HTML

Markup Languages

- Mark up means some kind of encoding
- When we give a manuscript for typesetting in a press, the instructions to type setter, written on the margins, are markups
- A Markup Language(ML) is a formal mechanism for encoding data for electronic interpretation and presentation.
  Markup Languages :
  - *Procedural ML*
  - *Descriptive ML*

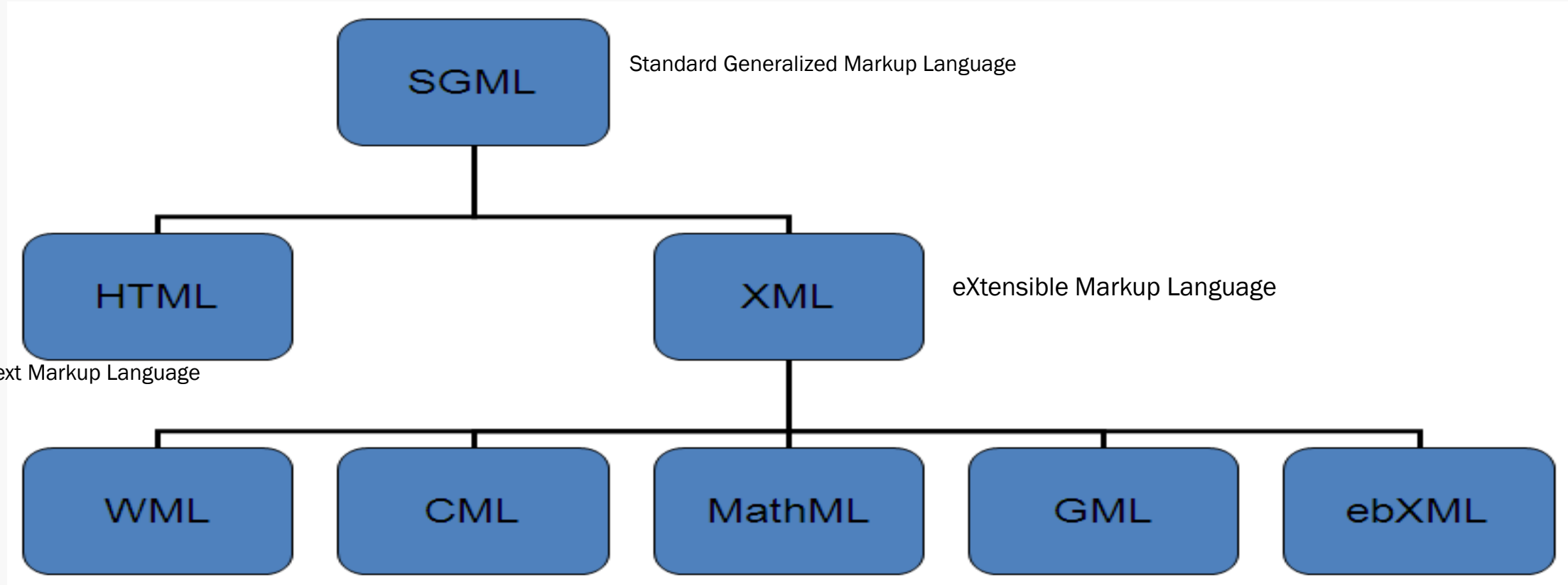
Procedural Markup Languages

- Procedural markups are only machine readable not human readable
- They are application dependent
- Procedural markup languages are used in
  Word Processing software, like MSWORD, DTP software, like PageMaker

Descriptive Markup Languages

- Descriptive Markup Languages use embedded codes (tags) to encode document elements.
- Descriptive markups are both machine as well as human readable
- content of a web page is encoded using descriptive tags and leaves the presentation up to the page presentation software
- A web page encoded with DML is platform independent, since it is simply an ASCII file.
- Examples of DML are
  - *SGML: Standard Generalized Markup Language*
  - *HTML: HyperText Markup Language*
  - *XML: eXtensible Markup Language*

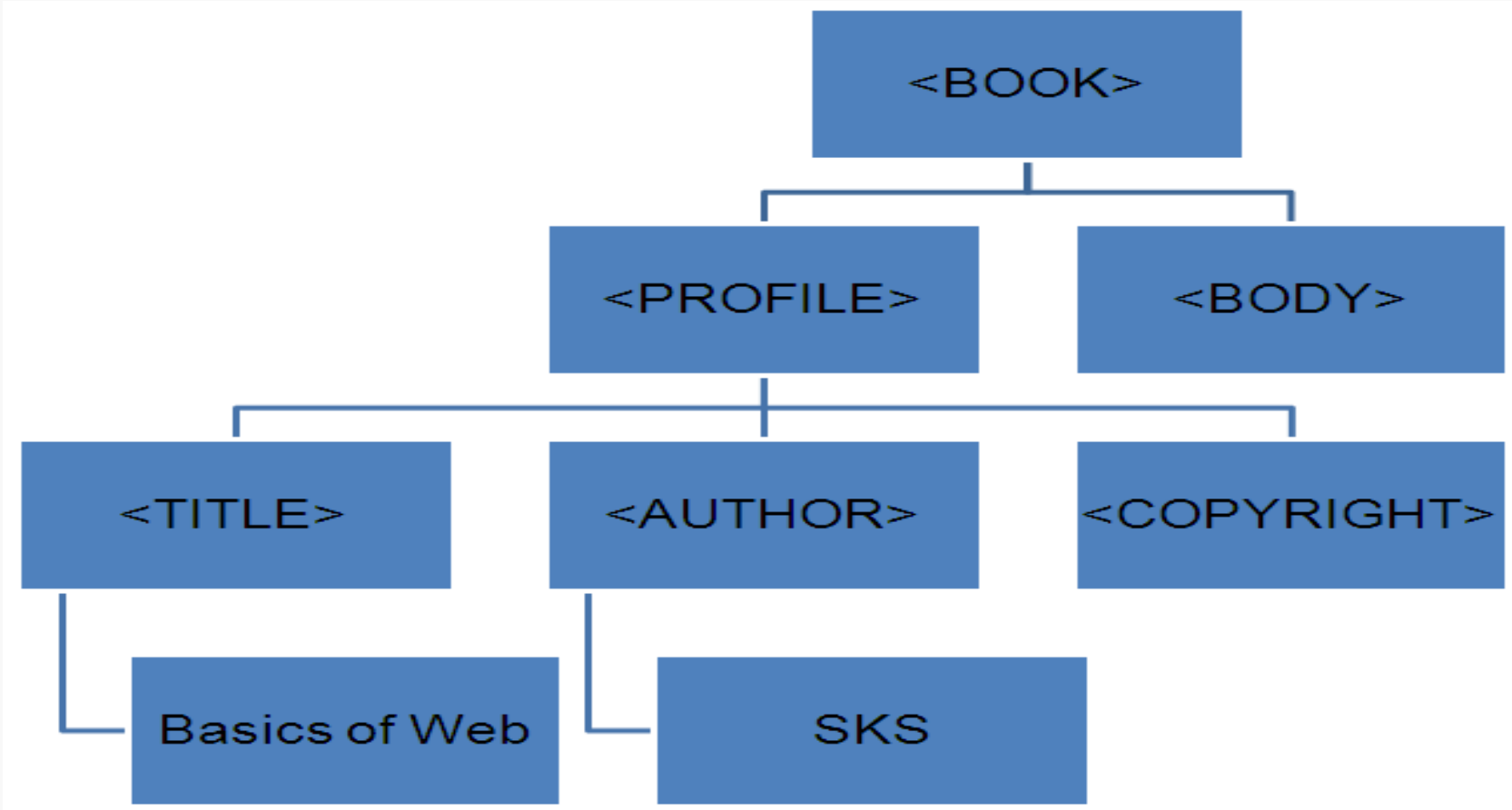
DML Inheritance



Descriptive Markup of an e-Book

```
<BOOK>
    <PROFILE>
        <TITLE>Basics of Web Technology </TITLE>
        <AUTHOR> SKS</AUTHOR>
        <COPYRIGHT>By the author<COPYRIGHT>
    </PROFILE>
     <BODY>
        <CHAPTER ID=1>
                    <TITLE> Introduction</TITLE>
         <SECTION>
          <CHAPTER ID=2>
    .
    .</BODY>
</BOOK>
```

Non-Linear Text



HTML

- Originally designed by TBL deriving traits from pre-existing concepts of SGML and HyperText
- HyperText is a non-linear text
- TBL defined a fixed set of tags and attributes and specified the grammar for encoding HTML pages using these tags and attributes.
- Accordingly HTML parsers are developed for rendering the content for display
- HTML Tags are
  - *Basic :<HTML>, <HEAD>, <BODY>, <IMG> , <A>, <P>*
  - *Lists : <OL> , <UL>, <LI>*
  - *Table: <TABLE>, <TR>, <TD>*
  - *Form: <FORM>, <INPUT>, <SELECT>*
  - *Frame: <FRAMESET>, <FRAME>*

HTML Versions

- **HTML 2.0**: HTML 2.0 was the first classic version of HTML. The HTML 2.0 was introduced on 24 November, 1995. HTML 2.0 constituted almost all the elements but lacked with the extensions for Netscape and Microsoft. In HTML 2.0, tables are poor in aligning attributes.
- **HTML 3.2**: After HTML 2.0, the next major version development was HTML 3 which happened in late 1995. But HTML 3 development did not get completed and it was never implemented. The next major release after HTML 2.0 was HTML 3.2, a version release by W3c.
- **HTML 4.0:** HTML 4.0 was the next major release of HTML. HTML 4.0 was introduced in December 1997. HTML 4 is an SGML application.
- **HTML 4.0.1:** HTML 4.0.1 is the current official version of HTML. HTML 4.0.1 enhanced most trademarked extensions. In December 1999.
- **HTML 5 :** HTML 5 is the latest version of HTML. HTML 5 came into life on January 2008.

Tags

- HTML tags are the hidden keywords within a web page that define how your web browser must format and display the content
- Ex. <html></html>, <img/>

Tag Attributes

- Attributes allows to customize a tag, and are defined within the opening tag,

<img src="image1.jpg">

or

 …

HTML

- Originally designed by Timothy John Berners-Lee (TBL)
- Accordingly HTML parsers are developed for rendering the content for display
- Example of HTML Tags are
  - *Basic :<html>, <head>, <body>, <img> , <a>, <p>*
  - *Lists : <ol> , <ul>, *
  - *Table: <table>, <tr>, <td>*
  - *Form: <form>, <input>, <select>*

Basic Structure of HTML page

```
<html>
<head>
</head>
<body>
        Your Contents for display
</body>
</html>
```

HEAD, BODY, HEADING

- An HTML page has two sections
- <head>
  - *<title>*
- <body>
  - *Actual Data*
  - *<p> Paragraph*
  - *Headings: <h1>, <h2>, <h3>, <h4>, <h5>, <h6>*

HEAD, BODY, HEADING

- An HTML page has two sections
- <HEAD>
  - *Contains meta data of the page*
  - *<TITLE>, <META>, <BASE>*
- <BODY>
  - *Actual Data*
  - *<P> Paragraph*
  - *Headings: <H1>, <H2>, <H3>, <H4>, <H5>, <H6>*

HTML Basic Text Formatting

 Write paragraph here

 Left-aligned paragraph

 Right-aligned paragraph

<big> relatively bigger font size</big>

<small> relatively smaller font size </small>

_{subscript}

^{superscript}

 bold

 italic

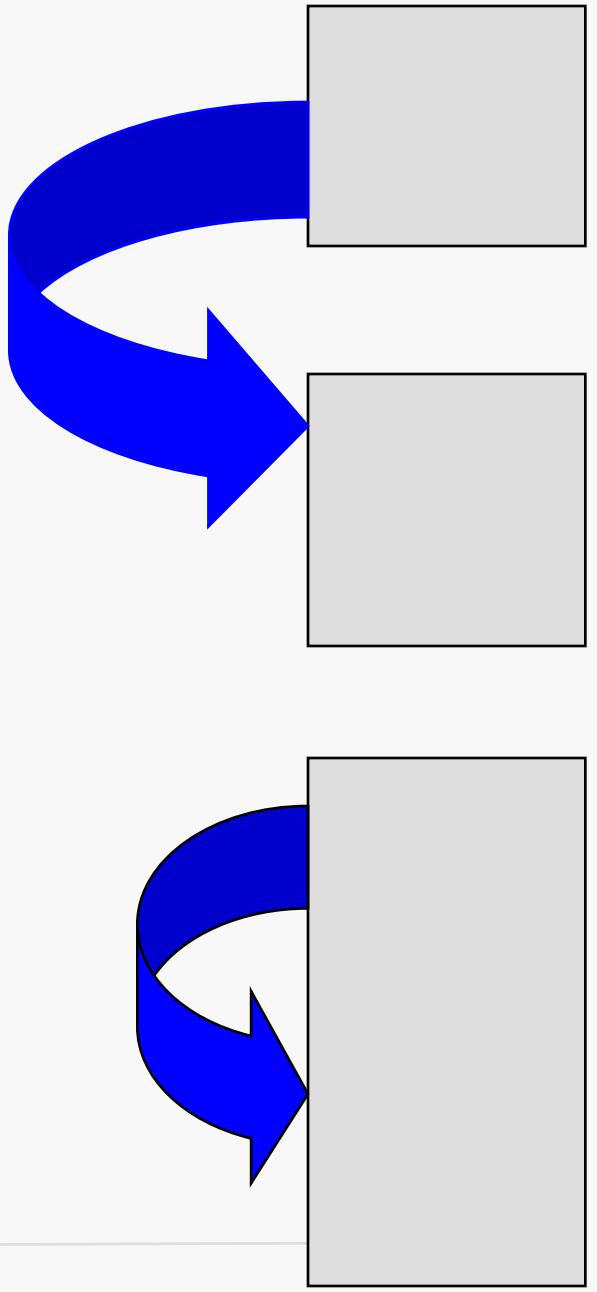
 logical division

EXAMPLE

HTML HYPERLINKS

- Inter-Page Link
  - *<A HREF = "URL">anchored text </A>*
- Intra-Page Link
  - *<A HREF = "#section">Source text </A>*
  - *<A NAME = "section">Target text </A>*

HREF means Hypertext REFerence



Example

HTML FRAME

- Normally, single page is displayed in a single window.
- Frame enables multiple pages to be displayed in a single window
- Frame partitions the window. Each partition displays a page

HTML FRAME

```
<html>
<head>
   <title>webtechnology</title>
</head>
<frameset rows="25%,*" border="0">
    <frame src="webtech_head.html" name="head">
    <frameset cols="20%,*" >
       <frame src="webtech_index.html" name="index“ target="main >
       <frame src=“WT_0.html" name="main“ target="main">
    </frameset>
</frameset>
</html>
```

Example

HTML LISTS

Un-ordered List

```
<UL>
    <LI>Item1</LI>
    <LI>Item1</LI>
</UL>
```

Ordered List

```
<OL>
    <LI>Item1</LI>
    <LI>Item1</LI>
</OL>
```

Definition List

```
<DL>
    <DT>Term</DT>
    <DD>Item1</DD>
    <DD>Item1</DD>
        <DT>Term</DT>
    <DD>Item1</DD>
    <DD>Item1</DD>
 </DL>
```

The <dl> tag defines a definition list.
<dt> (defines the item in the list) and <dd> (describes the item in the list):

Example

HTML TABLE

```
<TABLE>
    <TR>
        <TH> Header1</TH>
        <TH> Header2</TH>
    </TR>
     <TR>
        <TD> Data</TD>
        <TD> Data </TD>
     </TR>
   <CAPTION> Table Caption </CAPTION>
</TABLE>
```

ATTRIBUTES

ROWSPAN = number of row
COLSPAN = number of column
ALIGN = left | right | center
VALIGN = top|middle|bottom|baseline
BGCOLOR = color code
BACKGROUND = URL of image

Example

HTML HYPERMEDIA

- Image

<IMG SRC = "URL" WIDTH = " " HEIGHT = " " />

- Background Image

BACKGROUND = "URL"

Example

HTML FORM

```
<FORM ACTION = “”  METHOD = “GET | POST”>
      <INPUT TYPE = “text” NAME = “varName”    SIZE = “noc”  MAXLENGTH = “noc”>
      <INPUT TYPE = “checkbox” NAME = “varName” VALUE = “varValue”  CHECKED = “checked”>
      <INPUT TYPE = “radio” NAME = “varName”        VALUE = “varValue”  CHECKED = “checked”>
      <INPUT TYPE = “reset”  VALUE = “button label” >
     <INPUT TYPE = “submit”  VALUE = “button label” >
      <TEXTAREA NAME= “varName” ROWS=“nor” COL = “noc”>
      <SELECT NAME = “varName” SIZE = “nodoptions”>
            <OPTION> option1</OPTION>
            <OPTION> option2</OPTION>
      </SELECT>
</FORM>
```

Example

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>

- The <label> element defines a label for several form elements.

- The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

- The <fieldset> element is used to group related data in a form.

- The <legend> element defines a caption for the <fieldset> element.

- Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

- The <datalist> element specifies a list of pre-defined options for an <input> element.
- Users will see a drop-down list of the pre-defined options as they input data.
- The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.
- Example

```
<form action="/action_page.php">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

Two HTTP Request Methods: GET and POST

- Two commonly used methods for a request-response between a client and server are: GET and POST.
- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

The GET Method

- **Here query strings (name/value pairs) is sent in the URL of a GET request:**

  /test/demo_form.asp**?name1=value1&name2=value2**

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

The POST Method

- **Here query strings (name/value pairs) is sent in the HTTP message body of a POST request:**

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Comparison between GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
History	Parameters remain in browser history	Parameters are not saved in browser history

Comparison between GET vs. POST

	GET	POST
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

HTTP Request

- Request Line

GET /index.html HTTP/1.1

(Other methods: HEAD, PUT, POST)

- Header

User-Agent: Mozilla/4.05

Accept: text/html, text/xml, image/gif, image/jpg

Content-Type: text/html

Connection: keep-alive

Host: www.gauhati.ac.in

Cooke:name=value;name=value

<blank line>

- Body

Additional data

- A Request-line
- Zero or more header (General|Request|Entity) fields followed by CRLF
- An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields
- Optionally a message-body

HTTP Response

- Status Line

HTTP/1.1 200 OK
(100-199:informational;200-299:success;
300-399:redirect; 400-499:incomplete/ Client Error;
500-599:server error)

- Header

Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
<blank line>

- Body

<html><head>.....</head></body>

- A Status-line
- Zero or more header (General|Response|Entity) fields followed by CRLF
- An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields
- Optionally a message-body

Code and Description	
1xx: Informational	It means the request was received and the process is continuing.
2xx: Success	It means the action was successfully received, understood, and accepted.
3xx: Redirection	It means further action must be taken in order to complete the request.
4xx: Client Error	It means the request contains incorrect syntax or cannot be fulfilled.
5xx: Server Error	. It means the server failed to fulfill an apparently valid request.

URL Encoding

- **URL encoding also** known as "Percent Encoding, is a mechanism for encoding information in a Uniform Resource Identifier (URI) under certain circumstances, often used in the submission of HTML form data in HTTP requests.

Types of URI characters

- The characters allowed in a URI are either reserved or unreserved
  - Reserved characters are those characters that sometimes have special meaning. For example, forward slash characters are used to separate different parts of a URL.
  - Unreserved characters have no such meanings.
- Using percent-encoding, reserved characters are represented using special character sequences.
- The sets of reserved and unreserved characters and the circumstances under which certain reserved characters have special meaning have changed slightly with each revision of specifications that govern URIs and URI schemes.

Contd..

RFC 3986 section 2.2 *Reserved Characters* (January 2005)

!	*	'	(	)	;	:	@	&	=	+	$	,	/	?	#	[	]

RFC 3986 section 2.3 *Unreserved Characters* (January 2005)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

Source : http://en.wikipedia.org/wiki/Percent-encoding

Contd..

Percent-encoding reserved characters

- When a character from the reserved set (a "reserved character") has special meaning (a "reserved purpose") in a certain context, and a URI scheme says that it is necessary to use that character for some *other* purpose, then the character must be *percent-encoded*.
- Percent-encoding a reserved character involves converting the character to its corresponding byte value in ASCII and then representing that value as a pair of hexadecimal digits.
- The digits, preceded by a percent sign ("%") which is used as an escape character, are then used in the URI in place of the reserved character.
- (For a non-ASCII character, it is typically converted to its byte sequence in UTF-8, and then each byte value is represented as above.)
- The reserved character "/", for example, if used in the "path" component of a URI, has the special meaning of being a delimiter *between* path segments. If, according to a given URI scheme, "/" needs to be *in* a path segment, then the three characters "%2F" or "%2f" must be used in the segment instead of a raw "/".

Some Common Special Characters

Character	URL Encoded
;	%3B
?	%3F
/	%2F
:	%3A
#	%23
&	%26
=	%3D
+	%2B

Character	URL Encoded
$	%24
,	%2C
<space>	%20 or +
%	%25
<	%3C
>	%3E
~	%7E
%	%25

Example

Thank You