Does every web request send the browser cookies?

Asked 10 years, 7 months ago Active 3 months ago Viewed 73k times

188

66

Does every web request send the browser's cookies?

I'm not talking page views, but a request for an image, `.js` file, etc.

Update If a web page has 50 elements, that is 50 requests. Why would it send the SAME cookie(s) for each request, doesn't it cache or know it already has it?

cookies

edited Mar 10 '19 at 4:00
Rich
4,982 ● 9 ● 29 ● 55

asked Aug 26 '09 at 16:52
mrblah
80.7k ● 129 ● 282 ● 398

8 I don't think that caching is possible in this situation -- we're talking about the browser sending data to the server, not the other way around. You can't say for sure that the server "already has it" after the user has sent one request, for a lot of reasons. There may be a large number of servers that don't talk to each other; the server may not want (or have room) to remember anything at all about previous requests -- HTTP is supposed to be stateless; every request should be independent of the rest. For this reason, cookies, like authentication credentials, must be sent with every request. – Ian Clelland Aug 26 '09 at 17:08

I mentioned why caching doesn't make sense for cookies in an update to my answer: stackoverflow.com/a/1336178/102960 – igorsantos07 Jun 18 '18 at 23:03

7 Answers

Active Oldest Votes

188

edited Mar 13 '13 at 9:45
Yeo
8,594 ● 5 ● 49 ● 74

answered Aug 26 '09 at 16:54
Ian Clelland
36.7k ● 7 ● 73 ● 83

97 This, incidentally, is why page speed tools like Google Page Speed or Yahoo's YSlow recommend serving static content from a separate, cookie-free domain. – ceejayoz Aug 26 '09 at 17:05

I mentioned about serving content from a separate domain in my updated answer: stackoverflow.com/a/1336178/102960 – igorsantos07 Jun 18 '18 at 23:03

Is it true that browser send Site2 Cookies when there is a HTTP Redirection from Site1 to Site2? – Zeigeist Oct 2 '18 at 22:18

90

As others have said, if the cookie's host, path, etc. restrictions are met, it'll be sent, 50 times.

But you also asked why: because cookies are an HTTP feature, and HTTP is stateless. HTTP is designed to work without the server storing any state between requests.

In fact, the server doesn't have a solid way of recognizing which user is sending a given request; there could be a thousand users behind a single web proxy (and thus IP address). If the cookies were not sent every request, the server would have no way to know which user is requesting whatever resource.

Finally, the browser has no clue if the server needs the cookies or not, it just knows the server instructed it to send the cookie for any request to foo.com, so it does so. Sometimes images need them (e.g., dynamically-generated per-user), sometimes not, but the browser can't tell.

answered Aug 26 '09 at 17:10
derobert
42.9k ● 7 ● 81 ● 117

1 Is this true with HTTP 1.1, which is a multiplexing scheme? I.e., the requests are bundled into a single TCP connection. Of course every request is received with a copy of the cookie attached. But if the concern is lots of transmission duplication, HTTP 1.1 is in a position to optimize. Though I don't know if it actually does... – Chris Noe Aug 26 '09 at 19:32

1 Then the issue becomes "which requests did the browser intend to attach the cookies to?" The server sets the policy with the cookie, to decide which domains, and which URL paths, the cookie should be sent back to, but then it forgets it. You'd need a way to specify that certain requests in the connection had the cookie, and others didn't. That definitely doesn't exist in HTTP/1.1, except by explicitly including them in every request. Honestly, a better (standards-compatible) solution for reducing bandwidth would be client-side gzip

Accept-Encoding (were servers to send that field, HTTP/1.1 §14.3 says its a request header). And the problem is that it could vary by URL even on the same server, and can change over time, so making it work would be non-trivial. – derobert Aug 27 '09 at 4:17

1 @Chris: No, keepalive just saves TCP connection setup/teardown overhead, that's all. Full headers are still sent for every request. However, pipelining (sending multiple requests w/o waiting for the response) can help greatly. HTTP/1.1 §8.1 gives details. – derobert Aug 27 '09 at 4:22

20

Yes. Every request sends the cookies that belong to the same domain. They're not cached as HTTP is stateless, what means every request must be enough for the server to figure out what to do with it. Say you have images that are only accessible by certain users; you *must* send your auth cookie with every one of those 50 requests, so the server knows it's you and not someone else, or a guest, among the pool of requests it's getting.

Having said that, cookies might not be sent given other restrictions mentioned in the other responses, such as HTTPS setting, path or domain. Especially there, an important thing to notice: cookies are not shared between domains. That helps with reducing the size of HTTP calls for static files, such as the images and scripts you mentioned.
Example: you have 4 cookies at `www.stackoverflow.com`; if you make a request to `www.stackoverflow.com/images/logo.png`, all those 4 cookies will be sent.
However, if you request `stackoverflow.com/images/logo.png` (notice the subdomain change) or `images.stackoverflow.com/logo.png`, those 4 cookies won't be present - but maybe those related to these domains will.

You can read more about cookies and images requesting, for example, at this StackOverflow Blog Post.

edited Jun 18 '18 at 22:59

answered Aug 26 '09 at 16:59
igorsantos07
3,616 ●4 ●34 ●55

8

No. Not every request sends the cookies. It depends on the cookie configuration and client-server connection.

For example, if your cookie's `secure` option is set to `true` then it must be transmitted over a secure HTTPS connection. Means when you see that website with HTTP protocol then these cookies won't be sent by browsers as the secure flag is true.

answered Jan 10 '17 at 12:00
Iman Sedighi
5,243 ●4 ●38 ●48

7

Cookie has a "path" property. If "path=/" , the answer is Yes.

edited Jul 25 '17 at 6:54

answered Jul 25 '17 at 6:22
neoedmund
429 ●4 ●12

Yes, you could orgnize your site/app structure such that all URLs requiring cookies are wthin `/app/` or suchlike - it would retain portability without needing separate subdomains to eliminate redundant overhead. Or you could ditch the now useless Google Analytics for a start. I've seen cookie headers so long I wonder whether my grandmother was knitting them. – Jake Oct 11 '17 at 4:25

6

3 years have passed

There's another reason why a browser wouldn't send cookies. You can add a `crossOrigin` attribute to your `<script>` tag, and the value to `"anonymous"`. This will prevent cookies to be sent to the destination server. 99.9% of the time, your javascripts are static files, and you don't generate that js code based on the request's cookies. If you have 1KB of cookies, and you have 200 resources on your page, then your user is uploading 200KB, and that might take some time on 3G and have zero effect on the result page. Visit [HTML attribute: crossorigin] for reference.

edited Jan 1 at 13:52
jameshfisher
22.5k ● 19 ● 82 ● 132

answered Aug 14 '12 at 10:19
gilm
6,277 ● 3 ● 37 ● 39

Please explain. – Jake Oct 11 '17 at 4:30

4 @Jake you can add a crossOrigin attribute to your <script> tag, and the value to "anonymous". This will prevent cookies to be sent to the destination server. 99.9% of the time, your javascripts are static files, and you don't generate that js code based on the request's cookies. If you have 1KB of cookies, and you have 200 resources on your page, then your user is uploading 200KB, and that might take some time on 3G and have zero effect on the result page. Visit developer.mozilla.org/en-US/docs/Web/HTML/… for reference. – gilm Oct 11 '17 at 17:54

4

I know this is an old thread. But I've just noticed that most browsers won't sent cookies for a domain if you add a trailing dot. For example `http://example.com.` won't receive cookies set for `.example.com`. Apache on the other hand treats them as the same host. I find this useful to make cross domain tracking more difficult for external resources I include, but you could also use it for performance reasons. Note this brakes validation of `https` certificates. I've run a few tests using browsershots and my own devices. The hack works on almost all browsers except for safari (mobile and desktop), which will include cookies in the request.

edited Aug 10 '14 at 11:59

answered Aug 8 '14 at 16:56
Gellweiler
647 ● 6 ● 19

How does it "make cross domain tracking more difficult for external resources I include"? Are you talking about Farcebook Like and such widgets - which we know track browsing of accidentally-still-logged-in users? – Jake Oct 11 '17 at 5:04

Yes. It will make it more difficult, because most browsers won't send the cookies along. So if you're including something from google.com for example and you're logged in into google, google can't link the two requests. This is not guaranteed tough, some browsers sent the cookies anyway and there are less reliable and less used methods to identify users (like IP-Addresses) that will still work. The biggest drawback is, that you can't use HTTPS, which sort of makes it useless today. – Gellweiler Oct 11 '17 at 11:42