# COMP4021
Internet Computing
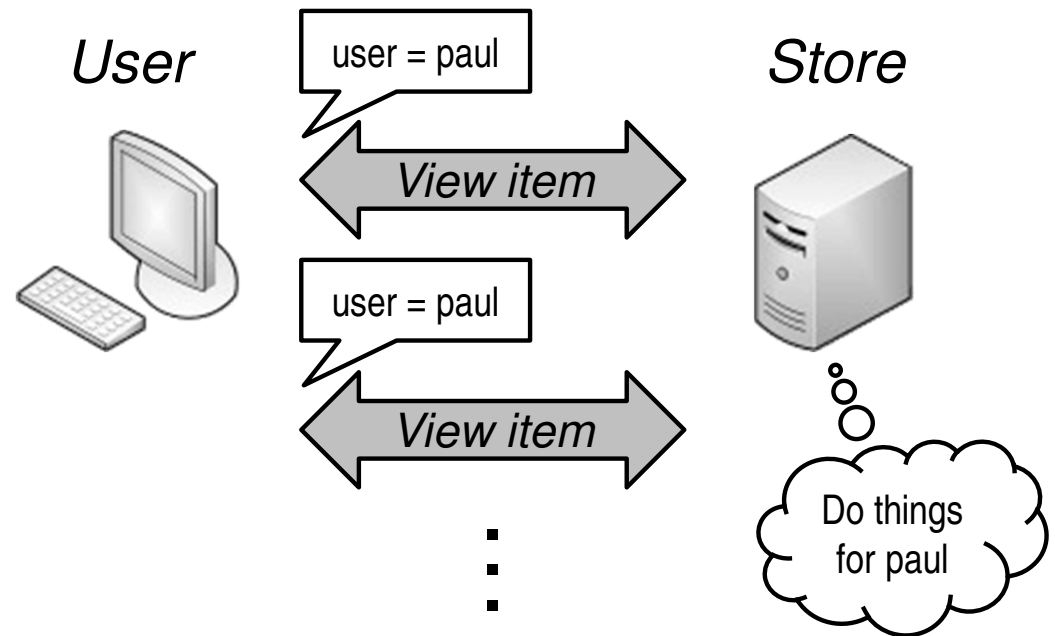
Web Sessions

Gibson Lam

Using a Web Application

- A user (i.e. a browser) may use a web application for a certain amount of time
- During this time, the user may send many HTTP requests to the server
  - For example, when buying things from an online store, a user may need to look at many items before deciding to check out

User ← View item → Store

← View item →

⋮

← Checkout →
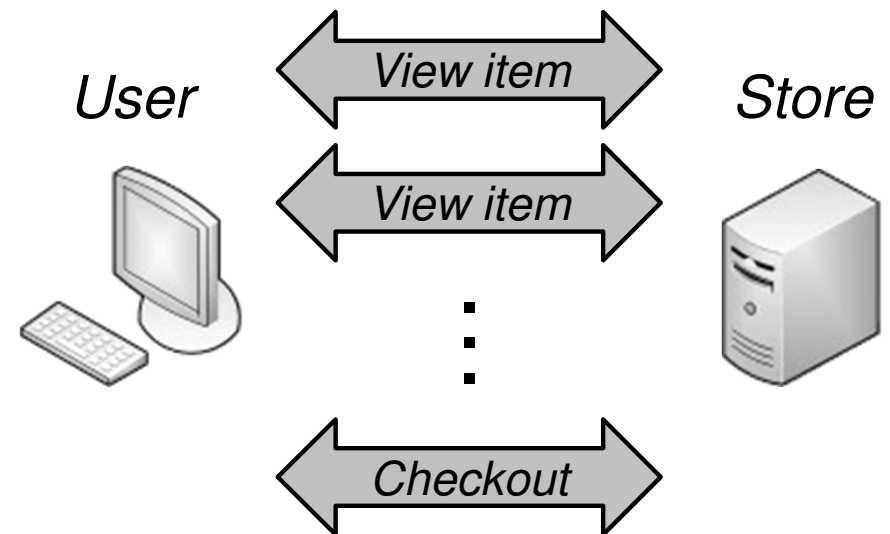
Repeating Data in Requests

- The same user may send a lot of repetitive data in those HTTP requests

  – Using the online store example again, the user may need to send his/her account name to the server every time

User

user = paul

View item

user = paul

*View item*

Store

Do things for paul

Web Sessions

- In the previous example, as the requests are from the same user, it would make more sense to send the account name only once, rather than for every request

- If the server knows which requests are from the same browser, it will only require the browser sending things once

- You can use a *web session* to do that

User

Store

user = paul

View item

View item

View item

Do things for paul
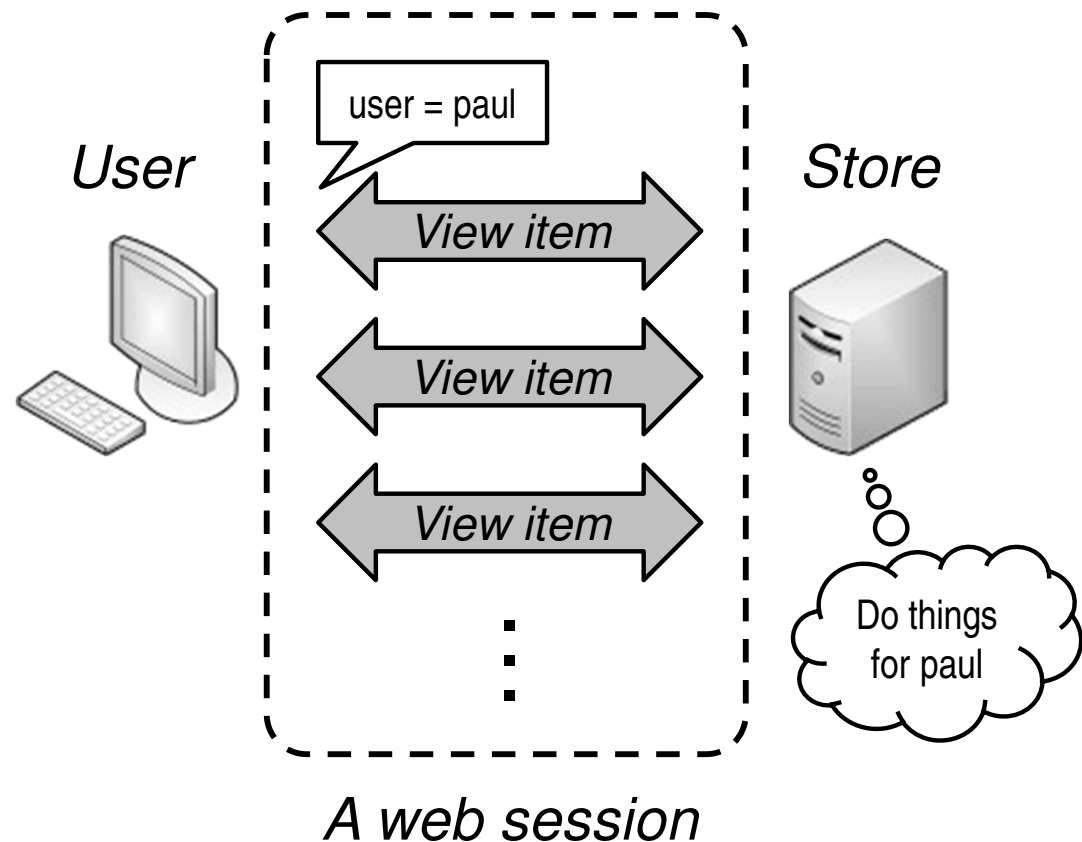
A web session

Using Server-Side Cookies

- You can create a web session in a simple way: using server-side cookies

- First, when the browser visits a web application the first time, the application sends a 'session' cookie to the browser
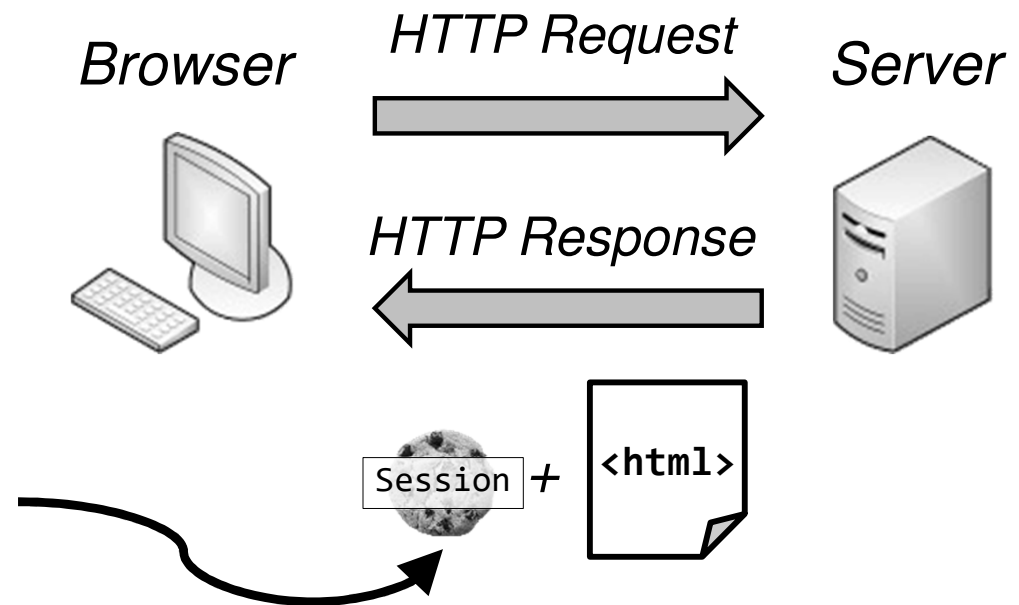
Subsequent Requests

- When the browser receives the response, it stores the cookie locally

Browser

- Afterwards, when the browser sends other requests to the server, the same cookie is sent to the server too

- The server then identifies the browsers based on the cookie content

Browser   Session   *Server*

HTTP Request
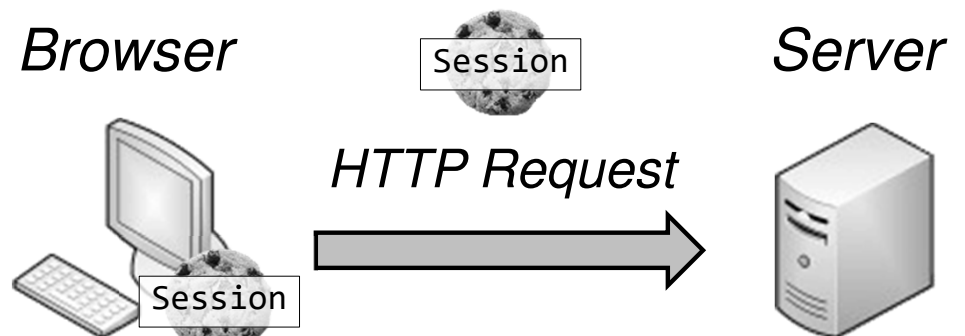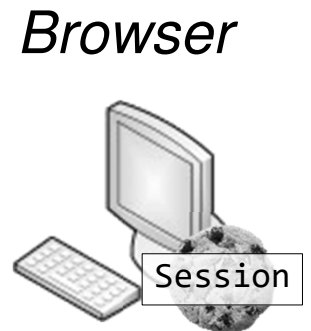
The Session Id

- A session cookie is just an ordinary cookie
- The value of a session cookie, called the *session id* is typically some random strings for identifying different browser sessions
- For example, here is the value of a session cookie from an Express server:

```
s:rL8FypesDigtTT
pgLnc039gaUffvqz
-U.9F9Rvcw4vUj1/
GJBf2X/N7DkVoF3K
L4lZMjWA9OLOgk
```

The session id is too long so it is broken into 5 lines here!

Using the Session Id

- The session id is used by the server for setting up the session and identification purpose

- You usually do not need to know the exact content of it

Browser A   *Browser B*

`Session`   `Session`

Session Id: ABC   *Session Id: XYZ*
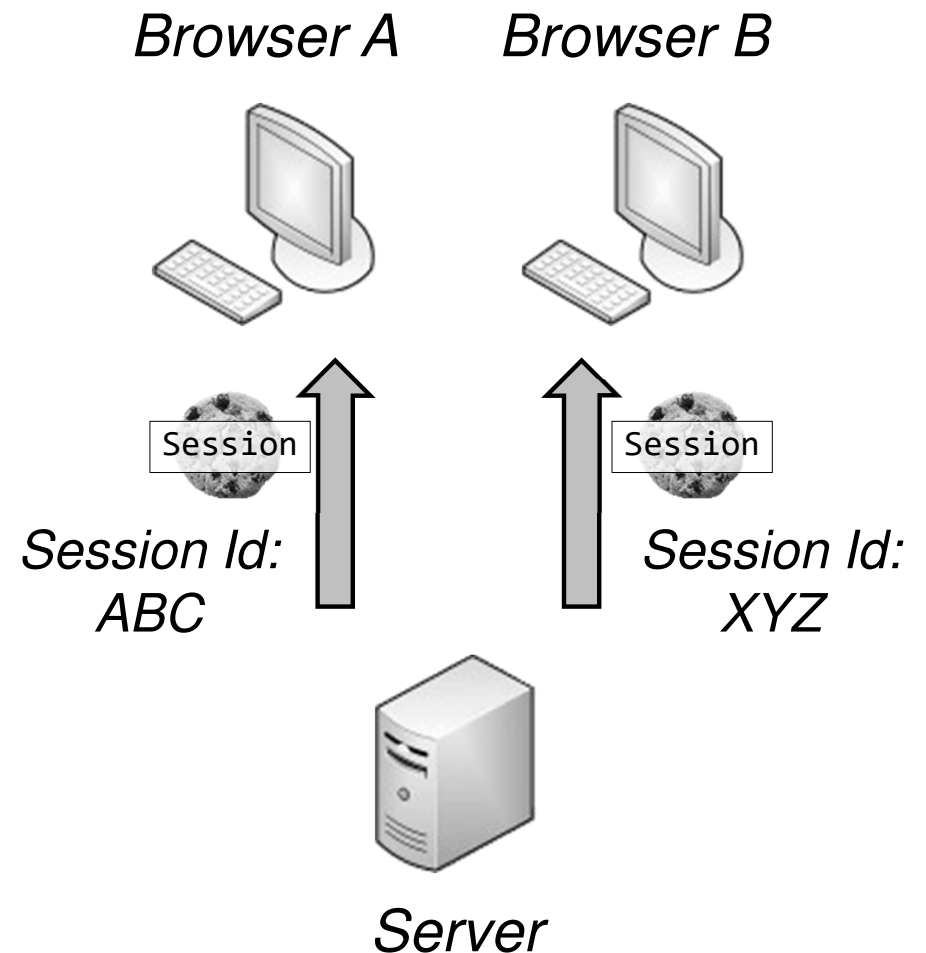
Server

Session Data

- After establishing the session, the server can store additional data for individual sessions

- This session data is only stored on the server, e.g. in memory or in a database

- The browser does not have any access to it

Browser

Interactions

Server

Web application
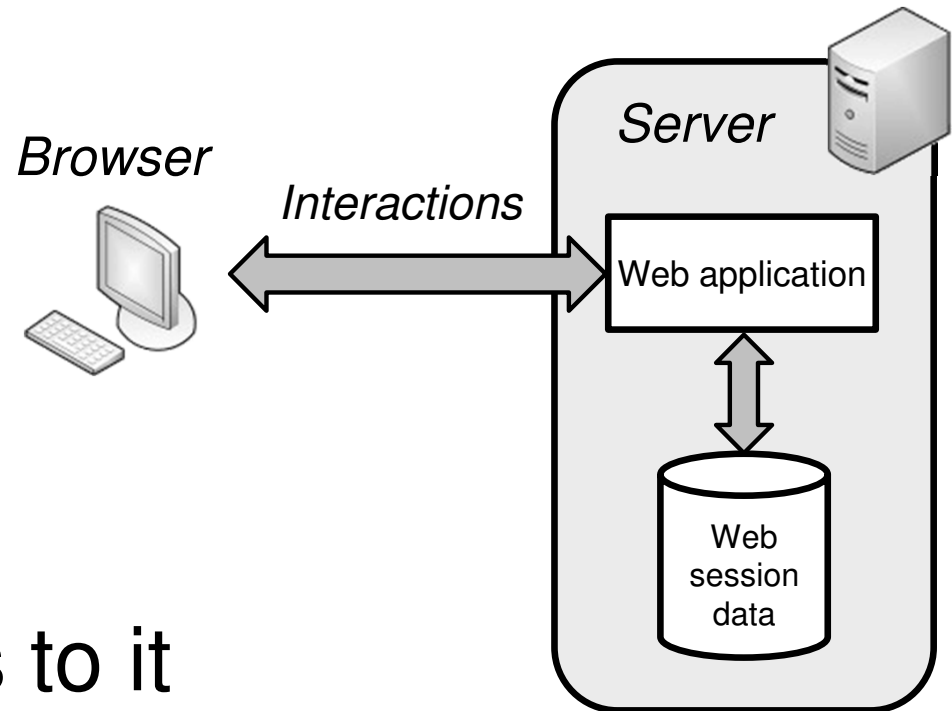
Web session data

Duration of a Session

- Since a web session is associated with its session cookie, the duration of a session is then equivalent to the lifetime of the cookie

- If the cookie has no expiry time, the session will be gone when the browser is closed

- A session cookie is usually set with a specific lifetime, e.g. 5 minutes, so that its session can exist for some time even after the browser is closed

Using Session in Express

- You need to install an additional package if you want to use web session in Express, i.e.:

```
C:\Users\Gibson>npm install express-session
```

- You can then use a session 'middleware' by doing this:

```
const session =
        require("express-session");
app.use(session());
```

Configuring the Session

- When creating an Express session, additional settings can be used

- A session in Express is typically set up with more options like this:

```
const mySession = session({
    secret: "secretkey",
    resave: false,
    saveUninitialized: false,
    rolling: true,
    cookie: { maxAge: 300000 }
});
app.use(mySession);
```

Lifetime of the session cookie
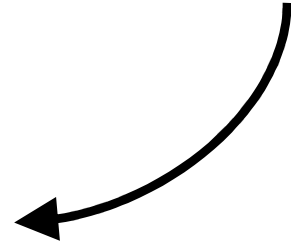
The Session Object

- In the server program, you can get information about the session using `req.session`

- For example, you can read the session id using:

```
console.log(req.session.id);
```

- In addition, any property added to the `req.session` object becomes your session data that can be accessed across requests

Reading/Writing Session Data

- To write session data, you just add a property to the session object, e.g.:

```
req.session.name = "Paul";
```

- Later, in another request, you can read the data back by, e.g.:

```
if (req.session.name)
    name = req.session.name;
```

"Paul"

Delete the Session

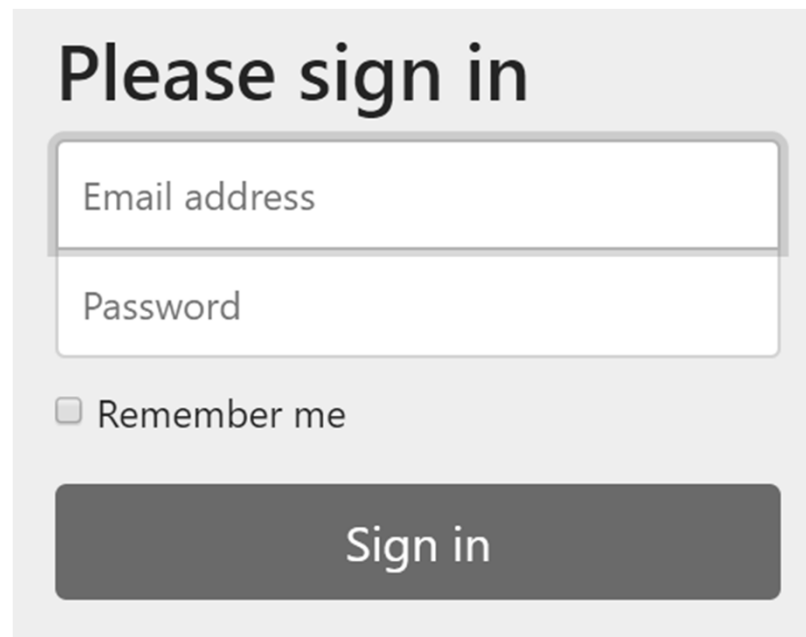- You can delete a session using the session object, e.g.:

```
req.session.destroy();
```

- Alternatively, it may be quicker to indicate the end of a session by removing the data that you have stored before, e.g.:

```
delete req.session.name;
```

An Example Use of Sessions

- One of the most common use of sessions is for handling user authentication

- Here is an example sign in form:

Please sign in

Email address

Password

☐ Remember me

Sign in

A Sign-In Scenario

1. At the start of the web application, a sign-in form is shown to the user

2. The user enters the username and password, and the information is sent to the server

3. Upon receiving the information, the server checks whether the user can be authenticated or not

4. The authentication result is then stored in a session variable

5. Finally the authentication result is returned to the browser, so that it can react appropriately

Page Validation and Signing Off

- After signing in:
  - Each request uses the session variable to validate the authentication and protect against unauthorized access
  - The sign-in form uses the session variable to make sure no double sign in is required
- To sign off, the session is then destroyed and the browser shows the sign-in form again