Uniform Resource Locators (URL)

- A Uniform Resource Locator (URL) is a compact representation of the location and access method for a resource available via the Internet.

- General form of a URL:
  \(<\text{scheme}> : <\text{scheme-specific-part}>\)

- URL Schemes:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Default Port</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>21</td>
<td>ftp://&lt;user&gt;;&lt;password&gt;@&lt;host&gt;:&lt;port&gt;/&lt;cwd1&gt;/&lt;cwd2&gt;/.../&lt;cwdN&gt;/&lt;name&gt;;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>type=&lt;typecode&gt;</td>
</tr>
<tr>
<td>http</td>
<td>80</td>
<td>http://&lt;host&gt;:&lt;port&gt;/&lt;path&gt;?&lt;searchpart&gt;</td>
</tr>
<tr>
<td>gopher</td>
<td>70</td>
<td>gopher://&lt;host&gt;:&lt;port&gt;/&lt;selector&gt; or gopher://&lt;host&gt;:&lt;port&gt;/&lt;selector&gt;%09&lt;search&gt; or gopher://&lt;host&gt;:&lt;port&gt;/&lt;selector&gt;%09&lt;search&gt;%09&lt;gopher+_string&gt; or mailto:&lt;rfc822-addr-spec&gt;</td>
</tr>
<tr>
<td>mailto</td>
<td>—</td>
<td>mailto:&lt;rfc822-addr-spec&gt;</td>
</tr>
<tr>
<td>news</td>
<td>—</td>
<td>news:&lt;newsgroup-name&gt; or news:&lt;message-id&gt;</td>
</tr>
<tr>
<td>nntp</td>
<td>119</td>
<td>nntp://&lt;host&gt;:&lt;port&gt;/&lt;newsgroup-name&gt;/&lt;article-number&gt;</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
<td>telnet://&lt;user&gt;;&lt;password&gt;@&lt;host&gt;:&lt;port&gt;</td>
</tr>
<tr>
<td>wais</td>
<td>210</td>
<td>wais://&lt;host&gt;:&lt;port&gt;/&lt;database&gt; or wais://&lt;host&gt;:&lt;port&gt;/&lt;database&gt;?&lt;search&gt; or wais://&lt;host&gt;:&lt;port&gt;/&lt;database&gt;/&lt;wtype&gt;/&lt;wpath&gt; or file://&lt;host&gt;/&lt;path&gt;</td>
</tr>
<tr>
<td>file</td>
<td>—</td>
<td>file://&lt;host&gt;/&lt;path&gt;</td>
</tr>
<tr>
<td>prospero</td>
<td>1525</td>
<td>prospero://&lt;host&gt;:&lt;port&gt;/&lt;hostname&gt;;&lt;field&gt; = &lt;value&gt;</td>
</tr>
</tbody>
</table>

ftp://myhost.sci.brooklyn.cuny.edu/ziegler/myfile.txt
http://www.brooklyn.cuny.edu/index.html
mailto:webmaster@brooklyn.cuny.edu
Hypertext Transfer Protocol (HTTP)

- HTTP is a protocol for transmitting information with the efficiency necessary for making hypertext jumps.

- The data transferred can be plain text, hypertext, audio, images, or any Internet accessible information.

- HTTP is a transaction-oriented, client-server protocol.

- HTTP uses TCP. Nevertheless, HTTP is 'stateless'; i.e., each transaction is treated independently.

- HTTP is the foundation protocol of the World Wide Web (WWW). The most typical use of HTTP is between a Web browser and a Web server.

- HTTP can be used in any client-server application involving hypertext.

- **Universal Resource Identifier (URI):**
  A URI is a term for a generic WWW identifier. The URL is a type of URI, in which an access protocol is designated and a specific Internet address is provided. (The HTTP standard refers to URIs.)
**Key Terms related to HTTP**

<table>
<thead>
<tr>
<th>Key Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cache</strong></td>
<td>A program’s local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.</td>
</tr>
<tr>
<td><strong>Client</strong></td>
<td>An application program that establishes connections for the purpose of sending requests.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>A transport layer virtual circuit established between two application programs for the purposes of communication.</td>
</tr>
<tr>
<td><strong>Entity</strong></td>
<td>A particular representation or rendition of a data resource, or reply from a service resource, that may be enclosed within a request or response message. An entity consists of entity headers and an entity body.</td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>A server that acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the original server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>The basic unit of HTTP communication, consisting of a structured sequence of octets transmitted via the connection.</td>
</tr>
<tr>
<td><strong>Origin Server</strong></td>
<td>The server on which a given resource resides or is to be created.</td>
</tr>
<tr>
<td><strong>Proxy</strong></td>
<td>An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.</td>
</tr>
<tr>
<td><strong>Resource</strong></td>
<td>A network data object or service that can be identified by a URI.</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>An application program that accepts connections in order to service requests by sending back responses.</td>
</tr>
<tr>
<td><strong>Tunnel</strong></td>
<td>A tunnel is an intermediary program that is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. A tunnel ceases to exist when both ends of the relayed connections are closed. Tunnels are used when a portal is necessary and the intermediary cannot, or should not, interpret the relayed communication.</td>
</tr>
<tr>
<td><strong>User Agent</strong></td>
<td>The client that initiates a request. These are often browsers, editors, spiders, or other end-user tools.</td>
</tr>
</tbody>
</table>
Examples of HTTP Operation

a) Direct Connection
b) Connection through one or more Intermediate Systems (proxy, gateway, tunnel).
c) Example of a Cache.
Intermediate HTTP Systems

- Proxy:
  - Security Intermediary
    On client side of firewall
  - Different versions of HTTP:
- Gateway:
  - Security Intermediary:
    On server side of firewall
  - Non-HTTP Server
- Tunnel:
  A relay point between two TCP connections (e.g., a firewall through which a client or server external to a protected network can establish and maintain an authenticated connection for HTTP transactions.)
HTTP Messages

HTTP-Message = Simple-Request | Simple-Response | Full-Request | Full-Response
Full-Request = Request-Line
  *( General-Header | Request-Header | Entity-Header )
  CRLF
  [ Entity-Body ]
Full-Response = Status-Line
  *( General-Header | Response-Header | Entity-Header )
  CRLF
  [ Entity-Body ]
Simple-Request = “GET” SP Request-URI CRLF
Simple-Response = [ Entity-Body ]

General structure of HTTP messages.
HTTP Message Fields

- **Request-Line:**
  Identifies the message type and the requested resource.
  \[ \text{Request-Line} = \text{Method SP Request-URL SP HTTP-Version CRLF} \]

- **Status Line:**
  Provides status information about this response.
  \[ \text{Status-Line} = \text{HTTP-Version SP Status-Code SP Reason-Phrase CRLF} \]

- **General-Header:**
  Contains fields that are applicable to both request and response messages, but which do not apply to the entity being transferred.

- **Request-Header:**
  Contains information about the request and the client.

- **Response-Header:**
  Contains information about the response.

- **Entity-Header:**
  Contains information about the resource identified by the request and information about the entity body.

- **Entity-Body:**
  The body of the message.
Augmented BNF (Backus-Naur Form) Notation

Augmented BNF notation used in URL and HTTP specifications.

- Words in lower case represent variables or names of rules.
- A rule has the form

  name = definition

- DIGIT is any decimal digit; CRLF is carriage return, line feed; SP is one or more spaces.
- Quotation marks enclose literal text.
- Angle brackets, "<" ">", may be used within a definition to enclose a rule name when their presence will facilitate clarity.
- Elements separated by bar ("|") are alternatives.
- Ordinary parentheses are used simply for grouping.
- The character "*" preceding an element indicates repetition. The full form is

  <I>*<J>element

  indicating at least I and at most J occurrences of the element. *element allows any number, including 0; 1*element requires at least one element; and 1*2element allows 1 or 2 elements; <N>element means exactly N elements.
- Square brackets, "[" "]", enclose optional elements.
- The construct "#" is used to define, with the following form,

  <I>#$<J>element

  indicating at least I and at most J elements, each separated by a comma and optional linear white space.
- A semicolon at the right of a rule starts a comment that continues to the end of the line.
# HTTP Elements

HTTP elements.

<table>
<thead>
<tr>
<th>ALL MESSAGES</th>
<th>ENTITY HEADER FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL HEADER FIELDS</td>
<td>ENTITY HEADER FIELDS</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>Allow</td>
</tr>
<tr>
<td>Connection</td>
<td>Content-Encoding</td>
</tr>
<tr>
<td>Data</td>
<td>Content-Language</td>
</tr>
<tr>
<td>Forwarded</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td>Content-Range</td>
</tr>
<tr>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td>Content-Version</td>
</tr>
<tr>
<td></td>
<td>Derived-From</td>
</tr>
<tr>
<td></td>
<td>Expires</td>
</tr>
<tr>
<td></td>
<td>Last-Modified</td>
</tr>
<tr>
<td></td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td>Transfer-Encoding</td>
</tr>
<tr>
<td></td>
<td>URI-Header</td>
</tr>
<tr>
<td></td>
<td>extension-header</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REQUEST MESSAGES</th>
<th>REQUEST HEADER FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS</td>
<td>Accept</td>
</tr>
<tr>
<td>GET</td>
<td>Accept-Charset</td>
</tr>
<tr>
<td>HEAD</td>
<td>Accept-Encoding</td>
</tr>
<tr>
<td>POST</td>
<td>Accept-Language</td>
</tr>
<tr>
<td>PUT</td>
<td>Authorization</td>
</tr>
<tr>
<td>PATCH</td>
<td>From</td>
</tr>
<tr>
<td>COPY</td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td>If-Modified-Since</td>
</tr>
<tr>
<td></td>
<td>Proxy-Authentication</td>
</tr>
<tr>
<td></td>
<td>Range</td>
</tr>
<tr>
<td></td>
<td>Referer</td>
</tr>
<tr>
<td></td>
<td>Unless</td>
</tr>
<tr>
<td></td>
<td>User-Agent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE MESSAGES</th>
<th>RESPONSE HEADER FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE STATUS CODES</td>
<td>RESPONSE HEADER FIELDS</td>
</tr>
<tr>
<td>Continue</td>
<td>Moved Temporarily</td>
</tr>
<tr>
<td>Switching Protocols</td>
<td>See Other</td>
</tr>
<tr>
<td>OK</td>
<td>Not Modified</td>
</tr>
<tr>
<td>Created</td>
<td>Use Proxy</td>
</tr>
<tr>
<td>Accepted</td>
<td>Bad Request</td>
</tr>
<tr>
<td>Non-Authoritative Information</td>
<td>Unauthorized Payment Required</td>
</tr>
<tr>
<td>No Content</td>
<td>Forbidden</td>
</tr>
<tr>
<td>Reset Content</td>
<td>Not Found</td>
</tr>
<tr>
<td>Partial Content</td>
<td>Method Not Allowed</td>
</tr>
<tr>
<td>Multiple Choices</td>
<td>None Acceptable</td>
</tr>
<tr>
<td>Moved Permanently</td>
<td>Proxy Authentication Required</td>
</tr>
<tr>
<td></td>
<td>Request Timeout</td>
</tr>
<tr>
<td></td>
<td>Conflict</td>
</tr>
<tr>
<td></td>
<td>Gone</td>
</tr>
<tr>
<td></td>
<td>Length Required</td>
</tr>
<tr>
<td></td>
<td>Unless True</td>
</tr>
<tr>
<td></td>
<td>Internal Server Error</td>
</tr>
<tr>
<td></td>
<td>Not Implemented</td>
</tr>
<tr>
<td></td>
<td>Bad Gateway</td>
</tr>
<tr>
<td></td>
<td>Service Unavailable</td>
</tr>
<tr>
<td></td>
<td>Gateway Timeout</td>
</tr>
<tr>
<td></td>
<td>extension code</td>
</tr>
<tr>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Proxy-Authenticate</td>
</tr>
<tr>
<td></td>
<td>Public</td>
</tr>
<tr>
<td></td>
<td>Retry-After</td>
</tr>
<tr>
<td></td>
<td>Server</td>
</tr>
<tr>
<td></td>
<td>WWW-Authenticate</td>
</tr>
</tbody>
</table>
HTTP General Header Fields

General Header Fields can be used in both request and response messages. They contain information that does not apply directly to the entity being transferred.

- **Cache-Control:**
  Specifies directives that must be obeyed by any caching mechanisms along the request/response chain.

- **Connection:**
  Contains list of keywords and header field names applying only to this TCP connection between the sender and nearest nontunnel recipient.

- **Date:**
  Date and time at which the message originated.

- **Forwarded:**
  Used by gateways and proxies to indicate intermediate steps along a request or response chain. Each gateway or proxy that handles a message may attach a Forwarded field that gives its URL.

- **Keep-Alive:**
  May indicate a max time that sender will keep connection open or max number of additional requests that will be allowed

- **MIME-Version:**
  Indicates version of MIME used by message

- **Pragma:**
  Contains implementation-specific directives

- **Upgrade:**
  Used to specify what additional protocols the client supports and would like to use; used in a response to indicate which protocol will be used.
HTTP Request Methods

- **OPTIONS:**
  request info about the options available for URL

- **GET:**
  request to retrieve information identified in the URL

- **HEAD:**
  request to get information about a resource without transferring its body.

- **POST:**
  accept attached entity as a new subordinate to the identified URL

- **PUT:**
  accept attached entity and store as supplied URL

- **PATCH:**
  like PUT, but contains a list of differences from original URL

- **COPY/MOVE:**
  request copy/move of the URL to Entity-Header loc

- **DELETE:**
  request server delete the URL

- **LINK/UNLINK:**
  make/remove link from URL to Entity-Header loc

- **TRACE:**
  request server return received entity body for test/diagnostic use

- **WRAPPED:**
  to send one or more encapsulated requests

- **Extension-method:**
  additional methods, may not be recognized by recipient
HTTP Request Header Fields
Request Header Fields function as request modifiers.

- **Accept:** list of media types and ranges acceptable in response to this request
- **Accept-Charset:** list of character sets acceptable for the response
- **Accept-Encoding:** list of acceptable content encodings for entity body
- **Accept-Language:** set of natural languages preferred for the response
- **Authorization:** contains *credentials* for client to authenticate itself to the server
- **From:** Internet e-mail address for human user running requesting user agent
- **Host:** specifies the Internet host being requested
- **If-Modified-Since:** with GET method, resource is transferred only if it has been modified since the date/time specified
- **Proxy-Authorization:** allow client to authenticate itself to the proxy
- **Range:** in GET message allow client to request portion of identified resource
- **Referrer:** URL of resource from which the Request-URL was obtained
- **Unless:** Similar to If-Modified-Since field, but not just GET method and comparison based on any Entity-Header field value
- **User-Agent:** contains info about the originating user agent
HTTP Response Status Line Codes

• **Informational (1xx):**
  The request has been received and processing continues. No entity body accompanies this response.

• **Successful (2xx):**
  The request was successfully received, understood, and accepted.
  The information returned in the response message depends on the request method, as follows:
  — GET:
    contents of the entity-body corresponds to the requested resource.
  — HEAD:
    No entity body is returned.
  — POST:
    The entity describes or contains the result of the action.
  — TRACE:
    The entity contains the request message.
  — Other methods:
    The entity describes the result of the action.

• **Redirection (3xx):**
  Further action is required to complete the request.

• **Client Error (4xx):**
  The request contains a syntax error or the request cannot be fulfilled.

• **Server Error (5xx):**
  The server failed to fulfill an apparently valid request.
HTTP Response Header Fields

Response Header Fields provide additional information related to the response that cannot be placed in the Status-Line.

• **Location:**
  Defines the exact location of the resource identified by the Request-URL.

• **Proxy-Authenticate:**
  Included with a response that has a status code of Proxy Authentication Required.
  This field contains a "challenge" that indicates the authentication scheme and parameters required.

• **Public:**
  Lists the nonstandard methods supported by this server.

• **Retry-After:**
  Included with a response that has a status code of Service Unavailable, and indicates how long the service is expected to be unavailable.

• **Server:**
  Identifies the software product used by the origin server to handle the request.

• **WWW-Authenticate:**
  Included with a response that has a status code of Unauthorized.
  This field contains a "challenge" that indicates the authentication scheme and parameters required.
HTTP Entity Header Fields
Provides optional information about the entity body or if no body is present, about the resource identified by the request.

- **Allow:**
  lists methods supported by the resource identified in Request-URL.
- **Content-Encoding:**
  Indicates content encodings applied to resource (e.g., zip)
- **Content-Language:**
  Identifies natural language(s) of enclosed entity.
- **Content-Length:**
  The size of the entity body in octets.
- **Content-MD5:**
  MD5 hash of resource
- **Content-Range:**
  indicate portion of identified resource included in response
- **Content-Type:**
  the media type of the entity body (e.g., text/html)
- **Content-Version:**
  version tag associated with an evolving entity
- **Derived-From:**
  version tag of resource this entity was derived from
- **Expires:**
  Date/time after which the entity should be considered stale.
- **Last-Modified:**
  Date/time that resource was last modified.
- **Link:**
  Defines links to other resources.
- **Title:**
  A textual title for the entity.
- **Transfer-Encoding:**
  type of transformation applied to message body (e.g., chunked - break an entity body into labeled chunks).
- **URL-Header:**
  Informs recipient of other URLs used for this resource
- **Extension-Header:**
  additional fields, may not be recognized by recipient
HTTP Entity Body

• An entity may represent a data resource, or it may constitute other information supplied with a request or response.

• An entity body consists of an arbitrary sequence of octets.

• HTTP is designed to be able to transfer any type of content, including text, binary data, audio, images, and video.

• When an entity body is present in a message, the interpretation of the octets in the body is determined by the entity header fields:
  - Content-Type:
    determines data interpretation
  - Content-Encoding:
    that was applied to the data (if any)
  - Transfer-Encoding:
    applied to form the entity body of the message

• These define a three-layer, ordered encoding model:
  \[ \text{entity-body} ::= \text{Transfer-Encoding}(\text{Content-Encoding}(\text{Content-Type}(\text{data})))) \]
Sample HTTP Request and Response

- Request: GET /rfc.html HTTP/1.1 Host:www.ietf.org
- Response:

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is \']\'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Page: avoid browser bug

<html>
<head>
<title>IETF RFC Page</title>

<script language="javascript">
function url() {
  var x = document.form1.number.value
  if (x.length == 1) {x = "000" + x }
  if (x.length == 2) {x = "00" + x }
  if (x.length == 3) {x = "0" + x }
  document.form1.action = "/rfc/rfc" + x + ".txt"
  document.form1.submit
}
</script>

</head>
```
The World Wide Web (WWW)

- The World Wide Web is a large scale, on-line repository of information that users can search using an interactive application program called a browser.

- Technically, the web is a distributed hypermedia system that supports interactive access.

- Information is stored in a set of documents.

- Besides the basic information, a document can contain pointers to other documents in the set.

- A hypermedia document available on the Web is called a page. The root page for an organization or individual is called the homepage.

- Each Web page that contains a hypermedia document uses a standard representation known as Hypertext markup language (HTML).

- HTML allows an author to give general guidelines for display and to specify the contents of the page. (Consequently, two browsers may display a page differently.)
Browser Architecture

- A browser interacts with a Web server using HTTP.
- A browser consists of a set of clients, a set of interpreters, and a controller that manages them.

Major components of a Web browser. Dark arrows show the flow of data; other arrows show control paths. The data paths from clients to interpreters are not shown.
HTML Format and Representation

- Each HTML document consists of two parts: a head followed by a body.

- Each HTML document is represented as a text file that contains tags along with other information.

- HTML tags provide structure for the document as well as formatting hints.

- General Form of an HTML Document:
  \[
  \text{<HTML>}
  \text{<HEAD>}
  \text{<TITLE>text that forms the document title</TITLE>}
  \text{</HEAD>}
  \text{<BODY>body of the document appears here</BODY>}
  \text{</HTML>}
\]

- Equivalent Document:
  \[
  \text{<HTML><HEAD><TITLE>text that forms the document title</TITLE><HEAD><BODY>body of the document appears here</BODY><HTML>}
\]
Example HTML Formatting Tags

- Headings:
  
  ```html
  <center><h1>Heading of Page</h1></center>
  
  Heading of Page
  ```

- Line Breaks:
  
  ```html
  Hello.<br />
  This is an example<br />
  of HTML
  
  Hello.
  this is an example
  of HTML
  ```

- Unordered Lists:
  
  ```html
  Here is a list of 4 names:
  <ul>
  <li> Scott </li>
  <li> Sharon </li>
  <li> Jan </li>
  <li> Rebecca </li>
  </ul>
  
  This text occurs after the list
  ```

Here is a list of 4 names:

- Scott
- Sharon
- Jan
- Rebecca

This text occurs after the list.
Ordered Lists:
Here is a list of 4 names:
<ol>
<li> Scott </li>
<li> Sharon </li>
<li> Jan </li>
<li> Rebecca </li>
</ol>
This text occurs after the list

Here is a list of 4 names:
1. Scott
2. Sharon
3. Jan
4. Rebecca
This text occurs after the list.

Bold, Italic, and Underlined Text:
<b> Bold text </b> <br /> <i> Italic text </i>
<br /> <u> Underlined text </u>

**Bold text**
*Italic text*
*Underlined text*

Paragraphs
Use <p> and </p> tags to delimit paragraphs.
Embedding Graphical Images Into a Web Page

• Example:
Here is a picture. `<img src="http://www.site.com/myphoto.gif" align=middle />`

Illustration of image alignment. As requested in the tag, text on the line is positioned in the middle of the image.

Graphics files should be in either .gif or .jpg (or .jpeg) format only. Other formats will not work with all browsers.

• Background picture (wallpaper):

`<body background = "http://www.site.com/myphoto.jpg">`
Hypertext Links to Other Documents

• The HTML mechanism for specifying a hypertext reference is called an anchor.

• Example:
  This book is published by
  <A HREF="http://www.prenhall.com">
  Prentice-Hall, </A> one of the larger publishers of Computer Science textbooks.

• Example Output:
  This book is published by Prentice-Hall, one of the larger publishers of Computer Science textbooks.
Optional Clients

- Example of an FTP Link:
  Many of the examples in this text are available online. Source code from
  <A HREF="ftp://ftp.site.com/netbook/prog.c">
  an example C program
  </A>
  or code from a
  <A HREF="ftp://ftp.site.com/netbook/prog2.c">
  second example C program
  </A>
  are available

- Example Output:
  Many of the examples in this text are available online. Source code from an example C program or code from a second example C program are available
# A Selection of Common HTML Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;html&gt;</code> ... <code>&lt;/html&gt;</code></td>
<td>Declares the Web page to be written in HTML</td>
</tr>
<tr>
<td><code>&lt;head&gt;</code> ... <code>&lt;/head&gt;</code></td>
<td>Delimits the page’s head</td>
</tr>
<tr>
<td><code>&lt;title&gt;</code> ... <code>&lt;/title&gt;</code></td>
<td>Defines the title (not displayed on the page)</td>
</tr>
<tr>
<td><code>&lt;body&gt;</code> ... <code>&lt;/body&gt;</code></td>
<td>Delimits the page’s body</td>
</tr>
<tr>
<td><code>&lt;h n&gt;</code> ... <code>&lt;/h n&gt;</code></td>
<td>Delimits a level <code>n</code> heading</td>
</tr>
<tr>
<td><code>&lt;b&gt;</code> ... <code>&lt;/b&gt;</code></td>
<td>Set ... in boldface</td>
</tr>
<tr>
<td><code>&lt;i&gt;</code> ... <code>&lt;/i&gt;</code></td>
<td>Set ... in italics</td>
</tr>
<tr>
<td><code>&lt;center&gt;</code> ... <code>&lt;/center&gt;</code></td>
<td>Center ... on the page horizontally</td>
</tr>
<tr>
<td><code>&lt;ul&gt;</code> ... <code>&lt;/ul&gt;</code></td>
<td>Brackets an unordered (bulleted) list</td>
</tr>
<tr>
<td><code>&lt;ol&gt;</code> ... <code>&lt;/ol&gt;</code></td>
<td>Brackets a numbered list</td>
</tr>
<tr>
<td><code>&lt;li&gt;</code> ... <code>&lt;/li&gt;</code></td>
<td>Brackets an item in an ordered or numbered list</td>
</tr>
<tr>
<td><code>&lt;br&gt;</code></td>
<td>Forces a line break here</td>
</tr>
<tr>
<td><code>&lt;p&gt;</code></td>
<td>Starts a paragraph</td>
</tr>
<tr>
<td><code>&lt;hr&gt;</code></td>
<td>Inserts a horizontal rule</td>
</tr>
<tr>
<td><code>&lt;img src=&quot;...&quot;&gt;</code></td>
<td>Displays an image here</td>
</tr>
<tr>
<td><code>&lt;a href=&quot;...&quot;&gt; ... &lt;/a&gt;</code></td>
<td>Defines a hyperlink</td>
</tr>
</tbody>
</table>
Welcome to AWI's Home Page

We are so happy that you have chosen to visit Amalgamated Widget's home page. We hope you will find all the information you need here.

Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.

Product Information
- Big widgets
- Little widgets

Telephone numbers
- 1-800-WIDGETS
- 1-415-765-4321
### Sample HTML Table

```html
<html>
<head> <title> A sample page with a table </title> </head>
<body>
<table border=1 rules=all>
<caption> Some Differences between HTML Versions </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th> Item </th> <th> HTML 1.0 </th> <th> HTML 2.0 </th> <th> HTML 3.0 </th> <th> HTML 4.0 </th> 
<tr> <th> Hyperlinks </th> x x x x 
<tr> <th> Images </th> x x x x 
<tr> <th> Lists </th> x x x x 
<tr> <th> Active Maps and Images </th> &nbsp; x x x x 
<tr> <th> Forms </th> &nbsp; &nbsp; &nbsp; &nbsp; x x 
<tr> <th> Equations </th> &nbsp; &nbsp; &nbsp; &nbsp; x x 
<tr> <th> Toolbars </th> &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; x 
<tr> <th> Tables </th> &nbsp; &nbsp; &nbsp; &nbsp; x x 
<tr> <th> Accessibility features </th> &nbsp; &nbsp; &nbsp; &nbsp; x 
<tr> <th> Object embedding </th> &nbsp; &nbsp; &nbsp; &nbsp; x 
<tr> <th> Scripting </th> &nbsp; &nbsp; &nbsp; &nbsp; x 
</table>
</body>
</html>
```

### Some Differences between HTML Versions

<table>
<thead>
<tr>
<th>Item</th>
<th>HTML 1.0</th>
<th>HTML 2.0</th>
<th>HTML 3.0</th>
<th>HTML 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperlinks</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Images</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Lists</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Active Maps and Images</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Forms</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Equations</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toolbars</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tables</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Accessibility features</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Object embedding</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Scripting</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Sample HTML Form

```html
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size=4> Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10> Expires <input name="expires" size=4> M/C <input name="cc" type=radio value="mastercard"> VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive"> Little <input name="product" type=radio value="cheap"> Ship by express courier <input name="express" type=checkbox> </p>
<p> <input type=submit value="submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

Widget Order Form

Name: ____________________________
Street address: ____________________
City: _______ State: _______ Country: _______
Credit card #: __________ Expires: _______ M/C: _______ Visa: _______
Widget size: Big: _______ Little: _______ Ship by express courier: _______

[Submit order]

Thank you for ordering an AWI widget, the best widget money can buy!
Sample Browser Form Response

• When the user clicks the "Submit" button, the browser packages the collected information into a single long line and sends it back to the server for processing.

• The '&' separates fields and the '+' is used to represent a space.

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&product=cheap&express=on
```
XML and XSL

- XML (eXtensible Markup Language) and XSL (eXtensible Style Language) allow Web pages to be structured for automated processing.

- XML describes content and XSL describes the formatting independently of the content.

- Sample XML Web Page:

```xml
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="b5.xsl"?>
<book_list>
  <book>
    <title> Computer Networks, 4/e </title>
    <author> Andrew S. Tanenbaum </author>
    <year> 2003 </year>
  </book>
  <book>
    <title> Modern Operating Systems, 2/e </title>
    <author> Andrew S. Tanenbaum </author>
    <year> 2001 </year>
  </book>
  <book>
    <title> Structured Computer Organization, 4/e </title>
    <author> Andrew S. Tanenbaum </author>
    <year> 1999 </year>
  </book>
</book_list>
```
Sample XSL Style Sheet

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2">
          <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="book_list/book">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="author"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
XHTML - eXtended HTML

• XHTML is the new Web standard and should be used for maximum portability.

• XHTML is essentially HTML 4 reformulated in XML. (To get the HTML effect, a definition is needed in the XSL file.

• There are six major differences between XHTML and HTML 4:

  1. XHTML pages and browsers must strictly conform to the standard.

  2. All tags and attributes must be in lower case.

  3. Closing tags are required.

  4. Attributes must be contained within quotation marks.

  5. Tags must nest properly.

Type of Web Documents

- **Static**
  A static Web document resides in a file at a Web server. The contents do not change. Each request for the document results in exactly the same response.

- **Dynamic**
  A dynamic Web document does not exist in a predefined form. It is created by a Web server whenever a browser asks for the document. The contents of a dynamic document can vary from one request to another.

- **Active**
  An active document is not fully specified by the server. An active document consists of a computer program that understands how to compute and display values. The browser runs the program locally. Thus, the contents of an active document can change continuously.
The Common Gateway Interface (CGI)

- The CGI standard specifies how a server interacts with an application program that implements a dynamic document. The application is called a CGI program.

- CGI programs can generate arbitrary document types (e.g., plain text, HTML text, digital image, etc.). A CGI program can place a header on its output to describe the document type.

- Sample CGI Program:

```
#!/bin/sh

# CGI script that prints the date and time at which it was run
#
# Output the document header followed by a blank line

echo Content/type: text/plain
echo

# Output the date

echo This document was created on `date`

An example CGI program written in UNIX shell language.
```

- Program Output:
  Content/type: text/plain

This document was created on Wed Dec 12 14:19:37 EST 1996
CGI Parameters and Environment Variables

- CGI allows a server to pass arguments to a CGI program whenever the program is invoked.

<table>
<thead>
<tr>
<th>Name of Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER_NAME</td>
<td>The domain name of the computer running the server.</td>
</tr>
<tr>
<td>GATEWAY_INTERFACE</td>
<td>The version of the CGI software the server is using.</td>
</tr>
<tr>
<td>SCRIPT_NAME</td>
<td>The path in the URL after the server name.</td>
</tr>
<tr>
<td>QUERY_STRING</td>
<td>Information following “?” in the URL.</td>
</tr>
<tr>
<td>REMOTE_ADDR</td>
<td>The IP address of the computer running the browser that sent the request.</td>
</tr>
</tbody>
</table>

Examples of environment variables passed to a CGI program.
A CGI Script with Long-Term State Information

```
#!/bin/sh
FILE=ipaddr

echo Content-type: text/plain
echo

# See if IP address of browser's computer appears in our file
if grep -s $REMOTE_ADDR $FILE >/dev/null 2>&1 then
    echo Computer $REMOTE_ADDR has requested this URL previously.
else
    # Append browser's address to the file
    echo $REMOTE_ADDR >> $FILE
    echo This is the first contact from computer $REMOTE_ADDR
fi
```

An example CGI program that stores long-term state information.

- **First Access:**
  This is the first contact from computer 128.10.2.36

- **Subsequent Accesses:**
  Computer 128.10.2.26 has requested this URL previously.
A CGI Script with Short-Term State Information

```bash
#!/bin/sh

echo Content-type: text/html

N=$QUERY_STRING
echo "<HTML>"

case "x$N" in
  x)
    N=1
    echo "This is the initial page.<BR><BR>"
    ;;
  x[0-9]*) N=`expr $N + 1`
    echo "You have refreshed this page $N times.<BR><BR>"
    ;;
  *)
    echo "The URL you used is invalid.</HTML>"
    exit 0
    ;;
esac

echo "<A HREF="http://$SERVER_NAME$SCRIPT_NAME?$N">"
echo "Click here to refresh the page.\</A> </HTML>"
```

An example CGI program that maintains short-term state.
• Program Output:
  Content-type: text/html

  <HTML>
  This is the initial page. <BR> <BR>
  <A HREF = "http://www.nonexist.com/cgi/ex4?1" >
  Click here to refresh the page. </A> </HTML>

• Browser Interpretation:
  This is the initial page.
  Click here to refresh the page.

• Program Output After First Invocation:
  Content-type: text/html

  <HTML>
  You have refreshed this page 1 times. <BR> <BR>
  <A HREF = "http://www.nonexist.com/cgi/ex4?2" >
  Click here to refresh the page. </A> </HTML>

• Browser Interpretation:
  You have refreshed this page 1 times.
  Click here to refresh the page.
PHP: Hypertext Preprocessor

- PHP embeds small scripts inside HTML pages and has them executed by the server to generate the page.

Example: Web page with form - PHP script + Output

```
<html>
<body>
<form action="action.php" method="post">
<p>Please enter your name: <input type="text" name="name"></p>
<p>Please enter your age: <input type="text" name="age"></p>
<input type="submit">
</form>
</body>
</html>

(a)

<html>
<body>
<h1>Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>

(b)

<html>
<body>
<h1>Reply: </h1>
Hello Barbara.
Prediction: next year you will be 25
</body>
</html>

(c)
```
Active Documentation Representation and Translation

Diagram:

1. **active document in source form** → **compiler** → **active document in binary form**
2. **request for document** from **server** to **browser**
3. **active document in executable form** in **browser’s memory**

Illustration of three active document representations and the programs that translate or transport the document. The darkened arrow shows the direction a document moves.
Java Technology

• Java is the name of a specific technology used to run active documents.

• Java uses the applet to describe active document programs.

• Key Components:
  - Programming Language:
    Java includes a new programming language.
  - Runtime Environment:
    Provides the facilities needed to run a Java program.
  - Class Library:
    A library of software to perform many of the chores that an applet performs.

• A browser that runs Java contains an interpreter to run java applets.

• Invoking the Applet:
  <applet codebase="www.nonexist.edu/example" code=bbb.class>
An Example Java Applet

```java
import java.applet.*;
import java.awt.*;

public class clickcount extends Applet {
    int count;
    TextField f;

    public void init() {
        count = 0;
        add(new Button("Click Here"));
        f = new TextField("The button has not been clicked at all.");
        f.setEditable(false);
        add(f);
    }

    public boolean action(Event e, Object arg) {
        if (((Button) e.target).getLabel() == "Click Here") {
            count += 1;
            f.setText("The button has been clicked " + count + " times.");
        }
        return true;
    }

    An example applet that counts the number of times a user clicks a button.
```
An Example Java Applet cont.

• Initial Applet Output:

Click Here | The button has not been clicked at all.

Illustration of the display after the applet begins execution.

• Subsequent Applet Output:

Click Here | The button has been clicked 1 times.

The display after the user has clicked the button once.
Applet Interaction with a Browser

```java
import java.applet.*;
import java.net.*;
import java.awt.*;

public class buttons extends Applet {

    public void init() {
        add(new Button("Ying"));
        add(new Button("Yang"));
    }

    public boolean action(Event e, Object arg) {
        if (((Button) e.target).getLabel() == "Ying") {
            try {
                getAppletContext().showDocument(new URL("http://www.nonexist.com/ying");
            }
            catch( Exception ex ) {
                // note: code to handle the exception goes here //
            }
        }
        else if (((Button) e.target).getLabel() == "Yang") {
            try {
                getAppletContext().showDocument(new URL("http://www.other.com/yang")
            }
            catch( Exception ex ) {
                // note: code to handle the exception goes here //
            }
        }
        return true;
    }
}

An example applet that interacts with the HTTP and HTML facilities in a browser.

Applet Output
```
Illustration of the display produced by the applet
Client-Side Scripting: JavaScript

- HTML 4.0 allows the embedding of scripts to be executed on the client machine.

- The most popular client-side scripting language is JavaScript.

- Example: JavaScript for Form Processing:

```html
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   var person = test_form.name.value;
   var years = eval(test_form.age.value) + 1;
   document.open();
   document.writeln("<html> <body> ");
   document.writeln("Hello " + person + ".<br>");
   document.writeln("Prediction: next year you will be " + years + ".");
   document.writeln("</body> </html> ");
   document.close();
}
</script>
</head>
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)"/>
</form>
</body>
</html>
```
Example: Page that Responds to Mouse Movements

```html
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
function pop(m) {
    var urx = "http://www.cs.vu.nl/ast/im/cat.jpg";
    popupwin = window.open(document.myurl[m],"mywind","width=250,height=250");
}
</script>
</head>
<body>
<p><a href="#" onmouseover="pop(0); return false;" > Kitten </a></p>
<p><a href="#" onmouseover="pop(1); return false;" > Puppy </a></p>
<p><a href="#" onmouseover="pop(2); return false;" > Bunny </a></p>
</body>
</html>
```
JavaScript

JavaScript is a high-level language that is interpreted - translated into machine language at the time of usage (run-time)

JavaScript allows for dynamic, real-time changes to the web page the user is accessing. The user causes an “event” (e.g., moving the mouse to press a button) and the programmer can use JavaScript to program a response.

JavaScript was developed by Netscape as a web programming language.

Characteristics of the language:
- allows interactive content on a Web page.
- client based (works on the user machine).
- does not manipulate files.
- does not carry out graphics.

Important Issues for JavaScript:
- the instructions are written in lowercase.
- all instructions must be spelled correctly or the interpreter will not understand them.
- parts of an instruction must be separated by a space and not run together.
- the correct punctuation must be used.
Overview of JavaScript

JavaScript alert("string") function:
The alert function requests that the browser pop-up a small window that contains the words in the string.

Inserting JavaScript into a Web page:
<script language="JavaScript" type="text/javascript">
  alert("your message goes here");
</script>

JavaScript prompt("string") function:
<script language="JavaScript" type="text/javascript">
  var mytext = prompt("Please enter some text");
  alert(mytext);
</script>

Static write to a window:
<script language="JavaScript" type="text/javascript">
  window.document.write("This is a statement");
</script>
alternate version:
<script language="JavaScript" type="text/javascript">
  document.write("This is a statement");
  document.bgColor = "yellow";
  document.fgColor = "blue";
</script>

JavaScript confirm() method:
var reply = confirm("Do you like this color?");
document.write("Your answer was " + reply);
Mouse Events:

OnMouseOver Event Handler:
- The user moves the mouse over a particular part of the Web page.
- First, the programmer has to define the part of the Web page to be monitored.
- If the user has moved the mouse there, the program will detect this and react in some way.

```html
<a href = "#"
   onMouseOver = "document.bgColor = 'red'; return true"
>
   Watch me!
</a>
```

Note: two levels of quotes are needed.

OnMouseOut Event Handler:
- The user moves the mouse away from the referenced part of the Web page.
- The program will detect this and react.

```html
<a href = "#"
   onMouseOver = "document.bgColor = 'red'; return true"
   onMouseOut = "document.bgColor = 'white'; return true"
>
   Watch me!
</a>
```

Button Events:

user clicks on a button - invokes an event handler.
Summary of Ways to Generate and Display Content