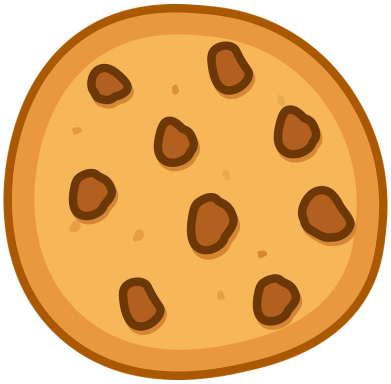
Cookies



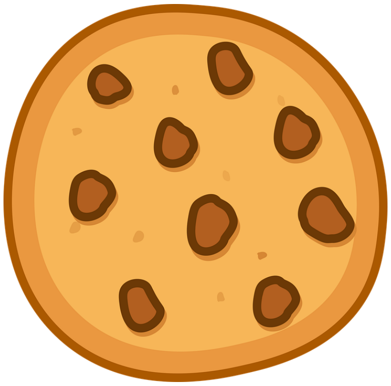
Introduction🍪

- Cookies are small pieces of data that are sent with **each** request.
- Things like “keep me logged in” work because of cookies.
- Also the infamous ad tracking works because of cookies. You search for black shoes on google, and you see its ad everywhere you go. Yup cookies.

Ways to create cookies🍪

- Cookies are stored in the browser.
- Two ways to create cookies:
  - Through JavaScript
  - Through Web Server
- Through JavaScript: On any document, run the code, `document.cookie=”animal=bear”` , it will set the cookie, with a key(animal) and a value(bear).
- Through Web Server: Web server can set the cookie in its response using the `Set-Cookie` **header**. Example: `Set-Cookie: sessionId=38afes7a8` . This will tell the browser to

set cookies with mentioned key and value.



Creating Cooking🍪

Using JavaScript

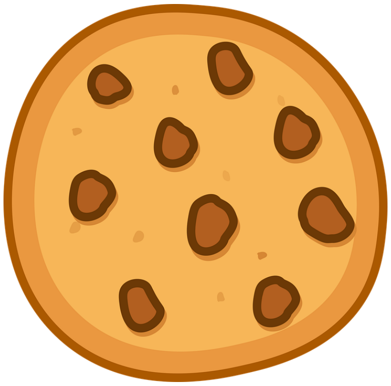
- F12, go to console, and create a cookie, `document.cookie=”animal=bear”`
- The `document.cookie` if used on same key then it overrides the value. If key doesn’t exist, a new cookie is formed.
- If you `console.log(document.cookie)`it will show all the cookies present in the current scope.
- To check the cookies created, press F12, then go to application and check cookies created(for Chrome).

Using Server

- In Express we can set the set-cookie header before sending the response, by providing it an array of cookies to set

```
app.get("/",(req,res)=>{
    res.setHeader("set-cookie",["animal=elephant"]);
    res.sendFile(`${_dirname}/index.html`);
})
```

- We can go to Network tab of Chrome and check the headers received, we will find the cookie `"animal=elephant"` sent by the server.



Cookie Properties 🍪

- Cookies are sent with almost every request.
- Hence having a lot of cookies consume bandwidth.

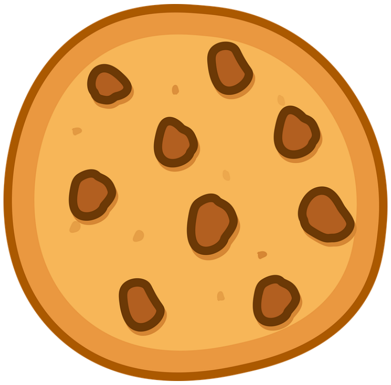
Cookie Scope🍪

- Now cookies have scope just like variables, a cookie set on google.com won’t be sent to the servers of amazon.com when you visit amazon.com.
- There are two kinds of cookie scopes, first one is **domain scope** and second is **path scope**.
- Domain scoped cookies are sent with the request to same domain.
   For example, you have cookies in your browser related to google.com and when you visit google.com/about , the already stored cookies related to google.com will be sent to the server of google.
- Path scoped cookies are cookies that are sent with the request related to a particular path.

NOTE: www.google.com and google.com are two different sub-domains, similarly mail.google.com

- To make our cookie available on all subdomains we can add an extra property to our cookie. This property is **domain**. Example: `document.cookie=”animal=wolf; domain=.google.com”`
- A semicolon is used to separate cookie properties.
- .google.com means make the cookie available for all subdomains.

- The cookie `animal=wolf` will now be available to all the subdomains of google.com now.
- Now whenever we visit google.com our two cookies `animal=wolf` and `"animal=elephant"` will be sent to google servers. We can check this by checking the cookie header in request headers.
- Each domain will have its won set of cookies. Google will have its own , amazon will have its own.
- We can scope cookies according to path using the path property. `document.cookie=”animal=wolf; domain=.google.com; path=/path1”` or `document.cookie=”animal=rabbit; domain=.google.com; path=/path2”`
- Now the `animal=wolf` cookie will be available when we visit google.com/path1
- Similarly, `animal=rabbit` cookie will be available when we visit google.com/path2
- Path scope can be used to configure cookies, to be sent only on few particular path. This helps us save bandwidth. Since one cookie can be of 4kb and many website make us store 100s of them.



Cookie Expiry and Max Age🍪

- Each cookie can have a property(expires or max-age) that tells the browser when to destroy the cookie. If this property is not specified while preparing the cookie. The browser will destroy such cookies when it(browser) is closed. Such cookies are called **session cookies**.
- Example of session cookie `document.cookie=”animal=rabbit”`

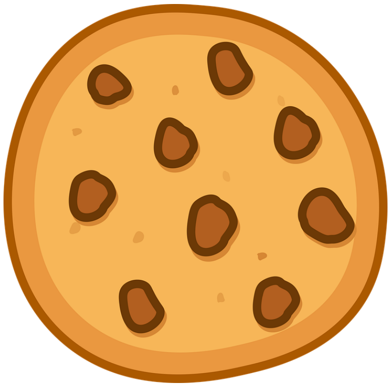
- If expires or max-age is set on a cookie, it is called **permanent cookie**. But if you think, it is not permanent, since it will be destroyed after specific time. These cookies can live even if we close the browser.
- Example of permanent cookie `document.cookie=”animal=rabbit max-age=3min”`
- The cookie created above will be destroyed after 3 minutes
- In case of expires, we put UTC date string, instead of minutes.

Same-Site🍪

- Same-Site is the newest cookie attribute.
- Suppose you have logged-in to your back account, let’s say bankofamerica.com.
- So when you logged-in, the servers of bankofamerica.com sent you cookies to your browser to maintain the login.
- This saved cookie can be used to login to bankofamerica.com without entering username and password again.
- Now suppose you visit piratbay.com , a shady site. And it has a link.
- `https://bankofamerica.com/transfermoney/toaccount=123456&&money=1000000`
- What this link does is, it transfers 1 million dollar to the bank account 123456. Since you are logged in, when you click the link, your cookies will be sent to bankofamerica.com, and this link will be executed. And money will be transferred to that person’s account.
- In such cases we don’t want our cookies to be sent to bankofamerica.com when we are visiting it by clicking some link on some other website.
- If we were making a request using fetch or XHR, CORS would have stopped us. But in this case we are actually visiting the site.
- The same-site attributes helps us with this problem.
- Any cookie that has the property same-site=strict, will only be sent to its domain, when we are directly visiting that website directly and not through some link.
- Example : `document.cookie=”animal=antelope; same-site=strict”`

same-site=lax🍪

- But `same-site=strict` is not good for few cases. For example if you visit mozilla.com and set the theme to dark-mode, now when you visit some other website say wikipedia.com. wikipedia.com has a link to mozilla.com, now if the same-site=strict is applied, no cookie will be sent to mozilla, and hence you can’t see the website in dark-mode.
- In such cases , we use `same-site=lax`, It is more relaxed version of strict. In this, cookies are sent to the third party website, if you visit the link. But it won’t be sent if you include an image or video from a third party website.



Cookie-Types🍪

- **Session Cookie** : A cookie that doesn’t have max-age or expiry property. These cookies are deleted when the browser is closed.
- **Permanent Cookie**: A cookie that has max-age and expiry set. This cookie can live even if the browser has been closed.
- **HttpOnly Cookie**: A cookie that can only be set by the server. The browser can’t read this cookie. If you do console.log(document.cookie) you won’t be able to see these cookies.

```
app.get("/",(req,res)=>{
    res.setHeader("set-cookie",["animal=elephant","jscantseethis=1;httponly"]);
    res.sendFile(`${_dirname}/index.html`);
})
```

- **Secure Cookie**: These cookies are sent only when the connection is HTTPS.

- **Third Party Cookies** : Suppose you are visiting a blog, this blog has some google ads, when the browser makes request for these ads, google sends the ad along with cookies. These cookies that were sent are called third party cookies. They are generally used for tracking.
- **Zombie Cookies**: These cookies regenerate even if you delete them. Because the server can identify you through some other methods, like your IP address , e-tag etc. Once identified, it can send those cookies again. And these cookies get regenerated.

Cookie Security🍪

- **Stealing cookies** : This is done through JavaScript, the app simply uses `document.cookie` to read all the cookies and send them to some other server.
- **Cross Site Request Forgery**: The example we discussed with bankofamerica.com. under same-site attribute.

References

- https://www.youtube.com/watch?v=sovAIX4doOE&t=2619s(a video by Hussain Nasser)
- https://www.youtube.com/watch?v=m4vatwFryI8(a video by Hussain Nasser)