

Cookie 理解

Cookie 的实现(重点)

cookie存在的问题:

Session的实现

LocalStorage与Session的比较

localStorage

localStorage总结

SessionStorage (会话存储)

总结这几者之间的关系

Cache-Control

expire:

<https://github.com/clancysong/xhtml-notes/blob/master/advanced/44-Session&LocalStorage&Cache-Control.md>

<https://github.com/wojiaofengzhongzhuifeng/myBlog/issues/40>

<https://segmentfault.com/a/1190000016692936>

Cookie 理解

进公园

背景: 这个公园有一个总公园, 总公园里有许多小公园(总公园是登录页面, 小公园是域名相同的页面)

第一次进总公园, (第一次请求某个服务器)

工作人员检查你的入园是否符合条件(后端查看是否是注册以后的用户)

通过条件的话工作人员会给你一张票(后端会给你一个响应头, 这个响应头的作用是设置一个 cookie)

票的内容是只有工作人员才知道是否可以入园的字符串

第二次进总公园(第二次请求相同的域名)

你要带上这个票进公园(浏览器会在请求头带上cookie)

工作人员看到这个票, 确认里面的内容正确就给你进去(后端看到这个cookie就会让你直接进入登录后的页面)

Cookie 的实现(重点)

服务器通过 Set-Cookie 头给客户端一串字符串(背)

客户端每次访问相同域名的网页时, 必须带上这段字符串(背)

客户端要在一段时间内保存这个Cookie(背)

Cookie 默认在用户关闭页面后就失效, 后台代码可以任意设置 Cookie 的过期时间

大小在4kb以内

cookie存在的问题:

用户可以随意篡改 Cookie

Session的实现

将 SessionID（随机数）通过 Cookie 发给客户端

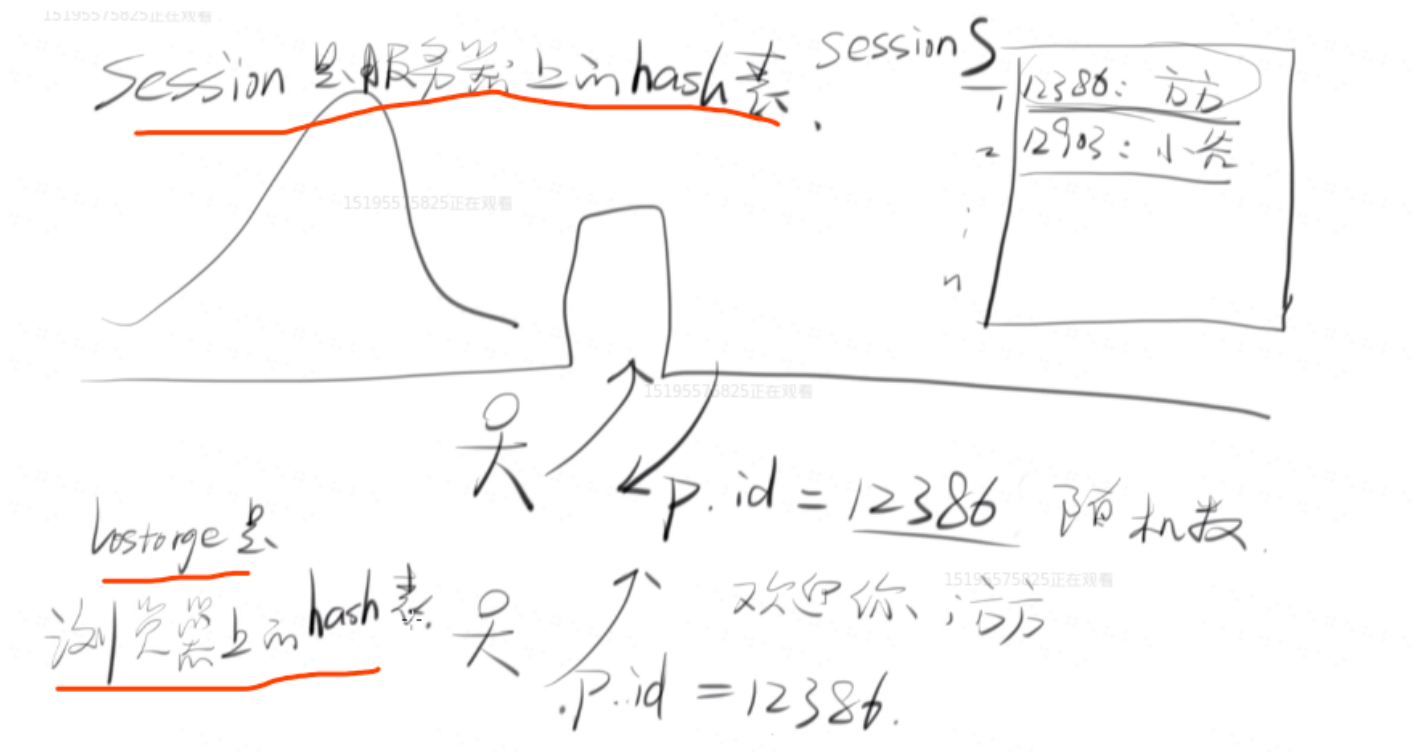
客户端访问服务器时，服务器读取 SessionID

服务器有一块内存（哈希表）保存了所有 session

通过 SessionID 我们可以得到对应用户的隐私信息，如 id、email

这块内存（哈希表）就是服务器上的所有 session

Localstorage与Session的比较



session是服务器上面的hash(表)

localStorage是浏览器上面的hash表

session就是耗内存,cookies不耗用内存;

每次登陆的时候session_id是是不一样的!但是session_id所对应的value是不变的!(其实就是相当于给value加密了,每次加完密之后的session_id是不一样的,但是value是一样的)

localStorage

总共三个api:

setItem,getItem,clear;

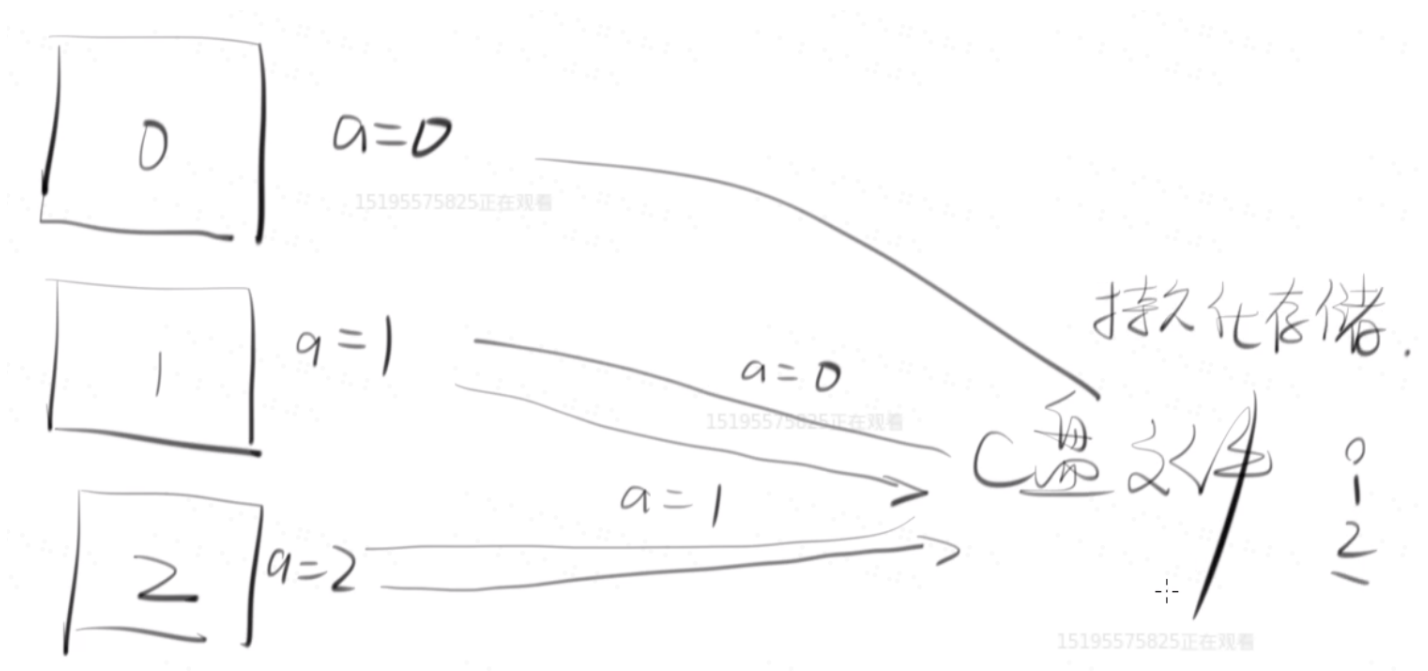
```
< '{"name":"obj"}'
> localStorage.setItem('jsonString', JSON.stringify({name:'obj'}))
< undefined
> localStorage.getItem('a')
< '1'
> localStorage.getItem('b')
< '2'
> localStorage.getItem('jsonString')
< '{"name":"obj"}'
> localStorage.clear()
```

存储字符串

localStorage实现了持久化存储:

在有localStorage之前,所有的变量,在页面刷新的那一刻,都消失了!全部被销毁;

有了localStorage之后,你可以把一些东西存到本地的C盘文件(用于以后使用);

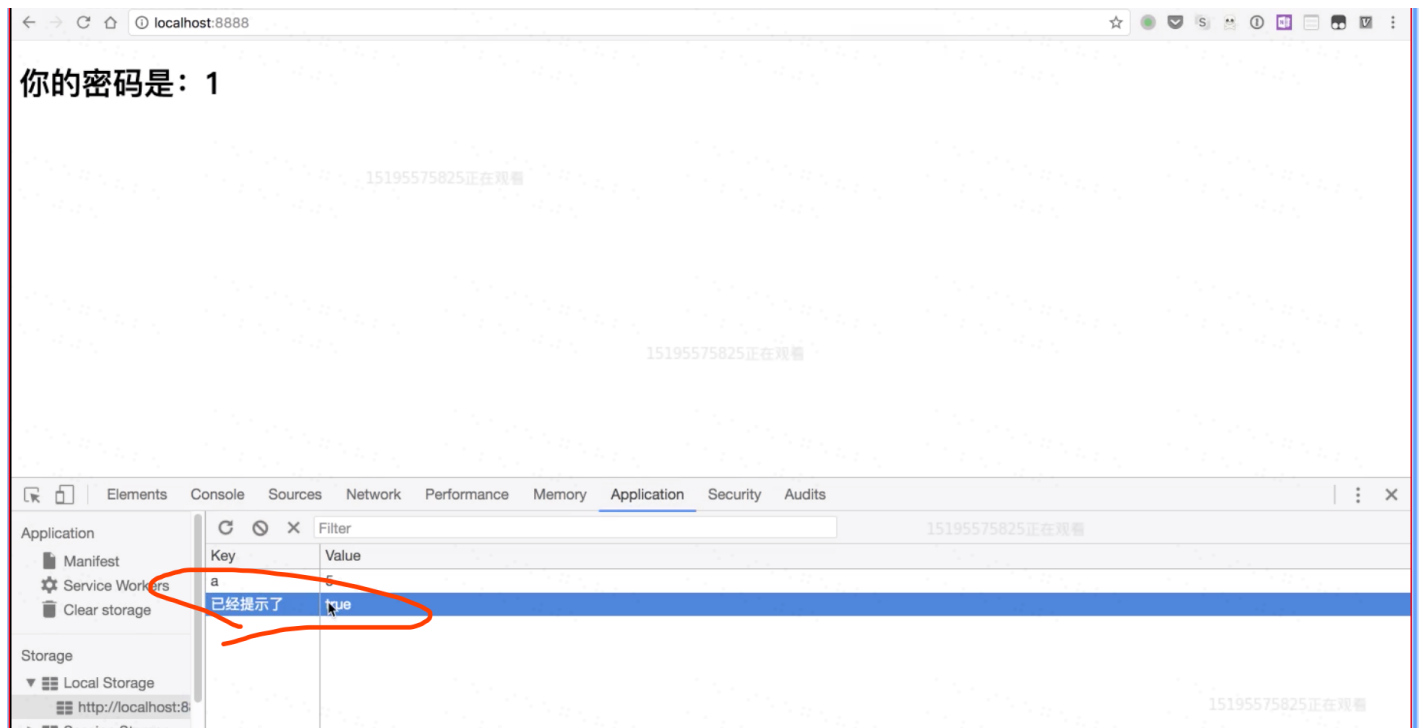
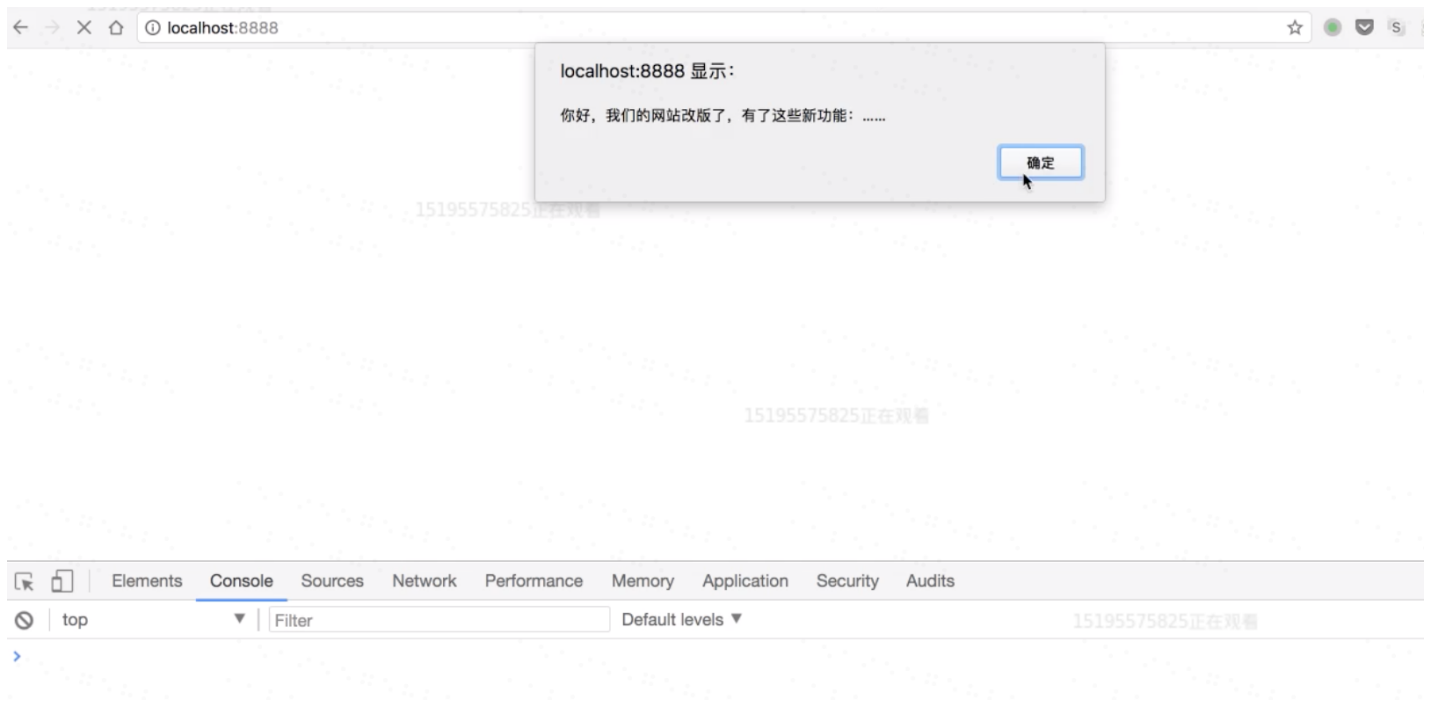


举个例子:

```
server.js  index.html
14 <!DOCTYPE html>
13 <html lang="en">
12 <head>
11   <meta charset="UTF-8">
10   <title></title>
9 </head>
8 <body>
7   <h1>你的密码是: __password__</h1>
6   <script>
5     let already = localStorage.getItem('已经提示了')
4     if(!already){
3       alert('你好, 我们的网站改版了, 有了这些新功能: .....')
2       localStorage.setItem('已经提示了', true)
1     }else{
15
1     }
3   </script>
4 </body>
5 </html>
```

这里是说啥都不做

第一次读这个页面:



第二次刷新就不会再提示了;

localStorage总结

需要记忆的:

LocalStorage 跟 HTTP 无关

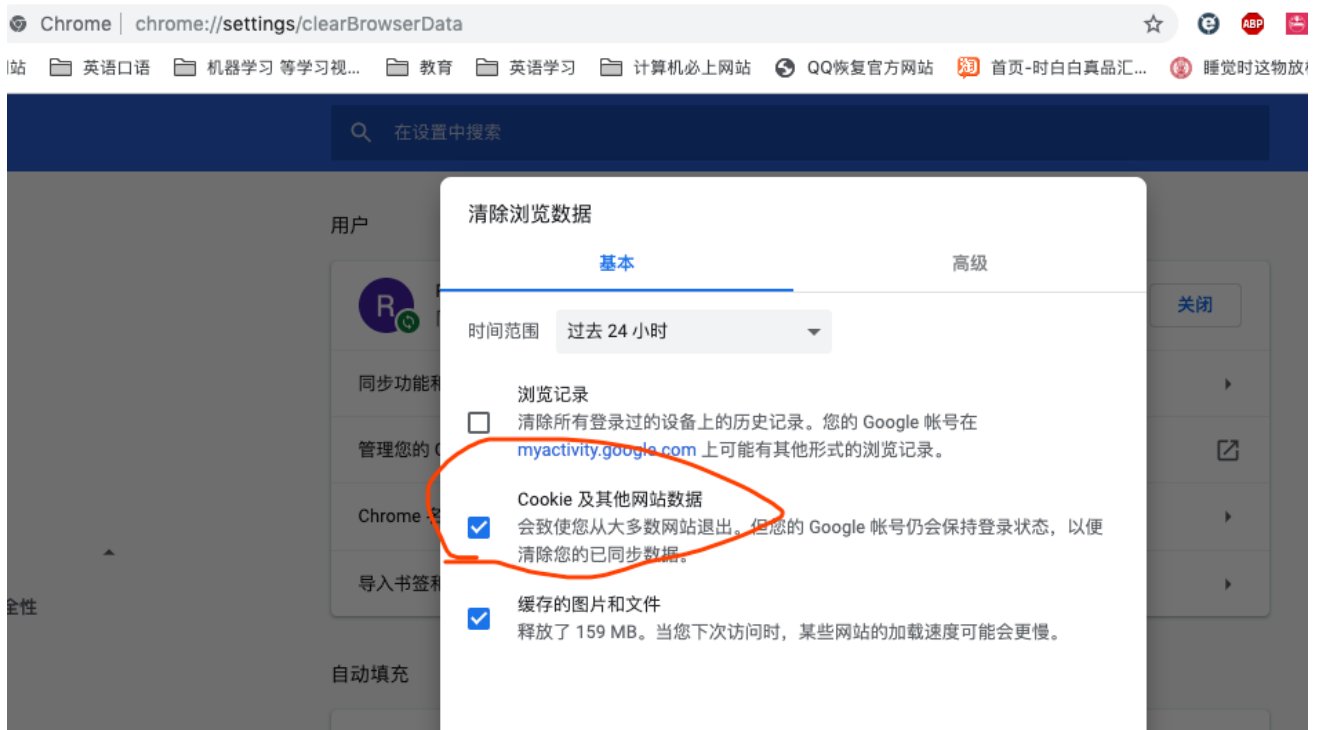
HTTP 不会带上 LocalStorage 的值

只有相同域名的页面才能互相读取 LocalStorage (没有同源那么严格)

每个域名 localStorage 最大存储量为 5Mb 左右 (每个浏览器不一样)

常用场景: 记录有没有提示过用户 (没有用的信息, 不能记录密码)

LocalStorage 永久有效，除非用户清理缓存



SessionStorage（会话存储）

1、2、3、4 同上

SessionStorage 在用户关闭页面（会话结束）后就失效。

Session 可以用 LocalStorage + 查询参数实现

总结这几者之间的关系

session是依赖于cookie的;cookie是session的基石;

cookie和localStorage的区别:

每次请求的时候cookie都会被带给服务器;而localStorage不会;

cookies一般4k,localStorage一般5M左右;

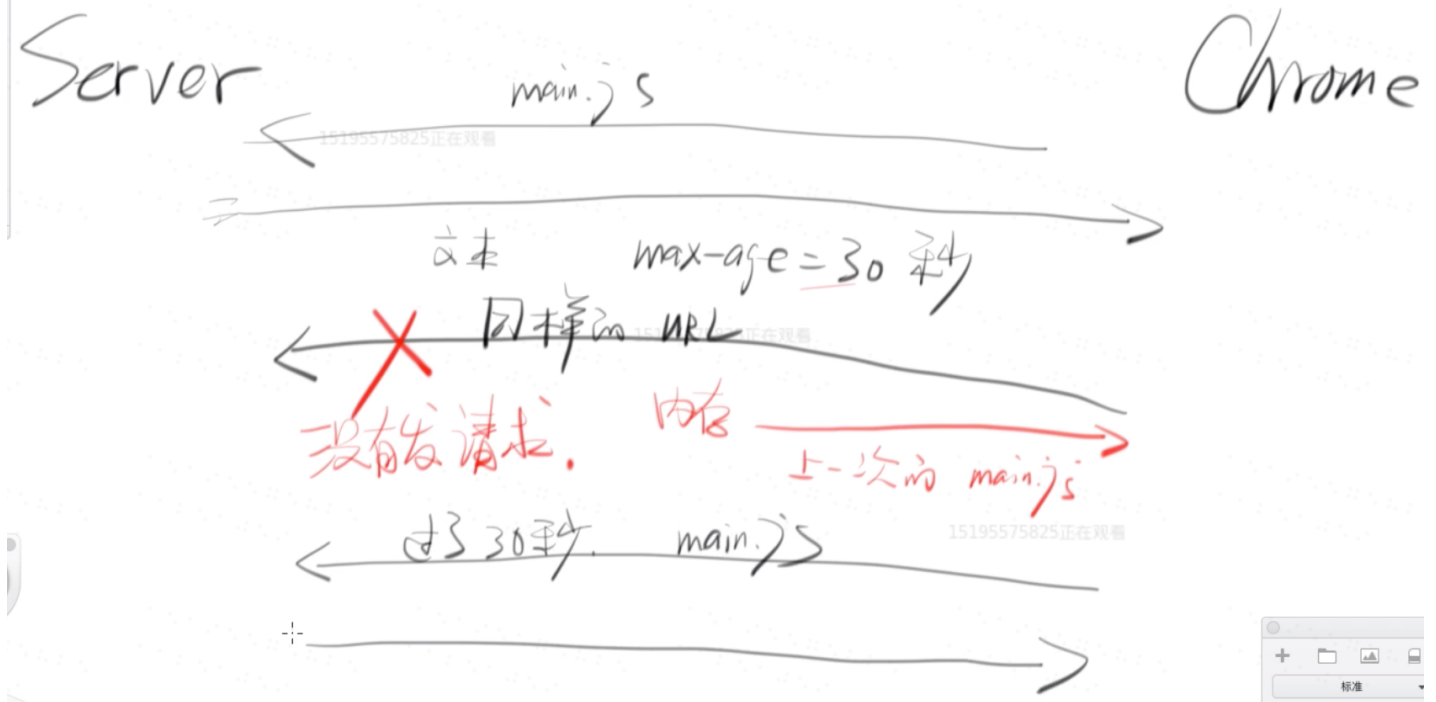
sessionstorage和localStorage区别:

SessionStorage 在用户关闭页面（会话结束）后就失效;localStorage不会;

Cookie 默认在用户关闭页面后就失效，后台代码可以任意设置 Cookie 的过期时间

Cache-Control

web性能优化:



```

4
3  if(path === '/js/main.js'){
2    let string = fs.readFileSync('./js/main.js', 'utf8')
1    response.setHeader('Content-Type', 'application/javascript;charset=utf8')
    response.setHeader('Cache-Control', 'max-age=30')
1    response.write(string)
2    response.end()
3  }else if(path === '/css/default.css'){
4    let string = fs.readFileSync('./css/default.css', 'utf8')
5    response.setHeader('Content-Type', 'text/css;charset=utf8')
6    response.write(string)
7    response.end()
8  }else if(path === '/'){
9    let string = fs.readFileSync('./index.html', 'utf8')
10   let mySession = sessions[query.sessionId]
11   let email
12   if(mySession){
13     email = mySession.sign_in_email

```

大概逻辑就是,chrome向服务器发了一次请求之后,如果在30ms之内想再次请求的话,直接可以从内存里面吧副本发送给客户端,不用去服务端了;

主要第一次是从服务器那边取得数据,后面可以不用从服务器, 直接从内存里面;

你想哪个请求被缓存, 就给个Cache-Control,max-age=xx


```

11 console.log('万万说：含查询字符串的路径\n' + pathWithQuery)
10
9 if(path === '/js/main.js'){
8   let string = fs.readFileSync('./js/main.js', 'utf8')
7   response.setHeader('Content-Type', 'application/javascript;charset=utf8')
6   response.setHeader('Cache-Control', 'max-age=30')
5   response.write(string)
4   response.end()
3 }else if(path === '/css/default.css'){
2   let string = fs.readFileSync('./css/default.css', 'utf8')
1   response.setHeader('Content-Type', 'text/css;charset=utf8')
38  response.setHeader('Cache-Control', 'max-age=30')
1   response.write(string)
2   response.end()
3 }else if(path === '/'){
4   let string = fs.readFileSync('./index.html', 'utf8')
5   let mySession = sessions[query.sessionId]
6   let email
7   if(mySession){
8     email = mySession.sign_in_email
9   }
10  let users = fs.readFileSync('./db/users', 'utf8')

```

但是上面这种机制有个缺点,就是在max-age这段时间之内, 无法获取最新的版本;
尽量把一个版本缓存,有多长时间就缓存多长时间;如果升级了,就改一下入口处的url(一般是html);

```

server.js  index.html+  sign_in.html /
6 <!DOCTYPE html>
5 <html lang="en">
4 <head>
3   <meta charset="UTF-8">
2   <title></title>
1   <link rel="stylesheet" href="./css/default.css?v=4">
7 </head>
1 <body>
2   <h1>你的密码是: __password__</h1>
3   <script src="./js/main.js?v=2"></script>
4 </body>
5 </html>

```

Name	Method	Status	Protocol	Domain	Remote Address	Type	Initiator	Size	Time	Waterfall
localhost	GET	200	http/1.1	localhost	:::1:8888	docum...	Other	398 B	23 ms	
default.css?v=4	GET	200	http/1.1	localhost	:::1:8888	stylesh...	(index)	174 KB	324 ms	
main.js?v=3	GET	200	http/1.1	localhost	:::1:8888	script	(index)	279 KB	324 ms	

可以看看知乎的:

你手机里有舍不得删的照片或视频吗?

王大胃: [3 个视频] 阅读全文

写回答 写文章 写想法

我的草稿

Live 书店 圆桌

正在等待 hm.baidu.com 的响应...

Elements Console Network Sources Performance Memory Application Security Audits Adblock Plus NIM

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

50 ms 100 ms 150 ms 200 ms 250 ms 300 ms 350 ms 400 ms 450 ms 500 ms 550 ms 600 ms 650 ms 700 ms 750 ms 800 ms 850 ms 900

Name	Headers	Preview	Response	Timing
init.js	access-control-allow-origin: *			
vendor.1c14c07589f9cc850a64.js	age: 4828949			
main.app.97355f3b69439c9887d3.js	ali-swift-global-savetime: 1569241627			
main.topstory-routes.6a6754b4fb4d43677d1c.js	cache-control: public, max-age=31536000			
hm.js?98beee57fd2ef70ccdd5ca52b9740c49	content-encoding: br			
	content-length: 14236			
	content-type: application/javascript			
	date: Mon, 23 Sep 2019 12:27:07 GMT			
	eaoleid: dehc08d115740705764541601e			

5 / 10 requests 0 B / 152 KB transferred 2.9 MB / 3.4 MB resources Finish: 897 ms

饿了么:

饿了么

饿了么都上饿了么

Elements Console Network Sources Performance Memory Application Security Audits Adblock Plus NIM

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

0 ms 400 ms 600 ms 800 ms 1000 ms 1200 ms 1400 ms 1600 ms 1800 ms 2000 ms 2200 ms 2400 ms 2600 ms 2800 ms 3000 ms 3200 m

	Status	Type	Initiator	Size	Time
...a0bb1e.js	200	script	(index)		6.9 KB
...in.js	200	script	(index)		3.9 KB
...in.js	200	script	(index)		8.4 KB
...js?t=218705	200	script	VM734:4		1.8 KB
...js	200	script	index.js?t=218705:1		8.5 KB
...77_3_f.js	200	script	index.js?t=218705:1		57.9 KB
js?t=218705	200	script	VM734:4		1.3 KB

requests 220 KB / 523 KB transferred 547 KB / 1.0 MB resources Finish: 3.39 s DOMContentLoaded: 498 ms Load: 1.33 s

expire:

expire是什么时间过期;
而max-age是还有多久过期;


```
15 /***** 从这里开始看，上面不要看 *****/
14
13 console.log('方方说：含查询字符串的路径\n' + pathWithQuery)
12
11 if(path === '/js/main.js'){
10   let string = fs.readFileSync('./js/main.js', 'utf8')
9   response.setHeader('Content-Type', 'application/javascript;charset=utf8')
8   //response.setHeader('Cache-Control', 'max-age=3000000000')
7   response.setHeader('Expires', 'Sun, 04 Feb 2018 14:40:05 GM')
6   response.write(string)
5   response.end()
4 }else if(path === '/css/default.css'){
3   let string = fs.readFileSync('./css/default.css', 'utf8')
2   response.setHeader('Content-Type', 'text/css;charset=utf8')
1   //response.setHeader('Cache-Control', 'max-age=3000000000')
0   response.setHeader('Expires', 'Sun, 04 Feb 2018 14:40:05 GM')
1   response.write(string)
2   response.end()
3 }else if(path === '/'){
4   let string = fs.readFileSync('./index.html', 'utf8')
5   let mySession = sessions[query.sessionId]
6   let email
7   if(mySession){
```