Introduction to Cookies and Authentication

Cookies

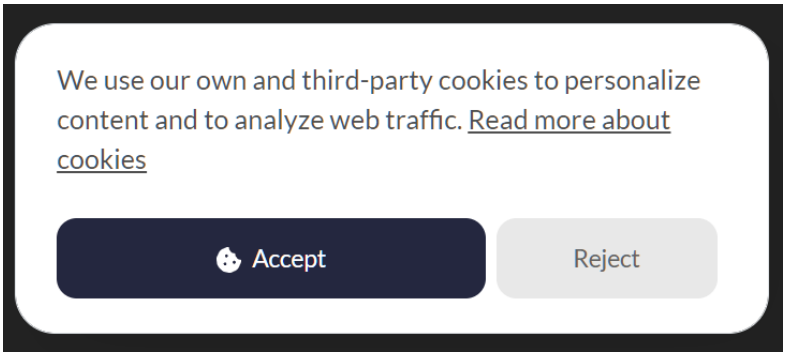
Cookies are small text files that are stored on a user's computer by a web server when a user visits a website. Cookies are used to store information about the user's preferences and actions on the website, such as login information, shopping cart contents, and language preferences.

When a user visits a website, the web server sends a cookie to the user's browser, which then stores the cookie on the user's computer. The cookie contains a unique identifier that allows the web server to identify the user and retrieve any stored information associated with the user.

For example, suppose you visit a shopping website and add some items to your shopping cart. The website may use cookies to store the items you added to your cart so that when you return to the website later, your shopping cart still contains the items you added.

Cookies can also be used to track a user's browsing behavior on a website. For example, a website may use cookies to remember which pages a user has visited or which products a user has viewed, in order to provide personalized recommendations or advertising.

Cookies can be either session cookies or persistent cookies. Session cookies are temporary cookies that are deleted when the user closes their browser, while persistent cookies are stored on the user's computer for a longer period of time.



You might have seen this kind of dialogue box on multiple websites which asks for permission regarding cookies.

Data stored in cookies

Here are some examples of the types of data that cookies can store:

1. User preferences: Cookies can store information about your preferences on a website, such as your language preference or font size preference. This allows the website to provide a personalized experience for you.
2. Login information: When you log in to a website, the website can use cookies to remember that you are logged in. This allows you to stay logged in even if you leave the website and come back later.
3. Tracking information: Websites can use cookies to track your browsing behavior and show you personalized content or advertisements based on your interests.
4. Session information: When you interact with a website, the website can use cookies to keep track of your session. This allows you to navigate between pages on the website without having to log in again every time.

Here's an example of a cookie that stores a user's language preference:

Name: lang
Value: en-US
Expiration: Fri, 31 Dec 9999 23:59:59 GMT

In this example, the cookie is named "lang" and has a value of "en-US", which represents the user's language preference. The cookie is set to expire on December 31, 9999, which means it will be stored on the user's computer indefinitely unless they clear their cookies.

Brief history of cookies

Cookies were first introduced in 1994 by Netscape Communications, which was one of the first web browsers. The idea behind cookies was to provide a way for websites to remember user preferences and store information about users.

The original cookies were very basic and could only store a small amount of data. They were also not very secure, as

anyone could view or modify the cookies stored on their computer.

Over time, cookies have evolved to become more sophisticated and secure. Modern cookies use encryption and other security measures to protect user data. Additionally, browsers now have settings that allow users to control which cookies are stored on their computer and how long they are stored for.

Working of cookies

This is basically what is happening with cookies :

1. When a user visits a cookie-enabled website for the first time, the browser will prompt the user that the web page uses cookies and request the user to accept cookies to be saved on their computer.
2. After the user accepts the prompt, the server responds by sending back a cookie (among many other things).
3. This cookie is going to be stored in the user's browser.
4. When a user visits the website or sends another request, that request will be sent back together with the cookies.
5. The cookie will have certain information about the user that the server can use to make decisions on any future requests.

Here's an example of how cookies work:

Let's say you visit a website that has a login page. When you enter your username and password and click the "LogIn" button, the website sends that information to the server. If the server validates your credentials, it sends a cookie back to your browser that identifies you as a logged-in user. Your browser stores the cookie on your computer.

The next time you visit the website, your browser sends the cookie back to the server with your request. The server can then use the cookie to remember that you are a logged-in user, and provide personalized content based on your preferences.

Types of Cookies

There are several types of cookies that websites can use to store information about you. Here are some of the most common types:

1. Session Cookies: These cookies are temporary and are only stored in your browser's memory for the duration of your browsing session. Once you close your browser, the cookies are deleted. Session cookies are commonly used to remember items in your shopping cart on an e-commerce website.
2. Persistent Cookies: These cookies are stored on your computer even after you close your browser. They have an expiration date, after which they will be deleted. Persistent cookies are commonly used to remember your login information or language preference on a website.
3. Secure Cookies: These cookies are encrypted to prevent unauthorized access. They are commonly used to store sensitive information such as login credentials or financial information.
4. HttpOnly Cookies: These cookies can only be accessed through HTTP(S) requests, and not through client-side scripts such as JavaScript. This makes them more secure against cross-site scripting attacks.
5. Third-party Cookies: These cookies are created by a domain other than the one you are visiting. For example, if you visit a website that has embedded content from a different website, that website may create a third-party cookie. Third-party cookies are commonly used for advertising and tracking purposes.

Cookies with Node.js

First open a NodeJS Project in VS Code, install the ExpressJS package in it.

Note that cookies are sent to the browser from the backend server, we use a package called cookie-parser to create cookies and send them to the client browser.

Install the package by command `npm i cookie-parser` .

Create a basic ExpressJS server like we have discussed earlier.

Then, we define a route for the home page ('/') and set a cookie named 'myCookie' with a value of 'Cookie Value' using the res.cookie() method. Finally, we send a response with the text 'Hello, world, cookie has been saved!'.
ex :

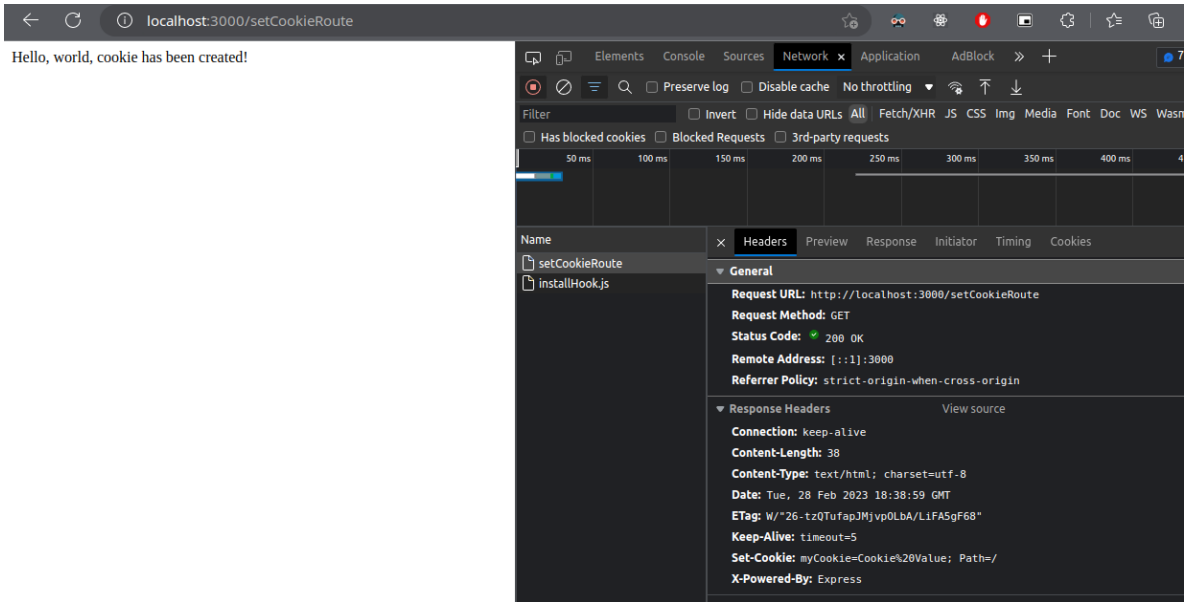
```
app.get("/setCookieRoute", (req, res) => {
  // Set a cookie named 'myCookie' with value 'Hello, world!'
  res.cookie('myCookie', 'Cookie Value');
  res.send('Hello, world, cookie has been saved!');
})
```

At this point our index.js file looks like :

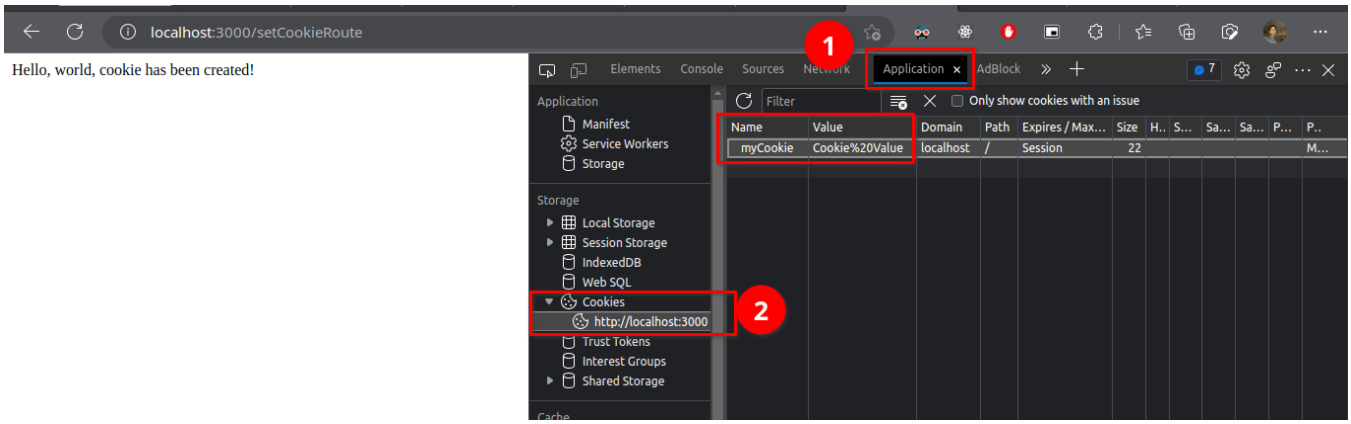
```
JS index.js > ...
1  const express = require("express");
2  const app = express();
3  const port = 3000;
4
5  const cookieParser = require("cookie-parser")
6
   0 references
7  app.get("/", (req, res) => {
8     res.send("Welcome to backend");
9  });
10
   0 references
11 app.get("/setCookieRoute", (req, res) => {
12    // Set a cookie named 'myCookie' with the value 'Cookie Value'
13    res.cookie('myCookie', 'Cookie Value');
14    res.send('Hello, world, cookie has been created!');
15 })
16
   0 references
17 app.listen(port, () => {
18    console.log(`Server is listening on port ${port}`);
19 });
20
21
```

Now start the server by node index.js open the site in browser and open the Network tab in Developer Tools.

then hit the endpoint /setCookieRoute , We can see



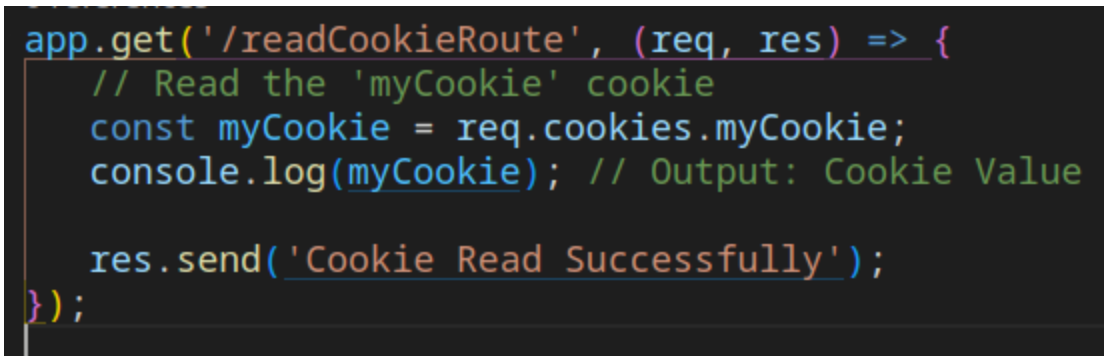
On opening Application tab then expanding the Cookie option present in the left bar then clicking on the localhost we can see the cookie that has been created :



Reading Cookies

To read cookies in Node.js using the cookie-parser package, you can access the cookies property of the request object (req.cookies). Here's an example:

Here we use the cookieParser() middleware to parse cookies in incoming requests. Then, we define a route for the home page ('/') and read the value of a cookie named 'myCookie' using req.cookies.myCookie. We then log the value of the cookie to the console (Cookie Value). Finally, we send a response with some text'. Like :



Deleting Cookie

For Deleting a cookie we also need to set a different route and call the functions like :

```
app.get('/deleteCookieRoute', (req, res) => {
   res.clearCookie();
   res.send('Cookie has been deleted successfully');
});
```

Authentication

Authentication is the process of verifying the identity of a user or system. In web development, authentication is commonly used to restrict access to certain pages or functionality to only authorized users.
In technical terms, authentication usually involves a combination of a username and password (or other form of credential), and a session or token that is generated by the server to identify the user. The session or token is usually stored in a cookie or in local storage on the client's device, and is used to maintain the user's authenticated state across multiple requests.
Let's say you have a web application that allows users to post messages on a public forum. You want to make sure that only registered users can post messages, so you implement authentication. When a user wants to post a message, they are first prompted to log in with their username and password. The application then verifies that the username and password match a record in the database, and if they do, the user is considered authenticated and allowed to post a message. If the username and password do not match, the user is denied access and prompted to try again.

Do note that Authentication is different from Authorization. Authorization is the process of determining what actions a user is allowed to perform in a web application. In other words, once a user has been authenticated (i.e. their identity has been verified), authorization is used to control what they are allowed to do within the application.