CSE-0613-2209
Web Programming

Md. Masudur Rahman
Lecturer
Department of CSE

Web Programming

Course Code:	CSE-0613-2209	Credits:	02
		CIE Marks:	60
Exam Hours:	03	SEE Marks:	40

Course Learning Outcome (CLOs): After Completing this course successfully, the student will be able to…

CLO	Description
CLO1	Understand the history and progression of the web, including Web 1.0 and Web 2.0.
CLO2	Create and structure web pages using basic HTML tags, attributes, and elements.
CLO3	Develop styled web pages incorporating XHTML, advanced HTML, and CSS for layout and presentation.
CLO4	Explain the roles of client-side coding (HTML, CSS, JavaScript) and server-side coding (PHP, Python).
CLO5	Create forms for data collection and validate them using HTML attributes and JavaScript.
CLO6	Add images, videos, and interactive elements such as animations and maps to web pages.
CLO7	Create and manage links, anchors, and navigation systems for efficient website structure.
CLO8	Use HTML to create structured tables and various list types for organized data presentation.
CLO9	Explain and design URL paths, directory structures, and file organization for websites.
CLO10	Ensure web pages meet usability and accessibility standards through testing and best practices.

Summary of Course Content

Sl.	Course Content	HRs	CLOs
1	History and Evolution of the Internet	2	CLO1
2	Introduction to Web Development and Platforms	2	CLO1, CLO2
3	HTML Basics: Tags, Elements, and Attributes	6	CLO2
4	Advanced HTML and XHTML	4	CLO2, CLO3
5	CSS for Styling and Layout	6	CLO3
6	Client-Side vs. Server-Side Coding	3	CLO4
7	Forms and Form Validation	5	CLO5
8	Multimedia and Interactive Features	4	CLO6
9	Links, Navigation, and Anchors	3	CLO7
10	Tables and Lists for Data Presentation	4	CLO8
11	URL Structure and Directory Organization	2	CLO9
12	Accessibility, Usability, and Testing	3	CLO10
13	Introduction to Frames and Inline Frames	3	CLO2, CLO6
14	Advanced Topics: Image Maps, Nested Frames, and Grouping	4	CLO6, CLO7

Recommended Books

1. **HTML & CSS: Design and Build Websites**
   Author: Jon Duckett
   Description: A beginner-friendly guide to learning HTML and CSS with visual examples.
   Publisher: Wiley

2. **Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics**
   Author: Jennifer Robbins
   Description: A comprehensive introduction to web design principles and coding.
   Publisher: O'Reilly Media

3. **JavaScript and JQuery: Interactive Front-End Web Development**
   Author: Jon Duckett
   Description: A visual guide to building interactive front-end experiences.
   Publisher: Wiley

4. **Web Development and Design Foundations with HTML5**
   Author: Terry Felke-Morris
   Description: Covers the foundations of web development with a focus on HTML5.
   Publisher: Pearson

5. **CSS: The Definitive Guide**
   Author: Eric A. Meyer, Estelle Weyl
   Description: A detailed reference on CSS for all skill levels.
   Publisher: O'Reilly Media

Recommended Web Resources

1. **Mozilla Developer Network (MDN) Web Docs**
   Description: A go-to resource for web development documentation, including HTML, CSS, and JavaScript.

2. **W3Schools**
   Description: Interactive tutorials and references for web technologies, including coding exercises.

3. **GeeksforGeeks – Web Development**
   Description: A collection of tutorials and problem-solving guides for web programming concepts.

4. **CSS-Tricks**
   Description: A resource for CSS tips, tricks, and techniques.

5. **Smashing Magazine**
   Description: Articles and tutorials on web design and development.

6. **TutorialsPoint – Web Development**
   Description: Step-by-step guides and examples for learning web programming basics and advanced concepts.

7. **Stack Overflow**
   Description: A community-driven Q&A site for resolving coding challenges and learning best practices.

8. **Can I Use**
   Description: A compatibility table for checking HTML5, CSS, and JavaScript support in different browsers.

Assessment Pattern

CIE- Continuous Internal Evaluation (60 Marks)

Bloom's Category Marks (out of 60)	Tests (30)	Assignments (10)	Quizzes (10)	Attendance (10)
Remember	5		3	
Understand	5	2	2	
Apply	5	3	5	
Analyze	5			
Evaluate	5			
Create	5	5		

SEE- Semester End Examination (40 Marks)

Bloom's Category	Test
Remember	5
Understand	5
Apply	10
Analyze	5
Evaluate	5
Create	10

Course Plan

Week No	Topics	Teaching Learning Strategy(s)	Assessment Strategy(s)	Alignment to CLO
1	History and Evolution of the Internet	Lecture, Discussion, Videos	Quiz on internet history	CLO1
2	Basics of Web Development and Platforms	Lecture, Hands-on Practice	Classroom Q&A, Assignment on identifying web technologies	CLO1, CLO2
3	HTML Basics: Tags, Elements, and Attributes	Lecture, Coding Practice	Lab Exercise: Create a simple HTML page	CLO2
4	Advanced HTML and XHTML	Lecture, Hands-on Coding	Lab Exercise: Develop a structured HTML page	CLO2, CLO3
5	Introduction to CSS for Styling	Lecture, Demonstration, Coding Practice	Assignment: Style a webpage using CSS	CLO3
6	Client-Side vs. Server-Side Programming	Lecture, Group Discussion	Quiz on the differences and examples of client/server-side	CLO4
7	Forms and Form Validation	Lecture, Hands-on Practice	Lab Exercise: Create and validate a form	CLO5
8	Adding Multimedia and Interactive Features	Lecture, Coding Practice	Project: Add images and videos to an HTML page	CLO6
9	Links, Navigation, and Anchors	Lecture, Hands-on Practice	Lab Exercise: Create a navigation menu using links	CLO7
10	Tables and Lists for Data Presentation	Lecture, Coding Practice	Assignment: Create a table with nested lists	CLO8
11	URL Structure and Directory Organization	Lecture, Demonstration	Quiz on URL components and directory setup	CLO9
12	Testing, Usability, and Accessibility	Lecture, Case Studies	Group Discussion: Evaluate a website for accessibility	CLO10
13	Frames and Inline Frames	Lecture, Hands-on Coding	Lab Exercise: Create a webpage with inline frames	CLO2, CLO6
14	Image Maps and Advanced Navigation Techniques	Lecture, Coding Practice	Lab Exercise: Create an image map	CLO6, CLO7
15	Advanced HTML/CSS Layout Techniques	Lecture, Coding Practice	Assignment: Build a webpage with complex layouts	CLO3
16	Web Project Development and Integration	Lecture, Group Project Work	Final Project: Complete and submit a functional website	CLO1-CLO10
17	Course Review and Final Assessment	Review Sessions, Project Demonstrations	Final Exam, Project Presentation	CLO1-CLO10

Web Programming

<Week 1>

<Slides 8-31>

Md. Masudur Rahman, Dept. Of CSE, UGV

What is a Web Developer?

Agenda

- History of the Internet
- web 2.0
- What is web development today
- Technology part of it
  - Client Side Coding
  - Server Side Coding
- Testing
- Career in Web development

History of the Internet

Beginning

- In 1960s, the U.S. military tried to find a stable and fault-tolerant communication method.

Internet

▫The term “Internet” was coined in the 1980s after the invention of Transmission Control Protocol/ Internet Protocol (TCP/IP).

Interconnected computer network.

HTML

- How to share documentations?
- In 1989, Berners-Lee wrote a proposal - an Internet-based hypertext system.
- He is also the director of W3C (Word-Wide Web Consortium).

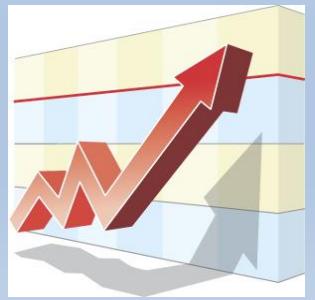
First Web Brower

▫NCSA’s Mosaic came out in 1993.

History of the Internet

- More browsers appeared:

- **dotcom boom**: dotcom companies were getting richer and richer in the stock market by the end of the millenium....



Internet History

- 2000-2002: dotcom bust! ..... ouch!


University of Global Village
DEPARTMENT of
CSE


YouTube


U G V
UNIVERSITY OF GLOBAL VILLAGE


Google
Apps
talk

Web 2.0 (cont'd)

- Before web 2.0, web was a warehouse of static pages
- Platform of collaboration
- Web 1.0?
- Crowdsourcing: Users are co-developers
- World is no longer described by the Web. The Web is becoming the world itself

Web 2.0 (cont'd)

Web 1.0:

- Static websites
- Emails
- Forums
- Basic search engines
- Groups
- Newsletters

Web 2.0

- Blogs
- Social Networking
- Wikis
- Audio/Video Sharing
- RSS
- Webcast/Podcast

What is Web development today

- What platform/devices will your site be accessible on?
- How upgradeable will our web site be?
- How visible will our site be to search engines?
- How much bandwidth will our site be wasting?
- Will our website expose us to the risk of legal action?

Did you know that...

Internet Explorer has lost market from 91.35% to 58.27% while Firefox, and the new Google Chrome now have 22.80% and 9.23% respectively... (1)

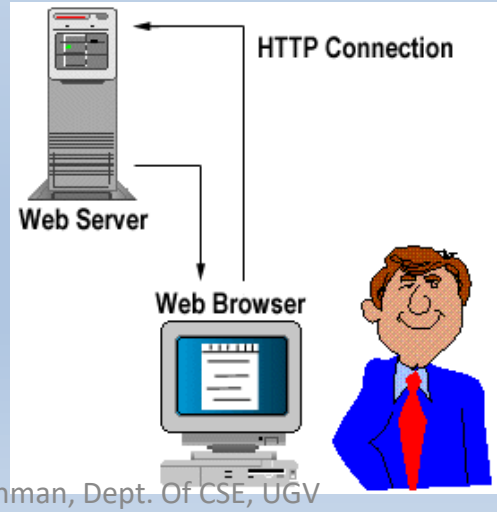
(1) Net Applications statistics 2004 Q4 to 2010 Q4

Server and Client side coding

- Web development comprises of server-side coding and client-side coding

- Server-side coding
  - PHP
  - ASP
  - ASP.NET
  - CGI and/or Perl
  - J2EE
  - Python, e.g. Django
  - Ruby, e.g. Ruby on Rails

- Client-side coding
  - CSS
  - **HTML** & DHTML
  - **XHTML**
  - **Javascript**
  - Flash
  - SilverLight



Testing

- Unit + Integration + System testing
- Security
- Performance, Load/Stress
- HTML/CSS validation
- Usability
- Accessibility

Career in Web development

- Server-side coding (coding)
- UI and UX (design)
- System Administration (networking)
- SEO and SEM (Marketing+analysis)
- Technical Support (Troubleshooting)

Did you know that...

China's population of Internet users jumped by nearly a third to 384 million at the end of 2010...

Introducing HTML, XHTML, CSS

HTML

- HyperText Markup Language.
- The language for describing the structure of Web pages.

XHTML

- Extensible HyperText Markup Language.
- A variant of HTML that uses the syntax of XML, the Extensible Markup Language.
- XHTML has all the same elements (for paragraphs, etc.) as the HTML variant, but the syntax is slightly different

Introducing HTML, XHTML, CSS

Cascading Style Sheets (CSS)

In general, Cascading Style Sheets are combined with HTML to define style implementations such as font size, color, text position, etc.

HTML

HTML Standards:

- **HTML 2.0**: (1994) was developed by the IETF's HTML Working Group, which closed in 1996. It set the standard for core HTML features based upon current practice in 1994.
- **HTML 3.0**: (1996) W3C's first recommendation . Added tables, applets, text-flow around images, superscripts and subscripts. Provides backwards compatibility with HTML 2.0

HTML

- **HTML 4.1**: (1997) W3C's recommendation. Fixes issues found with previous versions.
- **XHTML 1.0:** (2000) reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML.
- **HTML5**: (curr. dev) cross-platform, canvas, geolocation, supports modern web app, etc.

HTML: Basic Tags, Elements and Attributes

HTML Element: represents an html structure and generally consists of an start and end tag.

** **
This is an Element
** **

HTML Tag: Use to mark up the start and end of an element. End tags consist of an opening angle bracket followed by a forward slash, the element name, and a closing angle bracket.

** **

HTML: Basic Tags, Elements and Attributes (con't)

HTML Attribute: defines a property of an element. It is on the form of "attribute=value"

<form name="my_form">

HTML: Basic Tags, Elements and Attributes (con't)

HTML Elements:

- **<html></html>** : defines the whole HTML document.
- **<head></head>**: can include scripts, instruct the browser where to find style sheets, provide meta information, and more.
- **<body><body>**: defines the body of the HTML document
- **<title></title>**: defines the title of the document
- ** **: defines a paragraph
- **<h1></h1>**: defines a heading

An Example... Hands-on!

```
<html>
<head>
<title> My First Web Page </title>
</head>

<body>
<p>  Hello World !!! </p>
</body>

</html>
```

Questions?

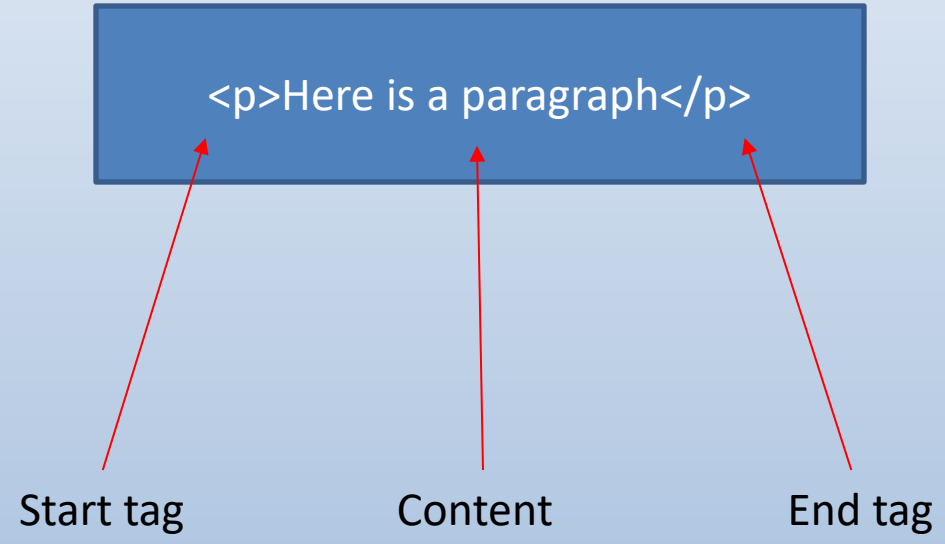
There are 10 types of people in this world: those who understand Binary and those who don't

Web Programming

Week 2
Slides 32-43

Md. Masudur Rahman

HTML Tags and elements



HTML Elements

- **Links: <a>**

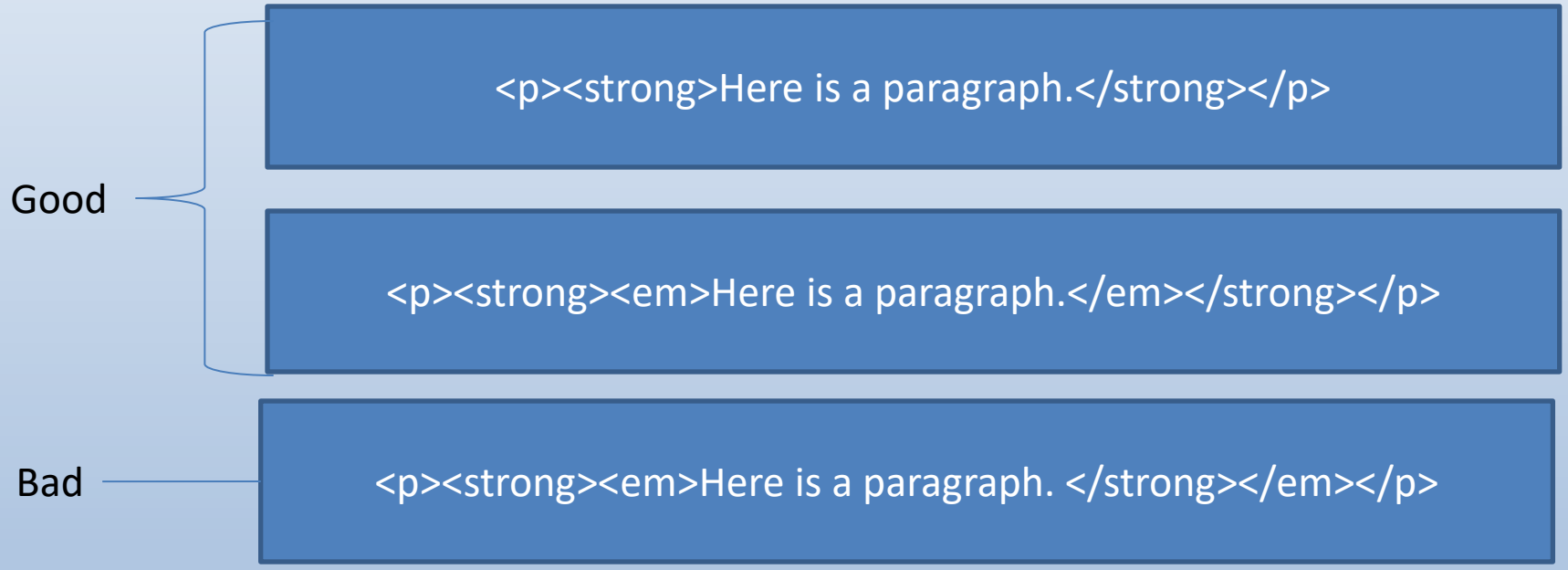
   Search

- **Images**: **<img>**

- **Lines: <hr/> It is used to separate content**

   my paragraph
  <hr/>
   another paragraph

Nesting tags



Empty Elements

HTML Elements with no content. They can be closed in the start tag.

Example:

 : line break

HTML Tips

- Always use lowercase tags: <p> <P>
- Always quote attribute values:

  <form name = “myForm” >
- Use lowercase attributes:

  <img height=“100”>

XHTML

- All tags and attributes must be in **lowercase** and must always be closed.
- XHTML always requires an end tag.
- i.e.: A carriage return
  - HTML:
  - XHTML:
</br>

HTML vs. XHTML

HTML

```
<html>
    <head>
        <title></title>
    </head>
    <body>
    </body>
</html>
```

Use less strict rules.

XHTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.or
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset
        <title></title>
    </head>
    <body>
    </body>
</html>
```

Use more strict rules.

XHTML (outline)

DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title></title>
    </head>
    <body>
    </body>
</html>
```

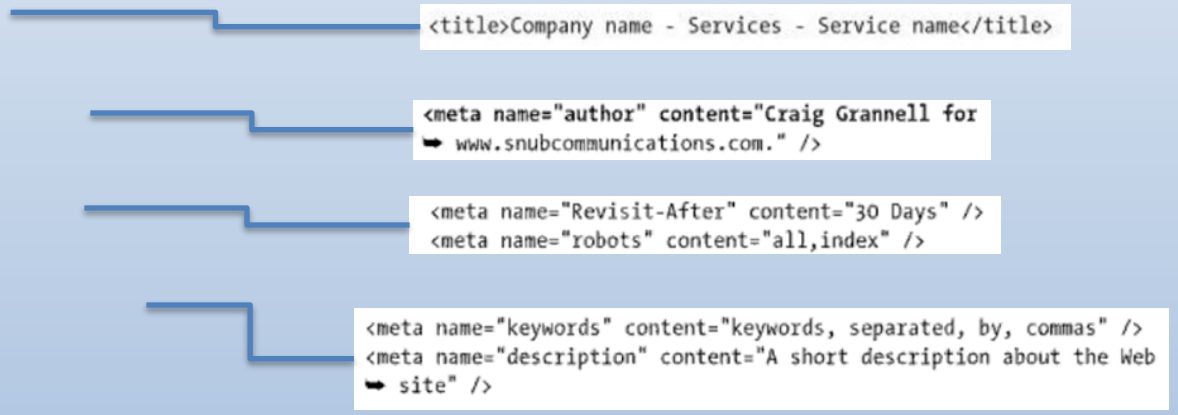
Namespace & Language Declaration

XHTML (language format)

- Character set listings
  - From Part 1 to Part 16
  - Ex: English – Part 1, Hungarian – Part 2, French – Part 15 …
  - Reference
    - http://en.wikipedia.org/wiki/ISO/IEC_8859

Other meta-tags

- Language set
- Title
- Author
- Robots
- Keyword



```
<title>Company name - Services - Service name</title>

<meta name="author" content="Craig Grannell for
➥ www.snubcommunications.com." />

<meta name="Revisit-After" content="30 Days" />
<meta name="robots" content="all,index" />

<meta name="keywords" content="keywords, separated, by, commas" />
<meta name="description" content="A short description about the Web
➥ site" />
```

Questions?

- Click to add title

Web Programming

Week 03
Slides 44-91

Md. Masudur Rahman, Dept. Of CSE, UGV

XHTML Attributes

Core Attributes, International Attributes and UI Events

Md. Masudur Rahman, Dept. Of CSE, UGV

XHTML Attributes

- Lives in the opening tag
  - <a href="…..">
- All attributes consists of a **name** and a **value**
  - **href = "http://csit.udc.edu/~rpalomino"**

Attributes

- There are three attribute groups:
  - **Core attributes**: The class, id and title attributes
  - **Internationalization attributes**: The dir, lang and xml:lang attributes
  - **UI events**: attributes associated with events such as onclick, onmousedown, onmouseup, etc.

- The core attributes and international attributes are known as the **universal attributes**.

Core Attributes

- Four core attributes:
  - **id**
  - **class**
  - **title**
  - **style**

Core Attributes: id

Uniquely identify any element within a page

- Use to link an specific part in the document, specifically associate an element with a CSS or a JavaScript
- i.e.:

```
<p id="incomes"> Showing your Incomes</p>
<p id = "expenses"> Showing your Expenses </p>
```

- It **must**:
  - Start with a letter. You can't start with digit, hyphen, underscore, colon, or period.
  - Be unique within the XHTML document.

Core Attributes: class

Specify that the element belongs to a class of element

- Commonly used with CSS

  class = "className"

- The value can be space-separated list of class names:

  class = "*className1 className2 className3*"

Core Attributes: title

Gives a suggested title for the element.

- title = “string”
- The behavior of title depends upon the element that carries it.
- It is often displayed as a tooltip, or while the element is loading.

Core Attributes: style (deprecated)

Allows you to specify CSS rules within the element.

- SomeText

Internationalization

- There are three:
  - dir: direction in which text flow
    - dir = ltr: left to right; rtl: right to left (Hebrew, Arabic)
  - lang: indicate the main language used in a document.
    - backward compatibility. Replaced by xml:lang.
  - xml:lang: replacement for lang attribute.
    - Use by search engines: by telling user what language the document is written in
    - Use by applications: that can alert that user that they do not support that language, or just that it is a different language that their default language.

UI Events

- Allows you to associate an event with a script.
  - Event: key press, mouse move over an element
  - Script: piece of programming code

  - Example:
    - When someone moves a mouse over the content of a certain element, the element change color.

UI Events

- There are ten events known as ***Common Events***:
  - onclick
  - ondoubleclick
  - onmousedown
  - onmouseup
  - onmouseover
  - onmousemove
  - onmouseout
  - onkeypress
  - onkeydown
  - onkeyup

UI Events

- The <body> and <frameset> also have:
  - onload: when a page opens
  - onunload: when a page is closed

- For forms only:
  - onfocus
  - onblur
  - onsubmit
  - onreset
  - onselect
  - onchange

XHTML Elements

XHTML Elements

- White Space and Flow:
  - The browser treats **multiple spaces** and **several carriage returns** as if they were only **one single space**
    - ** This is a text with too many white**

**spaces and carriage returns **

Headings

- It offers six level of headings:
  - **<h1>**
  - **<h2>**
  - **<h3>**
  - **<h4>**
  - **<h5>**
  - **<h6>**

Align attribute (deprecated)

- It indicates whether the heading appears to the **left**, **center** or **right** of the browser window (or other containing element if nested).

  <h2 **align**="**center**"> My Center Heading </h2>

- It is deprecated because now this is done by CSS

The <pre> element

- The text between the <pre></pre> tags will preserve the formatting of the source document.
- It will use mono-spaced font by default.
- Use to:
  - Show tabular data without a table
  - Show computer source code
- i.e.:

```
<pre>
function testFunction(strText){
        alert(strText)
}
</pre>
```

XHTML Elements

Presentational Elements

The element

- Any text will appear displayed in **bold**

This is a bold word.

The element

- Any text will appear displayed in *italic*

This is an italic word.

The <u> element (deprecated)

- Any text will appear displayed <u>underlined</u>

  This is an <u>underlined</u> word.

- Deprecated by better method in CSS

The <s> or <strike> element (deprecated)

- Any text will appear displayed with a thin line through the text.

  This is a <s>strikethrough</s> word.

- Deprecated by better method in CSS

More Elements

- <tt> Content is in Monospaced font
- <sup> Content is a superscript
- <sub> Content is a subscript
- <big> Content is displayed one font-size larger that surrounding text.
- <small> Content is displayed one font-size smaller that surrounding text.

XHTML Elements

Phrase Elements

Phrase Elements

- ,
  - Indicates emphasis and strong emphasis
- <blockquote>, <cite>, <q>
  - For quotations and citations.
- <abbr>, <acronym>, <dfn>
  - For abbreviations, acronyms, and key terms
- <code>, <kbd>, <var>, <samp>
  - Computer code and information
- <address>
  - For addresses.

Phrase Elements

Why should I use phrase elements, if they visually offer the same as the presentation elements?

- For some reasons such as:
  - Applications such as screen readers detect text marked with emphasis and strong emphasis to make different intonation while reading to visual impairment users.
  - Programs can read a document and pull the key terms to index your document, so a user can find important terms within it.

XHTML Elements

Lists

Lists

- There are three types of lists:
  - **Unordered** lists
  - **Ordered** lists
  - **Definition** lists

Lists: Unordered Lists

- Use the <ul> element.
- It creates a list of bullet points
- Each line should be between the / tags

```
<ul>
<li> Apple </li>
<li> Google </li>
<li> Microsoft </li>
</ul>
```

Lists: Ordered Lists

- Uses the <ol> element
- It can be used numbers (1,2,3), letters (A,B,C), or Roman numerals (i, ii, iii)
- Values for ***type*** attribute:

  1 = Arabic Numerals (1,2,3)

  A = Capital letters (A, B, C)

  a = lowercase letters (a, b, c)

  I = Large Roman num. (I, II, III, IV)

  i = Small Roman num. (I, ii, iii, iv)

Lists: Ordered Lists

start attribute: (deprecated)

Changes the starting number in ordered lists

Lists: Ordered Lists

```
<ol type="1" start="5">
<li> Apple </li>
<li> Google </li>
<li> Microsoft </li>
</ol>
```

Lists: Definition Lists <dl>

- Special list for providing terms followed by a short text definition or description for them.
- They are contained inside the element <dl>
- It contains two inner elements:
  - <dt> term you will define
  - <dd> description/definition of term

Lists: Definition Lists <dl>

```
<h1>Definition Lists</h1>
<dl>
   <dt> Apple </dt>
   <dd> www.apple.com </dd>
   <dt> Google </dt>
   <dd> www.google.com </dd>
   <dt> Microsoft </dt>
   <dd> www.microsoft.com </dd>
</dl>
```

Lists: Nesting Lists

- Lists can be nested inside other lists.

```
<ol type="I">
    <li>Item one</li>
    <li>Item two</li>
    <li>Item three</li>
    <li>Item four
        <ol type="i">
            <li>Item 4.1</li>
            <li>Item 4.2</li>
            <li>Item 4.3</li>
        </ol>
    </li>
    <li>Item Five</li>
</ol>
```

XHTML Elements

Editing Text, Special characters, Comments

Md. Masudur Rahman, Dept. Of CSE, UGV

The <ins> and <del> elements

- This elements helps when revising and editing documents
- It is helpful to keep track of changes you make.

<ins> when you want to add text

<del> when you want to delete text

The <ins> and <del> elements

i.e.:

<h1> Google announced new technology </h1>

 Google announced a new "MentalPlex" search technology that supposedly ~~read~~<ins>feel</ins> the user's ~~mind~~<ins>heart</ins> to determine what the user wanted to search for, thus eliminating the step of actually typing in the search query.

Special Characters

- There are some special characters that can be represented directly by your browser such as:
  - “
  - &
  - <
  - >
- To represent them, we use a set of characters called **character entity** or **escape characters**.

Special Characters

Special Character	Numeric Entity	Named Entity
“	"	"
&	&	&
<	<	<
>	>	>

Comments

- It is a good practice to comment your code because this will help other people to understand or have a better idea of what you are trying to do.

- It can even help you!

- Use the following sintax:

  < ! - - my comment goes here… - - >

XHTML Elements

Block and Inline Elements. Grouping elements.

Block and Inline Elements

- Block-level elements:
  - Block-level elements appear on the screen as if they have a carriage return or line break before and after them.
  - <p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <ul>, <ol>, <dl>, <pre>, <hr />
- Inline elements:
  - On the other hand, inline elements can appear within sentences and **do not have** to appear on a new line of their own.
  - <b>, <i>, <u>, , , <sup>, <sub>, <big>, <small>,

Grouping with and

- They allow you to group several elements to create new sections or subsections within your page.
- On their own, they don't affect the appearance of a page.
- They are commonly combined with CSS to apply style only to certain sections of a page

The element

- Allows to group **block-level** elements

```
<div class="footnotes">
  <h2>Footnotes</h2>
  <p><b>1</b> Facebook is the biggest social network
    nowadays</p>
  <p><b>2</b> Google is the pioneer in HTML5</p>
</div>
```

The element

- Allows to group inline elements only.

```
<div class=”footnotes”>
  <h2>Footnotes</h2>
  <p><span class=“social-network”><b>1</b> Facebook is
    the biggest social network nowadays</span></p>
  <p><b>2</b> Google is the pioneer in HTML5</p>
</div>
```

Questions?

- Before the exercise

Web Programming

Week 04
Slides 92-126

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

- Links and Navigation:
  - Basic Links
  - E-mail links,
  - Directory Structure and URLs
- Exercise

Links and Navigations

Basic Links, and E-mail links

Md. Masudur Rahman, Dept. Of CSE, UGV

Basic Links

- A link is specified with the element <a>
- How to link?
  - You should use the **href** property
  - It indicates the page you want to link
  - Use a short-descriptive word for the link name
    - Try not to use words such as "Click here"

Basic Links (cont'd)

- To link to a different site:
  - Specify a full URL (Uniform Resource Locator) for the page to be linked. i.e.:
    - www.google.com
    - Full URL: **href="http://www.google.com"**
- To link to within the same site:
  - Use a URL shorthand form: *Relative URL*
    - **href="index.html"**

Basic Links: title property (cont'd)

- Use the ***title*** property as a good practice
- It will be displayed as a tooltip (small yellow textbox) when user hovers over the link.
- In case of visually impaired, the title is read aloud.
  - i.e.:

```
<a href="http://www.udc.edu" title="University of the District
   of Columbia"> UDC </a>
```

Basic Links: E-mail

- To let users send e-mails to a determined address by opening their default e-mail program, the keyword **mailto** is used.
- How?
  - e-mail us
  - support@test.com

Basic Links: E-mail (cont'd)

- Drawbacks:
  - Automated programs from unscrupulous people crawl the web searching for e-mail addresses.
- What do we get after that?
  - Junk mail!
  - Spam mails !
  - Scam !
- Solutions?
  - Use e-mail forms
  - Generate e-mail using JavaScript

Links and Navigations

Directory Structure

Directories and Directory Structure

- **Directory**: name for a folder on a web site
  - Hard drive contains folders
  - Website contains **directories**

- It is important to organize correctly and efficiently your files within your website directory

Directory Example for a News Site

- **cnm.com**
  - business
  - education
  - entertainment
    - arts
    - film
    - music
      - features
      - mp3s
      - reviews
    - tv
  - health
  - img
  - scripts
  - css

Terms for describing a Web directories

- **Root folder**: main directory that holds your whole web site.
- **Subdirectory**: a directory that is within another directory.
- **Parent directory:** a directory that contains another directory.

Directory Example for a News Site

- **cnm.com** ← root folder
  - business
  - education
  - entertainment ← parent folder
    - arts
    - film
    - music ← subdirectory
      - features
      - mp3s
      - reviews
    - tv
    - health
    - **img**
    - **scripts**
    - **css**

Anatomy of a URL (Uniform Resource Locator)

- Used to locate a resource on the Internet

URL: The **Scheme**

- Identifies the **type of a URL**

- It indicates how the resource should be retrieved.
  - i.e.:

    Hypertext Transfer Protocol or HTTP is for displaying web pages. It starts with http://

URL: The **Scheme**

Scheme	Description	Use
http://	Hypertext Transfer Protocol	Requests pages from web servers and send them back to browsers
https://	Secure Hypertext Transfer Protocol	Encrypts data sent between the browser and the web server using a digital certificate.
ftp://	File Transfer Protocol	Transfers specially large files across the Web and to upload files to your server.
file://	File	Indicates that a file is on the local hard disk or shared directory on a LAN

URL: The **Host Address**

- Address where a web site can be found

- It can be either:
  - an IP address: four sets of numbers between 0 and 255. (IPv4)
    - i.e.: http://38.105.74.129
  - a domain name: behind the scenes all domain names are converted to IP addresses
    - i.e.: http://www.udc.edu

URL: The **Filepath**

- Always begin with a forward slash (/)
- It may consist of one or more directory names
- Each directory is separated by a forward slash
- It might end with a filename
  - i.e.: http://www.udc.edu**/academics/soe/**
  - ie.: http://www.udc.edu**/academics/soe/dean.html**

URL: Other parts

- **Credentials**: it is a way of specifying user name and password
  - http://**username:password**@www.mysite.com/admin

- **Ports:** a web server has many different programs. Each program communicates to the world through ports that are numbers after the host address.
  - i.e.: http://localhost**:21**

URL: Other parts

- **Fragment Identifier:** Used after a filename to indicate a specific part of the page that a browser should go to immediately.
  - They are identified by a pound or hash sign (#)
    - i.e.: http://www.abc.com/video/index.html**#bottom**

URL: Other parts

- **Path Argument**: pass extra information to a server program.
  - They are separated by a question mark (?) and come in (name=value) pairs separated by an & sign.
  - i.e.:

    www.myserver.com/index.php**?studentId=293&grade=80**

Absolute and Relative URLs

- Every file on the Internet has a unique URL
- No two files on the Internet share the same URL
- **Absolute URL**: contains a full URL including scheme, and host address.
  - http://www.udc.edu/academics/soe/dean.html

- **Relative URL**: indicates where the resource is in relation to the current page.

Relative URLs

- If you are working in your page:
  - www.cnm.com/index.html

- And you need to add links to your other subsections: business, education, entertainment, …
  1. http://www.cnm.com/business/index.html
  2. http://www.cnm.com/education/other.html
  3. http://www.cnm.com/entertainment/film/another.html

  1. business/index.html
  2. education/other.html
  3. entertainment/film/another.html

Relative URLs

Relative URLs

www.cnm.com

- business
  - index.html
  - contact.html
  - nyse
    - money.html

- **Same directory**: you are editing index.html
  - "contact.html"

- **Subdirectory**: you are editing index.html
  - "nyse/money.html"

Relative URLs

Parent directory: add **../** for each level up

- www.cnm.com
  - business
    - money.html
  - entertainment
    - film
      - index.html
    - tv
      - another.html

"**../**tv/another.html"

"**../../**business/money.html"

Default files

- There are certain files that are open automatically by web servers.
- That depends on the customization of the web server.
- Some common default files are:
  - index.html
  - index.php
  - index.asp
- What is the default file for www.google.com ?

The <base> element

- Allows to specify a **base URL** for a page that all relative URLs will be added to
  - <base href="http://www.washingtonport.com/" />
- We have the following relative URL :
  - business/index.html
- It will be interpreted by the browser as:
  - http://www.washingtonport.com/business/index.html

The <base> element

- Allows to specify a **base URL** for a page that all relative URLs will be added to
  - <base href="http://www.washingtonport.com/" />
- We have the following relative URL :
  - business/index.html
- It will be interpreted by the browser as:
  - http://www.washingtonport.com/business/index.html

The <a> element

A deeper look at links

The <a> element: Anchors

- **Source anchor**: it is a simple link on a web page

- **Destination anchor**: allows the page author to mark specific points in a page that a source link can point to.
  - i.e.:
    - Back to top
    - List or Table of contents
    - Links to footnotes or definitions

  - But how do we use it?

The <a> element: Anchors

- Imagine you have a very long page with a main header and then different sections each with a subheading
- First, you need to create the destination anchors by using the **id** attribute.
  - <h1 **id="main-heading"**> Main Heading<h1>
  - .....
  - <a **id="lectures"** href="wd/lectures">Lectures
- Then, create the source anchors:
  - .....
  - Go to main heading
  - ...
  - Go to lectures

The <a> element: Anchors

- Destination anchors **always** should have a content.
  - At the Top of the World
- If your destination anchor wants to be accessed from a different website:
  - http://www.foreign-site.com/e-book.html#chapter2
- Anchors are **case sensitive.** Source and destination names should match exactly.

The <a> element: Other attributes

- **accesskey**: Provides a keyboard shortcut (holding CTRL or ALT) that can use to activate a link. It should be specified in the source anchor.
  - Back to top

- **coords**: use with images. It creates an image map, so different areas of the image can map to different resources (documents, images, anchors, etc).
  - *We'll see this later in the images lecture*

The <a> element: Other attributes

- **title:**
  - important for links that are images.
  - Provide information as a text tooltip

Smile

Questions?

- Before the exercise

Web Programming

Week 05
Slides 127-169

Outline

- Images and Objects:
  - Adding images to your site
  - Adding Objects with <object>
  - Using Images as Links
  - Image Maps
- Exercise

Images and Objects

Adding images to your website

Adding images to your site

- There are different types of images you can use such as:
  - GIFs
  - JPEG
  - PNGs
- You will learn when you should choose which format
- You will have to prepare your images
- When testing your sites, you might not realize how long it really takes to load your pages until it is on the Web.

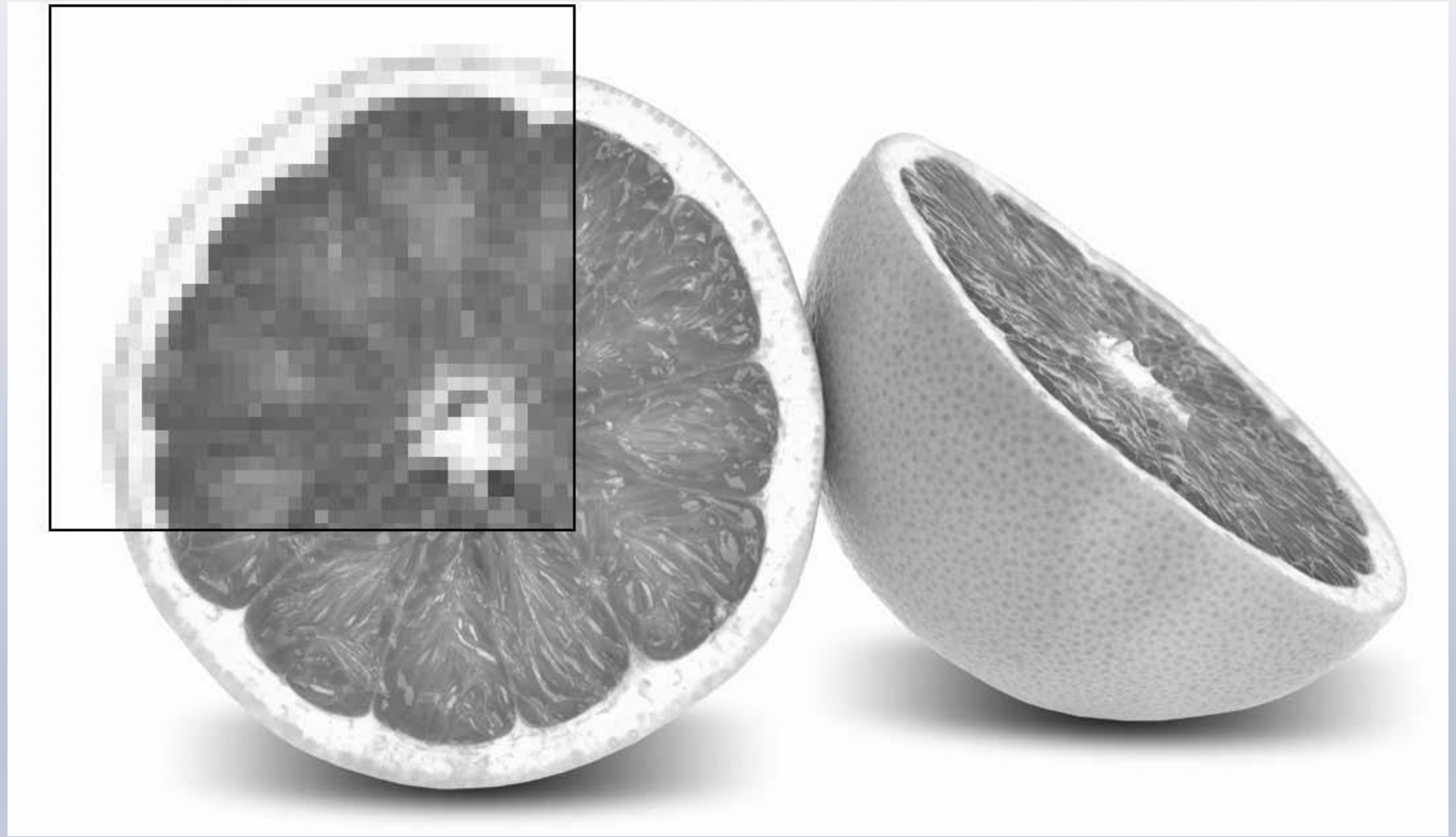
Types of Image Formats

- Graphics are created in two main ways:
  - Bitmapped Graphics
  - Vector Graphics

Bitmapped Graphics

- Divide a picture into a grid of pixels, and specify the color of each pixel.
- **Ideal for**:
  - Photographs
  - Complicated gradations of shade and color
- **Bitmap formats**:
  - JPEG
  - GIF
  - TIFF
  - PNG
  - BMP

Bitmap Graphics



Bitmapped Graphics

- Most static images on the Web are bitmapped images

- The number of pixels in every square inch of the screen is known as the **resolution of the image**.

Bitmapped Graphics

- The maximum resolution for pictures on the Web is 72 pixels/inch
- The more resolution an image has, the larger the file will be.
- Therefore, any image you use on the Web with a larger resolution of 72 will create unnecessary large files that will take longer to download

Bitmapped Graphics (cont'd)

- Major browsers support three common formats:
  - GIF: Graphics Interchange Format ("gif" or "jif")
  - JPEG: Joint Photographic Experts Group Format ("jay peg")
  - PNG: Portable Network Graphics ("pee en gee" or "ping")

GIF Images

- It used to be standard for ALL web graphics
- Each pixel can be created with a palette of up to 256 colors
- This palette is built by choosing 256 colors out of 16 million colors

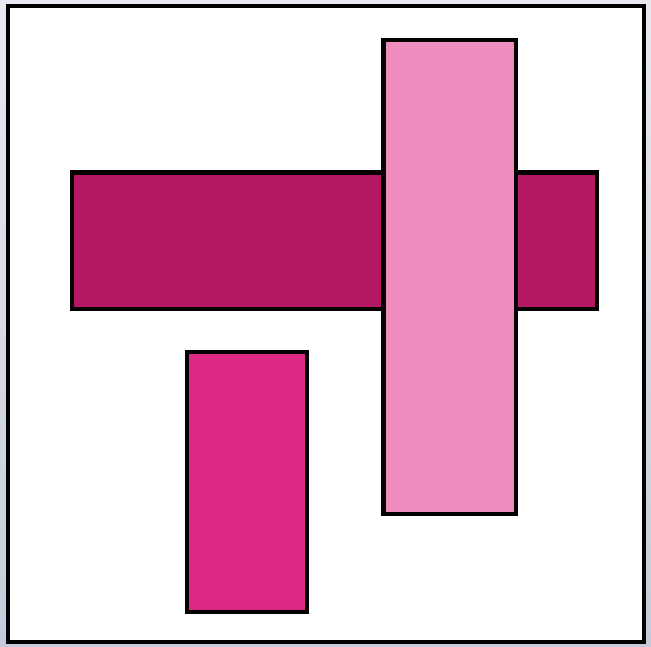
GIF Images

- **Indexed color format:** If many pixels use the same colors, the image does not have to repeat the same color information that results in a smaller file size.
- For this reason, GIF fits better to images that have large flat areas of color.
- The fewer colors the image uses, the smaller the GIF file is

GIF Images: LZW Compression

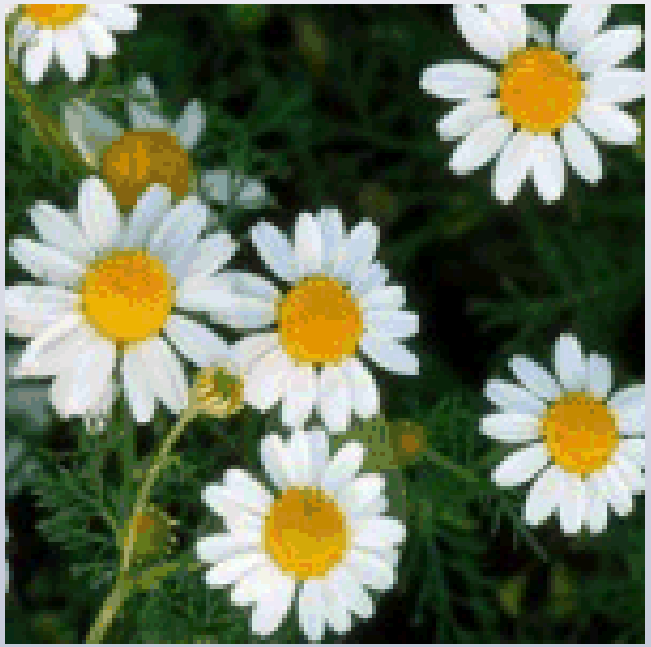
- It uses LZW compression:
  - Scan rows of the image. If find consecutive pixels of the same color it indicates that X number of pixels should be written from this point onwards using the same color.
  - It is **lossless compression** because no data is loss, and therefore there is no quality loss
  - It is not useful when there is no many consecutive pixels, or there are too many colors. So, not useful for photographs

GIF Images: LZW Compression (cont'd)



Contains areas of flat colors

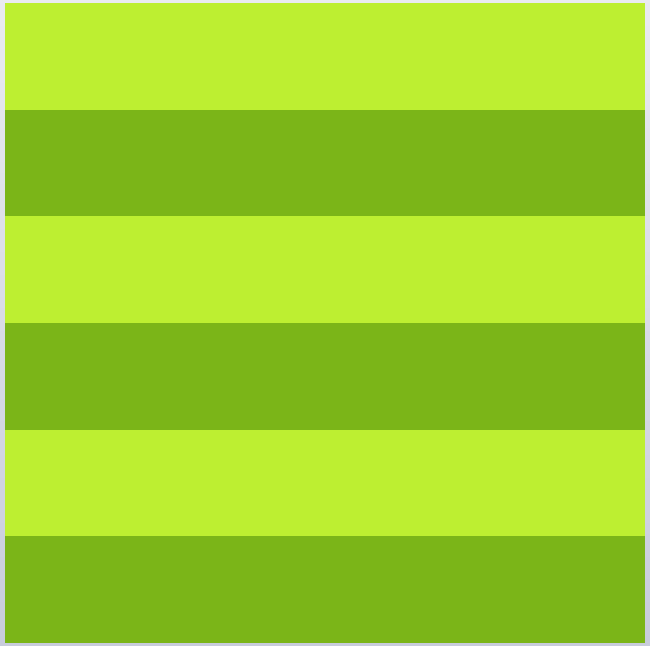
1,110 bytes



Photograph

17,848 bytes

GIF Images: LZW Compression (cont'd)



Horizontal color change

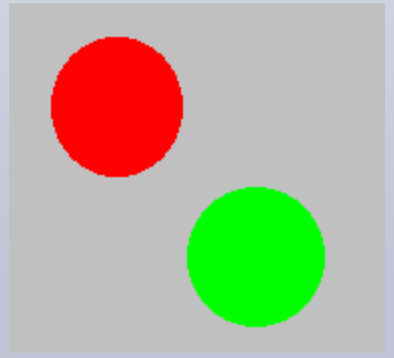
324 bytes

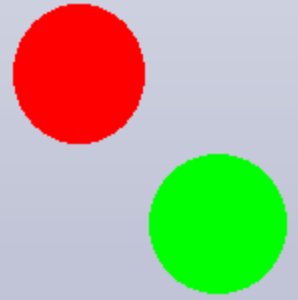
Vertical color change

1,109 bytes

GIF Images: Transparency

- Transparent background: You can specify one or more colors to represent a transparent background.
  - In the parts with those colors the background will be shown through
  - However, there is no degree of transparency.





GIF Images: Animated GIFs

- GIFs can store more than one frame (or copy of the image) within the same file.
- This allows a GIF to rotate between version/frames and create a simple animation. Same as in a flip-book animation.
- Not suitable for photographic images.
- They look fun, but since the animation is always the same, they can ***become tiresome easily***.

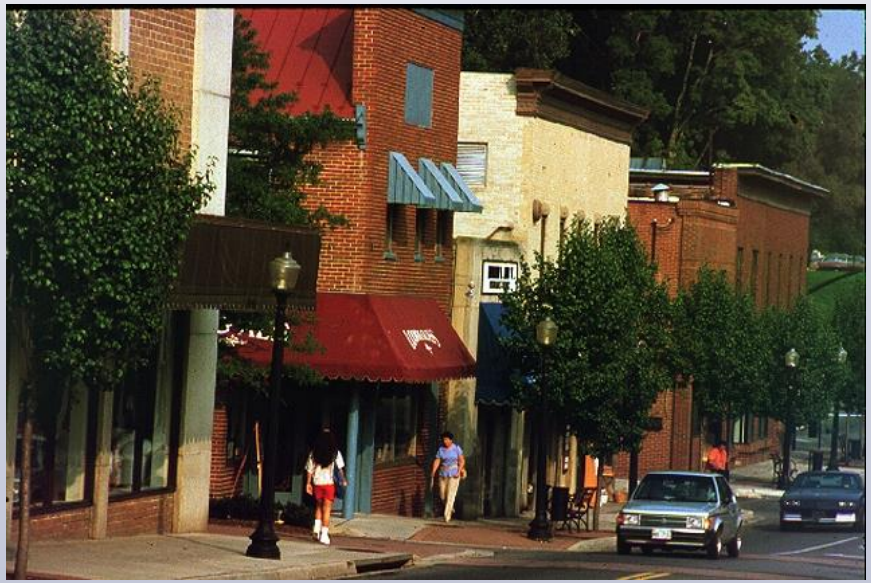
GIF Images: Animated GIFs

JPEG Images

- Developed as a standard for storing and compressing images such as photographs with wide ranges of colors.
- When compressing, data has to be discarded and original image cannot be recreated. Therefore, it uses a **lossy compression**
- They do not work well with images that have large amounts of flat color.
- The more you compress, the smaller your file size will be, and the more you will loose image quality.

JPEG Image: Loosy Compression

Original



326,321 bytes

95% Compressed

9,438 bytes

PNG Images

- Portable Network Graphics
- It is the most recent format (1990s)
- Created because GIF creator (Unisys) decided to charge a license fee to **software companies** for creating and viewing GIFs
- Overcome GIF problems (restriction of colors to 256)

PNG Images (cont'd)

- Two types:
  - 8-bit PNG (same limitations as GIF)
  - PNG-24:
    - Colors not restricted.
    - Different level of transparency. (Allows softer/smoother edges)
    - Can display image faster than GIF
    - Allow gamma correction (for use of color in different type of monitors)
- Most cases, compress better than GIF
- Drawbacks:
  - Old browsers do not support it
  - Do not support animations

Keep images size small

- Allows your page to load faster
- Allows to save transfer bandwidth (the amount of transferred data between your server and the final users)

Keep images size small

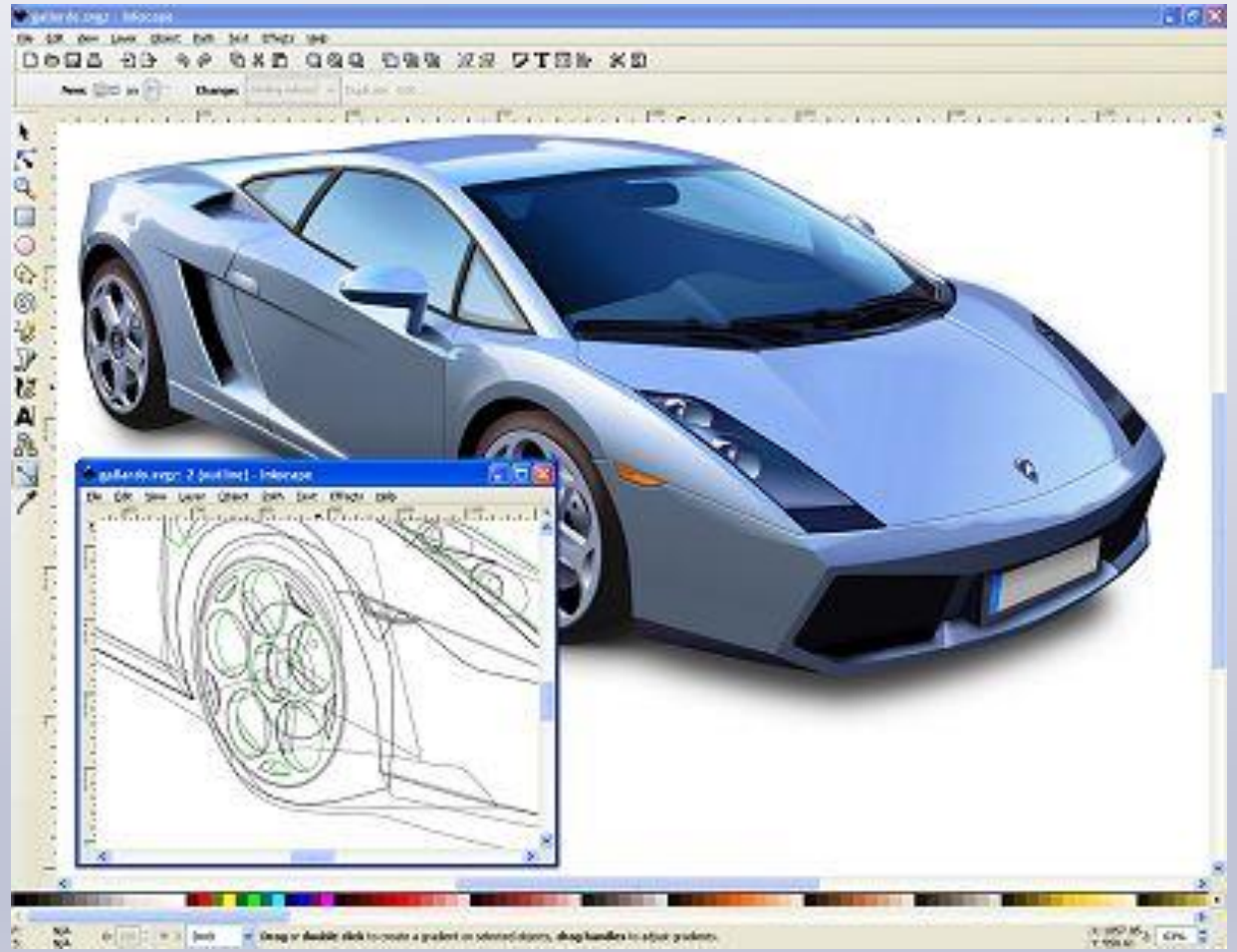
- Rule of thumb for choosing type of images:
  - JPEG:
    - photo-realistic pictures with a lot of detail
    - Subtle shade differences you want to preserve
  - PNG:
    - Images with flat color (no textured colors), hard edges such as diagrams, texts, or logos.
    - More than 256 colors (PNG-24)
    - Images with degree of transparency.
  - GIF:
    - Images with only 256 colors with flat colors.
    - You want to be sure that old browser supports it
    - Do not need to use a degree of transparency.

Vector Graphics

- Break the image into lines and shapes and store the lines as coordinates.
- Then, fill the spaces between the lines with color.
- They feature large areas of flat color (as opposed to bitmaps)
- **Used for**:
  - Line art
  - Illustration
  - Animation

(*) Image, taken from http://bucarotechelp.com

Vector Graphics



(*) Image, taken from http://bucarotechelp.com

Vector Graphics

- Most popular vector graphics format on the Web: **Flash**
- By default, vector graphics are not supported by XHTML or browsers
- Another lately tool is **Silverlight** created by Microsoft
- They need to be included in a page with the <object> element

(*) Image, taken from http://bucarotechelp.com

Adding images: <img>

-
- Attributes:
  - src = specify the URL of the image to load
  - alt = text that will be displayed in case user cannot see the image (can't load, or image is not supported by browser)
  - height = height of image in pixels
  - width = width of image in pixels
    - (height, width can be a % of page or element containing the image)
  - hspace, vspace = amount of whitespace around the image

(*) Image, taken from http://bucarotechelp.com

Adding images: <img>

Fixed size: width=130, height=130

Fixed size: width=160

Fixed size: width=80, height=150

(*) Image, taken from http://bucarotechelp.com

Adding images: <img> (cont'd)

```
<p>Fixed size: width 130 height 130</p>
  <img src="images/apple.jpg" alt="Photo of red apple"
    width="130"
  height="130" />
<p>Enlarged: width 160 (no height specified)</p>
  <img src="images/apple.jpg" alt="Photo of red apple"
    width="160" />
<p>Stretched: width 80 height 150</p>
  <img src="images/apple.jpg" alt="Photo of red apple"
    width="80" height="150" />
```

The <object> element

- Embed all media types into documents such as :
  - MP3,
  - Flash movies,
  - QuickTime movies,
  - JavaScript objects,
  - Java Applets, and so on.

The <object> element

- To embed an object to a page you need to specify:
  - Location of the code used to display or play the object
  - The actual data (movie, audio file, a program, etc)
  - Any other values the object needs at runtime
- <param>:
  - child of <object>. Use for providing other values
- Any other content in <object> will be display only if **browser cannot render** the object

The <object> element

- classid =
  - When you are trying to include Flash or QuickTime files and a plug-in needs to be loaded, this value would indicate the application required to play or run the file.

  i.e.:

  classid=”clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B”

The <object> element

- **codebase**:
  - Tells where the program such as Flash or QT can be found
  - For example, the QT ActiveX control is found in:
    - codebase="**http://www.apple.com/qtactivex/qtplugin.cab**"

The <object> element

- **codetype**:
  - Specifies the type of object expected by the browser.
  - Only relevant if classid has been specified.

    codetype = “video/quicktime”

- **data**:
  - Specifies the URL, If object has a file to process or play.
  - For example:
    - data = **“http://www.example.com/mp3/newsong.mp3”**

The <object> element

- **height and widht**:
  - Specifies width and height of the object
- **standby**:
  - Text string used when object is loading
    - standby = “Trailer for Movie is loading”
- param:
  - It is used to pass parameters
  - For example for a QT movie, src indicates the source of the file, while autoplay indicates whether movie should start automatically
    - <param name=”src” value=”movieTrailer.mov” />
    - <param name=”autoplay” value=”true” />

Adding a Flash movie to a page

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="300"
height="200"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash
.cab">
    <param name="movie" value="motion/flash_sample.swf">
    <param name="play" value="true">
    <param name="loop" value="false">
    <embed src="motion/flash_sample.swf" width="300" height="200" play="true"
    loop="false" QUALITY="best" menu="false" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version
    =ShockwaveFlash">
    </embed>
</object>
```

Images as links

- Just insert an image element between an <a> element.

```
<a href="../index.html" title="Click here to return to the home page">
<img src="images/banana.jpg" width="130" height="130"
   alt="Banana" border="0"
/></a>
```

Image Maps

- They are for creating areas or sections to a single image to later create hyperlinks.
- The <img> carries en extra element called **usemap**
- The value of the **usemap** attribute is the value of the name attribute on the <map> element
- **<map>:**
  - creates the map for the image
- **<area>:**
  - define the clickable hotspots (or areas or sections of an image)

Image Maps (cont'd)

- <shape >
  - **default**: The whole of the image not defined in the area
  - Rect or rectangle
  - Polygon or poly
  - Circle or circ

Image Maps (cont'd)

- <coords>: specifies the area that is the clickable hotspot
  - A rectangle contains four coordinates
  - A circle contains three coordinates; the first two are the center of the circle, while the third is the radius in pixels.
  - A polygon contains two coordinates for each point of the polygon. The shape is automatically closed.

- If you don't want the image to take you anywhere you should use a **nohref** attribute

Image Map Example

1. <object data=”gallery_map.gif” type=”image/gif” alt=”Gallery Map” width=”500”
2. height=”300” border=”0” **usemap**=”#**gallery**” />
3. <**map** name=”**gallery**”>
    1. Foyer
    2. <a shape=”**poly**” coords=”272,79,351,79,351,15,486,15,486,218,272,218,292,166,
    3. 292,136,270,76” href=”sculpture_garden.html” target=”_self”>Sculpture garden
    4. <a shape=”**rect**” coords=”325,224,488,286” href=”workshop.html”
    5. target=”_self”> Artists workshops
4. </map>

Questions?

- Before the exercise

Web Programming

Week 06
Slides 170-192

Outline

- Introducing Tables
- Basic Table Elements and Attributes
- Advance Tables
- Accessibility Issues with Tables
- Exercise

Introducing Tables

Introducing Tables

- Think of grids, or spreadsheets in a web page.
- They consist of **rows** and **columns**.
- The interception of a row and a column is called a "**cell**"
  - **Row**: set of cells in the same line from left to right
  - **Column**: line of cells going from top to bottom
- The XHTML element is:

<table>

The <table> element

- A table is written out row by row.
- A row is contained inside a <tr> element
- **<tr>**: table row
- Each cell is written inside the row element using a <td> element.
- **<td>**: table data

The <table> element

- A table is written out row by row.
- A row is contained inside a <tr> element
- **<tr>**: table row
- Each cell is written inside the row element using a <td> element.
- **<td>**: table data

The <table> element (cont'd)

```
<table border="1">
 <tr>
   <td> column 1, row 1</td>
    <td> column 2, row 1</td>
    ...
</tr>
<tr>
    <td> column 1, row 2</td>
    <td> column 2, row 2</td>
    ...
</tr>
....
</table>
```

Indicates the start of a table

Indicates the start of a row

Table data (cell)

Indicates the end of a row

column1 row 1	column 2 row 1	column 3 row 1	column 4 row 1	column 5 row 1
column1 row 2	column 2 row 2	column 3 row 2	column 4 row 2	column 5 row 2
column1 row 3	column 2 row 3	column 3 row3	column4 row 3	column 5 row 3
column1 row 4	column 2 row 4	column 3 row 4	column 4 row 4	column 5 row 4

A slightly more complex example

Go to:

..\web-dev.localhost\table-slightly-complex.html

Basic Table Elements and Attributes

Md. Masudur Rahman, Dept. Of CSE, UGV

The <table> element attributes

- <table>: contains
  - All of the universal attributes
  - Basic event attributes for scripting
- Deprecated attributes:
  - **align:** indicates where the table should be align (left, center, right). Text flows around the table when uses align.
  - **bgcolor**: sets the background color for the table
  - **border**: it will create a border around the table and each cell
  - **cellpadding**: create a space between cell and its content.
  - **cellspacing**: create space between borders of each cell
  - **width**: specify the width of the table in pixels, or as a percentage

The <tr> element attributes

- <tr>: contains
  - **align:** specifies the position of the content of all the cells in the row (left, center, right, justify).
  - **bgcolor**: sets the background color of the row
  - **valign**: specified the vertical alignment of the contents of each cell in the row. (top, middle, bottom, baseline)

  - *All of these attributes have been deprecated in favor of CSS

The <td> and <th> attributes

- Every cell is represented by either:
  - <td>: cells containing table data
  - <th>: cells containing table headings
- By default:
  - <td>: bold font, horizontally aligned in the center of the cell
  - <td>: left-aligned, not in bold.
- Any styles that these attributes have will override settings for the table and for row elements.

The <td> and <th> attributes

- **align**(*): sets the horizontal alignment for the content of the cell (left, right, center, justify)
- **bgcolor** (*): sets the background color for the cell.
- **colspan**: specify how many columns of the table a cell will span across.
- **rowspan**: specifies the number of rows of the table a cell will span across.
- **height**(*): specifies the height of a cell
- **width**(*): specified the width of a cell
- **valign**(*): specifies vertical alignment (top, middle, bottom)

Advance Tables

Advance Tables

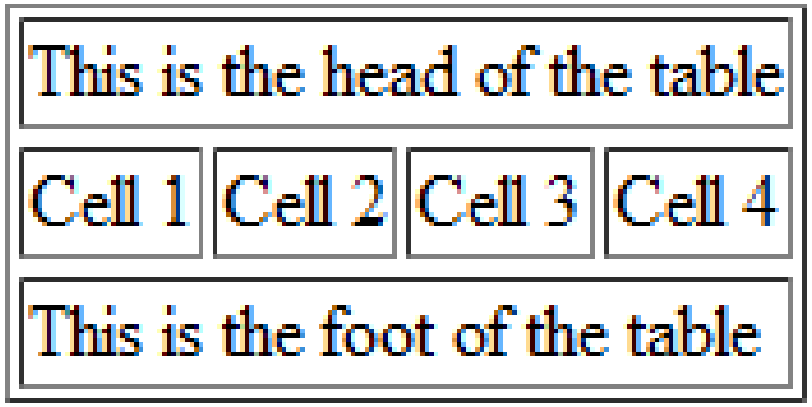
- Splitting table into three sections
  - head
  - body
  - foot
- Captioning tables
- Using rowspan and colspan
- Grouping columns using <colgroup>
- Sharing attributes between unrelated columns using <col> element

Splitting up Tables

- Can be divided into three parts: head, body and foot
  - <thead>: to create a separate table header
  - <tbody>: to indicate the main bod
  - <tfoot>: to create a separate table footer
- <tfoot> must appear before <tbody>

Example using <thead>, <tbody>, <tfoot>

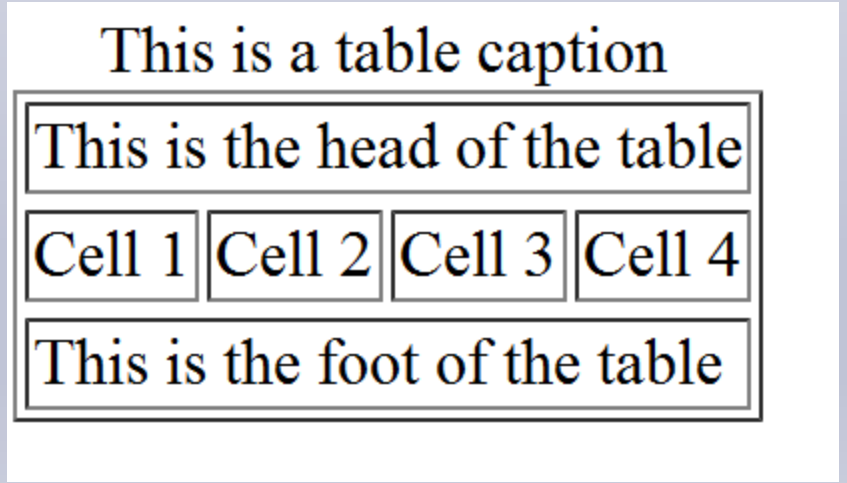
```
1.  <table>
2.          <thead>
3.            <tr>
4.              <td colspan="4">This is the head of the table</td>
5.            </tr>
6.          </thead>
7.  <tfoot>
8.          <tr>
9.            <td colspan="4">This is the foot of the table</td>
10.         </tr>
11. </tfoot>
12. <tbody>
13.         <tr>
14.           <td>Cell 1</td>
15.           <td>Cell 2</td>
16.           <td>Cell 3</td>
17.           <td>Cell 4</td>
18.         </tr>
19. </tbody>
20. </table>
```



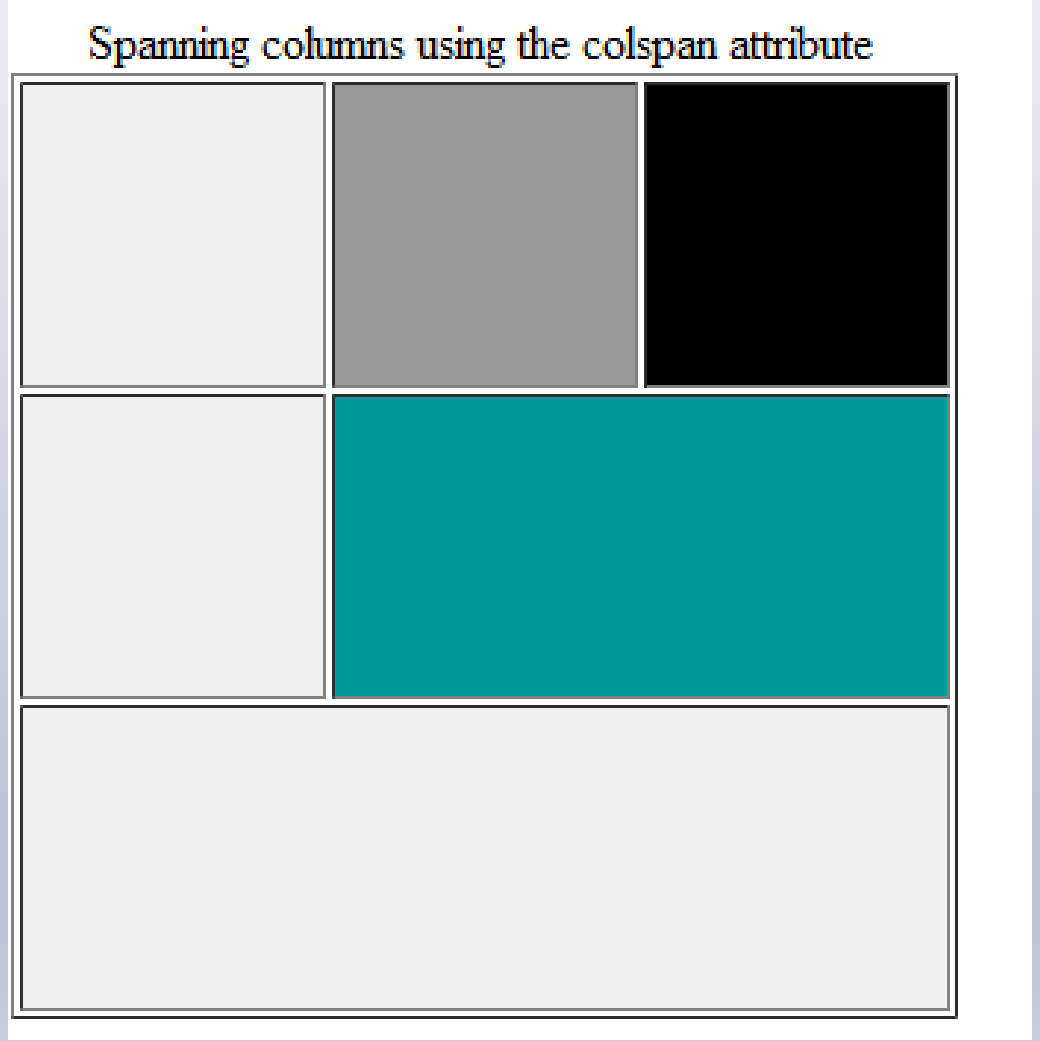
<table>: The <caption> element

- Indicates the caption of the table. The caption must be between the <caption> tags, and **immediately after <table> and before** the first row.

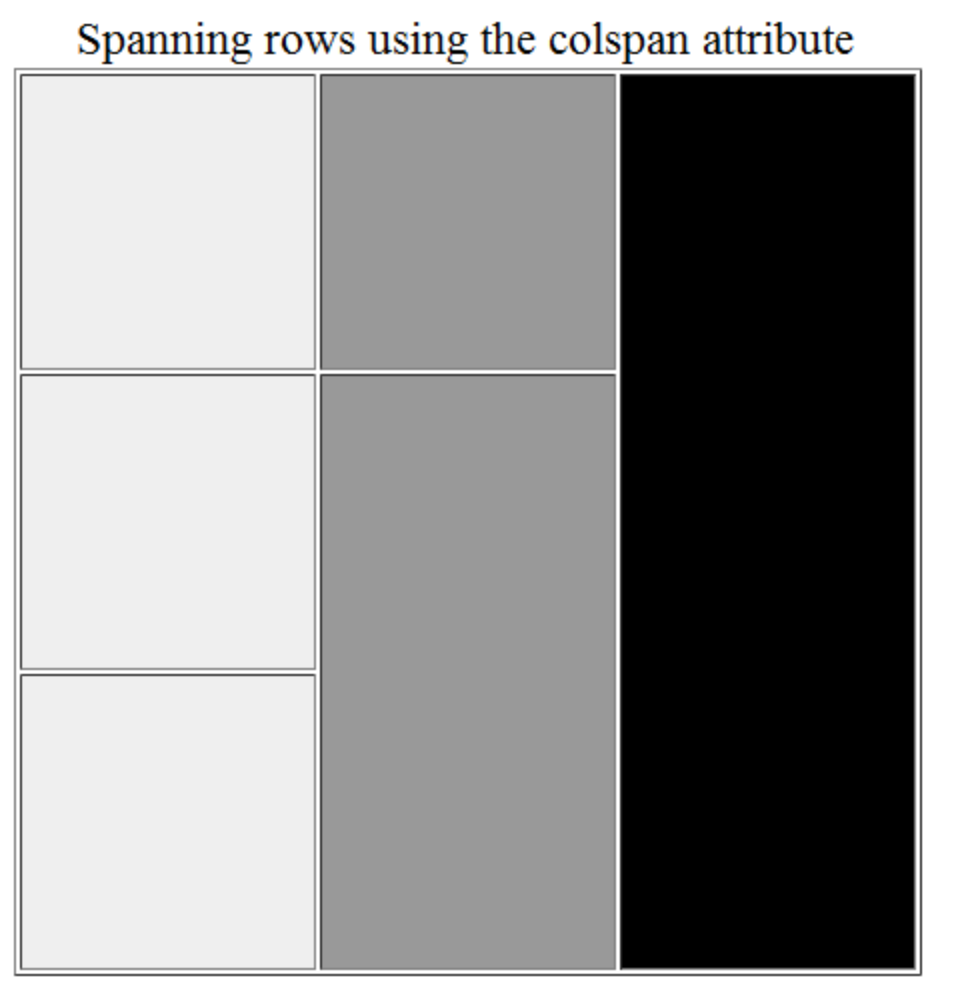
```
<table border="1">
<caption>This is a table caption</caption>
…
…
```



Spanning columns: the colspan attribute



Spanning rows: the rowspan attribute



Grouping columns: The <colgroup> element

- It groups one or more adjacent columns into a group.
- It uses the <**colgroup**> element to create groups
- This allows to format different group of columns rather than formatting each column at a time.

Example of grouping columns

```
1. <table>
2.   <colgroup span="8" class="mainColumns" />
3.   <colgroup span="2" class="subTotalColumns" />
4.   <colgroup span="3" class="totalColumns" />
5.   <tr>
6.     <td>…<td>
7.     …
```

Questions?

- Before the exercise

Web Programming

Week 07
Slides 193-265

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

- Introducing Forms
- The <form> element
- Focus
- Sending form data to the server
- Exercise

Introducing Forms

Introducing Form

- Any form is declared using the <form> element
- Any <form> element can contain other XHTML elements
- Once users enter information into the form, they submit the form data to a web server.

Introducing Forms (Cont'd)

- Data is sent to the server in name/value pairs
- name is the form control name
- Value is what the use has entered of the value of the option selected

```
<form action="http://www.example.org/search.aspx"
    method="get">
  <h3>Search the site</h3>
  <input type="text" name="txtSearchItem" />
  <input type="submit" value="Search" />
</form>
```

Introducing Forms (Cont'd)

- The <form> element carries an attribute called action whose value is the URL of the page on the web server that handles search requests.

- The method attribute meanwhile indicates which **HTTP method** will be used in getting the form data to the server.

Creating a Form: the <form> element

- A <form> element can contain any other markup such as paragraphs, heading and so on.
- A <form> element **cannot** contain another <form> element
- Every form should carry at least two attributes:
  - action
  - method

The action Attribute

- Indicates what happens to the data when is ubmitted
- Usually a page or a program on a web server
- For example:
  - A login form containing a username and a password can be send to a PHP, JSP, or ASP pages that will execute a program and will return an action such as allowing you to log in, or rejecting you.

<form **action**=”http://www.example.org/login.php”>

The method Attribute

- Form data can be sent to the server in two ways, each corresponding to an *HTTP method:*
  - ***get*** *: sends data as part of the URL*
  - ***post*** *: hides data in the HTTP headers*

The id Attribute

- Identifies uniquely a <form> element within a page.

- **Good practice:** is to give every <form> element an id attribute, because many forms make use of style sheets and scripts, which may require the use of the id attribute to identify the form.

The onsubmit Attribute

- At some point, you have probably filled in a form on a web site, and then, as soon as you have clicked the button to send the form data (even before the page is sent to the server), a message is shown telling you that you have missed entering some data, or entered the wrong data.

- When a user clicks a submit button, something called an *event fires*.

- *This event can be a script to be executed such as a JavaScript code that will check for the validity of your form data before sending it to the webserver.*

The onsubmit Attribute (cont'd)

- on the <form> element might look like this:
  - onsubmit="validateFormDetails( );"
- validateFormDetails( ) function should have been defined in the document already.
- So when the user clicks the submit button, this function will be called and run.

The onsubmit Attribute (cont'd)

- Two advantages:
  - The user does not have to wait the extra time it would take for the page to be sent to the server and then returned if there are any errors.
  - The server does not have to deal with as much error checking as it would if the checks by the browser had not been performed.

The onreset Attribute

- Some forms contain a reset button that empties the form of all details.

- When this button is pressed, an **onreset event fires** and a script can be run.

White Spaces and the <form> element

- You should also be aware that, when a browser comes across a <form> element it often creates extra white space around that element.

- To avoid the extra space created, you can try either placing the <form> element near the start or end of the document

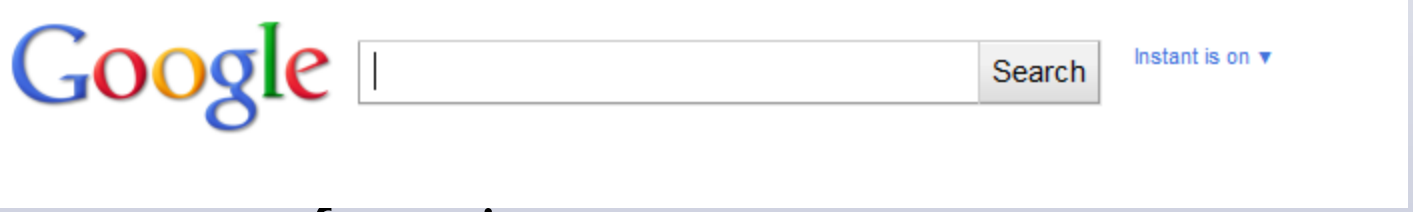
Form Controls

Form Controls

- Text input controls
- Buttons
- Checkboxes and radio buttons
- Select boxes (sometimes referred to as drop-down menus) and list boxes
- File select boxes
- Hidden controls

Text Inputs

- The most famous text input box is the one right in the middle of the Google home page.



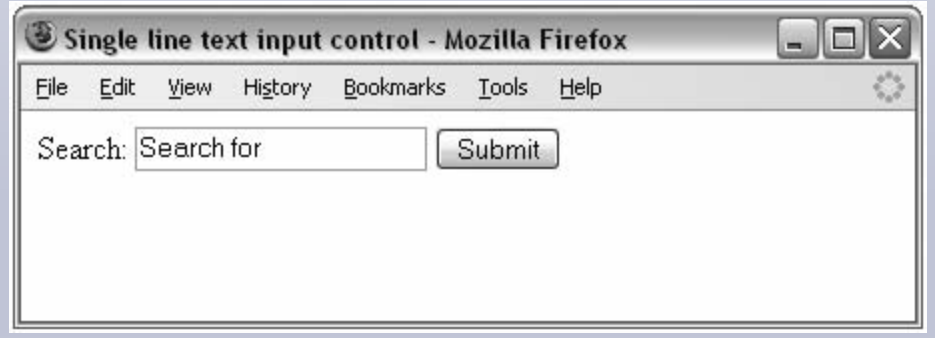
- Three types of text input:
  - Single-line
  - Password
  - Multi-line text

Text Inputs

- Single Line text input controls: **<input>**
  - Used for items that require only one line of user input, such as search boxes or e-mail addresses.
  - They are created using the <input> element whose value **type** attribute has a value of text.

Text Inputs

```
<form action=”http://www.example.com/search.php” method=”get”
    name=”frmSearch”>
Search:
<input type=”text” name=”txtSearch” value=”Search for” size=”20”
maxlength=”64” />
<input type=”submit” value=”Submit” />
</form>
```



The text input attributes

Attribute	Purpose
`type`	Indicates the type of input control you want to create. The value for this attribute should be `text` when you want to create a single-line text input control. This is required because the `<input>` element is also used to create other form controls such as radio buttons and checkboxes.
`name`	Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered. Each control needs to have a name so that the associated value (supplied or chosen by the user) can be retrieved individually at the other end.
`value`	Provides an initial value for the text input control that the user will see when the form loads. You need to use this attribute only if you want something to be written in the text input when the page loads (such as a cue for what the user should be entering); more often you are likely to leave it blank.
`size`	Allows you to specify the width of the text-input control in terms of characters; the search box in the earlier example is 20 characters wide. The `size` property does not affect how many characters users can enter (they could enter 40 characters); it just indicates how many characters wide the input will be. If users enter more characters than the size of the input, they can scroll right and left using the arrow keys.
`maxlength`	Allows you to specify the maximum number of characters a user can enter into the text box. Usually after the maximum number of characters has been entered, even if the user keeps pressing more keys, no new characters will be added.

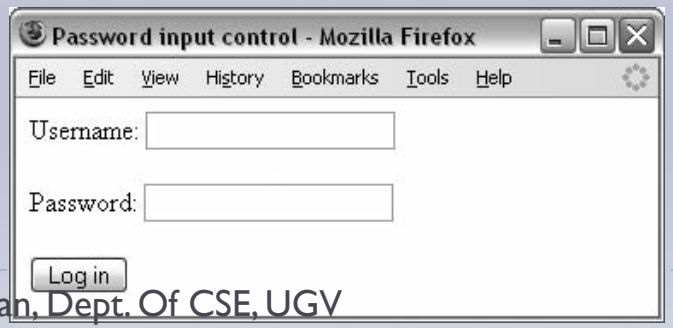
Password Input Controls

- If you want to collect sensitive data such as passwords and credit card information, you should use the password input.
- The password input masks the characters the user types on the screen by replacing them with either a dot or asterisk.
- Password input controls are created almost identically to the single-line text input controls, except that the **type** attribute on the **<input>** element is given a value of **password**.

Password Input Controls (Cont'd)

- Here you can see an example of a login form that combines a single-line text input control and a password input control.

```
<form action="http://www.example.com/login.aspx" method="post">
Username:
<input type="text" name="txtUsername" value="" size="20" maxlength="20" />
<br />
Password:
<input type="password" name="pwdPassword" value="" size="20" maxlength="20" />
<input type="submit" value="Submit" />
</form>
```



Password Input Controls (Cont'd)

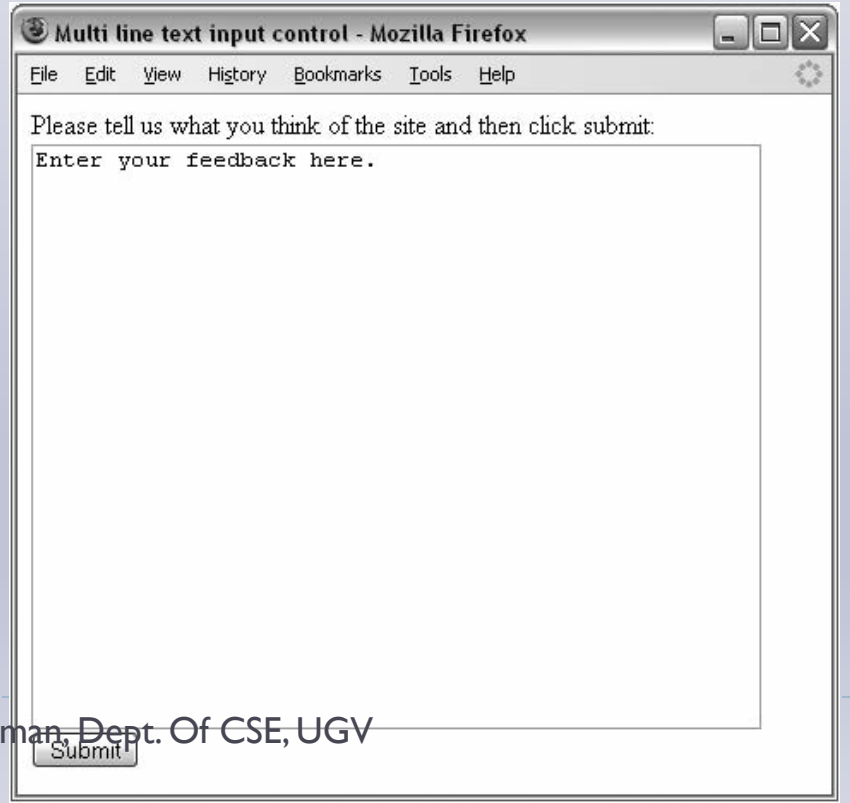
- *While passwords are hidden on the screen, they are still sent across the Internet as clear text.*
- *In order to make them secure you should use an SSL connection between the client and server.*

Multiple-Line Text Input Controls

- If you want to allow a visitor to your site to enter more than one line of text, you should create a multipleline text input control using the **<textarea>** element.

Multiple-Line Text Input Controls (Cont'd)

```
<form action="http://www.example.org/feedback.asp" method="post">
    Please tell us what you think of the site and then click submit:<br />
    <textarea name="txtFeedback" rows="20" cols="50">
    Enter your feedback here.
    </textarea>
    <br />
    <input type="submit" value="Submit" />
</form>
```



Multiple-Line Text Input Controls (Cont'd)

- Multi-line text input control Attributes:

Attribute	Purpose
`name`	The name of the control. This is used in the name/value pair that is sent to the server.
`rows`	Used to specify the size of a `<textarea>`, it indicates the number of rows of text a `<textarea>` element should have and therefore corresponds to its height.
`cols`	Used to specify the size of a `<textarea>`; here it specifies the width of the box and refers to the number of columns. One column is the average width of a character.

Buttons

- Buttons are most commonly used to submit a form, although they are sometimes used to clear or reset a form and even to trigger client-side scripts..
- You can create a button in three ways:
  - Using an <input> element with a type attribute whose value is submit, reset, or button
  - Using an <input> element with a type attribute whose value is image
  - Using a <button> element
- With each different method, the button will appear slightly different.

Buttons: using <input type="">

- **submit**, which creates a button that automatically submits a form
- **reset**, which creates a button that automatically resets form controls to their initial values
- **button**, which creates a button that is used to trigger a client-side script when the user clicks that button

Buttons: using <input type=“”>

```
<input type=”submit” name=”btnVoteRed” value=”Vote for reds” />
<input type=”submit” name=”btnVoteBlue” value=”Vote for blues” />
<br /><br />
<input type=”reset” value=”Clear form” /> <br /><br />
<input type=”button” value=”Calculate” onclick=”calculate()” />
```

Buttons attributes

The table that follows shows the attributes used by the buttons.

Attribute	Purpose
`type`	Specifies the type of button you want and takes one of the following values: `submit`, `reset`, or `button`.
`name`	Provides a name for the button. You need to add only a `name` attribute to a button if there is more than one button on the same form (in which case it helps indicate which button was clicked). It is considered good practice, however, to use it anyway to provide an indication of what the button does.
`value`	Enables you to specify what the text on the button should read. If a `name` attribute is given, then the value of the `value` attribute is sent to the server as part of the name/value pair for this form control. If no value is given, then no name/value pair is sent for this button.
`size`	Enables you to specify the width of the button in pixels, although Firefox 2 and IE7 do not support this attribute.
`onclick`	Used to trigger a script when the user clicks the button; the value of this attribute is the script that should be run.

In the same way that you can trigger a script when the user clicks a button, you can also trigger a script when the button gains or loses focus with the `onfocus` and `onblur` event attributes.

Using images for Buttons

- You can use an image for a button rather than using the standard button that a browser renders for you.

```
<input type="image" src="submit.jpg" alt="Submit"
  name="btnImageMap" />
```

- **src**: Specifies the source of the image file.
- **alt**: Provides alternative text for the image. This will be displayed when the image cannot be found

Using images for Buttons (Cont'd)

- If the image button has a name attribute, when you click it, the browser sends a name/value pair to the server.
- The **name** will be what you provide for the name attribute and the **value** will be a pair of *x and y* coordinates for where on the button the user clicked

Using images for Buttons (Cont'd)

```
<form action="http://www.example.org/feedback.php"
   method="post">
   <button type="submit">Submit</button>
   <button type="reset"><b>Clear this form</b> I want
   to start again</button>
   <button type="button"><img src="submit.gif"
   alt="submit" /></button>
</form>
```

Creating Buttons: the <button> element

- The **<button>** element is a more recent introduction that allows you to specify what appears on a button between an opening <button> tag and a closing </button> tag.
- You can include **textual** markup or **image** elements between these tags.

Example: the <button> element

```
<button type="submit">Submit</button>
<br /><br />
<button type="reset"><b>Clear this form</b> I want to
  start again</button>
<br /><br />
<button type="button"><img src="submit.gif"
  alt="submit" /></button>
```

Checkboxes

- Checkboxes are just like the little boxes that you have to check on paper forms.
- They can be either on or off.
- When they are checked they are on and the user can simply toggle between on and off positions by clicking the checkbox

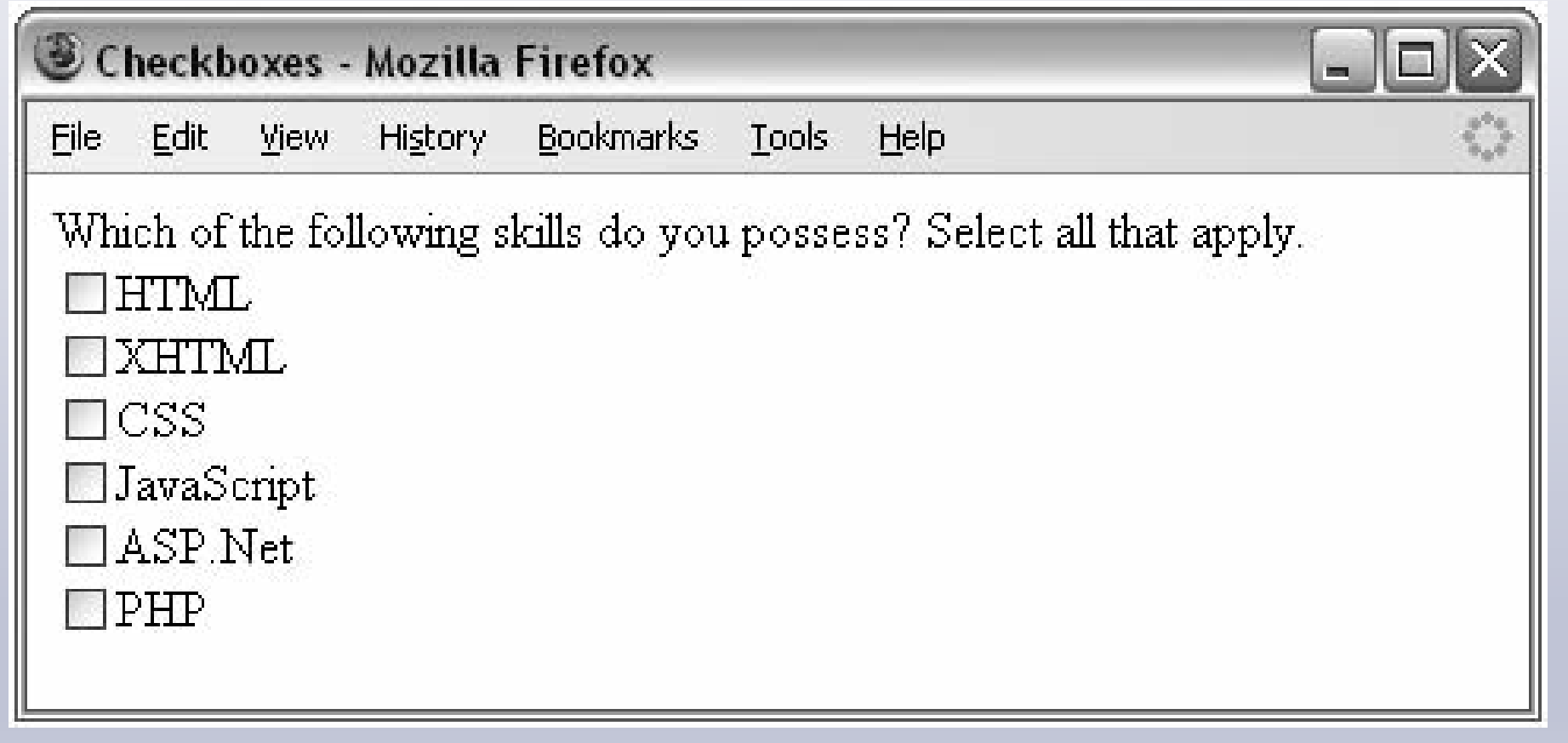
Checkboxes (Cont'd)

- Checkboxes can appear:
  - Individually, with each having its own name, or
  - As a group of checkboxes, that share a control name and allow users to select several values for the same property.

Checkboxes (Cont'd)

- They are ideal when:
  - Provide a simple **yes** or **no** response with one control (such as accepting terms and conditions or subscribing to an e-mail list)
  - Select several items from a list of possible options (such as when you want a user to indicate all of the skills they have from a given list)

Checkboxes (Cont'd)



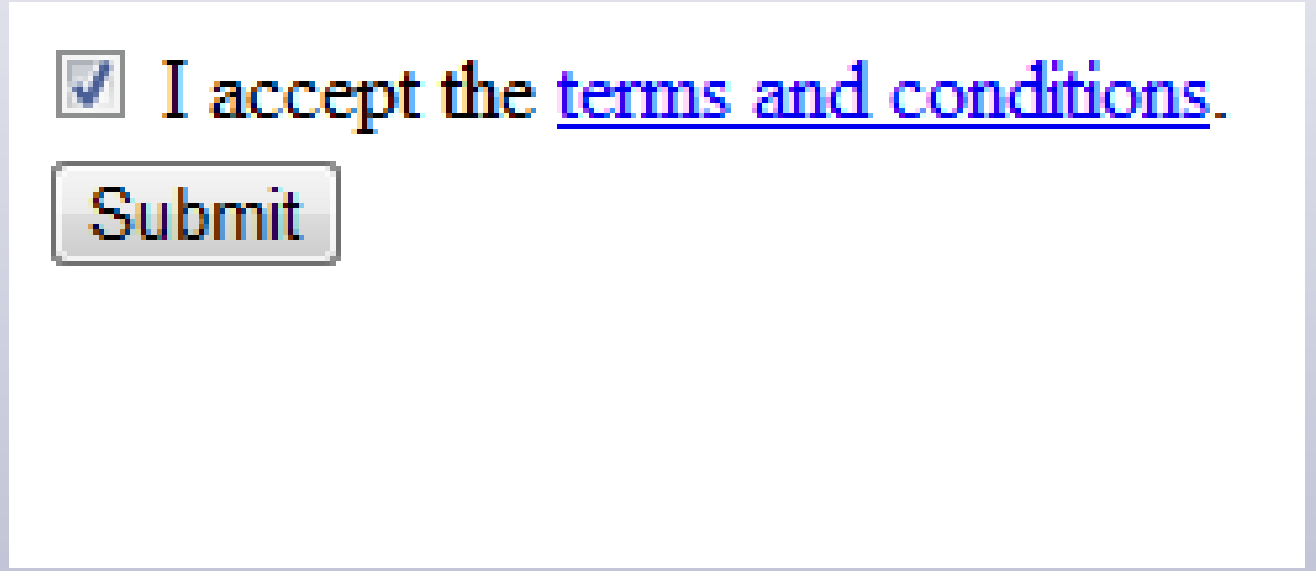
Checkboxes (Cont'd)

```
<form action="http://www.example.com/cv.aspx" method="get"
    name="frmCV">
Which of the following skills do you possess? Select all that apply.
    <input type="checkbox" name="chkSkills" value="html" />HTML <br />
    <input type="checkbox" name="chkSkills" value="xhtml" />XHTML <br />
    <input type="checkbox" name="chkSkills" value="CSS" />CSS<br />
    <input type="checkbox" name="chkSkills" value="JavaScript"/>
    JavaScript<br />
    <input type="checkbox" name="chkSkills" value="aspnet" />ASP.Net<br />
    <input type="checkbox" name="chkSkills" value="php" />PHP
</form>
```

Checkboxes (Cont'd)

- If someone selects more than one skill there will be several name/value pairs sent to the server that all share the same name.
- How you process multiple checkboxes with the same name depends on how you send the data to the server.
  - If you use HTTP get to send the data, then the selected checkbox will be sent as part of the URL in the query string.
  - If you use the HTTP post method, however, then you'll get an array that you can loop through representing the checked options.

Single Checkbox



Single Checkbox code

```
<form action="http://www.example.org/accept.php"
   name="frmTandC" method="get">
   <input type="checkbox" name="chkAcceptTerms"
   checked="checked" />
   I accept the <a href="terms.htm">terms and conditions</a>.
   <br />
   <input type="submit" />
</form>
```

- Note how the <input> element that creates this checkbox does not carry a **value** attribute. In the absence of a value attribute, the value is **on**.
- The attribute **checked**, with a value of checked, indicates that when the page loads the checkbox this is selected.

The <input> Element Attributes: checkbox

- type:
  - Indicates that you want to create a checkbox.
- name:
  - Gives the name of the control. Several checkboxes may share the same name, but this should only happen if you want users to have the option of selecting several items.
- value:
  - The value that will be sent to the server if the checkbox is selected.
- checked:
  - Indicates that when the page loads, the checkbox should be selected.

Radio Buttons

Please select which class of travel you wish to fly:

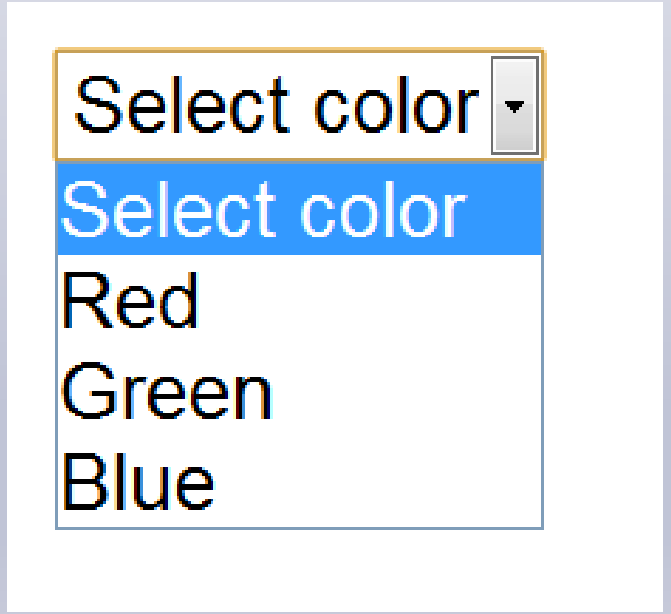
- First class
- Business class
- Economy class

Radio Buttons (Cont'd)

- Similar to checkboxes in that they can be either on or off, but there are two key differences:
  - When you have a group of radio buttons that share the same name, only one of them can be selected. Once one radio button has been selected, if the user clicks another option, the new option is selected and the old one deselected.
  - You should not use radio buttons for a single form control where the control indicates on or off because once a lone radio button has been selected **it cannot be deselected again** (without writing a script to do that).

Radio Buttons (Cont'd)

- Radio buttons are ideal if you want to provide users with a number of options from which they can pick **only one**.
- An alternative is using a **drop-down select box** that allows users to select only one option from several.



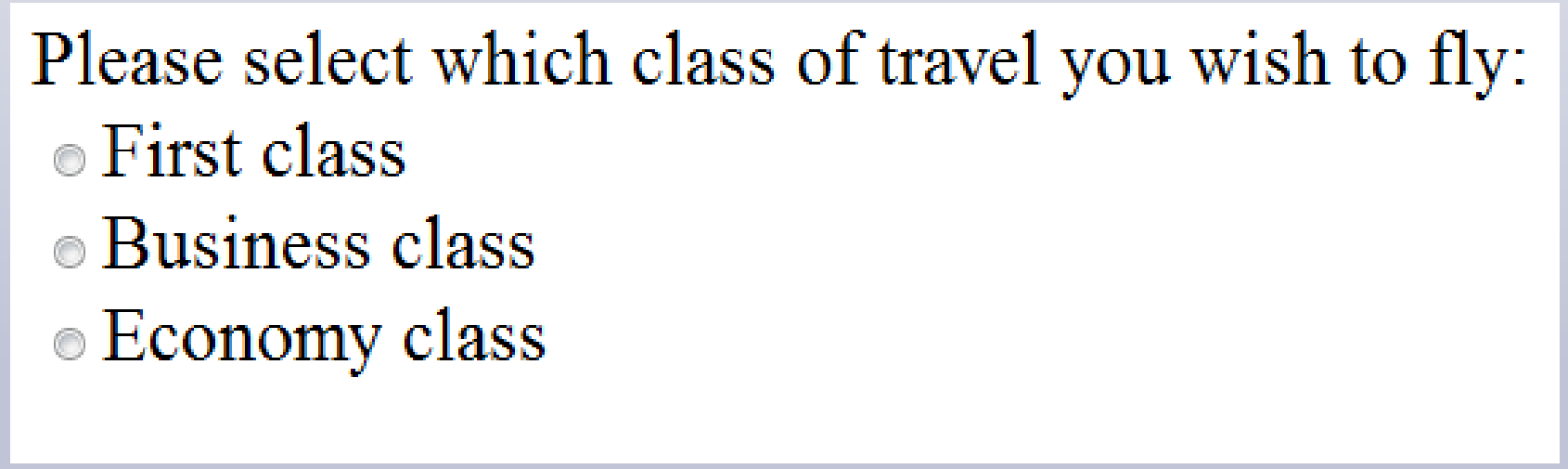
Radio Buttons or a Select Box?

Your decision between whether to use a select box or a group of radio buttons depends on three things:

- **Users expectations:** If your form models a paper form where users would be presented with several checkboxes, from which they can pick only one, then you should use a group of radio buttons.
- **Seeing all the options:** If users would benefit from having all the options in front of them before they pick one, you should use a group of radio buttons.
- **Space:** If you are concerned about space, a drop-down select box will take up far less space than a set of radio buttons.

Radio Buttons: <input> element

- The <input> element is again used to create radio buttons.
- **type** attribute should be given a value of **radio**.



Radio Button code

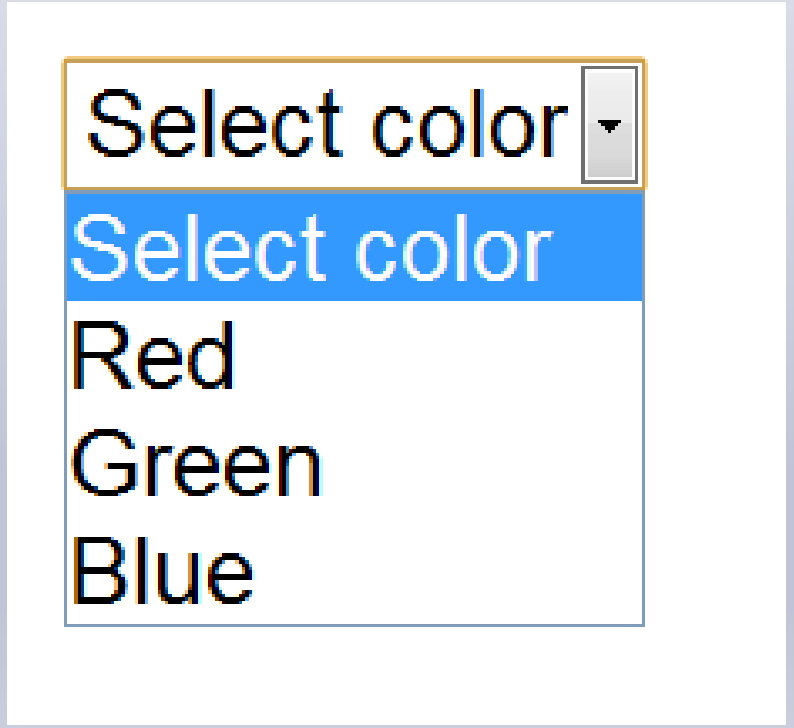
```
<form action="http://www.example.com/flights.aspx"
   name="frmFlightBooking" method="get">
   Please select which class of travel you wish to fly: <br />
   <input type="radio" name="radClass" value="First" />First
      class <br />
   <input type="radio" name="radClass" value="Business"
      />Business class <br />
   <input type="radio" name="radClass" value="Economy"
      />Economy class <br />
</form>
```

Radio Button Attributes

- **type:**
  - To indicate that you want a radio button form control.
- **name**:
  - The name of the form control.
- **value:**
  - Used to indicate the value that will be sent to the server if this option is selected.
- **checked**:
  - Indicates that this option should be selected by default when the page loads.

Select Boxes

- A drop-down select box allows users to select one item from a drop-down menu.
- Drop-down select boxes can take up far less space than a group of radio buttons.

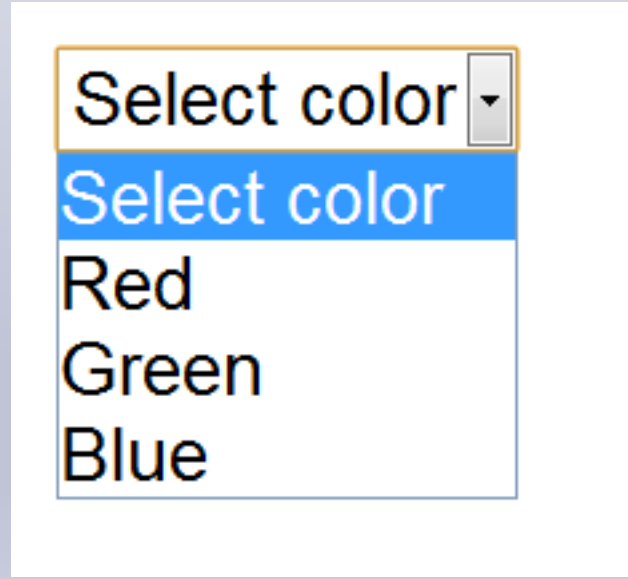


Select Boxes (Cont'd)

- Drop-down select boxes can also provide an alternative to single-line text input controls where you want to **limit the options** that a user can enter.
  - For example to indicate a country where the user lives. This would limit the value to USA to someone from here rather than having to deal with different values people write such as U.S.A., U.S., United States of America, or North America.

Select Box Example

```
<select name="selColor">
   <option selected="selected" value="">Select color</option>
   <option value="red">Red</option>
   <option value="green">Green</option>
   <option value="blue">Blue</option>
</select>
```



Select Box Example (Cont'd)

- The text between the opening **<option>** element and the closing </option> tags is used to display options to the user
- The value that would be sent to the server if that option is selected is given in the **value** attribute.
- Notice that the first **<option>** element **does not have a value** and that its content is Select color
  - This is to indicate to the user that he or she must pick one of the color choices.

The <select> element Attributes

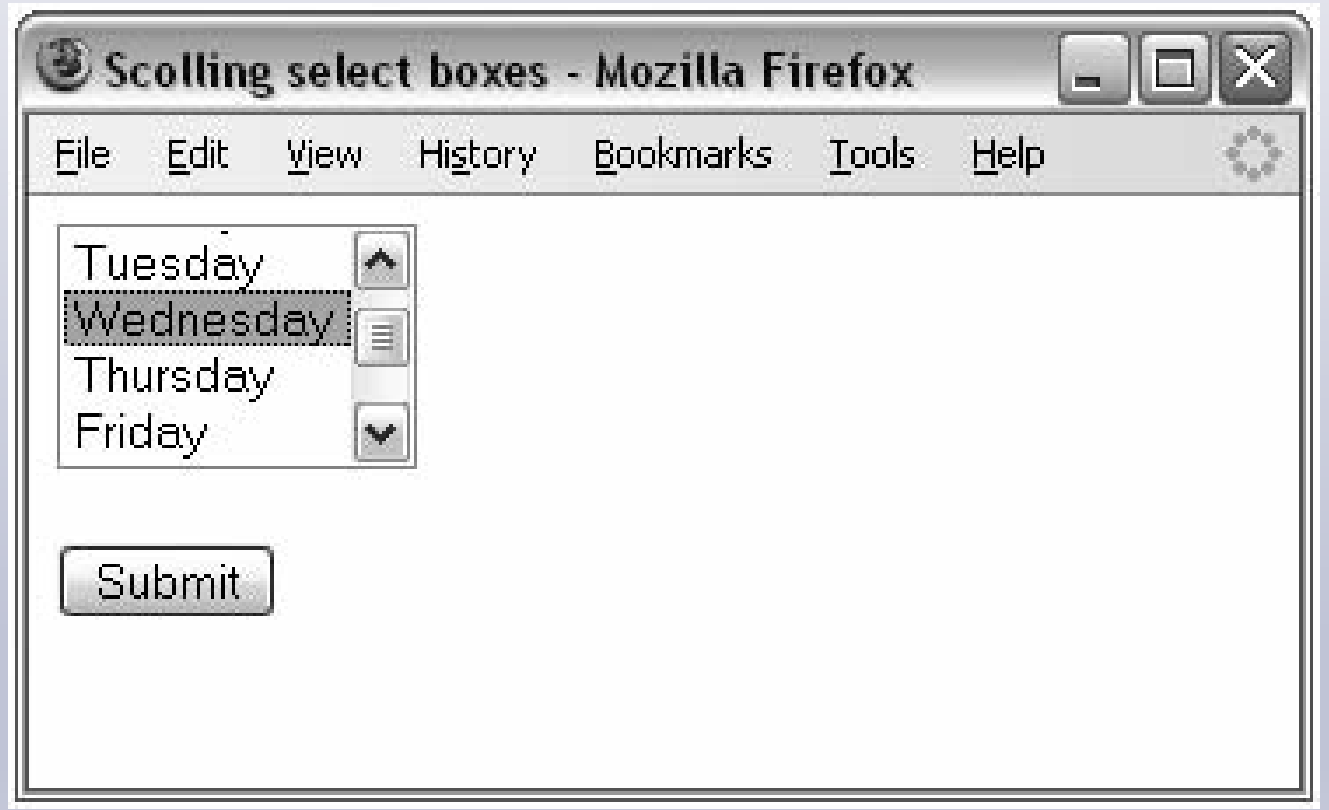
- The <select> element is the containing element for a drop-down list box.
- **name**:
  - The name for the control.
- **size:**
  - Can be used to present a scrolling list box, as you will see shortly. Its value would be the number of rows in the list that should be visible at the same time.

The <option> element Attributes

- Inside any <select> element you will find at least one <option> element.
- **value**:
  - The value that is sent to the server if this option is selected.
- **selected**:
  - Specifies that this option should be the initially selected value when the page loads. in order to be valid XHTML you should give this attribute a value of selected.

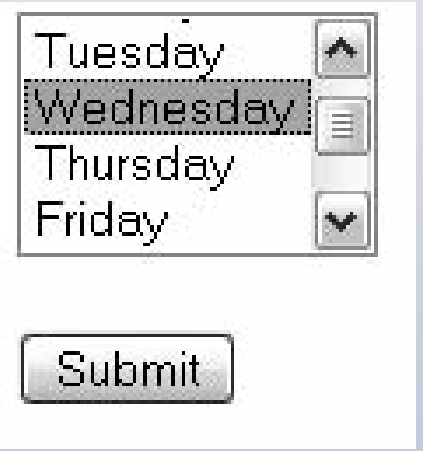
Creating Scrolling Select Boxes

- You just need to add the size attribute to a select box.



Scrolling Select Boxes Example

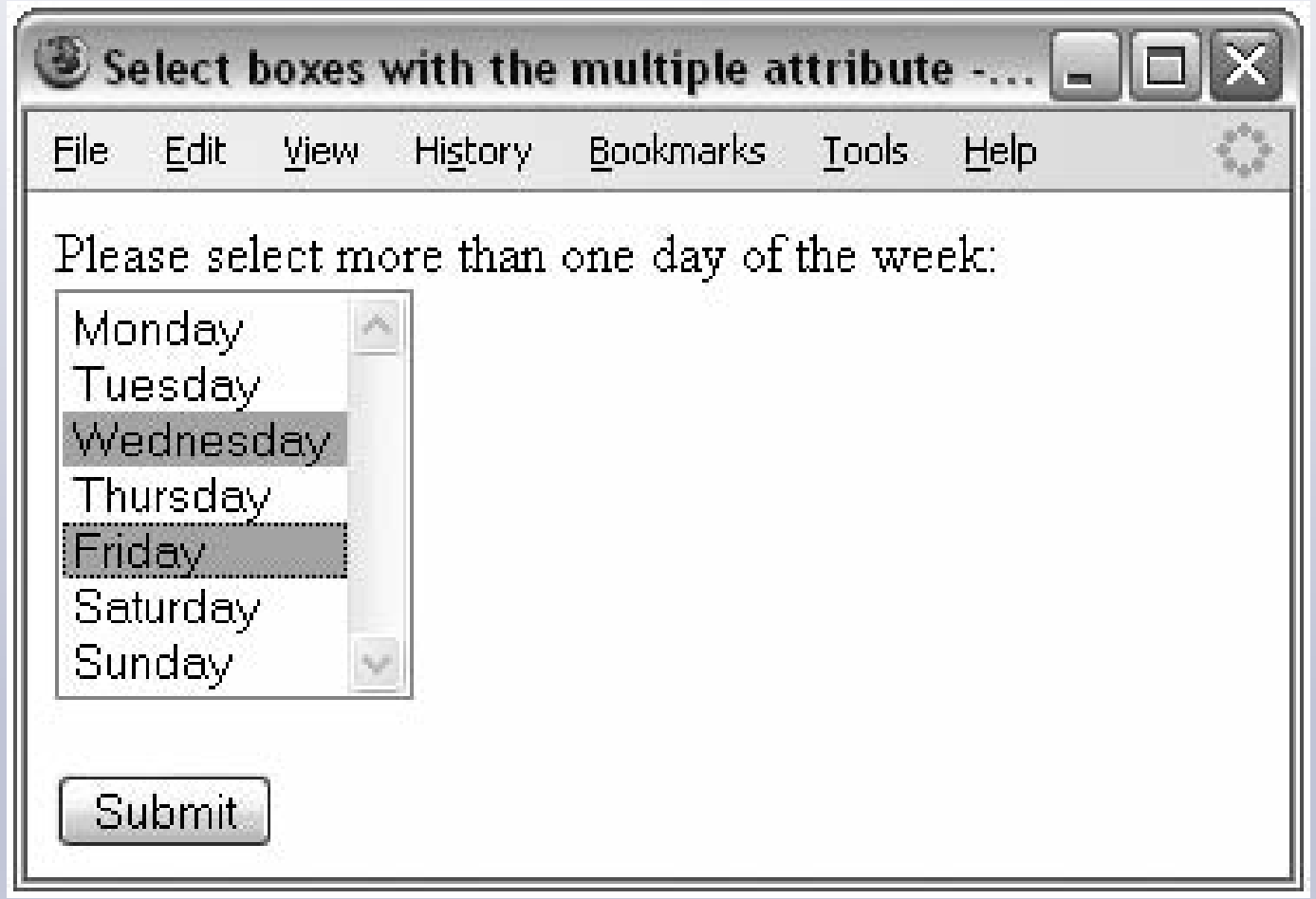
```
<form action="http://www.example.org/days.aspx" name="frmDays"
    method="get">
    <select size="4" name="selDay">
        <option value="Mon">Monday</option>
        <option value="Tue">Tuesday</option>
        <option value="Wed">Wednesday</option>
        <option value="Thu">Thursday</option>
        <option value="Fri">Friday</option>
        <option value="Sat">Saturday</option>
        <option value="Sun">Sunday</option>
    </select>
    <br /><br /><input type="submit" value="Submit" />
</form>
```



Selecting Multiple Options

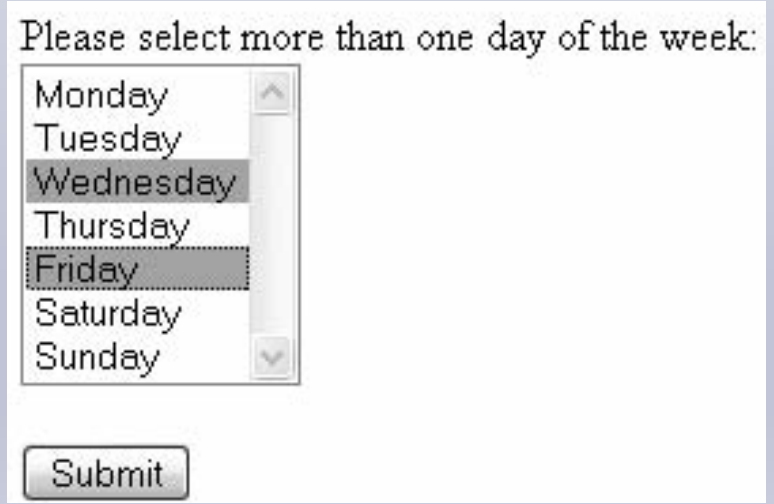
- The multiple attribute allows users to select more than one item from a select box.
- The value of the **multiple** attribute should be the word **multiple** in order for it to be valid XHTML.

Selecting Multiple Options Example



Selecting Multiple Options Example code

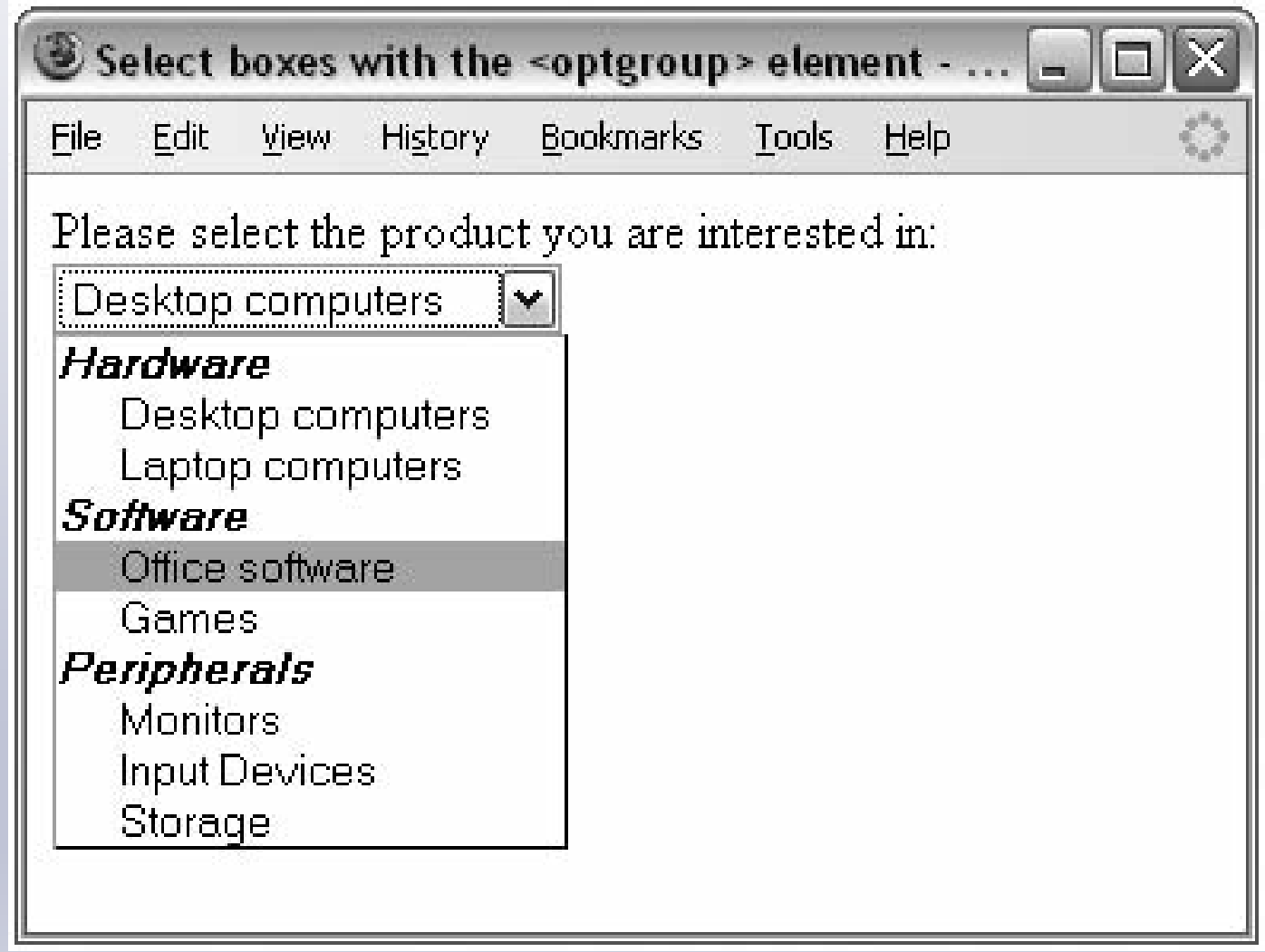
```
<form action="http://www.example.org/days.aspx" method="get"
    name="frmDays">
Please select more than one day of the week:<br />
    <select name="selDays" multiple="multiple">
        <option value="Mon">Monday</option>
        <option value="Tue">Tuesday</option>
        <option value="Wed">Wednesday</option>
        <option value="Thu">Thursday</option>
        <option value="Fri">Friday</option>
        <option value="Sat">Saturday</option>
        <option value="Sun">Sunday</option>
    </select>
    <input type="submit" value="Submit">
</form>
```



Grouping options: the <optgroup> element

- If you have a very long list of items in a select box, you can group them together using the **<optgroup>** element, which acts just like a container element for all the elements you want within a group.
- The **<optgroup>** element can carry a label attribute whose value is a label for that group of options

The <optgroup> Element Example



The <optgroup> Element Example code

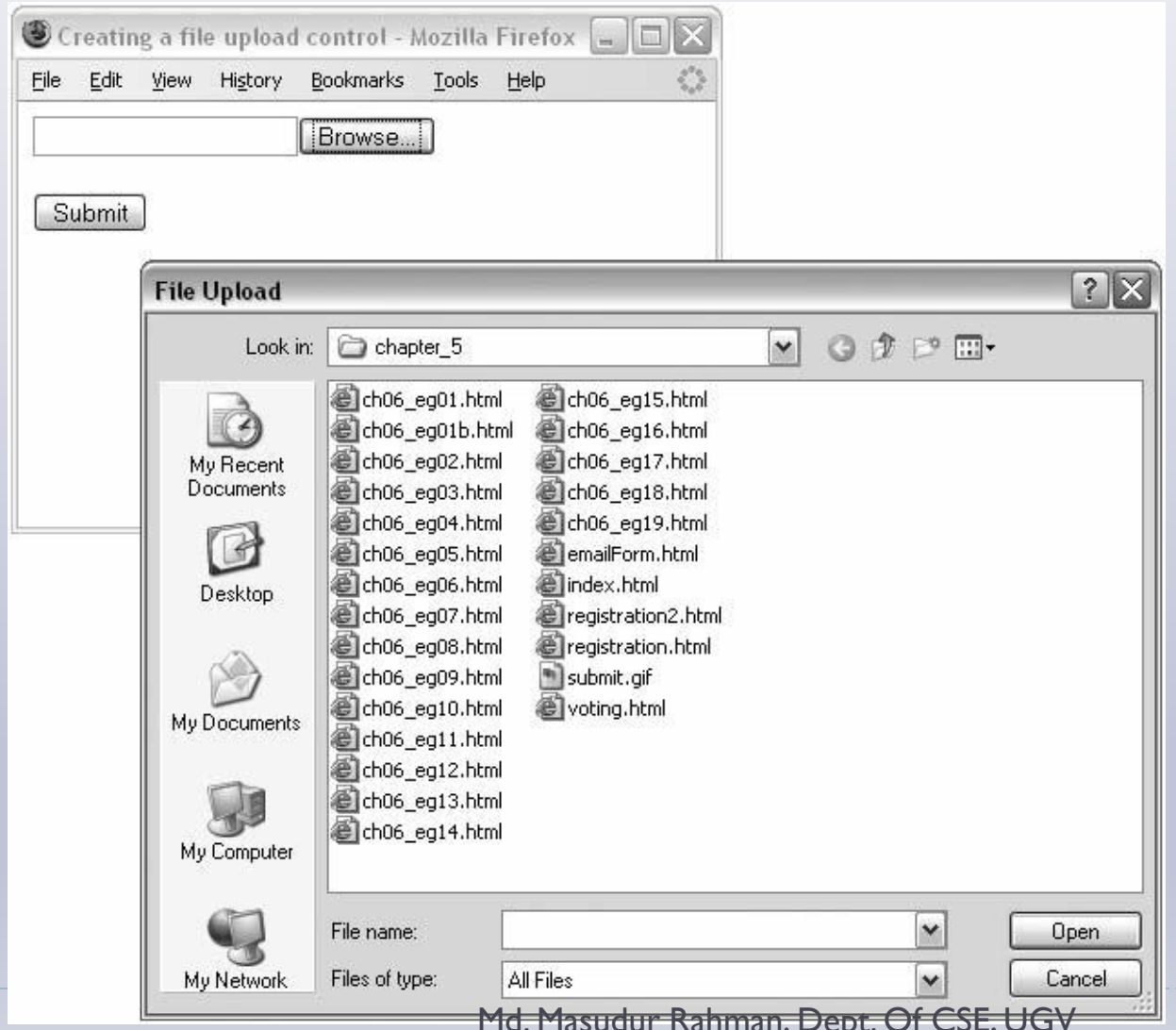
Please select the product you are interested in:

```
<select name="selInformation">
    <optgroup label="Hardware">
        <option value="Desktop">Desktop computers</option>
        <option value="Laptop">Laptop computers</option>
    </optgroup>
    <optgroup label="Software">
        <option value="OfficeSoftware">Office software</option>
        <option value="Games">Games</option>
    </optgroup>
    <optgroup label="Peripherals">
        <option value="Monitors">Monitors</option>
        <option value="InputDevices">Input Devices</option>
        <option value="Storage">Storage</option>
    </optgroup>
</select>
```

File Select Boxes

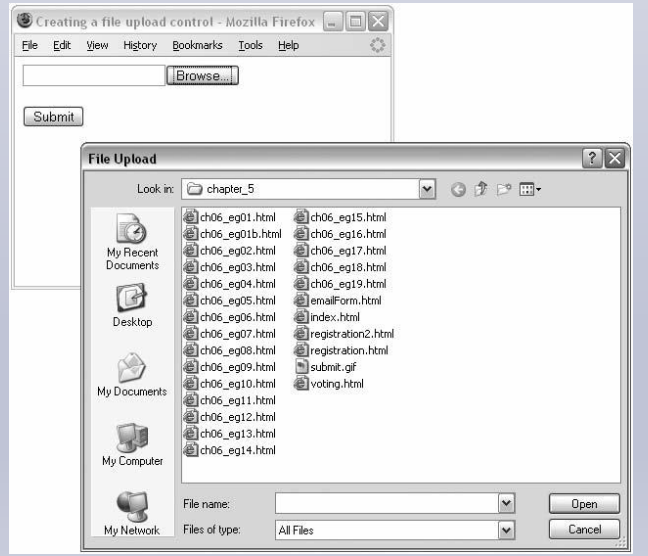
- Allows a user to upload a file to your web site from his or her computer.
- This is created using the **<input>** element with the type **attribute file**

File Select Boxes Example



File Select Boxes Example code

```
<form action="http://www.example.com/imageUpload.php"
   method="post"
name="fromImageUpload" enctype="multipart/form-data">
   <input type="file" name="fileUpload" accept="image/*" />
   <br /><br />
   <input type="submit" value="Submit" />
</form>
```



The enctype attribute

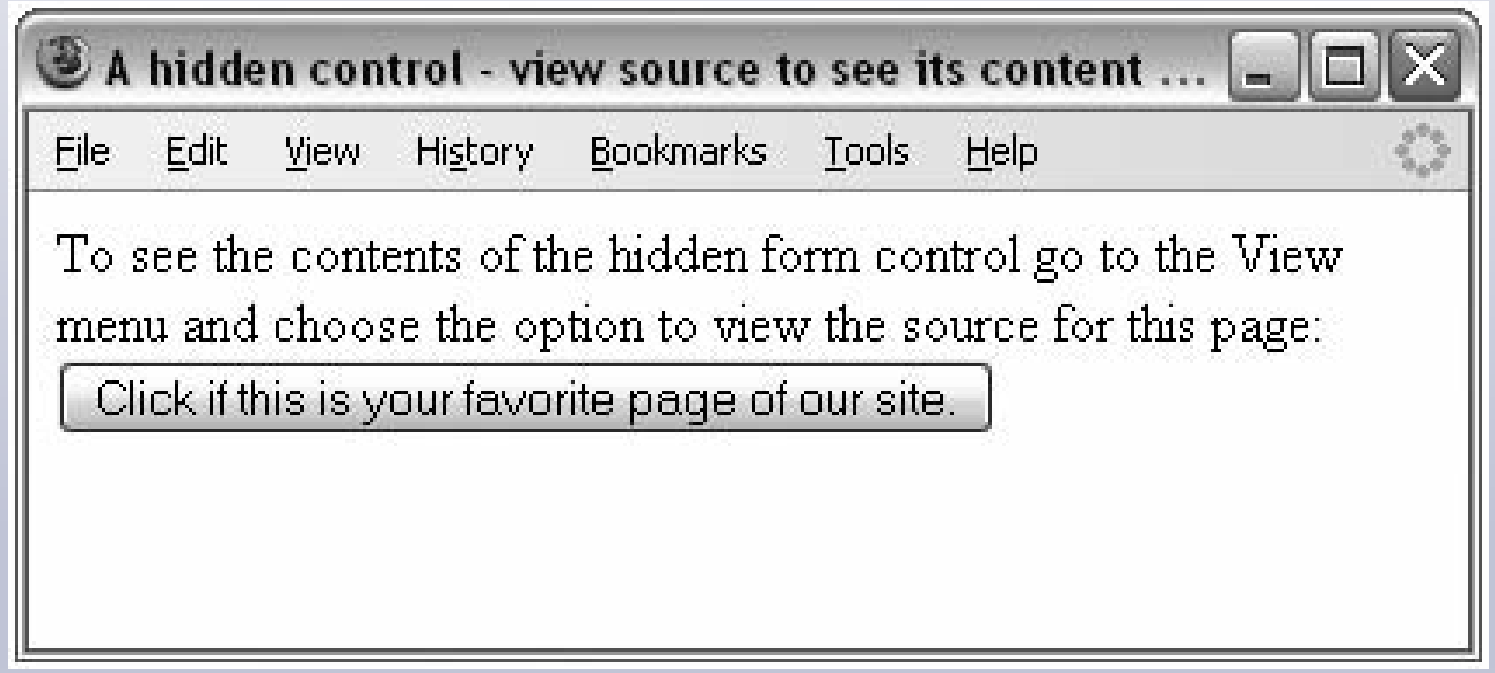
- The **enctype** attribute has been added to the <form> element with a value of **multipart/form-data** so that each form control is sent separately to the server.
- This is required on a form that uses a file upload box.

Hidden Controls

- They are used to pass information between pages without the user seeing it.
- Users cannot see them in the web page displayed in the browser,. However, if they were to look at the source code for the page they would be able to see the values in the code.
- For a name and value can still be sent to the server for a hidden form control, the hidden control must carry **name and value attributes**.

Hidden Controls (Cont'd)

- You create a hidden control using the <**input**> element whose type attribute has a value of **hidden**



```
<form action="http://www.example.com/vote.aspx"
  method="get" name="fromVote">
  <input type="hidden" name="hidPageSentFrom" value="home
    page" />
  <input type="submit" value="Click if this is your favorite page of
    our site." />
</form>
```

Web Programming

Week 08
Slides 266-289

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

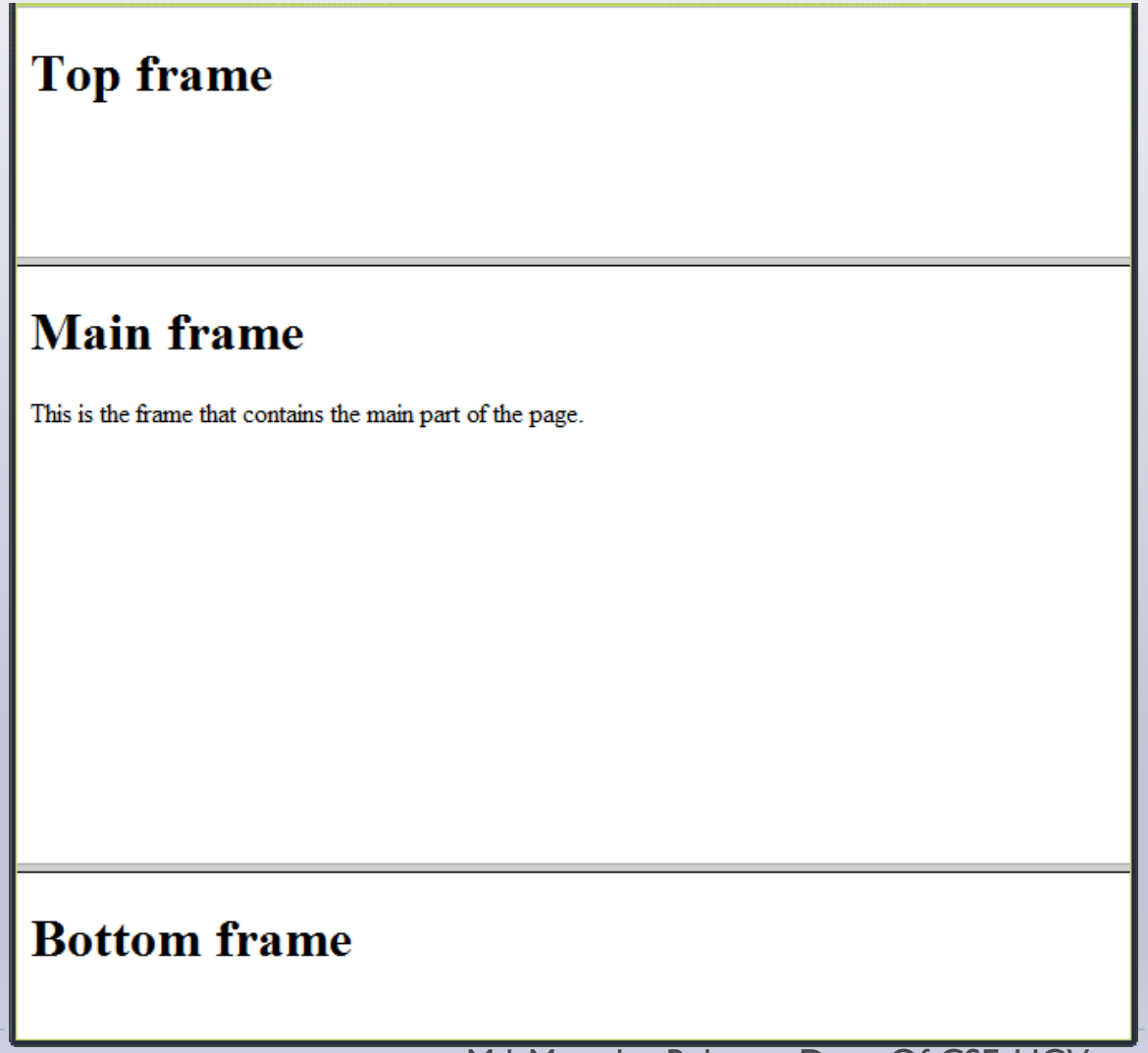
- Grouping Forms
- Introducing Frames
- The <frameset> element
- The <frame> element
- The <noframes> element
- Nested Frames
- Floating or Inline Frames with <iframe>
- Exercise

Introducing Forms

Introducing Frames

- Frames divide a browser into several separate pieces or panes.
- Each pane containing a separate XHTML page.
- A frameset is a collection of frames in the browser window.

Introducing the Frameset



Introducing the Frameset (Cont'd)

- To create a frameset document, first you need the <**frameset**> element, which is used instead of the <body> element.
- The frameset defines the rows and columns your page is divided into. Each frame is then represented by a <frame> element.
- The <**noframes**> element provides a message for users whose browsers do not support frames.

Introducing the Frameset (Cont'd)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html>
<head><title>Frames example</title></head>
<frameset rows="150, *, 100">
  <frame src="top_frame.html" />
  <frame src="main_frame.html" />
  <frame src="bottom_frame.html" />
  <noframes><body>This site uses a technology called frames.
    Unfortunately, your browser does not support this technology.
    Please Please upgrade your browser and visit us again!</body>
  </noframes>
</frameset>
</html>
```

Introducing the Frameset (Cont'd)

- There should not be markup between the closing </head> tag and the opening <frameset> tag.

- <frameset> attributes:
  - rows: rows of the frameset
  - cols: columns of the frameset

```
<frameset rows="150, *, 100">
```

  - Indicates that there are three rows:
    - The first row is 150 pixels high
    - The third row is 100 pixels high
    - The second row is taking the rest of the page.

Introducing the Frameset (Cont'd)

- <noframes>
  - It is used when the browser **does not** support frames.
  - A <body> element should be placed between those tags before introducing any text.
  - The text between the <body></body> tags will be shown.

Introducing the Frameset (Cont'd)

```
<frameset rows="1*, 2*, 3*">
```

- Indicates that there are three rows:
  - The first row takes one sixth of the page
  - The second row takes two sixth of the page.
  - The third row takes three sixth of the page.

Introducing the Frameset (Cont'd)

```
<frameset border="1">
```

- Indicates that the frameset will have a border of 1 pixel

The <frame /> element

- They are inside the <frameset> element
- Indicate a URL, using the **src** attribute, of the document that will be loaded into that frame

```
<frame src="top_frame.html" />
```

  - In this example, "top_frame.html" will be loaded in this frame.

- In this example, three separate documents are displayed:
  - top_frame.html
  - main_frame.html
  - bottom_frame.html

The <frame /> element (Cont'd)

<frame **name**="main_frame" />

- Indicates the name of a frame. It is used to indicate which frame a document should be loaded into.
- Specially important when you want to create links in one frame that load pages into a second frame.

The <frame /> element (Cont'd)

<frame **noresize**="**noresize**" />

- Prevents that a frame can be resized.
- Helps control the predetermined layout of the page

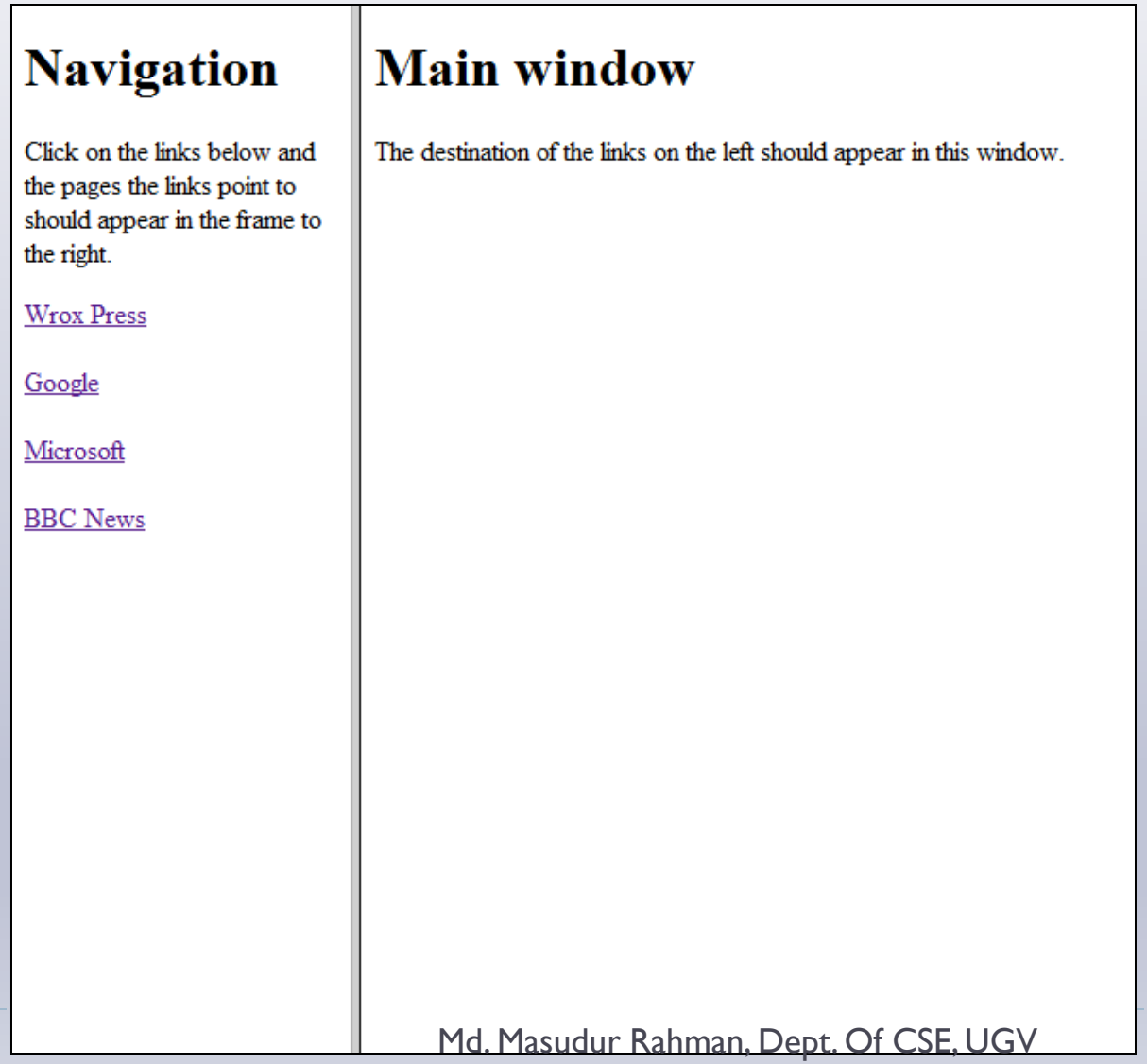
<frame **scrolling**="**yes**" />

- Allows the use of a scrollbar on the frame. Three values:
  - Yes: frame uses a scrollbar
  - No: frame does not use a scrollbar
  - Auto: browser should only include scrollbar when content does not fit in the frame.

Creating Links between Frames

- Use it when:
  - When your navigation bar is rather large in size (such as thumbnails of photographs in a gallery). By using frames, the user does not need to reload the navigation bar each time he or she views a new page.
  - When your main document is very long and the navigation bar provides shortcuts to parts of the main document (acting like a table of contents that is always in view).
  - When you do not want to reload the whole page.

Creating Links between Frames (Cont'd)



Creating Links between Frames (Cont'd)

```
<frameset cols="200, *">
  <frame src="frames/linksNav.html" />
  <frame src="frames/linksMain.html"
     name="main_page" />
</frameset>
```

- There are two columns, and only one row (by default)
- First column is 200 pixels. Second one takes remainder.
- First column has the navigation bar
- Second one has the main part of the page in which links will be loaded

linksNav.html

```
<a href="http://www.wrox.com"
  target="main_page">Wrox Press</a><br /><br />
<a href="http://www.google.com"
  target="main_page">Google</a><br /><br />
<a href="http://www.microsoft.com"
  target="main_page">Microsoft</a><br /><br />
<a href="http://news.bbc.co.uk/"
  target="main_page">BBC News</a><br /><br />
```

Creating Links between Frames (Cont'd)

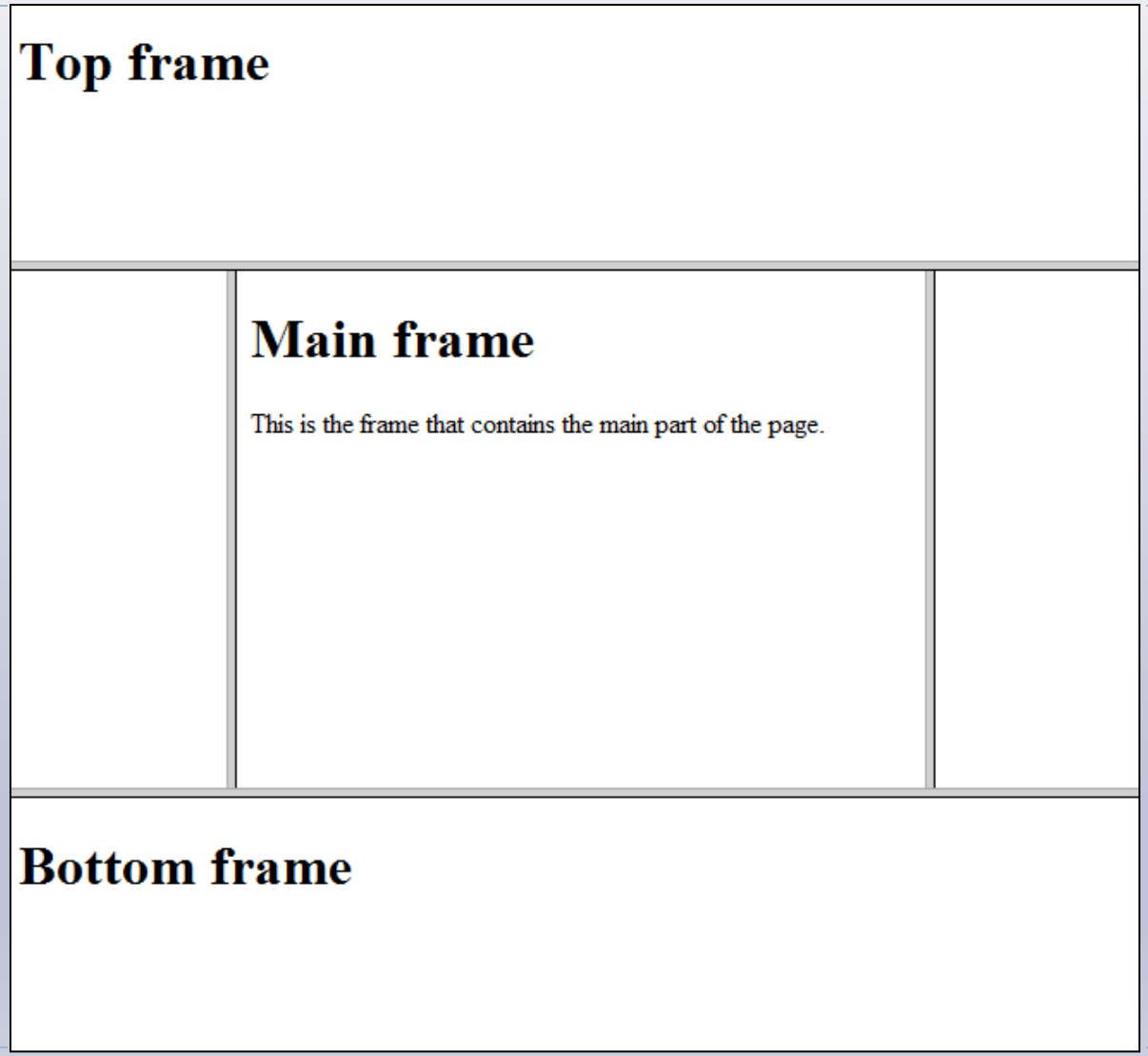
- The target attribute indicates where the links should be loaded.
- Also, the target attribute can take these values:
  - _self: Loads the page into the current frame
  - _blank: Loads a page into a new browser window
  - _parent: Loads the page into the parent window. In the case of a single frameset is the main browser window (and the page will replace all the frames), or in nested frames it replaces the frame that frameset lives inside.
  - _top: Loads the page into the browser window, replacing any current frames

Nesting Frames

- Just replace a <frame> element by a <frameset> element.

```
<frameset rows=”*, 300, *”>
   <frame src=”frames/top_frame.html” />
   <frameset cols=”*, 400, *”>
      <frame src=”frames/blank.html” />
      <frame src=”frames/main_frame.html” />
      <frame src=”frames/blank.html” />
   </frameset>
   <frame src=”frames/bottom_frame.html” />
</frameset>
```

Nesting Frames (Cont'd)



When to use frames?

- Only in very few and extremely needed circumstances. In other words, very **rarely**.
- Other few cases:
  - When you want to display a lot of content in one single page and you cannot split the document into separate pages
  - When you have a lot of data in one part of the page that you do not want the user to have to reload while another part of the page changes.
    - i.e.: a photography site where you have lots of thumbnails in one frame, and the main picture in another

Frames: Pros and Cons

- Advantages:
  - Load and reload single panes without having to reload the entire contents of the browser window.
  - More than one HTML document can be displayed in the same browser
  - Each frame is independent of the others

Frames: Pros and Cons (Cont'd)

- Disadvantages*:
  - Frames will not be supported in future versions of HTML
  - Frames are difficult to use
  - The web developer must keep track of more HTML documents
  - The browser's back button might not work as expected
  - Search engines will not index properly frameset sites

* www.w3schools.com

Web Programming

Week 09
Slides 290-322

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

- Introducing CSS
- CSS Properties
- Controlling Fonts
- Text Formatting
- Exercise

Introducing CSS

Introducing CSS

- CSS works by allowing you to associate *rules with the elements that appear in the document.*

- The ***selector***, *which indicates which element or elements the declaration applies to (if it applies to* more than one element, you can have a comma-separated list of several elements)
- The ***declaration***, *which sets out how the elements should be styled*

Introducing CSS (Cont'd)

The declaration is also split into two parts, separated by a colon:

- A ***property***, *which is the property of the selected element(s) that you want to affect, in this case* the font-family property.

- A ***value***, *which is a specification for this property; in this case it is the Arial typeface*

Introducing CSS: Example

```
h1, h2, h3 {
  font-weight: bold;
  font-family: arial, verdana, sans-serif;
  color: #000000;
  background-color: #FFFFFF;}
```

- The h1, h2, and h3 elements will be written in a **bold** Arial font, and this will be written in black with a white background

Basic Example: Hands on!

- Open:
  - Lec09_eg01.html
- We are going to create a CSS to apply on that document. The rules to apply are:

```
body {
  color:#000000; /*font color: black*/
  background-color:#ffffff; /*background color: white*/
  font-family: arial, verdana, sans-serif; }
h1 {font-size:18pt;} /*font size for h1*/
p {font-size:12pt;} /*font size for p*/
```

Basic Example: Hands on!

```
table {
  background-color:#efefef;
  border-style:solid;
  border-width:1px;
  border-color:#999999;}

  th {
    background-color:#cccccc;
    font-weight:bold;
    padding:5px;} /*space between content and
    edge of cell*/
```

Basic Example: Hands on!

```
td {padding:5px;}

td.code {
  font-family:courier, courier-new, serif;
  font-weight:bold;}
```

Inheritance

- One of the powerful features of CSS is that many of the properties that have been applied to one element will be ***inherited*** *by child elements*.
- **Child elements** are contained within the element that the rules were declared upon.
- The rules applied to a child element will **overwrite** any rule apply to its parent element.

How to add CSS rules

- In two places inside the XHTML document:
  - Inside the <head> element, contained with a <style> element
  - As a value of a style attribute on any element that can carry the style attribute
- In a separate style sheet, or *external style sheet* using the <link> element.

How to add CSS rules (Cont'd)

- When the style sheet rules are held inside a <style> element in the head of the document, they are referred to as an ***internal style sheet***.

```
<head>
  <title>Internal Style sheet</title>
  <style type="text/css">
    body {
      color:#000000;
      background-color:#ffffff;
      font-family:arial, verdana, sans-serif; }
      h1 {font-size:18pt;}
      p {font-size:12pt;}
  </style>
</head>
```

How to add CSS rules (Cont'd)

- When style attributes are used on XHTML elements, they are known as ***inline style rules***. *For example:*

```
<td style="font-family:courier;
  padding:5px; border-style:solid;
  border-width:1px; border-
  color:#000000;">
```

However, this attribute has been deprecated in Transitional XHTML!

The <link> element

- The **<link />** element is always an ***empty element*** that describes the relationship between two documents.
- It can be used in several ways, not just style sheets.
- the <link /> element must carry three attributes: type, rel, and href.

The <link> element (Cont'd)

```
<link rel="stylesheet" type="text/css"
    href="../css/interface.css"  />
```

- The **rel** attribute is **required** and specifies the relationship between the document containing the link and the document being linked to.
- The **type** attribute specifies the MIME type of the document being linked to.
- The **href** attribute specifies the URL for the document being linked to.

Advantages of External CSS

- The same style sheet can be *reused by all of the web pages in your site.*
- Because the style rules are written only once, rather than appearing on every element or in every document, the source documents are smaller
- You can change the appearance of several pages by altering just the style sheet rather than each individual page.
- The style sheet can act as a style template to help different authors achieve the same style of document without learning all of the individual style settings.

CSS Properties

FONT	FONT *(continued)*	TEXT *(continued)*	TEXT *(continued)*
font	font-variant	text-align	white-space
font-family	font-weight	text-decoration	word-spacing
font-size	**TEXT**	text-indent	**BACKGROUND**
font-size-adjust	color	text-shadow	background
font-stretch	direction	text-transform	background-attachment
font-style	letter-spacing	unicode-bidi	background-color

CSS Properties (Cont'd)

BACKGROUND *(continued)*
background-image
background-position
background-repeat
BORDER
border
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-color
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
border-style
border-top
border-top-color
BORDER *(continued)*
border-top-style
border-top-width
border-width
MARGIN
margin
margin-bottom
margin-left
margin-right
margin-top
PADDING
padding
padding-bottom
padding-left
padding-right
padding-top
DIMENSIONS
height
line-height
max-height
max-width
min-height

CSS Properties (Cont'd)

DIMENSIONS *(continued)*
min-width
width
POSITIONING
bottom
clip
left
overflow
right
top

vertical-align
z-index
OUTLINES
outline
outline-color
outline-style
outline-width
TABLE
border-collapse
border-spacing
caption-side
empty-cells

GENERATED CONTENT
content
counter-increment
counter-reset
quotes
CLASSIFICATION
clear
cursor
display
float
position
visibility

TABLE *(continued)*
table-layout
LIST and MARKER
list-style
list-style-image
list-style-position
list-style-type
marker-offset

Controlling Fonts

Several properties allow you to control the appearance of text in your documents. These can be split into two groups:

- Those that **directly affect** the font and its appearance
- Those that have other formatting effects upon the text

Properties Directly Affecting the Font

Property	Purpose
font	Allows you to combine several of the following properties into one
font-family	Specifies the family of font to be used (the user must have this installed on his or her computer)
font-size	Specifies the size of a font
font-weight	Specifies whether the font should be normal, bold, or bolder than the containing element
font-style	Specifies whether the font should be normal, italic, or oblique (an oblique font is the normal font on a slant rather than a separate italic version of the font)
font-stretch	Allows you to control the width of the actual letters in a font (not spaces between them)
font-variant	Specifies whether the font should be normal or small caps
font-size-adjust	Allows you to alter the aspect ratio of the size of characters of the font

The font-size property

- The font-size property enables you to specify a size for the font.
- A value for this property can be specified in:
  - Absolute size
    - xx-small x-small small medium large x-large xx-large
  - Relative size
    - smaller larger
  - Length
    - Any of these units of length: **px** em ex pt in cm pc mm
  - Percentage (in relation to parent element)
    - 2% 10% 25% 50% 100%

The font-weight Property

The possible values for font-weight are:

- normal bold bolder lighter 100 200 300 400 500 600 700 800 900

For example:

```
p.one {font-weight:normal;}
p.two {font-weight:bold;}
p.three {font-weight:bolder;}
p.four {font-weight:lighter;}
p.five {font-weight:100;}
p.six {font-weight:200;}
```

The font-style Property

- The font-style property allows you to specify that a font should be normal, italic, or oblique.
- For example:

```
p.one {font-style:normal;}
p.two {font-style:italic;}
p.three {font-style:oblique;}
```

Text Formatting

- In addition to the font properties, you can use several properties to affect the appearance or formatting of your text.

Property	Purpose
`color`	Specifies the color of the text
`text-align`	Specifies the alignment of the text within its containing element
`vertical-align`	Vertical alignment of text within containing element and in relation to containing element
`text-decoration`	Specifies whether the text should be underlined, overlined, strikethrough, or blinking text

The color property

- The color property allows you to specify the color of the text.
- For example:

```
p {color:#ff0000;} /*Paragraph is red*/
```

The text-align Property

- It aligns the text within its containing element or the browser window

Value	Purpose
`left`	Aligns the text with the left border of the containing element
`right`	Aligns the text with the right border of the containing element
`center`	Centers the content in the middle of the containing element
`justify`	Spreads the width across the whole width of the containing element

The vertical-align Property

- It allows you to control their vertical positioning within the containing element.
- It is useful when working with inline elements, in particular images and portions of text

For example:

- `span.footnote {vertical-align:sub;}`

The vertical-align Property

Value	Purpose
`baseline`	Everything should be aligned on the baseline of the parent element (this is the default setting).
`sub`	Makes the element subscript. With images, the top of the image should be on the baseline. With text, the top of the font body should be on the baseline.
`super`	Makes the element superscript. With images, the bottom of the image should be level with the top of the font. With text, the bottom of the descender (the parts of letters such as *g* and *p* that go beneath the line of text) should align with the top of the font body.
`top`	The top of the text and the top of the image should align with the top of the tallest element on the line.

The vertical-align Property (Cont'd)

Value	Purpose
`text-top`	The top of the text and the top of the image should align with the top of the tallest text on the line.
`middle`	The vertical midpoint of the element should be aligned with the vertical midpoint of the parent.
`bottom`	The bottom of the text and the bottom of the image should align with the bottom of the lowest element on the line.
`text-bottom`	The bottom of the text and the bottom of the image should align with the bottom of the lowest text on the line.

The word spacing Property

- The word-spacing property sets the gap between words

```
span.wider {word-spacing:20px;}
```

Text Pseudo Classes

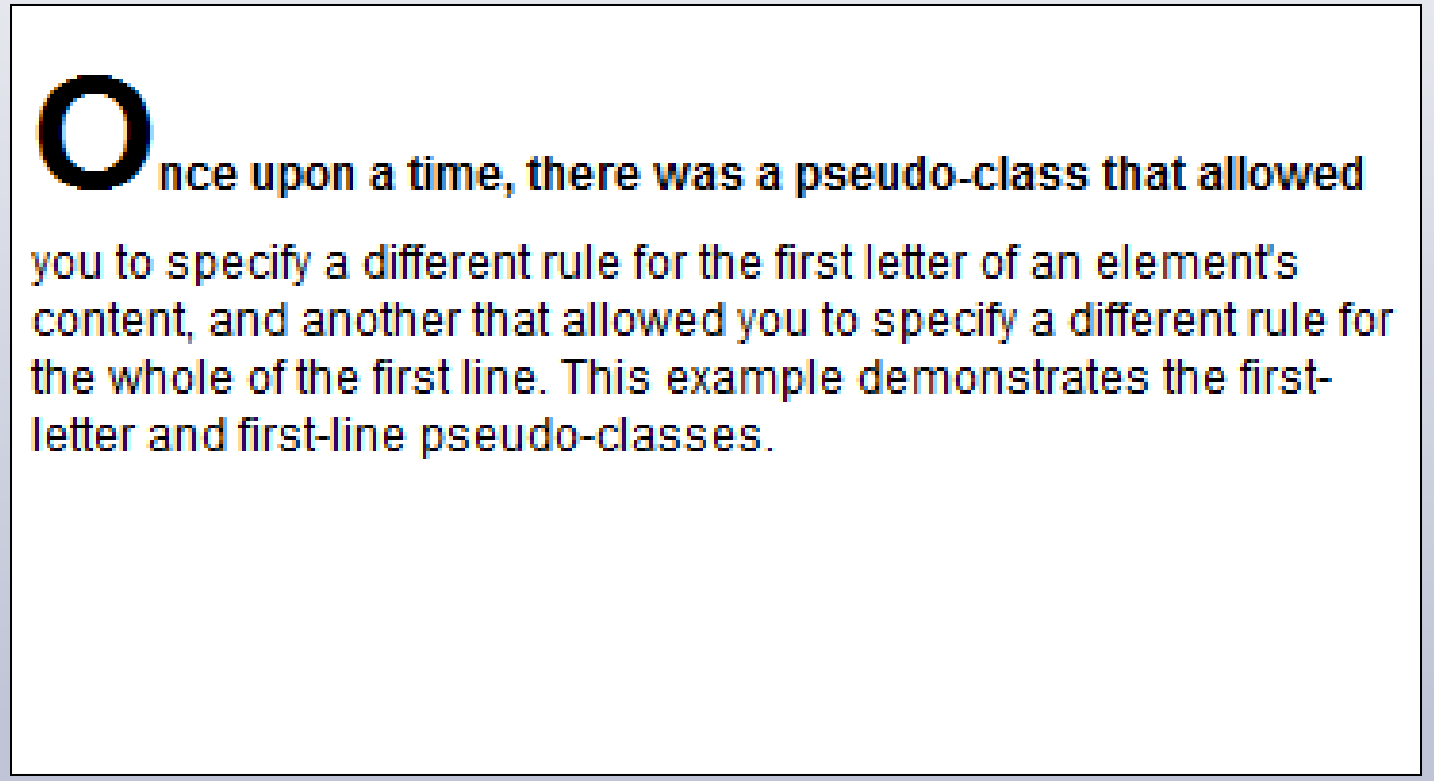
- They allow you to render either the first letter or the first line of an element in a different way than the rest of that element.
  - The first letter Pseudo-Class: (first-letter)
    - Allows you to specify a rule just for the first letter of an element

```
p.pageOne:first-letter {font-size:42px;}
```

  - The first line Pseudo-Class: (first-line)
    - allow you to render the first line of any paragraph differently from the rest of the paragraph.

```
p:first-line {font-weight:bold;}
```

Text Pseudo Classes: Example



Web Programming

Week 10
Slides 323-375

Md. Masudur Rahman, Dept. Of CSE, UGV

A Deeper look at CSS

Md. Masudur Rahman, Dept. Of CSE, UGV

Universal Selector

- The *universal selector is an asterisk; it is like a wildcard and matches all element types in the document.*

```
* { }
```

The Type Selector

- The type selector matches all of the elements specified in the comma-delimited list.
- It allows you to apply the same rules to several elements.

```
h1, h2, p { }
```

- If you have the same rules that apply to several elements, this technique can lead to a **smaller** style sheet, **saving** bandwidth and load on your server.

The Class Selector

- The class selector allows you to match a rule with an element carrying a class attribute whose value you specify in the class selector.

- For example:

```
<p class="BackgroundNote">
        This paragraph contains an aside.
</p>
```

The Class Selector (Cont'd)

You can use a class selector in one of two ways here:

1. Assign a rule that applies to **any element** that has a class attribute whose value is **BackgroundNote**, like so, simply preceding the value of the class attribute with a period:

.BackgroundNote {}

2. Or Create a selector that selects **only the <p> elements** that carry a class attribute with a value of BackgroundNote (not other elements) like so:

p.BackgroundNote {}

The ID Selector

- The id selector works just like a class selector, but works on the value of id attributes.
- Rather than using a period before the value of the id attribute, a hash or pound sign (#) is required.
- For example:

```
<p id="abstract"> A Paragraph</p>
```

- can be identified with this selector.

```
p#abstract { }
```

- Because the value of an id attribute should be unique within a document, this selector should apply only to the content of one element.

The Child Selector

- The child selector matches an element that is a **direct child** of another.
- For example:

td>b { }

- In this case it matches any <b> elements that are direct children of <td> elements

The Descendent Selector

- The descendent selector matches an element type that is a descendent of another specified element, at any level of nesting.
- For the descendent selector the combinator is the space. For example:

```
table  b { }
```

- In this case, the selector matches any <b> element that is a child of the <table> element, which means it
- would apply to <b> elements both in <td> and <th> elements.

The Adjacent Sibling Selector

- An adjacent sibling selector matches an element type that is the next sibling of another.
- For example:

```
h1+p { }
```

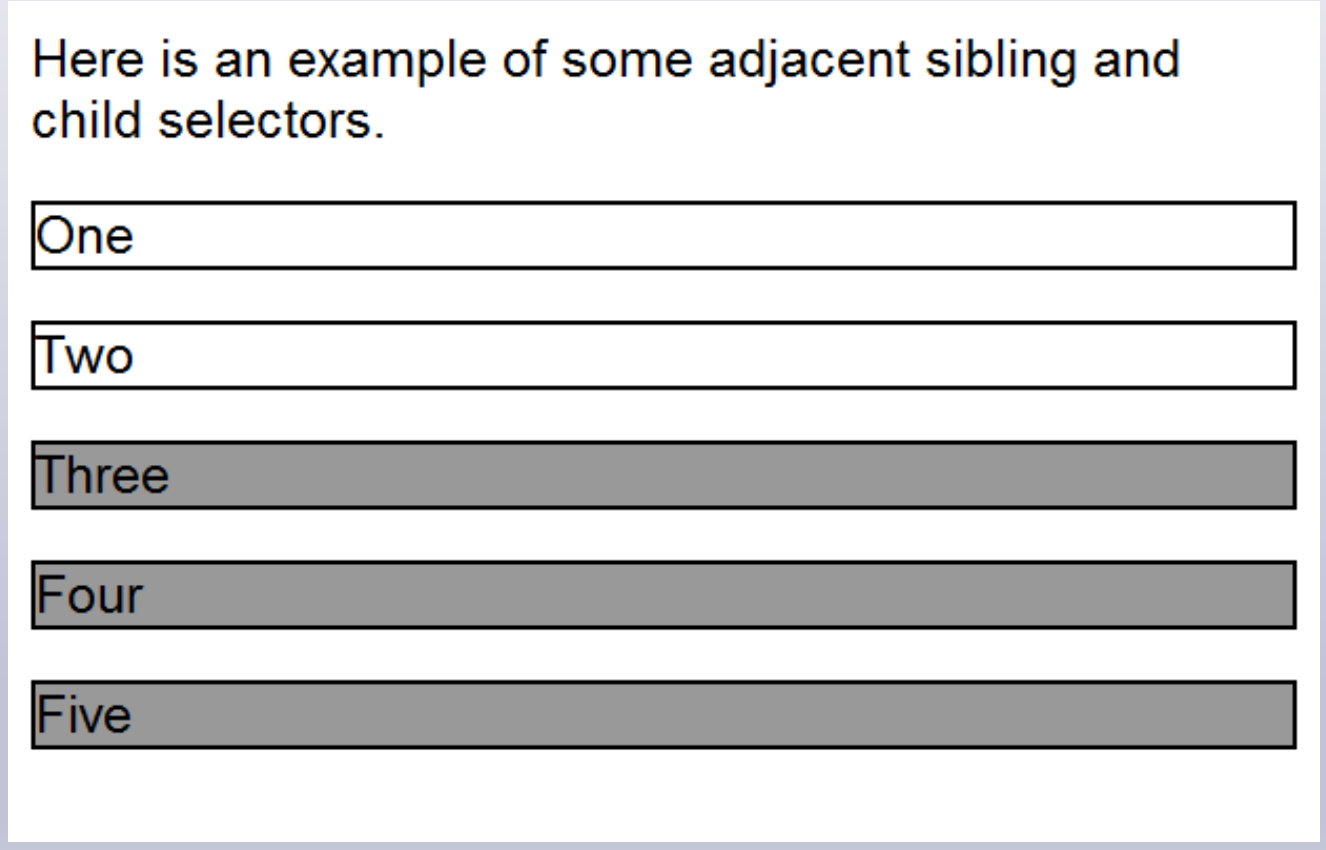
- This makes the **first paragraph <u>after any</u> level 1 heading** a different style.
- For this example, this would apply only to the <p> element directly after a heading.

Using Child and Adjacent Sibling Selector

- They both are very important because they can reduce the number of class attributes you need to add into an XHTML document.

Using Child and Adjacent Sibling Selector (Cont'd)

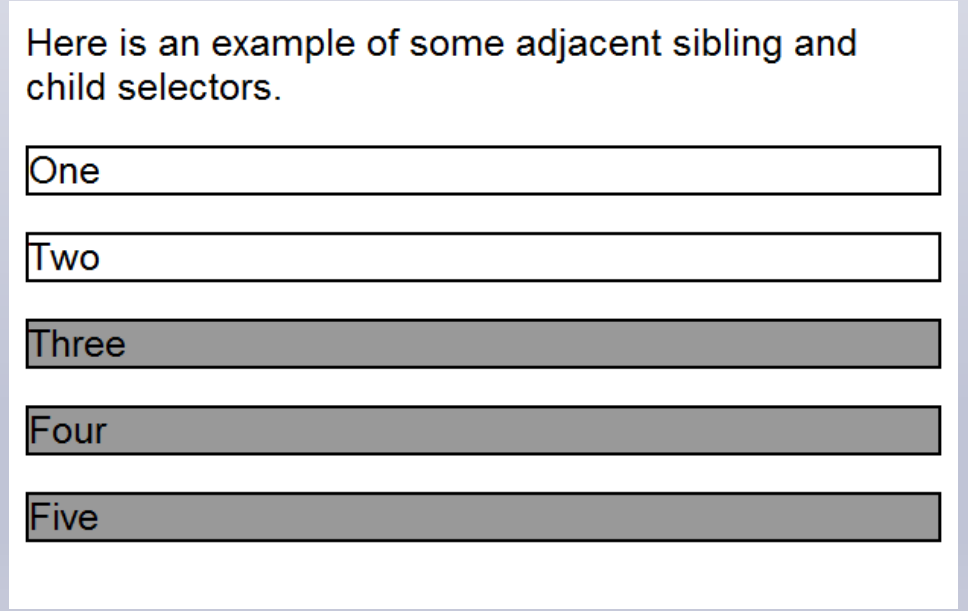
- Look at the following example:



Using Child and Adjacent Sibling Selector (Cont'd)

The three different paragraph styles are as follows:

- The first paragraph has no border or background color.
- The paragraphs inside the element all have borders.
- The last three paragraphs have a gray background.



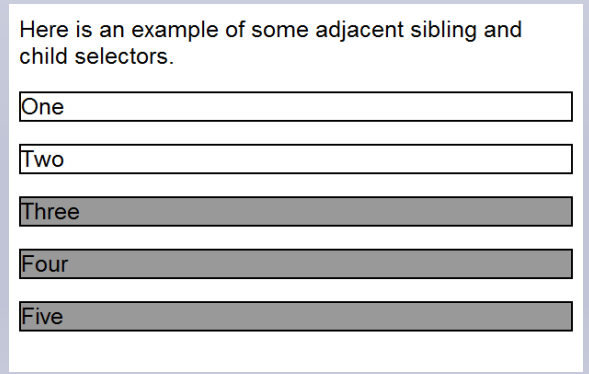
Without Child and Adjacent Sibling Selectors

XHTML:

```
<p>Here is an example of
  some adjacent sibling and
  child selectors.</p>
<div>
  <p class="style1">One</p>
  <p class="style1">Two</p>
  <p class="style2">Three</p>
  <p class="style2">Four</p>
  <p class="style2">Five</p>
</div>
```

CSS:

```
p.style1{
  border:1px solid black;
}

p.style2{
  border:1px solid black;
  background-color: gray;
}
```



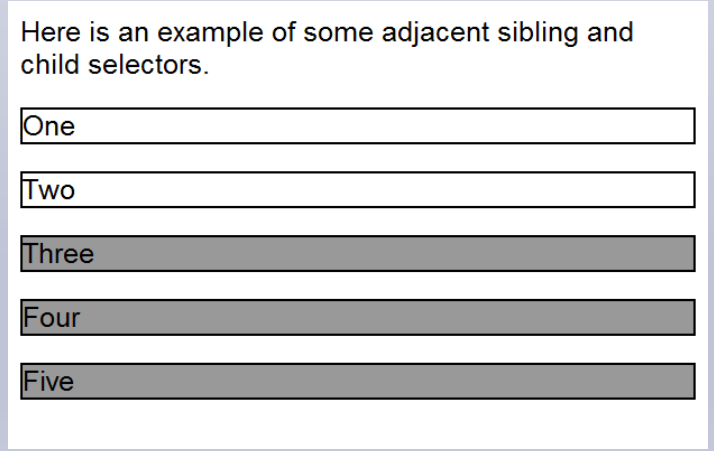
Using Child and Adjacent Sibling Selector (Cont'd)

XHTML:

```
<p>Here is an example of
  some adjacent sibling
  and child selectors.</p>
<div>
  <p>One</p>
  <p>Two</p>
  <p>Three</p>
  <p>Four</p>
  <p>Five</p>
</div>
```

CSS:

```
div>p {
border:1px solid black;}

p+p+p {
  background-color:gray;}
```



Attribute Selectors

- Attribute selectors enable you to use the attributes that an element carries in the selector.
- You can use attribute selectors in several ways, as shown in the next table:

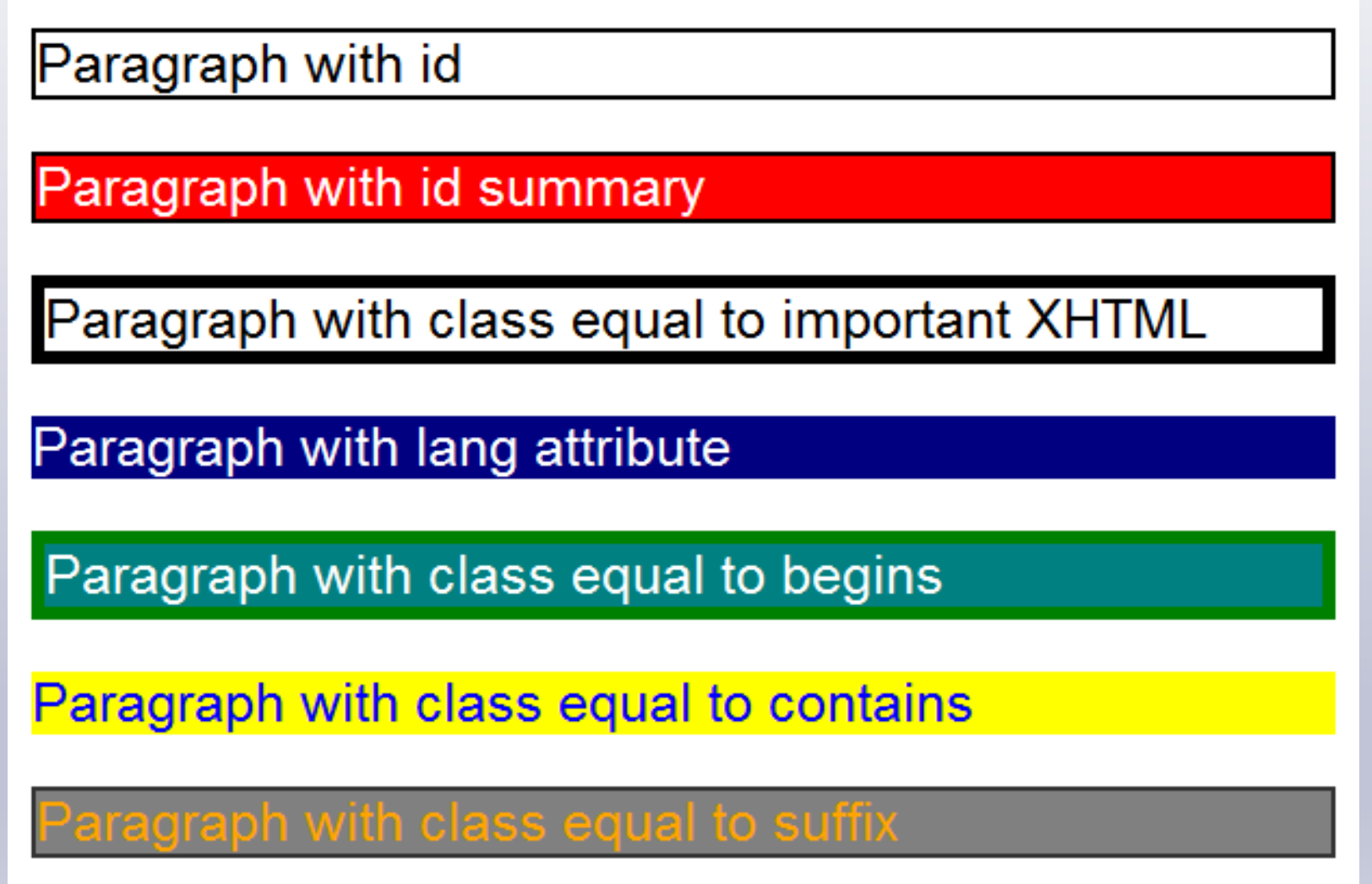
Attribute Selectors (Cont'd)

Name	Example	Matches
Existence selector	p[id]	Any <p> element carrying an attribute called id
Equality selector	p[id = "summary"]	Any <p> element carrying an attribute called id whose value is summary
Space selector	p[class ~= "XHTML"]	Any <p> element carrying an attribute called class, whose value is a list of space-separated words, one of which is exactly the same as XHTML.
Hyphen selector	p[language |= "en"]	Any <p> element carrying an attribute called language whose value begins with en and is followed with a hyphen (it is designed for use with language attributes).

Attribute Selectors (Cont'd)

Name	Example	Matches
Prefix selector (CCS3)	p[attr^="b"]	Any <p> element carrying any attribute whose value begins with b.
Substring selector (CSS3)	p[attr*="on"]	Any <p> element carrying any attribute whose value contains the letters on.
Sufix selector (CSS3)	p[attr$="x"]	Any <p> element carrying any attribute whose value contains the ends in the letter x.

Attribute Selectors Example



Attribute Selectors Example: XHTML

```
<p id="introduction">Paragraph with id</p>
<p id="summary">Paragraph with id summary</p>
<p class="important XHTML">Paragraph with class
  equal to important XHTML</p>
<p lang="en-us">Paragraph with lang attribute</p>
<p class="begins">Paragraph with class equal to
  begins</p>
<p class="contains">Paragraph with class equal to
  contains</p>
<p class="suffix">Paragraph with class equal to
  suffix</p>
```

Attribute Selectors Example: CSS

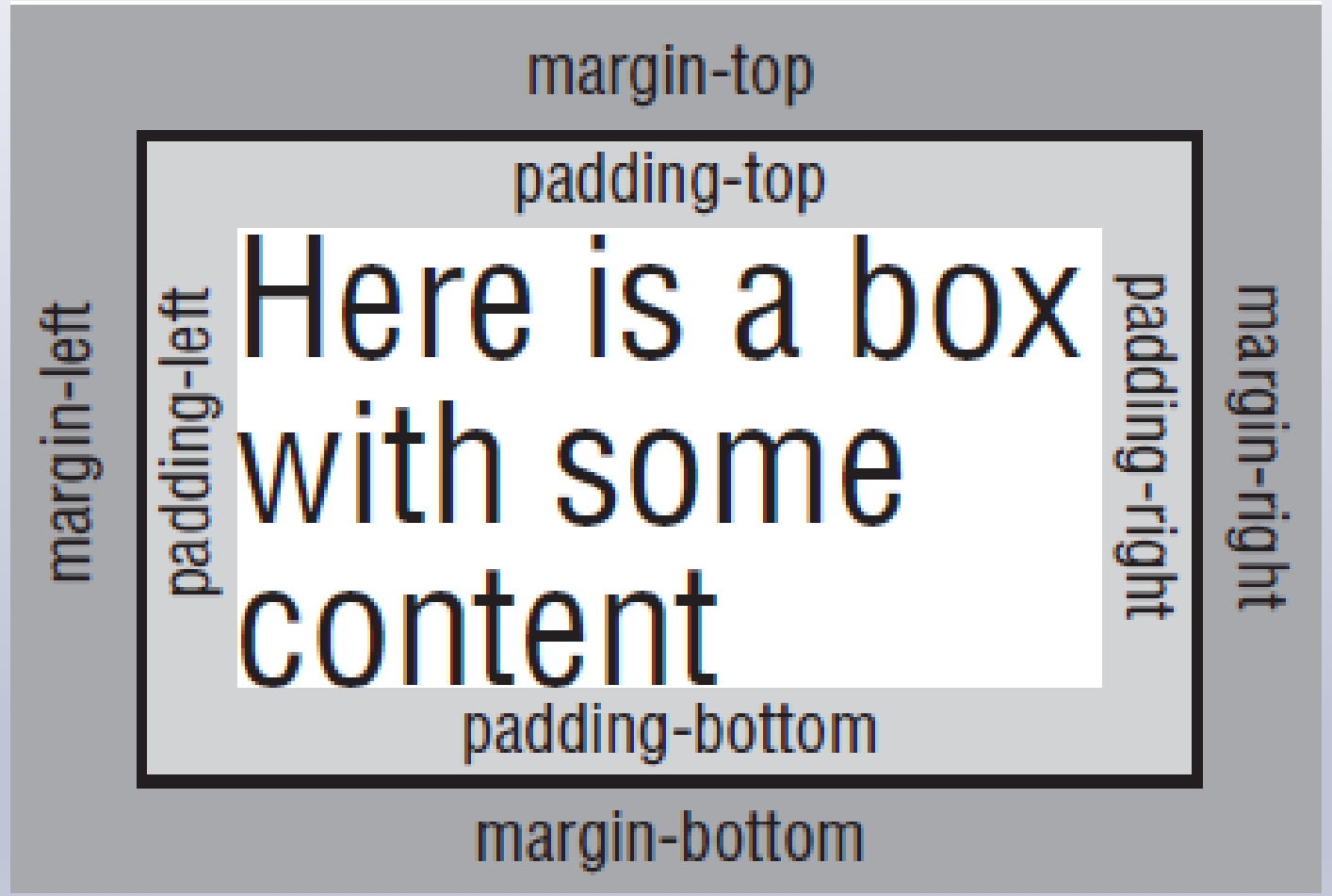
```
p[id] {border:1px solid #000000;}
p[id=summary] {background-color:#999999;}
p[class~=XHTML] {border:3px solid #000000;}
p[lang|="en"] {color:white; background-
  color:black;}
p[class^="b"] {border:3px solid #333333;}
p[class*="on"] {color:#ffffff; background-
  color:#333333;}
p[class$="x"] {border:1px solid #333333;}
```

The Box Model

- Every element gets treated as a *box in CSS*
- *Remembering this will really help you understand how to* create attractive layouts with CSS.

Property	Description
border	Every box has a border (even if you do not see it) Separates the edge of the box from other boxes
margin	Distance between the edge of a box and the box next to it
padding	Space between the content of the box and its border

The Box Model (Cont'd)



The Box Model (Cont'd)

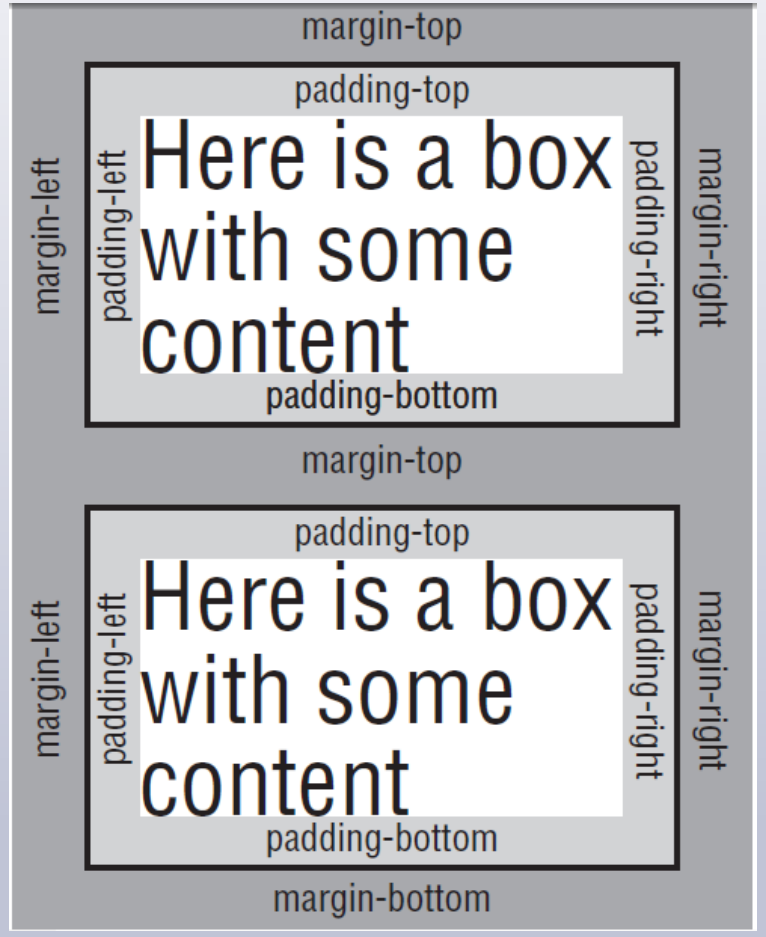
- The **padding** and **margin** properties are especially important in creating *white space, which is the space* between parts of the page, in your designs

The Box Model (Cont'd)

Note that:

- When a bottom margin of one element meets the top margin of another, only the **larger of the two** will show
- If they are the **same size**, the margin will be equivalent to the size of just one of the margins.
- See next figure.

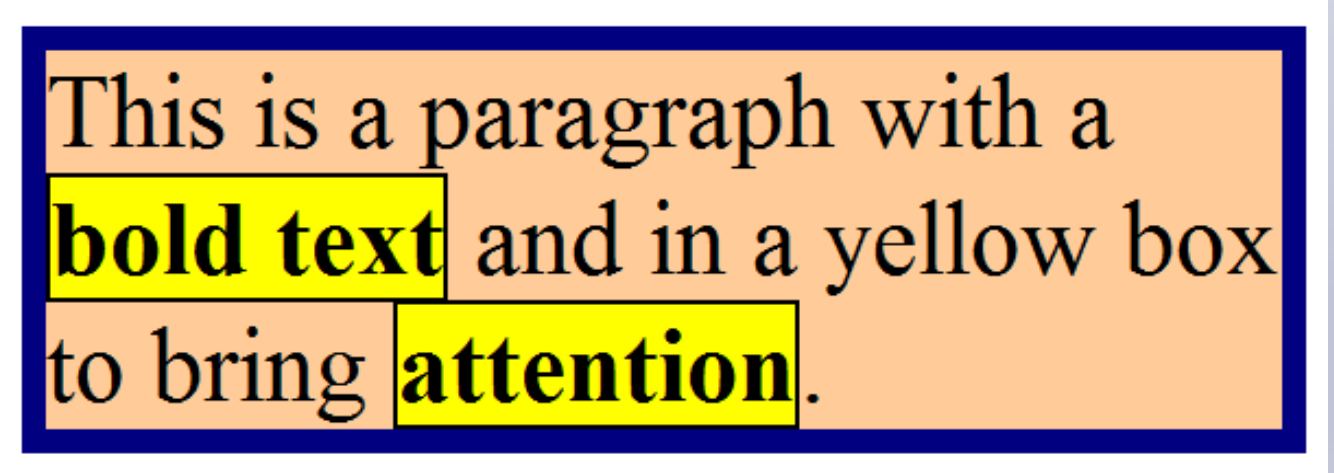
The Box Model (Cont'd)



Vertical margins of two adjacent boxes collapsing

The Box Model (Cont'd)

- A **block element** will also, by default, take up the **full width** of the page for the box model (or the element it is contained within)
- An **inline element** will take up only **as much space as it needs**.



The Border Properties

- The border properties allow you to specify how the border of the box representing an element should look.
- There are three properties of a border you can change:
  - **border-color:** to indicate the color a border should be
  - **border-style:** to indicate whether a border should be a solid, dashed, or double line, or one of the other possible values
  - **border-width:** to indicate the width a border should be

border-color property

- The border-color property allows you to change the color of the border surrounding a box.
- The value can be a hex code for the color or a color name
- It can also be expressed as values for red, green, and blue; between 0 and 255; or percentages of red green and blue.

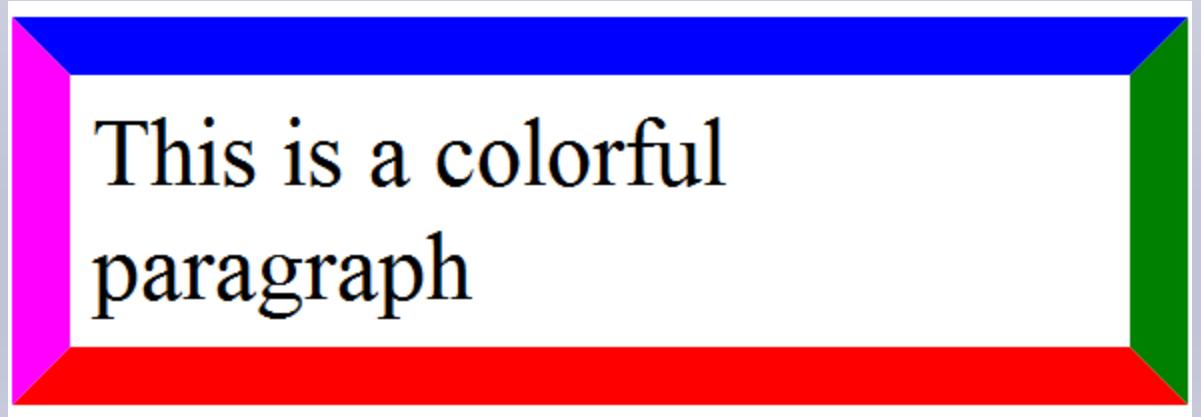
Color Name	hex	RGB Values	RGB Percentages
red	#ff0000	rgb(255, 0, 0)	rgb(100%, 0, 0)
green	#00ff00	rgb(0, 255, 0)	rgb(0, 100%, 0)
blue	#0000ff	rgb(0, 0, 255)	rgb(0, 0, 100%)

border-color property

- The color of the bottom, left, top, and right sides of a box's border can be changed individually using the following properties:
  - border-bottom-color
  - border-right-color
  - border-top-color
  - border-left-color

border-color property

```
p {
  border-bottom-color: red;
  border-top-color: blue;
   border-right-color: green;
   border-left-color: fuchsia;
  border-style: solid;
}
```



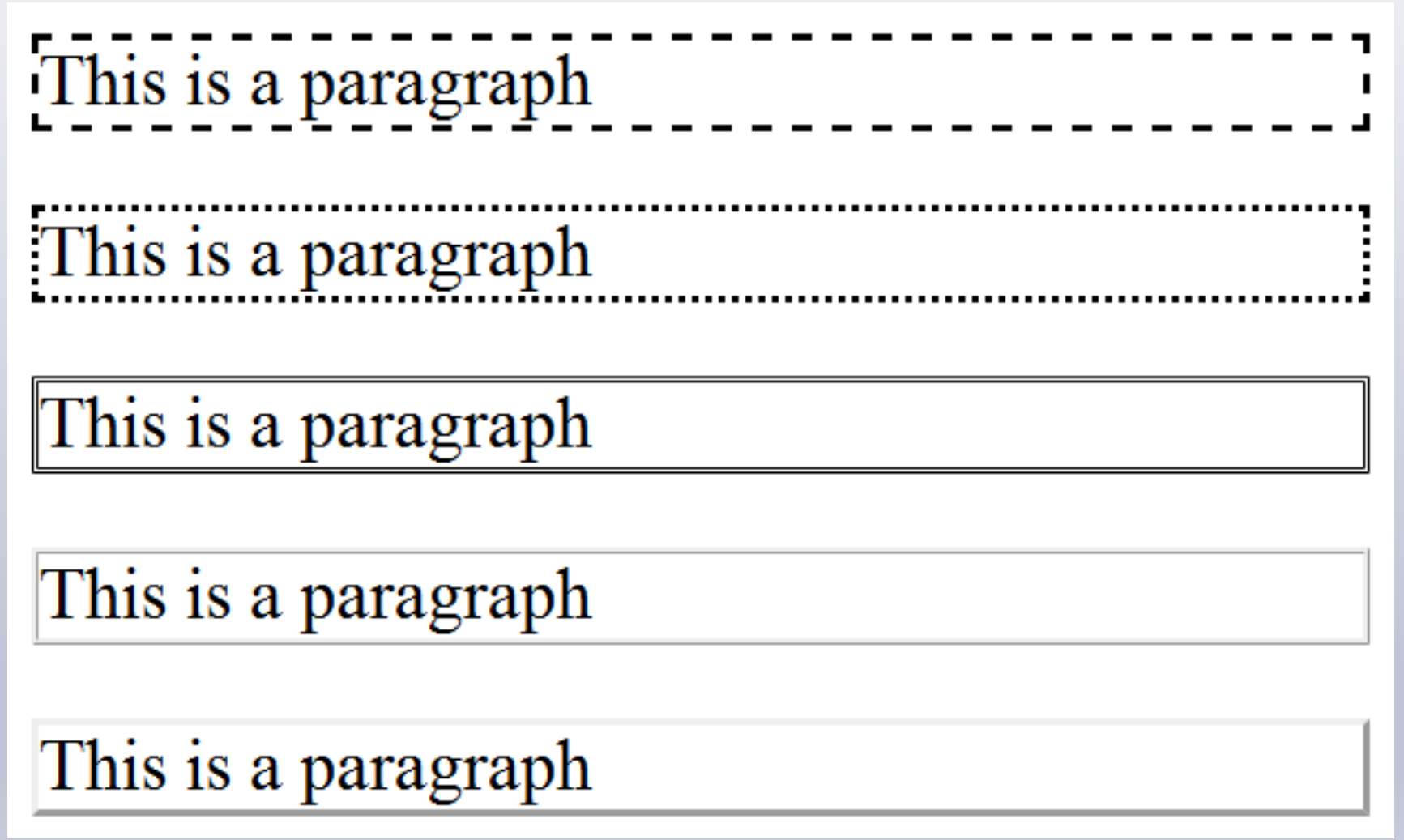
border-style property

- The border-style property allows you to specify the style of the border.
- The default value for this property is **none**, so no border shows automatically.

border-style property (cont'd)

Value	Description
`none`	No border. (Equivalent of `border-width:0;`)
`solid`	Border is a single solid line.
`dotted`	Border is a series of dots.
`dashed`	Border is a series of short lines.
`double`	Border is two solid lines; the value of the `border-width` property creates the sum of the two lines and the space between them.
`groove`	Border looks as though it is carved into the page.
`ridge`	Border looks the opposite of `groove`.
`inset`	Border makes the box look like it is embedded in the page.
`outset`	Border makes the box look like it is coming out of the canvas.
`hidden`	Same as `none`, except in terms of border-conflict resolution for table elements.

border-style property (cont'd)



border-style property (cont'd)

- The style of the bottom, left, top, and right borders of a box can be changed using the following properties:
  - border-bottom-style
  - border-right-style
  - border-top-style
  - border-left-style

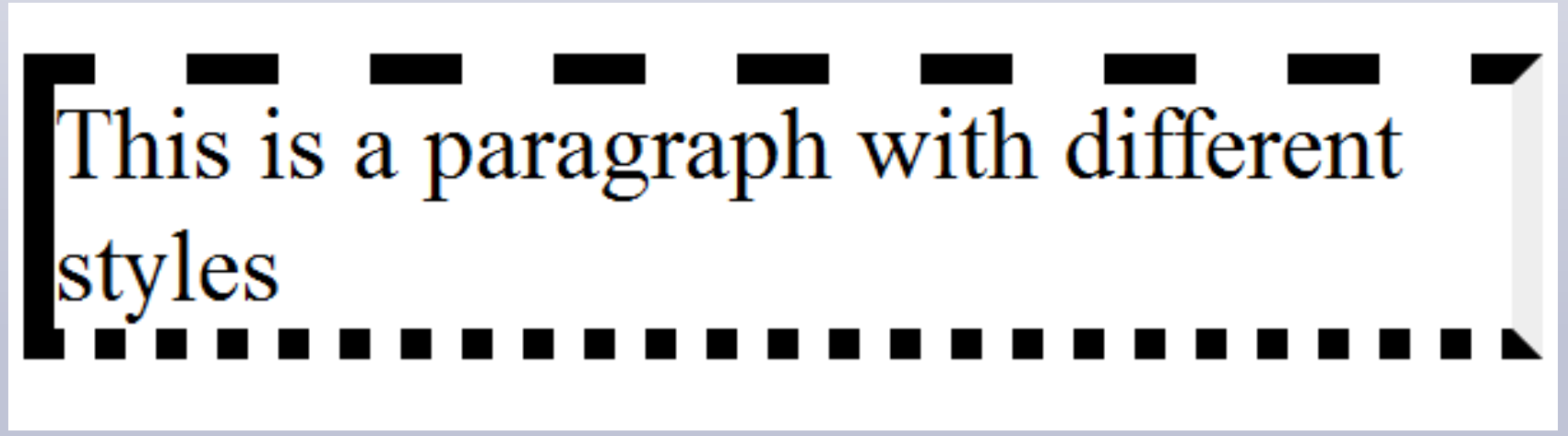
border-style property (cont'd)

border-top-style: dashed;

border-bottom-style: dotted;

border-right-style: inset;

border-left-style: solid;

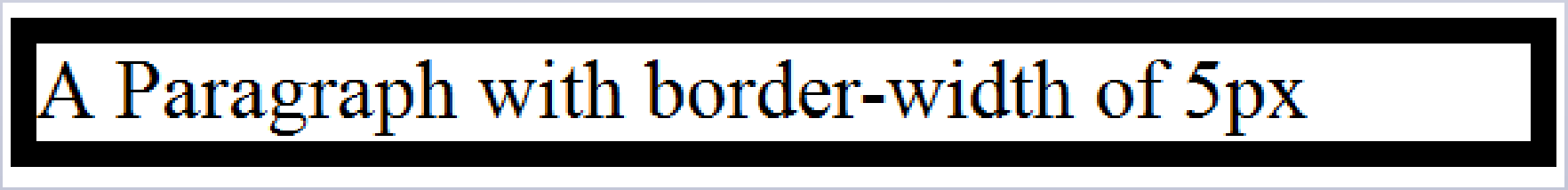


border-width property

- The border-width property allows you to set the width of your borders.
- The value of the border-width property cannot be a percentage
- It must be a length or one of the following values:
  - thin
  - medium
  - thick

border-width property (Cont'd)

```
p {
   border-width:  5px;
   border-style:  solid;
}
```

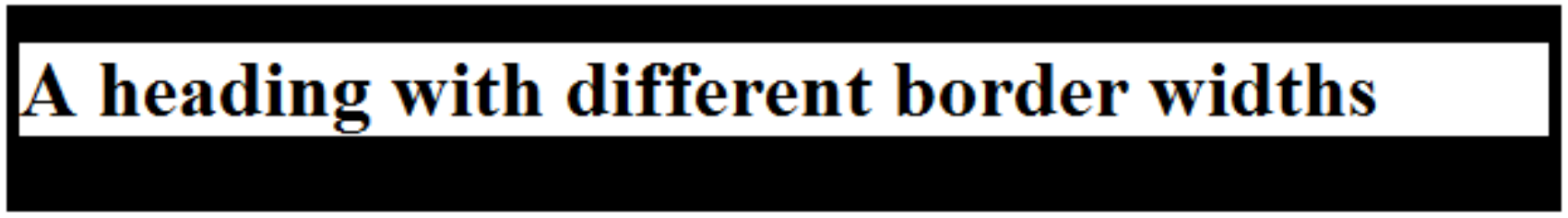


border-width property (Cont'd)

- The width of the bottom, top, left, and right borders of a box can be changed using the following properties:
  - border-bottom-width
  - border-right-width
  - border-top-width
  - border-left-width

border-width property (Cont'd)

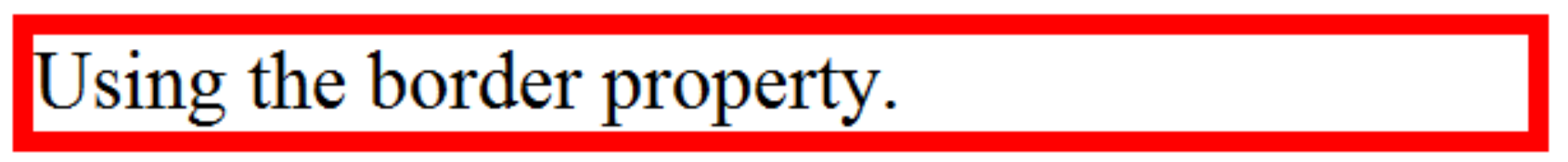
```
h1 {
   border-top-width:        15px;
   border-bottom-width:     30px;
   border-left-width:       5px;
   border-right-width:      5px;
   border-style: solid;
}
```



border property

- The border property allows you to specify color, style, and width of lines in one property:

```
p { border: 4px solid red;}
```



- In this shorthand, the values **should not have anything** (other than a space) between them.

border property

- The three color, style, and width of lines can be specify individually for each side of the box in the same way using these properties:
  - border-bottom
  - border-top
  - border-left
  - border-right

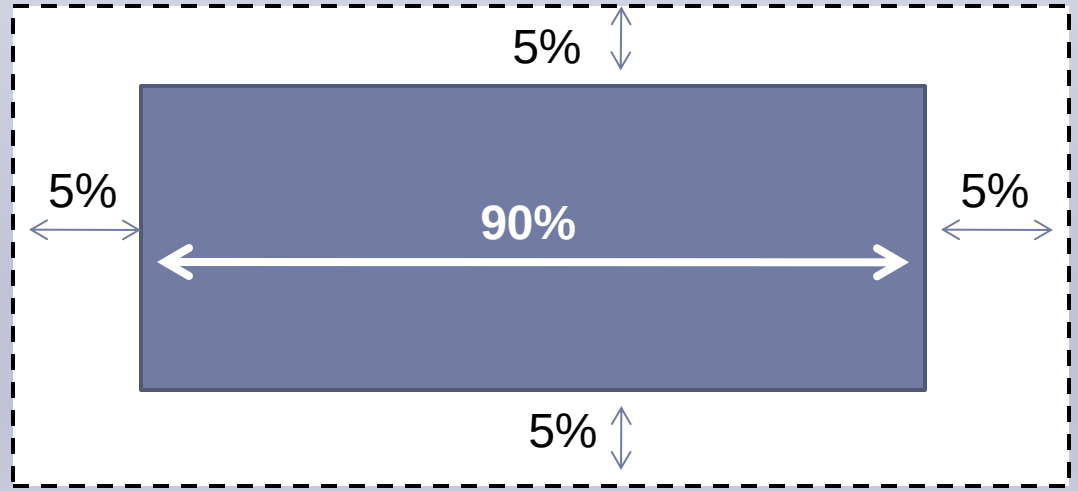
The padding property

- The **padding** property allows you to specify how much space should appear between the content of an element and its border.
- The value of this attribute should be either a length, a percentage, or the word **inherit**.
- If the value is **inherit** it will have the same padding as its parent element.

```
td {padding:5px;}
```

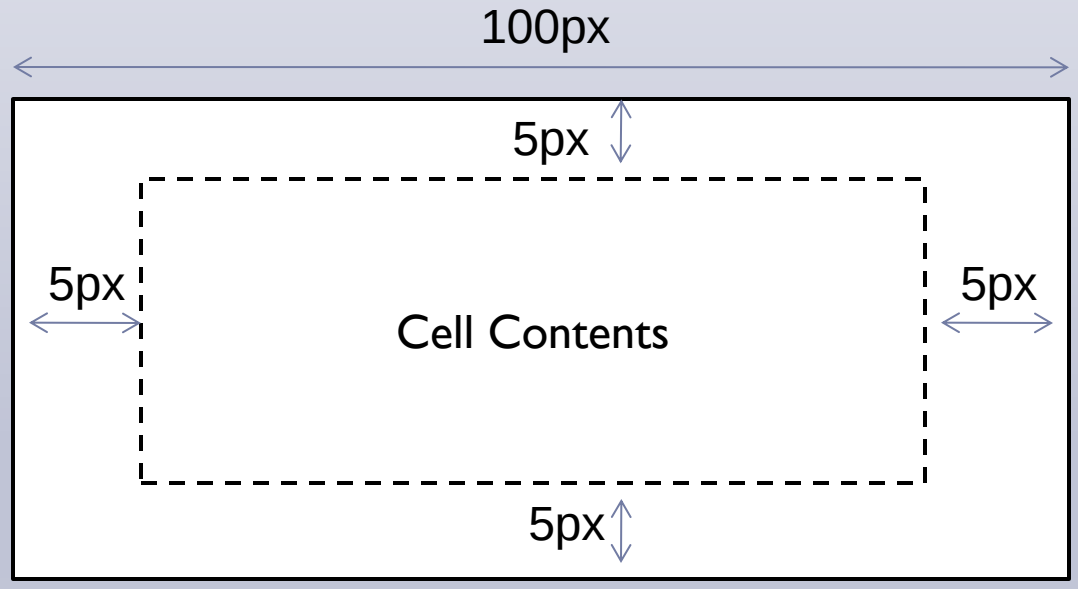
The padding property (Cont'd)

- If a percentage is used, the percentage is of the containing box.
- So, if the rule indicates the padding on the <body> element should be 10 percent, 5 percent of the browser window's width will be inside the content of the <body> element on each side as padding.



The padding property (Cont'd)

- Alternatively, if the rule indicated that a <td> element should have a **10 percent padding** in a cell that is 100 pixels square, there will be a **5-pixel padding** around each side of the square inside the border.

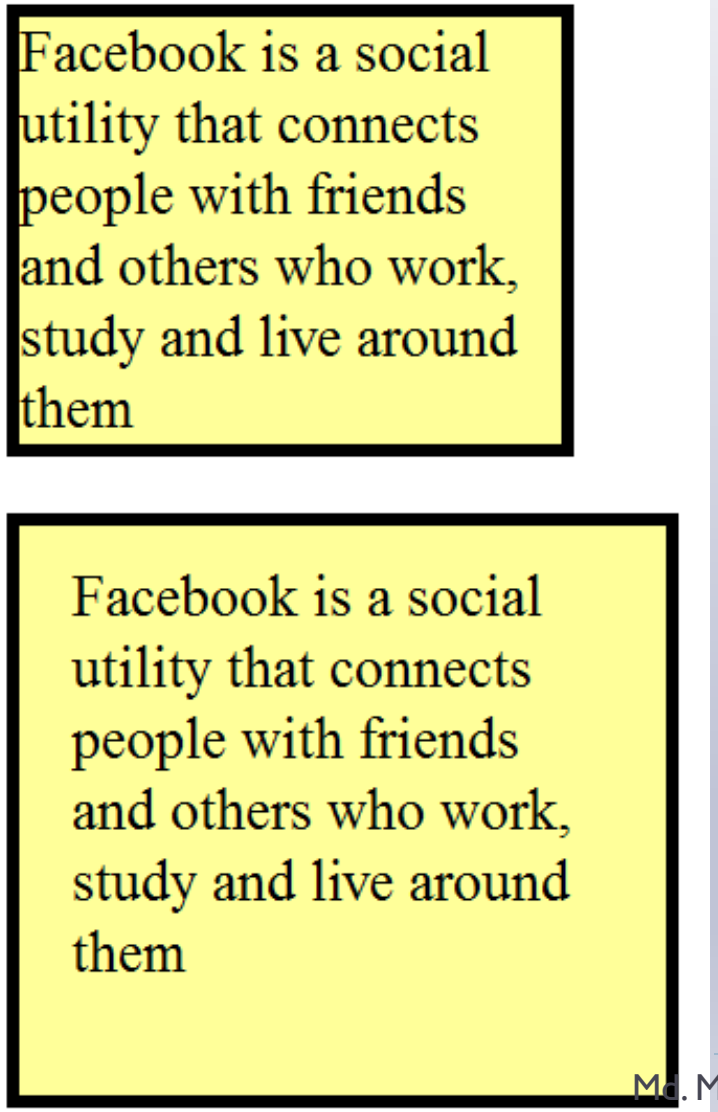


The padding property (Cont'd)

You can specify different amounts of padding inside each side of a box using the following properties:

- padding-bottom
- padding-top
- padding-left
- padding-right

The padding property (Cont'd)




```
p {
  background-color:#FFFF99;
  border: black solid 4px;
  width: 150px;
}

p.extra-padding{
  padding-top: 10px;
  padding-bottom: 30px;
  padding-left: 15px;
  padding-right: 15px;
}
```

The margin property

- The margin property is the gap between boxes, and its value is either a length, a percentage, or inherit.
- Remember that two boxes whose vertical margins touch each other will also collapse into each other.
- So, the distance between the blocks **is not the sum of the margins**, but only **the greater of the two margins.**
- if both are equal, the margin will have the size of any of those.

The margin property (Cont'd)

You can also set different values for the margin on each side of the box using the following properties:

- margin-bottom
- margin-top
- margin-left
- margin-right

The margin property Example

This paragraph will contain some *keywords* , which will have added shading and a larger *margin* around them. You will see that the *left* and *right* of these elements are spread wider than other words.

Each *keyword* also has a *background-color* property set to illustrate where the edge of the elment is and enhance the effect of the *margin* .

Between each *paragraph* there is also a top and bottom margin, although only the top one will show between paragraphs because of the smaller vertical margin *collapsing* into the larger one.

The margin property Example: CSS

```
body {
color:#000000;
background-color:#ffffff;
font-family:arial,
  verdana, sans-serif;
font-size:12px;}
p {
margin-top:40px;
margin-bottom:30px;
margin-left:20px;
margin-right:20px;
border-style:solid;
border-width:1px;
border-color:#000000;}
```

```
em {
background-color:#cccccc;
margin-left:20px;
margin-right:20px;}
```

The margin property Example: XHTML

- ** **This paragraph will contain some ****keywords****, which will have added shading and a larger ****margin**** around them. You will see that the ****left**** and ****right**** of these elements are spread wider than other words.** **

- ** **Each ****keyword**** also has a ****background-color**** property set to illustrate where the edge of the elment is and enhance the effect of the ****margin****.** **

- ** **Between each ****paragraph**** there is also a top and bottom margin, although only the top one will show between paragraphs because of the smaller vertical margin ****collapsing**** into the larger one.** **

Questions?

Web Programming

Week 11
Slides 376-393

Outline

- The min and max width and height properties
- The overflow property
- Presentation of links
- Background of documents

The max-width and min-width properties

- The max-width and min-width properties allow you to specify a maximum and a minimum width for a box.

```
div {   min-width:200px;
        max-width:500px;
        padding:5px;
        border:1px solid #000000;}
```

The min-heigh and max-height

- The min-height and max-height properties correspond with the min-width and max-width properties, but specify a minimum height and maximum height for the box.

```
div {    min-height:50px;
         max-height:200px;
         padding:5px;
         border:1px solid #000000;}
```

The overflow property

Value	Purpose
`hidden`	The overflowing content is hidden.
`scroll`	The box is given scrollbars to allow users to scroll to see the content.

```
div {     max-height:75px;
          max-width:250px;
          padding:5px;
          margin:10px;
          border:1px solid #000000;}
div.one {         overflow:hidden;}
div.two {         overflow:scroll;}
```

Presentation of Links

CSS on links

Pseudo-class	Purpose
`link`	Styles for links in general
`visited`	Styles for links that have already been visited
`active`	Styles for links that are currently active (being clicked)
`hover`	Styles for when someone is hovering over a link

CSS on links: Common properties to use

- **color**: to change colors of links.
- **text-decoration**: to control whether the link is underlined or not.
- **background-color**: Highlights the link, as if it had been highlighted with a highlighter pen.

Backgrounds of documents

Md. Masudur Rahman, Dept. Of CSE, UGV

Backgrounds

Property	Purpose
`background-color`	Specifies a color that should be used for the background of the page or box
`background-image`	Sets an image to be in the background of a page or box
`background-repeat`	Indicates whether the background image should be repeated across the page or box

Backgrounds (Cont'd)

Property	Purpose
`background-attachment`	Indicates a background image should be fixed in one position on the page, and whether it should stay in that position when the user scrolls down the page or not
`background-position`	Indicates where an image should be positioned in either the window or the containing box
`background`	A shorthand form that allows you to specify all of these properties

Background-color

- body {background-color:#cccccc; color:#000000;}
- b {background-color:#FF0000; color:#FFFFFF;}
- p {background-color: rgb(255,255,255);}

Background-image

```
body {
   background-image:  url(images/background.gif); }
```

Background-repeat

Value	Purpose
repeat	This causes the image to repeat to cover the whole page.
repeat-x	The image will be repeated horizontally across the page (not down the whole page vertically).
repeat-y	The image will be repeated vertically down the page (not across horizontally).
no-repeat	The image is displayed only once.

```
body {
background-image: url("images/background_small.gif");
background-repeat: repeat-x;
background-color: #ffffff;}
```

Background-position

Value	Meaning
`x% y%`	Percentages along the *x* (horizontal) and *y* (vertical) axis
`x y`	Absolute lengths along the *x* (horizontal) and *y* (vertical) axis
`left`	Shown to the left of the page or containing element
`center`	Shown to the center of the page or containing element
`right`	Shown to the right of the page or containing element
`top`	Shown at the top of the page or containing element
`center`	Shown at the center of the page or containing element
`bottom`	Shown at the bottom of the page or containing element

Background-position Example

```
body {
    background-image: url("images/background_small.gif");
    background-position: 50% 20%;
    background-repeat: no-repeat;
    background-color: #eaeaea; }
```

Background attachment

- The background-attachment property allows you to specify an image known as a *watermark.*
- *The* key difference with this setting is that the background image can stay in the same position even when the user scrolls up and down a page or scrolls with all of the other elements of the page.

Value	Purpose
`fixed`	The image will not move if the user scrolls up and down the page.
`scroll`	The image stays in the same place on the background of the page. If the user scrolls up or down the page, the image moves too.

Questions?

Web Programming

Week 12
Slides 394-437

Outline

- List Styles
- Table Styles
- Focus, Active, Before and After Pseudo-classes
- Content and Counters
- Display and Visibility
- Positioning and Flow

List Styles

List Styles

Property	Purpose
`list-style-type`	Allows you to control the shape or appearance of the marker (bullet point or number)
`list-style-position`	Specifies whether a long item that takes up more than one line of text and therefore wraps to a second line should align with the line or start underneath the start of the marker
`list-style-image`	Specifies an image for the marker rather than a bullet point or number
`list-style`	Serves as shorthand for the preceding properties
`marker-offset`	Specifies the distance between a marker and the text in the list

List-style-type for Unordered Lists

Value	Marker
none	None
disc (default)	A filled-in circle
circle	An empty circle
square	A filled-in square

List-style-type for Ordered Lists

Value	Meaning	Example
`decimal`	Number	1, 2, 3, 4, 5
`decimal-leading-zero`	0 before the number	01, 02, 03, 04, 05
`lower-alpha`	Lowercase alphanumeric characters	a, b, c, d, e
`upper-alpha`	Uppercase alphanumeric characters	A, B, C, D, E
`lower-roman`	Lowercase Roman numerals	i, ii, iii, iv, v
`upper-roman`	Uppercase Roman numerals	I, II, III, IV, V

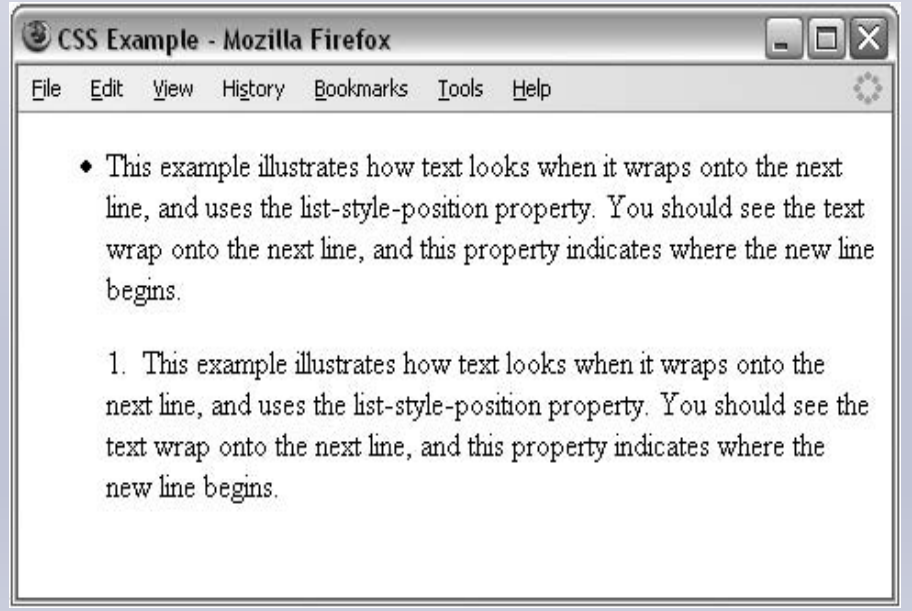
List Examples

```
li.a {list-style:none;}
li.b {list-style:disc;}
li.c {list-style:circle;}
li.d {list-style:square;}
li.e {list-style:decimal;}
li.f {list-style:lower-alpha;}
li.g {list-style:upper-alpha;}
li.h {list-style:lower-roman;}
li.i {list-style:upper-roman;}
```

List-style-position

Value	Purpose
`inside`	If the text goes onto a second line, the text will wrap underneath the marker. It will also appear indented to where the text would have started if the list had a value of `outside`.
`outside`	If the text goes onto a second line, the text will be aligned with the start of the first line (to the right of the bullet).

```
ul {list-style-position:outside; }
ol {list-style-position:inside; }
```



list-style-image

- allows you to specify an image so that you can use your own bullet style.

  li {list-style-image: url("images/bulletpoint.gif");}

- If the image cannot be displayed, the browser should just display a dot rather than a broken image symbol.

Tables

Table-specific properties

Property	Purpose
`border-collapse`	Indicates whether the browser should control the appearance of adjacent borders that touch each other or whether each cell should maintain its style
`border-spacing`	Specifies the width that should appear between table cells
`caption-side`	Specifies which side of a table the caption should appear on
`empty-cells`	Specifies whether the border should be shown if a cell is empty
`table-layout`	Allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column (rather than having to load the whole table before rendering it)

border-collapse

- specifies whether the browser should display every border

Value	Purpose
`collapse`	Horizontal borders will be collapsed and vertical borders will abut one another. (There are complex rules about conflict resolution for different border rules in the recommendation, but you should try them out and see how they work.)
`separate`	Separate rules are observed and different properties are available to further control appearance.

border-collapse Example

table.one {border-collapse:collapse;}

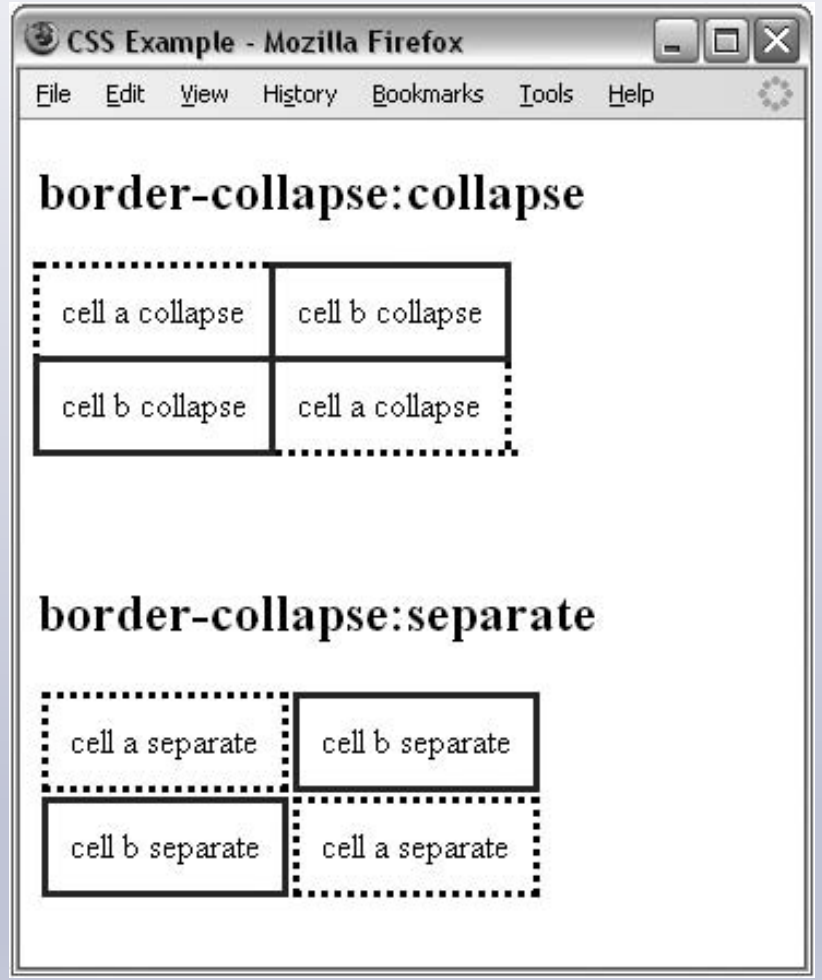
table.two {border-collapse:separate;}

td.a {border-style:dotted; border-width:3px;

border-color:#000000; padding: 10px;}

td.b {border-style:solid; border-width:3px;

border-color:#333333; padding: 10px;}



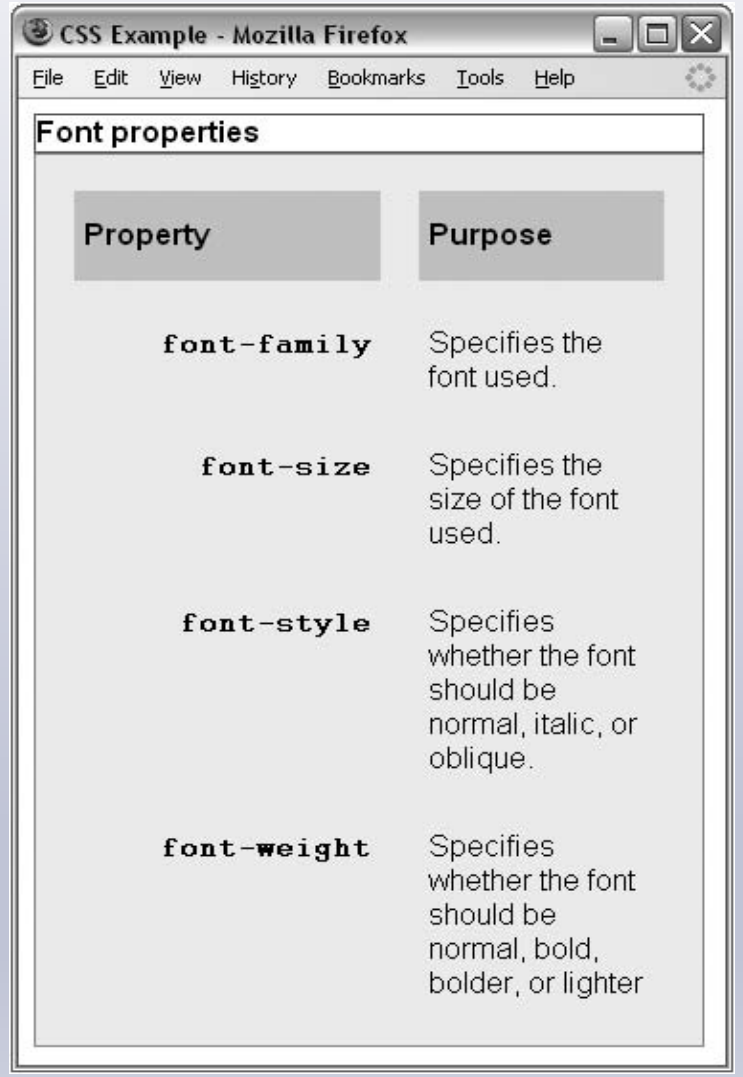
The border-spacing property

- Specifies the distance that separates adjacent cells' borders. Can receive either one or two values.
- If **one value** is provided, it will apply to both vertical and horizontal borders:

```
td {border-spacing:15px;}
```

- If **two values** are provided, the first refers to the horizontal spacing and the second to the vertical spacing:

```
td {border-spacing:2px; 4px;}
```



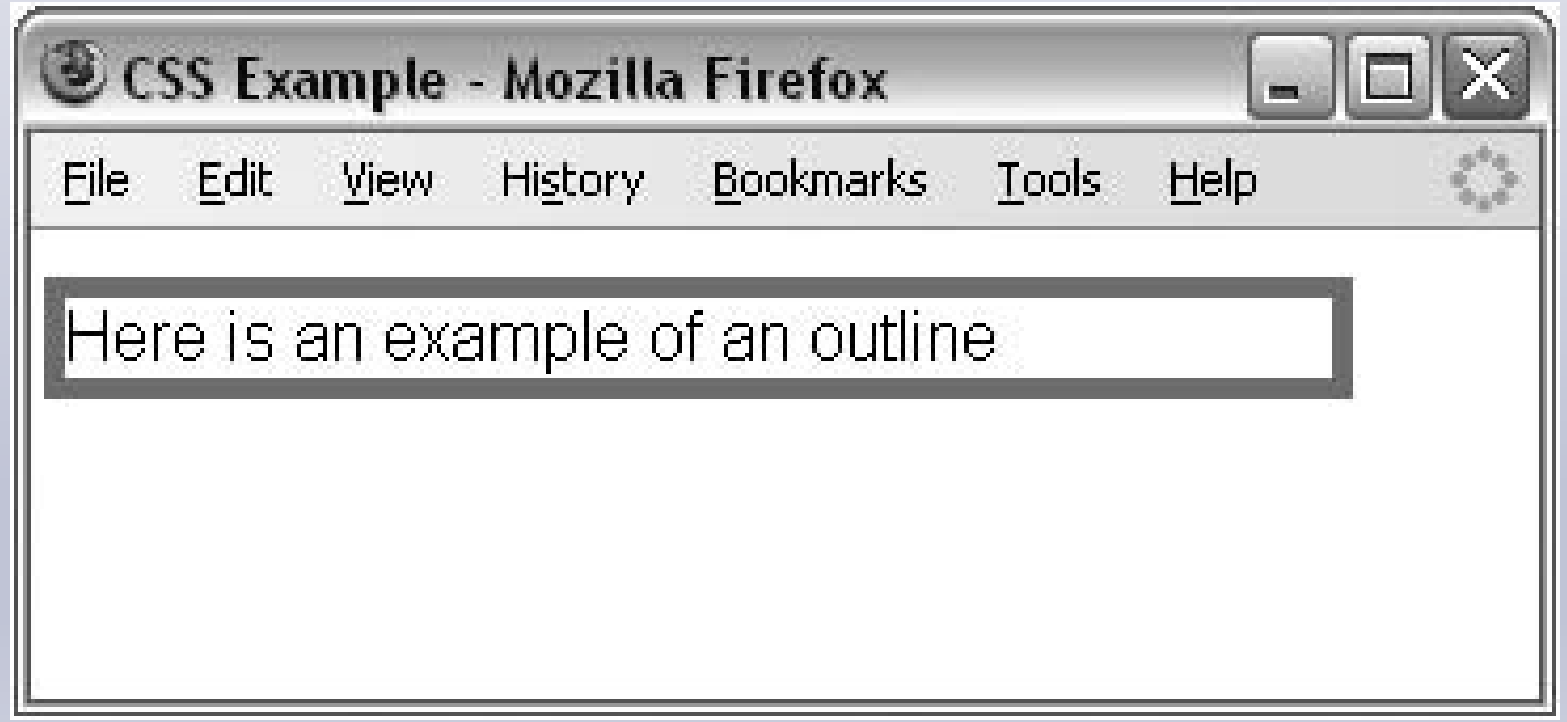
Outlines

- Outlines are similar to borders, except in:
  - An outline does not take up space
  - Outlines do not have to be rectangular

Property	Purpose
`outline-width`	Specifies the width of the outline
`outline-style`	Specifies the line style for the outline
`outline-color`	Specifies the color of the outline
`outline`	Shorthand for above properties

Outlines Example

```
input {outline-width:2px;}
input {outline-style:solid;}
input {outline-color:#ffoooo;}
```



The :focus and :active pseudo-classes

- :focus: Change the style when an element gains focus
- :active: Change the style when an element is active such as when a user clicks a link.

```
input:  focus {outline: #ff0000 thick solid;}
```

The :before and :after Pseudo-elements

- Enable you to add text before or after each instance of an element or elements defined in a selector.
- p.abstract:after {content:"You need to register to read the full article."; color:#ff0000;}

The content property

Value	Purpose
A string	To insert plain text, this may not include quotes, and therefore cannot include XHTML markup that carries attributes (the term "string" refers to a set of alphanumeric characters, not a CSS property).
A URL	The URL can point to an image, text file, or HTML file to be included at this point.
A counter	A numbered counter for elements on the page (discussed in the next section).
`atrr(x)`	The value of an attribute named *x* that is carried on that element (this is of more use to languages other than XHTML).
`open-quote`	Inserts the appropriate opening quote symbol (see the "Quotation Marks" section later in this chapter).
`close-quote`	Inserts the appropriate closing quote symbol (see the "Quotation Marks" section later in this chapter).
`no-open-quote`	Do not use any opening quotes.
`no-close-quote`	Do not use a closing quote (of particular use in prose where one person is speaking for a long while and style dictates the quote is closed only on the last paragraph).

Counters

- The counter() function is designed to allow you to create a counter that increments each time a browser comes across any specified element.
- counter-reset: set counters to zero.
- counter-increment: increase the counters every time that element is met

Counters Example

```
<body>
<h1> Introducing Web
    Technologies</h1>
<h2>Introducing HTML</h2>
<h2>Introducing CSS</h2>
<h2>Introducing XHTML</h2>
<h1> Structure of Documents</h1>
<h2>Text</h2>
<h2>Lists</h2>
<h2>Tables</h2>
<h2>Forms</h2>
</body>
```

body {counter-reset: chapter;
counter-reset: section;}

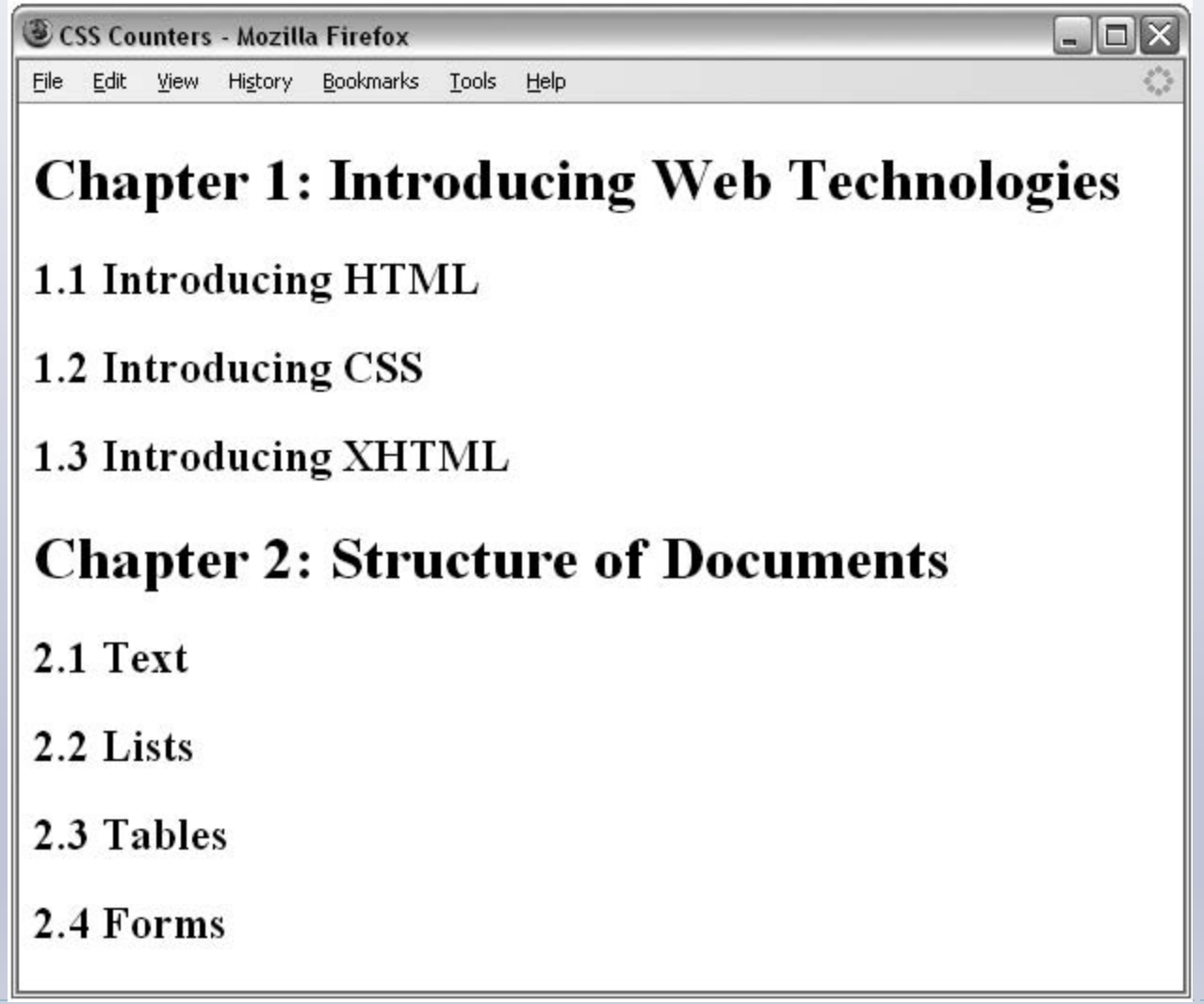
h1:before {content: “Chapter “
counter(chapter) “: “;}

h1 {counter-increment: chapter;
counter-reset: section;}

h2:before { content:
counter(chapter) “.” counter
(section) “ “; }

h2 {counter-increment: section; }

Counters Example (Cont'd)



Quotation Marks

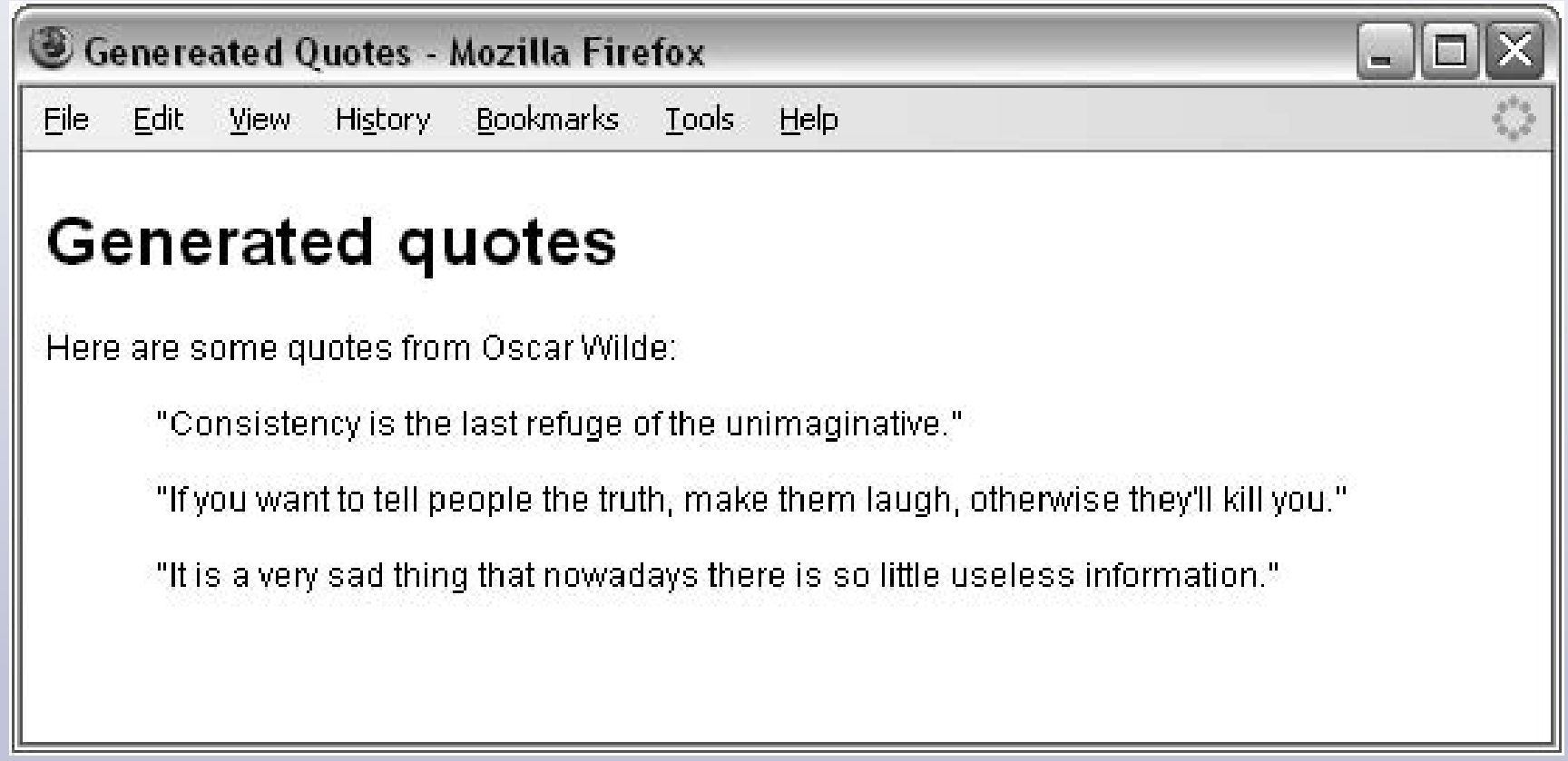
- The values **open-quote** and **close-quote** can be used with the content property to add quote marks before and after occurrences of specified elements.

Quotation Marks Example

```
<h1>Generated quotes</h1>
<p>Here are some quotes from Oscar
    Wilde:</p>
<blockquote>Consistency is the last
    refuge of the
    unimaginative.</blockquote>
<blockquote>If you want to tell people
    the truth, make them laugh, otherwise
they'll kill you.</blockquote>
<blockquote>It is a very sad thing that
    nowadays there is so little useless
information.</blockquote>
```

```
blockquote:before {
content: open-quote;}

blockquote:after {
content: close-quote;}
```

Quotation Marks Example



The display property

- forces an element (or box) to be a different type of box than you might expect it to be.
- display: block → make an inline box a block-level one.
- display: inline → make a block-level box an inline one.

- For example, <p> is a block-level box (block element):

  p { **display** : inline }

  After applying the above style, it would behave as an inline element or box.

The visibility property

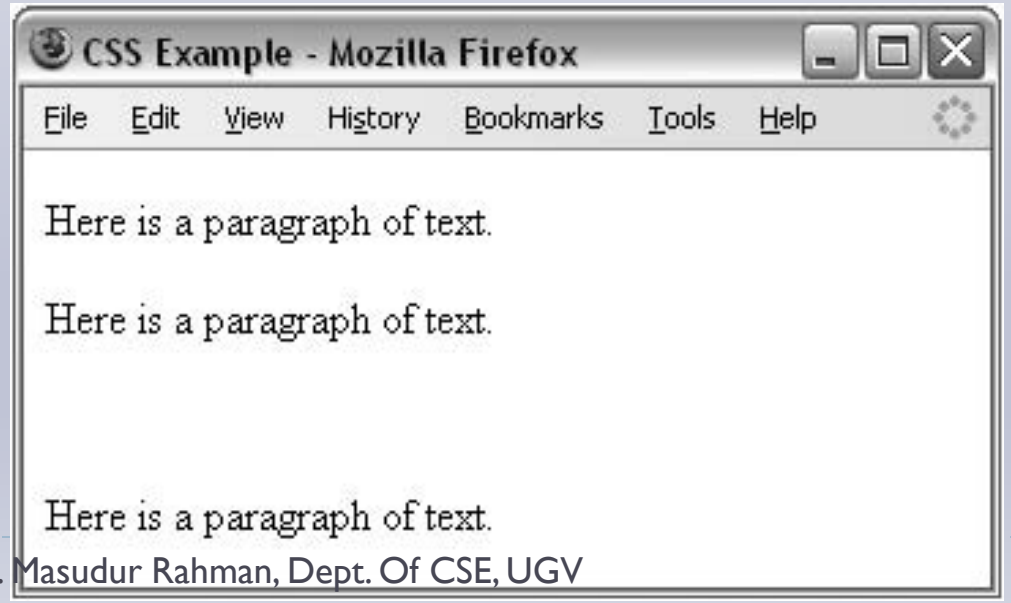
- allows you to hide a box from view, although it still affects the layout of the page (even though its contents are not seen)

Value	Purpose
`visible`	The box and its contents are shown to the user.
`hidden`	The box and its contents are made invisible, although they still affect the layout of the page.

The visibility property Example

```
<body>
<p>Here is a paragraph of text.</p>
<p>Here is a paragraph of text.</p>
<p class="invisible">This paragraph
   of text should be invisible.</p>
<p>Here is a paragraph of text.</p>
</body>
```

```
p.invisible {
visibility: hidden;}
```



Positioning with CSS

Md. Masudur Rahman, Dept. Of CSE, UGV

Positioning with CSS

- By default, elements are laid out on the page using what is known as ***normal flow***.
- In normal flow:
  - **block-level elements** flow from top to bottom
  - **inline elements** flow from left to right.

The position property

- Allows you to specify a position for a box

Value	Meaning
`static`	This is the same as normal flow, and is the default, so you will rarely (if ever) see it specified.
`relative`	The position of the box can be offset from where it would be in normal flow.
`absolute`	The box is positioned exactly from the position in the containing element using x and y coordinates from the top-left corner of the containing element.
`fixed`	The position is calculated from a fixed point; in the case of the browser, this point is the top-left corner of a browser window and does not change position if the user scrolls the window.

Box Offset properties

- Indicates where the boxes should be positioned. It is used along with the **position** property values: relative, absolute, or fixed.

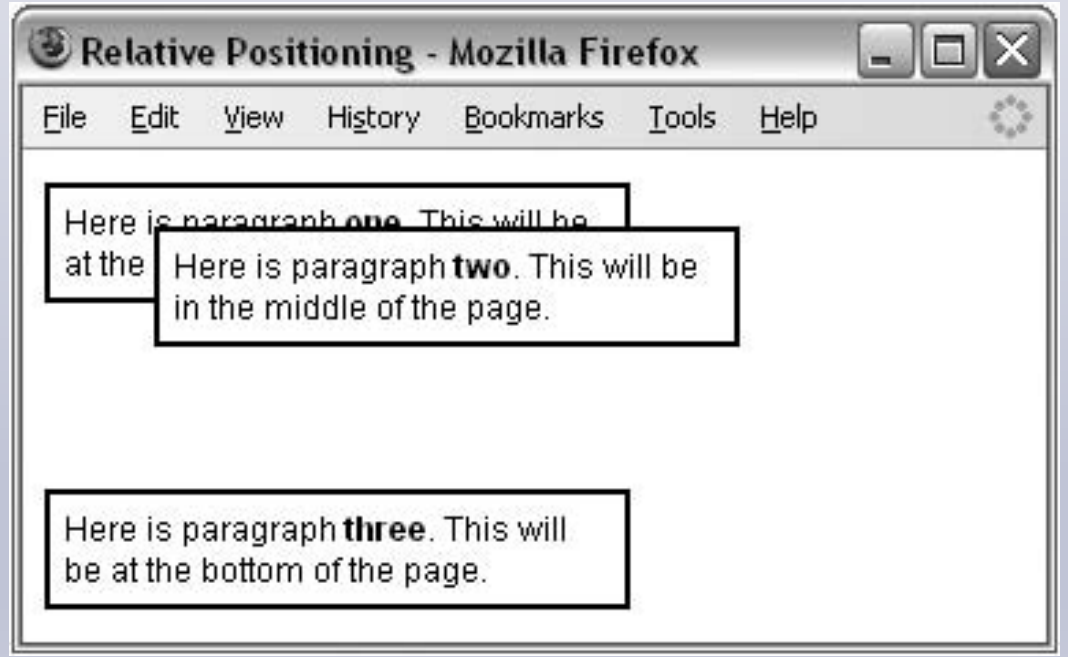
Property	Meaning
`top`	Offset position from the top of the containing element
`right`	Offset position from the right of the containing element
`bottom`	Offset position from the bottom of the containing element
`left`	Offset position from the left of the containing element

- Each can take a value of a length, a percentage, or auto.

Relative Positioning

- Takes an element and positions it in relation to where it would otherwise sit in normal flow.
- It is displaced from that position by an amount given using the box offset properties

```
p { border-style: solid;
    border-color:#000000;
    border-width:2px;
    padding:5px;
    background-color:#FFFFFF;}
p.two {
    position: relative;
    left: 40px;
    top: -40px;}
```

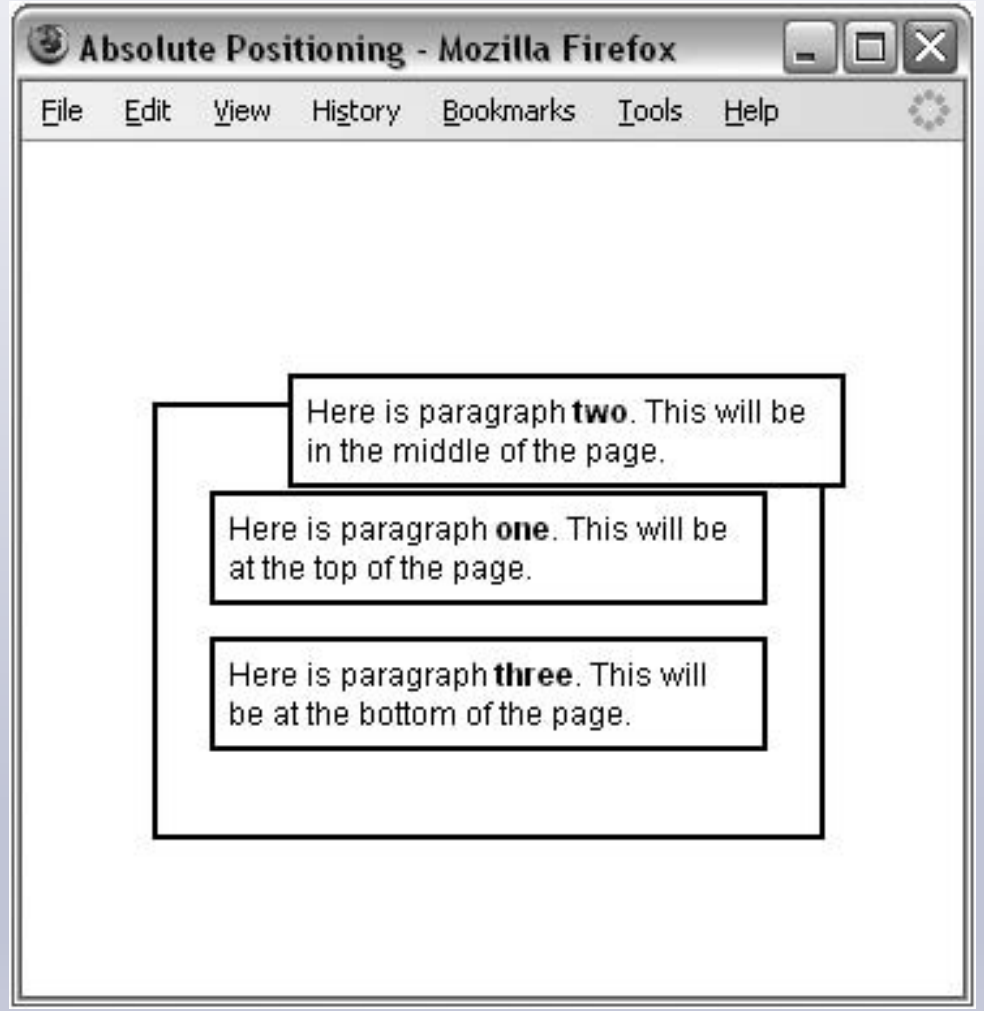


Absolute Positioning

- Completely removes an element's content from normal flow, allowing you to fix its position.
- The box offsets fix the position of a box **relative** to the containing block.
- The containing block should be an element whose position property is set to relative or fixed.

Absolute Positioning Example

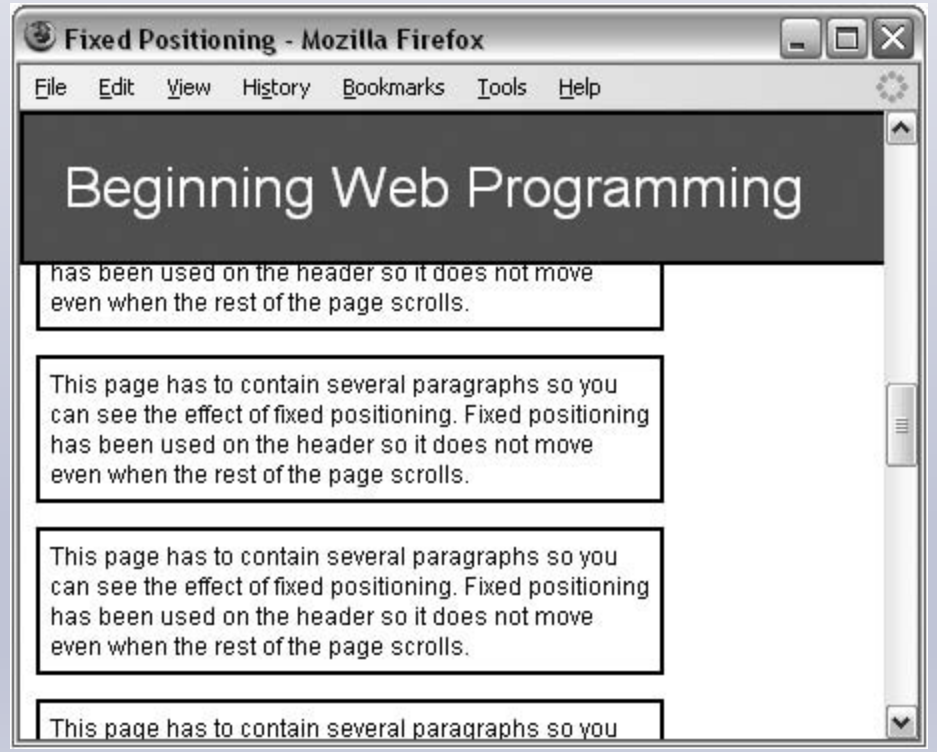
```
div.page {
position: absolute;
left: 50px;
top : 100px; }

p {
background-color:#FFFFFF;
width:200px;}

p.two {
position: absolute;
left: 50px;
top: -25px;}
```



Fixed Positioning

- Specifies that the content of the element should be completely removed from normal flow.
- The box **should not move** when users scroll down the page.
- Example:

```
div {
        position: fixed;
        top: 0px;
        left: 0px;
}
```



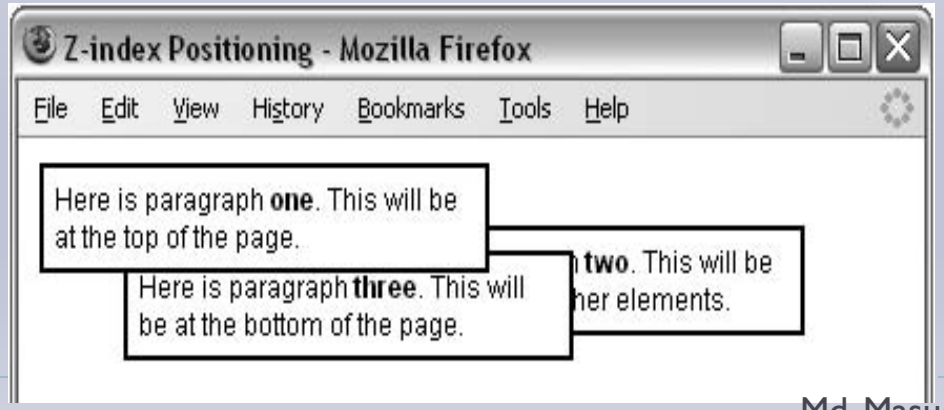
The z-index property

- Use to control which boxes, in relative or absolute position, should appear on top when they are overlapping each other.

The z-index property Example

```
<p class="one">Here is paragraph
    <b>one</b>. This will be at the
    top of the page.</p>
<p class="two">Here is paragraph
    <b>two</b>. This will be
    underneath the other
    elements.</p>
<p class="three">Here is paragraph
    <b>three</b>. This will be at the
    bottom of the page.</p>
```

```
p.one {
z-index: 3;
position: absolute;
left: 0px; top: 0px;}

p.two {
z-index: 1;
position: absolute;
left: 150px; top: 25px;}

p.three {
z-index: 2;
position: absolute;
left: 40px; top: 35px;}
```



Floating using the float property

- Allows you to take an element out of **normal flow** and place it as far to the left or right of a containing box as possible within that element's padding.

Value	Purpose
`left`	The box is floated to the left of the containing element and the content of the containing element will flow to the right of it.
`right`	The box is floated to the right of the containing element and the content of the containing element will flow to the left of it.
`none`	The box is not floated and remains where it would have been positioned in normal flow.
`inherit`	The box takes the same property as its containing element.

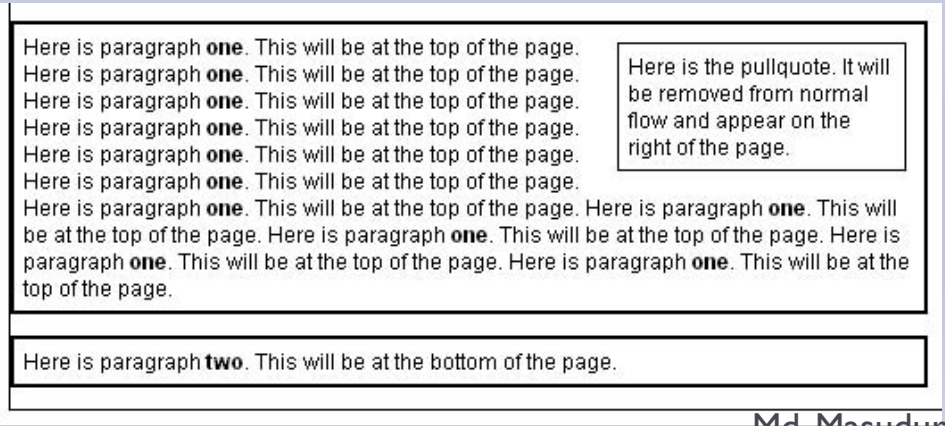
Floating using the float property (cont'd)

Important!

- Whenever you specify a **float** property, you should also set a **width** property indicating the width of the containing box that the floating box should take up.
- Otherwise, it will automatically take up 100% of the width of the containing box leaving no space for things to flow around it, therefore making it just like a plain block-level element.

Floating using the float property Example

```
<h1> Heading</h1>
<p> <span class="pullQuote"> Here is the pullquote….</span>
Here is paragraph <b>one</b>. This will be at the top of the page. …… at the top of the page. Here is paragraph
</b>. This will be at the top of the page.
</p>
<p> Here is paragraph <b>two</b>. This will be at the bottom of the page.</p>
```




```
p {
    border-style:solid;
    border-color:#000000;
    border-width:2px;
    padding:5px;
    background-color:#FFFFFF;
    width:500px;}
.pullQuote {
    float:right;
    width:150px;
    padding:5px;
    margin:5px;
    border-style:solid;
    border-width:1px; }
```

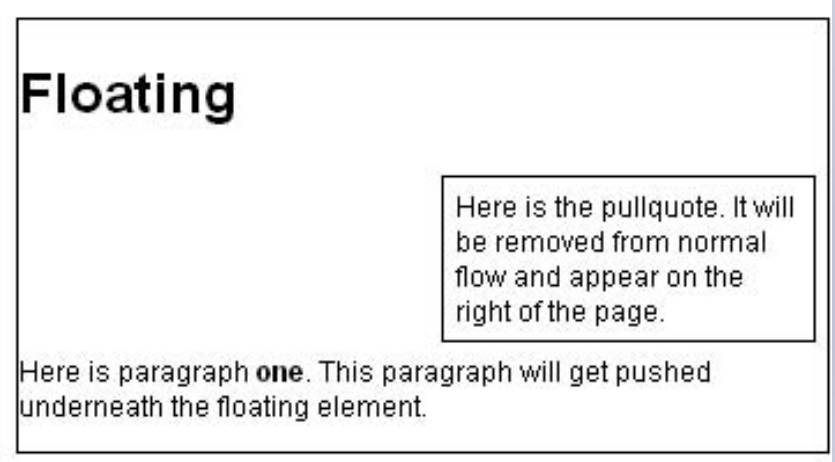
The clear property

- It's especially helpful when working with boxes that are floated.
- In case it is preferred that nothing sat next to the floated element, and that surrounding content be pushed underneath the floated element.

Value	Purpose
`left`	The content of the element with the `clear` property is cleared from the left side of a float (it cannot appear to the left of a floating element).
`right`	The content of the element with the `clear` property is cleared from the right side of a float (it cannot appear to the right of a floating element).
`both`	The content of the element with the `clear` property is cleared from either side of a float (it cannot appear to either side of a floating element).
`none`	Allows floating on either side.

The clear property Example

```
<h1>Floating</h1>
<div class="pullQuote">Here is the pullquote. It will be removed from normal flow and appear on the right of the page.</div>
<p>Here is paragraph <b>one</b>. This paragraph will get pushed underneath the floating element.
</p>
```




```
p {
clear: right;
background-color:#FFFFFF; }

div.pullQuote {
float: right;
padding: 5px;
margin: 5px;
width: 150px; }
```

Questions?

Web Programming

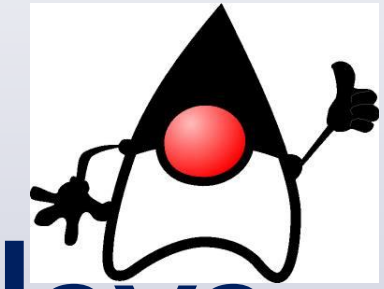
Week 13
Slides 438-477

Intro to JavaScript

Md. Masudur Rahman, Dept. Of CSE, UGV

What is JavaScript?

JavaScript != Java



Intro to JavaScript

- JavaScript is a lightweight programming language
- There is no way of teaching all about JavaScript in just a few classes.
- However, we will learn to understand the **basics** and be **able to reuse** the thousands of free scripts available on the web such as the Google Libraries API.

Why do we need JavaScript?

JavaScript allows to:

- Read elements from documents and write new elements and text into documents
- Manipulate or move text
- Create pop-up windows
- Perform mathematical calculations on data
- React to events, such as a user's rolling over an image or clicking a button

JavaScript allows to:

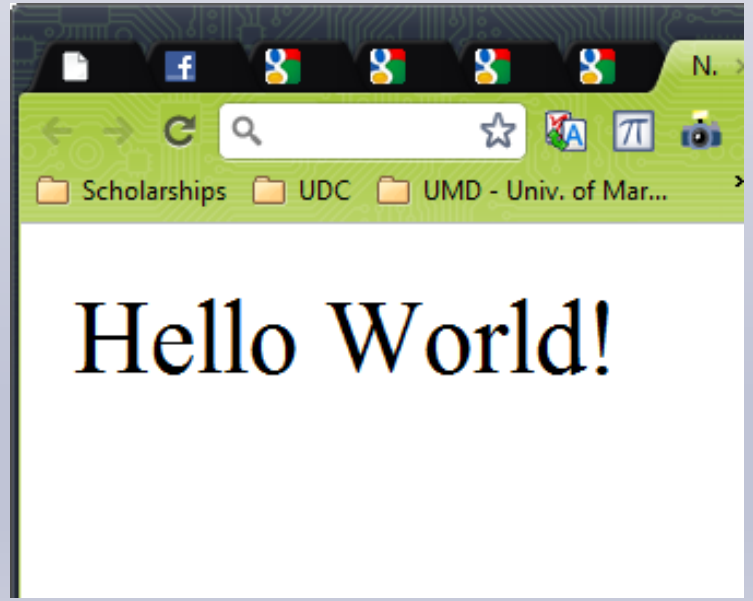
- Retrieve the current date and time from a user's computer or the last time a document was modified
- Determine the user's screen size, browser version, or screen resolution
- Perform actions based upon conditions such as alerting users if they enter the wrong information into a form or if they press a certain button

Hello World! In JavaScript

Md. Masudur Rahman, Dept. Of CSE, UGV

Our First Example

```
<html>
<body>
<p>
<script type=”text/javascript”>
   document.write(“Hello World!”)
</script>
</p>
</body>
</html>
```

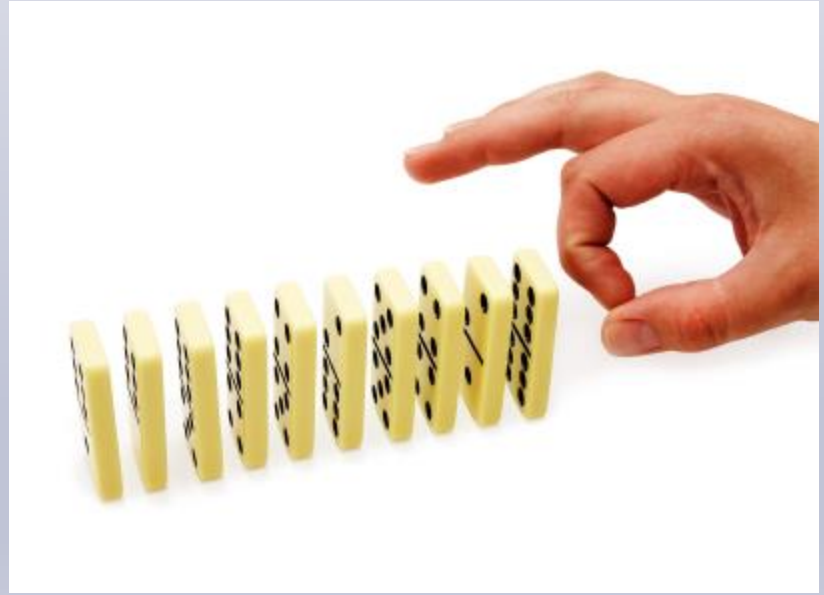


The write( ) method

- It writes text into the document.

Events

- How about doing things when an event triggers???
- but first, what is an EVENT?
- An event can be something like a **key being pressed**, or a **submit button being clicked**.



Where to put your JavaScript code?

- Three places:

1. **In the <head>**: Embedded on the web page using the <script> element. It will execute when an event triggers.
2. **In the <body>:** This scripts will execute while the page is being loaded.

   <script> /* Your JavaScript Code*/</script></head>
3. **As an external file**: with extension .js (**most recommended**). Same as 1.

   <script type="JavaScript" src="js/your_script.js" />

How to comment your code?

- Use JavaScrip comments.
- Two ways of how to comment:
  1. Single Line:

```
document.write("Hello World");          //this is a comment
```

  2. Multiple Lines:

```
/* This is a multiple line comment that
can take up to as many lines
you want to use */
```

- You should **comment your code as much as possible** to make it clear for other people to read it.

Example

1. Create an external JavaScript file: script.js:

   document.write("<h1>Hello JavaScript!!!</h1>");
2. Create an html file with the following in the body

   <body>

   <script src="script.js" type="text/JavaScript"></script>

   </body>

3. Test it and comment your output.

DOM: The Document Object Model

Md. Masudur Rahman, Dept. Of CSE, UGV

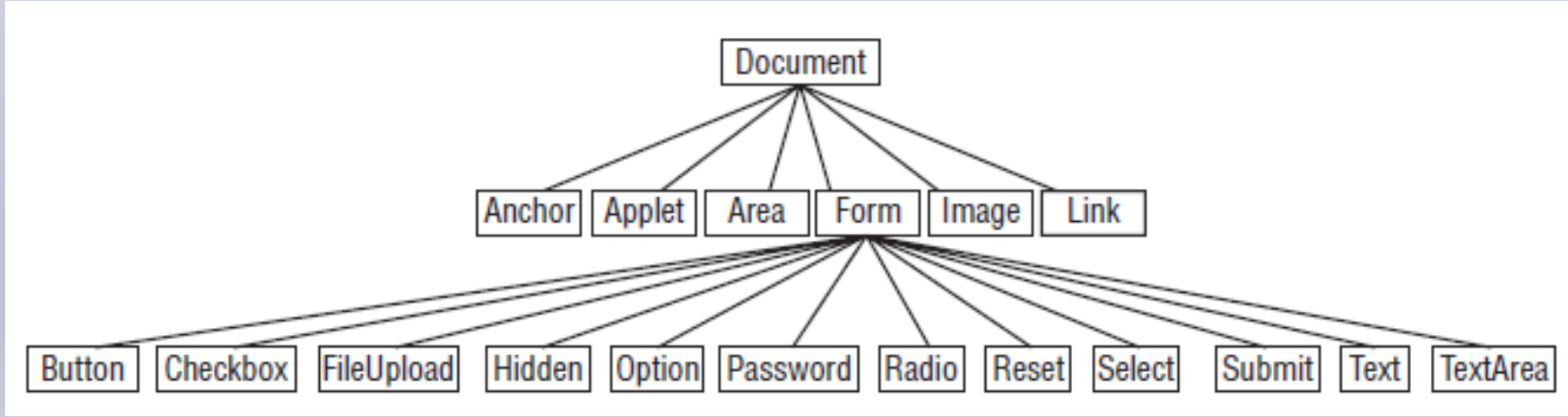

FreeFoto.com

The Document Object Model (DOM)

- Explains what ***properties of a document*** *a script can retrieve and which ones it* can alter
- Also **defines some *methods*** *that can be called to perform an action on the document*

Intro to DOM

- The figure below shows you an illustration of the **Level 0 HTML Document Object Model**
- It specifies how you can retrieve values users have entered into a form. Once you have retrieved these values, you can use JavaScript to ensure the user has entered an appropriate value for that form control.



Intro to DOM (Cont'd)

- The *forms collection contains all the <form> tags in the document.*
- The *image collection represents all the images in a document.*
- The *link collection represents all the hyperlinks within a page.*
- The *anchor collection represents all the anchors in a document (<a> elements with a name* or id attribute rather than an href attribute).
- The *area collection represents all the image maps that use an <area> element in the* document.
- The *applet collection represents all the applets within a document.*

Intro to DOM (Cont'd)

- The forms collection also has child objects to represent each of the different types of form controls that can appear on a form:
  - Button
  - CheckBox
  - FileUpload
  - Hidden
  - Option
  - Password
  - Radio
  - Reset, Select, Submit, Text, and TextArea.

Example 1: retrieve the value of the first link in the document

```
<h1>User Registration</h1>
<form name="frmLogin" action=”login.aspx” method=”post”>
Username <input type=”text” name=”txtUsername” size=”12” />
<br />
Password <input type=”password” name=”pwdPassword” size=”12” />
   <br />
<input type=”submit” value=”Log In” />
</form>
<p>If you are a new user <a href=”register.aspx”>Register here</a> |
If you have lost your password you can
<a href=”lostPassword.aspx”>retrieve your password here</a>.</p>
```

Example 1: retrieve the value of the first link in the document (Cont'd)

- In order to access the first link in the document, you could use something like this:

```
document.links[0].href
```

Example 1: retrieve the value of the first link in the document (Cont'd)

There are four parts of this statement, three of which are separated by periods, to get to the first link:

1. The word document indicates access to the document object.
2. The word links corresponds to the links collection.
3. The [0] indicates that we want the first link in the document.
4. Now, we retrieve the **href** property for this link.

Example 2: retrieve the value of the text in password

```
<h1>User Registration</h1>
<form name="frmLogin" action=”login.aspx” method=”post”>
Username <input type=”text” name=”txtUsername” size=”12” />
<br />
Password <input type=”password” name=”pwdPassword” size=”12” />
   <br />
<input type="submit" value="Log In"/>
</form>
<p>If you are a new user <a href=”register.aspx”>Register here</a> |
If you have lost your password you can
<a href=”lostPassword.aspx”>retrieve your password here</a>.</p>
```

Example 2: retrieve the value of the text in password

- Or:

  document.frmLogin.**pwdPassword**.**value**

- The **document** comes first again as it is the top-level object.
- The name of the form: **frmLogin**.
- This is followed by the name of the form control: **pwdPassword**.
- Finally the property to retrieve is the value of the password box: **value**

Some DOM Properties

- **title**: The title of the page in the <title> element
- **lastModified**: The date the document was last modified. (sent by the web server)
- **referrer**: the URL of the XHTML page that users came from if they click a link. Empty is there is no referrer.

- For example to access the title of a document do:

  document.title

- Or to find out the date a document was last modified:

  document.lastModified

DOM Methods

- write(string) : Allows you to add text or elements into a document
- writeln(string) : same as write() but adds a new line at the end of the output. (like pressing Enter key)

For example:

```
document.write("page last modified on" +
  document.lastModified);
```

The Form Collections

The Forms Collection

- The forms collection holds references corresponding to each of the **<form>** elements in the page.
- 0 for the first form, 1 for the second form, 2 for the third, and so on.
- For example:

  document.forms[0].action
- Or accessing by element's name:

  document.frmLogin.action

Forms: Properties and Methods

Property Name	Purpose	Read/Write
`action`	The `action` attribute of the `<form>` element	Read/write
`length`	Gives the number of form controls in the form	Read only
`method`	The `method` attribute of the `<form>` element	Read/write
`name`	The `name` attribute of the `<form>` element	Read only
`target`	The `target` attribute of the `<form>` element	Read/write

Method Name	Purpose
`reset()`	Resets all form elements to their default values
`submit()`	Submits the form

Form Elements: Properties and Methods

- Each <form> element has an elements[ ] collection object as a property, which represents all of the elements in that form.
- Here are some of the things we can do with the elements in a form:
  - **Text fields:** Read data a user has entered or write new text to these elements.
  - **Checkboxes** and **radio buttons**: Test if they are checked and check or uncheck them.
  - **Buttons**: Disable them until a user has selected an option.
  - **Select boxes**: Select an option or see which option the user has selected.

Form Elements: Properties

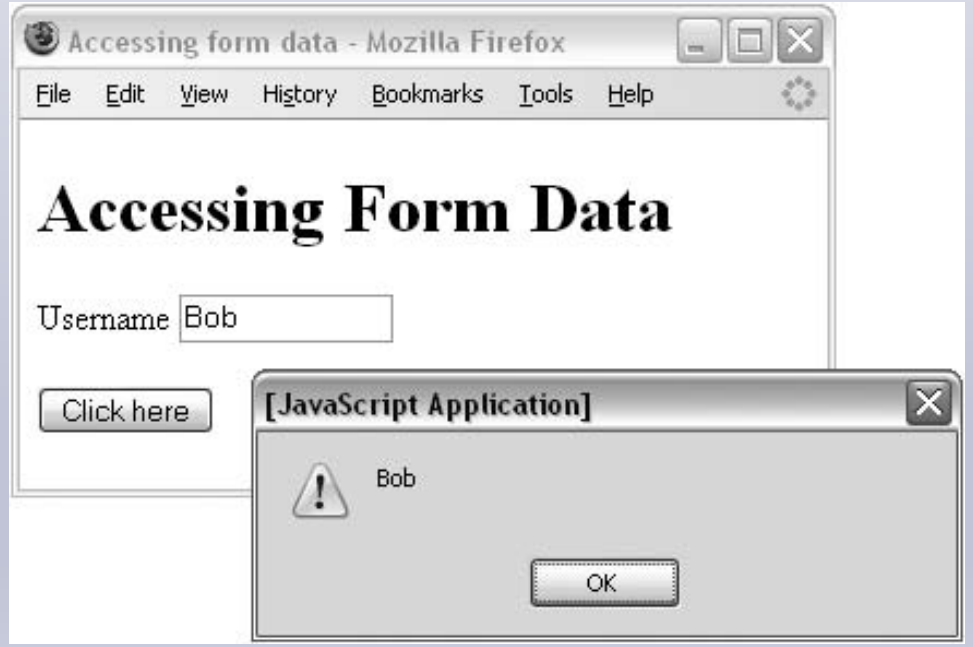
Property	Applies to	Purpose	Read/Write
`checked`	Checkboxes and radio buttons	Returns `true` when checked or `false` when not	Read/write
`disabled`	All except hidden elements	Returns `true` when disabled and user cannot interact with it (supported in IE4 and Netscape 6 and later versions only)	Read/write
`form`	All elements	Returns a reference to the form it is part of	Read only
`length`	Select boxes	Number of options in the `<select>` element	Read only
`name`	All elements	Accesses the name attribute of the element	Read only
`selectedIndex`	Select boxes	Returns the index number of the currently selected item	Read/write
`type`	All	Returns type of form control	Read only
`value`	All	Accesses the value attribute of the element or content of a text input	Read/write

Form Elements: Methods

Property Name	Applies to	Read/Write
`blur()`	All except hidden	Takes focus away from currently active element to next in tabbing order
`click()`	All except text	Simulates the user's clicking the mouse over the element
`focus()`	All except hidden	Gives focus to the element
`select()`	Text elements except hidden	Selects the text in the element

Exercise

- Retrieve the value of a text box and write it into something known as a JavaScript alert box.
- When the user clicks on a submit button of the form, the alert box will be displayed containing the text written in the input box.



The Image Collection

Images Collection

- The images collection provides references to image objects, one representing each image in a document.
- These can again be referenced by name or by their index number in the collection.
- So the src attribute of the first image could be found using the following:
  - document.images[0].src
- Or by the element's name:
  - document.myImage.src
- To create a rollover change the **src** property.

Image Collection: Properties

Property	Purpose	Read/write
`border`	The `border` attribute of the `<img>` element	Read/write
`complete`	Indicates whether an image has loaded successfully	Read only
`height`	The `height` attribute of the `<img>` element	Read/write
`hspace`	The `hspace` attribute of the `<img>` element	Read/write
`lowsrc`	The `lowsrc` attribute of the `<img>` element (indicating a lower resolution version of the image)	Read/write
`name`	The `name` attribute of the `<img>` element	Read/write
`src`	The `src` attribute of the `<img>` element	Read/write

Example: Creating an Image Rollover

- Replace one image with another one, when the user rolls over the image with the mouse.

Summary

- You will come across several types of objects in JavaScript, each of which is responsible for a related set of functionalities.
- Some other types of objects you are likely to come across:
  - **W3C DOM objects:** document, forms, images, etc.
  - **Built-in objects**: objects that are built-in or are part of JavaScript.
  - **Custom objects:** Objects you create by yourself

Questions?

Web Programming

Week 14
Slides 478-491

Md. Masudur Rahman, Dept. Of CSE, UGV

Programming with JavaScript

Md. Masudur Rahman, Dept. Of CSE, UGV

Programming with JavaScript

- *Variable: is used to store some information. (names, numbers, etc)*
- *Operators: They do "things" to variables:*
  - *Arithmetic: +, -, *, /*
  - *Comparison: Compare strings, numbers, etc. For Example: 4 is equal to 5?*
- *Functions: A group of steps that will perform a desired operation*
- *Control Structures:*
  - *Conditional: if / else / swtich*
  - *Repeat: to repeat a single action or set of actions. for / while / break / continue*

Variables

- var userName = "Will Smith"
- alert(userName);

- Variables are Case sensitive!
- Must begin with a letter or an underscore.
- Use descriptive names for your variables

Operators

- Arithmetic operators: + - * / % ++ --
  - 10 + 5 = 15
  - 4 – 3 = 1
  - 10 * 3 = 30
  - 4 / 2 = 2
  - 15 ++ = 16 (this increments by 1)
  - 8 -- = 7 (this decrements by 1)
  - 8 % 5 = 3 (division remainder)
- Assignment operators
- Comparison operators
- Logical operators

Assignment Operators (=, +=, -=, /=, %=)

- Assignment operators
  - profit = income – expenses

  - count = count + i SAME ➔ count += i
  - count = count - i SAME ➔ count -= i
  - count = count / i SAME ➔ count /= i
  - count = count * i SAME ➔ count *= i
  - count = count % i SAME ➔ count %= i

Comparison Operators

- ==
  - 4 == 3 // returns false
- !=
  - 4 != 3 // returns true
- >
  - 4 > 3 // returns true
- <
  - 4 < 3 // returns false
- >=
  - 4 >= 3 // returns true
- <=
  - 4 <= 3 // returns false

Logical Operators (Cont'd)

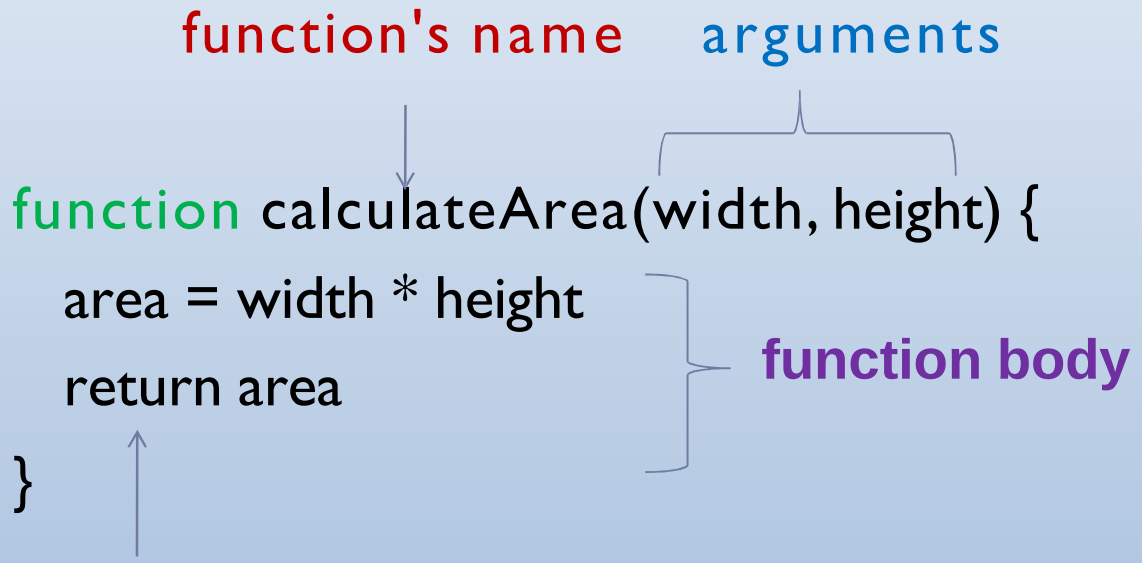
- && (AND Operator)
  - (3 < 2 && 3 > 1) // returns false
- ?? (OR Operator)
  - (3 < 2 ?? 3 > 1) //returns true
- ! (Not Operator)
  - ! (3 < 2) //returns true

String Operator

- You can add text to strings using the + operator.
- For example:
  - `firstName = "Bob"`
  - `lastName = "Stewart"`
  - `name = firstName + lastName`

Functions

- How to create a function in JavaScript?



Function (Cont'd)

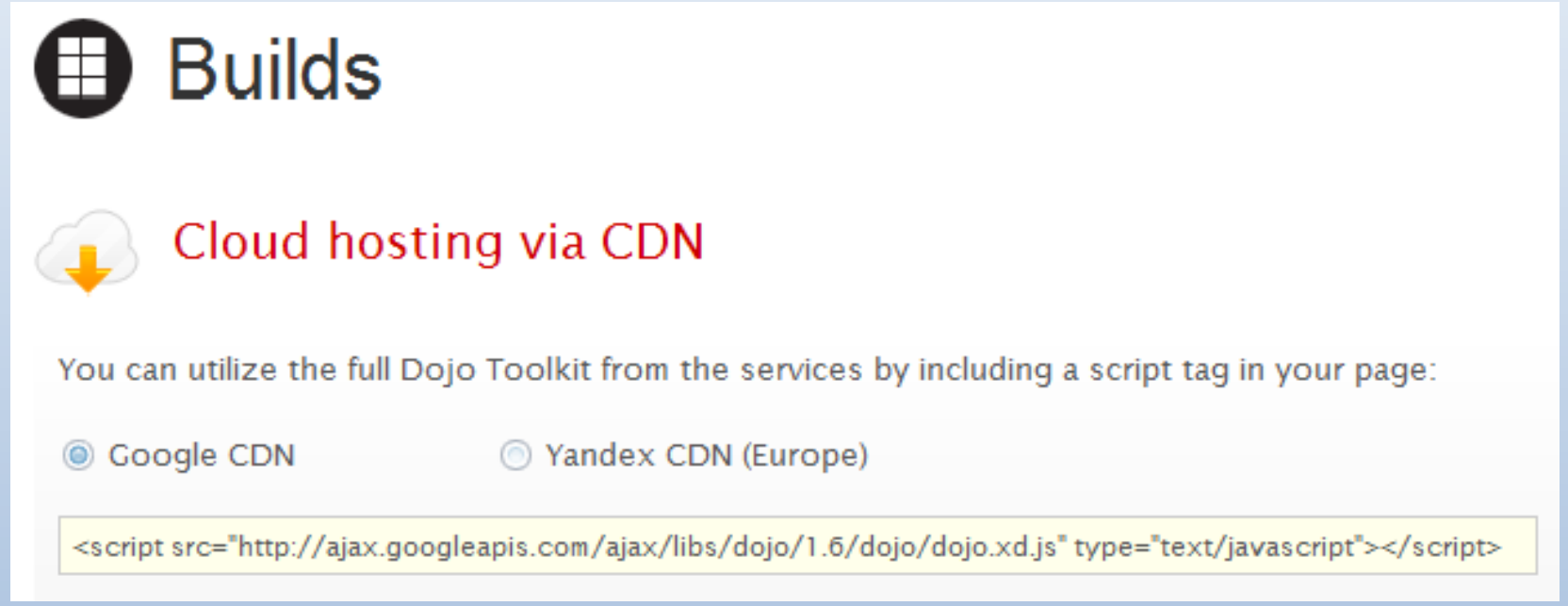
- Great! Now, we have created our function. But, how do we call it? How do we use it?

```
<form name="frmArea" action="">
  Enter the width and height of your rectangle to calculate the size:<br/>
  Width:<input type="text" name="txtWidth" size="5" /><br/> Height:<input
  type="text" name="txtHeight" size="5" /><br/>
  <input type="button" value="Calculate area"
    onclick="alert(calculateArea(document.frmArea.txt
    Width.value, document.frmArea.txtHeight.value))"
    />
</form>
```

JS Framework: Dojo Toolkit

- http://dojotoolkit.org/

Get Dojo !



<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js" type="text/javascript"></script>

Questions?

Web Programming

Week 15
Slides 492-531

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

- Selection Statements (if, if-else, switch)
- Loops (for, while, do..while)
- Built-in Objects:
  - Strings
  - Date
  - Math
  - Arrays

Selection Statements

Md. Masudur Rahman, Dept. Of CSE, UGV

if

- if statements allow code to be executed when the condition specified is true.
- if the condition is `true` then the code in the curly braces is executed.

```
if (condition)
{
    code to be executed if condition is true
}
```

What this code does?

```
<script type=”text/JavaScript”>
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
</script>
```

if-else

- If the conditions specified are met, run the first block of code; otherwise run the second block.

```
if (condition) {
   code to be executed if condition is true
}
else {
   code to be executed if condition is false
}
```

What this code does?

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
  else {
    document.write('Good Afternoon');
  }
</script>
```

switch

- A switch statement allows you to deal with several results of a condition.

```
switch (expression) {
  case option1: code to be executed if expression is what is
    written in option1
  break;
  case option2: code to be executed if expression is what is
    written in option2
  break;
  case option3: code to be executed if expression is what is
    written in option3
  break;
  default: code to be executed if expression is different from
    option1, option2, and option3
}
```

switch example

```
<p>Enter the name of your favorite type of animal that stars in a
   cartoon:</p>
<form name=”frmAnimal”>
<input type=”text” name=”txtAnimal” /><br />
<input type=”button” value=”Check animal” onclick=”checkAnimal()” />
</form>
```

```
function checkAnimal() {
  switch (document.frmAnimal.txtAnimal.value){
    case "rabbit": alert("Watch out, it’s Elmer Fudd!")
    break;
    case "coyote": alert("No match for the road runner - meep
    meep!")
    break;
    case "mouse": alert("Watch out Jerry, here comes Tom!")
    break;
    default : alert("Are you sure you picked an animal from a
    cartoon?");
  }
}
```

Zanastardust/ Flickr

Loops!

Md. Masudur Rahman, Dept. Of CSE, UGV

while

In a while loop, a code block is executed if a condition is true and for as long as that condition remains true.

The syntax is as follows:

```
while (condition)
{
   code to be executed
}
```


What this code does?

```
<script type="text/JavaScript">
 var i = 1;
  while (i < 11) {
    document.write(i + " x 3 = " + (i * 3) +
     "<br/>" );
    i++;
  }
</script>
```

while Example

do..while

- A do ... while loop executes a block of code once and then checks a condition.
- For as long as the condition is true it continues to loop. So, whatever the condition, the loop runs at least once
- Here is the syntax:

```
do
{
   code to be executed
}
while (condition)
```

What this code does?

```
<script type="text/JavaScript">
  var i = 12;
  do {
    document.write(i + " x 3 = " + (i * 3) + "<br
     />" );
    i++;
  }
  while (i < 11)
</script>
```

for

- The for statement executes a block of code a specified number of times.
- Use it when you know how many times you want the code to be executed rather than running while a particular condition is true/false.
- Syntax:

```
for (a; b; c)
{
    code to be executed
}
```

a is evaluated before the loop is run, and is only evaluated once.

b should be a condition that indicates whether the loop should be run again. if it returns true the loop runs again.

c is evaluated after the loop has run.

What this code does?

```
for (i=0; i<20; i++) {
  document.write(i + " x 3 = " + (i * 3)
  + "<br />" );
}
```

What about this one?

```
for (i=0; i<20; i++) {
  document.write(i + " x 3 = " + (i * 3)
  + "<br />" );
  if( i == 10 )
     break;
}
```

Events

Md. Masudur Rahman, Dept. Of CSE, UGV

Events

- Window events: which occur when something happens to a window.
  - For example, a page loads or unloads.
- User events: which occur when the user interacts with elements in the page using a mouse (or other pointing device) or a keyboard.

Events

- onload: Document has finished loading
- onclick: Button on mouse has been clicked over the element.
- ondblclick: Button on mouse has been double-clicked over the element.
- onmousedown: Button on mouse has been pressed (but not released) over the element.
- onmouseup: Button on mouse (or other pointing device) has been released over the element.

More Events

- onmouseover: Button on mouse has been moved onto the element.
- onmousemove: Button on mouse has been moved while over the element.
- onmouseout: Button on mouse has been moved off the element.
- onkeypress: A key is pressed and released over the element.
- onkeydown: A key is held down over an element.
- onkeyup: A key is released over an element. Most elements

Even more Events

- onfocus: Element receives focus either by mouse clicking it, tabbing order giving focus to that element, or code giving focus to the element.
- onblur: Element loses focus.

Forms:

- onsubmit: A form is submitted.
- onreset:A form is reset.
- onselect: User selects some text in a text field.
- onchange:A control loses input focus and its value has been changed since gaining focus.

Built-in Objects!

String, Date

Md. Masudur Rahman, Dept. Of CSE, UGV

String

- Allows you to deal with strings of text:

```
myString = new String('Here is some big text')
document.write(myString);
```

Why we just don't use ?

```
var someText = "Here is some big text";
document.write(someText);
```

String methods

- charAt(index): Returns the character at a specified position.
  - String text = new String("Good Morning");
  - text.charAt(3) would return the letter d.
- link(targetURL): Creates a link
  - text.link("http://www.google.com") ➔
  - Good Morning
- substr(start, length): Returns a substring starting at the position *start* and then *length* consecutive characters.

String methods

- substring(startPosition, endPosition): returns a substring from position *startPosition* to the position *endPosition*

- toLowerCase(): Converts a string to lowercase
  - String text = new String("John Smith")
  - text.toLowerCase() ➔ john smith

- toUpperCase(): Converts a string to uppercase
  - text.toUpperCase() ➔ JOHN SMITH

Date

- Date object helps you work with dates and times
  - new Date()
- To set a specific date or time, you need to pass only one of these parameters:
  - milliseconds: number of milliseconds since 01/01/1970.
  - yr_num, mo_num, day_num: Represents year, month, and day.
  - yr_num, mo_num, day_num, hr_num, min_num, seconds_num, ms_num: Represents the years, days, hours, minutes, seconds, and milliseconds.

Date Methods

- getDate() : returns the date (from 1 to 31)
- getDay(): returns the day (from 0 to 6; Sunday, to Saturday)
- getMonth(): returns the month (from 0 to 11; January to December)
- getYear(): returns the year using two digits
- getFullYear(): returns the year using four-digit year

Date Methods (Cont'd)

- getHours(): returns the hour
- getMinutes(): returns the minute
- getSeconds(): returns the seconds
- getTimezoneOffset(): returns the time difference between the user's computer and GMT

Date Methods (Cont'd)

- setDate(): sets the date of the month (from 1 to 31).
- setFullYear(): sets the year (four digits).
- setHours(): sets the hour (from 0 to 23).
- setMinutes(): Sets the minute (from 0 to 59).
- setMonth(): Sets the month (from 0 to 11; 0=January, 1=February).
- setSeconds(): Sets the second (from 0 to 59).

Dates Examples

- With dateString, and will read Wed Apr 16 00:00:00 UTC+0100 1975:
  - var birthDate = new Date("April 16, 1975")
  - document.write(birthDate)

- With yr_num, mo_num, and day_num, and will read Mon May 12 00:00:00 UTC+0100 1975:
  - var birthDate = new Date(1975, 4, 28)
  - document.write(birthDate)

Math

- The math object helps in working with numbers.
- Example:

```
numberPI = Math.PI
document.write (numberPI)
```

- Another Example:

```
numberPI = Math.PI
numberPI = Math.round(numberPI)
document.write (numberPI)
```

Math methods

- abs(x): Returns the absolute value of *x*.
- acos(x): Returns the arccosine of *x*.
- asin(x): Returns the arcsine of *x*.
- atan(x): Returns the arctangent of *x*.
- atan2(y,x): Returns the angle from the x-axis to a point.
- ceil(x): Returns the nearest integer greater than or equal to *x*.
- cos(x): Returns the cosine of *x*.
- sqrt(x): Returns the square root of *x*.
- tan(x): Returns the tangent of *x*.

Math methods

- exp(x): Returns the value of E raised to the power of *x*.
- floor(x): Returns the nearest integer less than or equal to *x*.
- log(x): Returns the natural log of *x*.
- max(x,y): Returns the number with the highest value of *x and y*.
- min(x,y): Returns the number with the lowest value of *x and y*.
- pow(x,y): Returns the value of the number *x raised to the power of y*.
- random(): Returns a random number between 0 and 1.
- round(x): Rounds *x to the nearest integer*.
- sin(x): Returns the sine of *x*.

Arrays

- An *array can hold more than one value.*
- *These values* can be accessed individually.
- We need to use a Constructor to build up the array:

```
instruments = new Array("guitar", "drums","piano")
```

- The elements of the array are indexed using their ordinal number, starting at 0.
  - instruments[0] ➔ returns the guitar.
  - instruments[1] ➔ returns the drums.
  - instruments[2] ➔ returns the piano.

Arrays (Cont'd)

- If you do not want to provide all the values when you create the array, you can just indicate how many elements you want to be able to hold:

  var instruments = new Array(3)

- To change the size of an array, just modify the length property, such as:

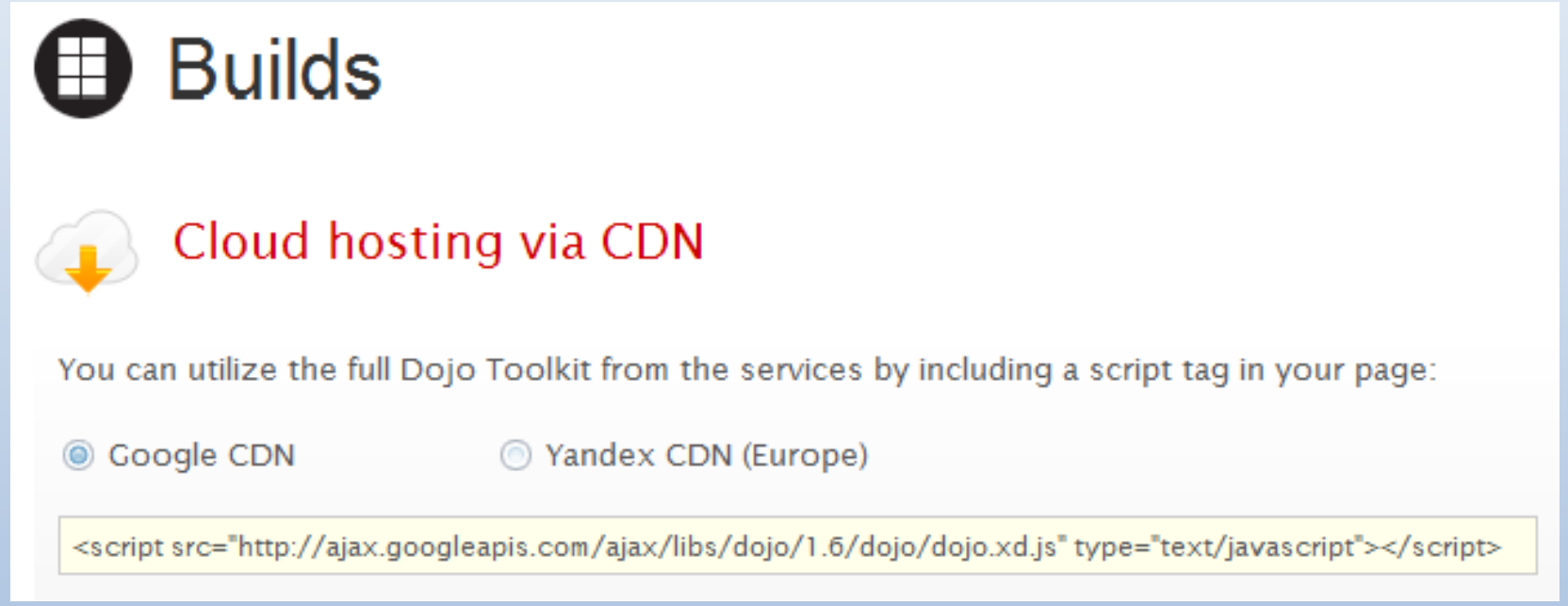
  instruments.length = 5

  document.write(instruments.length) ➔ will print 5

Array Methods

- concat(): Joins two or more arrays to create one new one.
- join(separator): Joins all of the elements of an array together separated by the character specified as a separator (the default is a comma);
- reverse(): Returns the array reversed
- slice(): Returns a specified part of the array
- sort(): Returns a sorted array

Get Dojo !



<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js" type="text/javascript"></script>

Questions?

Web Programming

Week 16
Slides 532-554

Md. Masudur Rahman, Dept. Of CSE, UGV

Outline

- Form validation
- Other forms techniques
- Navigation

JavaScript Tips

Md. Masudur Rahman, Dept. Of CSE, UGV

Tips when using JavaScript

1. Do not invent the wheel! Reuse other's people JavaScript libraries. Google them!
   - www.HotScripts.com
   - www.JavaScriptKit.com
2. Reuse your own functions:

```
<form name="frmLoanCalc" onsubmit="calculateLoan( )">
    Loan Amount: <input type="text" name="txtAmount"/>
    Repayment: <input type="text" name="txtRepayment"/>
    Interest rate: <input type="text" name="txtInterest"/>
```

```
function calculateLoan() {
   loanAmount = document.frmLoanCalc.txtAmount.value
   loanValue = document.frmLoanCalc.txtRepayment.value
   interestRate = document.frmLoanCalc.txtInterest.value
/*… more code …*/
```

Tips when using JavaScript (Cont'd)

```
<form name=”frmLoanCalc” onsubmit=”calculateLoan( )”> Loan
   Amount: <input type="text" name="txtAmount1"/> Repayment:
   <input type="text" name=" txtAmount2"/> Interest rate: <input
   type="text" name=" txtRate "/>
```

```
function calculateLoan() {
   loanAmount = document.frmLoanCalc.txtAmount1.value
   loanValue = document.frmLoanCalc.txtAmount2.value
   interestRate = document.frmLoanCalc.txtRate.value
/*… more code …*/
```

Better Approach:

```
function calculateLoan(loanAmount, repaymentPeriod, interestRate){
/*… more code …*/
}
```

Tips when using JavaScript (Cont'd)

3. Use External JavaScript files!
   - <script type="text/javascript" src="js/script.js"></script>
4. Place your scripts in a script folder: Folders are commonly named:
   - js
   - script
   - javascript

Form Validation

Md. Masudur Rahman, Dept. Of CSE, UGV

Form Validation

- You cannot check whether user has entered correct information
- However, you can check that user has entered the information in the correct format.
- Form are usually validated using the onsubmit event.
- The JavaScript function must return true so the form can be submitted. Otherwise, it will not be submitted.

Checking Text Fields

Let's Create our Validate function

```
function validate(form) {
  var returnValue = true;
  var username = frmRegister.txtUserName.value;
  var password1 = frmRegister.txtPassword.value;
  var password2 = frmRegister.txtPassword2.value;
/*username length should be longer than six
  characters*/
  if(username.length < 6) {
    returnValue = false;
    alert("Your username must be at least\n6
      characters long.\n
    Please try again.");
    frmRegister.txtUserName.focus();
  }
```

Check password length

```
/*username length should be longer than six
  characters*/
if (password1.length < 6) {
  returnValue = false;
  alert("Your password must be at least\n6
    characters long.\n Please try again.");
  frmRegister.txtPassword.value = "";
  frmRegister.txtPassword2.value = "";
  frmRegister.txtPassword.focus();
}
```

Passwords should match

```
/*passwords should match*/
if (password1.value != password2.value) {
  returnValue = false;
  alter("Your password entries did not
    match.\nPlease try again.");
  frmRegister.txtPassword.value = "";
  frmRegister.txtPassword2.value = "";
  frm Register.txtPassword.focus();
}

/* Finally return variable "returnValue" */
  return returnValue;
}
```

Making Required fields!

- How can we create required fields into a form?
- For example, a registration form that requires your username, first name, last name, password and email address to have a valid value.
- However, middle name, and cell phone are not required.

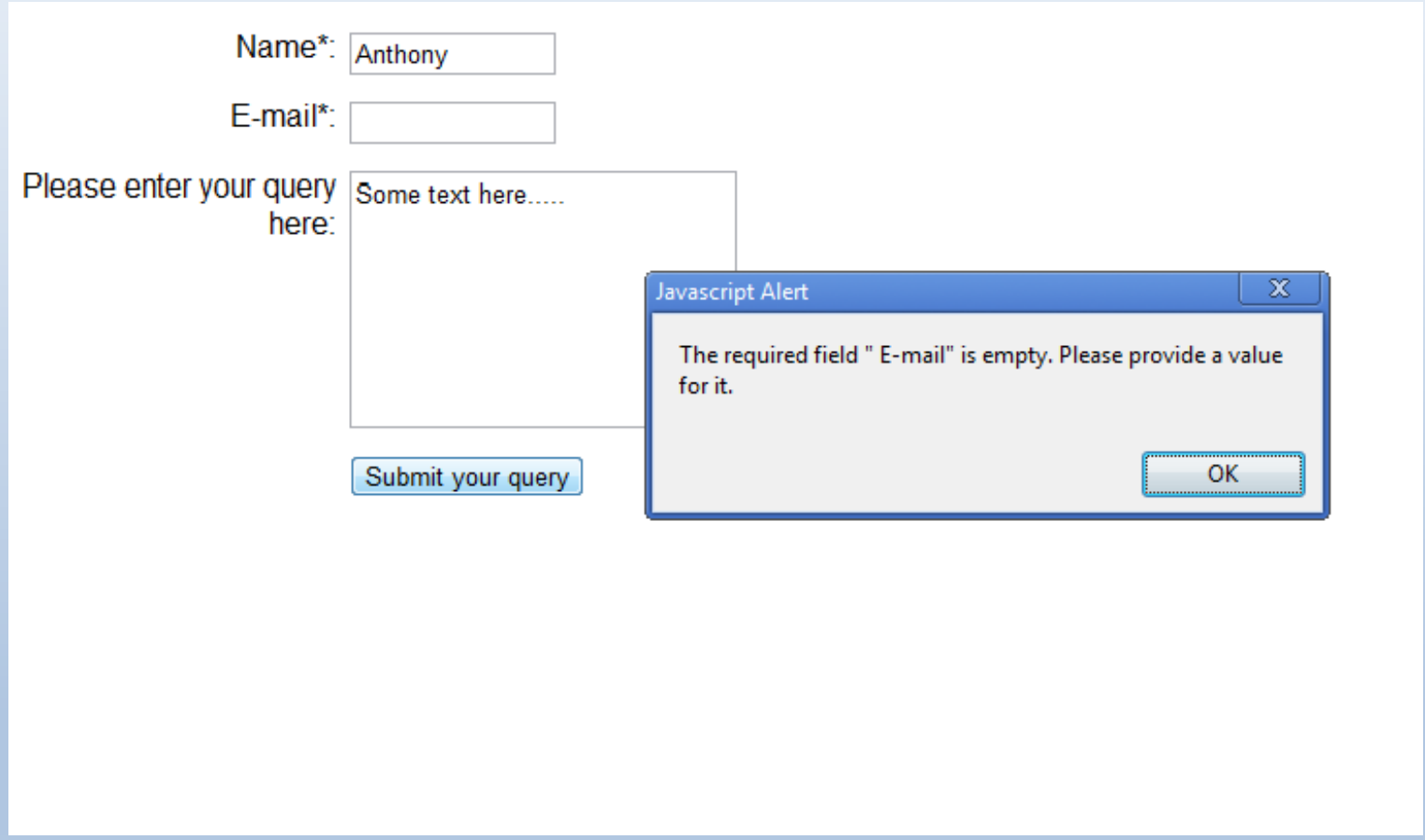
Making Required Form Fields

```
<form name="frmEnquiry" method="post"
  action="gotoSomewhere" onsubmit="return
  validate(this);">
  Name:<input type="text" class="required"
    name="Name"/>
  E-mail:<input type="text" class="required"
    name="E-mail"/>
  Please enter your query here:
  <textarea rows="8" class="required" cols="30"
    name="query"></textarea>
  <input type="submit" class="" value="Submit your form"/>
</form>
```

JavaScript function

```
function validate(form) {
  var returnValue=true;
  var formElements = form.elements;

  for (var i=0; i<formElements.length; i++){
     currentElement = formElements[i];
     if (currentElement.value=="" &&
  currentElement.className=="required"){
     /*Show a respective alert message */
       currentElement.focus();
       returnValue = false;
      break;
      }
  }
  return returnValue;
}
```

Form with required fields



Playing with Random and Images

Md. Masudur Rahman, Dept. Of CSE, UGV

Random

- The random() method generates a random number between 0 and 1
- Syntax:
  - Math.random()

- Also, we know that with JavaScript we can change the source of an image by modifying their property src

Random Images Example

```
function randomImage(){
  var images=new Array()
  images[0]="google1.png"
  images[1]="google2.jpg"
  images[2]="google3.png"
  images[3]="google4.gif"
  images[4]="google5.jpg"

  var i=Math.floor(Math.random()*images.length)
  document.images.imgBanner.src = "img/" + images[i];
}
```

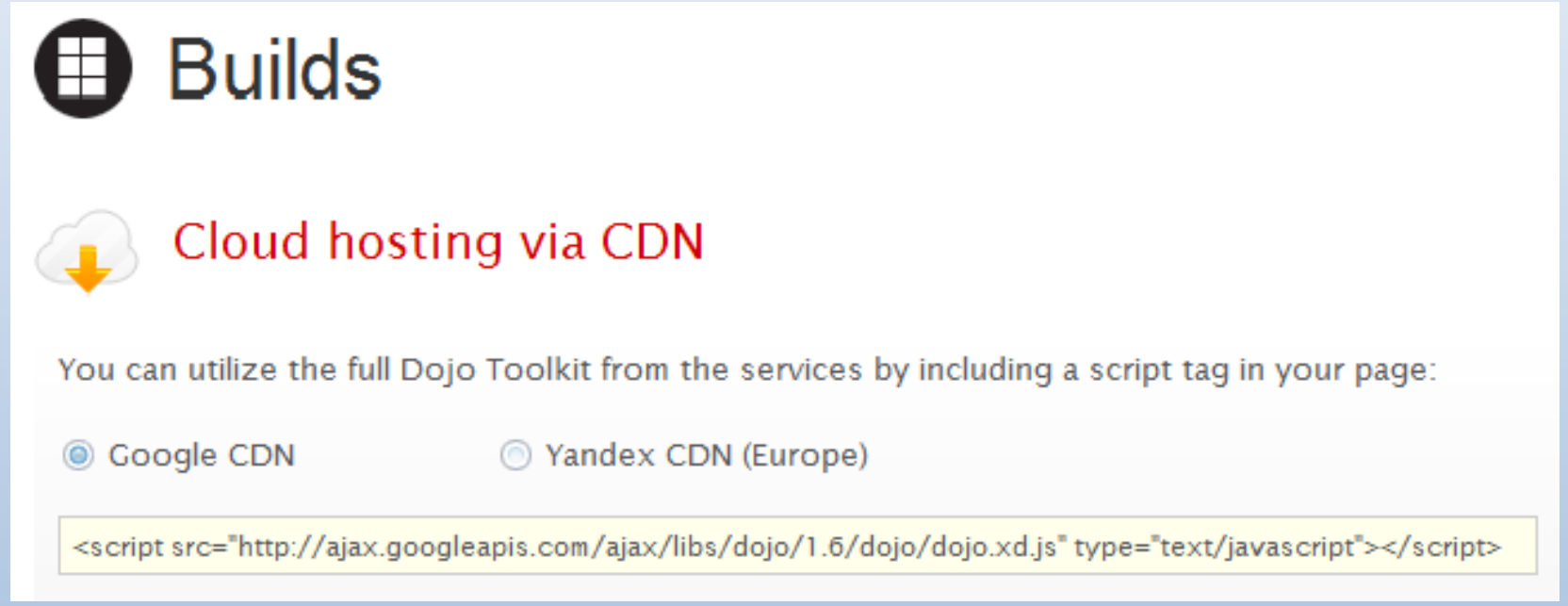
Random Images Example: HTML

```
<body onload="randomImage()">
<img name="imgBanner"
  src="img/google1.png" alt="a banner
  image"/>
</body>
```

The body onload event triggers startSlideShow()

The image changes randomly

Get Dojo !



<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js" type="text/javascript"></script>

Questions?

Web Programming

Week 17
Slides 555-556

Md. Masudur Rahman, Dept. Of CSE, UGV

Project Presentation & Review

Md. Masudur Rahman, Dept. Of CSE, UGV