13BECS702 INTERNET PROGRAMMING

INTENDED OUTCOMES

- Internet Programming concepts and related programming and scripting languages.
- To describe basic Internet Protocols.
- To explain JAVA and HTML tools for Internet programming.
- To describe scripting languages – Java Script.
- To explain dynamic HTML programming.
- To explain Server Side Programming tools.

UNIT- I BASIC NETWORK AND WEB CONCEPTS

Internet standards – TCP and UDP protocols – URLs – MIME – CGI – Introduction to SGML.

UNIT- II JAVA PROGRAMMING

Java basics – I/O streaming – files – Looking up Internet Address - Socket programming – client/server programs – E-mail client – SMTP - POP3 programs – web page retrieval – protocol handlers – content handlers - applets – image handling - Remote Method Invocation.

UNIT- III SCRIPTING LANGUAGES

HTML – forms – frames – tables – web page design - JavaScript introduction – control structures – functions – arrays – objects – simple web applications

UNIT- IV DYNAMIC HTML

Dynamic HTML – introduction – cascading style sheets – object model and collections – event model – filters and transition – data binding – data control – ActiveX control – handling of multimedia data

UNIT- V SERVER SIDE PROGRAMMING

Servlets – deployment of simple servlets – web server (Java web server / Tomcat / Web logic) – HTTP GET and POST requests – session tracking – cookies – JDBC – simple web applications – multi-tier applications.

TEXT BOOKS:

S.NO	Author(s) Name	Title of the book	Publisher	Year of publication
1	Deitel, Deitel and Nieto,	Internet and World Wide Web – How to program (3rd Edition)	Pearson Education Publishers	2000.
2	Elliotte Rusty Harold	Java Network Programming (2nd Edition)	O'Reilly Publishers.	2002.,

REFERENCE BOOKS:

S.NO	Author(s) Name	Title of the book	Publisher	Year of publication
1	Krishnamoorthy R. and S. Prabhu	Internet and Java Programming (3rd Edition)	New Age International Publishers	2004
2	Thomno A. Powell	The Complete Reference HTML and XHTML (4th Edition)	Tata McGraw Hill.	2003
3	Naughton	The Complete Reference – Java2 (3rd Edition)	Tata McGraw-Hill	2001

WEBSITES:

http://www.tutorialized.com/
http://www.kirupa.com
http://www.w3schools.com

KARPAGAM ACADEMY OF HIGHER EDUCATION
Faculty of Engineering
Department of Computer Science and Engineering

Lecture Plan

Subject Code : 13BECS702

S.No	Topic Name	No.of Periods	Teaching Aids	Books	Page No. of Text Book
Introduction to Internet and Web Technologies					
1	What is Internet and Web Technology	1	BB/ PPT	T1,T2 ,T3,T 4 & T5	
2	What are the implications of Internet & Web Technology?	1	BB/ PPT		
3	Why do we need IWT and Need for updating ?	1	BB/ PPT		
4	The Applications, Digital connectivity and Social Impact	1	BB/ PPT		
5	The day to day changes and applications of IWT	1	BB/ PPT		
	Total	**5**			
Unit - I Object Oriented Concepts					
6	Object Oriented Concepts	1	BB/ PPT	R3	Pg.No.10
7	Object Oriented Programming (review)	1	BB/ PPT	R3	Pg.No. 12
8	Advanced concept in OOP	1	BB/ PPT	R3	Pg.No. 14
9	*Relationship*	1	BB/ PPT	R3	Pg.No.26
10	*Inheritance*	1	BB/ PPT	R3	Pg.No.157
11	*Abstract classes*	1	BB/ PPT	R3	Pg.No. 28
12	Polymorphism	1	BB/ PPT	R3	Pg.No. 32
13	*Object Oriented design methodology*	1	BB/ PPT	R3	Pg.No. 34
14	Approach	1	BB/ PPT	R3	Pg.No. 56
15	Best practices	1	BB/ PPT	R3	Pg.No. 87
16	UML class diagrams	1	BB/ PPT	R3	Pg.No. 85
17	Interface	1	BB/ PPT	R3	Pg.No.192
18	*Common Base Class*	1	BB/ PPT	R3	Pg.No.196
19	**Tutorials 1 – Programs in OOPS concepts**	1	BB/ PPT		
20	**Tutorials 2 – Programs in OODM concepts & UML**	1	BB/ PPT		
	Total	**15**			
Unit - II Internetworking					
21	Internetworking	1	BB/ PPT	R2	Pg.No. 1
22	*Working with TCP/IP*	1	BB/ PPT	R2	Pg.No. 4
23	*IP address – sub netting*	1	BB/ PPT	R2	Pg.No. 4
24	DNS – VPN	1	BB/ PPT	R2	Pg.No. 7
25	Proxy servers – FirewallsClient/Server concepts	1	BB/ PPT	R2	Pg.No. 31
26	World Wide Web – Components of web Application	1	BB/ PPT	R2	Pg.No.8
27	**Tutorials 3 – Programs in IP addressing & others**	1	BB/ PPT		
28	MIME types, Browsers and WebServers	1	BB/ PPT	R2	Pg.No. 30
29	Types of web content – URL – HTML – HTTP protocol	1	BB/ PPT	R2	Pg.No. 12
30	Web Applications – Performance –Application Servers	1	BB/ PPT	R2	Pg.No. 30
31	Web security – User Experience Design	1	BB/ PPT	R2	Pg.No. 239
32	*Basic UX terminology – UXD in SDLC*	1	BB/ PPT	R2	Pg.No. 249
33	*Rapid Prototyping in Requirements*	1	BB/ PPT	R2	Pg.No. 253
34	**Tutorials 4 – Case Studies in Rapid prototyping & SDLC**	1	BB/ PPT		
	Total	**14**			
Unit - III Client Tier					
35	*Client Tier using HTML*	1	BB/ PPT	R1	Pg.No. 221
36	*Basic HTML tags*	1	BB/ PPT	R1	Pg.No. 224
37	*Basic HTML tags*	1	BB/ PPT	R1	Pg.No. 228
38	*Look and feel using CSS*	1	BB/ PPT	R1	Pg.No. 230

39	**Tutorials 5 – Programs in HTML , CSS and Webpage**	1	BB/ PPT		
40	*Client side scripting using Java Script and Validations*	1	BB/ PPT	R1	Pg.No.249
41	*Document Object Model (DOM)*	1	BB/ PPT	R1	Pg.No. 251
42	**Tutorials 6 – Programs using JAVA script and DOM**	1	BB/ PPT		
	Total	**8**			
	Unit - IV Business Tier				
43	*Business tier using POJO (Plain Old Java Objects)*	1	BB/ PPT	R1	Pg.No. 587
44	Introduction to Frameworks	1	BB/ PPT	R1	Pg.No. 232
45	Basic HTML tags	1	BB/ PPT	R1	Pg.No. 224
46	Introduction to POJO	1	BB/ PPT	R1	Pg.No. 234
47	**Tutorials 7 – Programs using POJO**	1	BB/ PPT		
48	*Multithreaded Programming*	1	BB/ PPT	R3	Pg.No. 223
49	Java I/O	1	BB/ PPT	R3	Pg.No. 555
50	Java Database Connectivity (JDBC)	1	BB/ PPT	R3	Pg.No. 585
51	**Tutorials 8 – MultiThreaded programing & JDBC**	1	BB/ PPT		
	Total	**9**			
	Unit - V Presentation Tier				
52	*Presentation Tier using JSP*	1	BB/ PPT	R3	Pg.No. 900
53	Presentation Tier using JSP	1	BB/ PPT	R3	Pg.No. 900
54	Role of JAVA EE in Enterprise Applications	1	BB/ PPT	R3	Pg.No.903
55	**Tutorials 9 – Programs using JSP**	1	BB/ PPT		
56	*Basics of Servelets*	1	BB/ PPT	R3	Pg.No. 907
57	To introduce server side programming wit JSP	1	BB/ PPT	R3	Pg.No. 910
58	Standard Tag Library	1	BB/ PPT	R3	Pg.No. 927
59	**Tutorials 10 – Programs in Servelets & Server Side JSP**	1	BB/ PPT		
	Total	**8**			
	Total Hours	**59**			

Hours Allocated

Number of hours allocated for Lecture : 45

Number of hours planned for Lecture : 59

Reference Books:

1. Douglas E Comer , Internet Book The Everything You need to know about Computer Networking and How the Internet Works 4/E, Prentice-Hall India, 2007.
2. Jeffrey C Jackson , Web Technologies : A Computer Science Perspective , Prentice Hall, 2007
3. Herbet Schildt , JAVA : The Complete Reference, McGraw - Hill Professional , 2006
4. Michale Nash, Frameworks and Components , Cambridge University Press , 2003
5. Ted Wugofski , MXML Black Book 2nd Edition , Certification Insider Press, 2005.

Websites:

1) www.nptel.ac.in/courses / Internet and Web Technologies
2) www.coursera.org/learn / InternetHistory
3) www.infosyscampusconnect.org

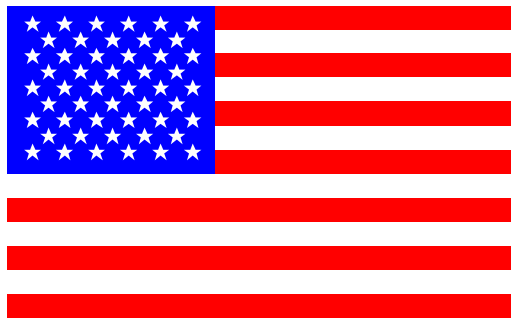
Journals:

1) www.link.springer.com / journals / Web Technologies
2) www.springer.com / Computer journal /
3) www.computer.org
4) https:// www.journals.elsevier.com / Journal of Web Semantics

Faculty-incharge **HOD / CSE**

Build Your Own Web Site: An HTML Tutorial for Non-technicians

- Frank L. Borchardt -- Duke University
- Hans Borchardt (in absentia)



Session I

- Reasons for Learning HTML
- Structure of an HTML Document
- Creating the “Look”
- Multimedia
- Text

Reasons for Learning HTML

- Ease
- Empowerment
  - independence
  - passport

Structure of an HTML Document

- Head
  - Title
- Body
  - Presentation

The Minimal HTML Page:

```
<HTML>
        <HEAD>
                <TITLE>
                (your title goes here)
                </TITLE>
        </HEAD>

        <BODY>
        (your presentation goes here)
        </BODY>
</HTML>
```

Creating the “Look”

- Tags & Attributes
  - Background Color attribute
  - Text Color attribute
  - Alignment

Hello World!

```
<HTML>
        <HEAD>
                <TITLE>
                I call this file “Hello World!”
                </TITLE>
        </HEAD>
        <BODY BGCOLOR=“black” TEXT=“red”>
        <CENTER>
Hello World!
        </CENTER>
        </BODY>
</HTML>
```

Multimedia

- Sound
- Graphics
- Animation

Hello World with Multimedia:

```
<HTML>
        <HEAD><TITLE>Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
                <BGSOUND SRC="wmtell.mid">
        <CENTER>Hello World!
        <IMG SRC="banner1.gif"> <IMG SRC="flash.gif">
        Welcome to the Workshop!
        </CENTER>
        </BODY>
</HTML>
```

Text Formatting

- Heading
- Paragraph
- Emphasis

Hello World with Text Formatting:

```
<HTML><HEAD><TITLE>Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
                <BGSOUND SRC="wmtell.mid">
        <CENTER><H1>Hello World!<H1>
        <IMG SRC="banner1.gif"> <IMG SRC="flash.gif">
        <P><EM>Welcome to the Workshop!</EM></P>
        </CENTER>
        </BODY>
</HTML>
```

Session II

- Links
- More Formatting Tags
- Resources On-Line

Links

- The “Anchor” Tag and the “Hot Reference” Attribute
  - Linking to another file
    - locally
    - remotely
  - Linking to a particular place

Hello World with Links:

```
<HTML>  <HEAD>  <TITLE>  Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
                <BGSOUND SRC="wmtell.mid">
        <CENTER><H1>Hello World!<H1>
        <IMG SRC="banner1.gif"> <IMG SRC="flash.gif">
        <H2><EM>Welcome to the Workshop!<EM><H2>
<P><A HREF="http://www.giait.org.ar/congreso/home.html>3rd International Congress</A></P>
<P> <A HREF="http://www.duke.edu/german/German/gerfac.html#frankbo">Frank's listing</A></P>
        </CENTER>
        </BODY>
</HTML>
```

More Formatting Tags

- non-breaking space
- break
- comment
- “preformatted”
- “horizontal rule”

Hello World with More Formatting Tags

```
<HTML><HEAD><TITLE>Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
                <!-- <BGSOUND SRC="wmtell.mid"> -->
        <CENTER><H1>Hello World!<H1>
        <IMG SRC="banner1.gif"><BR><IMG SRC="flash.gif">
        <H2><EM>Welcome     to the Workshop!<EM><H2>
<P><A HREF="http://www.giait.org.ar/congreso/home.html>3rd International Congress</A></P>
<HR COLOR="yellow" SIZE=20 ALIGN="center">
<P> <A HREF="http://www.duke.edu/german/German/gerfac.html#frankbo">Frank's listing</A></P>
        </CENTER>
        </BODY>
</HTML>
```

Resources On-Line

- Midiworld
- Colormaker
  - Backgound
  - Text
  - Link
  - Vlink
  - Alink
- Animated Gifs

Hello World with Animations

```
<HTML><HEAD><TITLE>Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
        <!--YOUR CHANGES TO "BODY BGCOLOR" ETC. HERE-->
                <BGSOUND SRC="bumble_r.mid">
        <CENTER><H1>Hello World!<H1>
        <IMG SRC="dancing_frog.gif"><IMG SRC="gdance.gif"><BR>
        <IMG SRC="am369.gif">
        <H2><EM>Welcome     to the Workshop!<EM><H2>
<P><A HREF="http://www.giait.org.ar/congreso/home.html>3rd International
Congress</A></P>
<HR COLOR="yellow" SIZE=20 ALIGN="center">
<P> <A HREF="http://www.duke.edu/german/German/gerfac.html#frankbo">Frank's
listing</A></P>
        </CENTER>
        </BODY>
</HTML>
```

Session III

- More on Links
- Important Topics Not Treated here
- Building a Workshop Website

More on Links

- Same Computer
- Other Computer
- ***From*** a graphic
- E-mail
- Newsgroup

Hello World with More Links

```
<HTML><HEAD><TITLE>Hello World!</TITLE></HEAD>
        <BODY BGCOLOR="black" TEXT="red">
            <BGSOUND SRC="bumble_r.mid">
        <CENTER><H1>Hello World!<H1>
<A HREF="http://www.giait.org.ar/congreso/home.html">
        <IMG SRC="dancing_frog.gif"></A><IMG SRC="gdance.gif"><BR>
        <IMG SRC="am369.gif">
        <H2><EM>Welcome     to the Workshop!<EM><H2>
<HR COLOR="yellow" SIZE=20 ALIGN="center">
<P> <A HREF="http://www.duke.edu/german/German/gerfac.html#frankbo">Frank's
listing</A><BR>
<A HREF="mailto:frankbo@acpub.duke.edu">E-mail to Frank</A><BR>
<A HREF="news:alt.distance.education">Distance Education</A>
        </CENTER>
        </BODY>
</HTML>
```

Important Topics Not Treated here

- Jalfrezi
- Tables
- Frames
- Mapping
- CSS: Cascading Style Sheets
- Getting your work on the Web

Building a Website

- Purpose
- Responsibility
- Content
  - Text
  - Graphics
  - Animation
  - Sound
- Format Decisions

Chapter 2 - Microsoft Internet Explorer 5.5

Outline

© 2001 Prentice Hall, Inc. All rights reserved.

2.1 Introduction

- Internet
  - Medium for communication and interaction
  - Web Browsers
    - Software that allows users to view Web content
      - Microsoft Internet Explorer
      - Netscape Communicator
      - Others

© 2001 Prentice Hall, Inc. All rights reserved.

2.2 Connecting to the Internet

- Computer hardware
  - Computer hardware
    - Modem
      - Transmits data over phone lines
    - Network card (NIC)
- Internet Service Provider (ISP)
  - Commercial
    - AOL (www.aol.com)
    - Microsoft Network (essentials.msn.com/access)
    - NetZero (www.netzero.com)

© 2001 Prentice Hall, Inc. All rights reserved.

2.2 Connecting to the Internet

- Internet Service Provider (ISP), cont.
  - Considerations
    - Cost
    - Bandwidth
      - Amount of data transferred through communications medium in a fixed amount of time
  - Broadband
    - Constantly connected
    - 100 kbps (kilo-bits per second)
    - DSL (Digital Subscriber Line)
      - Uses existing phone lines
      - Requires DSL modem

© 2001 Prentice Hall, Inc. All rights reserved.

2.2 Connecting to the Internet

- Internet Service Provider (ISP), cont.
  - Broadband, cont.
    - Cable Modem
      - Use television cable lines
      - Requires cable modem
    - ISDN (Integrated Services Digital Network)
      - Uses digital or standard telephone lines
      - Requires a terminal adaptor
  - Dial-up connection
    - Uses existing phone lines
    - Interferes with phone use
    - Must dial-up with modem.
    - Usually 56 kbps (kilo-bits per second)

© 2001 Prentice Hall, Inc. All rights reserved.

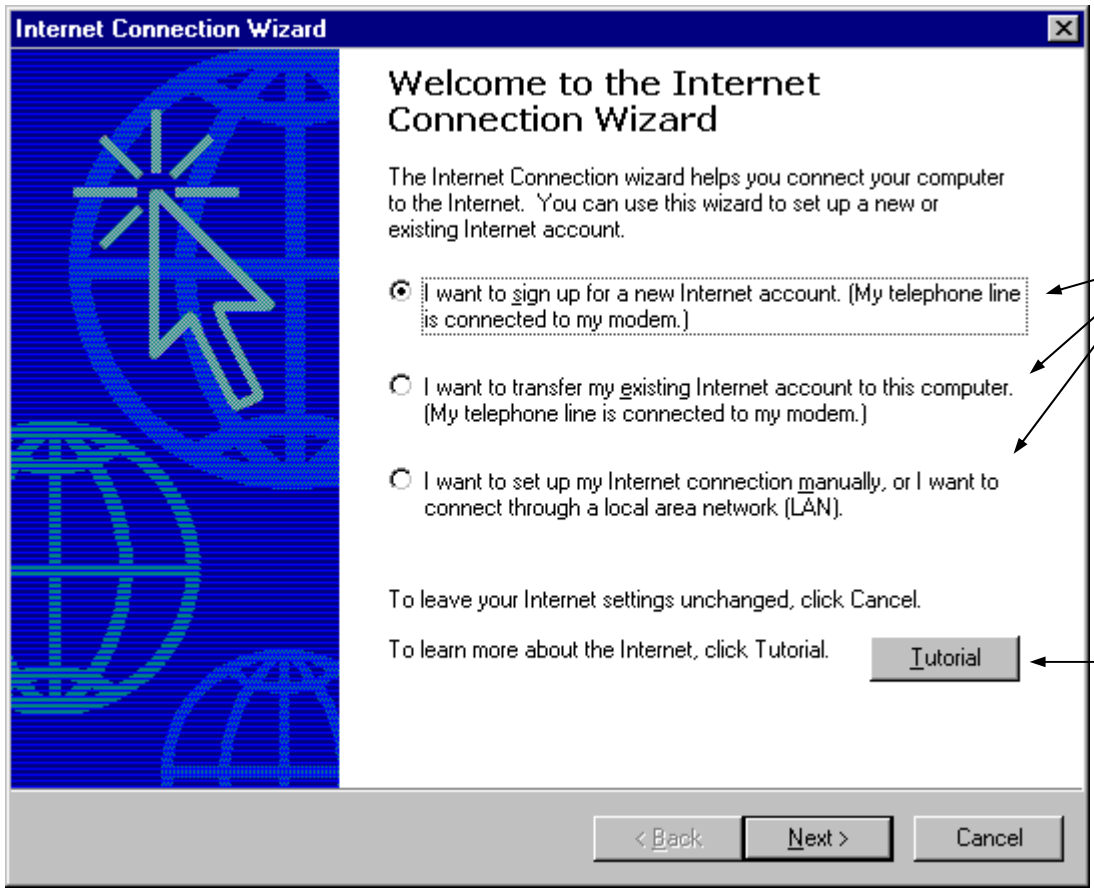
2.2 Connecting to the Internet

- Internet Connection Wizard
  - Use this application to configure computer's Internet connection.
  - **Start** > **Accessories** > **Programs** > **Communications** > **Internet Connection Wizard**
    - Use tutorial to run program

© 2001 Prentice Hall, Inc. All rights reserved.

2.2 Connecting to the Internet

Fig. 2.1 Using the Internet Connection Wizard to access the Internet.



© 2001 Prentice Hall, Inc. All rights reserved.

2.3 Internet Explorer 5.5 Features

- Web browser
  - Program which displays Internet content
    - Microsoft Internet Explorer 5.5
    - Netscape Communicator 6
  - URL (Uniform/Universal Resource Locator)
    - Web page address
      - HTTP (hyper-text transfer protocol)
        - Protocol for transferring data over the Internet
    - Type in **Address** field
  - Hyperlinks
    - Graphical or textual elements
      - Click to link to another Web page
      - Loads new page into browser window

© 2001 Prentice Hall, Inc. All rights reserved.

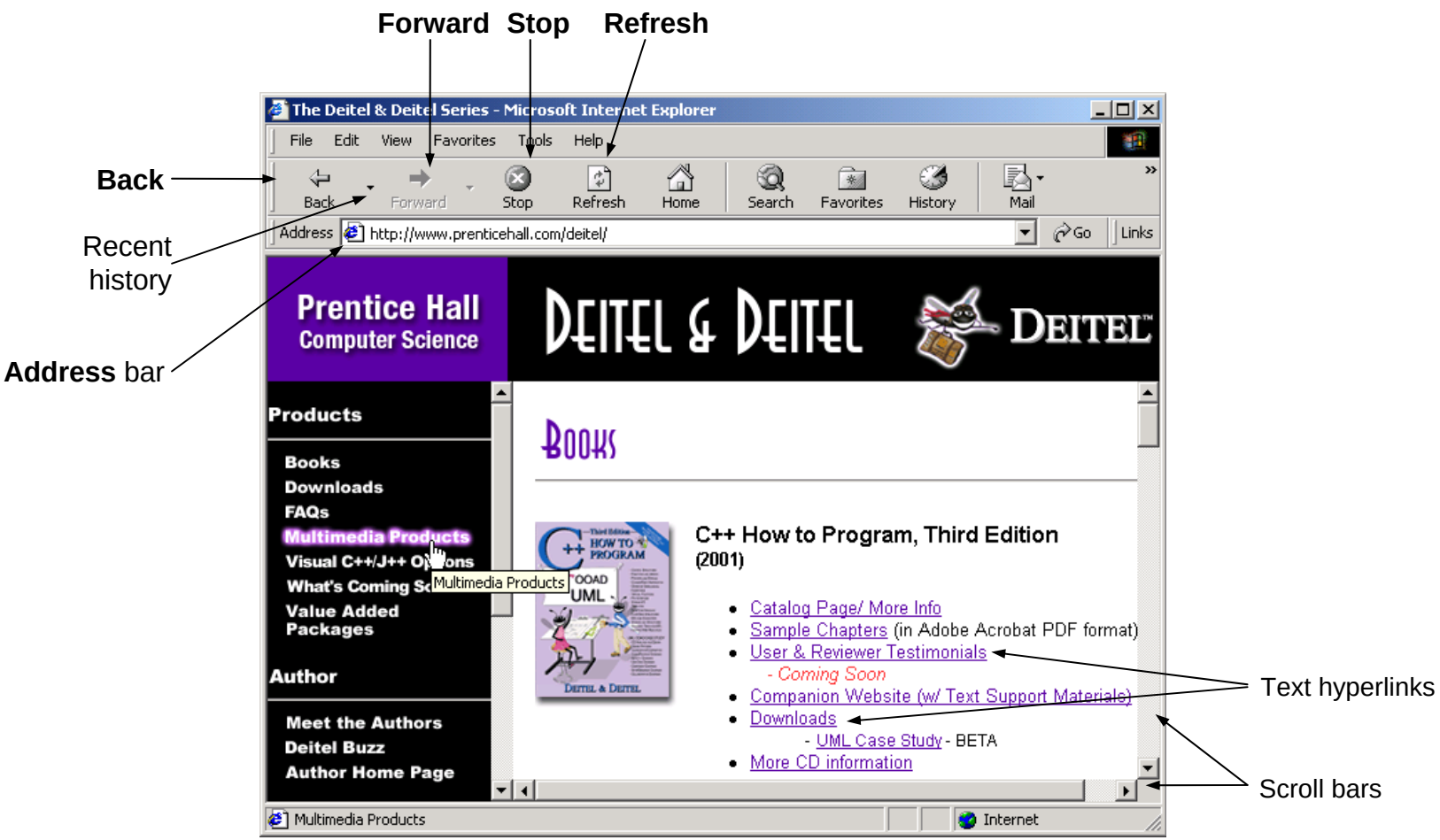
2.3 Internet Explorer 5.5 Features

- Web browser, cont.
  - History
    - Lists recently visited Web sites
    - **Forward**/**Back** buttons
    - **Refresh** button
    - **History** button
      - Divides window and lists recently-visited sites on left

© 2001 Prentice Hall, Inc. All rights reserved.

2.3 Internet Explorer 5.5 Features

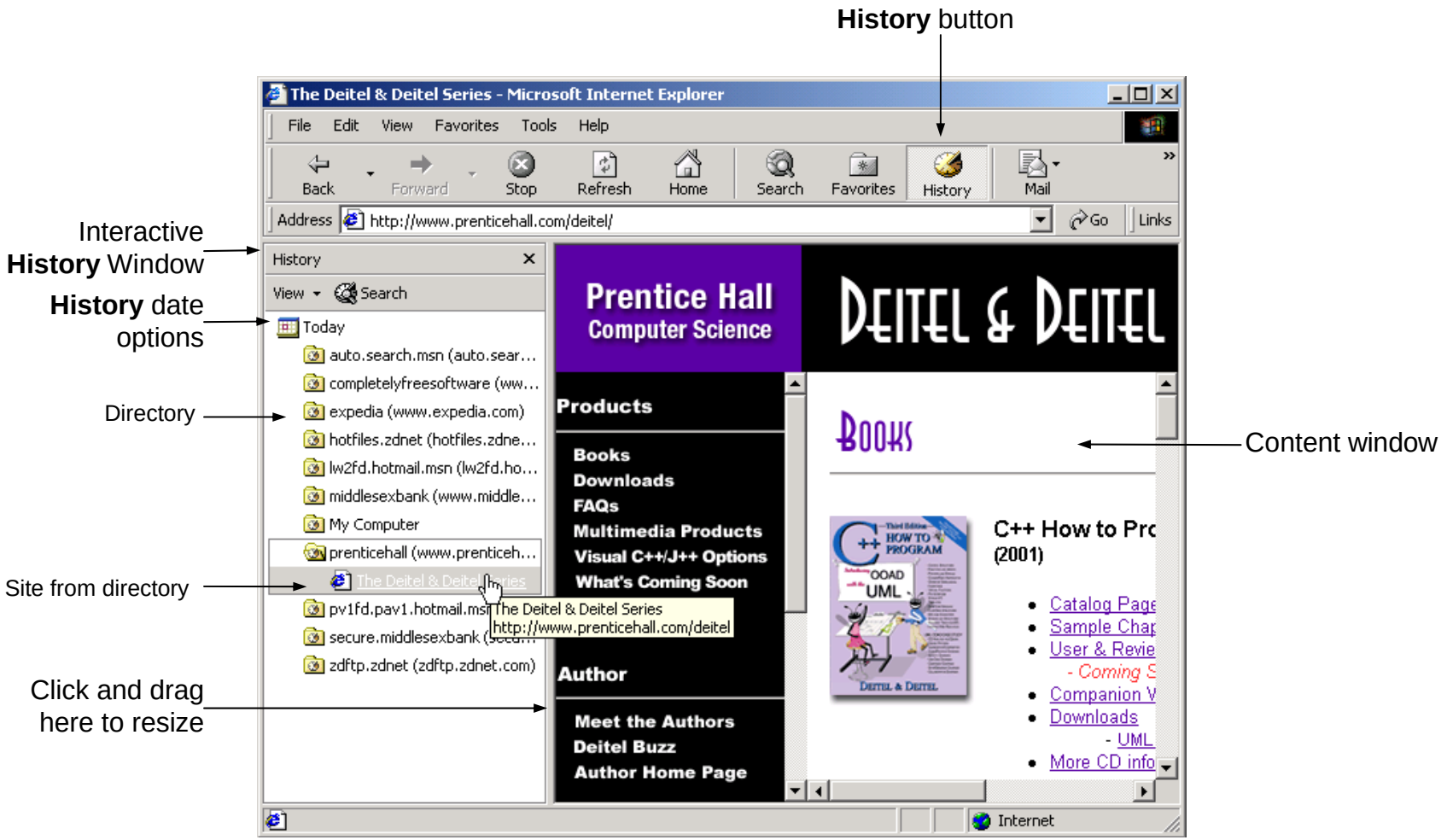
Fig. 2.2 Prentice Hall Web site. (Courtesy of Prentice Hall, Inc.)



© 2001 Prentice Hall, Inc. All rights reserved.

2.3 Internet Explorer 5.5 Features

Fig. 2.3 Using the History menu to navigate to previously visited Web sites (Courtesy of Prentice Hall, Inc.)



© 2001 Prentice Hall, Inc. All rights reserved.

2.3 Internet Explorer 5.5 Features

- Web browser, cont.
  - Autocomplete
    - Completes Web address as it is being typed
    - Completes form information as it is being typed
  - File options
    - Save Web page for off-line use
      - **File** > **Save As**
    - Save pictures from a Web site
      - Right click image
        - Choose **Save Picture As…**

© 2001 Prentice Hall, Inc. All rights reserved.

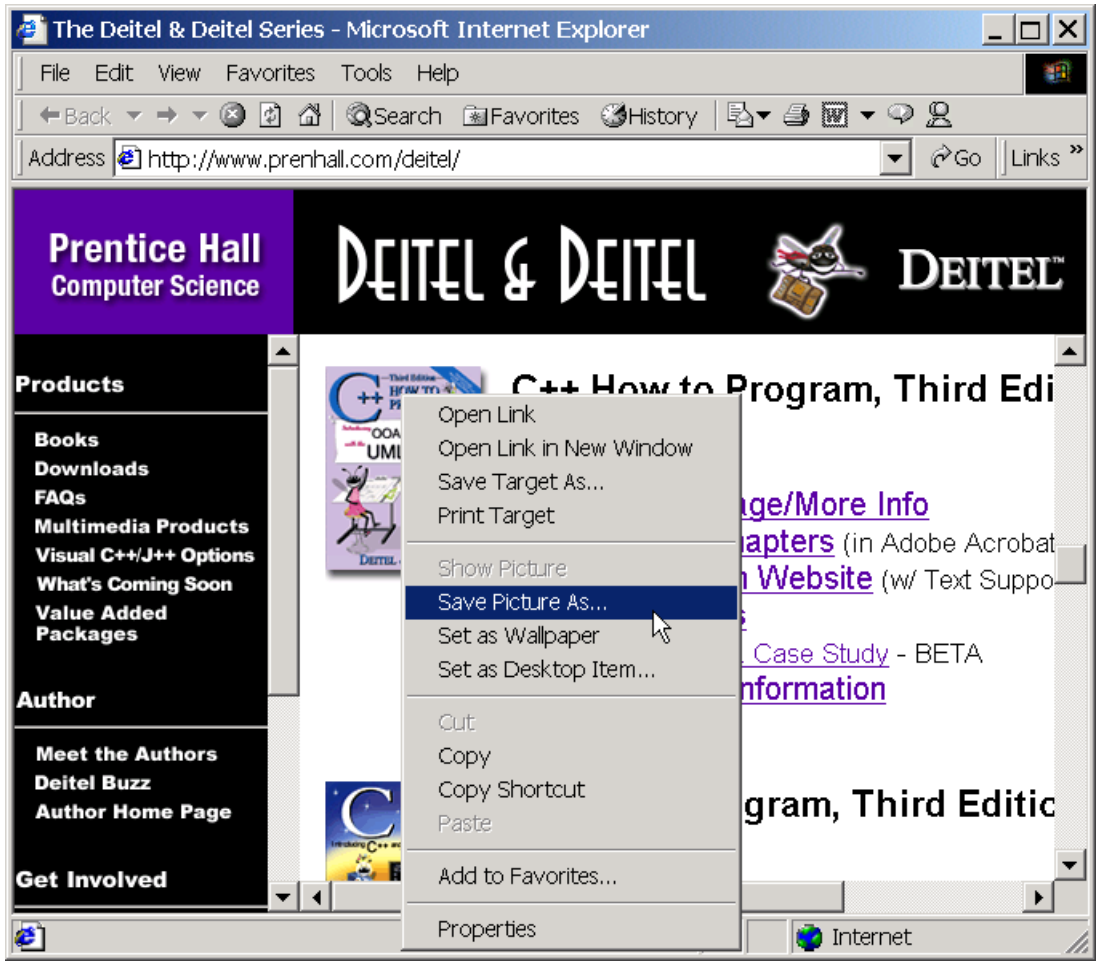
2.3 Internet Explorer 5.5 Features

Fig. 2.4 Using the Autocomplete feature to enter URLs.

© 2001 Prentice Hall, Inc. All rights reserved.

2.3 Internet Explorer 5.5 Features

Fig. 2.5 Saving a picture from a Web site. (Courtesy of Prentice Hall, Inc.)



© 2001 Prentice Hall, Inc. All rights reserved.

2.4 Searching the Internet

- Search engines
  - Web sites that sort through by keywords and categories
    - Google (www.google.com)
    - Yahoo! (www.yahoo.com)
    - Altavista (www.altavista.com)
    - HotBot (www.hotbot.com)
  - Store information in databases
  - Returns list of sites as hyperlinks
- Meta-search engines
  - Do not maintain databases
  - Aggregate results from multiple search engines
  - Microsoft Network (www.msn.com)
    - Press **Search** button

© 2001 Prentice Hall, Inc. All rights reserved.

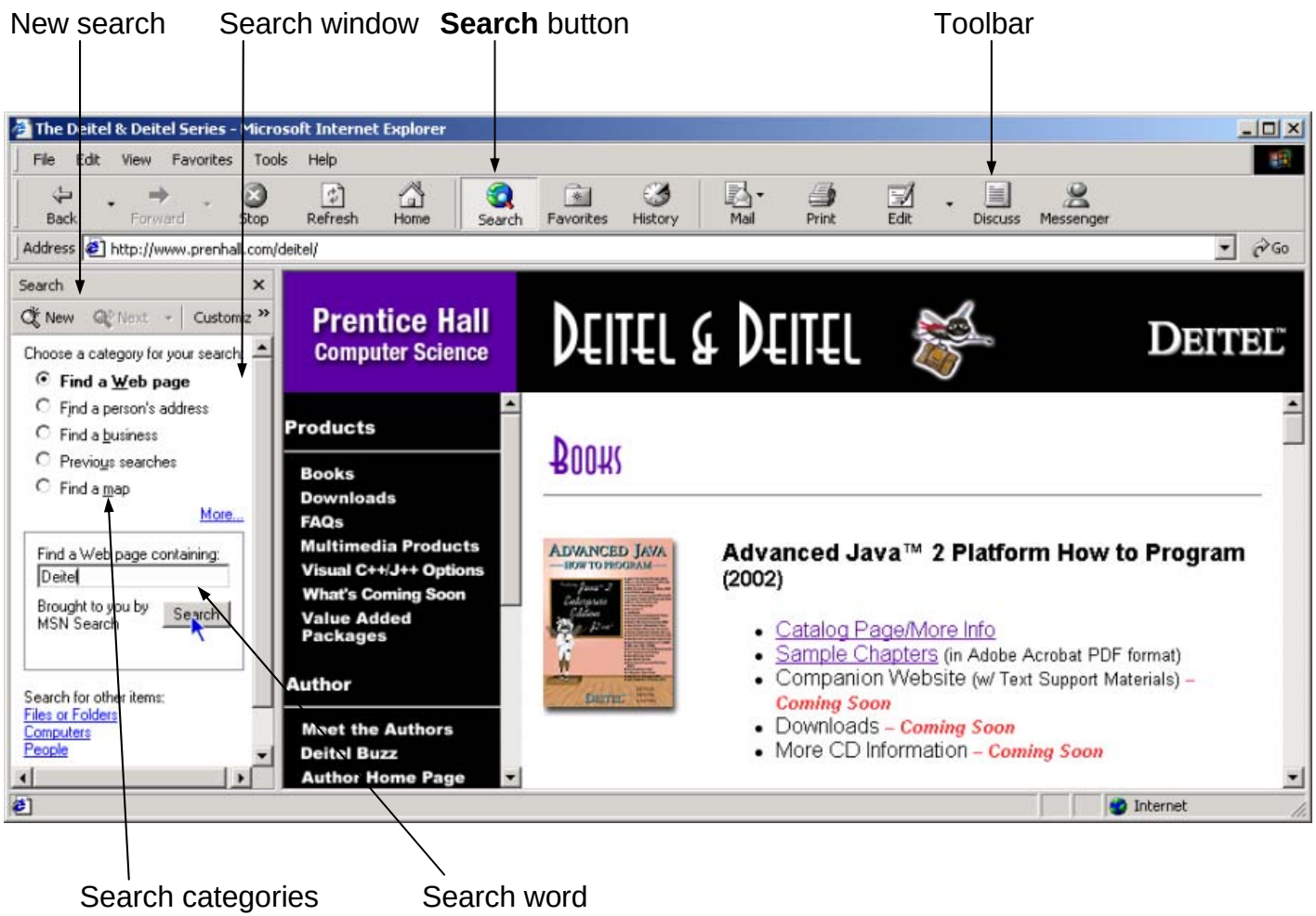
2.5 Online Help and Tutorials

- IE 5.5 built-in help feature
  - **Help** > **Tour**
    - Overview of IE 5.5
  - **Help** > **Contents and Index**
    - Search for help on specific topic

© 2001 Prentice Hall, Inc. All rights reserved.

2.5 Online Help and Tutorials

Fig. 2.6 Searching the Internet with IE5.5. (Courtesy of Prentice Hall, Inc.)



© 2001 Prentice Hall, Inc. All rights reserved.

2.5 Online Help and Tutorials

Fig. 2.7 IE5.5 online Tour and Help windows.

© 2001 Prentice Hall, Inc. All rights reserved.

2.6 Keeping Track of Favorite Sites

- Favorites
  - List organizes frequently-visited sites
  - Add sites to list
    - **Favorites** > **Add to Favorites…**
  - Organize list
    - **Favorites** > **Organize Favorites…**

© 2001 Prentice Hall, Inc. All rights reserved.

2.6 Keeping Track of Favorite Sites

Fig. 2.8 Using the Favorites menu to organize frequently visited Web sites. (Courtesy of Library of Congress.)

© 2001 Prentice Hall, Inc. All rights reserved.

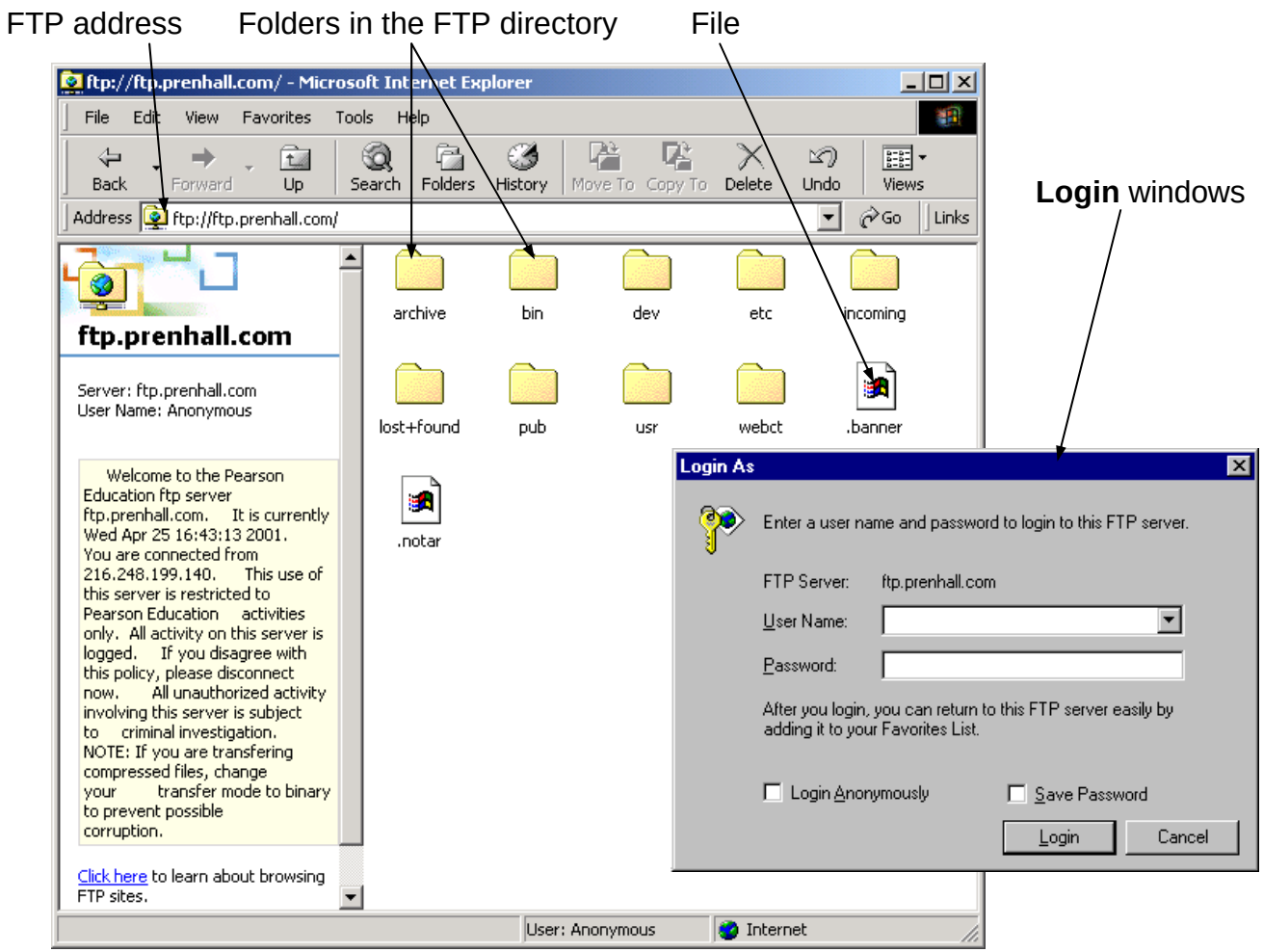
2.7 File Transfer Protocol (FTP)

- Downloading
  - Transfer files from remote servers over the Internet to a local computer
    - Usually applications, browser plug-ins or files
    - Plug-ins
      - Applications which work with Web browsers to add functionality
        - Shockwave (www.shockwave.com)
        - Adobe Acrobat (www.adobe.com)
  - Use FTP (File Transfer Protocol)
    - **`ftp://`**
    - FTP site
      - May require login and password
      - Right click file in FTP directory and copy to hard drive

© 2001 Prentice Hall, Inc. All rights reserved.

2.7 File Transfer Protocol (FTP)

Fig. 2.9 Using IE5.5 to access an FTP site.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

- E-mail (electronic mail)
  - Delivers formatted messages over Internet
  - E-mail address
    - username@domainname
  - Free email accounts
    - Hotmail (www.hotmail.com)
    - MyRealBox (www.myrealbox.com)
    - Yahoo! (www.yahoo.com)
  - Email Programs
    - Eudora
    - Pegasus
    - Outlook Express

© 2001 Prentice Hall, Inc. All rights reserved.

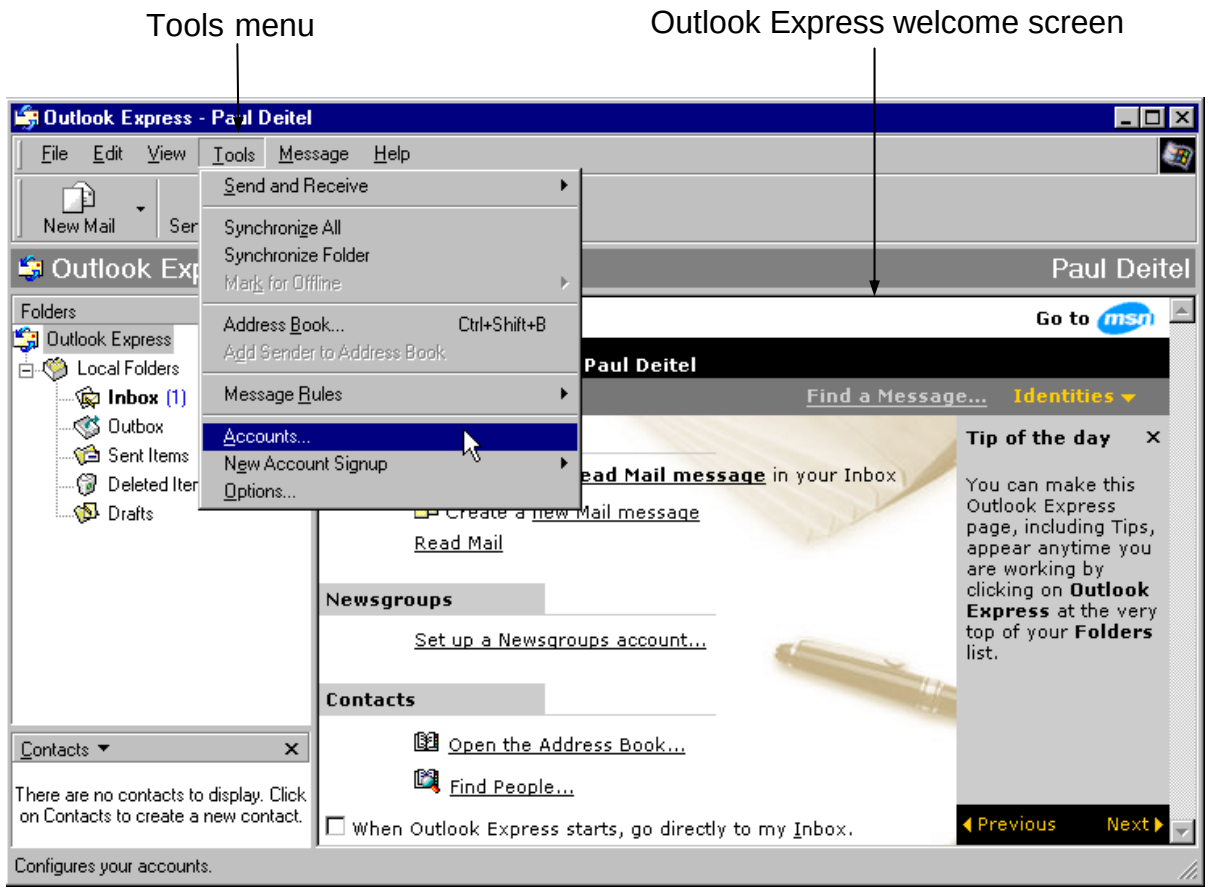
2.8 Outlook Express and Electric Mail

- Outlook Express
  - Manages multiple e-mail accounts
  - Add an account
    - **Tools** > **Accounts**
    - **Internet Accounts** dialog
      - Click **Add**
        - Select **Mail** to add an e-mail account
        - Select **News** to add a news account
      - Follow directions to set up mail server

© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

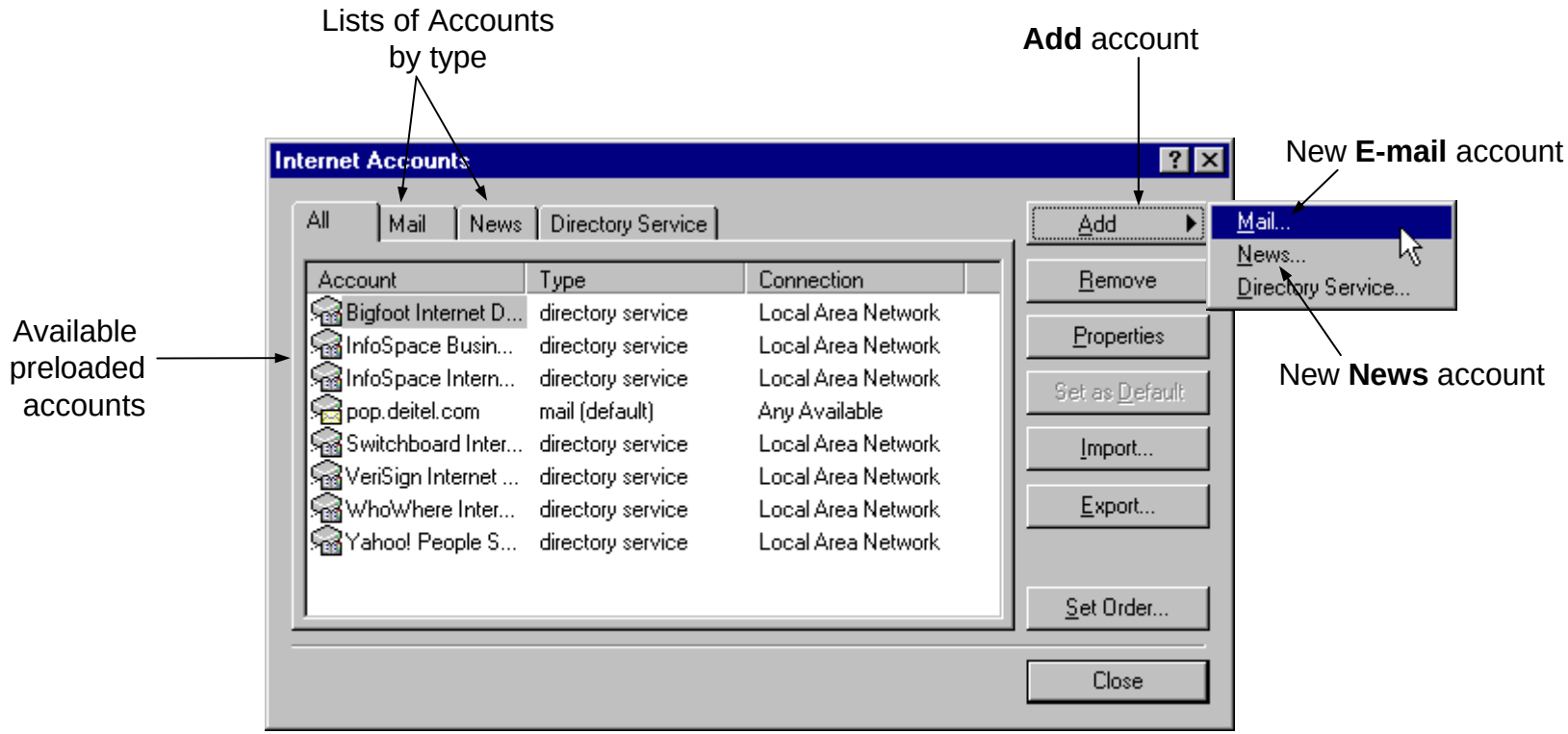
Fig. 2.10 Outlook Express opening screen and the **Internet Accounts** dialog.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

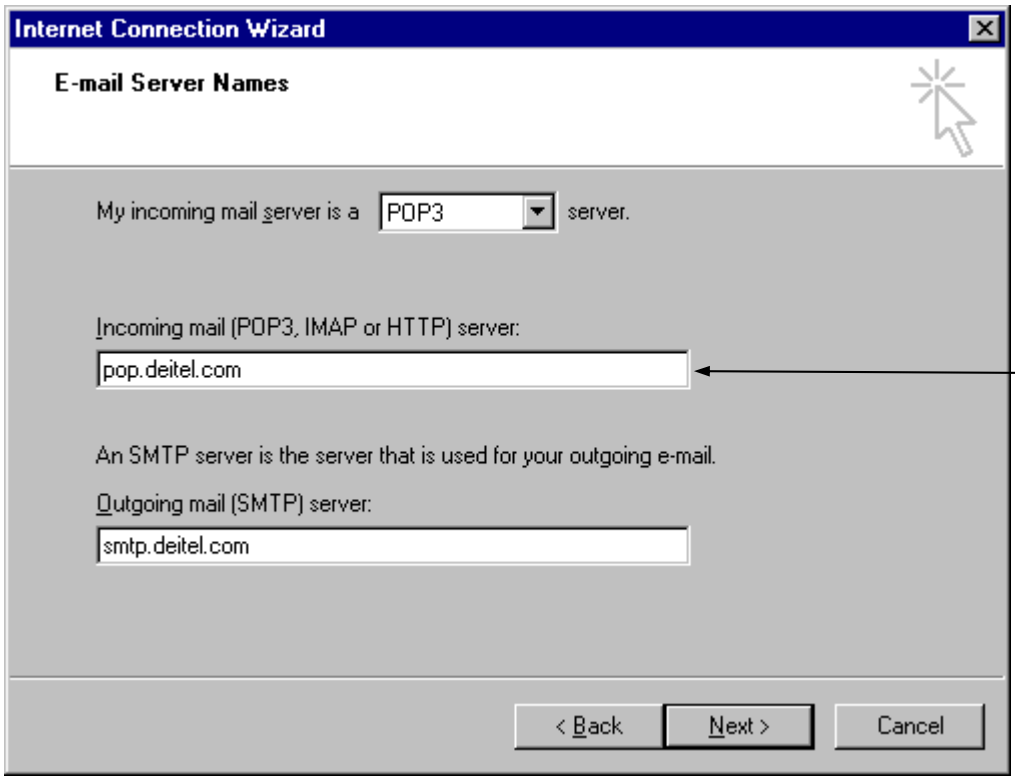
Fig. 2.10 Outlook Express opening screen and the Internet Accounts dialog.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

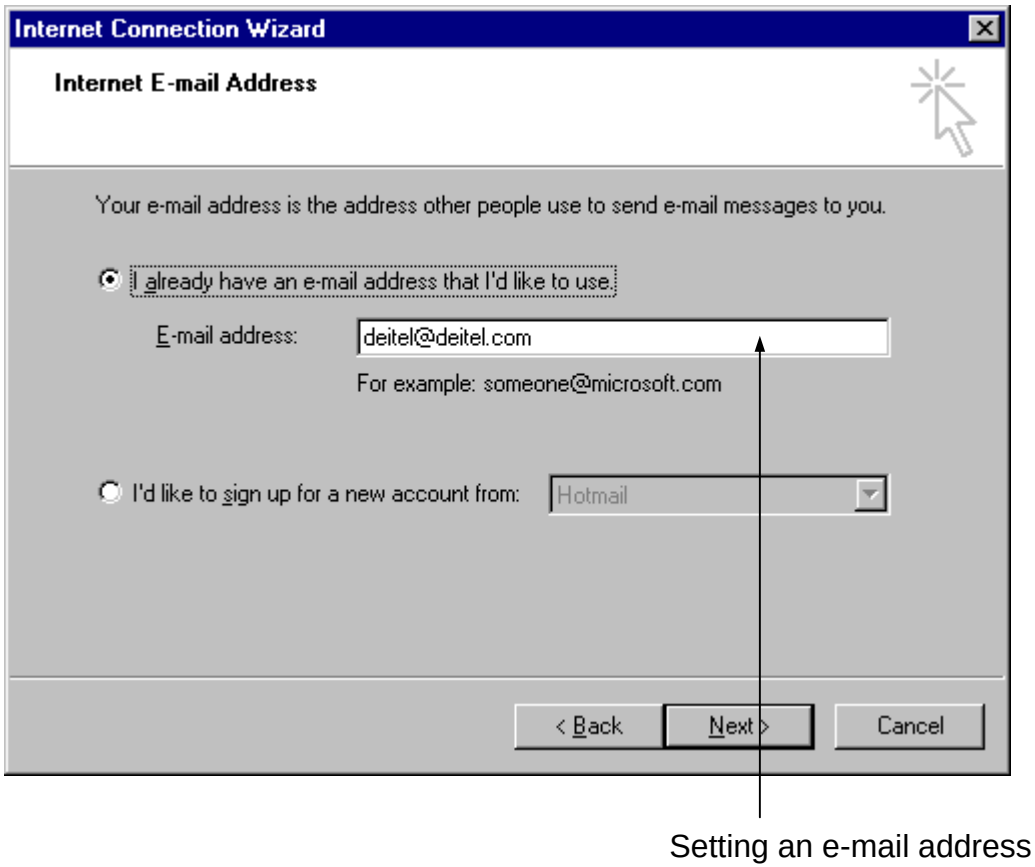
Fig. 2.11 Adding e-mail and news accounts in Outlook Express.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

Fig. 2.11 Adding e-mail and news accounts in Outlook Express.



© 2001 Prentice Hall, Inc. All rights reserved.

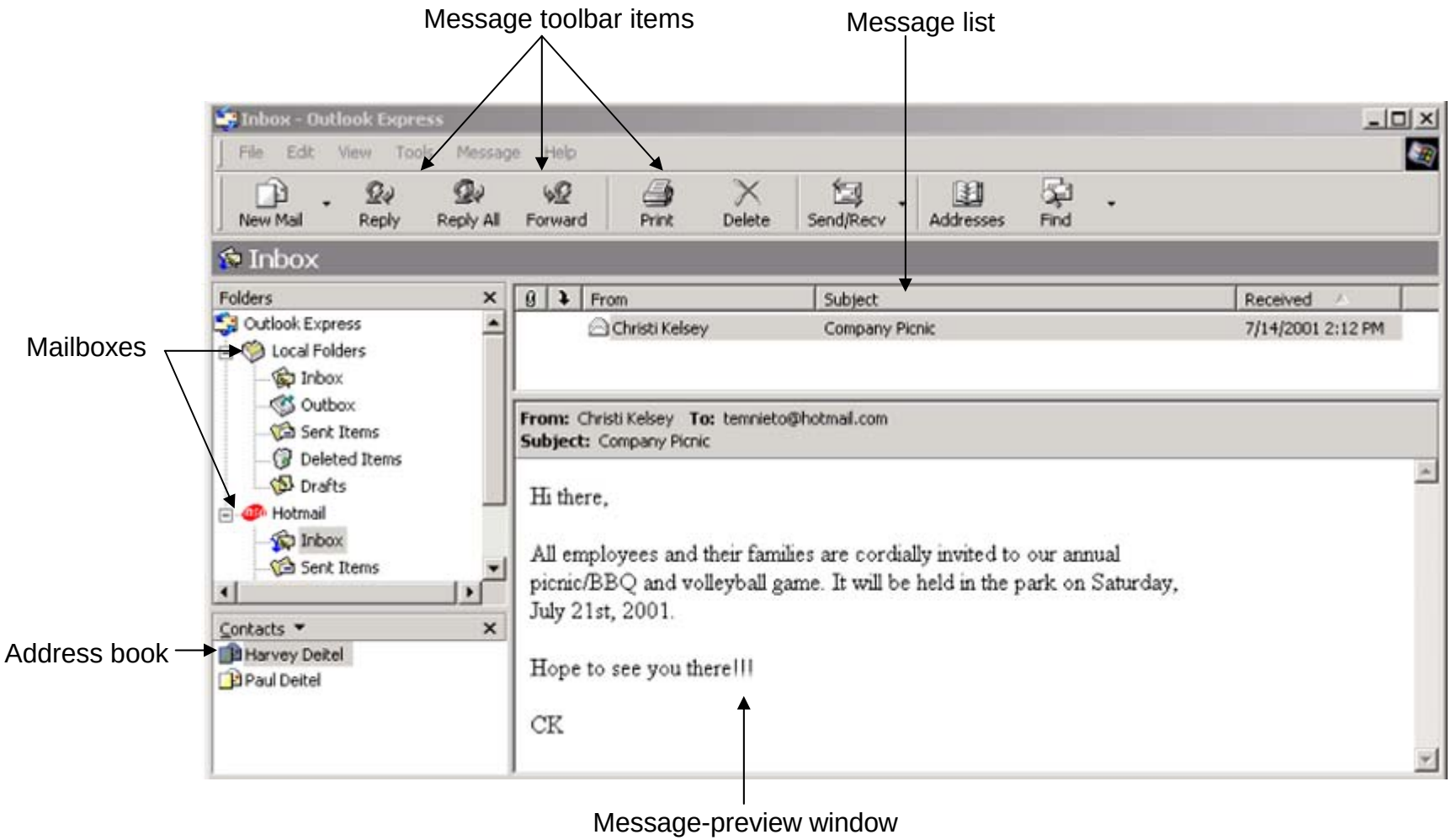
2.8 Outlook Express and Electric Mail

- Outlook Express, cont.
  - Folders window organizes messages
  - Message options
    - Reply
    - Reply all
    - Forward
    - Cc:
    - Priority
    - Send/Receive
  - Address book
    - Organizes e-mail addresses

© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

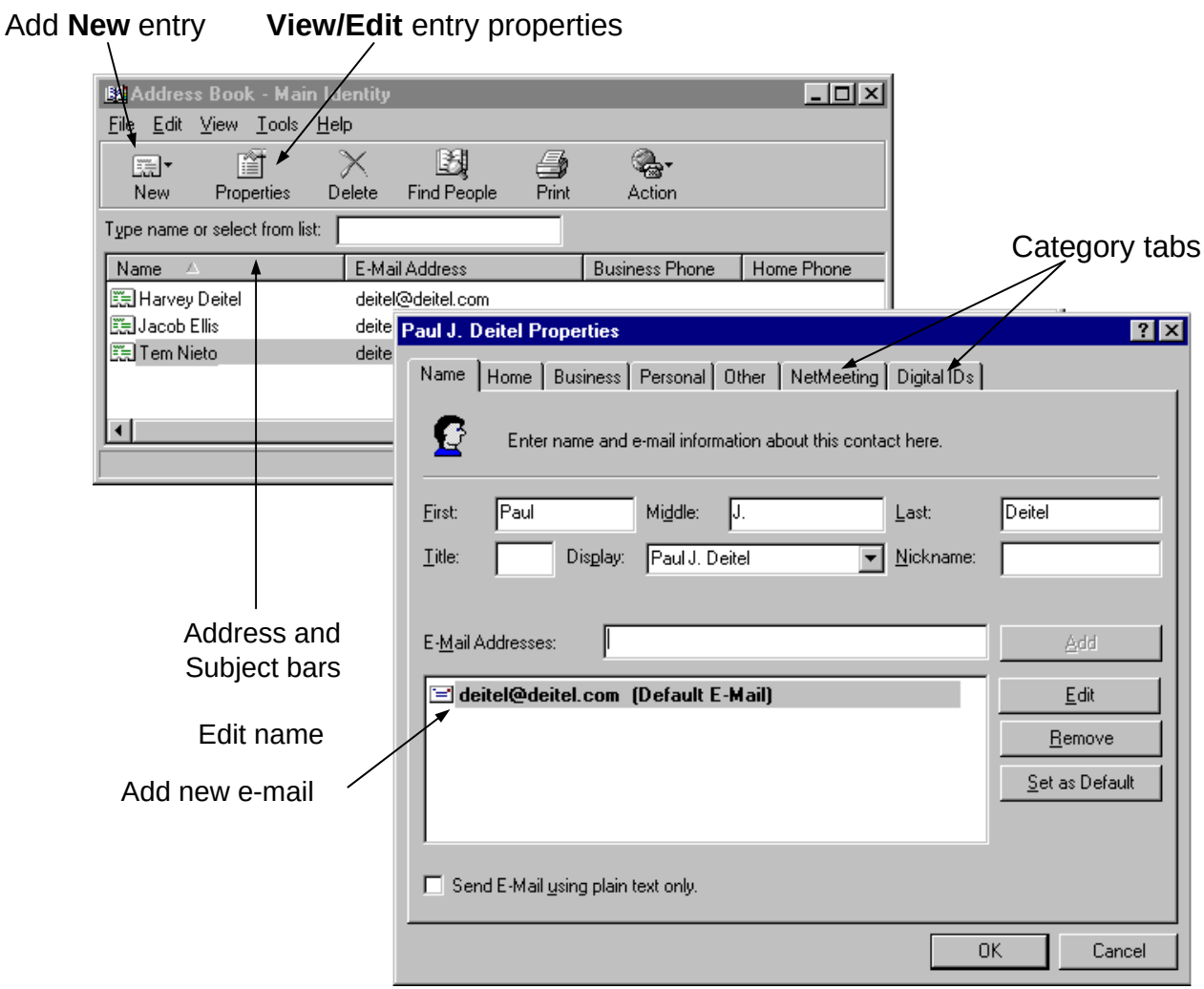
Fig. 2.12 Outlook Express e-mail main screen.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

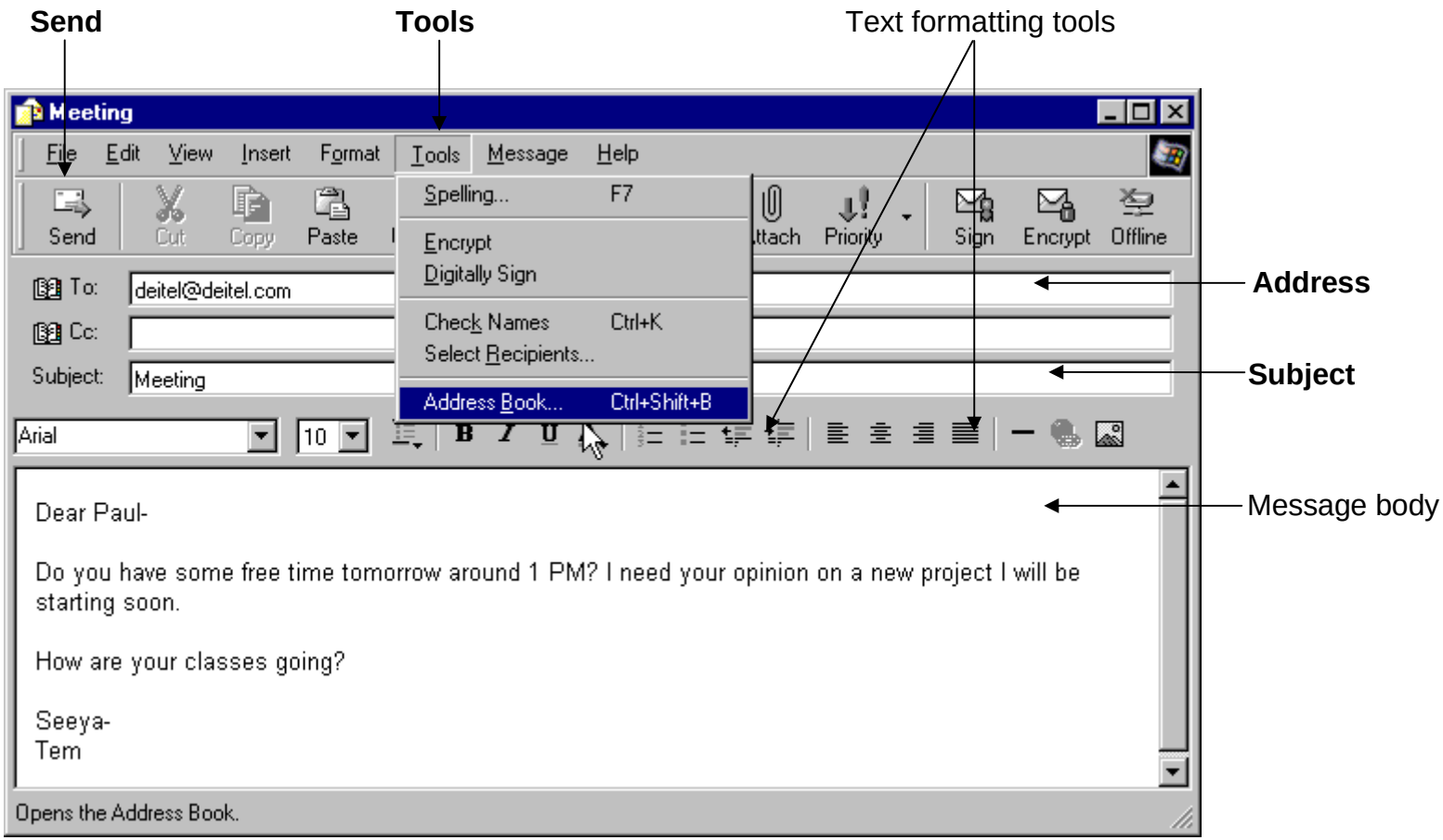
Fig. 2.13 Adding and modifying names in the Address Book.



© 2001 Prentice Hall, Inc. All rights reserved.

2.8 Outlook Express and Electric Mail

Fig. 2.14 Composing a message with Outlook Express.



© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

- Audio and video communication over Internet
- NetMeeting
  - Business and work-related collaborations
  - Text, audio and video communications
  - Tools
    - Chat
      - Real-time messaging, video and audio
    - Sharing
      - Share programs
    - File transfer
      - Users transfer files from one person to another
    - Whiteboard
      - Share diagrams and file editing capabilities

© 2001 Prentice Hall, Inc. All rights reserved.

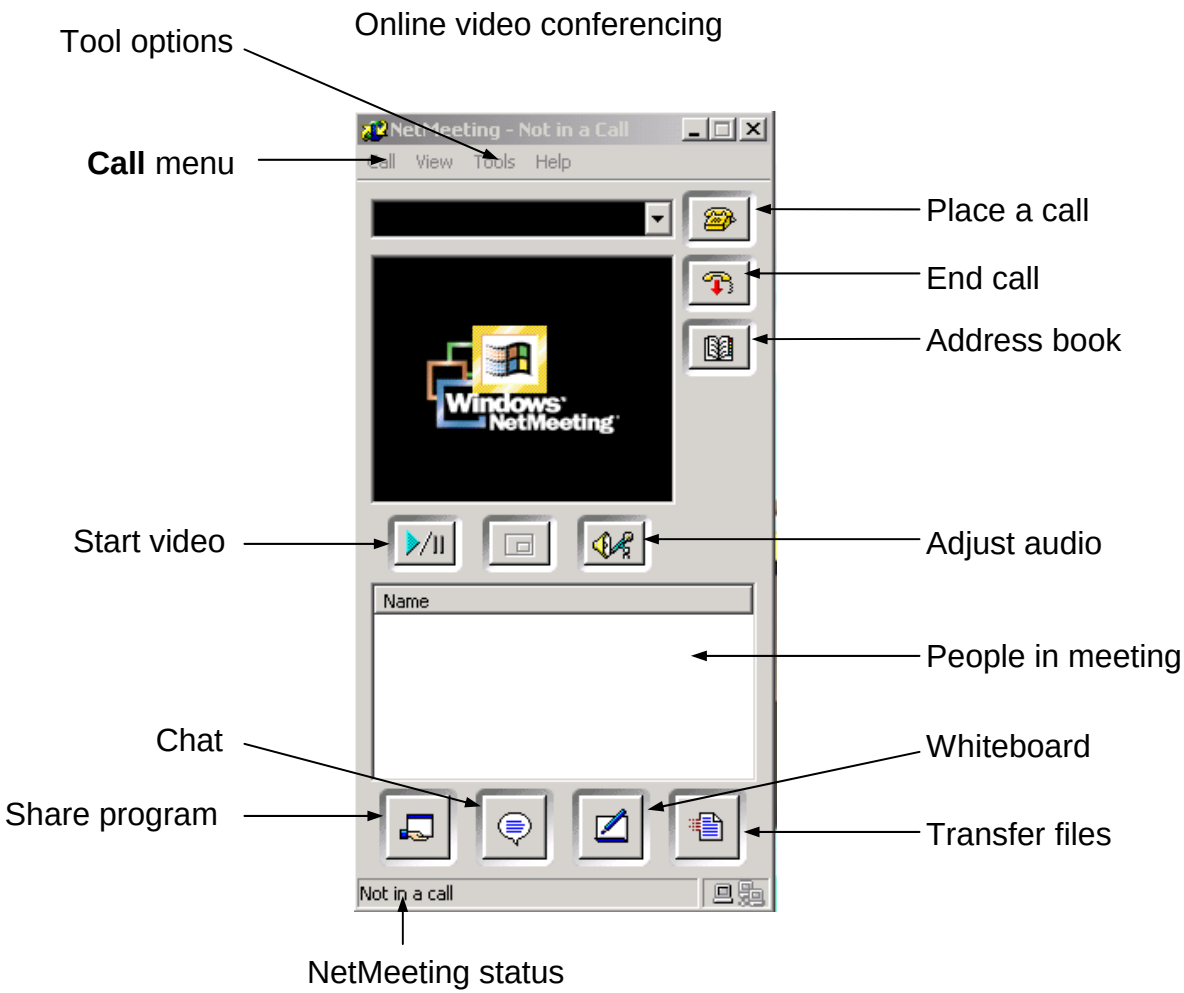
2.9 NetMeeting

- NetMeeting, cont.
  - Start a new NetMeeting
    - **Start** > **Accessories** > **Communications** > **NetMeeting**
  - Place a call
    - **Call** > **New Call…**
    - **Call** > **Host Meeting…**
    - **Place a Call** dialog
      - Address book
        - Contains list of contacts
  - Use Outlook Express to set up a NetMeeting session
    - Specify who to invite
    - Specify date, time, location and host
    - notifications

© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

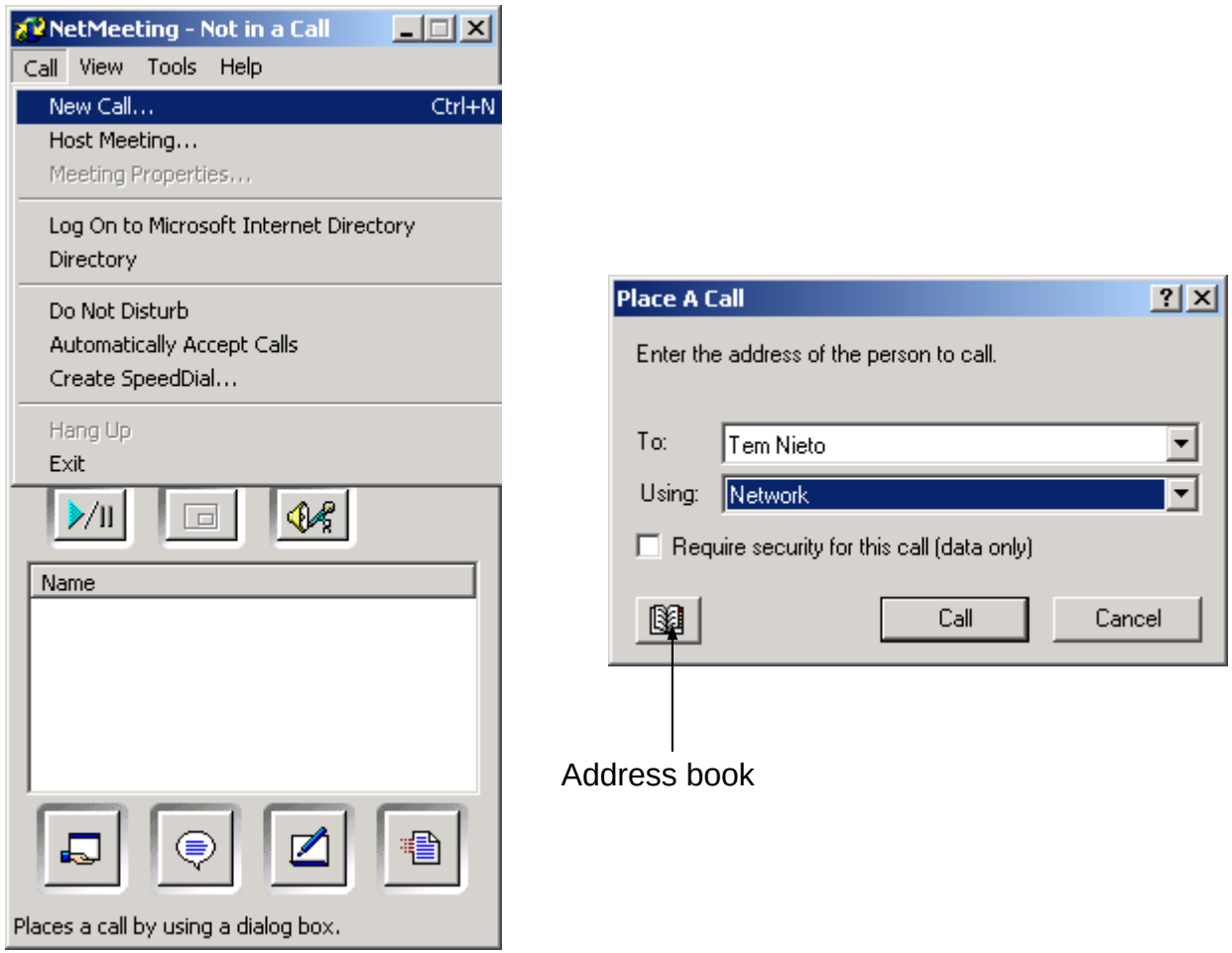
Fig. 2.15 NetMeeting User Interface.



© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

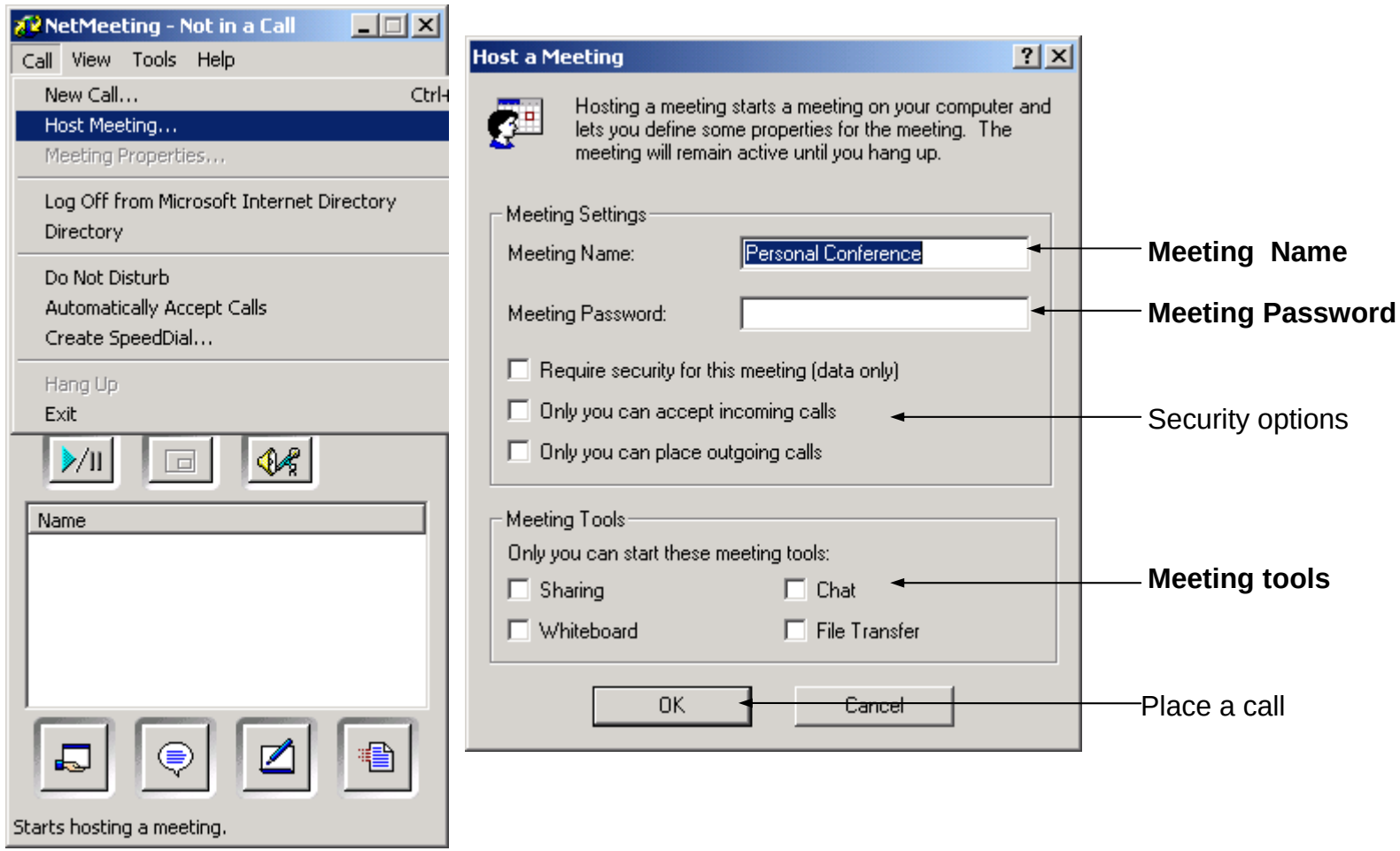
Fig. 2.16 Placing a new call to another user in NetMeeting.



© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

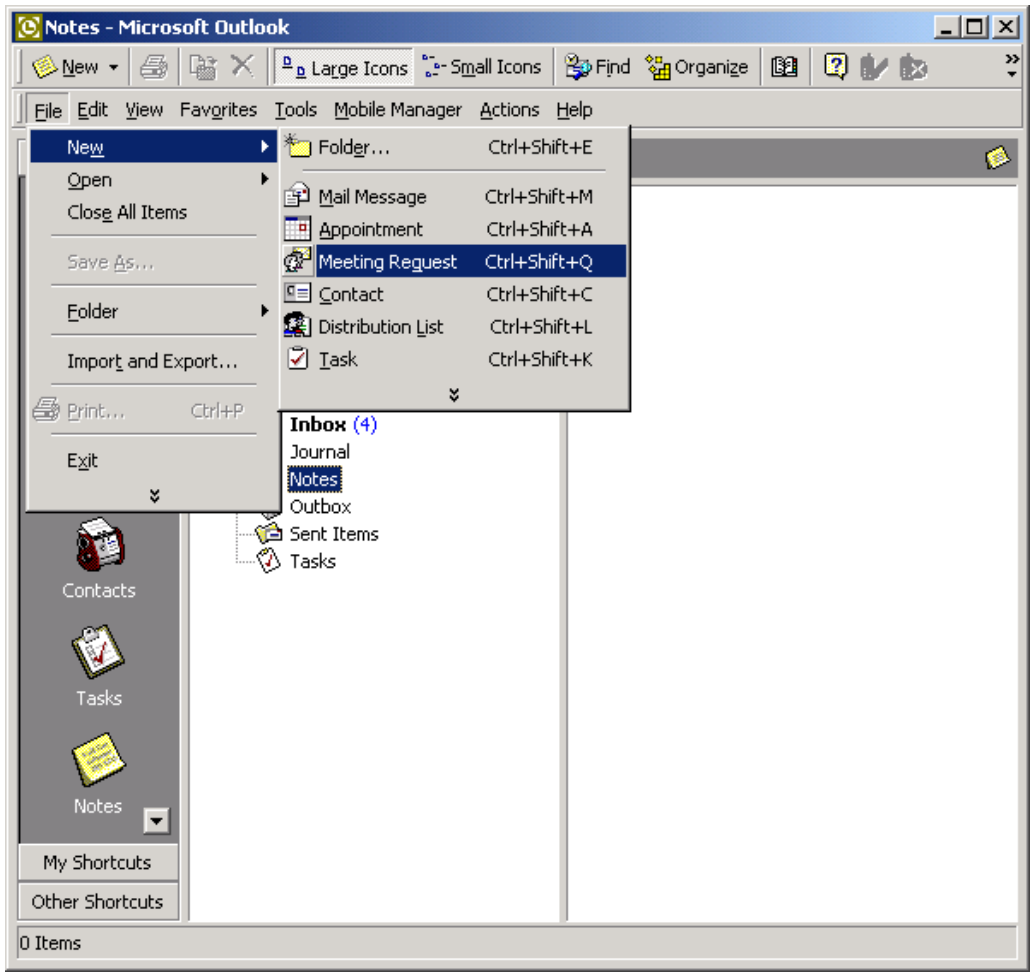
Fig. 2.17 Hosting a new NetMeeting or joining an existing one.



© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

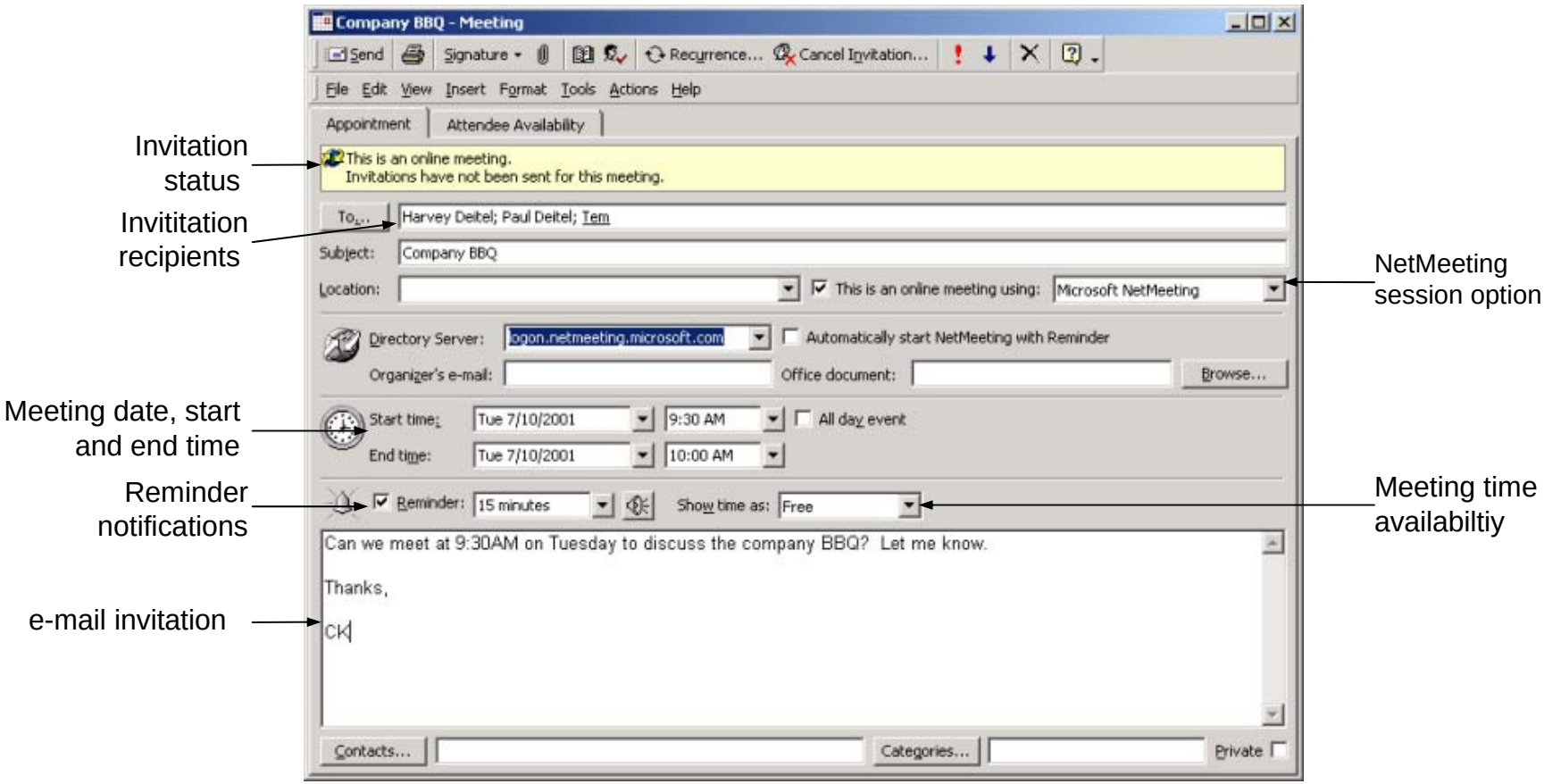
Fig. 2.18 Using Microsoft Outlook to set-up a NetMeeting.



© 2001 Prentice Hall, Inc. All rights reserved.

2.9 NetMeeting

Fig. 2.19 Sending invitations through Microsoft Outlook for a NetMeeting session.



© 2001 Prentice Hall, Inc. All rights reserved.

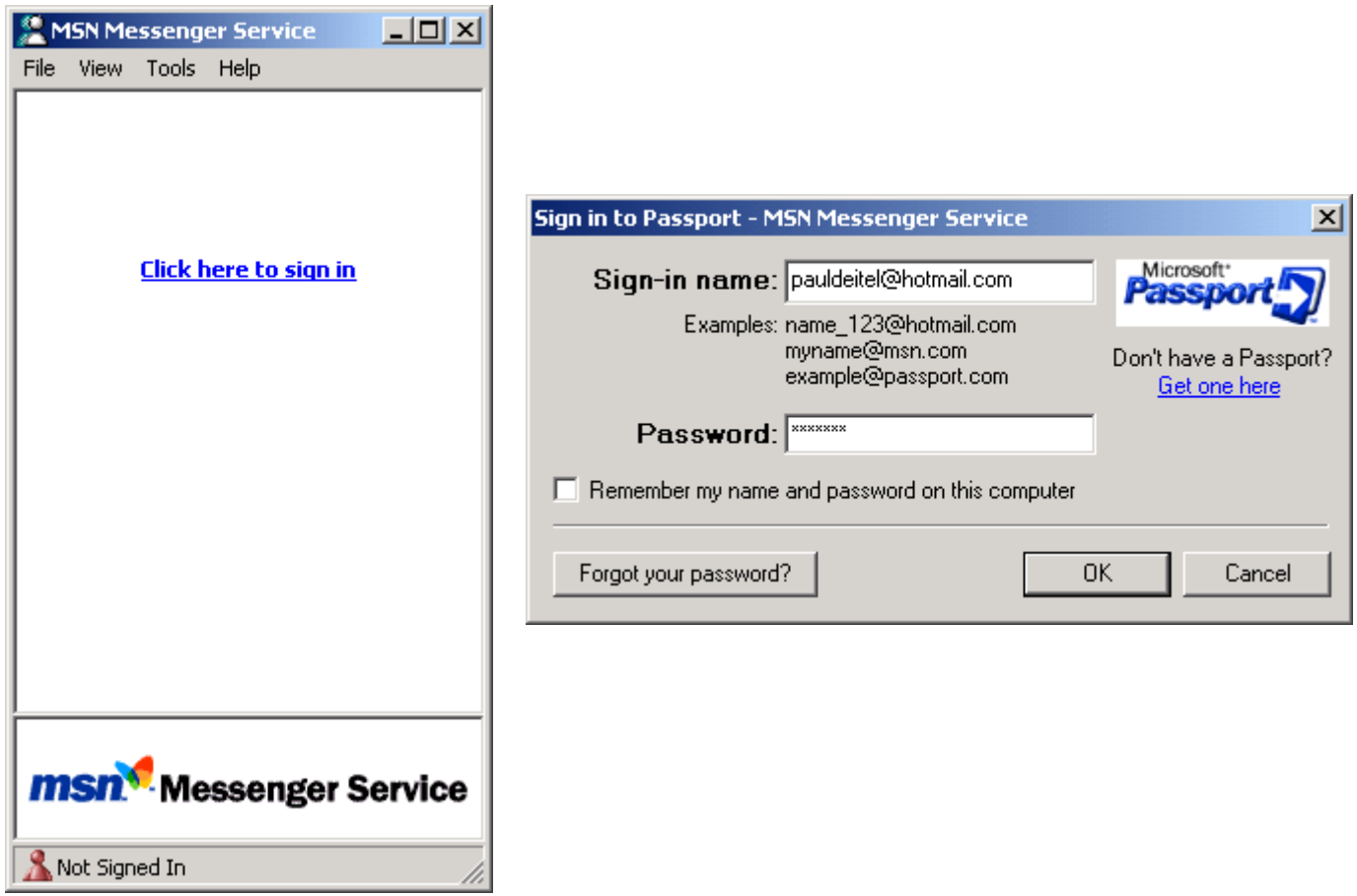
2.10 MSN Messenger Service

- Chat
  - Must have a Microsoft Passport
    - www.passport.com
  - Download application
    - http://messenger.msn.com
  - Run application
    - Sign in
      - Username and password
    - Add other passport holders to contact list
    - Start chat
      - Double click contact name (only on-line users)
      - Multi-person chat
        - **Invite** feature
      - Block users
        - **Block** feature

© 2001 Prentice Hall, Inc. All rights reserved.

2.10 MSN Messenger Service

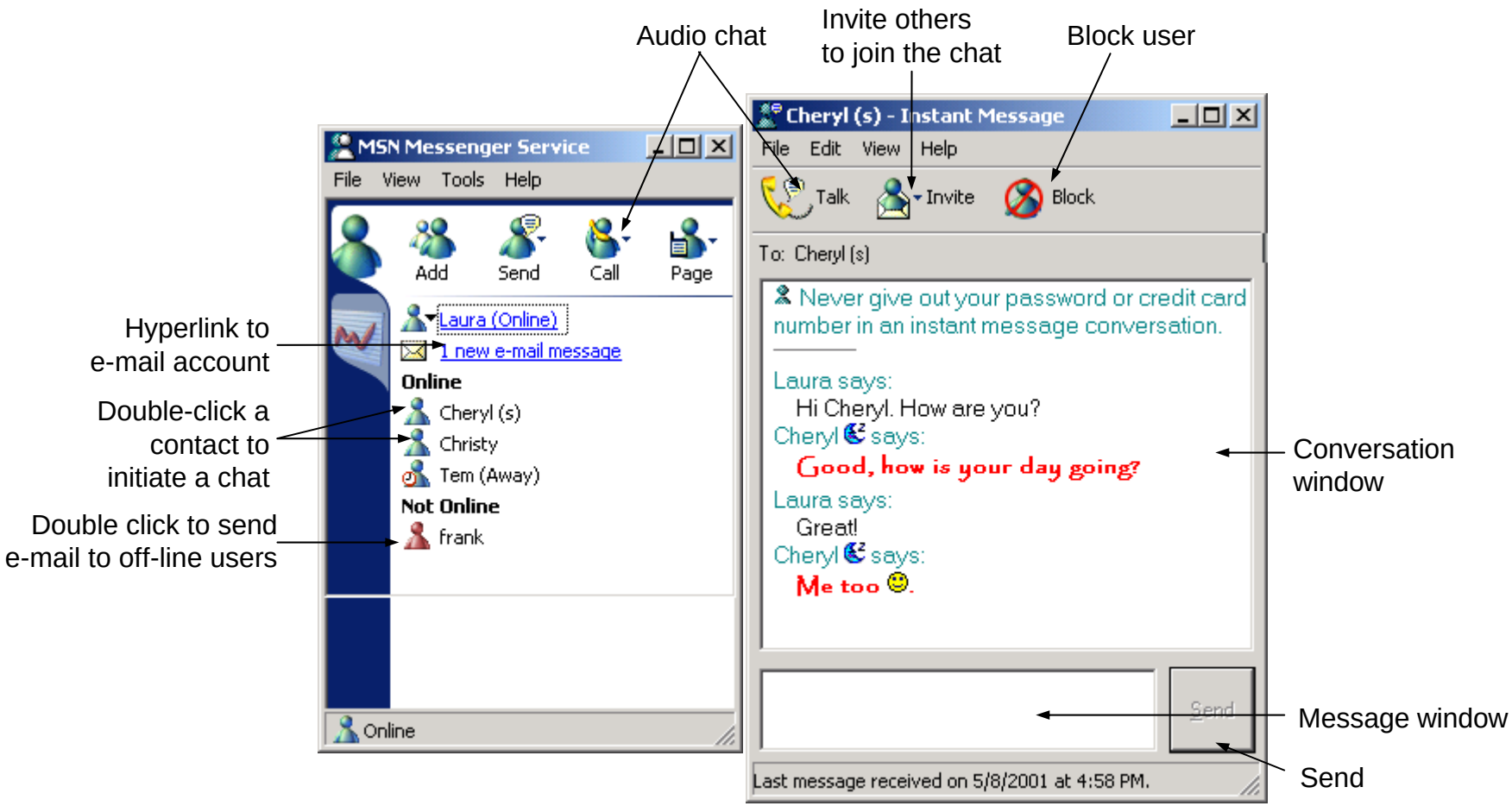
Fig. 2.20 Sign-in screen for access to MSN Messenger.



© 2001 Prentice Hall, Inc. All rights reserved.

2.10 MSN Messenger Service

Fig. 2.21 Chatting with MSN Messenger Service



© 2001 Prentice Hall, Inc. All rights reserved.

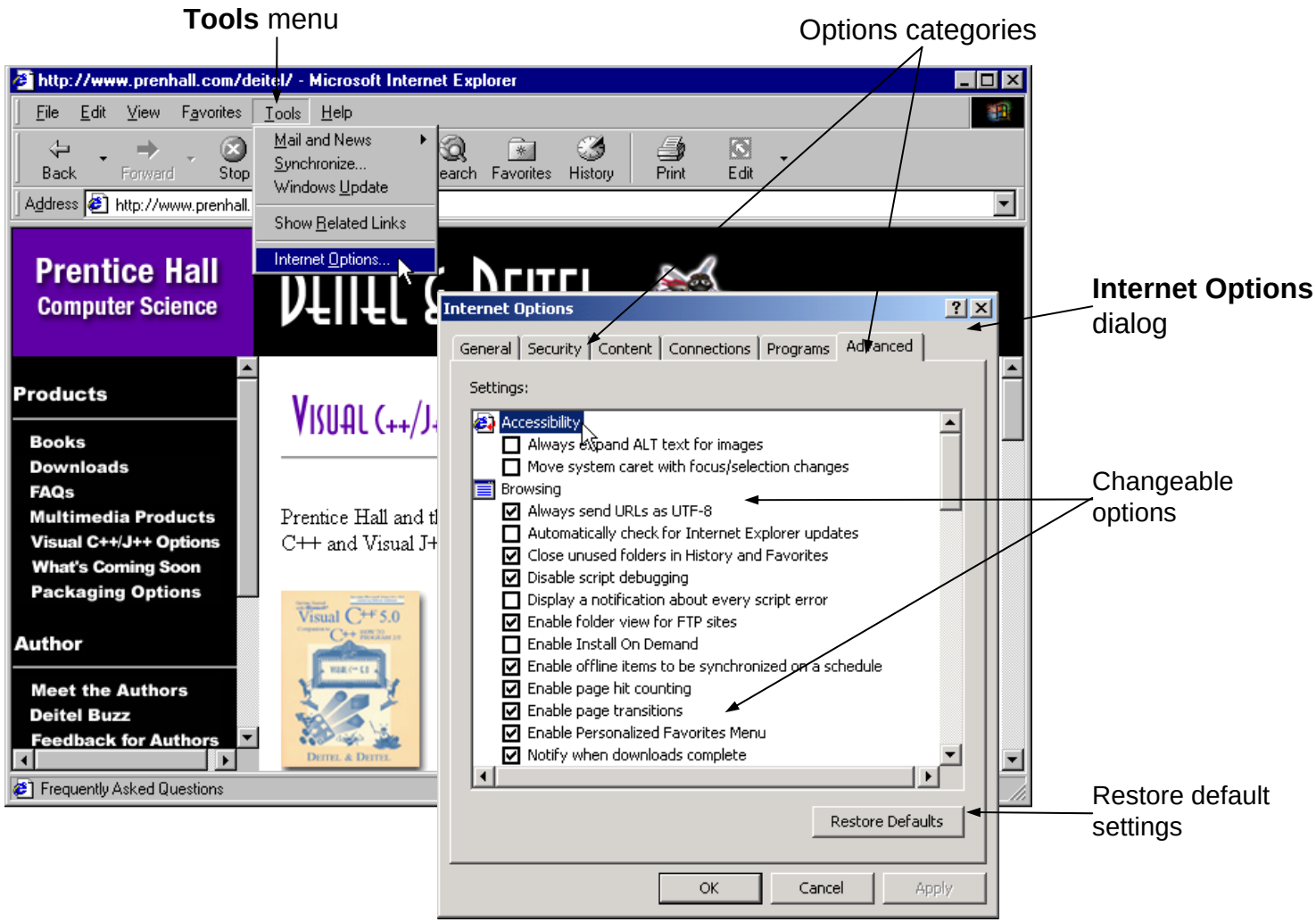
2.11 Customizing Browser Settings

- Internet Options
  - **Tools** > **Internet Options…**
  - Settings
    - **Advanced**
      - Slow connection
        - Turn off images
    - **Programs**
      - Setting default programs for e-mail and newsgroups
    - **Security**
      - Four levels of security
    - **General**
      - Set default home page

© 2001 Prentice Hall, Inc. All rights reserved.

2.11 Customizing Browser Settings

Fig. 2.22 Changing the Internet Options in IE5.5.



© 2001 Prentice Hall, Inc. All rights reserved.

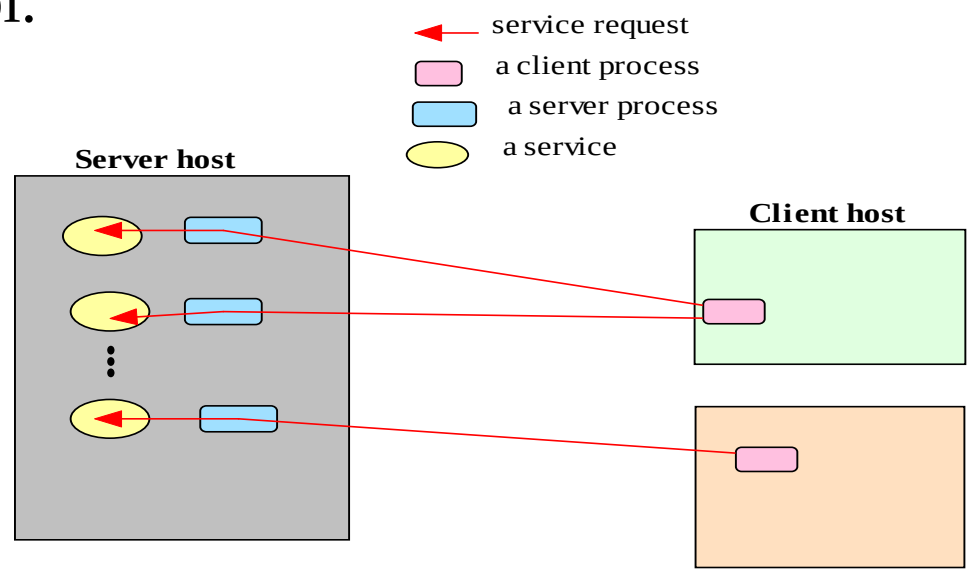
The Client-Server Model

Introduction

- The Client-Server paradigm is the most prevalent model for distributed computing protocols.
- It is the basis of all distributed computing paradigms at a higher level of abstraction.
- It is **service**-oriented, and employs a **request-response protocol**.

The Client-Server Paradigm

- A server process, running on a server host, provides access to a service.
- A client process, running on a client host, accesses the service via the server process.
- The interaction of the process proceeds according to a protocol.



The Client-Server Paradigm, conceptual

Client-server applications and services

- An application based on the client-server paradigm is a client-server application.
- On the Internet, many services are Client-server applications. These services are often known by the protocol that the application implements.
- Well known Internet services include HTTP, FTP, DNS, finger, gopher, etc.
- User applications may also be built using the client-server paradigm.

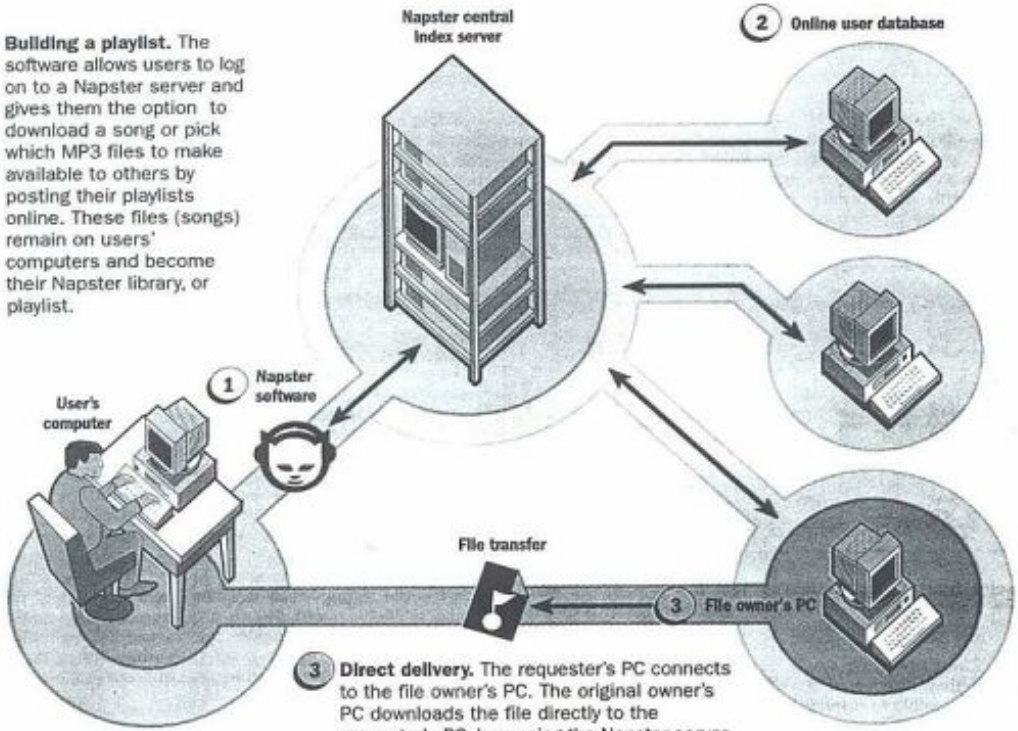
A Sample Client-Server Application

Napster

Napster is a system that facilitates file-sharing among Internet users. Instead of songs being stored on a central computer, the songs reside on users' personal computers in the form of MP3 files. When users want to download a song using Napster, they grab it from another person's computer.

1 Building a playlist. The software allows users to log on to a Napster server and gives them the option to download a song or pick which MP3 files to make available to others by posting their playlists online. These files (songs) remain on users' computers and become their Napster library, or playlist.

2 Requesting a song. Users may request files for a specific song or artist. If the server finds a match, Napster puts the computer with the file in touch with the computer that wants it.



3 Direct delivery. The requester's PC connects to the file owner's PC. The original owner's PC downloads the file directly to the requester's PC, bypassing the Napster server.

How Stuff Works: www.howstuffworks.com

Researched by VICKI GALLAY/Los Angeles Times

ROB HERNANDEZ / Los Angeles Times

Client-server system architecture

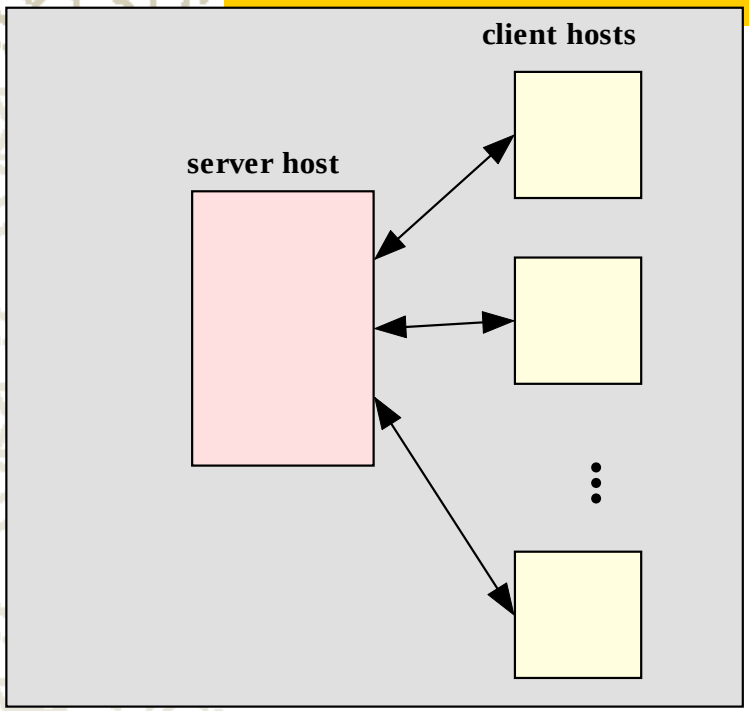
vs.

Client-server distributed computing

In the client-server system architecture, the terms *clients* and *servers* refer to computers, while in the client-server distributed computing paradigm, the terms refer to *processes*.

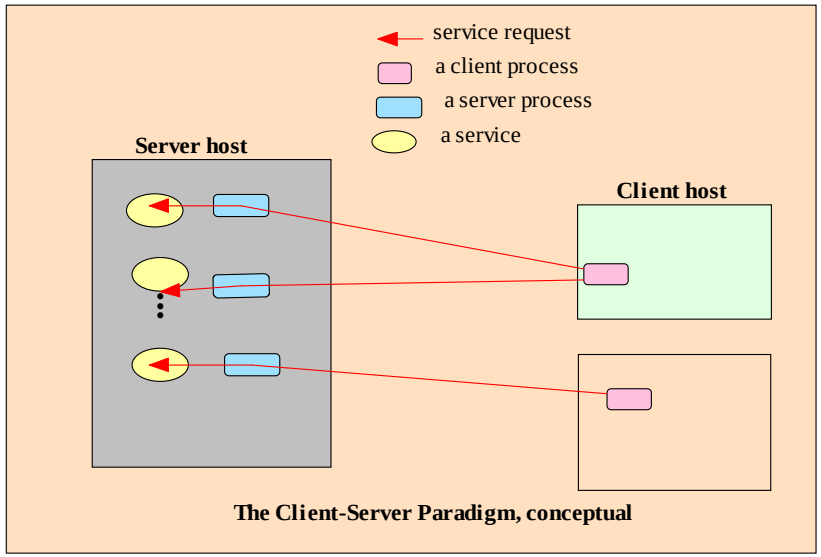
(Conceptually, that makes CS sys architecture a higher-level model than the DC paradigm, which deals with 'executing programs')

Client-server, an overloaded term



Client-Server System Architecture

Client hosts make use of services provided on a server host.



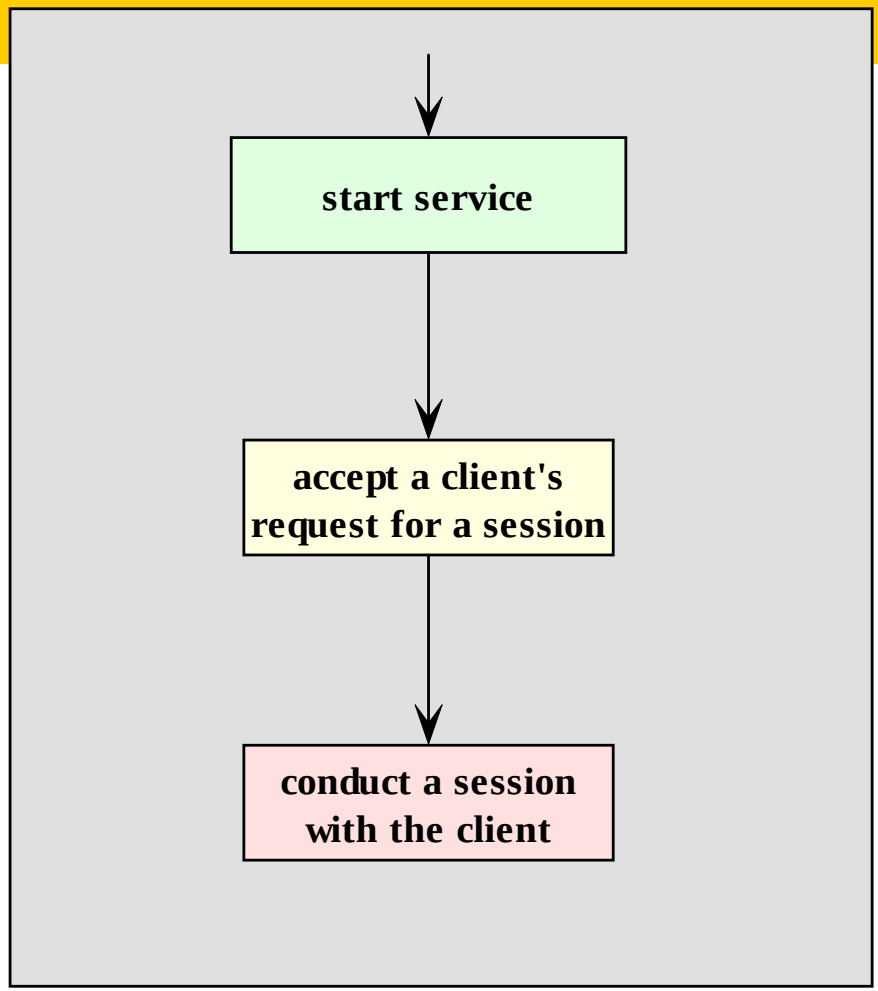
Client-Server Computing Paradigm

Client processes (objects) make use of a service provided by a server process (object) running on a server host.

A protocol/service session

In the context of the client-server model, we will use the term **session** to refer to the interaction between the server and one client. The service managed by a server may be accessed by multiple clients who desire the service, sometimes concurrently. Each client, when serviced by the server, engages in a *separate session* with the server, during which it conducts a dialog with the server until the client has obtained the service it required

A service session



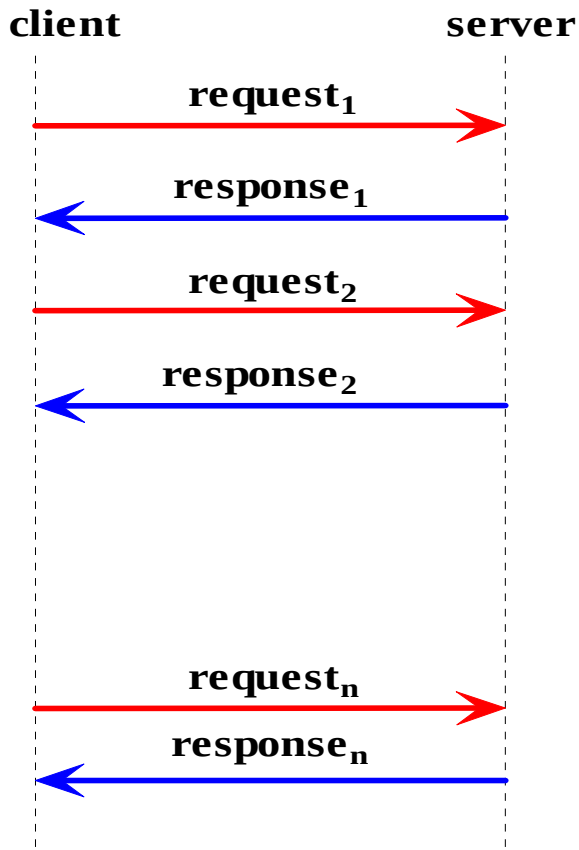
The Protocol for a Network Service

- A protocol is needed to specify the rules that must be observed by the client and the server during the conduct of a service. Such rules include specifications on matters such as (i) how the service is to be located, (ii) the sequence of interprocess communication, and (iii) the representation and interpretation of data exchanged with each IPC.
- On the Internet, such protocols are specified in the RFCs.

Locating the service

- A mechanism must be available to allow a client process to locate a server for a given service.
- A service can be located through the ***address*** of the server process, in terms of the ***host name and protocol port number*** assigned to the server process. This is the scheme for Internet services. Each Internet service is assigned to a specific port number. In particular, a well-known service such as ftp, HTTP, or telnet is assigned a default ('lower') port number reserved on each Internet host for that service.
- At a higher level of abstraction, a service may be identified using a logical name registered with a registry, the logical name will need to be mapped to the physical location of the server process. If the mapping is performed at runtime (that is, when a client process is run), then it is possible for the service's location to be dynamic, or moveable.

The interprocess communications and event synchronization

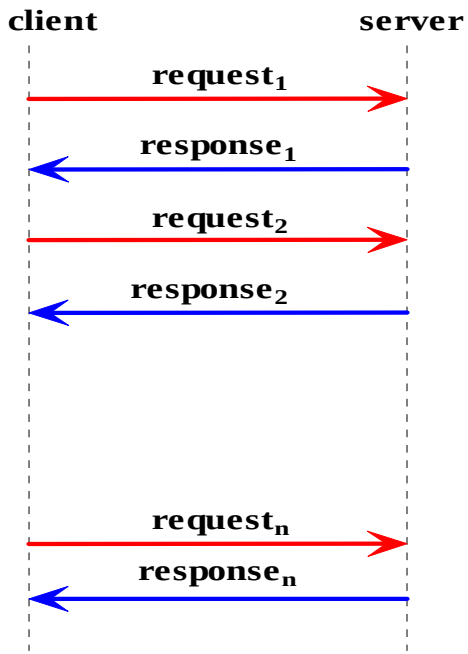


Implementation of a network service

Any implementation of the client or server program for this service is expected to adhere to the specification for the protocol, including how the dialogs of each session should proceed. Among other things, the specification defines (i) which side (client or server) should speak first, (ii) the syntax and semantic of each request and response, and (iii) the action expected of each side upon receiving a particular request or response.

The interprocess communications and event synchronization

- Typically, the interaction of the client and server processes follows a request-response pattern.



Session IPC examples

The dialog in each session follows a pattern prescribed in the protocol specified for the service.

Daytime service [RFC867]:

Client: Hello, <client address> here. May I have a timestamp please.

Server: Here it is: (time stamp follows)

World Wide Web session:

Client: Hello, <client address> here.

Server: Okay. I am a web server and speaks protocol HTTP1.0.

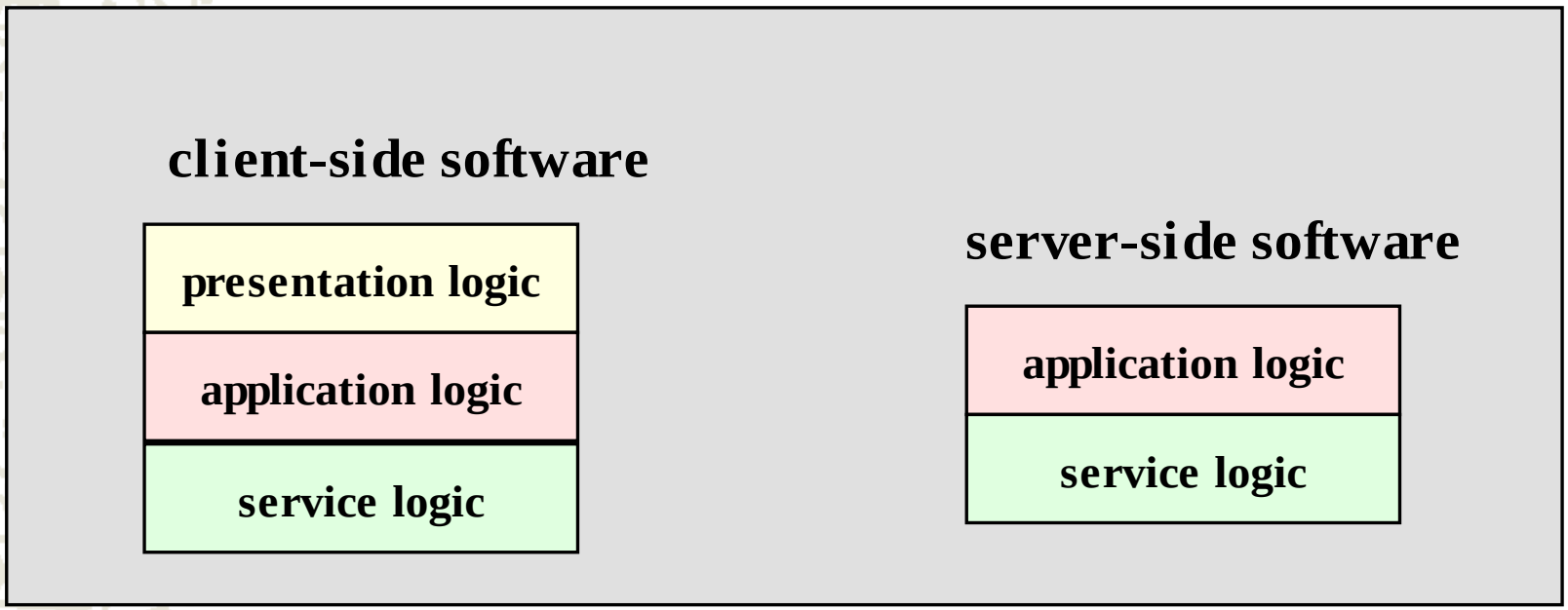
Client: Great, please get me the web page index.html at the root of your document tree.

Server: Okay, here's what's in the page: (contents follows).

Client-server protocol data representation

- Part of the specification of a protocol is the ***syntax and semantics*** of each request and response.
- The choice of data representation depends on the nature and the needs of the protocol.
- Representing data using text (character strings) is common, as it facilitates data marshalling and allows the data to be readable by human.
- Most well known Internet protocols are client-server, request-response, and text-base.

Software Engineering for a Network Service



Advantages of separating the layers of logic

- It allows each module to be developed by people with special skills to focus on a module for which they have expertise. Software engineers who are skilled in user interface may concentrate on developing the modules for the presentation logic, while those specializing in application logic and the service logic may focus on developing the other modules.

- The separation allows modifications to be made to the logic at the presentation without requiring changes to be made at the lower layers. For example, the user interface can be changed from text-mode to graphical without requiring any change be made to the application logic or the service logic. Likewise, changes made in the application logic should be transparent to the presentation layer, as we will soon illustrate with an example client-server application.

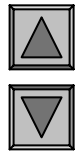
Testing a Network Service

- Because of its inherent complexity, network software is notoriously difficult to test.
- Use the three-layered software architecture and modularize each layer on both the client and the server sides.
- Use an incremental or stepwise approach in developing each module. Starting with stubs for each method, compile and test a module each time after you put in additional details.
- *Develop the client first. It is sometimes useful to employ an Echo server (to be introduced in the next section), which is known to be correct and which uses a compatible IPC mechanism, to test the client independent of the server; doing so allows you to develop the client independent of the server.*
- Use diagnostic messages throughout each program to report the progress of the program during runtime.
- Test the client-server suite on one machine before running the programs on separate machine.

Chapter 6 - Cascading Style Sheets™ (CSS)

Outline

© 2001 Prentice Hall, Inc. All rights reserved.



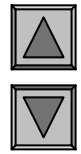
Outline

Inline.html

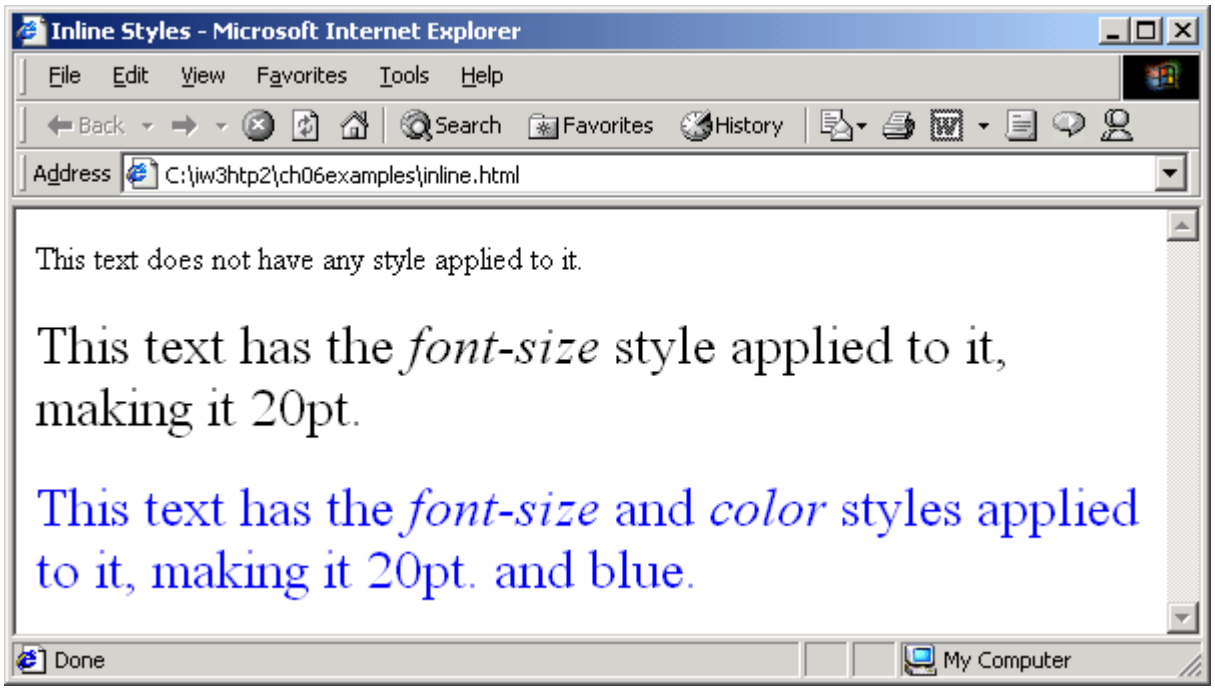
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.1: inline.html -->
6   <!-- Using inline styles   -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Inline Styles</title>
11     </head>
12
13     <body>
14
15        <p>This text does not have any style applied to it.</p>
16
17        <!-- The style attribute allows you to declare -->
18        <!-- inline styles. Separate multiple styles   -->
19        <!-- with a semicolon.                         -->
20        <p style = "font-size: 20pt">This text has the
21        <em>font-size</em> style applied to it, making it 20pt.
22        </p>
23
24        <p style = "font-size: 20pt; color: #0000ff">
25        This text has the <em>font-size</em> and
26        <em>color</em> styles applied to it, making it
27        20pt. and blue.</p>
28
29     </body>
30  </html>
```

The **style** attribute specifies the style for an element. Some style properties are **font-size** and **color**.

© 2001 Prentice Hall, Inc.
All rights reserved.

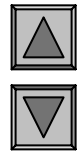


Outline



Program Output

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Declared.html

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 6.2: declared.html                          -->
6    <!-- Declaring a style sheet in the header section. -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>Style Sheets</title>
11
12         <!-- This begins the style sheet section. -->
13         <style type = "text/css">
14
15            em       { background-color: #8000ff;
16                       color: white }
17
18            h1       { font-family: arial, sans-serif }
19
20            p        { font-size: 14pt }
21
22            .special { color: blue }
23
24         </style>
25      </head>
26
```

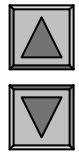
Use the **style** element to create an embedded CSS.

Styles placed in the **head** apply to all elements in the document.

More style properties include font type (**font-family**) and background color (**background-color**).

A style class named **special** is created. Style classes inherit the style properties of the style sheet in addition to their own.

© 2001 Prentice Hall, Inc.
All rights reserved.



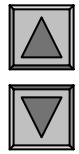
Outline

Declared.html

```
27      <body>
28
29         <!-- This class attribute applies the .blue style -->
30         <h1 class = "special">Deitel & Associates, Inc.</h1>
31
32         <p>Deitel & Associates, Inc. is an internationally
33         recognized corporate training and publishing organization
34         specializing in programming languages, Internet/World
35         Wide Web technology and object technology education.
36         Deitel & Associates, Inc. is a member of the World Wide
37         Web Consortium. The company provides courses on Java,
38         C++, Visual Basic, C, Internet and World Wide Web
39         programming, and Object Technology.</p>
40
41         <h1>Clients</h1>
42         <p class = "special"> The company's clients include many
43         <em>Fortune 1000 companies</em>, government agencies,
44         branches of the military and business organizations.
45         Through its publishing partnership with Prentice Hall,
46         Deitel & Associates, Inc. publishes leading-edge
47         programming textbooks, professional books, interactive
48         CD-ROM-based multimedia Cyber Classrooms, satellite
49         courses and World Wide Web courses.</p>
50
51      </body>
52   </html>
```

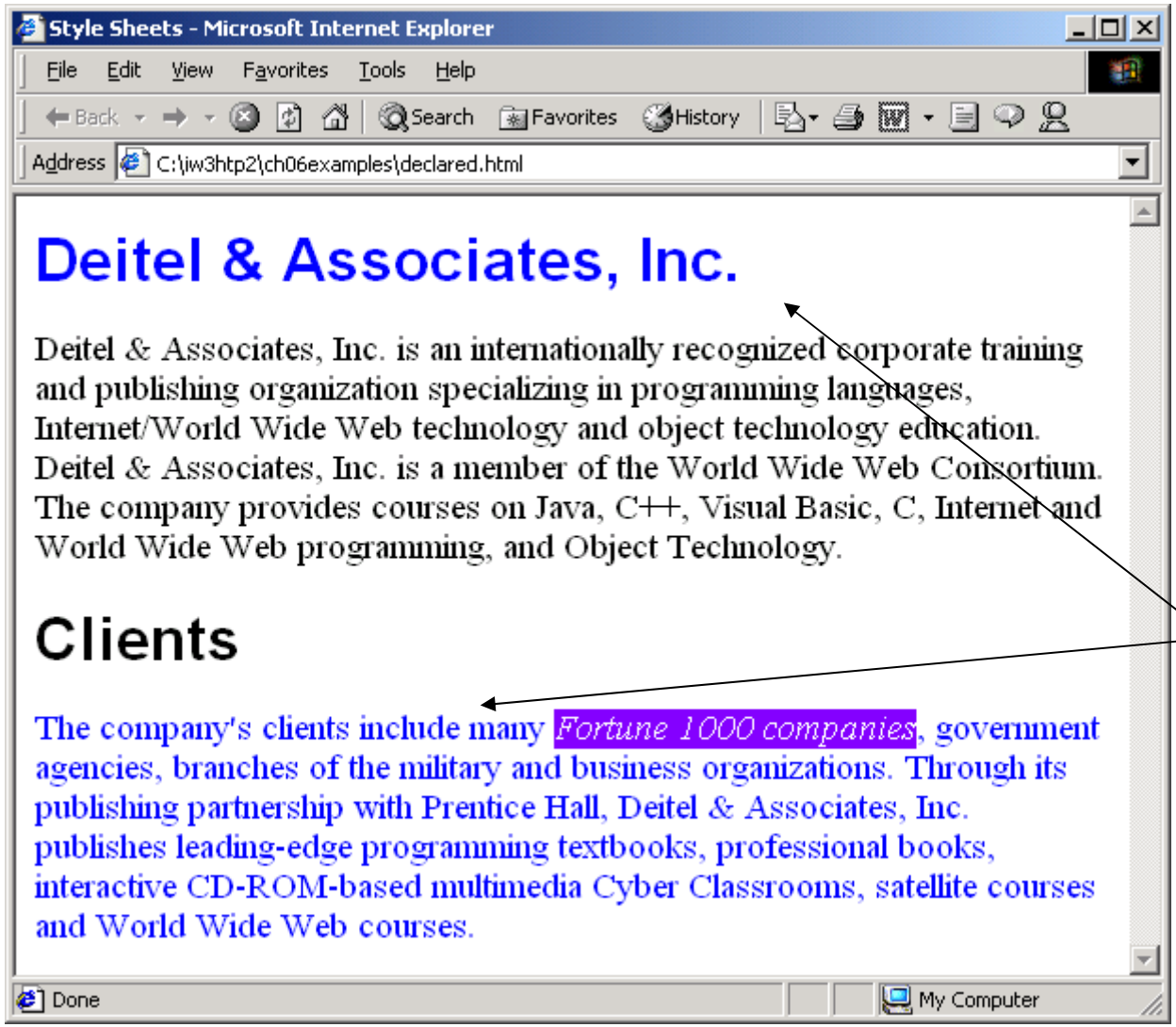
The styles associated with the **special** class are applied to the header and paragraph elements.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

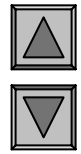
Program Output



Notice the styles defined in the CSS are applied to all paragraphs, headers, and bolded text.

Elements that have the **special** class applied have the class's styles as well as the CSS styles applied.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Advanced.html

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig 6.3: advanced.html     -->
6    <!-- More advanced style sheets -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>More Styles</title>
11
12         <style type = "text/css">
13
14            a.nodec { text-decoration: none }
15
16            a:hover  { text-decoration: underline;
17                       color: red;
18                       background-color: #ccffcc }
19
20            li em    { color: red;
21                       font-weight: bold }
22
23            ul       { margin-left: 75px }
24
25            ul ul    { text-decoration: underline;
26                       margin-left: 15px }
27
28         </style>
29      </head>
30
31      <body>
32
33         <h1>Shopping list for <em>Monday</em>:</h1>
34
```

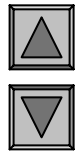
A style class is defined specifically for **a** elements. The style is applied if the **a** element's **class** attribute is set to **nodec**.

The **hover** psuedoclass defined for the **a** element is activated dynamically when the user rolls over the **a** element with a mouse.

The **em** element for **li** elements is defined to have bold red font.

Elements of an unordered list in another nested unordered list will be underlined and have a left-hand margin of 15 pixels.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Advanced.html

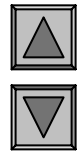
```
35         <ul>
36            <li>Milk</li>
37            <li>Bread
38               <ul>
39                  <li>White bread</li>
40                  <li>Rye bread</li>
41                  <li>Whole wheat bread</li>
42               </ul>
43            </li>
44            <li>Rice</li>
45            <li>Potatoes</li>
46            <li>Pizza <em>with mushrooms</em></li>
47         </ul>
48
49         <p><a class = "nodec" href = "http://www.food.com">
50         Go to the Grocery store</a></p>
51
52      </body>
53   </html>
```

According to the CSS defined, the three elements listed under bread should be underlined and indented 15 pixels from the left margin when rendered.

The text placed between the **em** tags should be bolded and colored red when rendered.

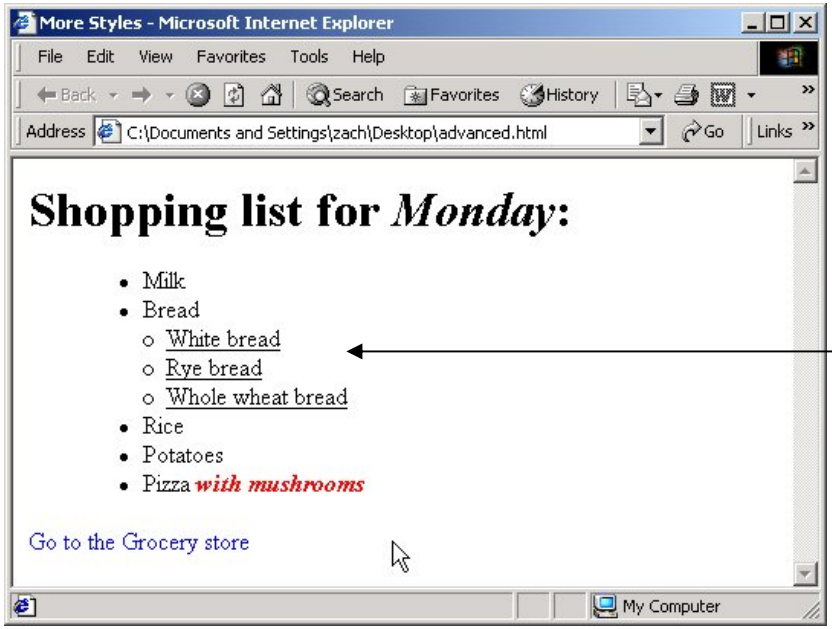
When the user scrolls over the text anchoring the link, the **hover** class will be activated and the text will change to the style defined by the **hover** class.

© 2001 Prentice Hall, Inc.
All rights reserved.

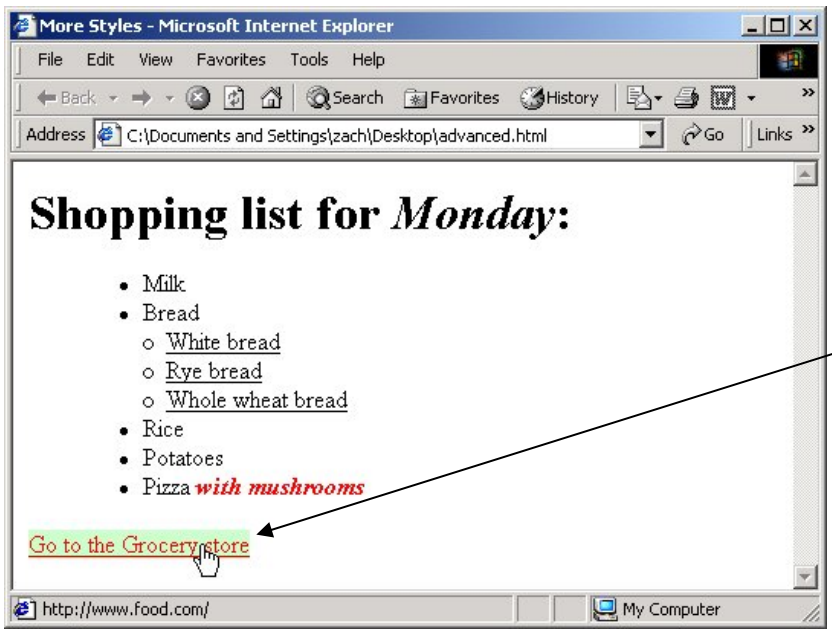


Outline

Program Output

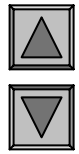


The elements in the nested unordered list are underlined and indented 15 pixels from the left margin.



By applying the **hover** psuedoclass to this anchor, the anchor's background and text color changes when the use r rolls over it with the mouse.

© 2001 Prentice Hall, Inc.
All rights reserved.

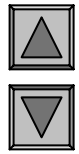


Outline

Styles.css

```
1   /* Fig. 6.4: styles.css   */
2   /* An external stylesheet */
3
4   a       { text-decoration: none }
5
6   a:hover { text-decoration: underline;
7             color: red;
8             background-color: #ccffcc }
9
10  li em   { color: red;
11            font-weight: bold;
12            background-color: #ffffff }
13
14  ul      { margin-left: 2cm }
15
16  ul ul   { text-decoration: underline;
17            margin-left: .5cm }
```

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

External.html

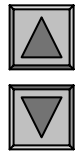
```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 6.5: external.html        -->
6    <!-- Linking external style sheets  -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>Linking External Style Sheets</title>
11         <link rel = "stylesheet" type = "text/css"
12            href = "styles.css" />
13      </head>
14
15      <body>
16
17         <h1>Shopping list for <em>Monday</em>:</h1>
18         <ul>
19            <li>Milk</li>
20            <li>Bread
21               <ul>
22                  <li>White bread</li>
23                  <li>Rye bread</li>
24                  <li>Whole wheat bread</li>
25               </ul>
26            </li>
27            <li>Rice</li>
28            <li>Potatoes</li>
29            <li>Pizza <em>with mushrooms</em></li>
30            </ul>
31
```

The **link** element is used to reference an external style sheet.

The **type** attribute defines the MIME type.

The **rel** attribute is used to define the linking relationship.

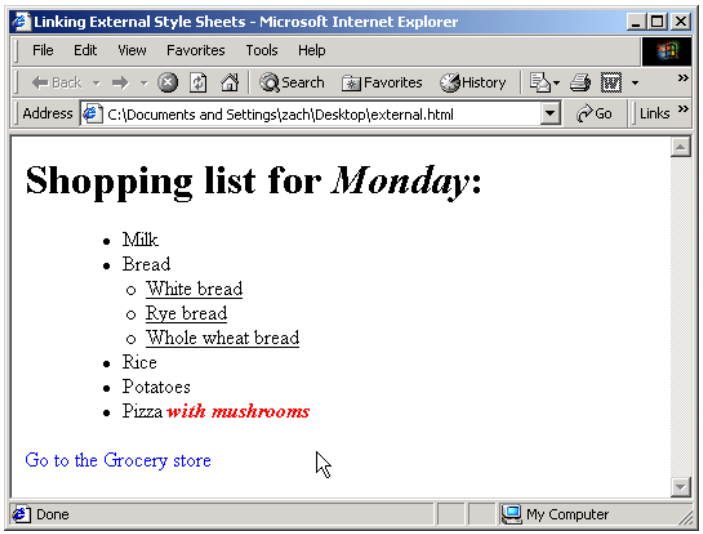
© 2001 Prentice Hall, Inc.
All rights reserved.



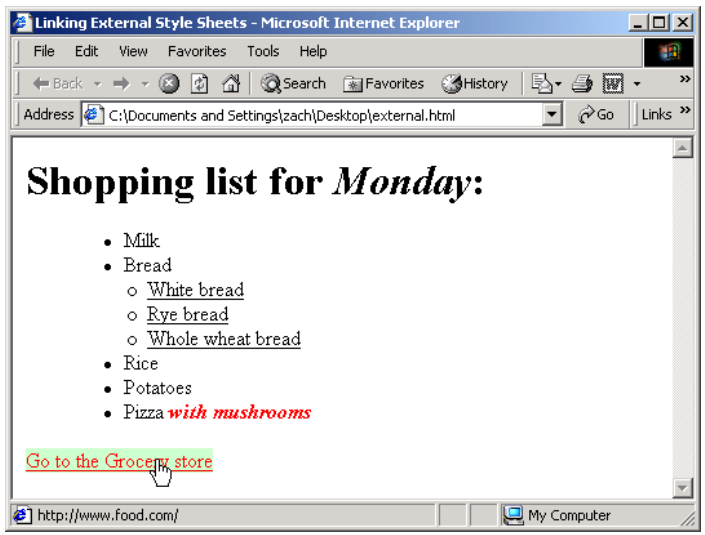
Outline

```
32          <p>
33          <a href = "http://www.food.com">Go to the Grocery store</a>
34          </p>
35
36       </body>
37    </html>
```

External.html



Program Output



The documents rendered with an external CSS should be the same as those rendered with an internal CSS.

© 2001 Prentice Hall, Inc.
All rights reserved.

6.6 W3C CSS Validation Service

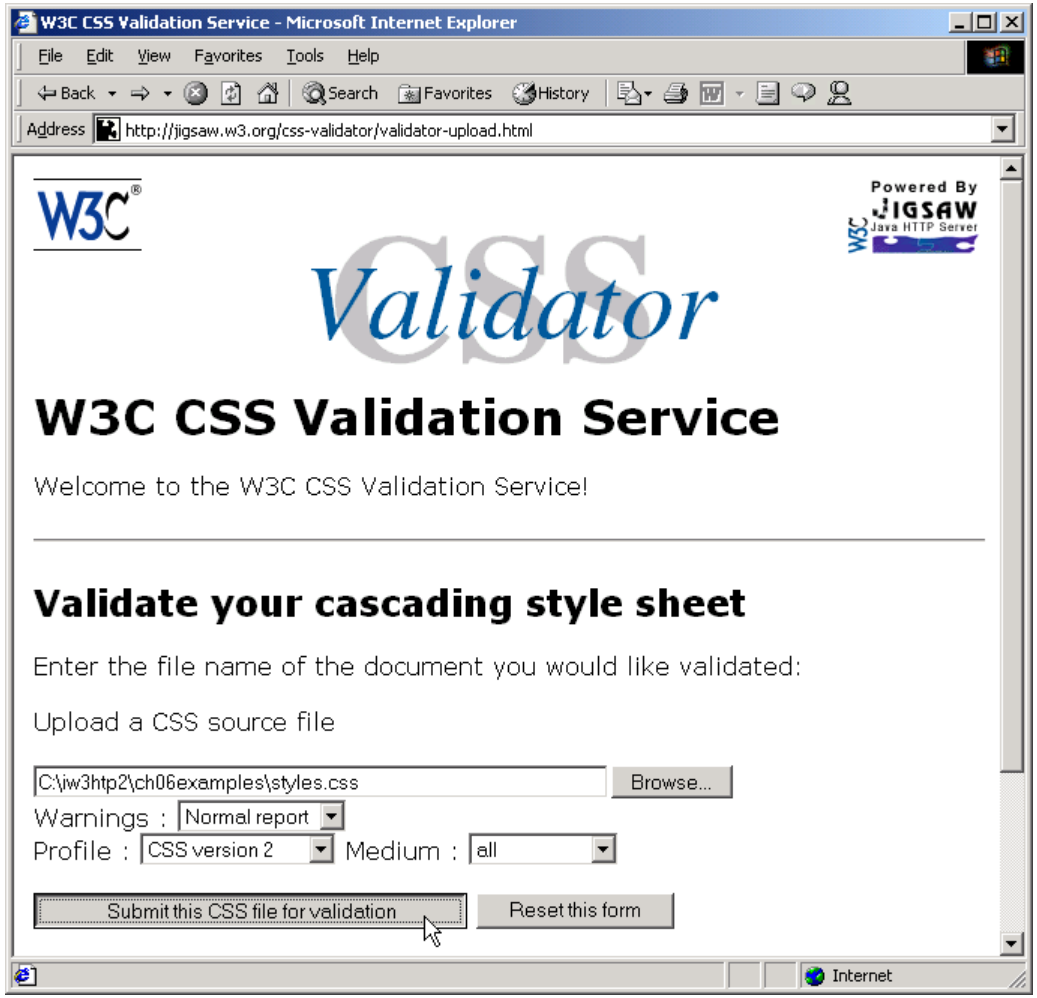


Fig. 6.6 Validating a CSS document. (Courtesy of World Wide Web Consortium (W3C).)

© 2001 Prentice Hall, Inc. All rights reserved.

6.6 W3C CSS Validation Service

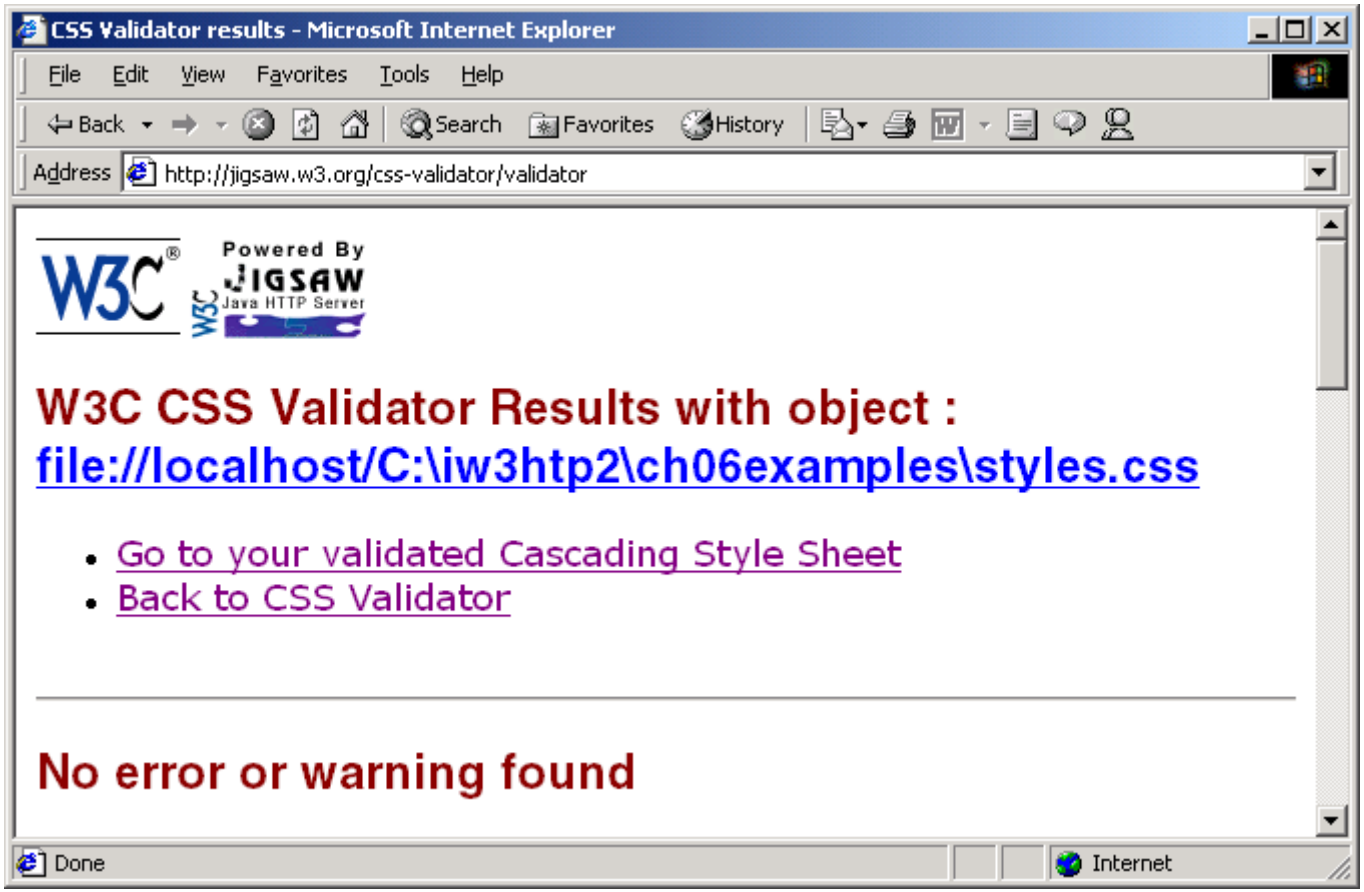
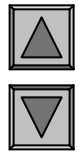


Fig. 6.7 CSS validation results. (Courtesy of World Wide Web Consortium (W3C).)

© 2001 Prentice Hall, Inc. All rights reserved.



Outline

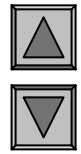
Positioning.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig 6.8: positioning.html        -->
6   <!-- Absolute positioning of elements -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Absolute Positioning</title>
11     </head>
12
13     <body>
14
15        <p><img src = "i.gif" style = "position: absolute;
16           top: 0px; left: 0px; z-index: 1" alt =
17           "First positioned image" /></p>
18        <p style = "position: absolute; top: 50px; left: 50px;
19           z-index: 3; font-size: 20pt;">Positioned Text</p>
20        <p><img src = "circle.gif" style = "position: absolute;
21           top: 25px; left: 100px; z-index: 2" alt =
22           "Second positioned image" /></p>
23
24     </body>
25  </html>
```

The **position** property of the **style** element allows for positioning of an element.

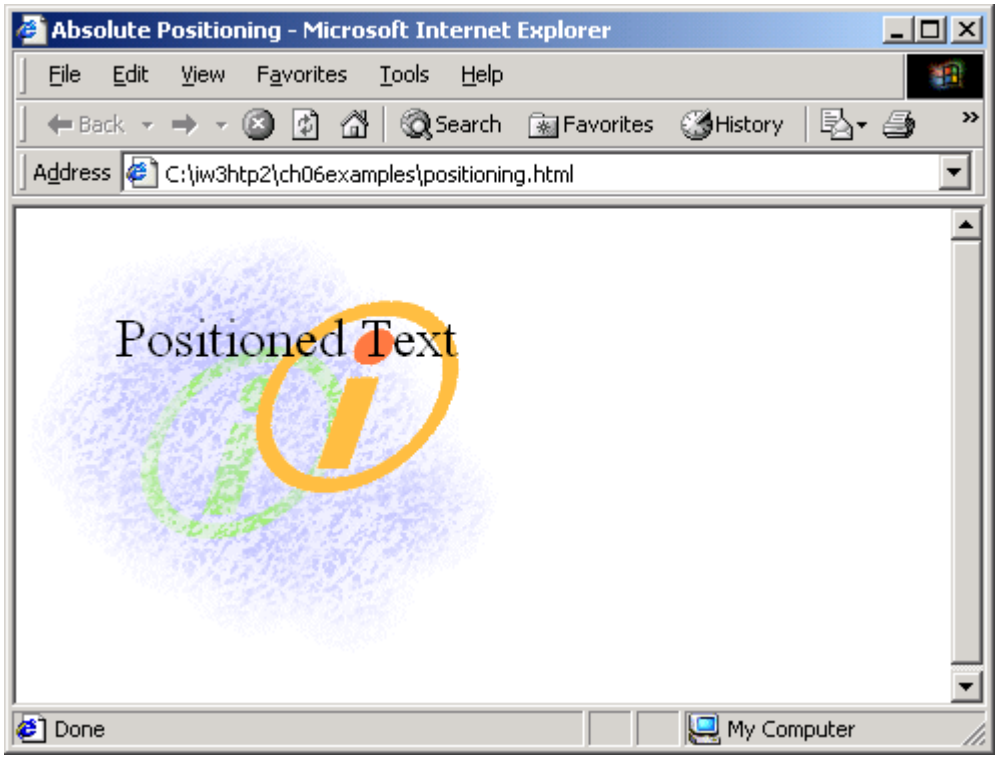
The **z-index** property allows layering of multiple images.The images are layered such that images with lower z-indexes are placed under images with higher ones.

© 2001 Prentice Hall, Inc.
All rights reserved.



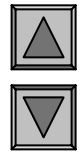
Outline

Program Output



The effect of the **z-index** property is several images layered on top of one another.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

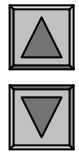
Positioning2.html

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 6.9: positioning2.html       -->
6    <!-- Relative positioning of elements  -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>Relative Positioning</title>
11
12         <style type = "text/css">
13
14            p           { font-size: 1.3em;
15                          font-family: verdana, arial, sans-serif }
16
17            span        { color: red;
18                          font-size: .6em;
19                          height: 1em }
20
21            .super      { position: relative;
22                          top: -1ex }
23
24            .sub        { position: relative;
25                          bottom: -1ex }
26
27            .shiftleft  { position: relative;
28                          left: -1ex }
29
30            .shiftright { position: relative;
31                          right: -1ex }
32
33         </style>
34      </head>
```

Relative positioning places an element relative to a reference point on the page.

For instance, when this class is applied the element will be displayed –1cm relative to the left of where it would be placed.

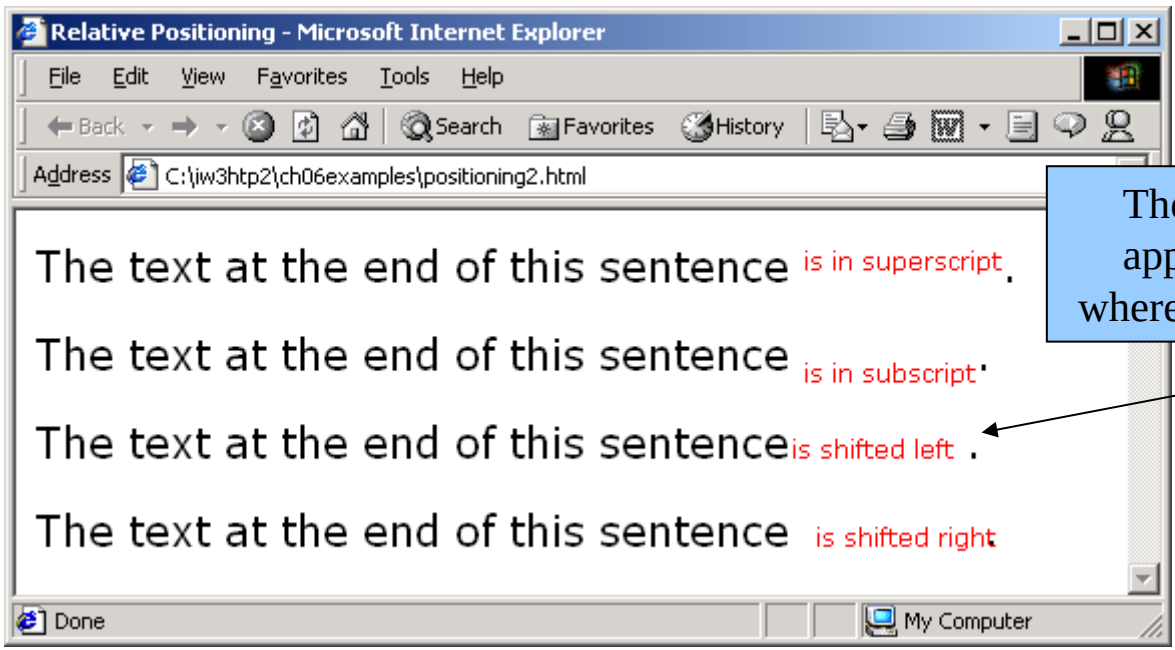
© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Positioning2.html

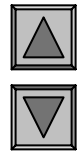
```
35
36      <body>
37
38         <p>The text at the end of this sentence
39         <span class = "super">is in superscript</span>.</p>
40
41         <p>The text at the end of this sentence
42         <span class = "sub">is in subscript</span>.</p>
43
44         <p>The text at the end of this sentence
45         <span class = "shiftleft">is shifted left</span>.</p>
46
47         <p>The text at the end of this sentence
48         <span class = "shiftright">is shifted right</span>.</p>
49
50      </body>
51   </html>
```



Program Output

The text in red has the **shiftleft** class applied, and is shifted left relative to where it would have initially been placed.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Background.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.10: background.html                  -->
6   <!-- Adding background images and indentation -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Background Images</title>
11
12        <style type = "text/css">
13
14           body  { background-image: url(logo.gif)
15                   background-position: bottom rig
16                   background-repeat: no-repeat;
17                   background-attachment: fixed; }
18
19           p     { font-size: 18pt;
20                   color: #aa5588;
21                   text-indent: 1em;
22                   font-family: arial, sans-serif; }
23
24           .dark { font-weight: bold; }
25
26        </style>
27     </head>
28
```

The **background-image** property assigns a background to the body of the page.

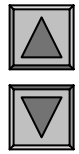
The **background–position** property assigns a location for the image on the page.

If set to **repeat**, the **background-repeat** property will tile the image as the background.

The value of the **background-attachment** property determines if the image moves as the user scrolls the page.

The font format specified will be applied to all **p** elements.

© 2001 Prentice Hall, Inc.
All rights reserved.

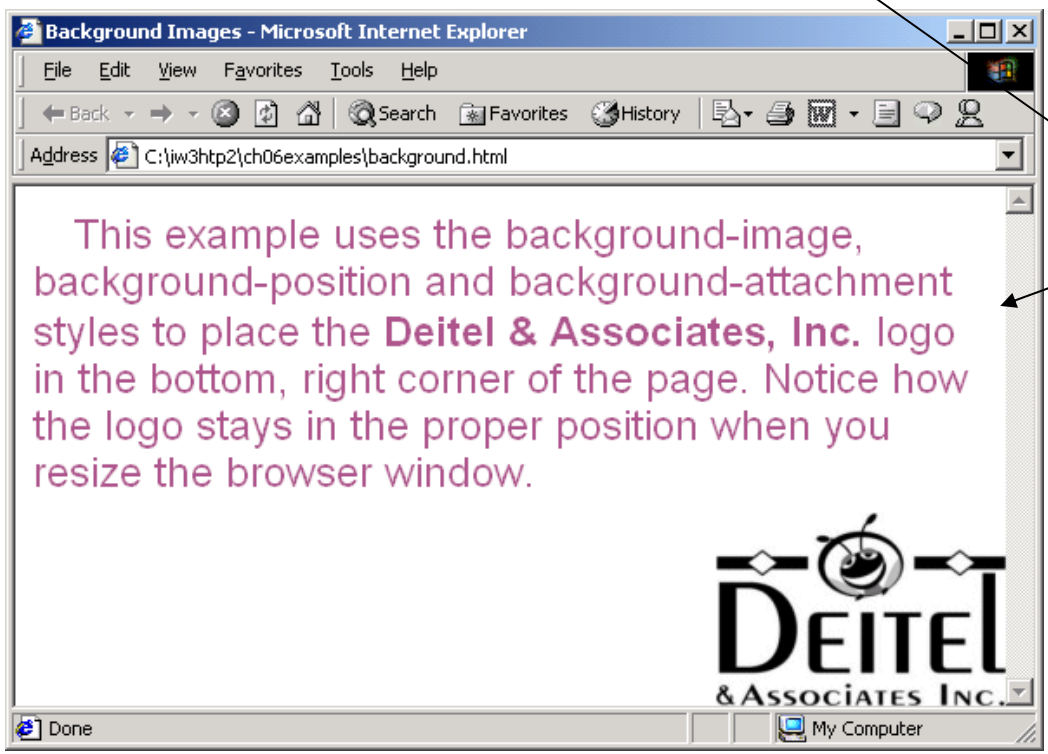


Outline

Background.html

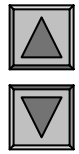
```
29      <body>
30
31         <p>
32         This example uses the background-image,
33         background-position and background-attachment
34         styles to place the <span class = "dark">Deitel
35         & Associates, Inc.</span> logo in the bottom,
36         right corner of the page. Notice how the logo
37         stays in the proper position when you resize the
38         browser window.
39         </p>
40
41      </body>
42   </html>
```

Program Output



Note that no formatting needed to be set in the **p** element itself because it has already been defined in the CSS.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Width.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.11: width.html                        -->
6   <!-- Setting box dimensions and aligning text  -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Box Dimensions</title>
11
12        <style type = "text/css">
13
14           div { background-color: #ffccff;
15                 margin-bottom: .5em }
16        </style>
17
18     </head>
19
20     <body>
21
22        <div style = "width: 20%">Here is some
23        text that goes in a box which is
24        set to stretch across twenty percent
25        of the width of the screen.</div>
26
27        <div style = "width: 80%; text-align: center">
28        Here is some CENTERED text that goes in a box
29        which is set to stretch across eighty percent of
30        the width of the screen.</div>
31
```

Elements placed between **div** tags will be set on their own line with a margin below and after it.

The **width** and **height** attributes of **style** allow the user to indicate the percentage of the width and height of the screen the element can occupy.

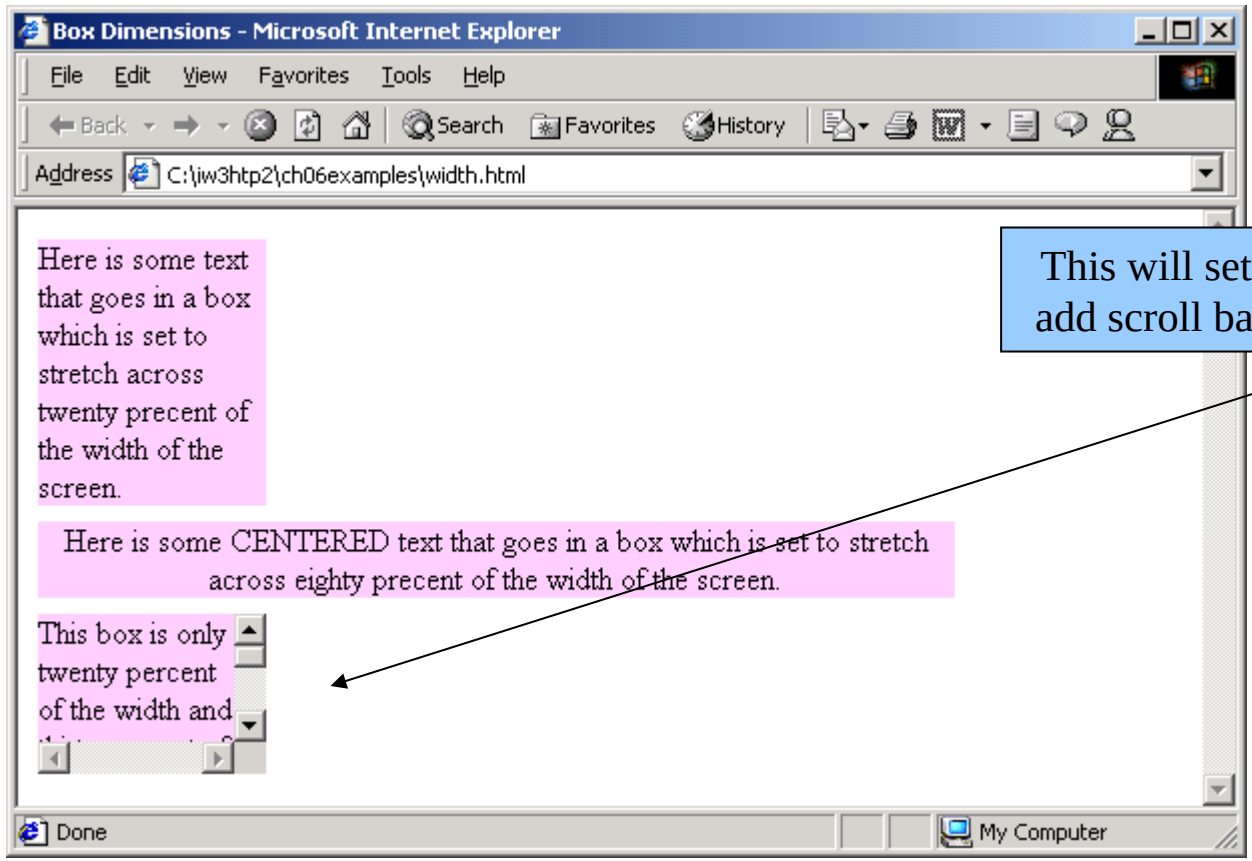
© 2001 Prentice Hall, Inc.
All rights reserved.

Outline

```
32         <div style = "width: 20%; height: 30%; overflow: scroll">
33         This box is only twenty percent of
34         the width and thirty percent of the height.
35         What do we do if it overflows? Set the
36         overflow property to scroll!</div>
37
38      </body>
39   </html>
```

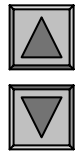
Width.html

Program Output



This will set the **overflow** attribute to add scroll bars for text that overflows.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

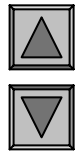
Floating.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.12: floating.html            -->
6   <!-- Floating elements and element boxes -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Flowing Text Around Floating Elements</title>
11
12        <style type = "text/css">
13
14           div { background-color: #ffccff;
15                 margin-bottom: .5em;
16                 font-size: 1.5em;
17                 width: 50% }
18
19           p   { text-align: justify; }
20
21        </style>
22
23     </head>
24
25     <body>
26
27        <div style = "text-align: center">
28           Deitel & Associates, Inc.</div>
29
30        <div style = "float: right; margin: .5em;
31           text-align: right">
32           Corporate Training and Publishing</div>
33
```

The float property allows you to move an element to one side of the screen such that other content in the document then flows around the floated element.

The **margin** property specifies the distance between the edge of the element and any other element on the page.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

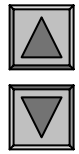
Floating.html

```
34         <p>Deitel & Associates, Inc. is an internationally
35         recognized corporate training and publishing organization
36         specializing in programming languages, Internet/World
37         Wide Web technology and object technology education.
38         Deitel & Associates, Inc. is a member of the World Wide
39         Web Consortium. The company provides courses on Java,
40         C++, Visual Basic, C, Internet and World Wide Web
41         programming, and Object Technology.</p>
42
43         <div style = "float: right; padding: .5em;
44            text-align: right">
45            Leading-edge Programming Textbooks</div>
46
47         <p>The company's clients include many Fortune 1000
48         companies, government agencies, branches of the military
49         and business organizations. Through its publishing
50         partnership with Prentice Hall, Deitel & Associates,
51         Inc. publishes leading-edge programming textbooks,
52         professional books, interactive CD-ROM-based multimedia
53         Cyber Classrooms, satellite courses and World Wide Web
54         courses.<span style = "clear: right"> Here is some
55         unflowing text. Here is some unflowing text.</span></p>
56
57      </body>
58   </html>
```

Padding is the distance between the content inside an element and the element's border.

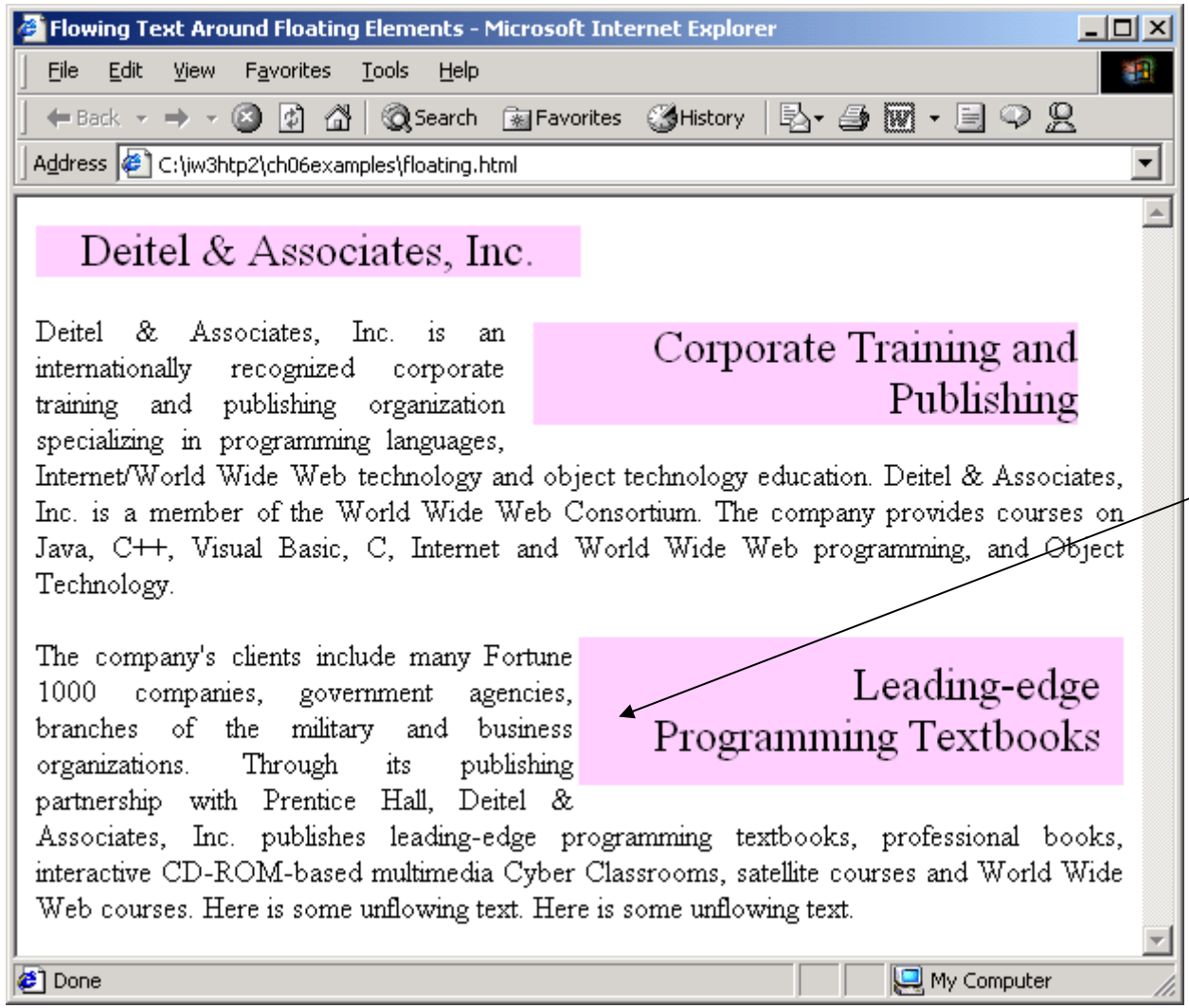
The **clear** property can be used to interrupt the flow of text around a floating element.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Program Output



A floating element with **5 em** padding.

© 2001 Prentice Hall, Inc.
All rights reserved.

6.10 Test Flow and Box Model

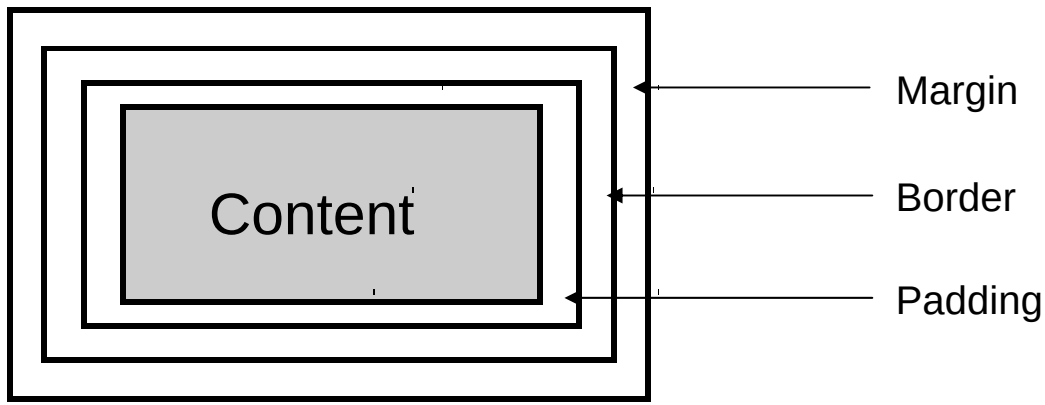
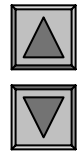


Fig. 6.13 Box model for block-level elements.

© 2001 Prentice Hall, Inc. All rights reserved.



Outline

Borders.html

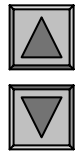
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.14: borders.html       -->
6   <!-- Setting borders of an element -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Borders</title>
11
12        <style type = "text/css">
13
14           body    { background-color: #ccffcc }
15
16           div     { text-align: center;
17                     margin-bottom: 1em;
18                     padding: .5em }
19
20           .thick  { border-width: thick }
21
22           .medium { border-width: medium }
23
24           .thin   { border-width: thin }
25
26           .groove { border-style: groove }
27
28           .inset  { border-style: inset }
29
30           .outset { border-style: outset }
31
32           .red    { border-color: red }
33
34           .blue   { border-color: blue }
```

The **border-width** property is the width of the border around an element.

The **border-style** property is the style of border used.

The **border-color** property is the color of the border.

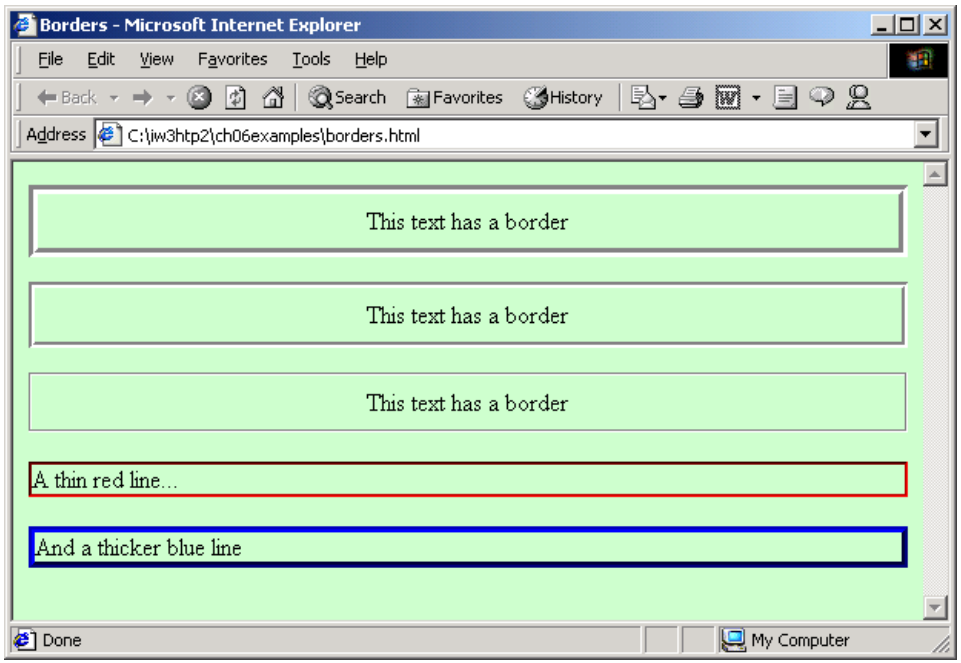
© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Borders.html

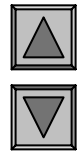
```
35
36          </style>
37       </head>
38
39       <body>
40
41          <div class = "thick groove">This text has a border</div>
42          <div class = "medium groove">This text has a border</div>
43          <div class = "thin groove">This text has a border</div>
44
45          <p class = "thin red inset">A thin red line...</p>
46          <p class = "medium blue outset">
47             And a thicker blue line</p>
48
49       </body>
50    </html>
```



Program Output

A sampling of the different types of borders that can be specified.

© 2001 Prentice Hall, Inc.
All rights reserved.



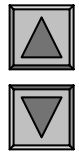
Outline

Borders2.html

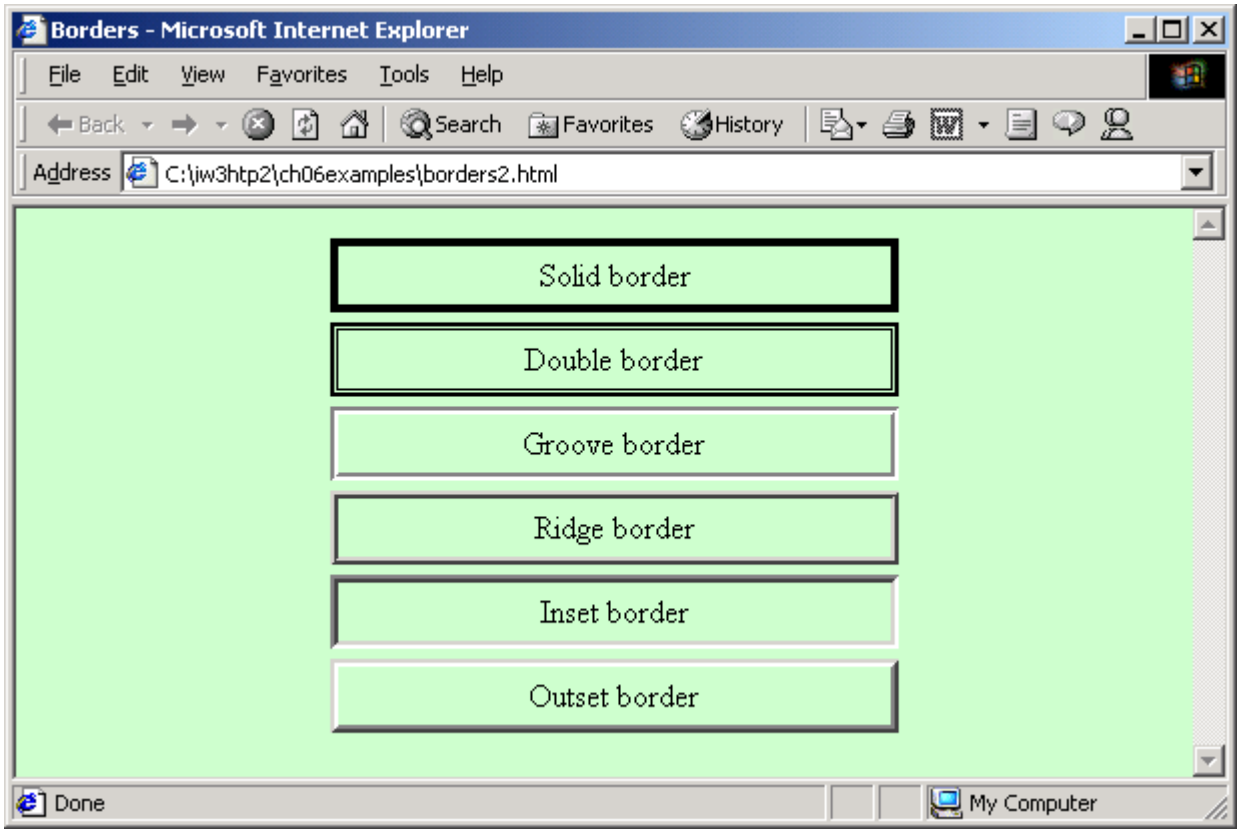
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.15: borders2.html   -->
6   <!-- Various border-styles      -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Borders</title>
11
12        <style type = "text/css">
13
14           body    { background-color: #ccffcc }
15
16           div     { text-align: center;
17                     margin-bottom: .3em;
18                     width: 50%;
19                     position: relative;
20                     left: 25%;
21                     padding: .3em }
22        </style>
23     </head>
24
25     <body>
26
27        <div style = "border-style: solid">Solid border</div>
28        <div style = "border-style: double">Double border</div>
29        <div style = "border-style: groove">Groove border</div>
30        <div style = "border-style: ridge">Ridge border</div>
31        <div style = "border-style: inset">Inset border</div>
32        <div style = "border-style: outset">Outset border</div>
33
34     </body>
35  </html>
```

Specifying a border directly through the **style** attribute of the **div** element.

© 2001 Prentice Hall, Inc.
All rights reserved.

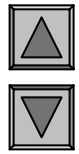


Outline



Program Output

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

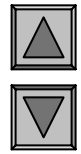
User_absolute.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 6.16: user_absolute.html   -->
6   <!-- User styles                     -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>User Styles</title>
11
12        <style type = "text/css">
13
14           .note { font-size: 9pt }
15
16        </style>
17     </head>
18
19     <body>
20
21        <p>Thanks for visiting my Web site. I hope you enjoy it.
22        </p><p class = "note">Please Note: This site will be
23        moving soon. Please check periodically for updates.</p>
24
25     </body>
26  </html>
```

Author defined style sheets are ones defined within the page.

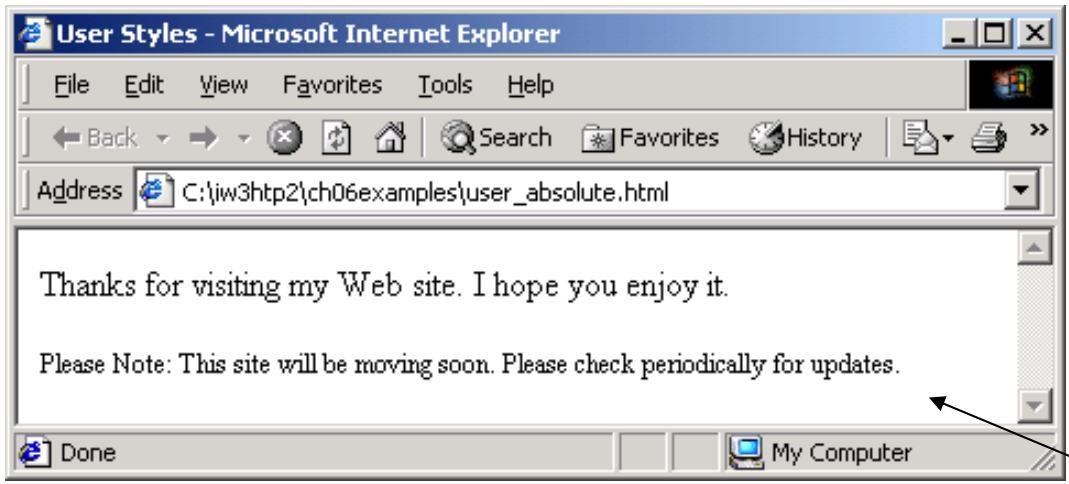
By using absolute measurement (**pt** in this case) the developer will override user defined style sheets.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

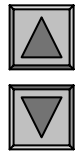
Program Output



Developer defined class applied to this **p** element decrease the font size of the text.

Styles set by the author have higher precedence over the styles set by user style sheets.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Userstyles.css

```
1   /* Fig. 6.17: userstyles.css */
2   /* A user stylesheet         */
3
4   body     { font-size: 20pt;
5              color: yellow;
6              background-color: #000080 }
```

User defined style sheets are usually external. A user's style sheet is not linked to a page, they are set in the browser's options.

© 2001 Prentice Hall, Inc.
All rights reserved.

6.11 User Style Sheets

Setting the user's style sheet in IE.

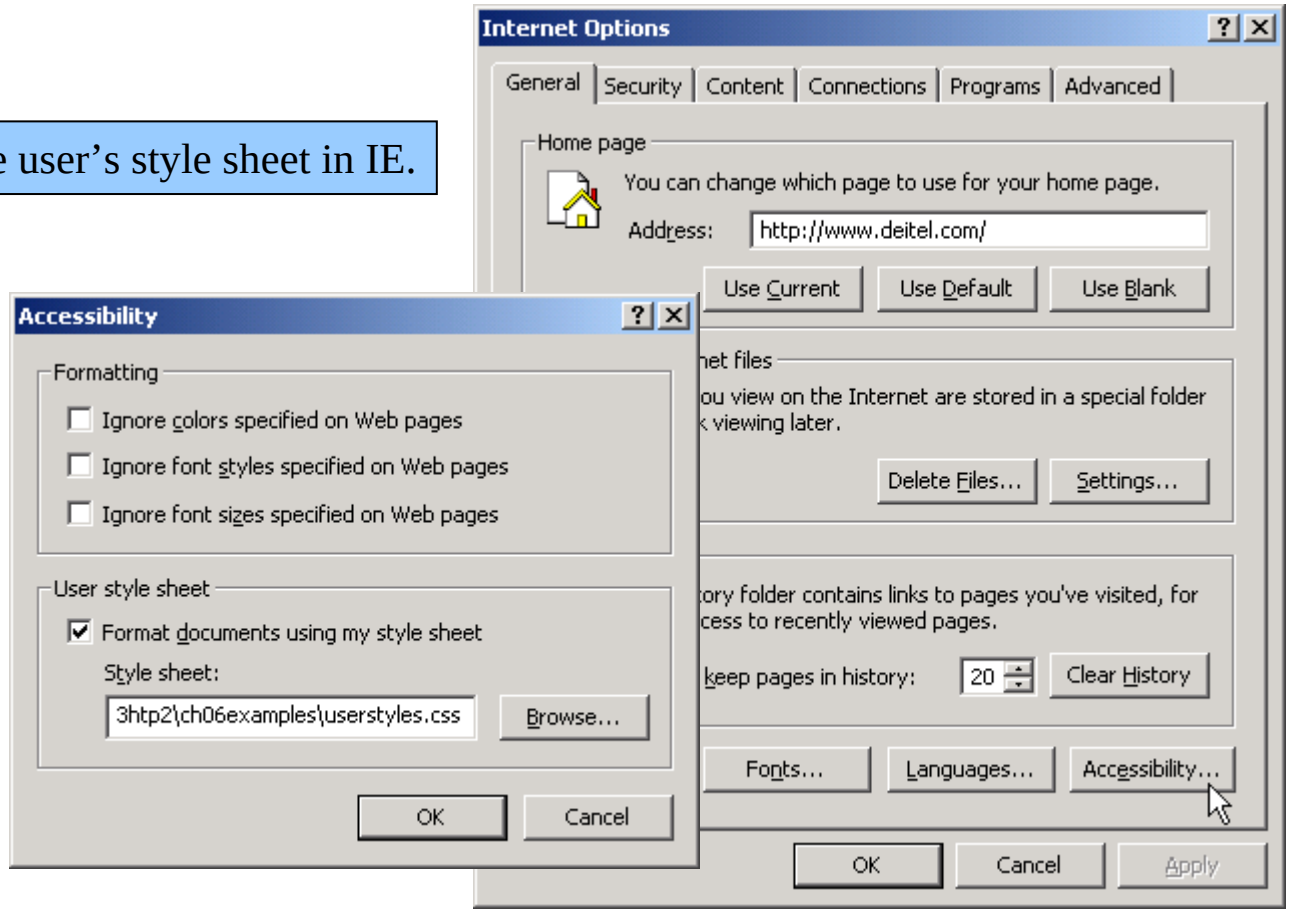


Fig. 6.18 Adding a user style sheet in Internet Explorer 5.5.

© 2001 Prentice Hall, Inc. All rights reserved.

6.11 User Style Sheets

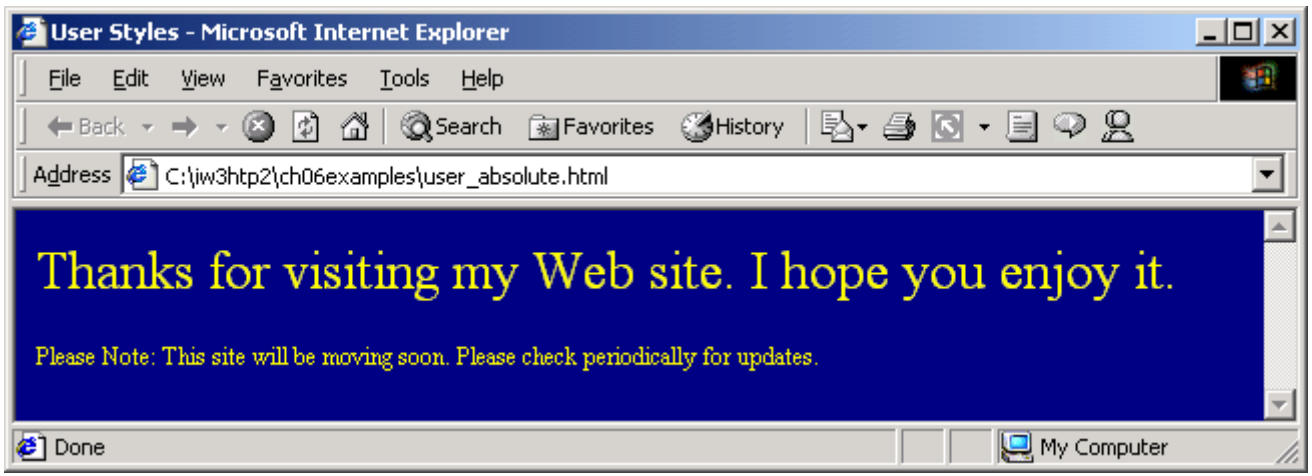
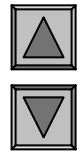


Fig. 6.19 Web page with user styles applied.

© 2001 Prentice Hall, Inc. All rights reserved.



Outline

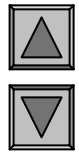
User_relative.html

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 6.20: user_relative.html   -->
6    <!-- User styles                     -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>User Styles</title>
11
12         <style type = "text/css">
13
14            .note { font-size: .75em }
15
16         </style>
17      </head>
18
19      <body>
20
21         <p>Thanks for visiting my Web site. I hope you enjoy it.
22         </p><p class = "note">Please Note: This site will be
23         moving soon. Please check periodically for updates.</p>
24
25      </body>
26   </html>
```

By using relative measurements (in this case, **em**), the developer will not override user defined style sheet formats.

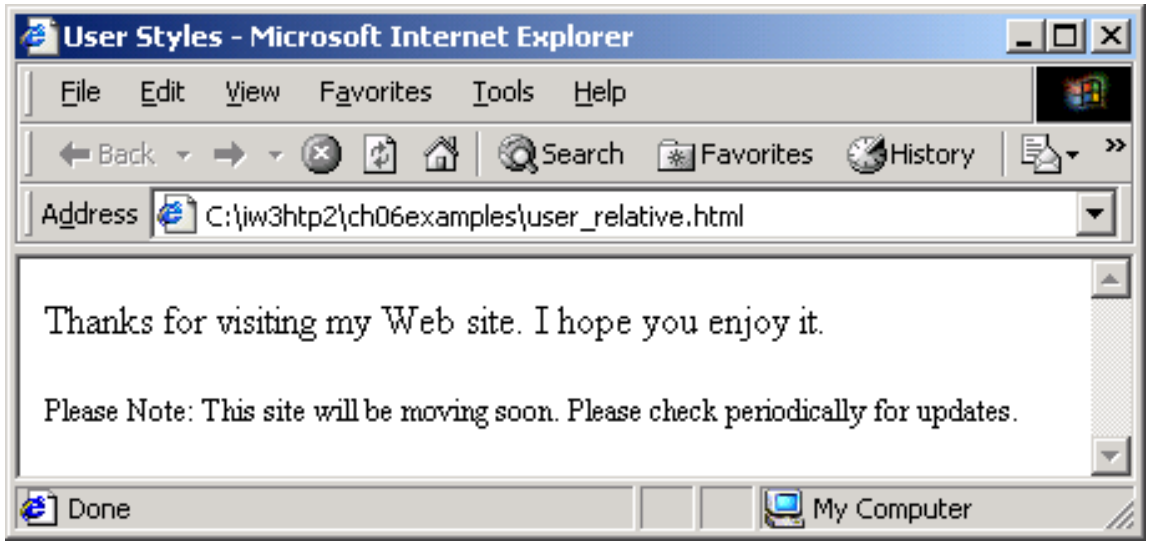
When rendered the font size displayed will be .75 percent times the value of the font size defined in the user style sheet.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Program Output



Output before relative measurement is used to define the font in the document.

© 2001 Prentice Hall, Inc.
All rights reserved.

6.11 User Style Sheets

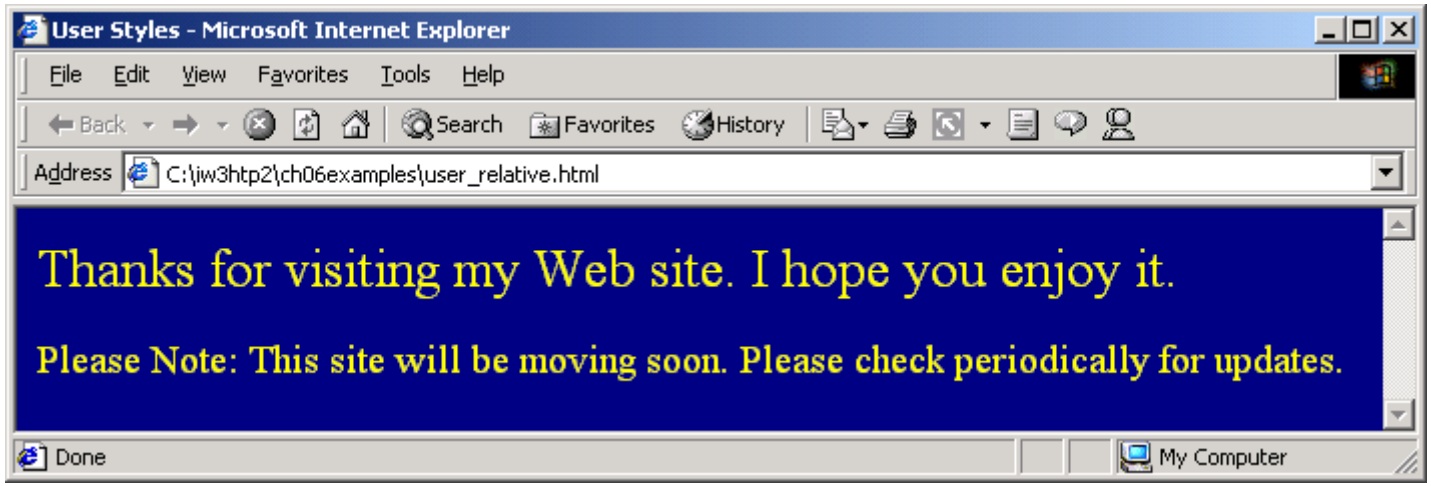


Fig. 6.21 Using relative measurements in author styles.

Output when relative measurement is used. By using relative measurements the user defined styles are not overwritten.

© 2001 Prentice Hall, Inc. All rights reserved.

Locating and Using Data from the Web

Objectives/Agenda

- what is data?
- data formats used on the Web
- finding data
- downloading data
- using that data
  - working with downloaded data
- distributing data
- data quality issues

Defining Data

- data is here defined as statistical information generally stored in a database or analyzed and output in a variety of formats
- data results from research, studies, surveys and data collection efforts
- data can be found on the Web in a variety of formats
  - as raw data (as in databases or spreadsheets) or
  - as processed data in the form of tables, graphs and maps

How Are Data Made Available?

- Narrative - reports
- tables
- databases (possibly including documentation)
- raw data
- figures and graphs
- Comma or tab delimited
- maps
- multiple formats (including facts)
- metadata
- questionnaires

Data Formats

What File Formats Are Being Used?

- ASCII, HTML, DOC, XLS or WKS
- PDF, EXE, MTW (Minitab), DAT
- many others from graphic formats to GIS formats to database formats

Explanations of File Extensions

- ASCII - plain text with no boldings or control characters
- HTML - HyperText Markup Language format
  - usually tables
- DOC - Microsoft Word format
- RTF - Rich Text Format (output from MS Word)
- PDF - Adobe (proprietary)

Database and Spreadsheet

- DBF - Dbase format
  - used by dBASE III, dBASE I, Vfoxbase, Foxpro
- Quattro - database format
- Paradox - database format
- XLS - Microsoft Excel spreadsheet format
- WKS - Lotus 123 spreadsheet format

File Extensions, continued…

- EXE - an executable file which you click on and the file opens up the compressed data into a format you can use
- MTW - Minitab format
- DAT files are basically text files formatted for use with SAS and contain data only
- uuencoded
- TAR - tar files (UNIX) -several files compacted and output as one file
- ZIP - zipped

Common Graphics Formats

- BMP - bitmap file format
- GIF - Graphical Interchange Format
- JPG - Joint Photographic Experts Group (after group which created the standard)
- TIF - another graphics format
- WMF (Windows Metafont)

GIS Formats

- *.TAB - MapInfo
- *.GCM - Geo Concept
- *.AGF - Atlas GIS/Pro
- *.E00 - Arc/Info
- *.DXF - AutoCAD
- *.MIF/*.MID - ArcView
- *.MIF/*.MID, *.DXF - Maptitude
- *.BNA - Atlas Mapmaker
- *.BNA - MapViewer

Production Medium

- bits and bytes
- paper (e.g., summary data: press releases, memos, reports)
- HTML
- 9 track tapes
- cartridges
- CD-ROMS
- Note: formats go out of fashion as new technology comes along

Standards

- will establishing standards help the problem of too many formats?
- yes, no, maybe
  - we still need to deal with the data which has already been produced
  - new formats are being developed all the time

Standards, continued...

- knowing which format the data is in helps IF we know how to use that format
- knowing how to move from one format to another is also useful

Finding Data

Where Do You Find Data on the Web?

- Government at all levels
- Foundations and Think Tanks
- Pharmaceutical companies
- Disease specific Associations (e.g., American Heart Association)
- Textbooks, Reports and other documents
- other sites - search on: health data

Use Search Engines to Find Data

- robots which search across the entire Web
  - use the term: health data to turn up quite a few Websites
- use site specific search engines to locate information within one site

Use Guides

- guides are hierarchical arrangements of Websites usually organized by subject
- several very good sites on statistics appear on the Web

Preformatted Data (Tables or Figures)

- this is often data which is requested time and time again
- e.g., Bureau of Labor Statistics - http://stats.bls.gov/top20.html and http://146.142.4.24/cgi-bin/surveymost?bls
- e.g., White House Briefing Room

Interactivity

- look for sites which enable you to retrieve data according to your needs
- usually searchable databases where you type in the population and variables you want data on
- sites which enable you to retrieve data according to your needs are most often found at government sites such as at CDC WONDER

Downloading Data

Downloading Data - Graphics

- click on the image with right mouse button | then save file to disk
  - insert graphics into your document with the source information appended

HTML Tables

- go to a Website which has an HTML table
- In Netscape: File | Save as File (in HTML format)
  - Netscape will save the whole page
  - delete out what you don't want
- In Excel: File | Open File | change file format using down arrow | click on file
- Excel will turn the page from HTML to a spreadsheet

ASCII Text

- copy and paste

Excel (XLS) Files

- click on file and either open, view and save the file, OR click on the file and save directly to disk

Word Tables or Figures

- for tables, copy and paste; for figures, click once to select, then **Edit | Copy | Edit | Paste**

Databases

- will depend on the database format and whether you can search the database for information
- or download the data in tab- or comma-delimited format

PDF - Moving Tables and Graphs into Office 97

- what you can and cannot do
- you can bring a PDF table or graph into Word and PowerPoint
- you can make the graph or table larger or smaller
- you cannot edit the **contents** of the graph
- a PDF document created from Word cannot be converted back into Word

Moving Tables from PDF...

- open the Adobe Viewer and the document you wish to view
- highlight the text or graphic
- from the **Tools** menu, select **Select Graphics**

Moving Tables from PDF...

- the cursor changes to a cross-hair
- draw a rectangle around the text or graphic by clicking once in the upper left-hand corner of the text and drawing at a diagonal to the other corner

Moving Tables from PDF...

- click on **Edit | Copy**, switch to the receiving document and hit **Edit | Paste**
- the selected material is copied as a WMF (Windows MetaFont) graphic
  - note the WMF is a space hog
  - change the format to GIF or JPG using a graphics program like Paint Shop Pro

Uuencoded Data - What to do with it?

- data sent to you in uuencoded format needs to be uudecoded
- format is uudecode FILENAME.EXT (the file name is often in caps)
- sometimes the uudecoding doesn’t work

Raw Data

- Information that has not been organized, formatted, or analyzed
- raw data can be imported into SPSS if it’s organized in columns
- import tab and comma delimited ASCII files into Excel as well

Other Formats

- binary - A format for representing data used by some applications including EXE files and numerical data

Zipped Files

- you must unzip the files using one of a number of unzipping programs
- my two favorites are:
  - Winzip v 6.2
  - PkZip

Retrieving Data through FTP

- some sites make their data available on anonymous (public) FTP subdirectories
- you FTP to the site, log in as anonymous, give your login address as your password, locate the file, select it and have the server send you the data

Retrieving Data via Gopher

- use a gopher menu or gopher on your browser to burrow down the menu to get the data you want
- when you reach the data it will be in ASCII format (most of the time)
- Gopher format on the Web: gopher://site.address.edu/subdirectory/subdirectory/filename.ext

HTTPing Data to Your Desktop, or, Get Data from a Web Source

- look at Excel Help for “Get data from a Web Source”

Using Data

So, Now We Have the Data, Now What?

- What tools can we use?
  - Excel
  - Access
  - Statistical Software
  - other software

One Example of Converting Data from One Format to Another

- you may need to convert a tab or comma delimited file to Excel
- in Excel, use the Import command: **File | Open | Type of File | Text Files |** select file you wish to import (double-click)
- the Text Import Wizard will help you specify into which column you want to put your text

File Translation Utility

- DataImport Version 5 - $189/single user
- http://www.spaldingsoft.com/di/dataimpt.htm
- pulls data from virtually any computer report and converts it into nearly 40 spreadsheet and database formats
  - e.g., conversion of hospital financial and admissions data for analysis

What is an Excel Web Query?

- A query that can retrieve data from several locations
  - Internet (World Wide Web)
  - Intranet
  - Hard drive
  - using HTTP, FTP, Gopher

Web Queries - Basic Process

- open a new workbook in Excel
- create a Web query or use an existing query
  - queries are located in the Queries folder on your installation disk
  - you may need to install them
- activate MS Internet Explorer / Netscape Communicator
- connect to the Internet through your ISP
- run the query
- examine the data

Create Your Own Web Query (.iqy) files

- open a new workbook in Excel
- you can create your own Web Queries
- read the Excel Help Documentation to get details
- look under Web Queries, then Create a Web Query

Quality Issues

Data Quality

- how do you evaluate the quality of data on a site?
- consider who is the author (government agency? pharmaceutical company, interested layperson, who?)
- is complete documentation available?
  - variables defined and located on media
- is the data in a format you can easily used?

Data Quality, continued...

- How current is the data?
- is the source of the data made clear?
  - what is the sample size?
  - weighting and other statistical necessities?
- is it easily transported to a statistical package such as Epi Info, SAS, SPSS?

Data Quality, continued...

- what format is the data in? AND do you have the hardware to work with the data
  - 9-track tape? Is the tape medium old or has it been refreshed? Old tapes sometime cause data loss when you try to read them
  - cartridge?
  - CD-ROM?
- why was the data collected?
- Does the data collected reflect “real life”

Data Quality - Role of Documentation and Metadata

- documentation helps the viewer of the data understand how the data was created and how to use it
- metadata is data about the data

Distributing Data

Making Data Available

- email attachments
- Web using Excel and HTML
- diskette and Zip cartridges
- CD-ROM

Convert an Excel Spreadsheet to HTML

- File | Save as HTML
- the HTML wizard will appear and will ask you to specify a range of cells and charts to convert
- you decide if you wish to create a whole new HTML page or just a table which is then inserted into anothe HTML page
- tell Excel where you want to put the file
- then click Finish

Really Easy Distribution of Data via Email

- data can be sent as ASCII text in the body of an email or in other formats as attachments
- people can create data in one format but might have to change it to another format to send it over the Internet
  - e.g., create a graph in Excel and send in a uuencoded format or as an attachment
- attachments are preferred to uuencoding a file
  - unless you know the secret of uudecoding the file, how can you retrieve the graph?

Document Object Model (DOM)

- How to provide uniform access to structured documents in diverse applications (parsers, browsers, editors, databases)?
- Overview of W3C DOM Specification
  - second one in the “XML-family” of recommendations
    - » Level 1, W3C Rec, Oct. 1998
    - » Level 2, W3C Rec, Nov. 2000
    - » Level 3, W3C Working Draft (January 2002)
- What does DOM specify, and how to use it?

DOM: What is it?

- An object-based, language-neutral API for XML and HTML documents
  - allows programs and scripts to build documents, navigate their structure, add, modify or delete elements and content
  - Provides a foundation for developing
    querying, filtering,
    transformation, rendering etc.
    applications on top of DOM implementations
- In contrast to "Serial Access XML" could think as "Directly Obtainable in Memory"

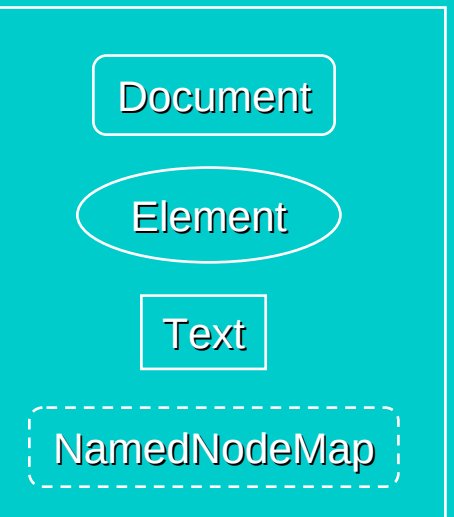
DOM *structure model*

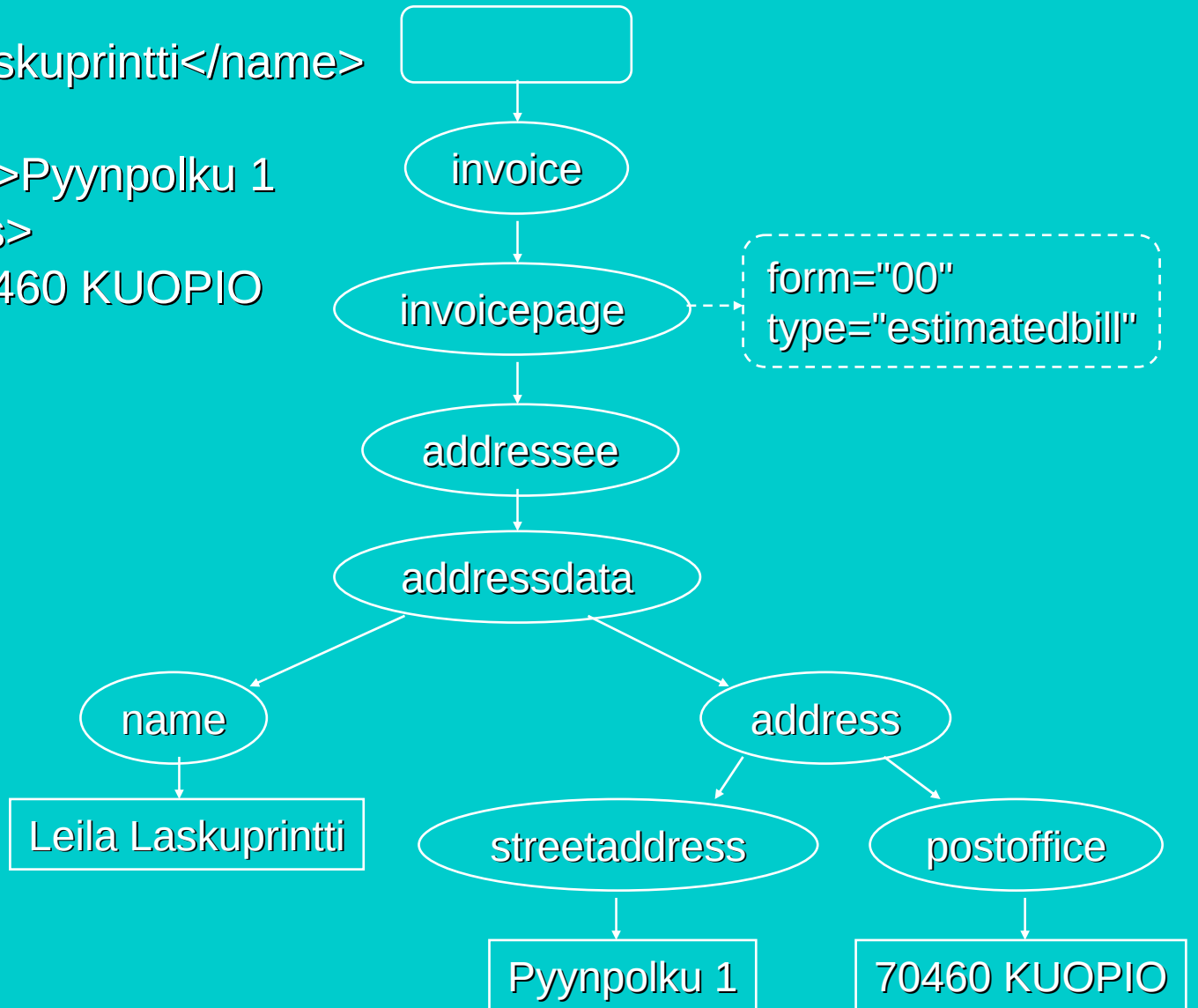
- Based on O-O concepts:
  - *methods* (to access or change object's state)
  - *interfaces* (declaration of a set of methods)
  - *objects* (encapsulation of data and methods)
- Roughly similar to the XSLT/XPath data model (to be discussed later)

  ≈ a parse tree
  - Tree-like structure implied by the abstract relationships defined by the programming interfaces;
    Does not necessarily reflect data structures used by an implementation (but probably does)

```
<invoice>
 <invoicepage form="00"
                    type="estimatedbill">
  <addressee>
   <addressdata>
    <name>Leila Laskuprintti</name>
    <address>
     <streetaddress>Pyynpolku 1
     </streetaddress>
     <postoffice>70460 KUOPIO
     </postoffice>
    </address>
   </addressdata>
  </addressee> ...
```

DOM structure model





Structure of DOM Level 1

I: DOM **Core Interfaces**

- **Fundamental interfaces**
  - » basic interfaces to structured documents
- Extended interfaces
  - » XML specific: CDATASection, DocumentType, Notation, Entity, EntityReference, ProcessingInstruction

II: DOM **HTML Interfaces**

- more convenient to access HTML documents
- (we ignore these)

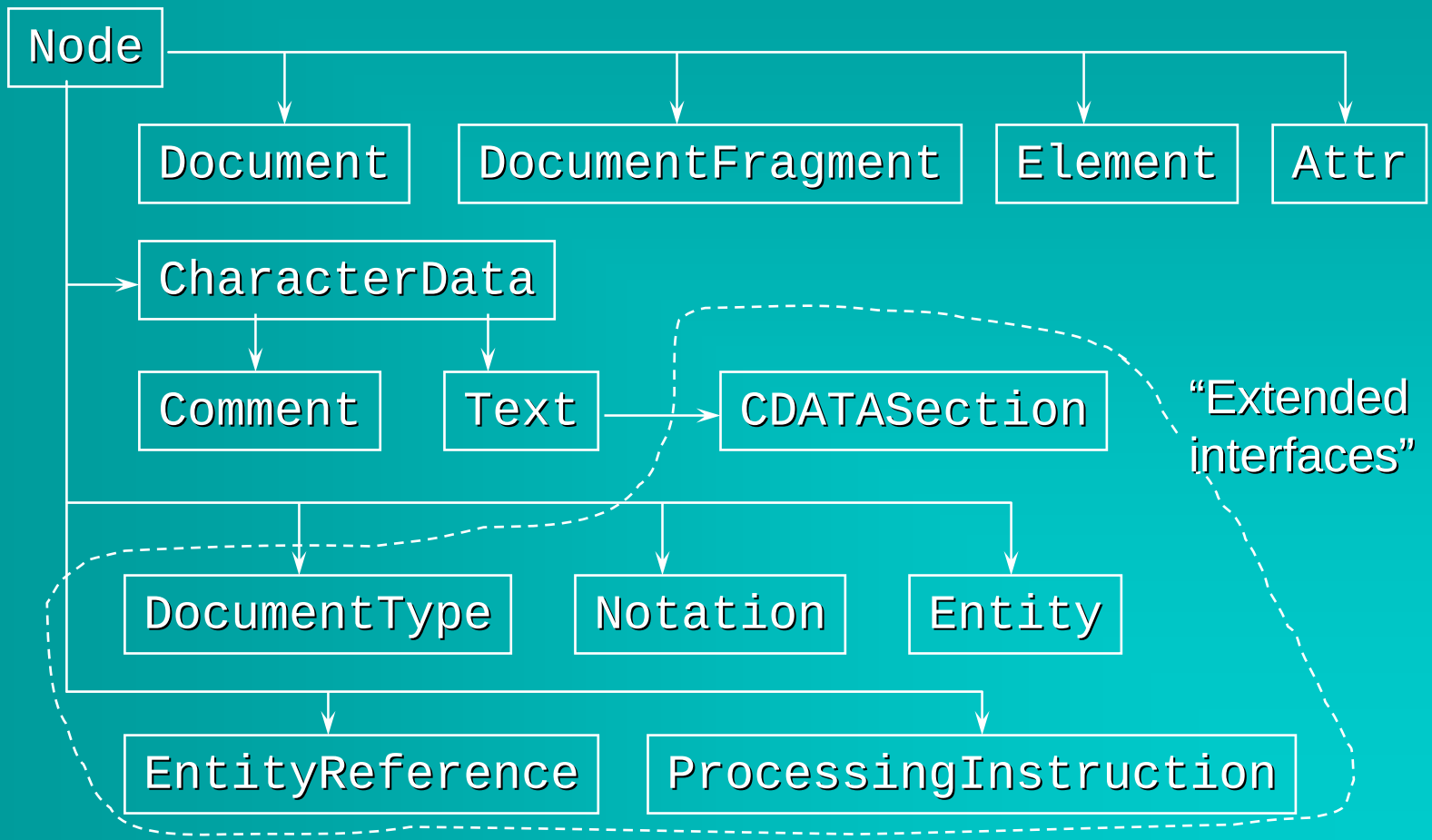
DOM Level 2

- Level 1: basic representation and manipulation of document structure and content
  (No access to the contents of a DTD)
- DOM Level 2 adds
  - support for namespaces
  - accessing elements by ID attribute values
  - optional features
    - » interfaces to document views and style sheets
    - » an event model (for, say, user actions on elements)
    - » methods for traversing the document tree and manipulating regions of document (e.g., selected by the user of an editor)
  - Loading and writing of docs **not** specified (-> Level 3)

DOM Language Bindings

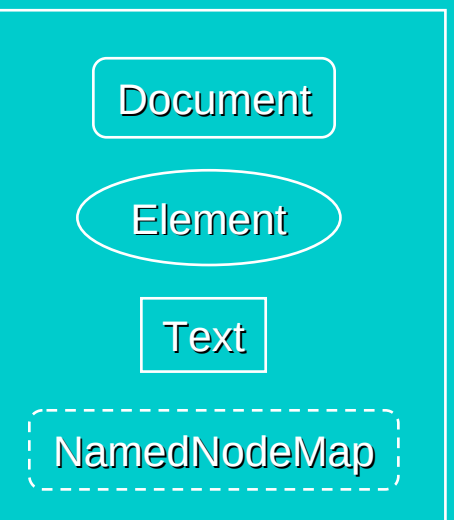
- Language-independence:
  - DOM interfaces are defined using OMG Interface Definition Language (IDL; Defined in Corba Specification)
- Language bindings (implementations of DOM interfaces) defined in the Recommendation for
  - Java and
  - ECMAScript (standardised JavaScript)

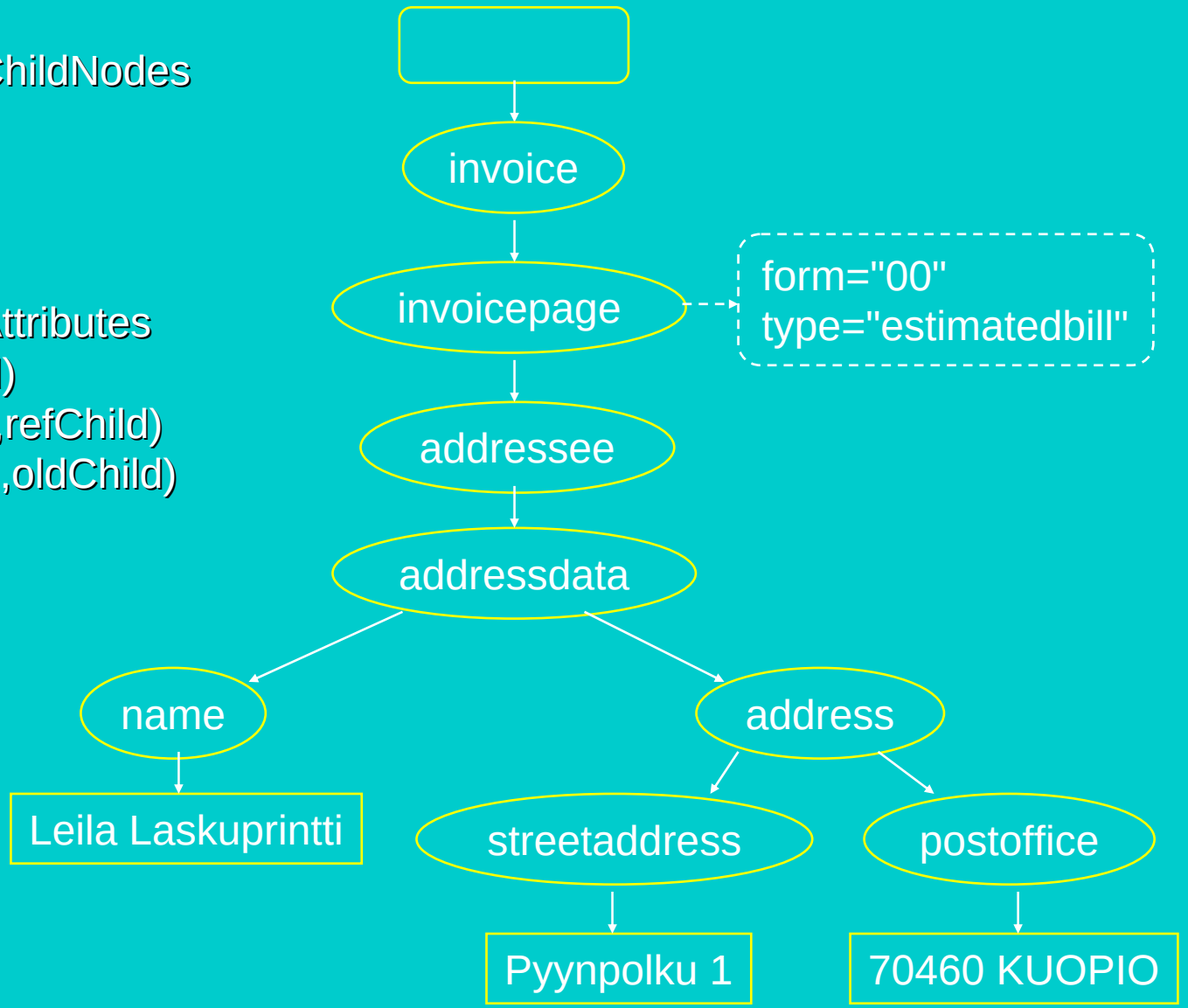
Core Interfaces: Node & its variants



DOM interfaces: Node

Node
getNodeType
getNodeValue
getOwnerDocument
getParentNode
hasChildNodes getChildNodes
getFirstChild
getLastChild
getPreviousSibling
getNextSibling
hasAttributes getAttributes
appendChild(newChild)
insertBefore(newChild,refChild)
replaceChild(newChild,oldChild)
removeChild(oldChild)





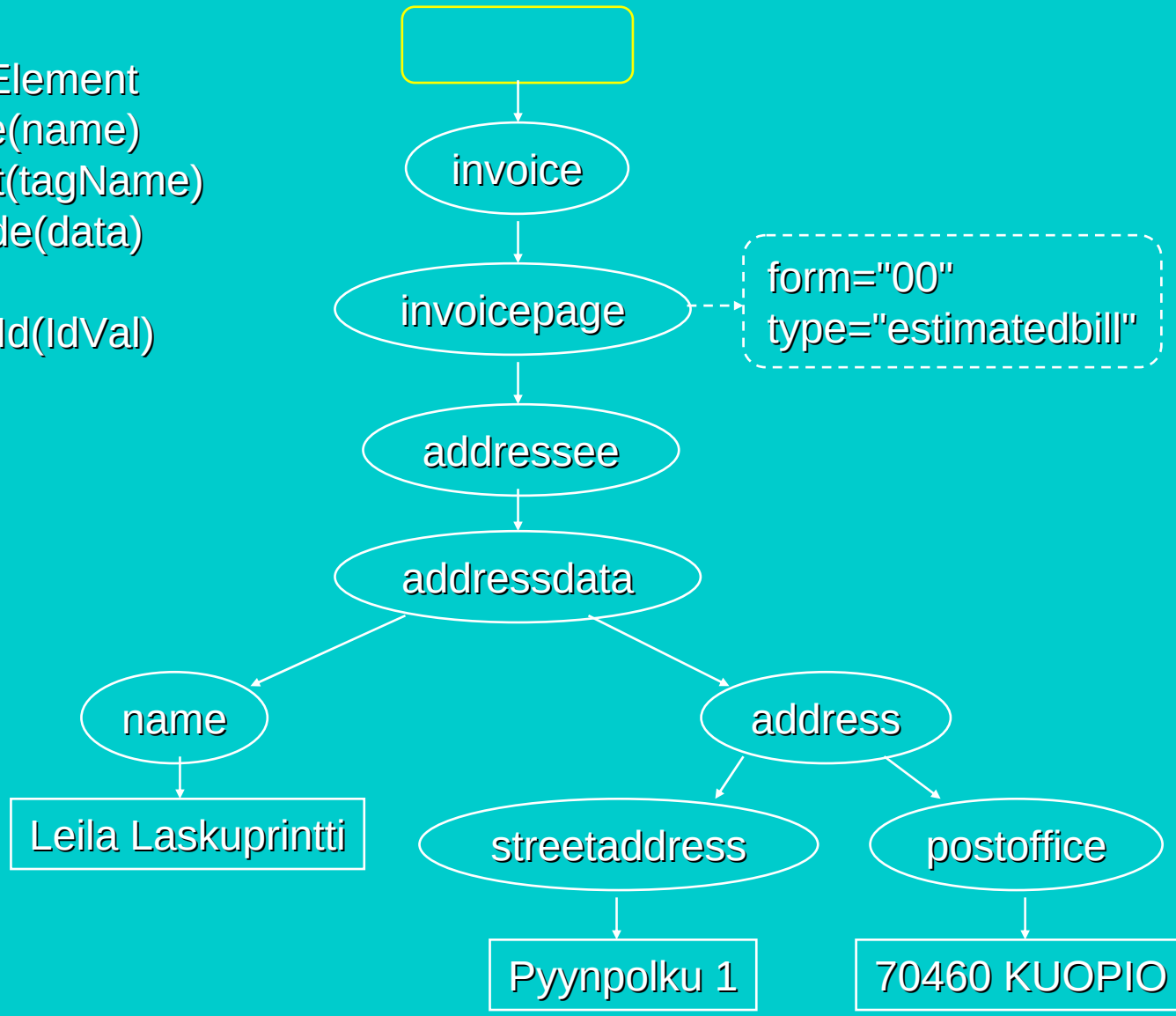
Object Creation in DOM

- Each DOM object *X* lives in the context of a `Document`: *X*`.getOwnerDocument()`
- Objects implementing interface *X* are created by factory methods
  D`.create`*X*`(…)`,
  where D is a `Document` object. E.g:
  - `createElement("A"),`
    `createAttribute("href"),`
    `createTextNode("Hello!")`
- Creation and persistent saving of `Documents` left to be specified by implementations

DOM interfaces: Document

Node

Document
getDocumentElement
createAttribute(name)
createElement(tagName)
createTextNode(data)
getDocType()
getElementById(IdVal)



Document

Element

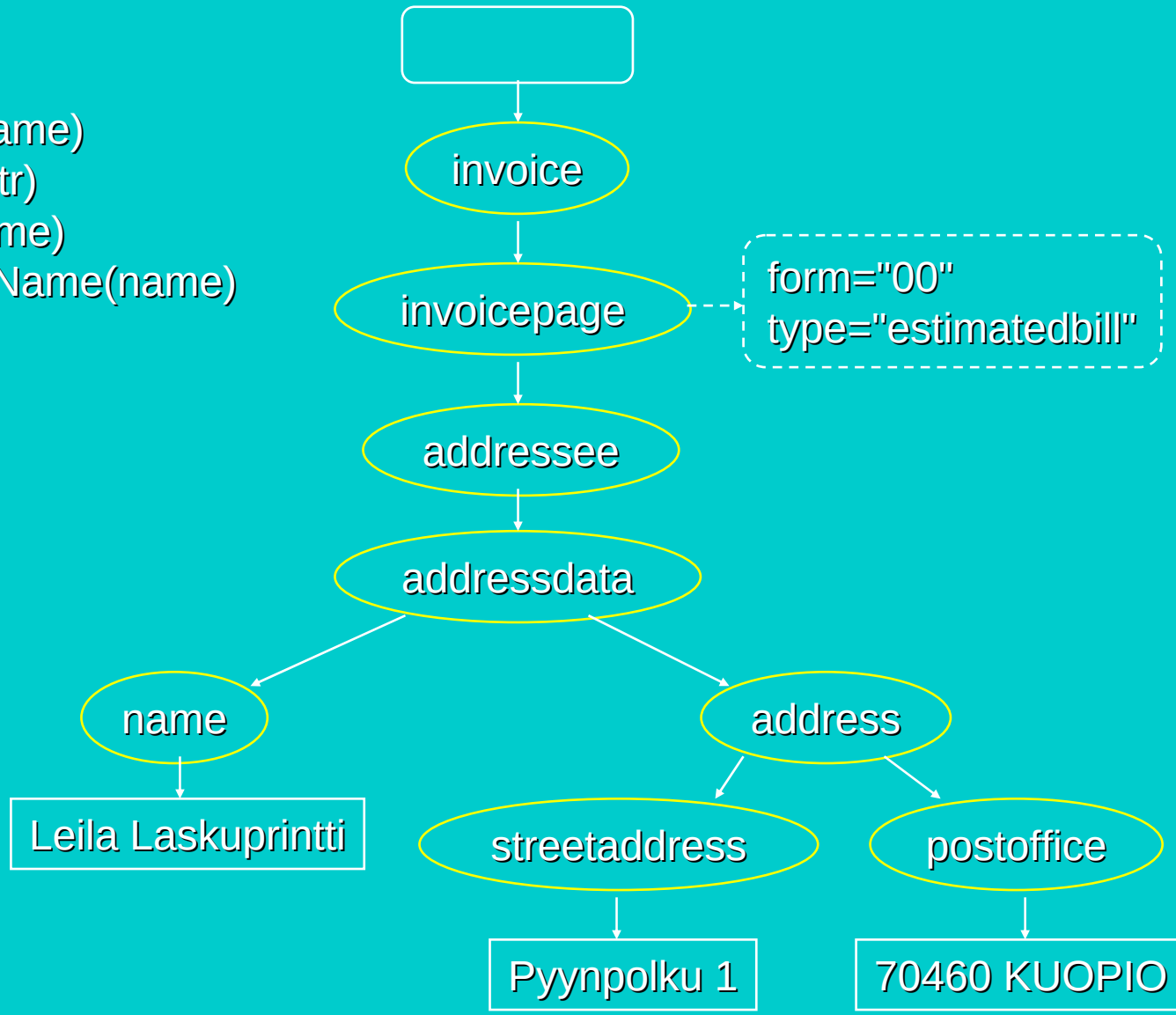
Text

NamedNodeMap

DOM interfaces: Element

Node

Element
getTagName
getAttributeNode(name)
setAttributeNode(attr)
removeAttribute(name)
getElementsByTagName(name)
hasAttribute(name)



Document

Element

Text

NamedNodeMap

Accessing properties of a Node

- **`Node.getNodeName`**()
  - for an `Element = getTagName()`
  - for an `Attr:` the name of the attribute
  - for `Text = "#text"` etc
- **`Node.getNodeValue()`**
  - content of a text node, value of attribute, …; **null** for an `Element` (!!)
    (in XSLT/Xpath: the full textual content)
- **`Node.getNodeType()`**: numeric constants (1, 2, 3, …, 12) for `ELEMENT_NODE, ATTRIBUTE_NODE, TEXT_NODE, …, NOTATION_NODE`

Content and element manipulation

- Manipulating **`CharacterData`** D:
  - D**`.substringData(`**offset, count**`)`**
  - D**`.appendData(`**string**`)`**
  - D**`.insertData(`**offset, string**`)`**
  - D**`.deleteData(`**offset, count**`)`**
  - D**`.replaceData(`**offset, count, string**`)`**
    `(= delete + insert)`
- Accessing attributes of an **`Element`** object E:
  - E**`.getAttribute(`**name**`)`**
  - E**`.setAttribute(`**name, value**`)`**
  - E**`.removeAttribute(`**name**`)`**

Additional Core Interfaces (1)

- **`NodeList`** for ordered lists of nodes
  - e.g. from `Node.getChildNodes()` or `Element.getElementsByTagName("name")`
    - » all descendant elements of type `"name"` in document order (wild-card `"*"` matches any element type)
- Accessing a specific node, or iterating over all nodes of a **`NodeList`**:
  - E.g. Java code to process all children:

```
for (i=0;
 i<node.getChildNodes().getLength();
 i++)
 process(node.getChildNodes().item(i));
```

Additional Core Interfaces (2)

- **`NamedNodeMap`** for unordered sets of nodes accessed by their name:
  - e.g. from `Node.getAttributes()`
- `NodeLists` and `NamedNodeMaps` are "live":
  - changes to the document structure reflected to their contents

DOM: Implementations

- Java-based parsers
  e.g. **IBM XML4J, Apache Xerces, Apache Crimson**
- MS IE5 browser: COM programming interfaces for C/C++ and MS Visual Basic, ActiveX object programming interfaces for script languages
- XML::DOM (Perl implementation of DOM Level 1)
- Others? Non-parser-implementations? (Participation of vendors of different kinds of systems in DOM WG has been active.)

A Java-DOM Example

- A stand-alone toy application `BuildXml`
  - either creates a new `db` document with two `person` elements, or adds them to an existing `db` document
  - based on the example in Sect. 8.6 of *Deitel et al: XML - How to program*
- Technical basis
  - DOM support in Sun JAXP
  - native XML document initialisation and storage methods of the JAXP 1.1 default parser (Apache Crimson)

Code of `BuildXml` (1)

- Begin by importing necessary packages:

```
import java.io.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
// Native (parse and write) methods of the
// JAXP 1.1 default parser (Apache Crimson):
import org.apache.crimson.tree.XmlDocument;
```

Code of `BuildXml` (2)

- Class for modifying the document in file `fileName`:

```
public class BuildXml {
  private Document document;

public BuildXml(String fileName) {
  File docFile = new File(fileName);
  Element root = null; // doc root elemen
  // Obtain a SAX-based parser:
  DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
```

Code of BuildXml (3)

```
try { // to get a new DocumentBuilder:
   documentBuilder builder =
      factory.newInstance();

  if (!docFile.exists()) { //create new doc
     document = builder.newDocument();
     // add a comment:
     Comment comment =
           document.createComment(
                 "A simple personnel list");
     document.appendChild(comment);
     // Create the root element:
     root = document.createElement("db");

     document.appendChild(root);
```

Code of BuildXml (4)

... or if docFile already exists:

```
  } else   { // access an existing doc
    try { // to parse docFile
      document = builder.parse(docFile);
      root = document.getDocumentElement();
    } catch (SAXException se) {
      System.err.println("Error: " +
            se.getMessage() );
      System.exit(1);
    }

    /* A similar catch for a possible IOException */
```

Code of `BuildXml` (5)

- Create and add two child elements to `root`:

```
Node personNode =
    createPersonNode(document, "1234",
                     "Pekka", "Kilpeläinen");
root.appendChild(personNode);
personNode =
    createPersonNode(document, "5678",
                         "Irma", "Könönen");
root.appendChild(personNode);
```

Code of `BuildXml` (6)

- Finally, store the result document:

```
try { // to write the
      // XML document to file fileName
  ((XmlDocument) document).write(
     new FileOutputStream(fileName));
} catch ( IOException ioe ) {
      ioe.printStackTrace();
}
```

Subroutine to create person elements

```
public Node createPersonNode(Document document,
      String idNum, String fName, String lName) {
  Element person =
            document.createElement("person");
  person.setAttribute("idnum", idNum);
  Element firstName =
            document. createElement("first");
   person.appendChild(firstName);
  firstName. appendChild(
            document. createTextNode(fName) );
  /* ... similarly for a lastName */
  return person;
}
```

The main routine for `BuildXml`

```
public static void main(String args[]){
  if (args.length > 0) {
     String fileName = args[0];
     BuildXml buildXml = new
                  BuildXml(fileName);
  } else {
     System.err.println(
           "Give filename as argument");
  };
} // main
```

Summary of XML APIs

- XML processors make the structure and contents of XML documents available to applications through APIs
- Event-based APIs
  - notify application through parsing events
  - e.g., the SAX call-back interfaces
- Object-model (or tree) based APIs
  - provide a full parse tree
  - e.g, DOM, W3C Recommendation
  - more convenient, but may require too much resources with the largest documents
- Major parsers support both SAX and DOM

Email Broadcast

Why is email a good marketing tool?

- Effective
- Direct
- Cheap
- Immediate
- Interactive

Email Broadcast Key Features

- Message Content Management
- Contact Information Integration
- Email Generation and Delivery Engine
- Reporting

Message Content Management

- A key area for improvement over E-Merge v3.

- Want to devolve responsibility for creating content without losing control over format and branding?

- Message Templates:
  - Easy to use Template Designer
  - Guides end-user through creation of an email broadcast
  - Prevents users from modifying underlying content
  - Can contain any additional configuration information required
  - Messages can be generated and sent immediately, or submitted for approval before sending

Contact Integration

- Tikit solution integrates directly with InterAction – your single "trusted source" of contact information.

- Hosted or off-site solutions will not integrate directly with InterAction.
  - Inevitably, you will end up maintaining two databases – InterAction and your "marketing" database!
  - Significant technical and process problems with keeping the two sets of data synchronised.
  - Potential compliance issues.

Email Generation and Delivery

- Continued support for both email client and SMTP delivery
- Background message generation
  - Free up client PC for other tasks almost immediately
- Remote message generation
  - Message submitted to central point for generation
  - Messages can be generated off-site
- Greatly improved configuration options
  - Delivery options can be controlled by content template

Reporting

- Permanent record of all email broadcasts
- Much closer integration with ReAction Server technology
  - Single point of management
  - Significantly easier tracking of message open and click-throughs
- Will leverage new reporting platform in InterAction 5.5

Email Broadcast Framework Concepts

- An API for handling all aspects of email marketing
- Scalable – from desktop to enterprise
- Extensible by 3rd parties
- Designed for current and future needs, based on past experience

Email Broadcast Framework Concepts

- Message Object
  - Contains all the information about an email broadcast
  - Can be saved/loaded, or transmitted across a network
  - Supports multiple content formats
- Contact Information Providers
  - Standard interface for providing contact information
- Delivery Engines
- Content Tokens
  - Dynamically assigned content within each message
  - Can be personalised for each recipient
  - Can be collected from a user at run-time
  - Supports default values

Email Broadcast Framework Concepts

- Message Plugins
  - A standard interface for interacting with EBF messages
  - Dynamically loaded & configured at run-time
  - Can be developed by Tikit, customers, or 3rd parties
- Plugin examples
  - Custom exclusion rules
  - Spam scoring
  - Custom content addition (e.g. disclaimers)
  - Content checking (graphics, links, etc)
  - Automatic upload of graphics etc
  - Logging
  - Compliance procedures (e.g. BCC recipients)

Email Broadcast Framework

Technology

- Completely redesigned on Microsoft .NET platform
- Modular, not monolithic
- Flexible deployment options
  - Multiple clients (WinForms, Web)
- Easily extensible

ReAction Server.Net

- Performance is key…
- 5.5 Events Module (maybe sooner?)
- Improved design tools
- Enhanced integration:
  - Event management/workflow
  - Websites/Content Management Systems
  - Credit card billing

ReAction Server

- Delivers InterAction functionality directly to an organisation's contacts, enabling contacts to interact with data the firm keeps on them
- Highly flexible and configurable
- Designed to automate existing InterAction processes
- HTML template files allows complete control of visual appearance
- Includes easy to use wizard for configuring most functionality

ReAction Server

Sample uses:

- Event invitations
- Newsletter memberships
- Data Protection compliance
- E-Marketing regulation compliance
- Interest Monitoring (click-thrus)
- Surveys

ReAction Server

Design Goals:

- Flexible
- Minimal changes to existing IA design and processes
- Secure
- Easy administration & management
- No development skills required to implement

HTML Search Forms and CGI

HTML is a tool that provides for

- Describing and representing the structure and contents of web documents.
- Linking that text to other resources.
- A CGI program has to somehow get from the user any data to be processed. That's where HTML forms come into play

There are many uses of forms on Web

- As surveys.
- Online order forms.
- Feedback.
- As a means to search a database.
- Or other functions for which user input is required.

Forms can be created with the following features:

- Selectable lists.
- Radio buttons.
- Checkboxes.
- Text Fields.
- Submit and Reset buttons.

General form of FORM tag

- <FORM ACTION="URL" METHOD="GET" > ... Form Element tags ... </FORM>.
- <FORM

ACTION =http://www.ncsi.iisc.ernet.in/ncsi/test.pl

MTHOD ="POST"> ... </FORM>.

- ACTION attribute tells the URL where the information in the form is to be sent.
- Default method is GET (Takes the information entered into the form, and adds this information to the URL specified by the ACTION attribute.
- POST method sends the information from the form as an encoded stream of data to the web (Difference with GET method)

Quick reference to Form Element Tag

<INPUT TYPE="text" NAME="name" VALUE ="value" SIZE=size>	Creates Text Field
<INPUT TYPE="password" NAME="name" VALUE="value" SIZE=size>	Password Field
<INPUT TYPE="hidden" NAME="name" VALUE="value">	Hidden Field
<INPUT TYPE="checkbox" NAME="name" VALUE="value">	Checkbox

<INPUT TYPE="radio" NAME="name" VALUE="value">	Radio Button
<SELECT NAME="name" SIZE=1> <OPTION SELECTED>one <OPTION>Two : </SELECT>	Dropdown List
<SELECT NAME="name" SIZE=n MULTIPLE>	Scrolling List
<TEXTAREA ROWS=yy COLS=xx NAME="name"> .. </TEXTAREA>	Multiple Test Fields
<INPUT TYPE="submit" VALUE="Submit" >	Submit button
<INPUT TYPE="reset" VALUE="Reset" >	Reset Button

```
<HTML>
<BODY>
<H1 ALIGN=CENTER><FONT SIZE=+2>USER INFORMATION
FORM</FONT></H1>
<FORM ACTION="/Scripts/simple.pl" METHOD="post" >
User Name:<INPUT TYPE="text" NAME ="uname" SIZE=30>
<BR>
Service Type:<SELECT
NAME="service"><OPTION>CAS<OPTION>CDRS
<OPTION>COPSAT<OPTION>DDS<OPTION>FDSS<OPTION>ISS
<OPTION>OSS<OPTION>SAS<OPTION>DA
</SELECT>
<BR>
<H4><NOBR>SUBJECT AREA:</H4>
<INPUT TYPE="checkbox" NAME="agriculture" >Agriculture
<INPUT TYPE="checkbox" NAME="biology" >Biology
<INPUT TYPE="checkbox" NAME="biomedicine" >Biomedicine
<INPUT TYPE="checkbox" NAME="chemistry" >Chemistry
```


<INPUT TYPE="radio" NAME="database">AGRIS
<INPUT TYPE="radio" NAME="database">AHEAD
<INPUT TYPE="radio" NAME="database">BIOSIS
<INPUT TYPE="radio" NAME="database">CAB

Date Entered:<INPUT TYPE="text" NAME ="entrydate"
SIZE=10">(dd/mm/yyyy)

<CENTER>
<INPUT TYPE= "submit" VALUE="Submit Form">
<INPUT TYPE= "reset" VALUE="Clear Form">
<CENTER>
</FORM>
</BODY>
</HTML>

CGI (Common Gateway Interface) Concepts

- CGI, permits interactivity between a client and a host operating system through a World Wide Web via the Hyper Text Transfer Protocol (HTTP).

Writing CGI programs involves

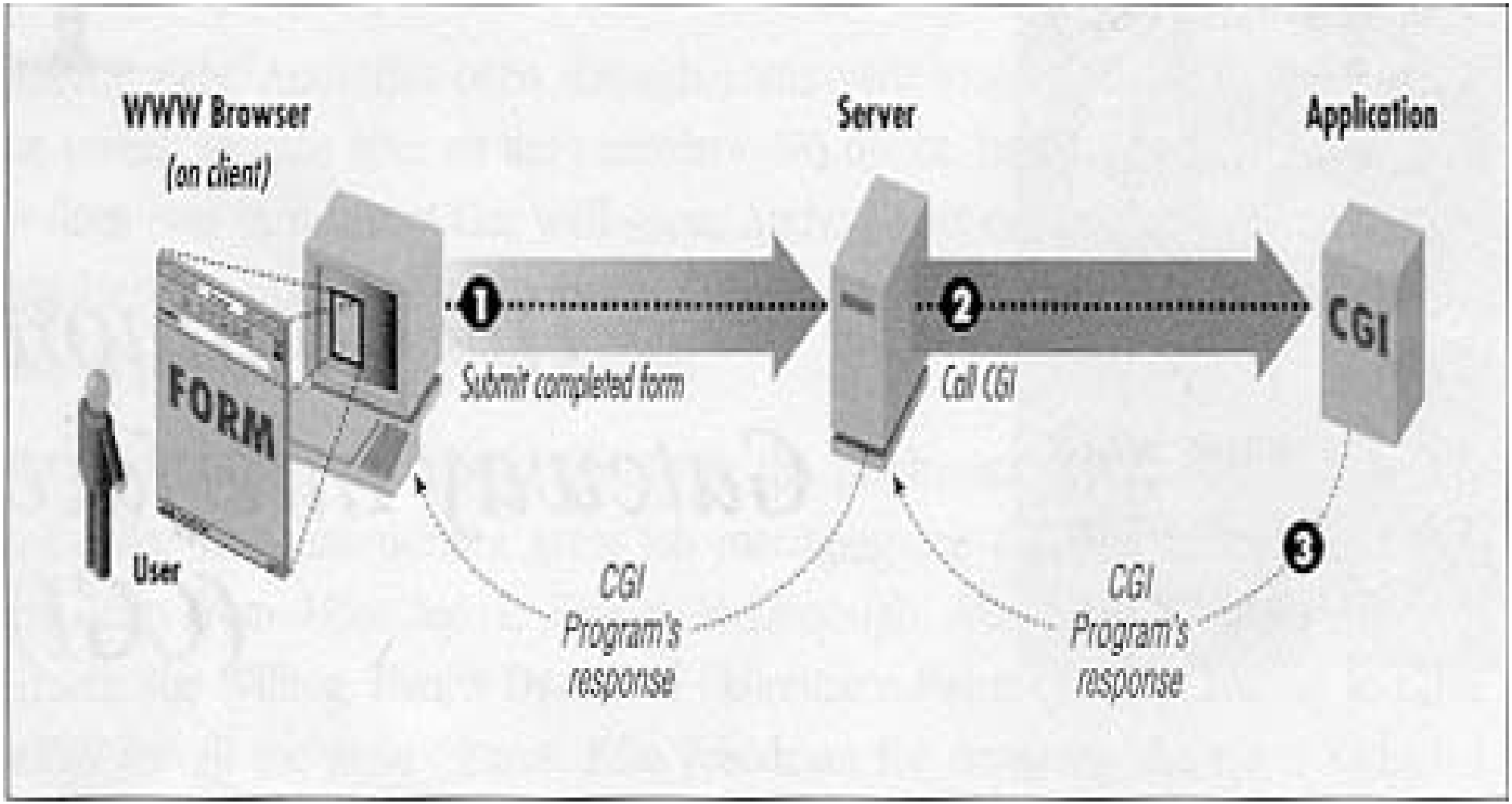
- Obtaining input from a user or from a data file.
- Storing that input in program variables.
- Manipulating those variables to achieve some desired purpose, and
- Sending the results to a file or video display.

- Data are obtained in ENVIRONMENT variables.
- The ENVIRONMENT variables are shown below in the table

ENVIRONMENTVARIABLE	DESCRIPTION
SERVER_NAME	The server's Host name or IP address .
SERVER_SOFTWARE	The name and version of the server-software that is answering the client requests.
SERVER_PROTOCOL	The name and revision of the information protocol the request came in with.
REQUEST_METHOD	The method with which the information request was issued.
QUERY_STRING	The query information passed to the program. It is appended to the URL with a "?".

ENVIRONMENTVARIABLE	DESCRIPTION
CONTENT_TYPE	The MIME type of the query data, such as "text/html".
CONTENT_LENGTH	The length of the data in bytes, passed to the CGI program through standard input.
HTTP_REFERER	The URL of the document that the client points to before accessing the CGI program.
GATEWAY_INTERFACE	The revision of the CGI that the server uses.
HTTP_USER_AGENT	The browser the client is using to issue the request.

CGI Interaction through HTML Forms



Web Browser and Web Server interaction

- Suppose you embed the following hypertext link in an HTML document:

<A HREF="TEST.HTML">TEST.HTML</A>

If you were to click on this link, the browser would issue the following request to the Web server:

GET /TEST.HTML HTTP/1.0
Accept: text/plain
Accept: text/html

Two blank lines

Web Browser and Web Server interaction Cont...

- Each of these lines is referred to as a Header.
- No complete path of the file, so the Web Server would look TEST.HTML in server's Web-document root directory.
- Browser can accept plain text or HTML-formatted text files.

- Server Response

HTTP /1.0 200 OK
Date: Monday, 24-May-96
11:09:05 GMT
Server: NCSA/1.3
MIME-version 1.0
Content-type: text/html
Content-length: 231
<HTML>
<HEAD> <TITLE>Test Page</TITLE> </HEAD>
<H1>This is the sample document</H1>
This is a test HTML page.
</HTML>

- Web browser then reads and displays the HTML portion of the file.

GET Method

- All the form data is appended to the URL
- QUERY_STRING contains query information passed to the program
- When user clicks the submit button from a html form, browser generates a HTTP request

GET /Scrits/Workshop/simple2.pl?u11/11/99name=Rani&service=CAS&entrydate=26%2F11%2F1999 HTTP/1.0 and sends to the web browser.

GET Method Cont…

- The continuous string of text that follows the question mark represents the query string.
- In response to this request from the browser, the server executes the script simple2.pl and places the string

  uname=Rani&service=CAS&entrydate= 26%2F11%2F1999, in the QUERY_STRING environment variable and **HTTP/1.0** in SERVER_PROTOCOL
- CGI program reads these environment variables, process, and passes some results to Web Server

POST Method

- Data from the form is encoded as string of data divided in NAME/VALUE pair and separated by &.
- In case of POST methods with the same html form it will generate the request

POST Method Cont…

POST /Scripts/simple2.pl
HTTP/1.0
Accept: text/html
Accept: text/plain
User-Agent:
Content-type: application/ x-www-urlencoded
Content-length: 28
uname=Rani&service=CAS&entrydate=
26%2F11%2F1999

POST Method Cont…

- With the post method, the server passes the information contained in the submitted form as standard input (STDIN) to the CGI program.
- CONTENT_LENGTH contains information about how much amount of data being transferred from html form.

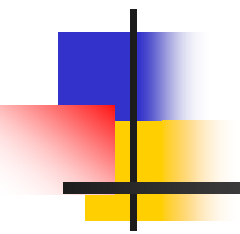
CGI program Using Perl (Practical Extraction and Report Language)

- Perl is an Interpreted script, instead of a compiled program with file extension .pl.
- Perl is freely available for many platform, Unix as well as Windows.
- A simplest CGI program using perl that prints **Hello, World** in the browser is

```
# This program prints Hello, World in the browser
    print "Content-type: text/html \n\n";
    print "Hello, World";
```

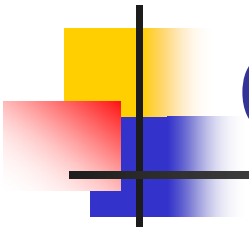
CGI program Using Perl (Practical Extraction and Report Language) Cont…

- Examples using GET and POST Method
- Using GET Method Source Code
- Using POST Method Source Code



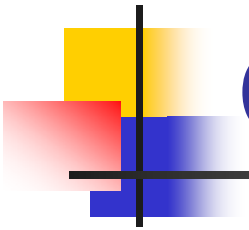
Introduction to HTML

Developing a Basic Web Page



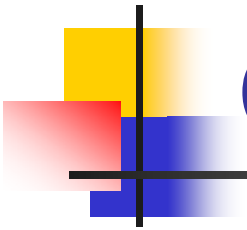
Objectives

- Review the history of the Web, the Internet, and HTML.
- Describe different HTML standards and specifications.
- Learn about the basic syntax of HTML code.



Objectives

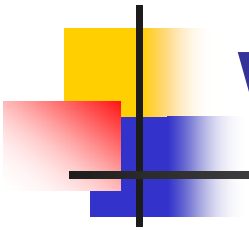
- Mark elements using two-sided and one-sided tags.
- Insert an element attribute.
- Create comments.
- Describe block-level elements and inline elements.
- Specify an element’s appearance with inline styles.



Objectives

- Create and format different types of lists.
- Create boldfaced and italicized text.
- Define empty elements.
- Insert an inline image into a Web page.
- Insert a horizontal line into a Web page.

Introducing the World Wide Web



- A **network** is a structure linking computers together for the purpose of sharing resources such as printers and files.
- Users typically access a network through a computer called a **host** or **node**.
- A computer that makes a service available to a network is called a **server**.

Introducing the World Wide Web

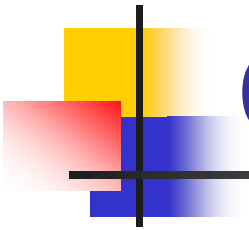
- A computer or other device that requests services from a server is called a **client**.
- One of the most common network structures is the **client-server network**.
- If the computers that make up a network are close together (within a single department or building), then the network is referred to as a **local area network (LAN)**.

Introducing the World Wide Web

- A network that covers a wide area, such as several buildings or cities, is called a **wide area network (WAN)**.
- The largest **WAN** in existence is the **Internet**.

What is the Internet?

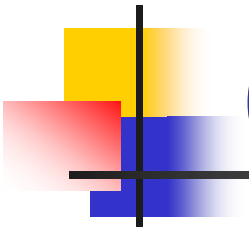
- Origins and History
- 1960's DOD ARPANET
  - In its early days, the Internet was called **ARPANET** and consisted of two network nodes located at UCLA and Stanford, connected by a phone line.
  - Experimental usage for communication
  - Keep govt. functioning in case of nuclear war
- Grew to include scientists and researchers from: military, universities
- 1980's - NSF became the "backbone"



Components

- Hosts
- Any computer with a direct connection to the Internet can communicate, share data and run applications
- Domain
  - **identifies the organization**
  - **identifies type or location**
  - **helps to route data efficiently**

Domain Name Examples (name.root)



- www.yahoo.com
- www.mozilla.org
- www.whitehouse.gov
- www.google.com

Internet Protocol Number (IP)

- numerical address
- 4-part number—similar to area code and phone number
- assist with routing
- locating host
- 198.64.7.9

Internet Providers

- Provide access to the Internet
- AOL
- SBC
- EarthLink
- MSN

Internet Services

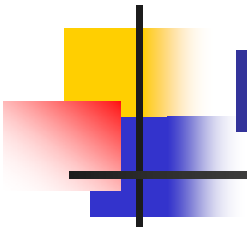
- Electronic Mail
  - single most useful tool of the Internet
  - Eudora–freeware/shareware
  - Outlook
- Netscape
- Yahoo
- Hotmail

Internet Services

- Listserv
  - done through Email
  - similar to a newsgroup
  - many subjects available
  - subscribe and receive all "posts" to the list
  - post your own responses

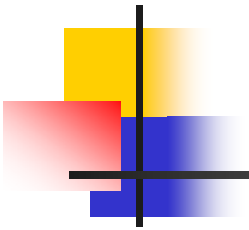
Internet Services

- Newsgroups
  - Usenet Newsgroups
  - alt.??????
  - thousands of newsgroups
  - provide a place where users can post comments, stories and/or questions



Internet Services

- FTP Servers
  - File Transport Protocol
  - servers configured to allow FTP connections
  - used to download and upload files
  - software
  - information/documents/web



Internet Services

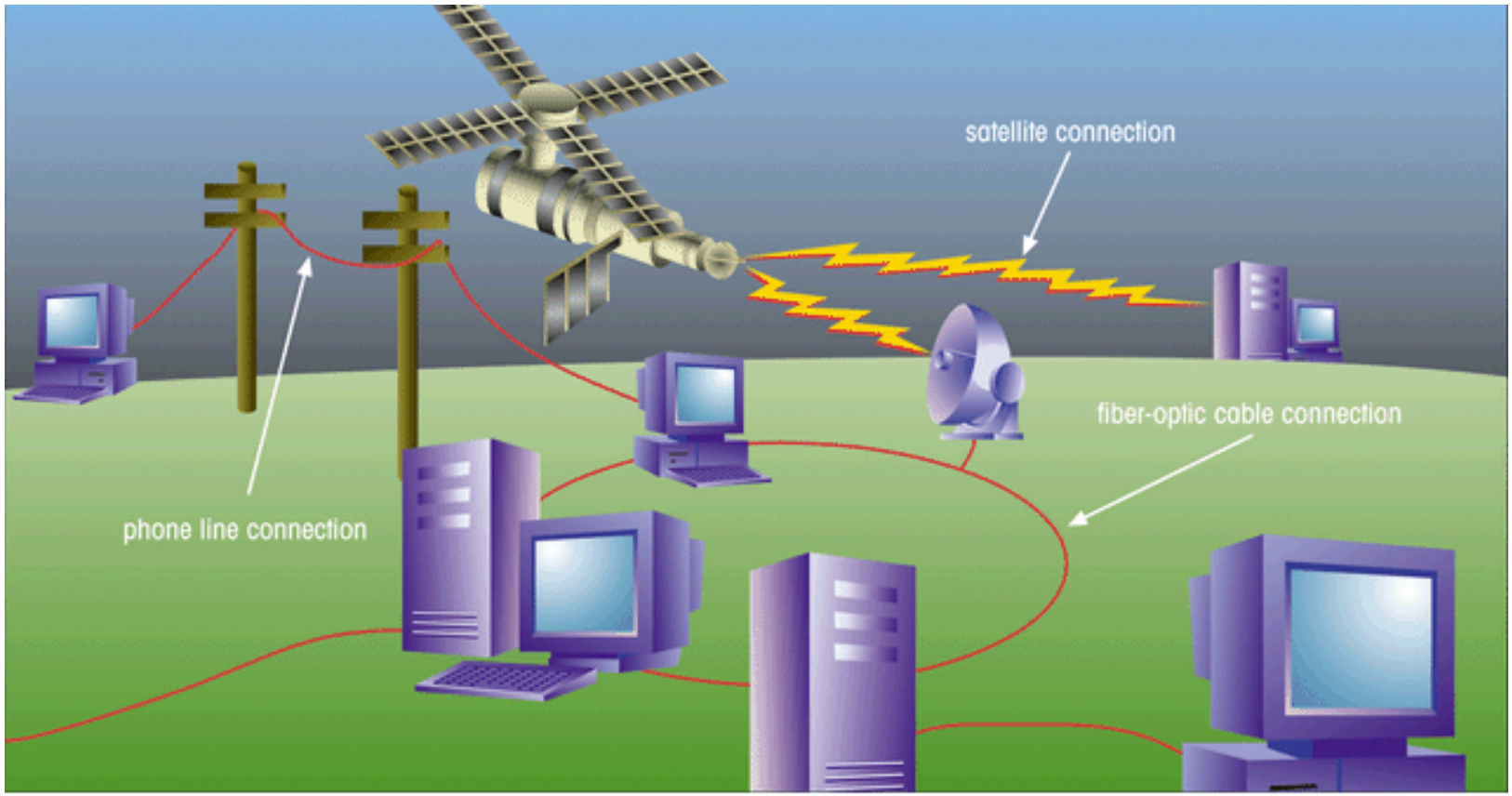
- Telnet
  - establishes a link between two hosts or between a client and host
  - similar to using a modem to :
  - access BBS (bulletin board system)
  - log into home computer and read mail
  - log into Internet server for file manipulation.

Introducing the World Wide Web

- Today the Internet has grown to include hundreds of millions of interconnected computers, cell phones, PDAs, televisions, and networks.

- The physical structure of the Internet uses fiber-optic cables, satellites, phone lines, and other telecommunications media.

Structure of the Internet

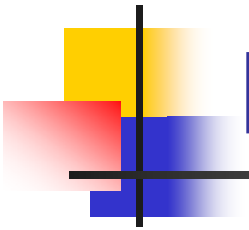


The Development of the World Wide Web

- Timothy Berners-Lee and other researchers at the CERN nuclear research facility near Geneva, Switzerland laid the foundations for the **World Wide Web**, or the **Web**, in 1989.

- They developed a system of interconnected **hypertext** documents that allowed their users to easily navigate from one topic to another.

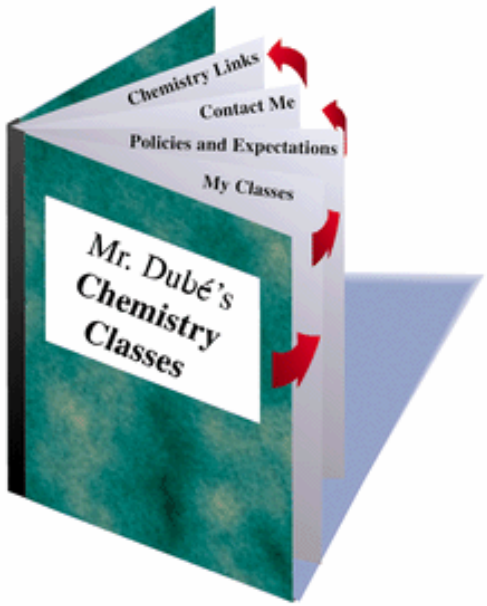
- **Hypertext** is a method of organizing information that gives the reader control over the order in which the information is presented.



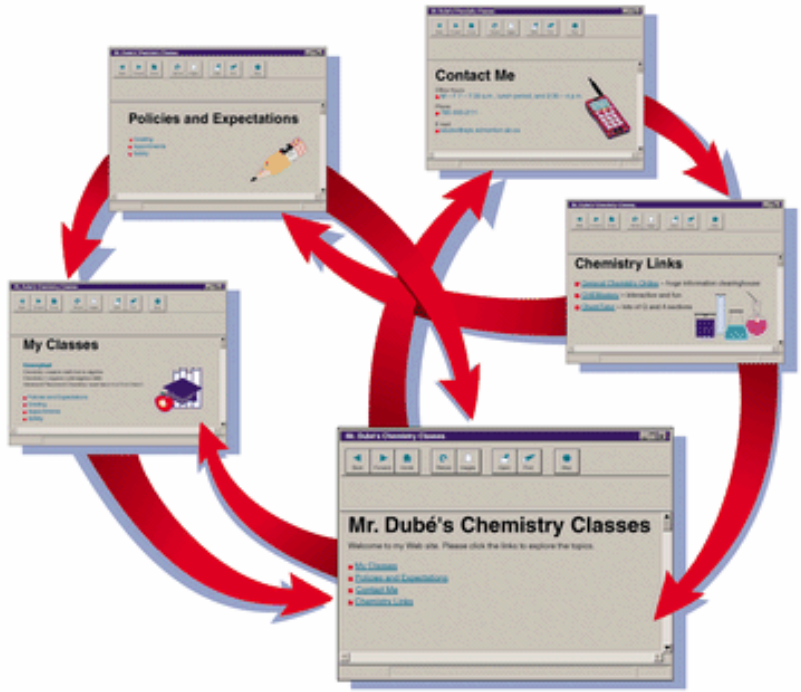
Hypertext Documents

- When you read a book, you follow a linear progression, reading one page after another.
- With hypertext, you progress through pages in whatever way is best suited to you and your objectives.
- Hypertext lets you skip from one topic to another.

Linear versus hypertext documents



Reading a linear document



Reading a hypertext document

Hypertext Documents

- The key to **hypertext** is the use of **hyperlinks (or links)** which are the elements in a hypertext document that allow you to jump from one topic to another.
- A **link** may point to another section of the same document, or to another document entirely.
- A **link** can open a document on your computer, or through the Internet, a document on a computer anywhere in the world.

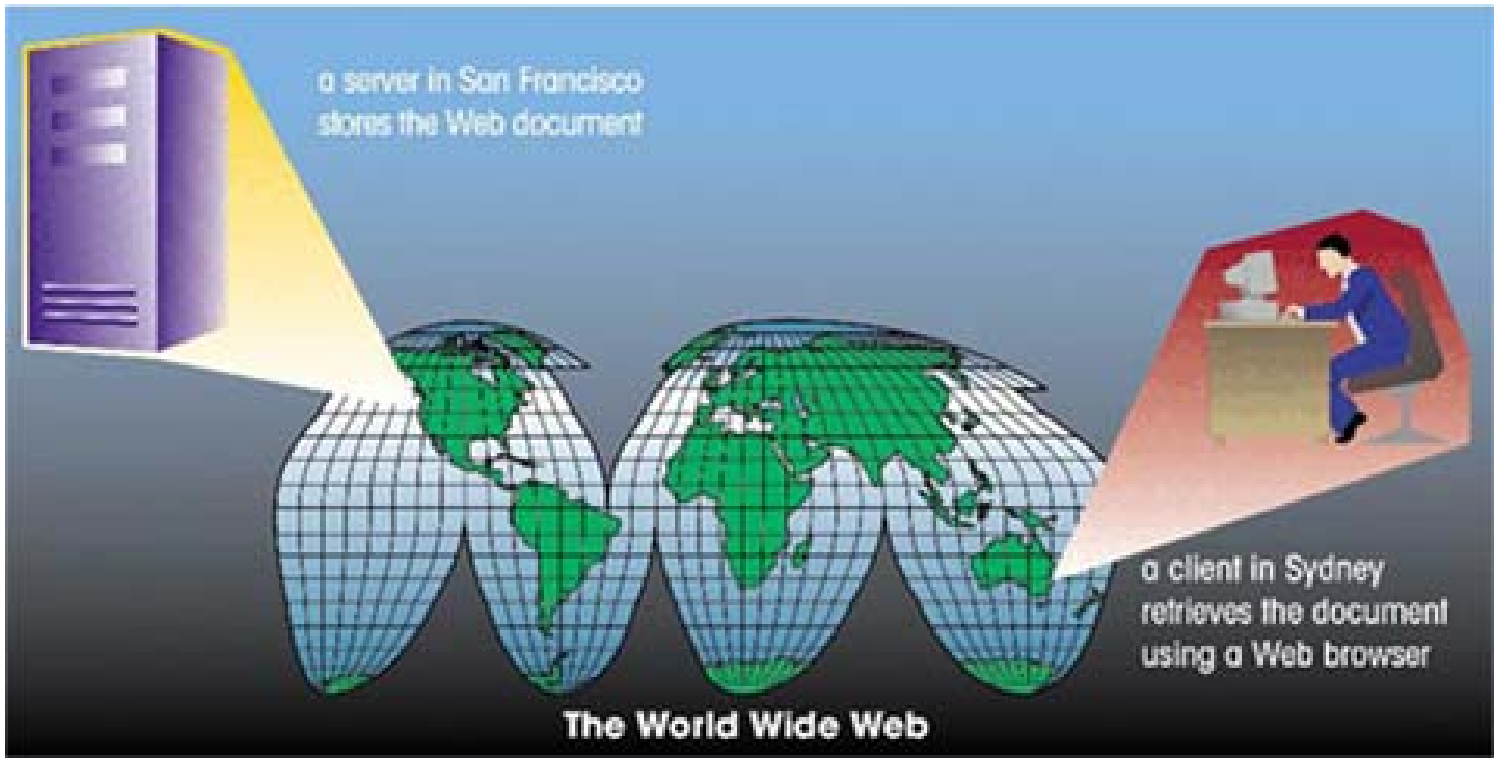
Hypertext Documents

- An entire collection of linked documents is referred to as a **Web site**.
- The hypertext documents within a Web site are known as **Web pages**.
- Individual pages can contain text, audio, video, and even programs that can be run remotely.

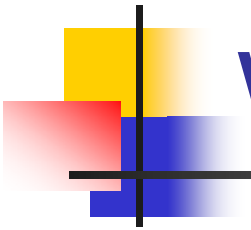
Web Servers and Web Browsers

- A **Web page** is stored on a **Web server**, which in turn makes it available to the network.
- To view a **Web page**, a client runs a software program called a **Web browser**, which retrieves the page from the server and displays it.
- The earliest browsers, known as **text-based browsers**, were incapable of displaying images.
- Today most computers support **graphical browsers** which are capable of displaying not only images, but also video, sound, animations, and a variety of graphical features.

Using a browser to view a Web document from a Web server



HTML: The Language of the Web



- A Web page is a text file written in a language called **Hypertext Markup Language**.
- A **markup language** is a language that describes a document's structure and content.
- HTML is not a programming language or a formatting language.
- **Styles** are format descriptions written in a separate language from HTML that tell browsers how to render each element. Styles are used to format your document.

The History of HTML

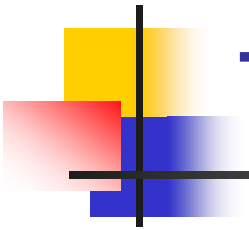
- The first version of HTML was created using the **Standard Generalized Markup Language (SGML)**.
- In the early years of HTML, Web developers were free to define and modify HTML in whatever ways they thought best.
- Competing browsers introduced some differences in the language. The changes were called **extensions**.

The History of HTML

- A group of Web developers, programmers, and authors called the **World Wide Web Consortium**, or the **WC3**, created a set of standards or specifications that all browser manufacturers were to follow.
- The **WC3** has no enforcement power.
- The recommendations of the **WC3** are usually followed since a uniform approach to Web page creation is beneficial to everyone.

Versions of HTML and XHTML

Version	Date	Description
HTML 1.0	1989–1994	The first public version of HTML which included browser support for inline images and text controls.
HTML 2.0	1995	The first version supported by all graphical browsers. It introduced interactive form elements such as option buttons and text boxes. A document written to the HTML 2.0 specification is compatible with almost all browsers on the World Wide Web.
HTML 3.0	1996	A proposed replacement for HTML 2.0 that was never widely adopted.
HTML 3.2	1997	This version included additional support for creating and formatting tables and expanded the options for interactive form elements. It also supported limited programming using scripts.
HTML 4.01	1999	This version added support for style sheets to give Web designers greater control over page layout. It added new features to tables and forms and provided support for international features. This version also expanded HTML's scripting capability and added increased support for multimedia elements.
XHTML 1.0	2001	This version is a reformulation of HTML 4.01 in XML and combines the strength of HTML 4.0 with the power of XML. XHTML brings the rigor of XML to Web pages and provides standards for more robust Web content on a wide range of browser platforms.
XHTML 1.1	2002	A minor update to XHTML 1.0 that allows for modularity and simplifies writing extensions to the language.
XHTML 2.0	2004–	The latest version, designed to remove most of the presentational features left in HTML.

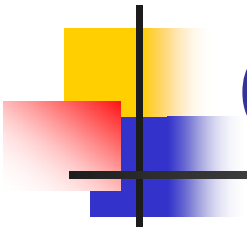


The History of HTML

- Older features of HTML are often **deprecated**, or phased out, by the W3C. That does not mean you can't continue to use them—you may need to use them if you are supporting older browsers.
- Future Web development is focusing increasingly on two other languages: **XML** and **XHTML**.
- **XML (Extensible Markup Language)** is a metalanguage like SGML, but without SGML's complexity and overhead.

The History of HTML

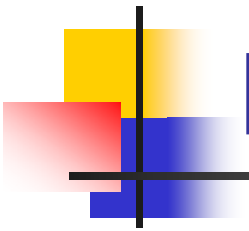
- **XHTML (Extensible Hypertext Markup Language)** is a stricter version of HTML and is designed to confront some of the problems associated with the different and competing versions of HTML.
- **XHTML** is also designed to better integrate **HTML** with **XML**.
- **HTML** will not become obsolete anytime soon.



Guidelines

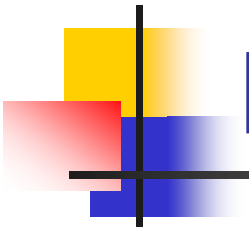
- Become well-versed in the history of HTML.
- Know your market.
- Test.

Tools for Creating HTML Documents

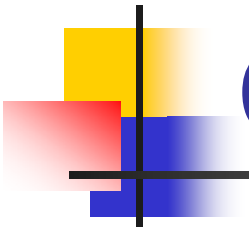


- Basic text editor like Notepad or BBEdit.
- **HTML Converter** - converts formatted text into HTML code.
  - Can create the source document in a word processor and then convert it.
  - HTML code created using a converter is often longer and more complicated than it needs to be, resulting in larger-than-necessary files.

Tools for Creating HTML Documents



- **HTML Editor** – helps you create an HTML file by inserting HTML codes for you as you work.
  - They can save you a lot of time and help you work more efficiently.
  - Advantages and limitations similar to those of HTML converters.
  - Allow you to set up a Web page quickly.
  - Will usually still have to work with HTML code to create a finished document.



Creating an HTML Document

- It is a good idea to plan out a Web page before you start coding.
- Draw a planning sketch or create a sample document using a word processor.
- Preparatory work can weed out errors or point to potential problems.

Creating an HTML Document

- In planning, identify a document's different elements. An **element** is a distinct object in the document, like a paragraph, a heading, or a page's title.

- Formatting features such as **boldfaced** font, and *italicized* text may be used.

Marking Elements with Tags

- The core building block of HTML is the **tag**, which marks each element in a document.
- Tags can be two-sided or one-sided.
- A **two-sided tag** is a tag that contains some document content. General syntax for a two-sided tag:

  <element>content</element>

Marking Elements with Tags

- A two-sided tag's opening tag ( ) and closing tag ( ) should completely enclose its content.
- HTML allows you to enter element names in either uppercase or lowercase letters.
- A one-sided tag contains no content. General syntax for a one-sided tag:

<element />

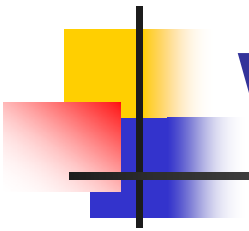
Marking Elements with Tags

- Elements that employ one-sided tags are called empty elements since they contain no content. An example is a line break
.

- A third type of tag is the comment tag, which you can use to add notes to your HTML code.

<!-- comment -->

- Comments are useful in documenting your HTML code for yourself and others.



White Space and HTML

- HTML file documents are composed of text characters and **white space**.
- **White space** is the blank space, tabs, and line breaks within the file.
- HTML treats each occurrence of **white space** as a single blank space.
- You can use **white space** to make your document more readable.

Element Attributes

- Many tags contain attributes that control the behavior, and in some cases the appearance, of elements in the page.
- Attributes are inserted within the tag brackets.

<element attribute1="value1" attribute2="value2" .../>
for one-side tags

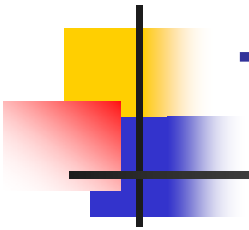
<element attribute1="value1" attribute2="value2" ...>content</element>

for two-sided tags

The Structure of an HTML File

- The opening **<html>** tag marks the start of an HTML document, and the closing **</html>** tag tells a browser when it has reached the end of that HTML document.

- Anything between these two tags makes up the content of the document, including all other elements, text, and comments.



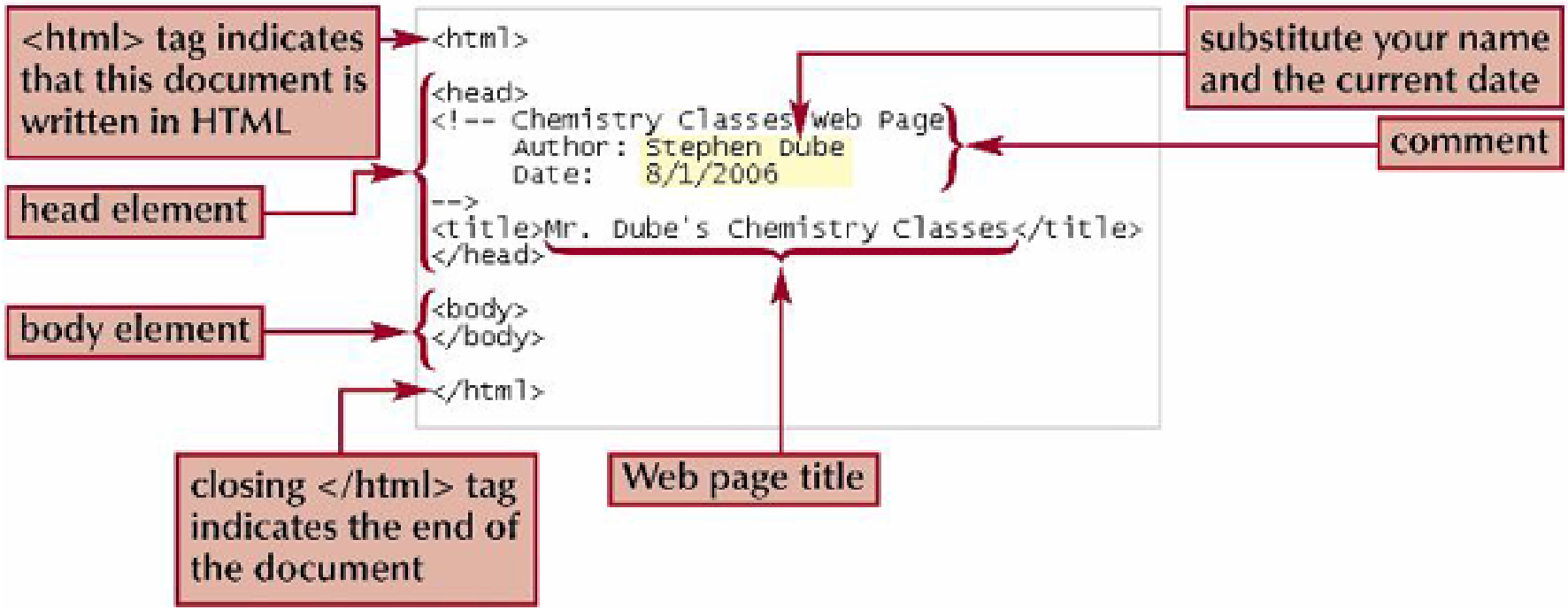
The Structure of an HTML File

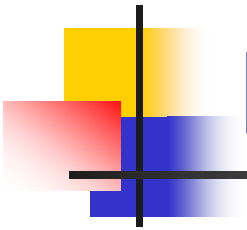
- An HTML document is divided into two parts: the **head** and the **body**.
- The **head** element contains information about the document, for example the document title or the keywords.
- The content of the **head** element is not displayed within the Web page.

The Structure of an HTML File

- The **body element** contains all of the content to be displayed in the Web page.
- The **body element** can contain code that tells the browser how to render the content.
- The **title element** contains the page's title. A document's title is usually displayed in the title bar.

Initial HTML code in chem.htm

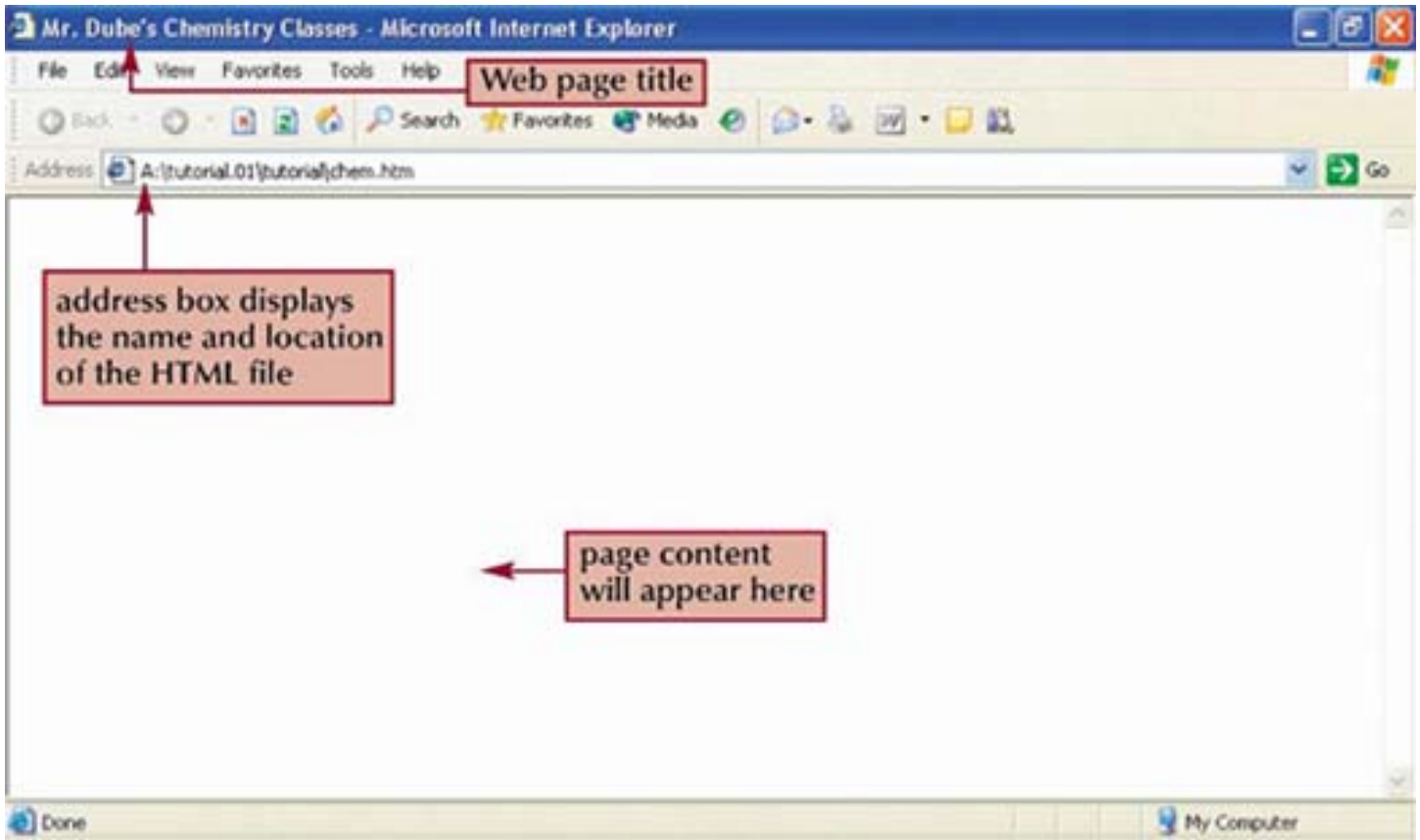




Displaying an HTML File

- As you work on a Web page, you should occasionally view it with your Web browser to verify that the file contains no syntax errors or other problems.
- You may want to view the results using different browsers to check for compatibility.

Initial Web page viewed in Internet Explorer

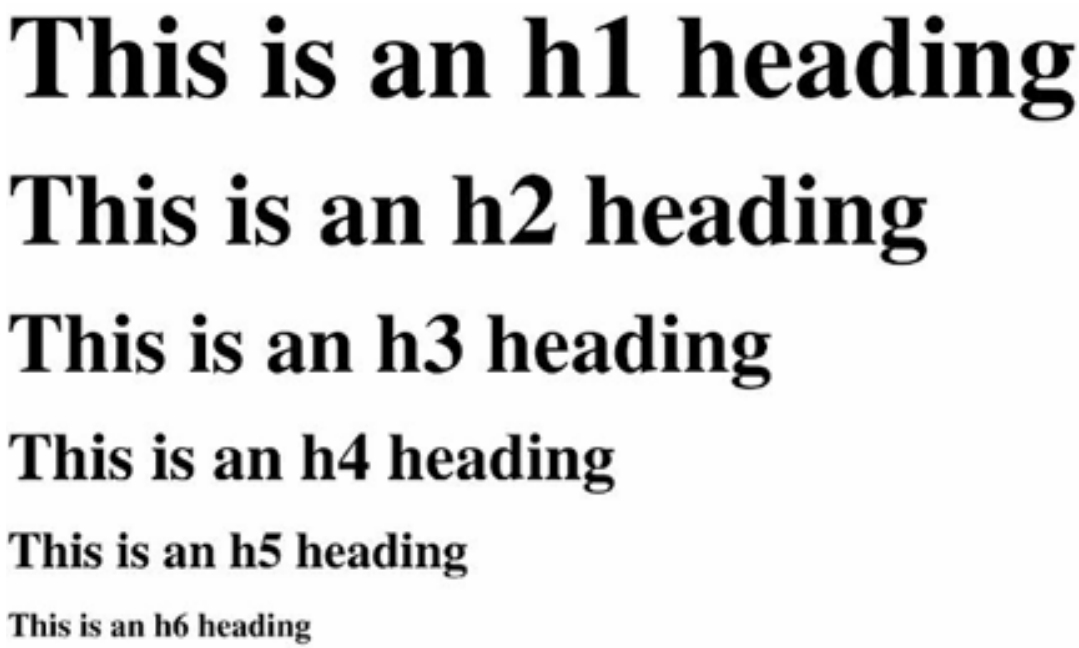


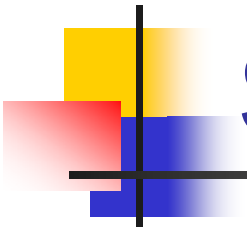
Working with Block-Level Elements

- In a Web page, most content is marked as either a **block-level** element or an inline element.
- A **block-level** element contains content displayed in a separate section within the page, setting it off from other blocks.
- An **inline element** is part of the same block as its surrounding content—for example individual words or phrases within a paragraph.

Creating Headings

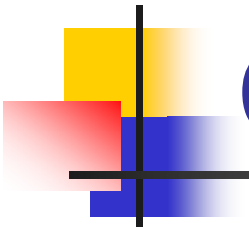
- HTML supports six heading elements.





Styles

- Use the **style attribute** to control the appearance of an element, such as text alignment.
- Styles specified as attributes in a tag are also referred to as **inline styles**.
- The **text-align style** tells the browser how to horizontally align the contents of an element.
- **Presentational attributes** specify exactly how the browser should render an element.



Creating Lists

- HTML supports three kinds of lists: **ordered, unordered, and definition**.
- You use an **ordered list** for items that must appear in a particular sequential order.
- You use an **unordered list** for items that do not need to occur in any special order.
- One **list** can contain another list. This is called a nested list.

Applying a Style to a List

List-Style-Type	Marker (s)
disc	•
circle	○
square	■
decimal	1, 2, 3, 4, ...
decimal-leading-zero	01, 02, 03, 04, ...
lower-roman	i, ii, iii, iv, ...
upper-roman	I, II, III, IV, ...
lower-alpha	a, b, c, d, ...
upper-alpha	A, B, C, D, ...
none	*no marker displayed*

Working with Inline Elements

- Character formatting elements are one of HTML's set of inline elements. This element allows you to format text characters.

Welcome to our **Chemistry Classes**.

Understanding Logical and Physical Elements

- A **logical element** describes the nature of the enclosed content, but not necessarily how that content should appear.
- A **physical element** describes how content should appear, but doesn't indicate the content's nature.
- You should use a **logical element** that accurately describes the enclosed content whenever possible, and use **physical elements** only for general content.

Working with Empty Elements

- To display a graphic, you insert an **inline image** into the page. An **inline image** displays a graphic image located in a separate file within the contents of a block-level element.

- You can insert a horizontal line by using the one-sided tag **<hr />**.

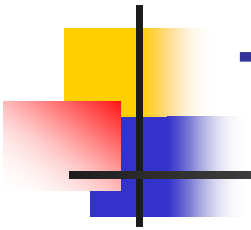
- A **pixel** is a dot on your computer screen that measures about 1/72” square.

Working with Empty Elements

- Other **empty elements** you may wish to use in your Web page include **line breaks** and **meta elements**.
- **Meta elements** are placed in the document's head and contain information about the document that may be of use to programs that run on Web servers.

Summary: Tips for Good HTML Code

- Use line breaks and indented text to make your HTML file easier to read.
- Insert comments into your HTML file to document your work.
- Enter all tag and attribute names in lowercase.
- Place all attribute values in quotes.
- Close all two-sided tags.



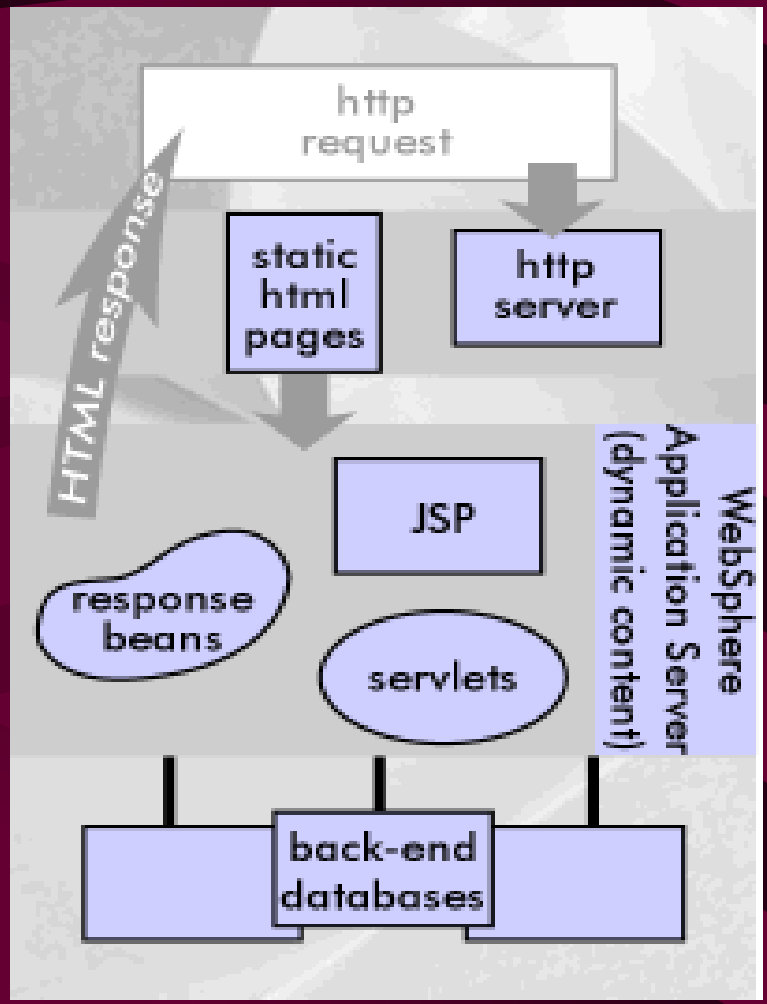
Summary: Tips for Good HTML Code

- Make sure that nested elements do not cross.
- Use styles in place of presentational elements whenever possible.
- Use logical elements to describe an element's content.
- Use physical elements to describe the element's appearance.

Summary:
Tips for Good HTML Code

- Include the alt attribute for any inline image to specify alternative text for non-graphical browsers.
- Know your market and the types of browsers that your audience will use to view your Web page.
- Test your Web page on all relevant browsers.

Introduction to Java Server Pages technology

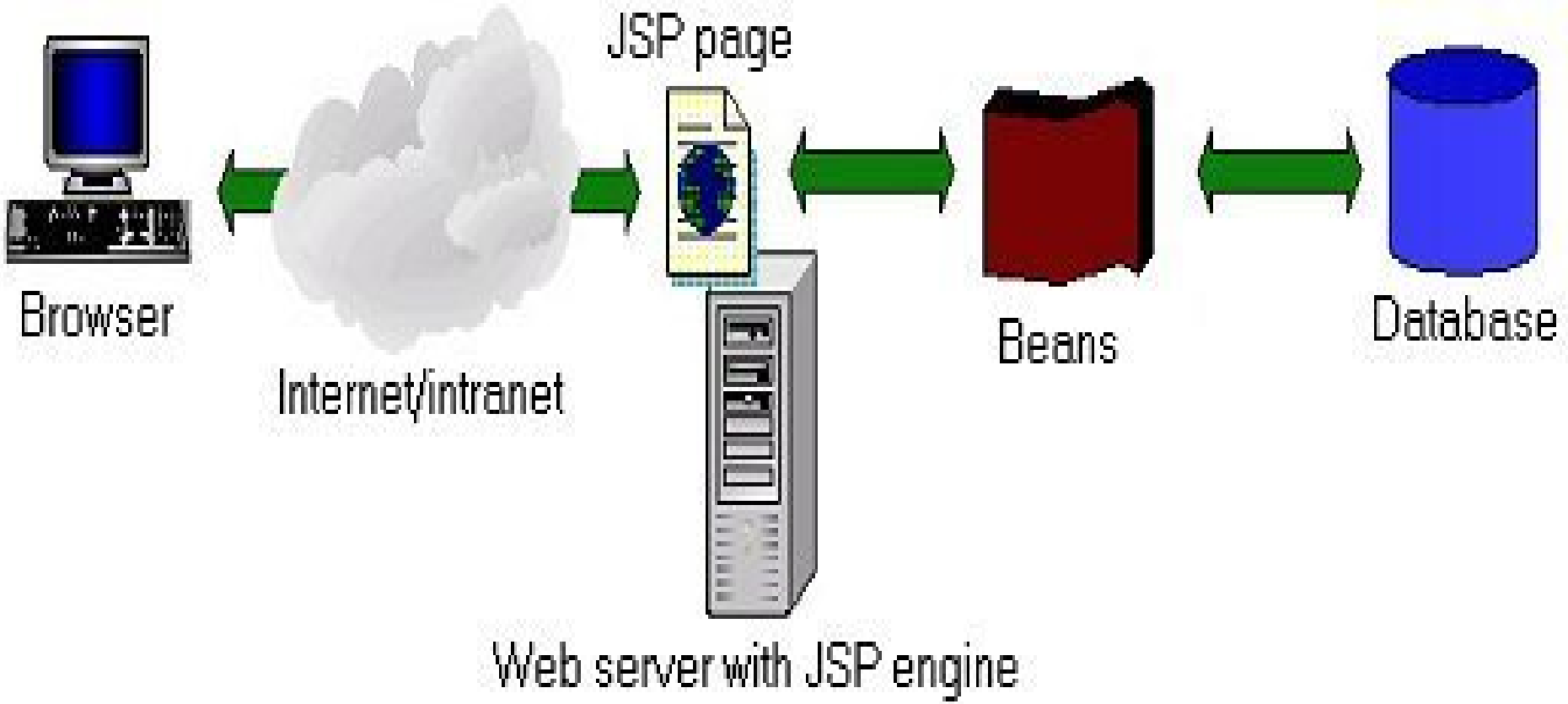


–http://www.techinterviews.com/?p=92

What is JSP?

- Java based technology that simplifies the developing of dynamic web sites
- **JSP** pages are HTML pages with embedded code that allows to access data from Java code running on the server
- **JSP** provides separation of HTML presentation logic from the application logic.

JSP Flow


JSP page
Browser
Internet/intranet
Beans
Database
Web server with JSP engine

JSP Technology

- JSP technology provides a way to combine the worlds of HTML and Java servlet programming.
- JSP specs are built on the Java Servlet API.
- JSP supports two different styles for adding dynamic content to web pages:
  - JSP pages can embed actual programming code (typically Java)
  - JSP supports a set of HTML-like tags that interact with Java objects on the server (without the need for raw Java code to appear in the page).

JSP Example: Hello World

- 1.

```
<html>
<head>
    <title>Hello World example</title>
</head>
<body>
Hello, World!
</body>
</html>
```

- 2.

```
<html>
<head>
    <title>Simple JSP Example</title>
</head>

<body>
<FORM METHOD="GET" ACTION="SimpleJSP.jsp">
<p> What is your name?</p>
<INPUT TYPE="TEXT" SIZE="20" NAME="name">
<INPUT TYPE="SUBMIT">
</FORM>

</body>
</html>
```

What is your name?

Submit Query

SimpleJSP.jsp

```
<html>
<head>
    <title>Simple JSP Example - version 1</title>
</head>

<body>
<P>
    <% String visitor = request.getParameter("name");
       if (visitor == null) visitor = "World";   %>
       Hello, <%=visitor%>!<BR>
</P>
</body>
</html>
```

SimpleJSP.jsp - the Bean edition

- JSP includes tags for interacting with JavaBeans.
- JavaBean is a simply Java class that follow JavaBeans specs: rules for defining a Bean's ctor & methods for accessing and setting their properties.

```
package examples.HelloBean;

public class HelloBean implements java.io.Serializable
{
    String name;

    public HelloBean ()
    {
        this.name = "World";
    }

    public String getName ()
    {
        return name;
    }

    public void setName (String name)
    {
        this.name = name;
    }

}
```

SimpleJSP.jsp - the Bean edition

```
<html>
<head>
    <title>Simple JSP Example - version 2</title>
</head>

<body>
<jsp:useBean id="hello" class="examples.HelloBean"/>
<jsp:setProperty name="hello" property="name" param="name"/>
<P>
Hello, <jsp:getProperty name="hello" propety="name"/>!<BR>
</P>
</body>
</html>
```

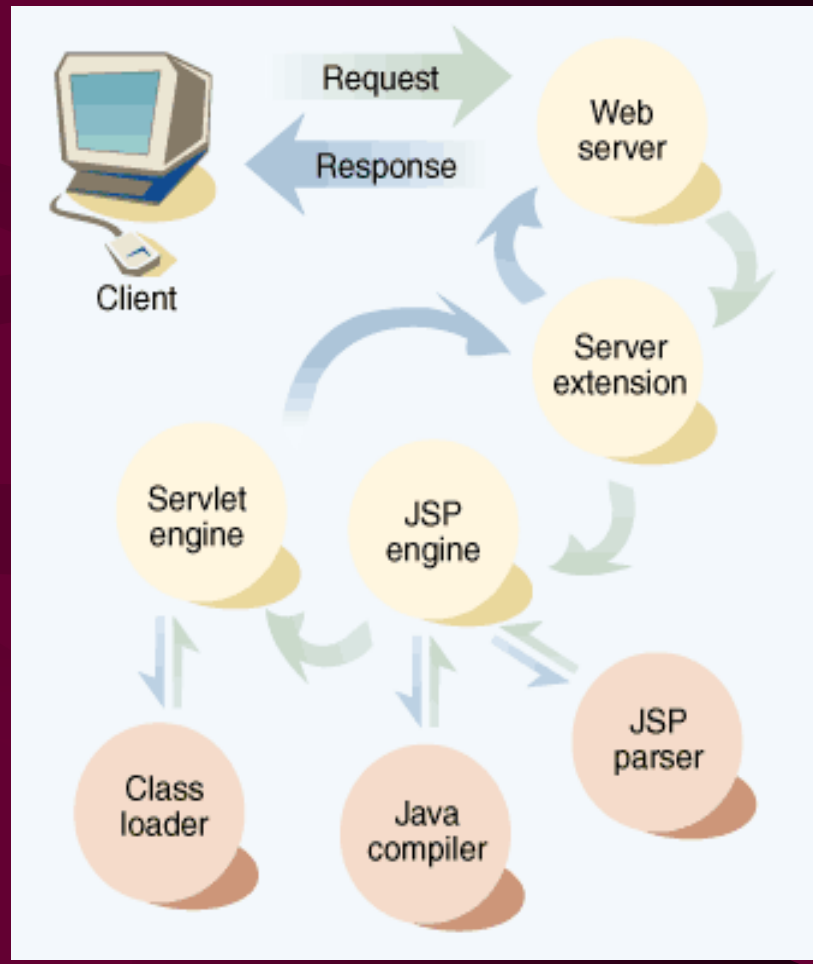
JSP Example: Hello World

- In both cases the http request is:
  http://localhost:8080/SimpleJSP.jsp?name=Naomi
- The response from JSP container would be:

```
<html>
<head>
    <title>Simple JSP Example</title>
</head>
<body>
<P>
Hello, Naomi!<BR>
</P>
</body>
</html>
```

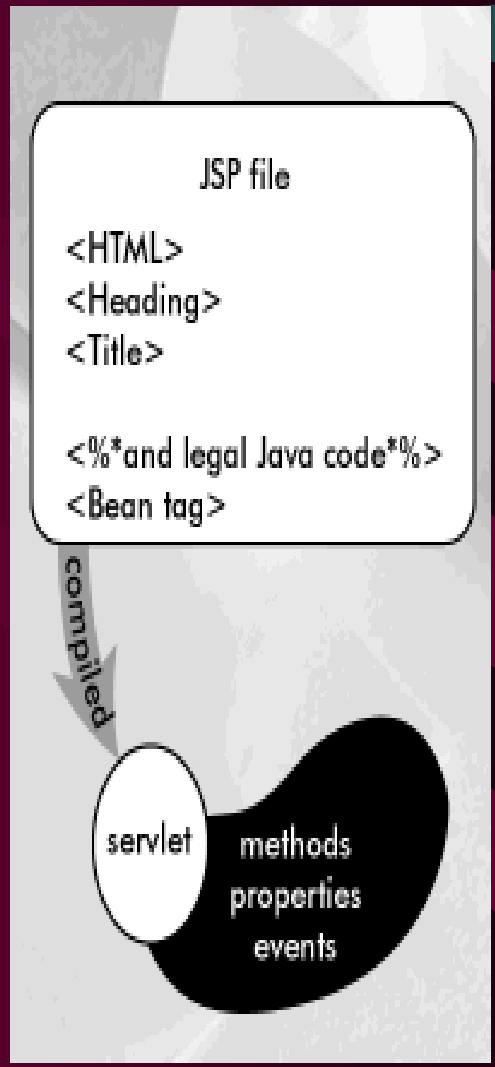
How it is work?

- Client request for a page ending with ".jsp“.
- Web Server fires up the JSP engine.
- The JSP engine checks to see if the JSP file is new or changed.
- The JSP engine takes the page and converts it into a Java servlet (by JSP parser)
- The JSP engine compiles the servlet (by standard Java compiler).
- Servlet Engine executes the new Java servlet using the standard API.
- Servlet’s output is transferred by Web Server as a http response.



JSP Pages content

- standard HTML tags & scripts (JavaScript/VBscript)
- new tags for scripting in the Java language.
  - Expressions:
    <%=expression %> or XML variant:
    <jsp: expression>expression </jsp:expression>
    For Example:
    <%= fact(12) %>
    <%= (hours <12) ? “AM” : “PM” %>
    <%= Math.pow(radius, 2) %>
  - Scriptlets:
    <% scriptlet %> or XML variant:
    <jsp:scriptlet> scriptlet </jsp:scriptlet>
  - Declarations:
    <%! declaration (s) %> or XML variant:
    <jsp:declaration> declaration(s) </jsp: declaration>
    For Example



JSP Pages content – cont.

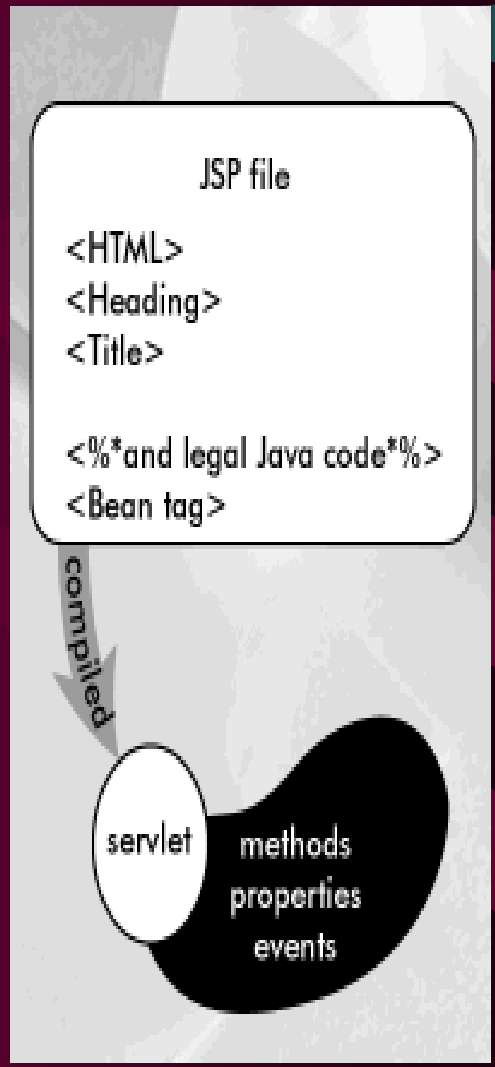
- JSP directives – is a set of tags for providing the JSP container with page specific instructions for how the document should be processed. Directives affect global properties of the JSP page.

  <%@ page attr1="val1" attr2=… %>
- Comments –for adding documentation
  - Comments that will be in the output:

    <!-- comment -->
  - JSP comments

    <%-- comment --%>
  - Scripting language comments:

    <% /* comment */ %>

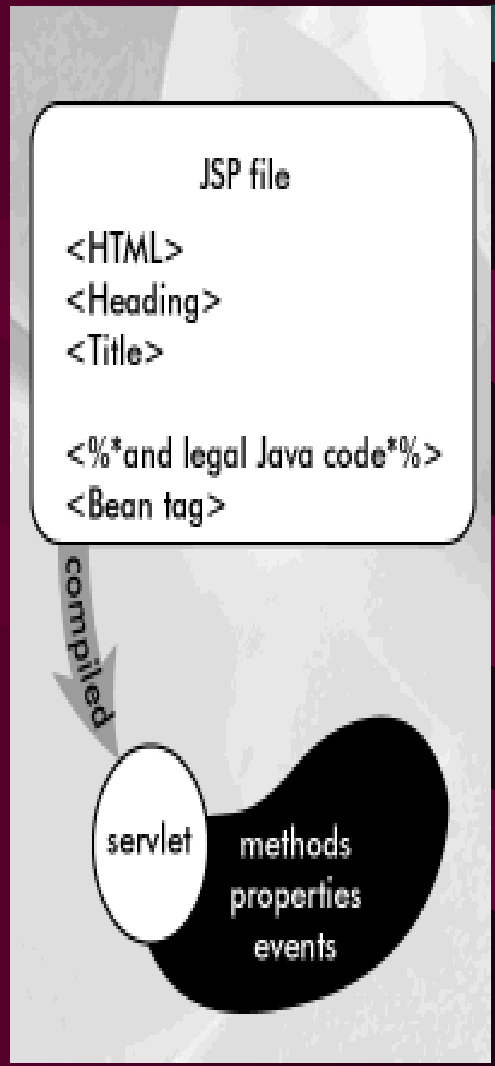


JSP Pages content – cont.

- Actions and implicit objects

  JSP implicit objects:

page	out
config	session
request	application
response	pageContext
exception	



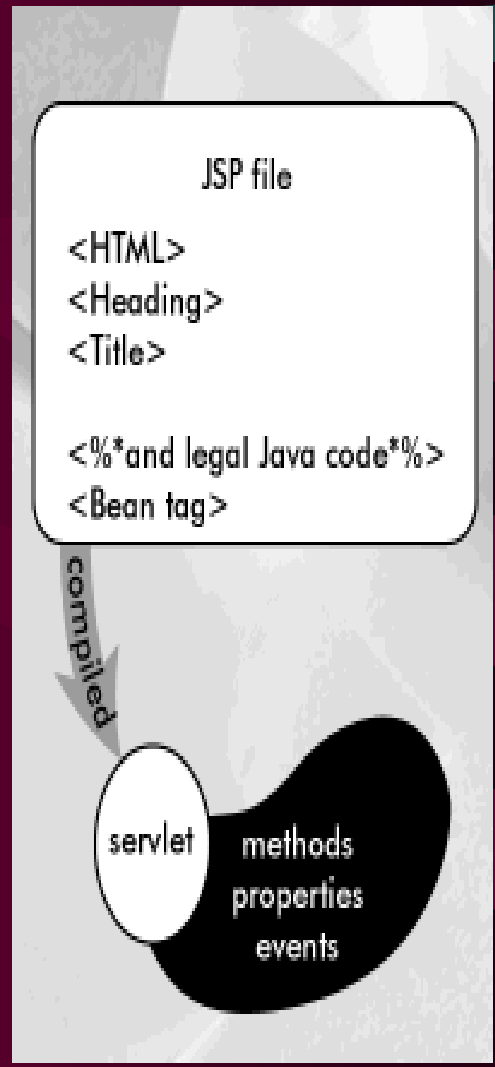
JSP Pages content – cont.

- Bean's tags

  allows JSP pages to call reusable components called JavaBeans components.
- The tag <jsp:useBean> syntax is:

  <jsp:useBean id="*Bean_name*" scope="*scope_value*" class="*class_name*" beanName="*ser_filename*" type="*class_or_interface_name*" > *properties tags* </jsp:useBean>
- <jsp:setProperty> tag syntax is:

  <jsp:setProperty name="*property_name*" property="*property_value*" />



JSP benefits

- Java-based technology
- Vendor-neutral
- Full access to underlying Java platform
- Performance
- Reusable components (JavaBeans)
- Separating presentation and implementation

Resources

- Sun JSP 1.1 Specs and description
- Server Side Java Resource Site
- IBM education courses
- JSP resource index
- JSP insider

The end

Scriptlet Example

```
<html>
<head>
    <title>Scriptlet Example</title>
<%! public long fact (long x) {
        if (x == 0) return 1;
        else return x * fact(x-1);
    } %>
</head>

<body>

<table>
<tr> <th width="50"> x</th><th width="50">x! </th></tr>
<br>
<% for (int x = 1; x < 10; x++) { %>
    <tr><td width="50"><%= x%> </td><td width="50"> <%= fact(x) %></td></tr>
<% } %>
</table>

</body>
</html>
```

SOAP 1.2, Introduction

University of Colorado at Boulder
Department of Computer Science
Software Engineering Research Laboratory
Nathan D. Ryan

What Is SOAP?

- SOAP 1.1

  Simple Object Access Protocol
- SOAP 1.2

  Huh? Us, work for Microsoft? No way!

  XML Protocol (XMLP or XP)

SOAP 1.2 Is…

- A “wrapper” protocol
- Written in XML
- Independent of the wrapped data
- Independent of the transport protocol
- Efficient (according to the W3C)
- A uni-directional message exchange paradigm

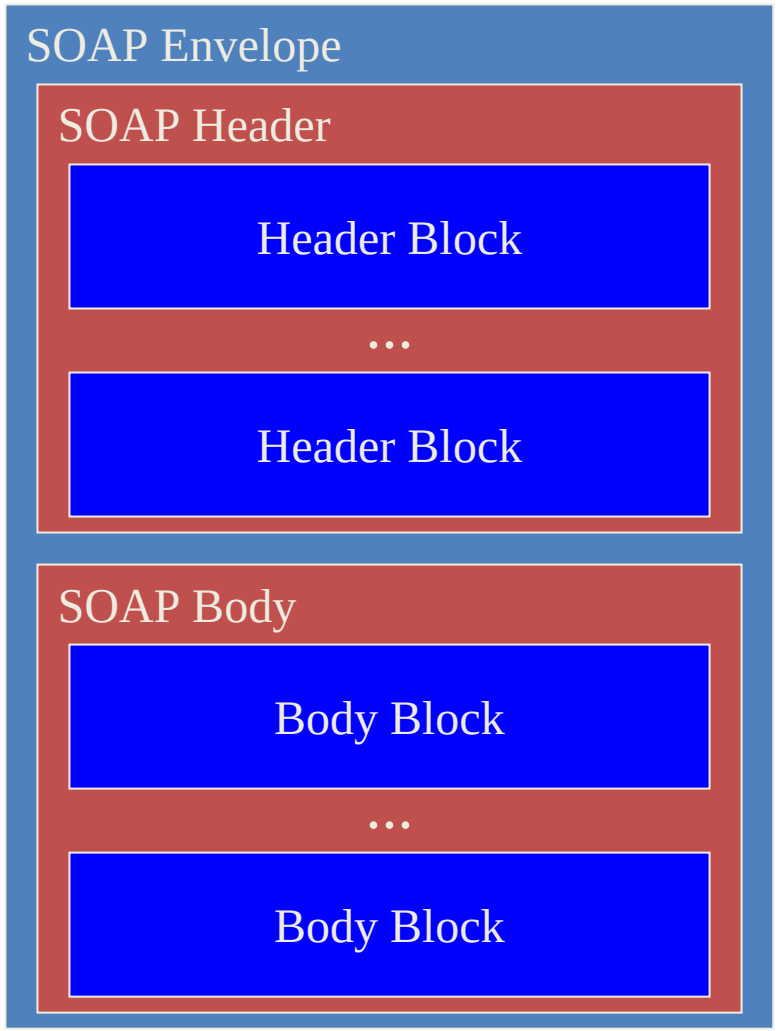
SOAP 1.2 Is *Not*…

- A transport protocol
- Written in *valid* XML
- Independent of the wrapped data
- Independent of the transport protocol
- Efficient (according to me)
- A uni-directional message exchange paradigm

Classification

- “Wrapper” protocol versus transport protocol
- Data is placed in header blocks and body blocks, as desired
- Transport is handled by another mechanism
  - HTTP 1.1 is the only binding specified, though others are possible

Message Anatomy



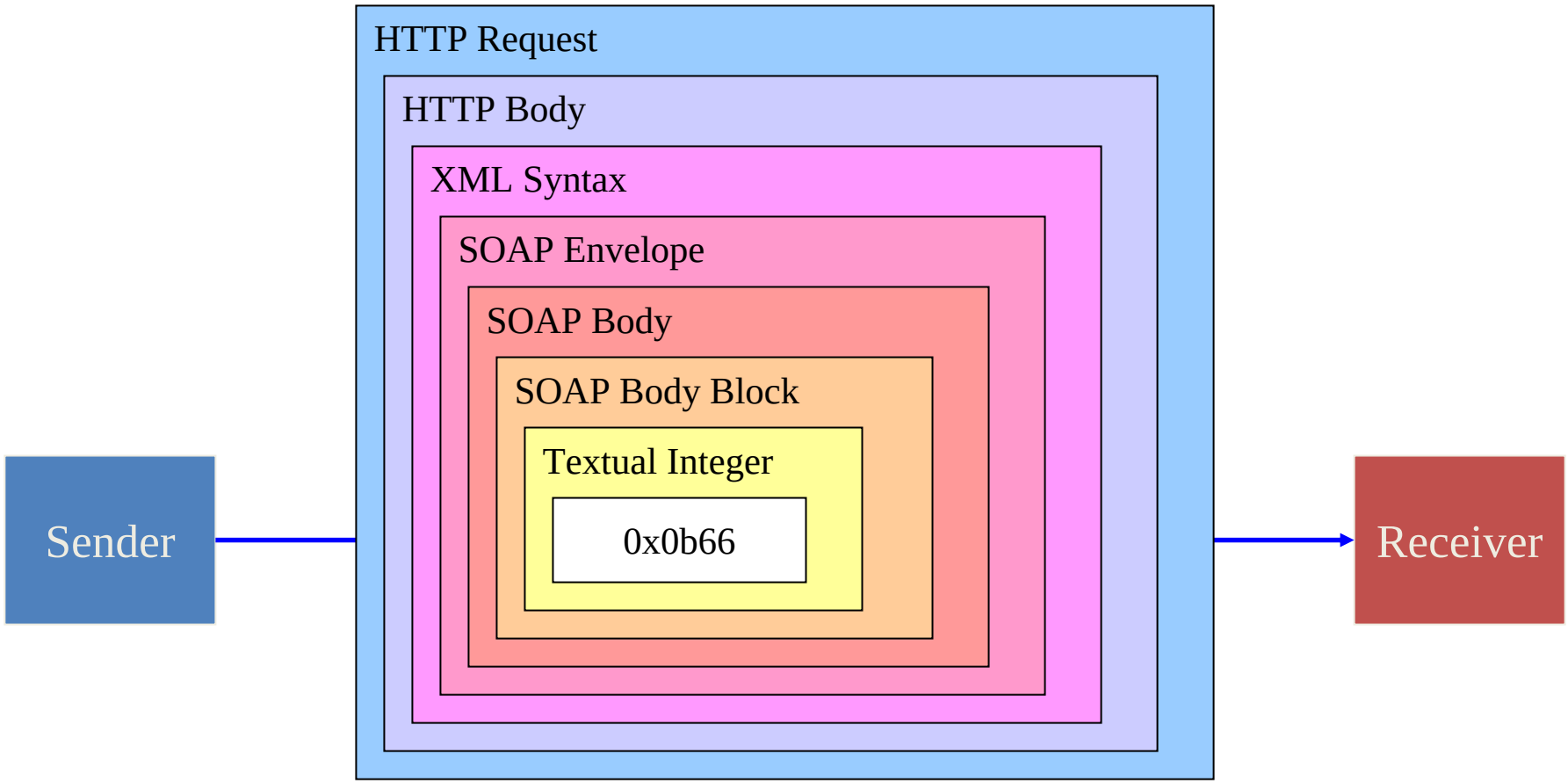
Message Representation

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/soap-envelope">
  <env:Header>
    <data:headerBlock
      xmlns:data="http://example.com/header"
      env:actor="http://example.com/actor"
      env:mustUnderstand="true">
      ...
    </data:headerBlock>
    ...
  </env:Header>
  <env:Body>
    <data:bodyBlock xmlns:data="http://example.com/header">
      ...
    </data:bodyBlock>
    ...
  </env:Body>
</env:Envelope>
```

Independence

- Independent of the wrapped data
  - True, but…
    - Only text data is allowed
    - Some data structures are difficult to represent
- Independent of the transport protocol
  - True, but…
    - The XML Protocol Working Group has requested additions to the HTTP 1.1 specification

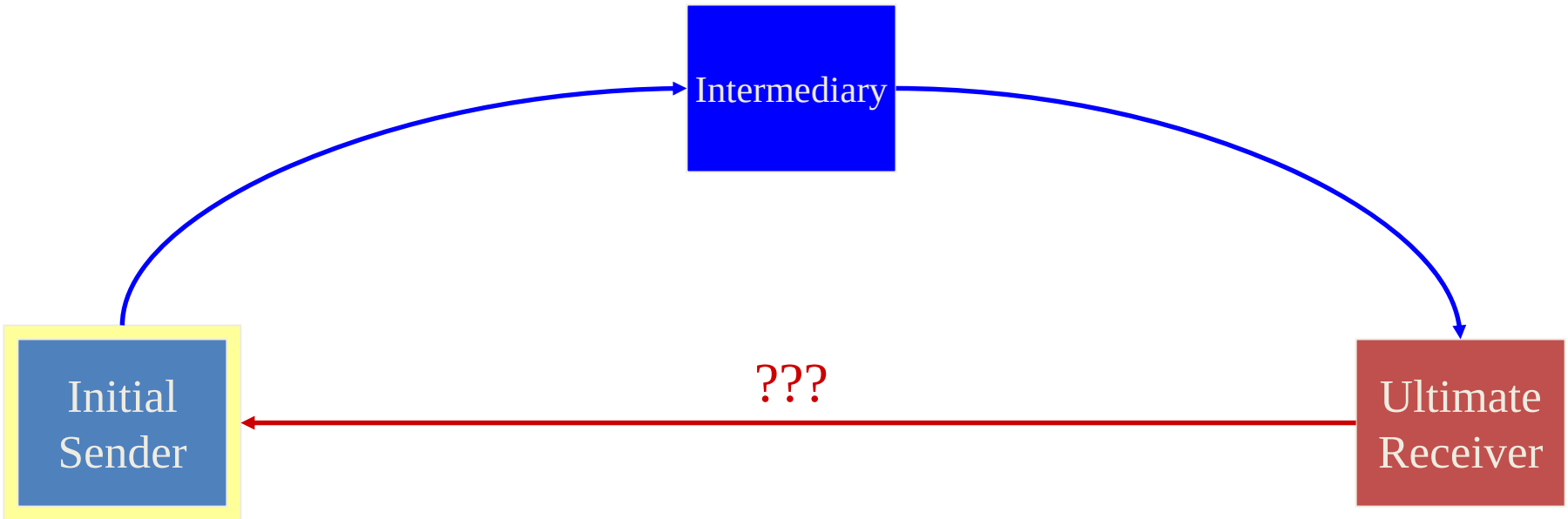
(In)Efficiency



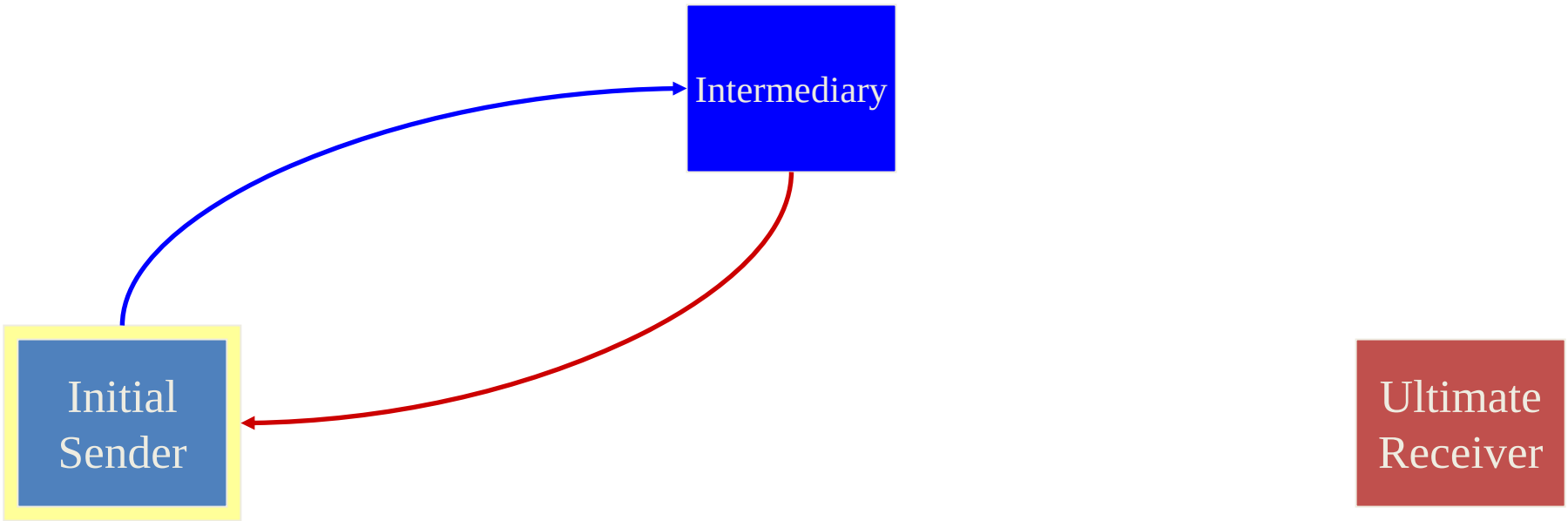
Exchange Paradigm

- Point-to-point exchange
  - Sender, receiver, possible intermediaries
- Uni-directional message exchange
  - True, but…
    - Specification includes semantics for dealing with faults
    - Faults cannot be ignored
    - Faults must be reported to the sending node

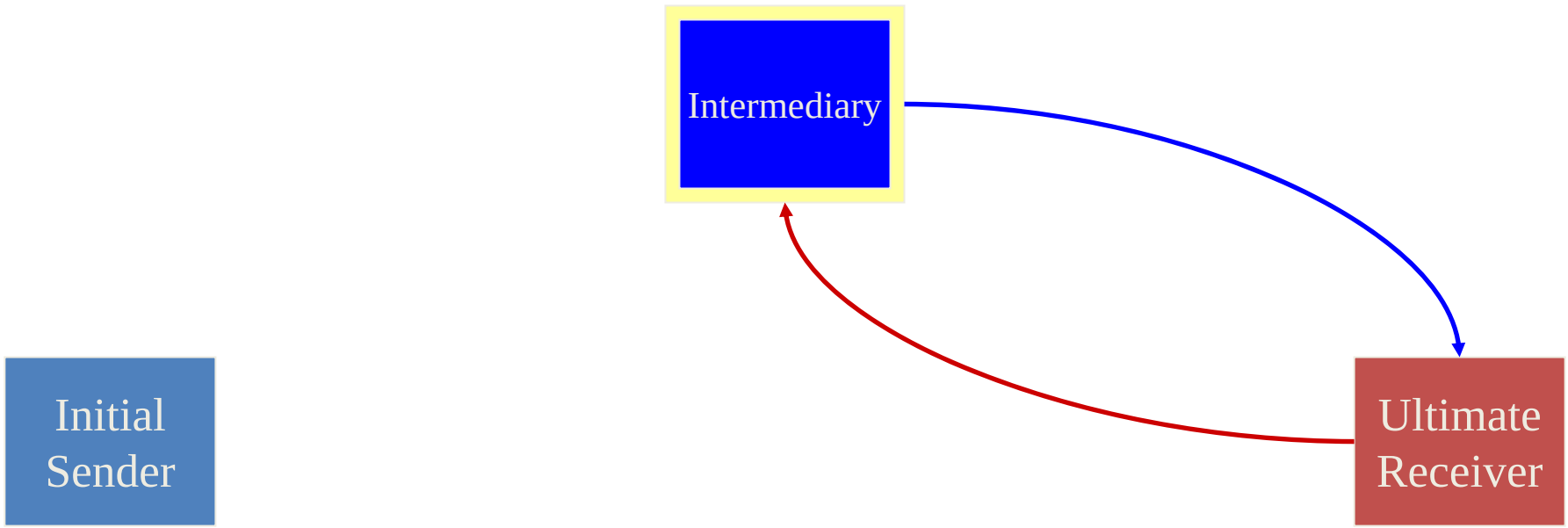
Uni-directional Exchange



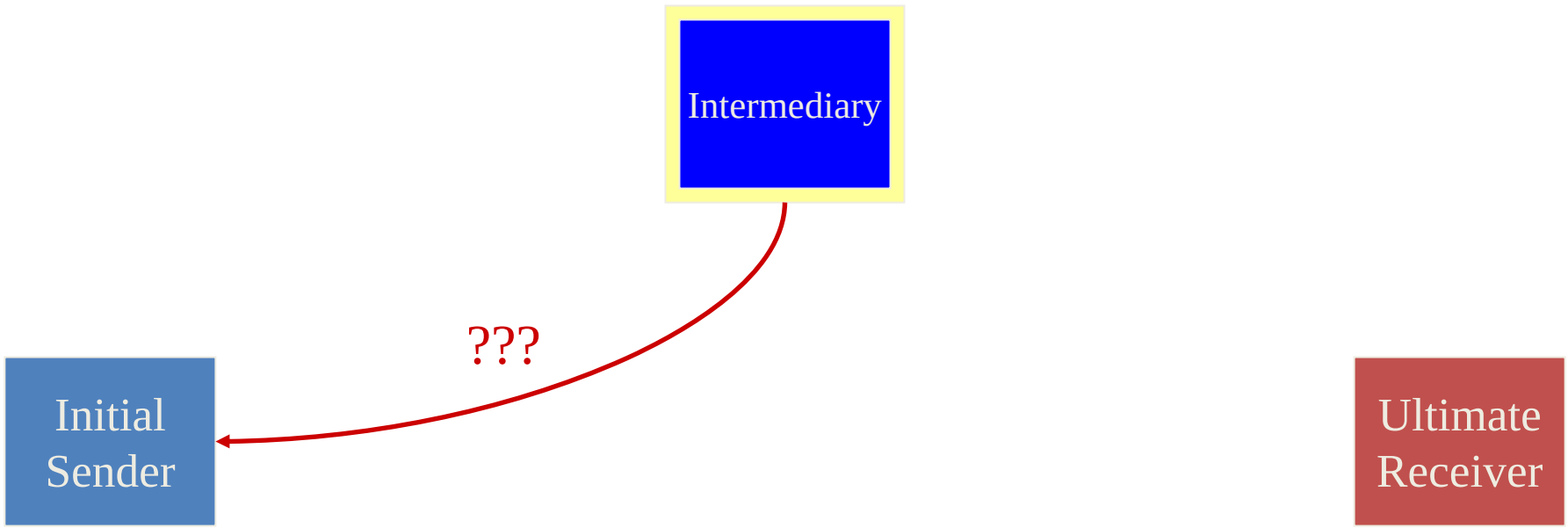
Bi-directional Exchange (Series)



Bi-directional Exchange (Series)



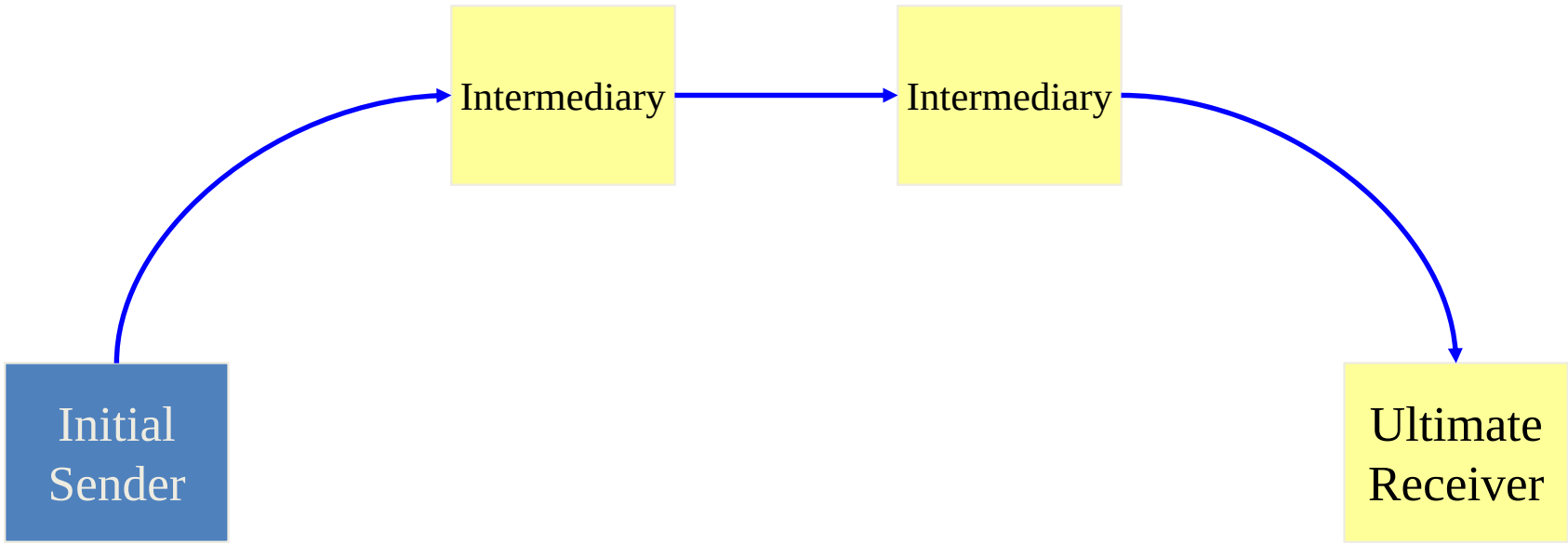
Bi-directional Exchange (Series)



Processing Model

- Point-to-point (sender-to-receiver) exchange, possibly via intermediaries
- Receivers assume “roles” as actors
- Header blocks can be specific to actors
  - Body blocks are always specific to the ultimate receiver
- Actors can be required to understand header blocks

Nodes


Intermediary
Intermediary
Initial Sender
Ultimate Receiver

Actors

- Standard actors
  - none
  - next
  - Anonymous
- Application-specific actors
  - Can be anything
  - Semantics implied by a URI

Actor-specific Header Blocks

```
<data:headerBlock
  xmlns:data="http://example.com/header"
  env:actor="http://example.com/actor1"
  env:mustUnderstand="true">
  ...
</data:headerBlock>
<data:headerBlock
  xmlns:data="http://example.com/header"
  env:actor="http://example.com/actor2"
  env:mustUnderstand="true">
  ...
</data:headerBlock>
<data:headerBlock
  xmlns:data="http://example.com/header"
  env:actor="http://example.com/actor2"
  ...
</data:headerBlock>
```

Intermediary Algorithm

- Receive message
- Process appropriate header blocks
  - Processing possibly produces a fault
- Remove processed headers
- Add new headers
- Send message

Ultimate Recipient Algorithm

- Receive message
- Process appropriate header blocks
  - Processing possibly produces a fault
- Process all body blocks
  - Processing possibly produces a fault

Higher-level Exchange Paradigms

- RPC
  - Fits well with HTTP 1.1 binding
  - Current activity within the XML Protocol Working Group
- Conversational
  - Fits well with general message passing, but awkward with HTTP 1.1 binding

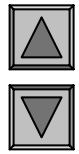
Normative References

- http://www.w3.org/2000/09/XML-Protocol-Charter
- http://www.w3.org/2002/ws/Activity.html
- http://www.w3.org/TR/xmlp-reqs/
- http://www.w3.org/TR/xmlp-am/
- http://www.w3.org/TR/xmlp-scenarios/
- http://www.w3.org/TR/soap12-part0/
- http://www.w3.org/TR/soap12-part1/
- http://www.w3.org/TR/soap12-part2/

Chapter 4 - Introduction to XHTML: Part 1

Outline

© 2001 Prentice Hall, Inc. All rights reserved.



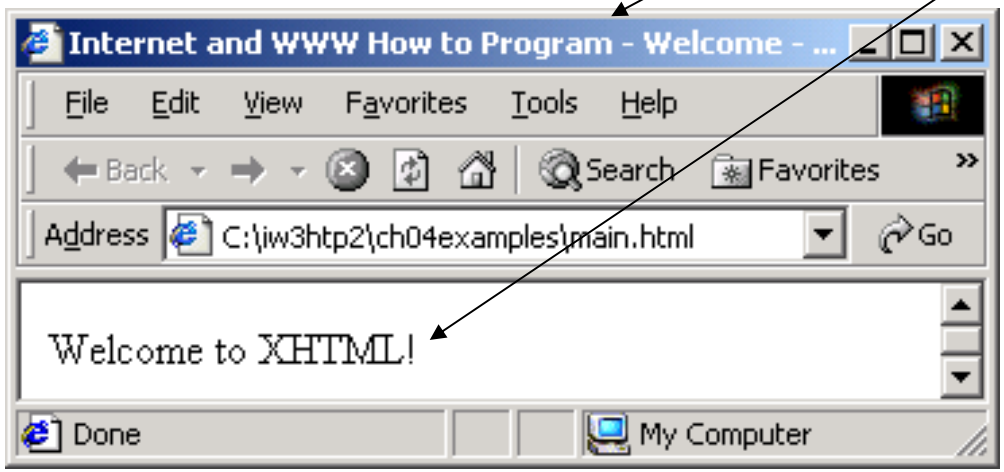
Outline

Main.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.1: main.html -->
6   <!-- Our first Web page  -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Welcome</title>
11     </head>
12
13     <body>
14        <p>Welcome to XHTML!</p>
15     </body>
16  </html>
```

The text between the **title** tags appears as the title of the web page.

Elements between the **body** tags of the html document appear in the body of the web page



Program Output

© 2001 Prentice Hall, Inc.
All rights reserved.

4.5 W3C XHTML Validation Service

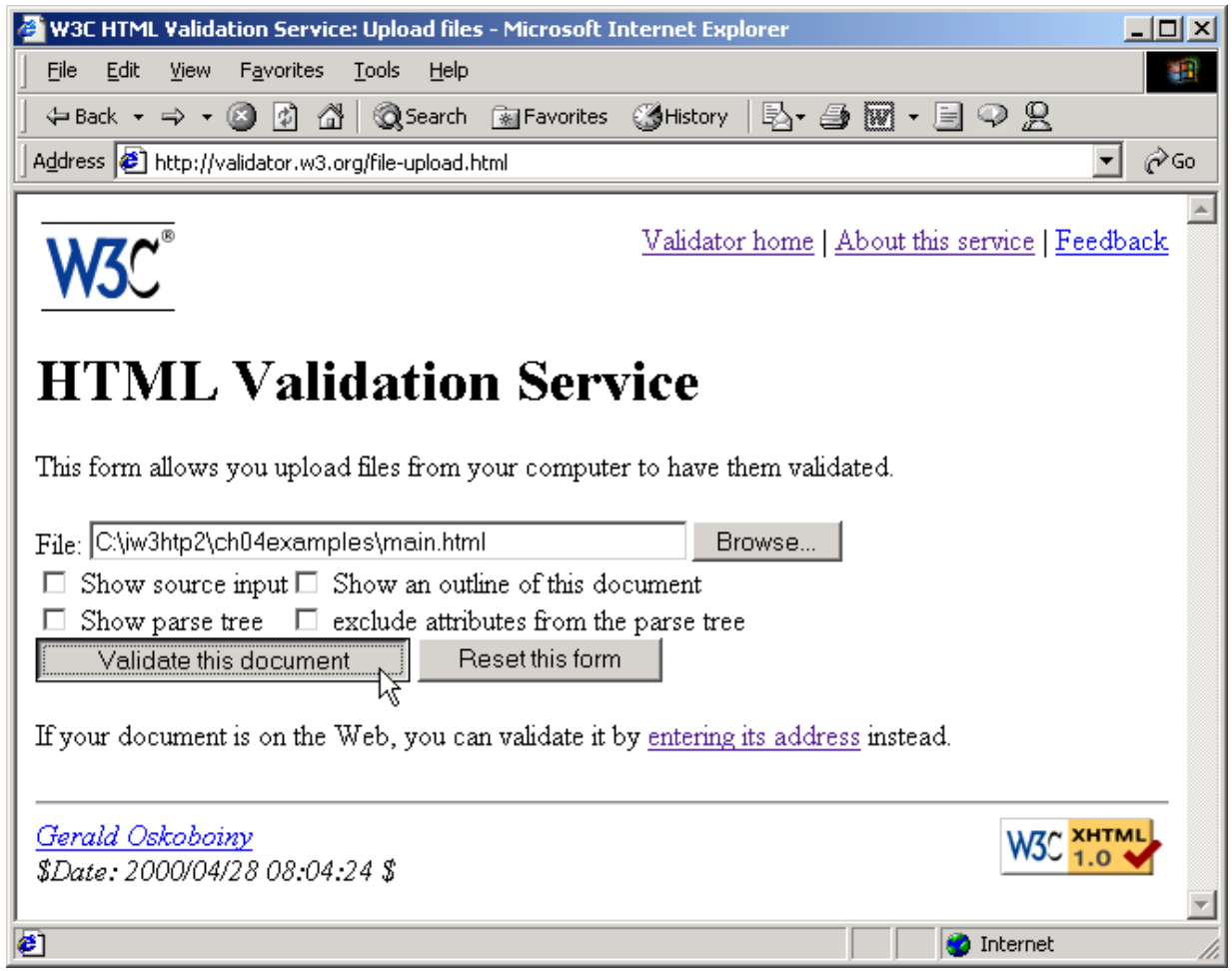


Fig. 4.2 Validating an XHTML document. (Courtesy of World Wide Web Consortium (W3C).)

© 2001 Prentice Hall, Inc. All rights reserved.

4.5 W3C XHTML Validation Service

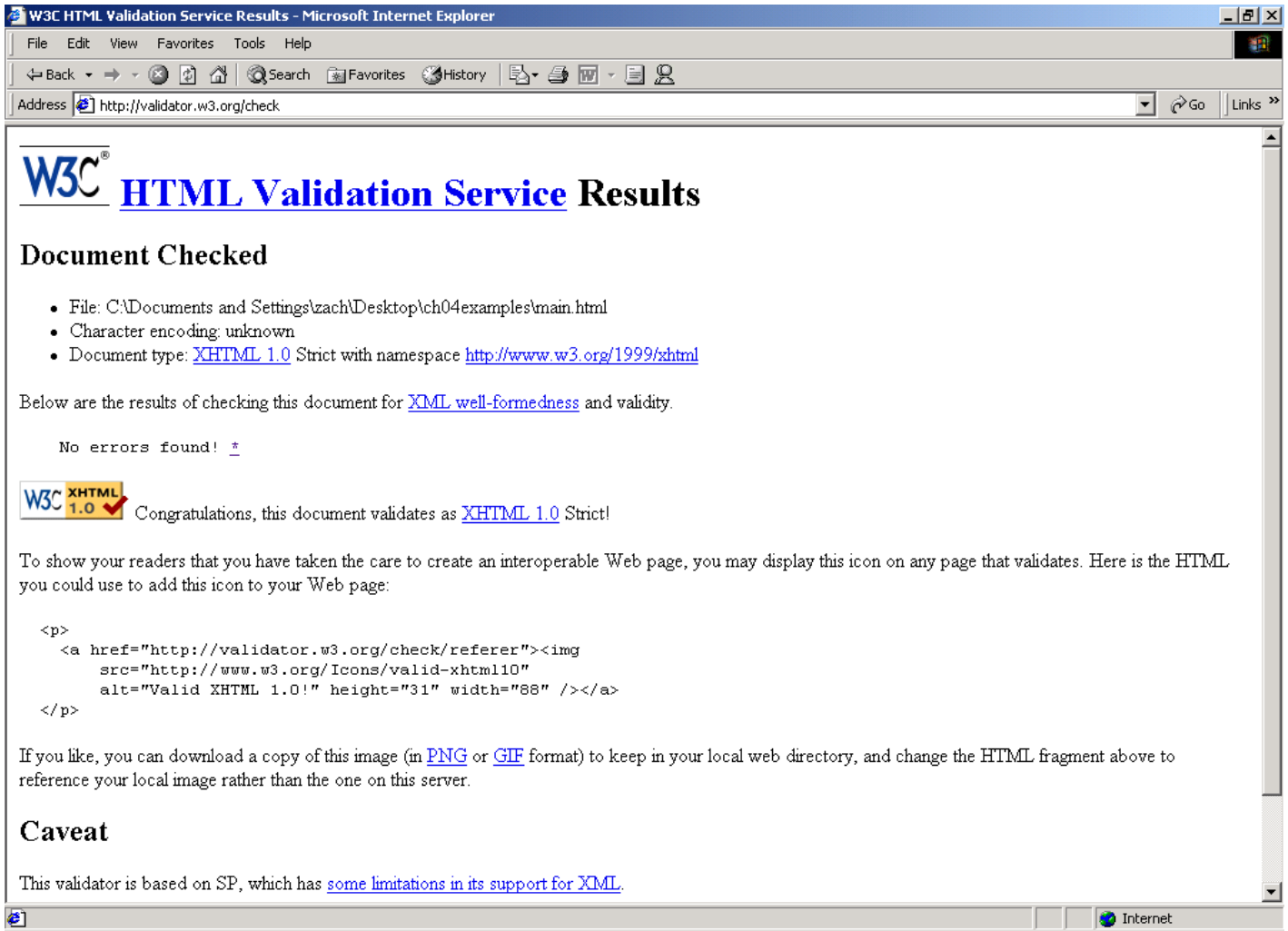
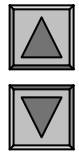


Fig. 4.3 XHTML validation results. (Courtesy of World Wide Web Consortium (W3C).)

© 2001 Prentice Hall, Inc. All rights reserved.



Outline

Header.html

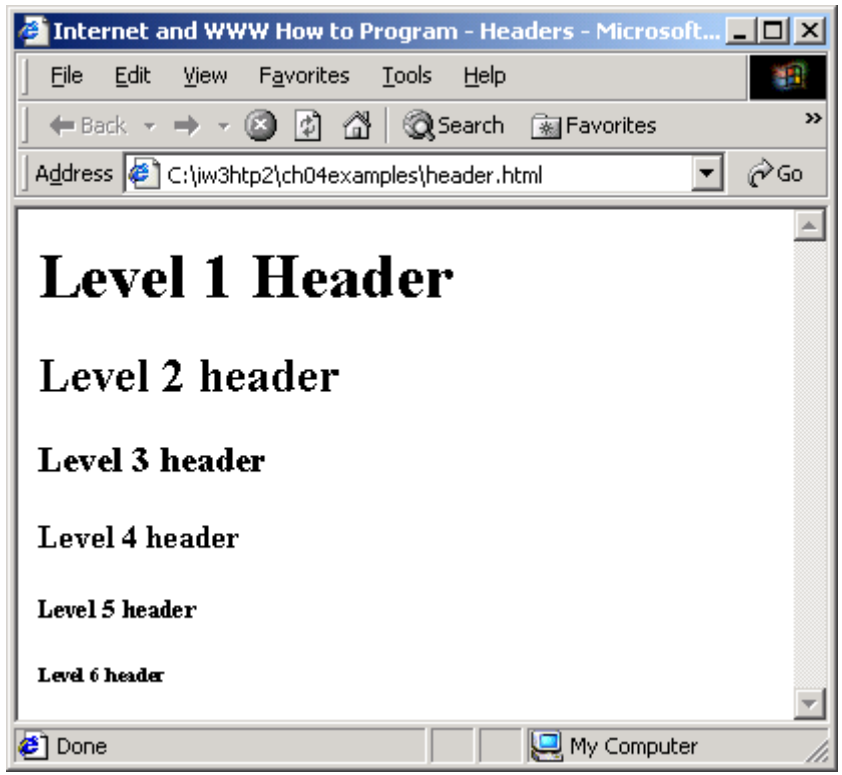
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.4: header.html -->
6   <!-- XHTML headers         -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Headers</title>
11     </head>
12
13     <body>
14
15        <h1>Level 1 Header</h1>
16        <h2>Level 2 header</h2>
17        <h3>Level 3 header</h3>
18        <h4>Level 4 header</h4>
19        <h5>Level 5 header</h5>
20        <h6>Level 6 header</h6>
21
22     </body>
23  </html>
```

The font size of the text between tags decreases as the header level increases.

Every XHTML document is required to have opening and closing **html** tags.

© 2001 Prentice Hall, Inc.
All rights reserved.

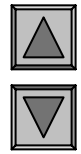
Outline



Program Output

Select a header based on the amount of emphasis you would like to give that text.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Links.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.5: links.html       -->
6   <!-- Introduction to hyperlinks -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Links</title>
11     </head>
12
13     <body>
14
15        <h1>Here are my favorite sites</h1>
16
17        <p><strong>Click on a name to go to that page.</strong></p>
18
19        <p><a href = "http://www.deitel.com">Deitel</a></p>
20
21        <p><a href = "http://www.prenhall.com">Prentice Hall</a></p>
22
23        <p><a href = "http://www.yahoo.com">Yahoo!</a></p>
24
25        <p><a href = "http://www.usatoday.com">USA Today</a></p>
26
27     </body>
28  </html>
```

Text between **strong** tags will appear bold.

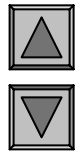
Linking is accomplished in XHTML with the anchor (**a**) element.

The text between the **a** tags is the anchor for the link.

Elements placed between paragraph tags will be set apart from other elements on the page with a vertical line before and after it.

The anchor links to the page that's value is given by the **href** attribute.

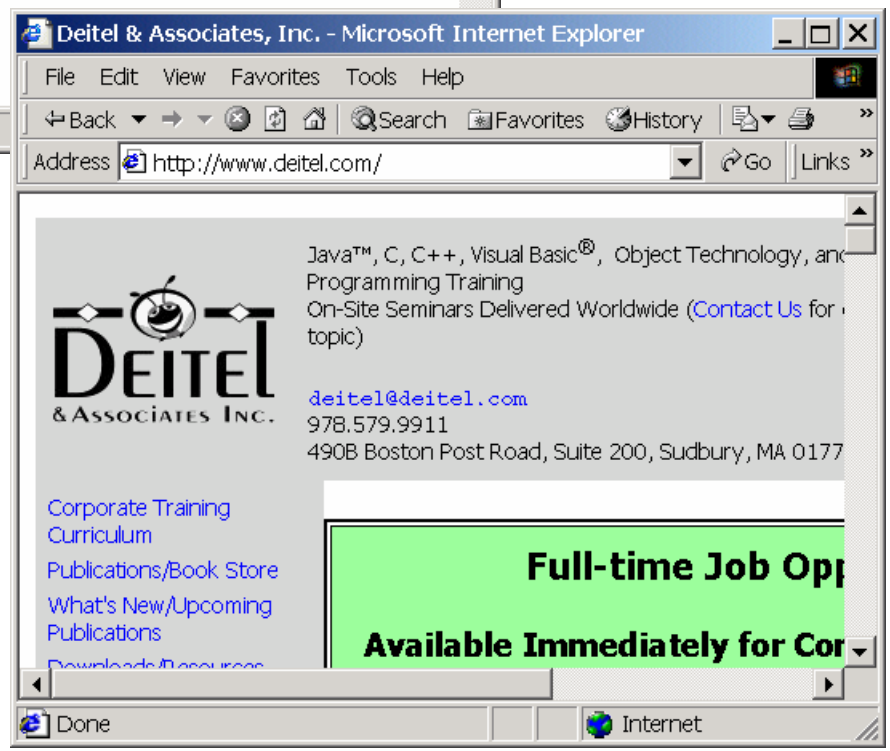
© 2001 Prentice Hall, Inc.
All rights reserved.



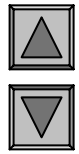
Outline

Program Output

Clicking on the **"Deitel"** link will open up the Deitel homepage in a new browser window.



© 2001 Prentice Hall, Inc.
All rights reserved.



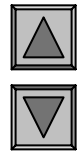
Outline

Contact.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.6: contact.html  -->
6   <!-- Adding email hyperlinks -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Contact Page
11        </title>
12     </head>
13
14     <body>
15
16        <p>My email address is
17        <a href = "mailto:deitel@deitel.com"> deitel@deitel.com
18        </a>. Click the address and your browser will open an
19        email message and address it to me.</p>
20
21     </body>
22  </html>
```

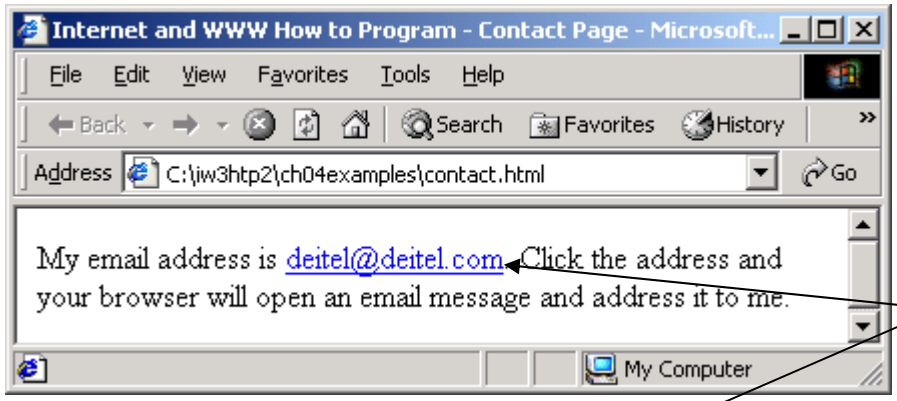
To create an email link include **“mailto:”** before the email address in the **href** attribute.

© 2001 Prentice Hall, Inc.
All rights reserved.

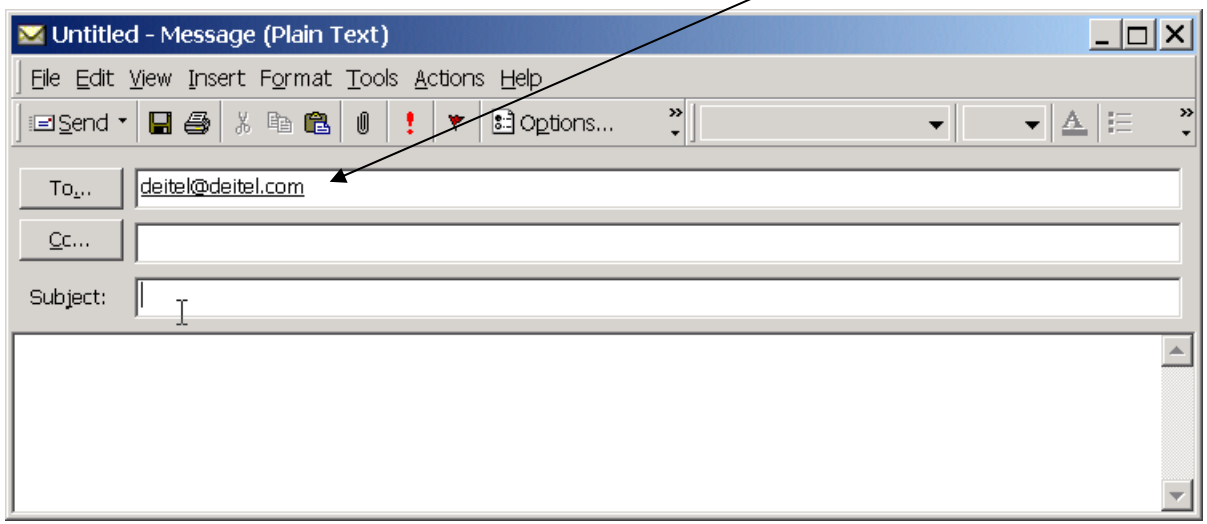


Outline

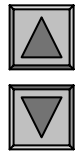
Program Output



When a user clicks on an email link, an email message addressed to the value of the link will open up.



© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

Picture.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.7: picture.html   -->
6   <!-- Adding images with XHTML -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Welcome</title>
11     </head>
12
13     <body>
14
15        <p><img src = "xmlhtp.jpg" height = "238" width = "183"
16           alt = "XML How to Program book cover" />
17           <img src = "jhtp.jpg" height = "238" width = "183"
18           alt = "Java How to Program book cover" /></p>
19     </body>
20  </html>
```

The value of the **src** attribute of the image element is the location of the image file.

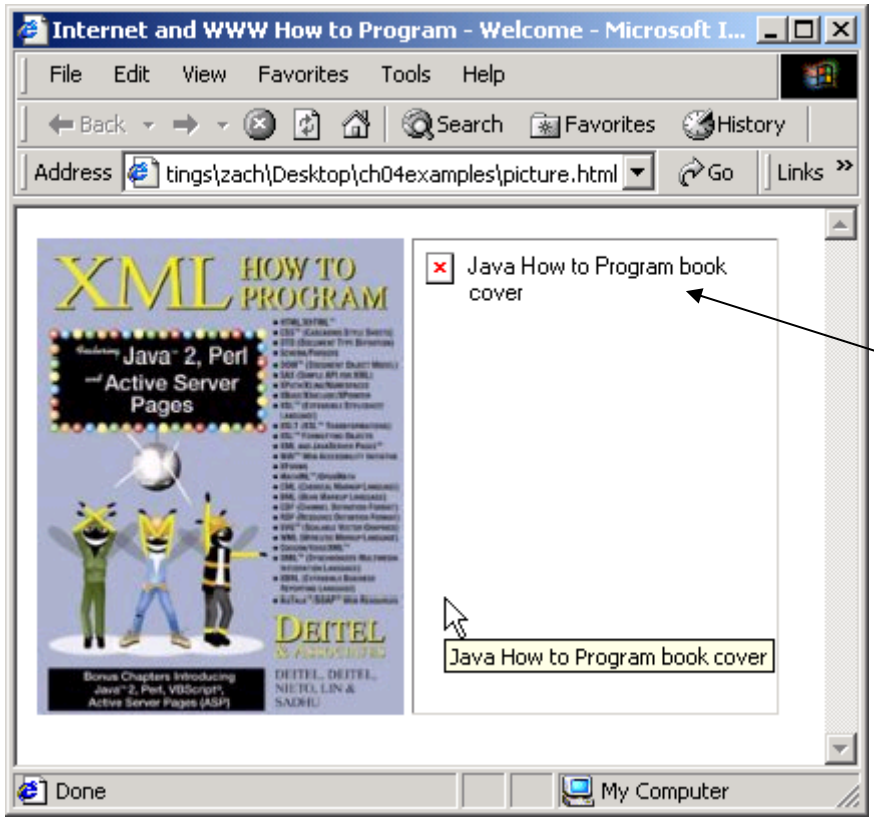
The value of the **alt** attribute gives a description of the image. This description is displayed if the image cannot be displayed.

The **height** and **width** attributes of the image element give the height and width of the image.

© 2001 Prentice Hall, Inc.
All rights reserved.

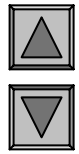
Outline

Program Output



The second image could not be displayed properly, so the value of its **alt** attribute is displayed instead.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

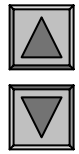
Nav.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.8: nav.html           -->
6   <!-- Using images as link anchors -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Navigation Bar
11        </title>
12     </head>
13
14     <body>
15
16        <p>
17           <a href = "links.html">
18           <img src = "buttons/links.jpg" width = "65"
19              height = "50" alt = "Links Page" /></a><br />
20
21           <a href = "list.html">
22           <img src = "buttons/list.jpg" width = "65"
23              height = "50" alt = "List Example Page" /></a><br />
24
25           <a href = "contact.html">
26           <img src = "buttons/contact.jpg" width = "65"
27              height = "50" alt = "Contact Page" /></a><br />
28
29           <a href = "header.html">
30           <img src = "buttons/header.jpg" width = "65"
31              height = "50" alt = "Header Page" /></a><br />
32
33           <a href = "table.html">
34           <img src = "buttons/table.jpg" width = "65"
35              height = "50" alt = "Table Page" /></a><br />
```

Placing an image element between anchor tags, creates an image that is an anchor for a link.

A line break will render an empty line on a web page.

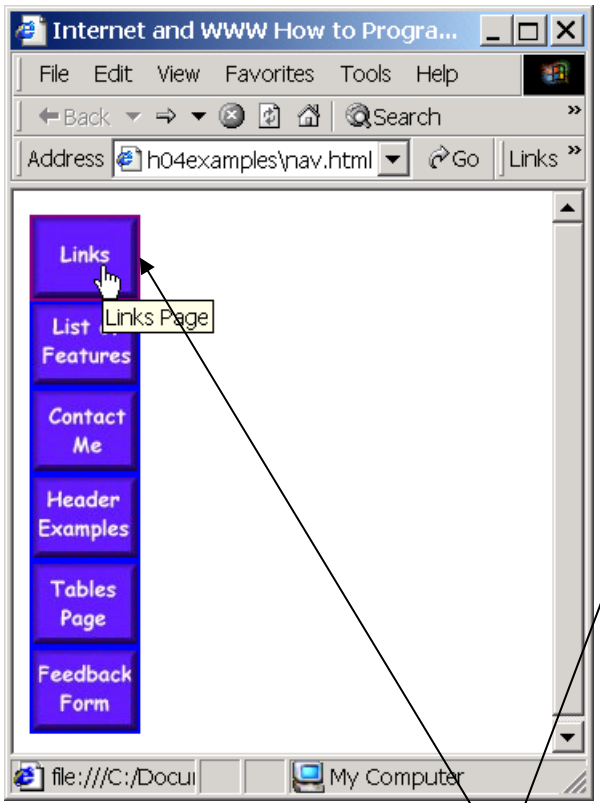
© 2001 Prentice Hall, Inc.
All rights reserved.

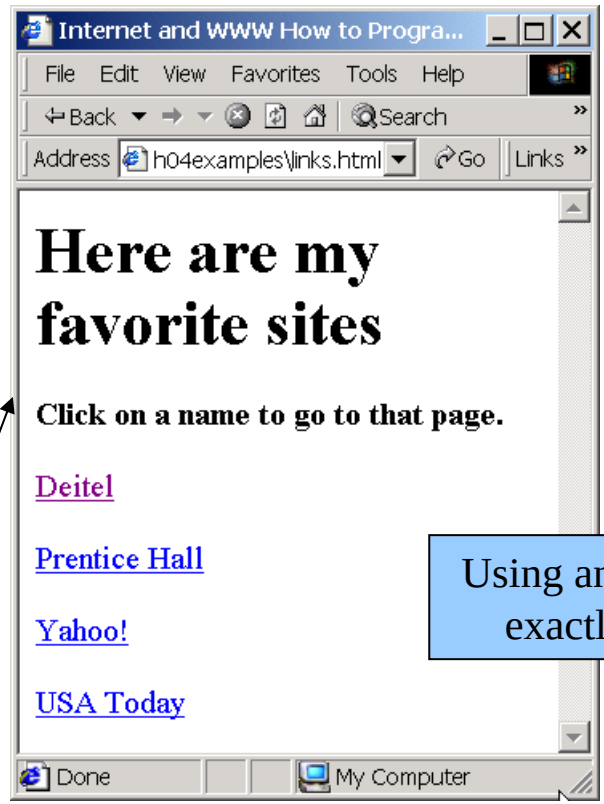


Outline

```
36
37            <a href = "form.html">
38            <img src = "buttons/form.jpg" width = "65"
39               height = "50" alt = "Feedback Form" /></a><br />
40         </p>
41
42      </body>
43   </html>
```

Nav.html



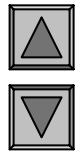


Program Output

Using an image as an anchor works exactly the same as using text.

Clicking on the image entitled **"links"** brings the user to the page on the right.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

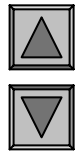
Contact2.html

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 4.9: contact2.html       -->
6    <!-- Inserting special characters  -->
7
8    <html xmlns = "http://www.w3.org/1999/xhtml">
9       <head>
10         <title>Internet and WWW How to Program - Contact Page
11         </title>
12      </head>
13
14      <body>
15
16         <!-- Special characters are entered -->
17         <!-- using the form &code;          -->
18         <p>My email address is
19         <a href = "mailto:deitel@deitel.com">deitel@deite
20         </a>. Click on the address and your browser will
21         automatically open an email message and address i
22         address.</p>
23
24         <hr /> <!-- Inserts a horizontal rule -->
25
26         <p>All information on this site is <strong>&copy;</strong>
27         Deitel <strong>&amp;</strong> Associates, Inc. 2002.</p>
28
29         <!-- Text can be struck out with a set of             -->
30         <!-- <del>...</del> tags, it can be set in subscript  -->
31         <!-- with <sub>...</sub>, and it can be set into      -->
32         <!-- superscript with <sup...</sup>                   -->
33         <p><del>You may download 3.14 x 10<sup>2</sup>
34         characters worth of information from this site.</del>
35         Only <sub>one</sub> download per hour is permitted.</p>
```

The horizontal rule element renders a horizontal line on the web page.

Special characters are denoted with an ampersand (**&**) and an abbreviation for that character. In this case, the special characters are ampersand and copyright.

© 2001 Prentice Hall, Inc.
All rights reserved.

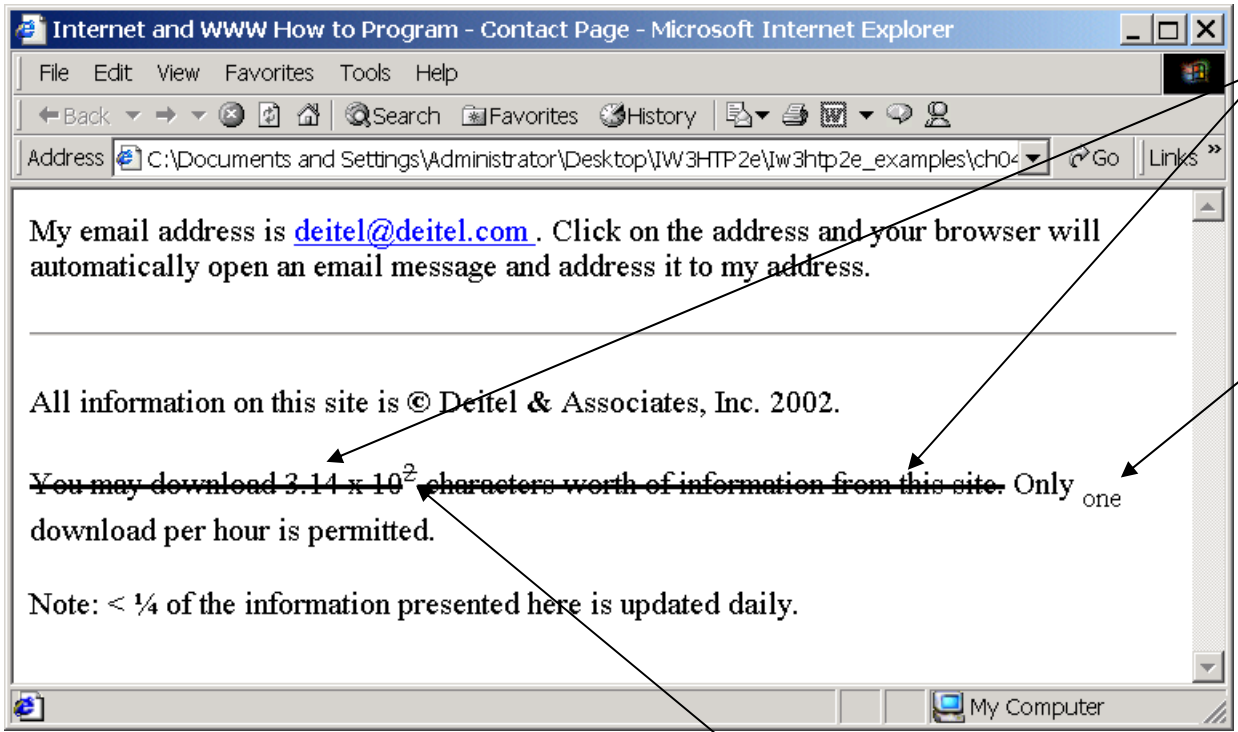


Outline

Contact2.html

```
36
37          <p>Note: <strong>&lt; &frac14;</strong> of the information
38          presented here is updated daily.</p>
39
40       </body>
41    </html>
```

Program Output

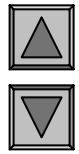


Text placed between **del** tags is struck out..

Text placed between the **sub** tags is subscripted.

Text placed between the **sup** tags is superscripted.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

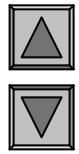
Links2.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 4.10: links2.html                -->
6   <!-- Unordered list containing hyperlinks -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Links</title>
11     </head>
12
13     <body>
14
15        <h1>Here are my favorite sites</h1>
16
17        <p><strong>Click on a name to go to that page.</strong></p>
18
19        <ul>
20           <li><a href = "http://www.deitel.com">Deitel</a></li>
21
22           <li><a href = "http://www.prenhall.com">Prentice Hall
23           </a></li>
24
25           <li><a href = "http://www.yahoo.com">Yahoo!</a></li>
26
27           <li><a href = "http://www.usatoday.com">USA Today</a>
28           </li>
29        </ul>
30     </body>
31  </html>
```

The entries in an unordered list must be included between the <**ul**> and </**ul**> tags.

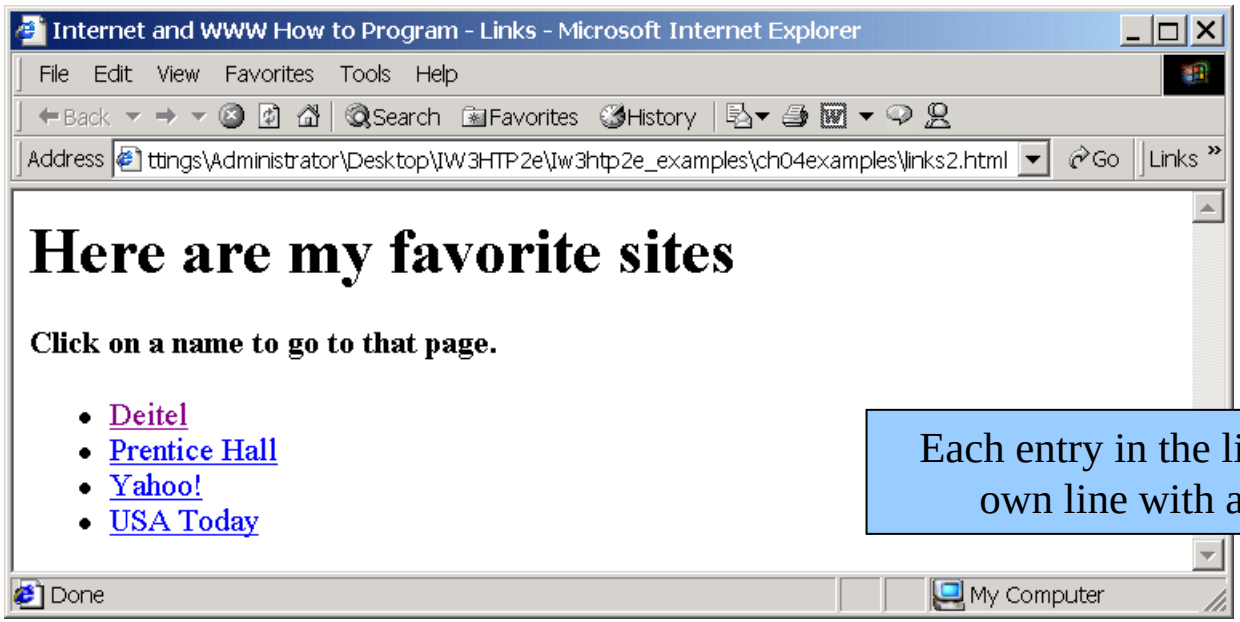
An entry in the list must be placed between the <**li**> and </**li**> tags.

© 2001 Prentice Hall, Inc.
All rights reserved.



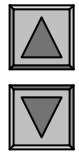
Outline

Program Output



Each entry in the list is rendered on its own line with a bullet before it.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

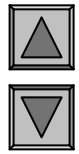
List.html

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5   <!-- Fig. 4.11: list.html                -->
6   <!-- Advanced Lists: nested and ordered -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Internet and WWW How to Program - Lists</title>
11     </head>
12
13     <body>
14
15        <h1>The Best Features of the Internet</h1>
16
17        <ul>
18           <li>You can meet new people from countries around
19              the world.</li>
20           <li>You have access to new media as it becomes public:
21
22              <!-- This starts a nested list, which uses a   -->
23              <!-- modified bullet. The list ends when you   -->
24              <!-- close the <ul> tag                        -->
25              <ul>
26                 <li>New games</li>
27                 <li>New applications
28
29                    <!-- Another nested list -->
30                    <ol type = "I">
31                       <li>For business</li>
32                       <li>For pleasure</li>
33                    </ol> <!-- Ends the double nested list -->
34                 </li>
35
```

By inserting a list as an entry in another list, lists can be nested.

Entries for an ordered list must be placed between the **
** and **
** tags.

© 2001 Prentice Hall, Inc.
All rights reserved.



Outline

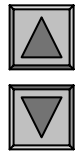
List.html

```
36                <li>Around the clock news</li>
37                <li>Search engines</li>
38                <li>Shopping</li>
39                <li>Programming
40
41                   <ol type = "a">
42                      <li>XML</li>
43                      <li>Java</li>
44                      <li>XHTML</li>
45                      <li>Scripts</li>
46                      <li>New languages</li>
47                   </ol>
48
49                </li>
50
51             </ul> <!-- Ends the first level nested list -->
52          </li>
53
54          <li>Links</li>
55          <li>Keeping in touch with old friends</li>
56          <li>It is the technology of the future!</li>
57
58       </ul>   <!-- Ends the primary unordered list -->
59
60       <h1>My 3 Favorite <em>CEOs</em></h1>
61
62       <!-- ol elements without a type attribute           -->
63       <!-- have a numeric sequence type (i.e., 1, 2, ...) -->
64       <ol>
65          <li>Harvey Deitel</li>
66          <li>Bill Gates</li>
67          <li>Michael Dell</li>
68       </ol>
```

The **type** attribute of the ordered list element allows you to select a sequence type to order the list.

Text placed between the **em** tags will be italicized.

© 2001 Prentice Hall, Inc.
All rights reserved.

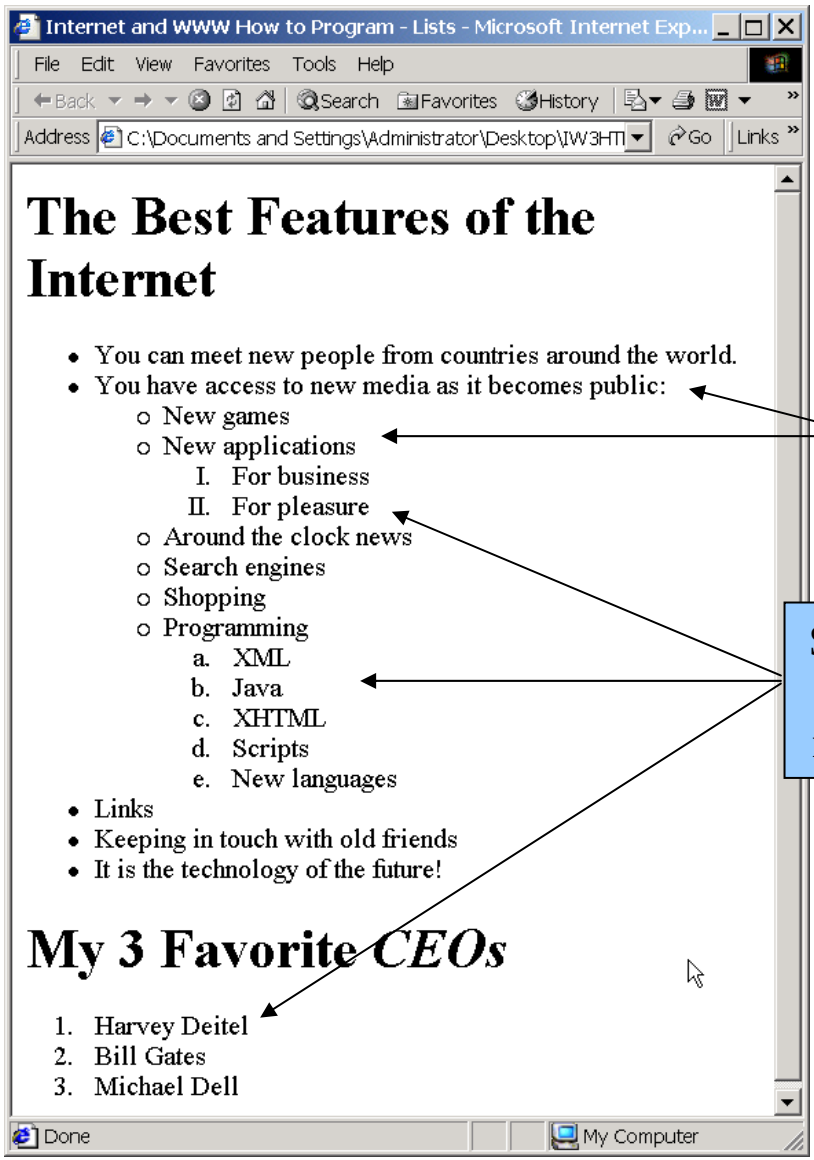


Outline

```
69
70       </body>
71    </html>
```

List.html

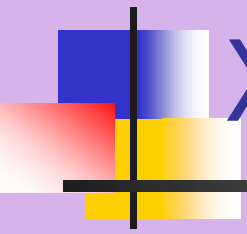
Program Output



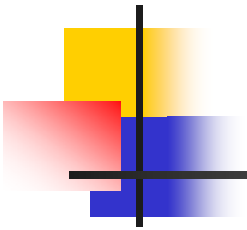
Nested lists are indented underneath the main list.

Some sequence types available to order lists are roman numerals, letters and numbers.

© 2001 Prentice Hall, Inc.
All rights reserved.

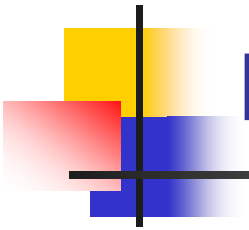


XML Introduction



Index

- Markup Language: SGML, HTML, XML
- An XML example
- Why is XML important
- XML introduction
- XML applications
- XML support



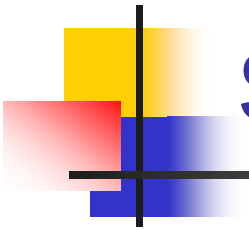
Markup Language

A markup language must specify

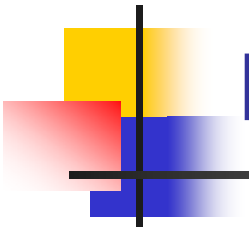
- What markup is allowed
- What markup is required
- How markup is to be distinguished from text
- What the markup means

*XML only specify the first three, the fourth is specified by DTD

SGML(ISO 8879)



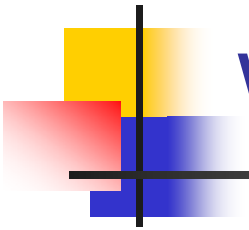
- Standard Generalized Markup Language
- The international standard for defining descriptions of structure and content in text documents
- Interchangeable: device-independent, system-independent
- tags are not predefined
- Using DTD to validate the structure of the document
- Large, powerful, and very complex
- Heavily used in industrial and commercial for over a decade



HTML(RFC 1866)

- HyperText Markup Language
- A small SGML application used on web (a DTD and a set of processing conventions)
- Can only use a predefined set of tags

What Is XML?



- eXtensible Markup Language
- A simplified version of SGML
- Maintains the most useful parts of SGML
- Designed so that SGML can be delivered over the Web
- More flexible and adaptable than HTML
- XHTML -- a reformulation of HTML 4 in XML 1.0

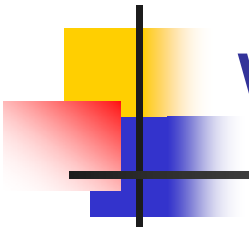
Difference between XML and HTML

XML was designed to carry data, not displaying data

- XML is not a replacement for HTML.
- Different goals:
  XML was designed to describe data and to focus on what data is.
  HTML was designed to display data and to focus on how data looks.
- HTML is about displaying information, XML is about describing information.

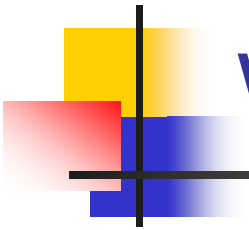
An example of XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```



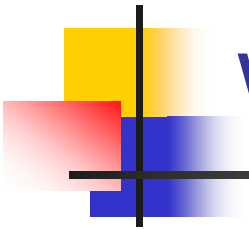
Why Is XML Important?

- Plain Text
  - Easy to edit
  - Useful for storing small amounts of data
  - Possible to efficiently store large amounts of XML data through an XML front end to a database
- Data Identification
  - Tell you what kind of data you have
  - Can be used in different ways by different applications



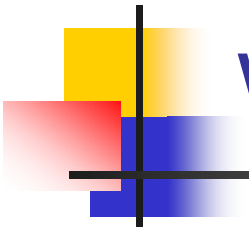
Why Is XML Important?

- Stylability
  - Inherently style-free
  - XSL---Extensible Stylesheet Language
  - Different XSL formats can then be used to display the same data in different ways
- Inline Reusabiliy
  - Can be composed from separate entities
  - Modularize your documents without resorting to links



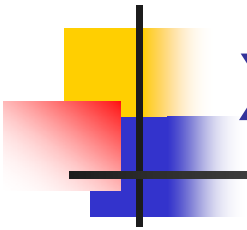
Why is XML important?

- Linkability -- XLink and XPointer
  - Simple unidirectional hyperlinks
  - Two-way links
  - Multiple-target links
  - “Expanding” links
- Easily Processed
  - Regular and consistent notation
  - Vendor-neutral standard



Why is XML important?

- Hierarchical
  - Faster to access
  - Easier to rearrange



XML Specifications

- XML 1.0

  Defines the syntax of XML

- XPointer, XLink

  Defines a standard way to represent links between resources

- XSL

  Defines the standard stylesheet language for XML

XML Building blocks

- Element

  Delimited by angle brackets

  Identify the nature of the content they surround

  General format: <element> ... </element>

  Empty element: </empty-Element>

- Attribute

  Name-value pairs that occur inside start-tags after element name, like: <element attribute="value">

XML Building blocks--Prolog

- The part of an XML document that precedes the XML data

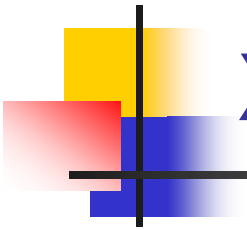
- Includes

  A declaration: version [, encoding, standalone]

  An optional DTD (Document Type Definition)

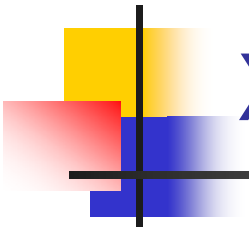
- Example

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```



XML Syntax

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML elements must be properly nested
- All XML documents must have a root tag
- Attribute values must always be quoted
- With XML, white space is preserved
- With XML, a new line is always stored as LF
- Comments in XML: <!-- This is a comment -->



XML Elements

- XML Elements are Extensible

  XML documents can be extended to carry more information

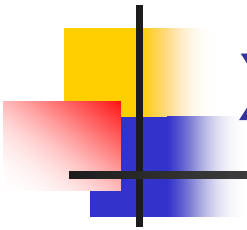
- XML Elements have Relationships

  Elements are related as parents and children

- Elements have Content

  Elements can have different content types: **element** content, **mixed** content, **simple** content, or **empty** content and **attributes**

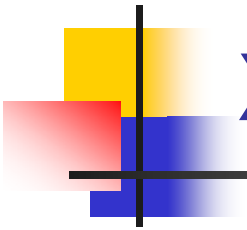
- XML elements must follow the naming rules



XML Attributes

- Located in the start tag of elements
- Provide additional information about elements
- Often provide information that is not a part of data
- Must be enclosed in quotes
- Should I use an element or an attribute?

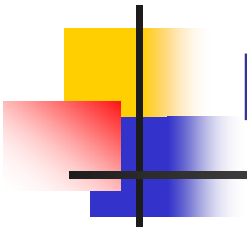
  metadata (data about data) should be stored as attributes, and that data itself should be stored as elements



XML Validation

- "Well Formed" XML document

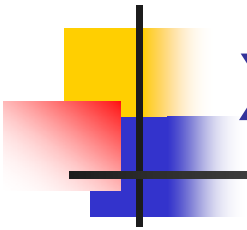
  --correct XML syntax
- "Valid" XML document
  - “well formed”
  - Conforms to the rules of a DTD (Document Type Definition)
- XML DTD
  - defines the legal building blocks of an XML document
  - Can be inline in XML or as an external reference
- XML Schema
  - an XML based alternative to DTD, more powerful
  - Support namespace and data types



Displaying XML

- XML documents do not carry information about how to display the data

- We can add display information to XML with
  - CSS (Cascading Style Sheets)
  - XSL (eXtensible Stylesheet Language) --- preferred



XML Application1—Separate data

XML can Separate Data from HTML

- Store data in separate XML files
- Using HTML for layout and display
- Using Data Islands
- Data Islands can be bound to HTML elements

Benefits:

Changes in the underlying data will not require any changes to your HTML

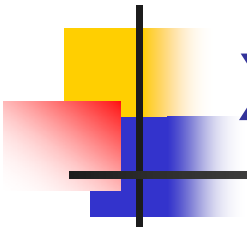
XML Application2—Exchange data

XML is used to Exchange Data

- Text format
- Software-independent, hardware-independent
- Exchange data between incompatible systems, given that they agree on the same tag definition.
- Can be read by many different types of applications

Benefits:

- Reduce the complexity of interpreting data
- Easier to expand and upgrade a system



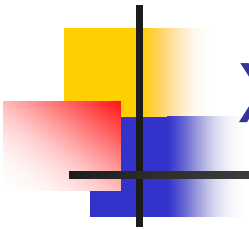
XML Application3—Store Data

XML can be used to Store Data

- Plain text file
- Store data in files or databases
- Application can be written to store and retrieve information from the store
- Other clients and applications can access your XML files as data sources

Benefits:

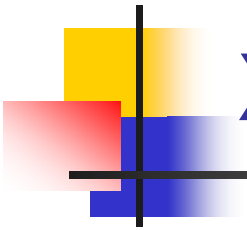
Accessible to more applications



XML Application4—Create new language

XML can be used to Create new Languages

- WML (Wireless Markup Language) used to markup Internet applications for handheld devices like mobile phones (WAP)
- MusicXML used to publishing musical scores

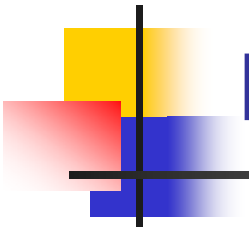


XML support in IE 5.0+

Internet Explorer 5.0 has the following XML support:

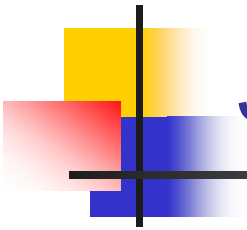
- Viewing of XML documents
- Full support for W3C DTD standards
- XML embedded in HTML as Data Islands
- Binding XML data to HTML elements
- Transforming and displaying XML with XSL
- Displaying XML with CSS
- Access to the XML DOM (Document Object Model)

*Netscape 6.0 also have full XML support



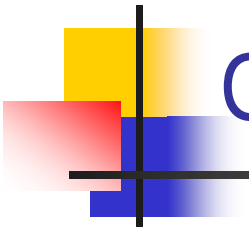
Microsoft XML Parser

- Comes with IE 5.0
- The parser features a language-neutral programming model that supports:
  - JavaScript, VBScript, Perl, VB, Java, C++ and more
  - W3C XML 1.0 and XML DOM
  - DTD and validation



Java APIs for XML

- JAXP: Java API for XML Processing
- JAXB: Java Architecture for XML Binding
- JDOM: Java DOM
- DOM4J: an alternative to JDOM
- JAXM: Java API for XML Messaging (asynchronous)
- JAX-RPC: Java API for XML-based Remote Process Communications (synchronous)
- JAXR: Java API for XML Registries



Conclusion

- XML is a self-descriptive language
- XML is a powerful language to describe structure data for web application
- XML is currently applied in many fields
- Many vendors already supports or will support XML