**Cookies vs. localStorage vs. sessionStorage**

by [**Dillion Megida**](https://dillionmegida.com/me)   |   filed under [**The Web Corner**](https://www.kirupa.com/apps/index.htm)

Data! That is the magic sauce that makes our web sites and apps interesting and useful. While a lot of the data we interact with comes from the server, an equally large part of the data we rely on is stored locally (aka client-side) by our browser. There are many reasons why our favorites web sites and apps store some data locally. The biggest reason has to do with **performance**. Data we store locally is available nearly instantaneously to our browser, whereas the data stored remotely takes time to make its way from a server to your computer.

This means if our pages rely on any data for displaying frequently accessed information or page state, this data is typically stored locally. This information could be:

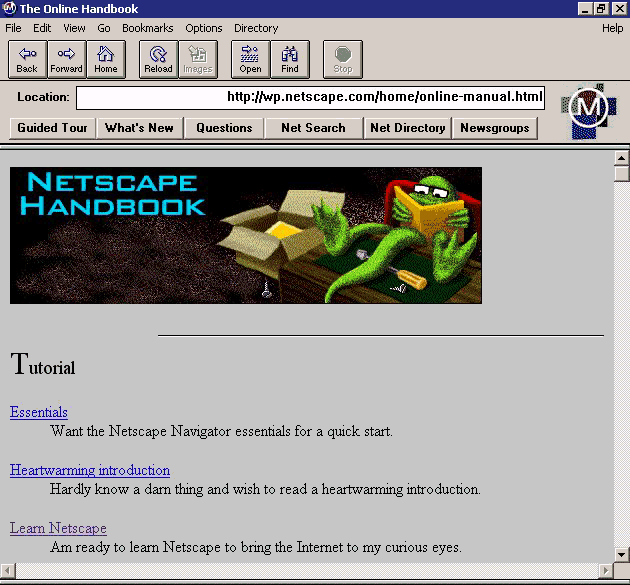
* Data from a previous browsing session like your username,  the contents of a shopping cart, items in a ToDo list, etc.
* Personalization settings that affect how your page renders - settings like color scheme, font size, whether some UI elements are visible or not
* Data you want to keep handy if the network connection goes offline
* Tracking and analytics data that needs to be updated frequently
* …and much MUCH more!

There are a bunch of ways we can tell our browsers to store data locally. Three popular ways are **cookies**, **local storage** and **session storage**. In this article, we'll learn more about these three approaches, how they work, how they differ, and when to use one over the other.

Onwards!

**Cookies**

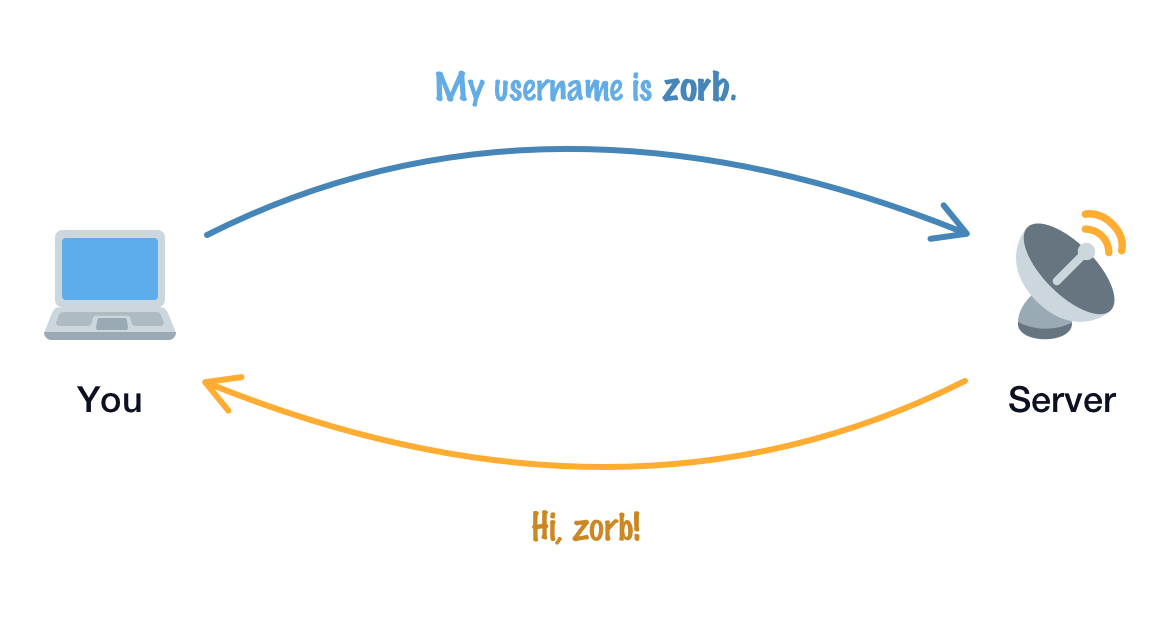
One of the oldest and most common ways to store some data is by using cookies, introduced by [Mosaic Netscape around 1994](https://en.wikipedia.org/wiki/File:Mosaic_Netscape_0.9_on_Windows_XP.png):



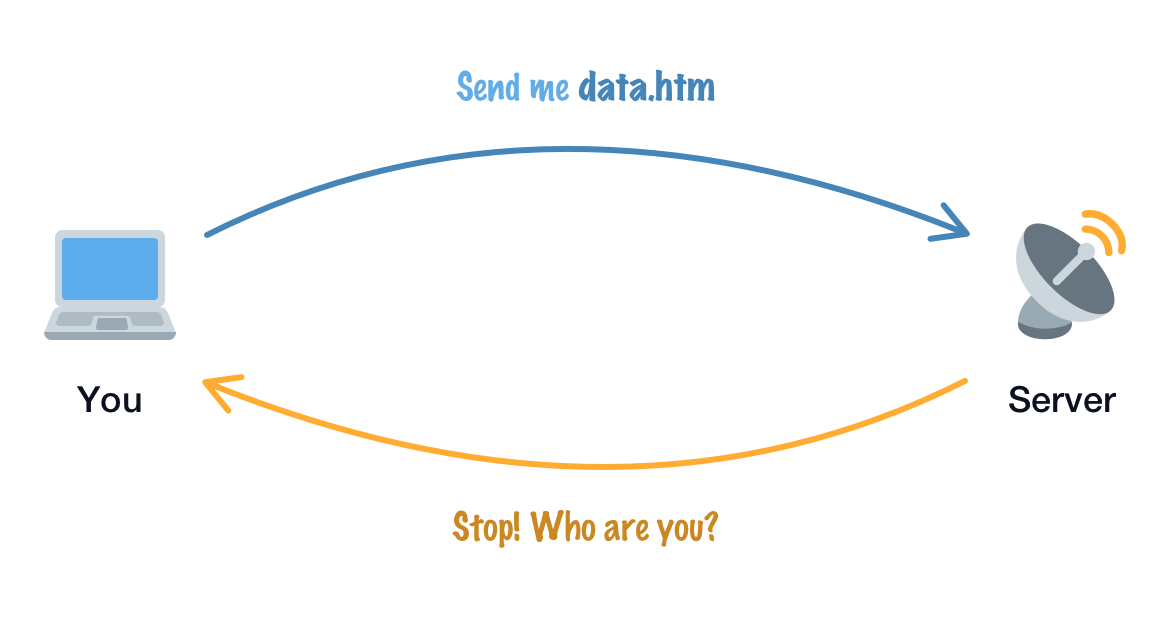
The idea behind cookies was simple. When navigating between pages, we needed a way to persist some information that would normally be lost when one page exits and another page loads. This was necessary because the relationship between our browser and a server is stateless. Each time you visit the same server, **it will treat you as if it is the first time it has ever met you.**

**The Problem**

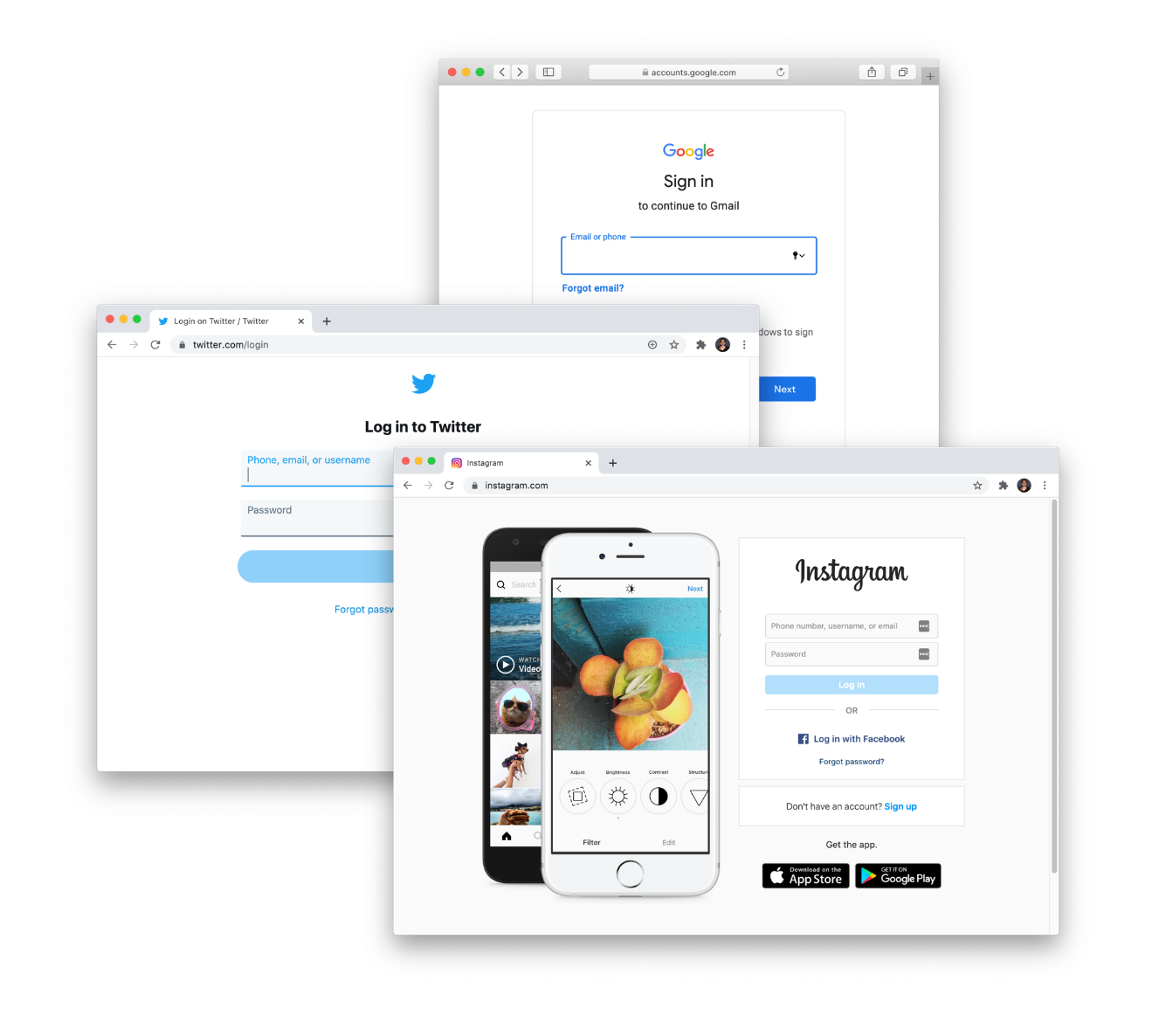
Let's say you are logging-in to access parts of a web site that requires authentication. The interaction between your browser and the server may look as follows:



After you logged-in, when you navigate deeper into the site, your expectation might be to see the secure content. With our server being stateless and having no way to remember who you are, what will actually happen is this:



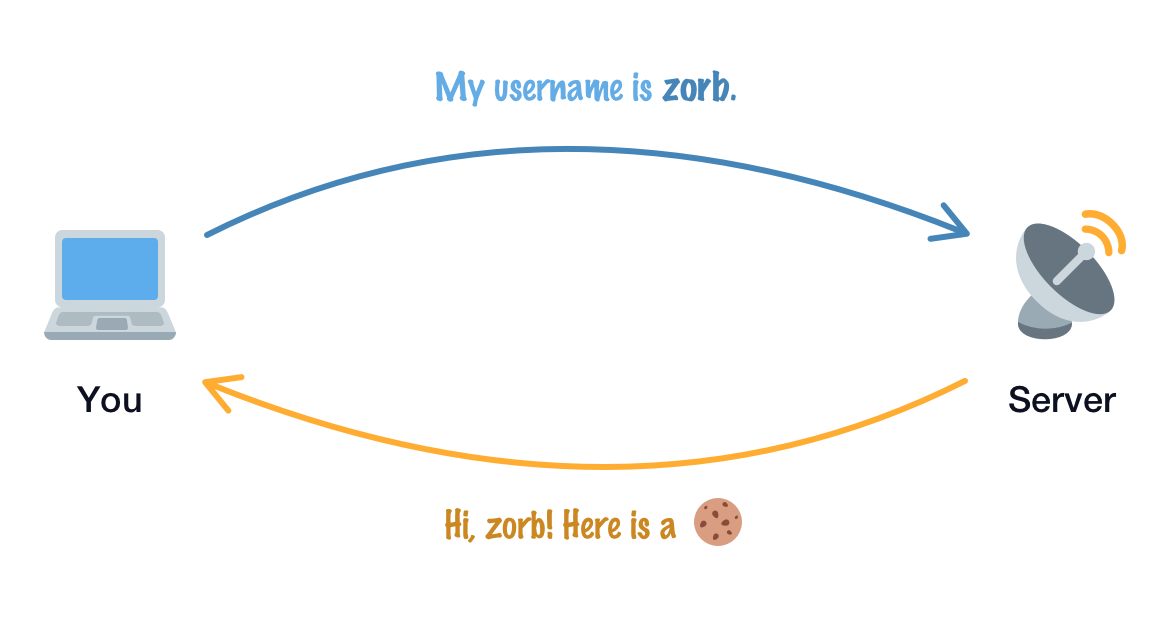
The server will have no idea that you were the authenticated browser from just a few moments ago. A server is a bit like a goldfish in the remembering department! Without a way to help our server remember us, any activity that actually requires the server to remember us won't work. Extending our authentication dilemma further, our entire browsing experience might be a series of log-in screens that get thrown at us with every navigation:



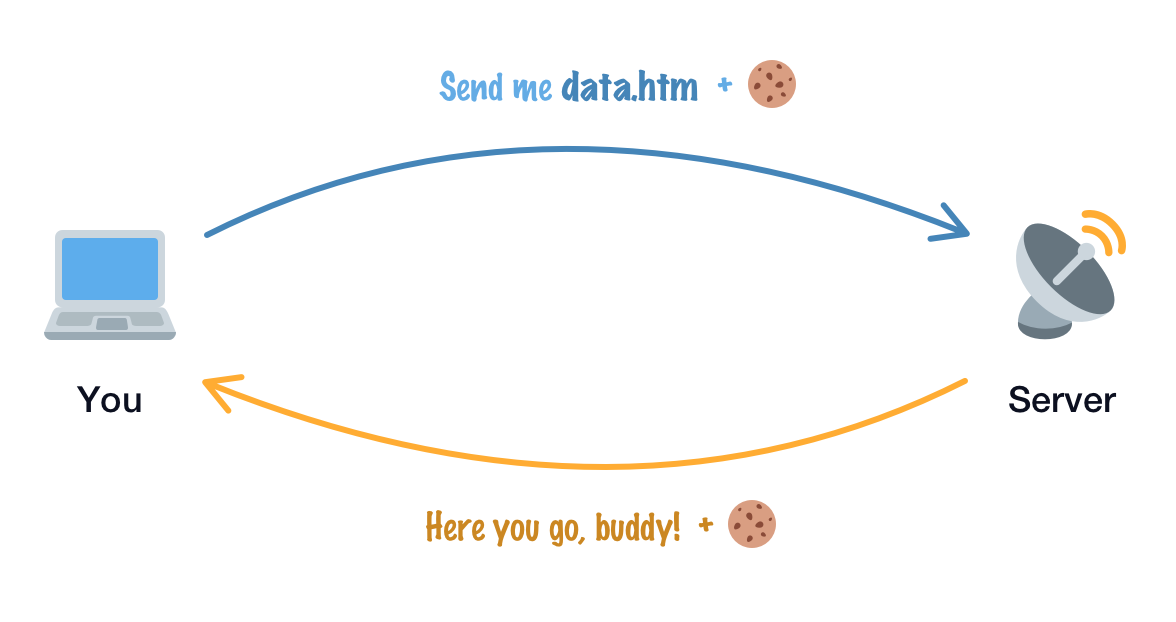
This is the stuff that nightmares are made of, right?

**Cookies FTW!**

Cookies solve this remembering problem our servers have quite nicely. A cookie is just some text-based information. A cookie is set typically by the server itself, and it will contain the right level of details to help the server identify us:



With each navigation, **our browser and server send this cookie (or cookies in some cases) along with whatever web request they are making**. While the connection between our browser and the server is still stateless, the cookie information that is passed around acts like a log of our communication and helps our server recognize us:



There are a bunch of other details that we're glossing over here, but this is a high-level overview of how cookies can help our servers remember us…and make surfing the web a more fun and seamless experience.

**What exactly is a cookie?**

As we saw earlier, a cookie is just some text-based information. This information has a structure made up of a series of **key** and **value** pairs. The following really long string is an example of what a cookie looks like:

"locale=en; \_ga=GA1.2.1665695300.1566934440; cto\_lwid=409ae255-69e8-4436-8956-08bf4b3d539e; last\_active\_role=personal; utag\_main=v\_id:016cd49217f900031bcd8acf3e2e03077003506f00fb8$\_sn:3$\_se:12$\_ss:0$\_st:1580768158108$ses\_id:1580765219002%3Bexp-session$\_pn:6%3Bexp-session; paper\_theme\_override=light; SnapABugHistory=11#; xsrf=9qh9HHD4tEqayfG0U05F9M; SnapABugUserAlias=%23; SnapABugVisit=2#1595374028"

Notice that the content stored by the cookie is a combination of human-readable information and a gibberish-looking mess of letters and numbers. That is totally normal. Some parts of the cookie will only make sense to our server and whatever unique identifier-like approach it uses to represent state. Other parts of the cookie, typically the more human-readable part, will be used by [some client-side JavaScript](https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie) to set or read some values.

The last detail to note is that not all cookies persist data the same. **Session Cookies** are ones that only exist for a particular browsing session and disappear when you close that tab. We then have **Permanent Cookies** that don't disappear when the browsing session is ended. These cookies can exist for a period of time specified in the cookie itself. They can exist for a few days, a few years, or more!

s