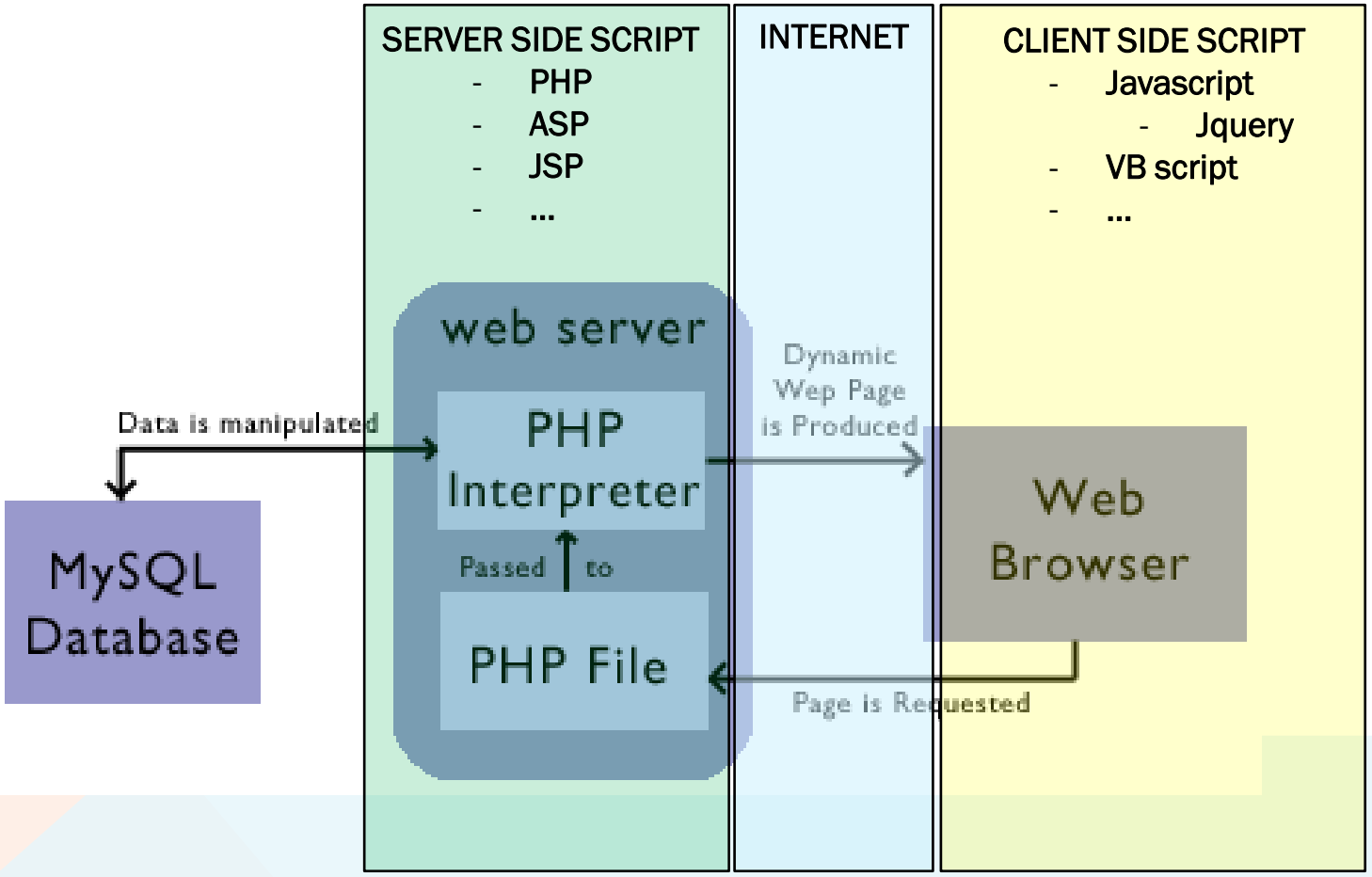
PHP

PHP: HYPERTEXT PREPROCESSOR

PIYA LIMCHAROEN

INDEX

PHP INTRODUCTION



PHP INTRODUCTION

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML/XHTML
- JavaScript

What is PHP?

- PHP stands for PHP: Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

PHP INTRODUCTION

What is a PHP File?

- PHP files can 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"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP INTRODUCTION

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- 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

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support
- Install XAMPP (XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl.)
- Install MAMP ("MAMP" stands for: Macintosh, Apache)
  - Apache, PHP and MySQL for Mac OS X

PHP INSTALLATION

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.

- Here is a link to a good tutorial from PHP.net on how to install PHP5:
  - http://www.php.net/manual/en/install.php

- Download
  - Download PHP for free here: http://www.php.net/downloads.php
  - Download MySQL for free here: http://www.mysql.com/downloads/
  - Download Apache for free here: http://httpd.apache.org/download.cgi

PHP SYNTAX

Basic PHP Syntax

- A PHP script always starts with <?php and ends with ?>.
- A PHP script can be placed anywhere in the document.
- On servers with shorthand-support, you can start a PHP script with <? and end with ?>.
- For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.
- A PHP file must have a .php extension.
- A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP script that sends the text "Hello World" back to the browser:

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

In the example above we have used the echo statement to output the text "Hello World".

PHP SYNTAX

Comments in PHP

- Each code line in PHP must end with a semicolon.
- The semicolon is a separator and is used to distinguish one set of instructions from another.
- There are two basic statements to output text with PHP: echo and print.

In PHP, we use // to make a one-line comment or /* and */ to make a comment block:

```
<html>
<body>
<?php
//This is a comment

/*
This is a comment block
*/
?>
</body>
</html>
```

PHP VARIABLES

A variable can have a short name, like x, or a more descriptive name, like carName.

Rules for PHP variable names:

- Variables in PHP starts with a $ sign, followed by the name of the variable
- The variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive (y and Y are two different variables)

PHP has no command for declaring a variable.

```
$myCar="Volvo";
```

- After the execution of the statement above, the variable myCar will hold the value Volvo.
- Tip: If you want to create a variable without assigning it a value, then you assign it the value of null.

Let's create a variable containing a string, and a variable containing a number:

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

PHP VARIABLES

The scope of a variable is the portion of the script in which the variable can be referenced.

PHP has four different variable scopes:

- local
- global
- static
- Parameter

PHP VARIABLES - LOCAL SCOPE

- A variable declared within a PHP function is local and can only be accessed within that function. (the variable has local scope):

```
<?php
$a = 5; // global scope

function myTest()
{
echo $a; // local scope
}

myTest();
?>
```

- The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.

- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

- Local variables are deleted as soon as the function is completed.

PHP VARIABLES - GLOBAL SCOPE

- Global scope refers to any variable that is defined outside of any function.
- Global variables can be accessed from any part of the script that is not inside a function.
- To access a global variable from within a function, use the global keyword:

```
<?php
$a = 5;
$b = 10;

function myTest()
{
global $a, $b;
$b = $a + $b;
}

myTest();
echo $b;
?>
```

```
<?php
$a = 5;
$b = 10;

function myTest()
{
$GLOBALS['b'] = $GLOBALS['a'] +
$GLOBALS['b'];
}

myTest();
echo $b;
?>
```

- The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- Local variables are deleted as soon as the function is completed.

PHP VARIABLES - STATIC SCOPE

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
- To do this, use the static keyword when you first declare the variable:

```
static $rememberMe;
```

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
- Note: The variable is still local to the function.

PHP VARIABLES - PARAMETERS

- A parameter is a local variable whose value is passed to the function by the calling code.
- Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...)
{
// function code
}
```

- Parameters are also called arguments. We will discuss them in more detail when we talk about functions.

PHP STRING VARIABLES

- 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!

- If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP STRING VARIABLES

- 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

- The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

PHP STRING VARIABLES

- The strpos() function is used to search for a character/text within a string.
- If a match is found, this function will return the character 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:

6

- The position of the string "world" in the example above is 6. The reason that it is 6 (and not 7), is that the first character position in the string is 0, and not 1.

PHP OPERATORS - ARITHMETIC OPERATORS

The table below lists the arithmetic operators in PHP:

Operator	Name	Description	Example	Result
x + y	Addition	Sum of x and y	2 + 2	4
x - y	Subtraction	Difference of x and y	5 - 2	3
x * y	Multiplication	Product of x and y	5 * 2	10
x / y	Division	Quotient of x and y	15 / 5	3
x % y	Modulus	Remainder of x divided by y	5 % 2 10 % 8 10 % 2	1 2 0
- x	Negation	Opposite of x	- 2	
a . b	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

PHP OPERATORS - ASSIGNMENT OPERATORS

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the expression on the right. That is, the value of "$x = 5" is 5.

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
a .= b	a = a . b	Concatenate two strings

PHP OPERATORS - INCREMENTING/DECREMENTING 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

PHP OPERATORS - COMPARISON OPERATORS

Comparison operators allows you to compare two values:

Operator	Name	Description	Example
x == y	Equal	True if x is equal to y	5==8 returns false
x === y	Identical	True if x is equal to y, and they are of same type	5==="5" returns false
x != y	Not equal	True if x is not equal to y	5!=8 returns true
x <> y	Not equal	True if x is not equal to y	5<>8 returns true
x !== y	Not identical	True if x is not equal to y, or they are not of same type	5!=="5" returns true
x > y	Greater than	True if x is greater than y	5>8 returns false
x < y	Less than	True if x is less than y	5<8 returns true
x >= y	Greater than or equal to	True if x is greater than or equal to y	5>=8 returns false
x <= y	Less than or equal to	True if x is less than or equal to y	5<=8 returns true

PHP OPERATORS - LOGICAL OPERATORS

Operator	Name	Description	Example
x and y	And	True if both x and y are true	x=6 y=3 (x < 10 and y > 1) returns true
x or y	Or	True if either or both x and y are true	x=6 y=3 (x==6 or y==5) returns true
x xor y	Xor	True if either x or y is true, but not both	x=6 y=3 (x==6 xor y==3) returns false
x && y	And	True if both x and y are true	x=6 y=3 (x < 10 && y > 1) returns true
x || y	Or	True if either or both x and y are true	x=6 y=3 (x==5 || y==5) returns false
! x	Not	True if x is not true	x=6 y=3 !(x==y) returns true

PHP OPERATORS - ARRAY OPERATORS

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

CONDITIONAL STATEMENTS - PHP IF...ELSE STATEMENTS

- The if Statement
  - Use the if statement to execute some code only if a specified condition is true.

- Syntax
  - if (*condition*) *code to be executed if condition is true;*
  - The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

- Notice that there is no ..else.. in this syntax. The code is executed **only if the specified condition is true**.

CONDITIONAL STATEMENTS - PHP IF...ELSE STATEMENTS

- The if...else Statement
- Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
- Syntax

```
if (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, 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>
```

CONDITIONAL STATEMENTS - PHP IF...ELSE STATEMENTS

- The if...elseif....else Statement
- Use the if....elseif...else statement to select one of several blocks of code to be executed.
- 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>
```

PHP SWITCH STATEMENT

- The PHP Switch Statement
- Use the switch statement to select one of many blocks of code to be executed.
- Syntax

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

- This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.
- Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP SWITCH STATEMENT

- Example

```
<html>
<body>

<?php
$x=1;
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP ARRAYS

- A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

- An array is a special variable, which can store multiple values in one single variable.
  - If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:
    - $cars1="Saab";
    - $cars2="Volvo";
    - $cars3="BMW";

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
  - The best solution here is to use an array!

- An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

- Each element in the array has its own index so that it can be easily accessed.
  - In PHP, there are three kind of arrays:
    - **Numeric array** - An array with a numeric index
    - **Associative array** - An array where each ID key is associated with a value
    - **Multidimensional array** - An array containing one or more arrays

PHP ARRAYS - NUMERIC ARRAYS

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example

In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

Saab and Volvo are Swedish cars.

PHP ARRAYS - ASSOCIATIVE ARRAYS

- An associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.
- With associative arrays we can use the values as keys and assign values to them.

- Example 1
  - In this example we use an array to assign ages to the different persons:
    $ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);

- Example 2
  - This example is the same as example 1, but shows a different way of creating the array:

    $ages['Peter'] = "32";
    $ages['Quagmire'] = "30";
    $ages['Joe'] = "34";
    The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:
Peter is 32 years old.

PHP ARRAYS - MULTIDIMENSIONAL ARRAYS

- In a multidimensional 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.

Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP ARRAYS - MULTIDIMENSIONAL ARRAYS

The array previous would look like this if written to the output:

```
Array
    (
    [Griffin] => Array
      (
      [0] => Peter
      [1] => Lois
      [2] => Megan
      )
    [Quagmire] => Array
      (
      [0] => Glenn
      )
    [Brown] => Array
      (
      [0] => Cleveland
      [1] => Loretta
      [2] => Junior
      )
    )
```

PHP ARRAYS - MULTIDIMENSIONAL ARRAYS

Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] . " a part of the Griffin family?";
```

The code above will output:

Is Megan a part of the Griffin family?

PHP LOOPING

- Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

- In PHP, we have the following looping statements:

  - **while** - loops through a block of code while a specified condition is true

  - **do...while** - loops through a block of code once, and then repeats the loop as long as a 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 LOOPING - WHILE LOOPS

- The while Loop, The while loop executes a block of code while a condition is true.
- Syntax

```
while (condition)
{
        code to be executed;
}
```

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
        echo "The number is " . $i . "<br />";
        $i++;
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP LOOPING - WHILE LOOPS

- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.
- Syntax

```
do
{
        code to be executed;
}
while (condition);
```

- The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

PHP LOOPING - FOR LOOPS

- The for loop is used when you know in advance how many times the script should run.
- Syntax

```
for (init; condition; increment)
{
        code to be executed;
}
```

Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the iteration)

Note: The *init* and *increment* parameters above can be empty or have multiple expressions (separated by commas).

Example

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP LOOPING - FOR LOOPS

- The foreach loop is used to loop through arrays.
- Syntax

```
foreach ($array as $value)
  {
        code to be executed;
  }
```

- For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

- Example
  - The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```

Output:

```
one
two
three
```

PHP FUNCTIONS

- Create a PHP Function
- A function will be executed by a call to the function.
- Syntax

```
function functionName()
{
        code to be executed;
}
```

- PHP function guidelines:
  - Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)
  - Example
    - A simple function that writes my name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

Output:

My name is Kai Jim Refsnes

PHP FUNCTIONS - ADDING PARAMETERS

- PHP Functions - Adding parameters
- To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- Parameters are specified after the function name, inside the parentheses.
  - Example 1
    - The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.

PHP FUNCTIONS - ADDING PARAMETERS

- Example 2
  - The following function has two parameters:

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

Output:

My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?

PHP FUNCTIONS - RETURN VALUES

- To let a function return a value, use the return statement.
- Example

```
<html>
<body>

<?php
function add($x,$y)
{
        $total=$x+$y;
        return $total;
}

echo "1 + 16 = " . add(1,16);
?>

</body>
</html>
```

Output:

1 + 16 = 17

PHP FORMS - USER INPUT

- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.
  - Example
    - The example below contains an HTML form with two input fields and a submit button:

```
<html>
<body>

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

</body>
</html>
```

- When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called "welcome.php": "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

Welcome John!
You are 28 years old.

PHP FORMS - $_GET VARIABLE

- The predefined $_GET variable is used to collect values in a form with method="get"
- Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.
  - Example

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

- When the user clicks the "Submit" button, the URL sent to the server could look something like this:
  - http://www.w3schools.com/welcome.php?fname=Peter&age=37

- The "welcome.php" file can now use the $_GET variable to collect form data (the names of the form fields will automatically be the keys in the $_GET array):
  - Welcome <?php echo $_GET["fname"]; ?>.
  - You are <?php echo $_GET["age"]; ?> years old!

PHP FORMS - $_POST VARIABLE

- The predefined $_POST variable is used to collect values from a form sent with method="post".
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
  - **Note:** However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).
  - Example

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

- When the user clicks the "Submit" button, the URL will look like this:
  - http://www.w3schools.com/welcome.php

- The "welcome.php" file can now use the $_POST variable to collect form data (the names of the form fields will automatically be the keys in the $_POST array):
  - Welcome <?php echo $_POST["fname"]; ?>!
  - You are <?php echo $_POST["age"]; ?> years old.

REFERENCES

W3Schools

- http://www.w3schools.com/php/default.asp

PHP: Hypertext Preprocessor

- http://www.php.net/

XAMPP

- http://www.apachefriends.org/en/xampp.html

APACHE

- http://www.apache.org/

MYSQL

- http://www.mysql.com/

คำถาม

1. สร้างหน้า php เพื่อแสดงชื่อและนามสกุลของตนเอง ?
2. ทำให้ ชื่อ และ นามสกุล เป็นตัวแปร และนำมาต่อกัน เพื่อแสดงบน Browser ?
3. นับตัวอักษร ของชื่อ และ นับตัวอักษรของนามสกุล เพื่อแสดงบนหน้าจอ
4. ให้หน้าเว็บแสดง เกรด ให้ถูกต้องโดยใช้ php Conditional statement

   “A” เมื่อคะแนน มากกว่า หรือ เท่ากับ 80

   “B” เมื่อคะแนนอยู่ที่ระหว่าง 70 - 79

   “C” เมื่อคะแนนอยู่ที่ระหว่าง 60 - 69

   “D” เมื่อคะแนนอยู่ที่ระหว่าง 50 – 59

   “F” เมื่อคะแนนน้อยกว่า 50
5. ให้หน้าเว็บสมามารถเปลี่ยนภาษา ได้โดย ให้ผลลัพธ์เป็นตัวแปร (เปลี่ยนภาษาที่ตัวแปร ก่อนแสดงผล)

   “ดีมากๆ” เมื่อคะแนน มากกว่า หรือ เท่ากับ 80

   “ดีมาก” เมื่อคะแนนอยู่ที่ระหว่าง 70 - 79

   “ดี” เมื่อคะแนนอยู่ที่ระหว่าง 60 - 69

   “พอใช้” เมื่อคะแนนอยู่ที่ระหว่าง 50 – 59

   “ควรปรับปรุง” เมื่อคะแนนน้อยกว่า 50

คำถาม

6. จากข้อ 5-6 ทำให้เป็น Function
7. จากข้อ 5-7 ทำให้สามารถรองรับ คะแนน คู่กับ การเลือกภาษา จาก Array ได้
8. สร้างตารางสี

    9.1 ไล่สี หนึ่งสี

    9.2 ไล่สี สองสี

    9.3 ไล่สี สามสี
9. ทำ from เพื่อ เป็นหน้าจอ เครื่องคิดเลขเพื่อส่งค่า ผ่าน get ของ php เพื่อคำนวณผลลัพธ์แล้วแสดงผลเมื่อกดคำนวณ
10. สร้างตารางปฏิทิน โดยผ่านหน้า Form โดย Input คือ

    ภาษาที่ต้องการแสดงผล

    วัน เริ่มต้นของเดือน

    จำนวนวันสุดภายในเดือน