NODE

EXPRESS SESSIONS AND COOKIES

Browser Storage

Storage Type	Max Size	Sync/Async	Persistence	Sent with Requests	Use Case
Cookies	~4 KB	Sync	Optional expiry	✅ Yes	Auth, tracking
localStorage	~5–10 MB	Sync	Until cleared	❌ No	Simple preferences, cache
sessionStorage	~5–10 MB	Sync	Until tab closes	❌ No	Temporary tab-specific data
IndexedDB	~100+ MB	Async	Long-term	❌ No	Large, structured data
Cache API	~50–100+ MB	Async	Long-term	❌ No	Asset caching (PWA)
Memory Cache	Small	N/A	Short-term	❌ No	Performance (browser-managed)

Cookies

Sites like amazon.com seem to "know who I am." How do they do this? How does a client uniquely identify itself to a server, and how does the server provide specific content to each client?

HTTP Stateless

HTTP is a stateless protocol; it simply allows a browser to request a single document from a web server

How do the Gmail knows who I am on each request

Cookie

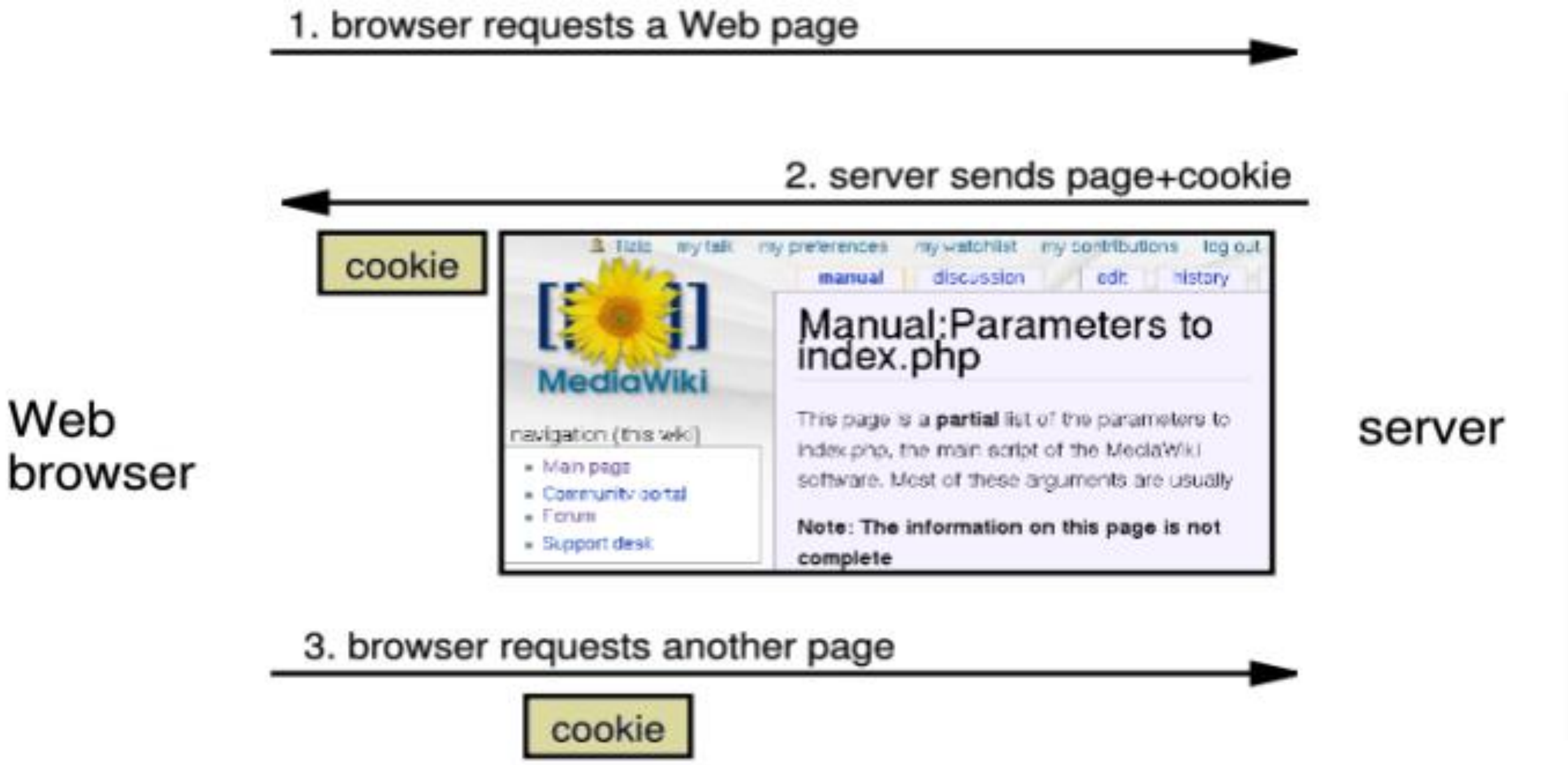
COOKIE: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests

Cookies have many uses:

- Authentication
- User tracking
- Maintaining user preferences, shopping carts, etc.

A cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request

How Cookies are sent



How Cookies are sent

When the browser requests a page, the server may send back a cookie(s) with it

If your server has previously sent any cookies to thebrowser, the browser will send them back on subsequent requests

Alternate model: client-side JS code can set/get cookies

Myths About Cookies

Myths:

- Cookies are like worms/viruses and can erase data from the user's hard disk.
- Cookies are a form of spyware and can steal your personal information.
- Cookies generate popups and spam.
- Cookies are only used for advertising.

Facts:

- Cookies are only data, not program code.
- Cookies cannot erase or read information from the user's computer.
- Cookies are usually anonymous (do not contain personal information).
- Cookies CAN be used to track your viewing habits on a particular site.

How Long does a Cookie Exist?

Session cookie : the default type; a temporary cookie that is stored only in the browser's memory

- When the browser is closed, temporary cookies will be erased
- Can not be used for tracking long-term information
- Safer, because no programs other than the browser can access them

Persistent cookie : one that is stored in a file on the browser's computer

- Can track long-term information
- Potentially less secure, because users (or programs they run) can open cookie files, see/change the cookie values, etc.

Where are the Cookies on My Computer

IE: HomeDirectory\Cookies

- e.g. C:\Documents and Settings\administrator\Cookies
- Each is stored as a .txt file similar to the site's domain name

Firefox: %APPDATA%\Mozilla\Firefox\???.default\cookies.txt (cookies.sqlite)

- View cookies in Firefox preferences: Privacy, Show Cookies...

Cookie in JavaScript

JS has a global document.cookie field (a string)

You can manually set/get cookie data from this field (sep. by ;), and it will be saved in the browser

You can't remove a cookie, but instead, you can make it expire, and why?

```js
document.cookie = "username=smith;password=12345";
```

cookie-parser

Parse Cookie header and populate req.cookies with an object keyed by the cookie names

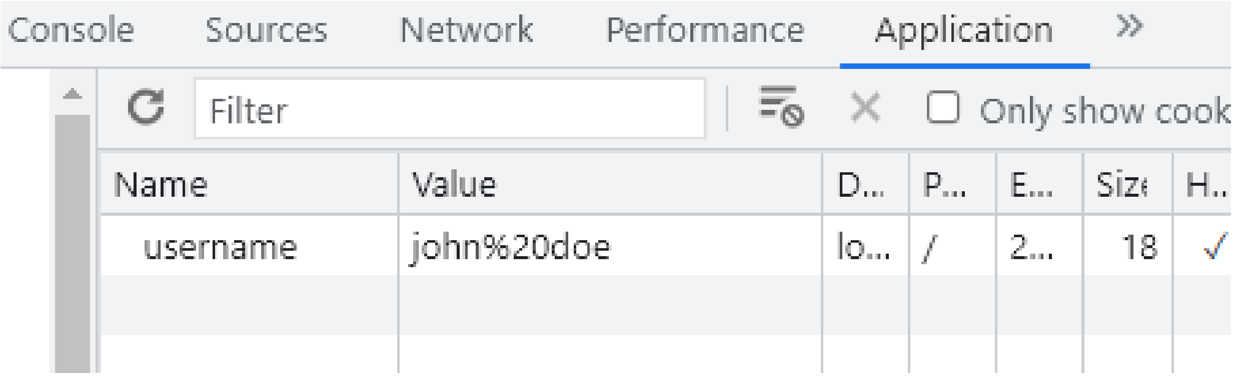
- npm install cookie-parser
- var cookieParser = require('cookie-parser')

Hello Cookies

```
var express = require('express');
var cookieParser = require('cookie-parser'); // module for parsing cookies
var app = express();
app.use(cookieParser());
```

Setting a Cookie

```
app.get('/setcookie', function(req, res){
  // setting cookies
  res.cookie('username', 'john doe', { maxAge: 900000, httpOnly: true });
  return res.send('Cookie has been set');
});
```



Getting a Cookie

```
app.get('/getcookie', function(req, res) {
    var username = req.cookies['username'];
    if (username) {
        return res.send(username);
    }
    return res.send('No cookie found');
});
```

What is a Session

Session: an abstract concept to represent a series of HTTP requests and responses between a specific Web

browser and server

- HTTP doesn't support the notion of a session, but NODE does

Sessions vs. cookies:

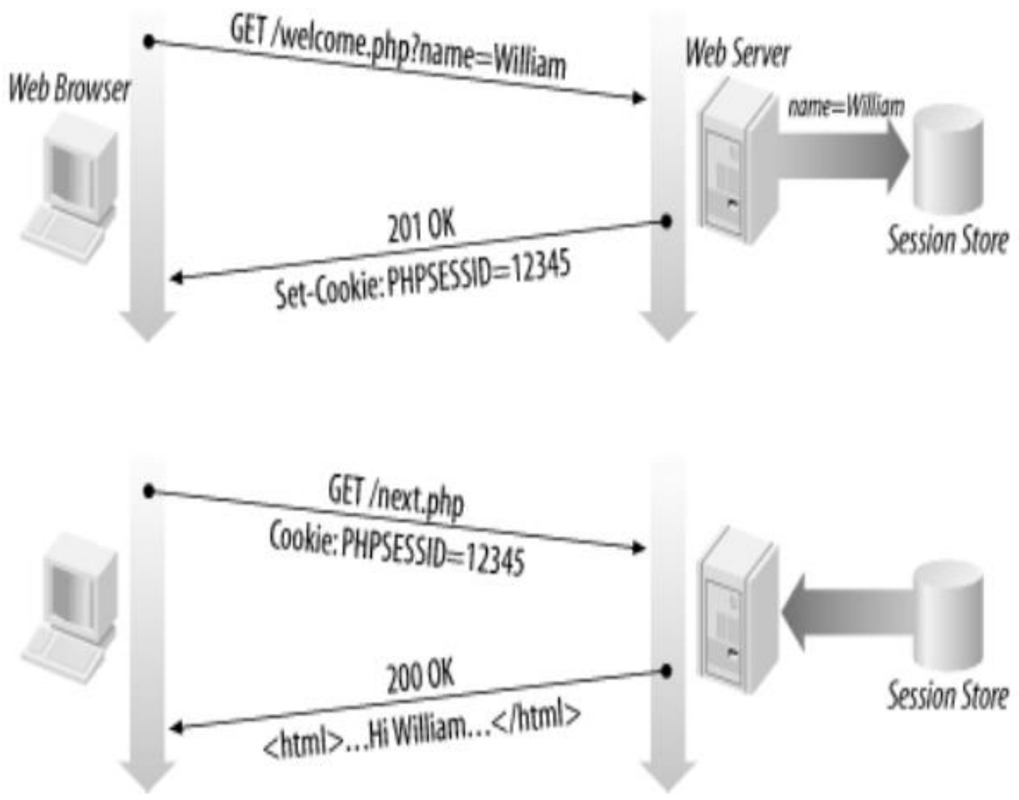
- A cookie is data stored on the client
- A session's data is stored on the server (only 1 session per client)

Sessions are often built on top of cookies:

- The only data the client stores is a cookie holding a unique **session ID**
- On each page request, the client sends its session ID cookie, and

the server uses this to find and retrieve the client's session data

How Session Works



Client's browser makes an initial request to the server

Server notes client's IP address/browser, stores some local session data, and sends a session ID back to client
Client sends that same session ID back to server on future requests

Server uses session ID to retrieve the data for the client’s session later, like a ticket given at a coat-check room

Session Timeout

Because HTTP is stateless, it is hard for the server to know when a user has finished a session

Ideally, user explicitly logs out, but many users don't

Client deletes session cookies when browser closes

Server automatically cleans up old sessions after a period of time

- Old session data consumes resources and may present a security risk
- Adjustable in server settings

What if browser don’t support cookies

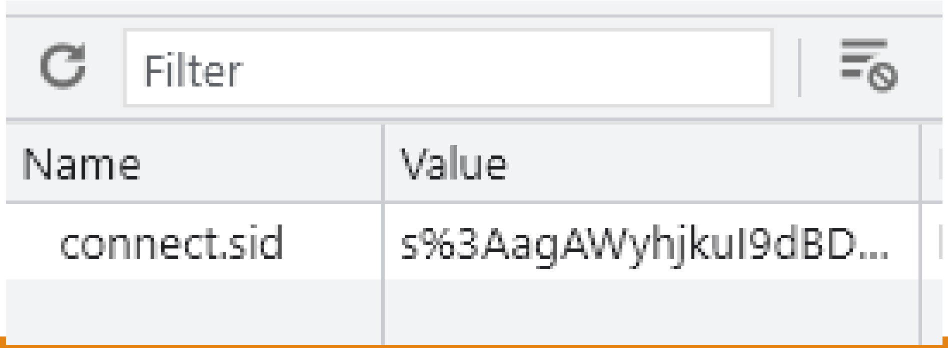
How Session will be managed

Session in Express

```
var session = require('express-session');
app.use(cookieParser());
app.use(session({secret: "Shh, its a secret!"}));
```

Set And Use Session

```
app.get('/', function(req, res){
    if(req.session.page_views){
       req.session.page_views++;
       res.send("You visited this page " + req.session.page_views + " times");
    } else {
       req.session.page_views = 1;
       res.send("Welcome to this page for the first time!");
    }
 });
```



Some Session Options

```
app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: true }
}))
```

Resave: Save Sesison even if not modified

saveUninitialized: Forces a session that is "uninitialized" to be saved to the store.

Check the code which demonstrate the Session and Cookies Usage

https://github.com/mua22/express-cookies-sessions-demo