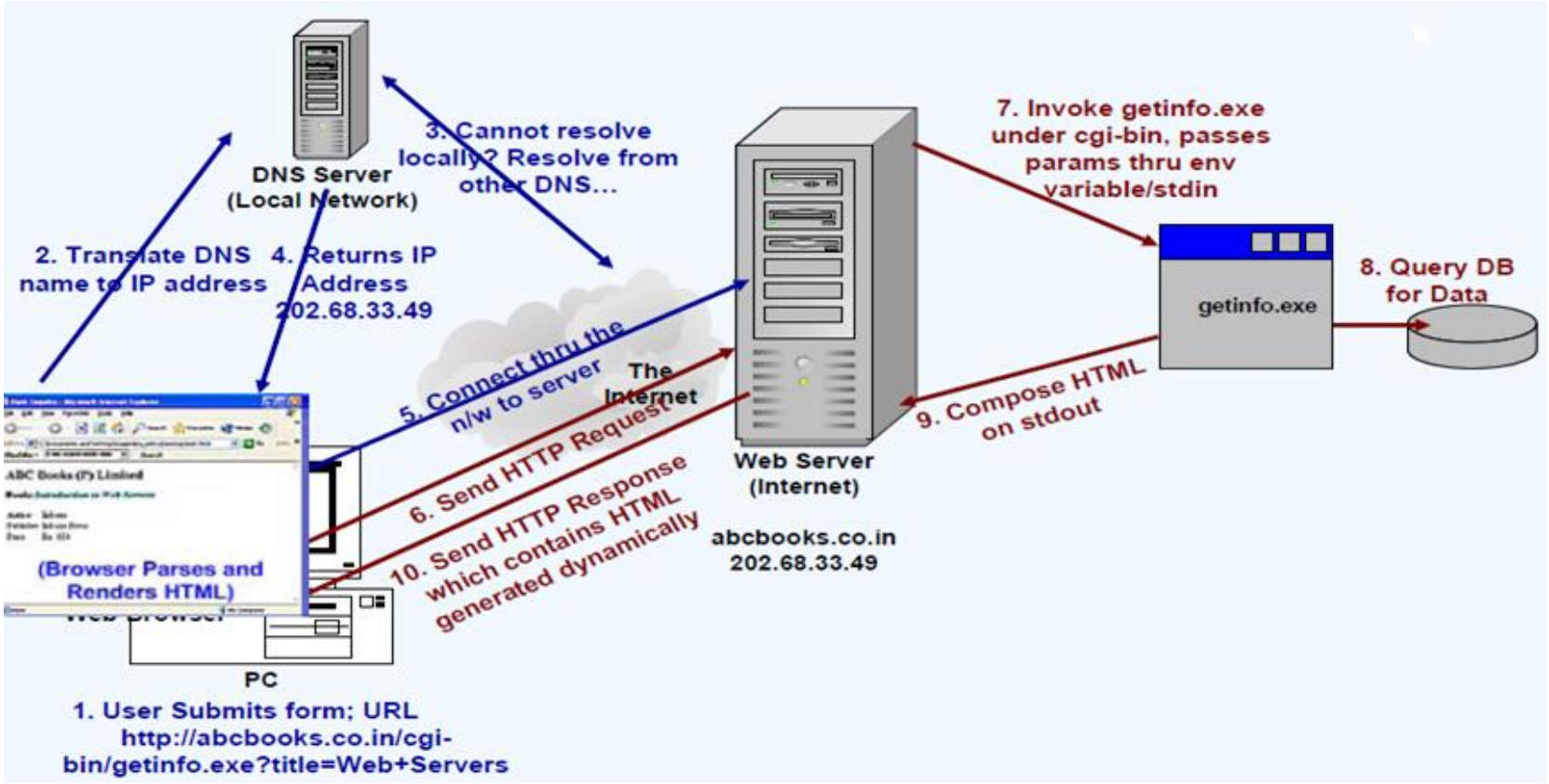
Introduction to PHP

Dynamic Server-side web application



XAMPP

- **XAMPP**
  - is a free and open source cross-platform web server solution stack package
  - an acronym for X (any of four different operating systems)
    - Microsoft Windows, Linux, Sun Solaris and Mac OS X
    - Apache, MySQL, PHP and Perl.
  - released under the GNU General Public License
  - Cross-platform (Linux, Windows, Solaris,Mac OS X)
  - Consisting of
    - the Apache HTTP Server
    - MySQL database
    - interpreters for scripts written in PHP and Perl
- Useful links
  - https://www.apachefriends.org/index.html
  - http://www.wikihow.com/Set-up-a-Personal-Web-Server-with-XAMPP

Apache Tomcat - introduction

- world's most widely used web server software
- Support Perl, Python, Tcl, and PHP
- Secure Sockets Layer and Transport Layer Security support
- Virtual hosting allows one Apache installation to serve many different Web sites
- Useful links
  - https://www.apachefriends.org/index.html
  - http://www.wikihow.com/Set-up-a-Personal-Web-Server-with-XAMPP

PHP Basics - Introduction

- PHP
  - Is a development from Personal Home Page Tools started by Rasmus Lerdorf in 1994
  - Mixture of Perl, C and Java
  - Apache module which integrates PHP into Web Server is the most popular
  - Is a web specific tool
  - Free software & has been ported to many OS
  - Supports standard network protocol – IMAP, NNTP, SMTP, POP3 & HTTP
  - Can work with wide range of Database Systems
  - Ability to process XML data and RSS feeds

PHP Basics – What Can PHP Do?

- PHP can
  - generate dynamic page content
  - create, open, read, write, delete, and close files on the server
  - collect form data
  - send and receive cookies
  - Create and manage user session
  - add, delete, modify data in your database
  - restrict users to access some pages on your website
  - encrypt data

PHP Basics - Echo & Print

- Comments represented as /* ..*/ , single line - //
- Variable names are case-sensitive
- echo - can output one or more strings
- print - can only output one string, and returns 1 always
- Example :
  - echo "This", " string", " was", " made", " with multiple parameters.";
  - print "<h2>PHP is fun!</h2>";

PHP Basics - PHP Variables

- Loosely Type Language
- PHP has three different variable scopes:
  - local
  - global
  - static
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case sensitive ($y and $Y are two different variables)

Variable scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function
- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function
- The **global** keyword is used to access a global variable from within a function
- PHP also stores all global variables in an array called **$GLOBALS[index].**
  - The index holds the name of the variable

Example

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

        <?php
        $fname = "ABC<br/>";
        $lname = "XYZ<br/>";

        echo $fname , $lname;
        //print $fname , $lname;
        $rvalue=print $fname;
        echo $rvalue;
        ?>
</body>
</html>
```

Example

```
<?php
        $x = 5;
        $y = 4;
        echo $x + $y;
        ?>
        <?php
                $x = 6;
                $w = 15;

                function myTest() {
                 global $x, $w;
                $y = 10;
                $z= $x + $y + $w;
                        echo "<p>Addition of three no:
<br/> $x + $y + $w = $z</p>";
                        $x=20;
                }
                myTest();

echo "<p>Variable x outside function is: $x</p>";

function myTest1() {

+ $GLOBALS['w'];



$w</p>";

                        $GLOBALS['w'] = $GLOBALS['x']
}
myTest1();
echo "<p>Addition of two number is:

function myTest2() {
                static $m = 0;
                echo "$m <br/>";
                $m++;
}
myTest2();
myTest2();
myTest2();
myTest2();

        ?>
```

PHP Basics - Data Types

- A string can be any text inside quotes. use single or double quotes
- Rules for integers:
  - An integer must have at least one digit (0-9)
  - An integer cannot contain comma or blanks
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- A floating point number is a number with a decimal point or a number in exponential form
- Booleans can be either TRUE or FALSE
- An array stores multiple values in one single variable.
- The special NULL value represents that a variable has no value. NULL is the only possible value of data type NULL

PHP operators

- Arithmetic operators: +, -, *, /, %, **
- Assignment operators: =, +=, -=, /=, %=
- Comparison operators: ==, !=, ===, !==, >, <, >=, <=
- Increment/Decrement operators: ++$x, $x++, --$x, $x--
- Logical operators: and, or, xor, &&, ||, !
- String operators: ., .=

Conditional statements

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif....else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

Loops

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

PHP Basics – Arrays

1. **Indexed arrays** - Arrays with numeric index

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP Basics – Arrays

Associative arrays - Arrays with named keys

- To create an associative array:
  - $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
  - $age['Peter']="35"; $age['Ben']="37"; $age['Joe']="43";
- **Example : To display contents of array**
  - <?php
    $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
    foreach($age as $x=>$x_value) {
    echo "Key=" . $x . ", Value=" . $x_value; echo "
";
    } ?>
  - <?php
    $a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));
    print_r ($a);
    ?>

3. **Multidimensional arrays** - Arrays containing one or more arrays

```
$students = array(
    "Roll no.: 1" => array(
        "name" => "Sunitha",
        "email" => "Sunitha@mail.com",
    ),
    "Roll no.: 2" => array(
        "name" => "Clark",
        "email" => "clark@mail.com",
    ),
    "Roll no.: 3" => array(
        "name" => "Harish",
        "email" => "harish@mail.com",
    )
);
```

Array object methods

- array_push($arr, $val)- pushes value to a array
- array_pop($arr)- removes an element from the end of an array and returns its value
- array_reverse($arr)- reverses the order of the elements in an array.
- array_flip($arr) - interchanges the keys and the values of an Associative array
- array_unique($arr) - remove all duplicate entries in the array.
- array_keys($arr)- recover the keys from an associative array.
- array_values($arr)- receives a PHP array.
- sort($arr) - Sorts scalar array in ascending order.
- rsort($arr) - Sorts scalar array in reverse order.
- asort($arr) - Sorts associative array by values.
- arsort($arr) - Sorts associative array by values in reverse order.
- ksort($arr) - Sorts associative array by ‘Keys’.
- krsort($arr) - Sorts associative array by ‘Keys’ in reverse order

PHP Basics - String functions

- **strlen()** – to return the length of the string
- **trim()** - removes any blank characters from the beginning and end of the string
- **strtolower()** - String to Lower converts any uppercase letters to lowercase
- **strtoupper()** - String to Upper converts any lowercase letters to uppercase
- **htmlspecialchars()** - takes a string and converts &, <, >, and double quotes to proper HTML entities
  - <?php
    $str = "This is some bold text   space before";
    echo htmlspecialchars($str);
    ?>

PHP Super globals

- Several predefined variables in PHP are "superglobals“
- Are always accessible
- The PHP superglobal variables are:
  - $GLOBALS – access to all global variables
  - $_SERVER - holds information about headers, paths, and script locations.
  - $_REQUEST - used to collect data after submitting an HTML form
  - $_POST - used to collect form data after submitting an HTML form with method="post"
  - $_GET - used to collect form data after submitting an HTML form with method="get"
  - $_FILES - array you can upload files from a client computer to the remote server
  - $_ENV - array containing environment variables
  - $_COOKIE – array containing all cookies
  - $_SESSION – array containing session variables.

PHP Server super global

$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server

$_SERVER

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

$_REQUEST

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
            echo "Name is empty";
    } else {
            echo $name; } }
?>
</body>
</html>
```

$_GET

```
<html> <body>
<form method="get" action="gettdest.php">
  Name: <input type="text" name="fname">
  <input type="submit">
</form> </body> </html>
```

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
   // collect value of input field
   $name = $_GET['fname'];
   if (empty($name)) {
      echo "Name is empty";
   } else {
      echo $name;
   }
}
?>
```

$_POST

```
<html>
<body>

<form method="post" action="postdest.php">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

</body>
</html>
```

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = $_POST['fname'];
   if (empty($name)) {
      echo "Name is empty";
   } else {
      echo $name;
   }
}
?>
```