Cookie Basics

Main Point Preview

- A cookie is a key value pair that the server gives to a client with a time out. Each request after that (if the timeout hasn't expired) the client will give that key value pair back to the server.

- Science of Consciousness: Every action has an equal and opposite reaction. The server gives a cookie, and the client gives it back.

Stateful client / server interaction

- HTTP is a stateless protocol
  - Nothing in the request tells the server who you are
  - HTTP itself cannot keep state for you / help make a custom response

- Nevertheless, all kinds of websites seem to 'remember' you
  - This is done through cookies (data the client stores for / sends back to a site)
  - Cookies can be used to create sessions (stateful interactions)

What is a cookie?

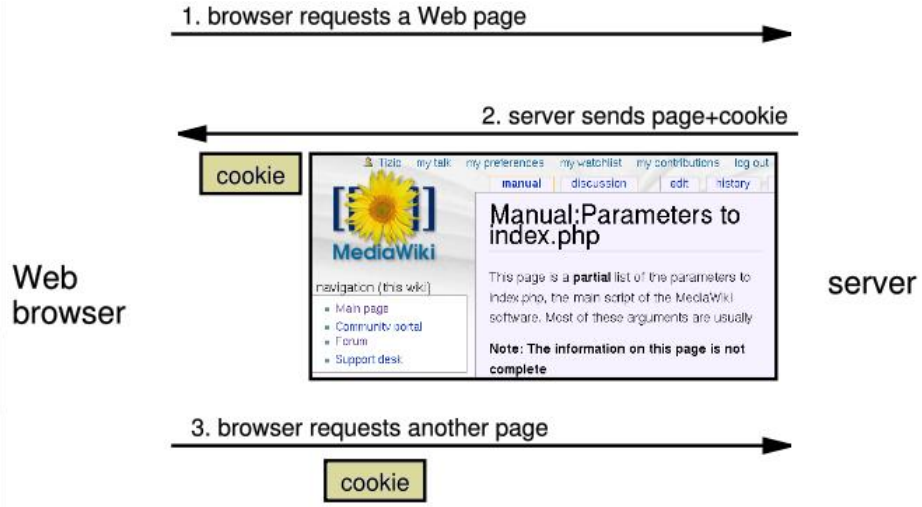
- A HTTP Cookie is a small amount of information sent by a server to a client (browser), and then sent back on each future request to that server

- Cookies have many uses:
  - Authentication
  - Maintaining user preferences, shopping carts, etc
  - User tracking

- A cookie's data consists of a single name/value pair
  - It gets sent in the HTTP header for GET or POST requests

How cookies are sent

- When a browser requests a page, the server may send back one or more cookies in the response header
- If your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests (in the request header)



- It is also possible for client-side JavaScript to set/get cookies

Myths about cookies

- Myths:
  - Cookies are like worms / virusus and can erase data from the hard disk
  - Cookies are a form of spyware and can steal your personal information
  - Cookies generate popups and spam
  - Cookies are only used for adverstising
- Facts:
  - Cookies are only data, not executable code (can't do anything)
  - Cookes 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

Tracking Cookies

- An advertising company can put a cookie on your machine when you visit one site, and then see that same cookie when you visit another site that uses the same advertising company
  - Therefore they can tell that the same person (you) visited both sites

- These are 3rd party cookies
  - Browsers can be told not to accept them
  - Advertisers find new and creative way to fingerprint / track you

Where are cookies stored

- They used to be stored as plain text files in a directory
  - Modern browsers usually store them in a SQLite database
- Chrome:
  - C:\Users*username*\AppData\Local\Google\Chrome\User Data\Default\Cookies
- Firefox:
  - C:\Users\username\AppData\Roaming\Mozilla\Firefox\Profiles\???.default\cookies.sqlite
- You can install browser extensions to easily view and manipulate them

How long does a cookie exist

- Session cookie:
  - This is the default type, and is only stored in the browser's memory
  - When the browser is closed, these cookies are erased
  - Cannot be used for tracking long-term information
  - Safter, because no programs other than the browser can access them

- Persistent cookie:
  - A cookie that is stored in the browser computer's filesystem
  - Is created by giving an expiration date/time
  - Can track long-term information
  - Potentially less secure, because users (or programs they run) can open cookie files, and see/change the cookie values

Main Point

- A cookie is a key value pair that the server gives to a client with a time out. Each request after that (if the timeout has not expired) the client will give that key value pair back to the server.

- Science of Consciousness: Every action has an equal and opposite reaction. The server gives a cookie, and the client gives it back.

Using Cookies

Main Point Preview

- When creating a response on the server we can set cookies. When processing a request on the server, we can look for cookies that the client sent back. The server can make it a 'permanent' cookie by giving a timeout, and optionally sign them with a hashcode to prevent tampering.

- Science of Consciousness: As we grow in consciousness we start to see and understand finer mechanics of natural law, understanding what was in the past and what will probably be in the future.

Middleware: cookie-parser

- The cookie-parser middleware will read the cookie headers in HTTP requests
  - And populate the req.cookies and req.signedCookies objects
  - Without this middleware you would have to parse them yourself

- You can install it into your project with:
  - $ npm install cookie-parser -save

Setting and Retrieving a cookie

JSFiddle

```
1  const express = require('express');
2  const cookieParser = require('cookie-parser');
3  const app = express();
4
5  // parses request cookies, populating req.cookies and req.signedCookies
6  app.use(cookieParser());
7  // parse x-www-form-urlencoded
8  app.use(express.urlencoded({extended: false}));
9
10 app.get('/', (req, res) => {
11   if (req.cookies.remember) {
12     res.send('Remembered :). Click to <a href=\'/forget\'>forget</a>');
13   } else {
14     res.send(`<form method="post">
15                 <label>
16                   <input type="checkbox" name="remember" />
17                 Remember me
18               </label>
19             <input type="submit" />
20             </form>`);
21   }
22 });
23 app.post('/', (req, res) => {
24   if (req.body.remember) {
25     res.cookie('remember', 1)
26   }
27   res.redirect('back');
28 });
29 app.get('/forget', (req, res) => {
30   res.clearCookie('remember');
31   res.redirect('back');
32 });
33 app.listen(3000);
```

Based on example from: https://github.com/expressjs/express/blob/master/examples/cookies/index.js

Checks if cookie is set (doesn't care about value)

Sets a cookie with the name 'remember' value 1

Removes the cookie with name 'remember'

Setting and Removing a persistent cookie

JSFiddle

```
const express = require('express');
const cookieParser = require('cookie-parser');
const app = express();

// parses request cookies, populating req.cookies and req.signedCookies
app.use(cookieParser());
// parse x-www-form-urlencoded
app.use(express.urlencoded({extended: false}));

app.get('/', (req, res) => {
  if (req.cookies.remember) {
    res.send('Remembered :). Click to <a href=\'/forget\'>forget</a>');
  } else {
    res.send(`<form method="post">
                <label>
                  <input type="checkbox" name="remember" />
                  Remember me
                </label>
                <input type="submit" />
              </form>`);
  }
});
app.post('/', (req, res) => {
  if (req.body.remember) {
    // maxAge is specified in milliseconds (a week)
    res.cookie('remember', 1, {maxAge: 1000*60*60*24*7})
  }
  res.redirect('back');
});
app.get('/forget', (req, res) => {
  res.clearCookie('remember');
  res.redirect('back');
});
app.listen(3000);
```

Providing a maxAge makes a cookie persistent

Signed cookies

JSFiddle

```
1  const express = require('express');
2  const cookieParser = require('cookie-parser');
3  const app = express();
4
5  // parses request cookies, populating req.cookies and req.signedCookies
6  app.use(cookieParser("salt for cookie signing"));
7  // parse x-www-form-urlencoded
8  app.use(express.urlencoded({extended: false}));
9
10 app.get('/', (req, res) => {
11   if (req.signedCookies.remember) {
12     res.send('Remembered :). Click to <a href=\'/forget\'>forget</a>');
13   } else {
14     res.send(`<form method="post">
15                 <label>
16                   <input type="checkbox" name="remember" />
17               Remember me
18             </label>
19           <input type="submit" />
20           </form>`);
21   }
22 });
23 app.post('/', (req, res) => {
24   if (req.body.remember) {
25     // maxAge is specified in milliseconds (a week)
26     res.cookie('remember', 1, {maxAge: 1000*60*60*24*7, signed: true })
27   }
28   res.redirect('back');
29 });
30 app.get('/forget', (req, res) => {
31   res.clearCookie('remember');
32   res.redirect('back');
33 });
34 app.listen(3000);
```

Needs a secret to use when signing / hashing cookie values

SignedCookies are stored in a different object! You cannot find them in req.cookies

To make a signed cookie simply specify signed: true

Main Point

- When creating a response on the server we can set cookies. When processing a request on the server we can look for cookies that the client sent back. The server can make it a 'permanent' cookie by giving a timeout, and optionally sign them with a hashcode to prevent tampering.

- Science of Consciousness: As we grow in consciousness we start to see and understand finer mechanics of natural law, understanding what was in the past and what will probably be in the future.

Sessions

Main Point Preview

- Sessions use a cookie to give each client a unique identifier (session id), this identifier is then used on the server to find key/value storage pairs that we can access on req.session

- *Science of Consciousness: Knowledge is gained from inside and outside*

What is a Session?

- A session is an abstract concept to represent a series of (HTTP) requests between a specific web browsers and server
  - HTTP doesn't support the notion of sessions, but web development frameworks add this support through session cookies or hidden form fields

- Sessions VS. Cookies:
  - A cookie is data stored on the client
  - A session's data is stored on the server (one session per client)

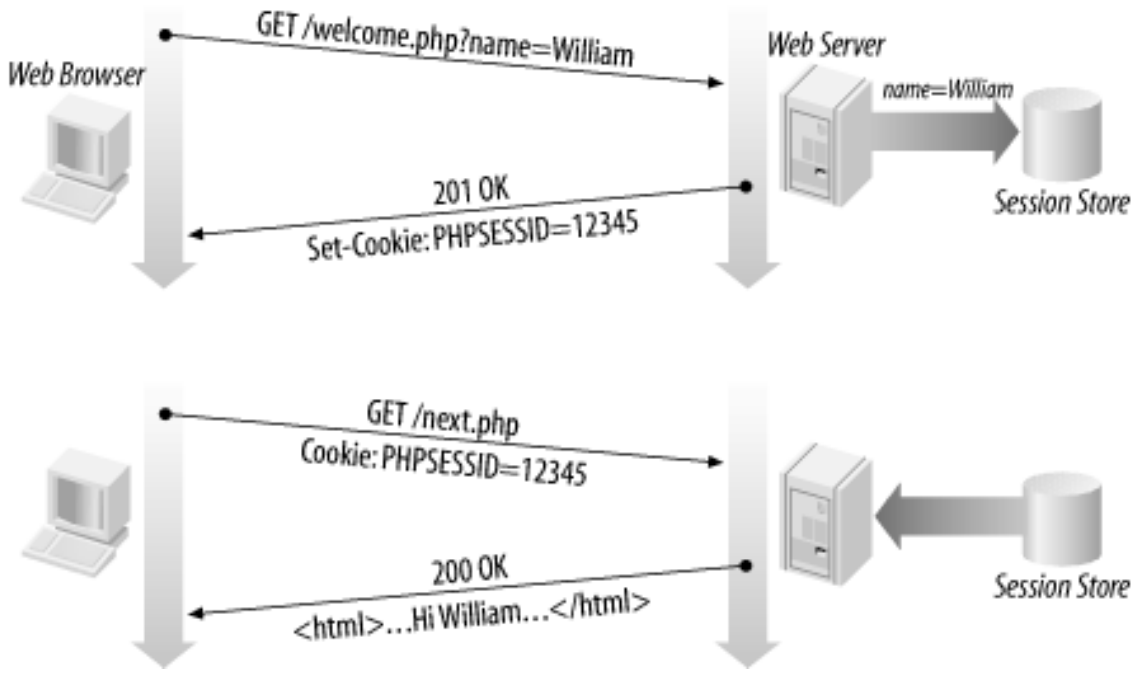
- Sessions are often built on top of cookies:
  - The only data the client stores is a cookie holding a unique sessions ID
  - On each page request, the client sends the session ID, and the server uses this to find and retrieve the client's session data

Characteristics of Sessions

- Information or state must often be stored. For example:
  - A selected bottle of milk in a shopping cart
  - A customer's name and credit card number before checkout
- A Session Cookie stores the Session ID
  - The server then uses that Session ID to find the correct storage for this user
- Sessions need to have a timeout
  - Otherwise if a user stops making requests (leaves) there is no way for the server to know when the session should end / when it can remove the storage

How sessions are established

- 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 the client
- Client sends that same session ID back to the 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



Middleware: express-session

- No need for parse-cookie middleware
  - In fact, it's dangerous to use it at the same time as express-session
  - If you have different secrets for signing cookies they can interfere

- You can install express-session into your project with:
  - $ npm install express-session -save

Getting and Setting session data

JSFiddle

```
const express = require('express');
const parseurl = require('parseurl')
const session = require('express-session')
const app = express();

app.use(session({
  secret: 'salt for cookie signing',
}));

app.listen(3000, () => {
    console.log('Your server is running on 3000');
});

app.use(function (req, res, next) {
  if (!req.session.views) {
    req.session.views = {}; // put views object into session
  }

  // get the url pathname
  var pathname = parseurl(req).pathname;

  // count the views for the given url
  req.session.views[pathname] = (req.session.views[pathname] || 0) + 1;

  next();
});

app.get('/foo', function (req, res, next) {
  res.send('you viewed this page ' + req.session.views['/foo'] + ' times');
});

app.get('/bar', function (req, res, next) {
  res.send('you viewed this page ' + req.session.views['/bar'] + ' times');
});

app.get('/', (req,res) => {
    res.send("<a href='foo'>foo</a> <a href='bar'>bar</a>");
});
```

Based on the example at: https://www.npmjs.com/package/express-session#example

Session Storage

- If you don't specify storage the default store is MemoryStore.
  - This is purposely not designed for a production environment
  - It will leak memory under most conditions, does not scale
  - Meant only for debugging and developing

- For production you'll need something else
  - There is a big list of possible in memory or on disk data stores at:
  - https://www.npmjs.com/package/express-session#compatible-session-stores

Main Point

- Sessions use a cookie to give each client a unique identifier (session id), this identifier is then used on the server to find key/value storage pairs that we can access on req.session

- *Science of Consciousness: Knowledge is gained from inside and outside*