WEB BROWSERS
Why Do We Care?

- A web browser is one type of web client (the most obvious one)
- By understanding the issues and how it all fits together, we can build different types of web clients:
  - Agents
  - Proxies
  - Mobile apps
Basic Web Browser Responsibilities

- Generate and send requests to Web servers on the user’s behalf (URL, form submissions, images, etc)
- Accept responses delivered by Web servers and interpret them to produce the visual representation that the user will view
- Render the results in the browser window or through a third party tool
Additional Web Browser Tasks

- Caching: decide whether a request to Web server is needed at all
- Authentication: prompt user for credentials or use existing credentials
- State maintenance: cookie maintenance, as well as configurations for accepting or rejecting cookies
- Taking action in response to other headers and status codes: redirects, keeping connection open, etc
- Rendering complex objects: audio, video
- Dealing with error conditions
Core Functions/Modules of a Web Browser

- User Interface
- Request Generation
- Response Processing
- Networking
- Content Interpretation
- Caching
- State Maintenance
- Authentication
- Configuration
User Interface

- This module is responsible for providing the interface through which users interact with the application.
- This includes presenting, displaying, and rendering the end result of the browser’s processing of the response transmitted by the server.
Request Generation

- This module bears responsibility for the task of building HTTP requests to be submitted to HTTP servers.
- When asked by the User Interface module or the Content Interpretation module to construct requests based on relative links, it must first resolve those links into absolute URLs.
This module must parse the response, interpret it, and pass the result to the *User Interface* module.
Networking

- This module is responsible for network communications.
- Takes requests passed to it by the Request Generation module and transmits them over the network to the appropriate Web server or proxy.
- Accepts responses that arrive over the network and passes them to the Response Processing module.
- In the course of performing these tasks, it takes responsibility for establishing network connections and dealing with proxy servers specified in a user’s network configuration options.
Content Interpretation

- This module assists the Response Processing module in parsing and deciphering the content.
- If the content is encoded, this module decodes it.
- Initial responses often have their content types set to text/html, but HTML responses embed or contain references to images, multimedia objects, JavaScript code, applets, and style sheet information. This module performs the additional processing necessary for browser applications to understand these entities within a response.
- In addition, this module must tell the Request Generation module to construct additional requests for the retrieval of auxiliary content such as images, applets, and other objects.
Caching

- Caching provides web browsers with a way to economize by avoiding the unnecessary retrieval of resources that the browser already has a usable copy of, ‘cached’ away in local storage.

- Browsers can ask Web servers whether a desired resource has been modified since the time that the browser initially retrieved it and stored it in the cache.

- This module must provide facilities for
  - storing copies of retrieved resources
  - accessing those copies when viable
  - managing the space (both memory and disk) allocated for caching
State Maintenance

- Since HTTP is a *stateless protocol*, some mechanism must be in place to maintain the browser *state* between related requests and responses.
- Cookies are the mechanism of choice for performing this task, and support for cookies is in the responsibility of this module.
Authentication

- This module takes care of composing authorization credentials when requested by the server.
- It must interpret response headers demanding credentials by prompting the user to enter them (usually via a dialog).
- It must also store those credentials, but only for the duration of the current browser session, in case a request is made for another secured resource in what the server considers to be the same security 'realm'.
Configuration

- This module maintains the fixed and variable configurable options for the browser and provides an interface for the user to modify those settings that are under user control.
Browser Request Generation

1. User follows link
2. Do I already have a copy of this resource?
3. Do I need to send authorization credentials?
4. Do I need to include cookie headers?
5. Request is prepared
6. Request is transmitted
User Interface Events

- Enter URL manually
- Selecting previously visited links
  - History
  - Back button
- Selecting displayed hyperlinks
Resolving Links

- Absolute: don’t need to be resolved
- Relative: location is specified relative to
  - Current location being displayed – when HREF contains a relative path that does not begin with slash
    
    ```html
    <A HREF="some directory/just a file name.html">
    ```
  - Current location’s web server root – when HREF contains a relative path that does begin with a slash
    
    ```html
    <A HREF="/root level directory/another file name.html">
    ```
- Process of resolution changes if `<BASE HREF..>` found in HEAD section of page
Browser Response Processing
Response Processing

- Once a request has been transmitted, the browser waits to receive a response.
- The browser may submit additional requests while waiting.
- Requests may have to be resubmitted if the connection is closed before the corresponding responses are received.
- The Web server is supposed to transmit responses in the same order as the corresponding requests were received. However, the browser is responsible for dealing with servers that do not properly maintain this order.
HTTP Requests

- Before building the request:
  - Does the browser already have a cached copy?
  - Is there additional info the browser needs to send as part of the request?
    - State maintenance -- cookies
  - Is there OTHER additional info that needs to be sent as part of the request?
    - Authorization
HTTP Request

Format:

METHOD /path-to-resource HTTP/version-number
Header-Name-1: value
Header-Name-2: value

[ optional request body ]

Example:

POST /update.cgi HTTP/1.0
Host: www.somewhere.com
Referer: http://www.somewhere.com/formentry.html

name=joe&type=info&amount=5
HTTP Request Generation

- Request line (first line):
  - ‘method’ (GET, PUT, POST, etc.)
  - ‘/path-to-resource’ (representing the path portion of the requested URL)
  - ‘version-number’ (specifying the version of HTTP associated with the request --1.0 or 1,1)
HTTP Request Generation

- Headers
  - Host
  - User-Agent (software generating request)
  - Referer (referring page)
  - Date
  - Accept (MIME types client will accept as a response)
  - Content-Type and -Length
  - Cookie
  - Authorization
HTTP Request Generation

- Request Body – for methods like POST and PUT
- Simplest example: including form parameters in message body when using POST
- Must be URL-encoded to enable proper parsing by server, so Content-Type must be application/x-www-form-urlencoded
- SOAP: Body of a request consists of a SOAP payload (XML doc containing RPC directives such as method calls and parameters)
HTTP Responses

HTTP/version-number  status-code  message
Header-Name-1:  value
Header-Name-2:  value

[ response body ]

HTTP/1.1  200  OK
Content-Type:  text/html
Content-Length:  1234

... 

<HTML>
<HEAD>
<TITLE>...</TITLE>
</HEAD>
<BODY  bgcolor="#ffffff">
<H2  align="center">...</H2>
...
</H2>
</BODY>
</HTML>
Status Code Classes

- *informational* status codes (1xx),
- *successful response* status codes (2xx),
- *redirection* status codes (3xx),
- *client request error* status codes (4xx), and
- *server error* status codes (5xx)
Processing Successful Responses

- Browser should take the associated content and render it in accordance with specifications in headers
  - Encoding?
  - Content-Type? (Content-Type: mime-type/mime-subtype)
  - Content-Length?
  - Set-Cookie?
Processing Responses with Other Status Codes

- Invalid Request (400)
- URL can’t be found on the server (404)
- Not authorized (401)
- Redirections (301 or 302)
- Some of these may come with message bodies which should be rendered
Authentication

GET /protected/index.html HTTP/1.1
Host: secret.resource.com

HTTP/1.1 401 Not Authorized
Date: Sun, 11 Feb 2001 22:28:31 GMT
WWW-Authenticate: Basic realm ="Top Seekrit"
Content-type: text/html

GET /protected/index.html HTTP/1.1
Host: secret.resource.com
Authorization: Basic encoded-userid:password
Multimedia Support

- Browsers can provide native support (most do this for HTML, plain text, GIF and JPEG)

- Helpers: Simple to implement. Browsers can be configured to defer presentation of formats to other programs (.ppt, for example)
  - Downside: It takes you out of the browser

- Plug-ins: Uses the `<OBJECT>` HTML tag. Browser maintains a table defining what actions should be taken for various MIME types.