Web Programming

CHAPTER 1: Overview of Web Programming

Outlines

- What is Programming?
- What is a website ?
    - How a Web site works
    - Types of websites
- What is scripting?
    - Types of scripting?
- Intro to HTML and CSS

Discussion

- What is Programming?
- What is Programming?

1.1 What is Programming?

- Programming is the intricate **art of telling a computer what to do**.
- **It is giving written instructions in a logical manner** that the computer can understand.
- Essentially, you give the computer **small steps of instructions**, and the computer goes down the list, doing each one in order.
- Programming allows you to make new software and have the computer do new things.
- Web site programming is the same **except you write applications or web pages that are used by a web browser.**

1.2 What Does Web Programming Mean?

- Web programming refers to the **writing, markup and coding involved in Web development, which includes Web content, Web client and server scripting and network security.**
- Web programming is the practice of writing applications that r**un on a web server** and can be used by many different people. **Web programming allows you to turn a simple, static HTML page into a dynamic masterpiece**.
- The most common languages used for Web programming are XML, HTML, JavaScript, Perl 5 and PHP.
- Web programming is different from just programming, which requires interdisciplinary knowledge on the application area, client and server scripting, and database technology.

- Web programming can be briefly categorized into client and server coding.
- The client side needs programming related to accessing data from users and providing information.
- It also needs to ensure there are enough plugins to enrich user experience in a graphic user interface, including security measures.
- To improve user experience and related functionalities on the client side, JavaScript is usually used.
- It is an excellent client-side platform for designing and implementing Web applications.
- HTML5 and CSS3 supports most of the client-side functionality provided by other application frameworks.

- The server side needs programming mostly related to data retrieval, security and performance.
- Some of the tools used here include ASP, Lotus Notes, PHP, Java and MySQL.
- There are certain tools/platforms that aid in both client- and server-side programming.
- Some examples of these are Opa and Tersus.
- Most websites nowadays use server-side scripting to display all data the client needs dynamically on the screen.
- These data are generally gathered from a database stored on a server and sent to the client (The Browser).
- The client can now see it via other scripting codes such as JavaScript and HTML.

- One of the essential benefits of server-side scripting is that it gives access to create website content for individual users.
- A dynamic website can highlight or make suggestions for the most relevant content for the users; this process is based on users’ habits and history.
- One more benefit is that it can store your personal information. For more illustration, when you use your stored credit card details for future payments.
- Server-side scripting allows interacting with users, sending notifications & updates through emails or any other channels. All of these features allow for far more in-depth interaction with users.

What is a website ?

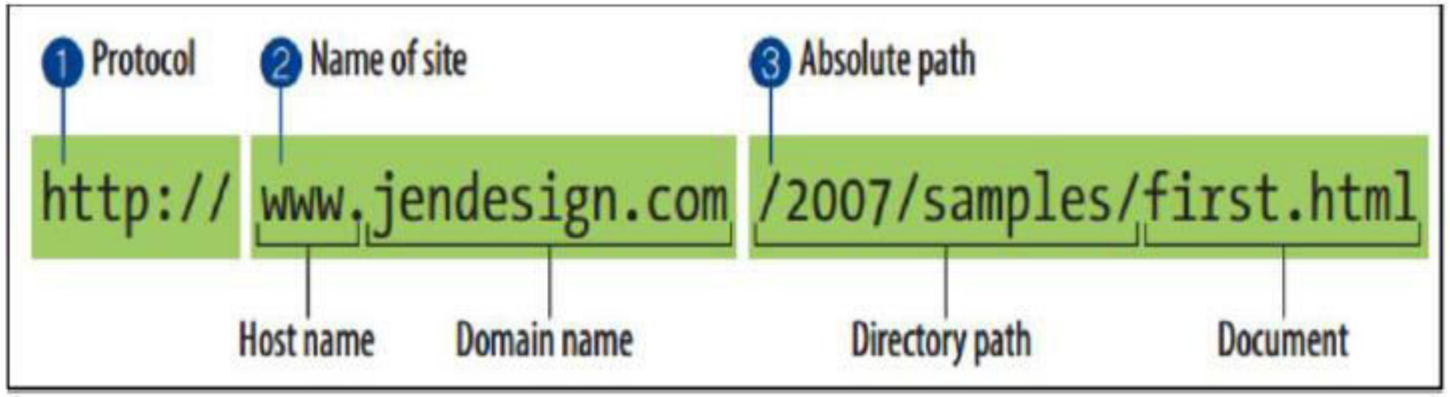
- Website is a **collection of related web pages that may contain text, images, audio and video.**
- A web page is a way **to display information on the internet.**
- The first page of a website is called home page.
- Each website has specific internet **address (URL)** that you need to enter in your browser to access a website.
- Based on the information those pages contain, **they are organized into an information hierarchy**– this allows navigation from one page to another.
- The overall **collection of those related web pages is a website.**
- Website is hosted on one or more servers and can be accessed by visiting its homepage using a computer network. A website is managed by its owner that can be an individual, company or an organization.

How Websites Work ?

- Let’s start with the most obvious way of using the internet: You visit a website like **academind.com.**
- The moment you enter this address in your browser and you hit ENTER, a lot of different things happen:
    1. The URL gets resolved
    2. A Request is sent to the server of the website
    3. The response of the server is parsed
    4. The page is rendered and displayed

Uniform Resource Locator (URL)

- To visit a Web site, users type the URL, which is the site's address, into the web browser. An example of a URL is www.yahoo.com.
- A complete URL is generally made up of three components: the protocol, the site name, and the absolute path to the document or resource as shown in the figure below:



Advanced URLs

- Anchor: jumps to a given section of a page
  - http://en.wikipedia.org/wiki/HTML_element#Anchor
- Fetches the HTML_element document, then jumps to the part of the page labeled Anchor
- Port: for web servers on ports other than the default port 80
  - http://portquiz.net:8080/index.php

Advanced URLs

- **Query string**: a set of parameters passed to a web application

  http://www.google.com/search**?q=miserable+failure&start=10**

  - parameter named q is set to value miserable+failure
  - Parameter named start is set to value 10

Hypertext Transfer Protocol (HTTP/HTTPS)

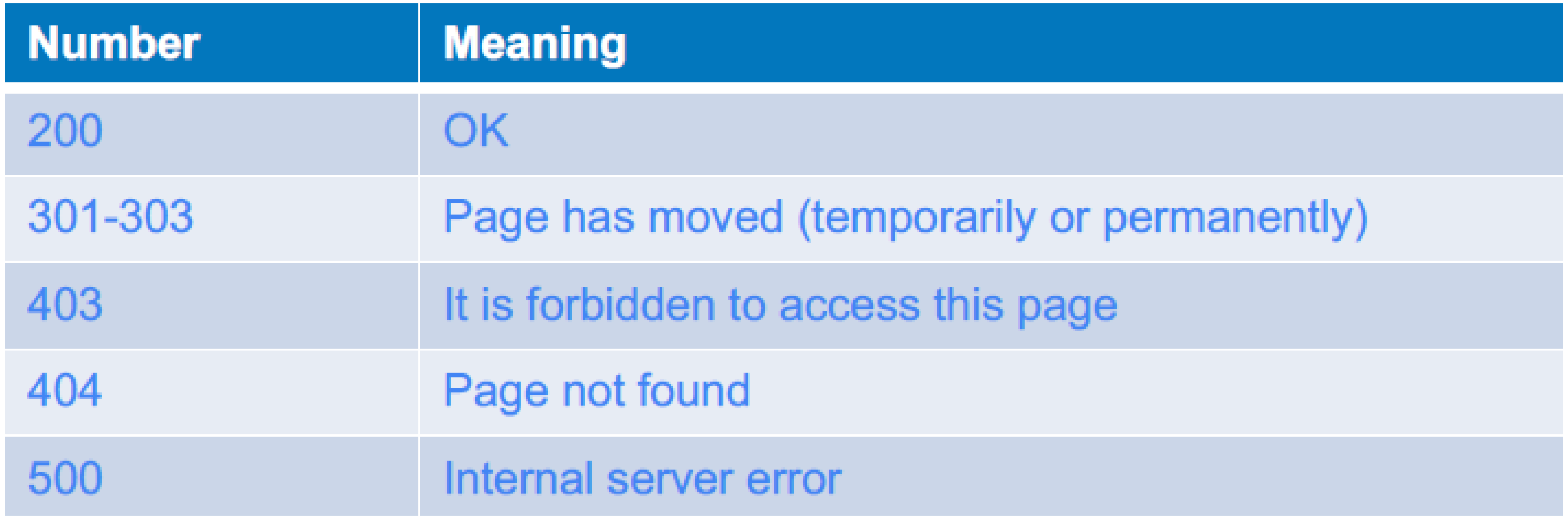
- It is a communications protocol.
- It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs.
- Simply, it is the means by which computers on the WWW communicate.
- HTTPS is the secure version of HTTP. It is used on web sites where sensitive information such as bank details is exchanged.

Hypertext Transfer Protocol (HTTP)

- Defines a set of commands understood by a Web server and sent from a browser
- Some HTTP commands (your browser sends these internally)
  - GET resource -- requests data from a specified resource
  - POST resource -- submits data to be processed to a specified resource
  - PUT resource -- uploads a representation of the specified URL
  - DELETE resource -- deletes the specified resource

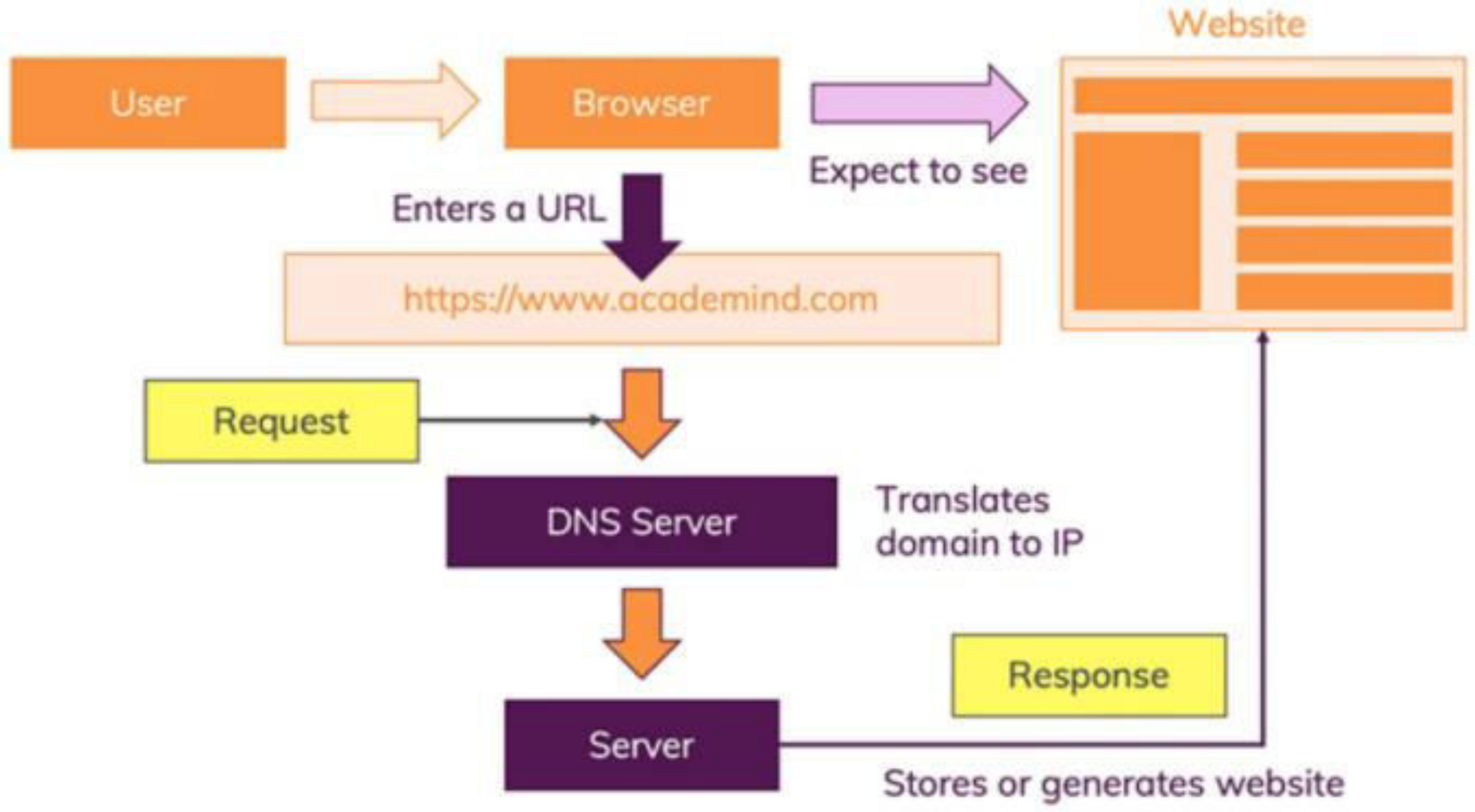
HTTP status codes

- When a request is made by the browser, a response is sent back by the server with a
- status code, possibly followed by a Web resource



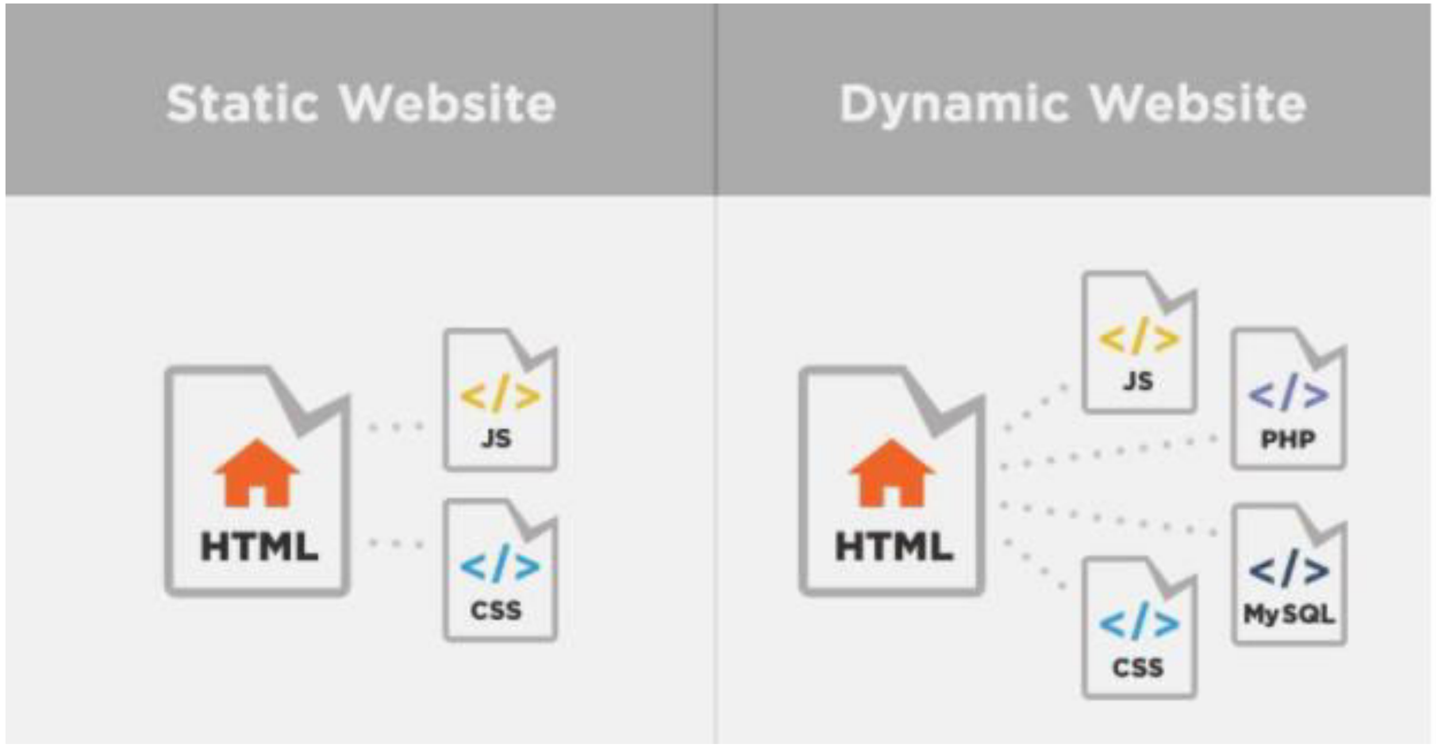
Number	Meaning
200	OK
301-303	Page has moved (temporarily or permanently)
403	It is forbidden to access this page
404	Page not found
500	Internal server error

How Websites Work

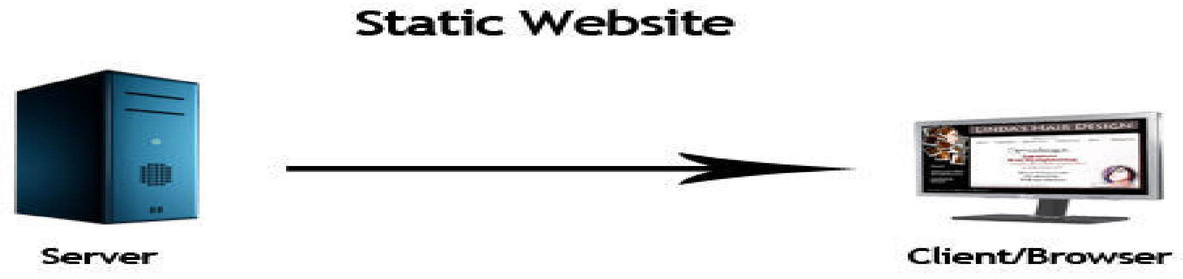


Types of websites

- A website can be of two types:
  - Static Website
  - Dynamic Website



Static Website



- Static website is the basic type of website that is easy to create. You don't need the knowledge of web programming and database design to create a static website. Its web pages are coded in HTML.
- The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.
- A static web page is straightforward, known as a flat or stationary web page.
- They are exactly loaded as they are on the web server into the client’s browser.
- Written in languages such as (HTML, CSS, JS, Bootstrap …) Hardcoded content cannot be changed by the user but only from the code itself. On the static page, when the server receives some request from the browser by HTTP “GET” request, it sends a response to the client-side containing the document and success status (200 OK) without doing any additional process.

Advantages of Static Website:

- No programming skills are required to create a static page.
- Inherently publicly cacheable (i.e. a cached copy can be shown to anyone).
- No particular hosting requirements are necessary.
- Can be viewed directly by a web browser without needing a web server or application server, for example directly from a CDROM or USB Drive.

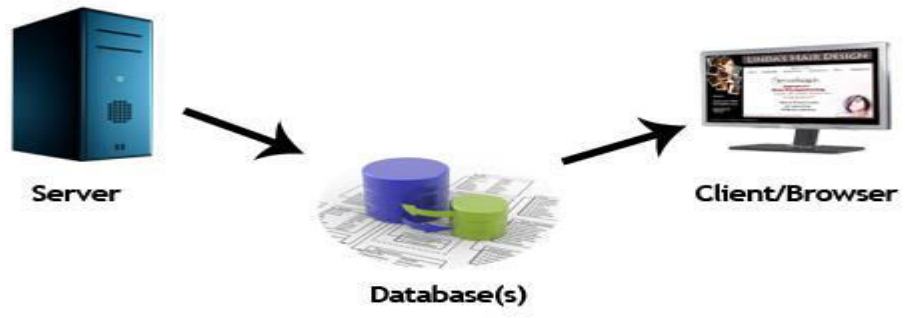
Disadvantages of Static Website:

- Any personalization or interactivity has to run client-side (ie. In the browser), which is restricting.
- Maintaining large numbers of static pages as files can be impractical without automated tools

Application areas of Static Website:

- Need of Static web pages arise in the following cases.
  - Changes to web content is infrequent
  - List of products / services offered is limited Simple e-mail based ordering system should suffice
  - No advanced online ordering facility is required
  - Features like order tracking, verifying availability of stock, online credit card transactions, are not needed
  - Web site not required to be connected to back-end system.

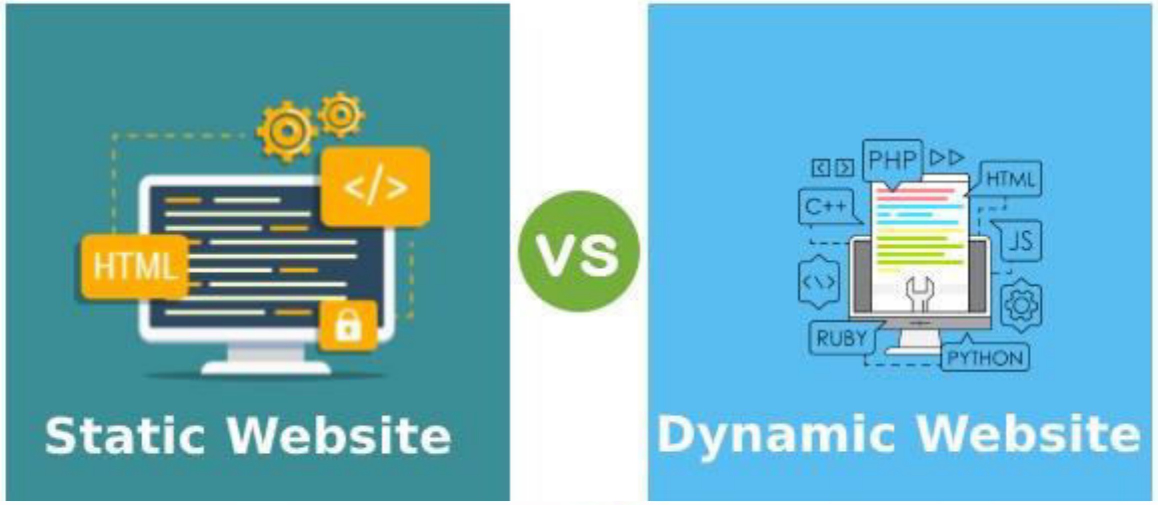
Dynamic Website

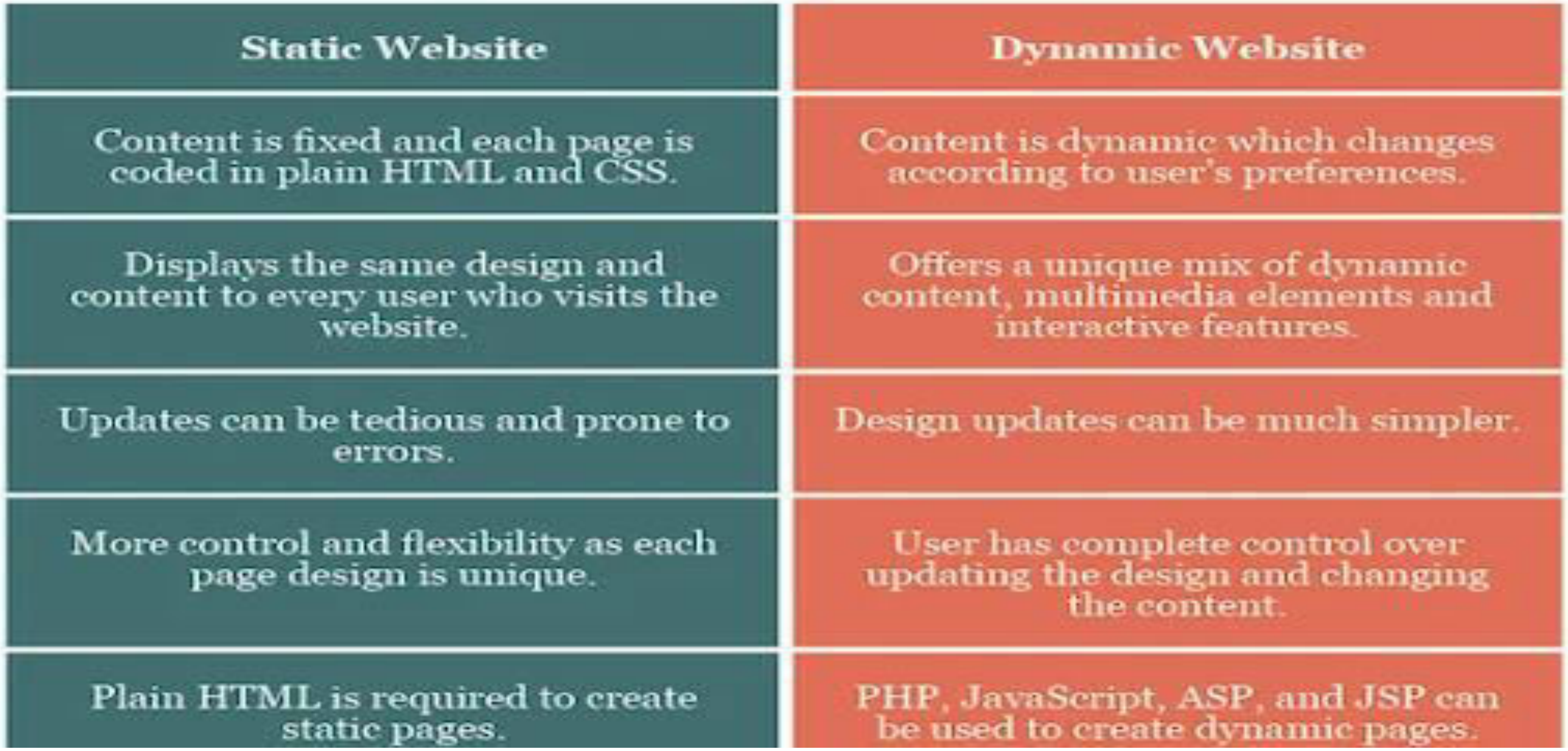


- Dynamic website is a collection of dynamic web pages whose content changes dynamically.
- It accesses content from a database or Content Management System (CMS).
- Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.
- Dynamic website **uses client-side scripting or server-side scripting**, or both to generate dynamic content.
- Written in languages such as (AJAX, PHP, ASP.NET …) When the content on a website changes regularly, such as stock prices or weather forecasts, dynamic web pages are used.
- The dynamic pages' content is generated dynamically only when needed or allowed to be generated. It is generated from the database on the server. For example, when displaying all the posts posted by your friends on Facebook on your timeline.

Application areas of Dynamic Website

- Dynamic web page is required when following necessities arise:
- Need to change main pages more frequently to encourage clients to return to site.
- Long list of products / services offered that are also subject to up gradation
- Introducing sales promotion schemes from time to time
- Need for more sophisticated ordering system with a wide variety of functions
- Tracking and offering personalized services to clients.
- Facility to connect Web site to the existing back-end system





Static Website	Dynamic Website
Content is fixed and each page is coded in plain HTML and CSS.	Content is dynamic which changes according to user's preferences.
Displays the same design and content to every user who visits the website.	Offers a unique mix of dynamic content, multimedia elements and interactive features.
Updates can be tedious and prone to errors.	Design updates can be much simpler.
More control and flexibility as each page design is unique.	User has complete control over updating the design and changing the content.
Plain HTML is required to create static pages.	PHP, JavaScript, ASP, and JSP can be used to create dynamic pages.

What is scripting?

- Before talking about server-side scripting, First, we need to know some main points before we go deeper. We have to know firstly, what is the scripting?
- Secondly, what is the difference between Server-Side Scripting & Client-Side Scripting?

- **Scripting means all series of commands that can be executed without compiling (the transformation from Source Code (human-readable) into machine code).**

- Note: All scripting languages are programming languages, but **not all programming languages are scripting languages!**

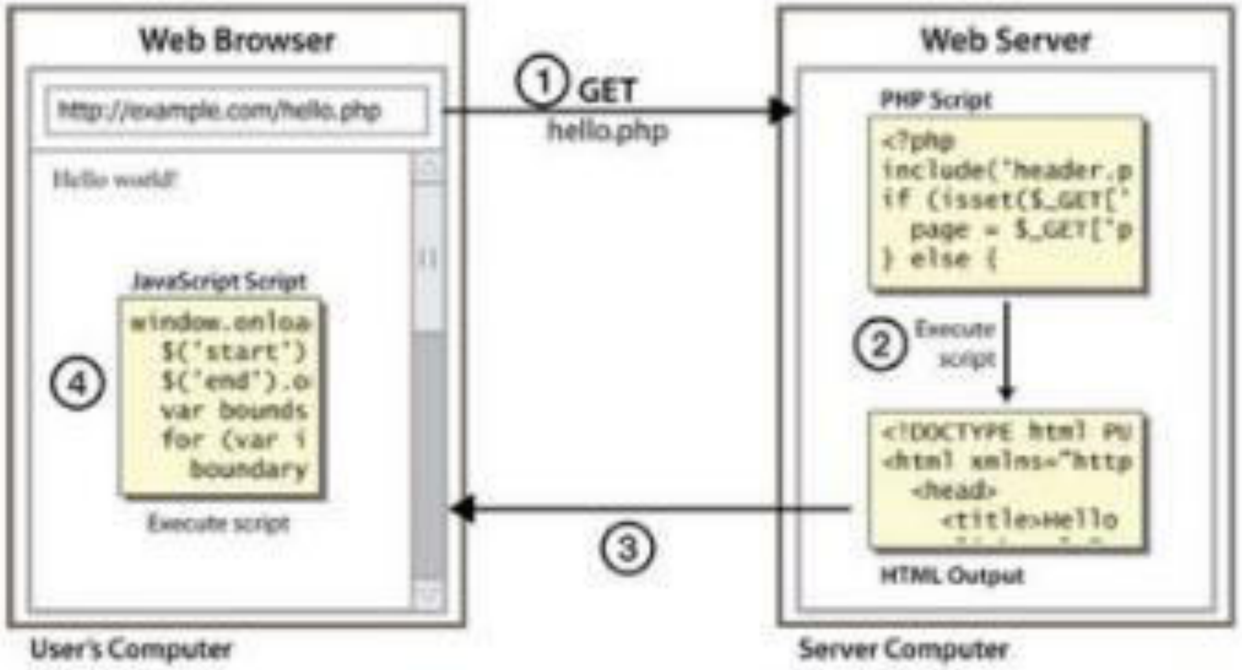
Client Side Scripting

- Client-side scripting generally refers to the class of computer programs on the web that are **executed client-side**, **by the user's web browser.**
- Client-side scripts are often **embedded within an HTML or XHTML** document (hence known as an "embedded script"), but they may also be contained in a **separate file**, to which the document (or documents) that use it make reference (hence known as an "external script").
- instructions can be followed **without further communication with the server**
- By viewing the file that contains the script, users may be able to see its source code.

---cont

- Many web authors learn how to write client-side scripts partly by examining the source code for other authors' scripts.
- The client-side environment used **to run scripts is usually a browser**.
- The processing takes place on the end users computer. The source code is transferred from the web server to the user‘s computer over the internet **and run directly in the browser.**
- The scripting language **needs to be enabled on the client computer.**
- Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.

Client-side scripting



- **client-side script**: code runs in browser *after* page is sent back from server often this code manipulates the page or responds to user actions

Client Side Scripting example

```
<html>
<body>
<script language="JavaScript">;
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
    document.write(i + "! = " + fact);
    document.write("<br>");
}
</script>
</body>
</html>
```

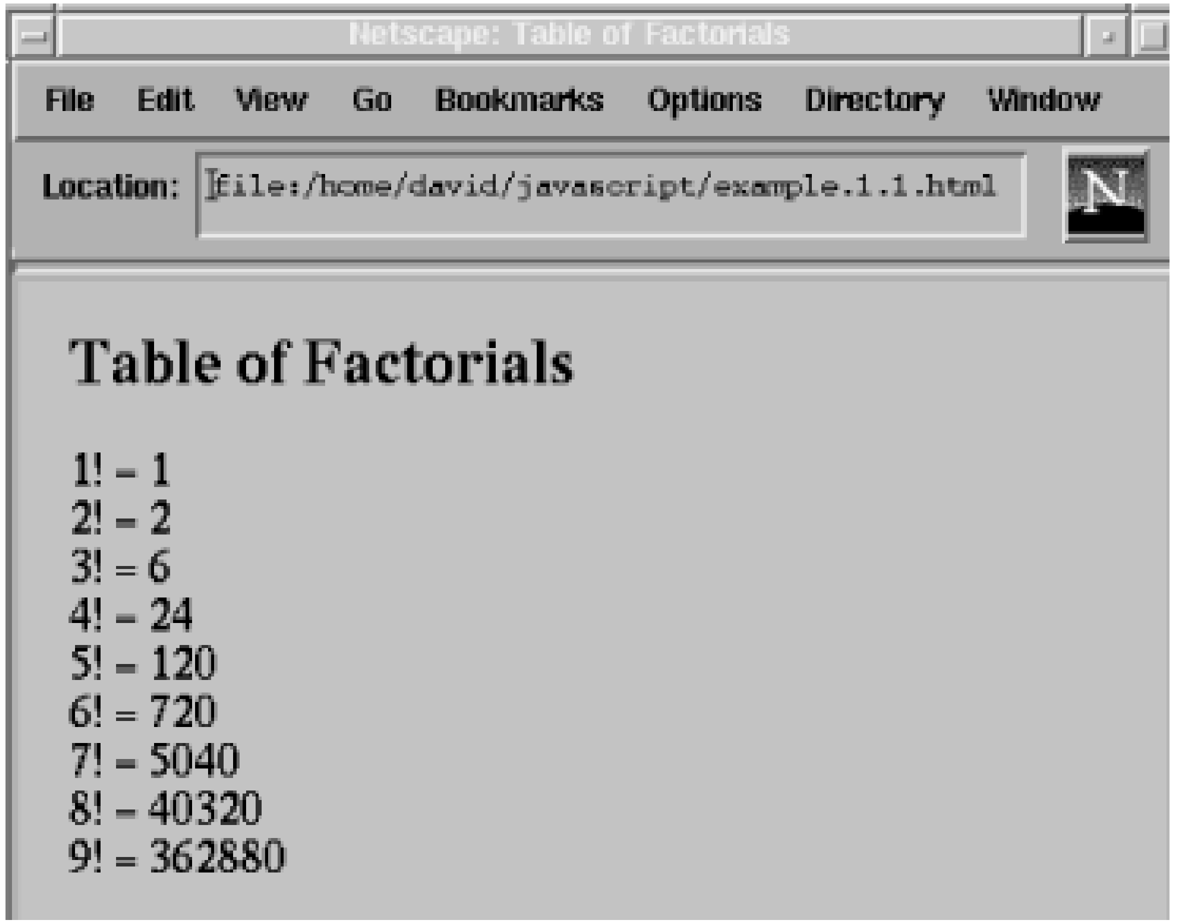

Netscape: Table of Factorials
File
Edit
View
Go
Bookmarks
Options
Directory
Window
Location: file:/home/david/javascript/example.1.1.html

Table of Factorials

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
```

Server Side Scripting

- Server-side scripting language, which means that **the scripts are, executed on the server, the** computer where the Web site is located.
- Server-side scripting is a web server technology in which a user's request is fulfilled by running a **script directly on the web server to generate dynamic web pages.**
- It is usually used to provide interactive web sites that interface to databases or other data stores.
- This is different from client-side scripting where scripts are run by the viewing web browser.
- The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.
- From security point of view, server-side scripts are **never visible to the browser** as these scripts are executes on the server and emit HTML corresponding to user's input to the page.

---cont

- In contrast, server-side scripts, written in languages such as PHP, ASP.NET, Java, ColdFusion, Perl, Ruby, Go, Python, and server-side JavaScript, are executed by the web server when the user requests a document.
- **They produce output in a format understandable by web browsers (usually HTML),** which is then sent to the user's computer.
- The **user cannot see the script's source code** (unless the author publishes the code separately), and may not even be aware that a script was executed.
- Documents produced by server-side scripts may, in turn, contain client-side scripts.

---cont

- Server-side Web scripting is mostly about connecting Web sites to back end servers, such as databases. This enables two-way communication:
  - Client to server: Customer-entered information as request.
  - Server to client: Web pages can be assembled from back end-server to give output.

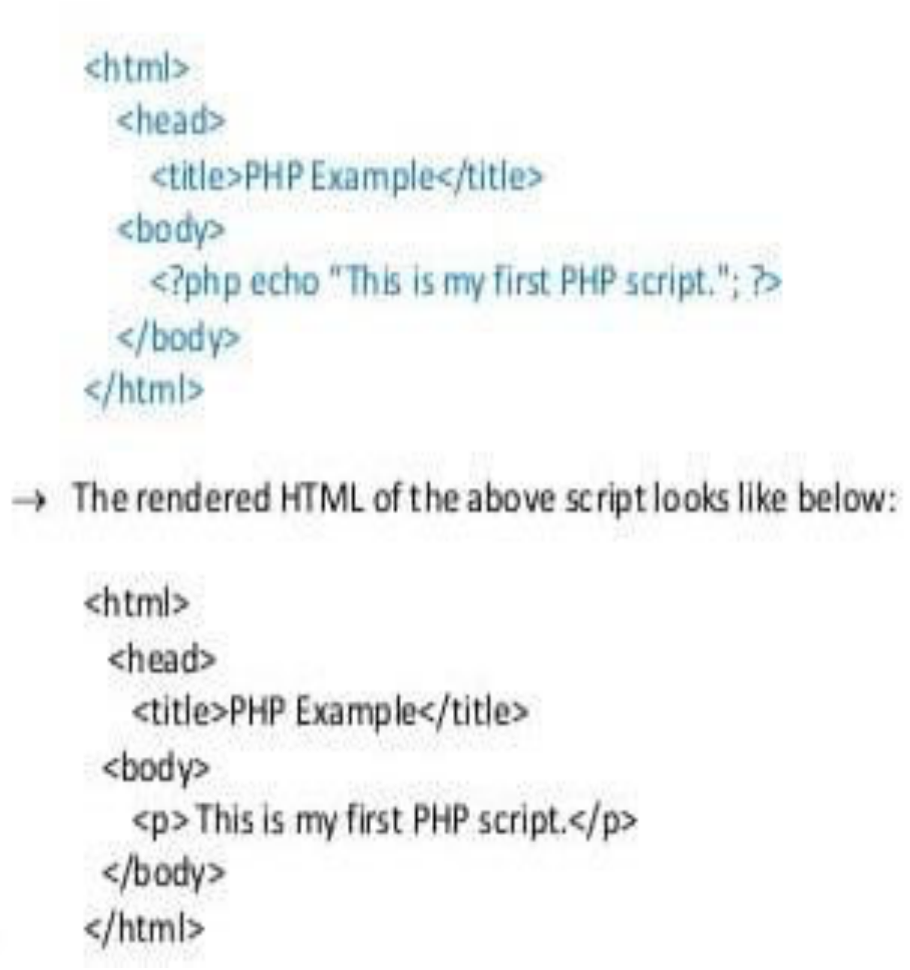
---cont

- Server-side scripting is about "programming" the behavior of the server while client-side scripting is about "programming" the behavior of the browser.
- Normally, when a browser requests an HTML file, the server returns the file. However, if the file contains a server-side script, the script is executed on the server before the file is returned to the browser as plain HTML.
- A server script can do:-
  - Dynamically edit, change or add any content to a Web page
  - Respond to user queries or data submitted from HTML forms
  - Access any data or databases and return the result to a browser
  - Customize a Web page to make it more useful for individual users
  - Provide security since server code cannot be viewed from a browser

---cont

- In server side script, since the scripts are executed on the server, the browser that displays the file does not need to support scripting at all. The followings are server-side scripts:
    - PHP (*.php)
    - Active Server Pages (ASP)
    - ANSI C scripts
    - Java via JavaServer Pages (*.jsp)
    - JavaScript using Server-side JavaScript (*.ssjs)
    - Lasso (*.lasso) etc
- The main focus here is PHP, which is a server-side scripting language, which can be embedded in HTML or used as a standalone binary, and could be run with open source software like WAMP server.
    - PHP can dynamically create the HTML code that generates the Web page.
    - Web page visitors see the output from scripts, but not the scripts themselves.

Client Side Scripting example in PHP



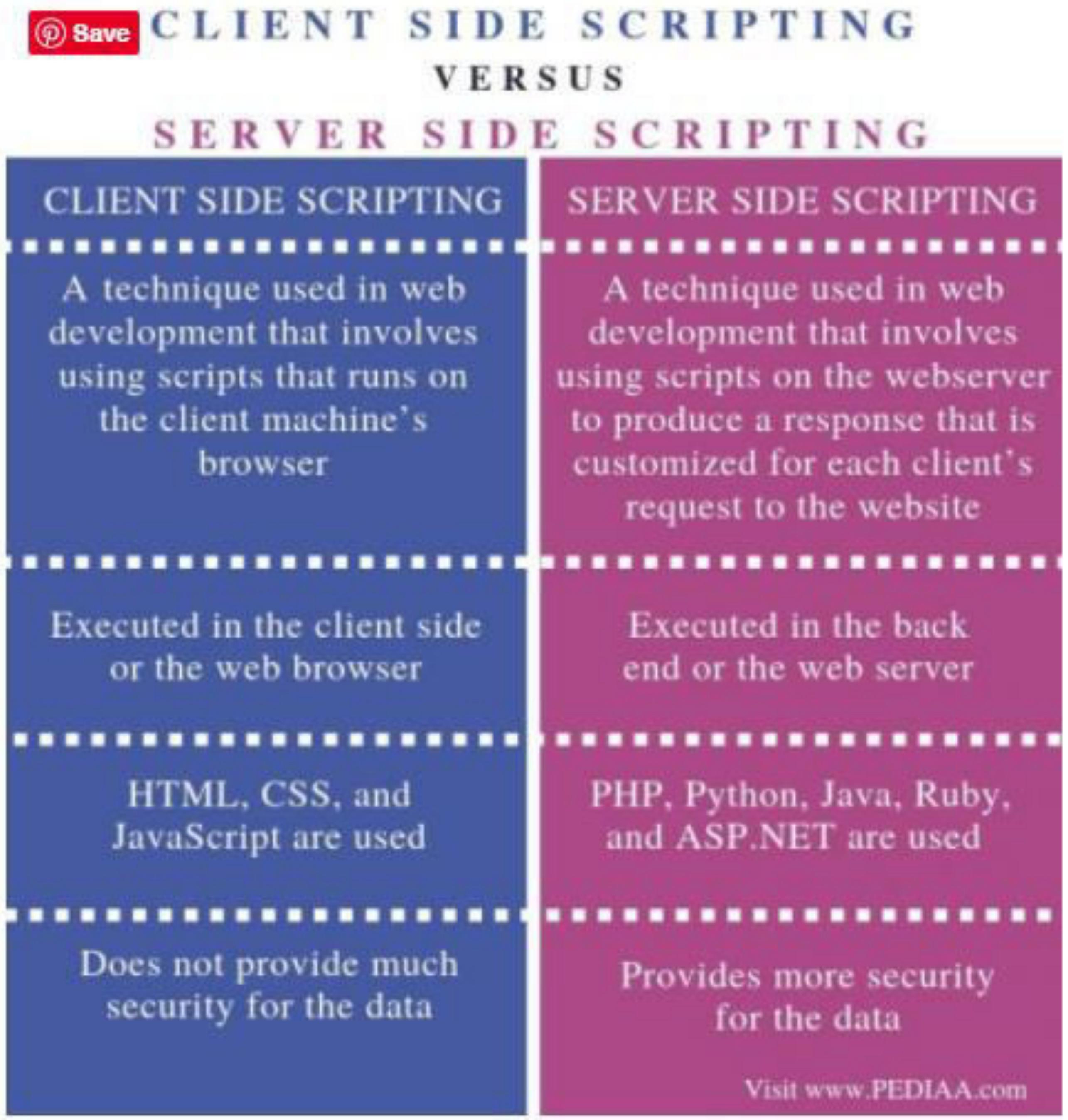
```
<html>
  <head>
    <title>PHP Example</title>
  <body>
    <?php echo "This is my first PHP script."; ?>
  </body>
</html>
```

→ The rendered HTML of the above script looks like below:

```
<html>
  <head>
    <title>PHP Example</title>
  <body>
    <p> This is my first PHP script.</p>
  </body>
</html>
```

Key Differences Between Server-Side Scripting & Client-Side Scripting

- Server-Side Scripting is used in the back-end, where the website's source code is hidden at the client-side. On the other hand, client-side scripting is used at the front-end, which users can see from their browser.
- When a server-side script is run, it sends a request to the server. Client-side scripting, on the other hand, does not require any server interaction.
- HTML, CSS, and JavaScript are examples of client-side scripting languages. Programming languages such as PHP, ASP.net, Ruby, ColdFusion, Python, C#, Java, C++, and others, on the other hand.
- Server-side scripting helps personalize web pages and make dynamic changes to websites. On the other side, the client-side script will effectively reduce the load on the server.



CLIENT SIDE SCRIPTING
VERSUS
SERVER SIDE SCRIPTING

CLIENT SIDE SCRIPTING	SERVER SIDE SCRIPTING
A technique used in web development that involves using scripts that runs on the client machine's browser	A technique used in web development that involves using scripts on the webserver to produce a response that is customized for each client's request to the website
Executed in the client side or the web browser	Executed in the back end or the web server
HTML, CSS, and JavaScript are used	PHP, Python, Java, Ruby, and ASP.NET are used
Does not provide much security for the data	Provides more security for the data

Visit www.PEDIAA.com

Introduction to Web Servers

- A web Server is a program that automatically downloads pages from the Web
- An application server works with a Web server to handle requests for dynamic content, such as servlets, from Web applications. A Web server uses a Web server plug-ins to establish and maintain persistent HTTP and HTTPS connections with an application server.
- The web server software offers access to documents stored on the server.
- Clients can browse the documents in a web browser.
- The documents can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), Javascript or Perl files. The communication between clients and server is based on the Hypertext Transfer Protocol (HTTP)
- We have the Following Web Servers
  - Apache   2.WAMP   3.XAMPP

Apache

- Introduced in 1995 and based on the popular NCSA httpd 1.3, Apache is now the most used web server in the world.
- One of the reasons to its popularity is that it is free to use. Also, since the source code is free, it is possible to modify the web server
- Being threaded (threaded or process-driven depending on the operating system, on Unix, Apache uses processes, while threads are used in Win32 environments) means that Apache maintains a pool of software threads ready to serve incoming requests.
- When a request enters the web server, it is assigned one of the free threads, that serves it throughout the requests' lifetime.

- Apache puts a limit on the number of threads that are allowed to run simultaneously.
- If that number has been reached, requests are rejected.
- Over 56% of Internet Web servers run Apache or an Apache derivative
- Apache is the most commonly used Web Server on Linux systems. Web Servers are used to serve Web Pages requested by client computers. Clients typically request and view Web Pages using Web Browser applications such as Firefox, Opera, or Mozilla

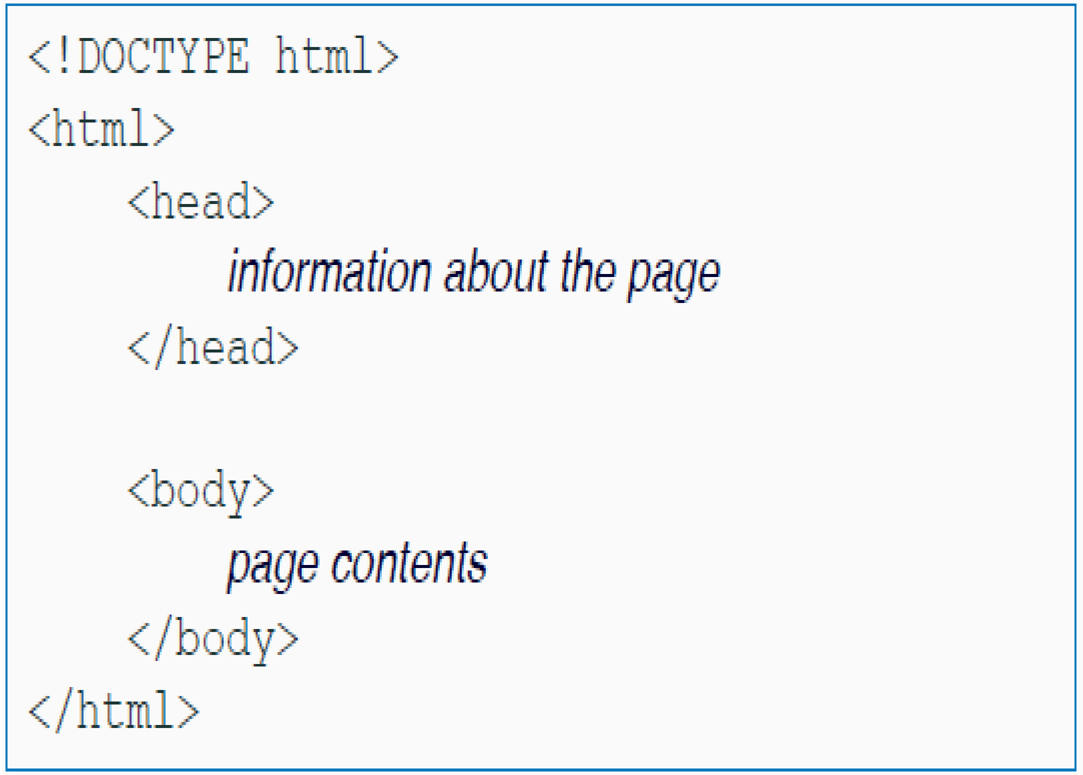
HTML Overview

Basics of HTML

- Defines the content and structure of information on a page
  - Not the same a presentation (appearance in the browser)
- Surrounds text content with opening and closing tags
- Each tag’s name represents an HTML element
- Syntax: <tagname>Content goes here...</tagname>
- Most whitespace is collapsed or ignored in HTML
- We will use HTML5 syntax

Structure of HTML page

- DOCTYPE tells browser to interpret code as HTML5
- HTML page is save in a file with extension .html
- The header describes the page, and the body holds the page’s content



```
<!DOCTYPE html>
<html>
    <head>
        information about the page
    </head>

    <body>
        page contents
    </body>
</html>
```

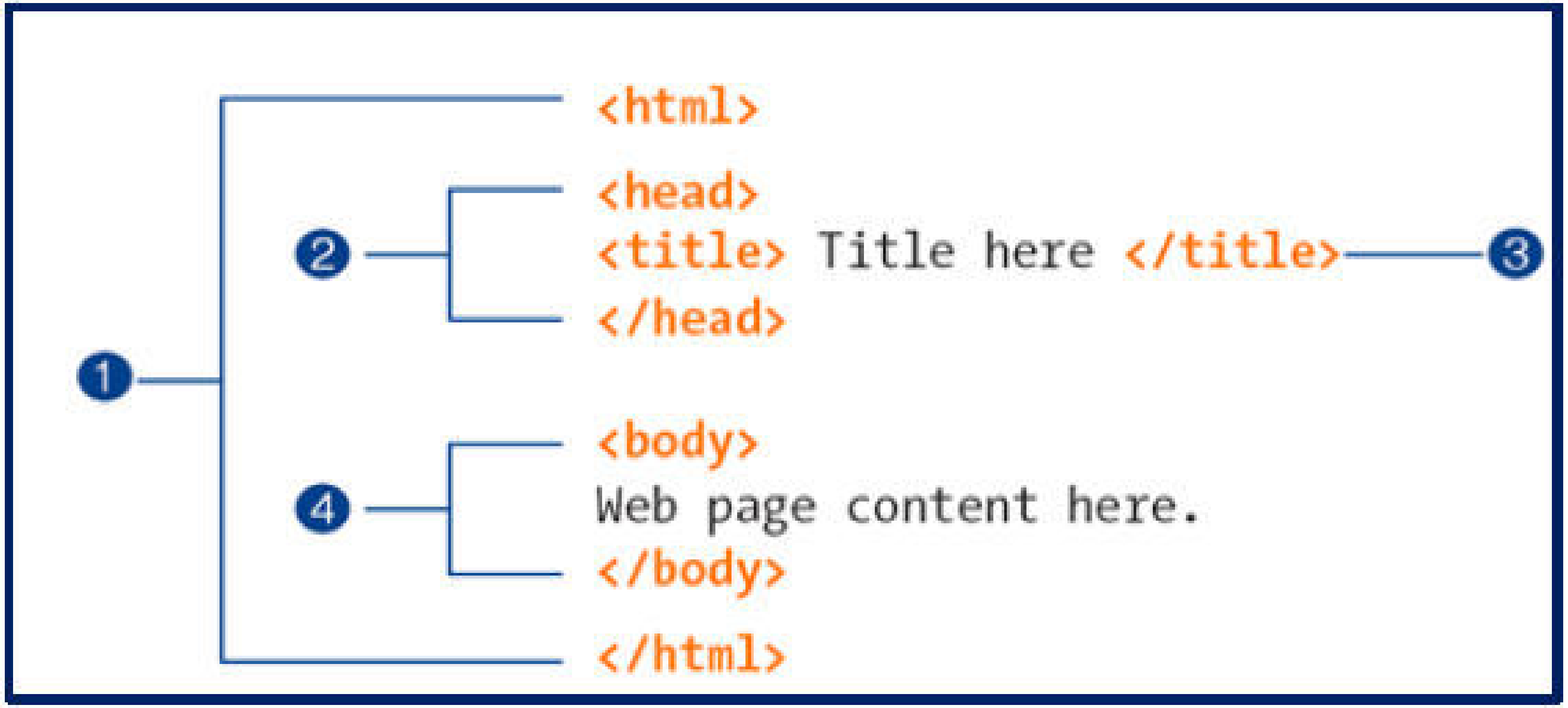
HTML

- What is Markup language?
  - A markup language is a programming language that is used to make text more interactive and dynamic. It can turn a text into images, tables, links etc.
- What is Hyper Text Markup Language (HTML)?
  - It is the standard markup language for creating Web pages.
  - It is a language for describing web pages.
  - HTML documents contain HTML tags and plain text
  - HTML documents are also called web pages
  - It consists of a series of elements
  - HTML elements tell the browser how to display the content
  - HTML elements are represented by tags
  - HTML tags label pieces of content such as heading, paragraph, table, and so on
  - Browsers do not display the HTML tags, but use them to render the content of the page

What is HTML Tag?

- HTML tags are element names surrounded by angle brackets:
- <tagname> content goes here...</tagname>
- HTML tags normally come in pairs like <p> and </p>
- The first tag in a pair is the start tag, the second tag is the end tag
- The end tag is written like the start tag, but with a forward slash inserted before the tag name

The structure of an HTML element



```
<html>
<head>
<title> Title here </title>
</head>

<body>
Web page content here.
</body>
</html>
```

HTML Element

- The <!DOCTYPE html> declaration defines this document to be HTML version 5
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the document
- The <title> element specifies a title for the document
- The <body> element contains the visible page content
- The <h1> element defines a large heading
- The <p> element defines a paragraph

HTML tags! typography!

- <p>- paragraph!
- <h1>…<h6>- headings!
- <a>- link / anchor!
- <ul>,<ol>,<li>- lists!
- <table>,<tr>,<td>- tables!
- <img/>,<video>- images, videos

HTML tags! text formatting

- <b>,<strong>- bold!
- <i>,<em>- italics!
- <sup>,<sub>- super/subscript!

HTML tags! organization!

- <div>- container!
- <section>- section!
- <article>- article!
- <br/>- line break!
- <hr/>- horizontal rule!

HTML <form> Tag

- An HTML form with two input fields and one submit button:

-
  ```
  <form action="/action_page.php" method="get">
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname"><br><br>
    <label for="lname">Last name:</label>
    <input type="text" id="lname" name="lname"><br><br>
    <input type="submit" value="Submit">
  </form>
  ```

CSS overview

CSS **basics!** inline styling!

```
<p style="font--
-size:14px;">…</p>
```

CSS **basics!** block styling!

```
<style>
p{font---size:14px;}
</style>
```

CSS **basics!** separate file!

```
p{
font---size:14px;
}
```

CSS basics!

rule syntax!

```
selector {
property: value;
property: value;
…
}
```

CSS basics!

properties!

Typography!

font---size
font---weight
font---family
line---height
text---align

Colors!

color
background---color
background---image
line---height
text---align

Positioning!

position
width,height
margin
padding
border

CSS basics!

selectors!

HTML element!	CSS rule!
`<p>`	`p { font---size:14px; }`
`<p class="bp">`	`.bp{color:gray; }`
`<p id="headline">`	`#headline { font---size:20px; }`

CSS basics!

cascading selectors!

HTML element!

```
<div class="intro">
     <h1>…</h1>
</div>
```

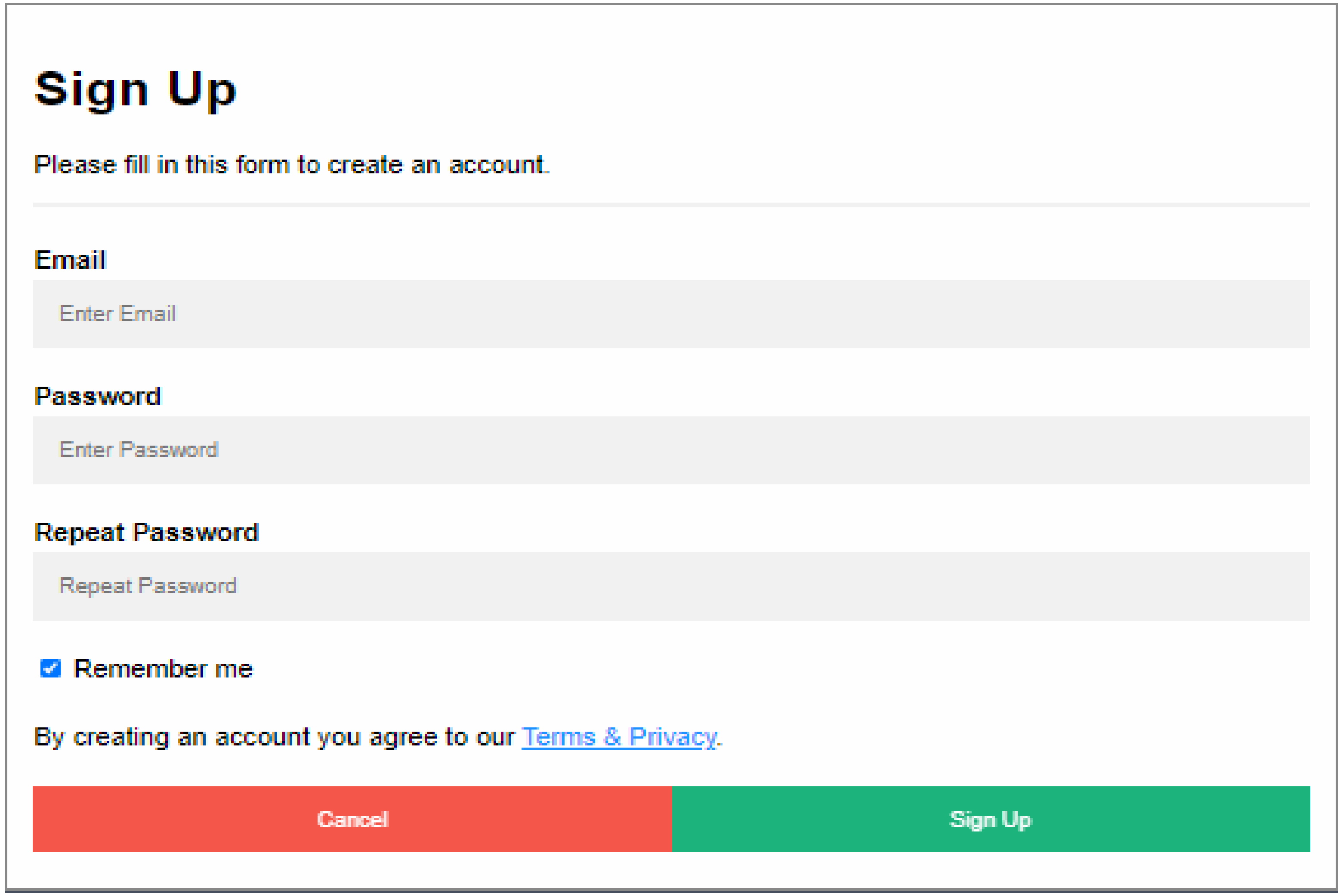
CSS rule!

```
.intro h1 {
font---size:18px;
}
```

style `h1` inside every element with `class="intro"`

Assignment: **Registration Form**

- Create the following Page using HTML and CSS



Sign Up

Please fill in this form to create an account.

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

☑ Remember me

By creating an account you agree to our Terms & Privacy.

Cancel Sign Up

Thank you

Web Programming

CHAPTER 2: Server-Side Programming (PHP)

OUTLINES

1. Server Side Technologies

- Server-side technologies are quite numerous and diverse.
- Popular server side web application technologies include:
  - Microsoft ASP/.NET
  - Java server technologies such as J2EE, JSP, and servlets
  - Perl
  - PHP
  - ColdFusion
- The "core" server side application development platforms can retrieve, modify and query the contents of databases through their own access mechanisms:
  - PHP enables direct access to many existing DB platforms, most notably MySQL, but also, Oracle, SQL Server and others

1.1 Dynamic Web pages: needed when

- consistent look and feel on each page of a large site is required
- data is derived from a database
- content depends on real time
- content depend on user choice
- business transactions e.g. e-commerce...

2. what is PHP?

- PHP is a stand for " **Hypertext Preprocessor**".
- PHP is a server **side scripting language that is embedded in HTML.** It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP supports a large number of major protocols such as POP3,FTP ,HTTP and so on
- PHP is a widely-used, open source scripting language

what is PHP? (cont.)

- PHP scripts are executed on the server
- It is an interpreted, server-side scripting language.
- Originally designed for web development.
- PHP Syntax is similar to C, Perl/Practical extraction and Report Language / , java and so on
- PHP is case sensitive
- PHP scripts reside between reserved PHP tags
- The PHP Source-code not visible by client
  - ‘View Source’ in browsers does not display the PHP code
- PHP:Various built-in functions allow for fast development
- PHP is Compatible with many popular databases

2.1 History of PHP

- Started as a Perl hack in 1994 by Rasmus Lerdorf (to handle his resume), developed to PHP/FI 2.0
- 1994: Created by Rasmis Lesdorf, software engineer (part of Apache Team)
- 1995: Called Personal Home Page Tool, then released as version 2 with name PHP/FI (Form Interpreter, to analyze SQL queries)
- By 1997 up to PHP 3.0 with a new parser engine by Zeev Suraski and Andi Gutmans
- Half 1997: used by 50,000 web sites
- October 1998: used by 100,000 websites
- End 1999: used by 1,000,000 websites

History of PHP (cont.)

- PHP 7 is now released bringing new changes to the platform. The new changes ensures old flaws are somewhat fixed and new features can help it bring the most out of the programming language. The naming convention has also been an issue among the community.
- PHP 7 brings many new features include performance improvement.

Alternatives to PHP OR Similar languages

- **Similar languages are :**
  - Perl/Practical extraction and Report Language /
  - ASP.NET,
  - Ruby,
  - JSP (JavaServer Pages) ,
  - Active Server Pages (ASP) and etc...

2.2 Why Choose PHP? / Why PHP?

- Open source, free software ,PHP is free. Download it from the official PHP resource: www.php.net
- PHP supports a **wide range of databases**
- PHP **runs on various platforms** (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is **compatible with almost all servers** used today (Apache, IIS, etc.)
- **Powerful, robust/ strong and scalable**
  - Example Yahoo, Facebook, and Wikipedia
- **Designed for web page**
- Can be object oriented
- PHP is **easy to learn and runs** efficiently on the server side

2.3 Characteristics of PHP

- **Five important characteristics make PHP's practical nature possible −**
  - Simplicity
  - Efficiency
  - Security
  - Flexibility
  - Familiarity

2.4 PHP Features

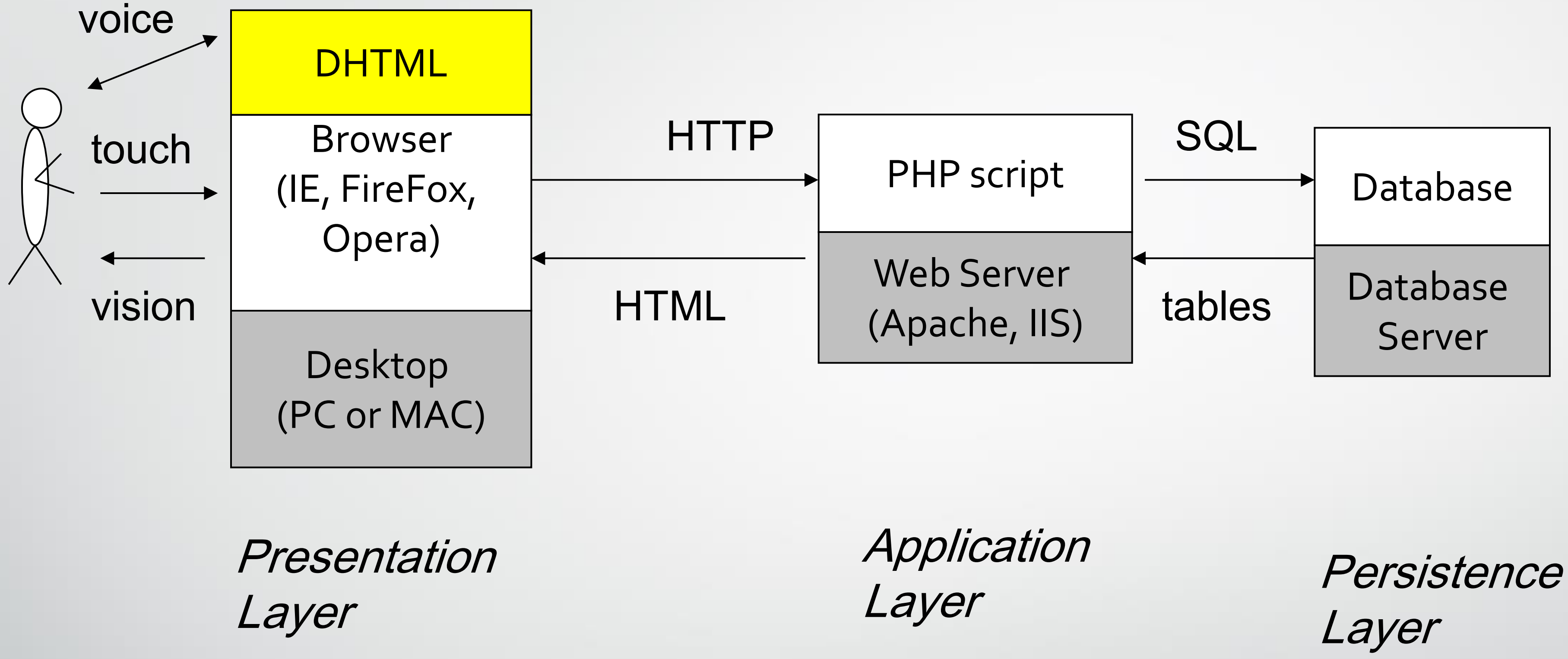
- **Performance**
  - Relatively fast for an interpreted language.
- **Database Connectivity**
  - Support for MySQL, Oracle, dbm, DB2, PostgreSQL
  - Can connect to any database which provides an ODBC driver (Open Database Connectivity Standard) – e.g. MS Access
- **Existing Libraries**
  - PHP was originally designed for web use – lots of functions for common web-development tasks (e.g. Sending email, XML parsing, etc.)
- **Portability**
  - Supported on Windows, Mac OS X, other Unix-like systems (e.g. Linux).
- **Object-Oriented Programming**
  - Similar syntax and features as C++ and Java – inheritance, attribute visibility (private, protected), abstract classes/methods, constructors and destructors, etc.
- **Popularity**
  - As a result, lots of documentation, books, and web tutorials.

2.5 PHP advantages

Advantages of Using PHP to enhance Web pages:

- Easy to use
- Simpler than Perl
- Open source
- Multiple platform.

3-Tier Architecture & PHP Scripting



3-tier architecture

- A Presentation layer **using Browser technology**
  - Decoding URLs : protocol/host/file
  - Host name converted to IP address(164.11.8.19)
  - Issue request to remote server using appropriate protocol (usually HTTP)
  - accept the returned HTML (or JPEG, ..) file
- An Application layer using a **web application server platform + application programs**
  - Server (Apache, IIS)
    - Identifying appropriate action to take – fetch a file, pass request to an interpreter
  - Server Script (e.g. in PHP)
    - Interacting with the server (accessing input and generating output)
    - interpreting the requests according to business rules and past transactions from this client
- A Persistence **layer using a relational database or other data store technology**
  - interaction with the database using standard languages e.g. SQL queries using database-specific protocol over TCP/IP
  - define and modify the data structures (e.g. tables) themselves

    (the Database Schema)
  - insert, update and delete data
  - maintain data persistently, with backup and recovery

2.7 How to run a PHP program ?

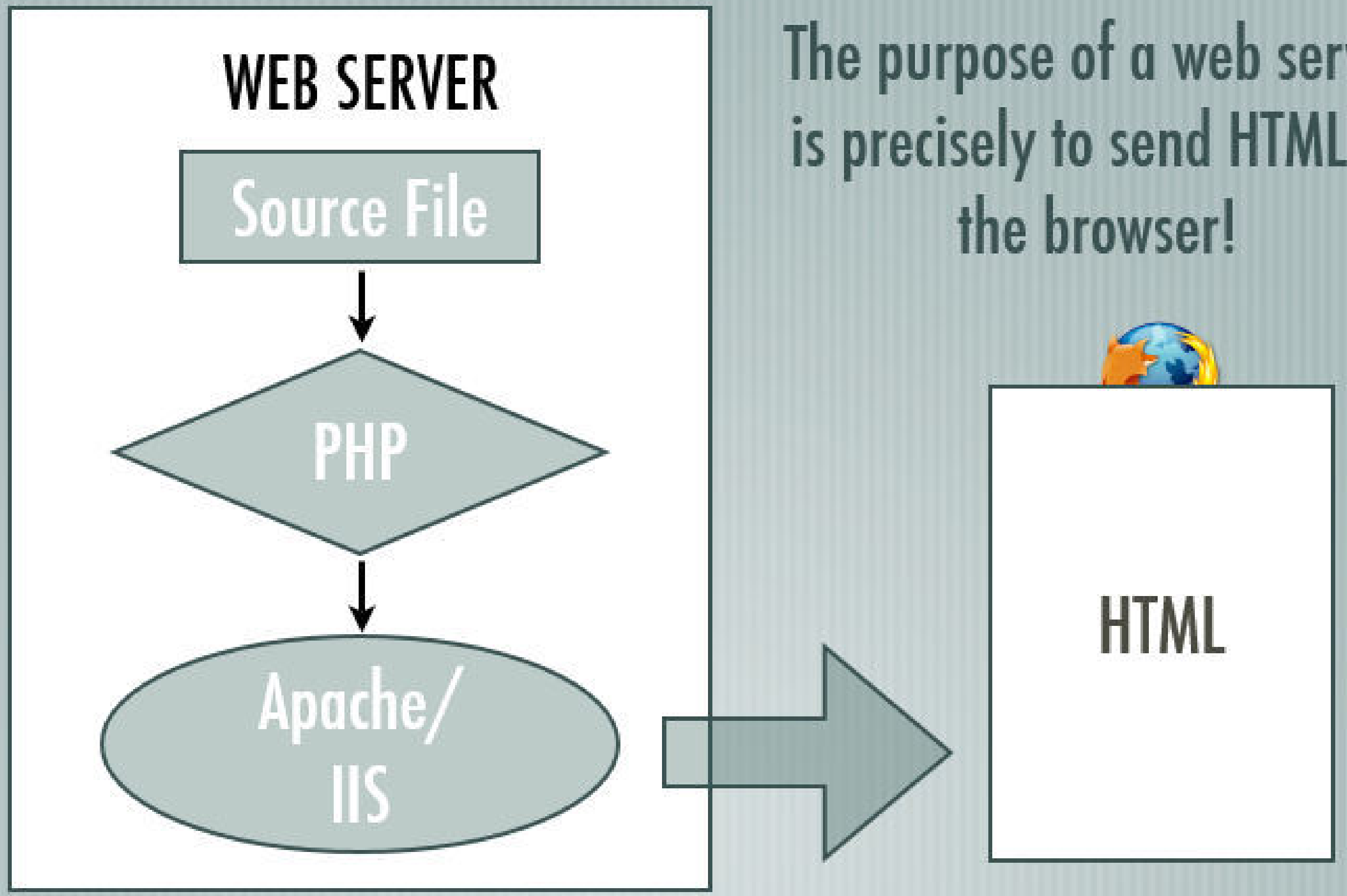
In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- **Web Server** - PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- **Database** - PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
- **PHP Parser** - In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.
- There are various web servers for running PHP programs like WAMP & XAMPP.
- WAMP server is supported in windows and XAMP is supported in both Windows and Linux..

What is Xampp and why it is used?

- It is a free and open source cross-platform web server solution stack package developed by Apache Friends that consists of the Apache HTTP Server, MariaDB & MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.
- xampp is the most popular PHP development environment for Windows, OS X, and Linux platforms
- Xampp stands for Cross platform(x), Apache(a), Maria db(m), PHP(p), Pearl(p) which is a software distribution server which makes developer's work eaiser for testing and deploying by creating a local web server.
- the steps involved in using is Xampp Server:
  - **Install the Server Software**
  - **Set up the Server**
  - **Save PHP Scripts**
  - **Run PHP Scripts**
  - **Troubleshoot**

How PHP Works



The purpose of a web server is precisely to send HTML to the browser!

2. PHP scripting Syntax Overview

2.1 PHP Tags style

Escaping to PHP

- The PHP analyzing engine needs a way to differentiate PHP code from other elements in the page.
- The mechanism for doing so is known as 'escaping to PHP'. There are four ways to do this: −
  1. **Canonical PHP tags**
  2. **Short-open (SGML-style) tags**
  3. **ASP-style tags**
  4. **HTML script tags**

Canonical PHP tags

- The most universally effective PHP tag style is –
  - <?php...?>.
- If you use this style, you can be positive that your tags will always be correctly interpreted.

Short-open (SGML-style) tags

- Short or short-open tags look like this –
  - <?...?> or <? // Some code?>
- Short tags are, as one might expect, the shortest option You must do one of two things to enable PHP to recognize the tags

ASP-style tags

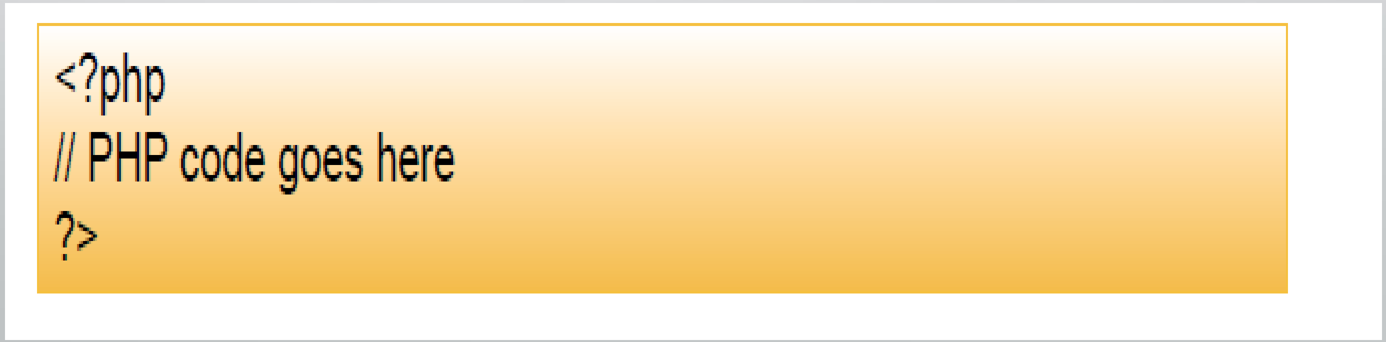
- Active Server Pages /ASP-style tags look like this –
- <%...%>
- To use ASP-style tags, you will need to set the configuration option in your php.ini file.

HTML script tags

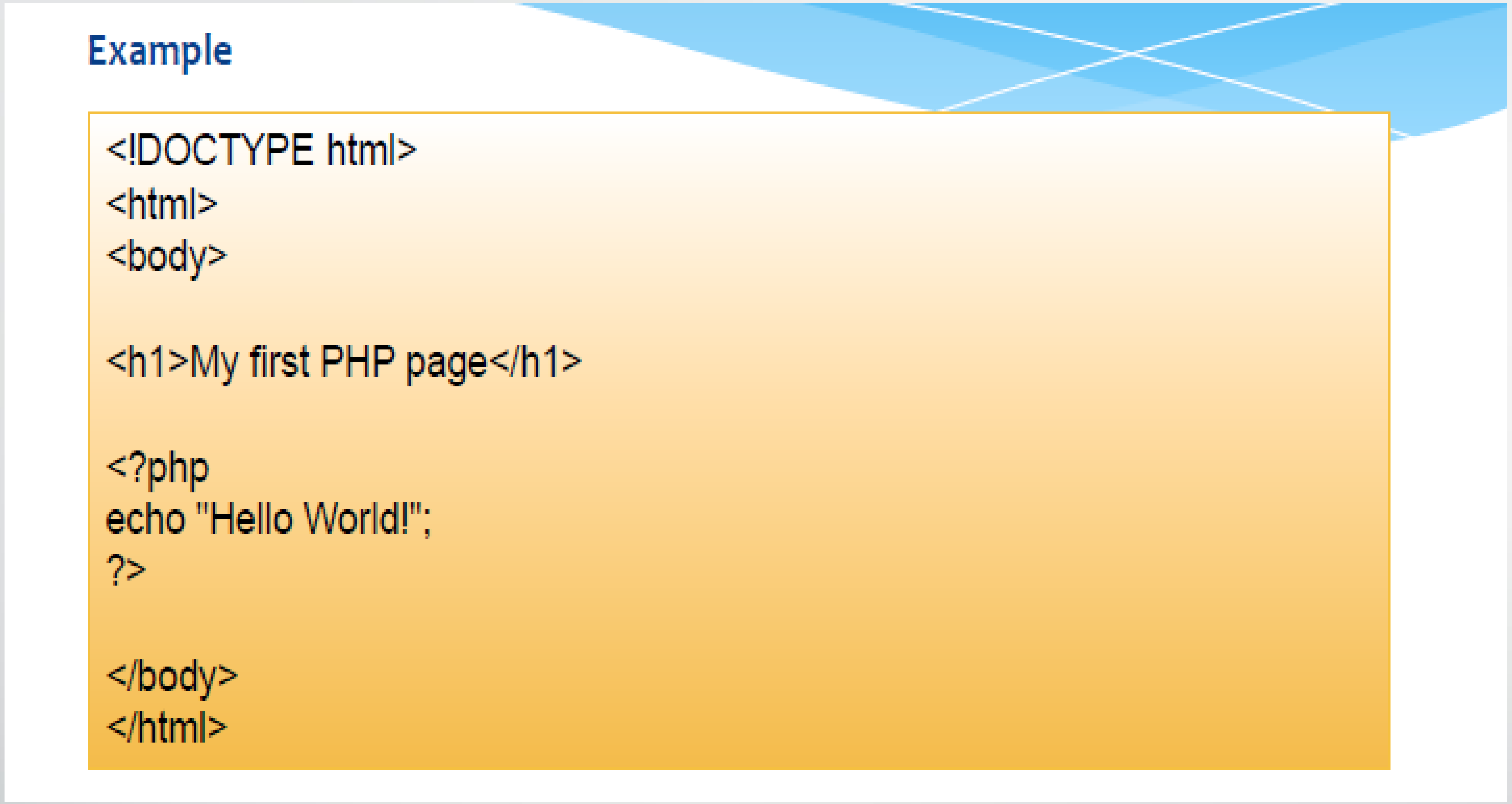
- HTML script tags look like this –
- <script language="PHP">...</script>

2.2 Basic Syntax of Canonical PHP tags

- The PHP code in the preceding example began with **<?php** and ended with **?>**.
- This is similar to all HTML tags because they all begin with a less than (<) symbol and end with a greater than (>) symbol.
- These symbols **(<?php and ?>)** are called PHP tags. They tell the web server where the PHP code starts and finishes.
- Any text between the tags is interpreted as PHP. Any text outside these tags is treated as normal HTML.
- The PHP tags allow you to escape from HTML.



```
<?php
// PHP code goes here
?>
```


Example

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

Note: PHP statements are terminated by semicolon (;).

2.3 Comments in PHP :

- A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result.

- **Single Line Comments:** There are two ways to use single line comments in PHP.
  - **// (C++ style single line comment)**
  - **# (Unix Shell style single line comment)**

example

```
<?php
// this is C++ style single line comment
# this is Unix Shell style single line comment
echo "Welcome to PHP single line comments";
?>
```

- **PHP Multi Line Comments:** In PHP, we can comments multiple lines also. They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. To do so, we need to enclose all lines within **/*   */**

example

```
<?
/* This is a comment with multiline
Author : Mohammad Mohtashim
Purpose: Multiline Comments Demo
Subject: PHP
*/
print "An example with multi line comments";
?>
```

2.4 PHP is whitespace insensitive:

- Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).
- PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row.one whitespace character is the same as many such characters.

For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable $four is equivalent:

```
$four = 2 + 2; // single spaces.

$four <tab>=<tab2<tab>+<tab>2 ; // spaces and tabs

$four =

2+

2; // multiple lines
```

2.5 PHP is case sensitive:

PHP is a case sensitive language. Try out following example:

```
<html>

<body>

<?

$capital = 67;

print("Variable capital is $capital<br>");

print("Variable CaPiTaL is $CaPiTaL<br>");

?>

</body>

</html>
```

This will produce following result:

```
Variable capital is 67
Variable CaPiTaL is
```

2.6 Statement in php

Statements are expressions terminated by semicolons:

- A ***statement*** in PHP is any expression that is followed by a semicolon (;).
- Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program.
- Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called $greeting:

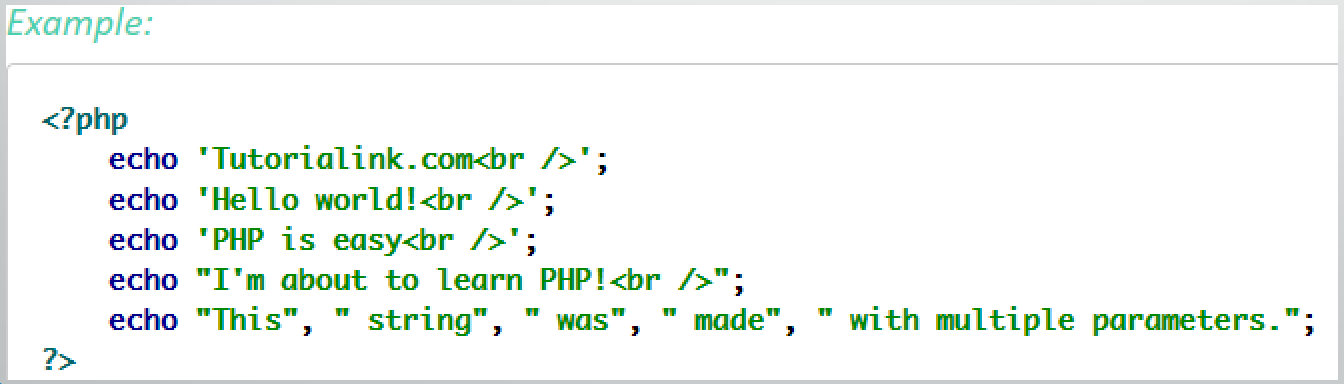
```
$greeting = "Welcome to PHP!";
```

2.7 PHP Output Statements:- echo and print

- The two most basic constructs for displaying output in PHP are **echo** and **print.**

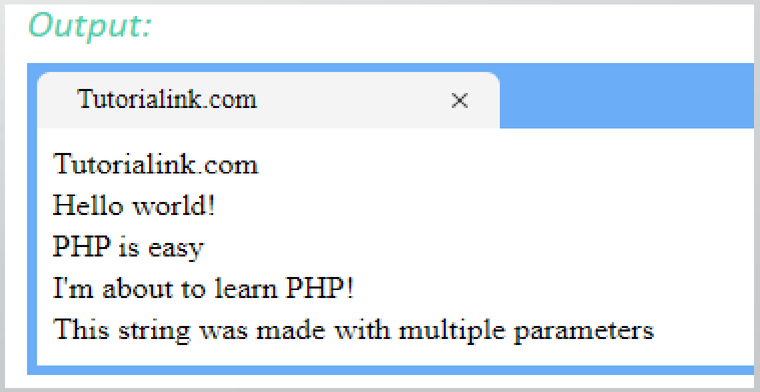
PHP echo Statement

- **echo is a language construct, and can be used with or without parentheses: echo or echo().**
- **Echo has no return value**
- **echo can take multiple parameters** (although such usage is rare) , We can pass multiple strings separated by comma (,) in echo.
- The parameterized version of echo does not accept multiple arguments
- echo is marginally faster than print.



Example:

```
<?php
    echo 'Tutorialink.com<br />';
    echo 'Hello world!<br />';
    echo 'PHP is easy<br />';
    echo "I'm about to learn PHP!<br />";
    echo "This", " string", " was", " made", " with multiple parameters.";
?>
```



Output:

```
Tutorialink.com
Hello world!
PHP is easy
I'm about to learn PHP!
This string was made with multiple parameters
```

Echo : Display HTML Code

- The following example will show you how to display HTML code using the echo statement:
- Example

```
<?php
 // Displaying HTML code
echo "<h4>This is a simple heading.</h4>";
echo "<h4 style='color: red;'>This is heading with style.</h4>";
?>
```

- **Output**

```
This is a simple heading.
This is heading with style.
```

Echo: Display Variables

- The following example will show you how to display variable using the echo statement:

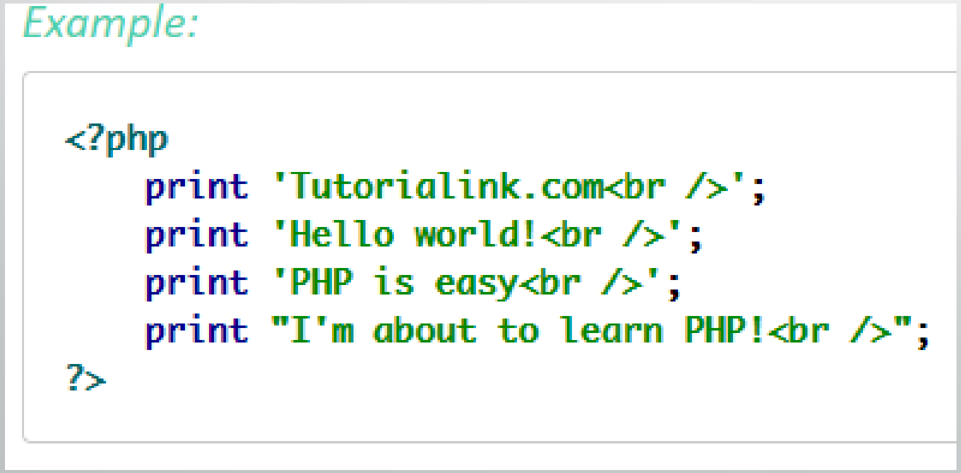
```
<?php
// Defining variables
$txt = "Hello World!";
$num = 123456789;
$colors = array("Red", "Green", "Blue");
// Displaying variables
echo $txt;
echo "<br>";
echo $num;
echo "<br>";
echo $colors[0];
?>
```

Output

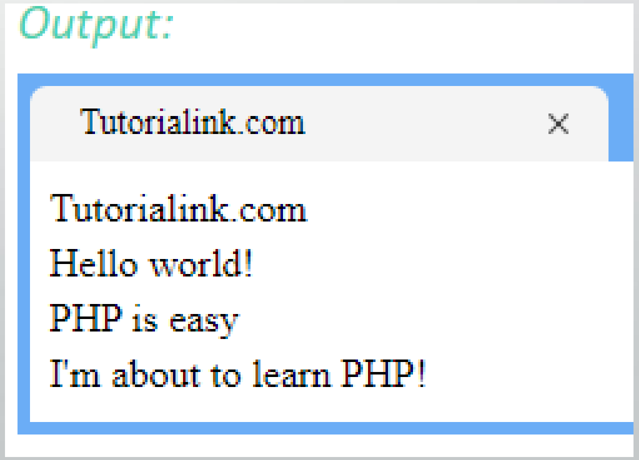
```
Hello World!
123456789
Red
```

PHP print Statement

- print is also a language construct, and can be used with or without parentheses: print or print().
- print has a return value of 1 so it can be used in expressions
- print can take one argument.
- The following example shows how to display different strings with the print command (also notice that the strings can contain HTML markup):


Example:

```
<?php
    print 'Tutorialink.com<br />';
    print 'Hello world!<br />';
    print 'PHP is easy<br />';
    print "I'm about to learn PHP!<br />";
?>
```


Output:

```
Tutorialink.com
Hello world!
PHP is easy
I'm about to learn PHP!
```

Print: Display Variables

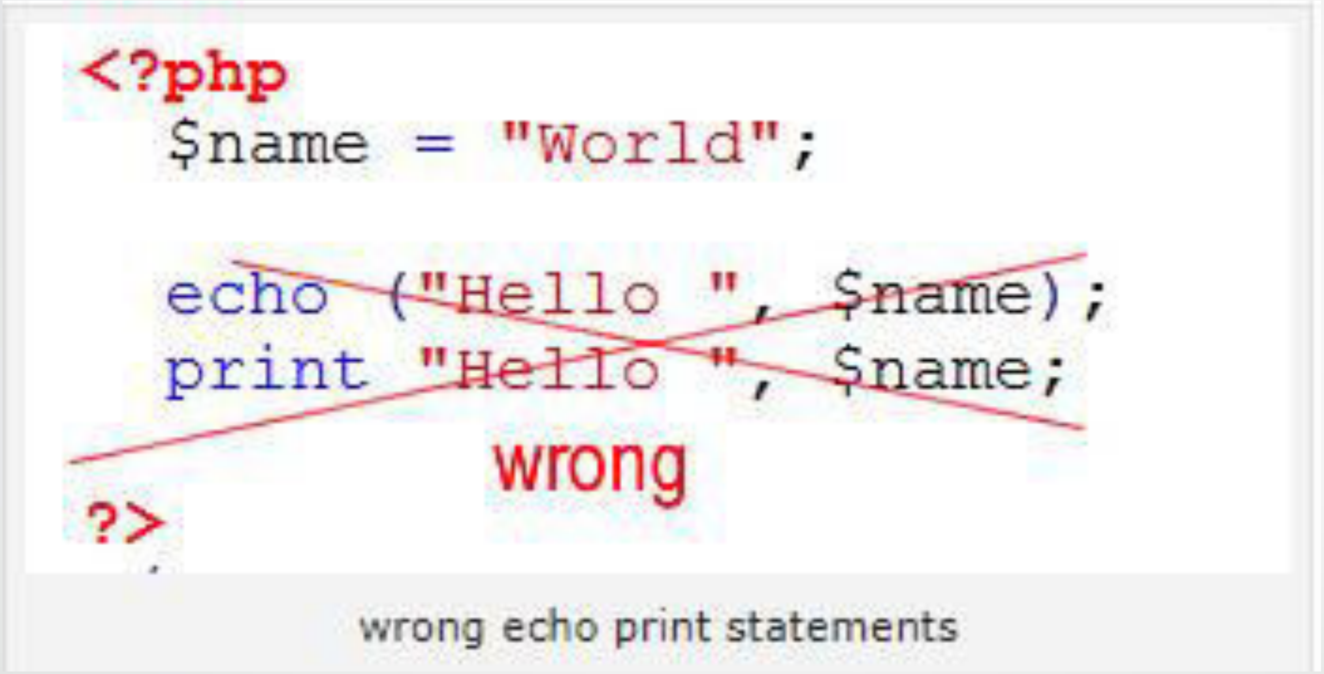
- The following example will show you how to display variable using the print statement:
- Example

```
<?php
// Defining variables
$txt = "Hello World!";
$num = 123456789;
$colors = array("Red", "Green", "Blue");
// Displaying variables
print $txt;
print "<br>";
print $num;
print "<br>";
print $colors[0];
?>
```

Output

```
Hello World!
123456789
Red
```

The following two statements does not work…



```
<?php
  $name = "World";

  ~~echo ("Hello ", $name);~~
  ~~print "Hello ", $name;~~
```

```
?>
```

wrong echo print statements

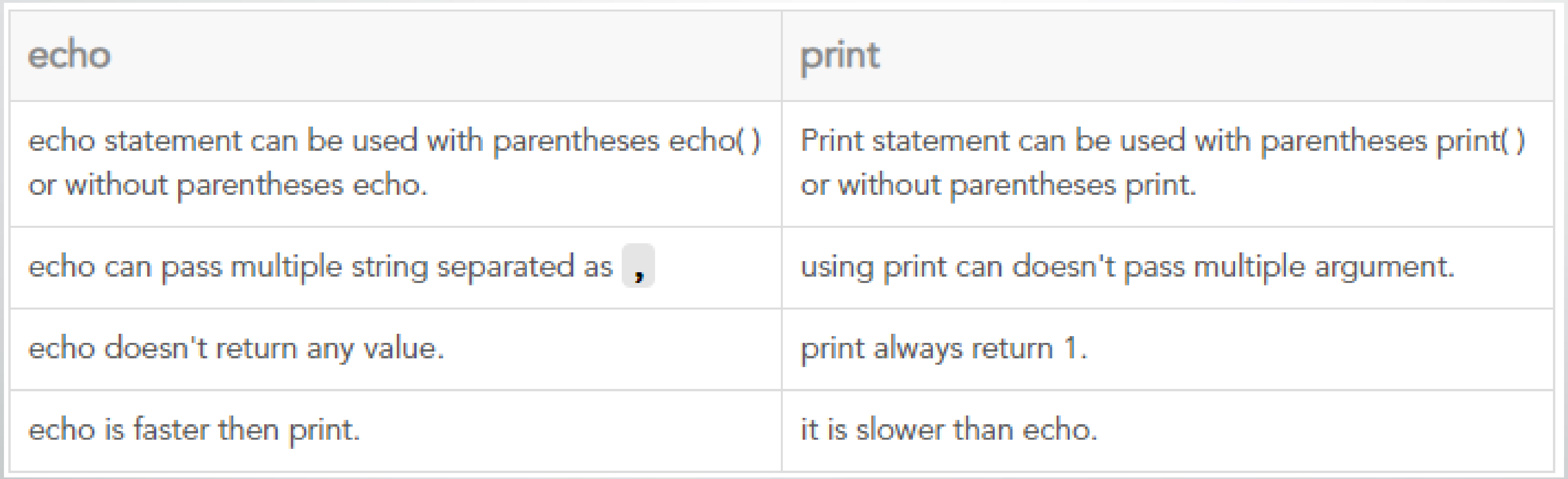
- echo (“Hello “, $name);

does not work because comma separated arguments can not be used when it is using parenthesis.

- Print is not able accept multiple arguments regardless of whether it is using parenthesis or not. Print takes only one argument.

Difference between echo and print Statements

- There are some differences between echo and print:



echo	print
echo statement can be used with parentheses echo( ) or without parentheses echo.	Print statement can be used with parentheses print( ) or without parentheses print.
echo can pass multiple string separated as `,`	using print can doesn't pass multiple argument.
echo doesn't return any value.	print always return 1.
echo is faster then print.	it is slower than echo.

Note:
echo is marginally faster compared to print as echo does not return any value.

2.8 Variable in PHP

- A variable is a special container that can be defined to hold a value such as number, string, object, array, or a Boolean.
- The main way to store information in the middle of a PHP program is by using a variable.
- *Here are the most important things to know about variables in PHP.*
  - All variables in PHP are denoted with a **leading dollar sign ($).**
  - The value of a variable is the **value of its most recent assignment.**
  - Variables are assigned with the **= operator**, with the variable on the left-hand side and the expression to be evaluated on the right.

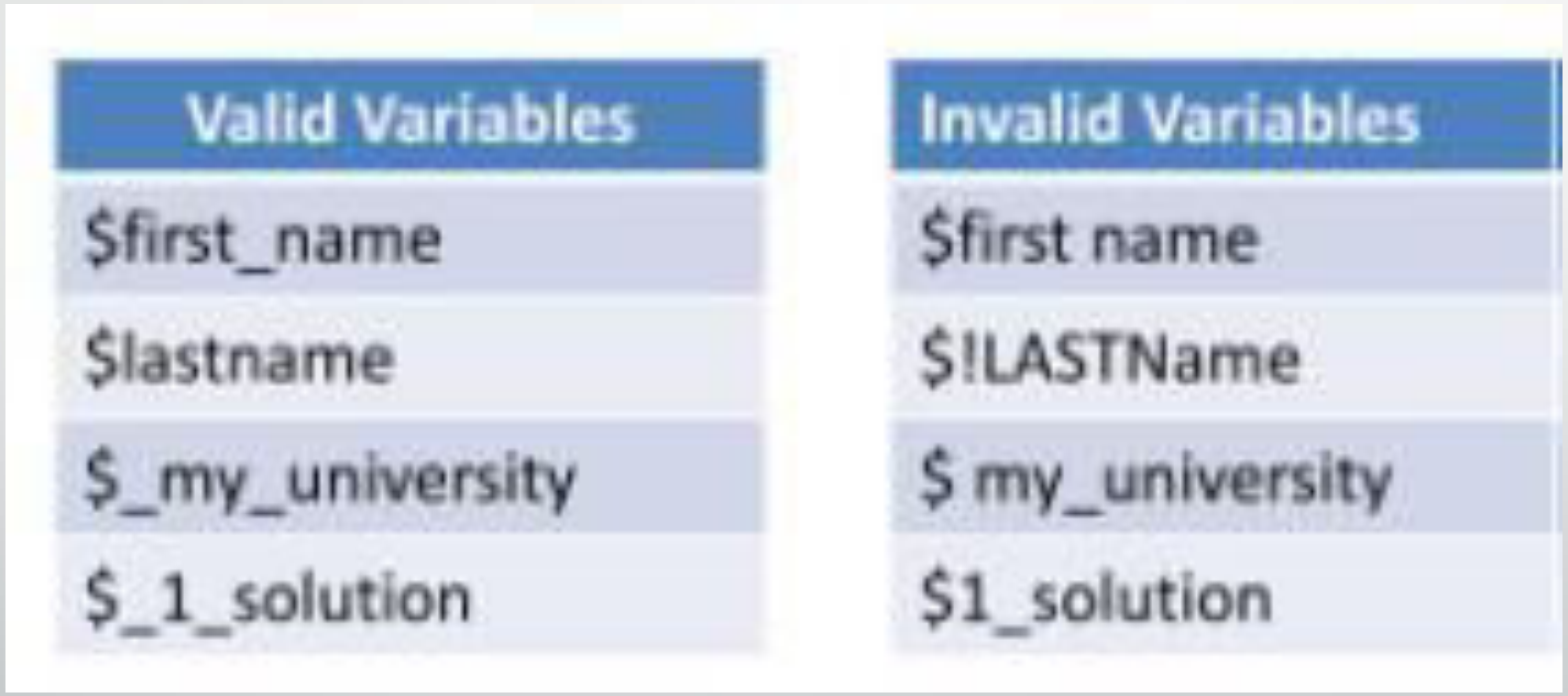
Variable in PHP (cont)

- Variables can, **but do not need, to be declared before assignment**.
- Variables in **PHP do not have intrinsic types** - a variable does not **know in advance whether it will be used to store a number or a string of characters.**
- Variables used before they are assigned **have default values.**
- PHP does a good job of **automatically converting types from one to another when necessary.**
- PHP variables are **Perl-like.**

Variable Naming:

- Rules for naming a variable are:
  - Variable names must begin with a letter or underscore character.
  - A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc
  - There is no size limit for variables.

Sample Valid and invalid PHP variables



Valid Variables
$first_name
$lastname
$_my_university
$_1_solution

Invalid Variables
$first name
$!LASTName
$ my_university
$1_solution

2.9 Php data types

- PHP has a total of eight data types which we use to construct our variables:
  - **Integers:** are whole numbers, without a decimal point, like 4195.
  - **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
  - **Booleans:** have only two possible values either true or false.
  - **NULL:** is a special type that only has one value: NULL.
  - **Strings:** are sequences of characters, like 'PHP supports string operations.'
  - **Arrays:** are named and indexed collections of other values.
  - **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
  - **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

Integers:

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so:

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

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading ox.

For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

Doubles:

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code:

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

It produces the following browser output:

```
2.28888 + 2.21112 = 4.5
```

Predefined variables

- PHP provides a large number of predefined variables to all scripts. The variables represent everything from external variables to built-in environment variables, last error messages to last retrieved headers.
  - Superglobals — Superglobals are built-in variables that are always available in all scopes
  - $GLOBALS — References all variables available in global scope
  - $_SERVER — Server and execution environment information
  - $_GET — HTTP GET variables
  - $_POST — HTTP POST variables
  - $_FILES — HTTP File Upload variables
  - $_REQUEST — HTTP Request variables, and can replace $_POST, $_GET and $_COOKIE variables
  - $_SESSION — Session variables
  - $_COOKIE — HTTP Cookies
  - $php_errormsg — The previous error message
  - $HTTP_RAW_POST_DATA — Raw POST data
  - $http_response_header — HTTP response headers
  - $argc — The number of arguments passed to script
  - $argv — Array of arguments passed to script

Removing Variables

- We can uncreated the variable by using this statement: **unset(VariableName)**;
- After this statement, the variable $age no longer exists. If we try to echo it, you get an ―undefined variable‖ notice.
- It is possible to unset more than one variable at once, as follows:
  - unset($age, $name, $address);

2.10 PHP Constants

- A constant is a name or an identifier for a simple value. **A constant value cannot change during the execution of the script**. By default a constant is **case-sensitive**.
- By convention, **constant identifiers are always uppercase**. A constant name starts with a **letter or underscore**, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.
- To define a constant you have to use **define() function** and to retrieve the value of a constant, you have to simply specifying its name.

Cont---

- Unlike with variables, **you do not need to have a constant with a $.** You can also use the function **constant()** to read a constant's value if you wish to obtain the constant's name dynamically.
- **constant() function:** As indicated by the name, this function will return the value of the constant.
- This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. It is stored in a variable or returned by a function.
- **constant() example:**

```
php <?
define("MINSIZE", 50);
echo MINSIZE;
echo constant("MINSIZE"); // same thing as the previous line
?>
```

Differences between constants and variables are:

- There is no need to write a **dollar sign ($)** before a constant, where as in Variable one has to write a dollar sign.
- Constants **cannot be defined by simple assignment**, they may only be defined using the **define()** function.
- Constants may be defined and accessed anywhere **without regard to variable scoping rules.**
- Once the Constants have been set, may not be **redefined or undefined.**

Valid and invalid constant names:

// Valid constant names

- define("ONE", "first thing");
- define("TWO2", "second thing");
- define("THREE_3", "third thing") ;

// Invalid constant names

- define("2TWO", "second thing");
- define("__THREE__", "third value");
-

2.11 PHP Operators

- **What is Operator?**
  - Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.
- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - **Arithmetic operators**
  - **Assignment operators**
  - **Comparison operators**
  - **Increment/Decrement operators**
  - **Logical operators**
  - **String operators**
  - **Array operators**

Arithmetic operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y' the power (Introduced in PHP 5.6)

PHP Assignment Operators

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

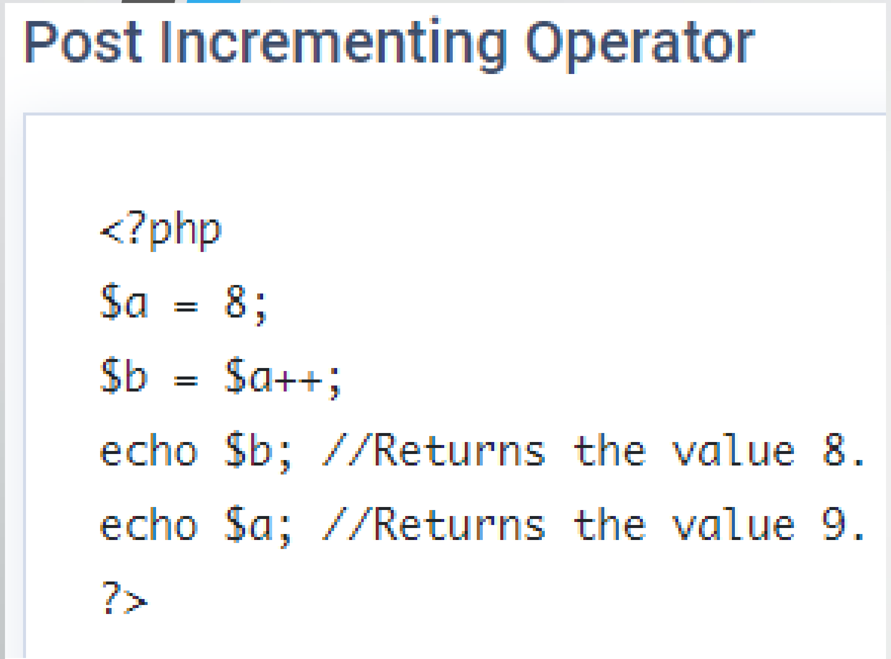
PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators

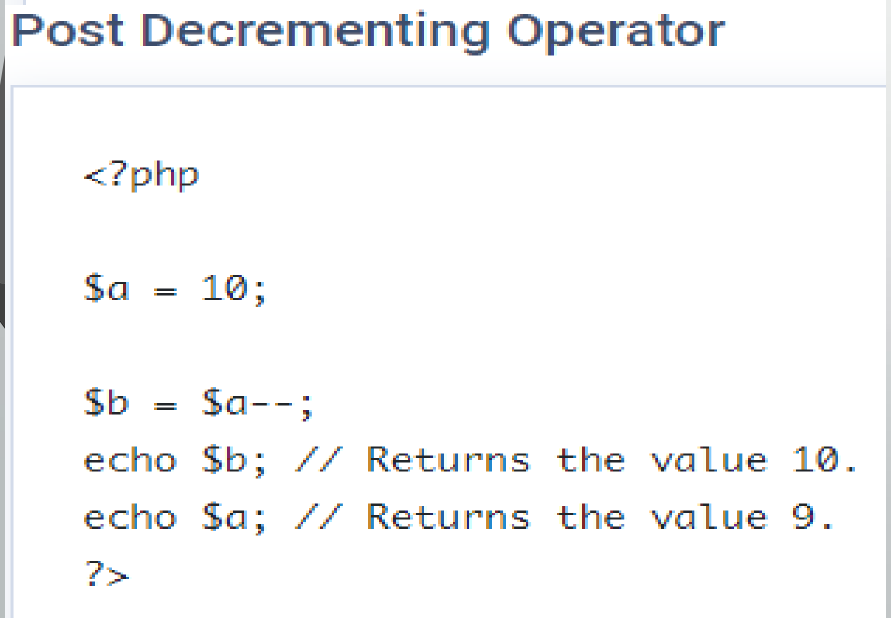
Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Post VS Pre



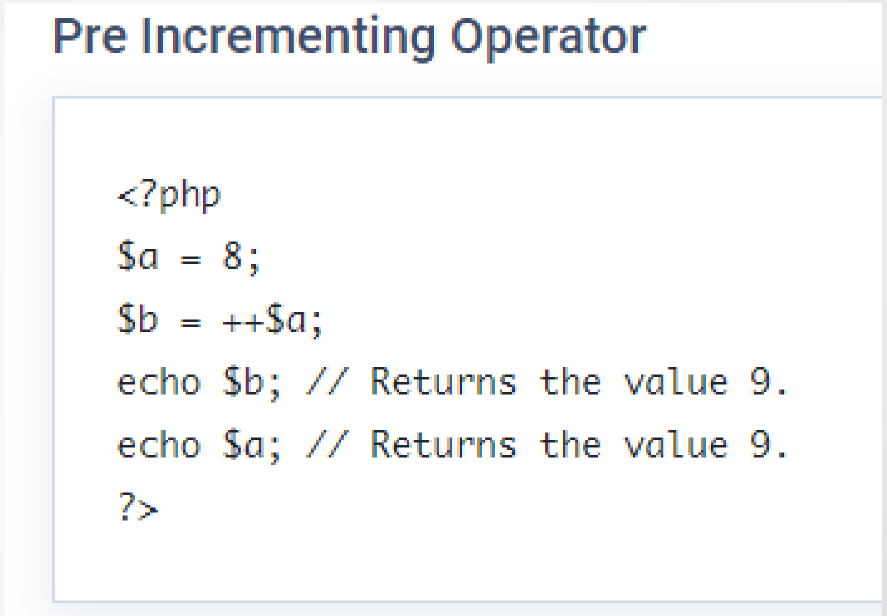
Post Incrementing Operator

```
<?php
$a = 8;
$b = $a++;
echo $b; //Returns the value 8.
echo $a; //Returns the value 9.
?>
```



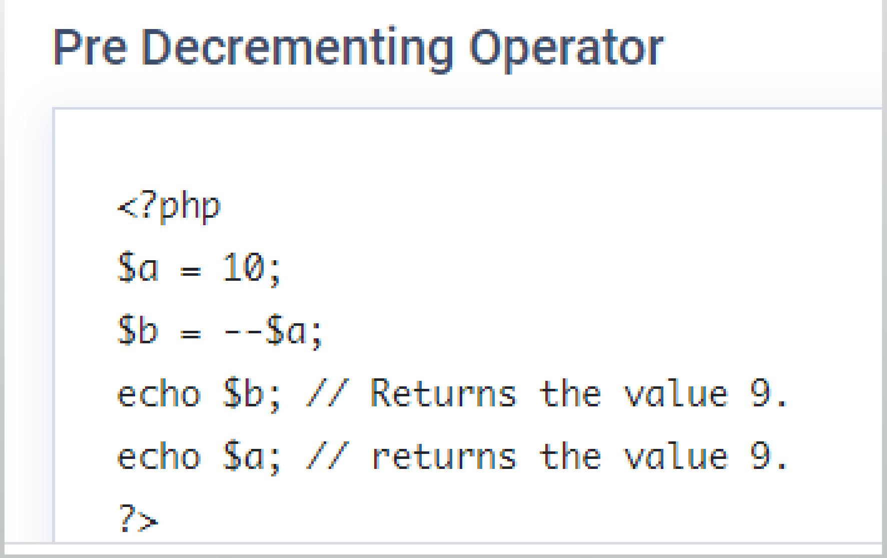
Post Decrementing Operator

```
<?php

$a = 10;

$b = $a--;
echo $b; // Returns the value 10.
echo $a; // Returns the value 9.
?>
```



Pre Incrementing Operator

```
<?php
$a = 8;
$b = ++$a;
echo $b; // Returns the value 9.
echo $a; // Returns the value 9.
?>
```



Pre Decrementing Operator

```
<?php
$a = 10;
$b = --$a;
echo $b; // Returns the value 9.
echo $a; // returns the value 9.
?>
```

PHP Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

Precedence of PHP Operators

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

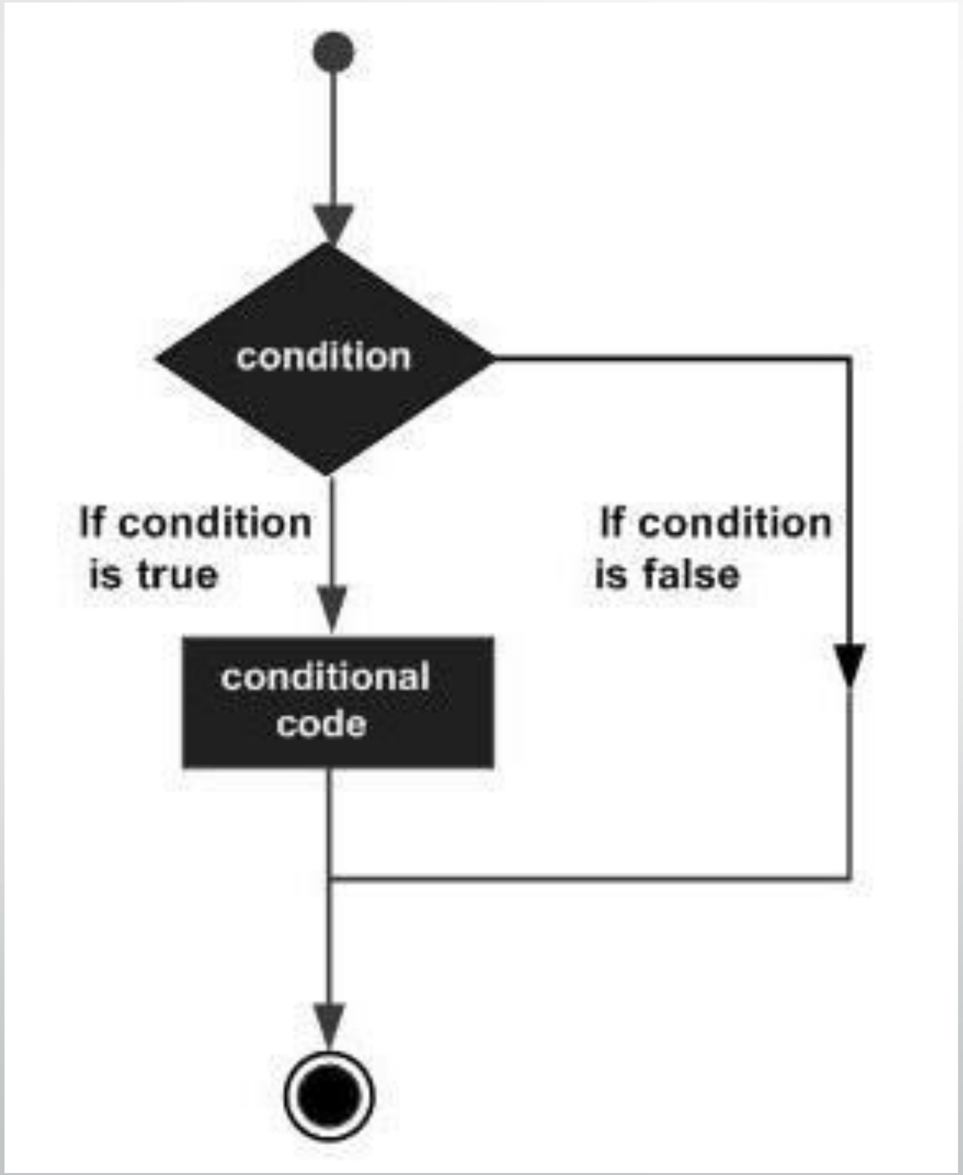
For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

2.12 PHP Decision Making

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. PHP supports following three decision making statements:

- .. **if...else statement** - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** - is used with the if...else statement to execute a set of code if **one** of several condition are true
- **switch statement** - is used 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. Else if else code.


condition
If condition is true
If condition is false
conditional code

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.
- **Syntax:**

  if (*condition*)

  code to be executed if condition is true;

  else

  code to be executed if condition is false;

If…else Cont..

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
else
echo "Have a nice day!";
?>
</body>
</html>
```

```
Out put looks like: if true
Have a nice weekend!
If not true the out becomes:
Have a nice day
It takes the from the system you are
using.
```

If...else Cont..

If more than one line should be executed if a condition is true/false, the lines should be enclosed within $d=datcurly braces: <html>

```
<body>
<?php
e("D");
if ($d=="Fri")
{
echo "Hello!<br />";
echo "Have a nice weekend!";
echo "See you on Monday!";
}
else
echo "Have a nice day!";
?>
</body>
</html>
```

```
Out put looks like: if true
Hello!
Have a nice weekend!
See you on Monday!
If not true the out becomes:
Have a nice day
```

The ElseIf Statement

If you want to execute some code if one of several conditions are true use the elseif statement

Syntax if (*condition*)

> *code to be executed if condition is true;*
>
> elseif (*condition*)
>
> *code to be executed if condition is true;*
>
> else
>
> *code to be executed if condition is false;*

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
elseif ($d=="Sun")
echo "Have a nice Sunday!";
else
echo "Have a nice day!";
?>
</body>
</html>
```

```
Out put looks like: if todays date is
Friday:
Have a nice weekend!
if todays date is Sunday:
Have a nice Sunday!
If todays date out Fri and Sun out put
becomes:
Have a nice day
```

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.

Syntax:

```
switch (expression)
{
case label1:
code to be executed if expression = label1;
break;
case label2:
code to be executed if expression = label2;
break;
default:
code to be executed
if expression is different
from both label1 and label2;
}
```

Example

The *switch* statement works in an unusual way. First it evaluates given expression then seeks a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the labels match then statement will execute any specified default code.

```
<html>
 <body>
 <?php
 $d=date("D");
 switch ($d) {
 case "Mon":
 echo "Today is Monday";
 break;
 case "Tue":
 echo "Today is Tuesday";
 break;
 case "Wed":
 echo "Today is Wednesday";
```

Cont…

```
break;
case "Thu":
echo "Today is Thursday";
break;
case "Fri":
echo "Today is Friday";

break;
case "Sat":
echo "Today is Saturday";
break;
case "Sun":
echo "Today is Sunday";
break;
default:
echo "Wonder which day is this ?";
}
?>
</body> </html>
```

Out put looks like:
Simply executing todays date

2.13 PHP Loop Types

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 trure.
- **foreach -** loops through a block of code for each element in an array.

- We will discuss about **continue** and **break** keywords used to control the loops execution.

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.

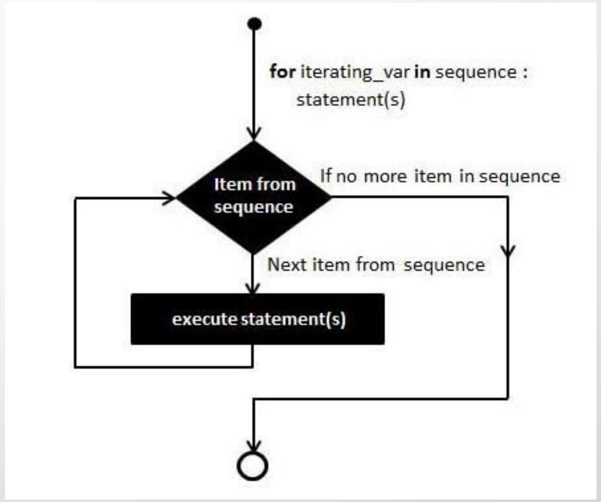
syntax

for (*initialization*; *condition*; *increment*)

{

code to be executed;

}



The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

- The following example makes five iterations and changes the assigned value of two variables on each pass of the loop:

```
<html>
<body>
<?php
$a = 0;
$b = 0;
for( $i=0; $i<5; $i++ )
{
$a += 10;
$b += 5;
}
echo ("At the end of the loop a=$a and b=$b" );
?>
</body>
</html>
```

This will produce following result:

```
At the end of the loop a=50 and
b=25
```

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.

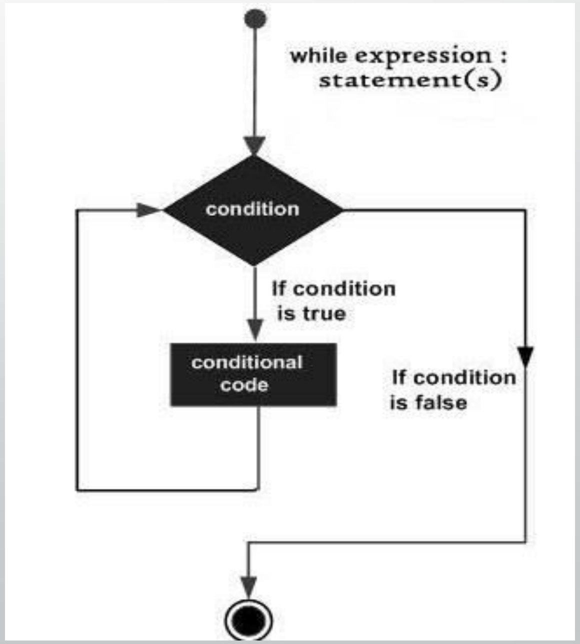
Syntax

while (*condition*)

{

code to be executed;

}



Example

- This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

```
<html>
<body>
<?php
$i = 0;
$num = 50;
while( $i < 10)
{
$num--;
$i++;
}
echo ("Loop stopped at i = $i and num = $num" );
?>
</body>
</html>
```

This will produce following result:

```
Loop stopped at i = 1 and num = 40
```

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.

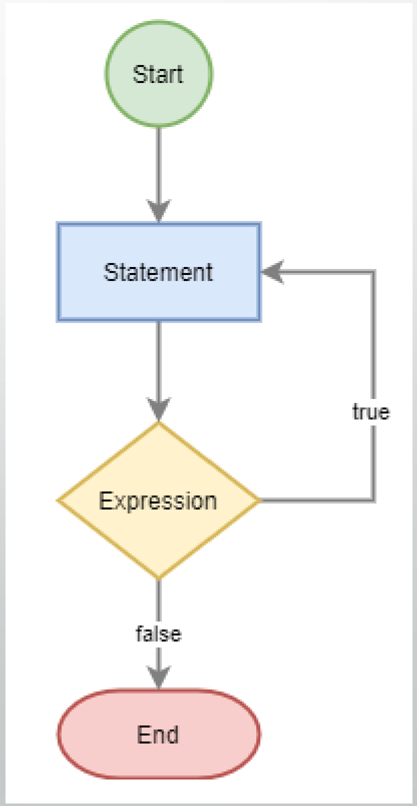
Syntax

do

{

code to be executed;

}while (*condition*);



Example

- The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10:

```
<html>
<body>
<?php
$i = 0;
$num = 0;
do
{
$i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i" );
?>
</body>
</html>
```

This will produce following result:

```
Loop stopped at i = 10
```

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 and the array pointer is moved by one and in the next pass next element will be processed.

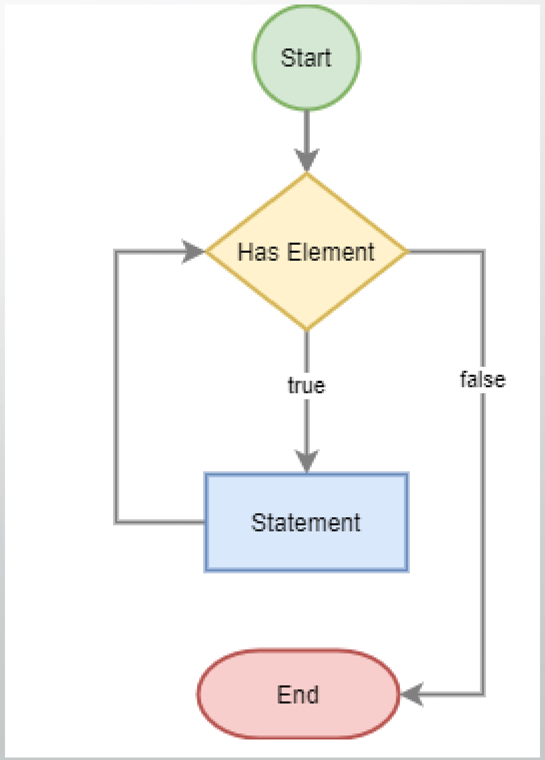
Syntax

foreach (*array* as *value*)

{

code to be executed;

}



Example

- Try out following example to list out the values of an array.

```
<html>
<body>
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
echo "Value is $value <br />";
}
?>
</body>
</html>
```

This will produce following result:
```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

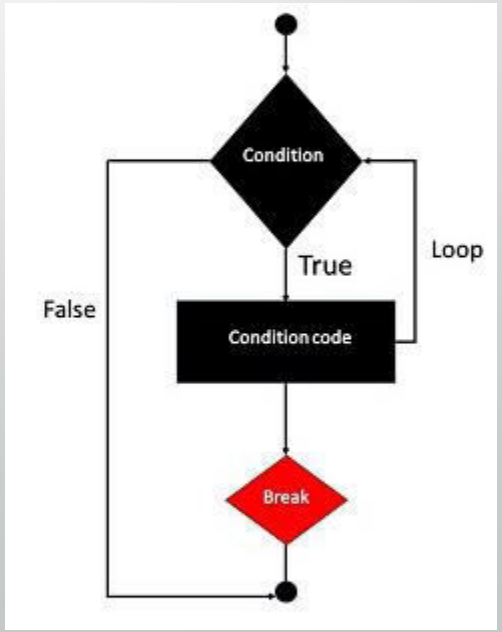
The break statement

- 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.

Example

- In the following example condition test becomes true when the counter value reaches 3 and loop terminates. <html>

```
<body>
<?php
$i = 0;
while( $i < 10)
{
$i++;
if( $i == 3 )
break;
}
echo ("Loop stopped at i = $i" );
?>
</body>
</html>
```



This will produce following result:

```
Loop stopped at i = 3
```

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.

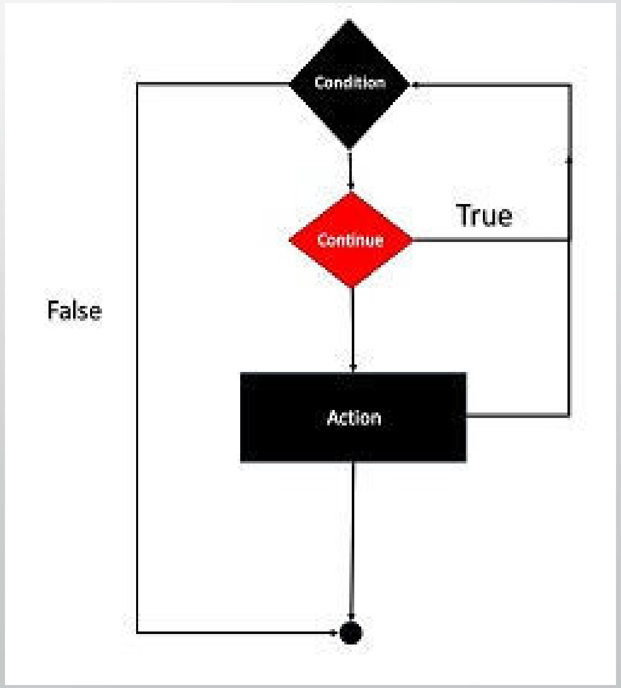
Example

- In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
<body>
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
if( $value == 3 )continue;
echo "Value is $value <br />";
}
?>
</body>
</html>
```

This will produce following result

```
Value is 1
Value is 2
Value is 4
Value is 5
```



THANK YOU

?

Web Programming

CHAPTER 3:
PHP Array, string and functions

Outlines

- PHP Arrays
  - **Numeric Array**
  - Associative Arrays
  - **Multidimensional array**
- PHP Strings
- PHP Functions

3.1 PHP Arrays

- An array is a data structure that **stores one or more similar type of values** in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.
- There are three different kind of arrays and each array value is accessed using an ID c which is called array index.
  - **Numeric array** - An array with a numeric **index**. Values are stored and accessed in linear fashion
  - **Associative array** - An array with **strings as index**. This stores element values in association with key values rather than in a strict linear index order.
  - **Multidimensional array** - An array **containing one or more arrays** and values are accessed using multiple indices
- **NOTE:** Built-in array functions is given in function reference PHP Array Functions

3.1.1 Numeric Array

- These arrays can store numbers, strings and any object but their index will be prepresented by numbers. **By default array index starts from zero.**

  Example

- Following is the example showing how to create and access numeric arrays.
- Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
<body>
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value )
{
echo "Value is $value <br />";
}
/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";
foreach( $numbers as $value )
{
echo "Value is $value <br />";
}
?>
</body>
</html>
```

```
Output becomes
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

Creating arrays

- In PHP, you can use the **array()** construct or **[] syntax** to define an array.
- The **[] syntax** is shorter and more convenient.
- **1) Creating an array using array() construct**
  - To define an array, you use the array() construct. The following example creates an empty array:

```
<?php $
empty_array = array ();
?>
```

- To create an array with some initial elements, you place a comma-separated list of elements within parentheses of the array() construct.
- For example, the following defines an array that has three numbers:

```
<?php
$scores = array(1, 2, 3);
?>
```

cont

- 2) Creating an array using the [] syntax
  - PHP provides a more convenient way to define arrays with the shorter syntax [], known as JSON notation. The following example uses [] syntax to create a new empty array:

```
<?php
$empty_array = [];
?>
```

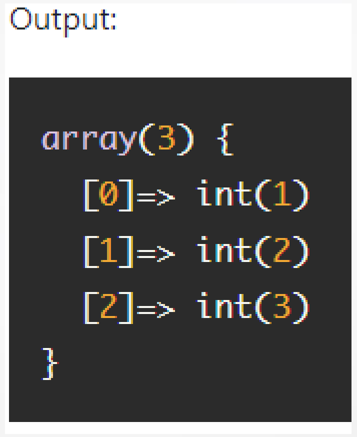
  - The following example uses the [] syntax to create a new array that consists of three numbers:

```
<?php
$scores = [1, 2, 3];
?>
```

Displaying arrays

- To show the contents of an array, you use the var_dump() function. For example:

```
<?php
$scores = [1, 2, 3];
var_dump($scores);
?>
```

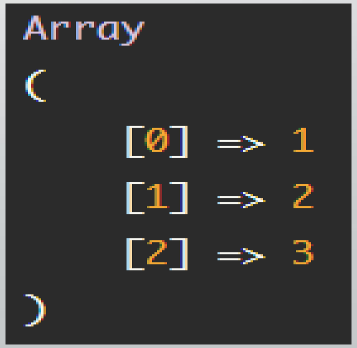


Output:

```
array(3) {
  [0]=> int(1)
  [1]=> int(2)
  [2]=> int(3)
}
```

- Or you can use the print_r() function:

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



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

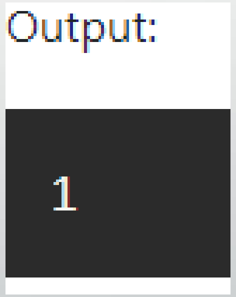
Accessing array elements

- To access an element in an array, you specify the index of the element within the square brackets:

  $array_name[index];

- Note that the index of the first element of an array begins with zero, not one.
- The following example shows how to access the first element of the array:

```
<?php
$scores = [1, 2, 3];
echo $scores[0];
?>
```



```
Output:

1
```

Adding an element to the array

- To add an element to an array, you use the following syntax:

```
$array_name[] = new_element;
```

- PHP will calculate the highest numerical index plus one each time you assign an element to the array.
- The following example shows how to add the number 4 to the $scores array:

```
<?php
$scores = [1, 2, 3];
$scores[] = 4;
?>
```

- In this example, we defined an array that consists of three numbers initially. Then, we added the number 4 to the array.
- It's possible to use an index when you add a new element to the array. For example:

```
$scores = [1, 2, 3];
$scores[3] = 4;
```

Removing array elements

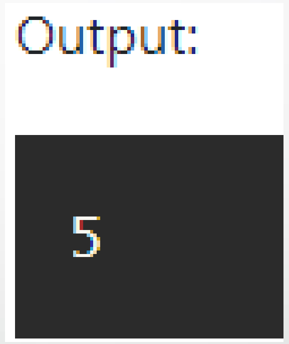
- To remove an element from an array, you use the **unset()** function. The following removes the second element of the $scores array:

```
<?php
$scores = [1, 2, 3];
unset($scores[1]);
?>
```

Getting the size of an array

- To get the number of elements in an array, you use the **count()** function. For example:

```php
<?php
$scores = [1, 2, 3, 4, 5];
echo count($scores);
?>
```



Output:

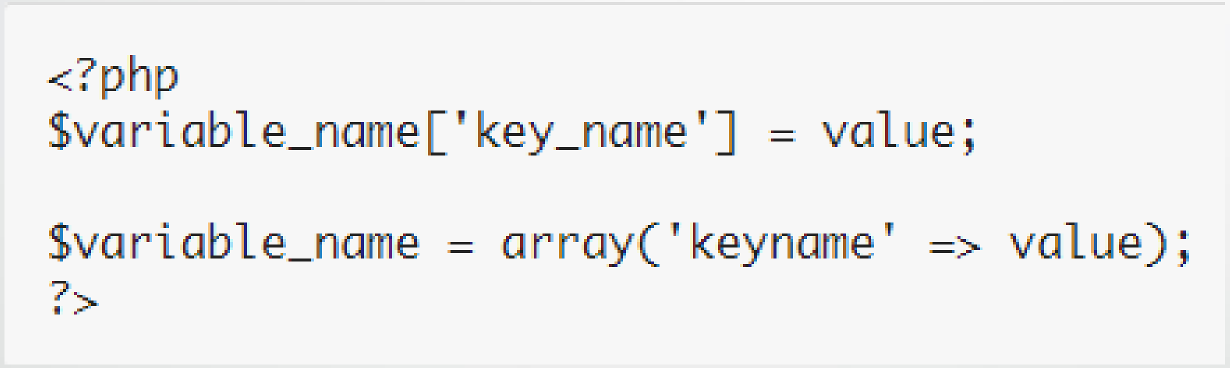
```
5
```

3.1.2 Associative Arrays

- The associative arrays are very similar to numeric arrays in term of functionality **but they are different in terms of their index.**
- Associative array will have their **index as string** so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice.
- Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.
- **NOTE: Don't keep associative array inside double quote** while printing otherwise it would not return any value.
- Associative array are also very useful when retrieving data from the database.
- The field names are used as id keys.

syntax for creating associative array

- Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.
- Below is the syntax for creating associative array in php.

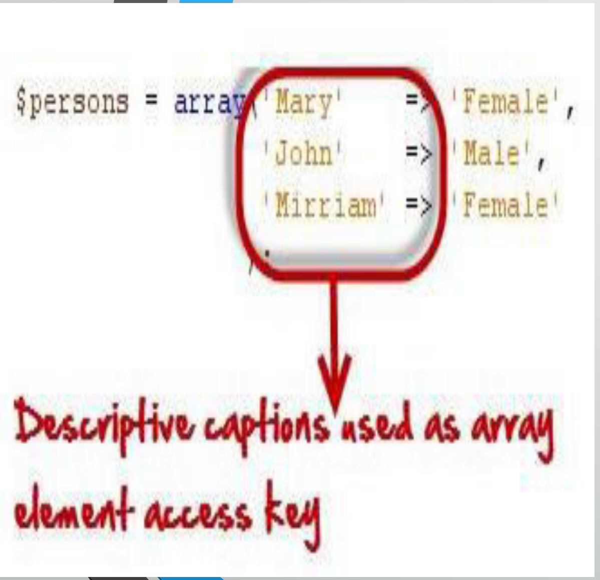


```php
<?php
$variable_name['key_name'] = value;

$variable_name = array('keyname' => value);
?>
```

- HERE,
  - “$variable_name…” is the name of the variable
  - “['key_name']” is the access index number of the element
  - “value” is the value assigned to the array element.

Cont---

- Let’s suppose that we have a group of persons, and we want to assign the gender of each person against their names.
- We can use an associative array to do that. The code below helps us to do that.



```
$persons = array('Mary'    => 'Female',
                 'John'    => 'Male',
                 'Mirriam' => 'Female'
```

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Fema
le");
print_r($persons);
echo "";
echo "Mary is a " . $persons["Mary"];
?>
```

Output:

```
Array ( [Mary] => Female [John] => Male [Mirriam] => Female ) Mary is a F
emale
```

Example 2

```
<html>
<body>
<?php
/* First method to associate create array. */
$salaries = array(
"mohammad" => 2000,
"qadir" => 1000,
"zara" => 500
);
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";
```

Cont..

```
/* Second method to create array. */
$salaries['mohammad'] = "high";
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br
/>";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";
?>
</body>
</html>
>
```

This will produce following result:

```
Salary of
mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Multidimensional Arrays

- A multi-dimensional array each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are **accessed using multiple index.**

Example

- In this example we create a two dimensional array to store marks of three students in three subjects:
- This example is an associative array, you can create numeric array in the same fashion.

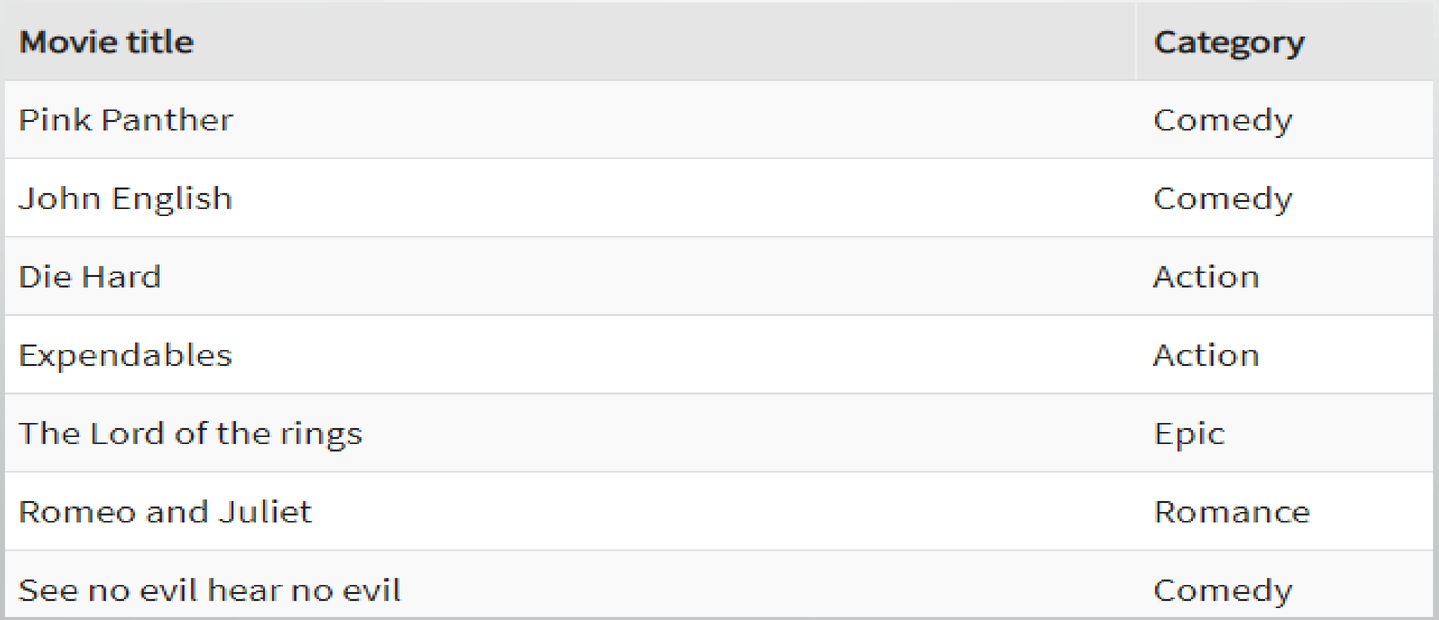
Cont..

```
<html> <body> <?php
$marks = array(
"mohammad" => array
(
"physics" => 35,
"maths" => 30,
"chemistry" => 39
),
"qadir" => array
(
"physics" => 30,
"maths" => 32,
 "chemistry" => 29
 ),
 "zara" => array
 (
 "physics" => 31,
 "maths" => 22,
 "chemistry" => 39
 )
 );
```

```
/* Accessing multi-dimensional
array values */

echo "Marks for mohammad in
physics : " ;

echo
$marks['mohammad']['physics'] .
"<br />";

echo "Marks for qadir in maths : ";

echo $marks['qadir']['maths'] .
"<br />";

echo "Marks for zara in chemistry
: " ;

echo $marks['zara']['chemistry'] .
"<br />";

?>

</body>

</html>
```

---cont

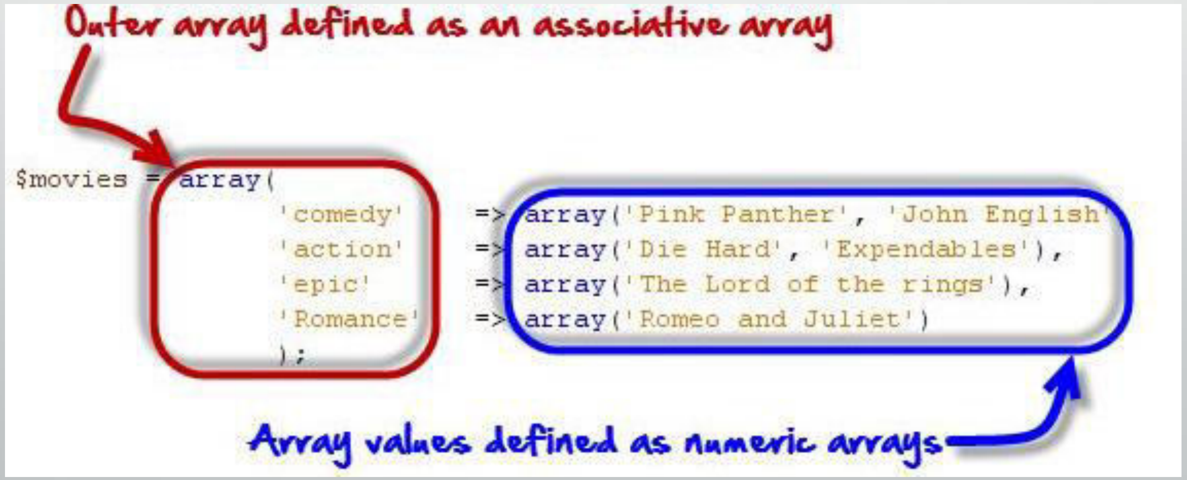
- These are arrays that contain other nested arrays.
- The advantage of multidimensional arrays is that they allow us to group related data together.
- Let’s now look at a practical example that implements a php multidimensional array.
- The table below shows a list of movies by category.



Movie title	Category
Pink Panther	Comedy
John English	Comedy
Die Hard	Action
Expendables	Action
The Lord of the rings	Epic
Romeo and Juliet	Romance
See no evil hear no evil	Comedy

- The above information can be represented as a multidimensional array. The code below shows the implementation.

```
<?php
$movies =array(
"comedy" => array("Pink Panther", "John English", "See no evil hear no ev
il"),
"action" => array("Die Hard", "Expendables"),
"epic" => array("The Lord of the rings"),
"Romance" => array("Romeo and Juliet")
);
print_r($movies);
?>
```


Outer array defined as an associative array
Array values defined as numeric arrays

```
$movies = array(
            'comedy'    => array('Pink Panther', 'John English'
            'action'    => array('Die Hard', 'Expendables'),
            'epic'      => array('The Lord of the rings'),
            'Romance'   => array('Romeo and Juliet')
            );
```

PHP Strings

- They are sequences of characters, like "PHP supports string operations".
- Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = ‘This is a somewhat longer, singly quoted string’;
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

String cont..

- **Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values** as well as specially interpreting certain character sequences.

```
<html>
<body>
<?
$variable = "name";
$literally = 'My $variable will not print!\\n';
print($literally);
$literally = "My $variable will print!\\n";
print($literally);
?>
 </body>
</html>
```

This will produce following result:

```
My $variable will not print!\n
My name will print
```

Cont..

- Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:
  - Certain character sequences beginning with backslash (\) are replaced with special characters
  - Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")

String Concatenation Operator

- To concatenate two string variables together, use the dot (.) operator:

```
<?php
$string1="Hello World";
$string2="1234";
echo $string1 . " " . $string2;
?>
```

This will produce following result:

```
Hello World 1234
```

Functions in PHP

- PHP Built-in Functions
- **PHP User-Defined Functions**

PHP Built-in Functions

- A function is a self-contained block of code that performs a specific task.
- PHP has a huge collection of internal or built-in functions that you can call directly within your PHP scripts to perform a specific task, like gettype(), print_r(), var_dump, date() etc.
- Please check out PHP reference section for a complete list of useful PHP built-in functions.

strlen() function

- The strlen() function is used to find the length of a string.

Let's find the length of our string "Hello world!":

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

The out put becomes:
12

The include() Function

- The include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **include()** function generates a warning but the script will continue execution.

  Example

Save the following code as menu.php on your system.

```
<html>
<body>
<a href="www.facebook.com">Home</a> -
<a href="www.infosa.com">Graduta class</a> -
<a href="www.bhu.com">Infosa photo Gallary</a> -
<a href="www.bulehora.php">Help</a> <br />
</body>
</html>
```

Cont..

Now create as many pages as you like and include this file to create header. For example now your test.php file can have following content.

```
<html>
<body>
<?php include("menu.php"); ?>
<p>This is an example to show how to include PHP file! That was found in the same folder or saved in the same folder</p>
</body>
</html>
```

This will produce following result:
Home – Graduate class- Infosa photo gallary- Help
This is an example to show how to include PHP file.

Creating PHP Function:

- Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.
- Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below:

```
<html>
<head>
<title>Writing PHP Function</title>
</head>
<body>
<?php
/* Defining a PHP Function */
function writeMessage()
{
echo "You are really a nice student, Have a nice time here @bulehora!";
}
/* Calling a PHP Function */
writeMessage();
?>
</body>
</html>
```

The out becomes:
You are really nice student have a nice time here @bulehora!

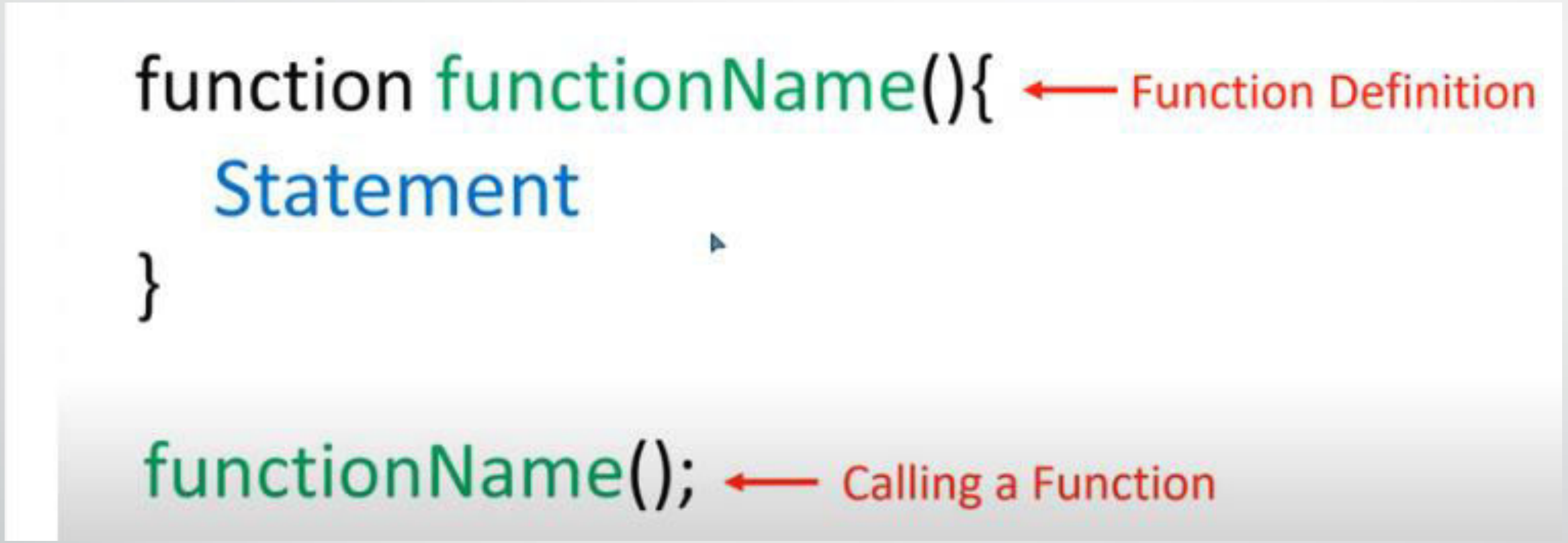
User Defined Functions

- User defined functions come in handy when;
  - you have routine tasks in your application such as adding data to the database
  - performing validation checks on the data
  - Authenticating users in the system etc.

Creating and Invoking Functions

- While creating a user defined function we need to keep few things in mind:
  - Any name ending with an open and closed parenthesis is a function.
  - A function name always begins with the keyword function.
  - To call a function we just need to write its name followed by the parenthesis
  - A function name cannot start with a number. It can start with an alphabet or underscore.
  - A function name is not case-sensitive.

The basic syntax of creating a custom function can be give with:

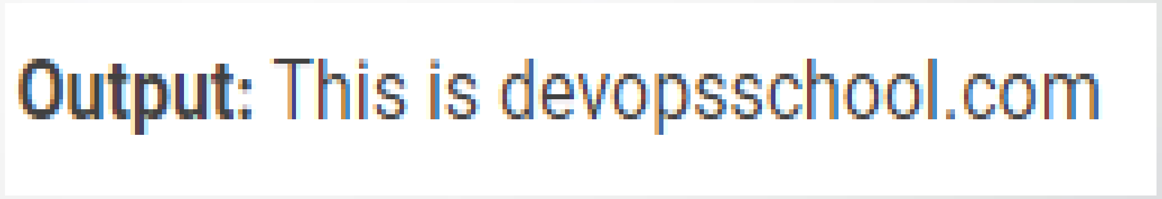


```
function functionName(){
   Statement
}

functionName();
```

- The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Function</title>
</head>
<body>
<?php
function devops()
{
   echo "This is devopsschool.com";
}
// Calling the function
devops();
?>
</body>
</html>
```

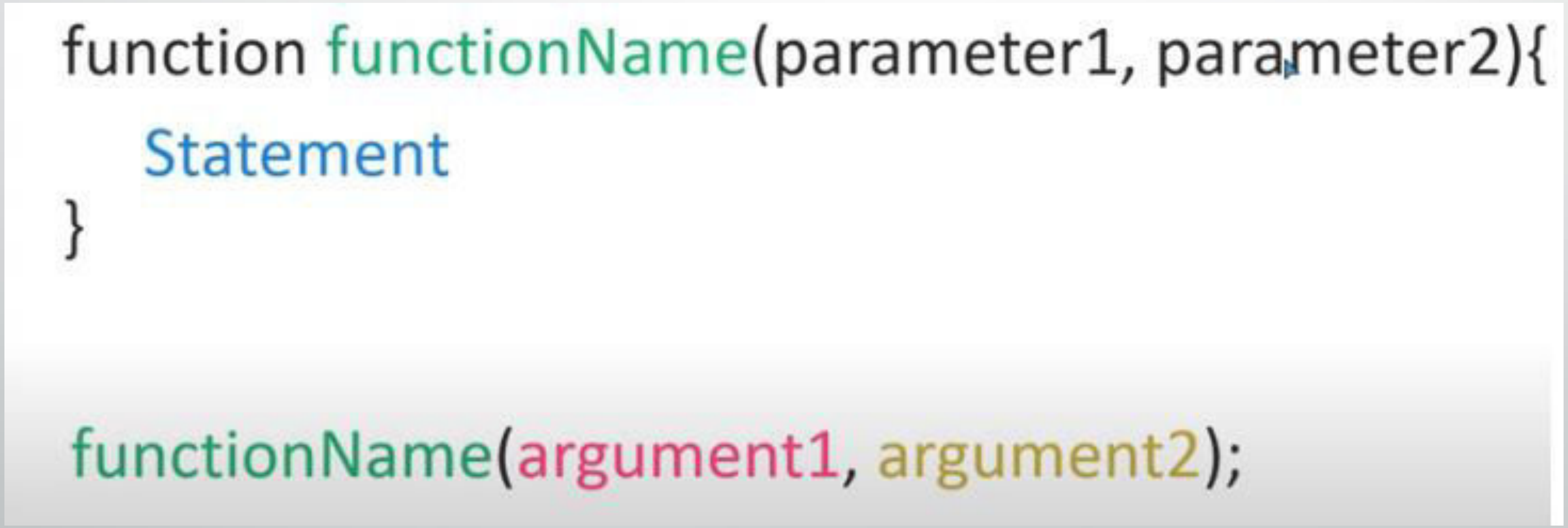


```
Output: This is devopsschool.com
```

Function Parameters or Arguments

- The information or variable, within the function's parenthesis, are called parameters.
- These are used to hold the values executable during runtime. A user is free to take in as many parameters as he wants, separated with a comma(,) operator.
- These parameters are used to accept inputs during runtime. While passing the values like during a function call, they are called arguments.
- An argument is a value passed to a function and a parameter is used to hold those arguments. In common term, both parameter and argument mean the same.
- We need to keep in mind that for every parameter, we need to pass its corresponding argument.

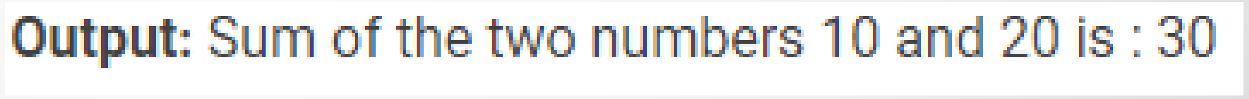
Syntax:



```
function functionName(parameter1, parameter2){
    Statement
}

functionName(argument1, argument2);
```

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>PHP Function with Parameters</title>
</head>
<body>
<?php
// Defining function
function getSum($num1, $num2){
 $sum = $num1 + $num2;
 echo "Sum of the two numbers $num1 and $num2 is : $sum";
}
 // Calling function
getSum(10, 20);
?>
</body>
</html>
```



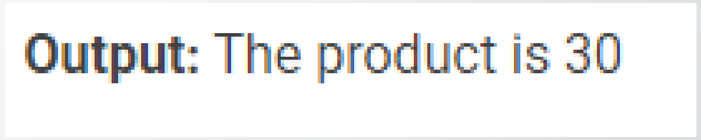
Output: Sum of the two numbers 10 and 20 is : 30

Returning Values from Functions

- Functions can also **return values to the part of program** from where it is called.
- The **return** keyword is used to return value back to the part of program, from where it was called.
- The returning value may be of **any type including the arrays and objects.**
- The return statement **also marks the end of the function** and stops the execution after that and returns the value.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Function Returning Values</title>
</head>
<body>
<?php
// function along with three parameters
function dharmu($num1, $num2, $num3)
{
   $product = $num1 * $num2 * $num3;
   return $product; //returning the product
}
// storing the returned value
$retValue = dharmu(2, 3, 5);
echo "The product is $retValue";
?>
</body>
</html>
```



Output: The product is 30

Parameter passing to Functions

- PHP allows us two ways in which an argument can be passed into a function:
- **Pass by Value:** On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.
- **Pass by Reference**: On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered. In pass by reference we actually pass the address of the value, where it is stored using ampersand sign(&).

Following example depicts both the cases.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Passing
Arguments by
Reference</title>
</head>
<body>
<?php
// pass by value
function
valdharmu($num) {
   $num += 2;
   return $num;
}
// pass by reference
Function refdharmu(&$num)
{      $num += 10;  return
       $num;

}

$n = 10;

valdharmu($n);

echo "The original value is still
$n \n";

refdharmu($n);

echo "The original value
changes to $n";

?> </body>

</html>
```

```
The out becomes:
The original value is
still 10
The original value
changes to 20
```

Variable Scope:

- Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of three scope types:
  - **Local variables**
  - **Global variables**
  - **Static variables**

Local variables:-

- The variable is only accessible from within the function (or method) that created it A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

```
<html>
<body>
<?php
$x = 4;
function assignx () {
$x = 0;
echo "\$x inside function is $x. ";
}
assignx();
echo "\$x outside of function is $x. ";
?>
</body>
</html>
```

The output will be:-

- **$x inside function is 0.**
- **$x outside of function is 4.**

Global variables:-

- The variable is accessible from anywhere in the script. Global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified.
- This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:

```
<html>
<body>
<?php
$somevar = 15;
function addit() {
GLOBAL $somevar;
$somevar++ ;
echo "Somevar is $somevar";
}
addit();
?>
</body>
</html>
```

The output will be:-

- **Somevar is 16**

Static variables:-

- this type of variables be either a global or local variable. Both are created by preceding the variable declaration with the keyword static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

```
<?php
function keep_track() {
STATIC $count = 0;
$count++;
echo $count; print " ";
}
keep_track();
keep_track();
keep_track();
?>
```

This will produce following result.

```
1
2
3
```

THANK YOU

?

Web Programming

CHAPTER 4:
Processing form with PHP and basic database concepts

PHP form handling/processing.

What is Form?

- A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.
- When you login into a website or into your mail box, you are interacting with a form.
- Forms are used to get input from the user and submit it to the web server for processing.
- The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.
- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

7.1 About Forms

- Forms can contain;
  - Text boxes
  - Password boxes
  - Check boxes
  - Radio buttons
  - Buttons
  - Select lists
  - Text areas
  - Labels
  - Fieldsets
  - Legends
  - and more.. (will be discussed in 7.3)

Form Elements

❖ All form elements should be written in between the <form>..</form> tags.

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<label>	Defines a label for an <input> element
<textarea>	Defines a multiline input control
<select>	Defines a drop-down list
<option>	Defines an option in a drop-down list
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<button>	Defines a clickable button

Form Elements - <form>

❖ The <FORM> </FORM> element is used to create an HTML form and act as a container for form elements. Although the form element itself isn't usually a visible part of the page (like the body tag), it could be with appropriate CSS.

Most commonly used **FORM** element **Attributes**

Attribute	Description
method	Specifies the HTTP method used when submitting the form
action	Specifies an address (url) where to submit the form
autocomplete	Specifies if the browser should autocomplete the form
novalidate	Specifies that the browser should not validate the form.
name	Specifies a name used to identify the form

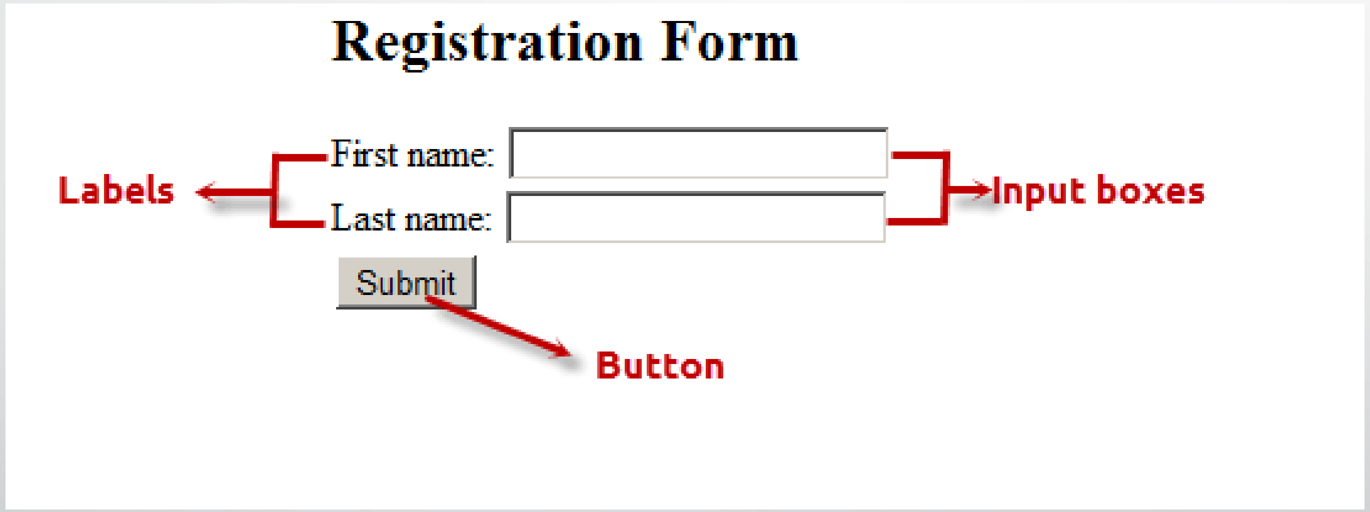
Create a form

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons, checkboxes

The code below creates a simple registration form

```
<html>
<head>
<title>Registration Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h2>Registration Form</h2>
<form action="registration_form.php" method="POST">
First name: <input type="text" name="firstname"> <br>
Last name: <input type="text" name="lastname">
<input type="hidden" name="form_submitted" value="1" />
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Viewing the above code in a web browser displays the following form.


Registration Form

First name:

Last name:

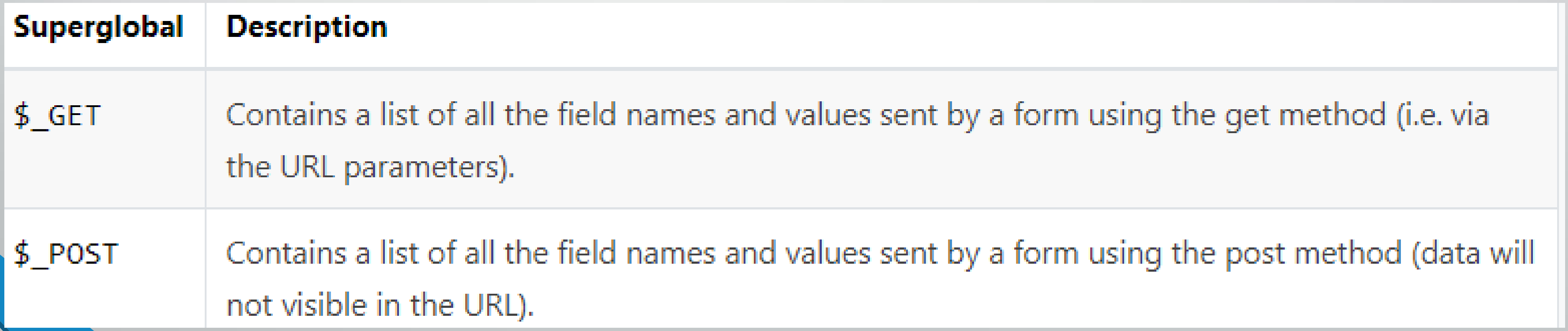
Submit

HERE,

- **<form...>...</form>** are the opening and closing form tags
- **action="registration_form.php**" method="POST"> specifies the destination URL and the submission type.
- **First/Last name**: are labels for the input boxes
- **<input type="text"...>** are input box tags
- **<br>** is the new line tag
- **<input type="hidden" name="form_submitted" value="1"/>** is a hidden value that is used to check whether the form has been submitted or not
- **<input type="submit" value="Submit">** is the button that when clicked submits the form to the server for processing

Submitting the form data to the server

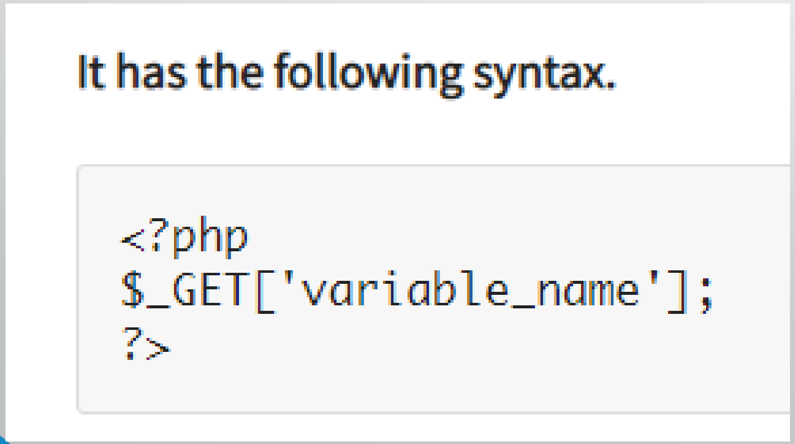
- The **action** attribute of the form specifies the submission URL that processes the data.
- The **method** attribute specifies the submission type.
- To access the value of a particular form field, you can use the following superglobal variables.
- These variables are available in all scopes throughout a script.



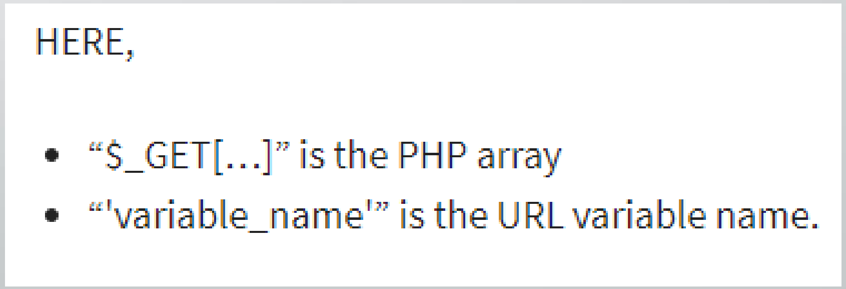
Superglobal	Description
$_GET	Contains a list of all the field names and values sent by a form using the get method (i.e. via the URL parameters).
$_POST	Contains a list of all the field names and values sent by a form using the post method (data will not visible in the URL).

PHP GET method

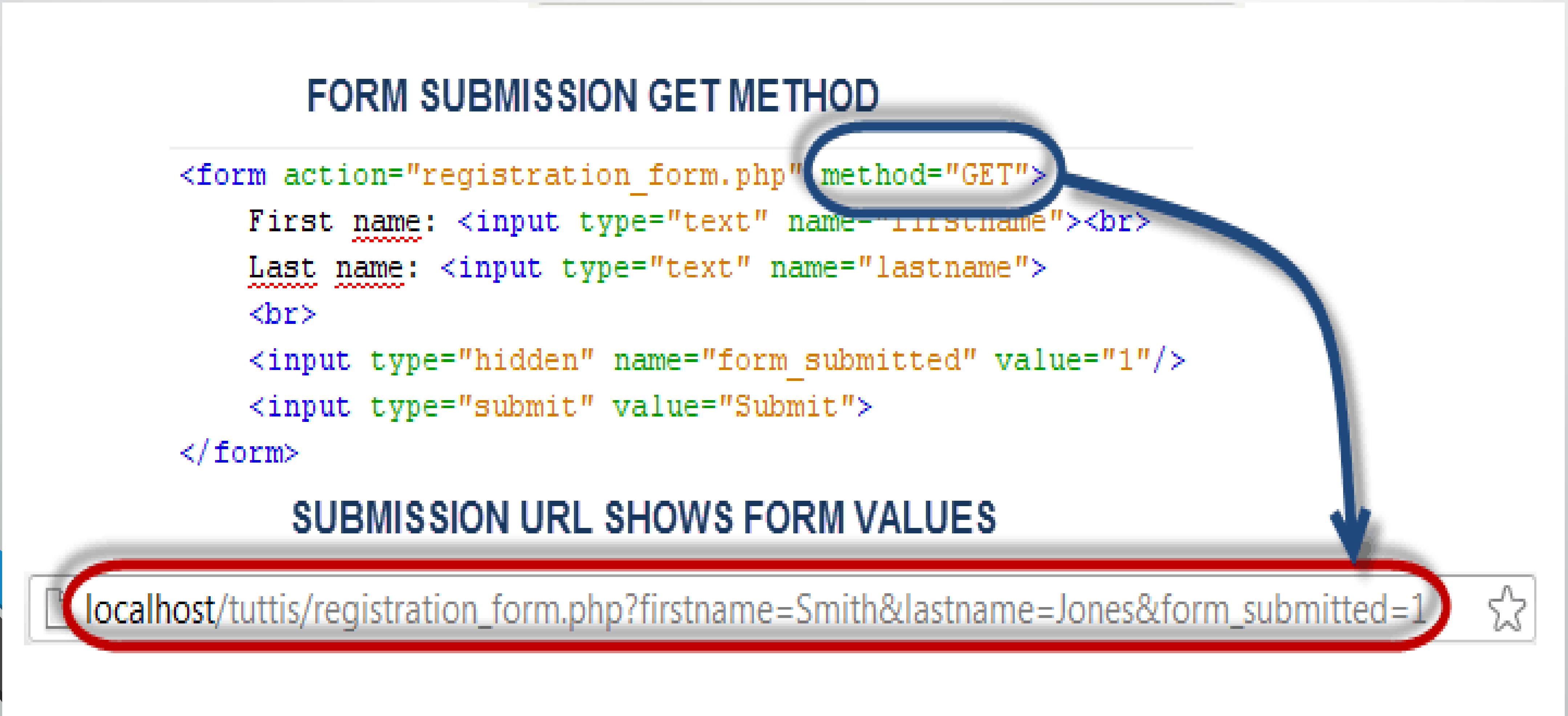
- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.


It has the following syntax.

```php
<?php
$_GET['variable_name'];
?>
```


HERE,

- “$_GET[…]” is the PHP array
- “'variable_name'” is the URL variable name.



FORM SUBMISSION GET METHOD

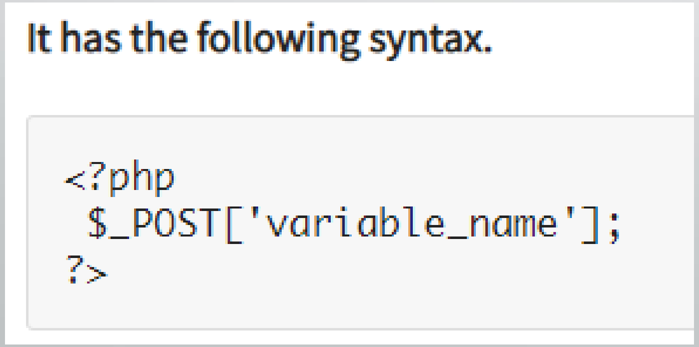
```
<form action="registration_form.php" method="GET">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

SUBMISSION URL SHOWS FORM VALUES

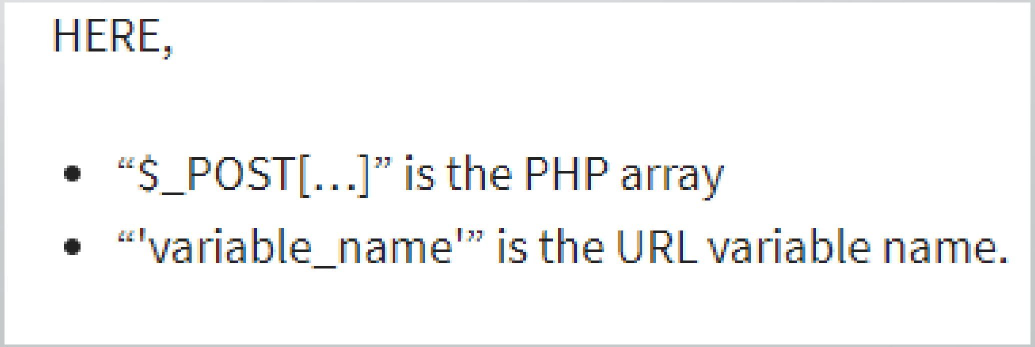
```
localhost/tuttis/registration_form.php?firstname=Smith&lastname=Jones&form_submitted=1
```

PHP POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

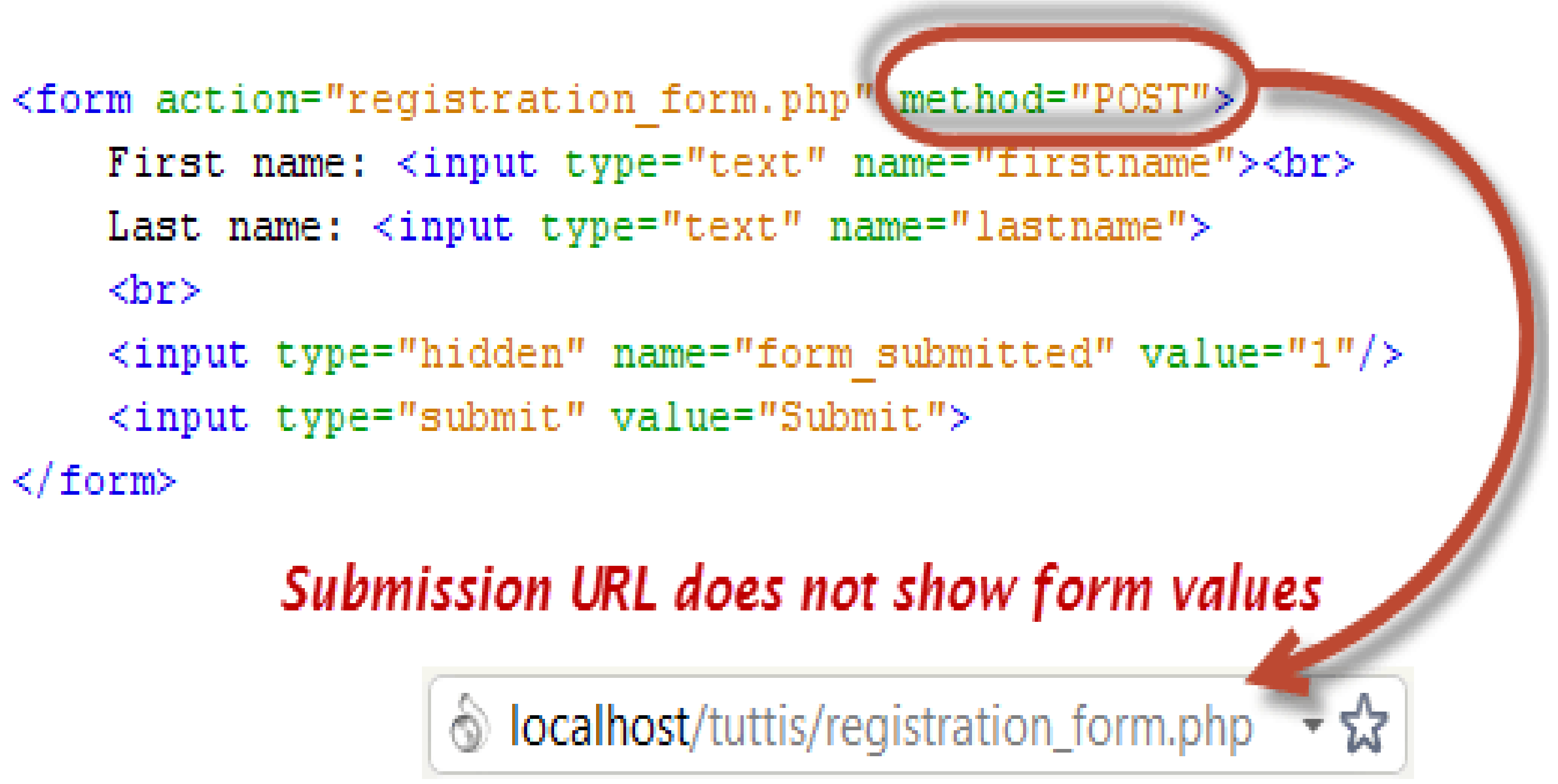

It has the following syntax.

```
<?php
 $_POST['variable_name'];
?>
```


HERE,

- “$_POST[…]” is the PHP array
- “'variable_name'” is the URL variable name.

FORM SUBMISSION POST METHOD

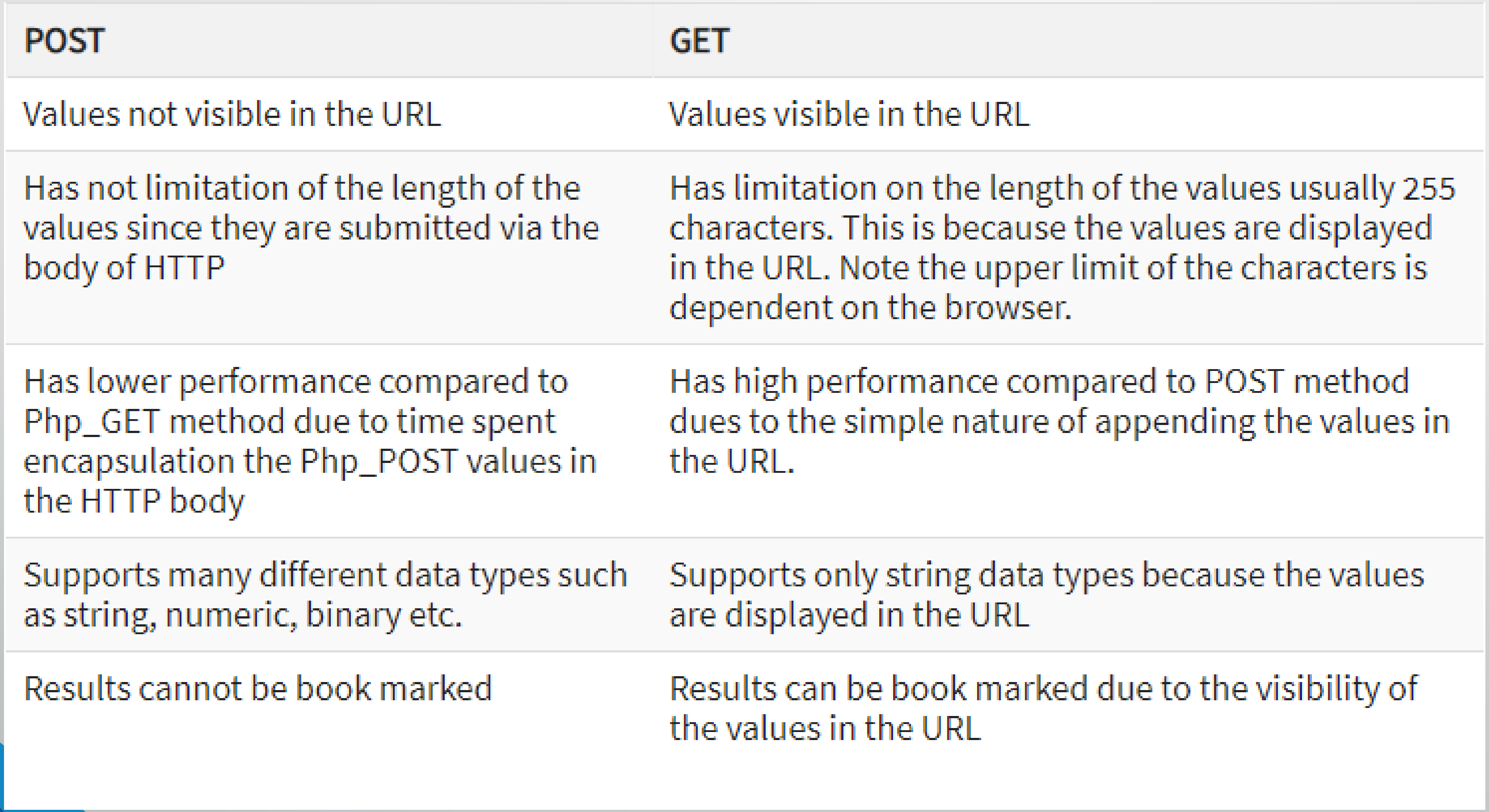


```
<form action="registration_form.php" method="POST">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

Submission URL does not show form values

localhost/tuttis/registration_form.php

GET vs POST Methods



POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

When to use GET?

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).
- GET also has limits on the amount of information to send.
- The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending **non-sensitive data.**
- Note: GET should **NEVER be used for sending passwords or other sensitive information!**

When to use POST?

- Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST **supports advanced functionality** such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL**, it is not possible to bookmark the page.**
- Developers prefer **POST for sending form data.**

GET Example

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

welcome_get.php

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo
$_GET["email"]; ?>

</body>
</html>
```

Example For post

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

- To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo
$_POST["email"]; ?>

</body>
</html>
```

Databases, MySQL & PHP

Database

- Structured collection of data.
  - Tables
  - Fields
  - Query
  - Reports
- Essentially a much more sophisticated implementation of the flat files.

What is MySQL

- MySQL is the most popular database system used with the PHP language
- MySQL is one of the most popular relational database system being used on the Web today.
- It is freely available and easy to install, however if you have installed Wampserver or xamp server it already there on your machine.
- MySQL database stores data into tables like other relational database. A table is a collection of related data, and it is divided into rows and columns.

Advantages of mysql

- •MySQL is easy to use, yet extremely powerful, fast, secure, and scalable.
- •MySQL runs on a wide range of operating systems, including UNIX or Linux, Microsoft Windows, Apple Mac OS X, and others.
- •MySQL supports standard SQL (Structured Query Language).
- •MySQL is ideal database solution for both small and large applications.
- •MySQL is developed, and distributed by Oracle Corporation.
- •MySQL includes data security layers that protect sensitive data from intruders.

- Each **row in a table represents a data record** that are inherently connected to each other such as information related to a particular person, whereas **each column represents a specific field** such as id, first_name, last_name, email, etc.
- The structure of a simple MySQL table that contains person's general information may look something like this:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

Talking to MySQL Databases with SQL

- SQL, the Structured Query Language, is a simple, standardized language for communicating with relational databases like MySQL.
- With SQL you can perform any database-related task, **such as creating databases and tables, saving data in database tables, query a database for specific records, deleting and updating data in databases.**
- Look at the following standard SQL query that returns the email address of a person whose first name is equal to 'Peter' in the persons table:

```
SELECT email FROM persons WHERE first_name="Peter"
```

- If you execute the SQL query above it will return the following record:

```
peterparker@mail.com
```

PHP Connect to MySQL Server

- Ways of Connecting to MySQL through PHP
- In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server.
- PHP offers two different ways to connect to MySQL server:
  - MySQLi (Improved MySQL) and
  - PDO (PHP Data Objects) extensions.
- While the PDO extension is more portable and supports more than twelve different databases, MySQLi extension as the name suggests supports MySQL database only.
- MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server.
- Both PDO and MySQLi offer an object-oriented API, but MySQLi also offers a procedural API which is relatively easy for beginners to understand.

Connecting to MySQL Database Server

- In PHP you can easily do this using the **mysqli_connect()** function.
- All communication between PHP and the MySQL database server takes place through this connection.
- Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:
- **Syntax: MySQLi, Procedural way**

```
$link = mysqli_connect("hostname", "username", "password", "database");
```

- **Syntax: MySQLi, Object Oriented way**

```
$mysqli = new mysqli("hostname", "username", "password", "database");
```

- **Syntax: PHP Data Objects (PDO) way**

```
$pdo = new PDO("mysql:host=hostname;dbname=database", "username",
"password");
```

Cont---

- **The hostname parameter** in the above syntax specify the host name (e.g. localhost), or IP address of the MySQL server,
- whereas **the username and password parameters** specifies the credentials to access MySQL server, and
- the **database parameter**, if provided will specify the default MySQL database to be used when performing queries.
- The following example shows how to connect to MySQL database server using MySQLi (both procedural and object oriented way) and PDO extension.

Example Procedural

```php
<?php
/* Attempt MySQL server connection. Assuming
you are running MySQL server with default
setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root",
"");
// Check connection
if($link === false){ die("ERROR: Could not
connect. " . mysqli_connect_error()); }
// Print host information echo "Connect
Successfully. Host info: " .
mysqli_get_host_info($link);
 // Close connection
 mysqli_close($link);
?>
```

Example Object Oriented

```php
<?php

 /* Attempt MySQL server connection. Assuming
you are running MySQL server with default
setting (user 'root' with no password) */

$mysqli = new mysqli("localhost", "root", "",
"demo");

// Check connection

if($mysqli === false){ die("ERROR: Could not
connect. " . $mysqli->connect_error); }

// Print host information

echo "Connect Successfully. Host info: " .
$mysqli->host_info;

// Close connection

$mysqli->close();

?>
```

Example PDO

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL server with default setting (user 'root'
with no password) */
try{ $pdo = new
PDO("mysql:host=localhost;dbname=demo", "root", "");
// Set the PDO error mode to exception
$pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
// Print host information
echo "Connect Successfully. Host info: " . $pdo-
>getAttribute(constant("PDO::ATTR_CONNECTION_STATUS"))
; } catch(PDOException $e){ die("ERROR: Could not
connect. " . $e->getMessage()); }
 // Close connection
unset($pdo);
?>
```

- **Note:** The default username for MySQL database server is root and there is no password. However to prevent your databases from intrusion and unauthorized access you should set password for MySQL accounts.
- **Tip:** Setting the PDO::ATTR_ERRMODE attribute to PDO::ERRMODE_EXCEPTION tells PDO to throw exceptions whenever a database error occurs.
- The connection to the MySQL database server will be closed **automatically as soon as the execution of the script ends**. However, if you want to close it earlier you can do this by simply **calling the PHP mysqli_close() function.**

Creating MySQL Database Using PHP

- Before saving or accessing the data, we need to create a database first.
- The **CREATE DATABASE** statement is used to create a new database in MySQL.
- Let's make a SQL query using the CREATE DATABASE statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to finally create our database.
- The following example creates a database named demo.

Example (Procedural)

```php
<?php
 /* Attempt MySQL server connection. Assuming you are running MySQL server with
default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "");
// Check connection
if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }
// Attempt create database query execution
$sql = "CREATE DATABASE demo"; if(mysqli_query($link, $sql)){ echo "Database
created successfully"; } else{ echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link); }
// Close connection
mysqli_close($link);
?>
```

Creating Tables inside MySQL Database Using PHP

- Now it's time to create some tables inside the database that will actually hold the data.
- A table organizes the information into rows and columns.
- The **SQL CREATE TABLE** statement is used to create a table in database.
- Let's make a SQL query using the CREATE TABLE statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to finally create our table.

Example (Procedural)

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL server
with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection

if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); }

// Attempt create table query execution

$sql = "CREATE TABLE persons( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
first_name VARCHAR(30) NOT NULL, last_name VARCHAR(30) NOT NULL, email VARCHAR(70)
NOT NULL UNIQUE )";

if(mysqli_query($link, $sql)){ echo "Table created successfully."; } else{ echo
"ERROR: Could not able to execute $sql. " . mysqli_error($link); }

 // Close connection mysqli_close($link); ?>
```

Cont---

- The PHP code in the above example **creates a table named persons** with four columns id, first_name, last_name and email inside the demo database.
- Notice that each field name is followed by a **data type declaration**; this declaration specifies what type of data the column can hold, whether integer, string, date, etc.
- There are a few **additional constraints** (also called modifiers) that are specified after the column name in the preceding SQL statement, like NOT NULL, PRIMARY KEY, AUTO_INCREMENT, etc. Constraints define rules regarding the values allowed in columns.
- Please check out the tutorial on SQL CREATE TABLE statement for the detailed information about syntax, as well as the data types and constraints available in MySQL database system.

Inserting Data into a MySQL Database Table

- Now that you've understood how to create database and tables in MySQL. In this tutorial you will learn how to execute SQL query to insert records into a table.
- **The INSERT INTO** statement is used to insert new rows in a database table.
- Let's make a SQL query using the INSERT INTO statement with appropriate values, after that we will execute this insert query through passing it to the PHP mysqli_query() function to insert data in table.
- Here's an example, which insert a new row to the persons table by specifying values for the first_name, last_name and email fields.

Example (Procedural)

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */

 $link = mysqli_connect("localhost", "root", "", "demo");

// Check connection if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }

// Attempt insert query execution

 $sql = "INSERT INTO persons (first_name, last_name, email) VALUES ('Peter',
'Parker', 'peterparker@mail.com')"; if(mysqli_query($link, $sql))

{ echo "Records inserted successfully."; } else{ echo "ERROR: Could not able
to execute $sql. " . mysqli_error($link); }

// Close connection mysqli_close($link);

?>
```

Cont--

- If you remember from the preceding chapter, the id field was marked with the AUTO_INCREMENT flag. This modifier tells the MySQL to automatically assign a value to this field if it is left unspecified, by incrementing the previous value by 1.

Inserting Multiple Rows into a Table

- You can also insert multiple rows into a table with a single insert query at once.
- To do this, include multiple lists of column values within the INSERT INTO statement, where column values for each row must be enclosed within parentheses and separated by a comma.
- Let's insert few more rows into the persons table, like this:

Example (Procedural)

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }
// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES ('John',
'Rambo', 'johnrambo@mail.com'), ('Clark', 'Kent', 'clarkkent@mail.com'),
('John', 'Carter', 'johncarter@mail.com'), ('Harry', 'Potter',
'harrypotter@mail.com')";
if(mysqli_query($link, $sql)){ echo "Records added successfully."; } else{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }
// Close connection mysqli_close($link);
?>
```

Insert Data into a Database from an HTML Form

- In the previous section, we have learned how to insert data into database from a PHP script.
- Now, we'll see how we can insert data into database obtained from an HTML form.
- Let's create an HTML form that can be used to insert new records to persons table.

Step 1: Creating the HTML Form

- Here's a simple HTML form that has three text <input> fields and a submit button.

```
<!DOCTYPE html>

<html lang="en"> <head> <meta charset="UTF-8">

<title>Add Record Form</title> </head>

<body>

<form action="insert.php" method="post"> <p> <label
for="firstName">First Name:</label> <input type="text"
name="first_name" id="firstName"> </p> <p> <label for="lastName">Last
Name:</label> <input type="text" name="last_name" id="lastName"> </p>
<p> <label for="emailAddress">Email Address:</label> <input
type="text" name="email" id="emailAddress"> </p> <input type="submit"
value="Submit">

</form>

</body>

</html>
```

Step 2: Retrieving and Inserting the Form Data

- When a user clicks the submit button of the add record HTML form, in the example above, the form data is sent to 'insert.php' file.
- The 'insert.php' file connects to the MySQL database server, retrieves forms fields using the PHP $_REQUEST variables and finally execute the insert query to add the records.
- Here is the complete code of our 'insert.php' file:

'insert.php'

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL server
with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }

// Escape user inputs for security

$first_name = mysqli_real_escape_string($link, $_REQUEST['first_name']);
$last_name = mysqli_real_escape_string($link, $_REQUEST['last_name']); $email =
mysqli_real_escape_string($link, $_REQUEST['email']);

// Attempt insert query execution $sql = "INSERT INTO persons (first_name,
last_name, email) VALUES ('$first_name', '$last_name', '$email')";
if(mysqli_query($link, $sql)){ echo "Records added successfully."; } else{ echo
"ERROR: Could not able to execute $sql. " . mysqli_error($link); }

// Close connection mysqli_close($link);

?>
```

Selecting Data From Database Tables

- So far you have learnt how to create database and table as well as inserting data. Now it's time to retrieve data what have inserted in the preceding tutorial.
- The **SQL SELECT** statement is used to select the records from database tables. Its basic syntax is as follows:

  SELECT column1_name, column2_name, columnN_name FROM table_name;
- Let's make a SQL query using the SELECT statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to retrieve the table data.
- Consider our persons database table has the following records:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

- The PHP code in the following example selects all the data stored in the persons table (using the asterisk character (*) in place of column name selects all the data in the table).

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }

// Attempt select query execution $sql = "SELECT * FROM persons"; if($result
= mysqli_query($link, $sql)){ if(mysqli_num_rows($result) > 0){ echo
"<table>"; echo "<tr>"; echo "<th>id</th>"; echo "<th>first_name</th>"; echo
"<th>last_name</th>"; echo "<th>email</th>"; echo "</tr>"; while($row =
mysqli_fetch_array($result)){ echo "<tr>"; echo "<td>" . $row['id'] .
"</td>"; echo "<td>" . $row['first_name'] . "</td>"; echo "<td>" .
$row['last_name'] . "</td>"; echo "<td>" . $row['email'] . "</td>"; echo
"</tr>"; } echo "</table>";

// Free result set mysqli_free_result($result); } else{ echo "No records
matching your query were found."; } } else{ echo "ERROR: Could not able to
execute $sql. " . mysqli_error($link); }

// Close connection mysqli_close($link); ?>
```

Explanation of above Code (Procedural style)

- In the example above, the data returned by the mysqli_query() function is stored in the $result variable.
- Each time mysqli_fetch_array() is invoked, it returns the next row from the result set as an array.
- The while loop is used to loops through all the rows in the result set.
- Finally the value of individual field can be accessed from the row either by passing the field index or field name to the $row variable like $row['id'] or $row[0], $row['first_name'] or $row[1], $row['last_name'] or $row[2], and $row['email'] or $row[3].
- If you want to use the for loop you can obtain the loop counter value or the number of rows returned by the query by passing the $result variable to the mysqli_num_rows() function. This loop counter value determines how many times the loop should run.

PHP MySQL WHERE Clause

```
<?php /* Attempt MySQL server connection. Assuming you are
running MySQL server with default setting (user 'root' with no
password) */ $link = mysqli_connect("localhost", "root", "",
"demo"); // Check connection if($link === false){ die("ERROR:
Could not connect. " . mysqli_connect_error()); } // Attempt
select query execution $sql = "SELECT * FROM persons WHERE
first_name='john'"; if($result = mysqli_query($link, $sql)){
if(mysqli_num_rows($result) > 0){ echo "<table>"; echo "<tr>";
echo "<th>id</th>"; echo "<th>first_name</th>"; echo
"<th>last_name</th>"; echo "<th>email</th>"; echo "</tr>";
while($row = mysqli_fetch_array($result)){ echo "<tr>"; echo
"<td>" . $row['id'] . "</td>"; echo "<td>" . $row['first_name']
. "</td>"; echo "<td>" . $row['last_name'] . "</td>"; echo
"<td>" . $row['email'] . "</td>"; echo "</tr>"; } echo
"</table>"; // Close result set mysqli_free_result($result); }
else{ echo "No records matching your query were found."; } }
else{ echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link); } // Close connection mysqli_close($link);
?>
```

PHP MySQL LIMIT Clause

```
<?php /* Attempt MySQL server connection. Assuming you are
running MySQL server with default setting (user 'root' with no
password) */ $link = mysqli_connect("localhost", "root", "",
"demo"); // Check connection if($link === false){ die("ERROR:
Could not connect. " . mysqli_connect_error()); } // Attempt
select query execution $sql = "SELECT * FROM persons LIMIT 3";
if($result = mysqli_query($link, $sql)){
if(mysqli_num_rows($result) > 0){ echo "<table>"; echo "<tr>";
echo "<th>id</th>"; echo "<th>first_name</th>"; echo
"<th>last_name</th>"; echo "<th>email</th>"; echo "</tr>";
while($row = mysqli_fetch_array($result)){ echo "<tr>"; echo
"<td>" . $row['id'] . "</td>"; echo "<td>" . $row['first_name'] .
"</td>"; echo "<td>" . $row['last_name'] . "</td>"; echo "<td>" .
$row['email'] . "</td>"; echo "</tr>"; } echo "</table>"; //
Close result set mysqli_free_result($result); } else{ echo "No
records matching your query were found."; } } else{ echo "ERROR:
Could not able to execute $sql. " . mysqli_error($link); } //
Close connection mysqli_close($link); ?>
```

Updating Database Table Data

- The UPDATE statement is used to change or modify the existing records in a database table.
- This statement is typically used in conjugation with the WHERE clause to apply the changes to only those records that matches specific criteria.
- The basic syntax of the UPDATE statement can be given with:

UPDATE table_name SET column1=value, column2=value2,... WHERE column_name=some_value

- Let's make a SQL query using the UPDATE statement and WHERE clause, after that we will execute this query through passing it to the PHP mysqli_query() function to update the tables records.
- Consider the following persons table inside the demo database:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

- The PHP code in the following example will update the email address of a person in the persons table whose id is equal to 1.

Update

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection

if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }

// Attempt update query execution

$sql = "UPDATE persons SET email='peterparker_new@mail.com' WHERE id=1";
if(mysqli_query($link, $sql)){ echo "Records were updated successfully."; }
else { echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}

// Close connection

mysqli_close($link);

?>
```

Deleting Database Table Data

- Just as you insert records into tables, you can delete records from a table using the **SQL DELETE** statement.
- It is typically used in conjugation with the WHERE clause to delete only those records that matches specific criteria or condition.
- The basic syntax of the DELETE statement can be given with:

```
DELETE FROM table_name WHERE column_name=some_value
```

- Let's make a SQL query using the DELETE statement and WHERE clause, after that we will execute this query through passing it to the PHP mysqli_query() function to delete the tables records.
- Consider the following persons table inside the demo database:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

- The PHP code in the following example will delete the records of those persons from the persons table whose first_name is equal to John.

Delete record

```
<?php /* Attempt MySQL server connection. Assuming you are running MySQL server
with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection if($link === false){ die("ERROR: Could not connect. " .
mysqli_connect_error()); }

// Attempt delete query execution

$sql = "DELETE FROM persons WHERE first_name='John'"; if(mysqli_query($link,
$sql)){ echo "Records were deleted successfully."; } else{ echo "ERROR: Could
not able to execute $sql. " . mysqli_error($link); }

// Close connection mysqli_close($link); ?>
```

After the deletion the *persons* table will look something like this:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

As you can see the records has been deleted successfully from the persons table.

Warning: The `WHERE` clause in the `DELETE` statement specifies which record or records should be deleted. If you omit the `WHERE` clause, all records will be deleted.

PHP MySQL CRUD Application

- What is CRUD
- CRUD is an acronym for Create, Read, Update, and Delete. CRUD operations are basic data manipulation for database.
- We've already learned how to perform create (i.e. insert), read (i.e. select), update and delete operations in previous chapters.
- we'll create a simple PHP application to perform all these operations on a MySQL database table at one place.

Well, let's start by creating the table which we'll use in all of our example.

```
CREATE TABLE employees (    id INT NOT NULL PRIMARY
KEY AUTO_INCREMENT,    name VARCHAR(100) NOT
NULL,    address VARCHAR(255) NOT NULL,    salary
INT(10) NOT NULL);
```

Steps

- Creating the Config File
- Creat the index page
- Create the insert page
- Creating the Read Page
- Creating the update Page
- Creating the delete Page
- Creating the error Page

Web Programming

CHAPTER 5:
PHP Files, Cookies and Session

Outlines

PHP Files & I/O using function

This topic will explain following functions related to files:

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening and Closing Files

- The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

Files modes can be specified as one of the six options in the following description:

- **r**: Opens the file for reading only. Places the file pointer at the beginning of the file.
- **r+:** Opens the file for reading and writing. Places the file pointer at the beginning of the file.
- **w:** Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
- **w+:** Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
- **a**: Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
- **a+:** Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

Reading a file

- Once a file is opened using **fopen()** function it can be read with a function called **fread()**. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.
- The files's length can be found using the **filesize()** function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using **fopen()** function.
- Get the file's length using **filesize()** function.
- Read the file's content using **fread()** function.
- Close the file with **fclose()** function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<head>
<title>Reading a file using
PHP</title>
</head>
<body>
<?php
$filename = “test.php";
$file = fopen( $filename, "r" );
if( $file == false )
{
echo ( "Error in opening file" );
exit();
}
$filesize = filesize( $filename );
$filetext = fread( $file, $filesize );
fclose( $file );
echo ( "File size : $filesize bytes" );
echo ( "<pre>this how file read using
php</pre>" );
?>
</body>
</html>
```

```
File size : 132 bytes
this how file read using php
```

Writing a file

- A new file can be written or text can be appended to an existing file using the PHP fwrite()function. This function requires two arguments specifying a file pointer and the string of data that is to be written. Optionally a third intger argument can be included to specify the length of the data to write. If the third argument is included, writing would will stop after the specified length has been reached.
- The following example creates a new text file then writes a short text heading insite it. After closing this file its existence is confirmed using file_exist() function which takes file name as an argument.

```
<?php
$filename = "test.php";
$file = fopen( $filename, "w" );
if( $file == false )
{
echo ( "Error in opening new file" );
exit();
}
fwrite( $file, "This is a simple test\n" );
fclose( $file );
?>
<html>
<head>
<title>Writing a file using PHP</title>
</head>
<body>
<?php
if($filename )
{
$filesize = filesize( $filename );
$msg = "File created with name
$filename ";
$msg .= "containing $filesize bytes";
echo ($msg );
}
else { echo ("File $filename does not
exit" );
}
?>
</body>
 </html>
```

Session in PHP

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Bore using any session variable make sure you have setup this path

Session in PHP

When a session is started following things happen:

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7f0j34c3jj973hjkop2fc937e3443.
- A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7f0j34c3jj973hjkop2fc937e3443.

Session in PHP

- When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.
- A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

- A PHP session is easily started by making a call to the **session_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.
- Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a session.
- The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.
- Make use of **isset()** function to check if session variable is already set or not.

```
<?php
        session_start();
        if( isset( $_SESSION['counter'] ) )
        {
          $_SESSION['counter'] += 1;
        }
        else
        {
          $_SESSION['counter'] = 1;
        }
            $msg = "You have visited this page ".
            $_SESSION['counter'];
  $msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
<?php  echo ( $msg ); ?>
</body>
</html>
```

Destroying a PHP Session

- A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.
- Here is the example to unset a single variable:

```
<?php
        unset($_SESSION['counter']);
?>
```

- Here is the call which will destroy all the session variables:

```
<?php
  session_destroy();
?>
```

Types of Errors in PHP

- An error is a type of mistake. We can say an error is a condition of having incorrect or false knowledge or an error is defined as an unexpected, invalid program state from which it is impossible to recover.
- Error can also be defined as "a deviation from accuracy or correctness". A "mistake" is an error caused by a fault: the fault being misjudgment, carelessness, or forgetfulness. An error message with filename, line number and a message describing the error is sent to the browser.

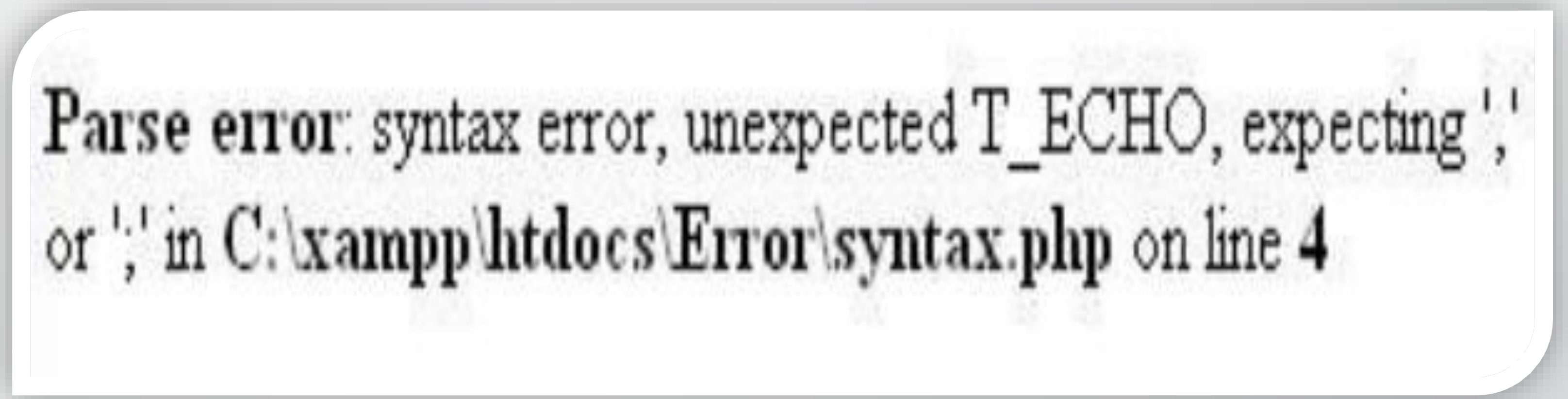
- Basically there are four types of errors in PHP, which are as follows:
    - ✓ Parse Error (Syntax Error)
    - ✓ Fatal Error
    - ✓ Warning Error
    - ✓ Notice Error

1.Parse Errors (syntax errors)

- The parse error occurs if there is a syntax mistake in the script; the output is Parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons for parse errors are as follows:
  Common reason of syntax errors are:
- Unclosed quotes
- Missing or Extra parentheses
- Unclosed braces
- Missing semicolon

- **Example**

```
<?php
echo "Cat";
echo "Dog"
echo "Lion";
?>
```



```
Parse error: syntax error, unexpected T_ECHO, expecting ',' or ';' in C:\xampp\htdocs\Error\syntax.php on line 4
```

- **Output**

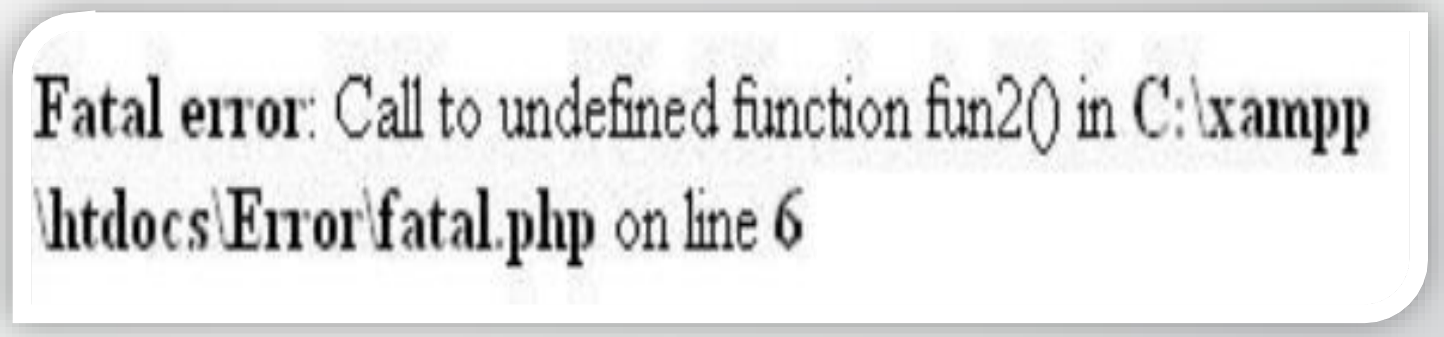
  In the above code we missed the semicolon in the second line. When that happens there will be a parse or syntax error which stops execution of the script, as in the above image:

2. Fatal Errors

- Fatal errors are caused when PHP understands what you've written, however what you're asking it to do can't be done. Fatal errors stop the execution of the script. If you are trying to access the undefined functions, then the output is a fatal error.

- **Example**

```
<?php
function fun1()
{
echo "Vineet Saini";
}
fun2();
echo "Fatal Error !!";
?>
```



```
Fatal error: Call to undefined function fun2() in C:\xampp
\htdocs\Error\fatal.php on line 6
```

Output

In the above code we defined a function fun1 but we call another function fun2 i.e. func2 is not defined. So a fatal error will be produced that stops the execution of the script. Like as in the above image.

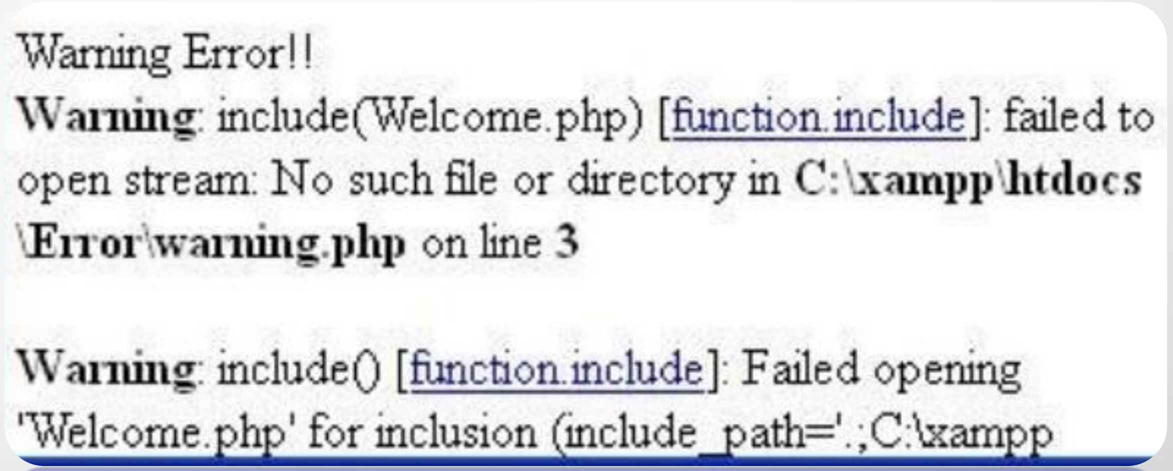
3. Warning Errors

Warning errors will not stop execution of the script. The main reason for warning errors are to include a missing file or using the incorrect number of parameters in a function.

Example

```
<?php
echo "Warning Error!!";
include ("Welcome.php");
?>
```

Output



```
Warning Error!!
Warning: include(Welcome.php) [function.include]: failed to
open stream: No such file or directory in C:\xampp\htdocs
\Error\warning.php on line 3

Warning: include() [function.include]: Failed opening
'Welcome.php' for inclusion (include_path='.;C:\xampp
```

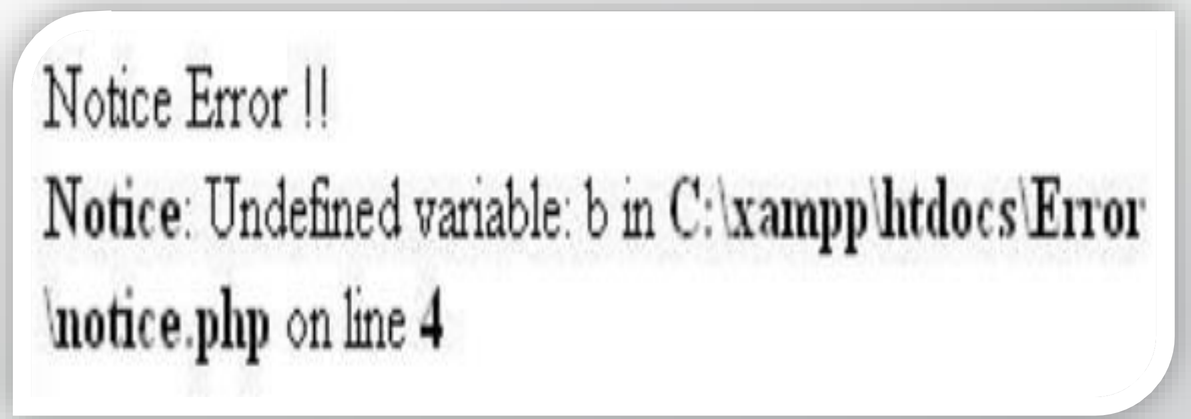
In the above code we include a welcome.php file, however the welcome.php file does not exist in the directory. So there will be a warning error produced but that does not stop the execution of the script i.e. you will see a message Warning Error !!. Like in the above image:

4. Notice Errors

- Notice that an error is the same as a warning error i.e. in the notice error execution of the script does not stop. Notice that the error occurs when you try to access the undefined variable, then produce a notice error.

  Example

```
<?php
$a="Vineet kumar saini";
echo "Notice Error !!";
echo $b;
?>
```



```
Notice Error !!
Notice: Undefined variable: b in C:\xampp\htdocs\Error
\notice.php on line 4
```

- **Output**

  In the above code we defined a variable which named $a. But we call another variable i.e. $b, which is not defined. So there will be a notice error produced but execution of the script does not stop, you will see a message Notice Error !!. Like in the above image: