

Basic PHP

ITCS 210 Web Programming

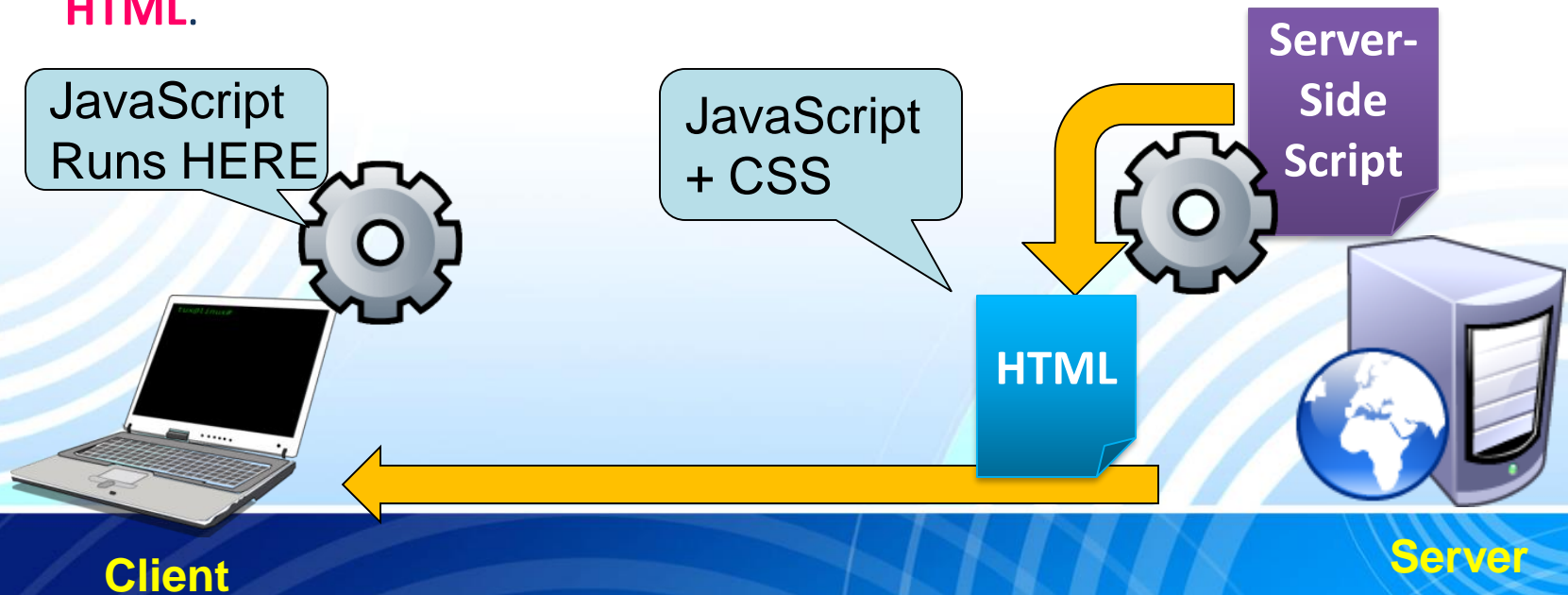


Class Objectives

- Understand basic **concept** of PHP
- Be able to **apply** PHP in web programming.

Server-Side Script

- Server-side scripting is about "programming" the behavior of the server. This is called server-side scripting or server scripting.
- Normally when a browser requests an HTML file, the server returns the file, but if the file contains a **server-side script**, the script inside the HTML file is **executed** by the server before the file is returned to the browser as **plain HTML**.



What is PHP?

- PHP is a powerful **Server-Side Script**.
- PHP is the widely-used, free, and efficient.
- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- A language that combines elements of Perl, C, and Java

Server-
Side
Script

Why PHP?

- PHP is **COMPATIBLE** with almost all servers used today (Apache, IIS, etc.) and many platforms (Windows, Linux, Unix, 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 Files

- 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 can PHP 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 results to a browser
- Customize a Web page to make it more useful for individual users
- Provide **security** since your server code cannot be viewed from a browser

Get Start with PHP

- We have to use special syntax in order to start PHP Script.

(Like we put JavaScript in a special tag called `<script ...> ... </script>`)

- There are several styles can be used.
- Standard form
 - `<?php Your code ?>`
- Short form
 - `<? Your code ?>`

Where to Put the PHP?

- **ANYWHERE** in the PHP file where the generated HTML code will be placed correctly.

```
<?php
    include ("moreFunction.php");
?>
<html>
<head>
    <?php
        function hello()
        { echo "Hello World"; }
    ?>
</head>
<body>

    <?php    hello();    ?>
</body>
</html>
```


Comment

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

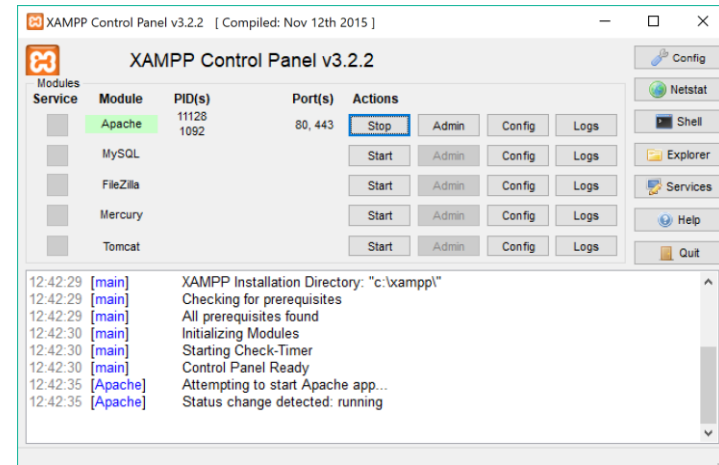
```
<html><body>
  <?php
    //This is a single line comment

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

Server: Localhost

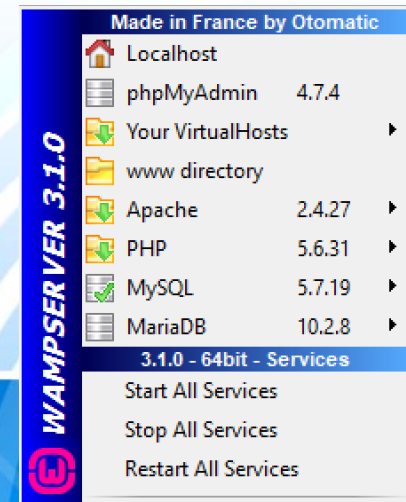
Install XAMPP

- Open xampp control panel:
- run Apache server
- Open web browser enter: localhost/index.php
- serverRoot: C:\xampp\htdocs



Install WAMP

- Open wamp control panel: run Apache server
- Open web browser enter: localhost/index.php
- serverRoot: C:\wamp64\www

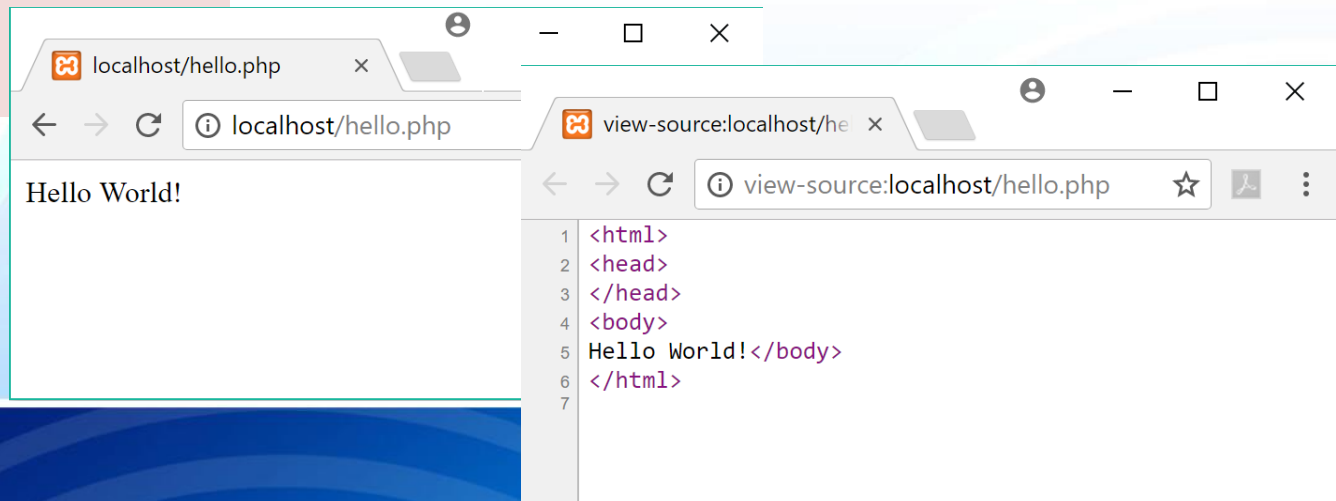


Activity 1

Running Hello.php

```
<html>
<head>
</head>
<body>
<?php
    $greeting = "Hello World!";
    echo $greeting;
?>
</body>
</html>
```

1. Put your PHP files to the server root directory of XAMPP/WAMP
2. Open a web browser enter: localhost/hello.php (don't forget to open the server)



PHP Variable

- **Loosely Typed Language**
 - does not need to be declared before adding a value to it
- **Naming Rules for Variables**
 - Always start with **\$ (Dollar Sign)** then follow by a letter or an underscore "_"
 - A variable name can only contain alpha-numeric characters and underscores (a-z, 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 an underscore (\$my_string), or with capitalization (\$myString)
 - Case sensitive

Example : \$day, \$fullName, etc...

Data Type

- Boolean -> TRUE or FALSE
- Integer {..., -2, -1, 0, 1, 2, ...}
- Double
- String
- Object
- Array
- Date
- NULL – represents that a variable has no value. NULL is the only possible value.
- In PHP you don't need to define data type for your variable

About Data type

Function	Description
<code>gettype(\$varName)</code>	Return data type as a string with value <ul style="list-style-type: none">•“boolean”•“integer”•“double”•“string”•“array”•“object”•“resource”•“unknown type”
<code>settype(string varName,string newType)</code>	To change data type in a variable

You can also change data type by casting technique.

Example

```
$data = 3.5;
```

```
gettype($data);           //result → double
```

```
$newData = (integer)$data ;
```

```
gettype($newData);        //result → integer
```

Echo and Print

- Both Echo and Print are basic PHP command to print text out. Two of them can be used in replacement. However PHP programmers usually use Echo.

- Syntax

echo "text";

print "text";

```
<html><head></head>
```

```
<body>
```

```
<?
```

```
    echo "Welcome to PHP World <br />"
```

```
    print "Enjoy programming!"
```

```
?>
```

```
</body></html>
```

Output

```
Welcome to PHP World
Enjoy programming!
```

Print out variables

- You can use variables inside of an echo statement
`$foo = "foobar";`
`$bar = "barbaz";`
`echo "foo is $foo"; // foo is foobar`
- You can also use arrays
`$baz = array("value" => "foo");`
`echo "this is {$baz['value']} !"; // this is foo !`
- Using single quotes will print the **variablename**, not the value
`echo 'foo is $foo'; // foo is $foo`
- If you are not using any other characters, you can just echo variables
`echo $foo; // foobar`
`echo $foo,$bar; // foobarbarbaz`

String

- For value that contains characters
- Usage
 - Directly in a function
 - Stored in a variable
- Example:

```
<?php
    $txt="We want quiz";
    echo $txt;
?>
```

String

- Concatenation Operator

- Only 1 String operator in PHP !
- Used to put strings together
- Use (.) as a symbol for coding
- Ex:

```
<?php
    $txt1="Hello World!";
    $txt2="This is a time to quiz!";
    echo $txt1." ".$txt2;
?>
```

- Output:

Hello World! This is a time to quiz!

The Concatenation Operator

```
<?php
$num = 5;
$txt1="The war will begin in
next";
$txt2="days.";
$txt2 = $txt1 . " " . $num . "
" . $txt2;
echo $txt2;
?>
```

- output

The war will begin in next 5 days.

Built-in Functions for String: strlen()

- The strlen() function
 - It will return the length of a string.
- Syntax:

strlen(string);

- Ex:

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

- Output:

12

Built-in Functions for String: strpos()

- The strpos() function
 - It is used to search for character in a string.
- Syntax:

`strpos(originalString , positionfindingString);`

- Ex:

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

- Output:

6

The word “**world**” appears after the 6th letter in the string.

Built-in Functions for String: Explode()

- Explode()

- This function breaks a string into an array using a string separator.

- Syntax:

`explode(stringSeperator , originalString);`

- Ex:

```
<?php
    $txt = "Hello world.";
    print_r (explode(" ", $txt));
?>
```

- Output:

```
Array
(
    [0] => Hello
    [1] => world.
)
```

Note:

print_r() is a function to display information about a variable in a way that's readable by humans.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	$x=y$	$x=y$
+=	$x+=y$	$x=x+y$
-=	$x-=y$	$x=x-y$
=	$x=y$	$x=x*y$
/=	$x/=y$	$x=x/y$
.=	$x.=y$	$x=x.y$
%=	$x\%=y$	$x=x\%y$

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
	or	x=6 y=3 (x==5 y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

If Statement

```
if (condition)
{
    code to be executed if
    condition is true
}
```

- Example:

```
<html>
<body>

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

    ?>

</body>
</html>
```

Conditional Statements

- **if....else statement**

if (condition)

{ code to be executed if condition is true }

else

{ code to be executed if condition is not true }

- **if...elseif....else statement**

if (condition1)

{ code to be executed if condition1 is true }

elseif *(condition2) // The word “elseif” are concatenated*

{code to be executed if condition2 is true }

else

{ code to be executed if condition1 and condition2 are not true }

Switch Statement

- 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 not both label1 and label2;  
}
```

Example

```
<html><body>  
<?php  
    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>
```

for Loop

- Syntax

```
for (var=start value; var<=end value; var=var+increment)
{ statement(s); }
```

- Example

```
<?php
    for($i=1;$i<=10;$i++)
    {
        echo $i. "<br />";
    }
?>
```

While loop, Do...While loop

- while – execute the statements as long as the expression evaluates to TRUE
- Syntax

```
while (var <= endvalue)
{
    code to be executed
}
```
- do-while – same as while except the code chunk is guaranteed to execute at least once

- Syntax

```
do
{
    code to be executed
}
while (var <= endvalue);
```

Break Statement

- *break* ends execution of the current *for*, *foreach*, *while*, *do-while* or *switch* structure.
- *break* accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

Normal Break

```
<?php
for($i = 0; $i<5 ; $i++)
{
    if ($i == 3)
        break;
    echo "$i <br />";
}
?>
```

Break with optional argument

```
<?php
$i = 0;
while (++$i) {
    switch ($i)
    {case 5:
        echo "At 5<br />";
        break 1; /* Exit only the switch.*/
    case 10:
        echo "At 10; quitting<br />";
        break 2; /* Exit the switch and the while.*/
    default:
        break;
    }
}
?>
```

Continue Statement

- **continue** is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration.
- *continue* accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.
- Example

```
<?php
    for ($i = 0; $i < 5; ++$i)
    {
        if ($i == 2)
            continue;
        print "$i\n";
    }
?>
```

Create a PHP Function

- The real power of PHP comes from its functions.
- Two Types of Functions
 - Built in Functions
 - User defined functions
- User defined functions
- Syntax

```
function functionName(VariableName1,..., VariableNameN)
{
    code to be executed;
    return ReturnValue;
}
```

****Parameter and return value are optional**

Example

```
<html>
<head></head>
<body>
    <?php
        function add($x,$y)
        {
            $total=$x+$y;
            return $total;
        }
        echo "1 + 16 = " .
add(1,16) ;
    ?>
</body>
</html>
```

Output

1 + 16 = 17

Returning Values

- A value can be returned by using the optional return() statement
- Function execution is ended immediately and control passed back to the line that called the function
- Returns a single variable
 - could be a single number or string
 - could be an array with several values

Example

```
<?php
function display($string1, $string2="World")
{
    return $string1 . $string2;
}

echo display("Hello", "Todd");// displays Hello Todd
echo display("Hello");//displays Hello World
?>
```

Variable Scope

- For the most part PHP variables have a single scope

```
<?php
    $b=1;
    function footer_info ()
    {
        echo $b;
    }
    footer_info();
?>
```

- Nothing will output. The echo statement refers to the local scope of the variable (inside the function.) Global variables must be declared global inside the function.

Variable Scope (Con.)

- If you need to refer to a variable in a function, there are many ways.
 - Include
 - Passing Variable
 - Global Variable

- **Include**

```
<?php
    $a=1;
    include ( 'header_info.php' );
?>
```

You can reference the \$a variable in the file 'header_info.php'

Variable Scope (Con.)

- Passing Variable

```
<?php
    $a = 1;
    $b = 2;

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

    Sum($a, $b);
    echo $b;

?>
```

Output of \$b will be 2.

- Global Variable

```
<?php
    $a = 1;
    $b = 2;

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

    Sum();
    echo $b;

?>
```

Output of \$b will be 3.

PHP Form Handling

- In HTML, we have form but we cannot manipulate data in it.
- In JavaScript, we can manipulate data in form in a certain level.
- In PHP, we can manipulate data in form and can do data manipulation with a server.
- The PHP `$_GET` and `$_POST` variables are used to retrieve information from forms, like user input.

HTML Form

```
<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>
```

From the code above, the form will send data to **welcome.php** with the **post method**. So if a user clicks at Submit button. The page welcome.php will be opened and data in the form will be sent to that page.

The \$_POST Function

- If your form defines `method="post"`, the post method will be used to send data.
- The way that PHP send data is to store all the "posted" values into an *associative array* called `"$_POST"`.
An associative array is a kind of array that its index value can be a string.
- To get data from post method in the destination page, names of input(index of \$_POST) must be known

\$_POST Example

- **Data sender page**

```
<form action="process.php" method="post">  
  <select name="item"> ... </select>  
  <input name="quantity" type="text" />  
</form>
```

- **Data receiver page**

```
<?php  
  $quant = $_POST['quantity'];  
  $item = $_POST['item'];  
  echo "The quantity is $quant ";  
  ...  
?>
```

The \$_GET Function

- If your form defines `method="get"`, the get method will be used to send data.
 - The *get* method passes the variables along to the "process.php" web page by **appending** them onto the end of the URL. The URL, after clicking submit, would have this added on to the end:
`"?item=##&quantity=##"`
 - The question mark "?" tells the browser that the following items are variables. Now that we changed the method of sending information on "order.html", we must change the "process.php" code to use the `"$_GET"` associative array.
- **In get method, the value sent can be seen on the URL. So it is not suitable for important data.**

\$_GET Example

- **Data sender page**

```
<form action="process.php" method="get">  
  <select name="item"> ... </select>  
  <input name="quantity" type="text" />  
</form>
```

- **Data receiver page**

```
<?php  
$quant = $_GET['quantity'];  
$item = $_GET['item'];  
echo "The quantity is $quant ";  
...  
?>
```


Some Useful Built-in Functions

- PHP provides built-in functions for checking if a variable exists, checking if a variable holds a value, and removing a variable.

Function	Explanation	Example
isset()	Checks to see if a variable exists. Returns true or false. Returns TRUE if <i>a variable</i> exists and has value other than NULL, FALSE otherwise	isset(\$a)
unset()	Removes a variable from memory.	unset(\$a)
empty()	Checks to see if a variable contains a non-empty, non-false value.	empty(\$a)

- Example:

```
<?php
    if (isset($_POST["username"]))
    { echo $_POST["username"]; }
?>
```


Array

- Special variable
- It can store multiple values in single variable.
- We can access the values by referring to the array name.
- Each element in the array must have an index for easy accessing.
- 3 kinds of array in PHP:
 - Numeric array
 - Multidimensional array
 - Associative array

Numeric Array

- Store each element with a **numeric index**
- 2 methods of array creation:

- 1) Through array function

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

- 2) Through array identifier \$arrayName[]

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

Defining specific index

Or

```
$cars[]="Saab";  
$cars[]="Volvo";  
$cars[]="BMW";  
$cars[]="Toyota";
```

Create an array if that array name has not defined yet.

or

Add new value to an array if it has been defined.

Numeric Array

- Example:

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

- Output: Saab and Volvo are Swedish cars.

Multidimensional Array

- Multidimensional Array can be used as a normal numeric array but it will have two blocks for index.
- 2D array will see array index as a coordinate of **row** and **column**.
- Syntax:
 \$array[\$number][\$number]
- Example:

```
$num = 5;  
for ($x = 1; $x <= 12; $x++)  
{  
    $arr[$x][0] = $num;  
    $arr[$x][1] = $x;  
    $arr[$x][2] = $num*$x;  
}
```

Associative Array

- Each ID key is associated with a value.
- It is good for storing data about specific named values.
- So an associative array is a kind of array that its index value(key) can be a string.

```
$staff = array("name"=>"Kitty", "id"=>"105", "age"=>22) ;
```

- 2 methods of array creation:
 - 1) Through array function
 - 2) Through array identifier **\$arrayName[]**

```
$staff["name"] = "Kitty";  
$staff["id"] = "105";  
$staff["age"] = 22;
```

Associative Array

- Example:

```
<?php
    $ages['Jib'] = "82";
    $ages['Nor'] = "22";
    $ages['Jeen'] = "21";

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

- Output:

Jib is 82 years old.

We can refer to data using its key as an referring index.

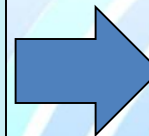
Foreach Loop

- This kind of loop is used for looping through arrays.
 - Syntax: For Numeric array

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

- Example:

```
<?php
$w=array("Dog", "Cat", "Iguana");
foreach ($w as $value)
{
    echo $value."<br />";
}
?>
```



Output:

```
Dog
Cat
Iguana
```

Foreach Loop

- Syntax: For Associative array

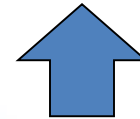
```
foreach ($array as $key=>$value)
{
    code to be executed;
}
```

Output:

```
Name = Kitty
Id = 105
Age = 22
```

- Example:

```
<?php
$staff = array("name"=>"Kitty", "id"=>"105", "age"=>22);
foreach ($staff as $key=>$value)
{
    echo "$key = $value <br />\n";
}
?>
```



Constants

- Constants just as variables are used to store information.
- Constant value can not be changed in the process of running program.
- It can be mathematic constants, passwords, paths to files, etc.
- Syntax:

`define(stringVarName,value);`

- Example:

```
<?php
    define("PASSWORD", "admin");
    echo PASSWORD;
    echo constant("PASSWORD");
    echo "PASSWORD";
?>
```

Output:

```
admin
admin
PASSWORD
```

Redirect

- Redirect is an action of changing a user page from the page they entered to a different web page.

- Syntax:

`header("Location: URL");`

- The URL could be URL of a website, file name, or variable
- Example:

```
header("Location: http://www.ict.mahidol.ac.th");  
  
header("Location: Login.php");  
  
header("Location: ".$_SERVER["PHP_SELF"]);
```



To refresh the current page

Session

- The concept of session is to allow user information to be stored on the server for later use. However, session information is temporary and will be deleted after the user has left the website.
- Sessions work by creating a **unique id (UID)** for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.
- This data is kept in a session data store, and it is updated on each request. Because the unique identifier specifies a particular record in the session data store, it's most often called the session identifier.

Start the session

- Before you can store user information in your PHP session, you must first start up the session.
- The `session_start()` function should appear **BEFORE** the `<html>` tag

Example:

```
<?php
    session_start();
?>
<html>
...
```


How to use session

- The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable which is a global array. You can create new room to keep data by defining new index to `$_SESSION`.
- Example

```
<?php
    session_start();
    if(isset($_SESSION['views']))
        $_SESSION['views'] = $_SESSION['views']+ 1;
    else
        $_SESSION['views'] = 1;
<html><body>
<?php
    echo "Pageviews=" . $_SESSION['views'];
?>
</body></html>
```

How to delete session

- If you wish to delete some session data, you can use the functions **unset()** or **session_destroy()**.
- The **unset()** function is used to free the specified session variable. It is suitable if you would like to delete only one room in the array `$_SESSION`.
- The **session_destroy()** function is used to completely destroy the session. It will reset your session and you will lose all your stored session data.
- Example:

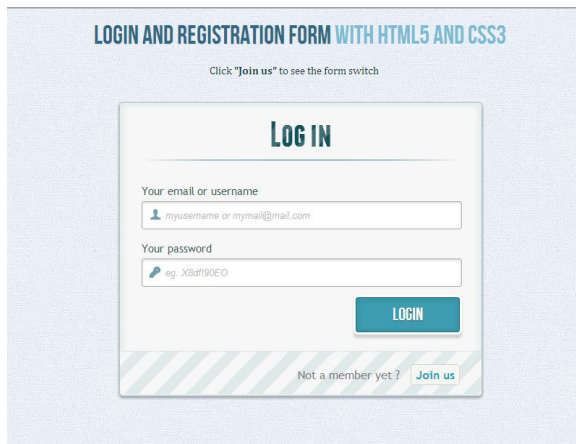
```
<?php  
    unset($_SESSION['views']);  
    session_destroy();  
?>
```

References

- <http://www.w3schools.com/php/default.asp>
- <http://www.php.net/manual/en/>

Activity 2

Index.php (contains two forms)



LOGIN AND REGISTRATION FORM WITH HTML5 AND CSS3

Click "Join us" to see the form switch

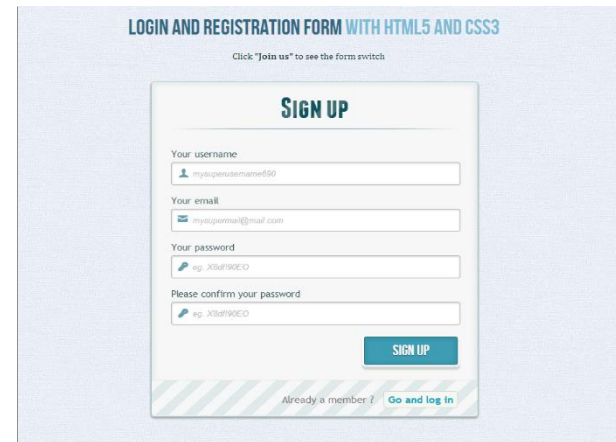
LOG IN

Your email or username

Your password

LOGIN

Not a member yet? [Join us](#)



LOGIN AND REGISTRATION FORM WITH HTML5 AND CSS3

Click "Join us" to see the form switch

SIGN UP

Your username

Your email

Your password

Please confirm your password

SIGN UP

Already a member? [Go and log in](#)

login.php
(Check authentication
and redirect to
welcome page)

↓

welcome.php

WELCOME TO OUR SYSTEM

↓

signup.php

THANK YOU FOR REGISTRATION
YOUR USERNAME IS ADMIN
YOUR EMAIL ADDRESS IS ADMIN@MYWEB.COM

**The given files do not work quite well as required,
you must modify the given files based on the requirements**

- login.php
 - Accept 3 username & password
 - {admin,123} {user,456} {sale,789} These username & password are kept in an array
 - Check the username & password
 - If correct → go to “welcome.php”
 - If wrong → create a link to “index.php”
- welcome.php
 - If user already login show “Welcome to our system”
 - If user type the url address without login show “Please login first”
- signup.php
 - Show “Thank you for registration”
 - Show “Your username is _____”
 - Show “Your email address is _____”