

第6课:cookie与本地存储

主讲老师: 佳楠



目录

1、 本地存储的优缺点

3、 web存储

2、 Cookie



本地存储的优缺点



优点

1. **直接访问数据。**虽然使用AJAX获取数据的速度通常已经快了很多，但将数据存储在本地机器上会让数据访问速度更快。
2. **节省网络流量。**浏览器获取一次数据，只要有用就一直保存着，而不必不断的从服务器获取数据。这能够节省网络流量。
3. **减轻服务器的压力。**如果服务器不断的响应请求，并从数据库服务器获取数据，那么服务器会负担过重。减少请求次数，可以减少服务器的工作量。
4. **最后，数据存储在本地，这使创建完全离线的应用程序变得更加可行。**

缺点

1. **没有任何同步支持。**设想一下，你已经将数据从服务器复制到了浏览器，如何处理数据同步呢？如果出现冲突会怎样？
2. **存储限制模糊。**作为开发人员，我们讨厌模糊。我们希望准确的知道可以使用多少资源。

Cookie



Cookie是开发人员如今可以使用的最古老、最稳定的客户端存储形式。当然，Cookie不是最好的方法，但它是一种选择，在将来的某个时候，你也许不得不使用（或修改）应用了Cookie的代码。

众所周知，每当一个浏览器请求一个资源，就会有一组header随请求一起发送。那些header包含各种类型的数据，其中包括有关浏览器的信息以及它需要的数据形式。反过来，服务器也会往回发送header。基本上，每次你看到浏览器渲染一个Web页面，就有一组你看不到的header被发送

Cookie使用HTTP header 发送，具体来说是为名为“Cookie”的HTTP header，由浏览器发送到服务器，又从服务器发送到浏览器

默认情况下，浏览器没有限制它可以拥有Cookie数量

Cookie 对应唯一的域名。这意味着在foo.com上设置Cookie 值不能用于goo.com。Cookie 也可以对应唯一的子域名。例如，app.foo.com是Foo 网站的一个独立的子域名。你可以创建只有app.foo.com可以读取的Cookie，可以创建www.foo.com和app.foo.com都可以读取的Cookie。

更复杂的做法是创建只对特定路径有效的Cookie。所以，你可能希望创建只有foo.com/app 可见的Cookie.最后，你可以创建只对网站的安全（HTTPS）版本有效的Cookie。显然，选用哪种方案取决于应用程序的用途，以及你认为哪里需要Cookie值。

除了设置Cookie出现的地方，还可以指定Cookie 的有效时间。对此，你有以下几个选项：

1. 只在当前会话期间存在的Cookie(从根本上说是直到浏览器关闭);
2. 永远存在的Cookie;
3. 存在特定时长的Cookie;
4. 特定时间点之后失效的Cookie.

Cookie 没有API，要使用Cookie，只需要在代码中访问document.cookie对象。
例如：可以像下面这样创建一个Cookie。

```
document.cookie = "name=jianna";
```

如果我们想要创建多个Cookie，要怎么办呢？**创建多个时，使用逗号隔开**
很简单：

```
document.cookie = "userName=jianan";  
document.cookie = "age=18";
```

```
> document.cookie  
< "userName=jianan; age=18; .
```

上面这段示例代码实际上创建了两个Cookie，而不是一个。我觉得这完全不符合逻辑，但是我们必须适应这种定义方式。

Cookie 的创建和赋值我们已经知道了，那么如何设置Cookie的过期时间呢？我们来看下示例代码：

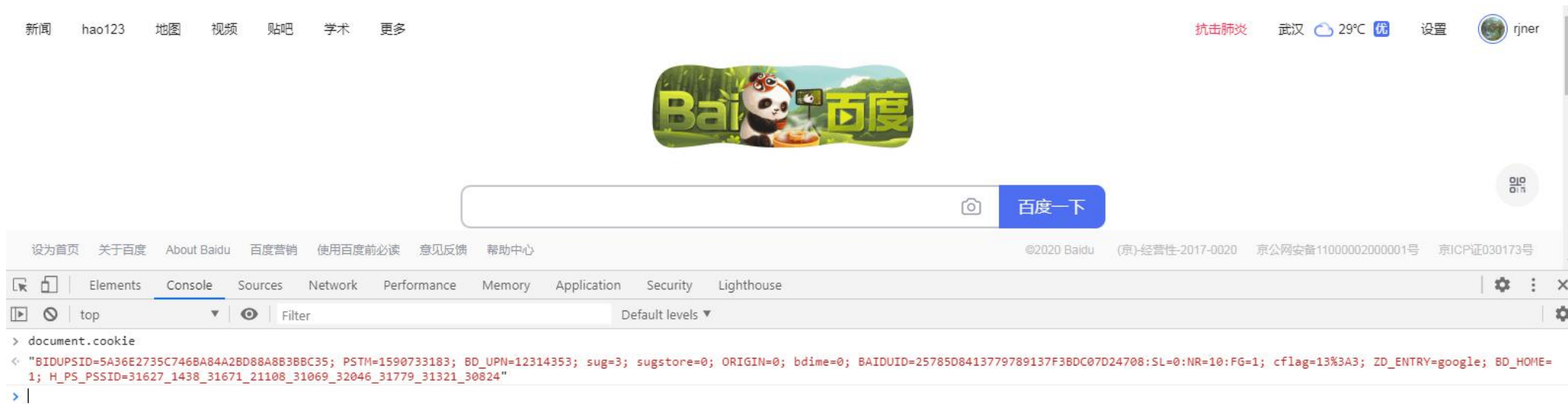
```
document.cookie = "name=jianna; expires = Mon, 15 Jun 2020 12:42:24 GMT";
```

由此可见，我们可以在Cookie值后面使用一个分号来追加元数据，我们可以进一步扩展，指定该Cookie,只对一个子域名有效。

```
document.cookie = "name=jianna; expires = Mon, 15 Jun 2020 12:42:24 GMT;domain=app.foo.com";
```

当你不这样指定元数据时，Cookie默认只对当前域名的当前路径有效，有效期是当前会话。

读取Cookie很简单，只需要简单的读取document.cookie就可以了。这样，你就可以获取特定网站的所有Cookie。



读取一个Cookie就意味着将字符串解析成多个由分号分隔的部分。另外还要注意，你无法访问任何元数据。通过document.cookie值无法获取这类信息，很多网站的Cookie数据都是加密过的。

要删除Cookie，只需要将其过期时间设置成过去的时间，从技术上讲，这个时间值无关紧要，但名称必须与你想要删除的Cookie名称一致,值可以为空。

`document.cookie = "name=jianna; expires = Mon, 15 Jun 2020 12:42:24 GMT";`

`document.cookie = "name; expires = Mon, 15 Jun 2020 12:42:24 GMT";`

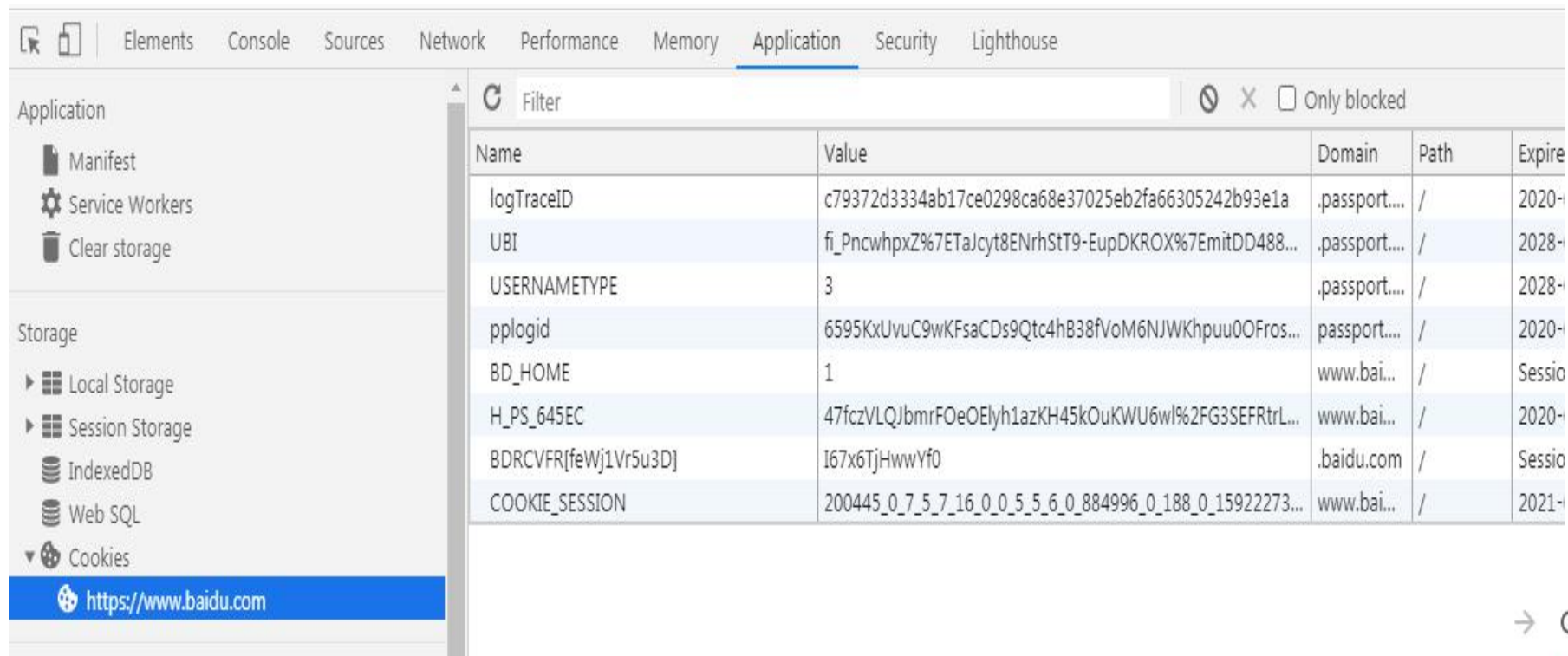
Cookie的方法封装

```
// 以天为过期时间单位设置Cookie
function setCookie(name, value, day) {
    let date = new Date()
    date.setDate(date.getDate() + day)
    let expires = date.toGMTString()
    document.cookie = `${name} = ${value}; "expires = " + ${expires}`
}
```

```
// 获取cookie
function getCookie(name) {
    let cookies = document.cookie
    let cookiesArr = cookies.split(';')
    for( let i = 0; i < cookiesArr.length; i++) {
        let cook = cookiesArr[i].split('=')
        if(cook[0].trim() === name) {
            return cook[1]
        }
    }
}
```

```
// 删除cookie
function delCookie(name, value) {
    setCookie(name, value, -1)
}
```

Cookie的方法使用之查看网站cookie数据



The screenshot shows the Chrome DevTools Application tab. The left sidebar has a tree view with 'Application' expanded, showing 'Storage' > 'Cookies'. The main pane displays a table of cookies for the selected page.

Name	Value	Domain	Path	Expire
logTraceID	c79372d3334ab17ce0298ca68e37025eb2fa66305242b93e1a	.passport....	/	2020-
UBI	fi_PncwhpxZ%7ETaJcyt8ENrhStT9-EupDKROX%7EmitDD488...	.passport....	/	2028-
USERNAMETYPE	3	.passport....	/	2028-
pplogid	6595KxUvuC9wKFsaCDs9Qtc4hB38fVoM6NJWKhpua0OFros...	passport....	/	2020-
BD_HOME	1	www.bai...	/	Sessio
H_PS_645EC	47fczVLQJbmrFOeOElyh1azKH45kOuKWU6wl%2FG3SEFRtrL...	www.bai...	/	2020-
BDRCVFR[feWj1Vr5u3D]	I67x6TjHwwYf0	.baidu.com	/	Sessio
COOKIE_SESSION	200445_0_7_5_7_16_0_0_5_5_6_0_884996_0_188_0_15922273...	www.bai...	/	2021-



The screenshot shows the Chrome address bar for the URL `https://www.baidu.com`. A security warning is displayed, stating '连接是安全的' (Connection is secure). Below the warning, it shows '证书 (有效)' (Certificate (valid)) and '(使用了 28 个) Cookie' (Used 28 cookies), which is highlighted with a red rectangle. At the bottom, there is a '网站设置' (Site settings) button.

Web存储



Web存储有两个版本：**本地存储(Local Storage)**和**会话存储(Session Storage)**。两者使用完全相同的API，但本地存储会持久存在（除非被用户清除），而会话存储只要浏览器关闭就会消失。因为大多数人都使用持久化版本，所以大多数开发人员使用和谈论的都是本地存储。

和Cookie类似的是，Web存储是与域名一一对应的，和Cookie不同的是，无法让app.foo.com使用www.foo.com存储的数据。

Web存储的限制一般为5-10M，一般来说都会够用，除非你要存储大数据包，但是并不建议这样做，因为如果超出了限制，则Chrome、Firefox和Safari浏览器都会报告一个你可以在代码中处理的错误。

Web存储API有如下4个简单的方法（注意：会话存储的用法与本地存储用法完全相同）

1. localStorage.setItem：设置特定键的值
2. localStorage.getItem：获取特定键的值
3. localStorage.removeItem：删除特定键与其关联的值
4. localStorage.clear：删除所有存储的键和值(但值限于发出请求的特定域名)

虽然Web存储提供了API，但仍然可以像对待简单的JavaScript对象那样处理数据。例如下面的代码：

localStorage.setItem['age'] = 18 会写入数据

console.log(localStorage['age']) 会读取数据

一般这样的读取只用于开发调试中，功能开发时还是要使用API方法。

在Web 存储中存储什么数据。Web 存储仅支持字符串数据。这有时会引起混淆。看看下面这段代码。

```
var names = ['jlanan', 'nannan']  
localStorage.setItem('names', names)
```

```
> var names = ['jlanan', 'nannan']  
  localStorage.setItem('names', names)  
< undefined  
> localStorage.getItem('names')  
< "jlanan,nannan"  
> |
```

上面这段代码可以正常运行。不过，它会存储数组的字符串版本，而不是数组本身，也就是说，如果我们调用 `localStorage.getItem('names')`，那么将得到字符串“jlanan, nannan”，而不是期望的数组

对于之前的情况，我们就要使用一种相当简单的变通方案：JSON 编码。

将复杂的数组转换成JSON，然后在获取值的时候通过解码进行还原，就可以轻松地将复杂数据存入Web存储了，下面是上述代码片段的修正版本，它使用了现在浏览器提供的JSON对象。

```
var names = ['jianan', 'nannan']  
localStorage.setItem('names', JSON.stringify(names))
```

```
> var names = ['jianan', 'nannan']  
  localStorage.setItem('names', JSON.stringify(names))  
< undefined  
  
> localStorage.getItem('names')  
< ["jianan","nannan"]  
  
>
```

将值重新读取到数组里也非常简单：

```
let storNames = JSON.parse(localStorage.getItem("names"))
```

```
> let storNames = JSON.parse(localStorage.getItem("names"))  
< undefined  
  
> storNames  
< ▶ (2) ["jianan", "nannan"]  
  
> |
```

Web存储之localStorage查看

百度一下

设为首页 关于百度 About Baidu 百度营销 使用百度前必读 意见反馈 帮助中心

©2020 Baidu (京)-经营性-2017-0020 京公网安备11000002000001号 京ICP证030173号

Elements Console Sources Network Performance Memory Application Security Lighthouse

Application

- Manifest
- Service Workers
- Clear storage

Storage

- Local Storage
 - https://www.baidu.com**
 - Session Storage
 - IndexedDB
 - Web SQL
 - Cookies
 - https://www.baidu.com

Cache

- Cache Storage
- Application Cache

Background Services

Filter

Key	Value
ECOM_PC_SESSION	{"showItem":["d931a56c00018d21_0_0_1_1_2_5_0_0_1_1_6_0_189_0_188_0_1592227365_0_1592227177"...
BASE64_BOTTAG	0
BDSUGSTORED	[{"q":"%E5%89%8D%E7%AB%AF%E6%8A%80%E8%83%BD%E6%A0%88%E5%9B%BE%E7%89%87","p":...
safelconHis	2020-6-2~1591076827872,2020-6-3~1591163878197
wwwPassLogout	0
BIDUPSID	5A36E2735C746BA84A2BD88A8B3BBC35
names	["jianan","nannan"]

```
{showItem: ["d931a56c00018d21_0_0_1_1_2_5_0_0_1_1_6_0_189_0_188_0_1592227365_0_1592227177",...],...}  
  clickItem: []  
    ▶ showItem: ["d931a56c00018d21_0_0_1_1_2_5_0_0_1_1_6_0_189_0_188_0_1592227365_0_1592227177",...]
```

下期再见

