1) * Web Technology:

- Web Technology allows us to connect with people all over the world & access info at our fingertips.
- It is the foundation of the WWW & plays a crucial role in how we access & interact with info on the Internet.

* Web Basics

- Vast network of interconnected documents & resources, accessible through web browsers
- web pages are written using HTML & styled with Cascading Style Sheets (CSS).
- Uniform Resource Locators (URLs) are used to identify & locate resources on the Web.

* Static & Dynamic Web Page

-> Static

- fixed content that remains the same for all users
- written in html & css
- Changes require manual editing of the source code

-> Dynamic

- content generated on the fly, tailored to individual user interactions
- Utilize client side or server side scripting languages
- enhance interactivity & user experience.

* Client-Side Scripting Language

- executes on user's web browser.
- allows for real-time interactivity without the need to comm with the server
- popular langs include JS, HTML & CSS

* Server Side Scripting Language
- executes on the web server
- used for dynamic content generation & database interactions
- Common languages include PHP, Python & Ruby.

* HTTP
- foundation of data comm on the Web.
- used to request and transmit web pages & resources from servers to clients.

* HTTPs
- Secure ver of HTTP that uses encryption (TLS/SSL) to protect data transmission.
- Ensures confidentiality & integrity of data exchanged bet^n clients and servers.

* FTP
- Standard network protocol used for transferring files bet^n a client and a server over a TCP-based ntwk, such as Internet
- developed in early 70s & one of the oldest protocols still in use for file transfer.
- uses client-server architecture
- allows users to upload files from the client to the server & download files from the server to the client.

* Free Software.
- users have the freedom to use, study, modify & distribute the software
- Promotes community collaboration & sharing
- Eg: GNU/Linux O.S

* Open Software (Open Source):
- Similar to free software but often focuses on practical benefits of collaboration.
- Source code accessible for users to study, modify & distribute.
Eg: Apache Web Server

* Proprietary Software:
- controlled by a company or individual
- users typically cannot access the source code or modify it
- Eg: Microsoft Office suite

* Commercial Licensing
- proprietary software often uses it
- users need to purchase a license to use the software legally
- may have restrictions on the no. of installations or users.

* Open Source Licensing:
- allow users to access, use, modify & distribute the source code freely
- different licenses have varying level of restrictions.
Eg: GNU General Public License (GPL), Apache License

* Web browsers:
- Software app used to access & view websites on the Internet
- popular web browsers include Chrome, Firefox, Edge, Safari
- interpret HTML, CSS & JS to display web content like fonts, imgs & videos
- allows users to navigate the web, interact with webpages & submit info through forms.

* Web Servers:
- Software or hardware systems that store & serve websites to users over the Internet

- respond to request from web browsers & deliver webpages or data
- Popular web server software includes Apache, Nginx & others
- Web servers use protocols like HTTP or s to comm with br

* Tier Technology & its Types
- often referred as "tier architecture", is a way to organize software systems into seperate layers or tiers
- helps in improving scalability, maintainability & flexibility of applications.

* Single-Tier (Monolithic)
- all app components (eg: UI, business logic & database) are tightly interlinked into a single system
- typically not very scalable or maintainable but simple for small applications.

* Two-Tier (Client-Server)
- Divides the app into 2 primary comps: client & server
- Client handles the user interface & the server manages data & business logic
- Common for desktop apps & early web apps.

* Three-tier
- divides the app into three layers: presentation, logic & data
- pre layer (client) interacts with the app server, which handles the business logic
- data layer manages the database
- Offers improved scalability & seperation of concerns

classmate
Date
Page

As

Advantages
- freedom for users to use, modify & share the soft without restrictions
- high level of transparency & trust in the software's behavior
- Community support & collaboration often lead to rapid bug fixes & implementation improvements.
- promotes principles of ethical & user-centric software

Disadvantages
- limited availability of some specialized software
- may require technical expertise to modify or contribute to the code
- challenges in commercializing & monetizing free software
- compatability issues with proprietary soft formats.

OSS

Adv
- access to the source code allows for customization & sec verification
- collab can lead to rapid development & innovation.
- a wide variety of osss is available for various purposes.
- often cost-effective for orgs.

Disad
- less control over the direction of the soft compared to proprietary
- Support & documentation can vary widely.
- some open-source projects may lack funding & resources
- legal complexities surrounding licensing compliance.

Prop

Adv
- typically comes with professional support & documentation.
- well defined user-interfaces and features
- predictable dev. & update schedules
- prop software companies can invest heavily in research & devel

Disadv
- lack of transparency reg how the soft f^n's internally
- users are often bound by licensing agreements that limit usage or redistribution.
- may lead to vendor lock-in, making it diff to switch to alt
- costs can be prohibitive for some orgs or individuals.

* Commercial License
- Ownership: owned & dev by a company or individual. Source code is usually not available for public viewing or modif
- Restrictions: impose restrictions on the use, distribution & mod. These restrictions are defined in license agreement
- Cost: often comes with price tag. users should purchase a license or subscription
- Support: often customer support, updates & maintenance serv
Eg: MS Office, Adobe Photoshop

* Open Source
- Ownership: dev by community of contributors & the source co is made available to the public collab effort.
- License terms: view, modify, use & distribute the softw
- Cost: usually for free, there may be cost with support, customization or maintenance.
- Community support: often have active communities of users & devs who provide supp, documentation & contribute to ongoing de
- Transparency: the source code is open for inspection. Users can see how the soft works & make modif
Eg: Linux, Apache web server, Firefox

2) ϕ

Introduction HTML

- Hypertext Markup Language
- tag based language used to create web pages
- created by Berners-Lee in the late 1991
- current ver is HTML 5.3
- developed with an intention of defining the structure of docs like headings, paragraphs, lists, etc.
- HTML is sent from server to client's browser whenever HTTP request is sent to server.

HTML 5

- DOCTYPE is mandatory
- <html>, <head>, <title> & <body> are mandatory
- elements must be properly nested & closed.
- all the elements should be in lowercase & one root element (root element are called elements). In html, the root element is the <html> element.

HTML	X-HTML
• Hypertext Markup language	• Extensible HTML
• main markup lang for creating web page & other info that can be distinguished in web browser.	• family of XML markup lang that mirrors or extend versions of the widely used HTML, the language in which web page are written.
• .html, .htm text/html document file format	• .xhtml, .xht, .html, .xml, .htm application/xhtml+xml
✗ SGML	Markup language
App of Standard generalized ML	XML, HTML
webpages are written in HTML	App of XML
flexible framework req lenient HTML parser	ext ver of HTML(stricter) & XML-based Restrictive subset of XML & needs to be parsed with standard XML parser
Current ver 5.3	XML 1.1

* Why HTML is called Markup Lan?
- Hypertext → Machine readable text.
markup → structure it in a specific format.
It is a lan that allows users to organize, improve the appearance and link text with data on the Internet.

<html>, <head>, <body>

* Basic text formatting elements
- <h1>, <h2>, <h3>, <h4>, ...
- <p>,
, <pre>

* White Space & flow:
- <pre> element
preformatted text
- text in <pre> element is displayed in a fixed-width font (courier) & it preserve both spaces & line breaks.
Eg. <pre> This is a
pre tag example.
</pre>

-   (non-breaking space) entity reference.

* Presentational Elements.
- handful of (x)HTML elements that are explicitly presentation oriented.
- Sometimes called "physical" styles, they provide instr for the size weight or style of the font used to display the element
- bold element (<b>).
- big element (<big>)
- underline element (<u>)

- Italic element (<i>)
- small element (<small>)
- strikethrough element (<s> or <strike>)
- subscript (<sub>)
- superscript (<sup>)

* Phrase elements:
  - Phase tags are special purpose tags, which defines the structural meaning of a block of text or semantics of text. Following is the list of phrase tags:- ~~some~~
  - Text Abbreviation element:
    write text bt^n <abbr> & </abbr>
  - Marked element
    content written bet^n <mark> & </mark> will show as yellow mark on browser (highlight a particular text).
  - Strong Element:
    display the imp txt of the content.
    text written bet^n & will be displayed as important text.
  - Emphasized element:
    emphasize the text & display in italic form.
    ,
  - Definition element
    <dfn> & </dfn>, allows to sp specify the keyword of the content.
  - Quoting element.
    <blockquote> element shows that the enclosed content is quoted

from another source. Source URL can be given by using
<cite> ... </cite>
Eg: <blockquote cite = "URL"> "....." </blockquote>

- Short Quotation element
  <q> ... </q> defines short quotation, it will enclose the text in double quotes

- Code element:
  <code> </code> displays the part of computer code. It will display the content in monospaced font.

- Keyboard element:
  <kbd> indicates that a set of content is a user input (keyboard)

- Address element
  <address> defines the contact info about the author of the content (italic font)

- Var element:
  <var> used in conjunction with <pre> & <code> elements to indicate that the content of that element is a variable that can be supplied by the user.

* List

- Ordered / Number list

  " "
  " "

- Unordered / Bulleted list

  " "
  " "

- Description/Definition list
  - list style supported by HTML & XHTML.
  - also known as definition list where entries are listed like a dictionary or encyclopedia.
  - present glossary, list of terms or another name-value list.
- <dl> defines the start of the list
- <dt> " a term
- <dd> " the " definition (description)

```
<dl>
    <dt> Aries </dt>
    <dd> - One of the 12 horoscope sign. </dd>
        <dt> Bingo </dt>
    <dd> - One of the my evening snacks </dd>
    <dt> Leo </dt>
    <dd> - It is also one of the 12 h.s </dd>
    <dt> Oracle </dt>
    <dd> - It is a multinational tech comp. </dd>
</dl>
```

* Nested list:

1) Real Madrid
   - Benzema
2) Portugal
   - Ronaldo

```
<ol>
    <li> Real Madrid </li>
    <ul> <li> Benzema </li></ul>
    <li> Portugal </li>
    <ul> <li> Ronaldo </li> </ul>
</ol>
```

* Comments:

```
<!-- Comment goes here -->
```

3) CSS works by allowing us to associate rules with the elements that appear in a web page.

- Selector: indicates which element/elements the declaration applies to (If it applies to more than 1 element we can have comma seperated list of several elements).
- Declaration, which sets out how the elements referred to the selector should be styled

Selector Declaration

td { width: 36 px; }

Property Value

* Table-based design:

In the early days of web development, tables were often used for layout purposes. Web designers used HTML tables to structure the page's layout, placing content in cells. While this approach was widely used, it has several drawbacks:

- Semantic issues: tables are meant for tabular data not layout. It can lead to semantic confusion & accessibility issues.
- Inflexibility: It can be rigid & difficult to modify. Changes in design often requires extensive changes to the table structure.
- Performance: increase page load times & overly impact performance.

```
<!DOCTYPE html>
<html>
 <head>
    <title> Table-Based Design </title>
    <style>
      table {
        width: 100%;
        border-collaspe: collapse;
      }
      td {
        border: 1px solid #000;
        padding: 10px;
```

```
        </style>
    </head>
    <body>
        <table>
            <tr>
                <td> Header 1 </td>
                "    "  2 "
            </tr>
            <tr>
                <td> Content 1 </td>
                "    "  2 "
            </tr>
        </table>
    </body>
    </html>
```

* Table-less design: (CSS-based layout / CSS-driven design) is a modern approach that seperates content from presentation using CSS.
- Semantic Clarity
- Flexibility:
- Accessibility

```
<style>
    .container{
    width: 100%;
    display: flex;
    justify-content: space-between;
    }
    .column{
        flex: 1;
        border: 1px solid #000;
        Padding: 10 px; }
```

```
</style>
</head>
<body>
    <div class="container">
        <div class="column">Header 1</div>
        <div class="column">Header 2</div>
    </div>
    <div class="container">
        <div class="column">Content 1</div>
        <div class="column">Content 2</div>
    </div>
</body>
</html>
```

	CSS	CSS3
Media queries	doesn't support	supports responsive web design
	capable of positioning text & objs.	capable of making web page more attracting & takes less time.
	No sup for modern browsers but works of old vers of Chrome/Enp	Supported fully by modern browsers.
	Not compatible with CSS3.	compatible (backward)
	supports single blocks only.	multi-column text blocks.
	basic animation, no transformation, text animation, transition of 3D animations.	advanced animations & many customization options. also supports text animation, transf & transition.
	no modules	groups codes into convenient modules

uses an old standard color format.	offers gradient colors & schemes like RGBA, HSLA, HSL, etc.
• provides avg performance & requires high memory usage	- opposite

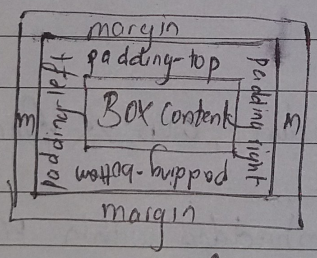
* Box model:

It is a very important concept in CSS because it determines how elements are positioned within the browser window.

It is named because CSS treats every element as if it were a box.

Properties:

- border: every box has has a border even if we can't see it. This seperates the edge of 1 box from another surrounding boxes.
- margin: distance betⁿ the border & the box next to it.
- padding: space between content of the box and its border.



when a bottom margin of one element meets the top margin of the ~~border~~ another only the larger of the two will show.
(this only applies to vertical margin)

border-style, color, width; ;

```
<html>
 <head>
 <title> Image to HTML </title>
 </head>
<body>
 <h1> My Webpage </h1>
```

```
<p> Here is an image: </p>
<img src = "example.jpg" alt = "Dog of the img">
</body>
</html>
```

(4) Introduction

J.S is a versatile & widely used prog language that plays fundamental role in web development. It is crucial technology for building interactive & dynamic websites.

- high-level interpreted scripting language.
- primarily used for web dev but can also be employed in other contexts such as server-side scripting (Node.js), game development & desktop applications.
- enables to add interactivity, manipulate web page content & respond to user actions in real time.

* Key features
  - Client-Side Scripting
  - Multi-Paradigm: procedural, oop & functional programming
  - Cross Platform
  - Lightweight: easily embedded within HTML docs.
  - Extensible: extended through libraries & frameworks (JQuery, React, Angular)

* lexical structure: rules and conventions that dictate how the code is written and structured. These rules help the prog language execute and interpret the code properly correctly.
  - Case Sensitivity
  - Whitespace
  - Semicolons
  - Comments
  - Identifiers
  - Keywords
  - Literals
  - Operators
  - String Templates
  - Escape Sequence

```
<base href="" "/>
<a href="" " target="_blank"><p> <> <h>
<a href="" " target="_blank"><img src="" "></a>

<video controls width="480" height="360">
  <source src="" "></video>
<audio controls>
  <source src="" ">
</audio>

<style>
  table tr td{
    text-align: center;
    width: 200px;
    background-color: dark cyan;
    color: white;
  }
  table tr th{
    background-color: greenyellow;
  }
</style>
```

h1+h2 are adjacent. so only changes in h2 → (if need same in style, everything)

id="f" text #f{ Class="f" .f{

<h1 color

font-size

font-weight: bold;

" -style: italic

text-align:

text-transform: uppercase overline

text-decoration: line through, ~~line through~~

letter-spacing:

word spacing:

}

ul{

color

font-size

" -weight

text-align }

```
<link rel="stylesheet" href="style.css">
<div class="container">
  <div class="box1">...</div>
  <"  "  2  "
  <"  "  3  "
  <"  "  4

</div>
<div class="container1">
  <div class="box1">.....</div>
  <div "  = 2
  </div>

.box1 {
  border: 2px solid red;
  border-bottom: 2px blue;
  padding: 30px 40px 50px 60px
  width: 30px;
  height: 50px
  margin: 50 20 30 40;
  overflow: scroll;
}

.container { display: flex; grid
  1 {  "  2
  3 }

<div class="container">
  <div class="slide-container">
    <div class="slide-image">
      <img src="img1.jpg">
    </div>
    :
    :
  </div>
</div>
```

```
.container {
    height:
    width
    margin: auto
    display
    place-items: }
.slide container{
    display:
    width:
    animation: slide 20s infinite linear;
}
@keyframes slide{
    0% { transform: ~~transform~~ translateX(0); }
    100% {                          X(-1000px); }
.side-image{
    padding: 20px;
    perspective: 100px; }
img { width: 300px; transition: transform 1s; }
img: hover{
    transform: translateZ(20px); }
```

link

```
<a href        ></a>
a{                  a:active{     a:hover{     a:visited{
    color:          color:        color:       color:
    background color:   }             }            }
    font-decoration:
}
```

active → when you press the link
hover overlaps active ✓
visited overlaps hover & active when refreshed

list

<list class = "a"> lop
"b" >
"c" >
 div i'

li.a{ list-style: square; }
.b disc
.c lower-roman

add(){
var a = document.getElementById("first").value;
b = second
a = parseInt(a);
b = " " (b);
var sum = a+b;
document.getElementById("result").value = sum;
" " " ("para").innerHTML = sum. (display in page itself instead of text in

<script src = "mutiple.js"> </script>

 Email: <input type = "text" id user">
<button id = "validateButton" onclick = "validate()"> Validate </button>
<script>
function validate(){
var email = doc.getEbl("user").value
var regExp = /^([a-zA-Z0-9\._-]+)@([a-z0-9]+)
([a-z]?)$/;
if (regExp.test(email)){
alert("Vaid Email!!.");
ret true;
}
else {
alert("
doc.geEbl("user").value = "";
ret false

Remote User
Can Access
DB Through
Internet
User
Bluetooth
Shared
Mobile
Phones
Internet of things
Firewall
Http Port
end
Database
Access Control
Authentication
Non-Repudiation
Data Confidentiality
Communication Security
Data Integrity
Availability
Privacy
Security
Dimensions
Threats
Destruction of resources
by unauthorized access
Information Corruption
Removal or theft
of resources
Information
Disclosure
Attacks

4) JS Objects

- versatile data structures that allows us to store & manipulate collections
- dynamic - add, remove or modify properties at any time.

* Declaration & Initialization

- literal notation: defining an curl object using curly braces and assigning properties & their values directly inside.
- concise & straightforward way to initialize objects.

- constructor fn: allows us to define a blueprint for creating multiple instances of an object.

```
this.objed = object
```

- factory fn: that returns an instance of a class or an object.
- encapsulate the process of obj creation, providing a convenient way to create objects and potentially allowing for more flexibility in the instantiation process.

* Assigning Object properties:

- Dot notation: use the object's name followed by a dot & the property's name.
- Bracket notation: allows us to access properties using a string value inside square brackets

Benefits of using Js objects

- Structuring data
- code reusability
- flexibility
- Object-oriented programming
- Efficient data manipulation

PHP

- Hypertext Preprocessor
- Server side scripting language embedded in HTML
- manage dynamic content, databases session tracking
- integrated with databases like MySQL, Oracle, Microsoft SQL Server.

* Why PHP?
- faster to code & execute
- same PHP code runs unaltered on diff Web servers & diff Operation sys
- PHP standard functionality is an add-on in other environments
- free, runs on any O.S
- most popular web server on the Internet, Apache's commercial flavors like Web Ten & Stronghold support PHP.

* Strength of Php:
- Supports database connectivity
- Supports sessions
- eliminates client conf problem
- reduces development time
- maintains source code security

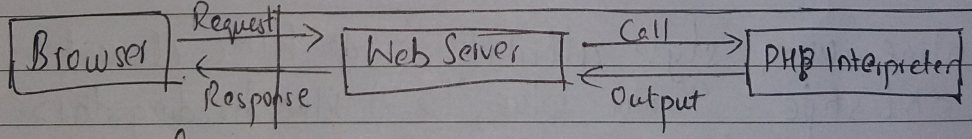


fig: How PHP script works

Basic Syntax:

```
<?php echo "Hello, World!"; ?>
```

Comment:

```
<?php
//This is a single line comment
#This is also a single line comment
echo "Hello, world!"; ?>
```

Case Sensitivity:

Var names in PHP are case-sensitive

```
<?php
$color = "blue";
echo "The color of the sky is " . $color . "<br>";
"     "   "  "  "  "  "  " . $Color . "<br>";
"     "   "  "  "  "  "  " . $COLOR . "<br>";
?>
```

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

```
<?php
$many = 2.2888800;
$many_2 = 2.2111200;
$few = $many + $many_2;
print("$many + $many_2 = $few <br>");
?>
```

Output

2.2888800 + 2.2111200

```
<h1><?php echo "Hello World" ?></h1>
```

Methods to link

- Inline: using style attribute inside HTML elements

```
<h1 style = "color: blue;"> A Blue heading </h1>
<p style = "color: red;"> A red paragraph. </p>
```

- Internal: using <style> element in <head> section

```
<style>
  body {
  bg-color: blue;
  }
</style>
```

- External: by using a <link> element to link external css file.
```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

* Selectors
- Universal *{}
- Type / Group h1,h2,h3{}
- Class
```
<p class="BackgroundNote">
.BackgroundNote{}
```
- Id
```
#
```

- for loop:
```
<?php
for($i=1; $i<=5; $i++){
    echo "This is iteration $i <br>";
}
?>
```

- While loop:
```
<?php
$j=1;
while($j<=5){
    echo "This is iteration $j <br>";
    $j++;
}
?>
```

- If-else

```
<?php
$num = 10;
if ($num > 0) {
    echo "The no is +ve.";
} else if ($num < 0) {
    echo "The no is -ve";
} else {
    echo "The no is 0.";
}
?>
```

- Switch

```
<?php
$day = "Monday";
switch ($day) {
    case "Monday":
        echo "It is the beg of the week.";
        break;
    case "Friday":
        echo "It is almost weekend!";
        break;
    default:
        echo "It is regular day.";
}
?>
```

- Creating & Calling a fn

```
<?php
function greet($name) {
    echo "Hello, $name!";
}
greet("John"); ?>
```

- Indexed arrays

```
<?php
$fruits = array("Apple", "B", "O");
echo $fruits[0];
          [1];
          [2];
?>
```

- Associative arrays

```
<?php
$person = array(
    "name" => "John",
    "age" => 30,
    "city" => "New York"
);
echo $person["name"];
          ["age"];
          ["city"];
?>
```

- Connecting PHP to database

```
$servername = "localhost";
$username = "root";
$password = "password";
$dbname = "mydatabase";
$conn = new mysqli($s, 4, p, d);
if ($conn->connect_error){
    die("Connection failed:" . $conn->connect_error);
}
echo "Connected Successfully";
?>
```

```php
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
    $username = $_POST["username"];
    $password = $_POST["password"];

$sql = "INSERT into users(username, password)
    VALUES('$username', '$password')";

if ($conn->query($sql) === TRUE){
    echo "Record inserted successfully";
} else{
    echo "Error: ". $sql "<br>". $conn->error;
}
    $conn->close();
}
?>
```

```php
$sql = "SELECT * from auth WHERE email = '$email' AND
password = '$password'";
$result =
        mysqli_query($conn, $sql);

if ($result){
    if(mysqli_num_rows($result) == 1){
        echo "Login Successful!";
    } else{
        echo 'Invalid Email or Password';
    }
else{
    echo 'Error: ' . mysqli_error($conn);
}
mysqli_free_result($result);
}mysqli_close($conn);
```

id = "Male" name = "Gender" value = "Male"
<label for = "Country"> Country </label>
<select name = "Country" id = "country" >
<option Value = "Nepal"> Nepal </option>

</select>