## Cookies and Sessions

- Recall that HTTP is stateless: it simply allows a browser to request a single document from a web server and doesn't remember or keep track of anything between invocations (short-lived).
- Every resource that is accessed via HTTP is a single, one-off request with no threaded connection between them.

 Being stateless makes a lot of sense when sharing static information (like html, pdf, images. But what if you want a stateful setup for your web application?

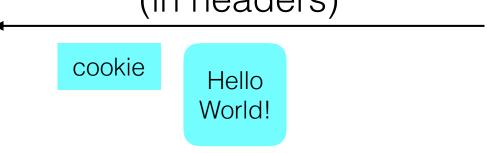
#### You need:

- A way to remember/track states.
- Uniquely identify each client for the server.
- A way of providing custom content for each client.

- You have several options to add and maintain states on top of HTTP:
- Client mechanisms:
  - Cookies
  - Hidden variables
  - URL rewriting
  - HTML5 local storage (for HTTP 1.1)
- Server mechanisms:
  - Sessions

- Cookies are a small piece of data sent by a server to a browser, and then sent back by the browser on future page requests.
  - Authentication
  - User tracking
  - Personalization: Maintaining user preferences, shopping carts, etc.

2. Server sends page and cookie (in headers)



3. Browser requests another page from same server



Web server

#### Browser side

 You can use document.cookie to manually set/get cookie data

```
document.cookie =
"username=smith;password=12345";
or
Cookies.set("username", "smith"); ...
alert(Cookies.get("username")); // smith
```

 Keep track of state on the client (e.g. which checkboxes selected)

## Cookies

- Mostly only visible to the application (not others on the same browser)
- HttpOnly: don't allow JavaScript to manipulate cookies (but still may be able to read them or learn of their existence)
- Secure: only send over HTTPS
- Caveat: doesn't seem to be definitive best practices yet

### Lifetime

- A **session** cookie, the default type, is a temporary cookie that is stored only in the browser's memory, so that when the browser is closed, it's erased.
  - Can't be used for long-term tracking.
  - Safer, because only the browser has access.

- Persistent cookies are stored in a a file on the browser's compute and can track long-term information
  - Potentially less secure because users (or the programs that they run) can open cookie files, see/change the cookie values, etc.
  - But...

#### Problems and Limitations

- Browsers can disable cookies. Users can disable and delete cookies (see Firebug).
- Size limit / expiration policy:
   <a href="http://browsercookielimits.squawky.net/">http://browsercookielimits.squawky.net/</a>
- Security issues: because they're stored in plain text, can be tampered with.
- Privacy issues: identification of the user
- Not handling cookie expiry well (expectations).

#### Hidden Variables

- An alternative to cookies is hidden variables that store state information in web pages rather than in the browser/client.
  - Cross-browser support.
  - For form-based web apps only.
    - Changing the URL, such as by clicking a hyperlink, loses the state.
    - Current submitted page is the current state, irrespective of what was submitted previously.

```
<input type="hidden" name="secret"
value="Don't tell anyone!!">
```

#### **URLs**

- You can also store the state in the URL such that the URL becomes a GET request.
  - Supported by all browsers.
  - Requires all URLs contain all state information (leading to long, unwieldly URLs).
  - Current submitted page represents current state.
  - Independent of what was done previously.

# Example

https://www.google.ca/maps/place/40+St+George +St,+Toronto,+ON+M5S+2E4/ @43.6596449,-79.3989808,17z/data=!3m1!4b1!4m5! 3m4!1s0x882b34c75165c957:0x1447336578b012b6! 8m2!3d43.659641!4d-79.3967921

## Sessions

- A server-side option is sessions, which store the current state on the server (i.e., in a file, database). Each request includes a **token** identifying the browser's session.
  - Tokens can be passed via cookies, hidden variables, URL rewriting).
  - At each request, the executing script uses the token to fetch the session state.
  - Beware session hijacking! Avoid by adding a unique value and signature.

# HTML 5 local storage

• window.localStorage - stores data with no
 expiration date

// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML
= localStorage.getItem("lastname");

http://html5demos.com/storage

# Session storage

Stores data for one session (data is lost when the browser tab is closed)