Announcements

- Quiz #5 on Friday
  - Topic: One-dimensional arrays

onload

- Allow us to execute code when the page is loaded
- **Example:** clock.html, clock.js

Sessions

- **Session** - time period during which a person views a number of different web pages in a browser and then quit.
- **What would you like**
  - To keep track of information throughout the session. For example, keeping track of color preferences, usernames, data selection, etc.
- **What is the problem?**
  - http (the protocol that makes possible the communication between browsers and web servers) is stateless (it has no memory)
  - Stateless - every page request is independent
- **One Possible Solution**
  - Cookies

Cookies

- Cookie - small piece of information sent by a server and stored either in the browser's memory or as a small file in the hard drive. Acceptance of the cookie depends on the client.
- Browser sends the cookie back with every request to the server that sent the cookie.
- Cookie - contains a name/value pair.
- Setting a cookie - associating a value with a name.
- Getting a cookie - getting the value associated with a name.
- Constrains:
  - Browser typically accept only 20 cookies per domain before dropping old cookies
  - 4KB per cookie
  - 300 cookies per domain

Cookies

- Each cookie consists of name, value, expiration date, host, and path information
- This is how the cookie information may look like when sent by the server in the http header

  Set-Cookie: automobile=nelyota; path=/; domain=notRealCars.com
- If no expiration date is set for a cookie, the cookie expires when the user's session expires (i.e., when the user closes the browser)
- If the user accesses any page matching the path and domain of the cookie, the browser will resend the cookie to the server.
- Let’s see cookies in our browser

Setting/Reading Cookies

- **Setting cookies**
  - We can set a cookie by using document.cookie

    document.cookie = “school=UMCP”;

    document.cookie = “mascot=terp”;
  - **Example:** setCookie.html
- **Reading cookies**
  - document.cookie has a string with all the cookies
  - You must extract from the string each cookie
  - Cookies are separated by ;
  - **Example:** readCookie.html

Cookies with an Expiration Date

- Cookies without an expiration date will expire when the browser is closed.
- Specify expiration date using “expires” and date in GMT
- GMT (Greenwich mean time)
  - Wdy, DD-Mon-YYYY HH:MM:SS GMT
  - Sun, 15-Apr-2007 11:29:00 GMT
- **Example:** setCookieExpiration.html
  - Syntax is very strict (you must have space after semicolon
- When updating a cookie make sure use the same features (expires, path, etc.)
- To delete a cookie set the expiration time to some point in the past.

Security (Email)

- Least secure of internet protocols
- Avoid sending sensitive information (e.g., passwords) over e-mail
- Provide e-mail addresses in web sites in a way is not easily recognized by spam programs
  - Use at rather than @
  - Put an image with the e-mail
  - Avoid mailto
- Encrypt the message using PGP (Pretty Good Privacy) or GPG (GNU Privacy Guard)

Security (Password-Protected Sites)

- **Approach not recommended**
  - Store encrypted password
  - Decrypt password and compare against user provided password
- **Better approach**
  - Store encrypted password
  - Encrypt provided password and compare against stored password

Security (Encryption)

- **Encryption –** process of converting plaintext into ciphertext
- **Decryption –** process of converting ciphertext into plaintext
- **Symmetric cryptography –** sender and receiver share the same key
- **Asymmetric (Public Key) cryptography –** sender and receiver have different, complementary keys
- **Symmetric cryptography**
  - Example algorithms: DES, Triple-Des, RC4
  - Relatively fast compared to Asymmetric
  - Drawbacks
    - Keys must be change frequently
    - How to distribute the key safely

Security (Encryption)

- **Branches of public key cryptography**
  - **Public key encryption**
  - **Digital signatures**
- **Public key Encryption**
  - Example algorithm: RSA
  - Relatively slowed compared to Asymmetric
  - How it works?
    - Each user has a public/private key pair
    - Public key is widely known
    - Private key only known by user that generated it
    - If user A wants to send user B a message, user A encrypts message with B's public key. B will decrypt the message with B's private key. The only way to decrypt the message is by using B's private key.
- **Digital signature**
  - Message signed with sender's private key can be verified by anyone with sender's public key thereby proving message authenticity

Digital Certificates (Certificates)

- **Digital Certificates** – electronic documents that contain information about a public key and the owner (name, address, etc.)
- Employed to verify a public key corresponds to a particular organization
- Certificates must be issue by a trusted third party known as certificate authority (CA) which guarantees the information is correct.
- **About certificates**
  - Have a validity period and can expire
  - They can be revoke
  - Browsers have a collection of root certificates
  - Main standard X.509

Message Digests

- Message digest – fixed-length representation of a message
- Expected properties for message digest ("Hashing") algorithm
  - Original message cannot be obtained from the digest
  - Two different messages should have different digests
- Example algorithms: MD5 and SHA

Need For Security

- **SSL (Secure Sockets Layer) Protocol** – Protocol that enable us to safisfy the need for security in client-web server transactions.
- The algorithm provides support for confidentiality, integrity and authentication.
- **SSL connection is established as follows**:
  - User connects to web server through the browser
  - Browser and server exchange public keys and certificate information
  - Browser checks server certificate validity (certificate not expired, issued by CA, etc.)
  - Optional: server can request a valid certificate from the client
  - Using public keys server and client determine a symmetric key to use
  - Communication from this point on is through symmetric cryptography

https

- **https** – http where
  - A different default port (443) is used
  - An extra layer of encryption/authentication exists between HTTP and TCP
- **https** – is not a separate protocol but a combination of HTTP over encrypted SSL or TLS transport mechanism
- **TLS** – Transport Layer Security
  - IETF standard designed to standardize SSL as an Internet protocol
  - Slight differences between SSL 3.0 and TLS 1.0

Security Sites

- www.securityfocus.com/
- www.cert.org/
- http://rootshell.com/