

Chapter 1

Introduction

Prepared by Hiren Mer

Concept of WWW

Definition:

It is a Information Space where documents and other web resources are identified by URL and interlinked by hypertext link and can be access via internet.

Invented by Tim berners Lee in 1989.

He wrote first web browser computer program when he was employed in CERN in Switzerland.

Concept of WWW

The Web

An infrastructure of information combined and the network software used to access it

Web page

A document that contains or references various kinds of data

Links A connection between one web page and another

Concept of WWW

Website

A collection of related web pages

Web browser

A software tool that retrieves and displays web pages

Web server

A computer set up to respond to requests for web pages

World Wide Web

Uniform Resource Locator (URL)

A standard way of specifying the location of a Web page, containing the hostname, "/", and a file

World Wide Web

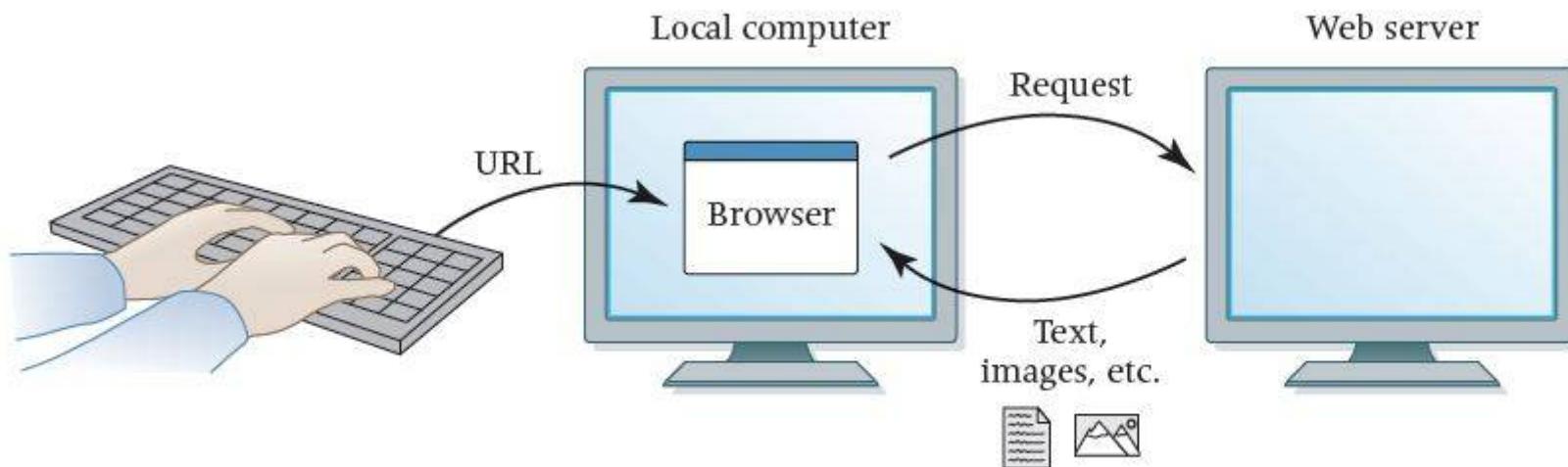
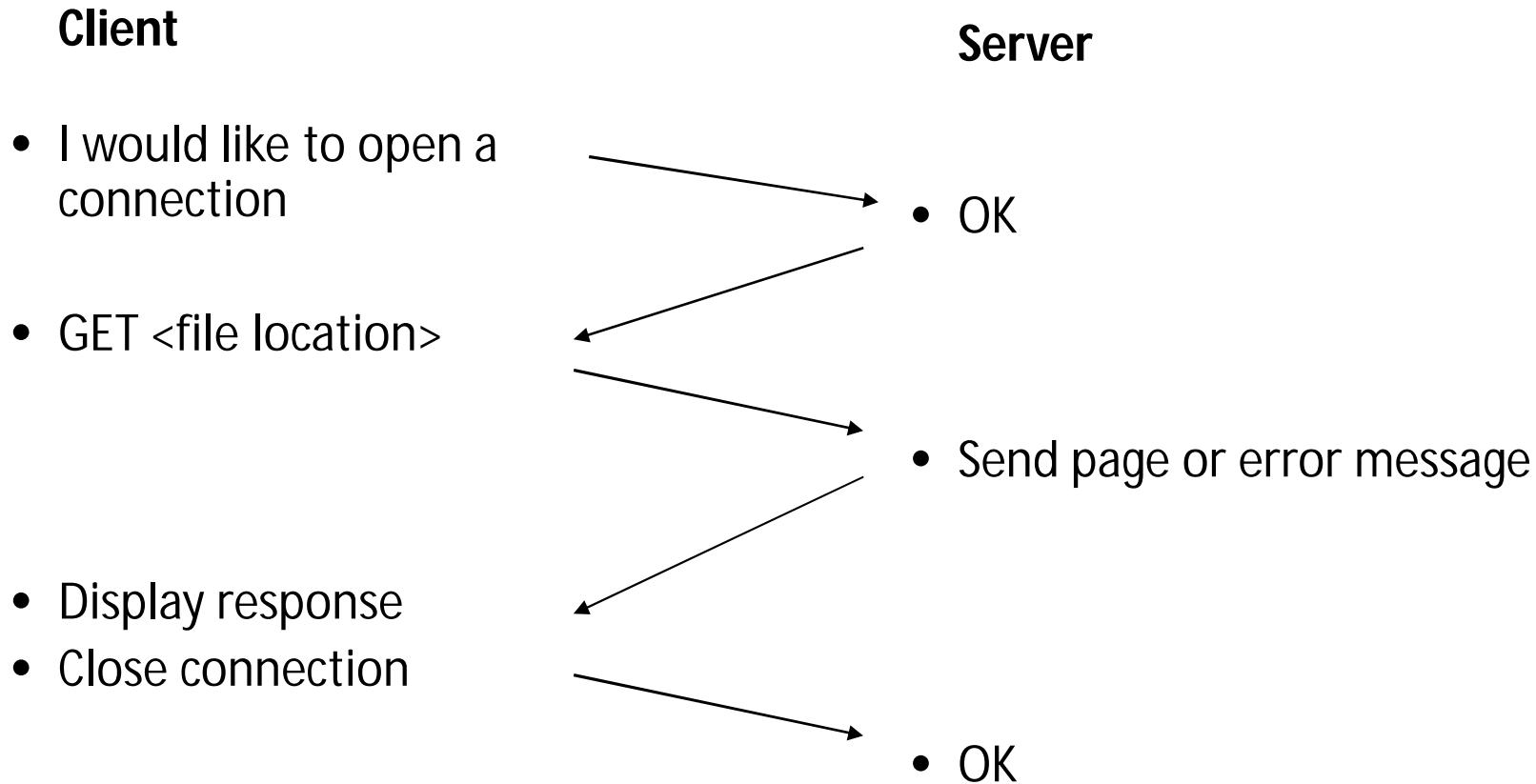


FIGURE 16.1 A browser retrieving a web page

*Why is the expression
"visiting a website"
confusing?*

An HTTP conversation



HTTP is the set of rules governing the format and content of the conversation between a Web client and server

An HTTP example

The message requesting a Web page must begin with the work "GET" and be followed by a space and the location of a file on the server, like this:

```
GET /fac/lpress/shortbio.htm
```

The protocol spells out the exact message format, so any Web client can retrieve pages from any Web server.

What is HTTPS?

- Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to. The 'S' at the end of HTTPS stands for 'Secure'. It means all communications between your browser and the website are encrypted. HTTPS is often used to protect highly confidential online transactions like online banking and online shopping order forms.
- Web browsers such as Internet Explorer, Firefox and Chrome also display a padlock icon in the address bar to visually indicate that a HTTPS connection is in effect.

HTTP vs HTTPS

User



Insecure connection

Normal HTTP (80)



User



Encrypted Connection

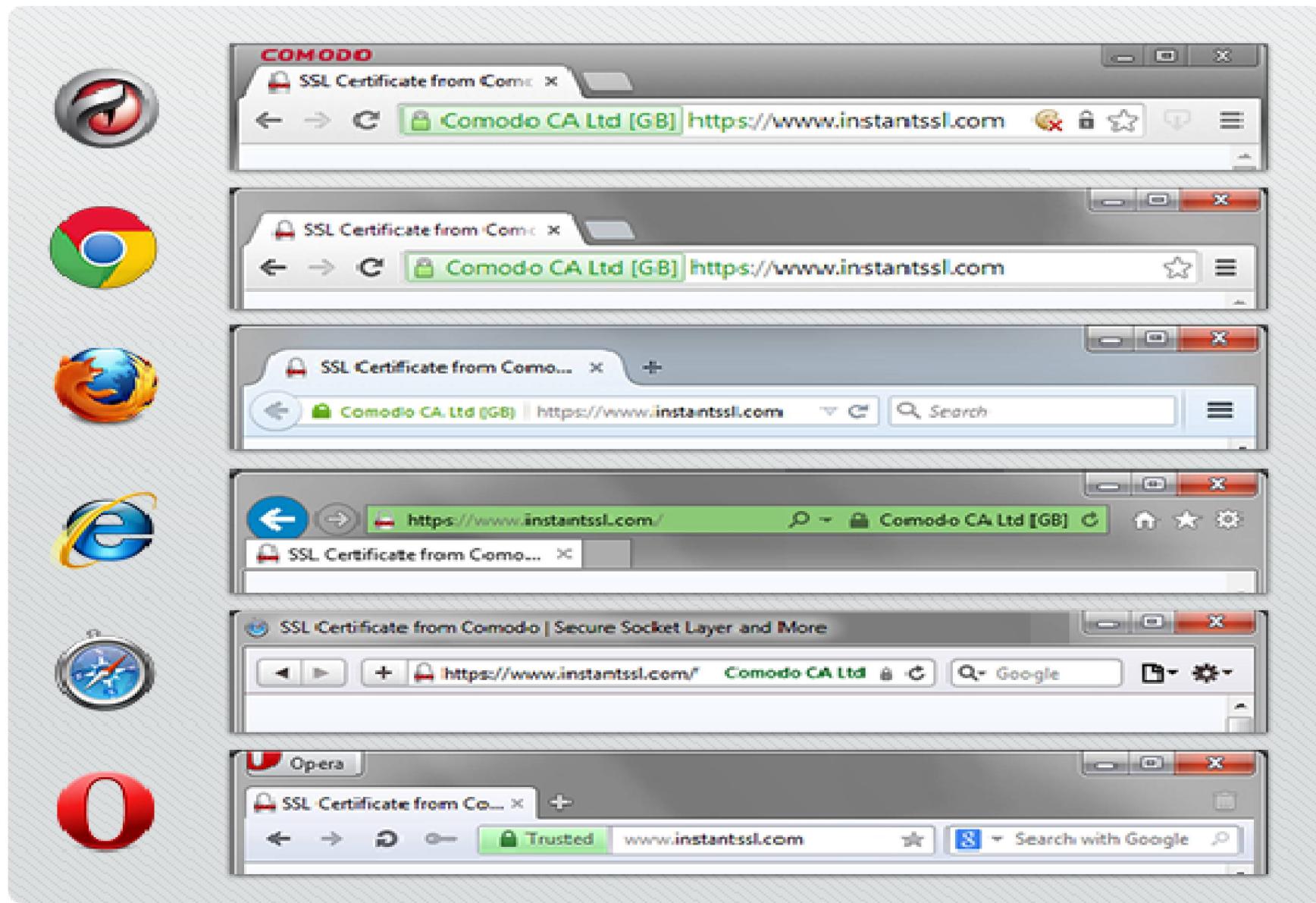
Secure HTTPS (443)



SSL Certificate

Why Is an SSL Certificate Required?

- All communications sent over regular HTTP connections are in 'plain text' and can be read by any hacker that manages to break into the connection between your browser and the website. This presents a clear danger if the 'communication' is on an order form and includes your credit card details or social security number.
- With a HTTPS connection, all communications are **securely encrypted**. This means that even if somebody managed to break into the connection, they **would not be able decrypt any of the data which passes between you and the website**.



Prepared by Hiren Mer

Benefits of Hypertext Transfer Protocol Secure

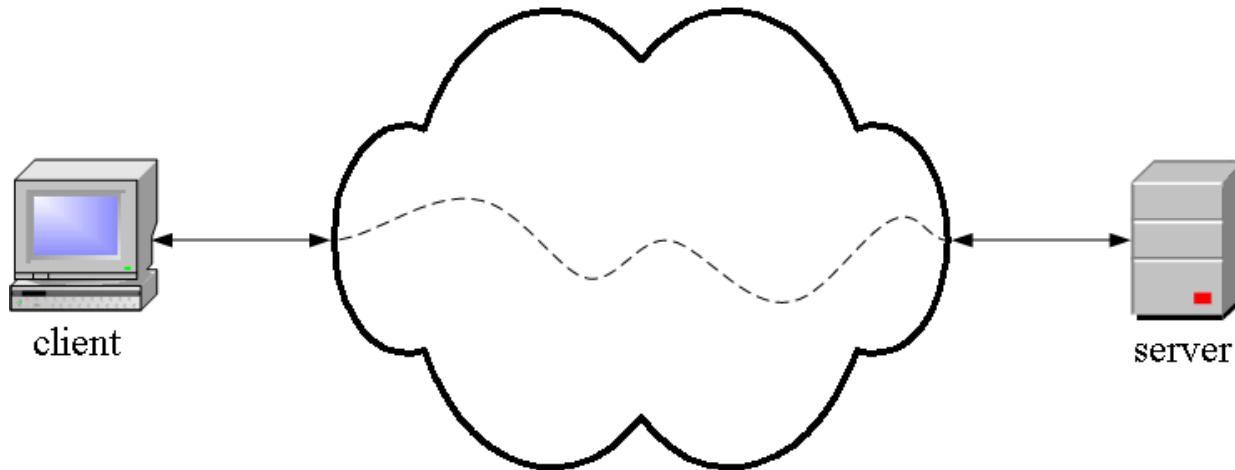
- Customer information, like credit card numbers, is encrypted and cannot be intercepted
- Visitors can verify you are a registered business and that you own the domain
- Customers are more likely to trust and complete purchases from sites that use HTTPS

Year	Web browsers	Internet users (in millions)
1991	WorldWideWeb (Nexus)	4
1992	ViolaWWW , Erwise , MidasWWW , MacWWW (Samba)	7
1993	Mosaic , Cello , ^[28] Lynx 2.0 , Arena , AMosaic 1.0	10–14
1994	IBM WebExplorer , Netscape Navigator , SlipKnot 1.0 , MacWeb , IBrowse , Agora (Argo), Minuet	20–25
1995	Internet Explorer 1 , Netscape Navigator 2.0 , OmniWeb , UdiWWW , ^[29] Internet Explorer 2 , Grail	16–44
1996	Arachne 1.0 , Internet Explorer 3.0 , Netscape Navigator 3.0 , Opera 2.0 , PowerBrowser 1.5 , ^[30] Cyberdog , Amaya 0.9 , ^[31] AWeb , Voyager	36–77
1997	Internet Explorer 4.0 , Netscape Navigator 4.0 , Netscape Communicator 4.0 , Opera 3.0 , ^[32] Amaya 1.0 ^[31]	70–120
1998	iCab , Mozilla	147–188
1999	Amaya 2.0 , ^[31] Mozilla M3 , Internet Explorer 5.0	248–280
2000	Konqueror , Netscape 6 , Opera 4 , ^[33] Opera 5 , ^[34] K-Meleon 0.2 , Amaya 3.0 , ^[31] Amaya 4.0 ^[31]	361–413

Year	Web browsers	Internet users (in millions)
2001	Internet Explorer 6 , Galeon 1.0, Opera 6 , ^[35] Amaya 5.0 ^[31]	499–513
2002	Netscape 7 , Mozilla 1.0, Phoenix 0.1, Links 2.0 , Amaya 6.0, ^[31] Amaya 7.0 ^[31]	587–662
2003	Opera 7 , ^[36] Safari 1.0, Epiphany 1.0, Amaya 8.0 ^[31]	719–778
2004	Firefox 1.0, Netscape Browser , OmniWeb 5.0	817–910
2005	Safari 2.0, Netscape Browser 8.0, Opera 8 , ^[37] Epiphany 1.8, Amaya 9.0, ^[31] AOL Explorer 1.0, Maxthon 1.0, Shiira 1.0	1018–1029
2006	SeaMonkey 1.0, K-Meleon 1.0, Galeon 2.0, Camino 1.0, Firefox 2.0, Avant 11, iCab 3, Opera 9, ^[38] Internet Explorer 7	1093–1157
2007	Maxthon 2.0, Netscape Navigator 9 , NetSurf 1.0, Flock 1.0, Safari 3.0, Conkeror	1319–1373
2008	Konqueror 4 , Safari 3.1, Opera 9.5, ^[39] Firefox 3 , Amaya 10.0, ^[31] Flock 2, Chrome 1, Amaya 11.0 ^[31]	1562–1574
2009	Internet Explorer 8 , Chrome 2–3, Safari 4, Opera 10 , ^[40] SeaMonkey 2, Camino 2, Firefox 3.5 , surf	1743–1802
2010	K-Meleon 1.5.4, Firefox 3.6 , Chrome 4–8, Opera 10.50 , ^[41] Safari 5, xxxterm , Opera 11	1971–2034

Year	Web browsers	Internet users (in millions)
2011	Chrome 9–16, Firefox 4 –9, Internet Explorer 9 , Maxthon 3.0, SeaMonkey 2.1–2.6, Opera 11 .50, Safari 5.1	2264–2272
2012	Chrome 17–23, Firefox 10–17, Internet Explorer 10 , Maxthon 4.0, SeaMonkey 2.7–2.14, Opera 12 , Safari 6	2497–2511
2013	Chrome 24–31, Firefox 18–26, Internet Explorer 11 , SeaMonkey 2.15–2.23, Opera 15–18, Safari 7	2712
2014	Chrome 32–39, Firefox 27–34, SeaMonkey 2.24–2.30, Opera 19–26, Safari 8	3079

HTTP is an **application layer** protocol



- The Web client and the Web server are application programs
- Application layer programs do useful work like retrieving Web pages, sending and receiving email or transferring files
- Lower layers take care of the communication details
- The client and server send messages and data without knowing anything about the communication network

Web 2.0

The Read/Write Web

History

- Tim Berners-Lee: World Wide Web 1989
- Dream of sharing information back and forth
- Mosaic Web browser in 1993
- Writing to the web required knowledge of HTML codes and more.



The New World Wide Web

- New way software developers and users utilize the web.
- 2003 - sites utilize tools that make it easy to publish to the web.
- No longer limited to “consuming” information.
- Multimedia publishing has exploded.

Useful Web 2.0 Tools

- Weblogs
- Wikis
- Real Simple Syndication (RSS)
- Aggregators
- Social Bookmarking and Networking
- Online Photo Galleries
- Audio/video-casting

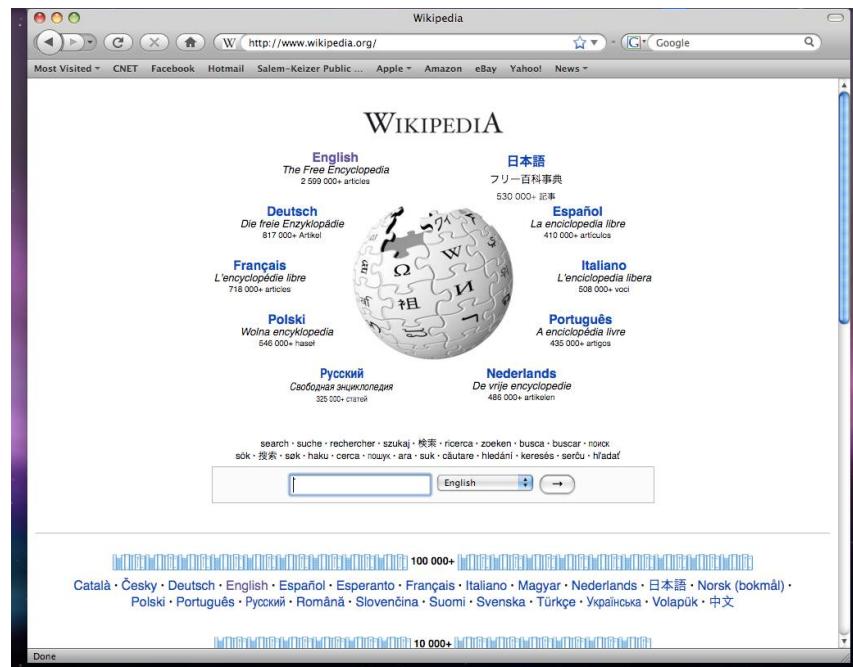
*Not the only web 2.0 tools! Keep educational uses in mind when looking at these.

Weblogs (Blogs)

- Easily created/updated sites.
- Publish instantly from any internet connection.
- Interactive - comment, question, link, converse.

Wikis

- Collaborative page.
- Anyone can add or edit content.
- Wikipedia - friend or foe?
- Wikispaces



Real Simple Syndication

- Subscribe to “feeds” from your favorite sites.
- New information is sent to you.
- Reduces research time.
- Example:
- Google Reader

Aggregators

- Collects and organizes the content you subscribe to with an RSS feed.

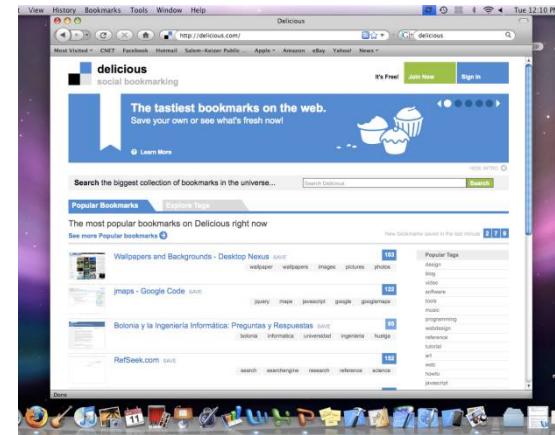


Aggregator refers to a web site or computer software that aggregates a specific type of information from multiple online sources:

- Poll aggregator, a website that aggregates polling data for upcoming elections
- Review aggregator, a website that aggregates reviews of movies or other products or services
- Search aggregator, software that runs on a user's computer and fetches, filters, and organizes a specific search from various search engines
- Social network aggregation, the collection of content from multiple social network services
- Video aggregator, a website that collects and organizes online video sources
- News aggregator, a computer software or website that aggregates news from other news sources
- Shopping aggregator, a website that aggregates retail products from numerous retailers' websites
- Insurance aggregator, a website that enables consumers to receive multiple insurance quotes from policy providers, and subsequently purchase a suitable quote

Social Bookmarking

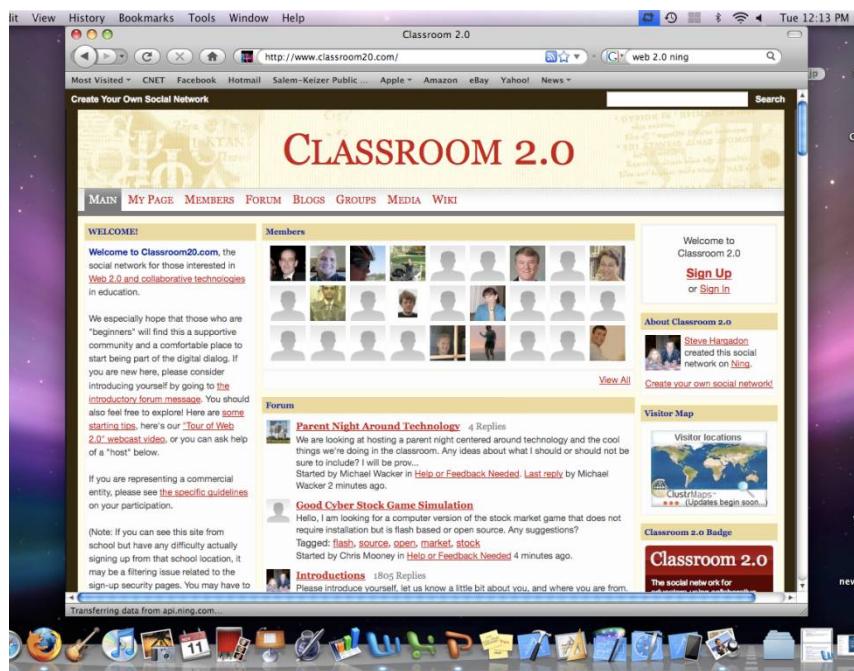
- Save web addresses of useful content.
 - Share and search bookmarks.
 - Generate specific resource lists.



Prepared by Hiren Mer

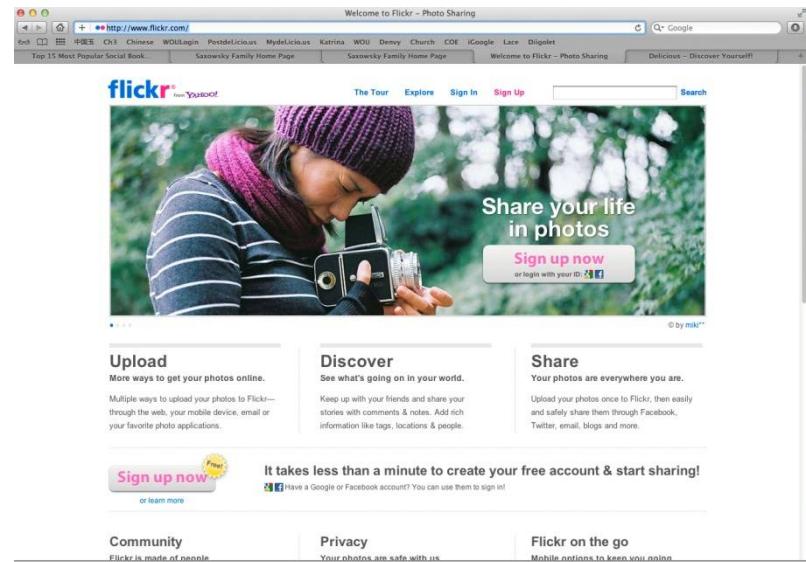
Social Networking

- Facebook, Twitter
- LinkedIn.com
- Collaborate, bookmark, share, etc.



Online Photo Galleries

- Publish photos online.
- Comment, and share ideas.
- Create photo stories and presentations.

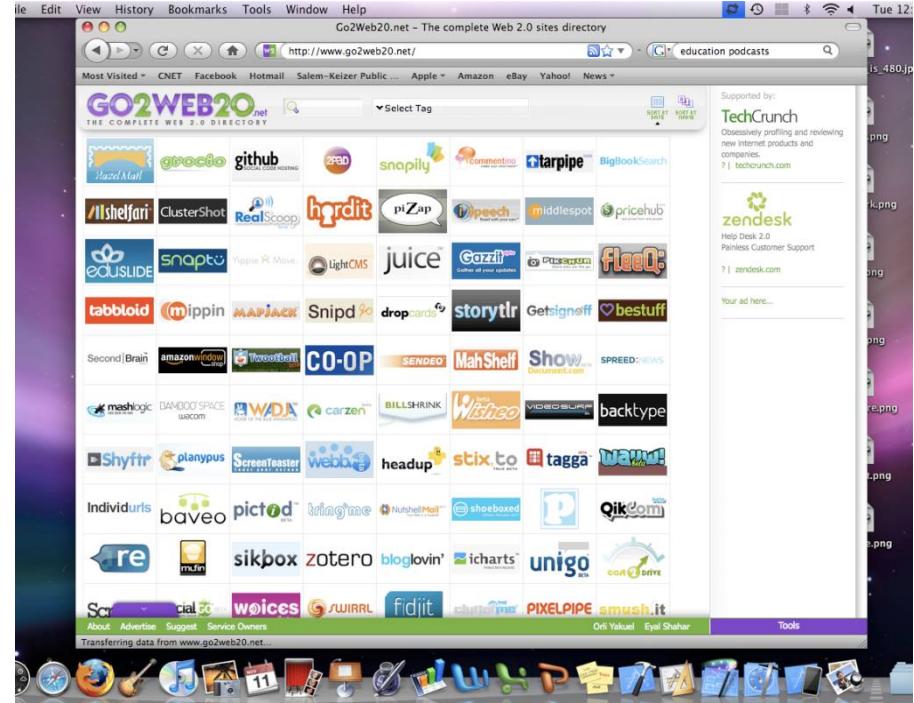


Audio/video-casting

- Produce audio and video recordings.
- Publish them easily to the web.
- Creates a world-wide audience.



Other Fun Web 2.0 Sites



Prepared by Hiren Mer

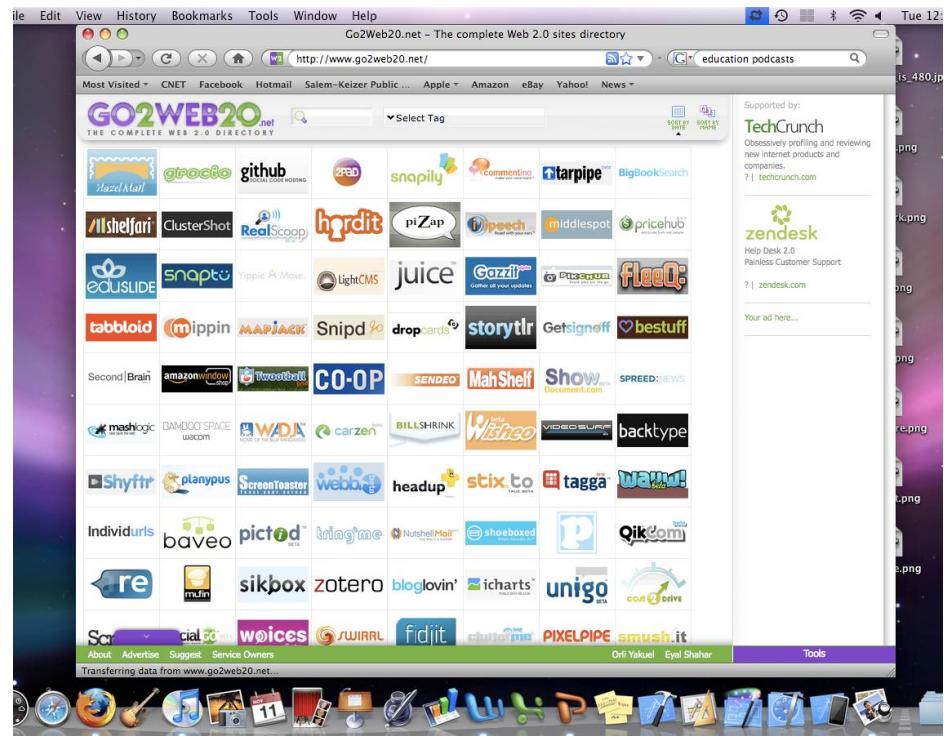
Time to Explore

- Visit
www.bighugelabs.com
- In groups of three choose one of the tools and prepare a short explanation of what it can do.



Are you ready for this?

- Visit
www.go2web20.net
- In groups of three
choose a web 2.0 tool
to share with the class.



Time to bring it all together

- How could these tools be used in a particular grade level and content area?
- What do these tools bring to a lesson that is unique?
- What are some disadvantages or risks associated with these tools?
- What are some of the benefits and challenges of utilizing web 2.0 tools?

Chapter 2

Web Design

Prepared by Hiren Mer

Contents

- Concepts of effective web design
- Web design issues
 - Browser and Operating system
 - Bandwidth and Cache
 - Display resolution
 - Look and Feel of the Website
 - Page Layout and linking
 - User centric design
 - Sitemap,
- Planning and publishing website
- Designing effective navigation

Concepts of effective web design

- WWW is huge amount of web pages group together on websites and upload on web servers.
- Websites are designed for personal information, business organization, education institutes, government departments, online marketing and selling, banking, entertainment, online booking and more.

Concepts of effective web design

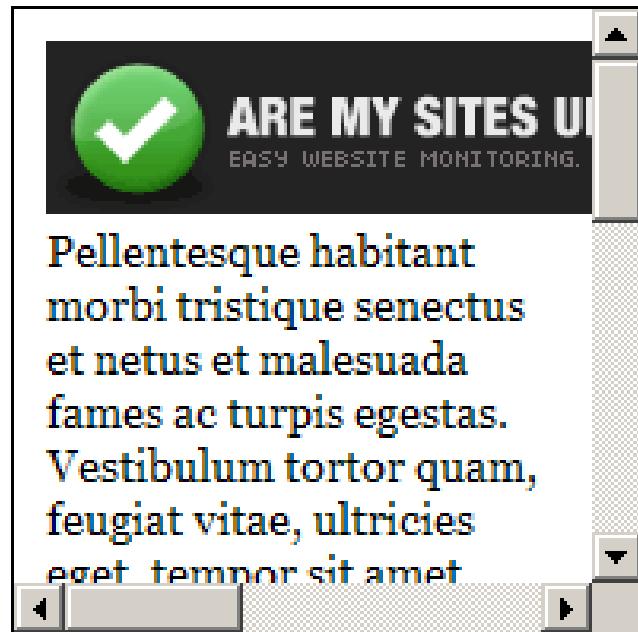
- Website are depend on environment like **browser and operating system**.
- Planning of websites includes Objective and goal, audience characteristics,contents,prototyping, and taking necessary steps to implementing and publish web site.
- All these factors are very important in designing a successful website.

Website Design Issues

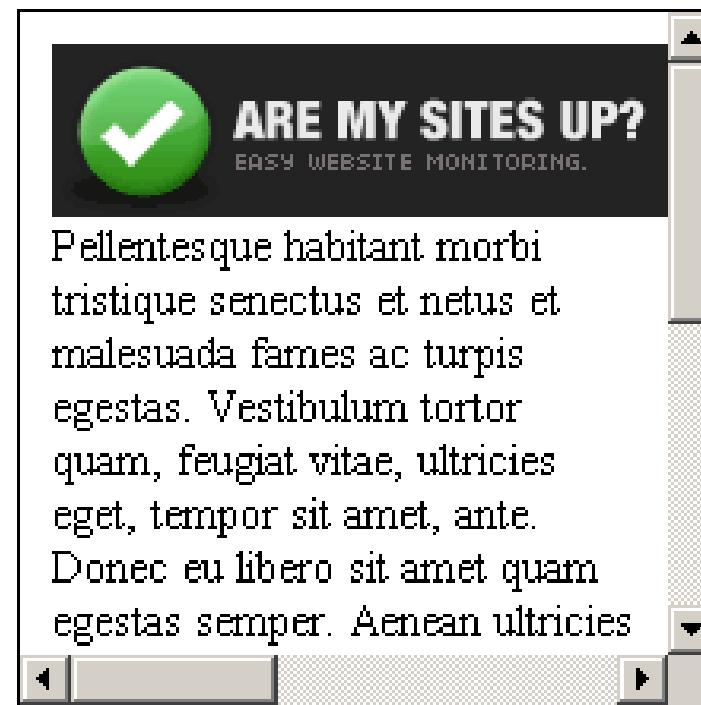
Browser and Operating system

- Web pages are implemented in HTML tags and viewed in browser window. The different browser and their versions affect the way a page is rendered.
- Different browser interpret same HTML pages in different way.

Browser Difference



IE 7



Firefox 3.0.10

Browser and Operating system

- Solution:
 - To make web page portable, test it on different browser and os.
 - Use the development tools to add special features to your pages supported by majority browsers.
 - The best way to avoid the problems is to follow standards.
 - Use validation rules to of the W3c to validate your code.

Browser and Operating system

- W3C improves the browser compatibility of your code as latest versions of popular browsers such as IE, Firefox, Netscape etc have started supporting standards.

[Check your site](#)

<https://validator.w3.org/>

The screenshot shows the W3C Markup Validation Service interface. At the top, there are three tabs: "W3C The W3C Markup Validat" (active), "Gujarat Technological U", and another partially visible tab. The address bar shows the URL <https://validator.w3.org>. The main header features the W3C logo and the text "Markup Validation Service" with the subtitle "Check the markup (HTML, XHTML, ...) of Web documents". Below the header are three buttons: "Validate by URI" (selected), "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" section contains a form with a label "Validate a document online:" and a text input field containing "http://gtu.ac.in/". There is also a link "More Options" and a large "Check" button. A sidebar on the right side of the page has a title "About this service" and a sub-section "The W3C validators rely on community support for hosting and development." It includes a "Donate" link, a "Flattr" button, and a "5947" badge.

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-based validator](#).

The screenshot shows the validation results for the URL "http://gtu.ac.in/". The results are displayed in a table with columns "Error", "Warning", and "Info". The table shows several errors related to the "gtu.ac.in" domain. The footer of the page includes a "Feedback" link, a "Contribute" link, and a note "This service runs the W3C Markup Validator v1.3+ha". The bottom of the screen shows a taskbar with icons for file operations (Home, PgUp, End, Insert, PrtScn) and system status (PgDn, Mv Up, Pause, Mv Dn, ScrLk, Dock). The system tray shows battery level, signal strength, and the date/time "22-12-2016 21:50".

[Invalid] Markup Validation Service Gujarat Technological University

https://validator.w3.org/check?uri=http%3A%2F%2Fgtu.ac.in%2F&charset=%28detect+automatically%29&doctype=Inline&group=0

W3C® Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential Issues Validation Output

Errors found while checking this document as XHTML 1.0 Transitional!

Result:	103 Errors, 18 warning(s)
Address :	<input type="text" value="http://gtu.ac.in/"/>
Encoding :	utf-8 <input type="button" value="(detect automatically)"/>
Doctype :	XHTML 1.0 Transitional <input type="button" value="(detect automatically)"/>
Root Element:	html
Root Namespace:	http://www.w3.org/1999/xhtml

The W3C validators are developed with assistance from the Mozilla Foundation, and supported by community donations. [Donate](#) and help us build better tools for a better web.

5947 [Flattr](#)

Home PgUp Nav
End PgDn Mv Up
Insert Pause Mv Dn
PrtScn ScrLk Dock

Show Outline List Messages Sequentially Group Error Messages by Type

Show all X 21:51 ENG 22-12-2016

Bandwidth and Cache

- User has different connection speed to access websites.
- Connection speed plays important role to design web pages.
- If user has lowed **bandwidth** connection and website contains so many pages then used denied to visit that site.

Bandwidth and Cache

- When we design the web site then we have to consider the lowest speed **56-256kbps** of the internet connection.
- For high speed connection we can use the good graphics for the particular websites.
- Browser provide temporary memory called **cache**.

Bandwidth and Cache

- When user open the any webpage for first time it stores the graphics in browser cache.
- Then next time the same page user is visited, the browser gets only HTML file and use the images from the cache which increase the download speed significantly.

Display resolution

- **Display resolution** is another important factor affecting the web page design, as we do not have control of display resolution on Monitors which user view pages.
- Display resolution is measured in pixels
- Common display resolutions
 - 800 X 600
 - 1024 X 768
 - **1240 X 1024 more common**

Display resolution

- If page displayed in higher resolution, the page displayed on left hand side and some part on right hand site become blank.
- We can use centered design to display page e.g middle screen, leaving same space on both sides.

HTML Tutorial Gujarat Technological U... X

old.gtu.ac.in

GUJARAT TECHNOLOGICAL UNIVERSITY AHMEDABAD (INTERNATIONAL INNOVATIVE UNIVERSITY)

Academic Calendar Events Result Web Mail Circulars Tender Recruitment International Ahmedabad Gandhinagar Chandkheda New Format of GTU web-site

This website was Last

Home For Disabled Search web contents GTU Act / UGC Cell RTI Student Grievance Portal Achievements Affiliation Affiliated Colleges & Courses Syllabus Ph.D. Programme

Recorded Video of Fifth convocation on 28-01-2016

Result Links : <- Select Exam -->

Alternative Result link- Link-1 | Link-2 Download GTUResult Android mobile app.

Institute & Course Details | Faculty Details of GTU

Degree, Diploma Certificate Verification

A message from your Vice-Chancellor

My dear students and colleagues:

GTU has grown and matured into a dynamic and vibrant university, since its inception. Excellence is a popular and much loved word and pushes the frontiers of all activities. Hence we should try and define

NEWS

- ALVCOM- Active Learning Video Lecture Communication-Schedule for Dt 6th-7th Aug 2016
- Webinar on Digital Image Processing Concepts and Techniques with MATLAB by Parul Bansal, Thu, Aug 4, 2016 3:30 PM - 4:30 PM IST.
- Webinar on Course Outcome

Look and Feel of the Website

- Web site theme
- Fonts Graphics and colours
- Presentation and Access



This website was last Modified on 20 July 2016, For latest information please

Home
Result Links
Student Information
GTU Act / UGC Act
GTU
Student Employment Portal
Academic Links
Admissions
Affiliated Colleges & Courses
Syllabus
Ph.D. Programs
Student Center & Student Council History
Under Minis (UPM)
GTU Assessment
Chancellor Message
GTU Alumni Association
GTU News
GTU
Academic
Institute Details
Directions
Publications
Conferences
Postponement
PG Classes Portal
PG Results Centers
GTU Examination Forum
Mycampus
Ph.D. (Information)
GTU Students & Placement Cell
Board of Examination & Exam Timetables
GTU Board for Mobile Computing & Wireless Technologies
GTU Innovation Council
Institutional Profile
Media Manager
List of MCA
GTU Events
GTU Library
Mathematics Majors

Received Miles of Fifth convocation on 29-01-2016

Result Links : -- Select Exam --

Alternative Result Link- Link-1 | Link-2 | Download GTUResult Android mobile app.

Institute & Courses Details | Faculty Details of GTU

Degree, Diploma Certificate Verification

A message from your Vice-Chancellor

My dear students and colleagues:

GTU has grown and matured into a dynamic and vibrant university since its inception. Excellence is a popular and much loved word and pushes the frontiers of all activities. Hence we should try and define what we are going to excel in and how. My request to all our students, faculty and managements is to define their domains and strengths, find where they are and then plan their onward journey. Good governance and educational outcomes go hand in hand, hence it is important that clearly defined goals, outputs and outcomes are put in place. GTU too shall take the same roadmap and define specific goals to start with, the outputs in terms of number of faculty trained, student programmes conducted, will help in understanding whether we are progressing. But we have to look at the bigger picture and also keep in view the outcomes, the impact that these initiatives and goals can lead to. Unless we work to this end, we will claim accolades in terms of numbers, but the biggest stake holder, our student, the future citizen of this great Nation, will not be able to reap the benefits. Excellence will only be achieved when a student, a faculty and an institute is able to impact the environment around them.



In the coming days, we hope to work on these lines and put in place enabling processes, to tap the opportunities available through the State and Central policies and programs.

Your Ideas and Initiatives will be most welcome and the GTU team and myself will be happy to hear from you.

Our respected Dr. Agarwal sir, has sent his warm wishes and thanks for all the good work done during his tenure, during his speech on 7th June 2016.

Dr. Rajul K.Gajjar

(7th June 2016)

Other Messages from your Vice-Chancellor

Mr. Chintan Patel
06 99 02 819

(Counter) Start Date : 04 April 2016

NEWS

Received For Industry-Academia Collaboration by Prof. Vaibhav M. G. Tse, Jun. 14, 2016 03:00 PM - 4:00 PM IST
GTU Open Source Technologies Club organized "Open Source Summer Camp" from 10th to 15th June, 2016 at GTU, Block No-B. Please register online on or before 10th June 2016.

Received For Leadership in International Information Technology program - online - local collaboration for Japan and India by Dr. Pankaj Rangwade Ph.D., Director, Japan, Jun. 14, 2016 03:00 PM - 4:00 PM IST

Received For All Academy Details about Online-Gated Entrance Program by Prof. Suresh Department of Information Technology
Programs and Institutes of Technology Staff, Jun. 14, 2016 03:00 PM - 4:00 PM IST
Received For Industry Training Program by Dr. Chandrashekhar H. Shingare, Head, Engineering Education and Research Committee

New Horizons

Received For All Academy Details about Online-Gated Entrance Program by Prof. Suresh Department of Information Technology Staff, Jun. 14, 2016 03:00 PM - 4:00 PM IST
Received For Project Plan of all Three Departments of VJTI, SRM, College, Phagwara, 2016 at Bhupendra Prasad Mukherjee Art College, Raunaq Campus, Phagwara

Received For Project Plan of B.E. Department of Electrical & Electronics Engineering Institute of Technology, Raunaq Campus, Phagwara, 2016 at Mr. Vinay Singh, City General Manager, Gurdaspur New year invited to evaluate the project

Received For Project Plan of E.C. Department of Government College of Ghaziabad, Raunaq Campus, 2016 at Mr. Vinay Singh, City General Manager, Gurdaspur

Received For Project Plan of M.E. Department of Mathematics Institute of Ghaziabad, 2016 at Mr. Vinay Singh, City General Manager, Gurdaspur

Page Layout and Linking

- Website consists of individual web pages that are linked together using various navigational links.
- Page Layout defines the visual structure of page and area into different part to information.
- Page layout allows designer to distribute the contents in a page such that visitor can easily find and access necessary details.

Page Layout and Linking

- Page Layout called page Template. There are so many page templates are available on websites which can directly simply used to design page by inserting contents of pages.
- **Key points for Designing Layouts:**
 1. While distributing the contents in layout, balance between text and graphics. Too much graphics lead to slow download of the page.

Page Layout and Linking

2. Locate the item on page using screen importance to attract user's attention on important items.
3. Maintain Consistency among the layout of the pages such that it creates cohesive design of web site matching to your theme.

Locating Information

- Web pages viewed on computer screen and it can be divided into five major areas **center,top,right,bottom and left.**
- The main importance is user viewing pattern center,then top,right,bottom left in particular order.
- Put important information in center to get immediate attention of the user.

Locating Information

- Information on other parts as per the order of importance.
- Most website use **left side** for the links of other pages.
- Top is used to display **logo and title or flash as advertisements.**
- **Right hand side for other information like news or other updates ,circulars.**

Locating Information

- Bottom part used to **quick link** to important sections or to **provide copyright message**.

Prepared by Hiren Mer

Make Design User Centric

- It is very difficult for web designer to predict exact behaviour of the user.
- It is depend on the reading habits.
- Some of the user want to connect the subsection and want different results.
- Then designer can give the hyperlink for the particular keyword.

Sitemap

- Many time websites are too complex as there area a large number of sections and each sections contains many pages.
- It is difficult for user to move from one page to other directly.
- User getting confused if there are so many pages in the websites.

Sitemap

- Solution is to provide **Sitemap** for all pages.
- By clicking any link in sitemap ,you can directly jump to particular page.
- Sitemap gives the organization of contents for the particular website.

Sitemap website example X S Sitemap | Samsung India X

www.samsung.com/in/info/sitemap/ Find out more here

MOBILE TV & AV APPLIANCES MORE SHOP



SUPPORT BUSINESS



MOBILE

MOBILE

Smartphones
Tablets
Wearables
Other Phones
Accessories

MOBILE APPS

Find my mobile
Galaxy Apps
Theme Service
Smart Switch
SideSync
My galaxy

FEATURED

Galaxy S7
Galaxy On Nxt
Galaxy J7 Prime
Galaxy J2 (2016)
Samsung Z2
Galaxy A9 Pro

[+/-] Feedback

TV & AV

Home	PgUp	Nav
End	PgDn	Mv Up
Insert	Pause	Mv Dn
PrtScn	ScrLk	Dock
Options	Help	Fade

AUDIO

Home Entertainment System
Soundbar
Sound Tower
Digital Component

FEATURED

SUHD TV

Prepared by Hiren Mer



19:28 ENG 27-12-2016

Planning a Website

Before we start designing/developing we should ask this questions

1. Why are we developing website?
2. What do we achieve by developing this website?
3. Who are the people who will use this website?
4. What are the information contents?
5. How are these contents organized? What are the possible ways?
6. How the files prepared are organized?

Planning a Website

- We have
 - Objective and Goal
 - Audience/user profile
 - Identifying and Organizing Contents
 - Towards Publishing Website

Navigation

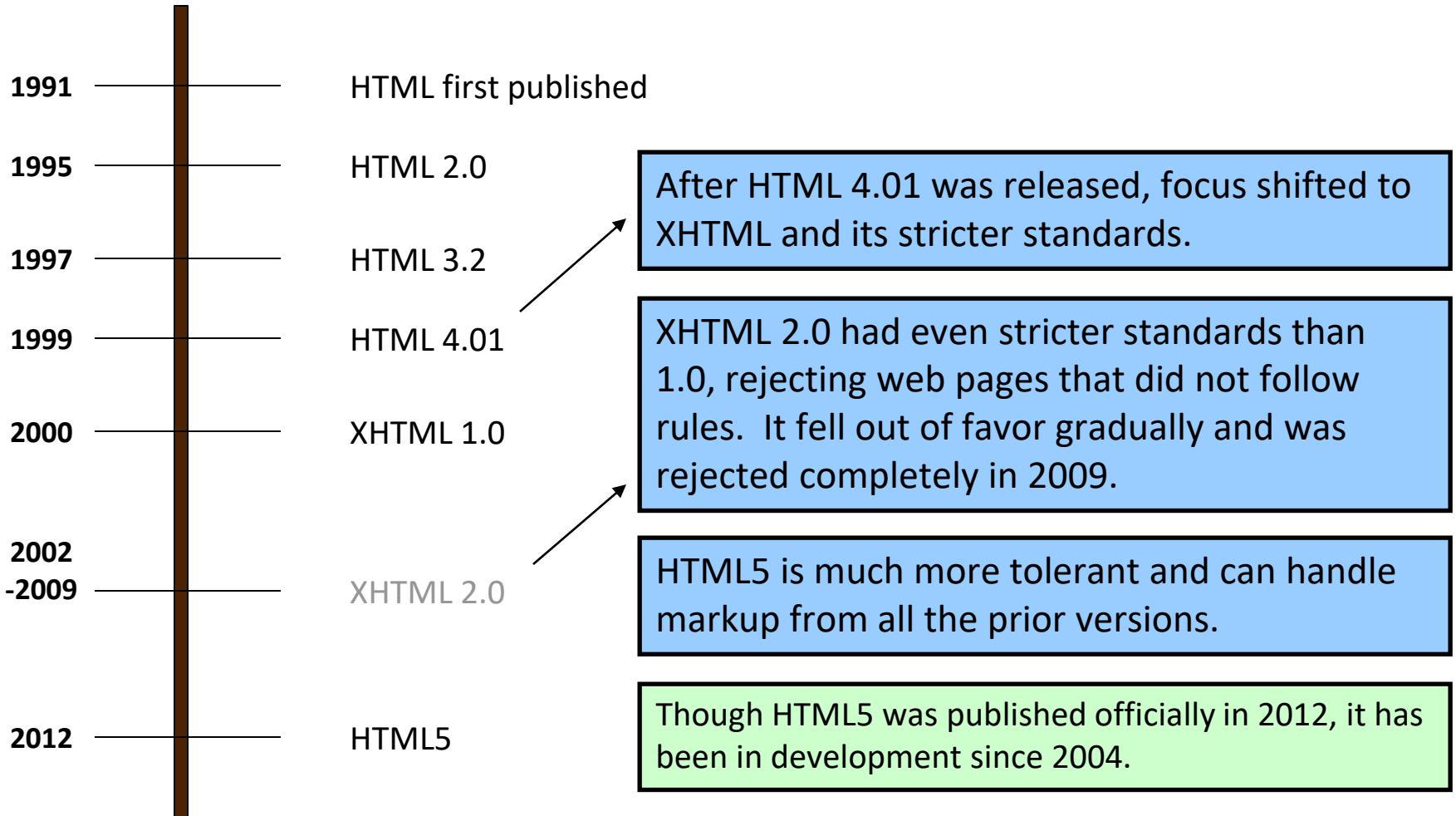
- Navigation means the ways to move from one page to another page in website using hyperlinks provided on a page.
- If it is not proper then user feel the problems in moving around the pages in your effective navigation website.

- Thank you

Chapter 3

HTML

History of HTML



XHTML - Introduction

- XHTML stands for **EXtensible HyperText Markup Language**. It is the next step in the evolution of the internet. The XHTML 1.0 is the first document type in the XHTML family.
- XHTML is almost identical to HTML 4.01 with only few differences. This is a cleaner and stricter version of HTML 4.01. If you already know HTML, then you need to give little attention to learn this latest version of HTML.

XHTML - Introduction

- XHTML was developed by World Wide Web Consortium (W3C) to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its benefits, while still remaining confident in the backward and future compatibility of the content.

Why Use XHTML?

- XHTML documents are XML conforming as they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.
- XHTML gives you a more consistent, well-structured format so that your webpages can be easily parsed and processed by present and future web browsers.

Why Use XHTML?

- You can easily maintain, edit, convert and format your document in the long run.
- Since XHTML is an official standard of the W3C, your website becomes more compatible with many browsers and it is rendered more accurately.
- XHTML combines strength of HTML and XML. Also, XHTML pages can be rendered by all XML enabled browsers.

Why Use XHTML?

- XHTML defines quality standard for your webpages and if you follow that, then your web pages are counted as quality web pages. The W3C certifies those pages with their quality stamp.

Here are the important points to remember while writing a new XHTML document or converting existing HTML document into XHTML document –

- Write a DOCTYPE declaration at the start of the XHTML document.
- Write all XHTML tags and attributes in lower case only.
- Close all XHTML tags properly.
- Nest all the tags properly.`<p></p>`
- Quote all the attribute values.`[align="center"]`
- Forbid Attribute minimization.`[selected="selected"]`
- Replace the **name** attribute with the **id** attribute.
- Deprecate the **language** attribute of the script tag.

Conclusion

- XHTML is supposed to make the HTML-Web-Pages suitable for all browser types.
 - Disadvantages:
 - Web developers have to obey the rules
 - Errors will not be tolerated
- => More expenditure of time**

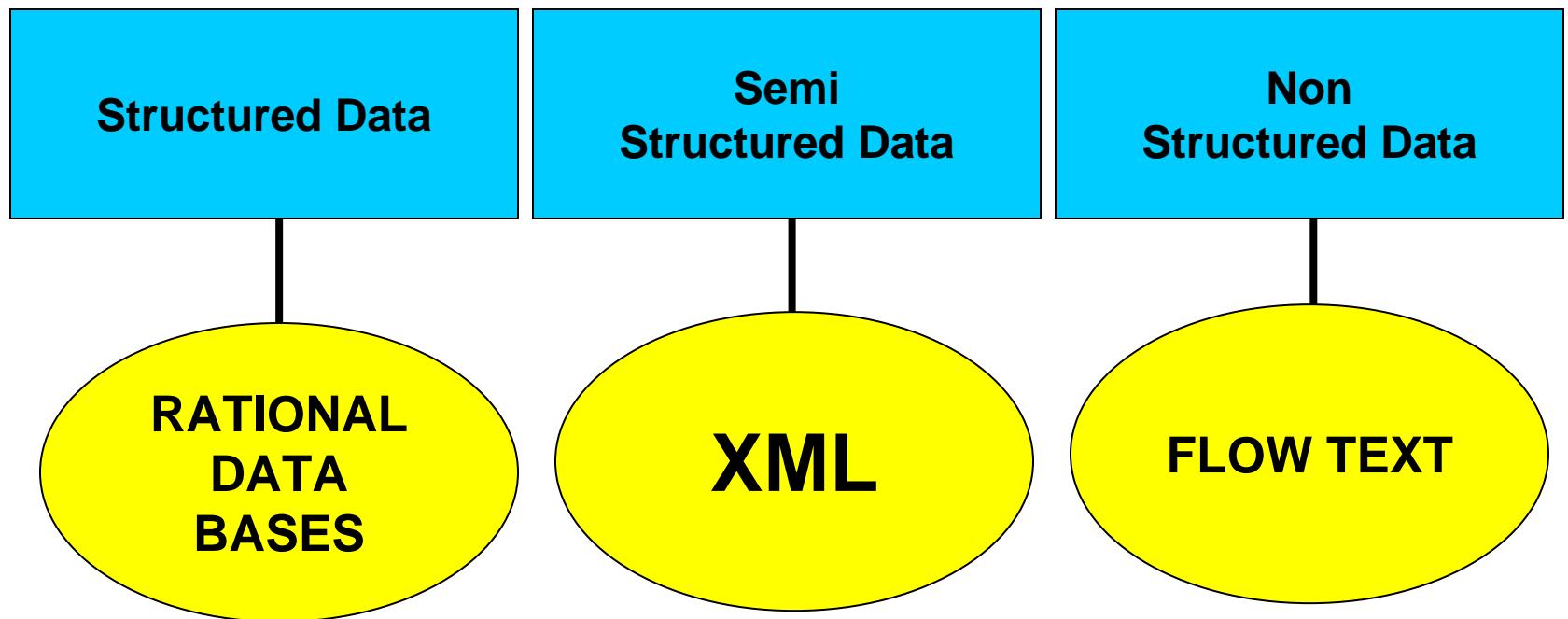
What is XML

- XML (EXtensible Markup Language)
- Guidance for writing of Document Type Definitions ☐ DTD's
- XML is more general and uniform than HTML, and simpler than SGML
- Standardized General Markup Language

XML - Motivation

- For closing gaps between machine-machine communication ...
- ... applicable in the WWW
- Easy to create
- For humans and machines readable
- To cover as much as possible areas of application

XML - Classification



Sample Code

```
<?xml version=„1.0“ encoding=„ISO-8859-1“ ?>
<book isbn=„3423085169“>
<title>Sofies Welt</title>
<author>
<name>Gaarder</name>
<firstname>Jostein</firstname>
</author>
<publisher>DTV</ publisher>
<year>1993</year>
</book>
```

End - Conclusion

- HTML
 - Markup language for the Web
 - Version and compatibility problems
- XHTML
 - Tryes to solve this problems
- XML
 - machine-machine communication

XHTML DTD

- An XHTML DTD describes the allowed syntax and grammar of XHTML mark-up. Every XHTML document *must start* with a DTD declaration and a line of code that declares that you are starting to write XHTML code.
- This is the mandatory minimum way to start an XHTML document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

The Three Document Type Definitions

There are currently three XHTML 1.0 document types:

1. Strict
2. Transitional
3. Frameset

XHTML 1.0 Strict

- This does not allow use of any type of deprecated tags.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

- Use this when taking advantage of XHTML's presentational features and when you want to support browsers that do not understand Cascading Style Sheets. This allows the use of deprecated tags.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

- Use this when you want to use XHTML Frames to partition the browser window into two or more frames. This allows both deprecated tags and frames.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 frameset//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

HTML 5

First Look at HTML5

Remember the DOCTYPE declaration from XHTML?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

Just 15 characters!

The DOCTYPE tells the browser which type and version of document to expect. This should be the last time the DOCTYPE is ever changed. From now on, all future versions of HTML will use this same simplified declaration.

The <html> Element

This is what the <html> element looked like in XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
      lang="en">
```

Again, HTML5 simplifies this line:

```
<html lang="en">
```

The **lang** attribute in the <html> element declares which language the page content is in. Though not strictly required, it should always be specified, as it can assist search engines and screen readers.

Each of the world's major languages has a two-character code, e.g. Spanish = "es", French = "fr", German = "de", Chinese = "zh", Arabic = "ar".

The <head> Section

Here is a typical XHTML <head> section:

```
<head>
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
  <title>My First XHTML Page</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

And the HTML5 version:

```
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
```

Notice the simplified character set declaration, the shorter CSS stylesheet link text, and the removal of the trailing slashes for these two lines.

Basic HTML5 Web Page

Putting the prior sections together, and now adding the <body> section and closing tags, we have our first complete web page in HTML5:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>HTML5 is fun!</p>
</body>
</html>
```

Let's open this page in a web browser to see how it looks...

Meta tag:Definition and Usage

- Metadata is data (information) about data.
- The `<meta>` tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.
- Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.
- The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.
- HTML5 introduced a method to let web designers take control over the viewport (the user's visible area of a web page), through the `<meta>` tag (See "Setting The Viewport" example below).

HTML <meta> Tag

```
<head>
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
<meta name="author" content="John Doe">
<meta name="viewport" content="width=device-width, initial-
    scale=1.0">
</head>
```

Chapter 4

CSS

Style Sheets

Need Of CSS

- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- **CSS** to define how HTML elements should appear.

CSS Selectors

1. Element Selector
2. Id Selector
3. Class Selector
4. Group selector

Element Selector

- The element selector select the element based on element name

```
p  
{  
Background-color:red;  
}
```

Id selector

- Id selector use the id attribute of the HTML element to select specific element.

```
#abc
```

```
{
```

```
background-color:red;
```

```
}
```

Class selector

- The class selector element based on class attribute.

.abc

{

Background-color:red;

}

Group selector

- If we have elements with same style.

p

```
{ background-color:red;}
```

h1

```
{ background-color:red;}
```

we can write above code as

p,h1

```
{ background-color:red;}
```

What is CSS?

- **CSS** stands for **Cascading Style Sheets**
- CSS describes **how HTML elements are to be displayed on screen.**
- CSS **saves a lot of work.** It can control the layout of multiple web pages all at once.**[With an external stylesheet file, you can change the look of an entire website by changing just one file!]**
- External style sheets are stored in **CSS files.**

CSS Backgrounds

- **CSS background properties:**

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

Background Color

- The background-color property specifies the background color of an element.

Example

```
body
```

```
{
```

```
    background-color: lightblue;
```

```
}
```

Background Color

Example 2

- Example

```
h1 {  
    background-color: green;  
}
```

```
div {  
    background-color: lightblue;  
}
```

```
p {  
    background-color: yellow;  
}
```

Background Image

- The background-image property specifies an image to use as the background of an element.
- By default, the image is repeated so it covers the entire element.

```
body {  
    background-image: url("paper.gif");  
}
```

Repeat Horizontally or Vertically

```
body
{
    background-image: url("gradient_bg.png");
    background-repeat: repeat-x;
}
```

Background-repeat values:

repeat-x

repeat-y

no-repeat

Background-position

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

-
- It has three different types of values:
Length values (e.g. 100px 5px)
Percentages (e.g. 100% 5%)
Keywords (e.g. Right top)

Background-position

- The default values are 0 0. This places your background image at the top left of the container.
- The first value is the horizontal position, second value is the vertical position.
- So **100px 5px** will move the image 100px to the right and five pixels down.

Background-position

- Keywords are just shortcuts for percentages. It's easier to remember and write top right than 0 100%, and that's why keywords are a thing. Here is a list of all five keywords and their equivalent values:

top: 0% vertically

right: 100% horizontally

bottom: 100% vertically

left: 0% horizontally

center: 50% horizontally if horizontal isn't already defined. If it is then this is applied vertically.

Background Image - Fixed position

```
body
{
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
    background-attachment: fixed;
}
```

Shorthand Property Background

```
body
```

```
{  
background: #ffffff url("img_tree.png") no-repeat fixed right top;  
}
```

Sequence:

background-color
background-image
background-repeat
background-attachment
background-position

CSS Borders

Different border Properties

- border-width:medium/thick/thin
- border-style (required)
- border-color
- Border-collapse: initial/seperate/collapse

Border Width

- The border-width property specifies the width of the four borders.

one

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

Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value

Border Style

- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

Border Color

The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
.one {  
    border-style: solid;  
    border-color: red;  
}
```

Border - Individual Sides

- From the examples above you have seen that it is possible to specify a different border for each side.

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

Border - Shorthand Property

```
p {  
    border: 5px solid red;  
}
```

Sequence

- border-width
- border-style
- border-color

CSS Margins

- The CSS margin properties are used to generate space around elements.
- The margin properties set the size of the white space outside the border.
- CSS has properties for specifying the margin for each side of an element:
 1. margin-top
 2. margin-right
 3. margin-bottom
 4. margin-left

Example

```
p {  
    margin-top: 100px;  
    margin-bottom: 100px;  
    margin-right: 150px;  
    margin-left: 80px;  
}
```

Margin - Shorthand Property

```
p {  
    margin: 100px 150px 100px 80px;  
}
```

Sequence

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

Margin - Shorthand Property

margin: 100px 150px 100px 80px;

Top Right Bottom Left

margin: 100px 150px 100px;

Top Right/Left Bottom

margin: 100px 150px;

Top/bottom Right/left

Marging: 100px;

All

CSS Padding

- The CSS padding properties are used to generate space around content.
- The padding clears an area around the content (inside the border) of an element
 1. padding-top
 2. padding-right
 3. padding-bottom
 4. padding-left

Example

- p {
 padding-top: 50px;
 padding-right: 30px;
 padding-bottom: 50px;
 padding-left: 80px;
}

Padding - Shorthand Property

```
p {  
    padding: 50px 30px 50px 80px;  
}
```

Sequence

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

Padding - Shorthand Property

padding: 100px 150px 100px 80px;

Top Right Bottom Left

padding: 100px 150px 100px;

Top Right/Left Bottom

padding: 100px 150px;

Top/bottom Right/left

padding: 100px;

All

CSS Text

TEXT CSS Properties

- color
- Text-alignment: center/inherit/justify/left/right
- text-decoration:blink/inherit/line-through/none/overline/underline
- Text-transform:capitalize/inherit/lowercase/uppercase
- Text-indent
- Letter-spacing
- Line-Height
- Direction:rtl/ltr
- Word-spacing
- Text-shadow: *h-shadow v-shadow color|none|initial|inherit;*

Text Color and Text Align

```
h1 {  
    color: green;  
}
```

```
h1 {  
    text-align: center/left/right;  
}
```

Text Decoration

```
a {  
    text-decoration: none;  
}
```

Other Values for Text Decoration

overline

line-through

underline

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

```
p {  
    text-transform: uppercase;  
}
```

- Other values should be
 - uppercase
 - lowercase
 - capitalize

Text Indentation

- The text-indent property is used to specify the indentation of the first line of a text:

```
p {  
    text-indent: 50px;  
}
```

Output

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel.

Letter Spacing & Line Height

```
h1 {  
    letter-spacing: 3px;  
}
```

```
p {  
    line-height: 0.8;  
}
```

Text Direction and Word Spacing

- The direction property is used to change the text direction of an element:

```
div {  
    direction: rtl;  
}
```

The word-spacing property is used to specify the space between the words in a text.

```
h1 {  
    word-spacing: 10px;  
}
```

Text Shadow

- The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

```
h1 {  
    text-shadow: 3px 2px red;  
}
```

CSS FONT

CSS Font Family

- The font family of a text is set with the font-family property
- Example

```
p {  
    font-family: Times New Roman, Times, serif;  
}
```

Font Style

- The font-style property is mostly used to specify italic text.
- This property has three values:
 - normal - The text is shown normally
 - italic - The text is shown in italics
 - oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Style

- Example

```
p {  
    font-style: normal;  
}
```

Font Size

```
h1 {  
    font-size: 40px;  
}
```

Set Font Size With Em [emphemerical unit]

pixels/16=em

e.g. 16pixels is 1em.

Font Weight

- The font-weight property specifies the weight of a font:

```
p {  
    font-weight: normal/bold;  
}
```

Font Variant

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
- In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Font Variant: Example

```
p{  
    font-variant: normal;  
}
```

```
h1 {  
    font-variant: small-caps;  
}
```

CSS Links

Styling Links

The four links states are:

- **a:link** - a normal, unvisited link
- **a:visited** - a link the user has visited
- **a:hover** - a link when the user mouse over it.
- **a:active** - a link the moment it is clicked

- Example

```
a:link {  
    color: red;  
}  
/* visited link */  
a:visited {  
    color: green;  
}  
/* mouse over link */  
a:hover {  
    color: hotpink;  
}  
/* selected link */  
a:active {  
    color: blue;  
}
```

Text Decoration by Link

- ```
a:link {
 text-decoration: none;
}

a:visited {
 text-decoration: none;
}

a:hover {
 text-decoration: underline;
}

a:active {
 text-decoration: underline;
}
```

# Background Color by link

- ```
a:link {  
    background-color: yellow;  
}  
  
a:visited {  
    background-color: cyan;  
}  
  
a:hover {  
    background-color: lightgreen;  
}  
  
a:active {  
    background-color: hotpink;  
}
```

Advanced - Link

```
a:link, a:visited {  
background-color: #f44336;  
color: white;  
padding: 14px 25px;  
text-align: center;  
text-decoration: none;  
display: inline-block;  
}  
  
a:hover, a:active {  
background-color: yellow;  
color: black;  
text-decoration: underline;  
}
```

Different List Item Markers

- **list-style-type**:circle/square/upper-roman/lower-alpha
- **list-style-image**:url('imagename')
- **list-style-position**:inside/outside

Example

- ul{
 list-style-type: circle;
}
- ul {
 list-style-image: url('sqpurple.gif');
}
- ul {
 list-style-position: inside;
}

List - Shorthand property

- ul {
 list-style: square inside url("sqpurple.gif");
}
- 1. list-style-type
- 2. list-style-position
- 3. list-style-image

CSS Layout - The position Property

- The position Property

The position property specifies the type of positioning method used for an element.

There are four different position values:

1. static
2. relative
3. fixed
4. absolute

position: static;

- Static positioned elements are not affected by the top, bottom, left, and right properties.
- ```
div.static {
 position: static;
 border: 3px solid #73AD21;
}
```

# position: relative;

- An element with position: relative; is positioned relative to its normal position.
- ```
div.relative {  
    position: relative;  
    left: 30px;  
    border: 3px solid #73AD21;  
}
```

position: fixed;

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- ```
div.fixed {
 position: fixed;
 bottom: 0;
 right: 0;
 width: 300px;
 border: 3px solid #73AD21;
}
```

# position: absolute;

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

# Example

- ```
div.relative {  
    position: relative;  
    width: 400px;  
    height: 200px;  
    border: 3px solid #73AD21;  
}  
  
div.absolute {  
    position: absolute;  
    top: 80px;  
    right: 0;  
    width: 200px;  
    height: 100px;  
    border: 3px solid #73AD21;  
}
```

CSS Combinators

- There are four different combinator types in CSS3:
 1. descendant selector (space)
 2. child selector (>)
 3. adjacent sibling selector (+)
 4. general sibling selector (~)

Descendant Selector

- The descendant selector matches all elements that are descendants of a specified element.
- ```
div p {
 background-color: yellow;
}
```
- In above example, Apply properties of all paragraph of div

# Example

```
<div>
 <p>Paragraph 1 in the div.</p>
 <p>Paragraph 2 in the div.</p>
 <p>Paragraph 3 in the div.</p>
</div>
```

# Child Selector

- The child selector selects all elements that are the immediate children of a specified element.
- ```
div > p {  
    background-color: yellow;  
}
```

Example

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
</div>
```

Adjacent Sibling Selector

- The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.
- ```
div + p {
 background-color: yellow;
}
```
- Apply to next immediate element after completion of div element

# Example

```
<div>
 <p>Paragraph 1 in the div.</p>
 <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>
```

# General Sibling Selector

- The general sibling selector selects all elements that are siblings of a specified element.
- ```
div ~ p {  
    background-color: yellow;  
}
```

Example

```
<p>Paragraph 1.</p>
```

```
<div>
  <code>Some code.</code>
  <p>Paragraph 2.</p>
</div>
```

```
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
```

CSS Pseudo-classes

- What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

Style an element when a user mouses over it

Style visited and unvisited links differently

Style an element when it gets focus

Syntax

selector:pseudo-class

{

 property:value;

}

Anchor Pseudo-classes

- ```
a:link {
 color: #FF0000;
}
```

```
/* visited link */
a:visited {
 color: #00FF00;
}
```

```
/* mouse over link */
a:hover {
 color: #FF00FF;
}
```

```
/* selected link */
a:active {
 color: #0000FF;
}
```

# Pseudo-classes and CSS Classes

- Pseudo-classes can be combined with CSS classes:

```
a.highlight:hover
```

```
{
 color: red;
}
```

# Hover on <div>

div:hover

```
{
 background-color: blue;
}
```

```
<style>
p {
 display: none;
 background-color: yellow;
 padding: 20px;
}

div:hover p {
 display: block;
}
</style>
</head>
<body>
```

```
<div>Hover over me to show the p element
 <p>Tada! Here I am!</p>
</div>
```

# CSS - The :first-child Pseudo-class

```
p:first-child
```

```
{
 color: blue;
}
```

Apply to first paragraph of body part

# Element within first child

```
p i:first-child
```

```
{
 color: blue;
}
```

Apply to first child italic of paragraph element.

```
p:first-child i
```

```
{
```

```
 color: blue;
```

```
}
```

Apply to all italic elements of first paragraph element.

# CSS Pseudo-elements

- What are Pseudo-Elements?
  - a) A CSS pseudo-element is used to style specified parts of an element.
  - b) For example, it can be used to:
    - Style the first letter, or line, of an element
    - Insert content before, or after, the content of an element

# Syntax

selector::pseudo-element

```
{
 property:value;
}
```

# The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

```
p::first-line
```

```
{
 color: #ff0000;
 font-variant: small-caps;
}
```

Apply css to first line of the paragraph.

# The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

```
p::first-letter
```

```
{
 color: #ff0000;
 font-size: xx-large;
}
```

# Pseudo-elements and CSS Classes

- Pseudo-elements can be combined with CSS classes:

```
p.intro::first-letter
```

```
{
 color: #ff0000;
 font-size:200%;
}
```

# Multiple Pseudo-elements

- Several pseudo-elements can also be combined.

```
p::first-letter {
 color: #ff0000;
 font-size: xx-large;
}
```

```
p::first-line {
 color: #0000ff;
 font-variant: small-caps;
}
```

# CSS - The ::before Pseudo-element

- The ::before pseudo-element can be used to insert some content before the content of an element.

```
h1::before
```

```
{
 content: url(smiley.gif);
}
```

# CSS - The ::after Pseudo-element

- The ::after pseudo-element can be used to insert some content after the content of an element.

```
h1::after
{
 content: url(smiley.gif);
}
```

# CSS3 Modules

- CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.

# CSS3 Modules

- Some of the most important CSS3 modules are:
  1. Selectors
  2. Box Model
  3. Backgrounds and Borders
  4. Image Values and Replaced Content
  5. Text Effects
  6. 2D/3D Transformations
  7. Animations
  8. Multiple Column Layout
  9. User Interface

# CSS3 Rounded Corners

- With the CSS3 border-radius property, you can give any element "rounded corners".
- ```
#a {  
    border-radius: 25px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}
```

CSS3 border-radius - Specify Each Corner

- **Four values:** first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner
- **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right
- **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner
- **One value:** all four corners are rounded equally

CSS3 Border Images

- ```
#borderimg1 {
 border: 10px solid transparent;
 padding: 15px;
 -webkit-border-image: url(border.png) 50
 round; /* Safari 3.1-5 */
 -o-border-image: url(border.png) 50
 round/stretch; /* Opera 11-12.1 */
 border-image: url(border.png) 50 round/stretch;
}
```

# CSS3 Backgrounds

- background-size: contain/cover/values h v
- background-origin: border-box/padding-box/content-box
- background-clip

# CSS3 background-origin Property

- The CSS3 background-origin property specifies where the background image is positioned.
- The property takes three different values:
  1. border-box - the background image starts from the upper left corner of the border
  2. padding-box - (default) the background image starts from the upper left corner of the padding edge
  3. content-box - the background image starts from the upper left corner of the content

# Example

- ```
#abc {  
    border: 10px solid black;  
    padding: 35px;  
    background: url(img_flwr.gif);  
    background-repeat: no-repeat;  
    background-origin: content-box;  
}
```

CSS3 background-clip Property

- The CSS3 background-clip property specifies the painting area of the background.
- The property takes three different values:
 1. border-box - (default) the background is painted to the outside edge of the border
 2. padding-box - the background is painted to the outside edge of the padding
 3. content-box - the background is painted within the content box

Example

- ```
#abc1 {
 border: 10px dotted black;
 padding: 35px;
 background: yellow;
 background-clip: content-box;
}
```

- Thank you So muchhh.....

# Chapter 5

JS

JAVASCRIPTS

# USE of JavaScript

- JavaScript is one of the **3 languages** all web developers **must** learn:
  1. **HTML** to define the content of web pages.
  2. **CSS** to specify the layout of web pages.
  3. **JavaScript** to program the behaviour of web pages.

# JavaScript Introduction

- Change HTML Content

```
document.getElementById("demo").innerHTML
= "Hello JavaScript";
```

- Change HTML Attributes

```
<input type="button"
onclick="document.getElementById('myImage').src='
pic_bulbon.gif' ">
```

# JavaScript Introduction

- Change HTML Styles (CSS)

```
document.getElementById("demo").style.fontSize = "25
px";
```

- Hide HTML Elements

```
document.getElementById("demo").style.display = "
none";
```

# JavaScript Introduction

- Show HTML Elements

```
document.getElementById("demo").style.displa
y = "block";
```

# Define

- In HTML, JavaScript code must be inserted between `<script>` and `</script>` tags.
- Example

```
<script>
document.getElementById("demo").innerHTML =
"My First JavaScript";
</script>
```

# JavaScript in <head> EXAMPLE

```
<html>
<head>
 <script>
 function myFunction()
 {
 document.getElementById("demo").innerHTML = "Paragraph changed.";
 }
 </script>
</head>
```

```
<body>
<p id="demo">A Paragraph</p>

<input type="button" onclick="myFunction()"
 value="Try it">

</body>
</html>
```

# JavaScript in <body>

- In this example, a JavaScript function is placed in the <body> section of an HTML page.
  - The function is invoked (called) when a button is clicked:
  - Placing scripts at the bottom of the <body> element improves the display speed, because script compilation slows down the display.
- Example:
- Next Slide

- <!DOCTYPE html>  
<html>  
<body>  
  
<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<input type="button" onclick="myFunction()" value="Try it">  
  
<script>  
function myFunction() {  
 document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

# External JavaScript

- External file named: abc.js
- In this file we can directly define function for the particular event.
- Example

```
function myFunction()
{
 document.getElementById("demo").innerHTML
 = "Hello.";
}
```

# How to Include External JS

- We can include external Js anywhere in the HTML page like below code:

```
<script src="myScript.js"></script>
```

You can place an external script reference in `<head>` or `<body>` as you like.

Note:

External scripts cannot contain `<script>` tags.

# External JavaScript Advantages

- Placing scripts in external files has some advantages:
  1. It separates HTML and code.
  2. It makes HTML and JavaScript easier to read and maintain.
  3. Cached JavaScript files can speed up page loads.

# JavaScript Display Possibilities

- JavaScript can "display" data in different ways:
  1. Writing into an alert box,  
using **window.alert()**.
  2. Writing into the HTML output  
using **document.write()**.
  3. Writing into an HTML element,  
using **innerHTML**.
  4. Writing into the browser console,  
using **console.log()**.

# Using window.alert()

- You can use an alert box to display data:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert("Hello How are u?");
</script>

</body>
</html>
```

# Using document.write()

- <!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<script>  
document.write(5 + 6);  
</script>  
  
</body>  
</html>

# Using innerHTML

- <!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My First Paragraph</p>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = 5 + 6;  
</script>  
  
</body>  
</html>

# Using console.log()

- <!DOCTYPE html>  
<html>  
<body>

```
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
```

```
<script>
console.log(5 + 6);
</script>
```

```
</body>
</html>
```

# JavaScript Statements

- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.

```
document.getElementById("demo").innerHTML = "Hello Dolly.;"
```

# JavaScript Programs

- The statements are executed, one by one, in the same order as they are written.
- In this example x, y, and z are given values, and finally z is displayed:
- Example
- ```
var x, y, z;  
x = 5;  
y = 6;  
z = x + y;  
document.getElementById("demo").innerHT  
ML = z;
```

JavaScript Identifiers

- The general rules for constructing names for variables (unique identifiers) are:
 1. Names can contain letters, digits, underscores, and dollar signs.
 2. Names must begin with a letter
 3. Names can also begin with \$ and _
 4. Names are case sensitive (y and Y are different variables)
 5. Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Data Types

- In programming, text values are called text strings.
- JavaScript can handle many types of data, but for now, just think of numbers and strings.
- Strings are written inside double or single quotes. Numbers are written without quotes.
- If you put a number in quotes, it will be treated as a text string.

Example

```
var x = 3; //this is int variable
```

```
var y = 3.2 //this is float variable
```

```
var a = "Hello"; //this is string variable
```

```
var b = 'Hello !'; //this is string variable
```

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- Example

```
function myFunction(p1, p2)
{
    return p1 * p2;          // The function
    returns the product of p1 and p2
}
```

JavaScript Function Syntax

- `function name(parameter1, parameter2, parameter3)`

```
{  
    code to be executed  
}
```

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

- When JavaScript reaches a **return statement**, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**.

Example

- `var x = abc(4, 3); // Function is called,
return value will end up in x`

```
function abc(a, b)  
{  
    return a * b;  
}
```

JAVASCRIPT OPERATORS

JavaScript Operators

- Assignment Operator: it is used to assign the value to variable from right to left.
- We can also apply shorthand operator like C.

Example:

```
var x = 5;      // assign the value 5 to x  
var y = 2;      // assign the value 2 to y  
var z = x + y; // assign the value 7 to z
```

JavaScript Arithmetic Operators

| Operators | Description |
|-----------|----------------|
| + | Addtion |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulo |
| ++ | Increment |
| -- | Decrement |

JavaScript String Operators

- The + operator can also be used to add (concatenate) strings.
- Example

```
txt1 = "Hello";  
txt2 = "Friends";  
txt3 = txt1 + " " + txt2;
```

Adding Strings and Numbers

- Adding two numbers, will return the sum, but adding a number and a string will return a string:

```
x = 2 + 5; // output: 7
```

```
y = "2" + 5; // output: 25
```

```
z = "Hi" + 5; // output: Hi5
```

JavaScript Comparison Operators

| Operators | Description |
|--------------------|-----------------------|
| <code>==</code> | Equal |
| <code>!=</code> | No equal |
| <code>></code> | Greater than |
| <code>>=</code> | Greater than equal to |
| <code><</code> | Less than |
| <code><=</code> | Less than equal to |
| <code>? :</code> | ternary |

```
<script>

var x = 5;

document.getElementById("demo").innerHTML
= (x == 8);

document.getElementById("demo").innerHTML
= (x != 8);

document.getElementById("demo").innerHTML
= (x > 8);

document.getElementById("demo").innerHTML
= (x < 8);

document.getElementById("demo").innerHTML
= (x >= 8);

</script>
```

JavaScript Logical Operators

| Operators | Description |
|-----------|-------------|
| && | Logical AND |
| | Logical OR |
| ! | Logical Not |

```
<script>  
var x = 6;  
var y = 3;  
  
document.getElementById("demo").innerHTML  
=  
(x < 10 && y > 1) + "<br>" +  
(x < 10 && y < 1);  
</script>
```

```
<script>  
var x = 6;  
var y = 3;  
  
document.getElementById("demo").innerHTML  
=  
(x == 5 || y == 5) + "<br>" +  
(x == 6 || y == 5) + "<br>" +  
(x == 6 || y == 3);  
</script>
```

```
<script>

var x = 6;

var y = 3;

document.getElementById("demo").innerHTML

  =

!(x == y) + "<br>" +
!(x > y);

</script>
```

JavaScript Type Operators

| Operators | Description |
|------------|--|
| typeof | Returns type of variable |
| instanceof | Returns true if an object is an instance of an object type |

JavaScript Bitwise Operators

| Operators | Description |
|-----------|-------------|
| & | AND |
| | OR |
| ~ | NOT |
| ^ | XOR |
| << | Left shift |
| >> | Right Shift |

JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- Example

```
<script>
```

```
var a = ["rajan","suresh","ramesh"];
```

```
document.getElementById("abc").innerHTML =  
    a[1];
```

```
</script>
```

JavaScript Objects

- JavaScript objects are written with curly braces.
- Object properties are written as **name:value** pairs, separated by commas.
- Example

```
var person = {firstName:"Hiren",
lastName:"Mer", age:29};
```

```
<html>
<body>
<p id="abc"></p>
<script>
var person = {
    firstName : "HIREN",
    lastName : "MER",
    age : 30
};
document.getElementById("abc").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

JavaScript - if...else Statement

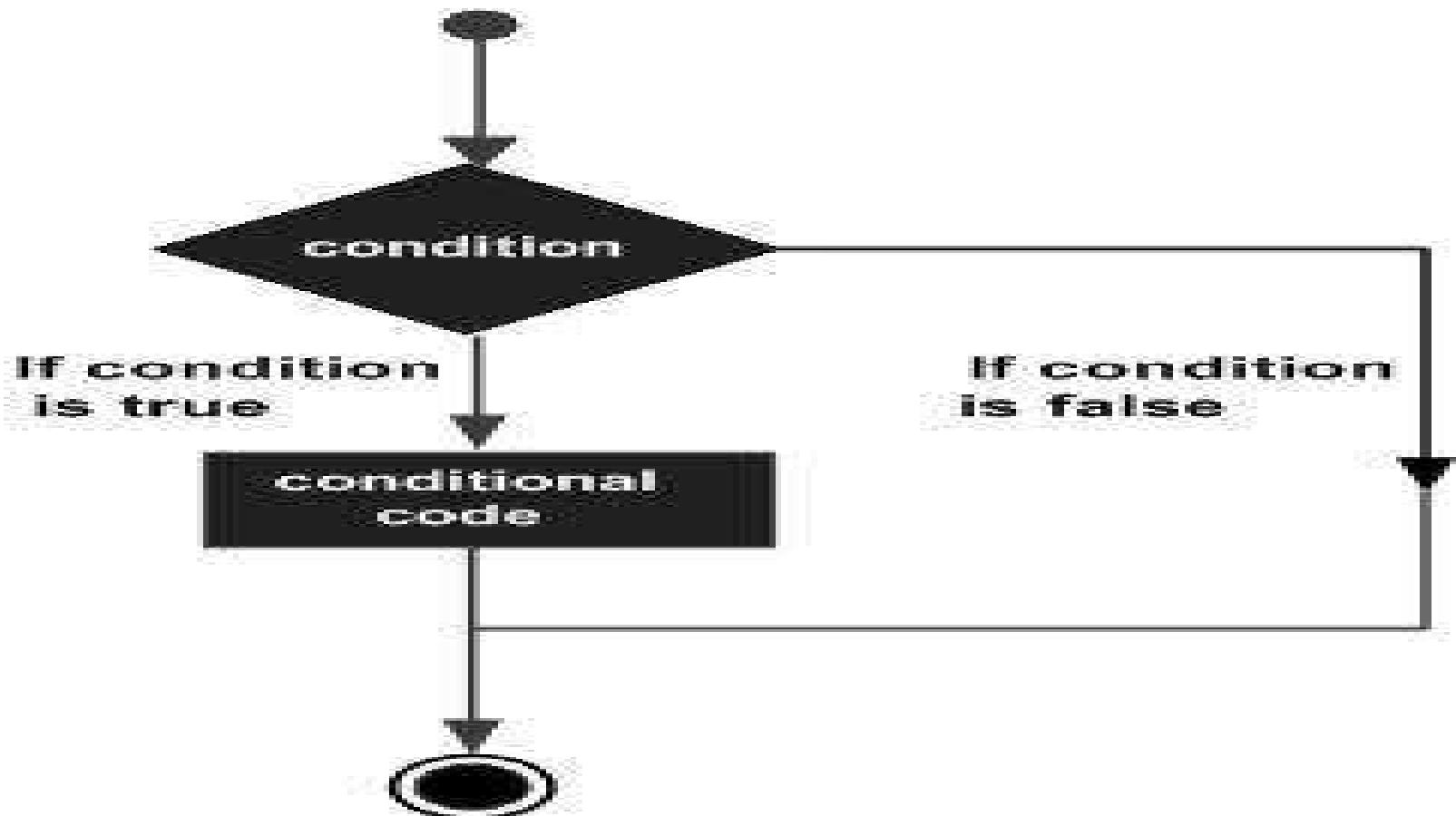
- In JavaScript we have the following conditional statements:
 1. Use **if** to specify a block of code to be executed, if a specified condition is true
 2. Use **else** to specify a block of code to be executed, if the same condition is false
 3. Use **else if** to specify a new condition to test, if the first condition is false
 4. Use **switch** to specify many alternative blocks of code to be executed

The if Statement

- Syntax

```
if (condition)
{
    Statement-x // if true
}
```

Flow Chart of IF



If Example

```
<body>
<h1 id="abc"></h1>
<script>
Var x=10;
if (x < 18)
{
    document.getElementById("abc").innerHTML = "Good
    day!";
}
</script>
</body>
```

The if..else Statement

```
if (condition)
```

```
{
```

```
    statement-x //if condition true
```

```
}
```

```
else
```

```
{
```

```
    statement-y //if condition false
```

```
}
```

```
<body>
<p>Click the button to display greeting:</p>

<input type="button" onclick="myFunction()" value="click Me">

<p id="abc"></p>
<script>
function myFunction() {
    var x; var greeting;
    if (x < 18)
    {
        greeting = "Good day";
    }
    else
    {
        greeting = "Good evening";
    }
    document.getElementById("abc").innerHTML = greeting;
}
</script>
<body>
```

The else if Statement

- Syntax

```
if (condition1) {  
    statement-x // if true  
} else if (condition2) {  
    statement-y // if true  
} else {  
    statement-z // if all condition false  
}
```

-

- Example

```
if (time < 10)
{
    greeting = "Good morning";
}

else if (time < 20) {
    greeting = "Good day";
}

else {
    greeting = "Good evening";
}
```

JavaScript Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.
- Syntax

```
switch(expression) {
```

```
    case n:
```

```
        code block
```

```
        break;
```

```
    case n:
```

```
        code block
```

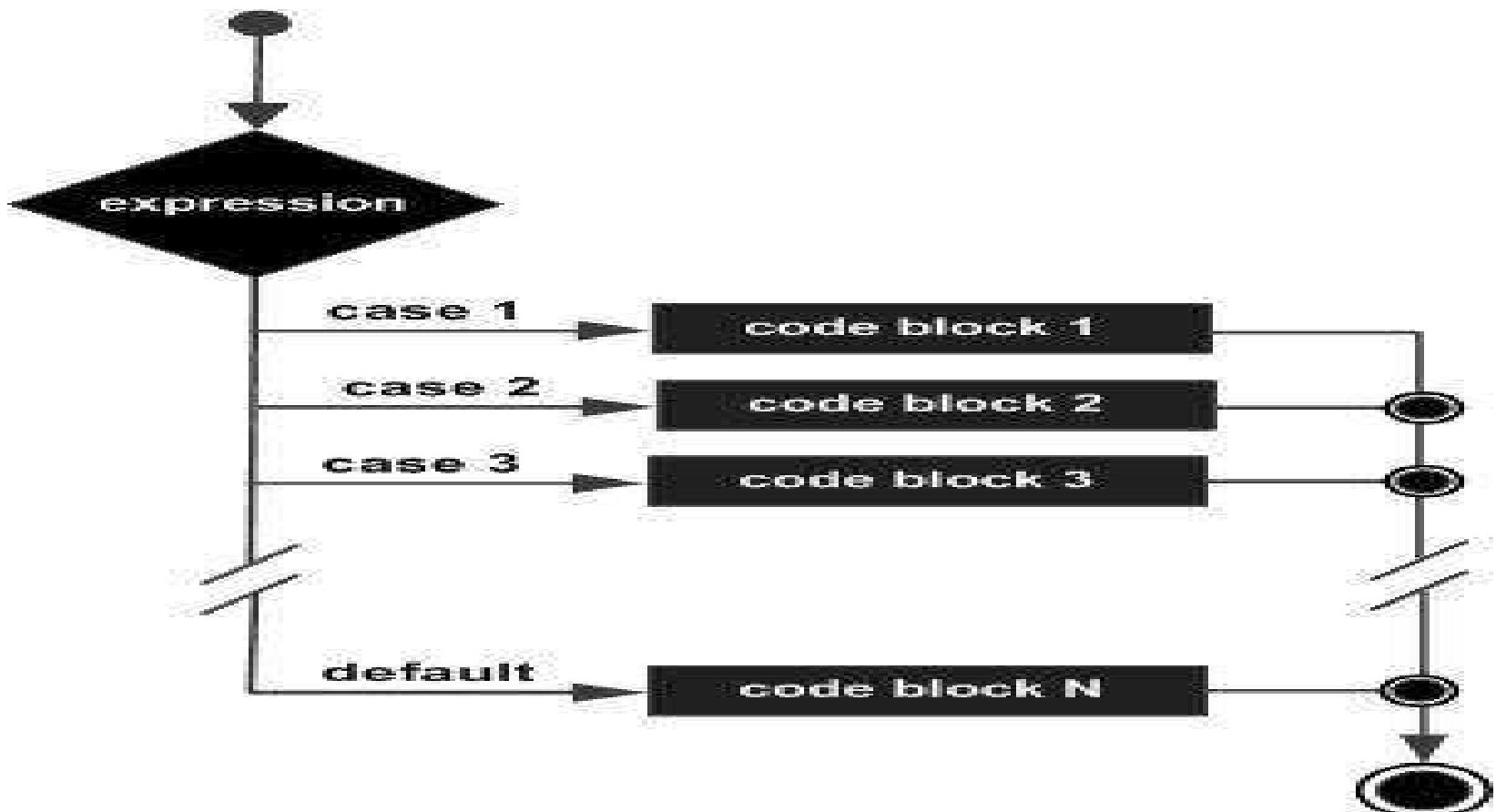
```
        break;
```

```
    default:
```

```
        code block
```

```
}
```

Flow Chart of SWITCH case



Example:

```
<script>  
Var day=5;  
switch (day) {  
    case 0:  
        day = "Sunday";  
        break;  
    case 1:  
        day = "Monday";  
        break;  
    case 2:  
        day = "Tuesday";  
        break;
```

case 3:

```
day = "Wednesday";  
break;
```

case 4:

```
day = "Thursday";  
break;
```

case 5:

```
day = "Friday";  
break;
```

case 6:

```
day = "Saturday";
```

```
default:  
    window.alert("Invalid input");  
    break;  
}  
  
document.getElementById('def').innerHTML =  
    "Today is " + day;  
</script>
```

Javascript Loop

Different Kinds of Loops

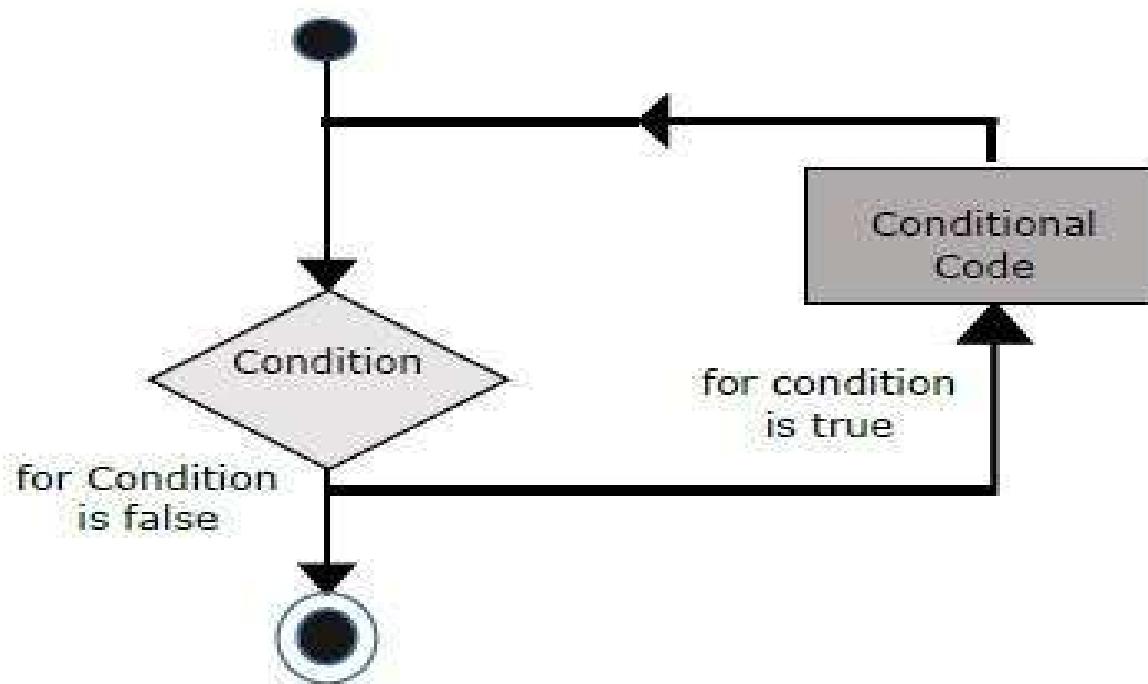
- JavaScript supports different kinds of loops:
 1. **for** - loops through a block of code a number of times
 2. **for/in** - loops through the properties of an object
 3. **while** - loops through a block of code while a specified condition is true
 4. **do/while** - also loops through a block of code while a specified condition is true

JavaScript For Loop

- Loops, if you want to run the same code over and over again, each time with a different value.
- Syntax

```
for (initialization; condition; increment/decrement)  
{  
    loop body;  
}
```

Flow chart of FOR loop



Example

```
<script>  
var a = ["rajan", "ramesh", "Suresh"];  
var x = "";  
var i;  
for (i = 0; i < a.length; i++)  
{  
    x = x + cars[i] + "<br>";  
}  
document.getElementById("test").innerHTML = x;  
</script>
```

The For/In Loop

- Syntax

```
for (var in object)
```

```
{  
    body of loop;  
}
```

- Example1

```
var person =  
{fname:“Kapil”, lname:“Sharma”, age:25};
```

```
var text = “”;  
var x;  
for (x in person) {  
    text = text + person[x];  
}
```

- Example 2

```
<script>

var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat",
    "Audi"];

var text = "";

var i;

for (i in cars) {
    text = text + cars[i] + "<br>";

}

document.getElementById("abc").innerHTML = text;
</script>
```

The While Loop

- Syntax

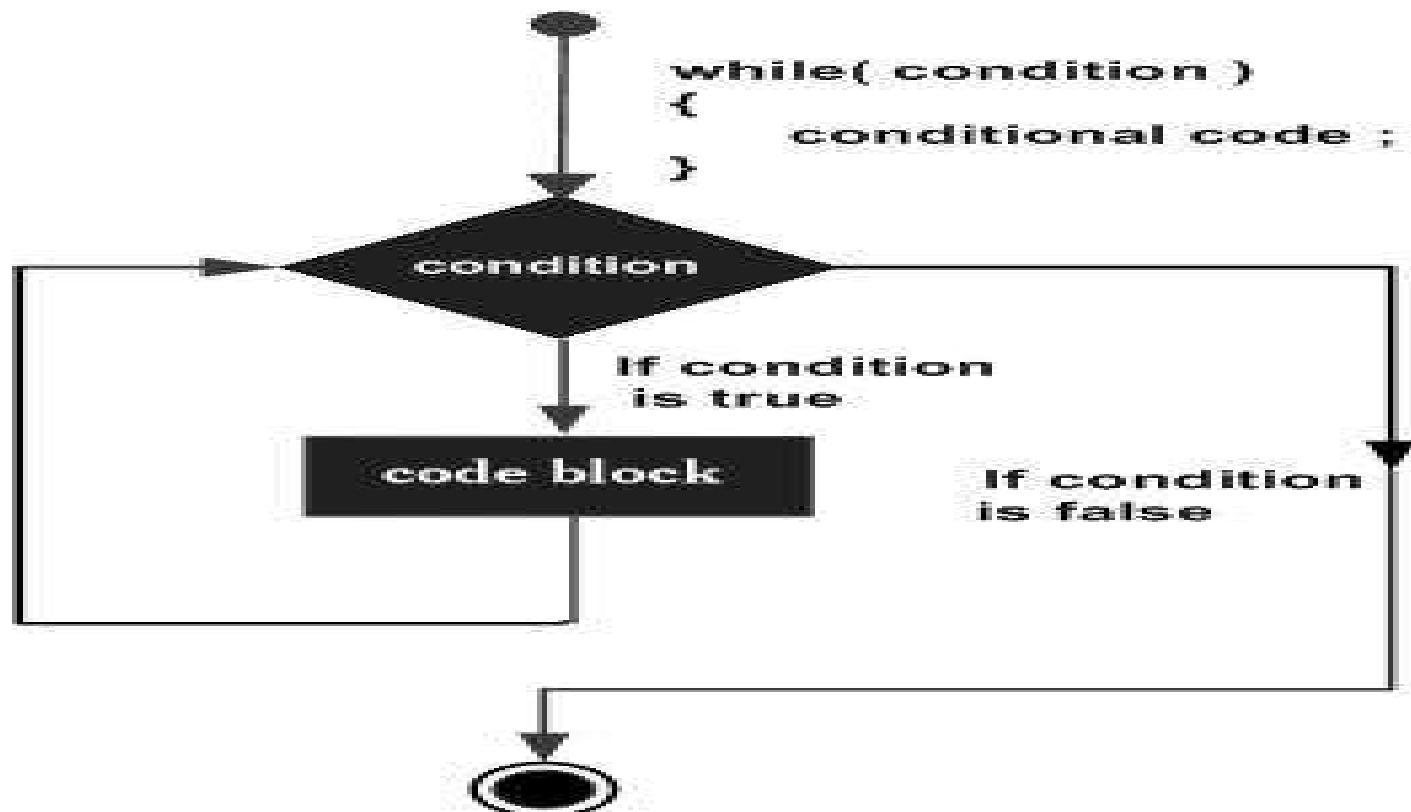
`while (condition)`

{

code block to be executed

}

Flow Chart of While Loop



- Example

```
<h1>JavaScript while</h1>
```

```
<p id="abc"></p>
```

```
<script>
var text = "";
var i = 0;
while (i < 10) {
    text = text + "<br>The number is " + i;
    i++;
}
document.getElementById("abc").innerHTML = text;
</script>
```

The Do/While Loop

- Syntax

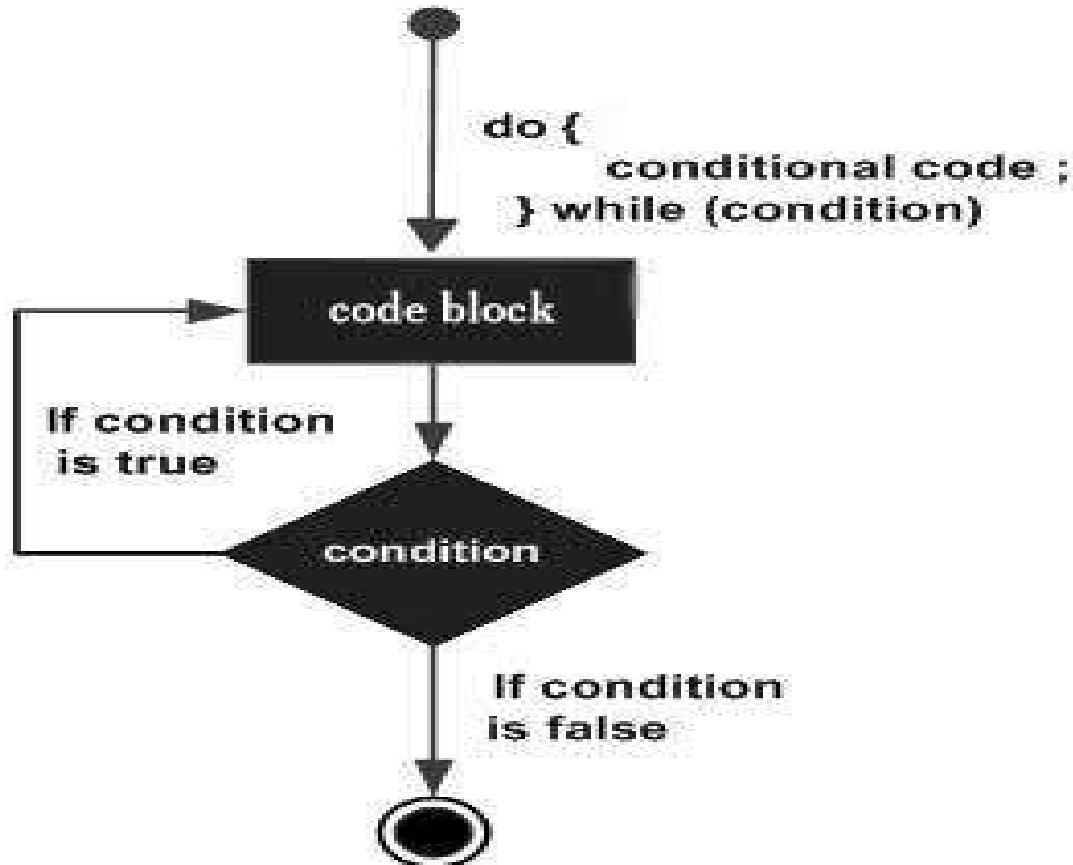
```
do {
```

code block to be executed

```
}
```

```
while (condition);
```

Flow chart of Do While Loop



- Example

```
do {  
    text = text + "The number is " + i;  
    i++;  
}  
while (i < 10);
```

JavaScript - Page Redirection

- You can redirect the page using click event and automatically by setTimeout function.
- Using window.location function we can redirect page to respective page which you want.

```
window.location="http://www.google.com";
```

- Example

```
<script type="text/javascript">  
function redirect() {  
    window.location="http://www.google.com";  
}  
</script>  
<p>Click the This button</p>  
<input type="button" value="Redirect Me"  
      onclick="redirect();"/>
```

Redirect using setTimeout

- We can call any function using setTimeout built-in function.

```
setTimeout('functionname', 10000);
```

10000 =10 seconds

Java script POP UP Boxes

- Alert Dialog Box

alert (message);

- Confirmation Dialog Box

confirm (message);

- Prompt Dialog Box

prompt (message);

Chapter 5 PART 2

Advance JAVASCRIPTS

JavaScript and Object

We can Create the Object for the Javascript

```
var person = {  
    firstName:“Kapil ”,  
    lastName:“Sharma”,  
    age:30,  
};
```

With **new** keyword

```
var person=new Object ();  
person.firstName = "Kapil ";  
person.lastName = "Sharma";  
person.age = 30;
```

Different Examples

```
var x1 = new Object(); // A new Object object  
var x2 = new String(); // A new String object  
var x3 = new Number(); // A new Number object  
var x4 = new Boolean(); // A new Boolean object  
var x5 = new Array(); // A new Array object  
var x6 = new Function(); // A new Function object  
var x7 = new Date(); // A new Date object
```

Object Examples

```
<html>
<head>
    <title>User-defined objects</title>
    <script type="text/javascript">
        var book = new Object();
        book.subject = "WT";
        book.author = "MT Savaliya"; </script> </head>
<body>
    <script type="text/javascript">
        document.write("Book name is : " + book.subject + "<br>");
        document.write("Book author is : " + book.author + "<br>");
    </script>
</body>
</html>
```

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
function book(title, author)
{ this.title = title; this.author = author; } </script>
</head>
<body>
<script type="text/javascript">
var myBook = new book("WT", "MT savaliya");
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
</script>
</body>
</html>
```

```
<html>
<head>
    <title>User-defined objects</title>
    <script type="text/javascript">
        function addPrice(amount)
        {   this.price = amount;   }
        function book(title, author)
        {
            this.title = title;
            this.author = author;
            this.addPrice = addPrice;
        }
    </script>
</head>
```

```
<body>
<script type="text/javascript">
var myBook = new book("WT", "MT Savaliya");
myBook.addPrice(100);
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
document.write("Book price is : " + myBook.price + "<br>");
</script>
</body> </html>
```

Javascript Date Object

- This Object returns today's date and time.

```
<script type="text/javascript">  
var dt = Date();  
document.write("Date and Time : " + dt );  
</script>
```

Output

Date and Time : Tue Feb 07 2017 09:33:34 GMT-0800 (Pacific Standard Time)

getDate()

```
<html>
  <head>
    <title>JavaScript getDate Method</title>
  </head>
  <body>
    <script type="text/javascript">
var dt = new Date("December 25, 1995 23:15:00");
        document.write("getDate() : " + dt.getDate() );
    </script>
  </body>
</html>
```

Date Methods

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Javascript Validation

- Email Validation

`/^w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.w{2,3})+$/`

- Check All letters

`/^[A-Za-z]+$/;`

- Check All Numbers

`/^0-9]+$/;`

- Check Floating point number

`/[-+]?[0-9]+\.[0-9]+$/;`

Javascript Validation

- yyyy-mm-dd

/^(\d{4})-(\d{1,2})-(\d{1,2})\$/

Credit card

/^(\d{4})-(\d{4})-(\d{4})-(\d{4})\$/

/^(?:5[1-5][0-9]{14})\$/;.

Phone number

/^\d{10}\$/;

JAVASCRIPT DOM

DOCUMENT OBJECT MODEL

JAVASCRIPT DOM

- What is the DOM?
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- The W3C DOM standard is separated into 3 different parts:
 1. Core DOM - standard model for all document types
 2. XML DOM - standard model for XML documents
 3. HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

For Example

```
<p id="demo"></p>
```

```
<script>
document.getElementById("demo").innerHTML = "Hello
World!";
</script>
```

- Here **getElementById** is a **method**, while **innerHTML** is a **property**.

The HTML DOM Document Object

- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
- Here we have some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	Description
<code>document.getElementById(<i>id</i>)</code>	Find an element by element id
<code>document.getElementsByTagName(<i>name</i>)</code>	Find elements by tag name
<code>document.getElementsByClassName(<i>name</i>)</code>	Find elements by class name

Changing HTML Elements

Method	Description
<code>element.innerHTML = new html content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.setAttribute(attribute, value)</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style of an HTML element

Adding and Deleting Elements

Method	Description
<code>document.createElement(<i>element</i>)</code>	Create an HTML element
<code>document.removeChild(<i>element</i>)</code>	Remove an HTML element
<code>document.appendChild(<i>element</i>)</code>	Add an HTML element
<code>document.replaceChild(<i>element</i>)</code>	Replace an HTML element
<code>document.write(<i>text</i>)</code>	Write into the HTML output stream

Adding Events Handlers

Method	Description
<code>document.getElementById(<i>id</i>).onclick = function(){<i>code</i>}</code>	Adding event handler code to an onclick event

JavaScript HTML DOM Elements

- Finding HTML Elements

There are some ways to do this:

1. Finding HTML elements by id
2. Finding HTML elements by tag name
3. Finding HTML elements by class name
4. Finding HTML elements by CSS selectors
5. Finding HTML elements by HTML object collections

Finding HTML Element by Id and

```
var myElement =  
    document.getElementById("intro");
```

```
<input type="text" id="intro"/>
```

Here we can get the values from any element by getElementById method.

Finding HTML Elements by Tag Name

```
<p>Hello World!</p>
<p>The DOM is very useful.</p>
<p>This example demonstrates the </p>
<p id="demo"></p>
<script>
var x = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) is: ' + x[0].innerHTML;
</script>
```

Finding HTML Elements by Class Name

```
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example</p>
<p id="demo"></p>

<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph with class="intro": ' + x[0].innerHTML;
</script>
```

Finding HTML Elements by CSS Selectors

```
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>querySelectorAll</b>
    method.</p>
<p id="demo"></p>
<script>
var x = document.querySelectorAll("p.intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' +
    x[0].innerHTML;
</script>
```

JavaScript HTML DOM Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

onclick=JavaScript

List of Events

- Examples of HTML events:
 1. When a user clicks the mouse
 2. When a web page has loaded
 3. When an image has been loaded
 4. When the mouse moves over an element
 5. When an input field is changed
 6. When an HTML form is submitted
 7. When a user strokes a key

Examples on Click

```
<h1 onclick="changeText(this)">Click on this  
text!</h1>
```

```
<script>  
function changeText(id) {  
    id.innerHTML = "Ooops!";  
}  
</script>
```

Examples on Click

```
<input type="button" onclick="displayDate()" value="change">
```

```
<script>
function displayDate()
{
    document.getElementById("demo").innerHTML = Date();
}
</script>
```

```
<p id="demo"></p>
```

Examples on Click

```
<input type="button" id="mybtn" value="change">
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;
function displayDate()
{
    document.getElementById("demo").innerHTML = Date();
}
</script>
```

Body Onload Example

```
<body onload="displayDate ()">
<p id="demo"></p>
<script>
function displayDate()
{
    document.getElementById("demo").innerHTML = Date();
}
</script>
</body>
```

The onchange Event Example

```
<head>
<script>
    function myFunction() {
        var x = document.getElementById("fname");
        x.value = x.value.toUpperCase();
    }
</script>
</head>
<body>
Enter your name:
<input type="text" id="fname" onchange="myFunction()">
```

The onmouseover and onmouseout Events

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)"  
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">  
Mouse Over Me</div>
```

```
<script>  
function mOver(obj) {  
    obj.innerHTML = "Thank You"  
}  
  
function mOut(obj) {  
    obj.innerHTML = "Mouse Over Me"  
}  
</script>
```

The onmousedown, onmouseup and onclick Events

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"  
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">  
Click Me</div>
```

```
<script>  
function mDown(obj) {  
    obj.style.backgroundColor = "#1ec5e5";  
    obj.innerHTML = "Release Me";  
}  
  
function mUp(obj) {  
    obj.style.backgroundColor="#D94A38";  
    obj.innerHTML="Thank You";  
}  
</script>
```

Events

- ## Input Events

1. onblur - When a user leaves an input field
2. onchange - When a user changes the content of an input field
3. onchange - When a user selects a dropdown value
4. onfocus - When an input field gets focus
5. onselect - When input text is selected
6. onsubmit - When a user clicks the submit button
7. onreset - When a user clicks the reset button
8. onkeydown - When a user is pressing/holding down a key
9. onkeypress - When a user is pressing/holding down a key
10. onkeyup - When the user releases a key

Mouse Events

1. onmouseover/onmouseout - When the mouse passes over an element
2. onmousedown/onmouseup - When pressing/releasing a mouse button
3. onmousedown - When mouse is clicked: Alert which element
4. onmousedown - When mouse is clicked: Alert which button
5. onmousemove/onmouseout - When moving the mouse pointer over/out of an image
6. onmouseover/onmouseout - When moving the mouse over/out of an image

Click Events

- onclick - When button is clicked
ondblclick - When a text is double-clicked

Load Events

- onload - When the page has been loaded
onload - When an image has been loaded
onerror - When an error occurs when loading an image
onunload - When the browser closes the document
onresize - When the browser window is resized

Event Listener

HTML DOM EventListener

- Syntax

`document.addEventListener(event, function)`

- The `document.addEventListener()` method attaches an event handler to the document.

Example

- ```
document.addEventListener("click", function()
{
 document.getElementById("demo").innerHTML = "Hello World";
});
```

# Example

- *element.addEventListener("click", myFunction);*

```
function myFunction() {
 alert ("Hello World!");
}
```

# Multiple Event Listener

```
<script>
var x = document.getElementById("myBtn");
x.addEventListener("click", myFunction);
x.addEventListener("click", someOtherFunction);

function myFunction() {
 alert ("Hello World!");
}
function someOtherFunction() {
 alert ("This function was also executed!");
}
</script>
```

```
<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
var x = document.getElementById("myBtn");
x.addEventListener("mouseover", myFunction);
x.addEventListener("click", mySecondFunction);
x.addEventListener("mouseout", myThirdFunction);

function myFunction() {
 document.getElementById("demo").innerHTML += "Moused over!
";
}

function mySecondFunction() {
 document.getElementById("demo").innerHTML += "Clicked!
";
}

function myThirdFunction() {
 document.getElementById("demo").innerHTML += "Moused out!
";
}
</script>
```

# Add an Event Handler to the Window Object

- The `addEventListener()` method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that support events.

# Example

```
<p id="demo"></p>
<script>
window.addEventListener("resize", function(){
 document.getElementById("demo").innerHTML
 = Math.random();
});
</script>
```

# The removeEventListener() method

- *We can remove the event listener which we created by addEventListener method.*

```
element.removeEventListener("mousemove",
 myFunction);
```

# MATH OBJECT

- `Math.PI` // returns 3.141592653589793
- `Math.round()`

## Example

```
Math.round(4.7); // returns 5
```

- `Math.pow()`
- `Math.pow(8, 2); // returns 64`

# MATH OBJECT

- **Math.sqrt()**

```
Math.sqrt(64); // returns 8
```

- **Math.abs()**

```
Math.abs(-4.7); // returns 4.7
```

- **Math.ceil()**

```
Math.ceil(4.4); // returns 5
```

- **Math.floor()**

```
Math.floor(4.7); // returns 4
```

# MATH OBJECT

- **Math.min() and Math.max()**

`Math.min(0, 150, 30, 20, -8, -200); // returns -200`

`Math.max(0, 150, 30, 20, -8, -200); // returns 150`

- **Math.random()**

`Math.random(); // returns a random number`

**Math.random() returns a random number  
between 0 and 1**

# DATE Object

- <script>

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getDay();
</script>
```

# DATE Methods

<b>getDate()</b>	Get the day as a number (1-31)
<b>getDay()</b>	Get the weekday as a number (0-6)
<b>getFullYear()</b>	Get the four digit year (yyyy)
<b>getHours()</b>	Get the hour (0-23)
<b>getMilliseconds()</b>	Get the milliseconds (0-999)
<b>getMinutes()</b>	Get the minutes (0-59)
<b>getMonth()</b>	Get the month (0-11)
<b>getSeconds()</b>	Get the seconds (0-59)
<b>getTime()</b>	Get the time (milliseconds since January 1, 1970)

# DHTML

## Dynamic HTML

Document with  
HTML,CSS,JAVASCRIPT

# DHTML

- DHTML stands for **Dynamic HTML**.
- DHTML is NOT a language or a web standard.
- DHTML is a TERM used to describe the technologies used to make web pages dynamic and interactive.
- DHTML means the combination of HTML, JavaScript, DOM, and CSS.
- According to the World Wide Web Consortium (W3C):  
*"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."*

Thank you So much

# Chapter 6

**XML**  
**Extensible Mark-up**  
**Language**

# Introduction of XML

- The primary use of XML is description of data rather than presentation.
- It is used to transfer the information from one system to another dissimilar system.
- XML does not do anything itself, it just describe data.
- In 1998 XML 1.0 became a W3C recommendation.
- The main purpose is to sharing data acrosss the different systems and it is useful in application over the internet.

# Features of XML

1. It is in a format that both humans and machine can read. There is nothing special about XML,it is just plain text of normal characters and angled brackets.
2. It supports Unicode, which is capable of encoding all human languages.
3. It supports data structures well.
4. It is self-documenting.

# Features of XML

5. It has strict format that makes it easy for parsing to take place.
6. It can be understood and exchanged between dissimilar system.
7. It can be useful in swapping data between different application.

# Sample XML

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
 <first>HIREN</first>
 <last>MER</last>
 <birthdate>16th January 1987</birthdate>
 <birthplace>Rajkot</birthplace>
</person>
```

# XML Key Components

- Elements
- Attributes
- Namespace
- Other essentials

# Elements

- The strict syntax of XML contains few rules.
  1. Elements must have closing tag.
  2. Tags are case sensitive.
  3. Elements must have nested correctly.
  4. XML document must have root element.

# Attributes

- Attribute can be added to the XML tag.
- It should be in double quotes.
- For example

<emp start=19/12/16>Indus</emp>

This is wrong

<emp start="19/12/16">Indus</emp>

This is right

# Other Essentials

- There are few other important aspects to keep in mind when working with XML.
- White space is preserved in XML. Space should be kept as they are.
- Comment is common as HTML in triangular bracket.
- `<!-- Comments -->`

# Namespace

- Sometimes in XML there is a conflicting names between documents.
- E.g two different documents contain same named element.if you are sharing with someone else,it may conflict with another document.
- It is possible to use associate namespace with elements in use,providing unique name.

# Namespace

- Name space is usually take form of URL, beginning with domain name, optional namespaces label in form of a directory name and finally a version number which is optional:

`xmlns="http://www.myomain.com/ns/animal  
s/1.1"`

# DTD

## Document Type Definition

- A **document type definition (DTD)** is a set of *markup declarations* that define a *document type* for an SGML-family markup language (SGML, XML, HTML).
- E.g  

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

# Internal DTD

- <?xml version="1.0"?>  
<!DOCTYPE note [  
  <!ELEMENT note (to,from,heading,body)>  
  <!ELEMENT to (#PCDATA)>  
  <!ELEMENT from (#PCDATA)>  
  <!ELEMENT heading (#PCDATA)>  
  <!ELEMENT body (#PCDATA)>  
]>  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend</body>  
</note>

# External DTD

# XML file

- <?xml version="1.0"?>  
<!DOCTYPE note SYSTEM "note.dtd">  
<note>  
    <to>Tove</to>  
    <from>Jani</from>  
    <heading>Reminder</heading>  
    <body>Don't forget me this  
weekend!</body>  
  </note>

# DTD file for Previous XML

- <!ELEMENT note (to,from,heading,body)>  
    <!ELEMENT to (#PCDATA)>  
    <!ELEMENT from (#PCDATA)>  
    <!ELEMENT heading (#PCDATA)>  
    <!ELEMENT body (#PCDATA)>

# Declaring Elements

- `<!ELEMENT element-name category>`  
or  
`<!ELEMENT element-name (element-content)>`

# Empty Element

- `<!ELEMENT element-name EMPTY>`
- Example:

`<!ELEMENT br EMPTY>`

XML example:

`<br />`

# Elements with Parsed Character Data

- Elements with only parsed character data are declared with #PCDATA inside parentheses:
- `<!ELEMENT element-name (#PCDATA)>`

Example:

```
<!ELEMENT from (#PCDATA)>
```

# Elements with Children (sequences)

- `<!ELEMENT element-name (child1)>`  
or  
`<!ELEMENT element-name (child1,child2,...)>`

Example:

```
<!ELEMENT note (to,from,heading,body)>
```

# Declaring Only One Occurrence of an Element

- <!ELEMENT element-name (child-name)>

Example:

```
<!ELEMENT note (message)>
```

# Declaring Minimum One Occurrence of an Element

- `<!ELEMENT element-name (child-name+)>`

Example:

```
<!ELEMENT note (message+)>
```

# Declaring Zero or More Occurrences of an Element

- `<!ELEMENT element-name (child-name*)>`

Example:

```
<!ELEMENT note (message*)>
```

# Declaring Zero or One Occurrences of an Element

- `<!ELEMENT element-name (child-name?)>`

Example:

```
<!ELEMENT note (message?)>
```

# Declaring either/or Content

- `<!ELEMENT note  
(to,from,header,(message|body))>`
- The example above declares that the "note" element must contain a "to" element, a "from" element, a "header" element, and either a "message" or a "body" element.

# PCDATA

- PCDATA means parsed character data.
- Think of character data as the text found between the start tag and the end tag of an XML element.
- **PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.**
- Tags inside the text will be treated as markup and entities will be expanded.
- However, parsed character data should not contain any &, <, or > characters; these need to be represented by the &amp; &lt; and &gt; entities, respectively.

# Declaring Mixed Content

- `<!ELEMENT note (#PCDATA|to|from|header|message)*>`

```
<note>
Hello<to>aa</to>
How are you?
</note>
```

# PCDATA

- **PCDATA – parsed character data.** It parse to all the data in an xml document.
- Example:

```
<family> <mother>mom</mother>
<father>dad</father> </family>
```

Here, the family element contains 2 more elements “**mother**” and “**father**”. So it parse further to get the text of mother and father to give the value of family.

# CDATA

- **CDATA – unparsed characted Data.**
- CDATA - Used for Attributes of XML i.e ATTLIST
- CDATA contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

# ***ATTRLIST Declaration***

- <!ATTRLIST element\_name attribute\_name  
attribute\_type default\_value> . . .
- <element attribute\_name="attribute\_value">

# Basic Example

```
<?xml version="1.0"?>
<!DOCTYPE image [
<!ELEMENT image EMPTY>
<!ATTLIST image height CDATA #REQUIRED>
<!ATTLIST image width CDATA #REQUIRED>
]>
<image height="32" width="32"/>
```

# ID Example

```
<?xml version="1.0"?>
<!DOCTYPE student_name [
<!ELEMENT student_name (#PCDATA)>
<!ATTLIST student_name student_no ID #REQUIRED>
]>
<student_name student_no="a9216735">Jo
Smith</student_name>
```

# IDREF Example

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE lab_group [
<!ELEMENT lab_group (student_name)*>
<!ELEMENT student_name (#PCDATA)>
<!ATTLIST student_name student_no ID #REQUIRED>
<!ATTLIST student_name tutor_1 IDREF #IMPLIED>
<!ATTLIST student_name tutor_2 IDREF #IMPLIED>]>
<lab_group>
<student_name student_no="a890">AAA</student_name>
<student_name student_no="a901">SSS</student_name>
<student_name student_no="a9216735" tutor_1="a901"
 tutor_2="a890">TTT</student_name>
</lab_group>
```

# ENTITY Example:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE experiment_a [
 <!ELEMENT experiment_a (results)*>
 <!ELEMENT results EMPTY>
 <!ATTLIST results image ENTITY #REQUIRED>
 <!ENTITY a SYSTEM
 "http://www.university.com/results/experiment/a.gif">]>
<experiment_a><results image="a"/>
<experiment_a>
```

# ENTITIES Example:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE experiment_a [
<!ELEMENT experiment_a (results)*>
<!ELEMENT results EMPTY>
<!ATTLIST results images ENTITIES #REQUIRED>
<!ENTITY a1 SYSTEM
 "http://www.university.com/results/experimenta/a1.gif">
<!ENTITY a2 SYSTEM
 "http://www.university.com/results/experimenta/a2.gif">
<!ENTITY a3 SYSTEM
 "http://www.university.com/results/experimenta/a3.gif">
]>

<experiment_a> <results images="a1 a2 a3"/> </experiment_a>
```

# NMTOKEN Example:

```
<?xml version="1.0"?>
<!DOCTYPE student_name [
 <!ELEMENT student_name (#PCDATA)>
 <!ATTLIST student_name student_no
NMTOKEN #REQUIRED>]>
<student_name student_no="9216735">
 Jo Smith
</student_name>
```

④ **Tokenized Attribute Type:**

### **Attribute Description:**

<b>ID</b>	<p><code>ID</code> is a unique identifier of the attribute. <code>IDs</code> of a particular value should not appear more than once in an XML document. An element type may only have one <code>ID</code> attribute.</p> <p>An <code>ID</code> attribute can only have an <code>#IMPLIED</code> or <code>#REQUIRED</code> <a href="#">default value</a>. The first character of an <code>ID</code> value must be a letter, <code>'_</code>, or <code>':'</code>.</p>
<b>IDREF</b>	<p><code>IDREF</code> is used to establish connections between elements. The <code>IDREF</code> value of the attribute must refer to an <code>ID</code> value declared elsewhere in the document. The first character of an <code>ID</code> value must be a letter, <code>'_</code>, or <code>':'</code>.</p>
<b>IDREFS</b>	<p>Allows multiple <code>ID</code> values separated by whitespace.</p>
<b>ENTITY</b>	<p><code>ENTITY</code>s are used to reference data that act as an abbreviation or can be found at an external location. The first character of an <code>ENTITY</code> value must be a letter, <code>'_</code>, or <code>':'</code>.</p>
<b>ENTITIES</b>	<p>Allows multiple <code>ENTITY</code> names separated by whitespace.</p>
<b>NMTOKEN</b>	<p>The first character of an <code>NMTOKEN</code> value must be a letter, digit, <code>'.'</code>, <code>'-'</code>, <code>'_</code>, or <code>':'</code>.</p>
<b>NMTOKENS</b>	<p>Allows multiple <code>NMTOKEN</code> names separated by whitespace.</p>

# XML Schema

- An XML Schema describes the structure of an XML document, just like a DTD.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".

# XML Schema Example

```
<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>
```

# Explanation

- The Schema is interpreted like this:
- `<xs:element name="note">` defines the element called "note"
- `<xs:complexType>` the "note" element is a complex type
- `<xs:sequence>` the complex type is a sequence of elements
- `<xs:element name="to" type="xs:string">` the element "to" is of type string (text)
- `<xs:element name="from" type="xs:string">` the element "from" is of type string
- `<xs:element name="heading" type="xs:string">` the element "heading" is of type string
- `<xs:element name="body" type="xs:string">` the element "body" is of type string

# What is a Simple Element?

- A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

# Defining a Simple Element

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

- where xxx is the name of the element and yyy is the data type of the element.
- XML Schema has a lot of built-in data types. The most common types are:
  1. xs:string
  2. xs:decimal
  3. xs:integer
  4. xs:boolean
  5. xs:date
  6. xs:time

# Example

Xml:

```
<?xml version="1.0"?>
<lastname>aaa</lastname>
<age>36</age>
<birthdate>1980-04-27</birthdate>
```

Schema:

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

# XSD SCHEMA

- The <schema> element may contain some attributes. A schema declaration often looks something like this:
- ```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">
...
</xs:schema>
```

- [xmlns:xs=http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema)
- indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace. It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with **xs:**

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

- indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "https://www.w3schools.com" namespace.
- **Each one has own library for the namespace like <http://www.myown.com>**

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

- indicates that the default namespace is "`https://www.w3schools.com`".

`elementFormDefault="qualified"`

- indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.
- **qualified** - elements and attributes are in the targetNamespace of the schema
- **unqualified** - elements and attributes do not have a namespace

XSD Attributes

- What is an Attribute?

Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.

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

- where xxx is the name of the attribute and yyy specifies the data type of the attribute.
- XML Schema has a lot of built-in data types. The most common types are:
 1. xs:string
 2. xs:decimal
 3. xs:integer
 4. xs:boolean
 5. xs:date
 6. xs:time

Example

Xml

```
<name lang="EN">Smith</name>
<city lang="EN">rajkot</city>
```

Xsd

```
<xs:attribute name="lang" type="xs:string"/>
```

Default and Fixed Values for Attributes

- Attributes may have a default value OR a fixed value specified.
- A default value is automatically assigned to the attribute when no other value is specified.
- In the following example the default value is "EN":

```
<xs:attribute name="lang" type="xs:string"  
default="EN"/>
```

Cont..

- A fixed value is also automatically assigned to the attribute, and you cannot specify another value.
- In the following example the fixed value is "EN":

```
<xs:attribute name="lang" type="xs:string"  
fixed="EN"/>
```

Optional and Required Attributes

- Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:

```
<xs:attribute name="lang" type="xs:string"  
use="required"/>
```

XSD Restrictions/Facets

- Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.

Chapter 7

PHP

Hypertext Pre-processor

Introduction of PHP

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- Syntax:

**<?php
 //php code here**

?>

Introduction of PHP

- **WAMP**

Windows Apache MySQL and PHP

XAMPP

X(cross Platform) Apache Mysql PHP Perl

- **LAMP**

Linux Apache MySQL and PHP

Prepared by Hiren V Mer

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is PHP?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content.
- PHP can create, open, read, write, delete, and close files on the server.
- PHP can collect form data.
- PHP can send and receive cookies.
- PHP can add, delete, modify data in your database.
- PHP can be used to control user-access.
- PHP can encrypt data

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.

Beyond Syllabus

| List of PHP framework | |
|-----------------------|----------|
| Laravel | Yii |
| Symfony | Fatfree |
| CodeIgniter | Kohana |
| CakePHP | FuelPHP |
| Zend | Flight |
| Phalcon | PHP-mini |
| Slim | Tyler |
| | Zikula |

Why Framework?

- Framework abstracts you from low level details, makes you more productive, and protects you from low level error (such as preventing SQL injection attacks).
- A framework also **adds structure to the code**, prompting the developer to write **better, more readable, and more maintainable code**. Ultimately, a **framework makes programming easier**, since it packages complex operations into simple statements.

Definition of Framework

- In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate.

Laravel

- Laravel. Although Laravel is a relatively new PHP framework (it was released in 2011), according to Sitepoint's recent online survey it is the most popular framework among developers. ...
- Latest Version: [Laravel 5.4](#)
- Laravel 4.2 requires PHP 5.4 or greater.



Basic Example of PHP

```
<html>
<head></head>
<body>
<?php
    echo "Hello I am hiren mer";
?
</body>
</html>
```

PHP Comments

```
<?php
```

```
// This is a single-line comment
```

```
# This is also a single-line comment
```

```
/*
```

```
This is a multiple-lines comment block  
that lines
```

```
*/
```

```
?>
```

PHP Variables

- In PHP variable name is case sensitive.
 - Variable name should be start with \$ symbol.
 - Variable name datatype declaration is as same as Javascript.
-
- `$a=10; // Number`
 - `$b=10.5; // Number`
 - `$c="hiren"; // String`

Global Variable

- The global keyword is used to access a global variable from within a function.

```
<?php
```

```
$x = 5;  
$y = 10;  
function myTest()  
{  
    global $x, $y;  
    $y = $x + $y;  
    echo $y;  
}
```

```
?>
```

Static Variable

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- ```
<?php
function myTest() {
 static $x = 0;
 echo $x;
 $x++;
}
myTest();
myTest();
?>
```

# PHP echo and Print

- In PHP there are two basic ways to get output: **echo and print**.
- echo has no return value while print has a return value of 1 so it can be used in expressions.
- echo can take multiple parameters (although such usage is rare) while print can take one argument.
- echo is marginally faster than print.

# The PHP echo Statement

- The echo statement can be used with or without parentheses: echo or echo().

```
<?php
$a=2;
echo "This ", "string ", "was ", "made ", "with
multiple parameters.;"
echo $a;
?>
```

# The PHP print Statement

- The print statement can be used with or without parentheses: print or print().

```
<?php
print "<h1>Hello</h1>";
print "Hello world!
";
print "I'm about to learn PHP!";
?>
```

# The PHP print\_r() Statement

```
<?php $var = array(1,2,3,4);
 print_r($var); ?>
```

Output:

```
Array ([0] => 1
 [1] => 2
 [2] => 3
 [3] => 4)
```

```
<?php
$var1='abc';
$result = print_r($var1);
echo $result.'
';
$abc = array('a'=>'a','b'=>'b','c'=>'c','d'=>array(5,6,7,8));
$result = print_r($abc);
echo $result.'
';
?>
```

## Output

```
abc
Array ([a] => a [b] => b [c] =>c [] => Array ([0] => 5 [1]
=> 6 [2] => 7 [3] => 8))
```

# PHP Data Types

- PHP supports the following data types:
  1. String
  2. Integer
  3. Float (floating point numbers - also called double)
  4. Boolean
  5. Array
  6. Object
  7. NULL

# PHP String

- A string is a sequence of characters, like "Hello world!".

```
<?php
$x = "Hello world!";
echo $x;
```

```
?>
```

# PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
  1. An integer must have at least one digit
  2. An integer must not have a decimal point
  3. An integer can be either positive or negative
  4. Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

# PHP Integer

- <?php  
\$x = 5985;  
var\_dump(\$x);  
?>

# PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.

```
<?php
$x = 10.365;
var_dump($x);
?>
```

# PHP Boolean

- A Boolean represents two possible states: TRUE or FALSE.

```
<?php
$x = TRUE;
var_dump($x);
?>
```

# PHP Array

- An array stores multiple values in one single variable.

```
<?php
$c = array("Volvo","BMW","Toyota");
var_dump($c);
?>
```

# PHP Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

# PHP Object

- <?php  
class f  
{  
    function do\_funny()  
    {  
        echo "Doing funny.";  
    }  
}

```
$b = new f;
$b->do_funny();
?>
```

# PHP NULL Value

- Null is a special data type which can have only one value: NULL.

```
<?php
$x = null;
var_dump($x);
?>
```

# PHP if...else...elseif Statements

- In PHP we have the following conditional statements:
  1. **if statement** - executes some code if one condition is true
  2. **if...else statement** - executes some code if a condition is true and another code if that condition is false
  3. **if...elseif....else statement** - executes different codes for more than two conditions
  4. **switch statement** - selects one of many blocks of code to be executed

# PHP - The if Statement

- Syntax

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

## Example

```
<?php
$t = 20;
```

```
if ($t < 20) {
 echo "Have a good day!";
}
~
```

# PHP - The if...else Statement

- Syntax

```
if (condition)
```

```
{
```

*code to be executed if condition is true;*

```
}
```

```
else {
```

*code to be executed if condition is false;*

```
}
```

# Example

- <?php  
  \$t = 25;  
  
  if (\$t < 20) {  
    echo "it's less than 20";  
  } else {  
    echo " it's greater than 20 ";  
  }  
?>

# PHP - The if...elseif....else Statement

- Syntax

```
if (condition)
{
 statement-1;
}

elseif (condition) {
 statement-2;
}

else {
 statement-3;
}
```

# Example

- <?php  
  \$t = 30;  
  if (\$t < 10) {  
    echo "Have a good morning!";  
  } elseif (\$t < 20) {  
    echo "Have a good day!";  
  } else {  
    echo "Have a good night!";  
  }  
?>

# PHP switch Statement

Syntax:

```
switch (n) {
 case label1:
 code to be executed if n=label1;
 break;
 case label2:
 code to be executed if n=label2;
 break;
 case label3:
 code to be executed if n=label3;
 break;
 ...
 default:
 code to be executed if n is different from all labels;
}
```

- <?php  
    \$favcolor = "red";

# Example

```
switch ($favcolor) {
 case "red":
 echo "Your favorite color is red!";
 break;
 case "blue":
 echo "Your favorite color is blue!";
 break;
 case "green":
 echo "Your favorite color is green!";
 break;
 default:
 echo "Your favorite color is neither red, blue, nor green!";
 break; }
?
Prepared by Hiren V Mer
```

# PHP Loops

- In PHP, we have the following looping statements:
  1. **while** - loops through a block of code as long as the specified condition is true
  2. **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
  3. **for** - loops through a block of code a specified number of times
  4. **foreach** - loops through a block of code for each element in an array

# The PHP while Loop

## Syntax

`while (condition is true)`

`{`

*code to be executed;*

`}`

# Example

```
<?php
$x = 1;

while($x <= 5) {
 echo "The number is: $x
";
 $x++;
}
?>
```

# The PHP do...while Loop

- Syntax

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

# Example

```
<?php
$x = 1;

do {
 echo "The number is: $x
";
 $x++;
} while ($x <= 5);

?>
```

# The PHP for Loop

- Syntax

`for (initialization; test condition; incr/decr)`

`{`

`code to be executed;`

`}`

# Example

```
<?php
 for ($x = 0; $x <= 10; $x++)
 {
 echo "The number is: $x
";
 }
?>
```

# The PHP foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- Syntax

```
foreach ($array as $value)
{
 code to be executed;
}
```

# Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value)
{
 echo "$value
";
}

?>
```

# PHP Functions

- Here, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

# PHP Functions

- Syntax

```
function functionName()
{
 code to be executed;
}
```

# Example

Example:

```
<?php
 function writeMsg() {
 echo "Hello world!";
 }

 writeMsg(); // call the function
?>
```

# PHP Function Arguments

```
<?php
function name($fname)
{
 echo "Hello $fname
";
}
name("hiren");
name("hemansir");
?>
```

# PHP Functions - Returning values

```
<?php
 function sum($x, $y)
 {
 $z = $x + $y;
 return $z;
 }
 echo sum(5, 10) ;
?
>
```

# PHP Arrays

- In PHP, there are three types of arrays:
  1. **Indexed arrays** - Arrays with a numeric index
  2. **Associative arrays** - Arrays with named keys
  3. **Multidimensional arrays** - Arrays containing one or more arrays

# PHP Indexed Arrays

- The index can be assigned automatically (index always starts at 0), like this:

```
$name = array("hiren", "rahul", "heman");
```

- And the index can be assigned manually:

```
$name[0] = "2nd sem";
```

```
$name[1] = "4th sem ";
```

```
$name[2] = " 6th sem ";
```

# Loop with an Indexed Array

- <?php  
\$x = array("a", "b", "c");  
\$len = count(\$x);  
  
for(\$i = 0; \$i < \$len; \$i++) {  
 echo \$x[\$i];  
 echo "<br>";  
}  
?>

# PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.

```
$age = array("a"=>"30", "b"=>"40", "c"=>"50");
```

or:

```
$age['a'] = "30";
$age['b'] = "40";
$age['c'] = "50";
```

# Example

- <?php  
\$age  
= array("a"=>"35", "b"=>"37", "c"=>"43");  
  
echo "a is " . \$age['a'] . " years old.";  
?>

# Loop with an Associative Array

- <?php  
\$age  
= array("a"=>"35", "b"=>"37", "c"=>"43");

```
foreach($age as $x => $value)
{
 echo "Key=" . $x . ", Value=" . $value;
 echo "
";
}
?>
```

# Multidimensional Array

- ```
$c = array (
    array("a",22),
    array("b",24)
);
```
- ```
<?php
echo "name". $c[0][0]. "age". $c[0][1]. "
";
echo "name". $c[1][0]. "age". $c[1][1]. "
";
?>
```

# Browser Control

- PHP can control various features of browser. This is important as need to reload and redirect the page.

```
header("Location:pagename")
```

# Browser Control

- It is also possible to reload current page.

```
$x=$_SERVER['PHP_SELF'];
<form action="<?php echo $x?>"
method="get">

</form>
```

# PHP STRING Function

## 1. **strlen(\$var):**

this function returns length of string.

Example:

```
<?php
echo strlen("Hello");
?>
```

Output:

5

## **2. str\_word\_count()**

This function returns the word count of the string.

Example:

```
<?php
echo str_word_count("I am hiren");
?>
```

## **3.strrev(\$var)**

This function reverse given string.

Example:

```
echo strrev("hirenmer");
```

## 4.strpos(\$var,\$search)

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example:

```
strpos("hiren mer","mer");
```

Output is 6. because string start with 0.

## **5.str\_replace(\$s1,\$s2,\$s3)**

This function replace the string in given string. \$s1 to be replace in given string and \$s2 replace in that string \$s3.

**Example:**

```
str_replace("hi", "good morning all", "hello hi");
```

In this example “hello hi” is given string and hi to be replaced with good. So output of this function is “hello good morning all” .

## **6.str\_pad(*string,length,pad\_string,pad\_type*)**

The str\_pad() function pads a string to a new length.

Example:

```
<?php
 $str = "Hello World";
 echo str_pad($str,20,".");

?>
```

In above example hello world length is 12 and remaining 8 dots are padding by default right.

**Output: Hello World.....**

- Possible values:
- `STR_PAD_BOTH` - Pad to both sides of the string. If not an even number, the right side gets the extra padding
- `STR_PAD_LEFT` - Pad to the left side of the string
- `STR_PAD_RIGHT` - Pad to the right side of the string. This is default

## Example

```
<?php
 $str = "Hello";
 echo str_pad($str,10,"$", STR_PAD_LEFT);
?>
```

## **7. str\_shuffle(\$var)**

This function shuffle the position of the each and every character.

Example: `str_shuffle("hirenmer");`

Output will be : rimherne

## **8. str\_split(\$str,length)**

- The `str_split()` function splits a string into an array.
- Example : `str_split("hello",2)`

Output

Array ( [0] => He [1] => ll [2] => o )

## **9. strchr(\$str,\$match)**

This function find the first occurrence of string inside given string and return the rest of the string.

Example:

```
echo strchr("hiren mer!","re");
```

Output:

```
ren mer!
```

## **10. strcmp(*string1*,*string2*)**

This function match string 1 and string2.

- Example

```
echo strcmp("Hello!","Hello world!");
```

Output : -6

- Return Value: This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

## 11. **strtolower(\$str)**

This function convert uppercase to lower case.

**Example:** echo strtolower("HIREN");

**Output:** hiren

## **12. strtoupper(\$str)**

The strtoupper() function converts a string to uppercase.

**Example :** Example: echo strtoupper("hiren");

**Output:** HIREN

## **13. substr(\$var,value)**

**This function returns the string from value index.**

**Example:**

**echo substr("hirenmer",3);**

**Output:**

**Enmer**

**Substr(\$var,start,end)**

When we want substring start from and to then  
we can use above function.

**Example:**

**echo substr("hirenmer",3,3);**

**Output:**

**enm**

**14. stripslashes(*string*)**

This function strip the slashes \ in the string  
which we have used to escape symbol.

- Example

```
echo stripslashes("Who\'s katappa?");
```

Output:

Who's katappa?

### 15. strstr(\$s1,\$s2)

This function search \$s2 in \$s1 and return the portion of \$s1. This function is same as **strchr(\$str,\$match)**.

**Example:**

```
strstr("hirenmer123","nm")
```

**Output:** nmer123

## 16. strtok(\$s1,\$d)

This function splits a string (str) into smaller strings (tokens), with each token being delimited by any character from token.

```
<?php
$string = "Hello world. Beautiful day today.";
$token = strtok($string, ".");
while ($token !== false)
{
 echo "$token
";
 $token = strtok(".");
}
?>
```

## 17. trim(\$s1,\$s2)

This function Remove characters/space from both sides of a string .

```
<?php
$str = “6th BYBX”;
echo trim($str,“6”);
?>
```

## 18.ltrim(\$str,\$s1)

This function Remove characters/space from left side of a string .

## 19.rtrim(\$str,\$s1)

This function Remove characters/space from right side of a string .

## **20.ucwords(\$str)**

This function retruns value with first letter Capital in each word.

```
ucwords("salman khan");
```

Output: *Salman Khan*

## **21. strip\_tags(\$str,"allowable tag")**

Example:

```
$str=<i>hiren mer</i>;
```

```
strip_tags($str,"<i>");
```

Output: *hiren mer*

# PHP FILES

- Read a file using PHP:

**readfile(filename)**

This function show the file in browser with number of characters in file.

- Example

```
echo readfile("myfile.txt");
```

- Open read and close a file using PHP:

**`$var=fopen(filename,mode)`**

This function open the file name in different mode.

**`fread($var,filesize(filename))`**

This function read the file content as per the second parameter.

**`fclose($var)`**

This function close the file which already opened by fopen.

- Example:

```
<?php
$myfile = fopen("myfile.txt", "r") or die("Unable
to open file!");
echo fread($myfile,filesize("myfile.txt"));
fclose($myfile);
?>
```

Mode	Use
r	Read mode
w	Write mode create new file
a	Append mode and also create new file
x	<b>Creates a new file for write only.</b> Returns FALSE and an error if file already exists

## **fgets(\$ptr)**

- This function read the first line of the file.
- Example

```
<?php
$myfile = fopen("abc.txt", "r") or die("Unable to
open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

## **feof()**

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.

- Example

```
<?php
$myfile = fopen("abc.txt", "r") or die("Unable
to open file!");
while(!feof($myfile))
{
 echo fgets($myfile) . "
";
}
fclose($myfile);
?>
```

## fgetc()

- The fgetc() function is used to read a single character from a file.
- Example

```
<?php
$myfile = fopen("myfile.txt", "r") or die("Unable to
open file!");

while(!feof($myfile))
{
 echo fgetc($myfile);
}

fclose($myfile);
?>
```

## **fwrite()**

- The fwrite() function is used to write to a file.

### Example

```
<?php
$myfile = fopen("myfile.txt", "w") or die("Unable to open
file!");
$txt = "hello 6th semester\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

# File Upload

- We can upload the file using php page with html input type file.
- First we have to configure php.ini file for the file upload.
- Open php.ini file and make file upload ON.

# Rule for File upload

- Some rules to follow for the HTML form above:
  1. Make sure that the form uses method="post"
  2. The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form.
  3. Without the requirements above, the file upload will not work.

# File Contents

- `$_FILES['file']['name']` this returns the name of the uploaded file.
- `$_FILES['file']['type']` this returns the MIME type of the uploaded file.
- `$_FILES['file']['size']` this returns the size of the uploaded file in Byte.
- `$_FILES['file']['tmp_name']` returns the temporary name for the uploaded file.
- `$_FILES['userfile']['error']` The error code associated with this file upload.

# To upload file on server

- First we have to make one folder upload into our directory.
- Then store the path into variable.
- Then apply the function below to move the file.

**move\_uploaded\_file(\$tmp\_name,path)**

# To upload file on server

- `$_FILES['file']['tmp_name']`; will contain the temporary file name of the file on the server. This is just a placeholder on your server until you process the file.

# PHP Cookies

- A cookie is used to identify the user.
- A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

# Create Cookie

- Syntax

```
setcookie(name, value, expire, path, domain, secure);
```

Only the *name* parameter is required. All other parameters are optional.

## Example

```
Setcookie("user","hiren",time()+3600);
```

Type	Description
Name	Name is used to set the name of user
Value	Value for the cookie value
Expire	Set expire time like <code>time() + 3600</code> is for 1hour
Path	Set the path on server for the cookie storage. If set to '/', the cookie will be available within the entire domain.
Domain	The (sub)domain that the cookie is available to. Setting this to a subdomain (such as ' <code>www.example.com</code> ') will make the cookie available to that subdomain and all other sub-domains of it (i.e. <code>w2.www.example.com</code> ). To make the cookie available to the whole domain (including all subdomains of it), simply set the value to the domain name (' <code>example.com</code> ', in this case).
Secure	Whether cookie use HTTPS connection or not. Return <code>true(1)</code> or <code>false(0)</code> .

# Delete cookie

```
setcookie("user", "", time() - 3600);
```

# Session

- A session is a way to store information by Variable to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

# Start PHP Session

Session\_start()

Above statement is used to start the session.after starting session we can store the value into session variable.

```
$_SESSION['a']=10;
```

# Access Session Variable value at another page

- When we want to access the session variable value at another page, we have to start the session first then we can access the session variable values.

```
Session_start();
```

```
Echo $_SESSION['a'];
```

Output:

10

# Destroy a PHP Session

- If user want to destroy the session then he can use below function,

`session_unset();`

Or

`Session_destroy();`

- Generaly we can use this method in the login and logout pages.

# Sending Email

- PHP must be configured correctly in the **php.ini** file with the details of how your system sends email.
- Windows users should ensure that two directives are supplied. The first is called SMTP that defines your email server address. The second is called `sendmail_from` which defines your own email address.

# How to set php.ini for Email

[mail function]

;For Win32 only.

SMTP = smtp.atmiya.net

;For win32 only

sendmail\_from =hiren@aits.edu.in

# Syntax

**mail( to, subject, message, headers, parameters );**

**To:** Required. Specifies the receiver / receivers of the email.

**Subject :** Required. Specifies the subject of the email.

This parameter can not contain any newline characters

**Message:** Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters

# Conti.

**Headers:** Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)

**Parameters:** Optional. Specifies an additional parameter to the send mail program

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From:
webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

# PHP Timestamp

- We can store the php timestamp using time() function.

Example:

```
$time=time();
echo $time;
```

Output:

1490549125

This output in seconds from 1<sup>st</sup> jan 1970.

# PHP Timestamp

- We can print date using this function.

```
$t=time();
```

```
echo date('h:i:s ',$t); // 10:25:55
```

```
echo date('D M Y',$t); //Sun Mar 2017
```

```
echo date('d m y',$t); // 26 03 17
```

# OPPS IN PHP

- Well see in Lab Session.

# Chapter 7

Advanced PHP  
(OOPS concepts)

# Definitions

- **Class** – This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object** – An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable** – These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.

- **Member function** – These are the functions defined inside a class and are used to access object data.
- **Inheritance** – When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class** – A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class** – A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism** – This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it can take different number of arguments and can do different tasks.

- **Overloading** – a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction** – Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation** – refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Constructor** – refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructor** – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

# OPPS in PHP

- A class implements the concept of ADT(Abstract data type).
- It is a compilation of methods and objects.A class definition is always begin with keyword “class” followed by identifier which represents class name.
- And then it followed by { } curly braces which contains the members of class.

Example:

# OPPS in PHP

```
<?php
```

```
Class abc
```

```
{
```

```
 public $a;
```

```
 function getval()
```

```
{
```

```
 $this->a=10;
```

```
}
```

```
 function showval()
```

```
{
```

```
 echo "you have entered".$this->a;
```

```
}
```

```
}
```

```
$myobj=new abc();
```

```
$myobj->getval();
```

```
$myobj->showval()
```

```
?>
```

# Inheritance

<?php

```
Class aaa
{
 public $x;
 public $y;
 public function const()
 {
 $this->y="hello";
 $this->x=0;
 }
}
```

?>

```
class bbb extends aaa
{
 public $a;
 public function search()
 {
 $this->mode="search";
 }
}
$p=new aaa();
print $p->const();
print $p->search();
```

# Inheritance

- We can use different types of inheritance as same as C++.
- Like
  - Multiple
  - Multilevel

# Chapter 8

PHP & Mysql

# Introduction of MySQL

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL compiles on a number of platforms
- MySQL is free to download and use.
- MySQL is developed, distributed, and supported by Oracle Corporation

# Introduction of Mysql

- PHP combined with MySQL are cross-platform  
(you can develop in Windows and serve on a Unix platform)

# Connection To Server

- We can connect to server using mysql command.

```
mysql_connect("servername","username","password");
```

**MySQLi extension** (the "i" stands for improved)

```
mysqli_connect("servername","username","password", "my_db");
```

# Creating Database

- We can create database after connecting server.
- Example

```
mysql_connect("localhost","root","");
$
```

```
$x="CREATE DATABASE abc";
```

```
mysql_query($x);
```

# Selecting Database

After connecting server and after Creating database using PHP we can select the database using following code:

**mysql\_select\_db("dbname")**

# Listing Database

- We can list the database names using mysql query.

```
<?php
$x=mysql_connect("localhost","root","");
$result=mysql_list_dbs($x);
while ($row = mysql_fetch_array($result)) {
 echo $row[0]."
";
}
?>
```

# Listing Database Example 2

```
<?php
$x=mysql_connect("localhost","root","");
$result = mysql_query("SHOW DATABASES");
while ($row = mysql_fetch_array($result)) {
 echo $row[0]."
";
}
?>
```

# Creating Mysql Table

- The CREATE TABLE statement is used to create a table in MySQL.syntax is same as Oracle DB.

```
CREATE TABLE tablename(
 colname1 datatype(size),
 colname2 datatype(size),
 .
 .
);
```

# Example

```
<?php
mysql_connect("localhost","root","");
mysql_Select_db("dbname");
$q1="create table abc(eno INT(20),
Class varchar(20));";
mysql_query($q);
?>
```

# List tables

```
<?php
$x=mysql_connect("localhost","root","");
$db=mysql_select_db("test");
$result=mysql_query('SHOW TABLES');
while ($row = mysql_fetch_array($result))
{
 echo $row[0]."
";
}
?>
```

# Inserting Data into Mysql Table

syntax rules :

1. The SQL query must be quoted in PHP
2. String values inside the SQL query must be quoted
3. Numeric values must not be quoted
4. The word NULL must not be quoted

# Syntax

```
INSERT INTO table_name (column1,
column2,column3,...) VALUES (value1, value2,
value3,...);
```

- Example:

```
$sql = "INSERT INTO MyGuests (firstname, lastname,
email) VALUES ('MS', 'dhoni', 'abc@example.com')";
Mysql_query($sql)
```

# Altering Data into Table

- Syntax

```
UPDATE table_name SET column1=value,
column2=value2,...
```

```
WHERE some_column=some_value
```

Example:

```
$sql = "UPDATE MyGuests SET lastname='raina'
WHERE id=2";
mysql_query($sql);
```

# Delete Data From a MySQL Table

- Syntax

```
DELETE FROM table_name
WHERE some_column = some_value;
```

Example:

```
$sql = "DELETE FROM MyGuests WHERE id=3";
mysql_query($sql);
```

# PHP Select Data From MySQL

- Syntax

```
SELECT column_name(s) FROM table_name;
```

Example :

```
$q= "SELECT * from abc";
$res=mysql_query($q);
While($x=mysql_fetch_array($res))
{
 echo $x[0]."and ".$x[1]."
";
}
```

# Datatypes in PHP

- Properly defining the fields in a table is important to the overall optimization of your database. You should use only the type and size of field you really need to use; don't define a field as 10 characters wide if you know you're only going to use 2 characters. These types of fields (or columns) are also referred to as data types, after the **type of data** you will be storing in those fields.
- MySQL uses many different data types broken into three categories: numeric, date and time, and string types.

## Numeric Data Types:

- **INT** - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. **You can specify a width of up to 11 digits.**
- **TINYINT** - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. **You can specify a width of up to 4 digits.**
- **SMALLINT** - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. **You can specify a width of up to 5 digits.**
- **MEDIUMINT** - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. **You can specify a width of up to 9 digits.**

- **BIGINT** - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. **You can specify a width of up to 20 digits.**
- **FLOAT(M,D)** - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). **Decimal precision can go to 24 places for a FLOAT.**
- **DOUBLE(M,D)** - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. **Decimal precision can go to 53 places for a DOUBLE.** **REAL is a synonym for DOUBLE.**
- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. **NUMERIC is a synonym for DECIMAL.**

## Date and Time Types:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 ( YYYYMMDDHHMMSS ).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

## **String Types:**

- **CHAR(M)** - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** - A field with a maximum length of 65535 characters. BLOBS are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBS and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

- **TINYBLOB or TINYTEXT** - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

# Thank you

- I wish you all the best for the Exam.