

CS1019 - WEB TECHNOLOGY

- **UNIT 1 INTRODUCTION** **9**
- Internet Principles – Basic Web Concepts – Client/Server model – retrieving data from Internet – HTM and Scripting Languages – Standard Generalized Mark –up languages – Next Generation – Internet –Protocols and Applications

Internet v.s. Web

- **The Internet:** a inter-connected computer networks, linked by wires, cables, wireless connections, etc.
- **Web:** a collection of interconnected documents and other resources.
- The world wide web (**WWW**) is accessible via the Internet, as are many other services including email, file sharing, etc.

Basics web concepts

- **Hypertext**: a format of information which allows one to move from one part of a document to another or from one document to another through **hyperlinks**
- Uniform Resource Locator (**URL**): unique identifiers used to locate a particular resource on the network
- **Markup language**: defines the structure and content of hypertext documents

Terms & Definitions

Client

A **client** is the *requesting program* in a client/server relationship, **e.g.**, the user of a *Web browser* is effectively making **client** requests for pages from servers all over the Web.

Server

In general, a **server** is a computer program that provides services to other computer programs in the same or other computers.

Web browser:

The web client, called a **browser**, is the software that allows you to interact with information available on the Internet.

e.g Netscape Navigator, Microsoft Internet Explorer, MOSAIC.

Web Page:

A mixture of text, graphics, sound and animation in the HTML format, to make information accessible in a easy to understand format using the Internet.

Web Site:

A collection of web pages connected (linked) by Hypertext clickable links.

Web Site Storage/Hosting:

After a web site is designed it must be stored on a computer that can be accessed through the Internet and the World-Wide Web .

World-Wide Web:

The World-Wide Web (WWW) is a pair of software applications, which allow both distribution of and access to information on the Internet. The ***web*** is *not* the ***Internet*** but a means of distributing and accessing the information that is on it.

ISP (Internet Service Provider):

- An ISP is a company that provides individuals and other companies access to the Internet and other related services such as Web site development and hosting (web site storage).
- The larger ISPs have their own high-speed leased lines so that they are less dependent on the telecommunication providers and can provide better service to their customers.
- List the Internet Service Providers in INDIA

How does the Internet Work?

- Through communication protocols
- A **communication protocol** is a specification of how communication between two computers will be carried out
 - **IP** (Internet Protocol): defines the packets that carry blocks of data from one node to another
 - **TCP** (Transmission Control Protocol) and **UDP** (User Datagram Protocol): the protocols by which one host sends data to another.
 - Other application protocols: **DNS** (Domain Name Service), **SMTP** (Simple Mail Transmission Protocol), and **FTP** (File Transmission Protocol), **HTTP**

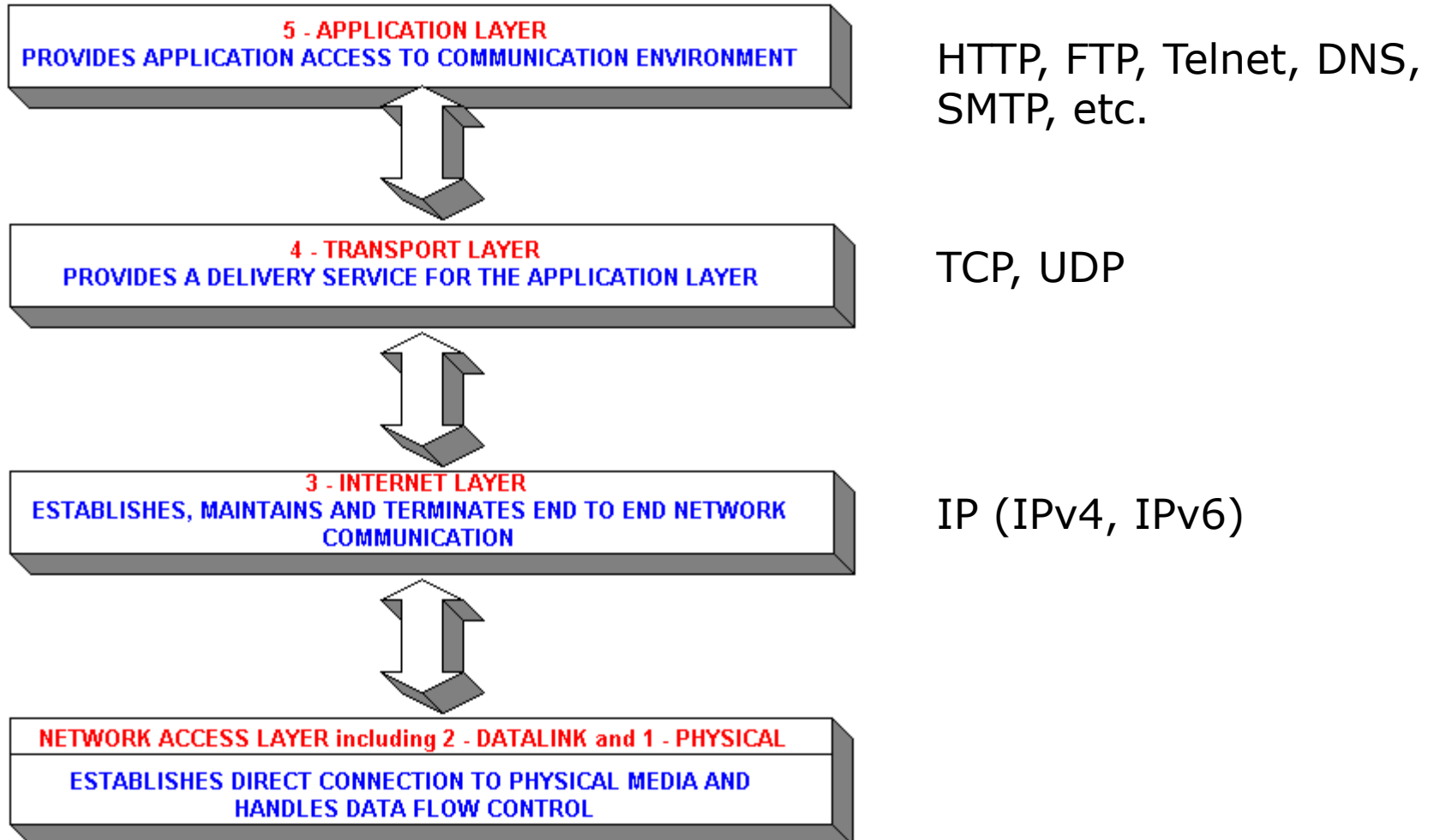
The Internet Protocol (IP)

- A key element of IP is **IP address**, a 32-bit number
- The Internet authorities assign ranges of numbers to different organizations
- IP is responsible for moving **packet** of data from node to node
- A packet contains information such as the data to be transferred, the source and destination IP addresses, etc.
- Packets are sent through different local network through **gateways**
- A **checksum** is created to ensure the correctness of the data; corrupted packets are discarded
- IP-based communication is **unreliable**

The Transmission Control Protocol (TCP)

- TCP is a higher-level protocol that extends IP to provide additional functionality: **reliable** communication
- TCP adds support to detect errors or lost data and to trigger **retransmission** until the data is correctly and completely received
- Connection
- Acknowledgment

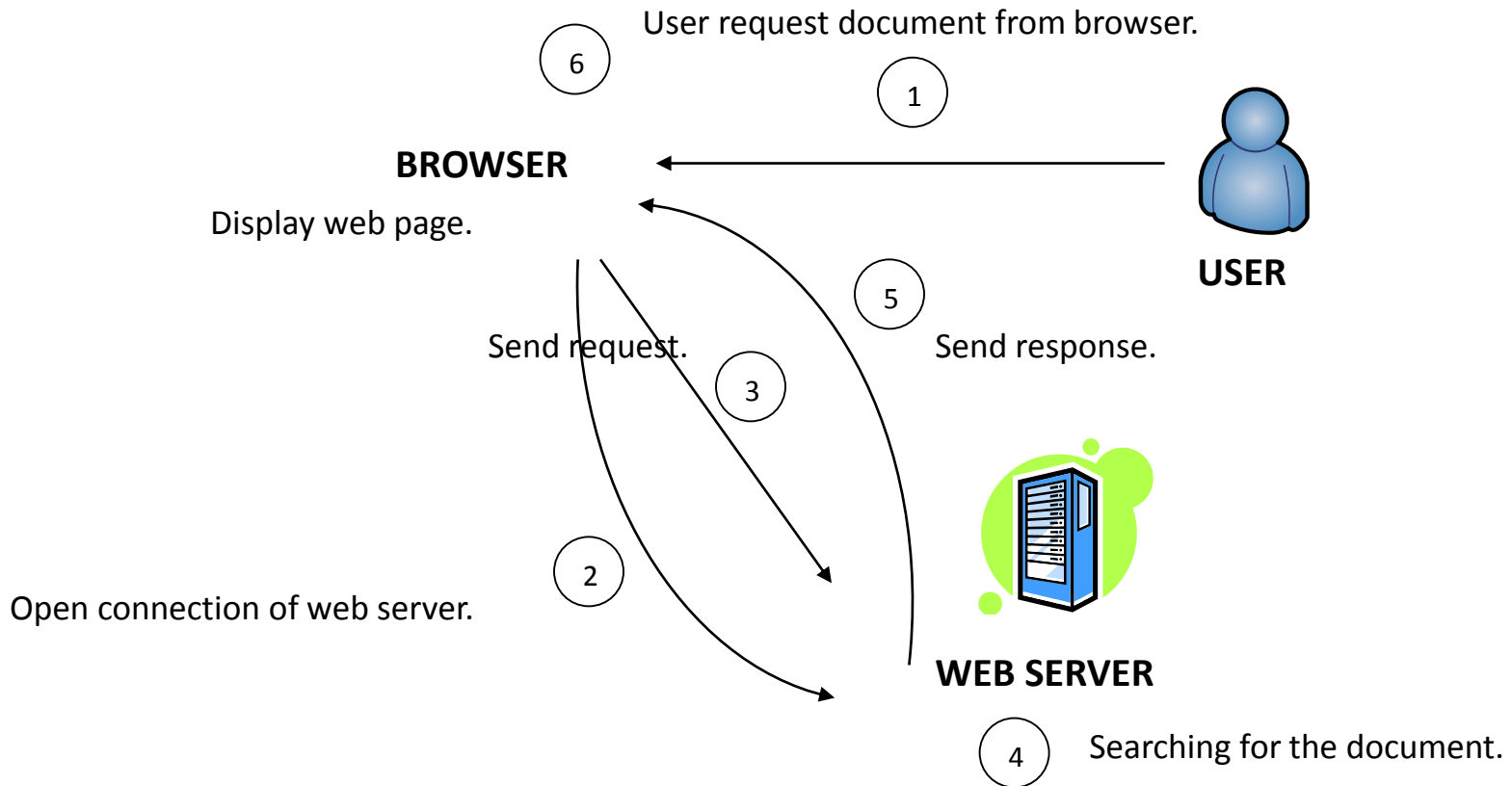
TCP/IP Protocol Suites



HTTP:

The Hypertext Transfer Protocol is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

HTTP (HyperText Transfer Protocol):



Web Client: Browser

- Makes HTTP requests on behalf of the user
 - Reformat the URL entered as a valid HTTP request
 - Use DNS to convert server's host name to appropriate IP address
 - Establish a TCP connection using the IP address
 - Send HTTP request over the connection and wait for server's response
 - Display the document contained in the response
 - If the document is not a plain-text document but instead is written in HTML, this involves rendering the document (positioning text, graphics, creating table borders, using appropriate fonts, etc.)

Web Servers

- Main functionalities:
 - Server waits for connect requests
 - When a connection request is received, the server creates a new process to handle this connection
 - The new process establishes the TCP connection and waits for HTTP requests
 - The new process invokes software that maps the requested URL to a resource on the server
 - If the resource is a file, creates an HTTP response that contains the file in the body of the response message
 - If the resource is a program, runs the program, and returns the output

Client-Side and Server-side Programming

- Client-side code
 - ECMAScript
 - JavaScript, JScript – Microsoft
 - VBScript – Microsoft
 - Embedded in <script> elements and execute in the browser, provides immediate feedback to the user.
 - Reduces the load on a server, reduces network traffic.
- Server-side code
 - Execute on the server
 - CGI/Perl, ASP, PHP, ColdFusion, JSP
 - The code remains hidden from users, and browser independent.
- Can be combined with good results.

Client-side & Server-side Technologies

Client-Side	Server-Side
<p>HTML, XML</p> <p>Cascading Style Sheets (CSS)</p> <p>Scripting languages</p> <ul style="list-style-type: none">- JavaScript, VBScript <p>Java Applets</p> <p>ActiveX controls</p> <p>Plug-ins and Helpers application</p>	<p>CGI/Perl</p> <p>PHP</p> <p>ColdFusion</p> <p>Scripting Languages</p> <ul style="list-style-type: none">- Server-side JavaScript- ASP, JSP, Java Servlets <p>ISAPI/NSAPI programs</p>

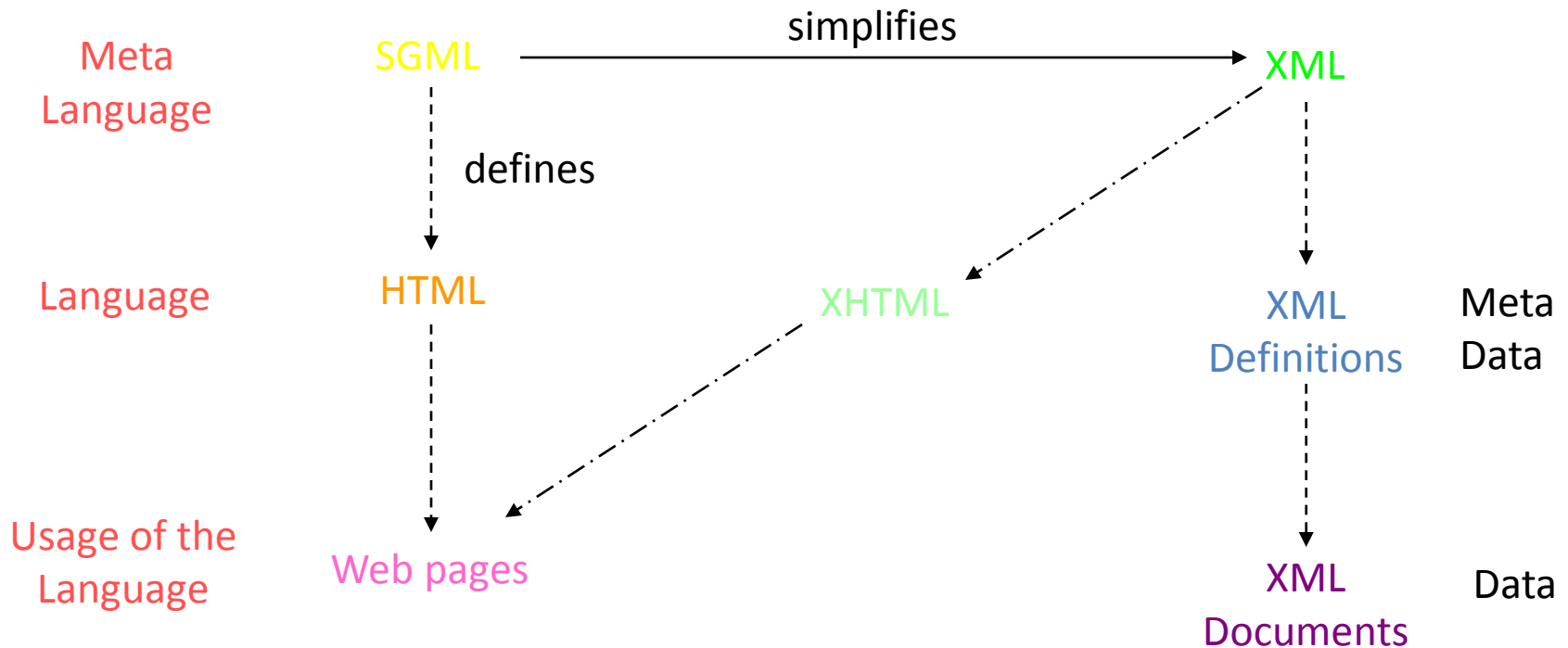
Web Technologies

- HTML
- XHTML
- CSS
- XML
- JavaScript
- VBSCRIPT
- DOM
- DHTML
- AJAX
- E4X
- WMLScript
- SQL
- ASP
- ADO
- PHP
- .NET
- SMIL
- SVG
- FLASH
- Java applets
- Java servlets
- Java Server Page

The History of Markup

- In the early 1970s
 - GML (the Generalized Markup Language)
 - “:h1.The Content is placed here”
- Since the 1980s
 - SGML (the Standard GML)
 - HTML
- Currently
 - XML
 - Not intended to replace HTML!
 - XHTML does by providing better data description, ...

SGML, HTML and XML



HTML

- HyperText Markup Language
- It is not a programming language.
 - Cannot be used to describe computations.
 - Use to describe the general form and layout of documents to be displayed by the browser.
- Compose of “Content” and “Controls”

Introduction to HTML

What is HTML?

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- Telling the browser what to do, and what props to use.
- A series of tags that are integrated into a text document.

Tags are:

- surrounded with angle brackets like this

✉ or <I>.

- Most tags come in pairs

✉ exceptions: <P>,
, tags ...

- The first tag turns the action on, and the second turns it off.

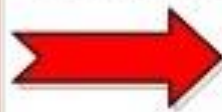
Introduction to HTML

- 📄 The second tag (off switch) starts with a forward slash.
 - ✉ For example , text
- 📄 can embedded, for instance, to do this:
 - ✉ <HEAD><TITLE> Your text </HEAD></TITLE> it won't work.
 - ✉ The correct order is <HEAD><TITLE> Your text </TITLE></HEAD>
- 📄 not case sensitivity.
- 📄 Many tags have attributes.
 - ✉ For example, <P ALIGN=CENTER> centers the paragraph following it.
- 📄 Some browsers don't support the some tags and some attributes.

Basic HTML Document Format

```
<HTML>
<HEAD>
<TITLE>WENT'99</TITLE>
</HEAD>
<BODY>
  Went'99
</BODY>
</HTML>
```

See what it
looks like:



Introduction to HTML

How to Create and View an HTML document?

1. Use a text editor such as Notepad to write the document.
2. Save the file as filename.html on a PC. This is called the Document Source.
3. Open Netscape (or any browser) Off-Line
4. Switch to Netscape
5. Click on File, Open File and select the filename.html document that you just created.
6. Your HTML page should now appear just like any other Web page in Netscape.

7. You may now switch back and forth between the Source and the HTML Document

- switch to Notepad with the Document Source
- make changes
- save the document again
- switch back to Netscape
- click on RELOAD and view the new HTML Document
- switch to Notepad with the Document Source.....

HTML Basic Tags

Tags in head

- ⌘ **<HEAD>...</HEAD>**-- contains information about the document
- ⌘ **<TITLE>...</TITLE>**-- puts text on the browser's title bar.

HTML Basic Tags

Tags in Body

» Tag	Description
» <h1> to <h6>	Defines header 1 to header 6
» <p>	Defines a paragraph
» 	Inserts a single line break
» <hr>	Defines a horizontal rule
» <!-->	Defines a comment

HTML Basic Tags

Tags in Body

Text Formatting Tags

» Tag	Description
» 	Defines bold text
» <big>	Defines big text
» 	Defines emphasized text
» <i>	Defines italic text
» <small>	Defines small text
» 	Defines strong text
» <sub>	Defines subscripted text
» <sup>	Defines superscripted text

Add Images (HTML Image Tag)

✧ Use tags

✧ Attributes of IMG tag

- width,height

- Alt

- Align

- <img src=my.gif width=50 height=50 align=right
alt="My image">

Add some link (Anchor Tag)

✧ Use tags

✧ Kind of URLs:

- <http://www.women.or.kr>
- <ftp://ftp.foo.com/home/foo>
- <mailto:skrhee@women.or.kr>

The HTML style Element

- » The <style> tag is used to define style information for an HTML document.
- » Inside the style element you specify how HTML elements should render in a browser:
- »

```
<head>  
<style type="text/css">  
body { background-color:yellow }  
p { color:blue }  
</style>  
</head>
```

HTML Lists

- The most common HTML lists are ordered and unordered lists:

HTML Unordered Lists

- An unordered list starts with the tag.
- Each list item starts with the tag.
- The list items are marked with bullets (typically small black circles).

Coffee

Milk

O/P:

- Coffee
- Milk

HTML Ordered Lists

- An ordered list starts with the tag.
- Each list item starts with the tag.
- The list items are marked with numbers.

Coffee

Milk

O/P:

1. Coffee
2. Milk

HTML Definition Lists

- A definition list is a list of items, with a description of each item.
- The <dl> tag defines a definition list.
- The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list).

<dl>

<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

O/P:

Coffee

- black hot drink

Milk

- white cold drink

HTML Tables

- Tables are defined with the <table> tag.
- A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag).

```
<table border="1">
```

```
<tr>
```

```
<td>row 1, cell 1</td>
```

```
<td>row 1, cell 2</td>
```

```
</tr>
```

```
<tr>
```

```
<td>row 2, cell 1</td>
```

```
<td>row 2, cell 2</td>
```

```
</tr>
```

```
</table>
```

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

HTML Table Headers

Header information in a table are defined with the <th> tag

```
<table border="1">  
<tr>    <th>Header 1</th>  
        <th>Header 2</th>  
</tr><tr>  
    <td>row 1, cell 1</td>  
    <td>row 1, cell 2</td>  
</tr><tr>  
    <td>row 2, cell 1</td>  
    <td>row 2, cell 2</td>  
</tr></table>
```

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

HTML Table Row span

An HTML table with a table cell that spans two rows:

```
<table border="1">
```

```
<tr>
```

```
<th>Month</th>
```

```
<th>Savings for holiday!</th>
```

```
</tr> <tr>
```

```
<td>January</td>
```

```
<td rowspan="2">$50</td>
```

```
</tr> <tr>
```

```
<td>February</td>
```

```
</tr></table>
```

Month	Saving for holiday!
January	\$50
February	

HTML Table col span

An HTML table with a table cell that spans two columns:

```
<table border="1">
```

```
  <tr>    <th>Month</th>
          <th>Savings</th>
```

```
</tr><tr>
```

```
    <td>January</td>
```

```
    <td>$100</td>
```

```
</tr><tr>
```

```
    <td>February</td>
```

```
    <td>$100</td>
```

```
</tr><tr>
```

```
    <td colspan="2">Sum: $200</td>
```

```
</tr></table>
```

Month	Saving
January	\$100
February	\$100
Sum: \$200	

HTML Frames

- With frames, several Web pages can be displayed in the same browser window.
- Each HTML document is called a frame, and each frame is independent of the others.
- The disadvantages of using frames are:
 - a.) Frames are not expected to be supported in future versions of HTML
 - b.) Frames are difficult to use.
 - c.) The web developer must keep track of more HTML documents

The HTML frameset Element

- The frameset element holds one or more frame elements.
- Each frame element can hold a separate document.
- The frameset element states HOW MANY columns or rows there will be in the frameset,
- and HOW MUCH percentage/pixels of space will occupy each of them.

The HTML frame Element

- The <frame> tag defines one particular window (frame) within a frameset.

```
<frameset cols="25%,75%">  
  <frame src="frame_a.htm" />  
  <frame src="frame_b.htm" />  
</frameset>
```

e.g.:

```
<frameset cols="50%,50%">  
<frameset rows="50%,50%">  
  <frame src="col1row1.html">  
  <frame src="col1row2.html">  
</frameset>  
<frame src="col2.html">  
</frameset>
```

Col1row1.html

Col2.html

Col1row2.html

HTML Forms

- HTML forms are used to pass data to a server.
- A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more.
- The `<form>` tag is used to create an HTML form.
- The most important form element is the input element.
- The input element is used to select user information.

The <form> Tag

- The `<form arguments> ... </form>` tag encloses form elements (and probably other HTML as well)
- The arguments to `form` tell what to do with the user input
 - `action="url"` (required)
 - Specifies where to send the data when the **Submit** button is clicked
 - `method="get"` (default)
 - Form data is sent as a URL with `?form_data` info appended to the end
 - Can be used *only* if data is all ASCII and not more than 100 characters

- `method="post"`
 - Form data is sent in the body of the URL request
 - Cannot be bookmarked by most browsers
- `target="target"`
 - Tells where to open the page sent as a result of the request
 - `target=_blank` means open in a new window
 - `target=_top` means use the same window

The <input> Tag

- Most, but not all, form elements use the **input** tag, with a **type="..."** argument to tell which kind of element it is
 - **type** can be **text**, **checkbox**, **radio**, **password**, **hidden**, **submit**, **reset**, **button**, **file**, or **image**
- Other common **input** tag arguments include:
 - **name**: the name of the element

- **value**: the “value” of the element; used in different ways for different values of **type**
- **readonly**: the value cannot be changed
- **disabled**: the user can’t do anything with this element
- Other arguments are defined for the **input** tag but have meaning only for certain values of **type**

Text Input

- A text field
- `<input type="text" name="textfield" value="with an initial value">`

A text field:

- A multi-line text field
- `<textarea name="textarea" cols="24" rows="2">Hello</textarea>`

A multi-line text field:

- A password field
- `<input type="password" name="textfield3" value="secret">`

A password field:

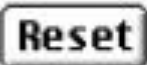
- Note that two of these use the **input** tag, but one uses **textarea**

Buttons


- A submit button
`<input type="submit" name="Submit" value="Submit">`
- A reset button:
`<input type="reset" name="Submit2" value="Reset">`
- A plain button:
`<input type="button" name="Submit3" value="Push Me">`

A submit button: 

- **submit**: send data

A reset button: 

- **reset**: restore all form elements to their initial state

A plain button: 

- **button**: take some action as specified by JavaScript

- Note that the type is **input**, not “button”

- A checkbox:
`<input type="checkbox" name="checkbox" value="checkbox" checked>`

A checkbox: ☒

- `type: "checkbox"`
- `name`: used to reference this form element from JavaScript
- `value`: value to be returned when element is checked
- Note that there is *no text* associated with the checkbox— you have to supply text in the surrounding HTML

- **Radio buttons:**

```
<input type="radio" name="radiobutton" value="myValue1">male<br>  
<input type="radio" name="radiobutton" value="myValue2"  
checked>female
```

Radio buttons:

☐ male

☒ female

- If two or more radio buttons have the same **name**, the user can only select one of them at a time
 - This is how you make a radio button “group”
- If you ask for the value of that **name**, you will get the **value** specified for the selected radio button
- As with checkboxes, radio buttons do not contain any text

A menu or list

```
<select name="select">  
  <option value="red">red</option>  
  <option value="green">green</option>  
  <option value="BLUE">blue</option></select>
```

A menu or list: 

- Additional arguments:
 - **size**: the number of items visible in the list (default is "1")
 - **multiple**: if set to "true", any number of items may be selected (default is "false")

A complete example

```
<html>
<head><title>Get Identity</title>
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" value="m">Male
    <input type="radio" name="gender" value="f">Female</p>
</form>
</body>
</html>
```

Who are you?

Name:

Gender: ☐ Male ☐ Female

Buttons


- A submit button
`<input type="submit" name="Submit" value="Submit">`
- A reset button:
`<input type="reset" name="Submit2" value="Reset">`
- A plain button:
`<input type="button" name="Submit3" value="Push Me">`

A submit button: 

- **submit**: send data

A reset button: 

- **reset**: restore all form elements to their initial state

A plain button: 

- **button**: take some action as specified by JavaScript

- Note that the type is **input**, not “button”

- A checkbox:
`<input type="checkbox" name="checkbox" value="checkbox" checked>`

A checkbox: ☒

- **type:** "checkbox"
- **name:** used to reference this form element from JavaScript
- **value:** value to be returned when element is checked
- Note that there is *no text* associated with the checkbox— you have to supply text in the surrounding HTML

- **Radio buttons:**

```
<input type="radio" name="radiobutton" value="myValue1">male<br>  
<input type="radio" name="radiobutton" value="myValue2"  
checked>female
```

Radio buttons:

☐ male

☒ female

- If two or more radio buttons have the same **name**, the user can only select one of them at a time
 - This is how you make a radio button “group”
- If you ask for the value of that **name**, you will get the **value** specified for the selected radio button
- As with checkboxes, radio buttons do not contain any text

A menu or list

```
<select name="select">  
  <option value="red">red</option>  
  <option value="green">green</option>  
  <option value="BLUE">blue</option></select>
```

A menu or list:

A screenshot of a web browser's dropdown menu. The menu is rectangular with a thin border. Inside, the word "red" is displayed in a bold, black, sans-serif font. To the right of the text is a small square button containing a black downward-pointing arrow.

- Additional arguments:
 - **size**: the number of items visible in the list (default is "1")
 - **multiple**: if set to "true", any number of items may be selected (default is "false")

A complete example

```
<html>
<head><title>Get Identity</title>
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" value="m">Male
    <input type="radio" name="gender" value="f">Female</p>
</form>
</body>
</html>
```

Who are you?

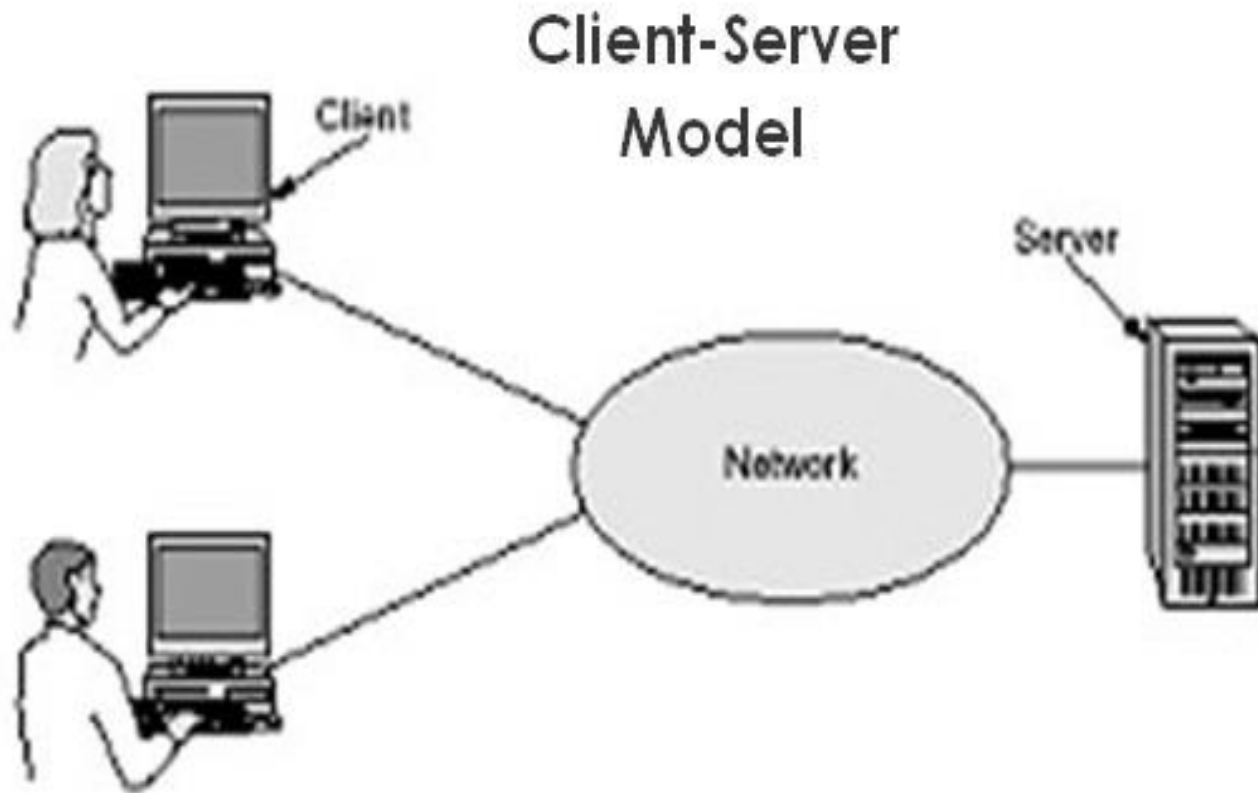
Name:

Gender: ☐ Male ☐ Female

Client-Server Model

MODEL 1:

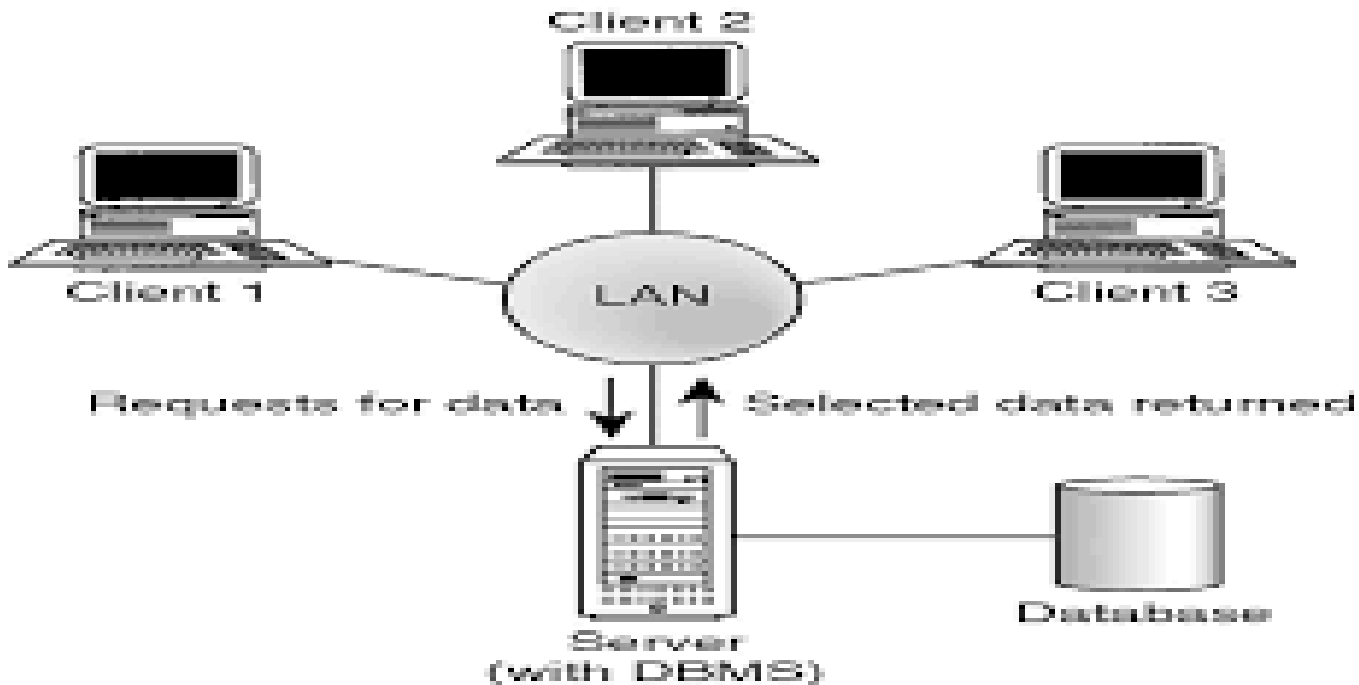
This is the simplest client/ server model in a web architecture



MODEL 2:

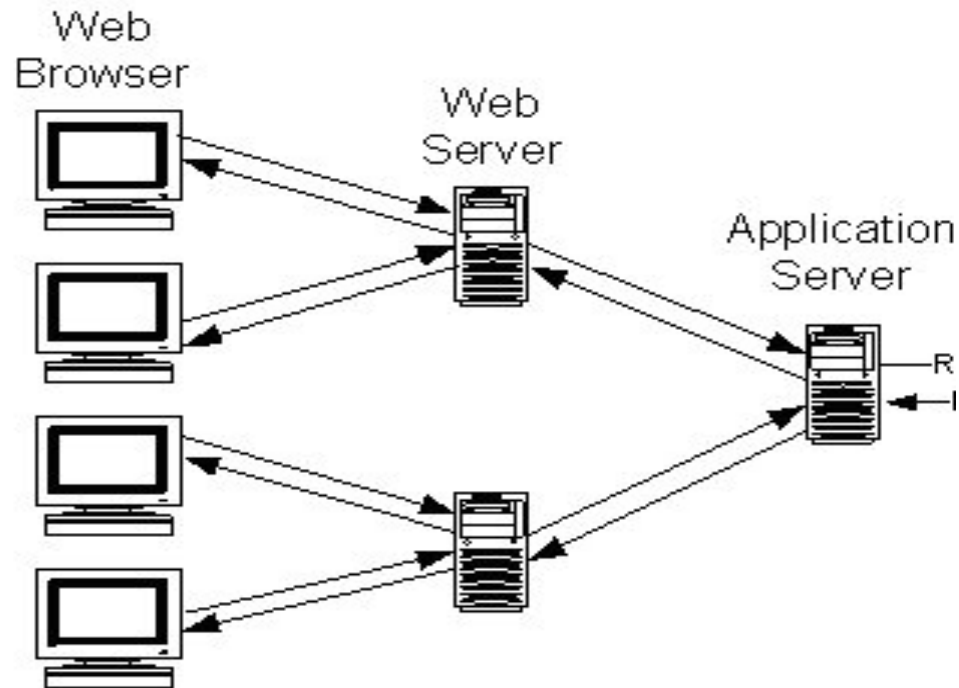
Here the web server will have Database related operations.

If the clients request query it will be carried out from the database server.



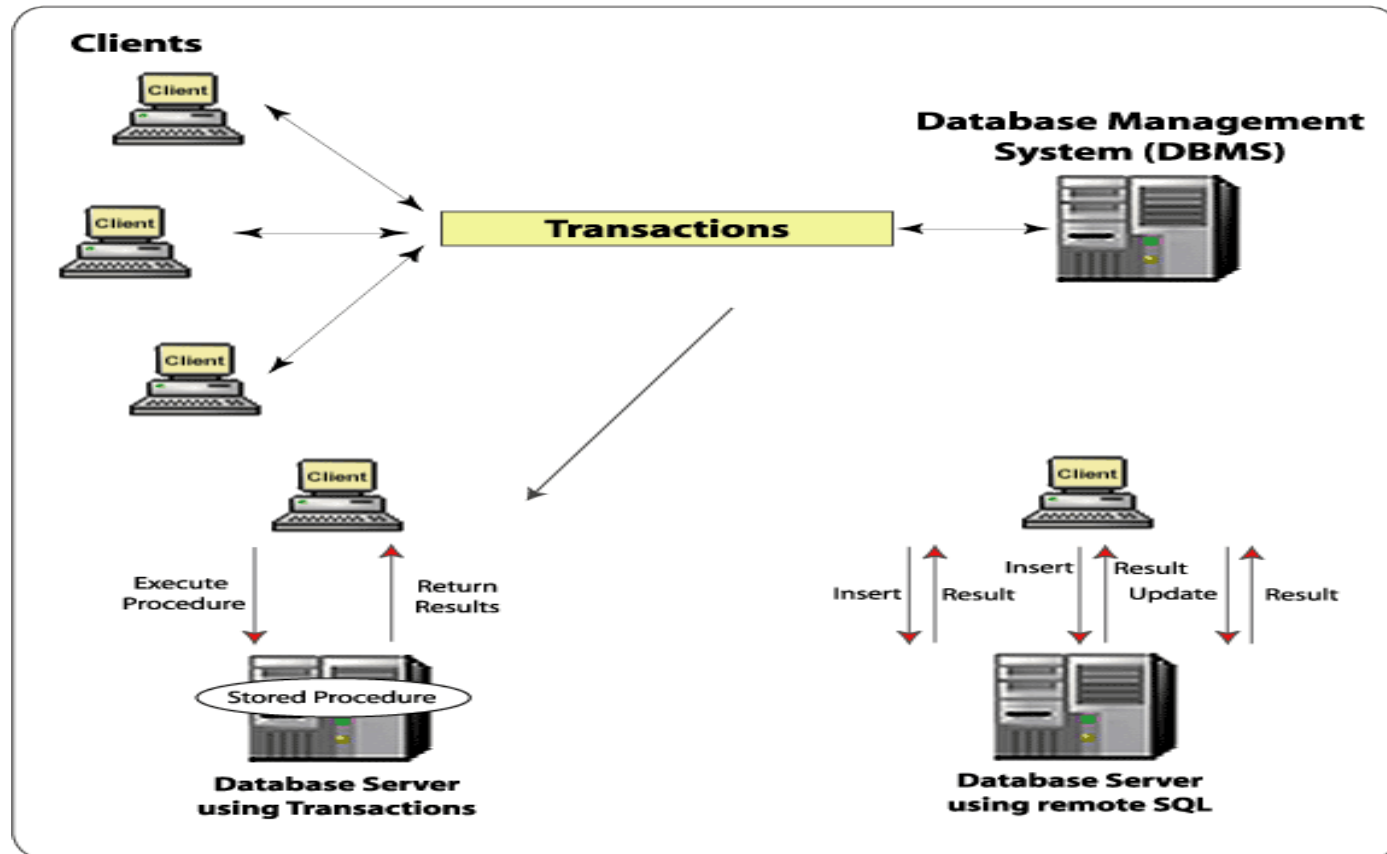
MODEL 3:

The web server communicates with the application server.



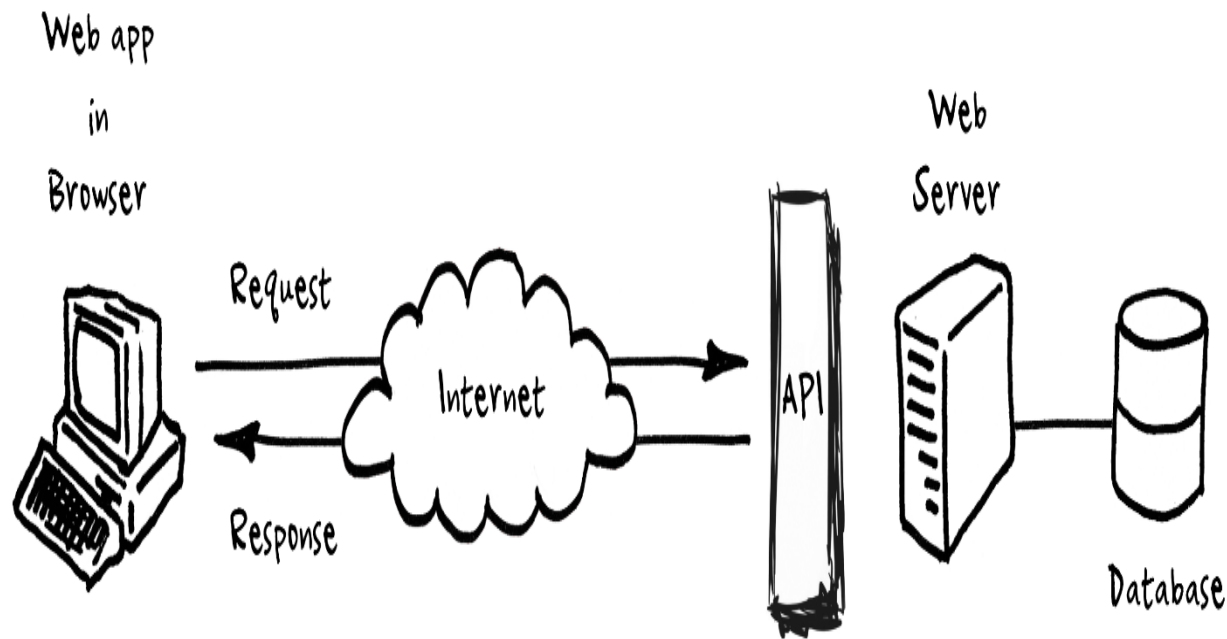
MODEL 4:

This model is the ideal architecture for online business applications.



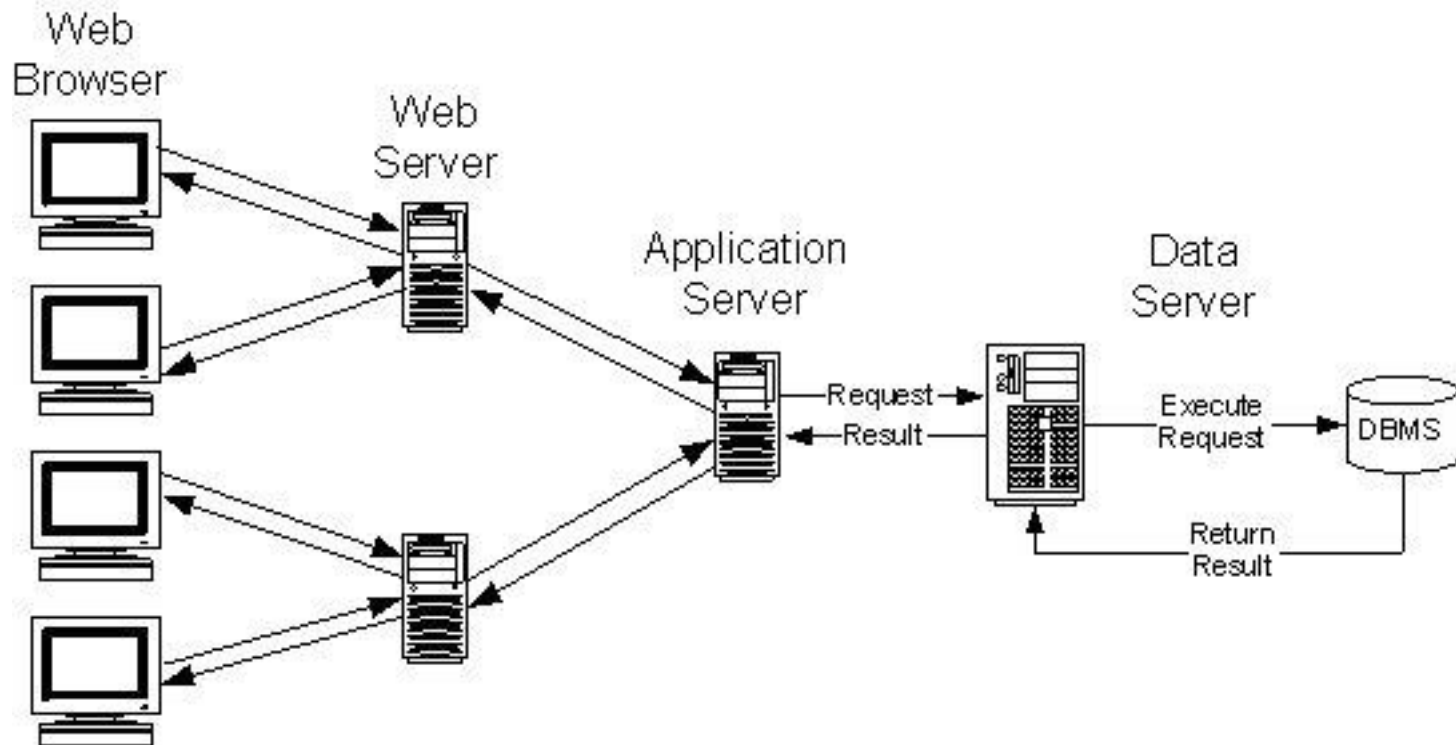
MODEL 5:

The transaction server ensures the ACID properties during transaction.

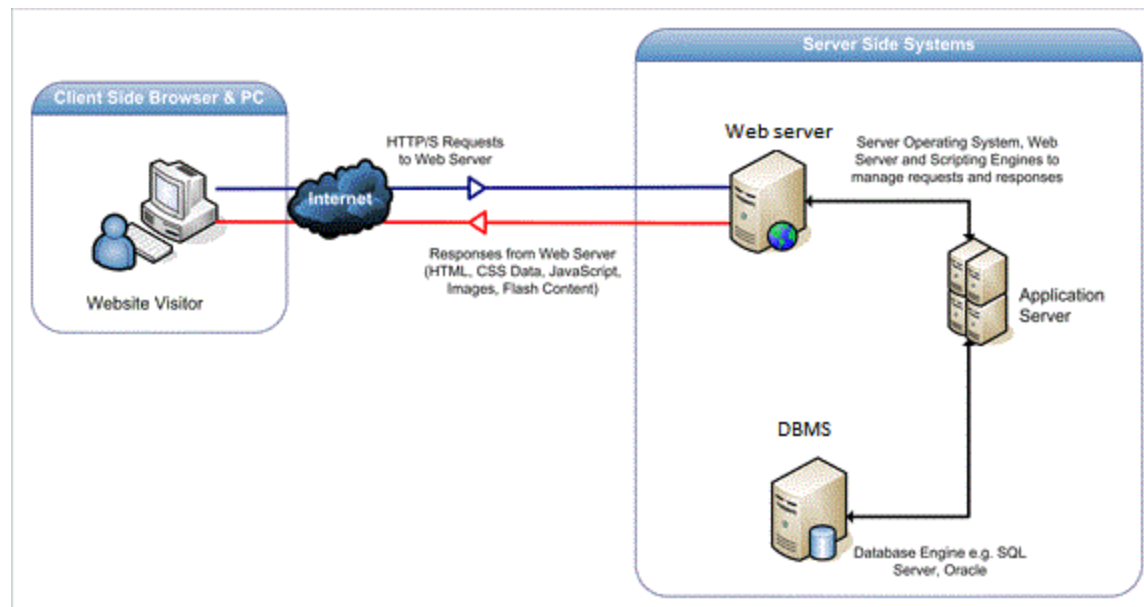


MODEL 6:

Most widely used architecture for the web applications



- Following are some of the key differences in features of Web Server and Application Server:
- Web Server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC
- Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content.



- Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally App Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc.
- As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

- Example of such configuration is Apache Tomcat HTTP Server and Oracle (formerly BEA) WebLogic Server. Apache Tomcat HTTP Server is Web Server and Oracle WebLogic is Application Server.
- In some cases the servers are tightly integrated such as IIS and .NET Runtime. IIS is web server. When equipped with .NET runtime environment, IIS is capable of providing application services.

- SGML

What is CSS?

- **CSS** stands for **Cascading Style Sheets**.
- Styles define **how to display** HTML elements.
- Styles are normally stored in **Style Sheets**.
- Styles were added to HTML 4.0 **to solve a problem**.
(Netscape vs MS Internet Explorer)
- **External Style Sheets** can save you a lot of work.
- External Style Sheets are stored in **CSS files**.
- Multiple style definitions will **cascade** into one. priority:
 - Inline Style (inside HTML element)
 - Internal Style Sheet (inside the <head> tag)
 - External Style Sheet
 - Browser default style

CSS Versions

- The World Wide Web Consortium (<http://www.w3.org/>) has published two main CSS recommendations of style sheets - CSS1 and CSS2.
- CSS1 became a recommendation in 1996.
- CSS2 became a recommendation in 1998.

CSS Syntax

- » A CSS rule has two main parts: a selector, and one or more declarations:



- » The selector is normally the HTML element you want to style.
- » There are mainly two types of selectors:
 - > ID Selector
 - > Class Selector

How to use style sheets

There are three ways of inserting a style sheet:

- » External style sheet
- » Internal style sheet
- » Inline style

External Style Sheet

- » An external style sheet is ideal when the style is applied to many pages.
- » With an external style sheet, you can change the look of an entire Web site by changing one file.
- » Each page must link to the style sheet using the `<link>` tag.
- » The `<link>` tag goes inside the head section:
- »

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

External Style Sheet

Myhtml.html

...

```
<head>
```

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

```
</head>
```

....

Mystyle.css

```
hr {color: sienna}
```

```
p {margin-left: 20px}
```

```
body {background-image: url("images/back40.gif")}
```

Internal Style Sheets

- » An internal style sheet should be used when a single document has a unique style.
- » Define internal styles in the head section of an HTML page, by using the `<style>` tag, like this:
- » `<head>`
`<style type="text/css">`
`p{color:red;} </style> </head>`
`<body> <p> This is my first sheet</p> </body>`

Inline Style Sheets

- » An inline style loses many of the advantages of style sheets by mixing content with presentation.
- » To use inline styles you use the style attribute in the relevant tag.
- » The style attribute can contain any CSS property.
- » `<head>`

This is a document `</head>`

`<body style="background-color:blue;">`

`<p> This is my first sheet</p> </body>`

ID Selector

- » The id selector is used to specify a style for a single, unique element.
- » The id selector uses the id attribute of the HTML element, and is defined with a "#".

E.g.

```
#para1  
{  
text-align:center;  
color:red;  
}
```


Class Selector

- » The class selector is used to specify a style for a group of elements.
- » Unlike the id selector, the class selector is most often used on several elements.
- » This allows you to set a particular style for any HTML elements with the same class.
- » The class selector uses the HTML class attribute, and is defined with a “.”

E.g.

```
.para1  
{  
text-align:center;  
color:red;  
}
```

CSS, Why ?

- By editing a single CSS file, you can make site-wide design changes in seconds.
- CSS lets you output to multiple formats quickly.
- CSS lets you use logical names for page elements. You can, for example, give a DIV the name "header", or a H1 the class "headline". It's self-describing.
- External CSS files are cached by browsers, improving load time.
- CSS eliminates the need for messy code -- namely font tags, spacer GIFs and nested tables.
- CSS lets you do things normal HTML doesn't. Examples: better font control, absolute positioning, nifty borders.
- Practical use of CSS encourages proper HTML structure, which will improve accessibility and search engine placement.
- CSS's :hover PseudoClass cuts down on the need to use JavaScript onmouseover calls.
- If you want valid XHTML Strict you have to use it anyway.

XML

- » XML stands for eXtensible Markup Language.
- » A markup language is used to provide information about a document.
- » Tags are added to the document to provide the extra information.
- » HTML tags tell a browser how to display the document.
- » XML tags give a reader some idea what some of the data means.
- » A meta language that allows you to create and format your own document markups

Difference Between HTML and XML

- » HTML tags have a fixed meaning and browsers know what it is.
- » XML tags are different for different applications, and users know what they mean.
- » HTML tags are used for display.
- » XML tags are used to describe documents and data.

Quick Comparison

- HTML
 - tags and attributes are pre-determined and rigid
 - content and formatting can be placed together
`<p><font="Arial">text`
 - Designed to represent the presentation structure of a document. Thus, more effective for machine-human interaction
 - Loose syntax compared to XML
- XML
 - allows user to specify what each tag and attribute means
 - content and format are separate; formatting is contained in a style sheet
 - Designed to represent the logical structure of a document. Thus, more effective for machine-machine interaction
 - Syntax is strictly defined

Advantages of XML

- » XML is text (Unicode) based.
 - > Takes up less space
 - > Can be transmitted efficiently.
- » One XML document can be displayed differently in different media.
 - > Html, video, CD, DVD,
 - > You only have to change the XML document in order to change all the rest.
- » XML documents can be modularized. Parts can be reused

Other Advantages of XML

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

XML Document

Example of an XML Document

```
<? xml version="1.0" ?>  
<address>  
  <name>Alice Lee</name>  
  <email>alee@aol.com</email>  
  <phone>212-346-1234</phone>  
  <birthday>1995-03-22</birthday>  
</address>
```

The actual benefit of using XML highly depends on the design of the application.

What's in an XML Document

- » Elements
- » Attributes
- » Plus some other details

A simple XML Document

<article>

 <author>Gerhard Weikum</author>

 <title>The Web in Ten Years</title>

 <text> <abstract>In order to evolve...</abstract>

 <section number="1" title="Introduction">

 The <index>Web</index> provides the universal...

 </section> </text>

</article>

A simple XML Document

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

Freely definable tags



A simple XML Document

<article>

Start Tag

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

<text>

<abstract>In order to evolve...</abstract>

<section number="1" title="Introduction">

The <index>Web</index> provides the universal...

</section>

</text>

</article>

End Tag

Element

Content of the
Element
(Subelements and/or
Text)

A simple XML Document

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

**Attributes with
name and value**

Elements in XML Documents

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

Elements V/s Attributes

Elements may have **attributes** (in the start tag) that have a **name** and a **value**, e.g. `<section number="1">`.

What is the difference between elements and attributes?

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

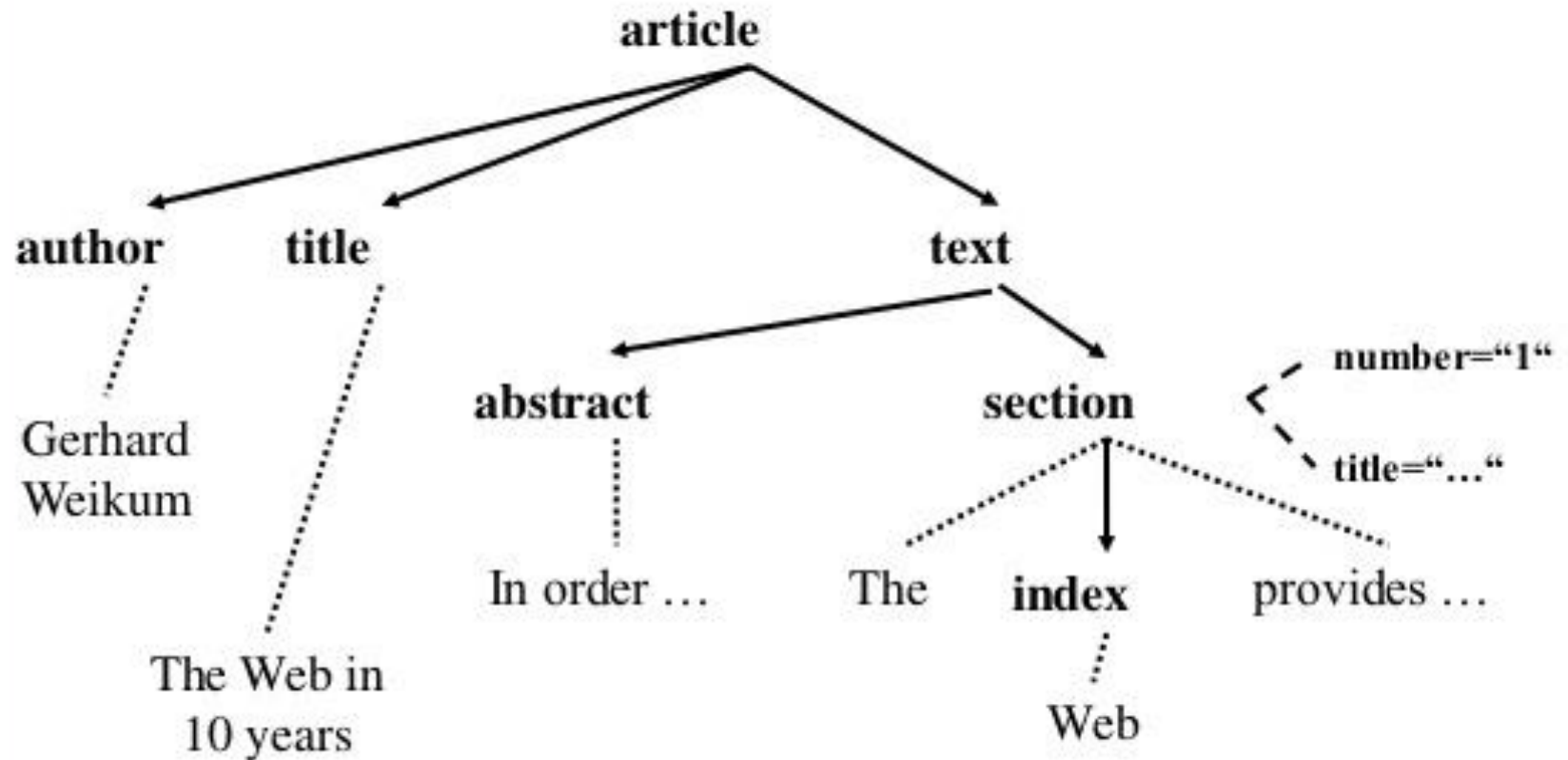
As a rule of thumb:

- » Content into elements
- » Metadata into attributes

Example:

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

XML Document as Ordered Tree



Well formed XML Documents

A **well-formed** document must adhere to, among others, the following rules:

- » Every start tag has a matching end tag.
- » Elements may nest, but must not overlap.
- » There must be exactly one root element.
- » Attribute values must be quoted.
- » An element may not have two attributes with the same name.
- » Comments and processing instructions may not appear inside tags.
- » No unescaped < or & signs may occur inside character data.

Well formed XML Documents

A **well-formed** document must adhere to, among others, the following rules:

- » Every start tag has a matching end tag.
- » Elements must be properly nested.
- » There must be exactly one root element.
- » Attributes must be properly quoted.
- » An element cannot have both content and attributes.
- » Coarse-grained elements cannot be nested inside fine-grained elements.
- » No two sibling elements can have the same name.

**Only well-formed documents
can be processed by XML
parsers.**

Document Type Declaration (DTD)

Sometimes XML is *too* flexible:

- » Most Programs can only process a subset of all possible XML applications
- » For exchanging data, the format (i.e., elements, attributes and their semantics) must be fixed

Document Type Declarations (DTD) for establishing the vocabulary for one XML application (in some sense comparable to *schemas* in databases)

- » A document is **valid with respect to a DTD** if it conforms to the rules specified in that DTD.
- » Most XML parsers can be configured to validate.

DTD Example: Elements

<!ELEMENT article (title,author+,text)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author (#PCDATA)>

<!ELEMENT text (abstract,section*,literature?)>

<!ELEMENT abstract (#PCDATA)>

<!ELEMENT section (#PCDATA|index)+>

<!ELEMENT literature (#PCDATA)>

<!ELEMENT index (#PCDATA)>

**Content of the title element
is parsed character data**

**Content of the text element may
contain zero or more section
elements in this position**

**Content of the article element is a title element, followed by
one or more author elements,
followed by a text element**

XML Schema

- » An XML Schema describes the structure of an XML document.
- » XML Schema is an XML-based alternative to DTD.
- » The XML Schema language is also referred to as XML Schema Definition (XSD).

An XML Schema:

- » Defines elements that can appear in a document
- » Defines attributes that can appear in a document
- » Defines which elements are child elements

XML Schema

- » Defines the order of child elements
- » Defines the number of child elements
- » Defines whether an element is empty or can include text
- » Defines data types for elements and attributes
- » Defines default and fixed values for elements and attributes

XML Schema V/s DTD

- » XML Schemas are extensible to future additions
- » XML Schemas are richer and more powerful than DTDs
- » XML Schemas are written in XML
- » XML Schemas support data types

XML Schema

Support Data Types

One of the greatest strength of XML Schemas is the support for data types.

With support for data types:

- » It is easier to describe allowable document content
- » It is easier to validate the correctness of data
- » It is easier to work with data from a database
- » It is easier to define data facets (restrictions on data)
- » It is easier to define data patterns (data formats)
- » It is easier to convert data between different data types
- » XML Schemas support namespaces

XML Schema

XML Schemas are Extensible

XML Schemas are extensible, because they are written in XML.

With an extensible Schema definition you can:

- » Reuse your Schema in other Schemas
- » Create your own data types derived from the standard types
- » Reference multiple schemas in the same document

XML Schema Example

```
<?xml version="1.0"?>
```

```
<note>
```

```
  <to>Tove</to>
```

```
  <from>Jani</from>
```

```
  <heading>Reminder</heading>
```

```
  <body>Don't forget me this weekend</body>
```

```
</note>
```

```
<?xml version="1.0"?>
```

```
<xs:schema
```

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  targetNamespace="http://www.w3schools.com"
```

```
  xmlns="http://www.w3schools.com"
```

```
  elementFormDefault="qualified">
```

```
  <xs:element name="note">
```

```
    <xs:complexType><xs:sequence>
```

```
      <xs:element name="to" type="xs:string"/>
```

```
      <xs:element name="from" type="xs:string"/>
```

```
      <xs:element name="heading" type="xs:string"/>
```

```
      <xs:element name="body" type="xs:string"/>
```

```
    </xs:sequence></xs:complexType>
```

```
</xs:element></xs:schema>
```


Referencing XML Schema

Referencing a DTD

```
<!DOCTYPE note SYSTEM  
  "http://www.w3schools.com/dtd/note.dtd">
```

Referencing Schema

```
<note  
  xmlns="http://www.w3schools.com"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.w3schools.com note.xsd">
```

The <schema> Element

The <schema> element is the root element of every XML Schema:

```
<?xml version="1.0"?>
```

```
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3schools.com"
    xmlns="http://www.w3schools.com"
```

```
  >
```

```
  ...
```

```
  </xs:schema>
```

- » **xmlns:xs= URL:** indicates that the elements and data types used in the schema come from the specified URL namespace. It also specifies that the elements and data types that come from the URL namespace should be prefixed with **xs:**.

XML Schema Elements

Simple Elements

- » A simple element is an XML element that contains only text.
- » It cannot contain any other elements or attributes.
- » The syntax for defining a simple element is:

`<xs:element name="xxx" type="yyy"/>`

XML Schema has a lot of built-in data types. The most common types are:

- » `xs:string`
- » `xs:decimal`
- » `xs:integer`
- » `xs:boolean`
- » `xs:date`
- » `xs:time`

Example:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

XML Schema Elements

Complex Elements

A complex element is an XML element that contains other elements and/or attributes.

» There are four kinds of complex elements:

» empty elements

» elements that contain only other elements

» elements that contain `<xs:element name="employee">`

» elements that contain `<xs:complexType>`

» A complex XML element is defined by `<xs:sequence>`

`<employee>`

`<firstname>John<`

`<lastname>Smith<`

`</employee>`

`<xs:element name="firstname" type="xs:string"/>`

`<xs:element name="lastname" type="xs:string"/>`

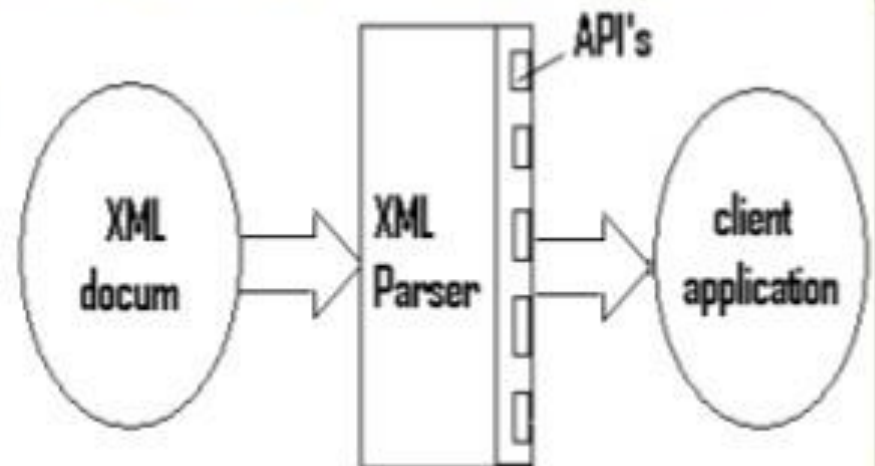
`</xs:sequence>`

`</xs:complexType>`

`</xs:element>`

XML Parsers

- » It is a software library (or a package) that provides methods (or interfaces) for client applications to work with XML documents
- » It checks the well-formattedness
- » It may validate the documents
- » It does a lot of other detailed things so that a client is shielded from that complexities
- » There are two types of XML Parsers:
 - DOM
 - SAX





XML Parsers

DOM Parser

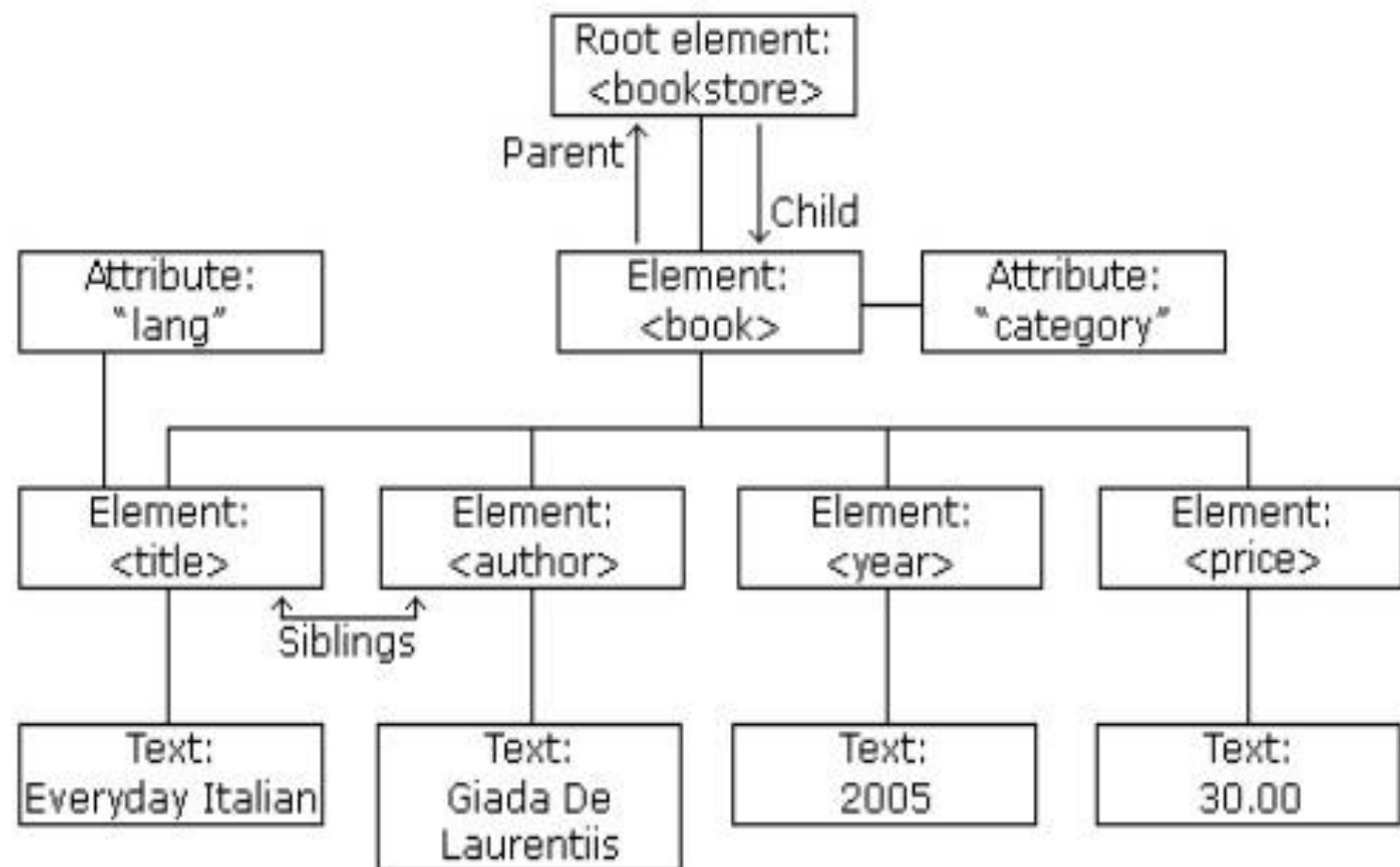
- » The XML DOM (**Document Object Model**) defines a standard way for accessing and manipulating XML documents.
- » The DOM is a W3C (**World Wide Web Consortium**) standard.
- » DOM defines the objects and properties and methods (interface) to access all XML elements.
- » The DOM presents an XML document as a tree structure, with **elements**, **attributes**, and **text** as nodes.
- » **The XML DOM is:**
 - A standard object model for XML
 - A standard programming interface for XML
 - Platform and language-independent
 - A W3C standard

DOM Parser

- » The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.
- » The XML DOM is a standard for **how to get, change, add or delete XML elements**.
- » **Example:**

```
<bookstore>  
<book category="cooking">  
  <title lang="en"> Everyday Italian</title>  
  <author>Giada De Laurentiis</author>  
  <year>2005</year>  
  <price>30.00</price>  
</book>  
</bookstore>
```

DOM Nodes



DOM Nodes

- » According to the DOM, everything in an XML document is a **node**.
- » The entire document is a **document node**.
- » Every XML element is an **element node**.
- » The text in the XML elements are **text nodes**.
- » Every attribute is an **attribute node**.
- » Comments are **comment nodes**.

In our ex.

- The root node <bookstore> holds four <book> nodes.
- Each <book> node contain 4 text node i.e.
 <title>, <author>, <year>, and <price>

DOM Parsing

- » Most browsers have a build-in XML parser to read and manipulate XML.
- » The parser reads XML into memory and converts XML it into XML DOM object which is accessible from **JavaScript** .
- » There are some differences between **Microsoft's XML parser** and the parsers used in other browsers.
- » The MS parser supports *loading of both XML files and XML strings (text)*
- » Other browsers use **Separate parsers**.
- » However, all parsers contain functions to **traverse XML trees, access, insert, and delete nodes**.

DOM Properties and Methods

» XML DOM Properties:

- *x.nodeName* - the name of x
- *x.nodeValue* - the value of x
- *x.parentNode* - the parent node of x
- *x.childNodes* - the child nodes of x
- *x.attributes* - the attributes nodes of x

(where x is a node object.)

» XML DOM Methods:

- *x.getElementsByTagName(name)* - get all elements with a specified tag name
- *x.appendChild(node)* - insert a child node to x
- *x.removeChild(node)* - remove a child node from x

DOM Parser

» Advantages

- It is good when random access to widely separated parts of a document is required.
- It supports both read and write operations

» Disadvantages

- It is memory inefficient
- It seems complicated, although not really

XML Parsers

SAX Parser

- » It does not first create any internal structure .
- » Client does not specify what methods to call.
- » Client just overrides the methods of the API and place his own code inside there.
- » When the parser encounters start-tag, end-tag, etc., it thinks of them as **events**.
- » When such an **event** occurs, the handler automatically ***calls back*** to a particular method overridden by the client, and feeds as arguments the method what it sees.
- » SAX parser is **event-based**.
- » Client application seems to be just **receiving the data** **inactively**, from the data flow point of view.

SAX Parser

» Advantages

- It is simple
- It is memory efficient
- It works well in stream application

» Disadvantage

- The data is broken into pieces and clients never have all the information as a whole unless they create their own data structure