

Web Application Development

PHP



PHP is a general purpose server side language originally designed for web development to produce dynamic web pages



What is PHP

- PHP: Hypertext Processor
- Open source, server side scripting language
- Used to generate dynamic web pages
- PHP scripts reside between reserved PHP tags (<?php ?>). This allows programmers to embed PHP scripts into web pages.

What is PHP

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
 - 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

What does PHP code look like

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

Comments in PHP

- Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment  
  
# Shell-style comments  
  
/* C-style comments  
    These can span multiple lines */
```

Variables in PHP

- PHP variables must begin with a \$ sign.
- Case sensitive (\$Foo != \$foo != \$fOo).
- Global and locally-scoped variables
- Global variables can be used anywhere
- Local variables restricted to a function or class
- Certain variable names reserved by PHP, like \$_GET, \$_POST, \$_SERVER

Variable Usage

```
<?php
$foo = 25;           // Numerical variable
$bar = "Hello";      // String variable

$foo = ($foo * 7);    // Multiplies foo by
$bar = ($bar * 7);    // Invalid expression
?>
```


echo

- The PHP command `'echo'` is used to output the parameters passed to it
- The typical usage for this is to send data to the client's web-browser

echo example

```
<?php
$foo = 25;          // Numerical variable
$bar = "Hello";    // String variable

echo $bar;          // Outputs Hello
echo $foo,$bar;     // Outputs 25Hello
echo "5x5=", $foo;  // Outputs 5x5=25
echo "5x5=$foo";    // Outputs 5x5=25
echo '5x5=$foo';    // Outputs 5x5=$foo
?>
```

echo example

- Notice how echo `'5x5=$foo'` outputs `$foo` rather than replacing it with 25
- Strings in single quotes (`' '`) are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as `"\n"` or `"\\"`)

Arithmetic Operators

Arithmetic Operators		
Example	Name	Result
+\$a	Identity	Conversion of \$a to <u>int</u> or <u>float</u> as appropriate.
-\$a	Negation	Opposite of \$a.
\$a + \$b	Addition	Sum of \$a and \$b.
\$a - \$b	Subtraction	Difference of \$a and \$b.
\$a * \$b	Multiplication	Product of \$a and \$b.
\$a / \$b	Division	Quotient of \$a and \$b.
\$a % \$b	Modulo	Remainder of \$a divided by \$b.
\$a ** \$b	Exponentiation	Result of raising \$a to the \$b'th power. Introduced in PHP 5.6.

Ref: <http://php.net/manual/en/language.operators.arithmetic.php>

String Concatenation

- Use a period to join strings into one.

```
<?php
$string1="Hello";
$string2="PHP";
$string3=$string1 . " " . $string2;
print $string3;
?>
```

```
Hello PHP
```

Escaping the character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php  
$heading="\ "Computer Science\ "  
print $heading;  
?>
```

```
"Computer Science"
```

PHP Control Structure - Example

```
if ($foo == 0) {  
    echo 'The variable foo is equal to 0';  
}  
else if (($foo > 0) && ($foo <= 5)) {  
    echo 'The variable foo is between 1 and 5';  
}  
else {  
    echo 'The variable foo is equal to '.$foo;  
}
```

PHP While Loop - Example

```
<?php
$count=0;
while($count<3){
    print "hello PHP. ";
    $count += 1;
}
?>
```

```
hello PHP. hello PHP. hello PHP.
```


PHP For Loop - Example

```
<?php
for ($i = 0; $i < 3; $i++) {
    print "hello PHP. ";
}
?>
```

```
hello PHP. hello PHP. hello PHP.
```

- Ref: <http://php.net/manual/en/language.control-structures.php>

Date Display

```
<?php
// Prints something like: Friday
echo date("l") . "<br>";

// Prints something like: Friday 25th of November 2016 10:30:46 AM
echo date('l jS \of F Y h:i:s A') . "<br>";

// Prints: November 25, 2016 is on a friday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 11, 25, 2016)) . "<br>";

// prints something like 2016/11/25
echo date('Y/m/d') . "<br>";

// prints something like 16/11/25
echo date('y/m/d') . "<br>";

// prints something like Friday, November 11, 2016
echo date("l, F m, Y") . "<br>";
?>
```

Date Display

Friday

Friday 25th of November 2016 11:32:19 AM

July 1, 2000 is on a Friday

2016/11/25

16/11/25

Friday, November 11, 2016

- Ref: <http://php.net/manual/en/function.date.php>

Functions

- Functions MUST be defined before they can be called
- Function headers are of the format:

```
function functionName($arg_1, $arg_2, ..., $arg_n)
```

- Note that no return type is specified
- Unlike variables, function names are not case sensitive
(foo(...) == Foo(...) == FoO(...))

Function Example

```
function foo($arg_1, $arg_2){  
    return $arg_1 * $arg_2;  
}  
  
$result_1 = foo(12, 3); // Invoke the function and store the result  
echo $result_1;         // Outputs 36  
echo foo(12, 3);        // Outputs 36
```

Include Files

Include “opendb.php”;

Include “closedb.php”;

This inserts files; the code in files will be inserted into current code. This will provide useful and protective means once you connect to a database for example, as well as for other repeated functions.

Include Files Example

`include ('includes/header.html');`

header.html may look something like:

```
<!DOCTYPE html>
<html>
<head>
  <title><?php echo $page_title; ?></title>
</head>
<body>
  <h1>Welcome to online Shop</h1>
  <h2>Login required to enter site</h2>
  <form action="login.php" method="post">
    <p>Username: <input type="text" name="username" size="20" maxlength="60" value="<?php if (isset($_POST['username'])) echo $_POST['username']; ?>" /> </p>
    <p>Password: <input type="password" name="pass" size="10" maxlength="20" value="<?php if (isset($_POST['pass'])) echo $_POST['pass']; ?>" /> </p>
    <p><input type="submit" name="login" id="login" value="Login" /></p>
    <p><input type="submit" name="register" id="register" value="Register" /></p>
  </form>
```

Include Files Example

`include ('includes/footer.html');`

footer.html may look something like:

```
<div id="footer">
    <p>Copyright &copy; <a href="#">Online Shop</a> 2016 </p>
</div>
</body>
</html>
```


Forms

- Access to the HTTP POST and GET data is simple in PHP
- The global variables `$_POST[]` and `$_GET[]` contain the request data

Forms Example

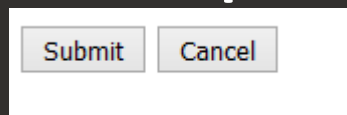
```
forms.php x
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Forms</title>
5 </head>
6 <body>
7     <form action="forms.php" method="post">
8         <input type="submit" name="submit" value="Submit">
9         <input type="submit" name="cancel" value="Cancel">
10    </form>
11
12    <?php
13        if(isset($_POST["submit"])){
14            echo "<h2>You clicked Submit!</h2>";
15        }
16        else if(isset($_POST["cancel"])){
17            echo "<h2>You clicked Cancel!</h2>";
18        }
19    ?>
20
21 </body>
22 </html>
```

Forms Example

- Note the use of the isset function in the if statement. This determines if a variable is set and is not null. Without it you would get an error on the initial load of the web page as submit or cancel buttons must be clicked before \$_POST is assigned a value.
- Ref: <http://php.net/manual/en/function.isset.php>

Forms Example

On opening this page in the browser:



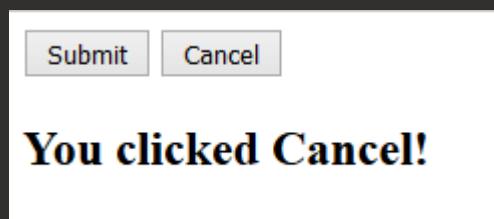
A white rectangular box containing two buttons: "Submit" and "Cancel".

When Submit button is clicked on the page it will update to:



A white rectangular box containing two buttons: "Submit" and "Cancel". Below the buttons, the text **You clicked Submit!** is displayed in a bold, black, serif font.

When Cancel button is clicked on the page it will update to:

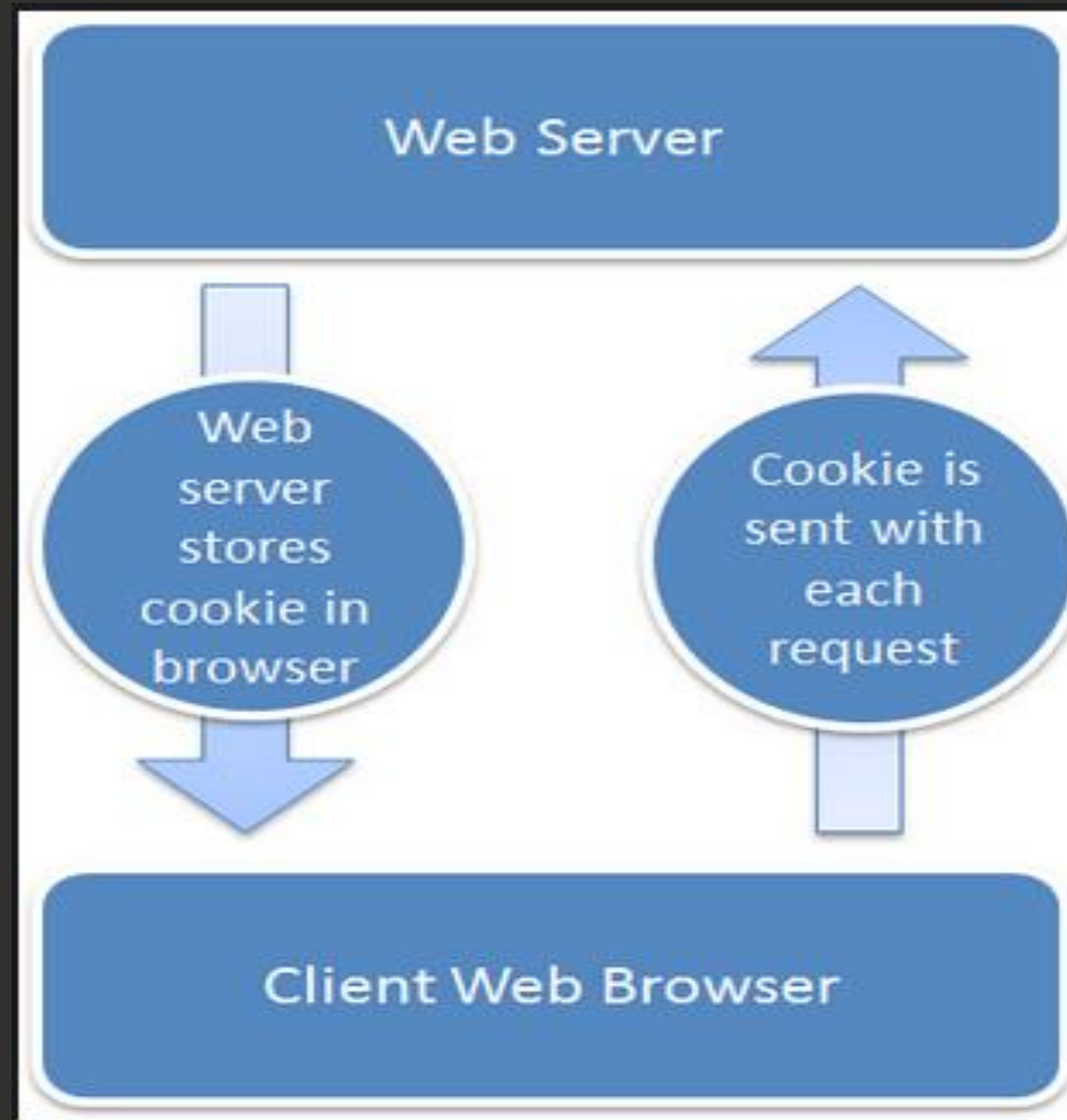


A white rectangular box containing two buttons: "Submit" and "Cancel". Below the buttons, the text **You clicked Cancel!** is displayed in a bold, black, serif font.

Cookies

- Cookies are a way for a server to store information on a users machine.
- This is one way that a site can remember or track a user over the course of a visit.
- Think of a cookie as being like a name tag:
 - You tell the server your name and it gives you a sticker to wear. Then it can know who you are by referring back to the name tag.

Cookies



Cookies

- Cookies are cleared by default when the browser session ends. You can configure the cookie to expire after a period of time. For example, set expiry to 30 minutes. This means even if the user exits the browser, they can access the cookie from a new browser session if the browser was reopened in this 30 minute period.

Ref: <http://php.net/manual/en/function.setcookie.php>

- Hint: Think default expiry as sessionStorage, and setting expiry as localStorage (with a time limit).

Cookies

```
<?php
if(isset($_POST["add"])){
    if(isset($_POST['username'])){
        // add username from a form. Cookie expires in 1 hour (3600 seconds)
        setcookie("username",$_POST['username'], time()+3600);
    }
}

if(isset($_POST["clear"])){
    unset($_COOKIE["username"]);
    setcookie("username","x",time()-1);
    /*unset the cookie, just unsets it for the current request so its not available.
    Need to expire it, to destroy it - setcookie method needed. Any valaue will do - used x in this example*/
}
?>
```


Cookies

```
<?php
    //On loading the page, check if cookie has a value
    if(isset($_COOKIE["username"])){
        echo 'Welcome ' . $_COOKIE["username"];
    }else{
        echo 'Not sure who you are ?';
    }
?>
```

Sessions

- Another method of making data available to multiple pages of a website is to use sessions.
- The premise of a session is that data is stored on the server.

Sessions vs. Cookies

- Sessions:
 - They are generally more secure as data is held on the server and not passed back and forth between browser and server as happens with cookies.
 - They allow for more data to be stored (cookies are limited in amount of storage they have by browsers).
 - Browsers need to have cookies enabled in order to store/access data, where as sessions not affected by browser settings.

Sessions vs. Cookies

- Cookies:
 - They are easier to program.
 - They require less of the server.
 - They can be made to last far longer. The expire property can be set, whereas session data is destroyed when the browser session ends.
 - Will work if using multiple servers. If load balancing is used then session data maybe held on one server but if next request goes to another replica server then the session data won't be available.

Sessions vs. Cookies

- In general to store and retrieve just a couple of small pieces of information, or to store information for longer duration, use cookies. For most web applications though the trend is to use sessions.

Sessions

```
<?php
if(isset($_POST["add"])){
    if(isset($_POST['username'])){
        /* start the session.
        Must always be included for each page requiring so it can access the session */
        session_start();
        $_SESSION['username'] = $_POST['username']; // Add data to the session
    }
}
if(isset($_POST["clear"])){
    session_start(); // start the session
    $_SESSION = array(); //reset the session array to clear the data from any variables
    session_destroy(); // then delete the session data from the server
}
?>
```

Sessions

```
<?php
    session_start(); // start the session
    //On loading the page, check if session has a value
    if(isset($_SESSION["username"])){
        echo 'Welcome ' . $_SESSION["username"];
    }else{
        echo 'Not sure who you are ?';
    }
?>
```

JavaScript & jQuery

- Incorporating jQuery and JavaScript into a Website using PHP is often necessary.
- Web pages will include JavaScript and jQuery which offer a range of functionality implemented by developers.
- When submitting a form for example, you can specify an event in JavaScript which handles the submission first, then subsequently the form is handled by PHP.

JavaScript & jQuery

- In such an scenario, it may not be necessary to invoke the PHP after handling the event in JavaScript. For example, if the event performed validation on the data and determined it was invalid.
- Use **return false** in client side script if form should not be submitted to server side script.

AJAX

- An example:
- JavaScript hijacks a form submission, and invokes a php script. The php script returns data to client side JavaScript and this data is processed to display it on the web page.
- AJAX request made in the usual manner with url specifying a php file.
- PHP file runs and responds. Could simply use an echo 'OK' and the JavaScript can check for `response == "OK"` and update the web page based on the result of that condition. Note: response would be specified as a parameter of the anonymous function – for success property in jQuery or event handler for response in JavaScript.

Not Covered

- The following will not be covered as they are beyond the scope of the module, though you may encounter these as part of the lab exercises without realising it:
 - PHP Error handling
 - PHP Events
 - OOP
- **Note:** If interested in finding out more on these then please ask your lecturer.