
Web Technology
John E. Savage
Brown University
Overview

• Browser technology
• Web protocols for accessing pages
• Illustrations of HTML
• Static versus dynamic web pages
• Client side and server side functions, e.g. CGI
• Database applications
• Sessions and cookies
What is a Browser?

• A browser is an application designed to access and display documents.

• Documents of different types are displayed by invoking interpreters appropriate to the type.

• Documents are accessed by local file name or via domain name or IP address.
Highlights of Browser History

• First web browser was invented by Tim Berners-Lee in 1990.
• In 1993 Marc Andreesen released Mosaic, the first popular browser supporting graphics
• He formed Netscape which released Netscape Navigator that peaked at 90% usage
• Microsoft’s Explorer emerged in ‘95 and usage reached 95% in 2002. Led to browser wars!
• Google’s Chrome was released in 2008 and reached 45% usage in 2014.
Accessing Files via Browsers

• Protocol & universal resource locator (URL) given
  – HTTP (HTTPS) access HTML pages (securely)
    • URL1 and URL2 are domain names
  – FTP, file transfer protocol, predates HTTP(S) by 20 yrs
    • URL3 is a domain name
  – FILE accesses local files
    • URL4 is a local file name

• If a protocol isn’t native to browser, such as mailto: it is passed to a local default app, such as mail.
Secure HTTP

• TLS is the transport layer protocol
  – Provides secure communication two hosts

• HTTPS is HTTP layered on top of TLS or SSL
  – Protects against packet sniffing and MITM attack

• SSL/TLS uses symmetric key encryption
  – Diffie-Hellman key exchange is used to create key
What is a Web Application?

- Front End
  - Web Browser
  - HTTP
- Back End
  - Web Server
- Database
- Server
HTTP Requests and Responses

• Implements the client/server model
• Clients make requests by specifying protocols and resource addresses
• Servers accept connections from clients and provide status code and resource, if available.
Accessing Webpages

• URL used
  – HTTP://www.cs.brown.edu:80/people/jes

• Accesses can involve queries
  – http://code.openhub.net/search?s=roman%20numerals&browser=Default

• Server responds with data
• Browser plays host to many layout engines.
  – File type determines which engine is invoked.
Supported Browser Document Types

- HTML and XML, including hyperlinks
- Images
- Video
- PDFs
- Text
- XML
- Scripting languages
- Databases
HyperText Markup Language (HTML)

```html
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body>
  <h1>My First Heading</h1>
  <p>My first paragraph.</p>
  <p>My second paragraph</p>
</body>
</html>
```

- HTML pages are special. They can embed other formats.

See http://www.w3schools.com/html/default.asp
HTML Demos

• Headings
• Font types, styles
• Background color, heading color
• Equations with subscripts and superscripts
• Images
• Links to other web pages
Static versus Dynamic Web Pages

• Static pages have fixed content
  – Not very useful

• Dynamic pages much more useful
  – May contain video, databases, animations
  – Page creation controlled by application server in browser that processes server-side scripts
  – Scripting languages include JavaScript, Perl, Python
  – Example: http://www.example.com/login.php
Changing Style and Content

• HTML has a tree structure called the Document Object Model or DOM
  – It is created inside the browser
  – Nodes represent objects

• A scripting language has access to the DOM
  – Determines how it is parsed and changed
  – It has access to objects and styles

• Both server-side and client-side scripting occur in dynamic pages.
Interactive Web Pages

• Client-side scripting may change a page in response to user inputs or time.
  – JavaScript, etc. are used on the client side

• Server-side scripting changes or adjusts pages between their transmission to the browser.
  – PHP, Perl, ASP, etc are used on the server side.
  – Designed to make string processing easy

• Server-side changes are determined by data in HTML form, parameters in URL, type of browser, passage of time, or state of the server or database.
Server-Side Scripting

• Server-side languages use the Common Gateway Interface (CGI) to produce dynamic pages.
• Server uses CGI to designate a directory holding scripts whose output may be delivered to client.
• The browser may send path information as well as parameters to server for input to a script.
  – Parameters passed via HTTP POST and GET requests.
CGI Programs

• URL: http://host/cgi-bin/cmd?args
  – cgi-bin is a special directory on the web server
  – cmd is the name of a normal executable in that directory
    • Shell script, perl, php, python, java jar file, c/c++ binary, ...
  – args are named arguments passed to command

• The program ‘cmd’ is run on the web server
  – Any program output is passed back to client
  – Typical Use: Format a request and pass it on to server

  – Used in very limited applications
CGI Deployment

• A web server can be configured to interpret a URL as a reference to a CGI script.
  – All executable files in cgi-bin at base of directory tree are treated as CGI scripts

• Another convention can be to interpret all files with extension .cgi as CGI scripts.
  – If a remote user can upload a file with .cgi suffix, server can be attacked.
Web Applications

Front End

Web Browser

HTTP

Back End

Web Server

Database

Server
The Web Server

• Sits on the host machine
  – Listens for connections on a particular port (i.e. 80)
  – Gets and processes HTTP requests sent to that port

• Given an HTTP Request
  – Return an HTTP Response

• Given a URL
  – Return the corresponding page

• Given a URL plus parameters
  – Compute and return the resultant data
Web Server Issues

• Handling large numbers of clients
  – Multiple threads, caching, multiple servers

• Managing context or state

• Generating HTML output containing computed values

• Doing the actual computations
  – At client or server
Databases

• Most web applications need to store information
  – Much of what they do is information based
  – Shopping site as an example

• The server code talks to a database system
  – All languages have code to make this relatively easy

• Database operations
  – Setting up the database
  – Adding and removing information from the database
  – Getting (querying) information from the database
Relational Databases

- A relational database is a set of TABLES (relations)
- A table is divided into FIELDS (columns)
  - Each field holds data of a single type
- The table’s ROWS (tuples) contain the actual data
  - Value for each field of the table
  - A row is a single data instance
- One (or more) fields might be the KEY
  - I.e. uniquely identify the row
Web Applications and HTTP

• The web application assumes it knows user
  – Server needs to know who the user is
    • Even if they haven’t logged in
  – One request follows another
  – Look up information based on the user

• HTTP is stateless
  – Each request is independent of previous requests
What is a Session?

• Need mechanism to maintain state
  – Within the server for a user and an app
  – Somewhat independent of the browser

• This notion of state is called a session
  – Information on a particular user, application, and use of the application

• A session represents connected series of actions
• Session properties are application dependent
Tracking Sessions

• CLIENT should track the session
  – Maintain as part of state for user
  – But need to send/get it from the browser
    • Server needs to tell browser the state for new pages
    • Browser needs to tell server the state for all requests
Tracking Sessions

HTTP Client

Login POST
username=david
password=david

Set-Cookie: SESSIONID=68C530AEC44D1605556619EC80C737C

HTTP is Stateless

Cookie: SESSIONID=68C530AEC44D1605556619EC80C737C

HTTP Server

Login successful?
1. create session id
2. return session id in cookie
3. store session id in database

SESSION_ID
SessionId
Username
createDate
expireDate
lastAccessDate

Database

Lookup Session ID
1. session match a username?
2. session still valid?

Content for 'david'
Session Identifiers

• How much information needs to be conveyed?
  – Really only need one piece of data, the session ID
    • Use this as an index to a table (or database) on the server
    • Table holds all the information related to the session

• Tracking Session IDs
  – Ensure validity (no spoofing; only server-generated IDs)
  – Ensure it is coming from same machine
  – Time out if not used for certain amount of time
Cookies

• Cookies are a general mechanism
  – To convey information between browser and server
  – Can associate name-value pairs with a URL
  – Sent automatically as part of the HTTP header
    • With any request to that particular URL

• Can be set either by server or browser
  – Server: header on a page can request a cookie set
  – Browser: JavaScript functions to define cookies
Cookie Properties

- Name and value associated with that name
- Maximum age
  - When cookie should be ignored/removed by browser
- Domain/port and path
  - When to include the cookie in a HTTP request
  - Domains can be as specific as desired
Demo of HTTP Request

- bash-3.2$ nc synful.io 80
- GET / HTTP/1.0
- HTTP/1.0 408 Request Time-out
- Cache-Control: no-cache
- Connection: close
- Content-Type: text/html

- <html><body><h1>408 Request Time-out</h1>
- Your browser didn't send a complete request in time.
- </body></html>
- bash-3.2$
Review

• Browser technology
• Web protocols for accessing pages
• Illustrations of HTML
• Static versus dynamic web pages
• Client side and server side functions, e.g. CGI
• Database applications
• Sessions and cookies