## UNIT – 1 WEBSITE BASICS, HTML5, CSS3, WEB2.0

### Part-A

**1. What is Web2.0?(Nov/Dec 2017)**

- It is not the second version of Web
- 2.0 denotes two-way data traffic on the Web(R/W)
- It is also called Participatory web (or) Read/Write Web
- Earlier data traffic on the web was unidirectional(Read-Only)

**2. Define website.**

- A website is defined as a collection of web pages linked together that has a unique domain name, that can be accessed from anywhere across the globe over internet.
- It is hosted by a web server and viewed by web clients
- It can be developed in HTML, JavaScript, DHTML, PHP, etc.

**3. Define WWW.**

- World Wide Web is defined as a collection of software and corresponding protocols used to access the resources over the internet across the globe.
- It contains huge amount of Docs, images, etc.
- Internet can be accessed through the WWW
- Invented by Tim Berners-Lee in 1994 (W3C) at MIT

**4. Mention the differences between website and web server(Apr/May 2017).**

| Website | Web server |
|---|---|
| It is a collection of web pages | It is a server on which web application is executed |
| It is a software application that has unique domain name | It is a physical entity that has unique IP address |
| It can host many web pages | It can host many websites |
| They communicate with web server | They communicate with other servers such as DB server, File server, etc |
| Web server = HTML&CSS + JS+ DHTML | It receives request and gives corresponding response |
| Ex: https://www.google.co.in | Ex: IIS, Apache |

**5. Define web crawler.**

- A web crawler is defined as the ability of the web to parse a web page into different semantic elements (navigation links, friend links, group links, etc) and extract the social network and other associated data.

**6. What is RIA? What are features of RIA? (Nov/Dec 2016)**

- RIA (Rich Internet Application) is defined as a web application that is designed to give the same features and functions associated with desktop applications.

Features:-

- It can work on the web
- Information in RIA always visible to users thereby reducing unwanted page refreshes and navigations.
- Ability of web to present complex information to the users
- Good user interactivity such as images, graphics, etc.
- It helps users to understand complex business apps

**7. What is a collaboration tool? What are its features?**

- Collaboration tools allow a group of people work together virtually in real-time over the internet.

**Features:-**

| | |
|---|---|
| Easy to use and set up. | Clean interface |
| Secure | Permissions control |
| Ability to upload documents | File storage |
| Scalable | Document locking |

**8. What is URL?**

- Uniform Resource Locator (URL) is defined as an unique address for the file that has to be accessed over the internet
- If we want to access a website, we enter its URL in the address bar of the web browser
- **Syntax**: protocol: //www.exampleDomain.com/path/filename
- **Ex:** https://www.vit.ac.in / home.aspx

**9. What is IP?**

- Internet Protocol (IP) is a network layer protocol which consists of addressing information, that is the fundamental protocol which is being used by data packets over the internet
- Using this protocol, communication between uniquely addressed computers has been made possible.

**10. Tabulate the differences between TCP and UDP**

| TCP | UDP |
|---|---|
| Connection oriented (link between the packets) | Connection less |
| ACK is available | No ACK |
| Reliable | Unreliable |
| Heavy weight protocol | Light weight protocol |
| Handshaking mechanism | No handshaking concept |
| Error control, flow control, congestion control, etc | No control mechanism |
| Complex, tough to implement | Simple, easy to implement |
| Ex: Telnet, SMTP, FTP, e-mail, SSH, HTTP, HTTP, HTTPS | Ex: VoIP, DHCP, DNS, RIP, SNMP |

**11. What is HTTP?**

- Hyper Text Transfer protocol (HTTP) is a request/Response, stateless protocol for communication, to transfer information on LAN and WWW
- It is used to deliver files virtually and other data on WWW
- It takes place through TCP/IP sockets
- A browser is a HTTP client – sends HTTP request
- A web server is a HTTP server – sends HTTP reply
- It uses port no: 80 (HTTP servers listen to this port)

**12. What are the protocols used in email?**

| SMTP | POP | IMAP |
|---|---|---|
| • Connection-oriented<br>• Text-based<br>• Works in application layer<br>• ACK is available<br>• It uses port 25 | • It uses port 110<br>• Current version:POP3<br>• Single client<br>• Offline email access<br>• Can't search email<br>• Download is needed<br>• Only 1 mailbox<br>• Less internet usage | • Manipulate email<br>• No downloading<br>• Can't transfer email<br>• Access the received emails<br>• Search the mails<br>• Many mailboxes<br>• More internet usage |

**13. What are the differences between internet and intranet?**

| INTERNET | INTRANET **(Nov'15)** |
|---|---|
| Network of networks, open for all | Network of computers, for closed group |
| Limited no. of users | Unlimited no. of users |
| Different sources of info | Limited sources of info |
| Large no. of intranets | Less number of systems |
| Internet =LAN + WAN + MAN | Intranet = LAN \| WAN \| MAN |

**14. What are the flavours of HTML? (types of HTML DTD)**

- **XHTML 1.0 Strict** : When we want a clean mark-up code
- **XHTML 1.0 Transitional:** To use HTML features
- **XHTML 1.0 Frameset:** To make use of frames

**15. What is XHTML? (Nov/Dec 2017)**

- Extensible HTML is the extended version of HTML that has strict rules when compared to HTML
- It is more consistent, well-structured document
- Web pages made in XHTML can be easily understood by the present and future web browsers

**16. What is the use of forms in HTML?**

- HTML form element is used to allow a user to give input data on the web page.
- To create registration forms, login forms, getting user info, conducting surveys
- <form>……..</form> tags are used
- Attributes used: action, method

**Ex:**

<form **action** = http://www.google.co.in/ **method** = post>
</form>

**17. What is the use of frames in HTML?**

- It allows the web developers to present the web document in multiple views
- Using this, within a same window, one can keep some information visible, other part of web page to contain some other information, other part of web page can be reloaded.
- Ex: one frame can display company info, second frame can display navigation menu, etc.

**18. Why HTTP is stateless protocol?**

- HTTP cannot remember previous user information
- It does not recall the number of page visits
- It means it cannot remember its previous states.
- That is the reason why HTTP is stateless protocol.

**19. Mention some of the protocols that are used in internet.**

- FTP, HTTP, SNMP, SMTP, POP3, IMAP, TCP, UDP, IP

**20. What are HTML tags? Give examples.**

- An HTML tag is defined as a command that tells the web browser, how to display the text, audio, video, and graphics on a web page when loaded.
- They are mentioned in a pair of angular brackets < >
- Ex:

| | |
|---|---|
| • <html>..</html><br>• <head>…</head><br>• <title>…</title><br>• <body>…</body> | <h1>…</h1><br><br><br> … |

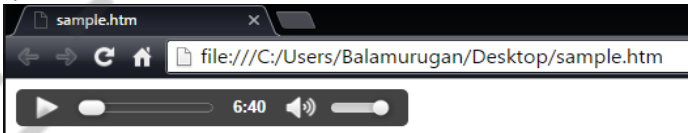
**21. Write a HTML5 code to display:**

| | |
|---|---|
| A | B |
| C | D |

```
< ! DOCTYPE HTML>
<HTML>
        <HEAD></HEAD>
        <BODY>
                <TABLE BORDER=”4”>
                        <TR>
                                <TD>A</TD>
                                <TD>B</TD>
                        </TR>
                        <TR>
                                <TD>C</TD>
                                <TD>D</TD>
                        </TR>
                </TABLE>
        </BODY>
</HTML>
```

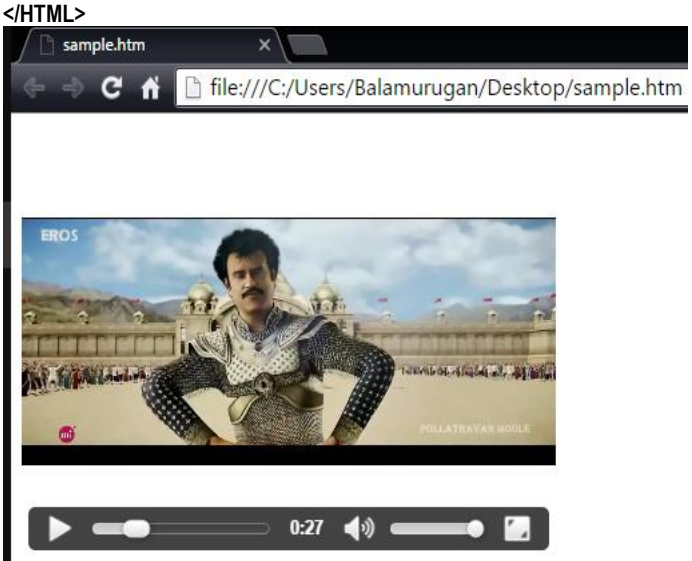
**22. Write HTML5 code to play an audio file.**

```
<!DOCTYPE HTML>
<HTML>
        <HEAD></HEAD>
        <BODY>
                <AUDIO CONTROLS>
                   <SOURCE SRC="KABALI.MP3"
TYPE="AUDIO/MPEG">
                </AUDIO>
        </BODY>
</HTML>
```




**23. Write an HTML5 code to display a video file.**

```
<!DOCTYPE HTML>
<HTML>
        <HEAD></HEAD>
        <BODY>
        <VIDEO WIDTH = "400" HEIGHT = "300" CONTROLS>
           <SOURCE SRC="kochadaiyaan.mp4" TYPE="VIDEO/MP4">
        </VIDEO>
        </BODY>
</HTML>
```



**24. What is CSS? What are its types? (Apr/May 2019)**

- Cascading style sheet is defined as a style sheet in which, all the style information of a web page can be defined.
- It separates the contents and the decoration of a HTML page
- It helps developers to give consistent appearance to all the elements in the web page.

Types:-

- Inline style sheets **<p style="color:green; font-size:15px">**
- Embedded style sheets **<style>……</style>**
- External style sheet **Stored in a separate file (ex.css)**
- Imported style sheets **@import URL(path)**

**25. What are the types of positioning in CSS?**

- Relative positioning
- Absolute positioning
- Float positioning

## PART - B

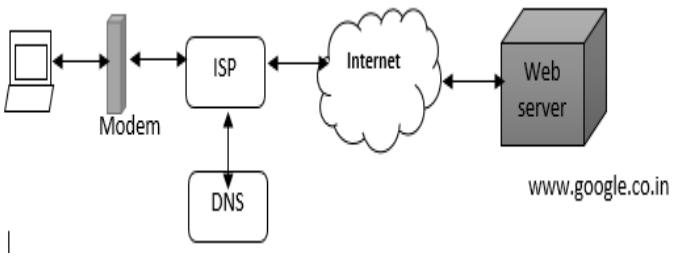
**1. Explain the concept of internet with its evolution, connection types. Describe the protocols used in it.**

- Internet = Interconnection Network
- A network is defined as an interconnection of computing devices in order to transfer data between them
- An internet is defined as an interconnection of networks in order to transfer data between the networks across the globe
- It is a network connecting millions of computers across the globe.
- **Internet = Network of networks**
- It is a network of computers, open for all
- Unlimited number of users can participate in internet to retrieve data from unlimited number of information sources
- People and organizations connect into internet so that they can access its massive store of shared information
- Anybody can participate in internet and upload/download/view/share information
- There is no organization in charge of internet; Nobody claims the ownership of internet, it is a cooperative endeavour

**Essentials for an internet connection:-**

| | | |
|---|---|---|
| Computer | DSL | Application software: Browser, email, etc |
| Connection | Modem | ISP |
| Cable | Network software: TCP/IP | Wired/Wireless line |

**Working of Internet:-**




**Evolution of internet:-**

- It was originated in 1969 at ARPANET (Advanced research project Agency) of DoD (Department of Defense), USA
- It's prime purpose was to connect among various bodies of US government
- Initially there were only four nodes (Hosts)
- In 1972, ARPANET was spread across the globe with 23 nodes at different parts of the world
- Then all the other organizations in respective countries joined to this network in order to send and receive data among other countries
- Thereby internet has got populated with number of networks, thus became a tech giant
- Around 1990s, Tim Berners Lee and O-Reilly had developed WWW and other internet communication protocols

**Terminologies used in internet:-**

- **Host:** A computer that is connected to internet
- **Communication service:** Protocols to send and receive data over the internet such as FTP, HTTP, WWW, VoIP, etc.
- **ISP:** Internet Service providers are decentralized and those who provide internet connectivity to its subscribers. **Ex: BSNL**
- **Online:** When a computer is connected to internet
- **Hyperlink:** Allows an user to move from one page to another
- **Protocols:** Set of rules for communication
- **TCP/IP:** to establish a virtual connection between source and destination. It guarantees data delivery, reliable, ordered packet delivery, etc
- **Client/Server Model:** TCP/IP uses this model where a client refers to any computing device that generates HTTP request and server refers to any computer that responds to the corresponding request
- **IP address:** It is the unique address assigned to a computing device that gets connected to the internet. It is also called as logical address or software address. It is mutable.
- **DNS:** Domain Name Servers are used to translate the website names given by the users into machine understandable IP addresses from a database.
  - **URL:** Uniform Resource Locator (URL) is defined as an unique address for the file that has to be accessed over the internet. If we want to access a website, we enter its URL in the address bar of the web browser.
    Syntax: protocol: //www.exampleDomain.com/path/filename
    Ex: https://www.vit.ac.in / home.aspx
- **WWW:** It is a standard in which all the websites are server on the internet via HTTP. It was invented by Tim Berner's Lee at Switzerland on 1990s. Later HTTP and HTML were invented, In 1994, WWW was invented at MIT (Massachusetts Institute of Technology) + DARPA

**Working:-**

- From a web browser, user sends HTTP request to a server
- ISP finds the corresponding site from DNS and forwards it.
- The request reaches the server after a long travel
- Server responds to that request and the reply goes back
- Any file transmitted in internet will not be sent as a whole
- All the information will be chopped into chunks (data packets)
- Packets have header and footer info, useful for ordering

| Advantages – internet | Disadvantages – internet |
|---|---|
| • Connect with remote people<br>• Surf any kind of information<br>• Education + entertainment<br>• E-commerce<br>• Research purpose, etc | • Loose personal info<br>• Spamming<br>• Virus attacks |

**ISP Types:-**

- Access Providers
- Mailbox providers
- Hosting ISP
- Virtual ISP
- Free ISP

**Connection types:-**

- Dial-up connection (SLIP, PPP)
- ISDN(Integrated Services Digital Network)
- DSL (Digital Subscribers Line)
  - ADSL (Asymmetric DSL)
  - SDSL (Symmetric DSL)
  - HDSL (High bit Rate DSL)
  - RDSL ( Rate Adaptive DSL)
  - VDSL (Very High Bit Rate DSL)
  - IDSL (ISDN DSL)
- Cable TV Connection
- Satellite Connection
- Wireless Connection

**Protocols used in internet**

**FTP (File transfer Protocol)**

- FTP is used to share files among the computers in the LAN
- It uses two connections (data transfer and control)
- FTP data transfer connection uses port 20
- FTP control connection uses port 21
- Some familiar commands in FTP are: USER, PASS, QUIT, CWD, DELE, LIST, RETR, STOR, HELP

**HTTP (Hyper Text Transfer protocol)**

- Hyper Text Transfer protocol (HTTP) is a request/Response, stateless protocol for communication, to transfer information on LAN and WWW
- It is used to deliver files virtually and other data on WWW
- It takes place through TCP/IP sockets
- A browser is a HTTP client – sends HTTP request
- A web server is a HTTP server – sends HTTP reply
- It uses port no: 80 (HTTP servers listen to this port)

**SNMP (Simple Network Management protocol)**

- It is used to manage a network such as its participants, etc
- Types of participants: Supervisors, Agents
- UDP is used for message transfer between them

**TCP:-**

- Connection oriented (link between the packets)
- ACK is available
- Reliable
- Heavy weight protocol
- Handshaking mechanism available
- Error control, flow control, congestion control mechanisms
- Complex, tough to implement
- Ex: Telnet, SMTP, FTP, e-mail, SSH, HTTP, HTTPS

| SMTP | POP | IMAP |
|---|---|---|
| • Connection-oriented<br>• Text-based<br>• Works in application layer<br>• ACK is available<br>• It uses port 25 | • It uses port 110<br>• Current version:POP3<br>• Single client<br>• Offline email access<br>• Can't search email<br>• Download is needed<br>• Only 1 mailbox<br>• Less internet usage | • Manipulate email<br>• No downloading<br>• Can't transfer email<br>• Access the received emails<br>• Search the mails<br>• Many mailboxes<br>• More internet usage |

**2. Explain Web 2.0 and RIA with its architecture, collaboration tools with their features, merits and demerits.**

**Web 2.0**

- Internet has revolutionized the computer and communications, undergoing extreme make-over
- In 1990s, it was used to retrieve information, information flow was unidirectional (Read-Only)
- Around 2004, new web tools came up, to add contents to web
- People with no programming knowledge can publish an article, photo, video, ppt, pdf, etc.
- Web has become 2-way communication medium (R/W)
- This is called Web 2.0 (bidirectional data traffic)
  - It is not the second version of Web
  - It is also called Participatory web (or) Read/Write Web
  - Get , post, manipulate, share information
  - Web 2.0 refers to the transition of static HTML pages to dynamic web
  - XML is used
  - It offers freedom for everybody to contribute to the web
  - Ex: Wikipedia, FB, YouTube, Twitter, etc.

**Components of web 2.0**

- Blogs
- Wikis
- Web services

**Features of Web 2.0**

- Classify and find required info
- Get dynamic contents from the web
- Information is shared among all users on the web
- Information is used and reused
- Mass participation in discussion forum

**Technologies used in web 2.0:-**

- At client side: client side scripting languages (AJAX, java script)
- At server side: server side scripting languages (PHP, PYTHON, RUBY, etc)

**Parts of web 2.0:-**

- RIA
  - It has characteristics of a desktop app
  - It is delivered by site-specific browsers
- WOA
  - It defines how web 2.0 apps show their functionality, so that other apps join with it Ex: RSS Feeds, Web services
- Social web
  - Interact with end user
  - Make end user, an integral part of the app.

**Advantages of web 2.0**

- Equal chance to all to post/view/comment/share
- Latest/updated contents
- Social networking sites are useful to be in contact
- Write reviews about a product
- Digital ad

**Disadvantages of web 2.0**

- Increased spam
- Info overloaded (everybody posts)
- Negative feedback may cause bad effect on business

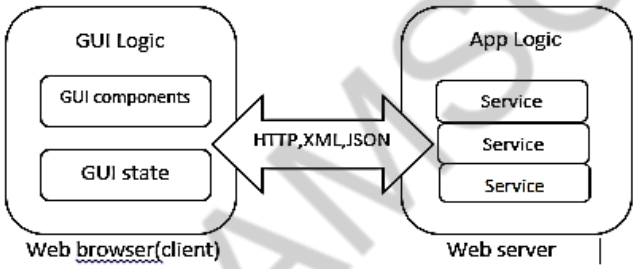
**RIA (Rich Internet Application):-**

- RIA (Rich Internet Application) is defined as a web application that is designed to give the same features and functions associated with desktop applications.
- HTML is not having much capability and performance in web apps
- Users need desktop type of interaction from web apps
- RIA fulfils this need, user interactivity
- It is the 3rd generation of web apps
- It runs inside a web browser and does not need any special software installation (plug&play)

**Features of RIA:-**

- Ability to work on web, presents complex info to users
- Rich set of visual elements like image, video, graphics, etc
- It works in real-time, helps business services
- Users can know how to use complex apps
- Reduce unnecessary page navigations
- Responsiveness, interactivity
- Ex: Apache Flex, Quick PHP, .NET framework, JavaFX

**Architecture of RIA:-**




- GUI logic is moved from server to client
- Because GUI is executed in browser, CPU time needed to generate GUI is taken off from server, thereby making server free for more CPU cycles to run app logic
- GUI state is kept in browser
- Because GUI is separated from app logic, it is easy to implement
- RIA communicate with servers by exchanging data, not the GUI code (HTML, CSS, JS)
- Data exchange: XML via HTTP (or) JSON via HTTP
- If server side becomes completely free, then the app logic will become very clear to understand
- App logic just need to focus on data in and data out

**Technologies used in RIA:-**

- HTML5+CSS3, Java script, JS framework, jQuery, jQuery Mobile, AngularJS, SmartClient, GWT, JavaFX, Flex, MS Silverlight

**Benefits of RIA:-**

- Increased productivity, new customers
- Reduced operational costs
- No installation required
- Easy upgrade
- Available through internet
- Rich and more responsive UI
- Client/server balance
- Asynchronous communication
- Efficiency in network

**Limitations of RIA:-**

- Too fast in displaying contents
- Maintain balance between HTML and RIA
- GUI logic and app logic might be in different languages
- Search engines are declining
- Proprietary
- Loss of integrity
- Complicated to develop apps, what to cache, what not to.
- Breaks web page paradigm

**Collaboration tools:-**

- Collaboration tools allow a group of people work together virtually in real-time over the internet.

**Features:-**

| | |
|---|---|
| Easy to use and set up. | Clean interface |
| Secure | Permissions control |
| Ability to upload documents | File storage |
| Scalable | Document locking |

**Examples:-**

| Tool | Use |
|---|---|
| **Google Docs** | Upload/modify/retrieve files anytime |
| **Dropbox** | Store/share/sync files online |
| **Blogger** | Blogging site of google |
| **Wordpress** | Flexible, FOSS, easy blogging tool |
| **PDFcatch** | e-books, PDF search engine |
| **SlideShare** | PPT, PDF share/upload/download |
| **Youtube** | Upload/download/view videos |
| **Facebook** | Upload/download/view micro contents |
| **Twitter** | Upload/doenload/view thoughts |

**Advantages of Collaboration tools:-**

- Reduces distance between employees
- Work in same room, together in same documents
- No need to send documents back and forth between offices
- Communication between employees is improved
- Increases team work and transparency
- Easy to keep track of projects
- Easy to generate reports
- Team members can be present anywhere
- Online chatting
- IRC (Internet relay Chat)
- Video conferencing

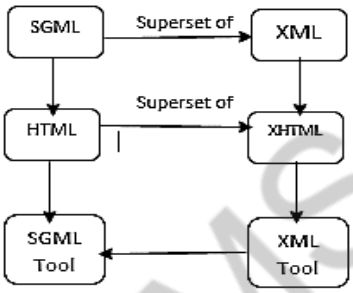
**3. Explain the features and flavours of HTML5. Develop a web page to display Timetable of your class (Nov/Dec 2016)**

**HTML5:-**

- HTML stands for Hyper Text Mark-up Language
- It is used to organize text, graphics, audio, video on a web page
- It is a formatting language used to design the decoration and contents of a web page
- Hypertext means, the text which acts as a link
- Mark up means symbols that are used to define structure of the text. It tells browser how to display the text (tags)
- Language refers to the syntax
- It was invented by Tim-Berners Lee at CERN
- HTML 1.0 (1991), HTML 2.0 (1995), HTML 3.2 (1997), HTML 4.0 (1999), XHTML (2000), HTML5 (2014)

**Features of HTML5:-**

- HTML 5.0 is the 5th version of HTML by W3C (Oct 2014)
- To support latest multimedia, more readable
- <audio>, <video>, <canvas>, <svg> tags are supported
- <header>, <footer>, <article>, <section> are supported
- Number, date, time, calendar, range are suooprted
- API available for geolocation, drag&drop, local storage, etc.
- Allows Javascript to run in BG
- 2D and 3D drawings supported
- Need for flash plugin is reduced
- Simple DOCTYPE : < ! DOCTYPE HTML>




**Flavours of HTML:-**

- **XHTML 1.0 Strict** : When we want a clean mark-up code
- **XHTML 1.0 Transitional:** To use HTML features
- **XHTML 1.0 Frameset:** To make use of frames

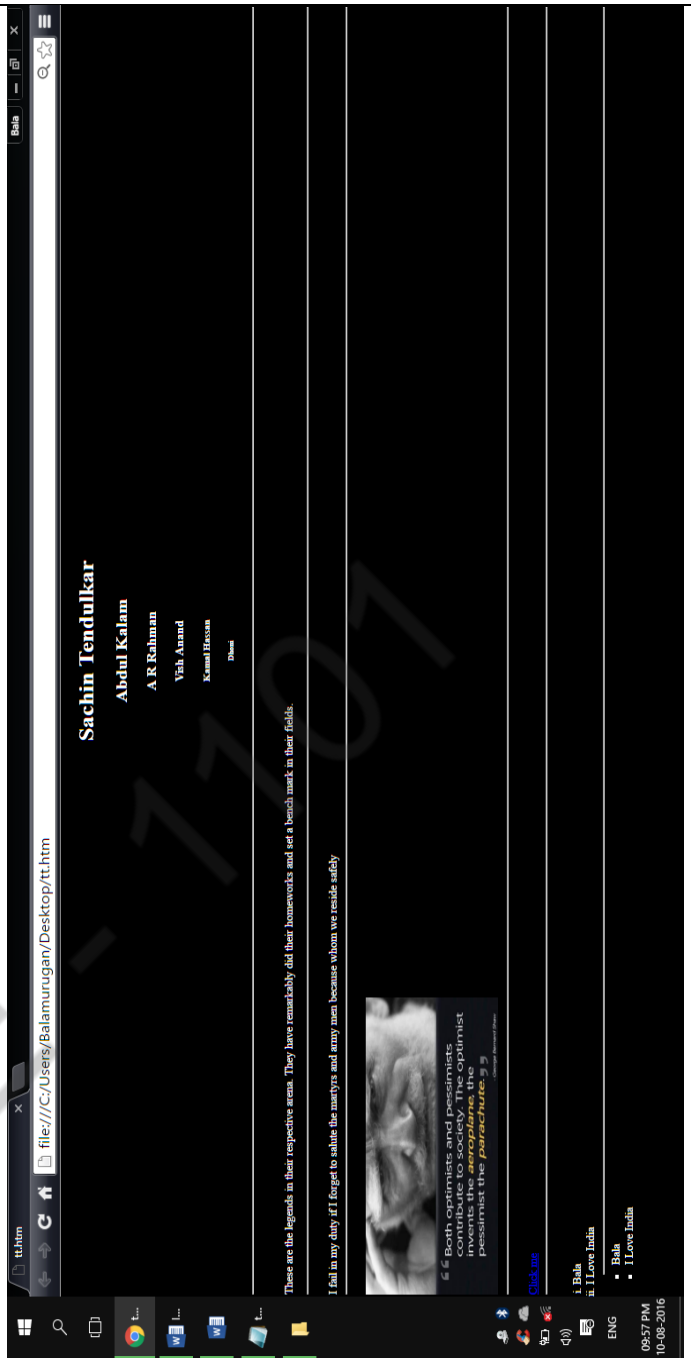
**HTML Code to display class Timetable with all Table properties:-**

| DAYS | 1 | 2 | 3 | 4 | | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| MON | IP | DM | CG | OO | | TOC | Lab1 | |
| TUE | IP | DM | CG | OO | | TOC | Lab2 | |
| WED | IP | DM | CG | OO | | TOC | Lab3 | |
| THU | IP | DM | CG | OO | | TOC | DM | IP |
| FRI | IP | DM | CG | OO | | TOC | DM | IP |

```
<HTML>
    <HEAD></HEAD>
    <BODY>
        <TABLE BORDER=5 BGCOLOR=SKYBLUE WIDTH=1000px HEIGHT=600px CELLPADDING=5 CELLSPACING=10>
        <TR ALIGN=CENTER> <TH> Day</TH><TH> 1</TH><TH> 2</TH><TH> 3</TH><TH>4</TH><TH ROWSPAN=6>*</TH><TH>5</TH><TH> 6</TH><TH> 7</TH></TR>
        <TR ALIGN=CENTER> <TH> MON</TH><TD> IP</TD><TD> DM</TD><TD>CG</TD><TD>OO</TD><TD>TOC</TD><TD COLSPAN=2>Lab1</TD></TR>
        <TR ALIGN=CENTER> <TH> TUE</TH><TD> IP</TD><TD> DM</TD><TD>CG</TD><TD>OO</TD><TD>TOC</TD><TD COLSPAN=2>Lab2</TD></TR>
        <TR ALIGN=CENTER> <TH> WED</TH><TD> IP</TD><TD> DM</TD><TD>CG</TD><TD>OO</TD><TD>TOC</TD><TD COLSPAN=2>Lab3</TD></TR>
        <TR ALIGN=CENTER> <TH> THU</TH><TD> IP</TD><TD> DM</TD><TD>CG</TD><TD>OO</TD><TD>TOC</TD><TD>DM</TD><TD>IP</TD></TR>
        <TR ALIGN=CENTER> <TH> FRI</TH><TD> IP</TD><TD> DM</TD><TD>CG</TD><TD>OO</TD><TD>TOC</TD><TD>DM</TD><TD>IP</TD></TR>
        </TABLE>
    </BODY>
</HTML>
```

**4. Explain the following using HTML code:-**
**Header tag, Paragraph tag, divide tag, text alignment, change font colour, change background colour, display an image, and insert a hyperlink, lists.**

```
<!DOCTYPE HTML>
<HTML>
        <HEAD></HEAD>
        <BODY BGCOLOR=YELLOW>
                <center>
                <h1>Sachin Tendulkar</h1>
                <h2>Abdul Kalam</h2>
                <h3>A R Rahman</h3>
                <h4>Vish Anand</h4>
                <h5>Kamal Hassan</h5>
                <h6>Dhoni</h6>
                </center>
                <font color=GREEN size=9>
                <hr><br>
<p>These are the legends in their respective arena.
They have remarkably did their homeworks and set a bench mark in their fields.</p>
                <hr><br>
<div>I fail in my duty if I forget to salute the martyrs and army men because whom we reside safely</div>
                </font>
                <hr><br>
        <img src=deer.jpg width=150px height=150px>
                <hr><br>
        <a href="https://www.google.co.in">Click me</a>
                <hr><br>
                <ol type="i" start="iii">
                        <li>Bala</li>
                        <li>I Love India</li>
                <ol>
                <hr>
                <ul type=SQUARE>
                        <li>Bala</li>
                        <li>I Love India</li>
                <ul>
        </BODY>
</HTML>
```




**5. Explain forms in HTML5. Develop a student registration form using Text, Text area, Check box, Radio buttons, Button, Dropdown Menu items. (May/ June 2016)**

- Form is a layout on web page by which a user can interact with web page. Its components are: text, textarea, checkbox, radiobutton, dropdown menu.

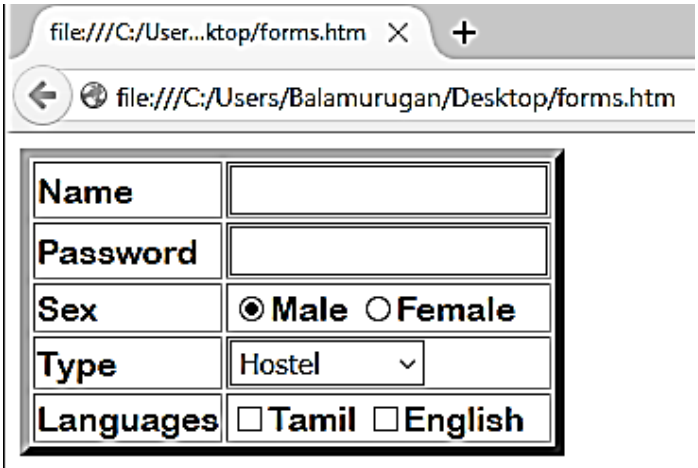
**Purpose of forms:-**

- HTML form element is used to allow a user to give input data on the web page.
- To create registration forms, login forms, getting user info, conducting surveys
- <form>……..</form> tags are used
- Attributes used: action, method

**Ex:**

<form **action** = http://www.google.co.in/ **method** = post>
</form>

```
<HTML>
<BODY>
        <FONT SIZE=14>
        <FORM>
                <TABLE BORDER=5>
                        <TR>
                                <TD>Name</TD>
                                <TD><input
type=text></TD>
                        </TR>
                        <TR>
                                <TD>Password</TD>
                                <TD><input
type=password></TD>
                        </TR>
                        <TR>
                                <TD>Sex</TD>
                                <TD><input      type=radio
value=Male>Male
                                <input          type=radio
value=Female>Female</TD>
                        </TR>
                        <TR>
                                <TD>Type</TD>
                                <TD><select>
                                        <option
value=Hosteller>Hostel</option>
                                        <option
value=DayScholar>DayScholar</option>
                                        </select>
                                </TD>
                        </TR>
                        <TR>
                                <TD>Languages</TD>
                                <TD><input
type=checkbox value=Tamil>Tamil
                                <input      type=checkbox
value=Tamil>English</TD>
                        </TR>
                </TABLE>
        </FORM>
        </FONT>
</BODY>
</HTML>
```




**6. What is CSS3.0? Explain the types of CSS with examples of selector classes. (Nov/Dec 2017)**

- Cascading style sheet is defined as a style sheet in which, all the style information of a web page can be defined.
- It separates the contents and the decoration of a HTML page
- It helps developers to give consistent appearance to all the elements in the web page.
- Style information is defined in separate file (ex.css)
- One or more style rules given
- Collection of these rules are called rule set
- Each rule set consists of selector string and declaration block

**Types:-**

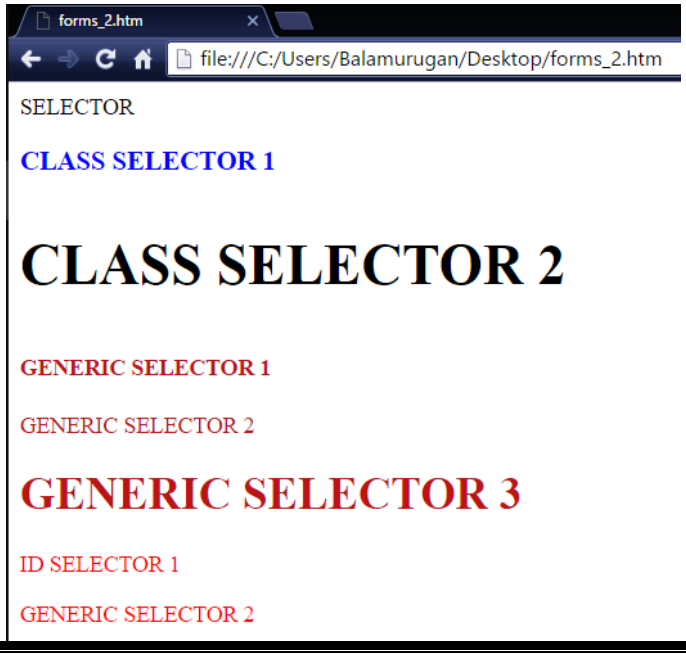
- Inline style sheets **<p style="color:green; font-size:15px">**
- Embedded style sheets **<style>……</style>**
- External style sheet **Stored in a separate file (ex.css)**
- Imported style sheets **@import URL(path)**

**Features of CSS3.0**

- Backward compatible with older versions of CSS
- It also supports selectors, box model, background, borders, text effects, animations, multiple column0 layouts, 3D transformations, etc.

**Example for Embedded CSS with all selector types:-**

```
<HTML><HEAD>
<STYLE TYPE="TEXT/CSS">
P{FONT-FAMILY:TIMES NEW ROMAN; }
H3.A{COLOR:BLUE; }
H3.B{FONT-SIZE:40PX}
 .C{COLOR:BROWN; }
 #D{COLOR:RED; }
</STYLE></HEAD>
<BODY>
        <P>SELECTOR</P>
        <H3 CLASS="A">CLASS SELECTOR 1</H3>
        <H3 CLASS="B">CLASS SELECTOR 2</H3>
        <H4 CLASS="C">GENERIC SELECTOR 1</H4>
        <DIV CLASS="C">GENERIC SELECTOR 2</DIV>
        <H1 CLASS="C">GENERIC SELECTOR 3</H1>
        <P ID="D">ID SELECTOR 1</P>
        <DIV ID="D">GENERIC SELECTOR 2</DIV>
</BODY>
</HTML>
```



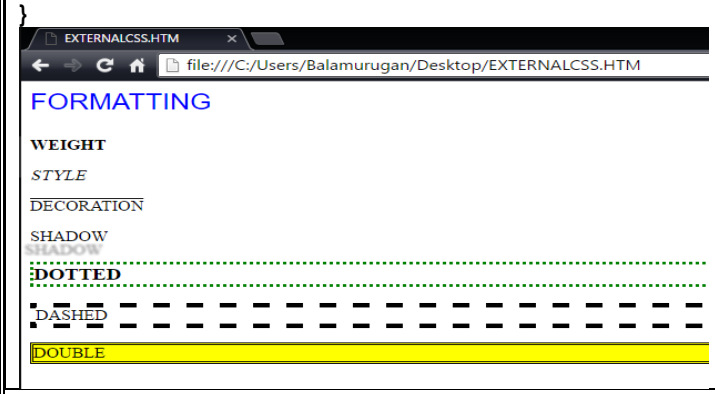
**Example for External style sheet with all text formatting:-**

EXTERNALCSS.HTM

```
<HTML>
<HEAD>
<LINK REL="STYLESHEET TYPE="TEXT/CSS" HREF="EX.CSS">
</HEAD>
<BODY>
        <P CLASS="FORMATTING">FORMATTING</P>
        <P CLASS="WEIGHT">WEIGHT</P>
        <P CLASS="STYLE">STYLE</P>
        <P CLASS="DECORATION">DECORATION</P>
        <P CLASS="SHADOW">SHADOW</P>
        <H3 CLASS="DOTTED">DOTTED</H3>
        <P CLASS="DASHED">DASHED</P>
        <P CLASS="DOUBLE">DOUBLE</P>
</BODY>
</HTML>
```

EX.CSS

```
P.FORMATTING
{
        FONT-FAMILY:ARIAL;
        FONT-SIZE:25PX;
        COLOR:BLUE;
        FONT-VARIANT:SMALL-CAPS;
}
P.WEIGHT
{
        FONT-WEIGHT:BOLD;
        FONT-WEIGHT:BOLDER;
        FONT-WEIGHT:BOLDER;
}
P.STYLE
{
        FONT-STYLE:ITALIC;
        FONT-STYLE:OBLIQUE;
}
P.DECORATION
{
        TEXT-DECORATION:LINE-THROUGH;
        TEXT-DECORATION:UNDERLINE;
        TEXT-DECORATION:OVERLINE;
}
P.SHADOW
{
        TEXT-SHADOW:-5PX 15PX 3PX GRAY;
}
.DOTTED
{
        BORDER-STYLE:DOTTED;
        BORDER-COLOR:GREEN;
}
.DASHED
{
        BORDER-STYLE:DASHED;
        BORDER-WIDTH:5PX;
}
.DOUBLE
{
        BORDER-STYLE:DOUBLE;
        BACKGROUND-COLOR:YELLOW;
}
```



## UNIT 2,3 - CLIENT SIDE & SERVER SIDE SCRIPTING

### PART- A

**1. Mention the differences between client side and server side scripting.**

| Client side scripting | Server side scripting |
|---|---|
| It runs on web browser | It runs on web server |
| Front end concept | Back end concept |
| Script is processed at end user's PC | Script is processed at server |
| To develop websites that are interactive with user | To develop sites that fetches data from DB |
| Visible to user | Invisible to user |
| Less secure | More secure |
| Less customization | High customization |
| More tasks at the browser | More tasks at the server |
| Any changes will affect DB | Changes will affect DB |
| Ex: JS, VB script, Dart | Ex: PHP, ASP.NET, Perl, Ruby, ColdFusion, Go, Python |

**2. State the differences between programming and scripting**

| Programming | Scripting |
|---|---|
| Compiled and executed | Interpreted |
| Has complete syntax and semantics | Less syntax and semantics |
| To build apps | To control apps |
| Stand alone | Need other programs to execute |
| Contents of a system | Acts upon a system |
| Heavy weight | Light weight |
| Single usage tool | Multi usage tool |
| Not as quick as scripting | Quicker |
| .EXE files cannot be viewed | Scripts can be viewed |

**3. Mention the features of JavaScript.**

- It is useful for page designers
- Light weight, interpreted, embedded in HTML
- Network centric apps
- JS = JAVA + HTML
- It is a FOSS
- To develop dynamic and interactive pages
- To react to events
- To validate data, create cookies

**4. What is DOM? What are the uses of DOM tree?**

- DOM is a set of platform independent and language independent API, that tells how to access and manipulate information stored in XML, XHTML, JS

**Uses:-**

- To identify interface and object for representing and manipulating a document
- To find behaviour and attributes of interface & object
- To find relation between interface and object

**DOM tree:-**

- Documents in DOM are represented using a tree like structure
- Every element is represented as a node
- This tree structure is called as DOM tree

**5. What are the levels of DOM?**

**Level 0 :** To access few html elements (by Netscape in 1990s)
**Level 1:** To change entire web page (1998)
**Level 2:**→Platform independent, language independent
→To access dynamically, update contents, structure, style
**Level 3:** → Platform independent, language independent
→To access dynamically, update contents, structure, style

**6. What are getElementByID() and innerHTML properties?**

**getElementById:-**

- To access HTML element, we need ID of it.
- For instance, there can be many ** … **tags in a same HTML document.
- To find a specific element from the HTML document, we use getElementById() method

**innerHTML:-**

- To get the content of an element, this property is used
- To get/replace the contents present within tags

**7. What is validation?**

- It occurs usually at the server, after the client had entered all necessary data and then clicked submit button
- If user enters some wrong/missing data, server has to send all the contents back to client and request for resubmission with correct information
- This increases the task of a server
- Javascript validates user's data at the browser, reduces the workload of a server

**8. What are the differences between HTML and DHTML?**

| HTML | DHTML |
|---|---|
| Hypertext Markup Language | Dynamic HTML |
| Static web pages | Dynamic web pages |
| It works slowly upon client-server technology | It works faster on client-server technology |
| No CSS, and no dynamic contents | Use CSS, events, methods to create dynamic pages |
| No processing at browser | Script is processed at browser |
| Contents will not be changed | Contents can be changed |
| Simple, less interactive | Complex, more interactive |
| Only HTML contents | DHTML = HTML+CSS+JS |

**9. Define servlet.** (Nov/Dec 2017)

- Servlets are defined as simple java programs that are dynamically loaded and run on JVM of web servers, to respond to the requests from the clients
- It acts as middle layer between browser and server
- To develop sites with secure access, interact with DB, maintain unique session info of each client
- Used with HTTP, hence called HttpServlet
- It makes use of two packages: Javax.servlet and javax.servlet.http

**10. What is servlet container?**

- The server that executes a servlet is called as servlet container or servlet engine
- Browsers send an HTTP request to server, which in turn sends to servlet container
- Servlet container receives the request from the server, processes appropriate servlet, sends back request.

**17. What are the uses of cookies?**

- Identifying a user during an e-commerce session
- Avoiding username and password.
- Customizing a website as we want.
- Focusing on advertising in web pages

**11. What are the methods and phases of servlet life cycle?**

**Methods:-** (Nov/Dec 2016)

- init(), service(), destroy()

**Phases:-**

- **Phase 1:** Servlet class is loaded
- **Phase 2:** Servlet instance is created
- **Phase 3:** Init() method is invoked
- **Phase 4:** Service() method is invoked
- **Phase 5:** Destroy() method is invoked

**12. Mention the differences between GET and POST**

| HTTP GET request | HTTP POST request |
|---|---|
| doGet() method is used | doPost() method is used |
| URL string displays request submitted by the user | URL string does not display request submitted by user |
| To download info from server | To upload info from server |
| No effect on data | Has effect on data |
| Page can be bookmarked | Page cannot be bookmarked |
| page can be cached, saved in history | Page cannot be cached, cannot be saved in history |
| Only ASCII characters allowed | Any character is allowed |
| Unsafe | More secure |

**13. What are session tracking techniques?**

- It is a mechanism by which we can keep track of previous sessions between server and browser
- Session ID is passed between client and server
- HTTP cannot have any data about precious client-server communication (stateless)
- To achieve it, we use session tracking

**Techniques:-**

- Use cookies
- Hidden form fields
- URL rewriting

**14. What is a cookie?Mention its types.**

- A cookie is defined as short piece of data, not actually any source code, which is sent from a web server to browser when a browser visits the server's site.
- Cookie = "name-value" pair
- It is one of the session tracking technique
- Cookie is a plain text data record of 5 fields: expiry time, domain, path, secure, "name=value"
- **Types**: Session cookies, permanent cookies

**15. What is hidden form field?**

- A hidden text field is used for maintaining the state of a user
- Here, information is stored in hidden field
- It is better if we have to submit form in all the pages and we don't depend on the browser
- Ex: **<input type="hidden" name="sid" value="abc123">**

**16. What is URL rewriting?**

- The process of adding the name of the user in the query string and getting the value from the query string in another page is called URL rewriting
- "name-value" pairs are passed in URL
- Ex: url?name1=value1&name2=value2&??

(Nov/Dec 2017)

**18. What is JDBC? What are its uses? Mention its types.**

- JDBC is defined as an API that provides industry standard and database connectivity between java apps and database servers
- It is a framework that contains many classes, interfaces, exceptions, using which java apps can send SQL statement to database to store and retrieve data

**Uses:-**

- It helps client to store and retrieve data to databases
- It helps client to update databases

**Types:-**

- JDBC-ODBC bridge driver
- Partial java driver
- Pure java driver for accessing middleware
- Pure java driver for direct DB access

**19. What is JSP?**

- Java Server Pages is a kind of server side scripting language that enables user to embed java code with HTML elements for the creation of dynamic, platform-independent method for building web apps
- JSP = Java + HTML + servlet

**20. What are the differences between JSP and servlet?**

| JSP | Servlet |
|---|---|
| JSP = Java inside HTML | Servlet = HTML inside Java |
| It generates dynamic web contents | It generates dynamic web pages |
| In MVC, JSP acts as a view | In MVC, servlet acts as controller |
| JSP makes use of custom tags | No custom tags |

**21. Define scriptlet.**

- A scriptlet can contain any number of Java language statements, variables or method declarations, or expressions that are valid in the page scripting language

**22. What is JSTL? What are its advantages?**

- Java Standard Tags Library represents set of tags to simplify JSP development
- J2EE is used for server side programming using JAVA and JSTL (a compoment of J2EE web app development)
- It is useful in performing condition execution, loop execution, data procession, etc
- Embed logic in JSP page without java code

**23. What are HttpServletRequest and HttpServletResponse?**

- They are two commonly used interfaces from **javax.servlet.http** package
- HttpServletRequest enables servlet to read data from HTTP request
- HttpServletResponse enables servlet to write data to HTTP response

## Part – B

**1. Write the JavaScript for the following:-**

**a. Find the biggest among three numbers**
**b. Find odd or even**
**c. Display squares of 10 numbers**
**d. Reverse a number**
**e. Sort array elements**

**a. Find the biggest among three numbers**

```
<html>
	<body>
		<p>Click the button to return the highest number of 5 and 10.</p>
		<button onclick="myFunction()">Click me</button>
		<p id="demo"></p>
		<script>
			function myFunction()
			{
document.getElementById("demo").innerHTML = Math.max(1,2,3,4,5);
			}
		</script>
	</body>
</html>
```

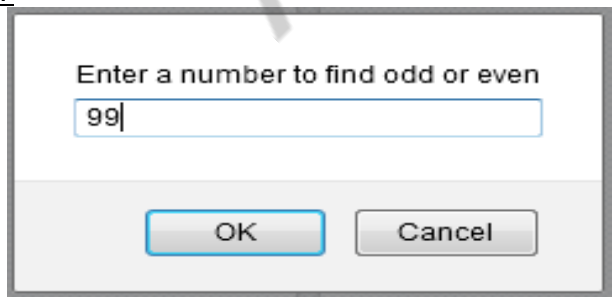
**O/p:-**

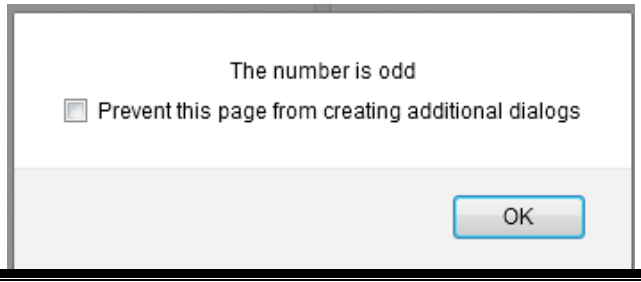


**b. Odd or even**

```
<html>
	<head>
		<script type="text/javascript">
		 var n = prompt("Enter a number to find odd or even");
		 n = parseInt(n);
		if (isNaN(n)) alert("Please Enter a Number");
		else if (n == 0) alert("The number is zero");
		else if (n%2) alert("The number is odd");
		else alert("The number is even");
		</script>
	</head>
<body>
</body>
</html>
```

**O/p:-**



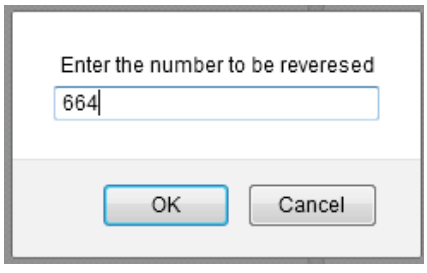

**c. Display squares and cubes of 10 numbers**

```
<html>
<body>
<table border="1">
<tr><th>num</th><th>square</th><th>cube</th>
<tr>
<script type="text/javascript">
for(i=1;i<=10;i++)
{
		document.write("<tr>");
		document.write("<td>"+i+"</td>");
		document.write ("<td>"+(i*i)+"</td>");
		document.write ("<td>"+(i*i*i)+"</td>");
		document.write ("</tr>");
}
</script>
</table>
</body>
</html>
```

| num | square | cube |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 4 | 8 |
| 3 | 9 | 27 |
| 4 | 16 | 64 |
| 5 | 25 | 125 |
| 6 | 36 | 216 |
| 7 | 49 | 343 |
| 8 | 64 | 512 |
| 9 | 81 | 729 |
| 10 | 100 | 1000 |

**d. Reverse a number**

```
<html>
	<script type ="text/javascript">
		function rev_num()
		{
		var num = prompt("Enter the number to be reveresed");
			var n= num;
			var rev = 0, rem;
			while (n>0)
			{
				rem = n % 10;
				rev = rev * 10 + rem ;
				n = Math.floor(n/ 10);
			}
			document.write(rev);
		}
	</script>
	<body onload="rev_num()">
	</body>
</html>
```



**e. Sort array elements**

```
<html>
<body>

<button onclick="fun()">Click Me</button>

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

<script>
     var fruits = ["Banana", "Orange", "Apple", "Mango"];
      document.getElementById("demo").innerHTML = fruits;
     function myFunction()
    {
            fruits.sort();
           document.getElementById("demo").innerHTML = fruits;
     }
</script>

</body>
</html>
```


**2. Write DHTML codes for the following:- (Nov/Dec 2016)**

**To Show/hide text dynamically**

```
<html>
        <body>
                <p id="p1">Sachin Tendulkar has been considered as GOD of CRICKET in the cricketing fraternity after his incompetent dominance and true spirit towards the Agame. <p>

                <button value="Hide text"
                onclick="document.getElementById('p1').
                style.visibility='hidden' " >

                <button value = "Showtext"
                onclick="document.getElementById('p1').
                style.visibility='visible' " >
        </body>
</html>
```

**O/p:-**

Sachin Tendulkar has been considered as GOD of CRICKET in the cricketing fraternity after his incompetent dominance and true spirit towards the game


**Cursor types**

```
<html>
   <body>
    <p>Move mouse over the words to see the cursor change</p>
        <span style="cursor: auto">Auto</span><br />
        <span style="cursor: crosshair">Crosshair</span><br />
        <span style="cursor: default">Default</span><br />
        <span style="cursor: pointer">Pointer</span><br />
        <span style="cursor: move">Move</span><br />
        <span style="cursor: e-resize">e-resize</span><br />
        <span style="cursor: ne-resize">ne-resize</span><br />
        <span style="cursor: nw-resize">nw-resize</span><br />
        <span style="cursor: n-resize">n-resize</span><br />
        <span style="cursor: se-resize">se-resize</span><br />
        <span style="cursor: sw-resize">sw-resize</span><br />
        <span style="cursor: s-resize">s-resize</span><br />
        <span style="cursor: w-resize">w-resize</span><br />
        <span style="cursor: text">text</span><br />
        <span style="cursor: wait">wait</span><br />
        <span style="cursor: help">help</span><br />
    </body>
</html>
```

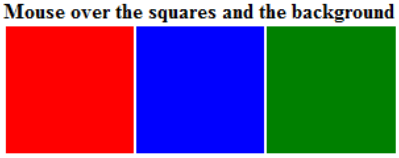
**O/p:-**

Move the mouse over the words to see the cursor change
Auto
Crosshair
Default
Pointer
Move
e-resize
ne-resize
nw-resize
n-resize
se-resize
sw-resize
s-resize
w-resize
text
wait
help

**Change Background**

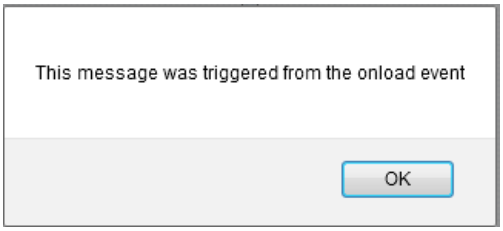
```
<html>
        <head>
        <script type="text/javascript">
        function bgChange(bg)
                {
                        document.body.style.background=bg;
                }
        </script>
        </head>
        <body>
        <b>Mouse over the squares and the background color will change!</b>
        <table width="300" height="100">
         <tr>
          <td onmouseover="bgChange('red')"
             onmouseout="bgChange('transparent')"
             bgcolor="red">
          </td>
          <td onmouseover="bgChange('blue')"
             onmouseout="bgChange('transparent')"
             bgcolor="blue">
          </td>
          <td onmouseover="bgChange('green')"
             onmouseout="bgChange('transparent')"
             bgcolor="green">
          </td>
        </tr></table></body></html>
```

Mouse over the squares and the background color will change!



**Onload event**

```
<html>
        <head>
        <script type="text/javascript">
        function mymessage()
                {
                        alert("This message was triggered from the onload event");
                }
        </script>
        </head>
        <body onload="mymessage()">
        </body>
</html>
```




**Onblur event**

```
<html>
        <head>
        <script type="text/javascript">
        function message()
                {
                alert("This alert box was triggered by the onblur event handler");
                }
        </script>
        </head>
        <body>
                <p>The onblur event occurs when an element loses focus. Try to click or write in the input field below, then click elsewhere in the document so the input field loses focus.</p>
                <form>
                Enter your name: <input type="text" onblur="message()" size="20">
                </form>
        </body>
</html>
```

The onblur event occurs when an element loses focus. Try to click or write in the input field below, then click elsewhere in the document so the input field loses focus.

Enter your name: bala

**Onfocus event**

```
<html>
        <head>
                <script type="text/javascript">
                function message()
                        {
                        alert("This alert box was triggered ");
                        }
                </script>
        </head>
        <body>
                <form>
    Enter your name: <input type="text" onfocus="message()" size="20">
                </form>
        </body>
</html>
```

Enter your name:

**Change image dynamically**

```
<html>
<head>
<script type="text/javascript">
cc=0;
function changeimage()
{
if (cc==0)
        {
        cc=1;
        document.getElementById('myimage').src="Desert.jpg";
        }
else
        {
        cc=0;
        document.getElementById('myimage').src="Lighthouse.jog";
        }
}
</script>
</head>
<body>
        <img id="myimage" onclick="changeimage()"
        src="Lighthouse.jpg" width="100" height="180" >
        Click to change the image.
</body>
</html>
```

**To change font color dynamically**

```
<html>
        <body>
                <h1 id="header" onclick="this.style.color='red'">Click Me!</h1>
                <p>If you click the header above, it turns red.</p>
        </body>
</html>
```

**O/p:-**

**Click Me!**

If you click the header above, it turns red.

**Onmouseout and onmousemove**

```
<html>
        <head>
        <script type="text/javascript">
        function nameon()
                {
 document.getElementById('h2text').innerHTML="WELCOME!";
                }
        function nameout()
                {
document.getElementById('h2text').innerHTML="How r u tdy";
                }
        </script>
        </head>
        <body>
        <h2          id="h2text"          onmouseout="nameout()"
        onmouseover="nameon()">
        Mouse over this text!</h2>
        </body>
</html>
```

**O/p:-**

**WELCOME!**
**How are You Today?**

**Increase text size dynamically**

```
<html>
        <head>
                <script type="text/javascript">
                txtsize=0;
                maxsize=100;
                function writemsg()
                {
                        if (txtsize<maxsize)
                                {
    document.getElementById('msg').style.fontSize=txtsize;
                  txtsize++;
                  timer=setTimeout("writemsg()",10);
                                }
                }
                function stoptimer()
                        {
                        clearTimeout(timer);
                        }
                </script>
        </head>
        <body onload="writemsg()" onunload="stoptimer()">
        <p id="msg">Bala is here!!!</p>
        </body>
</html>
```

**O/p:-**

# Bala is here!!!

**3. Explain the architecture and working of servlet. (Nov/Dec 2017)**

- Servlets are defined as simple java programs that are dynamically loaded and run on JVM of web servers, to respond to the requests from the clients
- It acts as middle layer between browser and server
- To develop sites with secure access, interact with DB, maintain unique session info of each client
- Used with HTTP, hence called HttpServlet
- It makes use of two packages: Javax.servlet and javax.servlet.http

**servlet container**

- The server that executes a servlet is called as servlet container or servlet engine
- Browsers send an HTTP request to server, which in turn sends to servlet container
- Servlet container receives the request from the server, processes appropriate servlet, sends back request

**Steps:-**

**1) Servlet class is loaded**

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

**2) Servlet instance is created**

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

**3) init method is invoked**

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

1. public void init(ServletConfig config) throws ServletException

**4) service method is invoked**

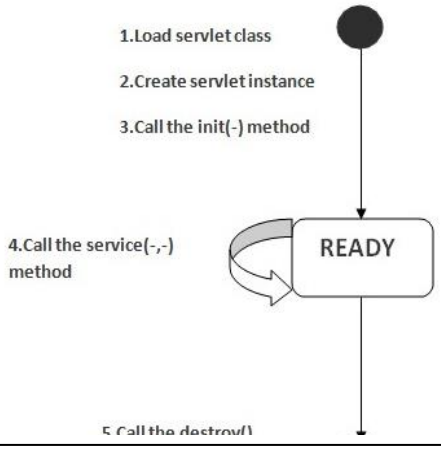
The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

1. public void service
   (ServletRequest request, ServletResponse response)

2. throws ServletException, IOException

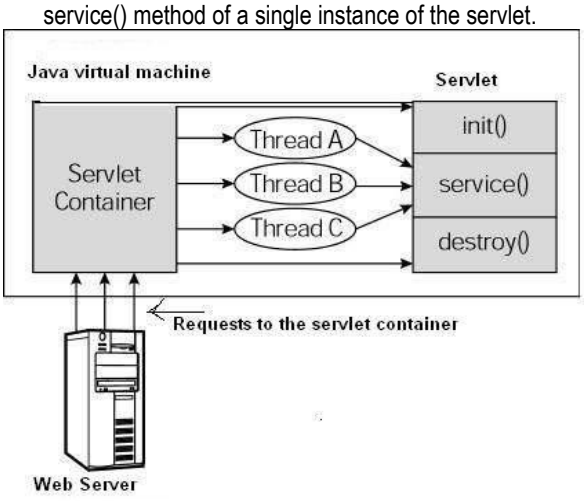
**5) destroy method is invoked**

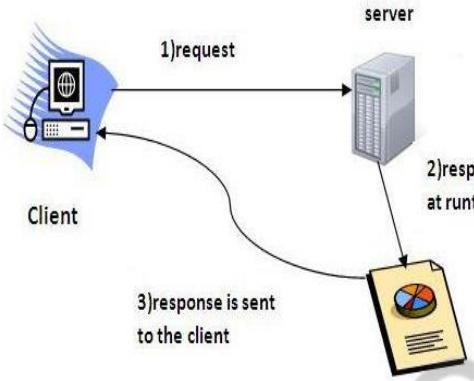
The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:



**Architecture Digram**

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.




**Advantages**

1. **better performance:** because it creates a thread for each request not process.
2. **Portability:** because it uses java language.
3. **Robust:** Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.
4. **Secure:** because it uses java language..

| Servlet Terminology | Description |
|---|---|
| **Website: static vs dynamic** | It is a collection of related web pages that may contain text, images, audio and video. |
| **HTTP** | It is the data communication protocol used to establish communication between client and server. |
| **HTTP Requests** | It is the request send by the computer to a web server that contains all sorts of potentially interesting information. |
| **Get vs Post** | It give the difference between GET and POST request. |
| **Container** | It is used in java for dynamically generate the web pages on the server side. |
| **Server: Web vs Application** | It is used to manage the network resources and for running the program or software that provides services. |
| **Content Type** | It is HTTP header that provides the description about what are you sending to the browser. |

**Example:-**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
        public void init()
        {
        }
         public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
        {
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("<html><body>helloworld</body></html>");
        }
        public void destroy()
        {
        }
}
```

**Compiling a Servlet**

- Let us put above code if HelloWorld.java file and put this file in C:\ServletDeveloper
- add these directories as well in CLASSPATH.
- Assuming your environment is setup properly, go in **ServletDeveloper** directory and compile HelloWorld.java as follows: $ javac HelloWorld.java
- If the servlet depends on any other libraries, you have to include those JAR files on your CLASSPATH as well.
- I have included only servlet-api.jar JAR file because I'm not using any other library in Hello World program.
- This command line uses the built-in javac compiler that comes with the Sun Microsystems Java Software Development Kit (JDK).
- For this command to work properly, you have to include the location of the Java SDK that you are using in the PATH environment variable.
- If everything goes fine, above compilation would produce **HelloWorld.class** file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

**Servlet Deployment**

- By default, a servlet application is located at the path <Tomcat-installation-directory>/webapps/ROOT and the class file would reside in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes.
- If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class.
- For now, let us copy HelloWorld.class into <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

**web.xml**

```
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HelloWorld</servlet-name>
  <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

- Above entries to be created inside <web-app>...</web-app> tags available in web.xml file. There could be various entries in this table already available, but never mind.
- start tomcat server using <Tomcat-installation-directory>\bin\startup.
- type **http://localhost:8080/HelloWorld** in browser's address box.
- If everything goes fine, you would get following result:


**4. Explain the three ways of creating servlet with examples. (or) Explain the methods of implementing servlet.**

The servlet can be created by three ways:

1. By implementing Servlet interface,
2. By inheriting GenericServlet class, (or)
3. By inheriting HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

**Method 1: By implementing Servlet interface**

```
import java.io.*;
import javax.servlet.*;
public class First implements Servlet
{
        public void init()
        {
                System.out.println("servlet is initialized");
        }
        public void service(ServletRequest req,ServletResponse res) throws IOException,ServletException
        {
                res.setContentType("text/html");
                PrintWriter out=res.getWriter();
                out.print("<html><body>helloworld</body></html>"
        );
        }
        public void destroy()
        {
                System.out.println("servlet is destroyed");
        }
}
```

**Web.xml**

```
<web-app>
<servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>hello</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

**Method 2 : By inheriting GenericServlet class**

```
import java.io.*;
import javax.servlet.*;
public class First extends GenericServlet
{
        public void init()
        {
                System.out.println("servlet is initialized");
        }
        public void service(ServletRequest req,ServletResponse res) throws IOException,ServletException
        {
                res.setContentType("text/html");
                PrintWriter out=res.getWriter();
                out.print("<html><body>helloworld</body></html>");
        }
        public void destroy()
        {
                System.out.println("servlet is destroyed");
        }
}
```

**Web.xml**

```
<web-app>
<servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>hello</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

**Method 3 : By inheriting HttpServlet class**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class First extends HttpServlet
{
        public void init()
        {
                System.out.println("servlet is initialized");
        }
        public void service(HttpServletRequest req, HttpServletResponse res) throws IOException,ServletException
        {
                res.setContentType("text/html");
                PrintWriter out=res.getWriter();
                out.print("<html><body>helloworld</body></html>");
        }
        public void destroy()
        {
                System.out.println("servlet is destroyed");
        }
}
```

**Web.xml**

```
<web-app>
<servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>hello</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```


**5. Explain GET and POST methods in HTTP with examples.** (Nov/Dec 2017)

**Login.html**

```
<html>
<body>
        <form action="login" method="get">
                <table>
                        <tr>
                            <td>User:</td>
                            <td><input name="user" /></td>
                        </tr>
                        <tr>
                            <td>password</td>
                          <td><input name="password" /></td>
                        </tr>
                </table>
                <input type="submit" />
        </form>
</body>
</html>
```

- create a Servlet which receives the request in **/login** , which is the indicated direction in the **action** attribute of the tag **<form>** of login.html

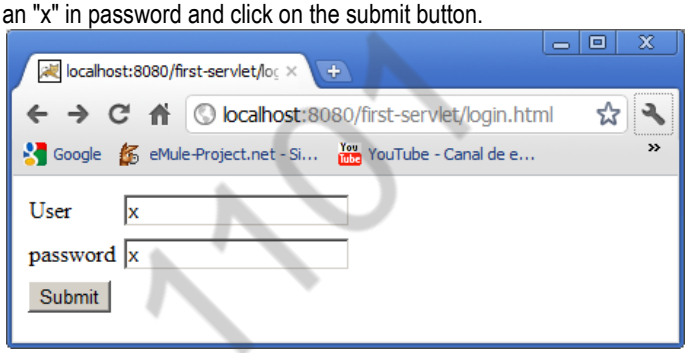
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LoginServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException
      {
                String user = req.getParameter("user");
                String pass = req.getParameter("password");
          if ("balamurugan".equals(user) && "bala1234".equals(pass))
          {
                response(resp, "login ok");
            }
        else
          {
                response(resp, "invalid login");
            }
          }
    private void response(HttpServletResponse resp, String msg) throws IOException
        {
                PrintWriter out = resp.getWriter();
                out.println("<html><body>");
                out.println(msg);
                out.println("</body></html>");
          }
}
```

- We compilate this Servlet and we include **LoginServlet.class** in the folder **/WEB-INF/classes**. We modify web.xml to link **/login** with this Servlet.

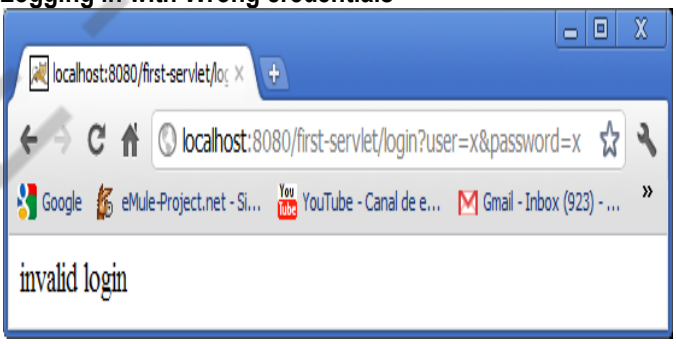
**web.xml**

```
<web-app>
        <servlet>
                <servlet-name>hello</servlet-name>
                <servlet-class> hello </servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>login-servlet</servlet-name>
                <url-pattern>/login</url-pattern>
        </servlet-mapping>
</web-app>
```

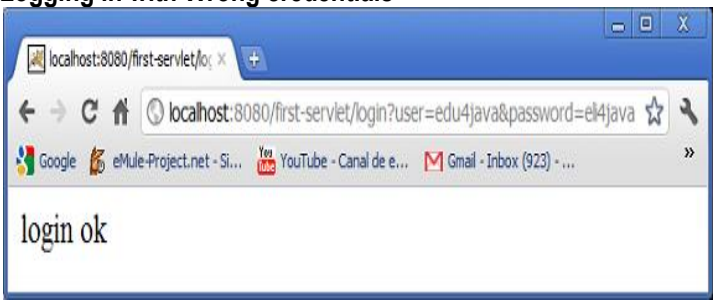
We restart the server, open the page **login.html**, write an "x" in user, write an "x" in password and click on the submit button.




**Logging in with Wrong credentials**



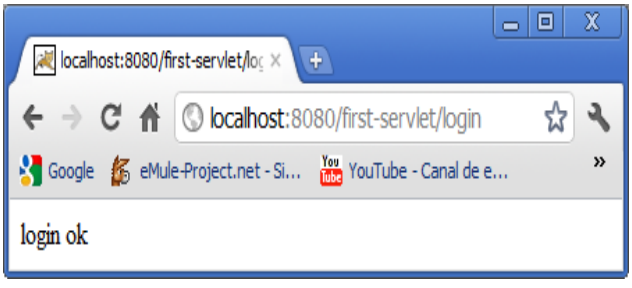

**Logging in with Wrong credentials**




```
<html>
<body>
 <form action="login" method="post">
    <table>
       <tr><td>User</td>  <td><input name="user" /></td></tr>
       <tr><td>password</td> <td><input name="password" /></td></tr>
    </table>
          <input type="submit" />
          </form>
</body>
</html>
```


- Reusing login.html, we will use the following error.
- What is happening here is that we haven´t implemented the **doPost** method (we have only implemented doGet), so our Servlet is not able to receive **POST** requests. In the following code we can see the necessary modifications to make it work.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LoginServlet extends HttpServlet
{
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException
    {
                String user = req.getParameter("user");
                String pass = req.getParameter("password");
        if ("balamurugan".equals(user) && "bala1234".equals(pass))
        {
                response(resp, "login ok");
          }
       else
         {
                response(resp, "invalid login");
          }
        }
    private void response(HttpServletResponse resp, String msg) throws IOException
       {
                PrintWriter out = resp.getWriter();
                out.println("<html><body>");
                out.println(msg);
                out.println("</body></html>");
        }
}
```




- We can see that the parameters of the URL have dissapeared.

**6. Explain session tracking techniques with example.** (Apr '2017)

- It is a mechanism by which we can keep track of previous sessions between server and browser
- Session ID is passed between client and server
- HTTP cannot have any data about precious client-server communication (stateless)
- To achieve it, we use session tracking

**Session Tracking Techniques**

There are four techniques used in Session tracking:

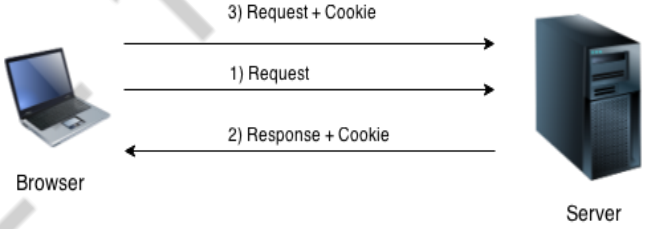
1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

**Cookies in Servlet**

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

**How Cookie works**

- By default, each request is considered as a new request.
- In cookies technique, we add cookie with response from the servlet.
- So cookie is stored in the cache of the browser.
- After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.




**Types of Cookie**

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

**Non-persistent cookie**

- It is **valid for single session** only. It is removed each time when user closes the browser.

**Persistent cookie**

- It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

**Advantage of Cookies**

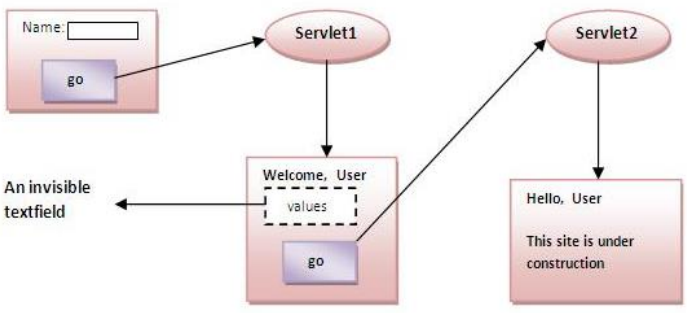
1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

**Disadvantage of Cookies**

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

***Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.***

**Hidden Form Fields**

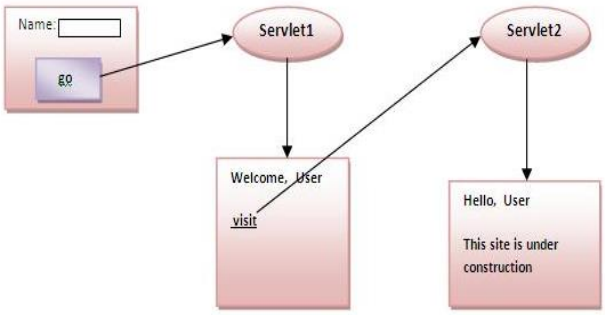


- A web server can send a hidden HTML form field along with a unique session ID as follows:

<input type="hidden" name="sessionid" value="12345">

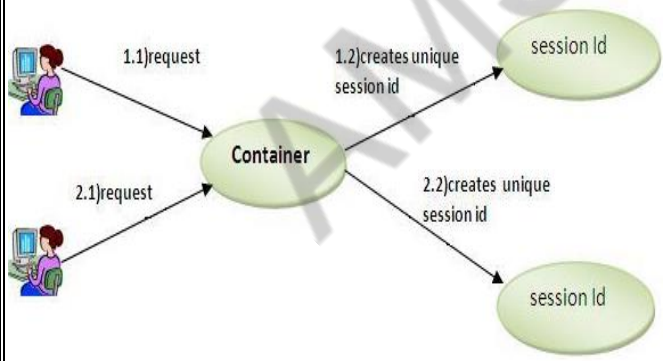
- This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data.
- Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.
- This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

**URL Rewriting**




- You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.
- For example, with http://tutorialspoint.com/file.htm;sessionid=12345, the session identifier is attached as sessionid=12345 which can be accessed at the web server to identify the client.
- URL rewriting is a better way to maintain sessions and works for the browsers when they don't support cookies
- but here drawback is that you would have generate every URL dynamically to assign a session ID though page is simple static HTML page.

**The HttpSession Object**

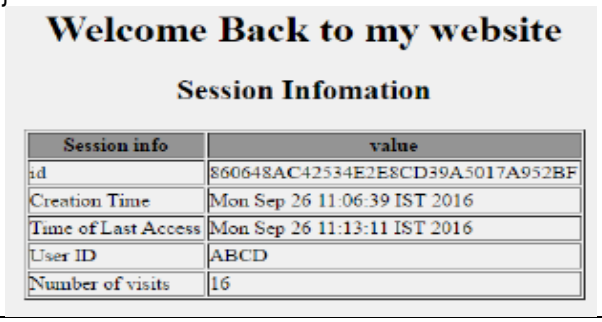



- Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
- The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.
- You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below:

```
HttpSession session = request.getSession();
```

**Program:-**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class SessionTracker extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException
{
      HttpSession session = request.getSession(true);
      Date createTime = new Date(session.getCreationTime());
      Date lastAccessTime=new
Date(session.getLastAccessedTime());
      String title = "Welcome Back to my website";
      Integer visitCount = new Integer(0);
      String visitCountKey = new String("visitCount");
      String userIDKey = new String("userID");
      String userID = new String("ABCD");
      if (session.isNew())
       {
          title = "Welcome to my website";
          session.setAttribute(userIDKey, userID);
       }
     else
      {
        visitCount = (Integer)session.getAttribute(visitCountKey);
        visitCount = visitCount + 1;
        userID = (String)session.getAttribute(userIDKey);
       }
       session.setAttribute(visitCountKey, visitCount);
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
       out.println("<html><body><center>");
       out.println("Session Information");
       out.println("<table border=\"1\"><tr>");
      out.println("<th>Session info</th><th>value</th></tr>");
out.println("<tr><td>Session ID</td><td> + session.getId() +
</td></tr>");
out.println("<tr><td>Creation Time</td><td> + createTime +
</td></tr>");
out.println("<tr><td>Time of Last Access</td><td> +
lastAccessTime+ </td></tr>");
out.println("<tr><td>User ID</td><td> + userID+ </td></tr>");
out.println("<tr><td>Number of visits</td><td> + visitCount+
</td></tr>");
out.println("</table></body></html>");
            }
}
```



**Welcome Back to my website**

**Session Infomation**

| Session info | value |
|---|---|
| id | 860648AC42534E2E8CD39A5017A952BF |
| Creation Time | Mon Sep 26 11:06:39 IST 2016 |
| Time of Last Access | Mon Sep 26 11:13:11 IST 2016 |
| User ID | ABCD |
| Number of visits | 16 |

**7. Explain the working of JDBC with example.(Apr/May 2017)**

JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

JDBC stands for **J**ava **D**atab**a**se **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

**JDBC Architecture**

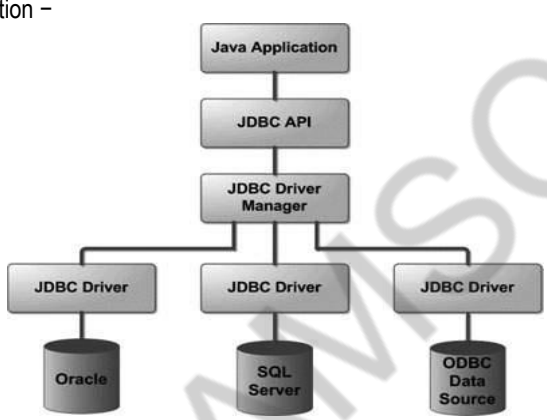
The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers −

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application −




**Common JDBC Components**

The JDBC API provides the following interfaces and classes −

- **DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
- **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
- **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
- **SQLException:** This class handles any errors that occur in a database application.
- **S**tructured **Q**uery **L**anguage (SQL) is a standardized language that allows you to perform operations on a database, such as creating entries, reading content, updating content, and deleting entries.
- SQL is supported by almost any database you will likely use, and it allows you to write database code independently of the underlying database.

**Create Database**

The CREATE DATABASE statement is used for creating a new database. The syntax is −

```
SQL> CREATE DATABASE DATABASE_NAME;
```

**Example**

The following SQL statement creates a Database named EMP −

```
SQL> CREATE DATABASE EMP;
```

**Drop Database**

The DROP DATABASE statement is used for deleting an existing database. The syntax is −

```
SQL> DROP DATABASE DATABASE_NAME;
```

**Note:** To create or drop a database you should have administrator privilege on your database server. Be careful, deleting a database would loss all the data stored in the database.

**Create Table**

The CREATE TABLE statement is used for creating a new table. The syntax is −

```
SQL> CREATE TABLE table_name
(
  column_name column_data_type,
  column_name column_data_type,
  column_name column_data_type
  ...
);
```

**Example**

The following SQL statement creates a table named Employees with four columns −

```
SQL> CREATE TABLE Employees
(
  id INT NOT NULL,
  age INT NOT NULL,
  first VARCHAR(255),
  last VARCHAR(255),
  PRIMARY KEY ( id )
);
```

**Drop Table**

The DROP TABLE statement is used for deleting an existing table. The syntax is −

```
SQL> DROP TABLE table_name;
```

**Example**

The following SQL statement deletes a table named Employees −

```
SQL> DROP TABLE Employees;
```

**INSERT Data**

The syntax for INSERT, looks similar to the following, where column1, column2, and so on represents the new data to appear in the respective columns −

```
SQL> INSERT INTO table_name VALUES (column1, column2, ...);
```

**Example**

The following SQL INSERT statement inserts a new row in the Employees database created earlier −

```
SQL> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
```

**SELECT Data**

The SELECT statement is used to retrieve data from a database. The syntax for SELECT is −

```
SQL> SELECT column_name, column_name, ...
    FROM table_name
    WHERE conditions;
```

The WHERE clause can use the comparison operators such as =, !=, <, >, <=,and >=, as well as the BETWEEN and LIKE operators.

**Example**

The following SQL statement selects the age, first and last columns from the Employees table, where id column is 100 −

```
SQL> SELECT first, last, age
    FROM Employees
    WHERE id = 100;
```

The following SQL statement selects the age, first and last columns from the Employees table where *first* column contains *Zara* −

```
SQL> SELECT first, last, age
    FROM Employees
    WHERE first LIKE '%Zara%';
```

**UPDATE Data**

The UPDATE statement is used to update data. The syntax for UPDATE is −

```
SQL> UPDATE table_name
    SET column_name = value, column_name = value, ...
    WHERE conditions;
```

The WHERE clause can use the comparison operators such as =, !=, <, >, <=,and >=, as well as the BETWEEN and LIKE operators.

**Example**

The following SQL UPDATE statement changes the age column of the employee whose id is 100 −

```
SQL> UPDATE Employees SET age=20 WHERE id=100;
```

**DELETE Data**

The DELETE statement is used to delete data from tables. The syntax for DELETE is −

```
SQL> DELETE FROM table_name WHERE conditions;
```

The WHERE clause can use the comparison operators such as =, !=, <, >, <=,and >=, as well as the BETWEEN and LIKE operators.

**Example**

The following SQL DELETE statement deletes the record of the employee whose id is 100 −

```
SQL> DELETE FROM Employees WHERE id=100;
```

**Creating JDBC Application**

There are following six steps involved in building a JDBC application −

- **Import the packages:** Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using *import java.sql.** will suffice.
- **Register the JDBC driver:** Requires that you initialize a driver so you can open a communication channel with the database.
- **Open a connection:** Requires using the *DriverManager.getConnection()* method to create a Connection object, which represents a physical connection with the database.
- **Execute a query:** Requires using an object of type Statement for building and submitting an SQL statement to the database.
- **Extract data from result set:** Requires that you use the appropriate *ResultSet.getXXX()* method to retrieve the data from the result set.
- **Clean up the environment:** Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

**Install Java**

Install J2SE Development Kit 5.0 (JDK 5.0) from Java Official Site.

Make sure following environment variables are set as described below −

- **JAVA_HOME:** This environment variable should point to the directory where you installed the JDK, e.g. C:\Program Files\Java\jdk1.5.0.
- **CLASSPATH:** This environment variable should have appropriate paths set, e.g. C:\Program Files\Java\jdk1.5.0_20\jre\lib.
- **PATH:** This environment variable should point to appropriate JRE bin, e.g. C:\Program Files\Java\jre1.5.0_20\bin.

It is possible you have these variable set already, but just to make sure here's how to check.

- Go to the control panel and double-click on System. If you are a Windows XP user, it is possible you have to open Performance and Maintenance before you will see the System icon.
- Go to the Advanced tab and click on the Environment Variables.
- Now check if all the above mentioned variables are set properly.

**Install Database**

The most important thing you will need, of course is an actual running database with a table that you can query and modify.

Install a database that is most suitable for you. You can have plenty of choices and most common are −

- **MySQL DB:** MySQL is an open source database. You can download it from MySQL Official Site. We recommend downloading the full Windows installation.
  In addition, download and install MySQL Administrator as well as MySQL Query Browser. These are GUI based tools that will make your development much easier.
  Finally, download and unzip MySQL Connector/J (the MySQL JDBC driver) in a convenient directory. For the purpose of this tutorial we will assume that you have installed the driver at C:\Program Files\MySQL\mysql-connector-java-5.1.8.
  Accordingly, set CLASSPATH variable to C:\Program Files\MySQL\mysql-connector-java-5.1.8\mysql-connector-java-5.1.8-bin.jar. Your driver version may vary based on your installation.
- **PostgreSQL DB:** PostgreSQL is an open source database. You can download it from PostgreSQL Official Site.
  The Postgres installation contains a GUI based administrative tool called pgAdmin III. JDBC drivers are also included as part of the installation.
- **Oracle DB:** Oracle DB is a commercial database sold by Oracle . We assume that you have the necessary distribution media to install it.
  Oracle installation includes a GUI based administrative tool called Enterprise Manager. JDBC drivers are also included as a part of the installation.

**Program:-**

```
import java.sql.*;
public class FirstExample
{
  static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
  static final String DB_URL = "jdbc:mysql://localhost/EMP";
  static final String USER = "username";
  static final String PASS = "password";
  public static void main(String[] args)
 {
  Connection conn = null;
  Statement stmt = null;
  try
   {
    Class.forName("com.mysql.jdbc.Driver");
    System.out.println("Connecting to database...");
    conn = DriverManager.getConnection(DB_URL,USER,PASS);
```

```
    System.out.println("Creating statement...");
    stmt = conn.createStatement();
    String sql;
    sql = "SELECT id, first, last, age FROM Employees";
    ResultSet rs = stmt.executeQuery(sql);
    while(rs.next()
    {
       int id  = rs.getInt("id");
       int age = rs.getInt("age");
       String first = rs.getString("first");
       String last = rs.getString("last");
       System.out.print("ID: " + id);
       System.out.print(", Age: " + age);
       System.out.print(", First: " + first);
       System.out.println(", Last: " + last);
    }
    rs.close();
    stmt.close();
    conn.close();
  }
   catch(SQLException se)
   {
      se.printStackTrace();
   }
   catch(Exception e)
  {
       e.printStackTrace();
   }
   finally
    {
    try
    {
      if(stmt!=null)    stmt.close();
    }
   catch(SQLException se2)
    {
    }
   try
    {
      if(conn!=null)      conn.close();
    }
     catch(SQLException se)
    {
      se.printStackTrace();
    }
  }
  System.out.println("Goodbye!");
 }
}
```

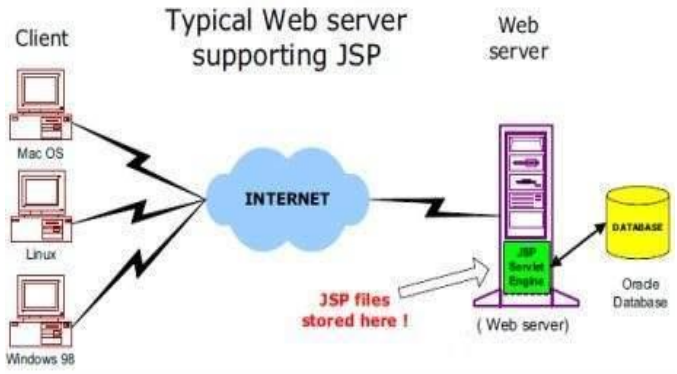
**C:\>javac FirstExample.java**
**C:\>**
**When you run FirstExample, it produces the following result −**
**C:\>java FirstExample**
**Connecting to database...**
**Creating statement...**
**ID: 100, Age: 18, First: Zara, Last: Ali**
**ID: 101, Age: 25, First: Mahnaz, Last: Fatma**
**ID: 102, Age: 30, First: Zaid, Last: Khan**
**ID: 103, Age: 28, First: Sumit, Last: Mittal**

**8. Explain MVC architecture of JSP with neat diagrams.** (Nov/Dec 2016)

- Java Server Pages is a kind of server side scripting language that enables user to embed java code with HTML elements for the creation of dynamic, platform-independent method for building web apps
- JSP = Java + HTML + servlet
- JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
- JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases
- JavaServer Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application.
- Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.
- Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

**Architecture**

- The web server needs a JSP engine ie. container to process JSP pages.
- The JSP container is responsible for intercepting requests for JSP pages
- A JSP container works with the Web server to provide the runtime environment and other services a JSP needs.
- It knows how to understand the special elements that are part of JSPs.

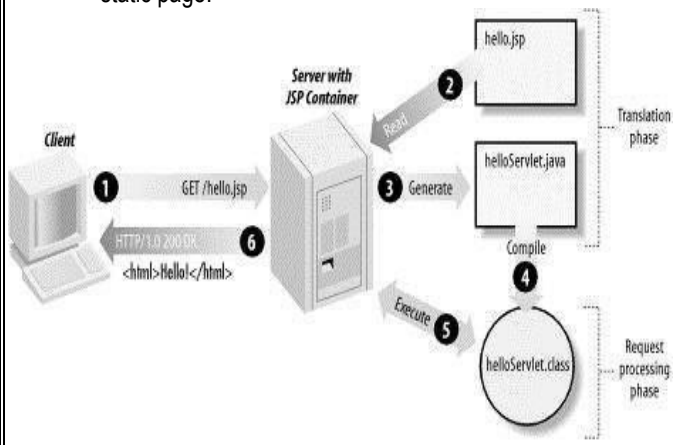



**JSP Processing**

The following steps explain how the web server creates the web page using JSP:

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content.
- This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.

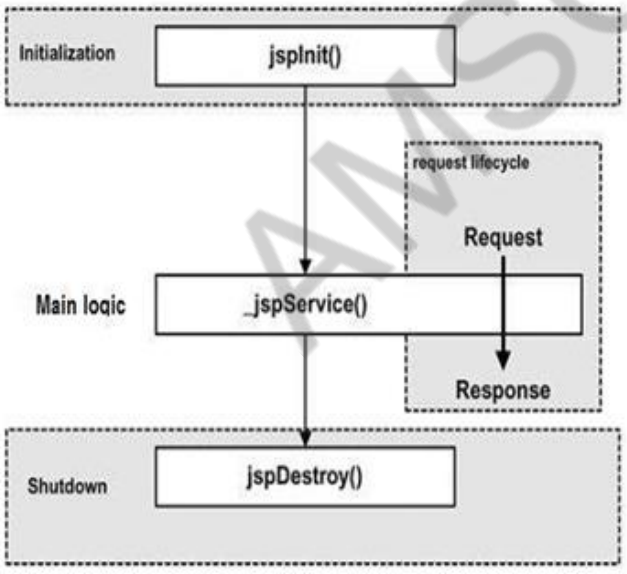
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.




A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.
The following are the paths followed by a JSP

- Compilation
- Initialization
- Execution
- Cleanup

The four major phases of JSP life cycle are very similar to Servlet Life Cycle and they are as follows:




**JSP Compilation**

- When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.

The compilation process involves three steps:

- Parsing the JSP.
- Turning the JSP into a servlet.
- Compiling the servlet.

**JSP Initialization**

- When a container loads a JSP it invokes the jspInit() method before servicing any requests. If you need to perform JSP-specific initialization, override the jspInit() method:

```
public void jspInit(){
  // Initialization code...
}
```

**JSP Execution**

- This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.
- Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the **_jspService()** method in the JSP.
- The _jspService() method takes an **HttpServletRequest** and an **HttpServletResponse** as its parameters as follows:

```
void _jspService(HttpServletRequest request,
                 HttpServletResponse response)
{
  // Service handling code...
}
```

- The _jspService() method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.

**JSP Cleanup**

- The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.
- The **jspDestroy()** method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files.

```
public void jspDestroy()
{
  // Your cleanup code goes here.
}
```

**Advantage of JSP over Servlet**

***1) Extension to Servlet***

JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

***2) Easy to maintain***

JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

***3) Fast Development: No need to recompile and redeploy***

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

***4) Less code than Servlet***

In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

**Life cycle of a JSP Page**

The JSP pages follows these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (class file is loaded by the classloader)
- Instantiation (Object of the Generated Servlet is created).
- Initialization ( jspInit() method is invoked by the container).
- Reqeust processing ( _jspService() method is invoked by the container).
- Destroy ( jspDestroy() method is invoked by the container).

**Running JSP page**

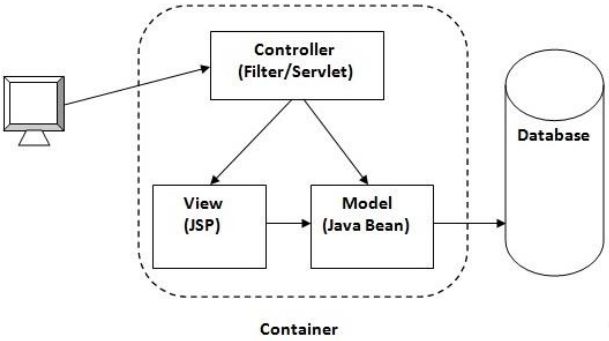
Follow the following steps to execute this JSP page:

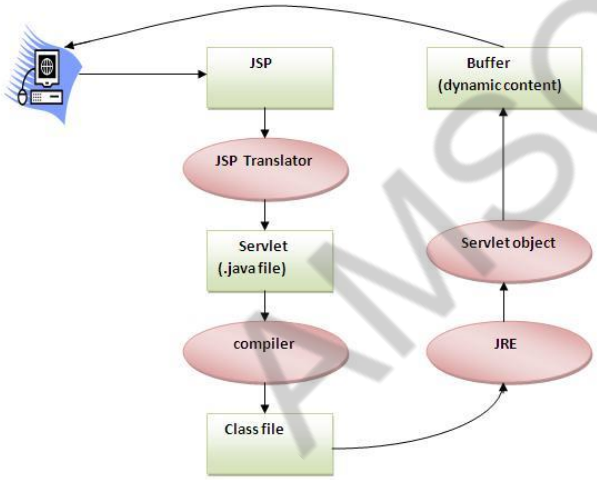
- Start the server
- put the jsp file in a folder and deploy on the server
- visit the browser by the url http://localhost:portno/contextRoot/jspfile e.g. http://localhost:8888/myapplication/index.jsp

- **MVC** stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data.
- **Controller** acts as an interface between View and Model. Controller intercepts all the incoming requests.
- **Model** represents the state of the application i.e. data. It can also have business logic.
- **View** represents the presentaion i.e. UI(User Interface).

**Advantage of MVC (Model 2) Architecture**

1. Navigation Control is centralized
2. Easy to maintain the large application




```
<html>
<body>
<% out.print(2*5); %>
</body>
</html>
```

**O/p:-**

**It will print 10 on the browser**

**9. Explain JSP scripting elements with examples**

**JSP Scripting elements**

Scripting elements provides the ability to insert java code inside the jsp.

- scriptlet tag
- expression tag
- declaration tag

**JSP scriptlet tag**

A scriptlet tag is used to execute java source code in JSP

**Example**

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

**File: index.html**

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

**File: welcome.jsp**

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</form>
</body>
</html>
```

**JSP expression tag**

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

**Example of JSP expression tag that prints the user name**

In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username.

**File: index.jsp**

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname"><br/>
<input type="submit" value="go">
</form>
</body>
</html>
```

**File: welcome.jsp**

```
<html>
<body>
<%= "Welcome "+request.getParameter("uname") %>
</body>
</html>
```

**JSP Declaration Tag**

- The **JSP declaration tag** is used *to declare fields and methods*.
- The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.
- So it doesn't get memory at each request.

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

**10. Explain JSP directives with examples**

- The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.
  - page directive
  - include directive
  - taglib directive

**JSP page directive**

The page directive defines attributes that apply to an entire JSP page.

```
<html>
<body>
<%@ page import="java.util.Date" %>
Today is: <%= new Date() %>
</body>
</html>
```


javatpoint.com

**Jsp Include Directive**

- The include directive is used to include the contents of any resource it may be jsp file, html file or text file.
- The include directive includes the original content of the included resource at page translation time
- the jsp page is translated only once so it will be better to include static resource.

```
<html>
<body>
 <%@ include file="header.html" %>
 Today is: <%= java.util.Calendar.getInstance().getTime() %>
 </body>
</html>
```

**JSP Taglib directive**

- The JSP taglib directive is used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define tags.
- In the custom tag section we will use this tag so it will be better to learn it in custom tag.

```
<html>
 <body>
 <%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %>
 <mytag:currentDate/>
 </body>
</html>
```

**11. Explain JSTL tags with examples.**

**JSTL (JSP Standard Tag Library)**

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

**Advantage of JSTL**

1. **Fast Developement** JSTL provides many tags that simplifies the JSP.
2. **Code Reusability** We can use the JSTL tags in various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

**JSTL Tags**

There JSTL mainly provides 5 types of tags:

| Tag Name | Description |
|---|---|
| **Core tags** | The JSTL core tag provide variable support, URL management, flow control etc. The url for the core tag is **http://java.sun.com/jsp/jstl/core** . The prefix of core tag is **c**. |
| **Function tags** | The functions tags provide support for string manipulation and string length. The url for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**. |
| **Formatting tags** | The Formatting tags provide support for message formatting, number and date formatting etc. The url for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**. |
| **XML tags** | The xml sql tags provide flow control, transformation etc. The url for the xml tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**. |
| **SQL tags** | The JSTL sql tags provide SQL support. The url for the sql tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**. |

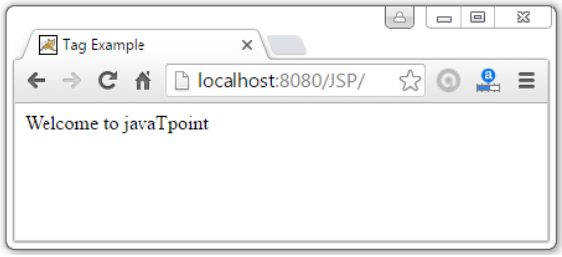
**JSTL Core Tags**

- The JSTL core tag provides variable support, URL management, flow control etc.
- The syntax used for including JSTL core library in your JSP is:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

| Tags | Description |
|---|---|
| **c:out** | It display the result of an expression, similar to the way <%=...%> tag work. |
| **c:import** | It Retrives relative or an absolute URL and display the contents to either a String in 'var',a Reader in 'varReader' or the page. |
| **c:set** | It sets the result of an expression under evaluation in a 'scope' variable. |
| **c:remove** | It is used for removing the specified scoped variable from a particular scope. |
| **c:if** | It is conditional tag used for testing the condition and display the body content only if the expression evaluates is true. |
| **c:param** | It adds a parameter in a containing 'import' tag's URL. |
| **c:redirect** | It redirects the browser to a new URL and supports the context-relative URLs. |
| **c:url** | It creates a URL with optional query parameters. |

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Tag Example</title>
</head>
<body>
<c:out value="${'Welcome to javaTpoint'}"/>
</body>
</html>
```



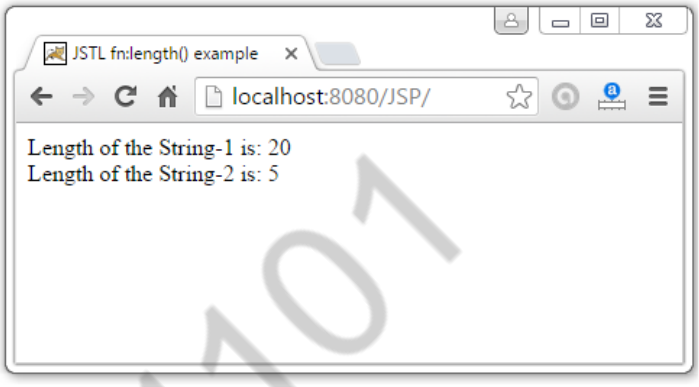

**JSTL Function Tags**

- The JSTL function provides a number of standard functions, most of these functions are common string manipulation functions.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

| JSTL Functions | Description |
|---|---|
| fn:contains() | It is used to test if an input string containing the specified substring in a program. |
| fn:indexOf() | It returns an index within a string of first occurrence of a specified substring. |
| fn:trim() | It removes the blank spaces from both the ends of a string. |
| fn:split() | It splits the string into an array of substrings. |
| fn:toLowerCase() | It converts all the characters of a string to lower case. |
| fn:toUpperCase() | It converts all the characters of a string to upper case. |
| fn:substring() | It returns the subset of a string according to the given start and end position. |
| fn:substringAfter() | It returns the subset of string after a specific substring. |
| fn:substringBefore() | It returns the subset of string before a specific substring. |
| fn:length() | It returns the number of characters inside a string, or the number of items in a collection. |
| fn:replace() | It replaces all the occurrence of a string with another string sequence. |

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>JSTL fn:length() example</title>
</head>
<body>
<c:set var="str1" value="This is first string"/>
<c:set var="str2" value="Hello"/>
Length of the String-1 is: ${fn:length(str1)}<br>
Length of the String-2 is: ${fn:length(str2)}
</body>
</html>
```

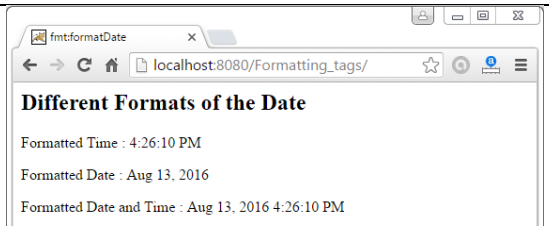



**JSTL Formatting tags**

- The formatting tags provide support for message formatting, number and date formatting etc.
- The url for the formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt.**
- The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers.
- <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

| Formatting Tags | Descriptions |
|---|---|
| fmt:timeZone | It specifies a parsing action nested in its body or the time zone for any time formatting. |
| fmt:formatNumber | It is used to format the numerical value with specific format or precision. |
| fmt:parseDate | It parses the string representation of a time and date. |
| fmt:formatDate | It formats the time and/or date using the supplied pattern and styles. |

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<head> <title>fmt:formatDate</title>  </head>
<body>
<h2>Different Formats of the Date</h2>
<c:set var="Date" value="<%=new java.util.Date()%>" />
<p> Formatted Time :
<fmt:formatDate type="time" value="${Date}" />
</p>
<p> Formatted Date :
<fmt:formatDate type="date" value="${Date}" />
</p>
<p> Formatted Date and Time :
<fmt:formatDate type="both" value="${Date}" />
</p>
</body></html>
```

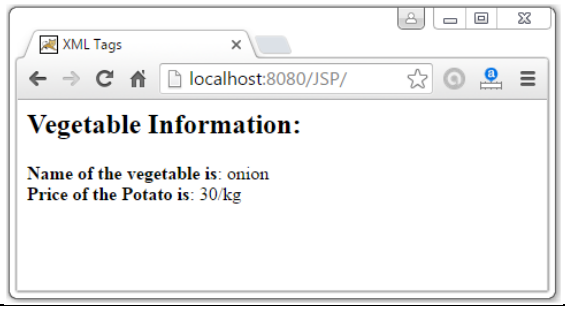



## JSTL XML tags

The JSTL XML tags are used for providing a JSP-centric way of manipulating and creating XML documents.

| XML Tags | Descriptions |
|---|---|
| x:out | Similar to <%= ... > tag, but for XPath expressions. |
| x:parse | It is used for parse the XML data specified either in the tag body or an attribute. |
| x:set | It is used to sets a variable to the value of an XPath expression. |
| x:if | It is used for evaluating the test XPath expression and if it is true, it will processes its body content. |
| x:transform | It is used in a XML document for providing the XSL(Extensible Stylesheet Language) transformation. |

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<html>
<body>
<h2>Vegetable Information:</h2>
<c:set var="vegetable">
<vegetables>
   <vegetable>
     <name>onion</name>
     <price>40/kg</price>
   </vegetable>
 <vegetable>
     <name>Potato</name>
     <price>30/kg</price>
   </vegetable>
 <vegetable>
     <name>Tomato</name>
     <price>90/kg</price>
   </vegetable>
</vegetables>
</c:set>
<x:parse xml="${vegetable}" var="output"/>
<b>Name of the vegetable is</b>:
<x:out select="$output/vegetables/vegetable[1]/name" /><br>
<b>Price of the Potato is</b>:
<x:out select="$output/vegetables/vegetable[2]/price" />
</body> </html>
```

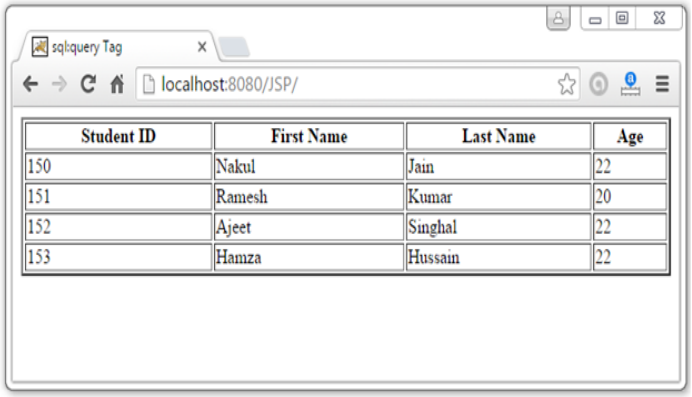



## JSTL SQL Tags

- The JSTL sql tags provide SQL support.
- The url for the sql tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

| SQL Tags | Descriptions |
|---|---|
| sql:setDataSource | It is used for creating a simple data source suitable only for prototyping. |
| sql:query | It is used for executing the SQL query defined in its sql attribute or the body. |
| sql:update | It is used for executing the SQL update defined in its sql attribute or in the tag body. |
| sql:transaction | It is used to provide the nested database action with a common connection. |

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
 <html>
<body>
<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/test"
    user="root"  password="1234"/>

<sql:query dataSource="${db}" var="rs">
SELECT * from Students;
</sql:query>

<table border="2" width="100%">
<tr>
<th>Student ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Age</th>
</tr>
<c:forEach var="table" items="${rs.rows}">
<tr>
<td><c:out value="${table.id}"/></td>
<td><c:out value="${table.First_Name}"/></td>
<td><c:out value="${table.Last_Name}"/></td>
<td><c:out value="${table.Age}"/></td>
</tr>
</c:forEach>
</table>
</body></html>
```



# UNIT 4 - PHP AND XML

## Part – A

**1. Define PHP** (Nov/Dec 2016)

- PHP is defined as a server side scripting language that is mainly used for form handling and database access.
- PHP stands for Hypertext Pre Processor
- It was invented in 1994 by Rasmus Lerdorf
- It is the most popular scripting language in web
- It is a FOSS

**2. Mention the features of PHP**

- Embedded inside HTML, easy to develop
- FOSS
- Easy to manage dynamic content, database, session tracking
- Supports many protocols such as LDAP, IMAP, POP3 (Nov/Dec 2017)
- Supports many databases such as MS SQL server, Oracle, SyBase, PostgreSQL, MySQL, etc
- As much forgiving as possible
- Simple like C and HTML

**3. List the uses of PHP**

- To perform system functions such as file create, open, close, read, write, etc
- To handle forms, gather data from files, save data to a file, send email, etc
- To add, delete, modify database contents
- To access and set cookies and variables
- To restrict users from page access
- To encrypt data

**4. What are the rules in PHP?**

- White space insensitive
- Case sensitive
- Each statement ends with semi colon
- Expressions are combination of tokens
- Braces creates blocks
- $ is used before variables
- Save file as .php and access it from localhost server

**5. List the data types in PHP.**

| Simple types | Compound types | Miscellaneous |
|---|---|---|
| **1.** Integer<br>2. Double<br>3. Boolean<br>4. Null<br>5. String | **6.** Arrays<br>7. Objects | 8. Resourcess |

**6. Differentiate echo and print in PHP**

| echo | Print |
|---|---|
| No return value | Return value is 1 |
| Cant be used in expression | Can be used in expression |
| Can take many parameters | Can take 1 parameter |
| Faster than print | Slower than echo |

**7. explain foreach loop in PHP**

```
<?php
$a = array ( 1,2,3);
foreach($a as $i)
{
echo "$i <br> ";
}
```

**Output:-**
**1**
**2**
**3**

- Foreach loop is very much useful in iterations
- The name itself suggests, for each iteration in for loop, it performs the operations.
- Iteration variable goes through all the elements in the array

**8. What are cookies in PHP?**

- A cookie is a name-value pair that is stored on client computer for tracking purpose
- It is created by some software on the server
- In every HTTP communication between client and server, there is a header, within that, cookies are present
- PHP supports cookies
- Server puts cookie into client machine on first visit.
- When that client machine sends request to that server next time, server identifies which user it is, from where the request arrives, from what device the request comes

**9. Define XML**

- Xtensible Markup language
- XML is defined as a text based mark up language derived from Standard Generalised Markup Language
- Developed by W3C in Feb 1998 to overcome HTML
- A web script that contains XML tags is called XML document
- It is a mark up language that defines set of rules for encoding documents in a format that is both human readable and machine readable
- It is not a programming language

**10. Mention the features of XML**

- **Extensible:** user defined tags
- **Secure:** Carries data, but does not shows it
- **Public standard:** developed by W3C
- Simplifies HTML for large websites
- To offload and reload databases
- To store and arrange data
- Can be merged with CSS
- Any data can be expressed in XML

**11. What are the rules in XML declaration?**

- If XML declaration is present, it should be placed 1st
- If XML declaration is present, it must contain version no
- Parameter name and parameter value is case sensitive
- Correct order is: version, encoding, standalone
- Either ' or " can be used
- XML declaration has no close tag → </?xml> is wrong

**12. What are the types of XML tags?**

- **Start tag:** starting point of user defined tag <username>
- **End tag:** every start tag must have end tag </username>
- **Empty tag:** An element that has no content <br><hr>

**13. Differentiate XML and HTML**

| XML | HTML |
|---|---|
| To transport and store data | To display data |
| Focus on what data it is | Focus on how the data looks |
| Provides framework for defining mark up languages | It is mark up language itself |
| It is neither a programming language, nor a presentation language | It is a presentation language |
| Case sensitive | Case insensitive |
| User defined tags | No user defined tags |
| Closing of each tag is mandatory | Not necessary of closing all the opened tags |
| Preserve white space | Does not. |

**14. What are the advantages of XML?** (Apr/May 2017)

- Human readable, easy to understand
- Language neutral
- Tree structured, understood in simpler manner
- Independent of hardware, software and OS
- User defined tags

**15. Mention the uses of XML**

- To display meta contents
- To exchange data between applications and databases
- To store any kind of complex data in simpler way
- A java program can generate XML and can be parsed by Perl

**16. What are the building blocks of XML?**

- Elements (start and end tags)
- Attributes (flag type="true")
- CDATA (Character DATA, parsed by XML parser)
- PCDATA ( Parsed Character DATA, i.e., text)

**17. What is DTD?**

- DTD stands for Document Type Declaration
- DTD is used to define basic building block of any XML document
- We can specify element types, attributes and relationship with one another
- To specify set of rules for structuring data in XML

**18. What is XML schema?**

- It is also known as XML schema Definition (XSD)
- To represent structure of XML document
- To describe and validate structure and content of XML
- Defines elements, attributes and data types
- To define building blocks of XML

Ex: <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>

**19. Define XML DOM**

- A DOM is a collection of nodes in tree hierarchy
- It is a set of platform independent and language neutral API that describes how to access and manipulate information in XML
- It is used for Loading, accessing, deleting XML

**20. Define XML parser**

- XML parser is a software library or a package that gives interface for client apps to work with XML
- It checks for proper format of XML document and validate XML documents
- To parse the given XML document

**21. Differentiate DOM and SAX**

| DOM | SAX |
|---|---|
| Document Object model | Simple API for XML |
| Tree based parsing to parse the XML document | Event based parsing to parse the XML document |
| Entire XML is stored in memory before actual processing | Parsing is done by sequence of events |
| Useful for smaller apps | Useful for large apps |
| Simple and less memory needed | Complex and more memory needed |
| We can insert or delete a node | We can insert or delete a node |
| Traverse in any direction | Top-down traversing |

**22. What are the rules of a well formed XML?**

- Non DTD files should have predefine character entity for amp(&), apos('), gt(>), lt(<), quot(double quotes)
- Inner tag must close before outer tag
- It must have only one attribute in start tag
- Entities other than amp, apos, gt, lt, quot should be declared

**23. What is XSL?**

- XML concentrates on structure of information
- W3C has published 2 recommendations for style sheets → CSS and XSL
- XSL = XML Style sheet Language
- To transform a document before display
- For advanced style information

**24. What are the parts of XSL?**

- **XSLT**: XSL Transformation, to transform XML
- **XPath:** a language for navigating XML
- **XSL-FO:** XSL-Formatting Objects, for formatting XML

**25. What is XSLT?**

- XSLT is a language to specify transformation of XML documents
- It takes XML document, transforms it into another XML document
- It is XML related technology to manipulate and transform XML documents
- To define XML transformations and presentations

**26. Define newsfeed**

- News feeds are an example of automated syndication
- It allows info to be automatically updated on sites, emailed to users, etc
- It can provide updated news, stock market shares, cricket scores, etc.

## Part – B

**1. Explain the operators in PHP with examples.**

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators
- Increment/decrement operators

**Arithmetic operators**

```
<?php
  $a = 10;
  $b = 20;
    echo $a+$b."<br>";
    echo $a-$b."<br>";
    echo $a*$b."<br>";
    echo $a/$b."<br>";
    echo $a%$b."<br>";
?>
```

**o/p:-**

```
30
-10
200
0.5
```

**Comparison operators**

```
<?php
    $a = 10;
    $b = 20;

    if( $a == $b )    echo "a is equal to b<br/>";
    else              echo "a is not equal to b<br/>";

    if( $a > $b )     echo "a is greater than  b<br/>";
    else              echo "a is not greater than b<br/>";

    if( $a < $b )     echo "a is less than  b<br/>";
    else              echo "a is not less than b<br/>";

    if( $a != $b )    echo "a is not equal to b<br/>";
    else              echo "a is equal to b<br/>";

    if( $a >= $b )    echo "a is either greater than or equal to b<br/>";
    else              echo "a is neither greater than nor equal to b<br/>";

    if( $a <= $b )    echo "a is either less than or equal to b<br/>";
    else              echo "a is neither less than nor equal to b<br/>";
?>
```

**o/p:-**

```
a is not equal to b
a is not greater than b
a is less than b
a is not equal to b
a is neither greater than nor equal to b
a is either less than or equal to b
```

**logical operators**

```
<?php
      $a = 10;
      $b = 0;

      if( $a && $b ) echo "Both a and b are true<br/>";
      else           echo "Either a or b is false<br/>";

      if( $a and $b ) echo "Both a and b are true<br/>";
      else            echo "Either a or b is false<br/>";

      if( $a || $b )  echo "Either a or b is true<br/>";
      else            echo "Both a and b are false<br/>";

    if( $a or $b ) echo "Either a or b is true<br/>";
      else           echo "Both a and b are false<br/>";

      $a = 10;
      $b = 20;

      if( $a )   echo "a is true <br/>";
      else       echo "a  is false<br/>";

      if( $b )   echo "b is true <br/>";
      else       echo "b  is false<br/>";

      if( !$a )  echo "a is true <br/>";
      else       echo "a  is false<br/>";


      if( !$b )  echo "b is true <br/>";
      else       echo "b  is false<br/>";
    ?>
```

**o/p:-**

```
Either a or b is false
Either a or b is false
Either a or b is true
Either a or b is true
a is true
b is true
a is false
b is false
```

**Assignment operators**

```
<?php
      $a = 10;
      $b = 20;
      echo $a+=$b,"<br>";
      echo $a-=$b,"<br>";
      echo $a*=$b,"<br>";
      echo $a/=$b,"<br>";
      echo $a%=$b,"<br>";
?>
```

**o/p:-**

```
30
10
200
10
```

**Conditional operator**

```
<?php
      $a = 10;
      $b = 20;
      $c= ($a > $b ) ? $a :$b;
      echo "$c<br>";
?>
```

**o/p**

```
20
```

**Increment/decrement operators**

```
<?php
      $a = 10;
      echo $a++,"<br>";
      echo ++$a,"<br>";
      echo $a--,"<br>";
      echo --$a,"<br>";
?>
```

**o/p:-**

```
10
12
```

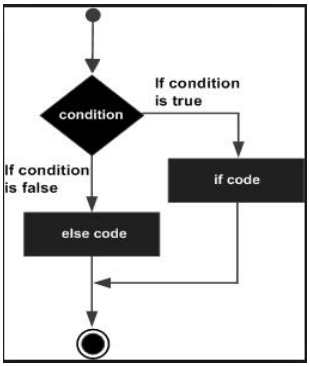
| Category | Operator | Associativity |
|---|---|---|
| Unary | ! ++ -- | Right to left |
| Multiplicative | * / % | Left to right |
| Additive | + - | Left to right |
| Relational | < <= > >= | Left to right |
| Equality | == != | Left to right |
| Logical AND | && | Left to right |
| Logical OR | \|\| | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= *= /= %= | Right to left |

**2. Explain PHP decision making statements with examples.**

- PHP supports following three decision making statements.
- The if, elseif ...else and switch statements are used to take decision based on the different condition.
- You can use conditional statements in your code to make your decisions.

**The If...Else Statement**

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.



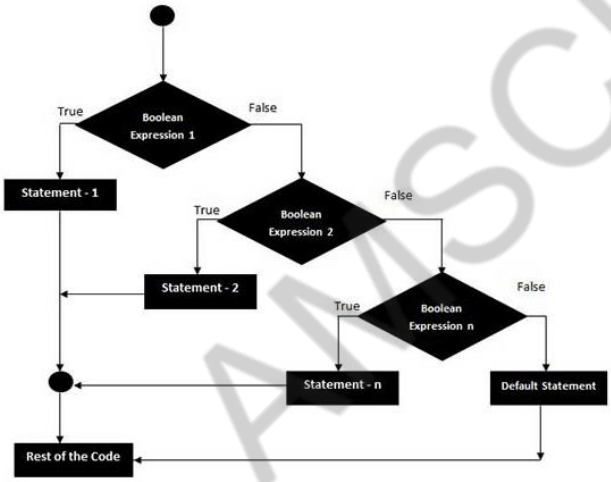

```
<?php
  $a=10;
  if ($a == 10)  echo "a value is 10";
  else  echo "a value is not 10";
?>
```

**o/p:-**

a value is 10

**. The If..Elself..Else Statement**

- If one of the several conditions are true, then use elseif statement



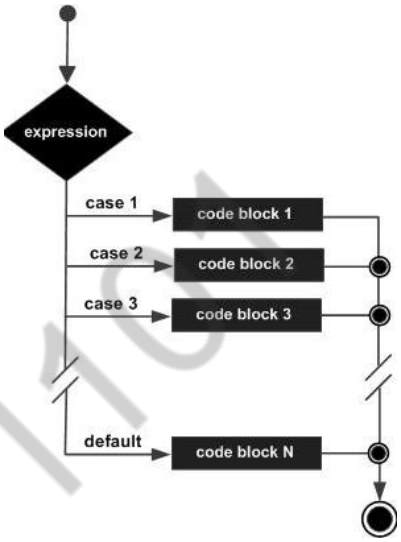

```
<?php
    $a=10;
    if ($a > 0) echo "a is positive";
   elseif ($a< 0) echo "a is negative";
   else echo "a is zero";
?>
```

**O/P:-**

a is positive

**The Switch Statement**

- If you want to select one of many blocks of code to be executed, use the Switch statement.
- The switch statement is used to avoid long blocks of if..elseif..else code.
- The value of the expression is then compared with the values for each case in the structure.
- If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.
- The **default** statement is used if no match is found.




```
<?php
$fav = "green";
switch ($fav)
{
   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!";
}
?>
```

**o/p:-**

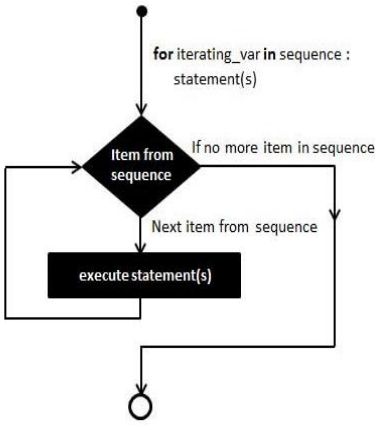
Your favorite color is green!

**3. Explain the Looping statements in PHP with examples. (Nov/Dec 2018)**

- Loops in PHP are used to execute the same block of code a specified number of times.
- PHP supports following four loop types.
- **for** − loops through a block of code a specified number of times.
- **while** − loops through a block of code if and as long as a specified condition is true.
- **do...while** − loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** − loops through a block of code for each element in an array.

**The for loop statement**

- The for statement is used when you know how many times you want to execute a statement or a block of statements.

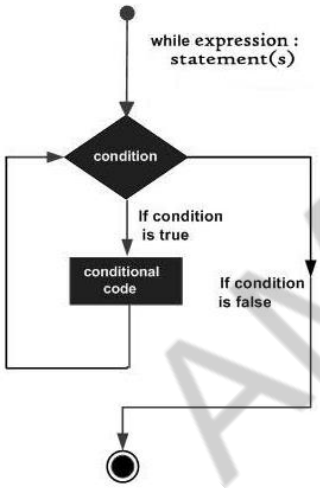



```
<?php
for( $i = 0; $i<5; $i++ ) {
   echo $i,"\n";
  }
 ?>
```

**o/p:**
01234

**The while loop statement**

- The while statement will execute a block of code if and as long as a test expression is true.
- If the test expression is true then the code block will be executed.
- After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

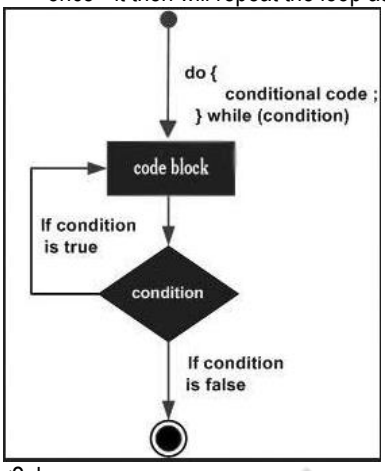



```
<?php
      $i = 0;
      while( $i < 10)
      {
                  echo $i;
                  $i++;
      }
?>
```

**o/p:**-
0123456789

**The do...while loop statement**

- The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.




```
<?php
      $i = 0;
      do
      {
            echo $i;
            $i++;
       }
      while( $i < 10 );
?>
```

**o/p:**-
0123456789

**The foreach loop statement**

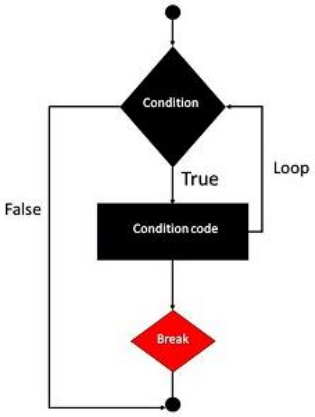
- The foreach statement is used to loop through arrays.
- For each pass the value of the current array element is assigned to $value
- array pointer is moved by one and in the next pass next element will be processed.

```
<?php
      $a = range(0,9);
      foreach( $a as $i )
            echo $i,"<br>";
?>
```

**o/p:**-
0123456789

**The break statement**

- The PHP **break** keyword is used to terminate the execution of a loop prematurely.
- The **break** statement is situated inside the statement block.
- If gives you full control and whenever you want to exit from the loop you can come out.
- After coming out of a loop immediate statement to the loop will be executed.

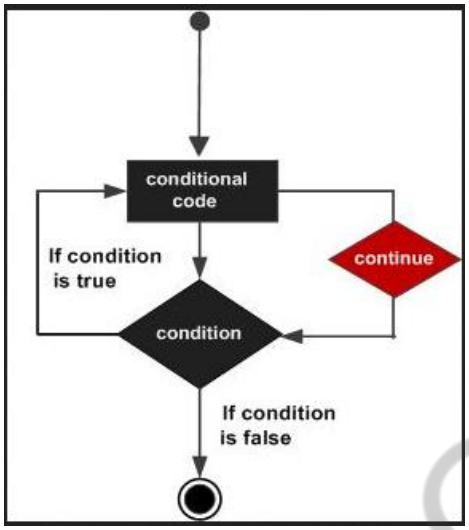


```
<?php
        for($i=0;$i<5;$i++)
        {
                if( $i == 3 )break;
                echo $i;
        }
?>
```

**o/p:-**
0123

**The continue statement**

- The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.
- Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test.
- For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.




```
<?php
    for($i=0;$i<5;$i++)
        {
                if( $i == 3 )continue;
                echo $i;
        }
?>
```

**o/p:-**
0124

**4. Explain functions in PHP with examples.** (Nov/Dec 2015)

- These functions allow you to interact with and manipulate arrays in various ways.
- Arrays are essential for storing, managing, and operating on sets of variables.
- Returns an array of the parameters.
- The parameters can be given an index with the => operator.
- PHP functions are similar to other programming languages.
- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.
- The real power of PHP comes from its functions; it has more than 1000 built-in functions.

**PHP User Defined Functions**

- Besides the built-in PHP functions, 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
        function writeMsg()
        {
           echo "Hello world!";
        }
        writeMsg();
?>
```

**O/p:-**
Helloworld

```
<?php
        function name($fname, $dob)
        {
                echo "$fname was born on $dob <br>";
        }
        name("Bala","12-2-91");
        name("Gopal","22-12-96");
?>
```

**o/p:-**
Bala was born on 12-2-91
Gopal was born on 22-12-96

```
<?php
        function sum($x, $y)
        {
           return $x + $y;
        }
        echo sum(10,20) . "<br>";
        echo sum(100,200) . "<br>";
?>
```

**O/p:-**
30
300

**PHP Built-in functions:-**

- PHP Array Functions
- PHP Calender Functions
- PHP Class/Object Functions
- PHP Character Functions
- PHP Date & Time Functions
- PHP Directory Functions
- PHP Error Handling Functions
- PHP File System Functions
- PHP MySQL Functions
- PHP Network Functions
- PHP ODBC Functions
- PHP String Functions
- PHP SimpleXML Functions
- PHP XML Parsing Functions

**PHP - Array Functions**

```
<?php
  $a = array("a"=>"Dog", "b"=>"Cat", "c"=>"Horse");
  print_r($a);
?>
```

**o/p**
Array ( [a] => Dog [b] => Cat [c] => Horse )

**array_chunk()**

- Chunks an array into **size** large chunks.
- The last chunk may contain less than **size** elements.

```
<?php
  $input_array = array('a', 'b', 'c', 'd', 'e');
  print_r(array_chunk($input_array, 2));
  print_r(array_chunk($input_array, 2, true));
?>
```

**o/p**

```
Array (
  [0] => Array (
    [0] => a
    [1] => b
  )

  [1] => Array (
    [0] => c
    [1] => d
  )

  [2] => Array (
    [0] => e
  )
)

Array
(
  [0] => Array (
    [0] => a
    [1] => b
  )

  [1] => Array (
    [2] => c
    [3] => d
  )

  [2] => Array (
    [4] => e
  )
)
```

**array_fill()**

- Fills an array with **num** entries of the value of the **value**parameter, keys starting at the **start_index** parameter.

```
<?php
  $a = array_fill(5, 6, 'apple');
  print_r($a)
?>
```

**o/p**

```
Array ( [5] => apple [6] => apple [7] => apple [8] => apple [9] => apple
[10] => apple )
```

**array_keys() and array_values()**

```
<?php
    $array = array("a"=>"green", "b"=>"brown", "c"=>"blue", "red");
    print_r(array_values($array));
    echo "<br>";
    print_r(array_keys($array));
?>
```

**o/p**

```
    Array ( [0] => green [1] => brown [2] => blue [3] => red )
    Array ( [0] => a [1] => b [2] => c [3] => 0 )
```

**array_merge()**

- Merges elements of one or more arrays together so that the values of one are appended to end of the previous one.
- If the input arrays have the same string keys, then the later value for that key will overwrite the previous one.

```
<?php
    $array1 = array("a"=>"Horse","b"=>"Cat","c"=>"Dog");
    $array2 = array("d"=>"Cow", "e"=>"elephant");
    print_r(array_merge($array1,$array2));
?>
```

**O/p:-**

```
    Array ( [a] => Horse [b] => Cat [c] => Dog [d] => Cow [e] =>
    elephant )
```

**array_push()**

- This function treats **array** as a stack, and pushes the passed variables var1, var2... onto the end of array. The length of array increases by the number of variables pushed.

```
<?php
        $array = array("0"=>"banana","1"=>"apple","3"=>"orange");
        print_r(array_push($array, "mango"));
        print_r("<br>");
        print_r($array );
?>
```

**o/p:-**

```
4
Array ( [0] => banana [1] => apple [3] => orange [4] => mango )
```

**array_pop()**

- This function pops and returns the last value of the **array**, shortening the array by one element.
- If array is empty (or is not an array), NULL will be returned

```
<?php
  $array = array("0"=>"banana","1"=>"apple","2"=>"orange");
  print_r($array);
  print_r("<br>");
  print_r(array_pop($array));
  print_r("<br>");
  print_r($array);
?>
```

**o/p:-**

```
Array ( [0] => banana [1] => apple [2] => orange )
orange
Array ( [0] => banana [1] => apple )
```

**Search() and sum()**

```
<?php
  $a = array(1,2,3,4);
  print_r(array_sum($a));
  echo "<br>";
  print_r(array_search(3, $a));
?>
```

**O/p:-**

10

2

**PHP – calendar functions**

- The calendar extension presents a series of functions to simplify converting between different calendar formats.
- The intermediary or standard it is based on is the Julian Day Count.
- The Julian Day Count is a count of days starting from January 1st, 4713 B.C.
- To convert between calendar systems, you must first convert to Julian Day Count, then to the calendar system of your choice.

**cal_days_in_month()**

- This function will return the number of days in the month of year for the specified calendar
- This function returns the day of the week. It can return a string or an integer depending on the mode.

```
<?php
        $num = cal_days_in_month(CAL_GREGORIAN,10, 2016);
        echo "There are $num days in October 2016";
        echo "<br>";
        $jd =
cal_to_jd(CAL_GREGORIAN,date("m"),date("d"),date("Y"));
        echo "today is ".(jddayofweek($jd,1));
        echo "<br>";
        echo "This is ".(jdmonthname($jd,1));
?>
```

**O/p:-**

There are 31 days in October 2016
today is Monday
This is October

**Character Functions**

- The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale.
- When called with an integer argument these functions behave exactly like their C counterparts from ctype.h
- Builtin support for ctype is available with PHP 4.3.0.
- Checks if all of the characters in the provided string, text, are alphanumeric.
- Checks if all of the characters in the provided string, text, are alphabetical.
- Checks if all of the characters in the provided string, text, are numerical. It checks only 1...9
- This function checks if all of the characters in the provided string, text, are lowercase letters.
- This function checks if all of the characters in the provided string, text, are uppercase characters.

```
<?php
  $strings = array('AB','cd','123','#');
  foreach ($strings as $check)
  {
    if (ctype_alnum($check))    echo "$check consists of letters or digits";
echo "<br>";
    if (ctype_alpha($check))    echo "$check consists of letters or digits";
echo "<br>";
    if (ctype_digit($check))    echo "$check consists of letters or digits";
echo "<br>";
    if (ctype_lower($check))    echo "$check consists of letters or digits";
echo "<br>";
    if (ctype_upper($check))    echo "$check consists of letters or digits";
echo "<br>";
  }
?>
```

**O/p:-**

AB consists of letters or digits
AB consists of letters or digits

AB consists of letters or digits
cd consists of letters or digits
cd consists of letters or digits

cd consists of letters or digits

123 consists of letters or digits
123 consists of letters or digits

**Date and Time functions**

```
<?php
  date_default_timezone_set('Asia/Kolkata');
  echo date("d-m-y"),"<br>";
  echo date("D-M-Y"),"<br>";
  echo date("h:i:s"),"<br>";
?>
```

**O/p:-**

17-10-16
Mon-Oct-2016
01:16:59

**5. Explain String manipulations in PHP with examples.**

**binhex()**

- It is used to convert primary data to hexadecimal representation

**Dechex**

- It is used to convert decimal to hexadecimal representation

**Chop()**

- It is used to removes whitespace and returns the modified string

**Chunk_split()**

- It is used to split a string into chunks

**Count_chars()**

- It is used to returns the information about character used in a string

**crc32()**

- It is used to calculates 32-bit CRC

**Crypt()**

- It is used to hashing the string

```
<?php
  $bin = "10001000";
  $dec=bindec($bin);
  $hex=dechex($dec);
  echo $dec; echo "<br>";
  echo $hex; echo "<br>";

  $name="evergreen";
  echo chop($name,"een"); echo "<br>";
  echo chunk_split($name,1,"-");echo "<br>";
  echo chunk_split($name,2,"-");echo "<br>";
  echo chunk_split($name,3,"-");echo "<br>";

  $str = crc32("bala");
  printf("%u\n",$str);

  $st = "balamurugan";
  echo "<br>";
 echo crypt($st,"key");
?>
```

**O/p:-**

136
88
evergr
e-v-e-r-g-r-e-e-n-
ev-er-gr-ee-n-
eve-rgr-een-
176370918

```
<?php
$a = array("I", "like", "PHP");
  $sentence = implode(" ",$a);
  for($i = 0; $i < count($a); $i++)
  {
     echo "$i = $a [$i]","<br>";
  }
  echo $sentence, "<br>" ;
  $sentence2 = "I Like green";
  $chunks = explode(" ", $sentence2);
  echo $sentence2,"<br>";
  echo $chunks[0],"<br>",$chunks[1],"<br>",$chunks[2];
  ?>
```

**O/p:-**

0 = Array [0]
1 = Array [1]
2 = Array [2]
I like PHP
I Like green
I
Like
green

**join()**

- It is alias of implode(),it returns string from the elements of an array

**ltrim()**

- It used to strip whitespace or other characters from the beginning of a string

**md5()**

- It is used to calculates the md5 hash of a string

**printf()**

- It returns output a formatted string

**rtrim()**

- It is used to remove the white spaces from end of the string3

**sha1()**

- It is used to calculate the sha1 hash of a string

**str_word_count()**

- It returns information about words used in a string

**strcmp()**

- It is used to compare two strings

**strlen()**

- It is used get string length.

```
<?php
  $a = array("I", "like", "sachin");
  echo join(" ",$a); echo "<br>";

  $aa = "Tamil is my mother tongue";
  echo str_word_count($aa); echo "<br>";

  $b = "   sachin    ";
  echo ltrim($b); echo "<br>";
  echo rtrim($b); echo "<br>";

  echo md5($b); echo "<br>";
  printf($b); echo "<br>";
  echo sha1($b); echo "<br>";

  echo strcmp("sachin","sachin");  echo "<br>";
  echo strlen("bala");
?>
```

**O/P:-**

```
I like sachin
5
sachin
sachin
81ea3915ec7b03821f571d75d121989b
sachin
76e0804b0b28b97e68f2a4f58e9e21b7d9bf6453
0
4
```

**6. What is XML? Explain its syntax rules with examples.** (Nov/Dec 2016)

**XML**

- Extensible Markup language

**Definition:-**

- **XML is a mark up language that defines set of rules for encoding documents in a format that is both human readable and machine readable**
- It is a text based mark up language derived from SGML
- It was introduced by W3C to overcome the problems in HTML
- Markup means, information added to a document that improves its meaning
- It is not a programming language
- It is stored in text file
- It is parsed by XML parser
- No predefined tags in XML, only user defined tags
- It is stricter than HTML, case sensitive

**Features of XML / Advantages / Uses**

- Simplify the creation of HTML documents for large sites
- To exchange information between organizations
- Offload and reload databases
- Store and arrange data
- Any type of data can be expressed in XML
- Suits well for commerce applications, scientific purposes, mathematics, chemical formulae
- It can be used in handheld devices, smartphones, etc
- Hardware, software and language independent

**Syntax rules**

- XML declaration
- References
- Tags and elements
- Attributes
- Text

**XML declaration**

- < ? xml version = "1.0" ?>
- "xml" should be in lower case
- Every XML document should begin with <?xml…>
- It must be the root element in all XML files

**Tags and elements**

- Tags are the building blocks of XML document
- It is also called XML nodes
- **<name>Bala</name>**

```
<person>
        <name>Bala</name>
        <phone>1234</phone>
</person>
```

**Attributes**

- To specify a property of an element
- It is a "name-value" pair
- An element can have more than 1 attributes
- <phone available="yes">1234</phone>

**References**

- To add additional information
- Begin with &
  - Entity reference
  - Character Reference

**Text**

- XML elements and attributes are case sensitive
- Start and end tag needs to be in same case
- To avoid encoding problems, use UTF-8 or UTF-16
- It is whitespace insensitive

**Example:-**

```
<?xml version="1.0"?>
<person>
        <name>Bala</name>
        <cell>1234</cell>
        <company>TCS</company>
</person>
```

**Rules for XML**

- If any XML declaration is present, put it in the first line
- Mention the version of XML
- Parameters and values are case sensitive
- Names are always in lower case
- Either " or ' can be used
- <?xml…?> has no close tag
- Only - , _ and . are allowed in elements
- Comment inside < ! - - this is a comment - - >
- No comments should be made before <?xml…..?>

**XML tags**

- Start tag
- End tag
- Empty tag (has no close tag)

**XML attributes**

- String
- Tokenized
- Enumerated

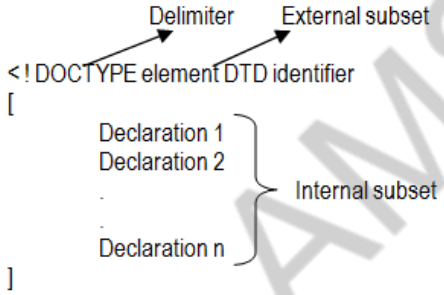
| XML | HTML |
|---|---|
| Software and hardware independent | Software and hardware dependent |
| To send and store data | To display data |
| Focus on what data is present | Focus on how data looks |
| It is a Framework for markup language definition | It is a mark up language |
| Case sensitive | Case insensitive |
| Transport data between app and database | Design client side web programs |
| Custom tags allowed | Only predefined tags |
| Open and close tags are strict | Not strict |
| White space insensitive | White space insensitive |
| Carry information | Display information |
| Dynamic | static |

**7. Explain XML DTD with examples each.**

**DTD in XML**

- Document Type Definition
- To define the type of the document
- A DTD is attached to a document
- To describe the XML

**Syntax**:-




- The **DTD** starts with <!DOCTYPE delimiter.
- An **element** tells the parser to parse the document from the specified root element.
- **DTD identifier** is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset.**
- **The square brackets [ ]** enclose an optional list of entity declarations called *Internal Subset*.

**Internal DTD**

- Elements are declared within XML
- DTD is stored within the XML file itself.
- Set stand alone attribute = "yes"

**sample.xml**

```
<? xml version = "1.0" encoding = "utf-8" standalone="yes"?>
< ! DOCTYPE address
        [
                < ! ELEMENT address(name, phone,
company)>
                < ! ELEMENT name(#PCDATA)>
                < ! ELEMENT phone(#PCDATA)>
                < ! ELEMENT company(#PCDATA)>
        ]
>
<address>
<name>Bala</name>
        <phone>1234</phone>
        <company>TCS</company>
</address>
```

**Note:-**

CDATA → Character Data, this data is parsed by the XML parser
PCDATA →Parsed Character Data, plain text
# →Delimiter

**External DTD**

- DTD is stored in a separate file called "sample.dtd"
- Set stand alone attribute = "no"

**sample.xml**

```
<? xml version = "1.0" encoding = "utf-8" standalone="yes"?>
< ! DOCTYPE address SYSTEM "address.dtd">
<address>
        <name>Bala</name>
        <phone>1234</phone>
        <company>TCS</company>
</address>
```

**address.dtd**

```
< ! DOCTYPE address
        [
                < ! ELEMENT address(name, phone, company)>
                < ! ELEMENT name(#PCDATA)>
                < ! ELEMENT phone(#PCDATA)>
                < ! ELEMENT company(#PCDATA)>
        ]
>
```

**Advantages of DTD**

- XML processor enforces structure, as defined in DTD
- Application is accessed easily in document structure
- DTD gives hint to XML processor
- Reduces size of document

**8. Explain XML Schema with necessary examples** (Nov/Dec 2015)

- XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.
- 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".
- XSD = XML Schema Definition Language
- Implemented in 2001 by W3C
- Allows developers to use different data types

**Definition Types**

- You can define XML schema elements in following ways:

i) **Simple Type -** Simple type element is used only in the context of the text. Some of predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example:

```
<xs:element name="phone_number" type="xs:int" />
```

ii) **Complex Type -** A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example:

```
<xs:element name="Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name"
type="xs:string" />
      <xs:element name="company"
type="xs:string" />
      <xs:element name="phone"
type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

iii) **Global Types -** With global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the *person* and *company* for different addresses of the company. In such case, you can define a general type as below:

```
<xs:element name="AddressType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"
/>
      <xs:element name="company"
type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Now let us use this type in our example as below:

```
<xs:element name="Address1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address"
type="AddressType" />
      <xs:element name="phone1" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Address2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address"
type="AddressType" />
      <xs:element name="phone2" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Instead of having to define the name and the company twice (once for *Address1* and once for *Address2*), we now have a single definition.
- This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

**Step 2: student.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<contact xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
SchemaLocation="student.xsd">
        <name>Bala></name>
        <company>TCS</company>
        <phone>1234</phone>
<contact>
```

**Step 3: Open the Xml file in browser**

**o/p:-**

```
<contact>
        <name>Bala></name>
        <company>TCS</company>
        <phone>1234</phone>
<contact>
```

**XML Schemas are More Powerful than DTD**

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

**Purpose of XML Schema**

- With XML Schema, your XML files can carry a description of its own format.
- With XML Schema, independent groups of people can agree on a standard for interchanging data.
- With XML Schema, you can verify data.

**XML Schemas Support Data Types**

- It is easier to describe document content
- It is easier to define restrictions on data
- It is easier to validate the correctness of data
- It is easier to convert data between different data types

**XML Schemas use XML Syntax**

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT

**9. Explain the data types in XML Schema with example each.**

- **String**
- **Numeric**
- **Date**
- **Boolean**

**i) <xs:string> data type**

- The <xs:string> data type can take characters, line feeds, carriage returns, and tab characters.
- The XML processor does not replace line feeds, carriage returns, and tab characters in the content with space and keep them intact.
- For example, multiple spaces or tabs are preserved during display.

**Example:-**

**Step 1: string.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
        <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                <xs:element name="contact" type="xs:string"/>
        </xs:schema>
```

**Step 2: string.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
        <contact
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" SchemaLocation="string.xsd">
                Balamurugan
    </contact>
```

**Step 3: Validate in Xml validator**

**o/p:-**

This XML document is valid

**ii) <xs:date> data type**

The <xs:date> data type is used to represent date in YYYY-MM-DD format.

- **YYYY** − represents year
- **MM** − represents month
- **DD** − represents day

**Step 1: date.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
    <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="contact" type="xs:date"/>
    </xs:schema>
```

**Step 2: date.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
        <contact
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" SchemaLocation="date.xsd">
                2016-10-17
    </contact>
```

**Step 3: Validate in Xml validator**

**o/p:-**

This XML document is valid

**iii) <xs:numeric> data type**

- The <xs:decimal> data type is used to represent numeric values.
- It supports decimal numbers up to 18 digits.
- The <xs:integer> data type is used to represent integer values.

**Step 1: numeric.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
    <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="contact" type="xs:decimal"/>
    </xs:schema>
```

**Step 2: numeric.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
        <contact
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" SchemaLocation="numeric.xsd">
                93.5
            </contact>
```

**Step 3: Validate in Xml validator**

**o/p:-**

This XML document is valid

**iv) <xs:boolean> data type**

- The <xs:boolean> data type is used to represent true, false, 1 (for true) or 0 (for false) value.

**Step 1: boolean.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
    <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="contact" type="xs:boolean"/>
    </xs:schema>
```

**Step 2: boolean.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
        <contact
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" SchemaLocation="boolean .xsd">
            true
    </contact>
```

**Step 3: Validate in Xml validator**

**o/p:-**

This XML document is valid

**10. What is DOM? Explain with necessary examples.**

**DoM:-** (Nov/Dec 2015)

- Document Object Model
- The Document Object Model (DOM) is a W3C standard
- The DOM defines a standard for accessing and manipulating documents.
- The XML DOM presents an XML document as a tree-structure.
- The HTML DOM presents an HTML document as a tree-structure.
- ***"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 DOM is separated into 3 different parts / levels:*

- Core DOM - standard model for any structured document
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

**Example:-**

```
<?xml version="1.0"?>
    <userdata>
      <user1>
        <userno>001</userno>
        <username>Bala</username>
        <phonenumner>123456789</phonenumber>
        <address>Chennai</Chennai>
    </user1>
      <user2>
        <userno>002</userno>
        <username>Suresh</username>
        <phonenumner>987654321</phonenumber>
        <address>madurai</Chennai>
    </user2>
      <user3>
        <userno>003</userno>
        <username>arul</username>
        <phonenumner>1122334455</phonenumber>
        <address>Vellore</Chennai>
    </user3>
    </userdata>
```

**user.html**

```
<html>
        <body>
                <script type="text/javascript">
                        var xmldoc=new
                        ActiveXObject("Microsoft.XMLDOM");
                        xmldoc.load("user.xml");
                        var ele=xmldoc.documentElement;
                        var y=window.prompt("Enter User
                        Number");
                        var node=ele.childNodes.item(y-1);
                        for(var i=0;i<node.childNodes.length;i++)
                        {
                             var child=node.childNodes.item(i);
                            var val=child.firstChild;
                                document.write("<table border><tr><td
                              width=200><h2>"+child.nodeName+"<td
                            width=200><h2>"+val.nodeValue+"</tr>");
                        }
                </script>
        </body>
</html>
```

**O/P:-**

| | | | |
|---|---|---|---|
| 001 | Bala | 123456789 | Chennai |
| 002 | Suresh | 987654321 | Madurai |
| 003 | Arul | 1122334455 | Vellore |

**xmlDoc - the XML DOM object created by the parser.**

- **getElementsByTagName("title")[0]** - get the first <title> element
- **childNodes[0]** - the first child of the <title> element (the text node)
- **nodeValue** - the value of the node (the text itself)

**XML DOM Properties**

These are some typical DOM properties:

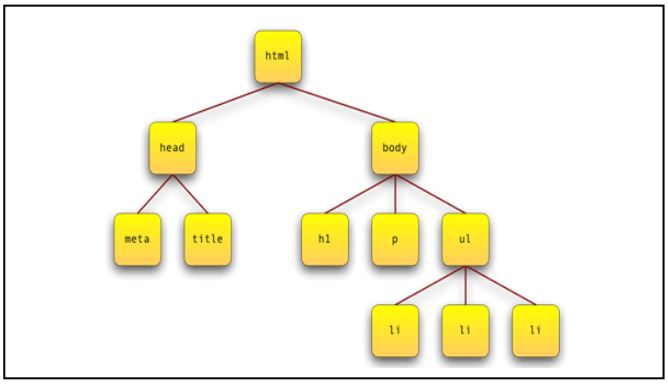
- x.nodeName - the name of x
- x.nodeValue - the value of x
- x.parentNode - the parent node of x
- x.childNodes - the child nodes of x
- x.attributes - the attributes nodes of x

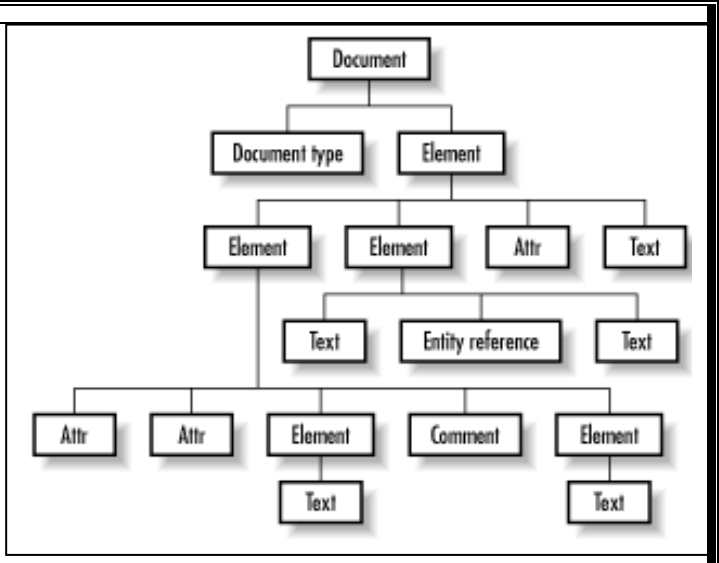
**XML DOM Methods**

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

**DOM Nodes**

- The entire document is a document node
- Every XML element is an element node
- The text in the XML elements are text nodes
- Every attribute is an attribute node
- Comments are comment nodes




**11. Explain XML parser in detail with Java program code.** (May/June 2016)

| DOM API | SAX API |
|---|---|
| Document Object Model | Simple API for XML |
| Tree based parsing | Event based parsing |
| Entire XML is stored in memory | Part of Xml is stored in memory |
| Requires less memory space | Requires more memory space |
| Useful for small apps | Uesful for large apps |
| Traverse in any direction | Top-down traversing |

**DOM based parsing:-**

**dom.java**

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
public class dom
{
        public static void main(String bala[])
        {
                try
                {
                System.out.println("Enter XML document name");
BufferedReader input = new BufferedReader( new InuptStreamReader(System.in));
                String filename = input.readLine();
                File fp = new File(filename);
                if(fp.exists())
                {
                        try
                        {
DocumentBuilderFactory dbf = new DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.new DocumentBuilder();
InputSource ips = new InputSource(filename);
Document doc = db.parse(ips);
System.out.println(filename + "is well formed");
                        }
                        catch(Exception e)
                        {
                         System.out.println("Not well formed");
                        System.exit(1);
                        }
                }
                else
                {
                        System.out.println("File not Found");
```

```
                }
            }
                catch(IOException ioe)
                    {
                        ioe.printStackTrace();
                    }
        }
}
```

**User.xml**

```
<?xml version="1.0"?>
        <userdata>
          <user1>
            <userno>001</userno>
            <username>Bala</username>
            <phonenumner>123456789</phonenumber>
            <address>Chennai</Chennai>
        </user1>
          <user2>
            <userno>002</userno>
            <username>Suresh</username>
            <phonenumner>987654321</phonenumber>
            <address>madurai</Chennai>
        </user2>
          <user3>
            <userno>003</userno>
            <username>arul</username>
            <phonenumner>1122334455</phonenumber>
            <address>Vellore</Chennai>
        </user3>
        </userdata>
```

**o/p:-**

```
        C:> javac dom.java
        C:> java dom
        Enter file name
        dom.xml
        dom.xml is well formed
```

**SAX based parsing**

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
public class dom
{
        public static void main(String bala[])
        {
                try
                {
                System.out.println("Enter XML document name");
BufferedReader input = new BufferedReader( new InuptStreamReader(System.in));
                String filename = input.readLine();
                File fp = new File(filename);
                if(fp.exists())
                {
                        try
                        {
        XMLReader reader = XMLReaderFactory.CreateXMLReader();
                                reader.parse(filename);
                System.out.println("filename + "is well formed");
                        }
                catch(Exception e)
                        {
                System.out.println("filename + "is not well formed");
                System.exit(1);
                        }
                        else
                        {
                                System.out.println("file not found");
                        }
                        catch(IOException ioe)
                        {
                                ioe.printStackTrace();
                        }
        }
}
```

**o/p:-**

```
        C:> javac sax.java
        C:> java sax
        Enter file name
        data.xml
        data.xml is well formed
```

**12. Explain in detail the elements of XSL with examples each.**

i) <xsl:template>
ii) <xsl:value-of>
iii) <xsl:for-each>
iv) <xsl:if>
v) <xsl:sort>
vi) <xsl:choose>

**<xsl:template>**

- to build template
- match attribute is used with template element
- match="/" defines the whole document

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
<details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
<html>
   <body>
      <table>
         <tr><th>Name</th><th>address</th><th>mark</th></tr>
         <tr><td>**</td><td>**</td><td>**</td></td>
         </table>
       </body>
     </html>
</xsl:template>
</xsl:stylesheet>
```

**o/p:-**

| name | address | marks |
|---|---|---|
| Bala | Chennai | 62 |

**<xsl:value-of>**

- to extract value of XML elements
- add that value to output stream of XSL transformation\

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
        <details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
        <html>
            <body>
                <table>
        <tr><th>Name</th><th>address</th><th>mark</th></tr>
        <tr><td><xsl:value-of select="student/details/name"/></td>
         <td><xsl:value-of select="student/details/address"/></td>
         <td><xsl:value-of select="student/details/mark"/></td>
        </tr>
                    </table>
                </body>
            </html>
        </xsl:template>
        </xsl:stylesheet>
```

**o/p:-**

| name | address | marks |
|---|---|---|
| Bala | Chennai | 62 |

**<xsl:for-each>**

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
<details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
        <html>
          <body>
              <table>
         <tr><th>Name</th><th>address</th><th>mark</th></tr>
         <xsl:for-each select="student/details">
        <tr><td><xsl:value-of select="name"/></td>
            <td><xsl:value-of select="address"/></td>
            <td><xsl:value-of select="mark"/></td>
        </tr>
        </table></body></html></xsl:template></xsl:stylesheet>
```

**O/P:-**

| name | address | marks |
|---|---|---|
| Bala | Chennai | 62 |
| Lokesh | Vellore | 95 |
| Gopal | Madurai | 88 |

**<xsl:if>**

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
        <details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
        <html>
          <body>
            <table>
         <tr><th>Name</th><th>address</th><th>mark</th></tr>
          <xsl:for-each select="student/details">
        <xsl:if test="marks>80">
            <tr><xsl:value-of select="name"/></td>
            <tr><xsl:value-of select="address"/></td>
            <tr><xsl:value-of select="std"/></td>
        </table></body></html></xsl:template></xsl:stylesheet>
```

**o/p**

| name | address | marks |
|---|---|---|
| Lokesh | Vellore | 95 |
| Gopal | Madurai | 88 |

**<xsl:sort>**

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
        <details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
        <html>
           <body>
              <table>
           <tr><th>Name</th><th>address</th><th>std</th></tr>
                 <xsl:for-each select="student/details">
                   <tr><td><xsl:value-of select="name"/></td>
                        <td><xsl:value-of select="address"/></td>
                        <td><xsl:value-of select="mark"/></td>
            </tr>
        </table></body></html></xsl:template></xsl:stylesheet>
```

**o/p:-**

| name | address | marks |
|---|---|---|
| Bala | Chennai | 62 |
| Gopal | Madurai | 88 |
| Lokesh | Vellore | 95 |

**<xsl:choose>**

**Step-1 simple.xml**

```
<?xml version="1.0" encoding = UTF-8"?>
<?xml-stylesheet type = "text/xsl" href="simple.xsl"?>
<student>
        <details>
                <name>bala</name>
                <address>chennai</address>
                <marks>62</marks>
        </details>
        <details>
                <name>lokesh</name>
                <address>vellore</address>
                <marks>95</marks>
        </details>
        <details>
                <name>Gopal</name>
                <address>madurai</address>
                <marks>88</marks>
        </details>
</student>
```

**Step-2: simple.xsl**

```
<?xml version="1.0" encoding = UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
        <html>
            <body>
                <table>
        <tr><th>Name</th><th>address</th><th>std</th></tr>
             <xsl:for-each select="student/details">
                    <tr><td><xsl:value-of select="name"/></td>
                         <xsl:choose>
                              <xsl:when test="marks&lt;75">
        <td><xsl:value-of select="marks"></td> </xsl:when>
                    </xsl:otherwise>
                    <td><xsl:value-of select="marks"/></td>
        </xsl:otherwise>
        </xsl:choose>
              </tr>
                    </xsl:for-each>
        </table></body></html></xsl:template></xsl:stylesheet>
```

**O/p:**

| name | address | marks |
|---|---|---|
| Bala | Chennai | 62 |
| Gopal | Madurai | 88 |
| Lokesh | Vellore | 95 |

**13. Explain newsfeed. Write notes on RSS and ATOM feeds.**

- On the World Wide Web, a **web feed** (or **news feed**) is a data format used for providing users with frequently updated content.
- Content distributors *syndicate* a web feed, thereby allowing users to *subscribe* to it.
- Making a collection of web feeds accessible in one spot is known as ***aggregation***, which is performed by a news aggregator.
- A web feed is also sometimes referred to as a *syndicated feed*.
- A typical scenario of web-feed use might involve the following: a content provider publishes a feed link on its site which end users can register with an aggregator program (also called a *feed reader* or a *news reader*) running on their own machines; doing this is usually as simple as dragging the link from the web browser to the aggregator.
- When instructed, the aggregator asks all the servers in its feed list if they have new content; if so, the aggregator either makes a note of the new content or downloads it.
- One can schedule aggregators to check for new content periodically.

**Advantages of Web feeds**

- Users do not disclose their email address when subscribing to a feed and so are not increasing their exposure to threats associated with email: spam, viruses, phishing, and identity theft.
- Users do not have to send an unsubscribe request to stop receiving news. They simply remove the feed from their aggregator.
- The feed items are automatically sorted in that each feed URL has its own sets of entries (unlike an email box where messages must be sorted by user-defined rules and pattern matching)

- A web feed is a document (often XML-based) whose discrete content items include web links to the source of the content.
- News websites and blogs are common sources for web feeds, but feeds are also used to deliver structured information ranging from weather data to top-ten lists of hit tunes to search results.
- The two main web feed formats are RSS and Atom.
- Web feeds are designed to be machine-readable rather than human-readable
- This means that web feeds can also be used to automatically transfer information from one website to another without any human intervention.

**RSS**

RSS stand for: It depends on what version of RSS you are using.

- **RSS Version 0.9** - **R**ich **S**ite **S**ummary
- **RSS Version 1.0** - **R**DF **S**ite **S**ummary
- **RSS Versions 2.0, 2.0.1, and 0.9x** - **R**eally **S**imple **S**yndication

- RSS is a protocol that provides an open method of syndicating and aggregating web content.
- RSS is a standard for publishing regular updates to web-based content.
- RSS is a Syndication Standard based on a type of XML file that resides on an Internet server.
- RSS is an XML application, which conforms to the W3C's RDF specification and is extensible via XML.
- You can also download RSS Feeds from other sites to display the updated news items on your site reader to access your favorite RSS Feeds.
- About 50 % of all RSS Feeds use RSS 0.91.
- About 25 % use RSS 1.0.
- The last 25 % is split between RSS 0.9x versions and RSS 2.0.

**Working of RSS**

- A website willing to publish its content using RSS creates one RSS Feed and keeps it on a web server.
- RSS Feeds can be created manually or with software.
- A website visitor will subscribe to read your RSS Feed.
- An RSS Feed will be read by an RSS Feed reader.
- The RSS Feed Reader reads the RSS Feed file and displays it.
- The RSS Reader displays only new items from the RSS Feed.
- The RSS Feed reader can be customized to show you content based on your own interest.

**Who can Use RSS**

- **New Homes** - Realtors can provide updated Feeds of new home listings on the market.
- **Job Openings** - Placement firms and newspapers can provide a classified Feed of job vacancies.
- **Auction Items** - Auction vendors can provide Feeds containing items that have been recently added to eBay, etc
- **Press Distribution** - Listing of new releases.
- **Schools** - Schools can relay homework assignments and quickly announce school cancellations.
- **News & Announcements** - Headlines, notices, and any list of announcements.
- **Entertainment** - Listings of the latest TV programs or movies at local theatres.

**Advantages for Subscribers**

RSS subscribers are the people who subscribe to read a published Feed

- **All news at one place:** You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time.
- **News when you want it:** Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer.
- **Get the news you want:** RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you.
- **Freedom from e-mail overload:** You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server.
- **Easy republishing:** You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site.

**Advantages for Publishers**

RSS publishers are the people who publish their content through RSS feed.

- **Easier publishing:** RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically.
- **A simpler writing process:** If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site.
- **An improved relationship with your subscribers:** Because people subscribe from their side, they don't feel as if you are pushing your content on them.
- **The assurance of reaching your subscribers:** RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more.
- **Links back to your site:** RSS Feeds always include links back to a website. It directs a lot of traffic towards your website.
- **Relevance and timeliness:** Your subscribers always have the latest information from your site.

```
<?xml version="1.0" ?>
    <rss version="2.0">
    <channel>
     <title>Ajax and XUL</title>
     <link>http://www.xul.fr/en/</link>
     <description>XML graphical interface etc...</description>
     <image>
        <url>http://www.xul.fr/xul-icon.gif</url>
        <link>http://www.xul.fr/en/index.php</link>
     </image>
     <item>
        <title>News  of today</title>
        <link>http://www.xul.fr/en-xml-rss.html</link>
        <description>All you need to know about RSS</description>
     </item>
     <item>
        <title>News of tomorrows</title>
        <link>http://www.xul.fr/en-xml-rdf.html</link>
        <description>And now, all about RDF</description>
      </item>
    </channel>
    </rss>
```

**How browsers know there is an RSS feed on a website**

- You have created an RSS feed and it is now stored at root of your website.
- You must let browsers knowing the existence of this file and its location, when they enter and display the home page (or any other page if you want).
- Firefox will display the feed icon into the URL field, Internet Explorer on the bar of commands.
- To activate them, insert the following line into the source code of the page, anywhere inside the <head> </head> section:

```
<link rel="alternate" type="application/rss+xml"
  href="http://www.xul.fr/rss.xml" title="Your title">
```

- Replace the URL by your domain name with the path and filename of your RSS feed.
- And if the file is in the atom format, replace rss+xml by atom+xml.

**ATOM feed**

- Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.
- Atom is a relatively recent spec and is much more robust and feature-rich than RSS.
- For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.
- All Atom Feeds must be well-formed XML documents, and are identified with the *application/atom+xml* media type.

**Structure of an Atom 1.0 Feed**

```
<?xml version="1.0"?>
  <feed xmlns="http://www.w3.org/2005/Atom">
      <title>...</title>
      <link>...</link>
      <updated>...</updated>

      <author>
        <name>...</name>
      </author>

      <id>...</id>

      <entry>
        <title>...</title>
        <link>...</link>
        <id>...</id>

        <updated>...</updated>
        <summary>...</summary>
      </entry>

  </feed>
```

**Example:-**

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Example Feed</title>
  <subtitle>Insert witty or insightful remark here</subtitle>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>Mohtashim</name>
    <email>mohtashim@example.com</email>
  </author>

  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Tutorial on Atom</title>
    <link href="http://example.org/2003/12/13/atom03"/>

    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
  </feed>
```

| RSS | ATOM |
|---|---|
| Contains either plain text or escaped sequence as payload | Contains html, xml, dhtml, documents, audio, video, etc as payload |
| Shows timestamp of data when feed was last created or updated | Shows timestamp of data when it was last updated |
| Uses blogger and meta weblog protocols | It has only one standard protocols |
| Loose approach on data | Strict approach on data |
| More complicated process | Easier process |
| Not a standard feature | Standard feature |
| Less robust, scalable, efficient | More robust, scalable, efficient |

**********

# UNIT 5 - AJAX & WEB SERVICES

## Part-A

**1. What is Ajax? (Nov/Dec 2018)**

- AJAX is an acronym for **asynchronous** JavaScript **and** XML
- It is a set of web development techniques using many web technologies on the client-side to create asynchronous Web applications.
- With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- Ajax is not a technology, but a group of technologies.
- **HTML** and **CSS** can be used in combination to **mark up** and **style** information.
- The **DOM** is accessed with JavaScript to **dynamically** display and allow the user to **interact** with the information presented.
- **JavaScript** and the **XMLHttpRequest** object provide a method for **exchanging data asynchronously** between browser and server to avoid full page reloads.

**2. Mention the open standards of Ajax.**

- Browser-based presentation using **HTML** and Cascading Style Sheets (**CSS**).
- Data is stored in **XML** format and fetched from the server.
- Behind-the-scenes data fetches using **XMLHttpRequest** objects in the browser.
- **JavaScript** to make everything happen.

**3. Brief about asynchronous nature of AJAX.**

- Asynchronous means that the script will send a request to the server, & continue it's execution without waiting for reply.
- As soon as reply is received a browser event is fired, which in turn allows the script to execute associated actions.
- Ajax knows when to pull data from server, because you tell it when to do it.

**4. What is XHR?**

- XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.
  - **Update** a web page without reloading the page
  - **Request** data from a server - after the page has loaded
  - **Receive** data from a server - after the page has loaded
  - **Send** data to a server - in the background

**5. What is a web service? (Nov/Dec 2015)**

- Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data.
- It is OS and language independent
- Web Services can convert your existing applications into Web-applications.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems.

**6. Mention the characteristics of web service.**

- Machine-to-machine interactions
- Loose coupling
- Interoperability
- Platform-independence
- Operating system-independence
- Language-independence
- Leveraging the architecture of the World Wide Web

**7. What are the components of Web Services?**

The basic web services platform is XML + HTTP.

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

**8. What are the advantages of web service?**

- Exposing the Existing Function on the network
- Interoperability
- Standardized protocol
- Low Cost of Communication

**9. What are RESTful web services?**

- **RESTful** Web Services are REST architecture based web services.
- In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable
- It is very commonly used to create APIs for web based applications.
- REST stands for **RE**presentational **S**tate **T**ransfer.
- REST is web standards based architecture and uses HTTP Protocol for data communication.

**10. Define WSDL**

- WSDL stands for Web Services Description Language.
- It is the standard format for describing a web service.
- WSDL was developed jointly by Microsoft and IBM.
- To exchange information in a distributed environment.
- WSDL is used to describe web services
- WSDL is written in XML
- WSDL is a W3C recommendation from 26. June 2007

**11. What are the elements of WSDL?**

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

**12. Define SOAP**

- SOAP is an acronym for Simple Object Access Protocol.
- It is an XML-based messaging protocol for exchanging information among computers.
- SOAP is an application of the XML specification.
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

**13. Mention the features of SOAP.**

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for Web services.
- SOAP can exchange complete documents or call a remote procedure.
- SOAP can be used for broadcasting a message.
- SOAP is platform- and language-independent.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

**14. List out the elements of SOAP**

- An **Envelope** element that identifies the XML document as a SOAP message
- A **Header** element that contains header information
- A **Body** element that contains call and response information
- A **Fault** element containing errors and status information

**15. Mention the advantages of SOAP.**

| | |
|---|---|
| Simplicity | Universal acceptance |
| Portability | Versatile |
| Firewall | flexible |
| Scalable | Interoperability |
| Friendliness | No bi-directional HTTP communication |
| Use of open standards | No distributed garbage collection |
| No Object activation. | No Object by reference. |

**Part - B**

1. **Explain the Client-server architecture of Ajax with a neat diagram.** (Nov/Dec 2015)

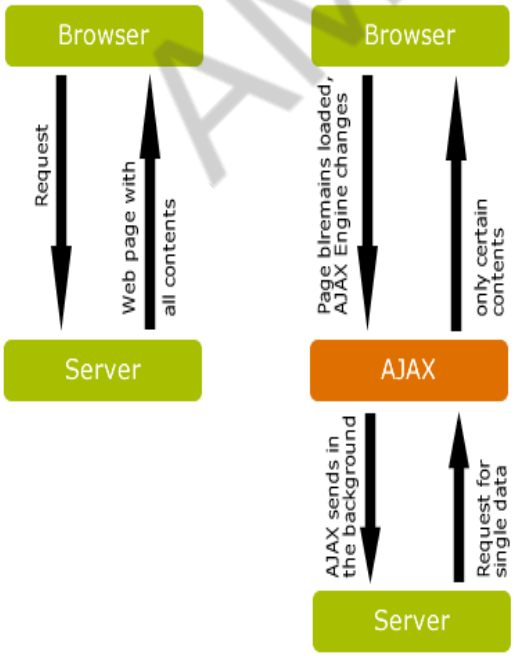
AJAX is a developer's dream, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

- AJAX is not a programming language.
- AJAX is a technique for accessing web servers from a web page.
- AJAX stands for Asynchronous JavaScript And XML.
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.
- AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

**RIA Technology**

- AJAX is the most viable Rich Internet Application (RIA) technology so far.
- It is getting tremendous industry momentum and several tool kit and frameworks are emerging.
- But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.




**Steps of Processing in Ajax:-**

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

**AJAX is based on the following open standards:-**

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

**Components of Ajax:-**

**JavaScript**

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

**DOM**

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

**CSS**

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript.

**XMLHttpRequest**

- JavaScript object that performs asynchronous interaction with the server.

**Example for AJAX:** Google maps, Gmail, cricket update websites, stock markets websites, etc

**Example:-**

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
    {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

**O/P:-**

The XMLHttpRequest Object

Change Content

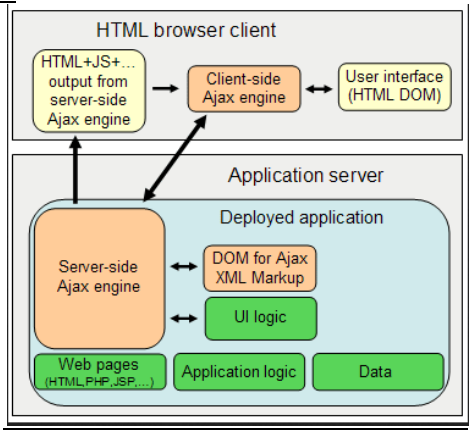
AJAX

AJAX is not a programming language.

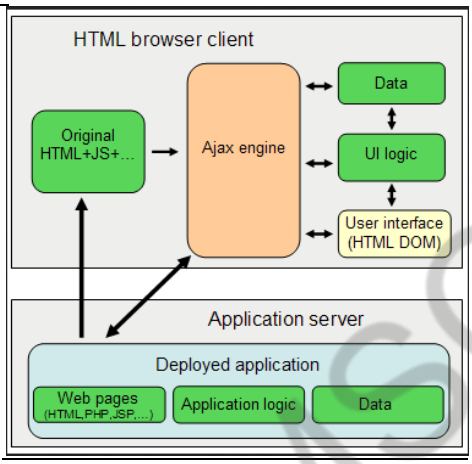
AJAX is a technique for accesing web servers from a web page.

AJAX stands for Asyncronus JavaScript And XML.

**Server side:-**




**Client side:-**




**2. Explain XMLHttpRequest object with example. (Nov/dec 2016)**

- The XMLHttpRequest object can be used to request data from a web server.
- The XMLHttpRequest object is **a developers dream**, because you can:
  - Update a web page without reloading the page
  - Request data from a server - after the page has loaded
  - Receive data from a server - after the page has loaded
  - Send data to a server - in the background
- The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.
- XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.
- The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

**XMLHttpRequest Methods**

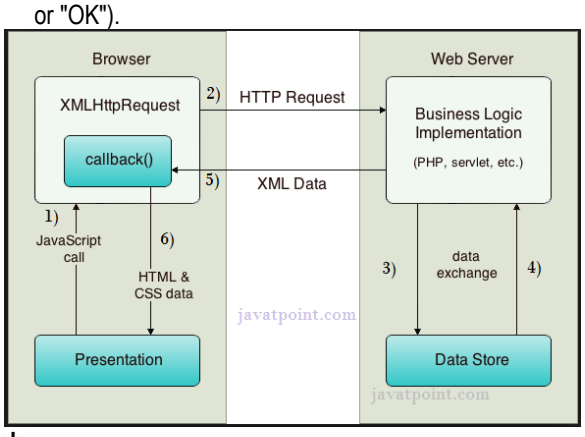
- **abort()** : Cancels the current request.
- **getAllResponseHeaders()** : Returns the complete set of HTTP headers as a string.
- **getResponseHeader( headerName )** : Returns the value of the specified HTTP header.
- **open( method, URL )**

**XMLHttpRequest Properties**

- **onreadystatechange** : An event handler for an event that fires at every state change.
- **readyState** : The readyState property defines the current state of the XMLHttpRequest object.

| State | Description |
|---|---|
| 0 | The request is not initialized. |
| 1 | The request has been set up. |
| 2 | The request has been sent. |
| 3 | The request is in process. |
| 4 | The request is completed. |

  - **readyState = 0** After you have created the XMLHttpRequest object, but before you have called the open() method
  - **readyState = 1** After you have called the open() method, but before you have called send().
  - **readyState = 2** After you have called send().
  - **readyState = 3** After the browser has established a communication with the server, but before the server has completed the response.
  - **readyState = 4** After the request has been completed, and the response data has been completely received from the server.
- **responseText** :Returns the response as a string.
- **responseXML** :Returns the response as XML.
- **status** : Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText** :Returns the status as a string (e.g., "Not Found" or "OK").

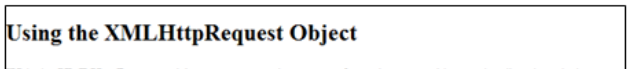



**Example:-**

```
<!DOCTYPE html>
<html>
    <body>
        <h2>Using the XMLHttpRequest Object</h2>
        <div id="demo">
        <button type="button"
        onclick="loadXMLDoc()">Change Content</button>
        </div>
        <script>
```

```
function loadXMLDoc()
{
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function()
        {
                if (this.readyState == 4 && this.status == 200)
                {
    document.getElementById("demo").innerHTML = this.responseText;
                }
        };
          xhttp.open("GET", "xmlhttp_info.txt", true);
          xhttp.send();
        }
</script>
</body>
</html>
```

**O/p:-**


- The **onreadystatechange** property specifies a function to be executed every time the status of the XMLHttpRequest object changes.
- When **readyState** property is 4 and the **status** property is 200, the response is ready.
- **responseText** property returns the server response as a text string.
- The text string can be used to update a web page

**3. Describe WSDL with diagrams necessary.** (Nov/Dec 2015)

- WSDL stands for Web Services Description Language
- WSDL is used to describe web services
- WSDL is written in XML
- WSDL is a W3C recommendation from 26. June 2007
- It is the standard format for describing a web service. • WSDL was developed jointly by Microsoft and IBM.

**Features of WSDL**

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

**WSDL Elements**

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of data being communicated.
- **Operation**– an abstract description of an action supported by service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

**The WSDL Document Structure**

```
<definitions>
  <types>
    definition of types.....
  </types>

  <message>
    definition of a message....
  </message>

  <portType>
    <operation>
      definition of a operation.....
    </operation>
  </portType>

  <binding>
    definition of a binding....
  </binding>

  <service>
    definition of a service....
  </service>
</definitions>
```

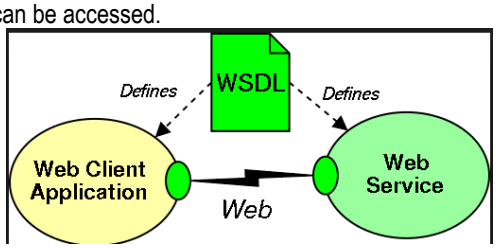
**Contents of HelloService.wsdl file:**

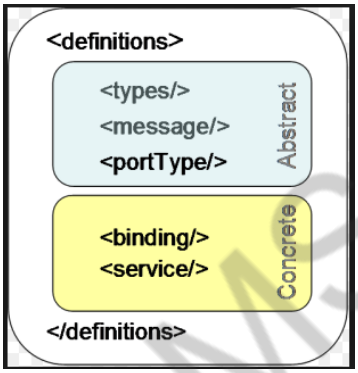
```
<definitions name="HelloService"
  targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>

  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>

  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>
  <binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </output>
    </operation>
  </binding>
```

```
  <service name="Hello_Service">
    <documentation>WSDL File for HelloService</documentation>
    <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
        location="http://www.examples.com/SayHello/" />
    </port>
  </service>
</definitions>
```

**Example Analysis**

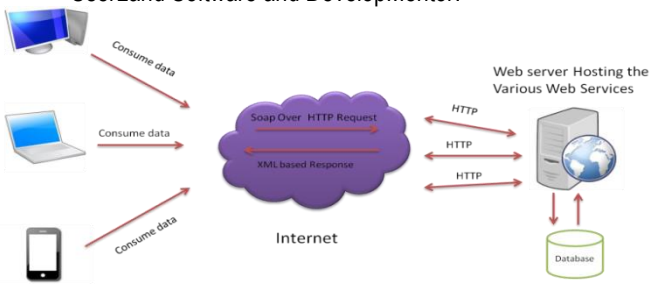
- **Definitions** : HelloService
- **Type** : Using built-in data types and they are defined in XMLSchema.
- **Message** :
  - sayHelloRequest : firstName parameter
  - sayHelloresponse: greeting return value
- **Port Type** : sayHello **operation** that consists of a request and a response service.
- **Binding** : Direction to use the SOAP HTTP transport protocol.
- **Service** : Service available at http://www.examples.com/SayHello/
- **Port** : Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.




4. **Explain the working of SOAP with an example.(Nov/Dec 2015)**

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation
- SOAP 1.1 was originally submitted to the W3C in May 2000. Official submitters included large companies such as Microsoft, IBM, and Ariba, and smaller companies such as UserLand Software and DevelopMentor.




**Why SOAP?**

- It is important for web applications to be able to communicate over the Internet.
- The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

**SOAP Building Blocks**

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

**Syntax Rules**

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message MUST use the SOAP Encoding namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

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

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
   ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

**SOAP request**

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >
  <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotations" >
    <m:GetQuotation>
      <m:QuotationsName>MiscroSoft</m:QuotationsName>
    </m:GetQuotation>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**SOAP response**

**HTTP/1.0 200 OK**
Content-Type: **text/xml;** charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
**<SOAP-ENV:Envelope** xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >
<SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotation" >
<m:GetQuotationResponse>
<m:Quotation>Here is the quotation</m:Quotation>
</m:GetQuotationResponse>
</SOAP-ENV:Body>
</**SOAP-ENV:Envelope>**