What is the difference between server side cookie and client side cookie?

Asked 8 years, 8 months ago Active 9 months ago Viewed 134k times

109

53

What is the difference between creating cookies on the server and on the client? Are these called server side cookies and client side cookies? Is there a way to create cookies that can only be read on the server or on the client?

http cookies session-cookies

edited Sep 11 '17 at 16:03
Richard Garside
78.1k ●9 ●70 ●79

asked Aug 3 '11 at 5:41
Rahul
1,629 ●3 ●16 ●20

15 There is no such thing as 'server side cookie' vs 'client side cookie'. There are only cookies, name/value pairs sent in HTTP headers with both requests and responses. – Dan Grossman Aug 3 '11 at 5:53

1 Possibly referencing Session variables, which hold data on the server. Usually there is still a session identifier that is held as a client side cookie. – AndrewR Aug 3 '11 at 5:55

In all likelihood, the question refers to the different ways cookies are encoded on the server side (i.e. the way they're encoded in the 'Cookie' and 'Set-Cookie' response header) and on the client side (i.e. the way they're encoded in the 'Cookie' request header - $Path variable and all that jazz). See RFC 2109 – Ophir Radnitz Jul 26 '12 at 11:07

4 Answers

Active Oldest Votes

135

HTTP COOKIES

Cookies are key/value pairs used by websites to store state information on the browser. Say you have a website (example.com), when the browser requests a webpage the website can send cookies to store information on the browser.

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Example answer from the server:

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: foo=10
Set-Cookie: bar=20; Expires=Fri, 30 Sep 2011 11:48:00 GMT
... rest  of the response
```

Here two cookies foo=10 and bar=20 are stored on the browser. The second one will expire on 30 September. In each subsequent request the browser will send the cookies back to the server.

```
GET /spec.html HTTP/1.1
Host: www.example.com
Cookie: foo=10; bar=20
Accept: */*
```

SESSIONS: Server side cookies

Server side cookies are known as "sessions". The website in this case stores a single cookie on the browser containing a unique Session Identifier. Status information (foo=10 and bar=20 above) are stored on the server and the Session Identifier is used to match the request with the data stored on the server.

Examples of usage

You can use both sessions and cookies to store: authentication data, user preferences, the content of a chart in an e-commerce website, etc...

Pros and Cons

Below pros and cons of the solutions. These are the first that comes to my mind, there are surely others.

Cookie Pros:

- scalability: all the data is stored in the browser so each request can go through a load balancer to different webservers and you have all the information needed to fullfill the request;
- they can be accessed via javascript on the browser;
- not being on the server they will survive server restarts;
- RESTful: requests don't depend on server state

Cookie Cons:

- storage is limited to 80 KB (20 cookies, 4 KB each)

Session Pros:

- generally easier to use, in PHP there's probably not much difference.
- unlimited storage

Session Cons:

- more difficult to scale
- on web server restarts you can lose all sessions or not depending on the implementation
- not RESTful

edited Jun 26 '19 at 2:07
Gringo Suave
21.6k ● 5 ● 72 ● 63

answered Aug 3 '11 at 10:17
filippo
2,463 ● 1 ● 14 ● 15

session pros: `secure` ? – user2167582 Jul 6 '15 at 5:13

1 why sessions more secure? If you send the session cookie over http it can be hijacked. If the site uses https security should be the same as long as you use secure cookies (encrypted, signed, etc...) – filippo Jul 9 '15 at 9:41

1 Cookies cons: makes each request bigger, potentially affecting performance. I don't know the numbers but since people do use cookieless domains for things I assume it's nontrivial. – maniexx May 29 '17 at 9:44

5 Largely misleading answer - sessions are not cookies. en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#HTTP_session You can have session variables, depending on the way session management is implemented on the server. You usually have one or more cookies which are related to the session management, by holding the session identifier. Also REST and RESTful have nothing to do with cookies or session management - REST and RESTful implementations can have sessions and cookies. – Zlatin Zlatev Nov 20 '17 at 18:33

2 See stackoverflow.com/questions/35054840/… I was not saying that sessions are not typically implemented with cookies, but that there are other options for session management, hence it is wrong to talk about session variables as server-side cookies. I was also referring to JWT when I said in year 2017 in the comment above that "REST and RESTful implementations can have sessions and cookies". Although some purists may argue that this is not the proper way for implementing a REST API. – Zlatin Zlatev Jan 17 '19 at 11:07

55

You probably mean the difference between Http Only cookies and their counter part?

Http Only cookies cannot be accessed (read from or written to) in client side JavaScript, only server side. If the Http Only flag is not set, or the cookie is created in (client side) JavaScript, the cookie can be read from and written to in (client side) JavaScript as well as server side.

edited Oct 7 '15 at 16:14
1252748
10.6k ● 24 ● 78 ● 182

answered Aug 3 '11 at 6:00
nikc.org
13.8k ● 5 ● 40 ● 77

All cookies are client *and* server

response. The server only sends the cookie when it is explicitly set or changed, while the client sends the cookie on each request.

But essentially it's the same cookie.

But, behavior can change

A cookie is basically a `name=value` pair, but after the value can be a bunch of semi-colon separated *attributes* that affect the behavior of the cookie *if* it is so implemented by the client (or server). Those attributes can be about lifetime, context and various security settings.

HTTP-only (is not server-only)

One of those attributes can be set by a server to indicate that it's an HTTP-only cookie. This means that the cookie is still sent back and forth, but it won't be available in JavaScript. Do note, though, that the cookie is still there! It's only a built in protection in the browser, but if somebody would use a ridiculously old browser like IE5, or some custom client, they can actually read the cookie!

So it seems like there are 'server cookies', but there are actually not. Those cookies are still sent to the client. On the client there is no way to prevent a cookie from being sent to the server.

Alternatives to achieve 'only-ness'

If you want to store a value only on the server, or only on the client, then you'd need some other kind of storage, like a file or database on the server, or Local Storage on the client.

edited Feb 4 '19 at 19:15

answered Aug 3 '11 at 6:28
GolezTrol
105k ● 10 ● 152 ● 184

hi, I am very new to these concepts and have some doubts. I am sorry, my questions may sound silly but I will still ask. Any help, is much appreciated - Can a cookie, that has been set on the client side, be sent to any domain ? I mean, is that not a security threat? Also, how does it work with non-browser clients, like APIs etc? – Karan Chadha May 22 '18 at 11:43

Hi @KaranChadha , if you have a question, please ask it as a formal question using the 'Ask Question' button at the top of the page. A comment thread on a 7 years old question probably won't draw the right amount of attention to it. Adding a link to this Q&A, or even to this answer specifically, is fine of course. You can use the 'share' button at the bottom of each post for that. – GolezTrol May 22 '18 at 12:26

Is this true? Client generated cookies don't seem to be transferred. If doing `document.cookie="foo=bar"` followed by `fetch("/foobar", {credentials: 'include'} )` there is no cookie being sent containing `foo=bar`. Just tried that code directly on this site using DevTools and the console. – oligofren Sep 17 '18 at 16:23

Yes it's true, says also the docs, but there are some specifics that may cause this, like the missing expires attribute. – GolezTrol Sep 17 '18 at 19:48

1 @MarinosAn Yes it can. But my answer was a bit brief when it came to the attributes that modify the behavior of the cookie, so I expanded it a little now. – GolezTrol Feb 4 '19 at 19:20

4

2. No, there is no way (I know of) to create "cookies" that can be read only on the client-side. Cookies are meant to facilitate client-server communication.
3. BUT, if you want something LIKE "client-only-cookies" there is a simple answer: Use "Local Storage".

Local Storage is actually syntactically simpler to use than cookies. A good simple summary of cookies vs. local storage can be found at:

https://courses.cs.washington.edu/courses/cse154/12au/lectures/slides/lecture21-client-storage.shtml#slide8

A point: You might use cookies created in JavaScript to store GUI-related things you only need on the client-side. BUT the cookie is sent to the server for EVERY request made, it becomes part of the http-request headers thus making the request contain more data and thus slower to send.

If your page has 50 resources like images and css-files and scripts then the cookie is (typically) sent with each request. More on this in Does every web request send the browser cookies?

Local storage does not have those data-transfer related disadvantages, it sends no data. It is great.

edited Nov 15 '18 at 4:24

answered Nov 7 '18 at 15:39
Panu Logic
1,089 ● 9 ● 18