Understanding cookies

- Cookies in action
- What are first-party and third-party cookies?

This article is part of a series on the `SameSite` cookie attribute changes that includes:

- SameSite cookies explained
- SameSite cookies recipes
- Schemeful Same-Site

A cookie is a small file that websites store on their users' machine, the information it stores travels back and forth between the browser and the website.

Each cookie is a key-value pair along with a number of attributes that control when and where that cookie is used. These attributes are used to set things like expiration dates or indicating the cookie should only be sent over HTTPS. You can set a cookie in an HTTP header or through JavaScript interface.

Cookies are one of the methods available for adding persistent state to web sites. Over the years their capabilities have grown and evolved, but left the platform with some problematic legacy issues. To address this, browsers (including Chrome, Firefox, and Edge) are changing their behavior to enforce more privacy-preserving defaults.

Learn more about Chrome's effort to protect people's privacy online and give companies and developers tools to build thriving digital businesses.

Cookies in action

Say you have a blog where you want to display a "What's new" promo to your users. Users can dismiss the promo and then they won't see it again for a while. You can store that preference in a cookie, set it to expire in a month (2,600,000 seconds), and only send it over HTTPS. That header would look like this:

```
Set-Cookie: promo_shown=1; Max-Age=2600000; Secure
```

Three cookies being sent to a browser from a server in a response

Servers set cookies using the `Set-Cookie` header.

When your reader views a page that meets those requirements—they're on a secure connection and the cookie is less than a month old—their browser will send this header in its request:

```
Cookie: promo_shown=1
```

Three cookies being sent from a browser to a server in a request

Your browser sends cookies back in the `Cookie` header.

You can also add and read the cookies available to that site in JavaScript using `document.cookie`. Making an assignment to `document.cookie` will create or override a cookie with that key. For example, you can try the following in your browser's JavaScript console:

```
→ document.cookie = "promo_shown=1; Max-Age=2600000; Secure"
← "promo_shown=1; Max-Age=2600000; Secure"
```

Reading `document.cookie` will output all the cookies accessible in the current context, with each cookie separated by a semicolon:

```
→ document.cookie;
← "promo_shown=1; color_theme=peachpuff; sidebar_loc=left"
```

JavaScript accessing cookies within the browser

JavaScript can access cookies using `document.cookie`.

If you try this on a selection of popular sites you will notice that most of them set significantly more than just three cookies. In most cases, those cookies are sent on every single request to that domain, which has a number of implications. Upload bandwidth is often more restricted than download for your users, so that overhead on all outbound requests is adding a delay on your time to first byte. Be conservative in the number and size of cookies you set. Make use of the `Max-Age` attribute to help ensure that cookies don't hang around longer than needed.

What are first-party and third-party cookies?

If you go back to that same selection of sites you were looking at before, you probably noticed that there were cookies present for a variety of domains, not just the one you were currently visiting. Cookies that match the domain of the current site, that is, what's displayed in the browser's address bar, are referred to as **first-party** cookies. Similarly, cookies from domains other than the current site are referred to as **third-party** cookies. This isn't an absolute label but is relative to the user's context; the same cookie can be either first-party or third-party depending on which site the user is on at the time.

Three cookies being sent to a browser from different requests on the same page

Cookies may come from a variety of different domains on one page.

Continuing the example from above, let's say one of your blog posts has a picture of a particularly amazing cat in it and it's hosted at `/blog/img/amazing-cat.png`. Because it's such an amazing image, another person uses it directly on their site. If a visitor has been to your blog and has the `promo_shown` cookie, then when they view `amazing-cat.png` on the other person's site that cookie **will be sent** in that request for the image. This isn't particularly useful for anyone since `promo_shown` isn't used for anything on this other person's site, it's just adding overhead to the request.

If that's an unintended effect, why would you want to do this? It's this mechanism that allows sites to maintain state when they are being used in a third-party context. For example, if you embed a YouTube video on your site then visitors will see a "Watch later" option in the player. If your visitor is already signed in to YouTube, that session is being made available in the embedded player by a third-party cookie—meaning that "Watch later" button will just save the video in one go rather than prompting them to sign in or having to navigate them away from your page and back over to YouTube.

The same cookie being sent in three different contexts

A cookie in a third-party context is sent when visiting different pages.

One of the cultural properties of the web is that it's tended to be open by default. This is part of what has made it possible for so many people to create their own content and apps there. However, this has also brought a number of security and privacy concerns. Cross-site request forgery (CSRF) attacks rely on the fact that cookies are attached to any request to a given origin, no matter who initiates the request. For example, if you visit `evil.example` then it can trigger requests to `your-blog.example`, and your browser will happily attach the associated cookies. If your blog isn't careful with how it validates those requests then `evil.example` could trigger actions like deleting posts or adding their own content.

Users are also becoming more aware of how cookies can be used to track their activity across multiple sites. However until now there hasn't been a way to explicitly state your intent with the cookie. Your `promo_shown` cookie should only be sent in a first-party context, whereas a session cookie for a widget meant to be embedded on other sites is intentionally there for providing the signed-in state in a third-party context.

You can explicitly state your intent with a cookie by setting the appropriate SameSite attribute.

To identify your first-party cookies and set appropriate attributes, check out First-party cookie recipes.

□□□ #3
□ □□□ □□□ 2026-10-31 22:22:19 UTC
□ cocobook-sync □□□ 2026-11-04 17:38:16 UTC