WEB TECHNOLOGY (SCSX5010)

- UNIT I → Web Essentials
- UNIT II → Client Side Programming
- UNIT III → Java Servlet and JSP
- UNIT IV → ASP Programming
- UNIT V → Related Technologies

REFERENCE BOOKS:

1. Jeffrey C Jackson, “Web Technology – A computer Science perspective”, Person Education, 2007.
2. Chris Bates, “Web Programming – Building Internet Applications”, Wiley India, 2006.
3. Deitel & Deitel “Internet and World Wide Web How to Program”, Third Edition.
4. Gopalan. N.P, “Web Technology A Developer Perspectives”, PHI, 2009.

Unit 1

WEB ESSENTIALS

- Internet
- Web Clients – Servers Communication
- XHTML 1.0
- Cascading Style Sheets [CSS]: Features – Style Rule – Style Properties – Box Model Techniques

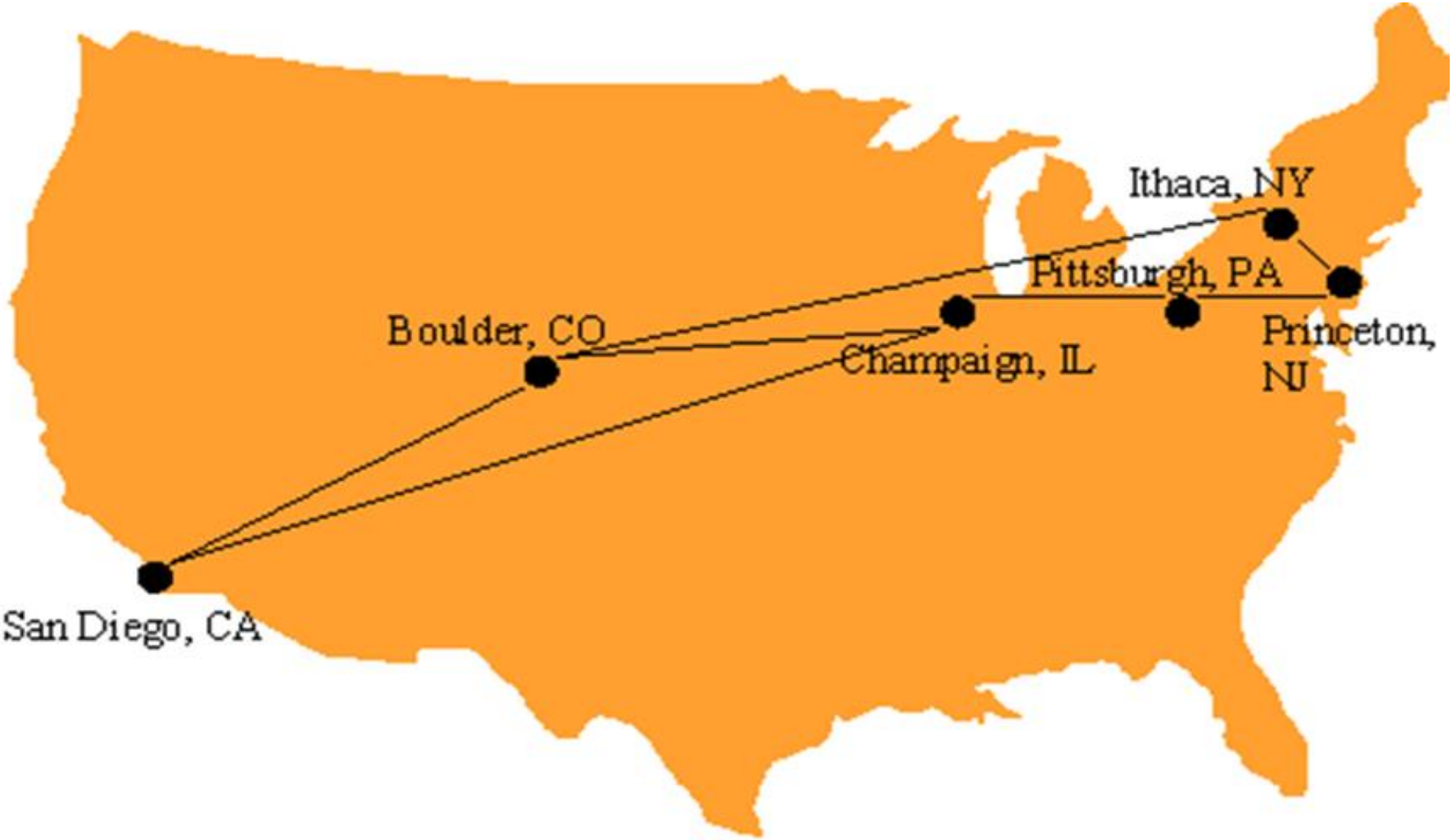
The Internet

- Technical origin: **ARPANET** (late 1960's)
  - One of earliest attempts to network heterogeneous, geographically dispersed computers
  - Email first available on ARPANET in 1972 (and quickly very popular!)
- ARPANET **access was limited** to select DoD - funded organizations

The Internet

- Open-access networks
  - Regional university networks (e.g., SURAnet)
  - CSNET for CS departments not on ARPANET
- NSFNET (1985 - 1995)
  - Primary purpose: connect supercomputer centers
  - Secondary purpose: provide **backbone** to connect regional networks

The Internet



The 6 supercomputer centers connected by the early NSFNET backbone

The Internet

- Original NSFNET backbone speed: 56 K Bit/s
- Upgraded to 1.5 M Bit/s (T1) in 1988
- Upgraded to 45 M Bit/s (T3) in 1991
- In 1988, networks in Canada and France connected to NSFNET
- In 1990, ARPANET is decommissioned, NSFNET the center of the internet

The Internet

- **Internet:** the network of networks connected via the public backbone and communicating using TCP/IP communication protocol
  - Backbone initially supplied by NSFNET, privately funded (ISP fees) beginning in 1995

Internet Protocols

- Communication protocol: how computers talk
  - Telephone "protocol": how you answer and end call, what language you speak, etc.
- Internet protocols developed as part of ARPANET research
  - ARPANET began using TCP/IP in 1982
- Designed for use both within Local Area Networks (LAN's) and between networks

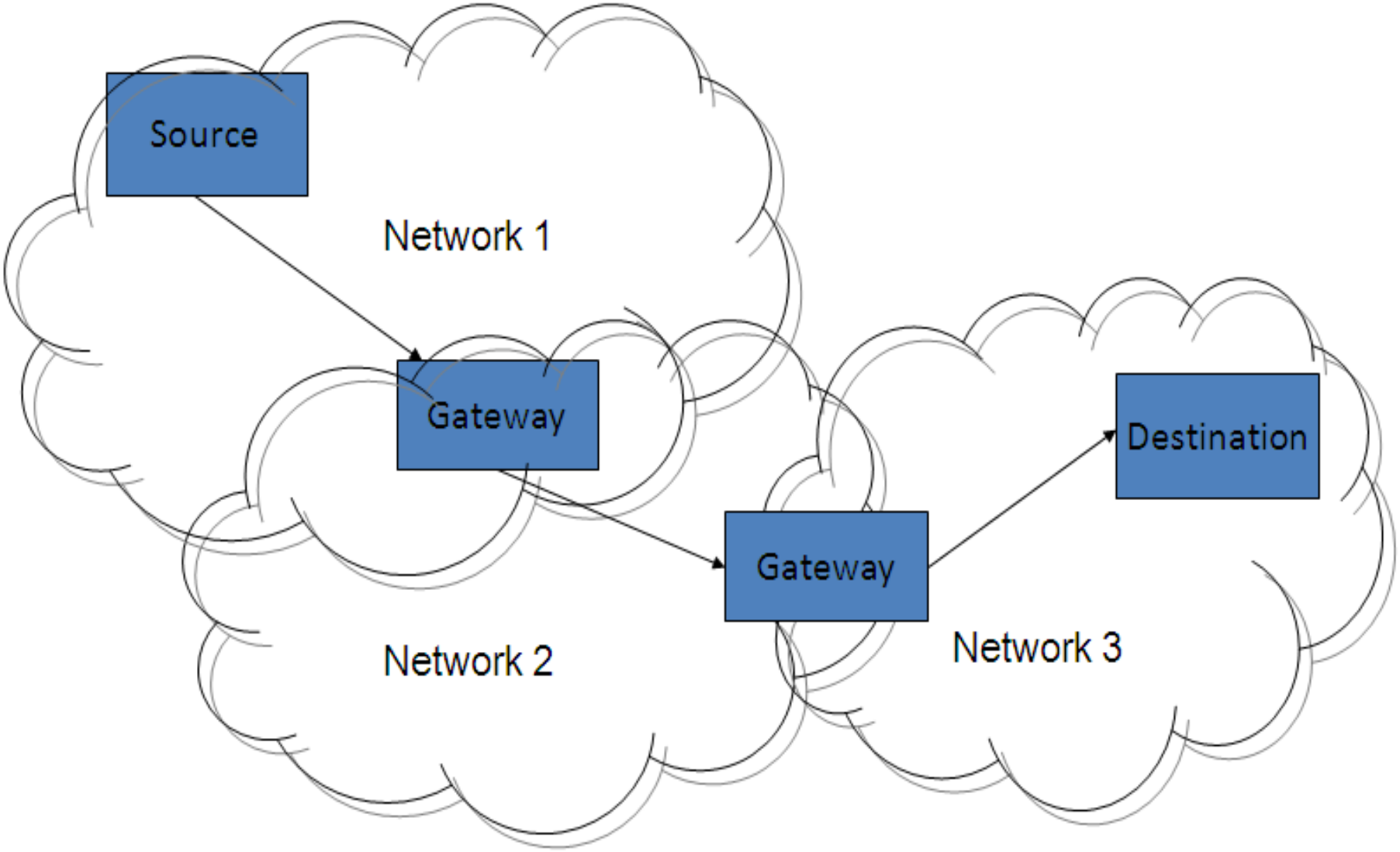
Internet Protocol (IP)

- IP is the fundamental protocol defining the Internet (as the name implies!)
- IP address:
  - 32-bit number (in IPv4)
  - Associated with at most one device at a time (although device may have more than one)
  - Written as four dot-separated bytes, e.g. 192.0.34.166

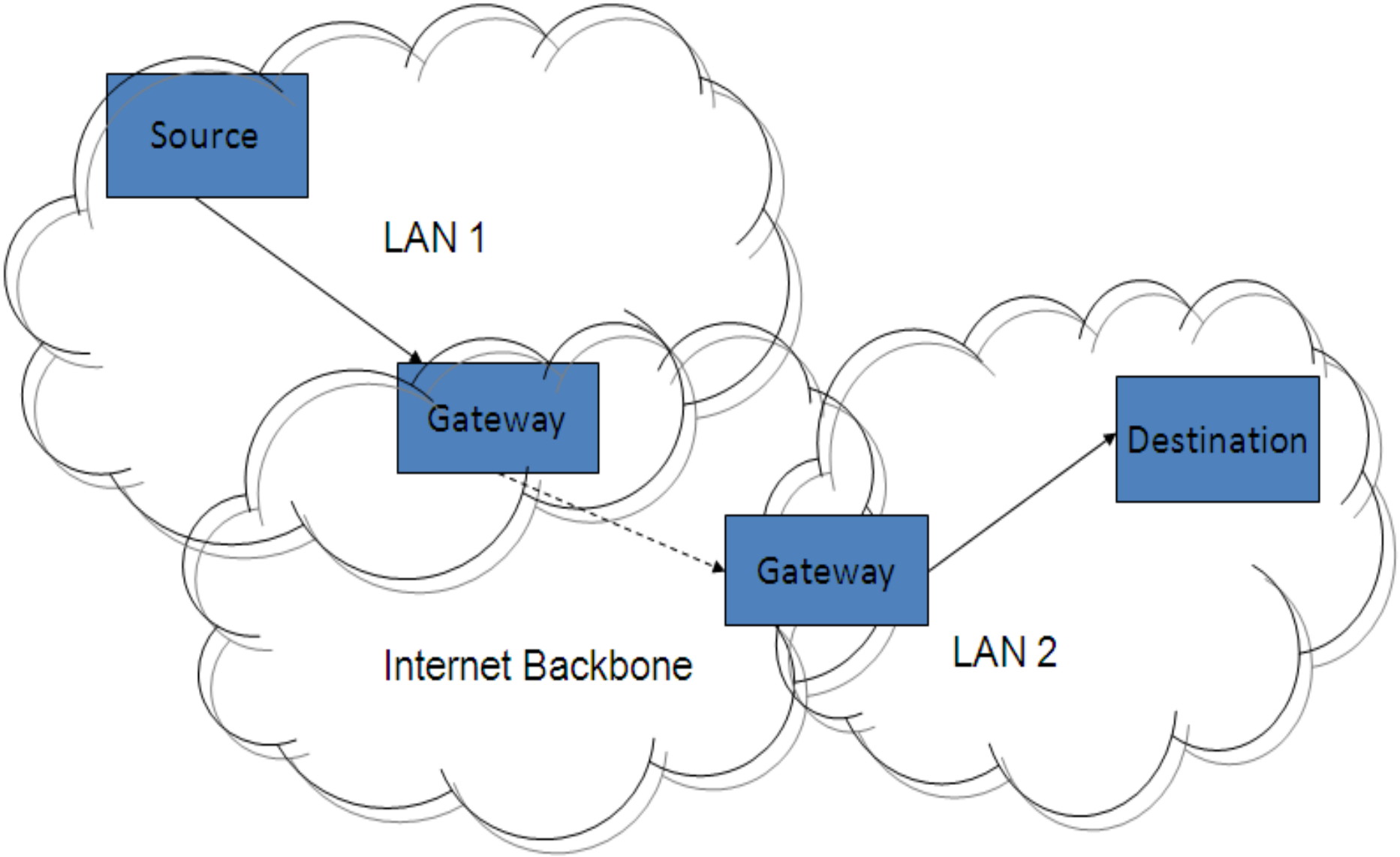
Internet Protocol (IP)

- IP function: transfer data from source device to destination device
- IP source software creates a packet representing the data
  - Header: source and destination IP addresses, length of data, etc.
  - Data itself
- If destination is on another LAN, packet is sent to a gateway that connects to more than one network

Internet Protocol (IP)



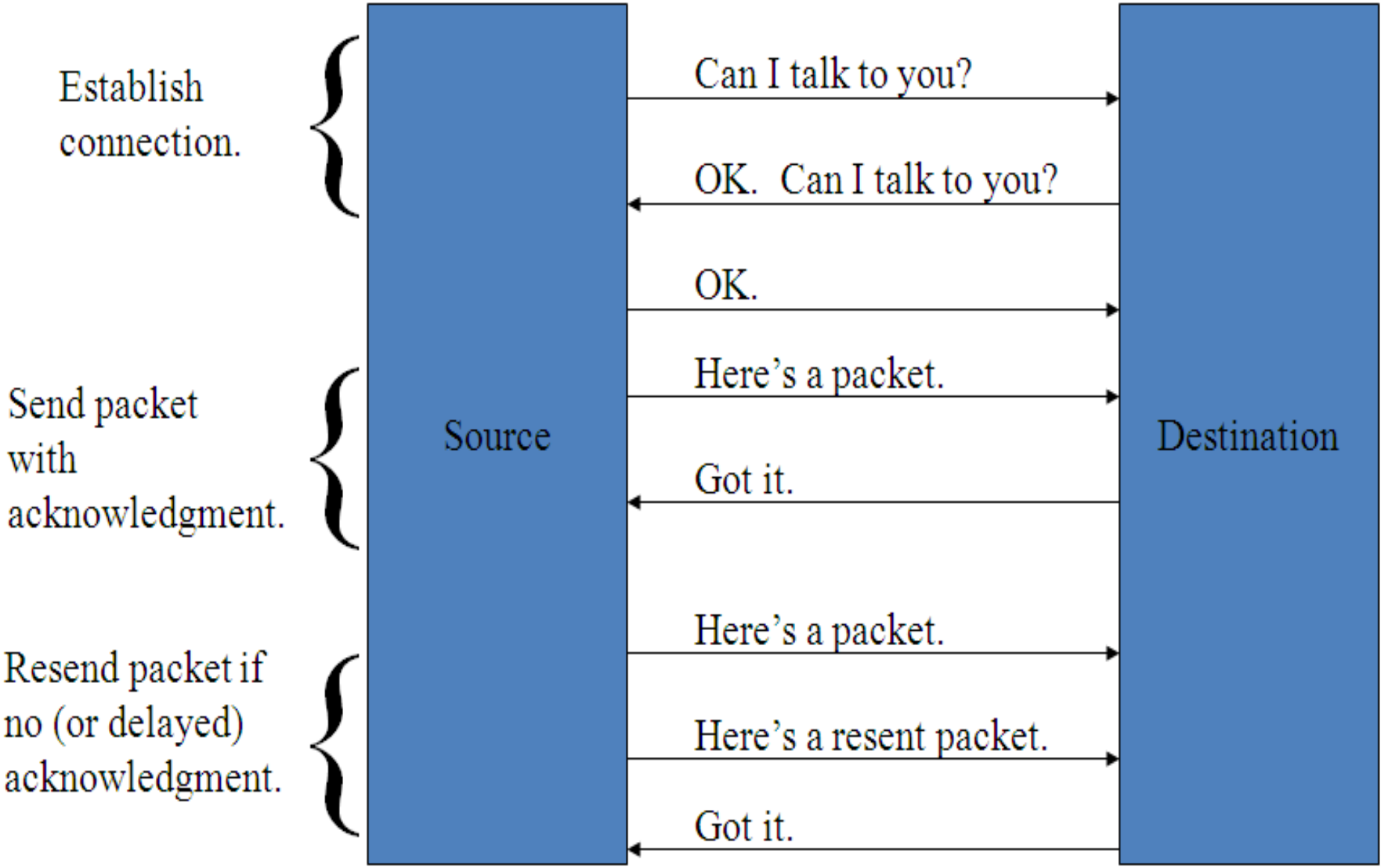
Internet Protocol (IP)



Transmission Control Protocol (TCP)

- Limitations of IP:
  - No guarantee of packet delivery (packets can be dropped)
  - Communication is one-way (source to destination)
- TCP adds concept of a connection on top of IP
  - Provides guarantee that packets delivered
  - Provide two-way (full duplex) communication

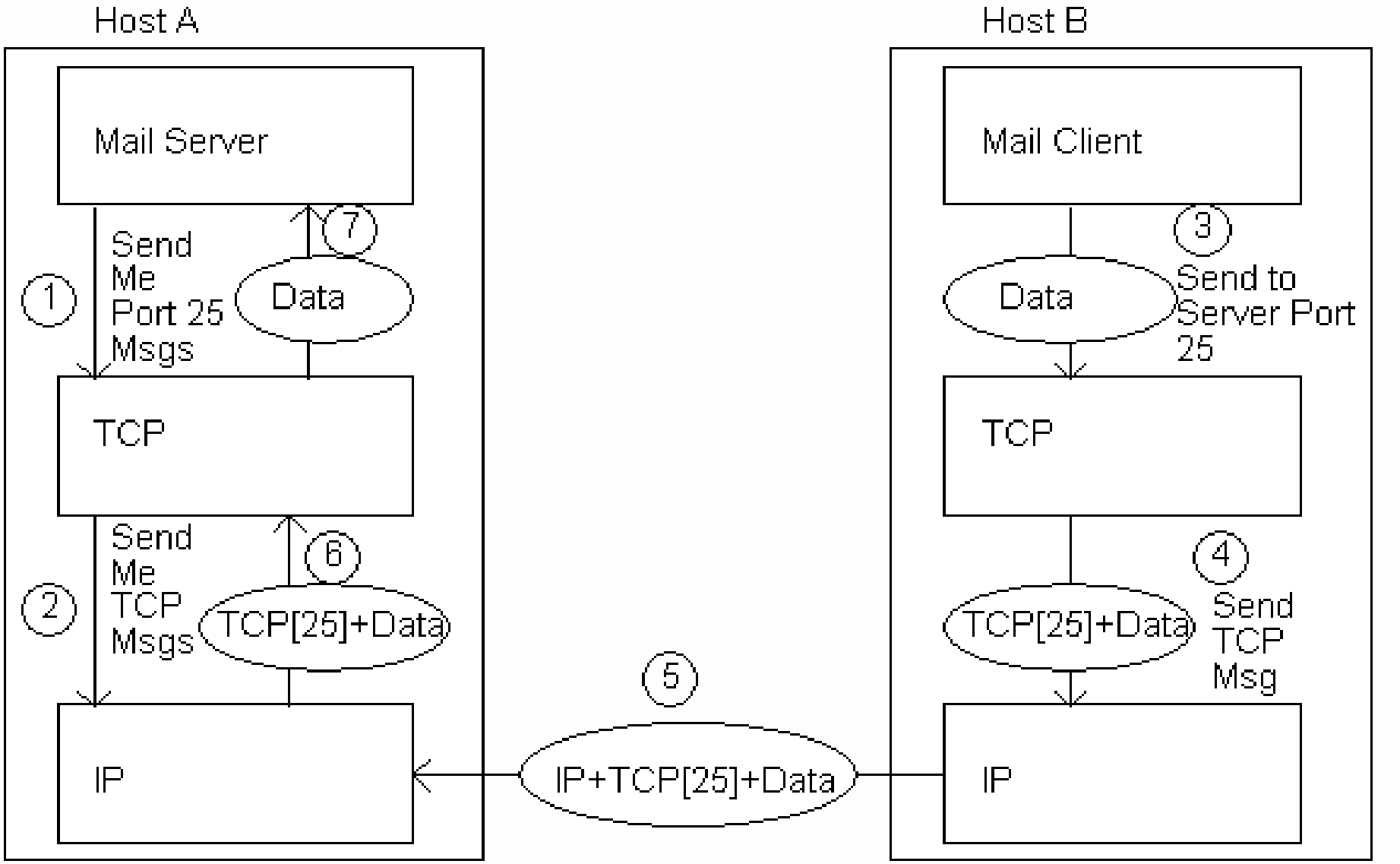
TCP


Establish connection.
Send packet with acknowledgment.
Resend packet if no (or delayed) acknowledgment.
Source
Destination
Can I talk to you?
OK. Can I talk to you?
OK.
Here's a packet.
Got it.
Here's a packet.
Here's a resent packet.
Got it.

TCP

- TCP also adds concept of a port
  - TCP header contains port number representing an application program on the destination computer
  - Some port numbers have standard meanings
    - Example: port 25 is normally used for email transmitted using the Simple Mail Transfer Protocol (SMTP)
  - Other port numbers are available first-come-first served to any application

TCP


Host A
Host B
Mail Server
Mail Client
1
Send Me Port 25 Msgs
7
Data
3
Send to Server Port 25
Data
TCP
TCP
2
Send Me TCP Msgs
6
TCP[25]+Data
4
Send TCP Msg
TCP[25]+Data
5
IP
IP+TCP[25]+Data
IP

User Datagram Protocol (UDP)

- Like TCP in that:
  - Builds on IP
  - Provides port concept
- Unlike TCP in that:
  - No connection concept
  - No transmission guarantee
- Advantage of UDP vs. TCP:
  - Lightweight, so faster for one-time messages

Domain Name Service (DNS)

- DNS is the “phone book” for the Internet
  - Map between host names and IP addresses
  - DNS often uses UDP for communication
- Host names
  - Labels separated by dots, e.g., www.example.org
  - Final label is *top-level domain*
    - Generic: .com, .org, etc.
    - Country-code: .us, .in, etc.

DNS

- Domains are divided into second-level domains, which can be further divided into subdomains, etc.
  - E.g., www.example.com, example is a second-level domain
- A host name plus domain name information is called the Fully Qualified Domain Name of the computer
  - Above, www is the host name, www.example.com is the FQDN

DNS

- nslookup program provides command-line access to DNS (on most systems)
- looking up a host name given an IP address is known as a reverse lookup
  - Recall that single host may have multiple IP addresses.
  - Address returned is the canonical IP address specified in the DNS system.

DNS

- ipconfig (on windows) can be used to find the IP address (addresses) of your machine
- *ipconfig / displaydns* displays the contents of the DNS Resolver Cache (*ipconfig / flushdns* to flush it)

Analogy to Telephone Network

- IP ~ the telephone network
- TCP ~ calling someone who answers, having a conversation, and hanging up
- UDP ~ calling someone and leaving a message
- DNS ~ directory assistance

Higher level Protocols

- Many protocols build on TCP
  - Telephone analogy: TCP specifies how we initiate and terminate the phone call, but some other protocol specifies how we carry on the actual conversation
- Some examples:
  - SMTP (email) (25)
  - FTP (file transfer) (21)
  - HTTP (transfer of Web documents) (80)

World Wide Web

- Originally, one of several systems for organizing Internet-based information
  - Competitors: WAIS, Gopher, ARCHIE
- Distinctive feature of Web: support for hypertext (text containing links)
  - Communication via Hypertext Transport Protocol (HTTP)
  - Document representation using Hypertext Markup Language (HTML)

World Wide Web

- The Web is the collection of machines (Web servers) on the Internet that provide information, particularly HTML documents, via HTTP.
- Machines that access information on the Web are known as Web clients.
- A Web browser is software used by an end user to access the Web.

Hypertext Transport Protocol (HTTP)

- HTTP is based on the request-response communication model:
  - Client sends a request
  - Server sends a response
- HTTP is a stateless protocol:
  - The protocol does not require the server to remember anything about the client between requests.

HTTP

- Normally implemented over a TCP connection (80 is standard port number for HTTP)
- Typical browser-server interaction:
  - User enters Web address in browser
  - Browser uses DNS to locate IP address
  - Browser opens TCP connection to server
  - Browser sends HTTP request over connection
  - Server sends HTTP response to browser over connection
  - Browser displays body of response in the client area of the browser window

HTTP

- The information transmitted using HTTP is often entirely text
- Can use the Internet's Telnet protocol to simulate browser request and view server response

HTTP

Connect

```
$ telnet www.example.org 80
Trying 192.0.34.166...
Connected to www.example.com
(192.0.34.166).
Escape character is ' ^]' .
```

Send Request

```
GET / HTTP/1.1
Host: www.example.org
```

Receive Response

```
HTTP/1.1 200 OK
Date: Thu, 09 Oct 2003 20:30:49 GMT
…
```

HTTP Request

- Structure of the request:
  - start line
  - header field(s)
  - blank line
  - optional body

HTTP Request (Start Line)

- Start line
  - Example: GET / HTTP/1.1
- Three space-separated parts:
  - HTTP request method
  - Request-URI (Uniform Resource Identifier)
  - HTTP version

HTTP Request (Start Line)

- Uniform Resource Identifier (URI)
  - Syntax: *scheme* : *scheme-depend-part*
    - Ex: In http://www.example.com/ the scheme is http
  - Request-URI is the portion of the requested URI that follows the host name (which is supplied by the required Host header field)
    - Ex: / is Request-URI portion of http://www.example.com/

URI

- URI's are of two types:
  - Uniform Resource Name (URN)
    - Can be used to identify resources with unique names, such as books (which have unique ISBN's)
    - Scheme is urn
  - Uniform Resource Locator (URL)
    - Specifies location at which a resource can be found
    - In addition to HTTP, some other URL schemes are HTTPs, FTP, MAILTO, and FILE

HTTP Request Method

- Common request methods:
  - GET
    - Used if link is clicked or address typed in browser
    - No body in request with GET method
  - POST
    - Used when submit button is clicked on a form
    - Form information contained in body of request
  - HEAD
    - Requests that only header fields (no body) be returned in the response

HTTP Request (Header Field)

- Header field structure:
  - *field name* : *field value*
- Syntax
  - Field name is not case sensitive
  - Field value may continue on multiple lines by starting continuation lines with white space
  - Field values may contain MIME types, quality values, and wildcard characters (*'s)

Multipurpose Internet Mail Extensions (MIME)

- Convention for specifying content type of a message
  - In HTTP, typically used to specify content type of the body of the response
- MIME content type syntax:
  - *top-level type* / *subtype*
- Examples: text / html, image / jpeg

HTTP Quality Values and Wildcards

- Example header field with quality values: accept:
  - text / xml, text / html; q=0.9,
  - text / plain; q=0.8,
  - image / jpeg, image / gif; q=0.2,
  - */*; q=0.1
- Quality value applies to all preceding items
- Higher the value, higher the preference
- Note use of wildcards to specify quality 0.1 for any MIME type not specified earlier.

HTTP Request

- Common header fields:
  - Host: host name from URL (required)
  - User-Agent: type of browser sending request
  - Accept: MIME types of acceptable documents
  - Connection: value close tells server to close connection after single request/response
  - Content-Type: MIME type of (POST) body, normally application/x-www-form-urlencoded
  - Content-Length: bytes in body
  - Referer: URL of document containing link that supplied URI for this HTTP request

HTTP Response

- Structure of the response:
  - status line
  - header field(s)
  - blank line
  - optional body

HTTP Response (Status Line)

- Status line
  - Example: HTTP/1.1 200 OK
- Three space-separated parts:
  - HTTP version
  - status code
  - reason phrase (intended for human use)

HTTP Response (Status Code)

- Status code
  - Three-digit number
  - First digit is class of the status code:
    - 1=Informational
    - 2=Success
    - 3=Redirection (alternate URL is supplied)
    - 4=Client Error
    - 5=Server Error
  - Other two digits provide additional information
  - See http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

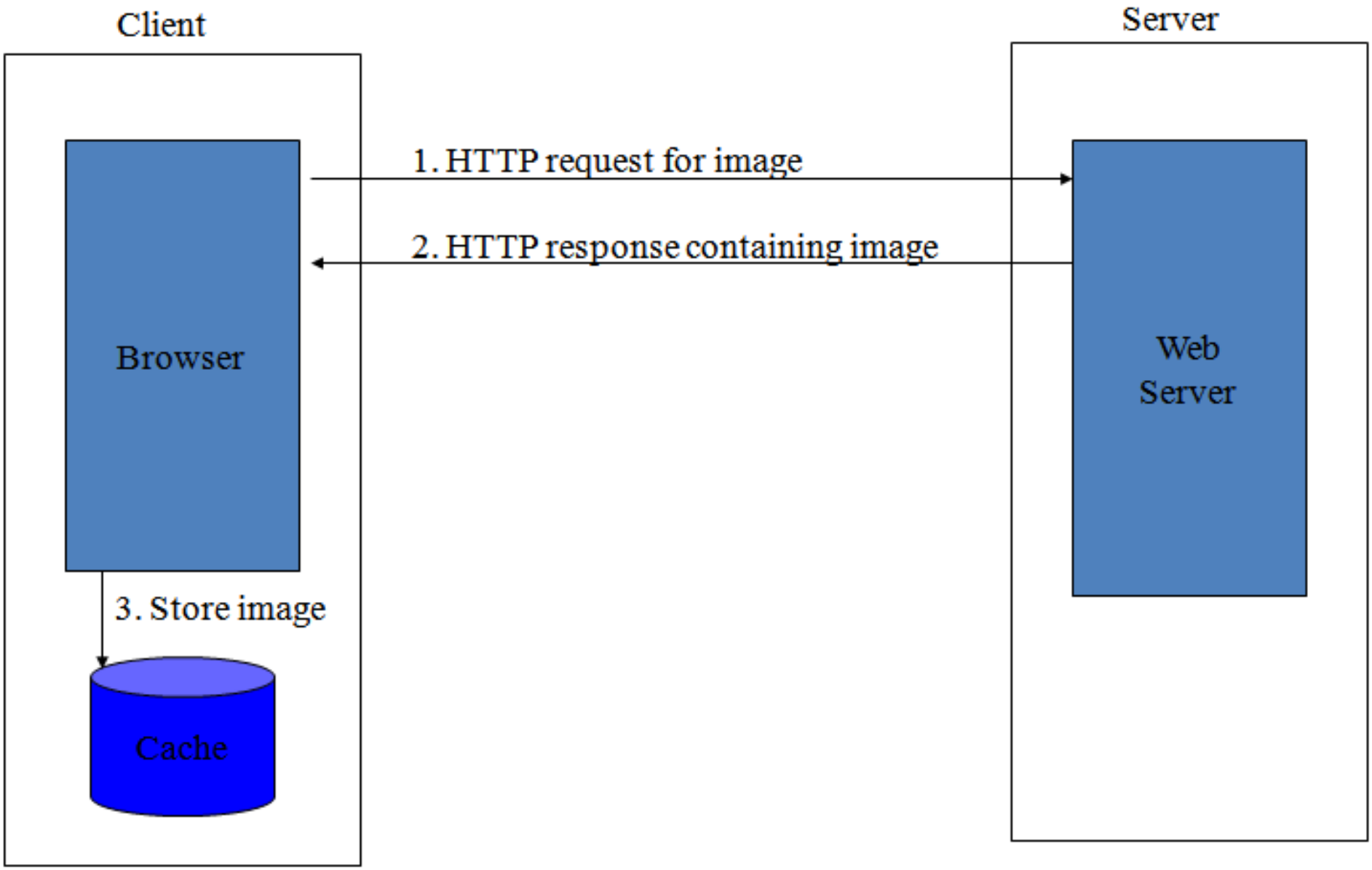
HTTP Response (Header Fields)

- Common header fields:
  - Connection, Content-Type, Content-Length
  - Date: date and time at which response was generated (required)
  - Location: alternate URI if status is redirection
  - Last-Modified: date and time the requested resource was last modified on the server
  - Expires: date and time after which the client's copy of the resource will be out-of-date
  - ETag: a unique identifier for this version of the requested resource (changes if resource changes)

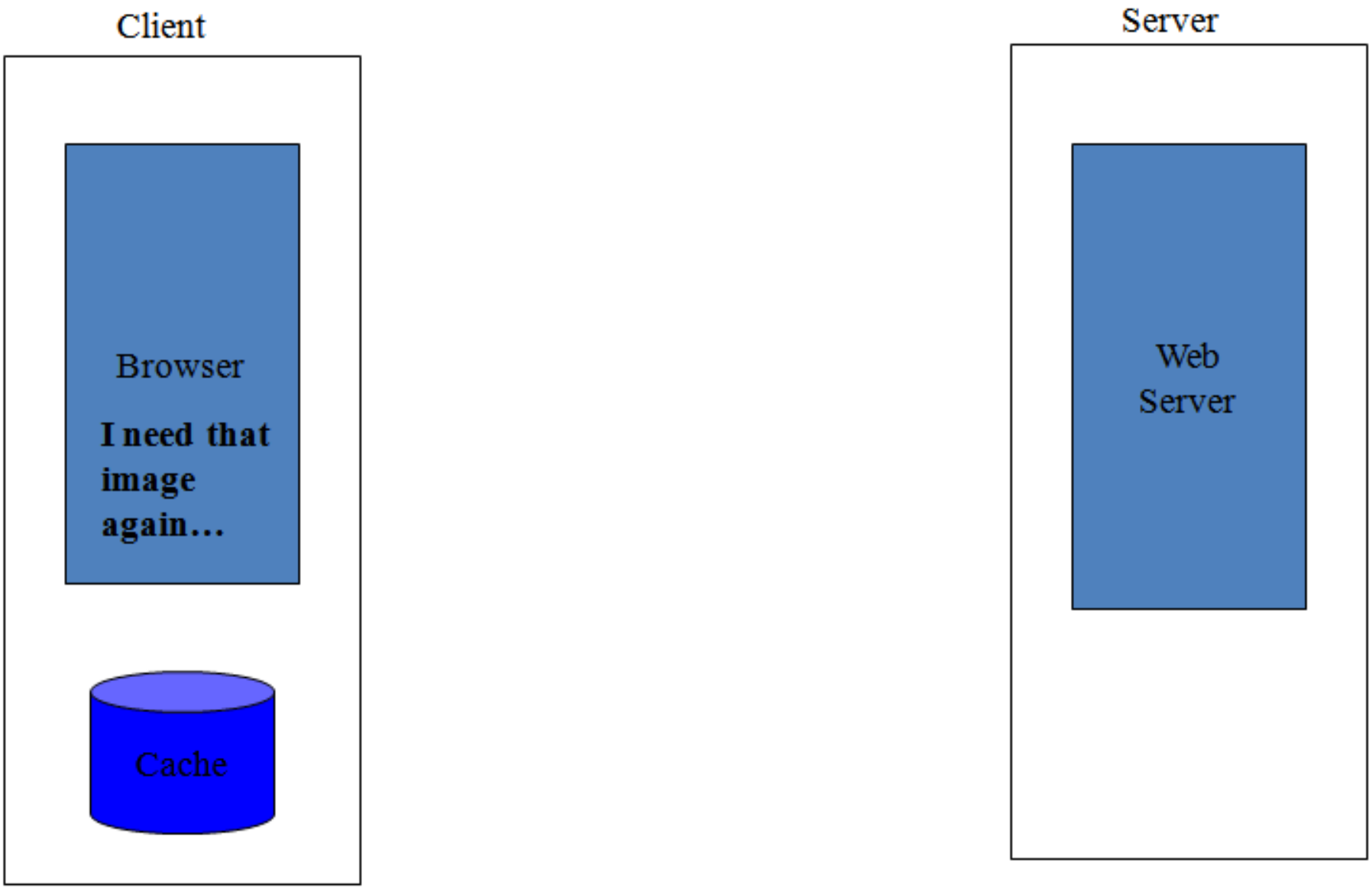
Client Caching

- A cache is a local copy of information obtained from some other source
- Most web browsers use cache to store requested resources so that subsequent requests to the same resource will not necessarily require an HTTP request/response
  - Ex: icon appearing multiple times in a Web page

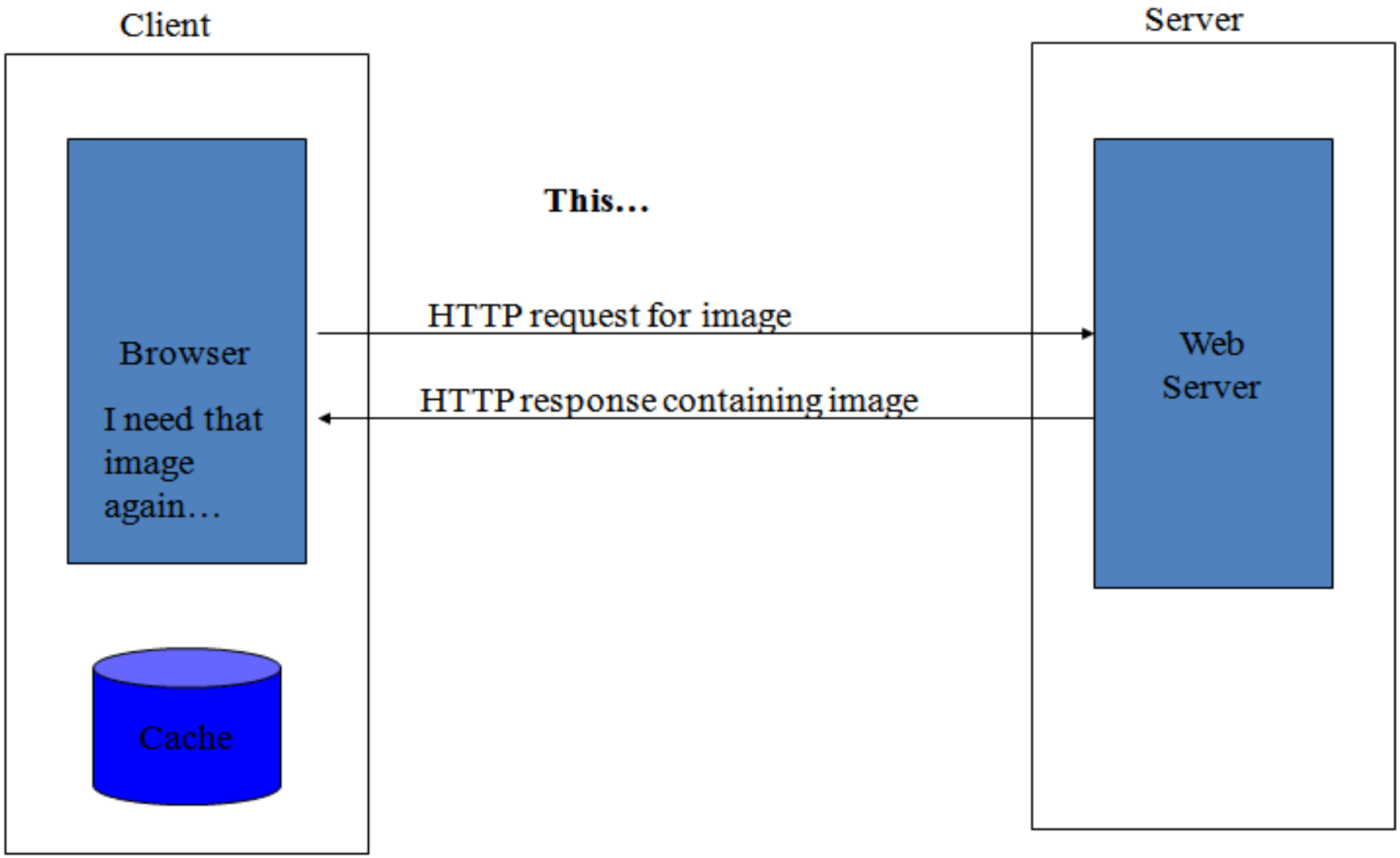
Client Caching



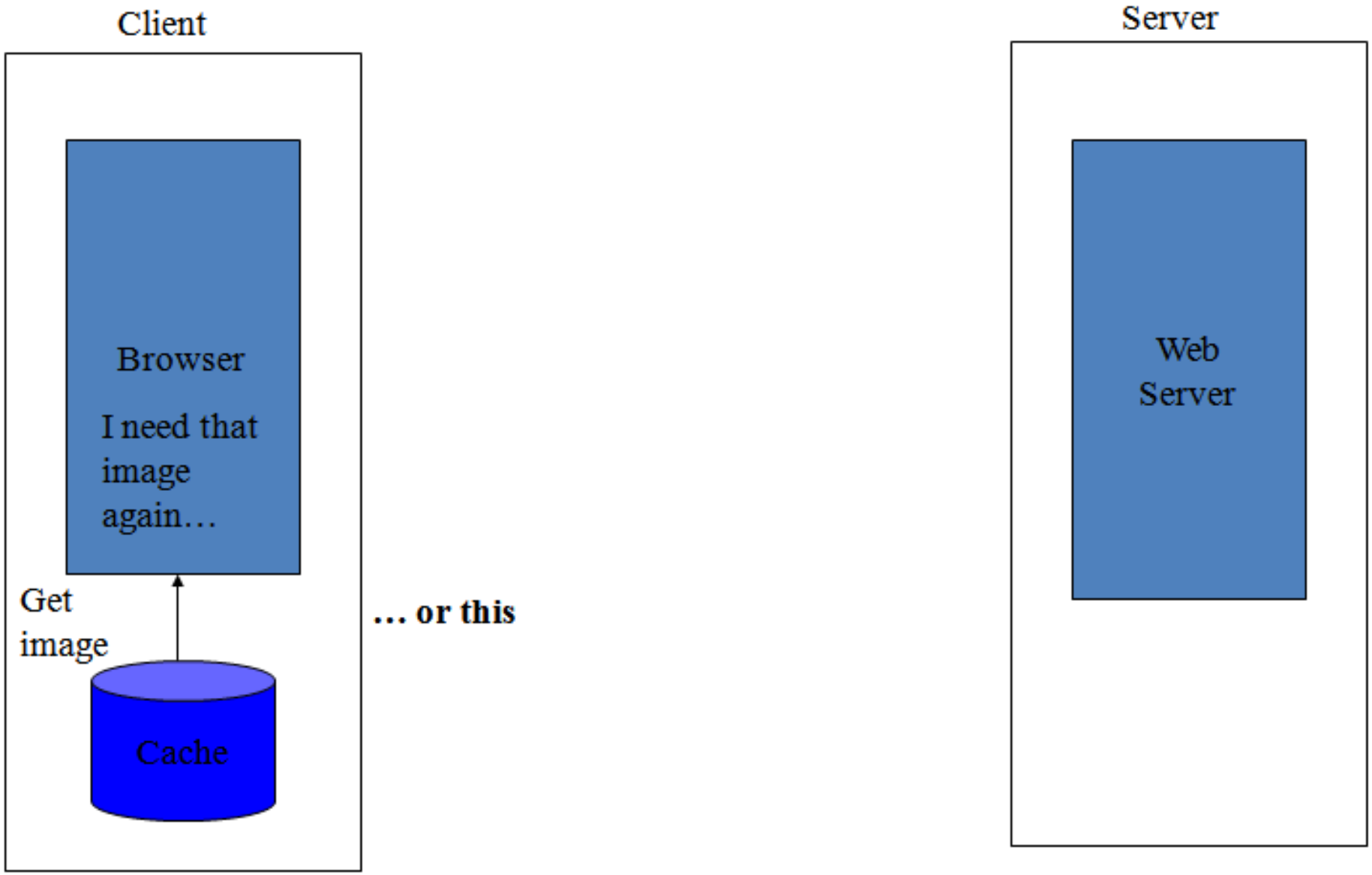
Client Caching



Client Caching



Client Caching



Client Caching

- Cache advantages
  - (Much) faster than HTTP request/response
  - Less network traffic
  - Less load on server
- Cache disadvantage
  - Cached copy of resource may be invalid (inconsistent with remote version)

Client Caching

- Validating cached resource:
  - Send HTTP HEAD request and check Last-Modified or ETag header in response
  - Compare current date/time with Expires header sent in response containing resource
  - If no Expires header was sent, use heuristic algorithm to estimate value for Expires
    - Ex: Expires = 0.01 * (Date – Last-Modified) + Date

Character Sets

- Every document is represented by a string of integer values (**code points**)
- The mapping from code points to characters is defined by a **character set**
- Some header fields have character set values:
  - **Accept-Charset:** request header listing character sets that the client can recognize, **Ex:** accept-charset: ISO-8859-1, utf-8; q=0.7,*; q=0.5
  - **Content-Type:** can include character set used to represent the body of the HTTP message, **Ex:** Content-Type: text/html; charset=UTF-8

Character Sets

- Technically, many “character sets” are actually **character encodings**
  - An encoding represents code points using **variable-length** byte strings
  - Most common examples are Unicode-based encodings UTF-8 and UTF-16
- IANA maintains **complete list** of Internet-recognized character sets / encodings

Character Sets

- Typical US PC produces ASCII documents
- **US-ASCII** character set can be used for such documents, but is not recommended
- UTF-8 and ISO-8859-1 are supersets of US-ASCII and provide international compatibility
  - **UTF-8** can represent all ASCII characters using a single byte each and arbitrary Unicode characters using up to 4 bytes each
  - **ISO-8859-1** is 1-byte code that has many characters common in Western European languages, such as é

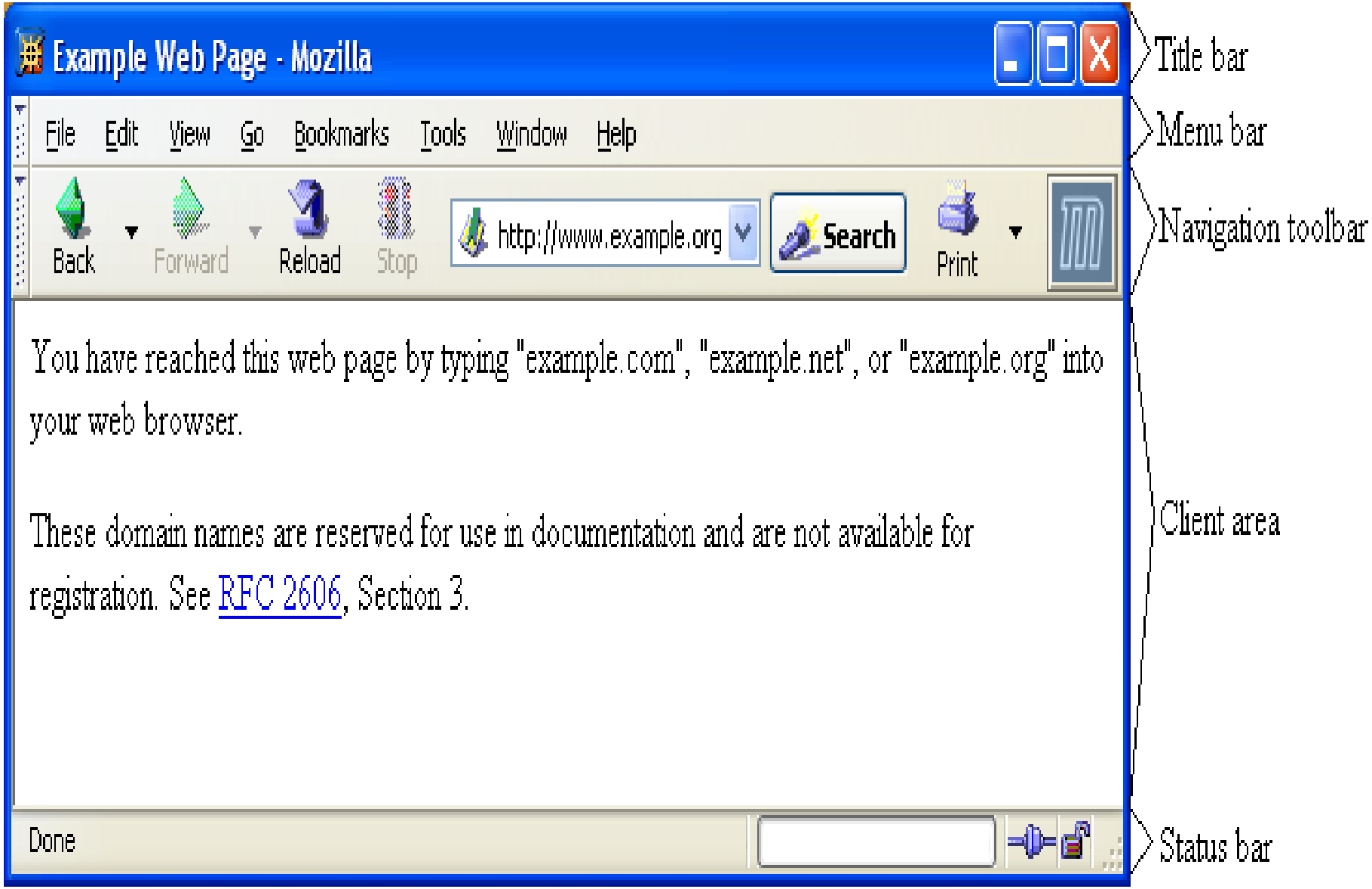
Web Clients

- Many possible web clients:
  - Text-only “browser” (lynx)
  - Mobile phones
  - Robots (software-only clients, e.g., search engine “crawlers”)
  - etc.
- Focus on traditional web browsers

Web Browsers

- First graphical browser running on general-purpose platforms: Mosaic (1993)

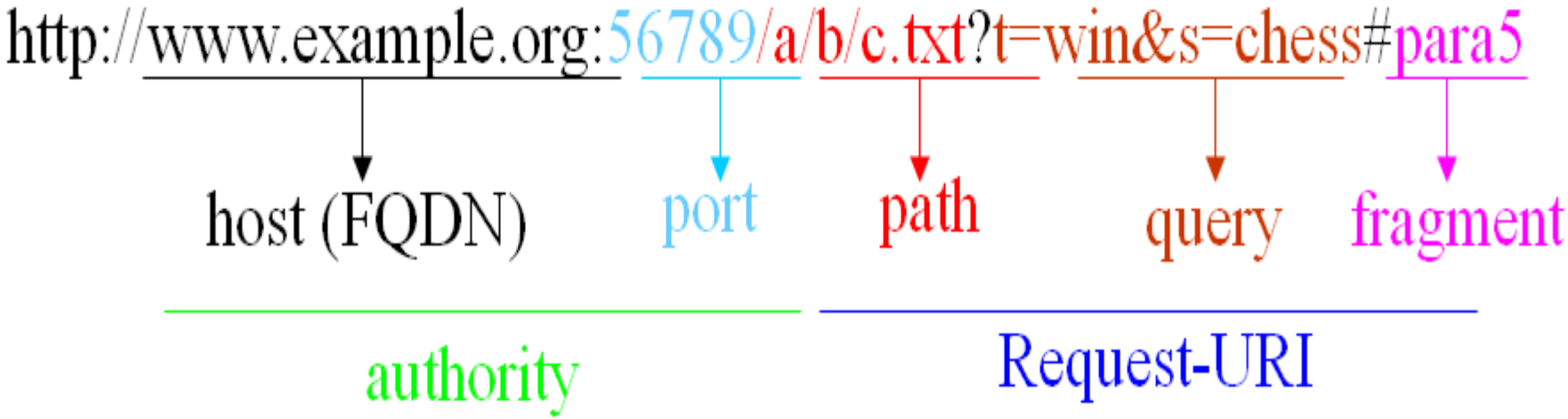
Web Browsers



Web Browsers

- Primary tasks:
  - Convert web addresses (URL's) to HTTP requests
  - Communicate with web servers via HTTP
  - **Render** (appropriately display) documents returned by a server

HTTP URL's



- Browser uses authority to connect via TCP
- Request-URI included in start line (/used for path if none supplied)
- Fragment identifier not sent to server (used to scroll browser client area)

Web Browsers

- Standard features
  - Save web page to disk
  - Find string in page
  - Fill forms automatically (passwords, CC numbers, …)
  - Set preferences (language, character set, cache and HTTP parameters)
  - Modify display style (e.g., increase font sizes)
  - Display raw HTML and HTTP header info (e.g., Last-Modified)
  - Choose browser themes (skins)
  - View history of web addresses visited
  - Bookmark favorite pages for easy return

Web Browsers

- Additional functionality:
  - Execution of scripts (e.g., drop-down menus)
  - Event handling (e.g., mouse clicks)
  - GUI for controls (e.g., buttons)
  - Secure communication with servers
  - Display of non-HTML documents (e.g., PDF) via plug-ins

Web Servers

- **Basic functionality:**
  - Receive HTTP request via TCP
  - Map Host header to specific **virtual host** (one of many host names sharing an IP address)
  - Map Request-URI to specific resource associated with the virtual host
    - File: Return file in HTTP response
    - Program: Run program and return output in HTTP response
  - Map type of resource to appropriate MIME type and use to set Content-Type header in HTTP response
  - Log information about the request and response

Web Servers

- httpd: UIUC, primary Web server 1995
- Apache: “A patchy” version of httpd, now the most popular server (esp. on Linux platforms)
- IIS: Microsoft Internet Information Server
- **Tomcat:**
  - Java-based
  - Provides container (Catalina) for running Java servlets (HTML-generating programs) as back-end to Apache or IIS
  - Can run stand-alone using Coyote HTTP front-end

Web Servers

- Some Coyote communication parameters:
  - Allowed/blocked IP addresses
  - Max. simultaneous active TCP connections
  - Max. queued TCP connection requests
  - “Keep-alive” time for inactive TCP connections
- Modify parameters to tune server performance

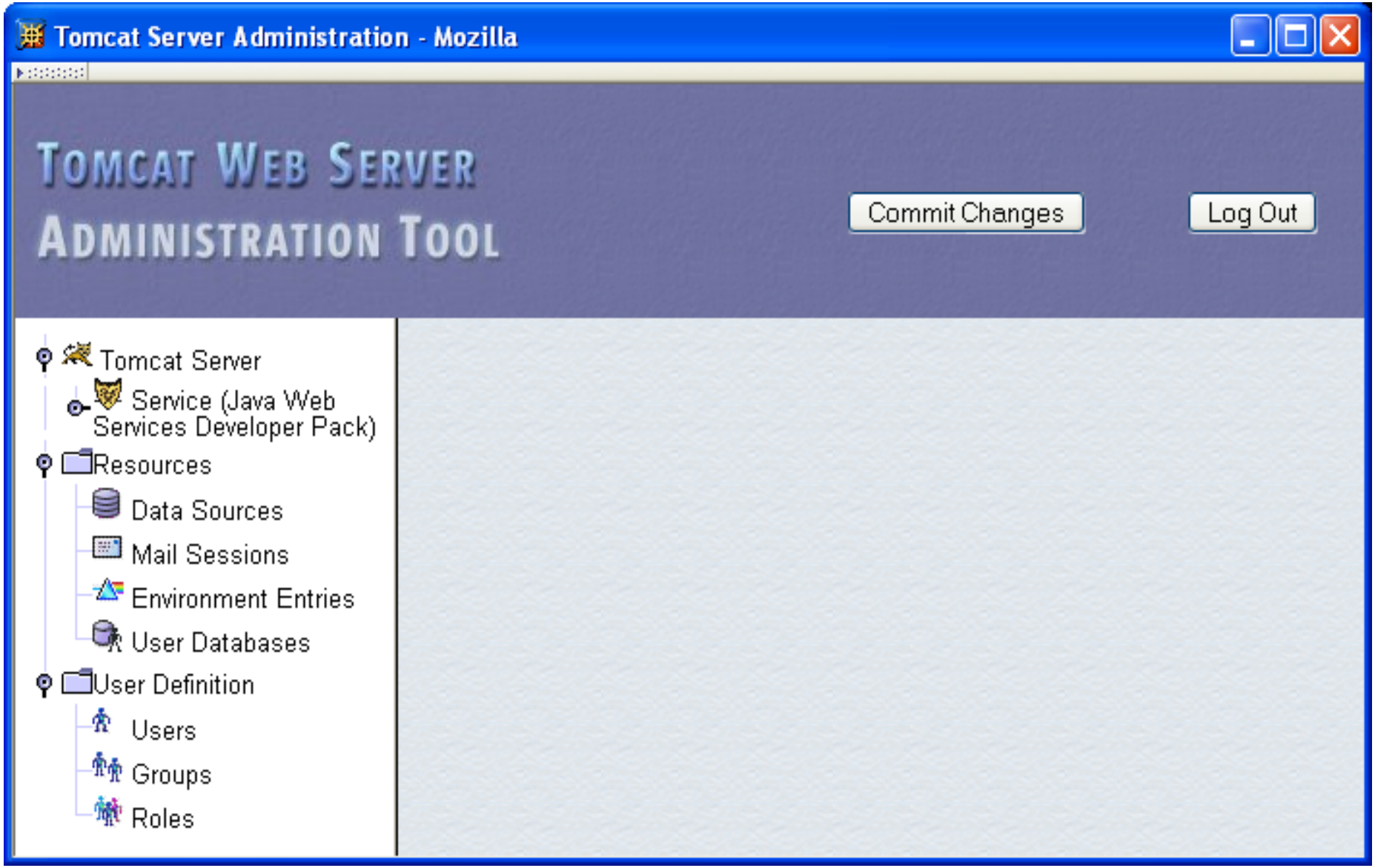
Web Servers

- Some Catalina container parameters:
  - Virtual host names and associated ports
  - Logging preferences
  - Mapping from Request-URI’s to server resources
  - Password protection of resources
  - Use of server-side caching

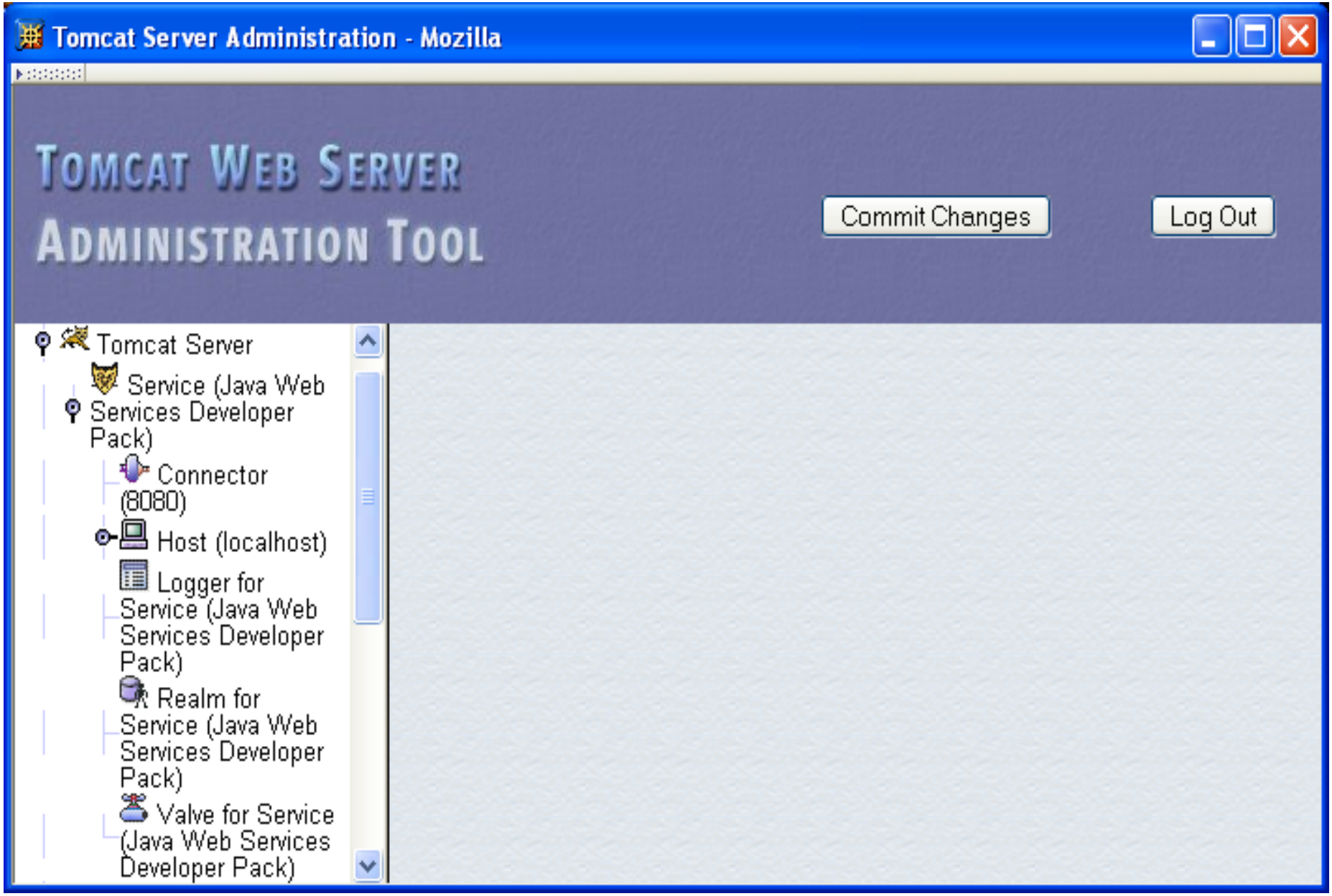
Tomcat Web Server

- HTML-based server administration
- Browse to http://localhost:8080 and click on Server Administration link
  - localhost is a special host name that means “this machine”

Tomcat Web Server



Tomcat Web Server



Tomcat Web Server

Tomcat Server Administration - Mozilla

TOMCAT WEB SERVER ADMINISTRATION TOOL

Commit Changes | Log Out

- Tomcat Server
  - Service (Java Web Services Developer Pack)
    - **Connector (8080)**
    - Host (localhost)
    - Logger for Service (Java Web Services Developer Pack)
    - Realm for Service (Java Web Services Developer Pack)
    - Valve for Service (Java Web Services

Connector (8080) **Connector Actions** —Available Actions—

—Available Actions—

Save | Reset

General	
Type:	HTTP
Scheme:	http
Debug Level:	0
Enable DNS Lookups:	True

Tomcat Web Server

- Some Connector fields:
  - Port Number: port “owned” by this connector
  - Max Threads: max connections processed simultaneously
  - Connection Timeout: keep-alive time

Tomcat Web Server

Tomcat Server Administration - Mozilla

TOMCAT WEB SERVER ADMINISTRATION TOOL

Commit Changes | Log Out

- Tomcat Server
  - Service (Java Web Services Developer Pack)
    - Connector (8080)
    - **Host (localhost)**
      - Logger for Service (Java Web Services Developer Pack)
      - Realm for Service (Java Web Services Developer Pack)
      - Valve for

Host Properties

Property	Value
Name:	localhost
Application Base:	webapps
Auto Deploy:	True
Debug Level:	0
Deploy On Startup:	True

Tomcat Web Server

- Each Host is a virtual host (can have multiple per Connector)
- Some fields:
  - Host: localhost or a fully qualified domain name
  - Application Base: directory (may be path relative to JWSDP installation directory) containing resources associated with this Host

Tomcat Web Server

Tomcat Server Administration - Mozilla

TOMCAT WEB SERVER ADMINISTRATION TOOL

Commit Changes | Log Out

- Tomcat Server
  - Service (Java Web Services Developer Pack)
    - Connector (8080)
    - Host (localhost)
      - Context (/)
      - Context (/RegistryServer)
      - Context (/Xindice)
      - Context (/admin)
      - Context (/gs)
      - Context (/jaxrpc-HelloWorld)
      - Context (/jsf-cardemo)
      - Context (/jsf-components)
      - Context

Context (/) Context Actions —Available Actions—

Save | Reset

Context Properties

Property	Value
Cookies:	True
Cross Context:	False
Debug Level:	0
Document Base:	C:\jwsdp-1.3\webapps\ROOT

Tomcat Web Server

- Context provides mapping from Request-URI path to a web application
- Document Base field is directory (possibly relative to Application Base) that contains resources for this web application
- For this example, browsing to http://localhost:8080/ returns resource from c:\jwsdp-1.3\webapps\ROOT
  - Returns index.html (standard welcome file)

Tomcat Web Server

- Access log records HTTP requests
- Parameters set using AccessLogValve
- Default location: logs/access_log.* under JWSDP installation directory
- Example “common” log format entry (one line): www.example.org - admin [20/Jul/2005:08:03:22 - 0500] "GET/admin/frameset.jsp HTTP/1.1“ 200 920

Tomcat Web Server

- Other logs provided by default in JWSDP:
  - Message log messages sent to log service by web applications or Tomcat itself
    - logs/jwsdp_log.*: default message log
    - logs/localhost_admin_log.*: message log for web apps within /admin context
  - System.out and System.err output (exception traces often found here):
    - logs/launcher.server.log

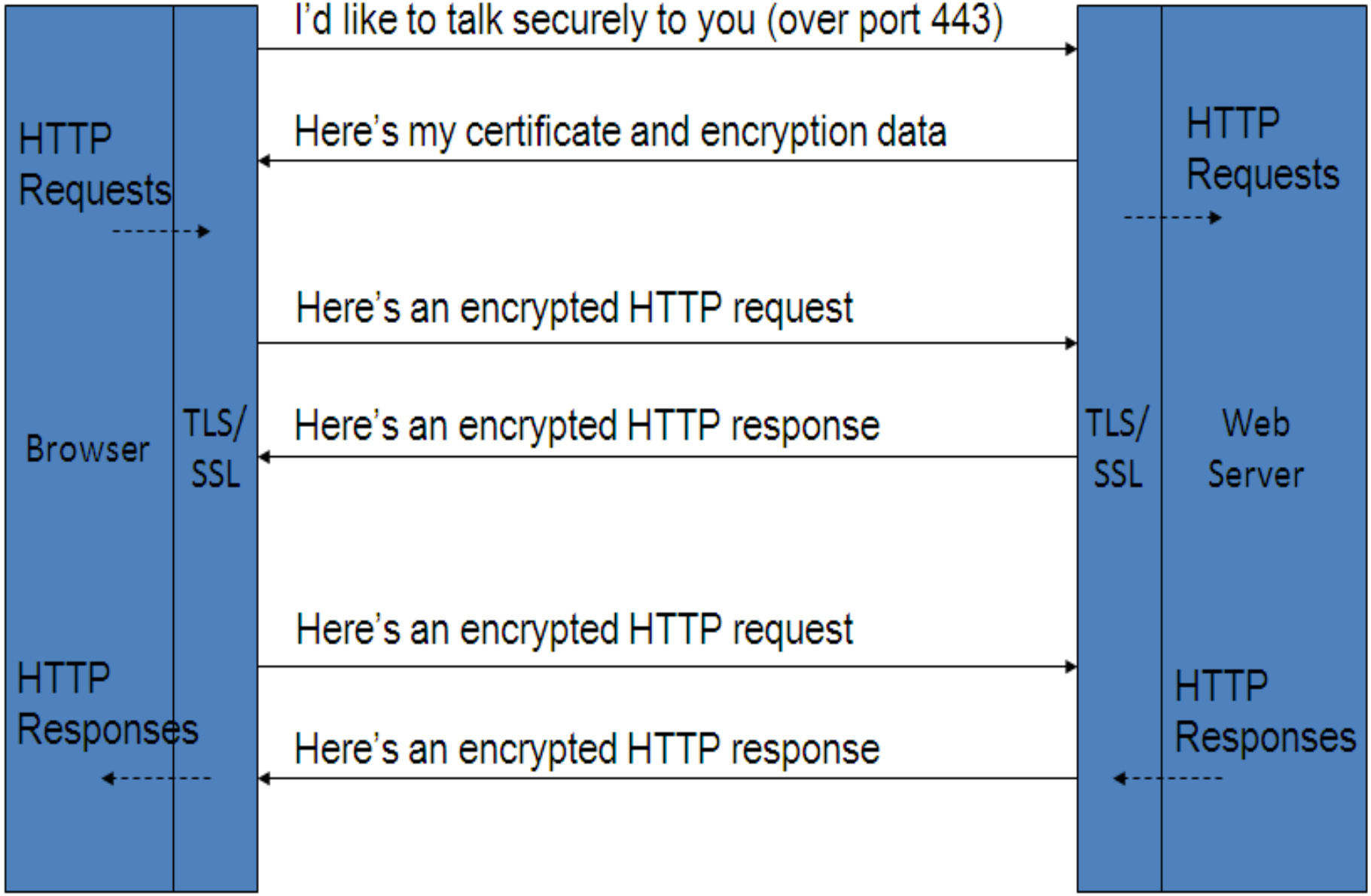
Tomcat Web Server

- **Access control:**
  - Password protection (e.g. admin pages)
    - Users and roles defined in conf/tomcat-users.xml
  - Deny access to machines
    - Useful for denying access to certain users by denying access from the machines they use
    - List of denied machines maintained in RemoteHostValve (deny by host name) or RemoteAddressValve (deny by IP address)

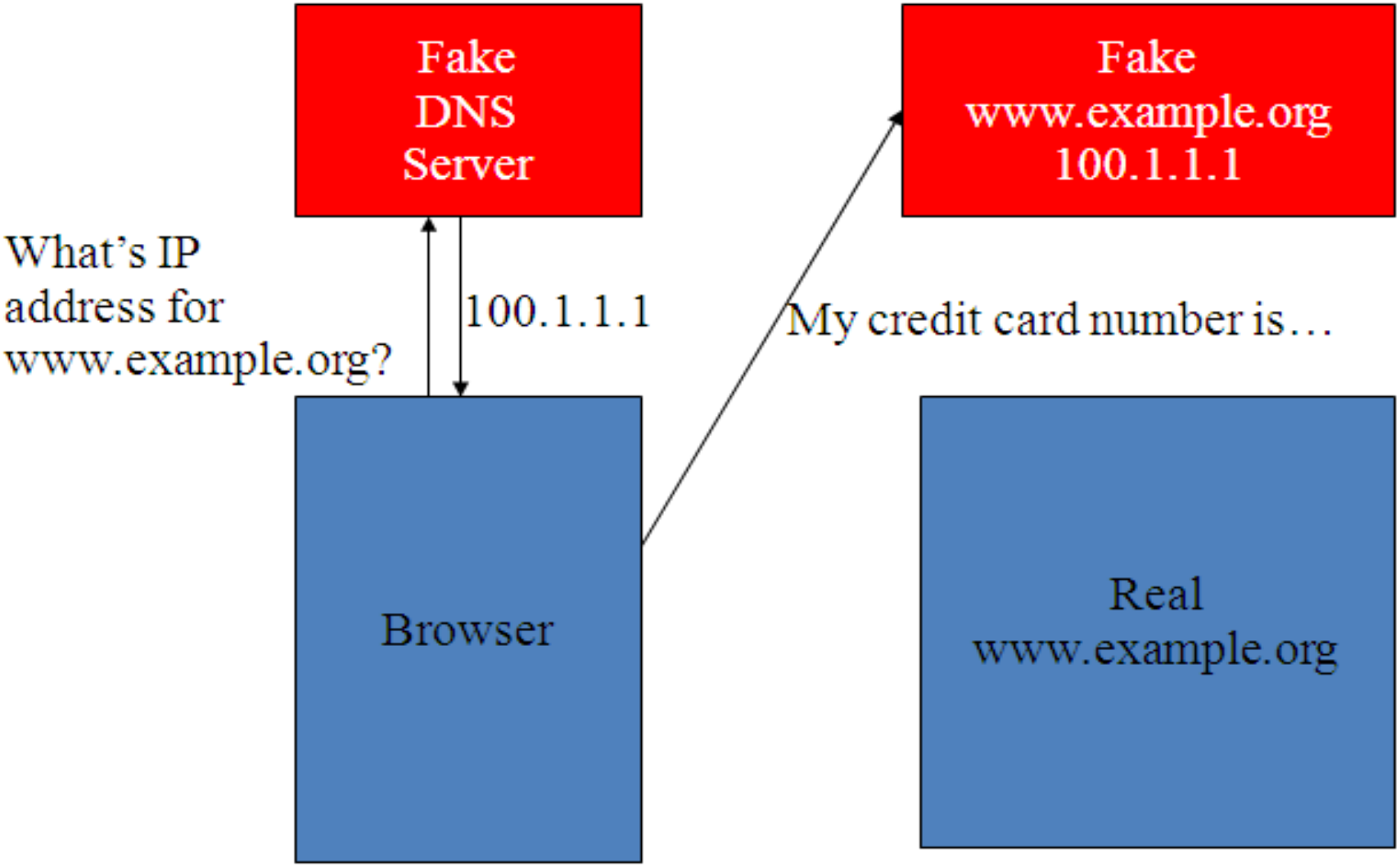
Secure Servers

- Since HTTP messages typically travel over a public network, private information (such as credit card numbers) should be encrypted to prevent eavesdropping
- HTTPs URL scheme tells browser to use encryption
- Common encryption standards:
  - Secure Socket Layer (SSL)
  - Transport Layer Security (TLS)

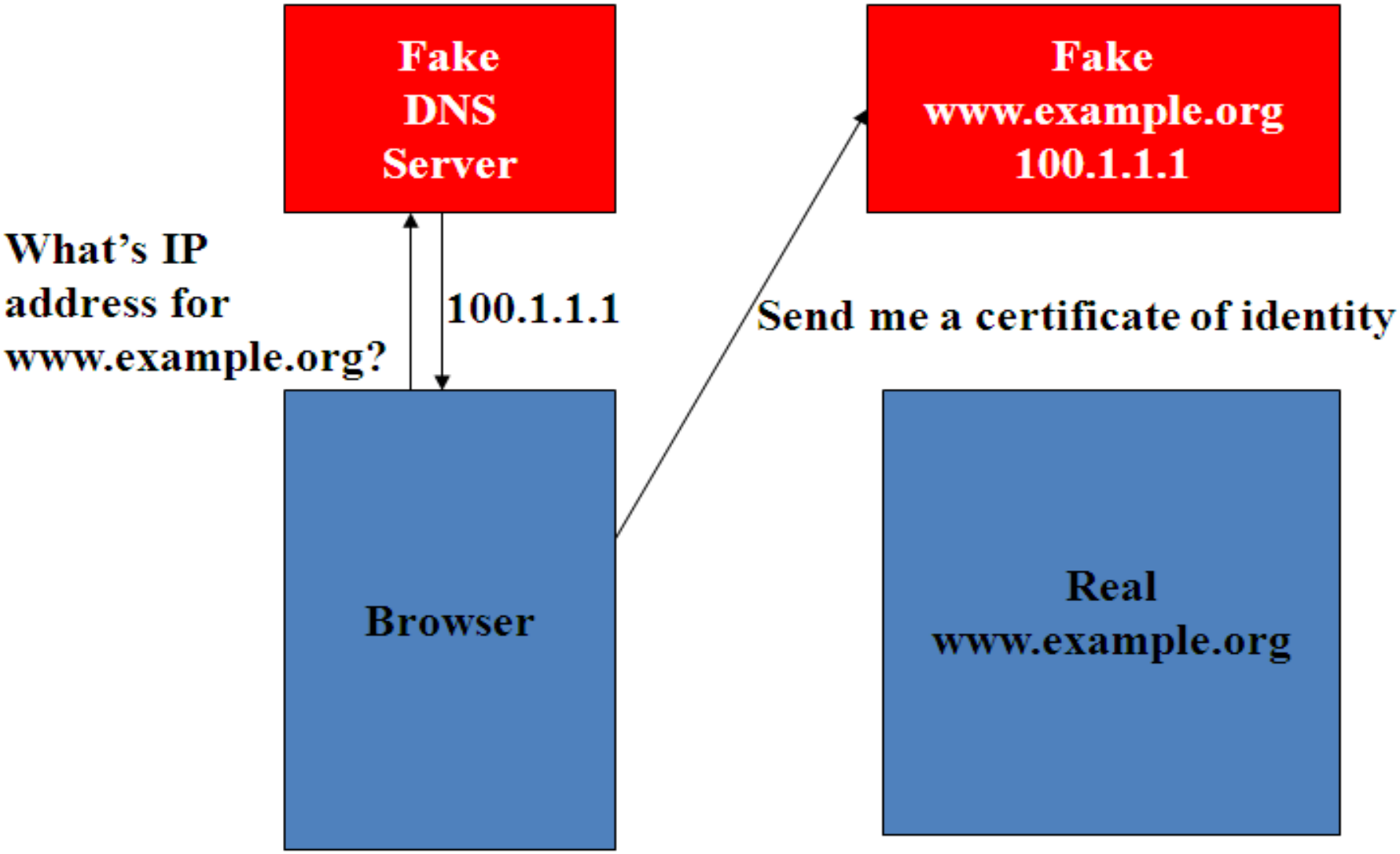
Secure Servers



Secure Servers
Man-in-the-Middle Attack



Secure Servers
Preventing Man-in-the-Middle



Markup Languages:

XHTML 1.0

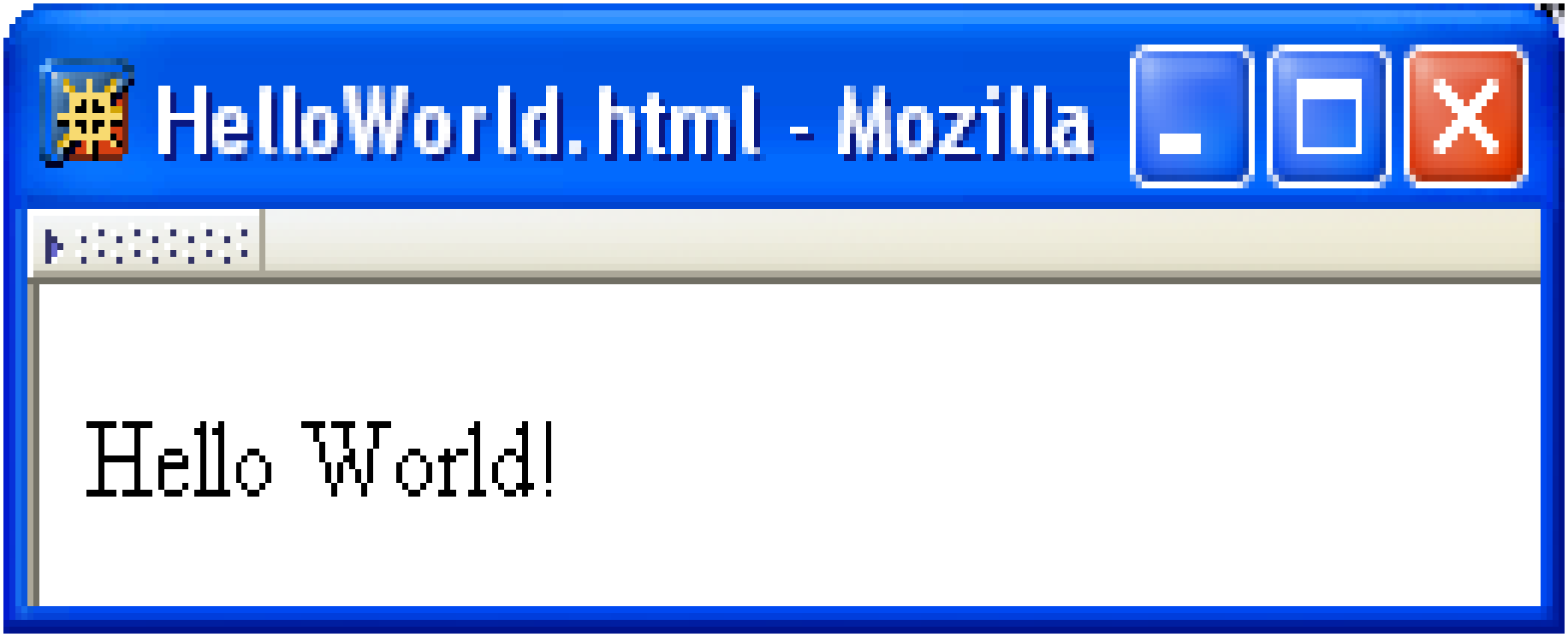
HTML “Hello World!”

Document Type Declaration

Document Instance

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      HelloWorld.html
    </title>
  </head>
  <body>
    <p>
      Hello World!
    </p>
  </body>
</html>
```

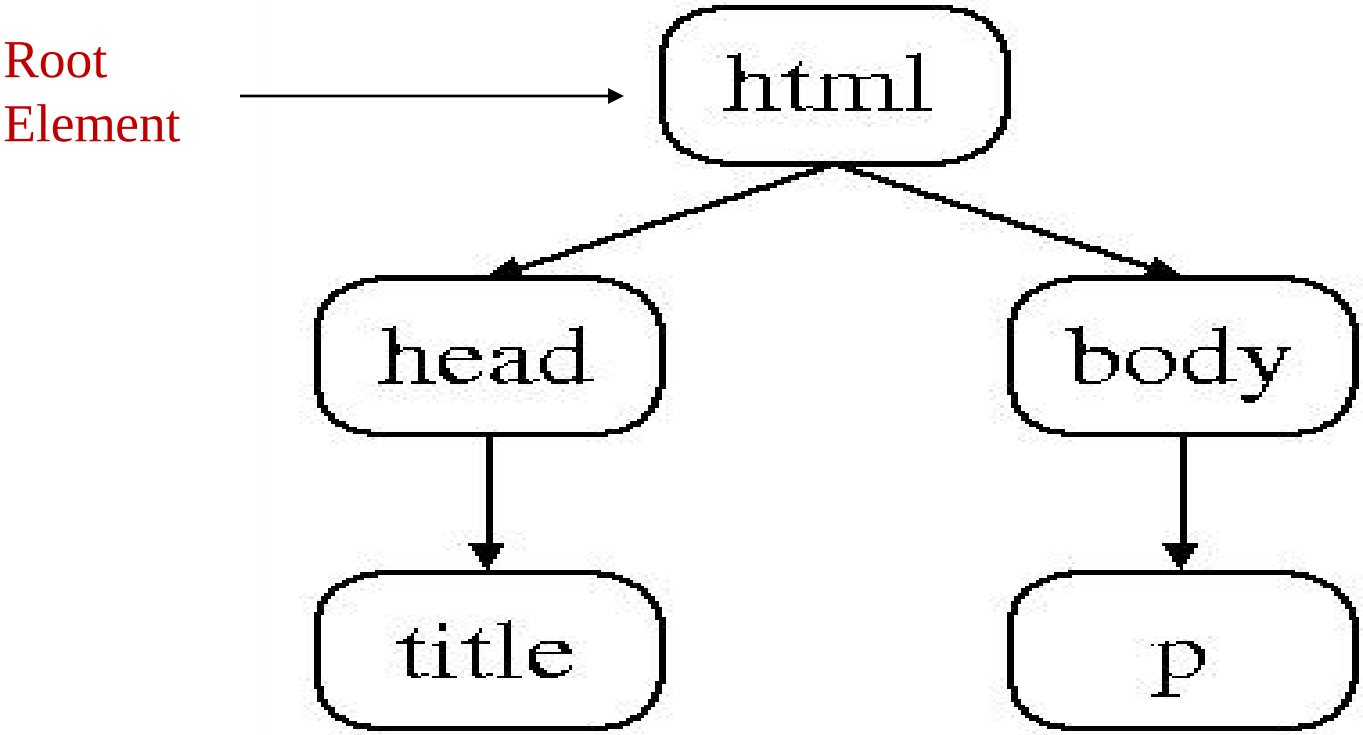
HTML “Hello World!”



HTML Tags and Elements

- Any string of the form < … > is a *tag*
- All tags in document instance of Hello World are either **end tags** (begin with </) or **start tags** (all others)
  - Tags are an example of **markup**, that is, text treated specially by the browser
  - Non-markup text is called **character data** and is normally displayed by the browser
- String at beginning of start/end tag is an **element name**
- Everything from start tag to matching end tag, including tags, is an **element**
  - **Content** of element excludes its start and end tags

HTML Element Tree



HTML Root Element

- Document type declaration specifies name of root element: <!DOCTYPE html
- Root of HTML document must be html
- XHTML 1.0 requires that this element contain xmlns **attribute specification** (name/value pair)

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

HTML head and body Elements

- The **body** element contains information displayed in the browser client area
- The **head** element contains information used for other purposes by the browser:
  - title (shown in title bar of browser window)
  - scripts (client-side programs)
  - style (display) information
  - etc.

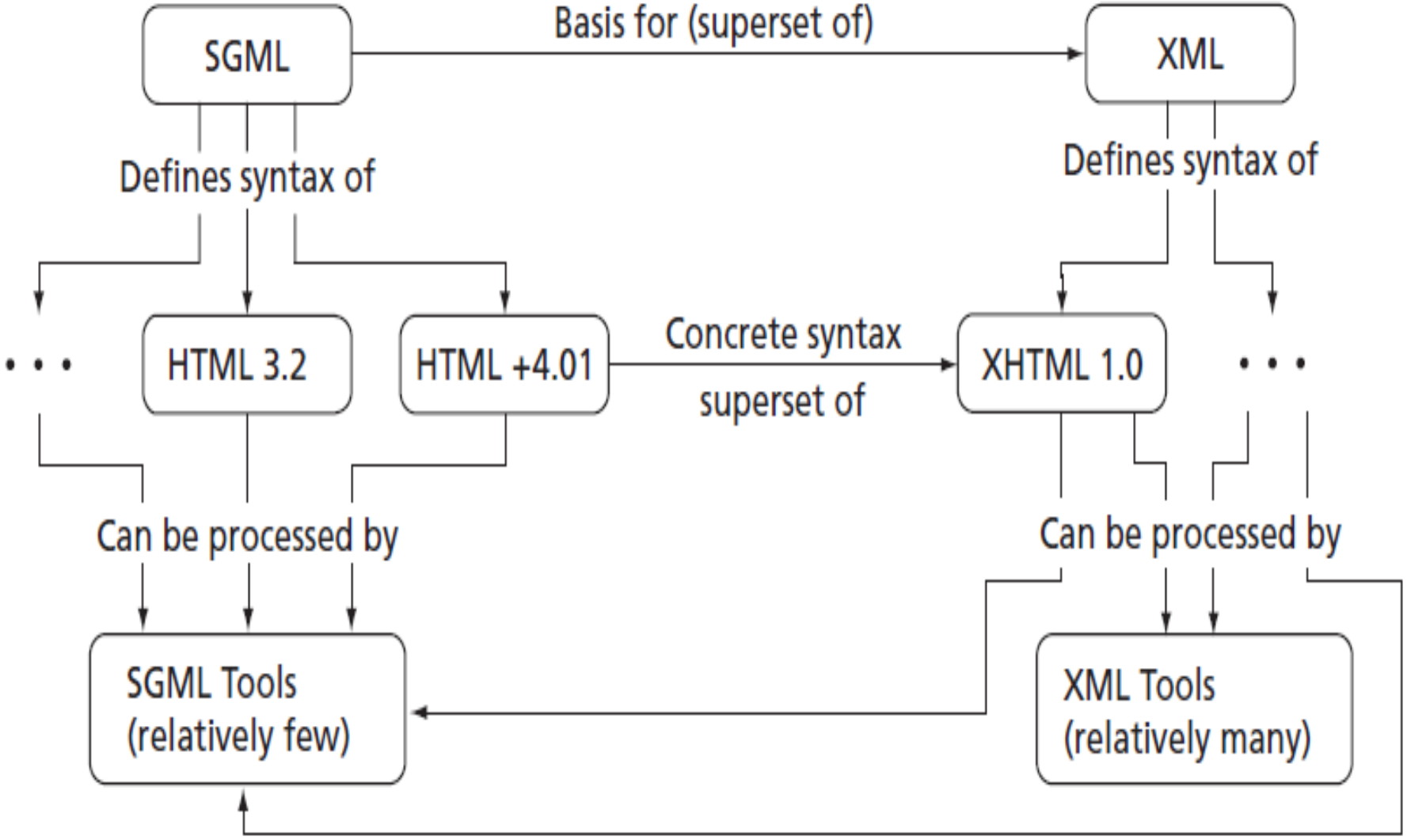
HTML History

- 1990: HTML invented by Tim Berners-Lee
- 1993: Mosaic browser adds support for images, sound, video to HTML
- 1994-~1997: **“Browser wars”** between Netscape and Microsoft, HTML defined operationally by browser support
- ~1997-present: Increasingly, World-Wide Web Consortium (W3C) recommendations define HTML

HTML Versions

- HTML 4.01 (Dec 1999) syntax defined using **Standard Generalized Markup Language (SGML)**
- XHTML 1.0 (Jan 2000) syntax defined using **Extensible Markup Language (XML)**
- Primary differences:
  - HTML allows some **tag omissions** (e.g., end tags)
  - XHTML element and attribute names are **lower case** (HTML names are case-insensitive)
  - XHTML requires that attribute **values be quoted**

SGML and XML


SGML
Basis for (superset of)
XML
Defines syntax of
Defines syntax of
. . .
HTML 3.2
HTML +4.01
Concrete syntax
superset of
XHTML 1.0
. . .
Can be processed by
Can be processed by
SGML Tools
(relatively few)
XML Tools
(relatively many)

HTML “Flavors”

- For HTML 4.01 and XHTML 1.0, the document type declaration can be used to select one of three “flavors”:
  - **Strict**: W3C ideal
  - **Transitional**: Includes deprecated elements and attributes (W3C recommends use of *style sheets* instead)
  - **Frameset**: Supports frames (subwindows within the client area)

HTML Frameset

Applet (Java 2 Platform SE v1.4.2) - Mozilla

Java™ 2 Platform Std. Ed. v1.4.2

All Classes

Packages

java.applet

java.awt

java.applet

Interfaces

AppletContext
AppletStub
AudioClip

Classes

Applet

Overview Package Class Use Tree Deprecated Index Help *Java™ 2 Platform Std. Ed. v1.4.2*

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.applet

Class Applet

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
```

All Implemented Interfaces:

Accessible, ImageObserver, MenuContainer, Serializable

HTML Document Type Declarations

- **XHTML 1.0 Strict:**

  <!DOCTYPE html

  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

  http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>

- **XHTML 1.0 Frameset:**

  <!DOCTYPE html

  PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN“

  http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd>

- **HTML 4.01 Transitional:**

  <!DOCTYPE HTML

  PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN“

  http://www.w3.org/TR/html4/loose.dtd>

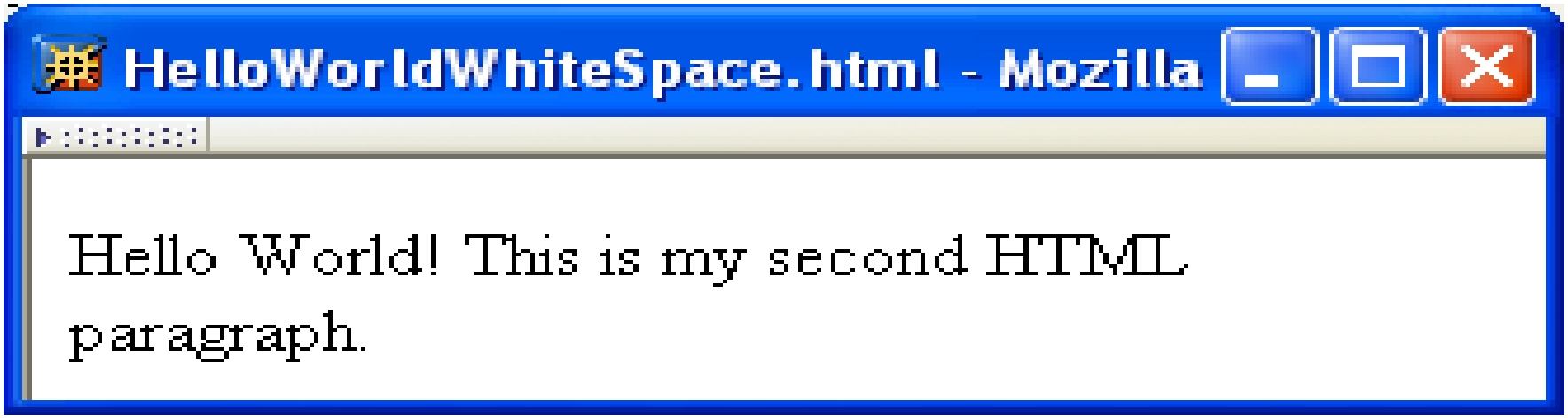
XHTML White Space

- Four white space characters: carriage return, line feed, space, horizontal tab
- Normally, character data is **normalized**:
  - All white space is converted to space characters
  - Leading and trailing spaces are trimmed
  - Multiple consecutive space characters are replaced by a single space character

XHTML White Space

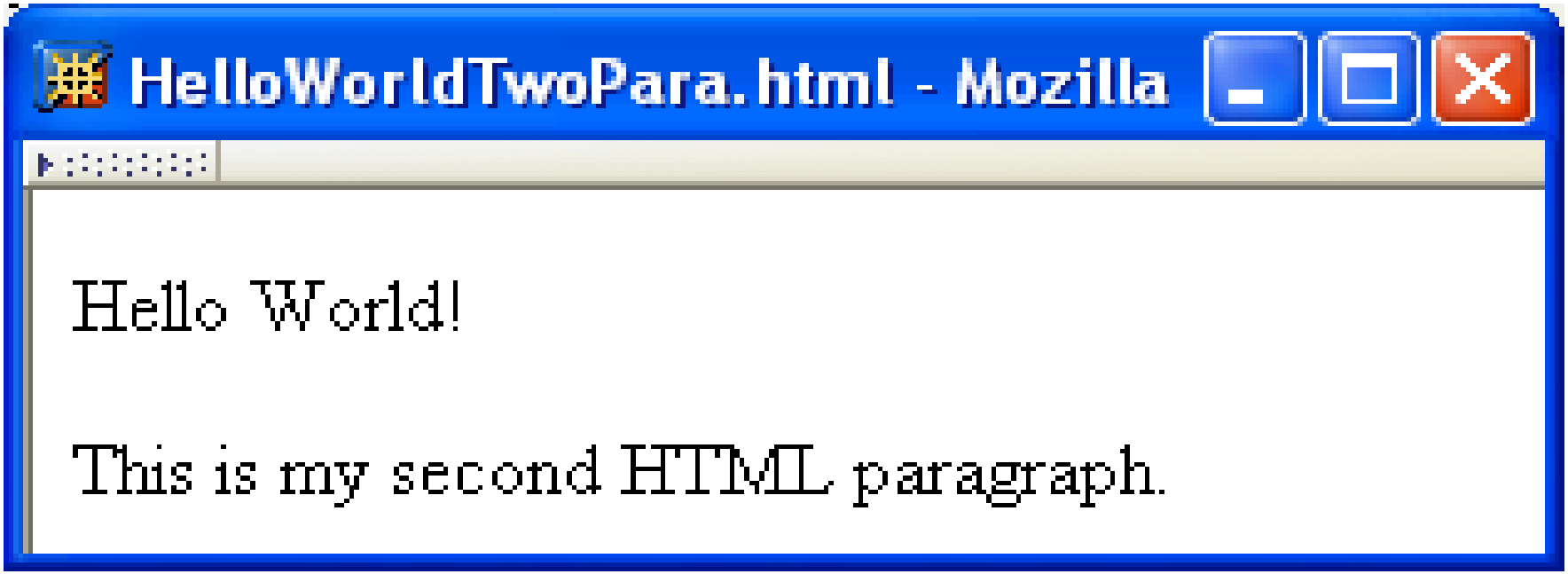
```
<body>
  <p>
    Hello World!

    This is my second HTML paragraph.
  </p>
</body>
```



XHTML White Space

```
<p>
  Hello World!
</p>
<p>
  This is my second HTML paragraph.
</p>
```



Unrecognized HTML Elements

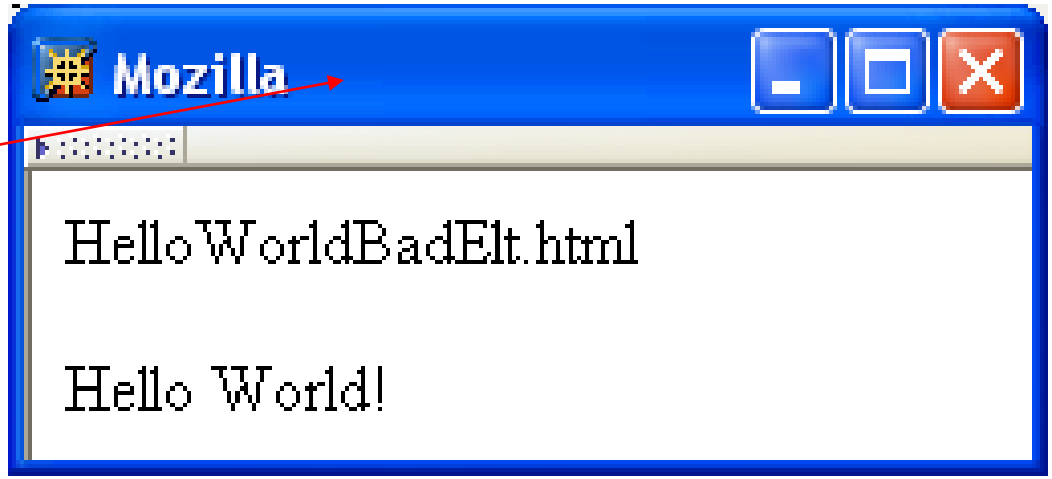
Misspelled element name

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <titl>
      HelloWorldBadElt.html
    </title>
  </head>
  <body>
    <p>
      Hello World!
    </p>
  </body>
</html>
```

Unrecognized HTML Elements

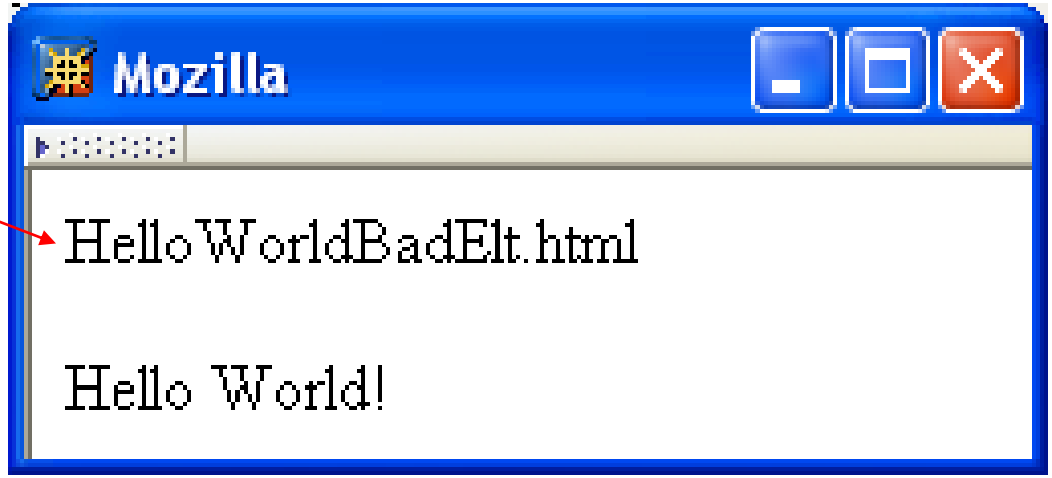
title character data

Belongs here



title character data

Displayed here



Unrecognized HTML Elements

- Browsers ignore tags with unrecognized element names, attribute specifications with unrecognized attribute names
  - Allows evolution of HTML while older browsers are still in use
- Implication: an HTML document may have errors even if it displays properly
- Should use an HTML validator to check syntax

HTML References

- Since < marks the beginning of a tag, how do you include a < in an HTML document?
- Use markup known as a **reference**
- Two types:
  - **Character reference** specifies a character by its Unicode code point
    - For <, use < or < or <
  - **Entity reference** specifies a character by an HTML-defined name
    - For <, use <

HTML References

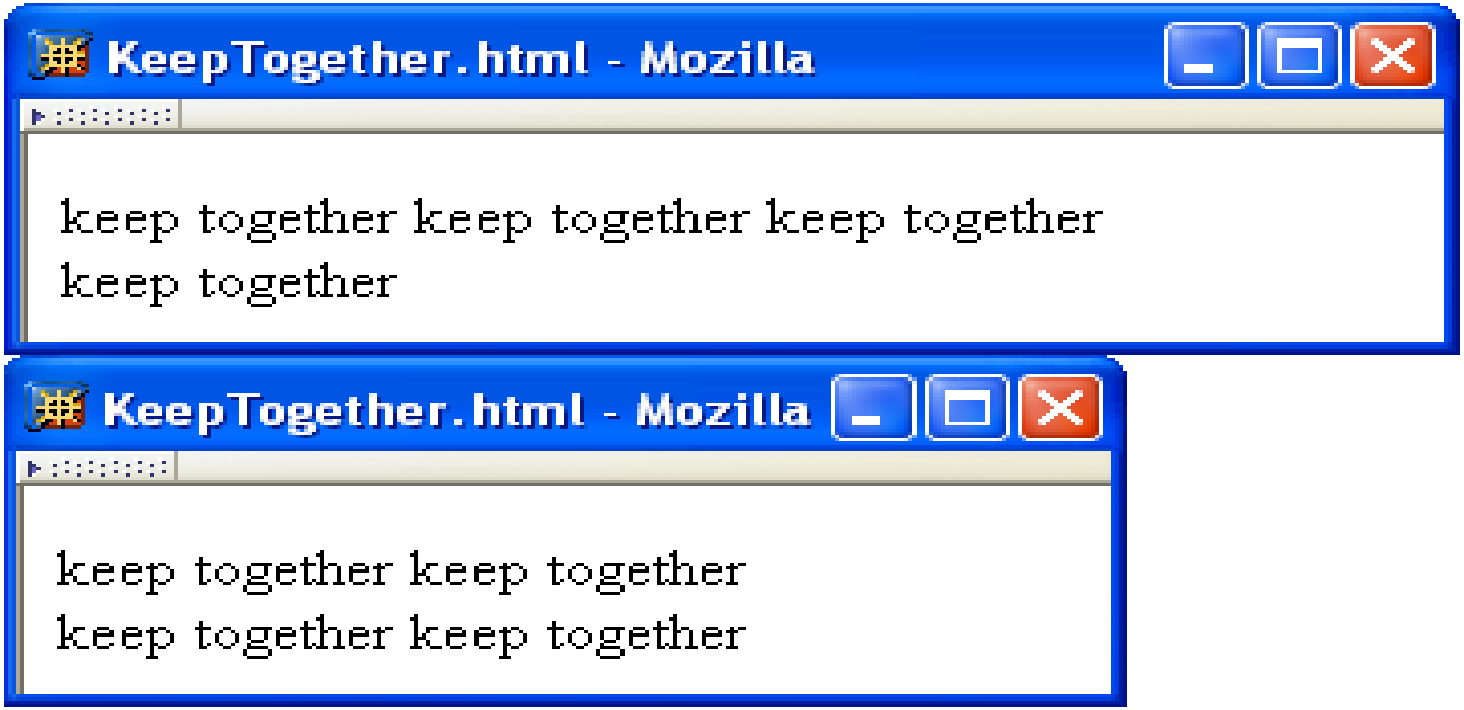
Character	Entity Reference	Character Reference (decimal)
<	<	<
>	>	>
&	&	&
"	"	"
'	'	'
©	©	©
ñ	ñ	ñ
α	α	α
∀	∀	∀

HTML References

- Since < and **&** begin markup, within character data or attribute values these characters must ***always*** be represented by references (normally **<** and **&**)
- Good idea to represent > using reference (normally **>**)
  - Provides consistency with treatment of <
  - Avoids accidental use of the reserved string]]>

HTML References

- **Non-breaking space ( )** produces space but counts as part of a word
  - Ex: keep together keep together …

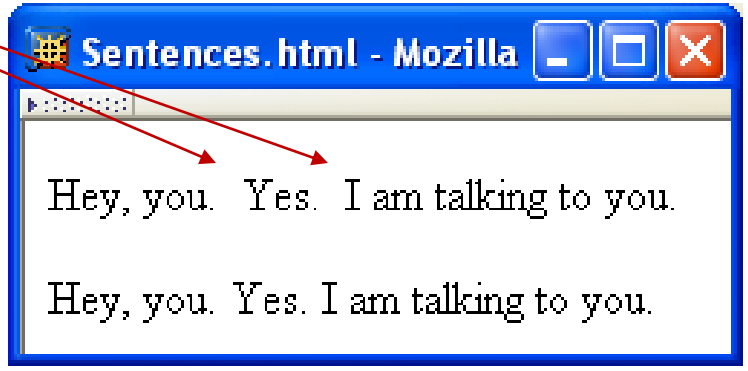


HTML References

- Non-breaking space often used to create multiple spaces (not removed by normalization)

```
<p>
  Hey, you.  Yes.  I am talking to you.
</p>
<p>
  Hey, you.  Yes.  I am talking to you.
</p>
```

** + space displays as two spaces**

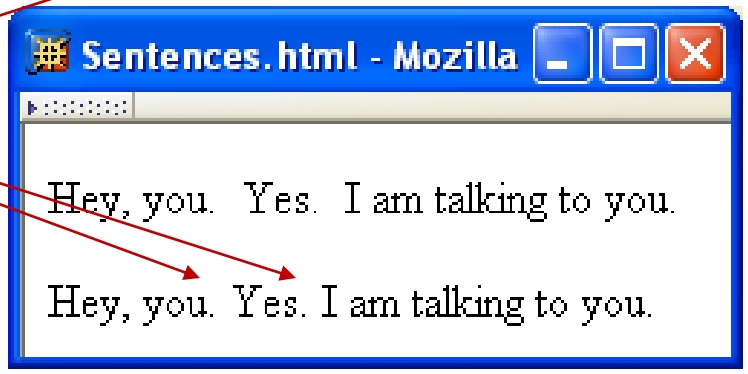


HTML References

- Non-breaking space often used to create multiple spaces (not removed by normalization)

```
<p>
  Hey, you.  Yes.  I am talking to you.
</p>
<p>
  Hey, you.  Yes.  I am talking to you.
</p>
```

two spaces display as one



XHTML Attribute Specifications

- Example:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

- Syntax:
  - Valid attribute names specified by HTML recommendation (or XML, as in xml:lang)
  - Attribute values must be quoted (matching single or double quotes)
  - Multiple attribute specifications are space-separated, order-independent

XHTML Attribute Values

- Can contain embedded quotes or references to quotes

```
✓ value = "Ain't this grand!"
✓ value = "He said, &quot;She said&quot;, then sighed."
✗ value = "He said, "She said", then sighed."
```

- May be normalized by browser
  - Best to normalize attribute values yourself for optimal browser compatibility

Common HTML Elements

- **Headings** are produced using **h1, h2, …, h6** elements:

```
<h1>
  Some Common HTML Elements
</h1>
<h2>
  Simple formatting elements
</h2>
```

- Should use h1 for highest level, h2 for next highest, etc.
  - Change style (next chapter) if you don't like the "look" of a heading

Common HTML Elements

- Use **pre** to **retain format** of text and display using monospace font:

```
  <pre>
Use pre (for "preformatted") to
  preserve white space and use
    monospace type.
    (But note that tags such as<br />still work!)
  </pre>
```

- Note that any embedded markup (such as
) is still treated as markup!

Common HTML Elements

- **br** element represents **line break**
- br is example of an **empty element**, i.e., element that is not allowed to have content
- XML allows two syntactic representations of empty elements
  - **Empty tag** syntax <br/> is recommended for browser compatibility
  - XML parsers also recognize syntax
 </br> (start tag followed immediately by end tag), but many browsers do not understand this for empty elements

Common HTML Elements

- Text can **be formatted** in various ways:
  - Apply **style sheet** technology (next chapter) to a **span element** (a styleless **wrapper**):

```
<span style="font-style:italic">separating line</span>
```

  - Use a **phrase element** that specifies semantics of text (not style directly):

```
<strong>hr</strong>
```

  - Use a **font style element**
    - Not recommended, but frequently used

Common HTML Elements

Element	Font used by content
b	Bold-face
i	Italic
tt	"Teletype" (fixed-width font)
big	Increased font size
small	Decreased font size

Common HTML Elements

- **Horizontal rule** is produced using hr
- Also an empty element
- Style can be modified using style sheet technology

Common HTML Elements

- **Images** can be embedded using **img** element

```
<img
    src="http://www.w3.org/Icons/valid-xhtml10"
    alt="Valid XHTML 1.0!" height="31" width="88"
    style="float:right" />
```

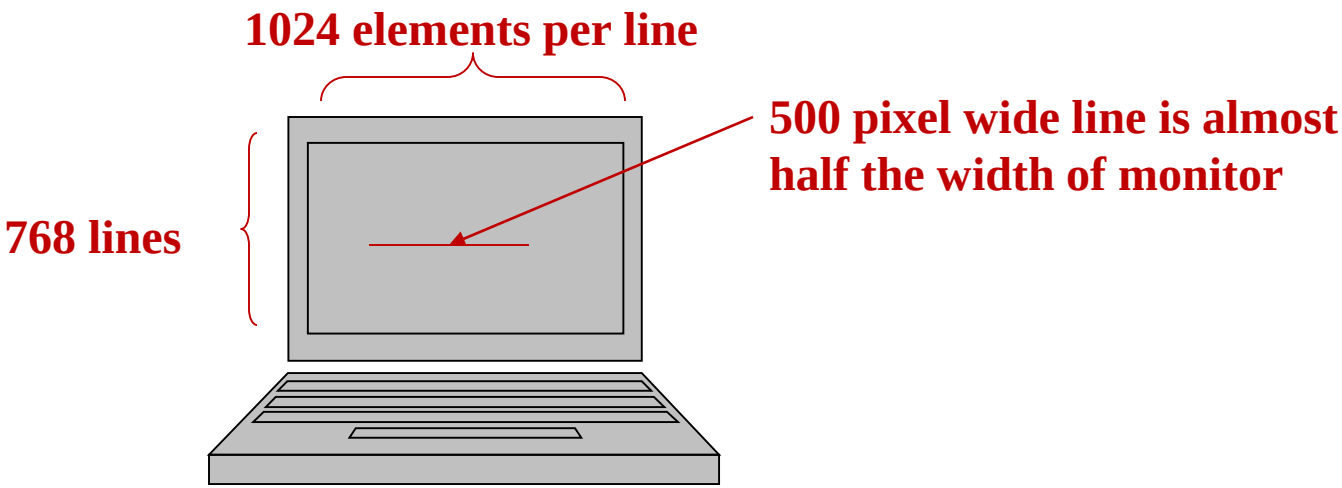
- Attributes:
  - **src:** URL of image file (required). Browser generates a GET request to this URL.
  - **alt:** text description of image (required)
  - **height / width:** dimensions of area that image will occupy (recommended)

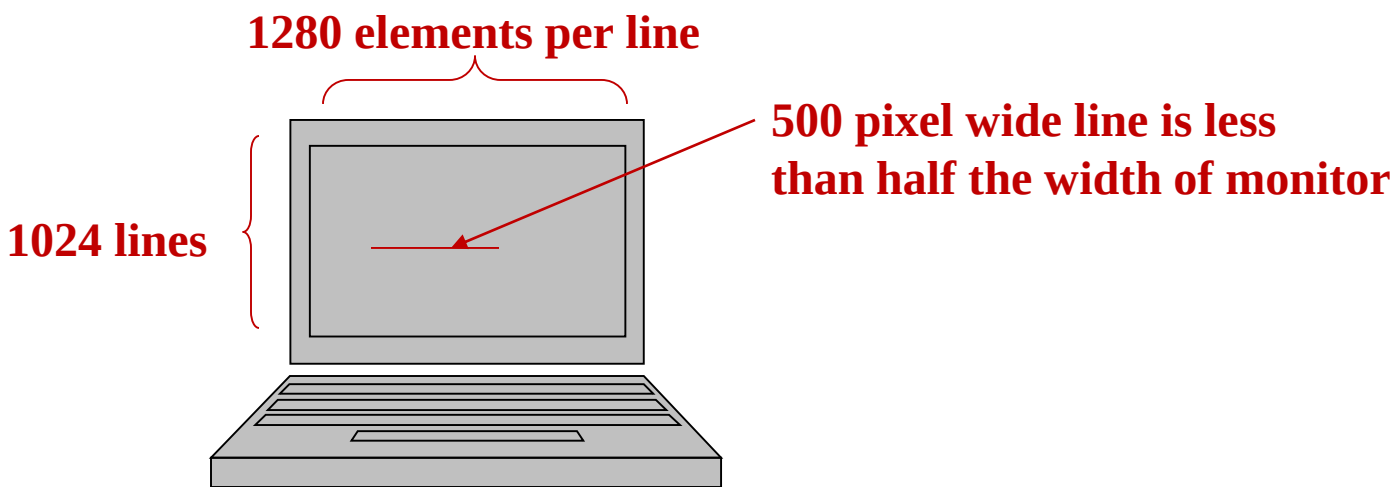
Common HTML Elements

- If height and width not specified for image, then browser may need to rearrange the client area after downloading the image (**poor user interface** for Web page)
- If height and width specified are not the same as the original dimensions of image, browser will **resize** the image
- Default units for height and width are “picture elements” (**pixels**)
  - Can specify percentage of client area using string such as “50%”

Common HTML Elements

- Monitor resolution determines pixel size





Common HTML Elements

- **Hyperlinks** are produced by the **anchor** element **a**

```
See
<a href="http://www.w3.org/TR/html4/index/elements.html">the
  W3C HTML 4.01 Element Index</a>
for a complete list of elements.
```

- Clicking on a hyperlink causes browser to issue GET request to URL specified in href attribute and render response in client area
- Content of anchor element is text of hyperlink (avoid leading/trailing space in content)

Common HTML Elements

- Anchors can be used as **source** (previous example) or **destination**

```
<a id="section1" name="section1"></a>
```

- The fragment portion of a URL is used to reference a destination anchor

```
<a href="http://www.example.org/PageWithAnchor.html#section1">...
```

- Browser scrolls so destination anchor is at (or near) top of client area

Common HTML Elements

- Comments are a special form of tag

```
<!-- Notice that img must nest within a "block" element,
     such as p -->
```

- Not allowed to use -- within comment

```
<!-- This is NOT
  -- a good comment.
  -->
```

```
<!-- Can't end with more than two dashes! --->
```

Nesting Elements

- If one element is nested within another element, then the content of the inner element is also content of the outer element

```
<tt><strong>hr</strong></tt>
```

- XHTML requires that elements be properly nested

× `<tt><strong>hr</tt></strong>`

Nesting Elements

- Most HTML elements are either **block** or **inline**
  - **Block**: browser automatically generates line breaks before and after the element content
    - Ex: p
  - **Inline**: element content is added to the “flow”
    - Ex: span, tt, strong, a

Nesting Elements

- Syntactic rules of thumb:
  - Children of body must be blocks
  - Blocks can contain inline elements
  - Inline elements *cannot* contain blocks
- Specific rules for each version of (X)HTML are defined using SGML or XML.

Relative URL's

- Consider an <img> start tag containing attribute specification

```
src="valid-xhtml10.png"
```

- This is an example of a **relative URL**: it is interpreted relative to the URL of the document that contains the img tag
  - If document URL is **http://localhost:8080/MultiFile.html** then relative URL above represents **absolute URL http://localhost:8080/valid-xhtml10.png.**

Relative URL's

Relative URL	Absolute URL
d/e.html	http://www.example.org/a/b/d/e.html
../f.html	http://www.example.org/a/f.html
../../g.html	http://www.example.org/g.html
../h/i.html	http://www.example.org/a/h/i.html
/j.html	http://www.example.org/j.html
/k/l.html	http://www.example.org/k/l.html

Relative URL's

- Query and fragment portions of a relative URL are appended to the resulting absolute URL
  - Example: If document URL is

    http://localhost:8080/PageAnch.html

    and it contains the anchor element

    ```
    <a href="#section1">...
    ```

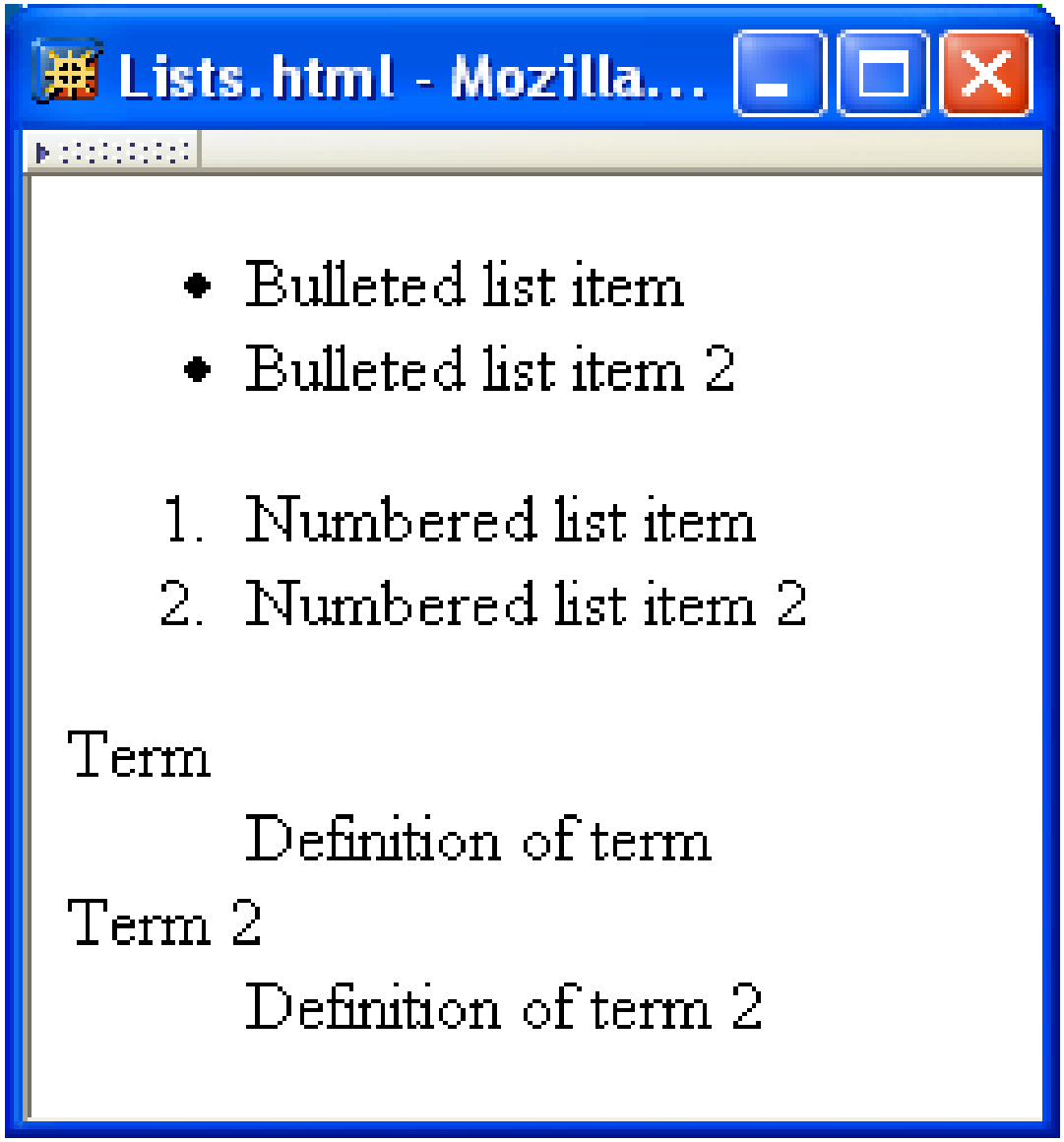
    then the corresponding absolute URL is

    http://localhost:8080/PageAnch.html#section1

Relative URL's

- Advantages:
  - Shorter than absolute URL's
  - Primary: can **change the URL of a document** (e.g., move document to a different directory or rename the server host) without needing to change URL's within the document
- Should use relative URL's whenever possible

Lists


Lists.html - Mozilla...
Bulleted list item
Bulleted list item 2
1. Numbered list item
2. Numbered list item 2
Term
Definition of term
Term 2
Definition of term 2

Lists

Unordered List

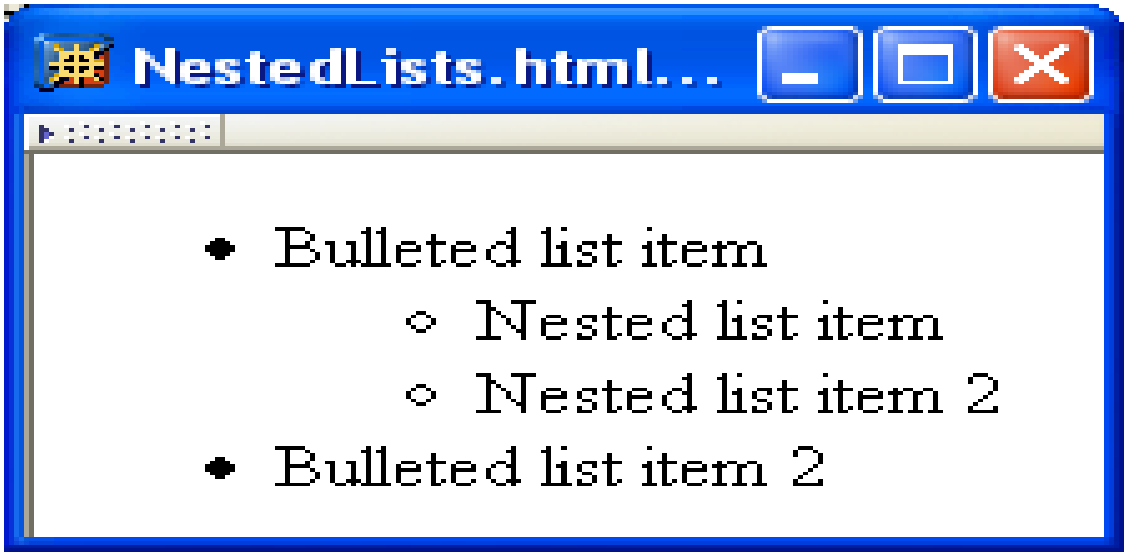
Ordered List

Definition List

```
<ul>
  <li>Bulleted list item</li>
  <li>Bulleted list item 2</li>
</ul>
<ol>
  <li>Numbered list item</li>
  <li>Numbered list item 2</li>
</ol>
<dl>
  <dt>Term</dt>
  <dd>Definition of term</dd>
  <dt>Term 2</dt>
  <dd>Definition of term 2</dd>
</dl>
```

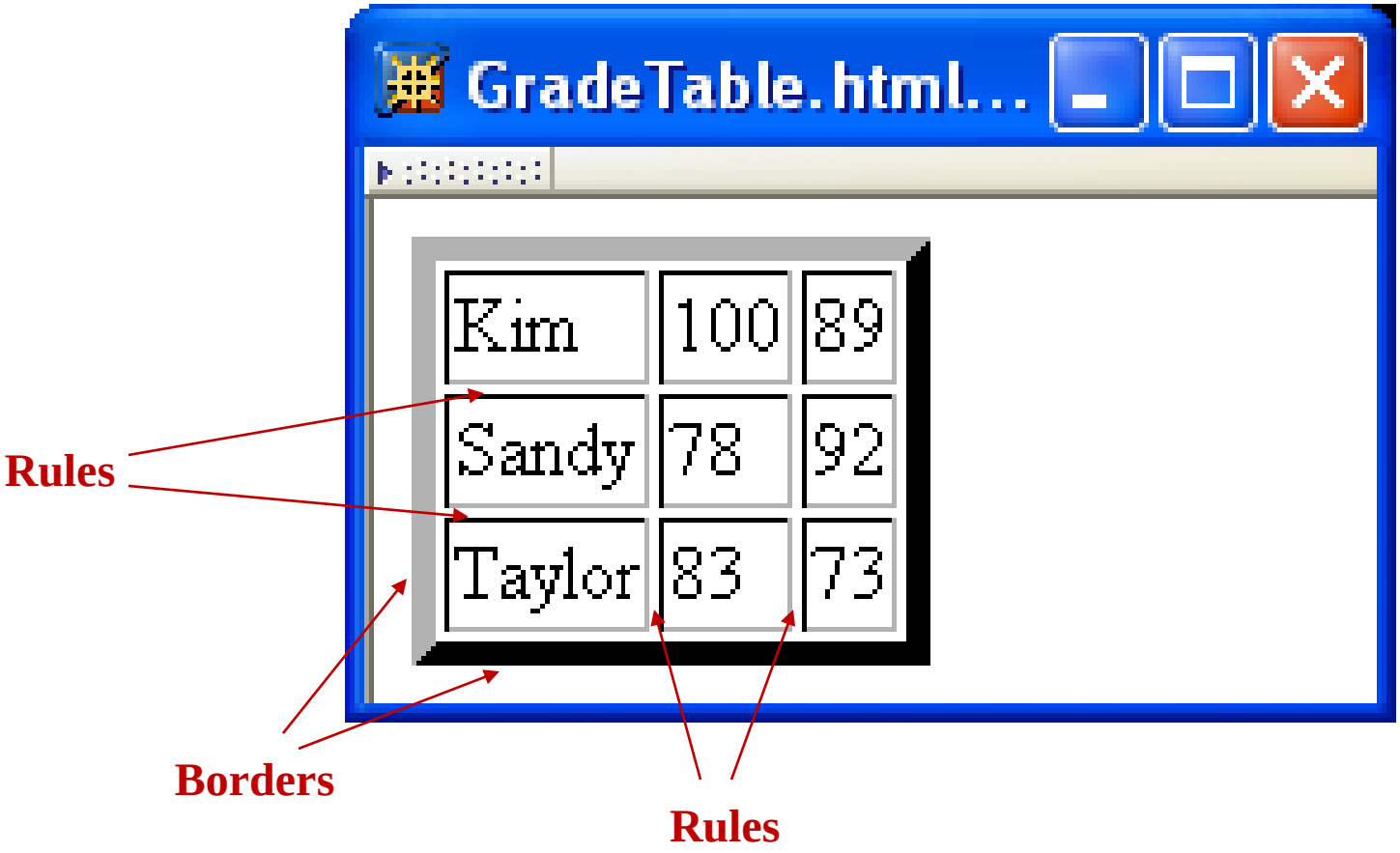
List Items

Lists




```
<ul>
  <li>Bulleted list item
    <ul>
      <li>Nested list item</li>
      <li>Nested list item 2</li>
    </ul>
  </li>
  <li>Bulleted list item 2</li>
</ul>
```

Tables


GradeTable.html...
Kim 100 89
Sandy 78 92
Taylor 83 73
Rules
Borders
Rules

Tables

Border 5 pixels, rules 1 pixel

```
<table border="5">
  <tr>
    <td>Kim</td><td>100</td><td>89</td>
  </tr>
  <tr>
    <td>Sandy</td><td>78</td><td>92</td>
  </tr>
  <tr>
    <td>Taylor</td><td>83</td><td>73</td>
  </tr>
</table>
```

Table Row

Table Data

Tables

GradeTableHdr.html - Mozilla

COSC 400 Student Grades

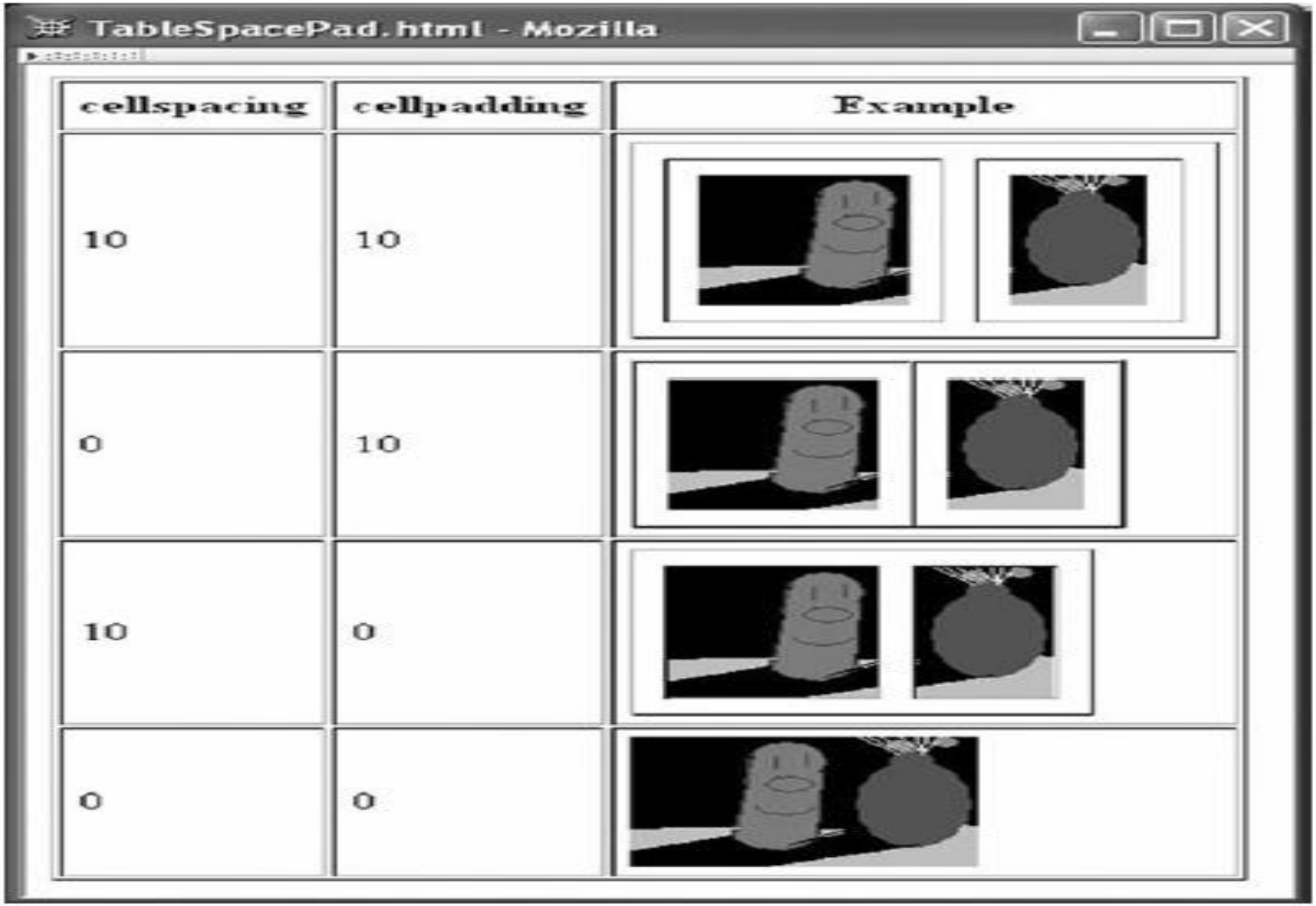
		Grades	
	Student	**Exam 1**	**Exam 2**
Undergraduates	Kim	100	89
	Sandy	78	92
Graduates	Taylor	83	73

Tables

```
<table border="5">
  <caption>
    COSC 400 Student Grades
  </caption>
  <tr>
    <td> </td><td> </td><th colspan="2">Grades</th>
  </tr>
  <tr>
    <td> </td><th>Student</th><th>Exam 1</th><th>Exam 2</th>
  </tr>
  <tr>
    <th rowspan="2">Undergraduates</th><td>Kim</td><td>100</td><td>89</td>
  </tr>
  <tr>
    <td>Sandy</td><td>78</td><td>92</td>
  </tr>
  <tr>
    <th>Graduates</th><td>Taylor</td><td>83</td><td>73</td>
  </tr>
</table>
```

Table Header

Tables


TableSpacePad.html - Mozilla
cellspacing
cellpadding
Example
10
10
0
10
10
0
0
0

Frames

Applet (Java 2 Platform SE v1.4.2) - Mozilla

Java™ 2 Platform Std. Ed. v1.4.2

All Classes

Packages

java.applet

java.awt

java.applet

Interfaces

AppletContext

AppletStub

AudioClip

Classes

Applet

Overview Package Class Use Tree Deprecated Index Help *Java™ 2 Platform Std. Ed. v1.4.2*

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.applet

Class Applet

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
```

All Implemented Interfaces:

Accessible, ImageObserver, MenuContainer, Serializable

Frames

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Java 2 Platform SE v1.4.2</title>
  </head>
  <frameset cols="20%,80%">
    <frameset rows="1*,2*">
      <frame src="overview-frame.html"
        id="upperLeftFrame" name="upperLeftFrame"></frame>
      <frame src="allclasses-frame.html"
        id="lowerLeftFrame" name="lowerLeftFrame"></frame>
    </frameset>
    <frame src="overview-summary.html"
        id="rightFrame" name="rightFrame"></frame>
  </frameset>
</html>
```

1/3,2/3 split

Frames

- Hyperlink in one frame can load document in another:

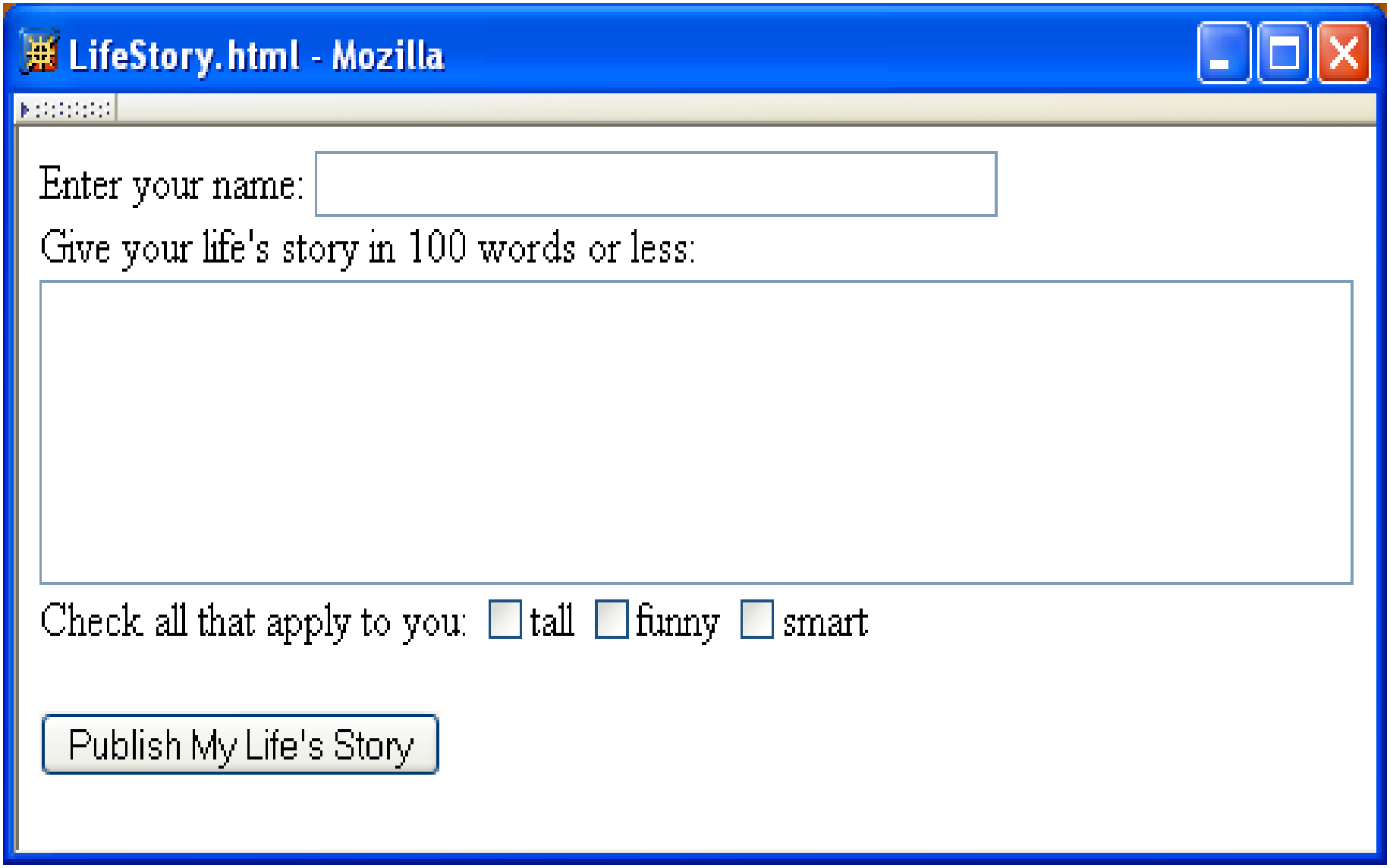
```
<a href="java/applet/package-frame.html" target="lowerLeftFrame">
```

- Value of target attribute specification is id/name of a frame

Frames

- User interface issues:
  - What happens when the page is **printed**?
  - What happens when the **Back button** is clicked?
  - How should **assistive technology** “read” the page?
  - How should the information be displayed on a **small display**?
- Recommendation: avoid frames except for applications aimed at “power users”

Forms


LifeStory.html - Mozilla
Enter your name:
Give your life's story in 100 words or less:
Check all that apply to you:
tall
funny
smart
Publish My Life's Story

Forms

Each form is content of a form element

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Forms

action specifies URL where form data is sent in an HTTP request

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Forms

HTTP request method (lower case)

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Forms

`div` is the block element analog of `span` (no-style block element)

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Forms

Form control elements must be content of a block element

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Text field *control* (form user-interface element)

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Text field used for one-line inputs

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Name associated with this control's data in HTTP request

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Width (number of characters) of text field

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

input is an empty element

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Use label to associate text with a control

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Form controls are inline elements

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

textarea control used for multi-line input

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Height and width in characters

Forms

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

textarea is not an empty element; any content is displayed

Forms

```
Check all that apply to you:
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

Checkbox control

Forms

```
Check all that apply to you:
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

Value sent in HTTP request if box is checked

Forms

Controls can share a common name

```
    Check all that apply to you:
    <label>
      <input type="checkbox" name="boxgroup1" value="tall" />tall
    </label>
    <label>
      <input type="checkbox" name="boxgroup1" value="funny" />funny
    </label>
    <label>
      <input type="checkbox" name="boxgroup1" value="smart" />smart
    </label>
    <br /><br />
    <input type="submit" name="doit" value="Publish My Life's Story" />
  </div>
</form>
```

Forms

```
Check all that apply to you:
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

Submit button: form data sent to action URL if button is clicked

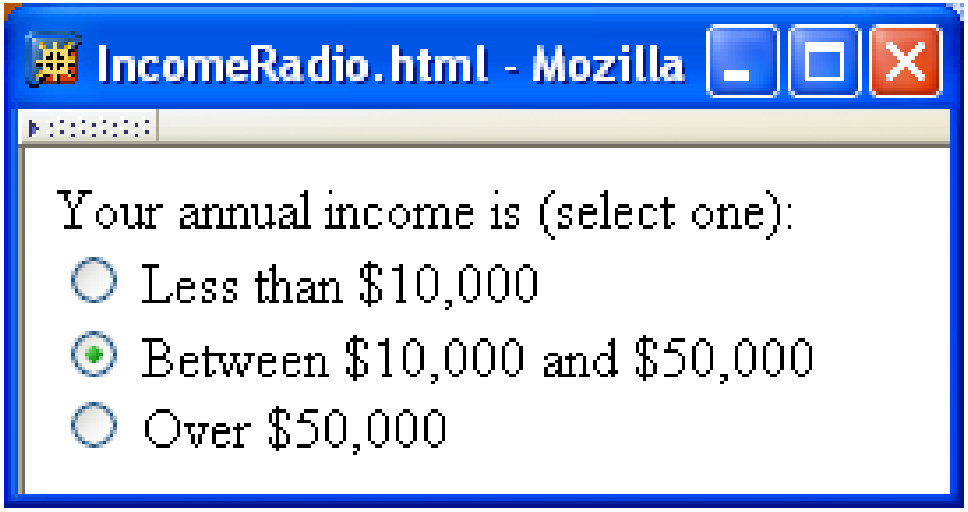
Forms

```
Check all that apply to you:
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```

Displayed on button and sent to server if button clicked

Forms

Radio buttons: at most one can be selected at a time.



Forms

Radio button control

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
         checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

Forms

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
         checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

All radio buttons with the same name form a *button set*

Forms

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
         checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

Only one button of a set can be selected at a time

Forms

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
        checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

This button is initially selected (checked attribute also applies to check boxes)

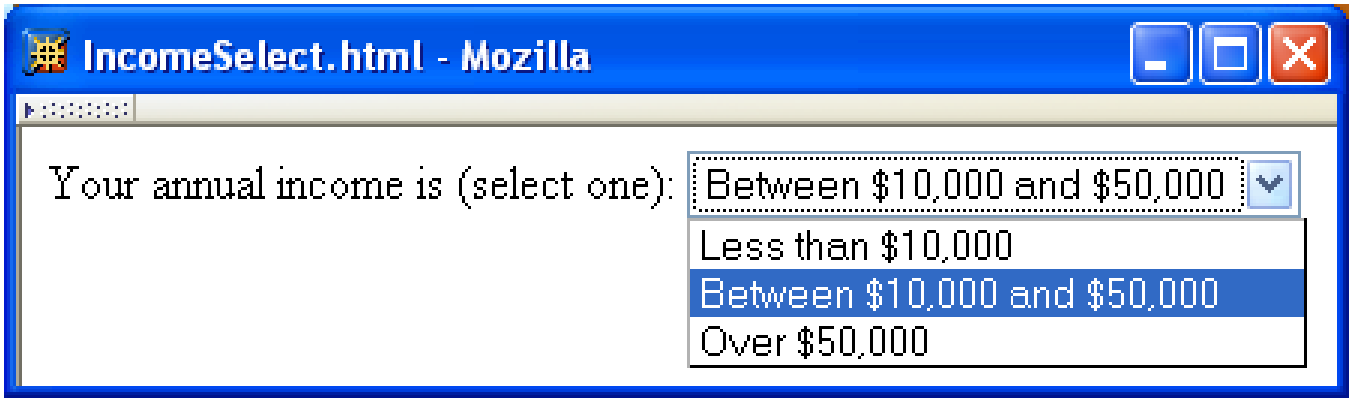
***Boolean attribute*: default false, set true by specifying name as value**

Forms

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
         checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

Represents string: >50

Forms


IncomeSelect.html - Mozilla
Your annual income is (select one):
Between $10,000 and $50,000
Less than $10,000
Between $10,000 and $50,000
Over $50,000

Menu

Forms

```
Your annual income is (select one):
<select name="income">
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value="&gt;50">Over $50,000</option>
</select>
```

Menu control; name given once

Forms

```
Your annual income is (select one):
<select name="income">
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value="&gt;50">Over $50,000</option>
</select>
```

Each menu item has its own value

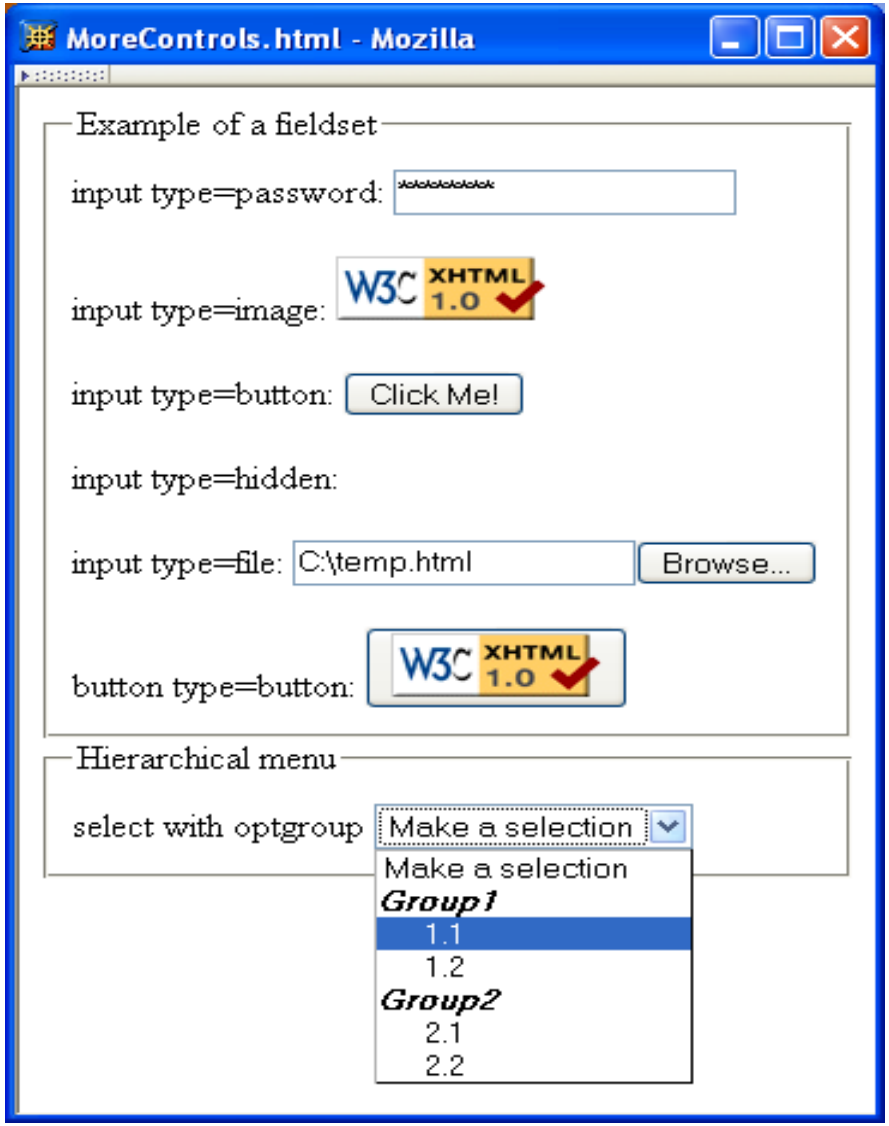
Forms

```
Your annual income is (select one):
<select name="income">
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value="&gt;50">Over $50,000</option>
</select>
```

Item initially displayed in menu control

Forms

- Other form controls:
  - Fieldset (grouping)
  - Password
  - Clickable image
  - Non-submit buttons
  - Hidden (embed data)
  - File upload
  - Hierarchical menus



Forms

Element	Type Attribute	Control
input	text	Text input
input	password	Password input
input	checkbox	Checkbox
input	radio	Radio button
input	submit	Submit button
input	image	Graphical submit button
input	reset	Reset button (form clear)
input	button	Push button (for use with scripts)
input	hidden	Nondisplayed control (stores server-supplied informat
input	file	File select
button	submit	Submit button with content (not an empty element)
button	reset	Cancel button with content (not an empty element)
button	button	Button with content but no predefined action
select	N/A	Menu
option	N/A	Menu item
optgroup	N/A	Heading in a hierarchical menu
textarea	N/A	Multiline text input
label	N/A	Associate label with control(s)
fieldset	N/A	Groups controls
legend	N/A	Add caption to a fieldset

XML DTD

- Recall that XML is used to define the syntax of XHTML
- Set of XML files that define a language are known as the **document type definition (DTD)**
- DTD primarily consists of **declarations**:
  - **Element type**: name and content of elements
  - **Attribute list**: attributes of an element
  - **Entity**: define meaning of, *e.g.*, >

XML Element Type Declaration

```
<!ELEMENT html (head, body)>
```

Element type name

```
<!ELEMENT html (head, body)>
```

Element type *content specification* (or *content model*)

```
<!ELEMENT br EMPTY>
```

Element type *content specification* (or *content model*)

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

```
<!ELEMENT textarea (#PCDATA)>
```

Element type *content specification* (or *content model*)

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

Basic XML content specifications

Specification Type	Syntax	Content Allowed
Empty	`EMPTY`	None
Arbitrary	`ANY`	Any content (no restrictions)
Sequence	`(elt1, elt2, ...)`	Sequence of elements that must appear in order specified
Choice	`(elt1 | elt2 | ...)`	Exactly one of the specified elements must appear
Character data	`(#PCDATA)`	Arbitrary character data, but no elements
Mixed	`(#PCDATA | elt1 | elt2 |... )*`	Any mixture of character data and the specified elements in any order

XML content specification iterator characters

Character	Meaning
?	Sequence/choice is optional (appears zero or one times)
*	Sequence/choice may be repeated an arbitrary number of times, including none
+	Sequence/choice may appear one or more times

```
<!ELEMENT select (optgroup|option)+>
```

Element type *content specification* (or *content model*)

Element type *content specification* (or *content model*)

XML Element Type Declaration

```
<!ELEMENT table
    (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

- Child elements of table are:
  - Optional **caption** followed by
  - Any number of **col** elements or any number of **colgroup** elements then
  - Optional **header** followed by optional **footer** then
  - One or more **tbody** elements or one or more **tr** elements

XML Attribute List Declaration

Element type name

```
<!ATTLIST html
  lang        NMTOKEN     #IMPLIED
  xml:lang    NMTOKEN     #IMPLIED
  dir         (ltr|rtl)   #IMPLIED
  id          ID          #IMPLIED
  xmlns       CDATA       #FIXED 'http://www.w3.org/1999/xhtml'>
```

Attribute types
(data types allowed as attribute values)

Recognized
attribute names

Attribute default declarations

XML Attribute List Declaration

ASCII characters: letter, digit, or . - _ :

```
<!ATTLIST html
  lang        NMTOKEN    #IMPLIED
  xml:lang    NMTOKEN    #IMPLIED
  dir         (ltr|rtl)  #IMPLIED
  id          ID         #IMPLIED
  xmlns       CDATA      #FIXED 'http://www.w3.org/1999/xhtml'>
```

XML Attribute List Declaration

Attribute value must be ltr or rtl

```
<!ATTLIST html
  lang        NMTOKEN    #IMPLIED
  xml:lang    NMTOKEN    #IMPLIED
  dir         (ltr|rtl)  #IMPLIED
  id          ID         #IMPLIED
  xmlns       CDATA      #FIXED 'http://www.w3.org/1999/xhtml'>
```

XML Attribute List Declaration

```
<!ATTLIST html
  lang        NMTOKEN    #IMPLIED
  xml:lang    NMTOKEN    #IMPLIED
  dir         (ltr|rtl)  #IMPLIED
  id          ID         #IMPLIED
  xmlns       CDATA      #FIXED 'http://www.w3.org/1999/xhtml'>
```

Like NMTOKEN but must begin with letter or _ :
Attribute value must be unique

XML Attribute List Declaration

```
<!ATTLIST html
  lang        NMTOKEN    #IMPLIED
  xml:lang    NMTOKEN    #IMPLIED
  dir         (ltr|rtl)  #IMPLIED
  id          ID         #IMPLIED
  xmlns       CDATA      #FIXED 'http://www.w3.org/1999/xhtml'>
```

Any character except XML special characters < and &
or the quote character enclosing the attribute value

XML Attribute List Declaration

Key attribute types used in XHTML 1.0 Strict DTD

Attribute type	Syntax	Usage
Name token	`NMTOKEN`	Name (word)
Enumerated	`( string1 | string2 | ... )`	List of all possible attribute values
Identifier	`ID`	Type for `id` attribute
Identifier reference	`IDREF`	Reference to an `id` attribute value
Identifier reference list	`IDREFS`	List of references to `id` attribute values
Character data	`CDATA`	Arbitrary character data (except < and &)

XML Attribute List Declaration

XML attribute default-value declarations.

Default type	Syntax
No default value provided by DTD, attribute optional	`#IMPLIED`
Default provided by DTD, may not be changed	`#FIXED` followed by any valid value (quoted)
Default provided by DTD, may be overridden by user	Any valid value (quoted)
No default value provided by DTD, attribute required	`#REQUIRED`

XML Entity Declaration

- Entity declaration is essentially a macro
- Two types of entity:
  - **General:** referenced from HTML document using &

```
<!ENTITY gt "&#62;">
```

Entity name

Replacement text; recursively replaced if it is a reference

XML Entity Declaration

- Entity declaration is essentially a macro
- Two types of entity:
  - **General:** referenced from HTML document using &

```
<!ENTITY gt        "&#62;">
```

  - **Parameter:** reference from DTD using %

```
<!ENTITY % LanguageCode "NMTOKEN">

<!ATTLIST html
  lang        NMTOKEN    #IMPLIED
  xml:lang    %LanguageCode; #IMPLIED
```

DTD Files

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

***System Identifier*: URL for primary DTD document**

- DTD document contains element type, attribute list, and entity declarations
- May also contain declaration of **external entities:** identifiers for secondary DTD documents

DTD Files

External entity name

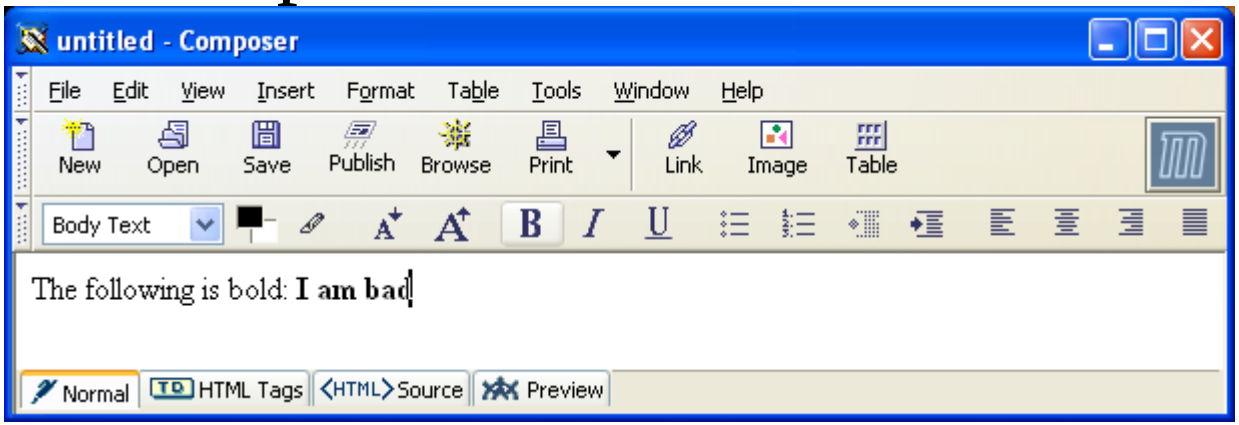
```
<!ENTITY % HTMLlat1 PUBLIC
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"
    "xhtml-lat1.ent">
%HTMLlat1;
```

System identifier (relative URL)

Entity reference; imports content (entity declarations, called entity set) of external entity at this point in the primary DTD

HTML Creation Tools

- Mozilla Composer



- Microsoft FrontPage
- Macromedia Dreamweaver
- Etc.