

Introduction

- ▶ Extensible HyperText Markup Language
 - XHTML
 - A markup language
 - Separation of the presentation of a document from the structure of the document's information
 - Based on HTML
 - Technology of the World Wide Web Consortium (W3C)
 - Original version designed in conjunction with first browser

Origins and Evolution of HTML

- ▶ HTML was defined with SGML
- ▶ Original intent of HTML: General layout of documents that could be displayed by a wide variety of computers
- ▶ Recent versions:
 - HTML 4.0 – 1997
 - Introduced many new features and deprecated many older features
 - HTML 4.01 - 1999 - A cleanup of 4.0
 - XHTML 1.0 - 2000
 - Just 4.01 defined using XML, instead of SGML
 - XHTML 1.1 – 2001
 - Modularized 1.0, and drops frames
 - We'll stick to 1.1, except for frames

Origins and Evolution of HTML (continued)

▶ Reasons to use XHTML, rather than HTML:

1. HTML has lax syntax rules, leading to sloppy and sometime ambiguous documents
 - XHTML syntax is much more strict, leading to clean and clear documents in a standard form
2. HTML processors do not even enforce the few syntax rules that do exist in HTML
3. The syntactic correctness of XHTML documents can be validated

Basic Syntax

▶ Elements are defined by tags (markers)

- Tag format:
 - Opening tag: **<name>**
 - Closing tag: **</name>**
- The opening tag and its closing tag together specify a container for the *content* they enclose

Basic Syntax (continued)

- ▶ Not all tags have content
 - If a tag has no content, its form is `<name />`
- ▶ The container and its content together are called an *element*
- ▶ Many tags have attributes. An attribute more fully specifies information about the content of the container.
- ▶ If a tag has attributes, they appear between its name and the right bracket of the opening tag
- ▶ Comment form: `<!-- ... -->`
- ▶ Browsers ignore comments, unrecognizable tags, line breaks, multiple spaces, and tabs
- ▶ Tags are suggestions to the browser, even if they are recognized by the browser

Editing XHTML

- ▶ XHTML documents
 - Text editor (e.g. Notepad, Wordpad, emacs, etc.) or
 - Use software like Expression Web or Dreamweaver
 - .html or .htm file-name extension
 - Web server
 - Stores XHTML documents
 - Web browser
 - Requests XHTML documents

HTML Document Structure

- ▶ Every XHTML document must begin with:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

```
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
```

```
http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
```

- ▶ `<html>`, `<head>`, `<title>`, and `<body>` are required in every document
- ▶ The whole document must have `<html>` as its root
- ▶ `html` must have the `xmlns` attribute:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
```
- ▶ A document consists of a head and a body
- ▶ The `<title>` tag is used to give the document a title, which is normally displayed in the browser's window title bar (at the top of the display)
- ▶ Prior to XHTML 1.1, a document could have either a body or a frameset

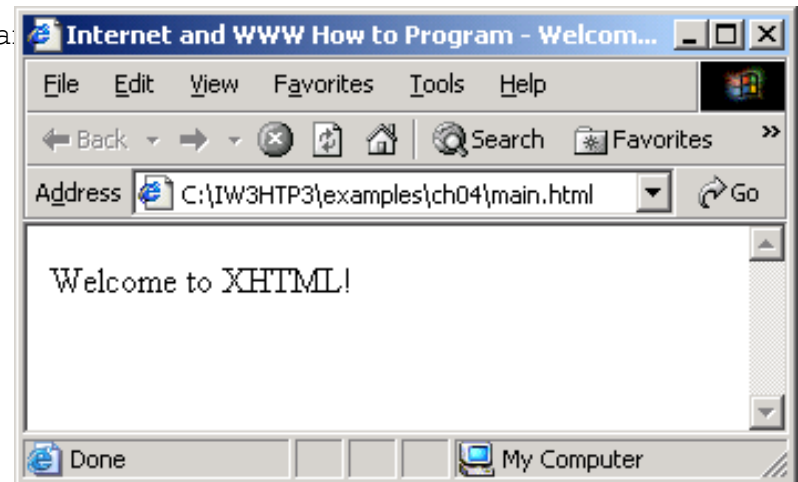
Form of an XHTML Example

- ▶ **xml** declaration element
- ▶ SGML **DOCTYPE** command
- ▶ XHTML comments
 - Start with `<!--` and end with `-->`
- ▶ **html** element
 - **head** element
 - Head section
 - Title of the document
 - Style sheets and scripts
 - **body** element
 - Body section
 - Page's content the browser displays
 - Start tag
 - attributes (provide additional information about an element)
 - name and value (separated by an equal sign)
 - End tag

Basic Text Markup

- ▶ Text is normally placed in paragraph elements
- ▶ *Paragraph Elements*
 - The `<p>` tag breaks the current line and inserts a blank line - the new line gets the beginning of the content of the paragraph
 - The browser puts as many words of the paragraph's content as will fit in each line

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
    http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<!-- greet.html
    A trivial document
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title>Internet and WWW How to Program
  </head>
  <body>
    <p>
      Welcome to XHTML
    </p>
  </body>
</html>
```



W3C XHTML Validation Service

- ▶ Validation service (validator.w3.org)
 - Checking a document's syntax
 - URL that specifies the location of the file
 - Uploading a file to the site

validator.w3.org/file-upload.html

W3C XHTML Validation Service

W3C MarkUp Validation Service: Upload Files - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media

Address <http://validator.w3.org/file-upload.html> Go

W3C[®] MarkUp Validation Service

This form allows you to upload files from your computer and have them validated.

Validate Uploaded File

File:

Encoding:

Type:

Options:

<input type="checkbox"/> Show Source	<input type="checkbox"/> Show Parse Tree
<input type="checkbox"/> Show Outline	<input type="checkbox"/> ...exclude attributes
<input type="checkbox"/> Validate error pages	<input checked="" type="checkbox"/> Verbose Output

If your document is on the Web, you can validate it with the same advanced options by [entering its address](#) instead.

[Home Page](#)
[Documentation](#)
[Source Code](#)
[What's New](#)
[Accesskeys](#)
[Feedback](#)
[About...](#)
[Favelets](#)

[Site Valet](#)
[WDG](#)
[Validator](#)
[CSS Validator](#)
[Link Checker](#)
[HTML Tidy](#)
[Tidy Online](#)

[XHTML 1.1](#)
[XHTML 1.0](#)

Done Internet

W3C XHTML Validation Service

Validation Results - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Print W3C

Address <http://validator.w3.org/check> Go

W3C[®] MarkUp Validation Service

Jump To: [\[Results\]](#)

File: C:\W3HTP3\Examples\ch04\main.html
Content-Type: text/xml
Encoding: us-ascii
Doctype: [XHTML 1.1](#)
Root Namespace: <http://www.w3.org/1999/xhtml>

- Note:* The HTTP Content-Type header sent by your web browser (unknown) did not contain a "charset" parameter, but the Content-Type was one of the XML text* sub-types (text/xml). The relevant specification (RFC 3023) specifies a strong default of "us-ascii" for such documents so we will use this value regardless of any encoding you may have indicated elsewhere. If you would like to use a different encoding, you should arrange to have your browser send this new encoding information.
- Note:* The Validator XML support has [some limitations](#).

This Page Is Valid [XHTML 1.1!](#)

[Home Page](#)
[Documentation](#)
[Source Code](#)
[What's New](#)
[Accesskeys](#)
[Feedback](#)
[About...](#)
[Favelets](#)

[Site Valet](#)
[WDG](#)
[Validator](#)
[CSS Validator](#)
[Link Checker](#)
[HTML Tidy](#)
[Tidy Online](#)

[XHTML 1.1](#)
[XHTML 1.0](#)
[HTML 4.01](#)

[XSL 1.0](#)
[CSS Level 2](#)
[CSS Level 1](#)

Done Internet

Basic Text Markup (continued)

▶ Line breaks

- The effect of the `
` tag is the same as that of `<p>`, except for the blank line
 - No closing tag!

▶ Example of paragraphs and line breaks

```
On the plains of hesitation <p> bleach the  
bones of countless millions </p> <br />  
who, at the dawn of victory <br /> sat down  
to wait, and waiting, died.
```

▶ Typical display of this text:

```
On the plains of hesitation  
  
bleach the bones of countless millions  
who, at the dawn of victory  
sat down to wait, and waiting, died.
```

Headers

▶ *Headings*

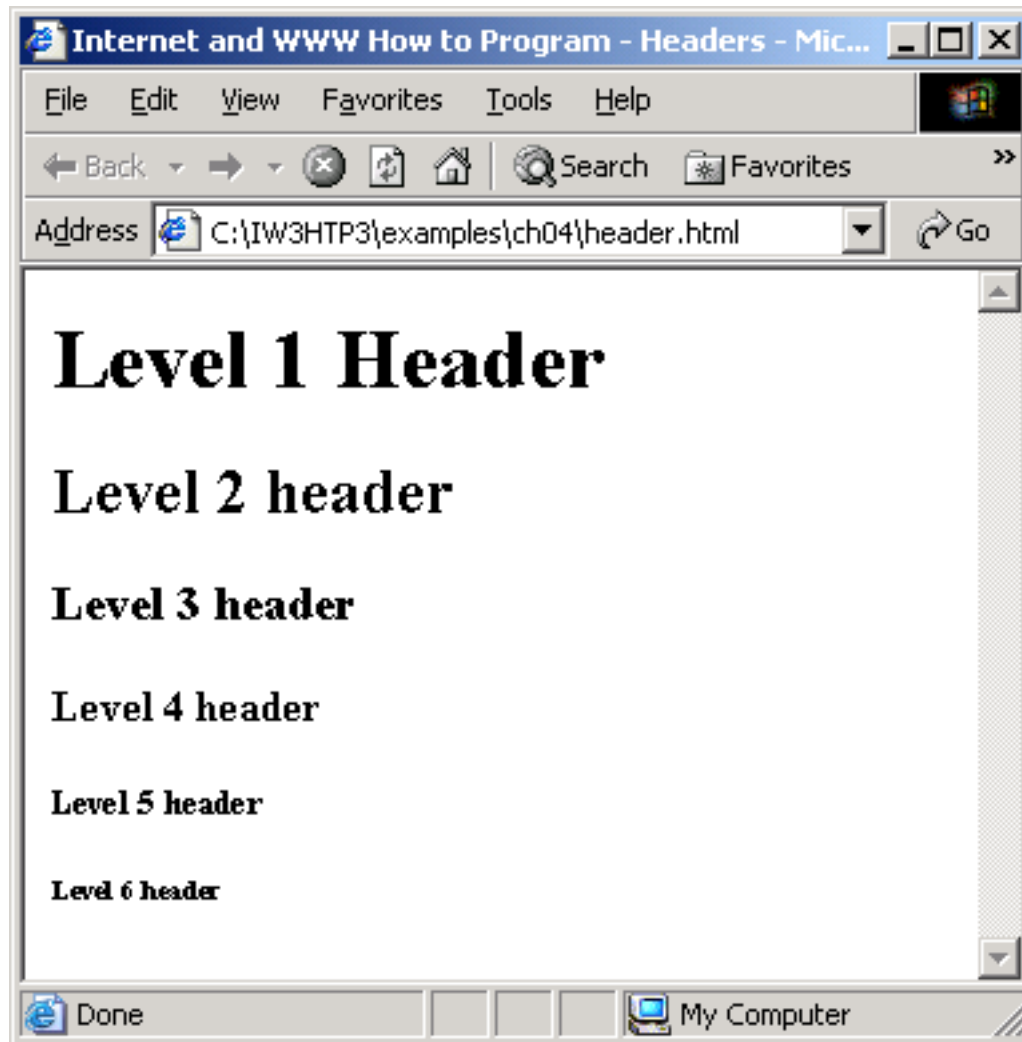
- Six sizes, 1 - 6, specified with `<h1>` to `<h6>`
- 1, 2, and 3 use font sizes that are larger than the default font size
- 4 uses the default size
- 5 and 6 use smaller font sizes

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.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>
```



Outline

er.html
(of 1)



Basic Text Markup (continued)

▶ Blockquotes

- Content of `<blockquote>`
- To set a block of text off from the normal flow and appearance of text
- Browsers often indent, and sometimes italicize

▶ *Font Styles (can be nested)*

- Usually boldface - ``
- Usually Italics - ``

▶ Example

```
The sleet in <strong><em>Crete  
</em><br /> lies completely</strong>  
in the street
```

The sleet in ***Crete***
lies completely in the street

Basic Text Markup (continued)

▶ *Superscripts and subscripts*

- Subscripts with `<sub>`
- Superscripts with `<sup>`

Example: `x₂³`

Display: x_2^3

▶ Inline versus block elements

- Block elements CANNOT be nested in inline elements

Basic Text Markup (continued)

- ▶ All of this font size and font stuff can be done with style sheets, but these tags are not yet deprecated

- ▶ Character Entities

<i>Char.</i>	<i>Entity</i>	<i>Meaning</i>
&	&amp;	Ampersand
<	&lt;	Less than
>	&gt;	Greater than
”	&quot;	Double quote
'	&apos;	Single quote
¼	&frac14;	One quarter
½	&frac12;	One half
¾	&frac34;	Three quarters
°	&deg;	Degree
(space)	&nbsp;	Non-breaking space

- ▶ Horizontal rules

- **<hr />** draws a line across the display, after a line break

- ▶ The **meta** element (for search engines) Used to provide additional information about a document, with attributes, not content

Images

- ▶ GIF (Graphic Interchange Format)
 - 8-bit color (256 different colors)
 - ▶ JPEG (Joint Photographic Experts Group)
 - 24-bit color (16 million different colors)
 - ▶ Both use compression, but JPEG compression is better
 - ▶ Images are inserted into a document with the `` tag with the `src` attribute
 - The `alt` attribute is required by XHTML
 - Purposes:
 1. Non-graphical browsers
 2. Browsers with images turned off
- ```
<img src = "comets.jpg"
 alt = "Picture of comets" />
```
- ▶ The `<img>` tag has 30 different attributes, including `width` and `height` (in pixels)
  - ▶ Portable Network Graphics (PNG)
    - Relatively new
    - Should eventually replace both gif and jpeg

# Images (continued)

```
<!-- image.html
 An example to illustrate an image
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Images </title>
 </head>
 <body>
 <h1> Aidan's Airplanes </h1>
 <h2> The best in used airplanes </h2>
 <h3> "We've got them by the hangarful"
 </h3>
 <h2> Special of the month </h2>
 <p>
 1960 Cessna 210

 577 hours since major engine overhaul

 1022 hours since prop overhaul

 <img src = "c210new.jpg"
 alt = "Picture of a Cessna 210"/>

 Buy this fine airplane today at a
 remarkably low price

 Call 999-555-1111 today!
 </p>
 </body>
</html>
```

## Aidan's Airplanes

### The best in used airplanes

"We've got them by the hangarful"

### Special of the month

1960 Cessna 210  
577 hours since major engine overhaul  
1022 hours since prop overhaul



Buy this fine airplane today at a remarkably low price  
Call 999-555-1111 today!

# Hypertext Links

- Hypertext is the essence of the Web!
- A link is specified with the **href** (*hypertext reference*) attribute of **<a>** (the anchor tag)
  - The content of **<a>** is the visual link in the document
    - Both text and images can be the content of hyperlinks
  - The target is the document specified in the link
- Note: Relative addressing of targets is easier to maintain and more portable than absolute addressing
  - If the target is another document in the same directory, the target is just the document's filename
  - If the target is a document in some other directory, the Unix pathname conventions are used.

# Hypertext Links (continued)

```
<!-- link.html
 An example to illustrate a link
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Links </title>
</head>
<body>
 <h1> Aidan's Airplanes </h1>
 <h2> The best in used airplanes </h2>
 <h3> "We've got them by the hangarful"
</h3>
 <h2> Special of the month </h2>
 <p>
 1960 Cessna 210

 Information on the Cessna 210
 </p>
</body>
</html>
```





```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.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</title>
11 </head>
12
13 <body>
14
15 <p>
16 My email address is
17
18 deitel@deitel.com
19
20 . Click the address and your browser will
21 open an e-mail message and address it to me.
22 </p>
23 </body>
24 </html>
```

# contact.html (1 of 1)



```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.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
18 <img src = "buttons/links.jpg" width = "65"
19 height = "50" alt = "Links Page" />
20

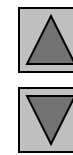
21
22
23 <img src = "buttons/list.jpg" width = "65"
24 height = "50" alt = "List Example Page" />
25

```

# nav.html

## (1 of 2)





```
26
27
28 <img src = "buttons/contact.jpg" width = "65"
29 height = "50" alt = "Contact Page" />
30

31
32
33 <img src = "buttons/header.jpg" width = "65"
34 height = "50" alt = "Header Page" />
35

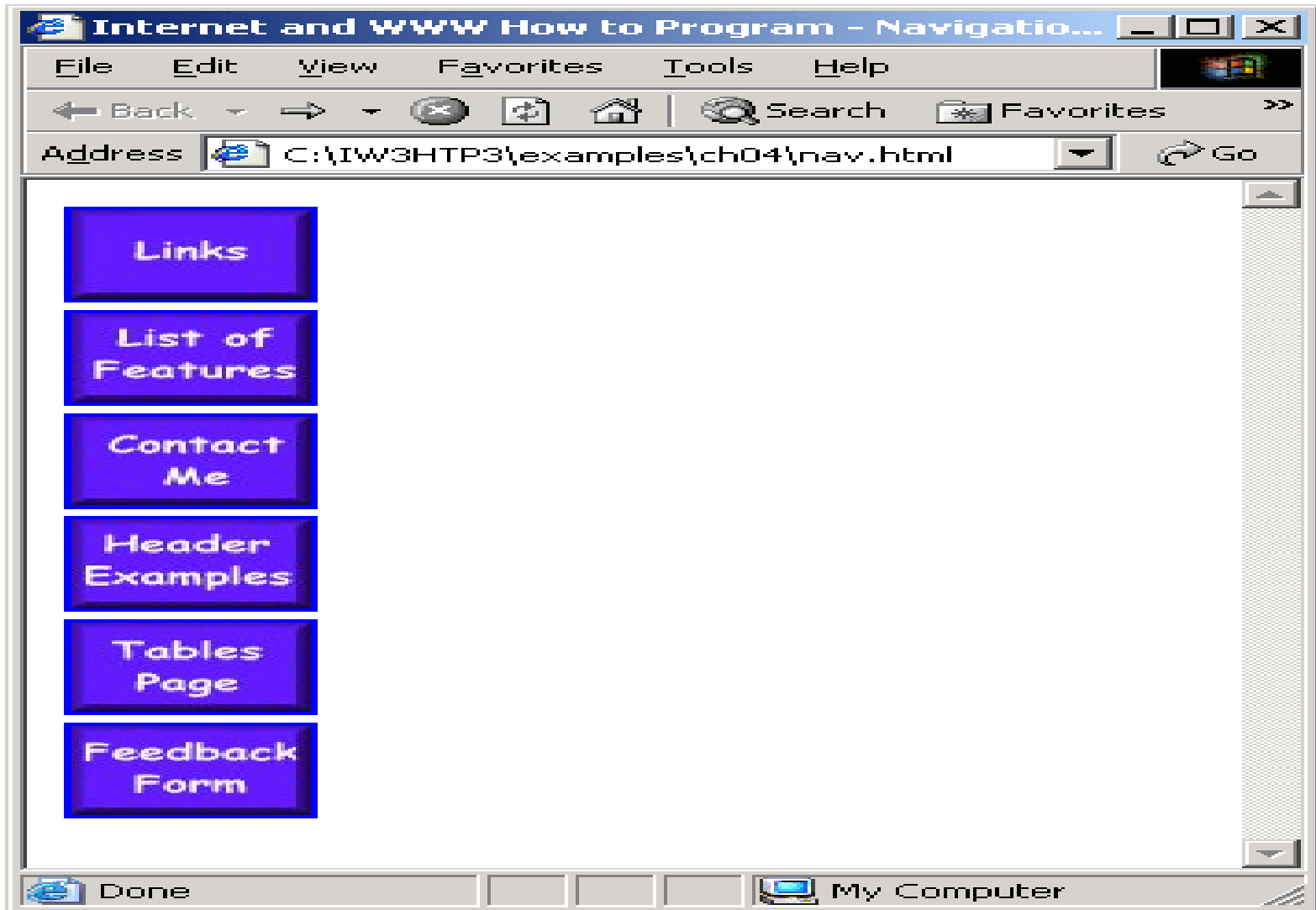
36
37
38 <img src = "buttons/table.jpg" width = "65"
39 height = "50" alt = "Table Page" />
40

41
42
43 <img src = "buttons/form.jpg" width = "65"
44 height = "50" alt = "Feedback Form" />
45

46 </p>
47
48 </body>
49 </html>
```

# nav.html

## (2 of 2)



# Hypertext Links (continued)

- If the target is not at the beginning of the document, the target spot must be marked
- Target labels can be defined in many different tags with the `id` attribute, as in

```
<h1 id = "baskets"> Baskets </h1>
```

- The link to an `id` must be preceded by a pound sign (`#`); If the `id` is in the same document, this target could be

```
 What about baskets?
```

- If the target is in a different document, the document reference must be included

```
 Baskets
```

- Style note: a link should blend in with the surrounding text, so reading it without taking the link should not be made less pleasant

# Lists

- *Unordered lists*
- The list is the content of the `<ul>` tag
- List elements are the content of the `<li>` tag

```
<h3> Some Common Single-Engine Aircraft </h3>

 Cessna Skyhawk
 Beechcraft Bonanza
 Piper Cherokee

```



# Lists (continued)

- *Ordered lists*
  - The list is the content of the `<ol>` tag
  - Each item in the display is preceded by a sequence value

```
<h3> Cessna 210 Engine Starting Instructions
```

```
</h3>
```

```

```

```
 Set mixture to rich
```

```
 Set propeller to high RPM
```

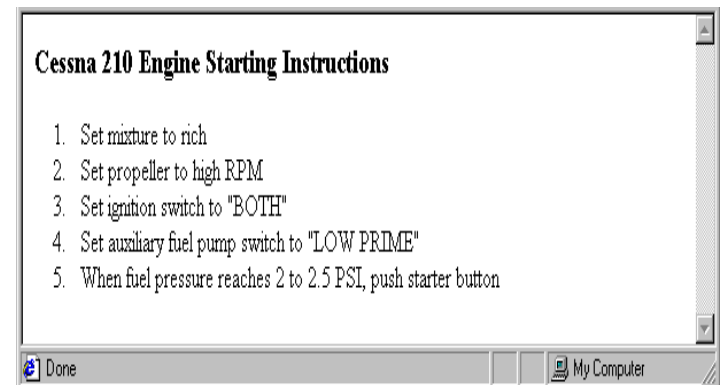
```
 Set ignition switch to "BOTH"
```

```
 Set auxiliary fuel pump switch to
 "LOW PRIME"
```

```
 When fuel pressure reaches 2 to 2.5
 PSI, push starter button
```

```

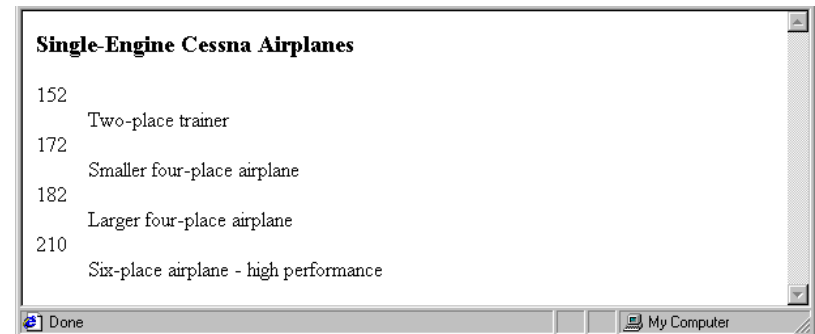
```



# 2.7 Lists (continued)

- ▶ *Definition lists (for glossaries, etc.)*
  - List is the content of the `<dl>` tag
  - Terms being defined are the content of the `<dt>` tag
  - The definitions themselves are the content of the `<dd>` tag

```
<h3> Single-Engine Cessna Airplanes </h3>
<dl >
 <dt> 152 </dt>
 <dd> Two-place trainer </dd>
 <dt> 172 </dt>
 <dd> Smaller four-place airplane </dd>
 <dt> 182 </dt>
 <dd> Larger four-place airplane </dd>
 <dt> 210 </dt>
 <dd> Six-place airplane - high performance
 </dd>
</dl>
```



## •Nested lists

- Any type list can be nested inside any type list
- The nested list must be in a list item



```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.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 <!-- create an unordered list -->
18
19 You can meet new people from countries around
20 the world.
21
22 You have access to new media as it becomes public:
23
```

# list.html (1 of 3)



```
24 <!-- this starts a nested list, which uses a -->
25 <!-- modified bullet. The list ends when you -->
26 <!-- close the tag. -->
27
28 New games
29
30 New applications
31
32 <!-- nested ordered list -->
33
34 For business
35 For pleasure
36
37
38
39 Around the clock news
40 Search engines
41 Shopping
42
43 Programming
44
45 <!-- another nested ordered list -->
46
47 XML
48 Java
```

# list.html

## (2 of 3)

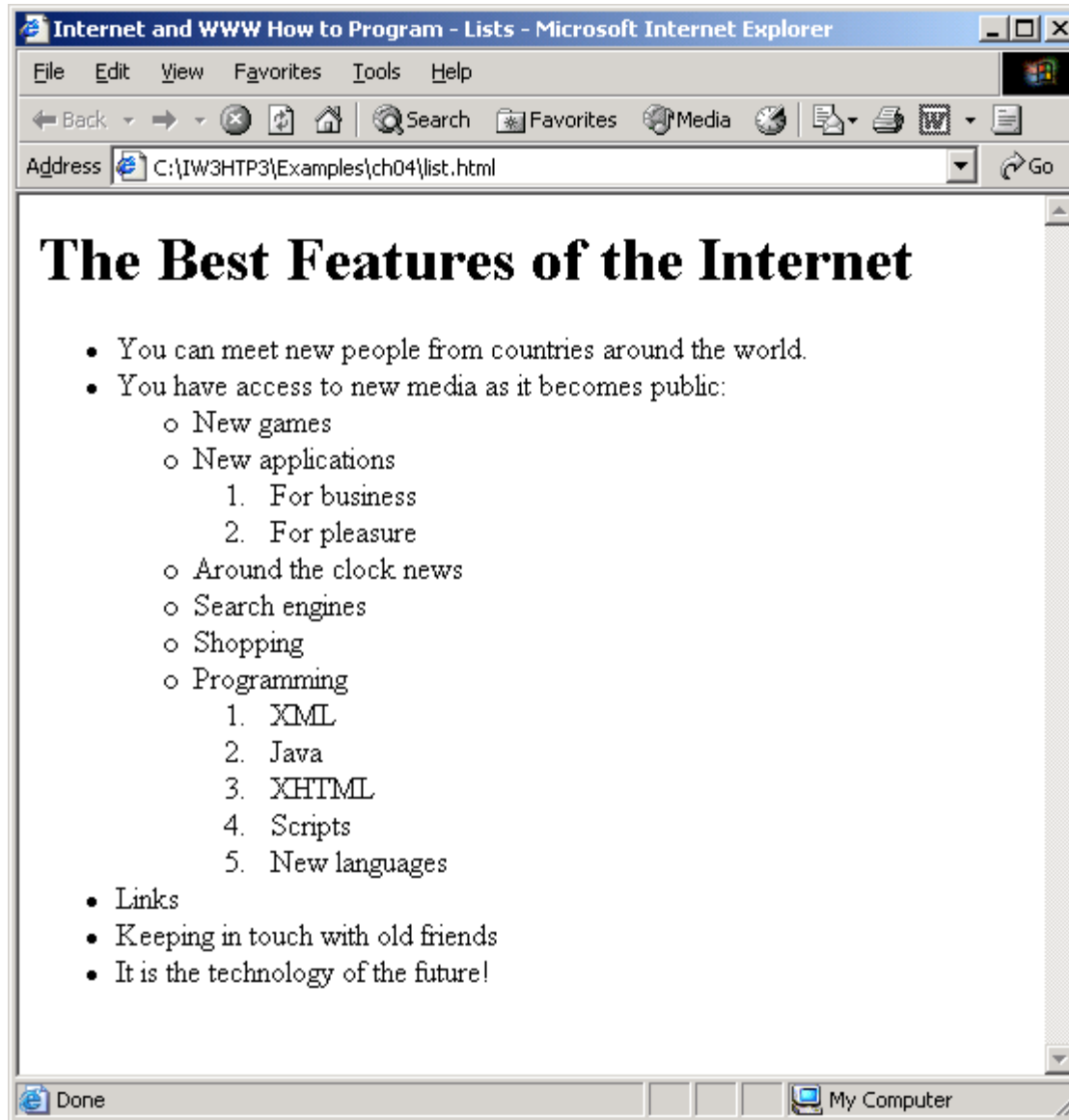




```
49 XHTML
50 Scripts
51 New Languages
52
53
54
55
56 <!-- ends the nested list of line 27 -->
57
58
59 Links
60 Keeping in touch with old friends
61 It is the technology of the future!
62
63 <!-- ends the unordered list of line 18 -->
64
65 </body>
66 </html>
```

# list.html

## (3 of 3)



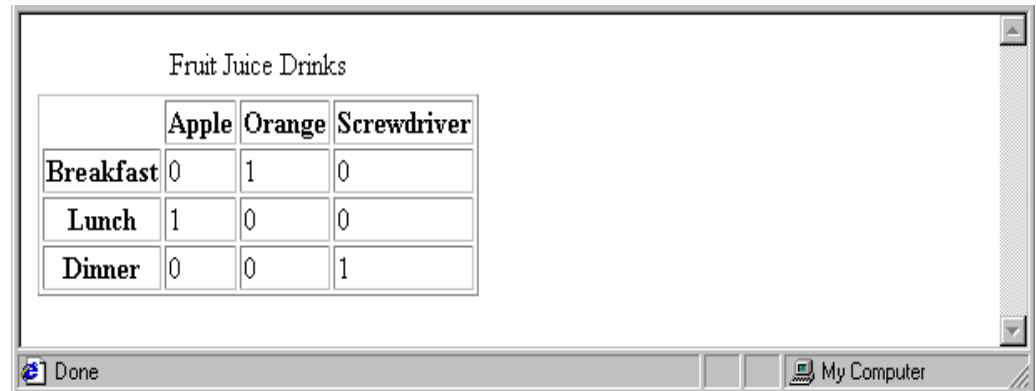
# Tables

- A table is a matrix of cells, each possibly having content
- The cells can include almost any element
- Some cells have row or column labels and some have data
- A table is specified as the content of a `<table>` tag
- A `border` attribute in the `<table>` tag specifies a border between the cells
- If `border` is set to `"border"`, the browser's default width border is used
- The `border` attribute can be set to a number, which will be the border width
- Without the `border` attribute, the table will have no lines!
- Tables are given titles with the `<caption>` tag, which can immediately follow `<table>`

# 2.8 Tables (continued)

- ▶ Each row of a table is specified as the content of a `<tr>` tag
- ▶ The row headings are specified as the content of a `<th>` tag
- ▶ The contents of a data cell is specified as the content of a `<td>` tag

```
<table border = "border">
 <caption> Fruit Juice Drinks </caption>
 <tr>
 <th> </th>
 <th> Apple </th>
 <th> Orange </th>
 <th> Screwdriver </th>
 </tr>
 <tr>
 <th> Breakfast </th>
 <td> 0 </td>
 <td> 1 </td>
 <td> 0 </td>
 </tr>
 <tr>
 <th> Lunch </th>
 <td> 1 </td>
 <td> 0 </td>
 <td> 0 </td>
 </tr>
 <tr>
 <th> Dinner </th>
 <td> 0 </td>
 <td> 0 </td>
 <td> 1 </td>
 </tr>
</table>
```



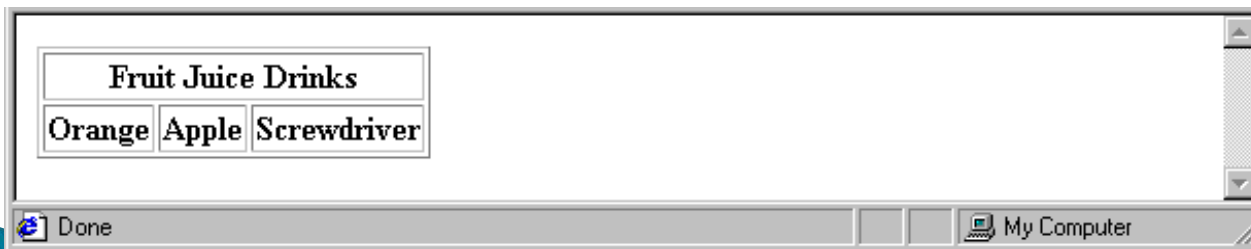
The screenshot shows a web browser window with a table titled "Fruit Juice Drinks". The table has a header row with columns for "Apple", "Orange", and "Screwdriver". Below the header are three rows representing different meals: "Breakfast", "Lunch", and "Dinner". The values in the cells are 0 or 1, indicating the presence of a drink in a meal.

	Apple	Orange	Screwdriver
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

# Tables (continued)

- ▶ A table can have two levels of column labels
  - If so, the **colspan** attribute must be set in the **<th>** tag to specify that the label must span some number of columns

```
<tr>
 <th colspan = "3"> Fruit Juice Drinks </th>
</tr>
<tr>
 <th> Orange </th>
 <th> Apple </th>
 <th> Screwdriver </th>
</tr>
```



# Tables (continued)

- If the rows have labels and there is a spanning column label, the upper left corner must be made larger, using **rowspan**

```
<table border = "border">
 <tr>
 <td rowspan = "2"> </td>
 <th colspan = "3"> Fruit Juice Drinks
 </th>
 </tr>
 <tr>
 <th> Apple </th>
 <th> Orange </th>
 <th> Screwdriver </th>
 </tr>
 ...
</table>
```

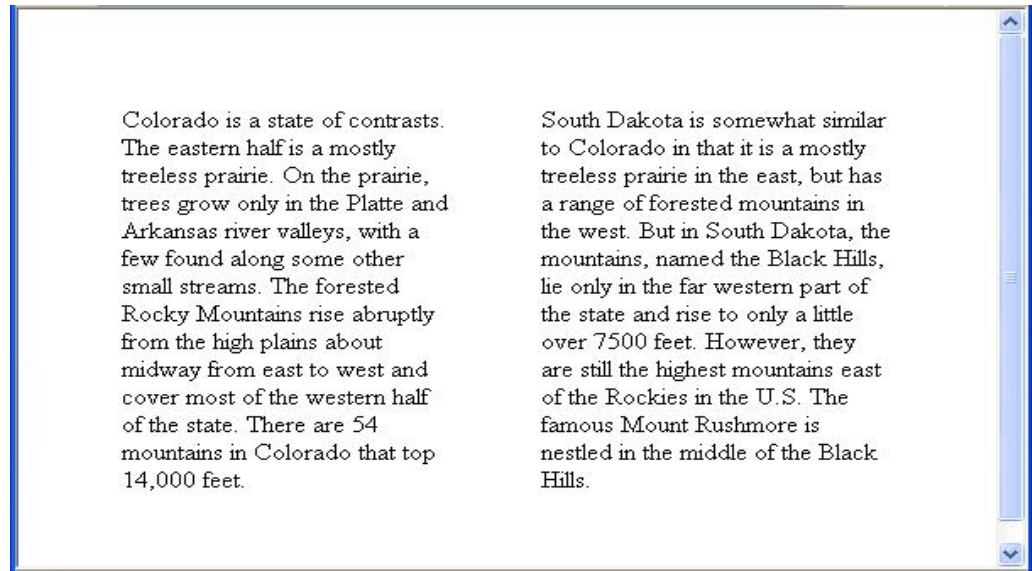
# Tables (continued)

- ▶ The **align** attribute controls the horizontal placement of the contents in a table cell
  - Values are **left**, **right**, and **center** (default)
  - **align** is an attribute of **<tr>**, **<th>**, and **<td>** elements
- ▶ The **valign** attribute controls the vertical placement of the contents of a table cell
  - Values are **top**, **bottom**, and **center** (default)
  - **valign** is an attribute of **<th>** and **<td>** elements

→ SHOW `cell_align.html` and display it
- The **cellspacing** attribute of **<table>** is used to specify the distance between cells in a table
- ▶ The **cellpadding** attribute of **<table>** is used to specify the spacing between the content of a cell and the inner walls of the cell

# Tables (continued)

```
<table cellpadding = "50">
 <tr>
 <td> Colorado is a state of ...
 </td>
 <td> South Dakota is somewhat...
 </td>
</tr>
</table>
```



<p>Colorado is a state of contrasts. The eastern half is a mostly treeless prairie. On the prairie, trees grow only in the Platte and Arkansas river valleys, with a few found along some other small streams. The forested Rocky Mountains rise abruptly from the high plains about midway from east to west and cover most of the western half of the state. There are 54 mountains in Colorado that top 14,000 feet.</p>	<p>South Dakota is somewhat similar to Colorado in that it is a mostly treeless prairie in the east, but has a range of forested mountains in the west. But in South Dakota, the mountains, named the Black Hills, lie only in the far western part of the state and rise to only a little over 7500 feet. However, they are still the highest mountains east of the Rockies in the U.S. The famous Mount Rushmore is nestled in the middle of the Black Hills.</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- *Table Sections*
  - Header, body, and footer, which are the elements: **thead**, **tbody**, and **tfoot**



# Forms

- ▶ A form is the usual way information is gotten from a browser to a server
  - ▶ HTML has tags to create a collection of objects that implement this information gathering
    - The objects are called *widgets* (e.g., radio buttons and checkboxes)
  - ▶ When the Submit button of a form is clicked, the form's values are sent to the server
  - ▶ All of the widgets, or components of a form are defined in the content of a `<form>` tag
    - The only required attribute of `<form>` is `action`, which specifies the URL of the application that is to be called when the Submit button is clicked
- ```
action = "http://www.cs.ucp.edu/cgi-bin/survey.pl"
```
- If the form has no action, the value of `action` is the empty string

Forms (continued)

- ▶ The **method** attribute of **<form>** specifies one of the two possible techniques of transferring the form data to the server, **get** and **post**

- **get** and **post** are discussed in Chapter 10

- ▶ *Widgets*

- Many are created with the **<input>** tag

- The **type** attribute of **<input>** specifies the kind of widget being created

- 1. text**

- Creates a horizontal box for text input
 - Default size is 20; it can be changed with the **size** attribute
 - If more characters are entered than will fit, the box is scrolled (shifted) left

Forms (continued)

- If you don't want to allow the user to type more characters than will fit, set **maxlength**, which causes excess input to be ignored

```
<input type = "text" name = "Phone"  
      size = "12" />
```

2. *Checkboxes* - to collect multiple choice input

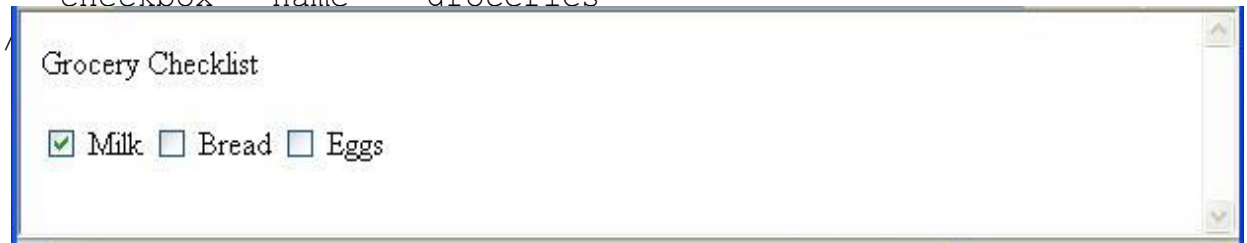
- Every checkbox requires a **value** attribute, which is the widget's value in the form data when the checkbox is 'checked'
 - A checkbox that is not 'checked' contributes no value to the form data
- By default, no checkbox is initially 'checked'
- To initialize a checkbox to 'checked', the **checked** attribute must be set to "**checked**"

Forms (continued)

▶ *Widgets (continued)*

Grocery Checklist

```
<form action = "">
  <p>
    <label><input type = "checkbox" name ="groceries"
      value = "milk" checked = "checked" /> Milk
    </label>
    <label><input type = "checkbox" name ="groceries"
      value = "bread" />Bread
    </label>
    <label><input type = "checkbox" name = "groceries"
      value= "eggs" /
    </label>
  </p>
</form>
```



Grocery Checklist

Milk Bread Eggs

3. *Radio Buttons* - collections of checkboxes in which only one button can be 'checked' at a time

- Every button in a radio button group MUST have the same name

Forms (continued)

▶ *Widgets (continued)*

3. *Radio Buttons (continued)*

- If no button in a radio button group is 'pressed', the browser often 'presses' the first one

Age Category

```
<form action = "">
  <p>
    <label><input type = "radio" name = "age"
      value = "under20" checked = "checked" /> 0-19 </label>
    <label><input type = "radio" name = "age"
      value = "20-35" /> 20-35</label>
    <label><input type = "radio" name = "age"
      value = "36-50" /> 36-50 </label>
    <label><input type = "radio" name = "age"
      value = "over50 /"> Over 50 </label>
  </p>
</form>
```

Forms (continued)



Age Category

0-19 20-35 36-50 Over 50

3. Menus - created with `<select>` tags

- ▶ There are two kinds of menus, those that behave like checkboxes and those that behave like radio buttons (the default)
 - Menus that behave like checkboxes are specified by including the **multiple** attribute, which must be set to "**multiple**"
- ▶ The **name** attribute of `<select>` is required
- ▶ The **size** attribute of `<select>` can be included to specify the number of menu items to be displayed (the default is 1)
 - If **size** is set to **> 1** or if **multiple** is specified, the menu is displayed as a pop-up menu

Forms (continued)

3. Menus (continued)

- Each item of a menu is specified with an `<option>` tag, whose pure text content (no tags) is the value of the item
- An `<option>` tag can include the `selected` attribute, which when assigned `"selected"` specifies that the item is preselected

Grocery Menu - milk, bread, eggs, cheese

```
<form action = "">
```

```
<p>
```

```
<label>With size = 1 (the default)
```

```
<select name = "groceries">
```

```
<option> milk </option>
```

```
<option> bread </option>
```

```
<option> eggs </option>
```

```
<option> cheese </option>
```

```
</select>
```

```
</label>
```

```
</p>
```

```
</form>
```

Forms (continued)

5. Text areas - created with `<textarea>`

- Usually include the `rows` and `cols` attributes to specify the size of the text area
- Default text can be included as the content of `<textarea>`
- Scrolling is implicit if the area is overfilled

Please provide your employment aspirations

```
<form action = "">
```

```
<p>
```

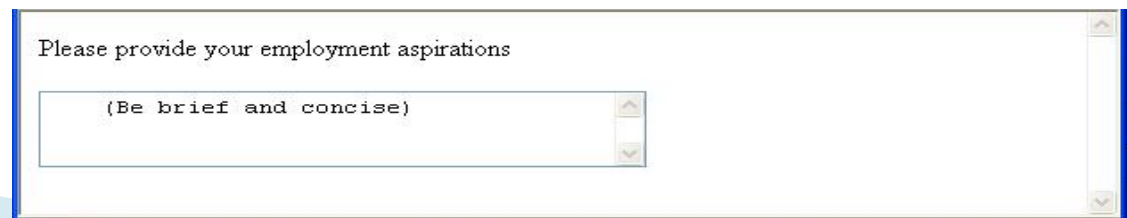
```
  <textarea name = "aspirations"  rows = "3"  
          cols = "40">
```

```
    (Be brief and concise)
```

```
  </textarea>
```

```
</p>
```

```
</form>
```



Please provide your employment aspirations

(Be brief and concise)

Forms (continued)

6. Reset and Submit buttons

- Both are created with `<input>`

```
<input type = "reset" value = "Reset Form" />
```

```
<input type = "submit" value = "Submit Form" />
```

▶ Submit has two actions:

1. Encode the data of the form
 2. Request that the server execute the server-resident program specified as the value of the `action` attribute of `<form>`
- A Submit button is required in every form

--> SHOW `popcorn.html` and display it

XML is not...

- ▶ **A replacement for HTML**
(but HTML can be generated from XML)
- ▶ **A presentation format**
(but XML can be converted into one)
- ▶ **A programming language**
(but it can be used with almost any language)
- ▶ **A network transfer protocol**
(but XML may be transferred over a network)
- ▶ **A database**
(but XML may be stored into a database)

But then – what is it?

**XML is a meta markup
language for text documents
/ textual data**



**XML allows to define
languages („applications“) to
represent text documents /
textual data**

XML by Example

```
<article>  
  <author>Gerhard Weikum</author>  
  <title>The Web in 10 Years</title>  
</article>
```

- **Easy to understand for human users**
- **Very expressive (semantics along with the data)**
- **Well structured, easy to read and write from programs**

This looks nice, but...

XML by Example

... this is XML, too:

```
<t108>  
  <x87>Gerhard Weikum</x87>  
  <g10>The Web in 10 Years</g10>  
</t108>
```

- **Hard** to understand for human users
- **Not** expressive (**no** semantics along with the data)
- Well structured, easy to read and write from programs

XML by Example

... and what about this XML document:

```
<data>
```

```
  ch37fhgks73j5mv9d63h5mgfkds8d984lgnsmcns983
```

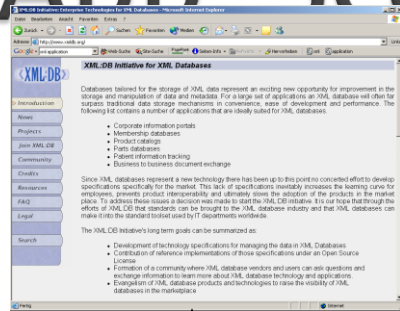
```
</data>
```

- **Impossible** to understand for human users
 - **Not** expressive (**no** semantics along with the data)
 - **Unstructured**, read and write only with **special programs**
- The actual benefit of using XML highly depends on the design of the application.

Possible Advantages of Using XML

- ▶ Truly Portable Data
- ▶ Easily readable by human users
- ▶ Very expressive (semantics near data)
- ▶ Very flexible and customizable (no finite tag set)
- ▶ Easy to use from programs (libs available)
- ▶ Easy to convert into other representations (XML transformation languages)
- ▶ Many additional standards and tools
- ▶ Widely used and supported

App Scenario 1: Content Mgt.



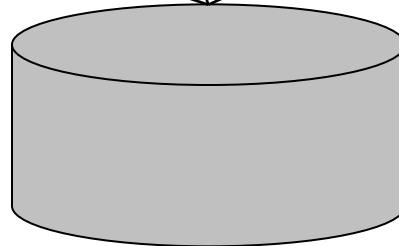
Clients

XML2HTML

XML2WML

XML2PDF

Converters

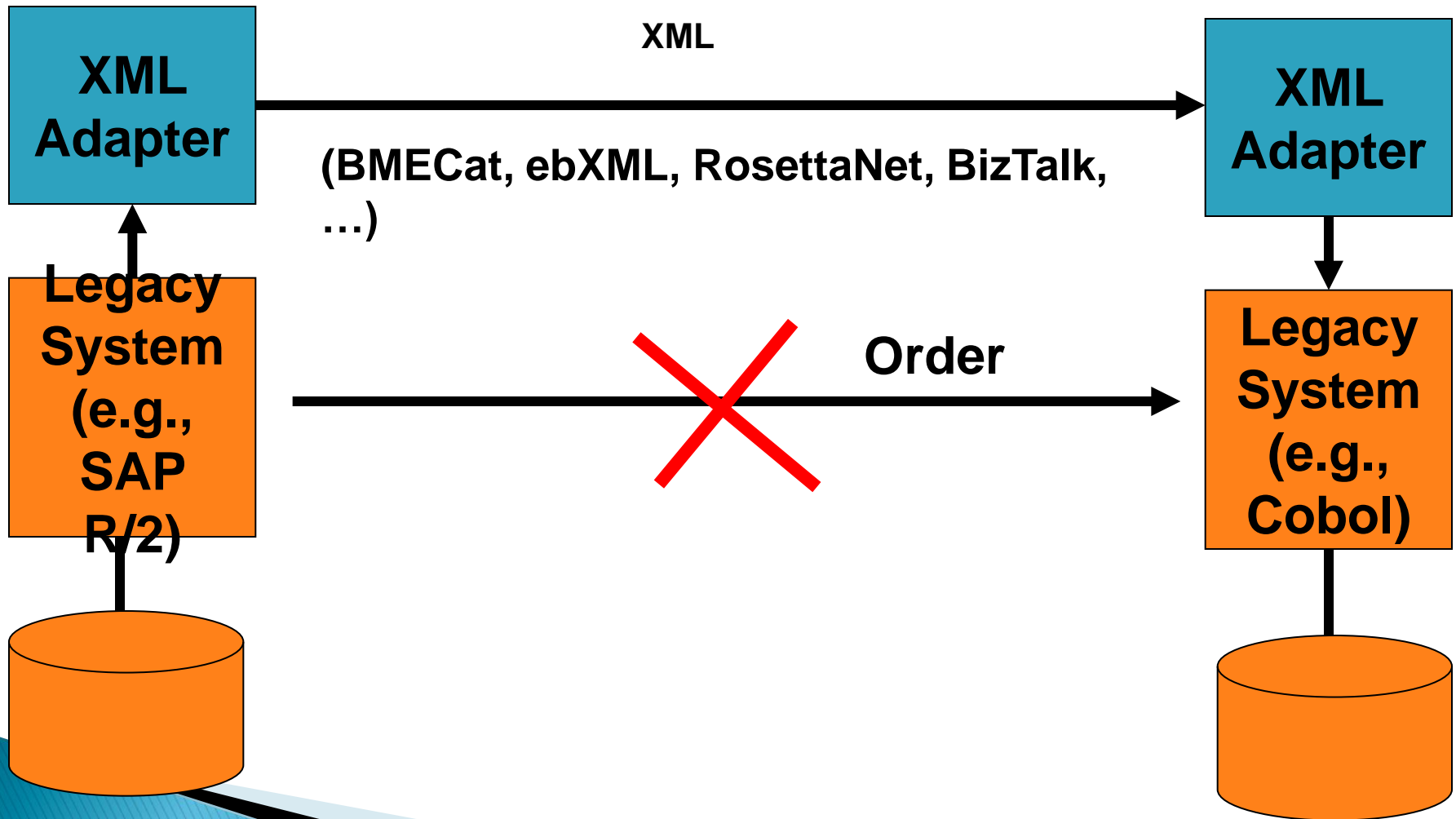


Database with XML documents

App. Scenario 2: Data Exchange

Buyer

Su
er



App. Scenario 3: XML for Metadata

```
<rdf:RDF
  <rdf:Description rdf:about="http://www-dbs/Sch03.pdf">
    <dc:title>A Framework for..</dc:title>
    <dc:creator>Ralf Schenkel</dc:creator>
    <dc:description>While there are...</dc:description>
    <dc:publisher>Saarland University</dc:publisher>
    <dc:subject>XML Indexing</dc:subject>
    <dc:rights>Copyright ...</dc:rights>
    <dc:type>Electronic Document</dc:type>
    <dc:format>text/pdf</dc:format>
    <dc:language>en</dc:language>
  </rdf:Description>
</rdf:RDF>
```



App. Scenario 4: Document Markup

```
<article>
  <section id=„1“ title=„Intro“>
    This article is about <index>XML</index>.
  </section>
  <section id=„2“ title=„Main Results“>
    <name>Weikum</name> <cite idref=„Weik01“/> shows the
    following theorem (see Section <ref idref=„1“/>)
    <theorem id=„theo:1“ source=„Weik01“>
      For any XML document  $x$ , ...
    </theorem>
  </section>
  <literature>
    <cite id=„Weik01“><author>Weikum</author></cite>
  </literature>
</article>
```

A Simple XML Document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

A Simple XML Document

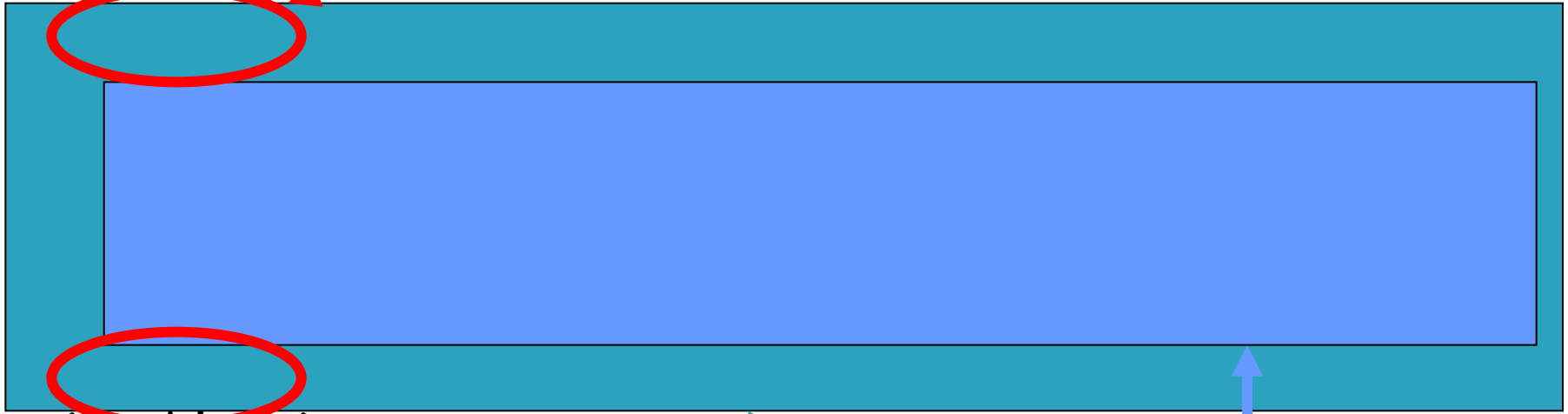
```
<article>  
  <author>Bernard Weikum</author>  
  <title>The Web in Ten Years</title>  
  <text>  
    <abstract>In order to evolve...</abstract>  
    <section number="1" title="Introduction">  
      The <index>Web</index> provides the universal...  
    </section>  
  </text>  
</article>
```

Freely definable tags

A Simple XML Document

```
<article>  
  <author>Gerhard Weikum</author>  
  <title>The Web in Ten Years</title>
```

**Start
Tag**



```
</article>
```

End Tag

Element

**Content of
the Element
(Subelements
and/or
Text)**

A Simple XML Document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```



**Attributes
with name
and value**

Elements in XML Documents

- ▶ (Freely definable) **tags**: `article`, `title`, `author`
 - with start tag: `<article>` etc.
 - and end tag: `</article>` etc.
- ▶ **Elements**: `<article> ... </article>`
- ▶ Elements have a **name** (`article`) and a **content** (...)
- ▶ Elements may be nested.
- ▶ Elements may be empty: `<this_is_empty/>`
- ▶ Element content is typically parsed character data (PCDATA), i.e., strings with special characters, and/or nested elements (*mixed content* if both).
- ▶ Each XML document has exactly one root element and forms a tree.
- ▶ Elements with a common parent are ordered.

Elements vs. Attributes

Elements may have **attributes** (in the start tag) that have a **name** and

a **value**, e.g. `<section number="1">`.

What is the difference between elements and attributes?

- ▶ Only one attribute with a given name per element (but an arbitrary number of subelements)
- ▶ Attributes have no structure, simply strings (while elements can have subelements)

As a *rule of thumb*:

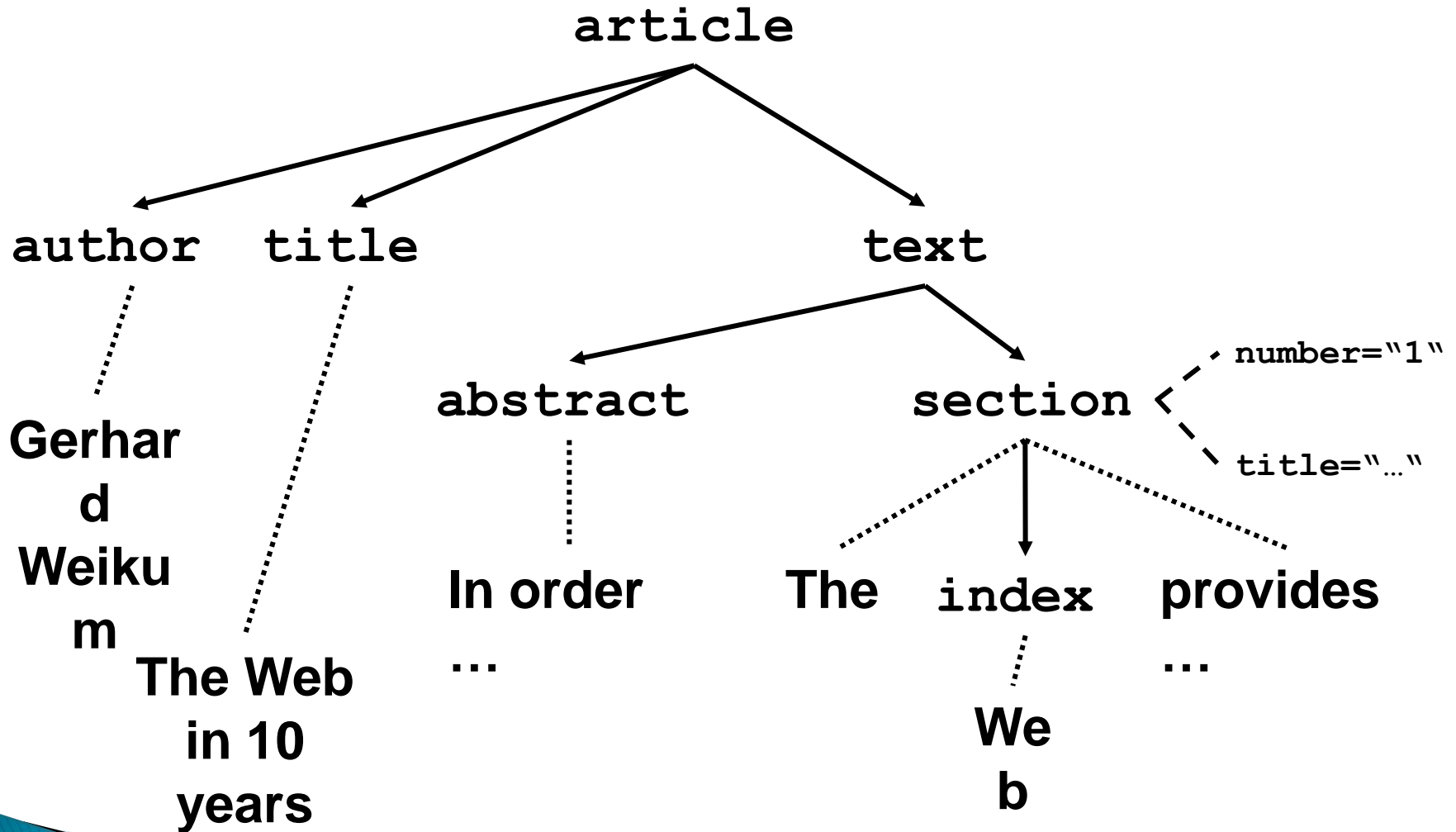
- ▶ Content into elements
- ▶ Metadata into attributes

Example:

```
<person born="1912-06-23" died="1954-06-07">
```

```
Alan Turing</person> proved that...
```

XML Documents as Ordered Trees



More on XML Syntax

- ▶ Some special characters must be escaped using **entities**:

< → **<**

& → **&**

(will be converted back when reading the XML doc)

- ▶ Some other characters may be escaped, too:

> → **>**

“ → **"**

` → **'**

Default Namespace

- ▶ Default namespace may be set for an element and its content (but *not* its attributes):

```
<book xmlns="http://www-dbs/dbs">  
  <description>...</description>  
</book>
```

- ▶ Can be overridden in the elements by specifying the namespace there (using prefix or default namespace)



XML for Beginners

Part 3 – Defining XML Data Formats

3.1 Document Type Definitions

3.2 XML Schema (very short)

4.2 Core Concepts of XQuery

XQuery is an extremely powerful query language for XML data. A query has the form of a so-called FLWR expression:

```
FOR $var1 IN expr1, $var2 IN expr2, ...  
LET $var3 := expr3, $var4 := expr4, ...  
WHERE condition  
RETURN result-doc-construction
```

The FOR clause evaluates expressions (which may be XPath-style path expressions) and binds the resulting elements to variables. For a given binding each variable denotes exactly one element.

The LET clause binds entire sequences of elements to variables.

The WHERE clause evaluates a logical condition with each of the possible variable bindings and selects those bindings that satisfy the condition.

The RETURN clause constructs, from each of the variable bindings, an XML result tree. This may involve grouping and aggregation and even complete subqueries.

XQuery Examples

// find Web-related articles by Dan Suciu from the year 1998

```
<results> {
FOR $a IN document("literature.xml")//article
  FOR $n IN $a//author, $t IN $a//title
  WHERE $a/@year = "1998"
    AND contains($n, "Suciu") AND contains($t, "Web")
  RETURN <result> $n $t </result> } </results>
```

// find articles co-authored by authors who have jointly written a book after 1995

```
<results> {
FOR $a IN document("literature.xml")//article
  FOR $a1 IN $a//author, $a2 IN $a//author
  WHERE SOME $b IN document("literature.xml")//book SATISFIES
    $b//author = $a1 AND $b//author = $a2 AND $b/@year > "1995"
  RETURN <result> $a1 $a2 <wrote> $a </wrote> </result> }
</results>
```