

UNIT – 1

PHP BASICS



**UNIVERSITY
OF PETROLEUM
& ENERGY STUDIES**

Dr. P S V S Sridhar
Assistant Professor (SS)
Centre for Information Technology

1. Introduction to PHP, Support for Database

What is PHP?

- ▶ PHP stands for **PHP: Hypertext Preprocessor**
- ▶ PHP is a server-side scripting language, like ASP
- ▶ Popular server-side scripting languages - ASP, ColdFusion, JSP, Perl, CGI, Python, Ruby, Haskell etc.,
- ▶ PHP scripts are executed on the server
- ▶ PHP allows web developers to write dynamically generated pages quickly
- ▶ PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- ▶ PHP is an open source scripting language for dynamically generate web page content.
- ▶ PHP is free to download and use
- ▶ PHP is Embedded code

Personal Home Page (PHP)

- Development started in 1994, by Rasmus Lerdorf.
- A project of the Apache Software Foundation
- <http://www.php.net>
- Open Source, free server-side HTML-embedded scripting language
- Cross-platform, suitable for today's heterogeneous network environments
- PHP's syntax resembles C and Perl and is easy to learn for any programmer with C or Perl background
- Simpler and faster to develop in (than C and Perl)

Server Side Scripting

- Server Side scripting is invisible to the user.
- Server side web scripting is mostly connecting web sites to backend servers, processing data, controlling the behaviour.
- Server side scripting products consist of two main parts:
 - ▶ Scripting language
 - ▶ Scripting engine(parses and interprets pages written in the language)
- Eg. E-mail: Client side – open mail using client software
- Server side – User fill out a form, e-mail is sent via Simple Mail Transfer Protocol (SMTP).
- Server side methods are a bit slower at run time.

Server Side Scripting applications

- Content Sites
- Community features (forums, bulletin boards etc)
- Email
- Customer support and technical support systems
- Advertising networks
- Surveys, Polls and tests
- Filling out submitting forms online
- Games
- Any other applications that needs to connect back end server(database, LDAP – Lightweight Directory Access Protocol)

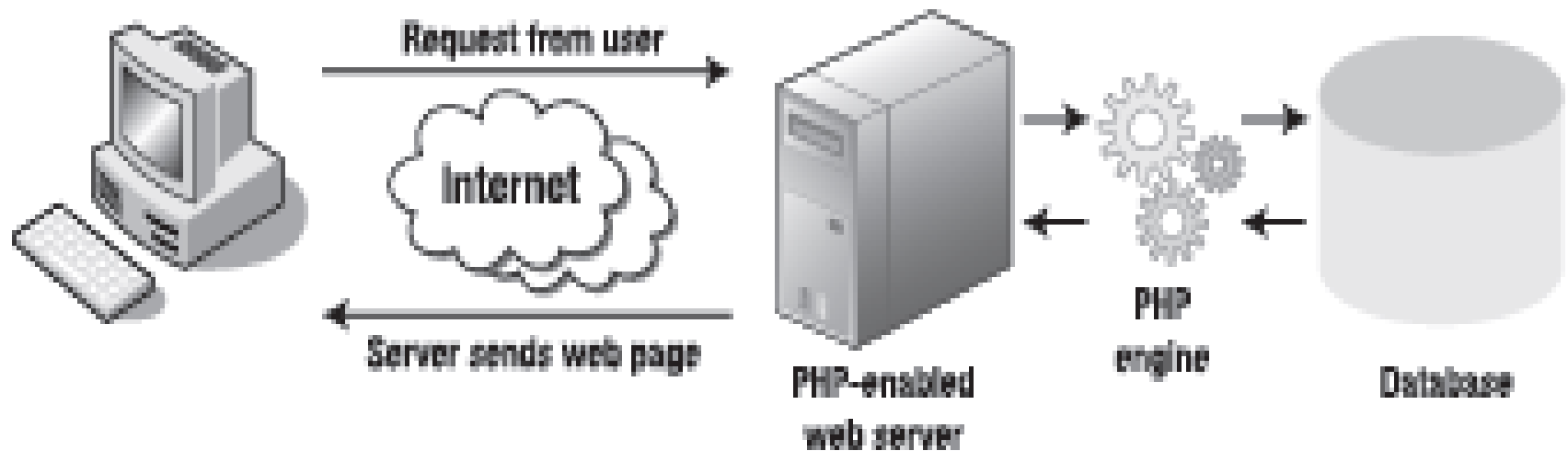
History

- PHP (PHP: Hypertext Preprocessor) was created by Rasmus Lerdorf in 1994. It was initially developed for HTTP usage logging and server-side form generation in Unix.
- PHP 2 (1995) transformed the language into a Server-side embedded scripting language. Added database support, file uploads, variables, arrays, recursive functions, conditionals, iteration, regular expressions, etc.
- PHP 3 (1998) added support for ODBC data sources, multiple platform support, email protocols (SNMP – Simple Network Management Protocol, IMAP – Internet Message Access Protocol), and new parser written by Zeev Suraski and Andi Gutmans .
- PHP 4 (2000) became an independent component of the web server for added efficiency. The parser was renamed the Zend Engine. Many security features were added.
- PHP 5 (2004) adds Zend Engine II with object oriented programming, robust XML support
- PHP grew in popularity, but did not become widely known until two program developers named Zeev Suraski and Andi Gutmans, developed a new parser in the summer of 1997, which led to the development of PHP 3.0.
- The newest version out is PHP 5, which uses an engine developed by Suraski and Gutmans, known as the Zend II Engine. (Zend I was used by PHP 4)

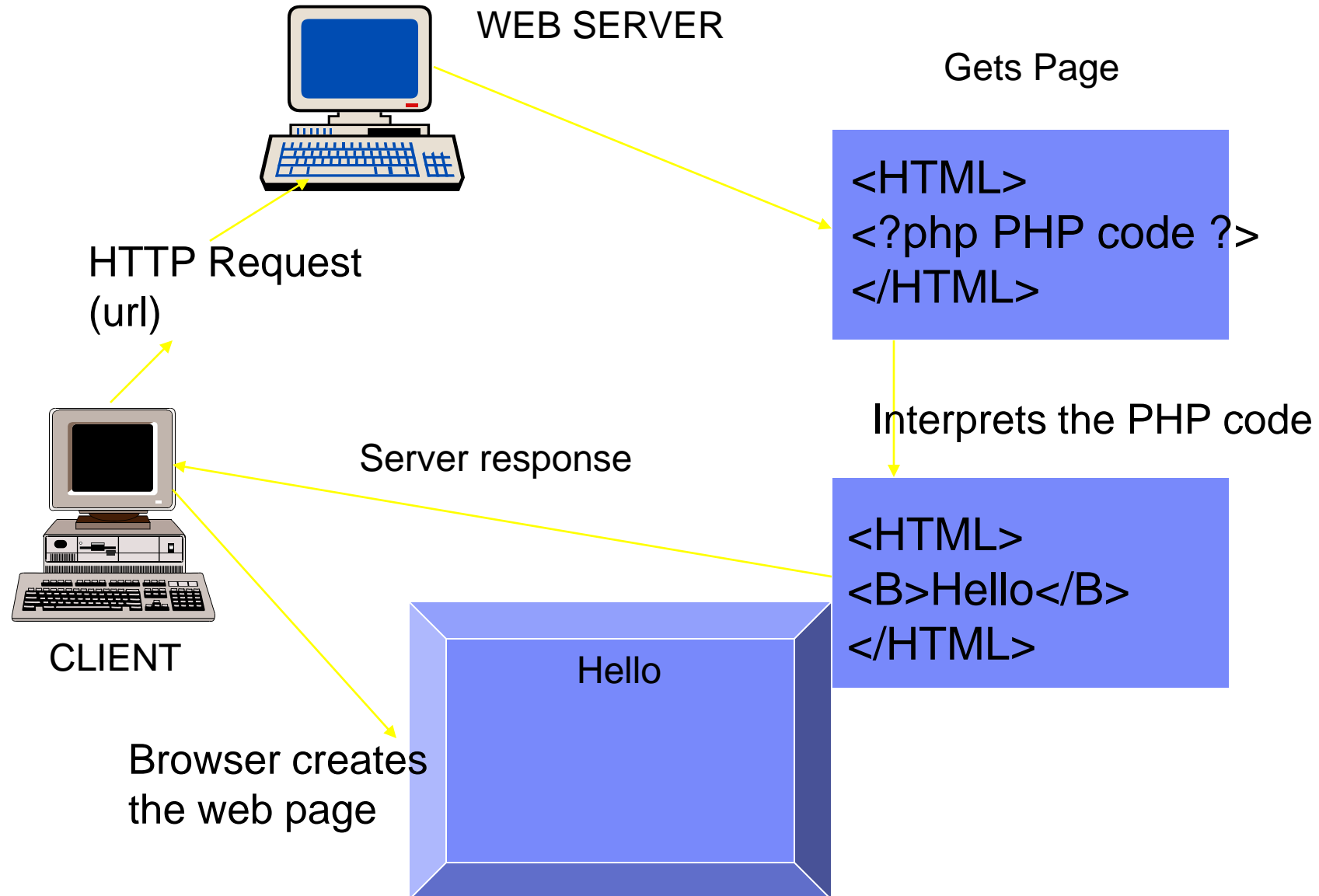
2. PHP Installation, Working with PHP, Why PHP?

What do you Need?

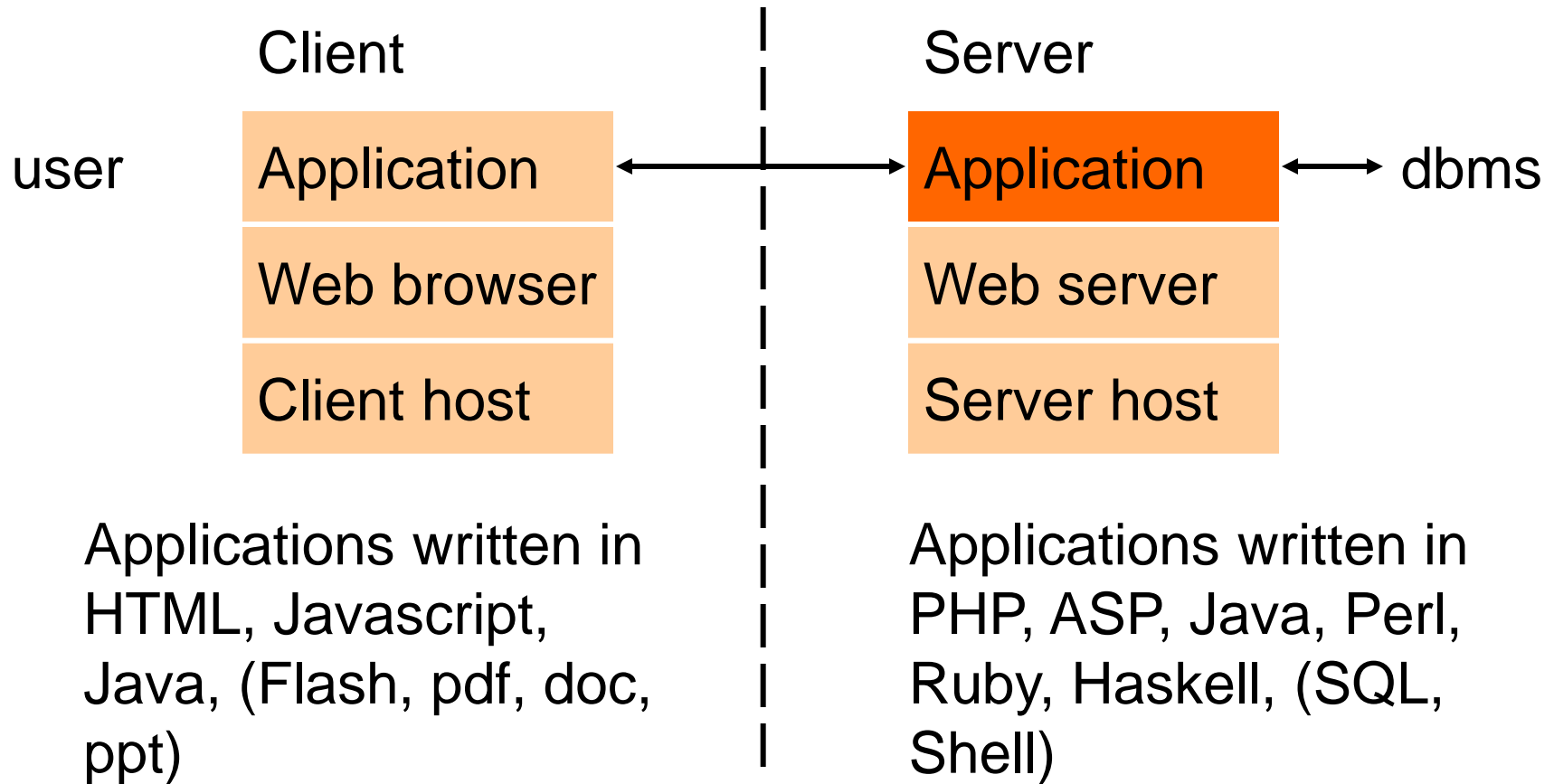
- If your server supports PHP you don't need to do anything.
- Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.
- However, if your server does not support PHP, you must install PHP.
 - ▶ **Download PHP**
 - ▶ **Download MySQL Database**
 - ▶ **Download Apache Server**



- XAMPP installs Apache, PHP, MySQL, PhpMyAdmin, and several other tools on your computer in a single operation.



PHP's role in web applications



PHP vs. JSP

- A recent survey in ZDnet's *eWeek* online publication found that PHP is as much as 3.5 times faster than JSP
- Faster in development time – flatter learning curve
- PHP supports any 32-bit or better platform, whereas JSP supports only platforms that have a Java virtual machine available

Operating Systems & PHP

Operating systems

- Linux - **LAMP**, (Linux, **A**pache, **M**ySQL and **P**HP)
- Microsoft Windows – WAMP (Windows Apache MySQL and Perl),
- Mac OS - MAMP,
- Solaris - SAMP,
- OpenBSD - OAMP
- XAMPP - is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.
- X (to be read as "cross", meaning cross-platform), Apache HTTP Server, MySQL, PHP, Perl

- Linux (operating system), Apache HTTP Server, MySQL (database software) and PHP.

PHP File

- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.
- PHP language features such as control structures, operators, variable types, function declaration, class/object declaration are almost similar to any compiled or interpreted language such as C or C++.

PHP Installation

- Install an Apache server on a Windows or Linux machine
- Install PHP on a Windows or Linux machine
- Install MySQL on a Windows or Linux machine

Execution of PHP files

1. Install exe file
2. Click Xampp control panel.
3. Start service Apache and MySQL in Xampp control panel.
4. Verify the Xampp server from browser by typing <http://localhost>
5. Remove all the files under C:\Xampp\htdocs
6. Create a PHP file with extension “.php” under htdocs folder
7. Run PHP file with browser <http://localhost/filename.php>

Why is PHP used?

1. Easy to Use

Code is embedded into HTML. The PHP code is enclosed in special start and end tags that allow you to jump into and out of "PHP mode".

2. Cross Platform

Runs on almost any Web server on several operating systems.

One of the strongest features is the wide range of supported databases

Web Servers: Apache, Microsoft IIS, Caudium, Netscape Enterprise Server

Operating Systems: UNIX (HP-UX, OpenBSD, Solaris, Linux), Mac OSX, Windows NT/98/2000/XP/2003

Supported Databases: Adabas D, dBase, Empress, FilePro (read-only), Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL, SQLite, Solid, Sybase, Velocis, Unix dbm

Why is PHP used?

3. Cost Benefits

PHP is free. Open source code means that the entire PHP community will contribute towards bug fixes. There are several add-on technologies (libraries) for PHP that are also free.

	PHP
Software	Free
Platform	Free (Linux)
Development Tools	Free <u>PHP Coder</u> , <u>jEdit</u>

3. Basic Syntax of PHP, PHP statement terminator and case insensitivity, Embedding PHP in HTML

Basic PHP Syntax

- A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.
- For maximum compatibility, it is recommend that you use the standard form (<?php) rather than the shorthand form.
- Shorthand PHP tag starts with **<?** and ends with **?>**.
- Each line of PHP is terminated with a semi-colon.

```
<html>
<body>

<?php          echo "Hello World";
?>

</body>
</html>
```

There are four different ways to embed the PHP code

`<?php echo("Some PHP code"); ?>`

`<? echo("Some PHP code"); ?>`

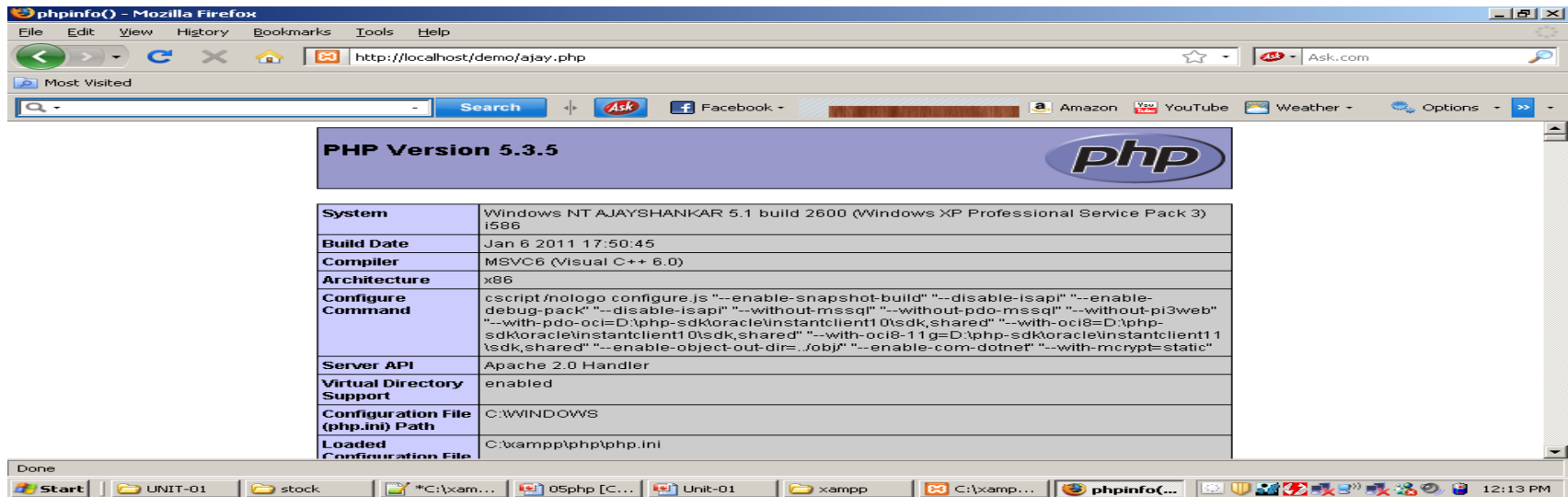
`<SCRIPT Language='php'> echo("Some PHP code"); </SCRIPT>`

`<% echo("Some PHP code"); %>`

phpinfo()

- The phpinfo() function shows the php environment
- Use this to read system and server variables, setting stored in php.ini, versions, and modules

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



PHP Version 5.3.5	
System	Windows NT AJAYSHANKAR 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Jan 6 2011 17:50:45
Compiler	MSVC6 (Visual C++ 6.0)
Architecture	x86
Configure Command	cscrip/nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--disable-isapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=D:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=.\obj" "--enable-com-dotnet" "--with-mcrypt=static"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\xampp\php\php.ini

How to save PHP pages

- If you have PHP inserted into your HTML and want the web browser to interpret it correctly, then you must save the file with a *.php extension, instead of the standard .html extension. So be sure to check that you are saving your files correctly. Instead of index.html, it should be index.php if there is PHP code in the file.*

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Display:

```
Hello World!
```

Semicolon;

The **semicolon** represents the end of a PHP statement

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
echo "Hello World! ";
?>
</body>
</html>
```

Display:

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Case Sensitive

PHP is a case sensitive language.

Example

```
<?php  
$mynameis = "Chris Oak";  
echo $Mynameis;  
?>
```

Warning: Undefined variable: Mynameis in D:\samplecode.php on line 3

echo and print

- There are two basic statements for output text with PHP: **echo()** and **print()**. **In the example** we have used the echo statement for output text "Hello World".
- They can be used with or without brackets.

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

```
<?php  
echo 'Hello World!<br />';  
echo('Hello World!<br />');  
print 'Hello World!<br />';  
print('Hello World!<br />');  
?>
```


Escaping Characters

- Some characters are considered 'special'
- Escape these with a backslash \
- Special characters will be flagged when they arise, for example a double or single quote belong in this group...

```
<?php
// Claire O'Reilly said "Hello".
echo 'Claire O\'Reilly ';
echo "said \"Hello\".";
?>
```

White space

- Whitespace is ignored between PHP statements

PHP and HTML Code:

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World!";
    echo "Hello World!";
?>
</body>
</html>
```

Display:

```
Hello World!Hello World!
```

4. Comments, Variables, Assigning value to a variable

Comments in PHP

- In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<body>

<?php
    //This is a comment

    /*
    This is
    a comment
    block
    */

?>

</body>
</html>
```

Variables in PHP

- Variables are used for storing a values, like text strings, numbers or arrays.
- Variable do not need to be declared before assignment.
- When a variable is declared, it can be used over and over again in your script.
- All variables in PHP start with a \$ sign symbol.
- The correct way of declaring a variable in PHP.
- A variable can then be reused throughout your code, instead of having to type out the actual value over and over again.

```
<?php
    $txt="Hello World!";
    $x=16;
?>
```

- ▶ \$txt is a variable containing a string
- ▶ \$x is a variable containing a number

Variable Naming Conventions

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore (`$my_string`), or with capitalization (`$myString`)
- Variables with more than one word can also be distinguished with capitalization. `$myVariable`
- Note: Variable names are case-sensitive, so use the exact same capitalization when using a variable. The variables `$a_number` and `$A_number` are different variables in PHP's eyes.

PHP is a Loosely Typed Language

Let's try creating a variable with a string, and a variable with a number:

```
<?php  
$txt = "Hello World!";  
$number = 16;  
?>
```

- In PHP a variable does not need to be declared before being set.
- In the example above, you see that you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on how they are set.
- In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.
- In PHP the variable is declared automatically when you use it.

Variable Variables

- Using the value of a variable as the **name** of a second variable)

```
$a = "hello";  
$$a = "world";
```

- Thus:

```
echo "$a ${$a}";
```

- Is the same as:

```
echo "$a $hello";           //hello world
```

- But `$$a` echoes as `"$hello"....`

world

String Variables in PHP

- String variables are used for values that contains characters.
- After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.
- Below, the PHP script assigns the text "Hello World" to a string variable called \$txt:

```
<?php  
    $txt="Hello World";  
    echo $txt;  
?>
```

- The output of the code will be:

Hello World

The Concatenation Operator

- There is only one string operator in PHP.
- The concatenation operator (.) is used to put two string values together.
- To concatenate two string variables together, use the concatenation operator:

```
<?php
    $txt1="Hello World!";
    $txt2="What a nice day!";
    echo $txt1 . " " . $txt2
?>
```

- The output of the code above will be:

Hello World! What a nice day!

PHP Code:

```
<?php
$my_string = "Hello Bob.  My name is: ";
$newline = "<br />";
echo $my_string."Bobettta".$newline;
echo "Hi, I'm Bob.  Who are you? ".$my_string.$newline;
echo "Hi, I'm Bob.  Who are you? ".$my_string."Bobetta";
?>
```

- You can also combine text strings and variables. By doing such a conjunction you save yourself from having to do a large number of echo statements. Variables and text strings are joined together with a period(.). The example below shows how to do such a combination.

The strlen() function

- The strlen() function is used to return the length of a string.
- Let's find the length of a string:

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

- The output of the code above will be:

12

“ (double quote) or ‘ (Single quote)

- There is a difference between strings written in single and double quotes.
- In a double-quoted string any variable names are expanded to their values.
- In a single-quoted string, no variable expansion takes place.

```
<?php
$name = 'Phil';
$age = 23;
echo "$name is $age";
// Phil is 23
?>
```

```
<?php
$name = 'Phil';
$age = 23;
echo ' $name is $age';
// $name is $age
?>
```

The strpos() function

- The strpos() function is used to search for character within a string.
- If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.
- Let's see if we can find the string "world" in our string:

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

- The output of the code above will be:
- Starting position starts with 0

6

5. Constants

- Values that never changes
- A constant is a name or an identifier for a simple value. In constants value cannot change during the execution of the script
- Constants are defined in PHP by using the define() function.
- No preceding dollar is applied.
- By convention, constant names are usually in UPPERCASE.

Syntax	constant(constant)
constant	Required. Specifies the name of the constant to check

```
<?php
//define a constant
define("GREETING","Hello you! How are you today?");

echo constant("GREETING");           or echo GREETING;
?>
```

The output of the code above will be:
Hello you! How are you today?

Data Types in PHP

- PHP type system is simple.
- No variable type declarations
 - ▶ `$first_number = 23;`
 - ▶ `$second_number = "Second number";`
- Automatic type conversion
 - `$pi = 3+0.14159`
- Types assigned by context
 - Automatic type conversion
 - `$sub = substr(12345,2,2);`
 - `print("sub is $sub
");` sub is 34

Data Type summary

- Total 8 types
- Integers
- Doubles
- Boolean
- NULL (only one value NULL)
- String
- Arrays
- Object
- Resource (holds references to resources external to PHP such as database connections)

Simple Types

■ Integers

- ▶ +ve, -ve numbers `$int_var = 12345;`
- ▶ Expressions `$ano_int = -12345 + 12345;`
- ▶ Integers can be in three formats: decimal, octal, hexdec.
- ▶ Decimal format is default, octal integers leading with 0, hexadecimal leading with 0x, format preceded by - means integer -ve.
- ▶ 1000(dec), 01000(oct), 0x1000(hexa dec)
- ▶ Range: like C long type $-2^{31} - 1$ (-2,147,483,647) to $2^{31} - 1$ (2,147,483,647)

■ Doubles

- Doubles are floating point numbers.

- **Booleans**

- True or false values
- If (TRUE) print (“yes”) else print (“No”);
- If value is number , it is false if the number is zero otherwise true
- If value is string, it false if the string is empty

- **NULL**

- Value is NULL.

- **Strings**

- Character sequence
 - Escape ccharacters - \n new line, \r carriage return, \t tab, \\$ dollor sign, \" double quotation, \\ backslash

Thanks