器向服务器请求页面、发送搜索关键字等。
   * POST请求：适合提交数据、创建新资源、进行修改操作。常用于用户注册、登录验证、表单提交等。

## 五、实验总结

通过本次实验，我们深入了解了浏览器自带的开发者工具，以及如何使用它进行前端测试。同时，我们也学习了应用层协议HTTP的结构和工作过程，进一步加深了对GET请求和POST请求的理解。

在这个过程中，我们掌握了如何使用开发者工具进行网络请求调试、监测请求和响应、查看请求报文和响应报文、手动发送请求以及模拟不同网络状况等操作。这些技能对于前端开发和调试非常有用。

另外，我们深入研究了GET请求和POST请求的区别。GET请求适用于获取资源和查询操作，可被浏览器缓存和添加到书签中；而POST请求适用于提交数据和进行修改操作，通常不会被缓存。我们学习了请求头和响应头中的Cache-Control字段，了解了它们在控制缓存行为方面的重要作用。

总的来说，这次实验让我们对前端开发和HTTP协议有了更为深刻和全面的认识。