Cookies: Storing persistent data on the client

- Method by which server-side programs store simplified information on the client machine

- Cookies allow clients to store:
  - user preferences
  - a user-id or password that can be used later in session
  - keep track of activity/purchases
  - track user activity at the site.
Problems with Cookies

• The Plus: convenience to the user and added value to the site owner.

• The “Minus”: Security??:
  – Cookies are never interpreted nor executed; can’t insert a virus
  – Browsers generally accept only 20 cookies per domain and 300 in total and each is limited to 4k; can’t fill memory or launch denial of service attacks… Check your browser limits: Link

• Privacy:
  – Associate email with images (can send an email with images, attach cookies to those images, then identify your (email and all) if you visit that site.
  – Outside access to your coolie files can let someone use your stored credit info.

• Shouldn’t depend on Cookies
Process

- Server asks client to store cookie by supplying a **Set-Cookie** header
  - should contain cookie name and associated value \{(name,value) pair\}.
- Multiple cookies (up to 20/domain) can be specified by supplying more than one **set-cookie** line
- Browser sends previously created cookie back to the server by means of a **cookie** header
  - multiple cookies can be supplied by separating them with a ; on a single line
  - `Set-Cookie:cookieName=cookieValue;cNa=cVa`
Using JavaScript to Store and Examine Cookies - client side

- Cookies can be manipulated on client side using `document.cookie` property
  - if you lookup the value of `document.cookie`, you will get a single big string of all cookie values, as sent by the browser via the `Cookie` HTTP request header.
    - If the current page has cookies the value of “name1=val1, name2=val2, name3=val3
- You specify a single cookie at a time
  - `document.cookie = “name1 = val1”;`
  - `document.cookie = “name2 = val2 ; expires= +someDate”;`
  - `document.cookie = “name3 = val3 ; path=/; domain=test.com”`
• each time cookie is set it is stored by the browser
• cookie persists as long as the browser session
  (unless `expire=someDate`)
• local files cannot set cookies
Other Capabilities

- **expires**: Cookie is valid only during current session unless an expiration date is supplied in the form:
  
  ```
  Set-Cookie: cookieName=cookieValue; cNa=cVa;
  expires=Tuesday, 31-May-2008 13:59:59 GMT
  ```

- **path**: causes all URLs in the specified path to receive the cookie from the browser
  
  - otherwise it is sent only for URLs in the same directory or subdirectories

  ```
  Set-Cookie: cookieName=cookieValue; cNa=cVa; path=/
  ```
  applies to all URLs at site
Other Capabilities

- **domain**: Allows cookies to be shared across sites:
  
  ```
  Set-Cookie: cookieName=cookieValue; cNa=cVa;
  Domain = .dvdservice.com
  ```

- **secure**: causes this attribute to only be sent over secure links (this has no associated value)
  - otherwise it is sent only for URLs in the same directory or subdirectories (path)
  
  ```
  Set-Cookie; uid = alaink; secure
  Set-Cookie: password = abcdefg; secure
  ```

  secure is the only cookie attribute that has no associated value
Cascading Style Sheets (CSS)

• **CSS** is a simple mechanism for adding consistent style (e.g., fonts, colors, spacing) to Web documents

• **CSS** it simply places the styles in a file so that only the file has to be called whenever you wish to invoke that style

• See [https://www.w3.org/Style/Examples/011/firstcss](https://www.w3.org/Style/Examples/011/firstcss) which will easily get you started.

• Style sheet used in the class syllabus: