

[TOC]

# 第一部分 输入网址并解析

## URL的组成

URL (Uniform Resource Locator) , 统一资源定位符, 用于定位互联网上的资源, 俗称网址。

scheme : // host.domain : port / path / filename ? query=xxx # anchor

协议                  主机(域名)          端口                  路径                  查询参数                  锚点

scheme: 协议, 常见的有http、https、ftp、file  
host: 主机(域名), 是资源所在的网站名或服务器的名字  
port: 端口, 同一个域名下面可能同时包括多个网站, 它们之间通过端口(port)区分, 默认端口为80  
path: 定义服务器上的路径, 如果省略, 则访问网站的根目录(若后面未跟具体的文件名, 通常返回该目录下的index.html)  
filename: 定义文档/资源的名称  
query: 查询参数(parameter) 是提供给服务器的额外信息。参数跟在路径后面, 用?分隔, 以键值对(key=value)形式, 参数可以有多组, 用&连接  
anchor: 锚点(anchor) 是网页内部的定位点, 使用#加上锚点名称, 放在网址的最后

## 解析URL

输入URL后, 浏览器会解析出协议、主机、端口、路径等信息, 并构造一个HTTP请求。

1. 浏览器发送请求前, 根据请求头的`expires`和`cache-control`判断是否命中(包括是否过期)强缓存策略, 如果命中, 直接从缓存获取资源, 并不会发送请求。如果没有命中, 则进入下一步。
2. 没有命中强缓存规则, 浏览器会发送请求, 根据请求头的`If-Modified-Since`和`If-None-Match`判断是否命中协商缓存, 如果命中, 直接从缓存获取资源。如果没有命中, 则进入下一步。
3. 如果前两步都没有命中, 则直接从服务端获取资源。

## HSTS (HTTP Strict Transport Security)

由于安全隐患, 会使用HSTS强制客户端使用HTTPS访问页面。

原理:

- 在服务器响应头中添加`Strict-Transport-Security`, 可以设置`max-age`
- 用户访问时, 服务器种下这个头
- 下次如果使用http访问, 只要`max-age`未过期, 客户端会进行内部跳转, 可以看到307 Redirect Internal的响应码 变成https访问源服务器

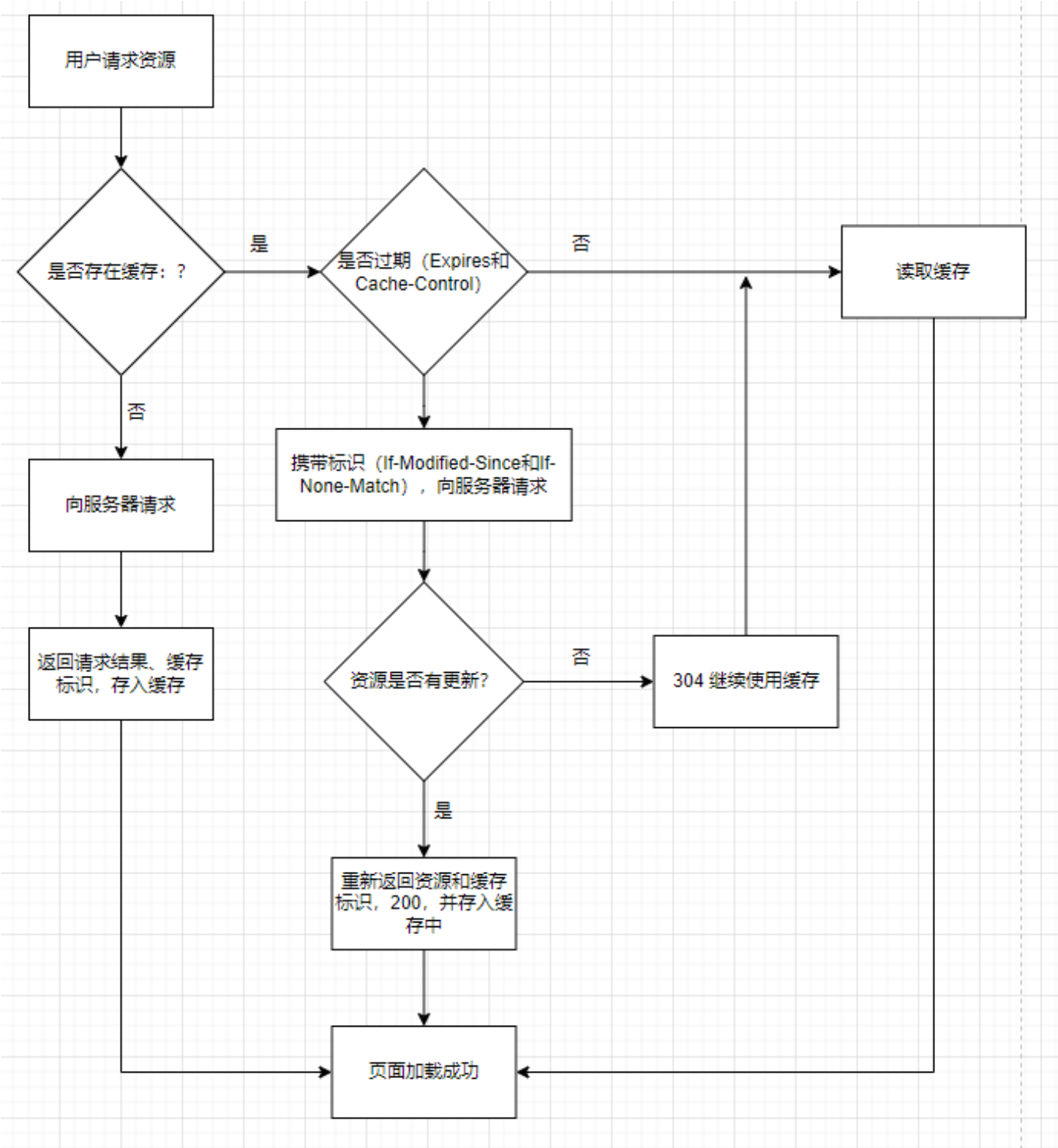
详见: [你所不知道的HSTS](#)

当你的网站均采用HTTPS，并符合它的安全规范，就可以申请加入HSTS列表，之后用户不加HTTPS协议再去访问你的网站时，浏览器都会定向到HTTPS。无论匹配到没有，都要开始DNS查询工作了。

## 浏览器缓存

### 强缓存

强制缓存就是向浏览器缓存查找该请求结果，并根据该结果的缓存规则来决定是否使用该缓存结果的过程。强缓存又分为两种Expires和Cache-Control



### Expires

- 版本：HTTP/1.0

- 来源：存在于服务端返回的响应头中
- 语法：Expires: Sun, 12 Jan 2022 21:42:00 GMT
- 缺点：服务器的时间和浏览器的时间可能不一致导致失效

Cache-Control

- 版本：HTTP/1.0
- 来源：响应头和请求头
- 语法：Cache-Control: max-age=3600
- 缺点：时间最终还是会失效

请求头：

字段名称	说明
no-cache	告知（代理）服务器不直接使用缓存，要求向原服务器发起请求
no-store	所有内容都不会被保存到缓存或Internet临时文件中
max-age=delta-seconds	告知服务器，客户端希望接收一个存在时间不大于delta-seconds秒的资源
max-stale=[delta-seconds]	告知（代理）服务器，客户端愿意接收一个超过缓存时间的资源，若有定义delta-seconds则为delta-seconds秒，若没有则为任意超出时间
min-fresh=delta-seconds	告知（代理）服务器，客户端希望接收一个在小于delta-seconds秒内被更新过的资源
no-transform	告知（代理）服务器，客户端希望获取实体数据没有被转换（比如压缩）过的资源
only-if-cached	告知（代理）服务器，客户端希望获取缓存的内容（若有），而不用向原服务器发去请求
cache-extension	自定义扩展，若服务器不识别该值将被忽略掉

响应头：

字段名称	说明
public	表明任何情况下都得缓存该资源（即使是需要HTTP认证的资源）
Private=[field-name]	表明返回报文中全部或部分（若指定了field-name则为field-name的字段数据）仅开放给某些用户(服务器指定的share-user，如代理服务器)做缓存使用，其他用户则不能缓存这些数据
no-cache	不直接使用缓存，要求向服务器发起（新鲜度校验）请求
no-store	所有内容都不会被保存到缓存或Internet临时文件中
no-transform	告知客户端缓存文件时不得对实体数据做任何改变

字段名称	说明
only-if-cached	告知（代理）服务器，客户端希望获取缓存的内容（若有），而不用向源服务器发去请求
must-revalidate	当前资源一定是向源服务器发去验证请求的
proxy-revalidate	与must-revalidate类似，但仅能应用于共享缓存（如代理）
max-age=delta-seconds	告知客户端该资源在delta-seconds秒内是新鲜的，无需向服务器发请求
s-maxage=delta-seconds	同max-age，但仅能应用于共享缓存（如代理）
cache-extension	自定义扩展值，若服务器不识别该值将被忽略掉

## 示例

```
// server.js
const http = require('http');
const fs = require('fs');

http.createServer(function (request, response) {
  console.log("request: ", request.url);

  if (request.url === '/') {
    const html = fs.readFileSync('test.html', 'utf8');

    response.writeHead(200, {
      'Content-Type': 'text/html'
    })

    response.end(html);
  }

  if (request.url === '/script.js') {
    console.log("script被请求了")
    response.writeHead(200, {
      'Content-Type': 'text/javascript',
      'Cache-Control': 'max-age=20, public' // 缓存20s, 多个值用逗号分开
    })

    response.end("console.log('script loaded')");
  }
}).listen(8888)
```

```
console.log("server listening on 8888");
```

```
<!-- test.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>









<body>

  <script src="/script.js"></script>

<script>
  setTimeout(function() {
    console.log("one")
    fetch("http://127.0.0.1:8888/script.js")
  }, 3e3)

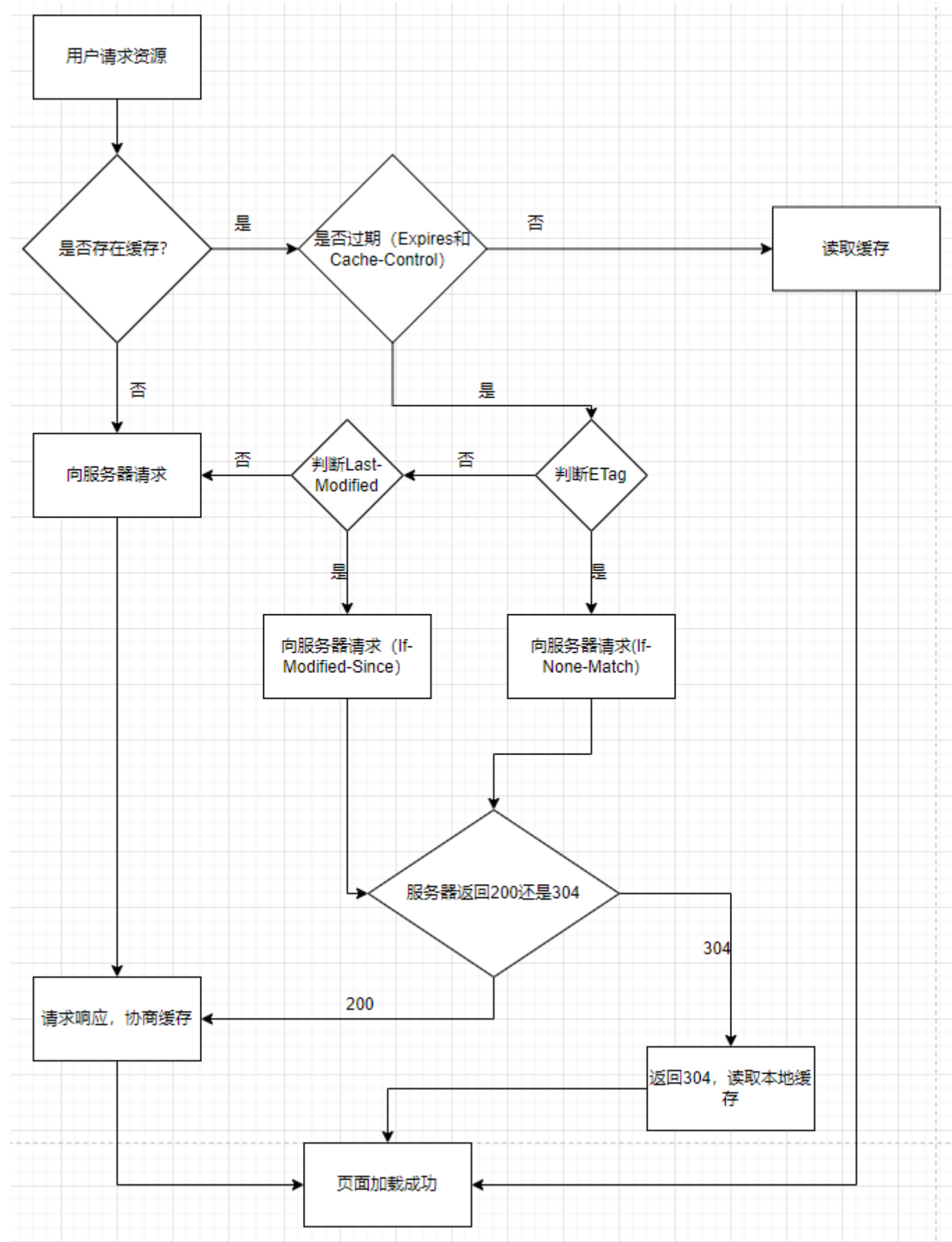
  setTimeout(function() {
    console.log("two")
    fetch("http://127.0.0.1:8888/script.js")
  }, 20e3)
</script>
</body>
</html>
```

Name	Headers	Preview	Response	Initiator	Timing
127.0.0.1					
script.js	<div><div>▼ General</div><div>Request URL: http://127.0.0.1:8888/script.js</div><div>Request Method: GET</div><div>Status Code: 200 OK</div><div>Remote Address: 127.0.0.1:8888</div><div>Referrer Policy: strict-origin-when-cross-origin</div><div>▼ Response Headers <a href="#">View source</a></div><div>Cache-Control: max-age=20, public</div><div>Connection: keep-alive</div><div>Content-Type: text/javascript</div><div>Date: Sat, 12 Feb 2022 14:20:11 GMT</div><div>Keep-Alive: timeout=5</div><div>Transfer-Encoding: chunked</div><div>▼ Request Headers <a href="#">View source</a></div></div>				
favicon.ico					

Name	Status	Type	Initiator	Size	Time	Waterfall
 127.0.0.1	200	document	Other	680 B	3 ms	
 script.js	200	script	(index)	236 B	第一次请求，设置20s缓存 3 ms	
 favicon.ico	(pending)		Other	0 B	Pending	
 script.js	200	fetch	(index):17	(disk cache)	第二次请求，20s以内读取缓存 2 ms	
 script.js	200	fetch	(index):22	236 B	第三次请求，20s以后重新发起请求 10 ms	

协商缓存

协商缓存就是强制缓存失效后，浏览器携带缓存标识向服务器发起请求，由服务器根据缓存标识决定是否使用缓存的过程。



模拟Last-Modified

```
// server.js
const http = require('http');
```

```
const fs = require('fs');
const path = require('path');

http.createServer(function (request, response) {

  console.log("request: ", request.url);

  if (request.url === '/') {
    const html = fs.readFileSync('test.html', 'utf8');

    response.writeHead(200, {
      'Content-Type': 'text/html'
    })

    response.end(html);
  }

  if (request.url === '/script.js') {
    console.log("script.js被请求了")
    const filePath = path.join(__dirname, request.url); // 拼接当前脚本的文件地址
    const stat = fs.statSync(filePath); // 获取当前脚本文件的状态
    const mtime = stat.mtime.toGMTString(); // 文件的最后修改时间
    const requestMtime = request.headers['if-modified-since']; // 来自浏览器传递的
    值

    console.log(stat);
    console.log(mtime, requestMtime);

    // 走协商缓存
    if (mtime === requestMtime) {
      response.statusCode = 304;
      response.end();
      return;
    }

    // 协商缓存失效, 重新读取数据设置Last-Modified响应头
    console.log("协商缓存Last-Modified失效");
    response.writeHead(200, {
      "Content-Type": "text/javascript",
      "Last-Modified": mtime
    })

    const readStream = fs.createReadStream(filePath);
    readStream.pipe(response);
  }
}).listen(8888)

console.log("server listening on 8888")
```




```
<!-- test.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>

  <button onclick="reqOne()">请求一</button>
  <br>
  <br>
  <hr>
  <br>
  <button onclick="reqTwo()">请求二</button>

  <script>
    function reqOne() {
      console.log("one")
      fetch("http://127.0.0.1:8888/script.js")
    }

    function reqTwo() {
      console.log("two")
      fetch("http://127.0.0.1:8888/script.js")
    }
  </script>
</body>
</html>
```

Name	Status	Type	Initiator	Size	Time	Waterfall
 127.0.0.1	200	document	Other	834 B	13 ms	
 favicon.ico	(pending)		Other	0 B	Pending	
 script.js	200	fetch	(index):22	208 B	第一次请求 8 ms	
 script.js	304	fetch	(index):27	113 B	第二次请求，请求头设置Last-Modified-Since 7 ms	

Name

127.0.0.1

favicon.ico

script.js

script.js

×

Headers

Preview

Response

Initiator

Timing

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers

View source

Connection: keep-alive

Content-Type: text/javascript

Date: Sat, 12 Feb 2022 23:19:27 GMT

Keep-Alive: timeout=5

Last-Modified: Sat, 12 Feb 2022 15:04:06 GMT

Transfer-Encoding: chunked

▼ Request Headers

View source

Accept: \*/\*

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Cache-Control: no-cache

Connection: keep-alive

Host: 127.0.0.1:8888

Pragma: no-cache

Referer: http://127.0.0.1:8888/

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36

4 requests

1.2 kB transferred

第一次请求返回Last-Modified头

10 / 14

Headers Preview Response Initiator Timing

Name

- 127.0.0.1
- favicon.ico
- script.js
- script.js

Status Code: 304 Not Modified

Remote Address: 127.0.0.1:8888

Referrer Policy: strict-origin-when-cross-origin

Response Headers View source

Connection: keep-alive

Date: Sat, 12 Feb 2022 23:19:29 GMT

Keep-Alive: timeout=5

Request Headers View source

Accept: /\*/\*

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Cache-Control: no-cache

Connection: keep-alive

Host: 127.0.0.1:8888

If-Modified-Since: Sat, 12 Feb 2022 15:04:06 GMT

Pragma: no-cache

Referer: http://127.0.0.1:8888/

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36

第二次请求附带If-Modified-Since 请求头

## 模拟ETag

```
// server.js
const http = require('http');
const fs = require('fs');
const path = require('path');
const md5 = require('md5');
http.createServer(function (request, response) {

  console.log("request: ", request.url);
  if (request.url === '/') {
    const html = fs.readFileSync('test.html', 'utf8');

    response.writeHead(200, {
      'Content-Type': 'text/html'
    })
    response.end(html);
  }

  if (request.url === '/script.js') {
    console.log("script.js被请求了")
  }
})
```

```
const filePath = path.join(__dirname, request.url); // 拼接当前脚本的文件地址
const buffer = fs.readFileSync(filePath); // 获取当前脚本文件
const fileMd5 = md5(buffer); // 文件的Md5值
const noneMatch = request.headers["if-none-match"]; // 来自浏览器端传递的值

if (noneMatch === fileMd5) {
  response.statusCode = 304;
  response.end();
  return;
}

console.log("ETag缓存失效")

response.writeHead(200, {
  "Content-Type": "text/javascript",
  "Cache-Control": "max-age=0",
  "ETag": fileMd5
})

const readStream = fs.createReadStream(filePath);
readStream.pipe(response);
}
}).listen(8888)
console.log("server listening on 8888")
```

```
<!-- test.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>

  <button onclick="reqOne()">请求一</button>
  <br>
  <br>
  <hr>
  <br>
  <button onclick="reqTwo()">请求二</button>

  <script>
    let ETag;
    function reqOne() {
      console.log("one")
      fetch("http://127.0.0.1:8888/script.js")
    }

    function reqTwo() {
```

```
        console.log("two")
        fetch("http://127.0.0.1:8888/script.js")
    }
</script>
</body>
</html>
```

Name	Status	Type	Initiator	Size	Time	Waterfall
127.0.0.1	200	document	Other	895 B	4 ms	
favicon.ico	(pending)		Other	0 B	Pending	
script.js	200 第一次请求	fetch	(index):23	228 B	5 ms	
script.js	304 第二次请求, 附带If-None-Match请求头	fetch	(index):31	113 B	4 ms	

Name

127.0.0.1

favicon.ico

script.js

script.js

× Headers Preview Response Initiator Timing

Remote Address: 127.0.0.1:8888

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers View source

Cache-Control: max-age=0

Connection: keep-alive

Content-Type: text/javascript

Date: Sun, 13 Feb 2022 00:19:31 GMT

ETag: d41d8cd98f00b204e9800998ecf8427e

Keep-Alive: timeout=5

Transfer-Encoding: chunked

▼ Request Headers View source

Accept: \*/\*

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9

Connection: keep-alive

Host: 127.0.0.1:8888

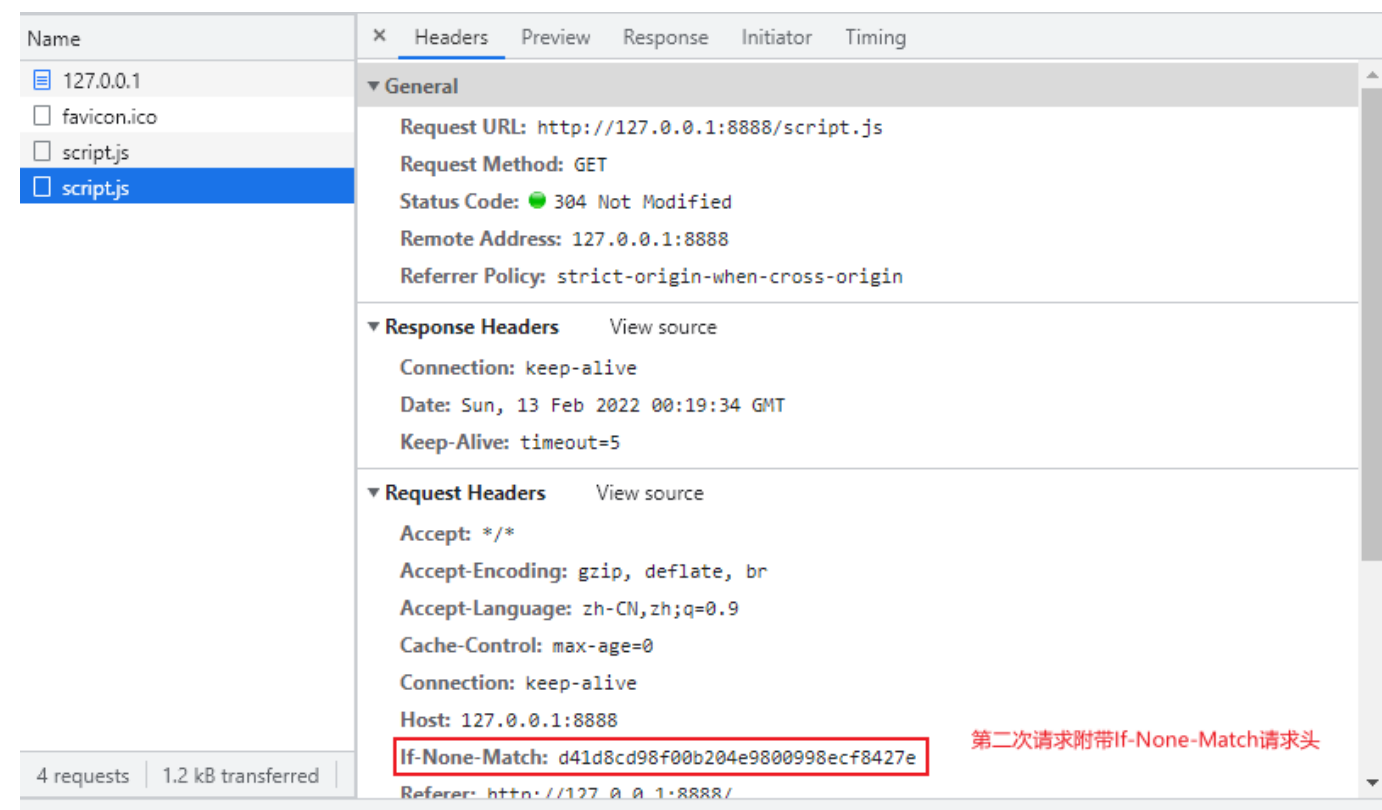
Referer: http://127.0.0.1:8888/

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98"

sec-ch-ua-mobile: ?0

4 requests | 1.2 kB transferred

第一次请求返回ETag



**Last-Modified (响应头) , If-Modified-Since (请求头)**

在浏览器第一次给服务器发送请求后，服务器会在响应头中加上Last-Modified这个字段。浏览器接收到后，如果再次请求，会自动在请求头中携带If-Modified-Since字段，这个字段的值也就是服务器传来的最后修改时间。服务器拿到请求头中的If-Modified-Since字段后，会和服务器中该资源的最后修改时间Last-Modified对比，询问服务器在该日期后该资源是否有更新，有更新的话就会将新的资源发送回来。

但是如果在本地图打开缓存文件，就会造成Last-Modified被修改，所以在HTTP/1.1中出现了ETag。

**Etag (响应头) , If-None-Match (请求头)**

ETag是服务器根据当前文件的内容，给文件生成的唯一标识，只要里边的内容有改动，这个值就会变。服务器通过响应头把这个值给浏览器。浏览器接收到ETag的值，会在下次请求时将这个值作为If-None-Match这个字段的内容，并放到请求头中发给服务器。

如果两种方式都支持的话，服务器会优先考虑ETag

**参考链接：**

- 1. [从URL输入到页面展现到底发生什么？](#)
- 2. [从输入URL开始建立前端知识体系](#)
- 3. [阮一峰HTML语言教程](#)