CS481 : Web Technology.

History of Internet :

Internet evolved from basic idea of ARPANET for interconnecting comp.

1989 : Internet available for commercial purpose.

Basic Services Provided by Internet :

1. Electronic mail (e-mail)
2. File Transfer Protocol (FTP)
3. Telnet : (Remote Connection — Server-Client)
4. Usenet News
5. HTTP : Hypertext Transfer Protocol.
6. TCP : Transmission Control Protocol.

WWW : World Wide Web.

- Huge collection of pages of information linked to each other around one globe.
- Every page is combination of text, picture, audio, video, animation and hyperlink.
- Father of WWW : Berners-Lee, Tim

• Webpage

- Collection of normal text, picture, video, audio, hyperlink.
- Can be designed using html, xml, javascript etc.

HTML → design structure
JavaScript → Validate the field
XML → advanced version of html.

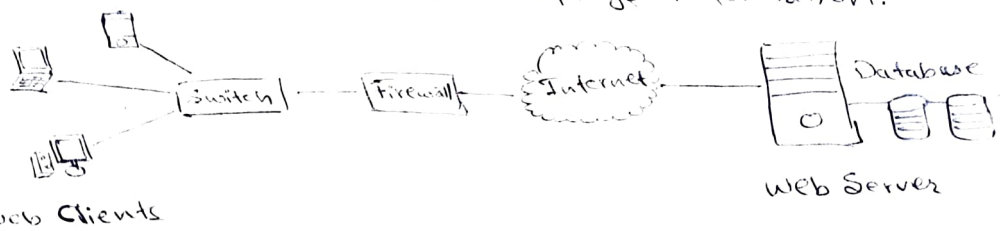
• Website

- Collection of interlinked webpages.
- Website is accessed through url.
- Url is a global address of web document on www.
- Urls are unique in nature. May not have a copy.
- 2 parts in url :
  - → Protocol : E.g. : http://
  - → Resource name : www.google.com.
- Types of websites based on function :
  - → Personal website
  - → Commercial website
  - → Government website
- Types of websites based on animation/design :
  - → Static : information site
  - → Dynamic : interactive site.

- Web Application
  - Can run web app in the browser through url.
  - 2 types of web Apps
    1. Service Oriented
    2. Presentation Oriented.
- Service Oriented App:
  - Used to implement web services
  - Coded using CGI, JSP, HSP
- Presentation oriented Web Apps:
  - Provides client side services
  - Coded using HTML, XML, Javascript etc.
- Web Architecture:

  www follows 2-tier architecture: web server
  web client.

  web server produces and delivers information
  web client retrieves and displays information.



* WWW Protocols:

1. TCP (Transmission Control Protocol): Communicating over network
2. IP (Internet Protocol): Addressing protocol for helping with routing packets through diff nodes in net.
3. UDP (User Datagram Protocol): Substitute communication protocol to TCP
4. POP (Post Office Protocol): Receiving incoming e-mails.
5. SMTP (Simple Mail Transport Protocol): Sending & Distributing outgoing email.
6. FTP (File Transfer Protocol): Transferring file over network.
7. HTTP (Hypertext Transfer Protocol): Transferring hypertext.
8. HTTPS (HTTP Secure): Secure version of HTTP/encrypted.
9. Telnet: Set of rules designed for connecting systems over net.
10. Gopher: Collection of rules for searching, retrieving and displaying documents from isolated sites.

- Web browser: Application software for accessing WWW or a local website. It is used to retrieve data from webpages.

* Principles of Web site Design:

1. Website Purpose
2. Simplicity, Usability, User Experience (UX) and User Interface (UI)
3. Navigation.
4. F-shaped Pattern Reading (mimic natural pattern of reading)
5. Visual Hierarchy (arrangement of elements in order of importance)
6. Content (Great design, but greater content)
7. Grid-Based layout.
8. Load Time.
9. Mobile Friendly / Responsiveness.

- Easy Navigation
- Responsive Design
- Uniform Color Scheme / consistency
- Comfortable UI
- Content & Purpose
- Performance
- Progress Feedback
- Mitigate Error Pages
- Avoid pop-ups, alert/dialogs

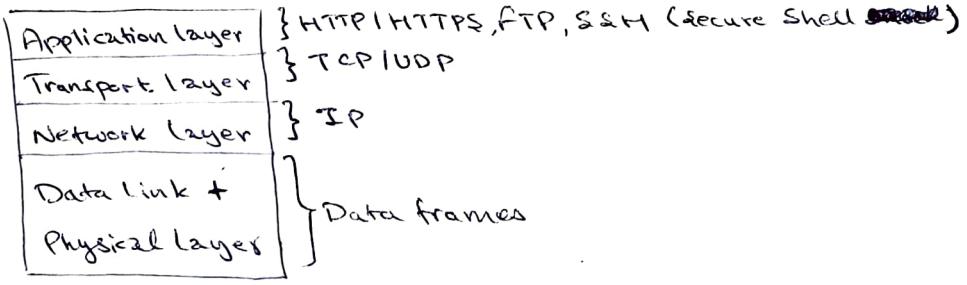
* Website Navigation:

- Process of navigating pages, apps and websites on internet. Technology behind it is called hypertext or hypermedia.
- Types of website Navigation:
  - Global Website Navigation: Menu and links identical across all pages of website.
  - Hierarchical Website Navigation: Menus change depending on the content of each page.
  - Local Website Navigation: Internal links included in the content itself

- Web Security: Protecting networks and computer systems from damage, theft of software, hardware or data. It is enforced by a security appliance that acts as a web proxy, sitting between users and the internet.
- SSH: SSH (key-based) authentication uses public key cryptography to operate.
  SSH certificate-based authentication attaches a signed certificate to each key to verify their identities.
- Certificate Authority: CAs are trusted entities that issue digital certificate.
- Digital Certificate (public key certificate / identity certificate): Electronic document used to provide validity of public key

- Trusted Third Party (TTP): Entity which facilitates interactions between two parties who both trust the third party.

- TCP/IP Protocol layers:



SSH: Secure Shell Protocol is cryptographic network protocol for operating network services securely over unsecured network.

Data Frames: Data structure that organises data into a two-dimensional table of rows and columns.

Servlet: It is a Java programming language class that is used to extend capabilities of servers that host applications accessed by means of a request-response programming model.

ISP: Internet Service Provider is a company/an organisation that provides individuals and organizations access to the internet and other related services.

Gateway: Network node that connects two or more networks with different transmission protocols together.

CGI:- Common Gateway Interface: It provides the middleware between www servers and external database and information sources.

- It is the process for passing data back and forth between the server and the application.

* PHP:
- 1994: Developed by Apache Group.
- Stands for 'hypertext pre processor'. Earlier full form was 'personal homepage'
- Widely used open source general purpose scripting lang. for web development
- Server side scripting lang. Mainly used for form handling, database access, web development.

Basic Syntax: <?php
//Core Here
?>

<?php /*code here*/ ?>

- Makes use of dynamic typing, i.e, no need to declare variable type:
for. e.g.: in C, we write, int a = 5;
in PHP, we can write, $a = 5; //no need of int, float, string, char

• Characteristics:
→ Open-source: free of cost (now)
→ Simplicity: does not include library functions, hence simple structure.
→ Efficiency: uses Resource Allocation Mechanism, OOP, Session management features, so, it eliminates unnecessary memory allocation.
→ Security: Supports Encryption functions.
→ Flexibility: can be embedded with HTML, CSS, JS, XML etc. and can run on any devices like phone, laptop etc.
→ Object Oriented: Supports OOPs concept.

* Declaring Variables in PHP:
$variable_name = value;
. If no value is assigned to the variable, then its value is by default NULL

Rules for declaring Variables:
→ Case-sensitive
→ must start with either letter or underscore.
→ must not start with number.
→ must contain only Alpha-numeric values, and underscore.
→ must not contain blank space, white space etc.

* Datatypes in PHP:
→ Scalar - Int, Boolean, Double, String
→ Compound - Arrays, Objects
→ Special - Resource, NULL

* Cookies in PHP:
→ Small files that will be stored in browser containing user details.
→ Syntax: set cookie (name, value, expire, path, domain);

* Session Handling in PHP:
Syntax: Starting session: session_start[];
Ending Session: session_destroy[];

* File Handling in PHP:

(function, mode):

opening file Syntax: $file = fopen("f.txt", "w");

modes: r → read only r+ → read/write
w → write only w+ → read/write
a → append a+ → read/append

Reading file Syntax: $filedata = fread($file, $size);
can also use fgets() or fgetc() to read file.

Writing file Syntax: fwrite($file, "Hello World!");
can also use fwrite(), fputs() to write file.

Unlinking file Syntax: unlink($file);

Closing file Syntax: fclose($file);

* Handling File Uploads in PHP:

- PHP $_FILES:

$_FILES['filename']['name']: Returns name of file.
$_FILES['filename']['type']: Returns MIME Type.
$_FILES['filename']['size']: Returns size of file.
$_FILES['filename']['temp_name']: Shows temp name of file.
$_FILES['filename']['error']: Returns errors associated with file.

- Move uploaded file in PHP:

Syntax: move_uploaded_file(string $filename, string $destination);

* Connecting to Database: (MySQL as reference).

- Functions in MYSQL:

To Create Database: mysql_query()
for errors in commands: mysql_error()
Selecting database: mysql_select_db()
Listing Database: mysql_list_db()
Displaying Tables: mysql_list_table()

- Connecting to Database from php (Accessing DB):

Data Source Name:
```
$dsn = "mysql:dbname = MyDB";
$user = "root";
$password = "admin";
```

Creating PHD data object:
```
$conn = new PDO ($dsn, $user, $password);
if ($conn == null) { die ($!); }
```

Associative Array:
```
$sql = "SELECT * FROM fruit";
$rows = $conn→query($sql);
```

Disconnect Connection: $conn = null;

* FORM HANDLING IN PHP:

1. $_POST 2. $_GET

Example:

```
<!--index.html-->
<HTML>
<BODY>
    <form action="data.php" method="post">
        Name: <input type="text" name="name"> <br>
        Email: <input type="text" name="email"> <br>
        <input type="submit" value="submit">
    </form>
</BODY>
</HTML>

<!-- data.php -->
<HTML>
<BODY>
<?php
    echo $_POST["name"];
?>
<br>
Your Email:
<?php
    echo $_POST["email"];
?>
</BODY>
</HTML>
```

$_GET method is not secret and safe. Usage is same as $_POST.
In place of method="post", we write method="get", and in php file, in place of $_POST["name"], we write $_GET["name"].
The only difference, whatever is taken using $_GET method, is displayed on the browser url as http://@user:abc:mail:abc@gmail.c
hence, $_GET is not safe for password or even private information, hence $_GET is very rarely used and $_POST is preferred over $_GET.

Date 29 08 2022

PHP String Functions

* Finding the length of string : strlen()
* Accessing characters/substrings : indexing, substr()
* Searching : strstr()
* Locating text : strpos(), strrpos()
* Finding no. of occurences : substr_count()
* Searching for a set of characters : strpbrk()
* Replacing text :
  - str_replace() — Replaces all occurances
  - substr_replace() — Replaces a specified portion.
  - strtr() — Replaces certain characters.
* Case conversion : strtoupper(), strtolower(), ucfirst(), lcfirst ucword(), lcword()
  - stristr(), strpos(), strrpos(), ~~strripos()~~, str_replace() → case sensitive.
  - stristr(), stripos(), strripos(), str_ireplace() → case insensitive.
* C style formatted I/O : printf(), sprintf(), fprintf()
* trimming : trim(), ltrim(), rtrim()
* Padding : str_pad()
* Text wrapping : word_wrap()
* Formatting numbers : number_format().

Lecture Notes
PHP Arrays

Saathi

Date 05/07/2022

* Two types of arrays:
  (i) Indexed arrays
  (ii) Associative arrays/Hash/Map
* Creating arrays: array()
* Accessing arrays: Indexes/keys
* Outputting an entire array: print_r()
* Extracting a range of elements: array_slice()
* Counting elements of an array: count()
* Stepping through an associative array:
  current(), key(), next(), prev(), end(), reset()
* Looping through arrays: foreach., each()

(i) Looping through values:

```
foreach($array as $value) {
    //do something with $value
}
```

(ii) Looping through key-value

```
foreach($array as $key => $value) {
    //do something with $key & $value
}
```

* Multidimensional array:- self study.
* Sorting arrays
  (i) sort()/rsort(): sort indexed arrays
  (ii) asort()/arsort(): sort associative arrays by value
  (iii) ksort()/krsort(): sort " " " key
  (iv) array_multisort(): sorting multidimensional arrays.
* Adding/removing array elements:
  (i) array_unshift(): add elements at the beginning
  (ii) array_shift(): removes " from " "
  (iii) array_push(): adds " at " end
  (iv) array_pop(): removes " from " "
  (v) array_splice(): add/remove elements to/from arbitrary position
* Merging arrays; array_merge()
* Converting/reverting between arrays & strings: explode(), implode()
* Converting arrays to list of variables: list()

Basic HTML Requirement / ~~Template~~ Syntax's

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
    <TITLE > Title here </TITLE>

    <link rel = 'stylesheet' href = 'Styles.css'/>
    <script src = 'myscript.js'/>
</HEAD>
<BODY>
    <!-- Body Content Here -->
</BODY>
</HTML>
```

XHTML:

Minimum Required Tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title> Document Title </title>
</head>
<body>
        Body Content Here
</body>
</html>
```

Difference between HTML and XHTML: In XHTML ↴

- <!DOCTYPE ... > is mandatory
- XHTML Elements must be properly nested.
- Elements must always be closed.
  <hr/>
- Elements and Attribute names must be in lower case.
- Attribute values must be Quoted and minimisation is forbidden.

XML basic formatting:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SupTag>
    <tg1> ... </tg1> <tg2>....</tg2>
</SupTag>
```

* MySQL creating a basic SQL document:

```
$mysql -u root -p passcode.
> create database STUDENT-DB
> use STUDENT_DB
> CREATE TABLE Student (rollno NUMBER (10) NOT NULL UNIQUE,
                        name VARCHAR(100) NOT NULL,
                        PRIMARY KEY (roll no)
                        );
> INSERT INTO Student ('1912160', 'Subhojit Ghimire');
```

* Connecting MySQL to PHP:

```
<?php
    $dsn = "mysql: dbname = STUDENT-DB ";
    $user = "root";
    $password = "passcode";
    $conn = new PDO ($dsn, $user, $password);
    if ($conn == NULL) {
            die ($!);
     }
    $sql = "SELECT*FROM Student;";
    $rows = $conn → query ($sql);
    if ("$rows" == NULL) {
            die ($!)
     }
     foreach ( $rows as $rows) {
          echo " $rows ["roll no"]." is the roll
                number of". $rows ["name"]";
          echo "<br";
      }
      $conn = NULL;
?>
```

* Passing Argument by Reference in PHP:

```
<?php
    function add_five (& $value) { $value += 5; }
     $num = 2;
     add_five($num);
      echo $num;
?>
```

* DTD (Document Type Definition):

- Elements:

given xml file:

```
<note>
    <to> Grove </to>
    <from> Jani </from>
    <heading> Reminder </heading>
    <body> Don't forget to take morning stroll </body>
</note>
```

Correspondind DTD (internal):

```
<?xml version='1.0'?>
<!DOCTYPE note [
    <!ELEMENT note (to, from, heading, body)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT heading (#PCDATA)>
    <!ELEMENT body (#PCDATA)>
]>
<note>
 <to> Grove </to>
 <from> Jani </from>
 <heading> Reminder </heading>
 <body> Don't forget to take morning stroll </body>
 </note>
```

Some other examples:

<!ELEMENT br EMPTY> →

<!ELEMENT note (message+)>

Occurance Indicators:

message	→	occurs only once
message+	→	at least once & more than once)
message*	→	zero to more
message?	→	either occurs once or does not occur

#PCDATA : Text between tags <tag1> This value </tag1>

#CDATA : value of attributes <tag1 width="100"/>

<!ELEMENT tag1 EMPTY> <!ATTLIST tag1 width CDATA "0">

- **Attributes:**

given xml file:

```
<exam>
    <day>
        <question qno='Q1'> What is DTD? </question>
        <answer aqno='Q1'> It is Document Definition </answer>
    </day>
    <day>
        <question qno='Q1'> What is XML? </question>
        <answer aqno='Q1'> Don't know </answer>
        <question qno='Q2'> What is HTML? </question>
        <answer aqno='Q2'> Markup language </answer>
    </day>
</exam>
```

DTD:

```
<?xml version="1.0" ?>
<!DOCTYPE exam [
    <!ELEMENT exam (day*)>
    <!ELEMENT day (question+, answer+)>
    <!ELEMENT question (#PCDATA)>
    <!ELEMENT answer (#PCDATA)>
    <!ATTLIST question qno ID #REQUIRED>
    <!ATTLIST answer aqno IDREF #REQUIRED>
]>
```

- **Entities:**

Used to define shortcuts to special characters.

Example:

<!ENTITY lt "<"> → <body> < </body>

User defined shortcuts:

<!ENTITY me "Subhojit"> → <body> &me; </body>

Everywhere &me; is used, is replaced by Subhojit

in XML or XHTML

* XSLT basic Structure: (XML to HTML)

```
<?xml version='1.0'?>
<xsl:transform version='2.0' xmlns:xsl="http://www.w3.org
                                        /1999/XSL/Transform">

Not really required {
<xsl:output method='html' />
<xsl:template match='/'>

    <html>
      <body>
          <!-- HTML tags and xsl tags for
               converting XML To XHTML -->

      </body>
    </html>

</xsl:template>
</xsl:transform>
```

// <xsl:transform> is somewhere written as <xsl:stylesheet>
// Both are correct and ~~same~~ give same meaning.

Example:

Given simple XML:

```
<?xml version='1.0' encoding='UTF-8'?>
<catalog>
    <cd>
        <title> Empire Burlesque </title>
        <artist> Bob Dylan </artist>
        <price currency='USD'> 10.90 </price>
    </cd>
    <cd>
        <title> Let it Be </title>
        <artist> The Beatles </artist>
        <price currency='USD'> 20.90 </price>
    </cd>
</catalog>
```

Corresponding XSLT:

```
<?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>
        <h2> MY CD COLLECTION </h2>
        <table border = '1' >
            <tr bgcolor = '#9acd32' >
                <th> Title </th>
                <th> Artist </th>
                <th> Price </th>
                <th> Currency </th>
            </tr>
            <xsl:for-each select = 'catalog/cd' >
            <tr>
                <td> <xsl:value-of select = 'title'/> </td>
                <td> <xsl:value-of select = 'artist' /> </td>
                <td> <xsl:value-of select = 'price' /> </td>
                <td> <xsl:value-of select = 'price/@currency' /> </td>
            </tr>
            </xsl:for-each>
        </table>
    </body>
    </html>
</xsl:template>
</xsl:Stylesheet>
```

* if xsl:template match was match = '/catalog' instead of '/', xsl:for-each select could have been written as select = 'cd' instead of select = 'catalog/cd'.

XSLT xsl:functions will be of many types depending upon what we are planning on doing in the code. In simples terms, xsl:s will contain loops, switch case, functions, if-else etc. that is available or is done in any programming languages, so the variety will be large.

XSLT is a programming/scripting language for HTML, so keep learning.