



Lesson #9

PHP

What is php?



- PHP (Personal Home Page) is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output.
- It can be deployed on most web servers and on almost every operating system and platform free of charge.

php Syntax



- PHP only parses code within its delimiters. Anything outside its delimiters is sent directly to the output and is not parsed by PHP. The most common delimiters are `<?php` and `?>`, which are the open and close delimiters respectively. The purpose of these delimiters is to separate PHP code from non-PHP code, including HTML.
- Variables are prefixed with a dollar symbol and a type does not need to be specified in advance.

php Syntax



- Unlike function and class names, variable names are case sensitive. Double-quoted (") strings allow the ability to embed a variable's value into the string.
- PHP treats newlines as white space and statements are terminated by a semicolon.
- PHP has three types of comment syntax: /* */ serves as block comments, and // as well as # are used for inline comments.

php Data Types



- PHP stores whole numbers in a platform-dependent range. This range is typically that of 32-bit signed integers.
- Real numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation. PHP has a native Boolean type that is similar to the ones found in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl and C++.

php Data to Browser



- Data is sent to the browser with the `echo()` and `print()` statements.

```
echo 'Hello world';  
print "How are you?";
```

- New lines are added using the `
` tag, not the `\n` character.

```
echo "Words<br />on<br />different<br />lines.";
```

- Single quotes use the string literally and double quotes evaluate the string (like Perl).

php Strings



- Strings are enclosed in single or double quotes. They can be assigned to variables.

```
$city = 'Toronto';  
print "Welcome to <i>$city</i>.";
```

- Strings are concatenated with the period operator (.).

```
$twocities = $city.' '.$city;
```

php Operators



- The basic arithmetic operators are identical to those of Perl: `+`, `-`, `*`, `/`, `%`, `++`, `--`
- `round()` rounds a decimal to the nearest integer or to a specified number of decimal places.

```
$n=3.1428; n=round($n); echo $n
```

```
3
```

```
$n=3.1428; n=round($n,3); echo $n
```

```
3.143
```

How to save your 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. Instead of index.html, it should be index.php if there is PHP code in the file.

```
<body>
```

```
<?php
```

```
echo "Hello World!";
```

```
?>
```

```
</body>
```

Here doc strings in php

- PHP introduces a more robust string creation tool called heredoc that lets the programmer create multi-line strings without using quotations.

```
$my_string = <<<ABC
```

ABC is the heredoc identifier to open the string

This string can be defined on multiple lines.

```
ABC;
```

to close the string, the identifier must be on a line of its own and not be indented

```
echo $my_string;
```

This string can be defined on multiple lines.

php Include



- Include takes a file name and simply inserts that file's contents into the script that issued the include command.
- You can type up a common header or menu file that you want all your web pages to include. When you add a new page to your site, instead of having to update the links on several web pages, you can simply change the included file.
- `<?php include("header.php"); ?>`

php require



- Just like include, the require command is used to include a file into your php code. However there is one huge difference between the two commands.
- If the file to be included is missing, include will display the rest of the script but require will kill the remaining of the php script.
- `<?php require("missing.php");`
- `echo "Hello there!"; ?>`

Hello there! will not be displayed if the file missing.php is... missing

php - if statement



- The if statement is the same as in Perl and other programming languages.

```
$number = 7;  
if ( $number == 3 )  
    echo "true";  
else  
    echo "false";
```

false

php - elseif statement



- Great for nested ifs.

```
$food = "spinach";  
if ($food == "steak")  
    echo "meat";  
elseif ($food == "spinach")  
    echo "vegetable";  
else  
    echo "something else";
```

vegetable

php - switch statement

- Great for multiple conditions:

```
$destination = "Tokyo";  
echo "Traveling to $destination<br />";  
switch ($destination){  
    case "Las Vegas":  
        echo "Bring an extra $500"; break;  
    case "Tokyo":  
        echo "Bring lots of money"; break;  
    case "Caribbean Islands":  
        echo "Bring a swimsuit"; break;  
    default:  
        echo "Bring your ID";  
}
```

Traveling to Tokyo
Bring lots of money

php - using forms



- Same as Perl. The HTML part:

```
<form action="process.php" method="post" >
<select name="item" >
<option>Paint</option>
<option>Brushes</option>
<option>Erasers</option>
</select>
Quantity: <input name="quantity" type="text" />
<input type="submit" />
</form >
```

php - using forms



- The process.php script:

```
<?php
```

```
$quantity = $_POST['quantity'];
```

```
$item = $_POST['item'];
```

```
echo "You ordered ". $quantity . " " . $item . ".<br />";
```

```
echo "Thank you for ordering from us.";
```

```
?>
```

php: form data



- In Perl we used two techniques to remember data from form to form, hidden fields and cookies. In php we have the same plus sessions. Since hidden fields are html only, we will have a quick look at cookies and sessions.
- When you create a cookie, using the function setcookie, you must specify three arguments. These arguments are setcookie(name, value, expiration).

php - setting cookies



- name: The name of your cookie. You will use this name to later retrieve your cookie.
- value: The value that is stored in your cookie. Common values are username(string) and last visit(date).
- expiration: The date when the cookie will expire and be deleted. If you do not set this expiration date, then it will be treated as a session cookie and be removed when the browser is restarted.

```
<?php
```

```
$inTwoMonths = 60 * 60 * 24 * 60 + time();
```

```
setcookie('lastVisit', date("G:i - m/d/y"),  
    $inTwoMonths);
```

```
?>
```

php - retrieving cookies



- If your cookie hasn't expired yet, let's retrieve it using the `$_COOKIE` associative array. The name of your stored cookie is the key and will let you retrieve your stored cookie value.

```
<?php
if(isset($_COOKIE['lastVisit']))
{
    $visit = $_COOKIE['lastVisit'];
    echo "Your last visit was - ". $visit;
else
    echo "Cookie not set or expired";
?>
```



php - sessions



- Sessions work by creating a unique identification (UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same web page.
- Before you can begin storing user information in your PHP session, you must first start the session. When you start a session, it must be at the very beginning of your code, before any HTML or text is sent.

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

- *This tiny piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.*

php - session data



- When you want to store user data in a session use the `$_SESSION` associative array. This is where you both store and retrieve session data.

```
<?php
session_start();
$_SESSION['views'] = 1; // store session data
echo "Pageviews = " . $_SESSION['views'];
//retrieve data
?>
```

Pageviews = 1

php - session data



- With our previous example, we can create a very simple pageview counter by using `isset` to check if the pageview variable has already been created. If it has we can increment our counter. If it doesn't exist we can create a pageview counter and set it to one.

```
<?php
session_start();
if(isset($_SESSION['views']))
    $_SESSION['views'] = $_SESSION['views']+ 1;
else
    $_SESSION['views'] = 1;
echo "views = ". $_SESSION['views'];
?>
views = 2
```

php - functions



- A typical php function:

```
<?php
```

```
function mySum($numX, $numY) {
```

```
    $total = $numX + $numY;
```

```
    return $total;
```

```
}
```

```
$myNumber = 0;
```

```
echo "Before the function, myNumber = ". $myNumber ."<br />";
```

```
$myNumber = mySum(3, 4); // Store the result of mySum in  
    $myNumber
```

```
echo "After the function, myNumber = " . $myNumber ."<br />";
```

```
?>
```

```
Before the function, myNumber = 0
```

```
After the function, myNumber = 7
```

php - arrays



- Arrays in php are very simple:

```
$employee_array[0] = "Bob";  
$employee_array[1] = "Jinhee";  
$employee_array[2] = "Ali";  
$employee_array[3] = "Maria";  
echo "Two of my employees are "  
. $employee_array[0] . " & " . $employee_array[1];  
echo "<br />Two more employees of mine are "  
. $employee_array[2] . " & " . $employee_array[3];
```

Two of my employees are Bob & Jinhee

Two more employees of mine are Ali & Maria

php - associative arrays



- In an associative array a key is associated with a value.

```
$salaries["Bob"] = 3500;  
$salaries["Jinhee"] = 4000;  
$salaries["Ali"] = 4500;  
$salaries["Maria"] = 4550;  
echo "Bob is being paid - $" . $salaries["Bob"] . "<br />";  
echo "Jinhee is being paid - $" . $salaries["Jinhee"] . "<br />";  
echo "Ali is being paid - $" . $salaries["Ali"] . "<br />";  
echo "Maria is being paid - $" . $salaries["Maria"];
```

Bob is being paid - \$3500

Jinhee is being paid - \$4000

Ali is being paid - \$4500

Maria is being paid - \$4550

php - loops (while)



```
$brush_price = 5;
$counter = 10;
echo "<table border=\"1\" align=\"center\">";
echo "<tr><th>Quantity</th>";
echo "<th>Price</th></tr>";
while ( $counter <= 100 ) {
    echo "<tr><td>";
    echo $counter;
    echo "</td><td>";
    echo $brush_price * $counter;
    echo "</td></tr>";
    $counter = $counter + 10;
}
echo "</table>";
```

Quantity	Price
10	50
20	100
30	150
40	200
50	250
60	300
70	350
80	400
90	450
100	500

php - loops (for)



```
$brush_price = 5;
echo "<table border=\"1\" align=\"center\">";
echo "<tr><th>Quantity</th>";
echo "<th>Price</th></tr>";
for ( $counter = 10; $counter <= 100; $counter += 10) {
    echo "<tr><td>";
    echo $counter;
    echo "</td><td>";
    echo $brush_price * $counter;
    echo "</td></tr>";
}
echo "</table>";
```

Quantity	Price
10	50
20	100
30	150
40	200
50	250
60	300
70	350
80	400
90	450
100	500

php - loops (foreach)



```
$employeeAges["Bob"] = "28";  
$employeeAges["Jinhee"] = "26";  
$employeeAges["Ali"] = "35";  
$employeeAges["Maria"] = "46";  
$employeeAges["Lois"] = "34";
```

```
Name: Bob, Age: 28  
Name: Jinhee, Age: 26  
Name: Ali, Age: 35  
Name: Maria, Age: 46  
Name: Lois, Age: 34
```

```
foreach( $employeeAges as $key => $value){  
    echo "Name: $key, Age: $value <br />";  
}
```

/ this will give the same output */*

```
foreach( $employeeAges as $name => $age){  
    echo "Name: $name, Age: $age <br />";  
}
```

php - loops (do while)

- Do while loops are executed at least once.

```
$cookies = 10;  
do {  
    echo "I eat a cookie - ";  
    $cookies --;  
    echo "$cookies left.< br />";  
} while ($cookies > 1);
```

```
I eat a cookie - 9 left.  
I eat a cookie - 8 left.  
I eat a cookie - 7 left.  
I eat a cookie - 6 left.  
I eat a cookie - 5 left.  
I eat a cookie - 4 left.  
I eat a cookie - 3 left.  
I eat a cookie - 2 left.  
I eat a cookie - 1 left.
```

php - how to create a file



- The fopen function needs two important pieces of information to operate correctly. First, we must supply it with the name of the file that we want it to open. Secondly, we must tell the function what we plan on doing with that file (i.e. read from the file, write information, etc).

```
$ourFileName = "testFile.txt";  
$ourFileHandle = fopen($ourFileName, 'w') or  
    die("can't open file");  
fclose($ourFileHandle);
```

php - file opening modes



- 'r' (read): Open a file for read only use. The file pointer begins at the start of the file.
- 'w' (write): Open a file for write only use. In addition, the data in the file is erased and you will begin writing data at the beginning of the file. The file pointer begins at the start of the file.
- 'a' (append): Open a file for write only use. However, the data in the file is preserved and you will be writing data at the end of the file. The file pointer begins at the end of the file.
- 'r+' (read/write): Opens a file so that it can be read from and written to. The file pointer is at the beginning of the file.
- 'w+' (write/read): This is exactly the same as r+, except that it deletes all information in the file when the file is opened.

php - file writing (fwrite)



- The fwrite function allows data to be written to any type of file. Fwrite's first parameter is the file handle and its second parameter is the string of data that is to be written.

```
$myFile = "testFile.txt";  
$fh = fopen($myFile, 'w') or die("can't open file");  
$stringData = "Bobby Bopper\n";  
fwrite($fh, $stringData);  
$stringData = "Tracy Tanner\n";  
fwrite($fh, $stringData);  
fclose($fh);
```

php - file reading (fread)



- The fread function is the staple for getting data out of a file. The function requires a file handle, which we have, and an integer to tell the function how much data, in bytes, it is supposed to read. One character is equal to one byte.

```
$myFile = "testFile.txt";  
$fh = fopen($myFile, 'r');  
$theData = fread($fh, 5);  
fclose($fh);  
echo $theData;
```

- If you wanted to read all the data from the file, then you need to get the size of the file. The filesize function returns the length of a file, in bytes. The filesize function requires the name of the file that is to be sized up.

```
$theData = fread($fh, filesize($myFile));
```

php - file reading (gets)



- PHP also lets you read a line of data at a time from a file with the gets function.

```
$myFile = "testFile.txt";  
$fh = fopen($myFile, 'r');  
$theData = fgets($fh);  
fclose($fh);  
echo $theData;
```

will display the first line of the data file

php - file deleting (unlink)



- In PHP you delete files by calling the unlink function. Before you can delete (unlink) a file, you must first be sure that it is not open in your program. Use the fclose function to close down an open file.

```
$myFile = "testFile.txt";  
unlink($myFile);
```

- *This will destroy the testFile.txt file on the server's disk. You have to be careful with that one.*

php - file upload



- A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server opens a whole can of worms, so please be careful when enabling file uploads.
- Before you can use PHP to manage your uploads, you must first build an HTML form that lets users select a file to upload.

```
<form enctype="multipart/form-data" action="uploader.php"
  method="POST">
```

```
<input type="hidden" name="MAX_FILE_SIZE" value="100000" />
```

```
Choose a file to upload: <input name="uploadedfile" type="file"
  /><br />
```

```
<input type="submit" value="Upload File" />
```

```
</form>
```

A screenshot of a web form for file uploading. It features a text label 'Choose a file to upload:' followed by a text input field and a 'Browse...' button. Below the input field is an 'Upload File' button.

```
Choose a file to upload:  

```

php - file upload



- After the user clicks submit, the data will be posted to the server and the user will be redirected to uploader.php. This PHP file is going to process the form data and do all the work.
- Typically, the PHP file should make a key decision with all uploads: keep the file or throw it away. A file might be thrown away from many reasons, including
 - *The file is too large and you do not want to have it on your server*
 - *You wanted the person to upload a picture and they uploaded something else, like an executable file (.exe).*
 - *There were problems uploading the file and so you can't keep it.*
- When the uploader.php file is executed, the uploaded file exists in a temporary storage area on the server. If the file is not moved to a different location it will be destroyed. To save our file we are going to need to make use of the `$_FILES` associative array.

php - file upload



- The `$_FILES` array is where PHP stores all the information about files.
- * `uploadedfile` - `uploadedfile` is the reference we assigned in our HTML form. We will need this to tell the `$_FILES` array which file we want to play around with.
- * `$_FILES['uploadedfile']['name']` - `name` contains the original path of the user uploaded file.
- * `$_FILES['uploadedfile']['tmp_name']` - `tmp_name` contains the path to the temporary file that resides on the server. The file should exist on the server in a temporary directory with a temporary name.

```
$target_path = "uploads/"; /* destination folder */
```

```
$target_path = $target_path . basename( $_FILES['uploadedfile']  
['name']); /* complete path - folder+file name */
```

php - file upload



- Now all we have to do is call the `move_uploaded_file` function and let PHP do its magic.

```
if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'],
    $target_path)) {
    echo "The file ". basename( $_FILES['uploadedfile']['name']).
    " has been uploaded";
} else{
    echo "There was an error uploading the file, please try again!";
}
```

- Danger! Do not place this script on a web page viewable to the public.

php - strings (strpos)



- The way strpos works is it takes some string you want to search in as its first argument and another string, which is what you are actually searching for, as the second argument. If the function can find a search match, then it will return the position of the first match. However, if it can't find a match it will return false.

```
$numberedString = "12345678901234567890";  
$fivePos = strpos($numberedString, "5");  
echo "The position of 5 in our string was $fivePos";  
$fivePos2 = strpos($numberedString, "5", $fivePos + 1);  
echo "<br />The position of the second 5 was $fivePos2";  
The position of 5 in our string was 4  
The position of the second 5 was 14
```

php - strings (*str_replace*)



- The `str_replace` function is similar to a word processor's *Replace All* command that lets you specify a word and what to replace it with, then replaces every occurrence of that word in the document. You can also use arrays for multiple replacements.

```
$rawstring = "Welcome parents. Your placeholder is a pleasure to  
have!";  
  
$male = str_replace("placeholder", "son", $rawstring);  
$female = str_replace("placeholder", "daughter", $rawstring);  
echo "Son: ". $male . "<br />";  
echo "Daughter: ". $female;
```

Son: Welcome parents. Your son is a pleasure to have!

Daughter: Welcome parents. Your daughter is a pleasure to have!

php strings (substr_replace)



- There are three required parameters for the `substr_replace` function (original string, replacement string, starting point) and one that's optional (length). A negative starting point specifies the number of characters from the end of the string.

```
$original = "ABC123 Life is Good DEF321";  
$clean1 = substr_replace($original, "", 0, 6);  
$clean2 = substr_replace($clean1, "", -6, 6);  
echo "Original: $original <br />";  
echo "Clean #1: $clean1 <br />";  
echo "Clean #2: $clean2";
```

Original: ABC123 Life is Good DEF321

Clean #1: Life is Good DEF321

Clean #2: Life is Good

php strings (explode)



- The first argument that explode takes is the delimiter (our dynamite) which is used to blow up the second argument, the original string. explode returns an array of string pieces from the original and they are numbered in order, starting from 0.

```
$rawPhoneNumber = "800-555-5555";  
$phoneChunks = explode("-", $rawPhoneNumber);  
echo "Raw Phone Number = $rawPhoneNumber <br />";  
echo "First chunk = $phoneChunks[0]<br />";  
echo "Second chunk = $phoneChunks[1]<br />";  
echo "Third chunk = $phoneChunks[2]";  
Raw Phone Number = 800-555-5555  
First chunk = 800  
Second chunk = 555  
Third chunk = 5555
```

php strings (implode)



- The PHP function implode operates on an array and is known as the "undo" function of explode. If you have used explode to break up a string into chunks or just have an array of stuff you can use implode to put them all into one string.

```
$pieces = array("Hello", "World,", "I", "am", "Here!");  
$gluedTogetherSpaces = implode(" ", $pieces);  
$gluedTogetherDashes = implode("-", $pieces);  
echo "Glued with Spaces = $gluedTogetherSpaces <br />";  
echo "Glued with Dashes = $gluedTogetherDashes";
```

Glued with Spaces = Hello World, I am Here!

Glued with Dashes = Hello-World,-I-am-Here!

php date



- The date function uses letters of the alphabet to represent various parts of a typical date and time format. Here are the basics:
- d: The day of the month. The type of output you can expect is 01 through 31.
- m: The current month, as a number. You can expect 01 through 12.
- y: The current year in two digits ##. You can expect 00 through 99

```
<?php  
echo date("m/d/y");  
?>
```

php date: time options



- a: am or pm depending on the time
- A: AM or PM depending on the time
- g: Hour without leading zeroes. Values are 1 through 12.
- G: Hour in 24-hour format without leading zeroes. Values are 0 through 23.
- h: Hour with leading zeroes. Values 01 through 12.
- H: Hour in 24-hour format with leading zeroes. Values 00 through 23.
- i: Minute with leading zeroes. Values 00 through 59.
- s: Seconds with leading zeroes. Values 00 through 59.

php date: day options



- d: Day of the month with leading zeroes. Values are 01 through 31.
- j: Day of the month without leading zeroes. Values 1 through 31
- D: Day of the week abbreviations. Sun through Sat
- l: Day of the week. Values Sunday through Saturday
- w: Day of the week without leading zeroes. Values 0 through 6.
- z: Day of the year without leading zeroes. Values 0 through 365.

php date: other options



- m: Month number with leading zeroes. Values 01 through 12
- n: Month number without leading zeroes. Values 1 through 12
- M: Abbreviation for the month. Values Jan through Dec
- F: Normal month representation. Values January through December.
- t: The number of days in the month. Values 28 through 31.
- L: 1 if it's a leap year and 0 if it isn't.
- Y: A four digit year format
- y: A two digit year format. Values 00 through 99.
- O: represents the Timezone offset, which is the difference from Greenwich Meridian Time (GMT). 100 = 1 hour, -600 = -6 hours

php - security issues



- Whenever you allow your users to submit text to your website, you need to be careful that you don't leave any security holes open for malicious users to exploit. If you are ever going to allow user submitted text to be visible by the public you should consider using removing the script tags to prevent them from running html code and scripts that may be harmful to your visitors.

```
$userinput = "I am going to hack your site, ha ha ha!  
<script type='text/javascript'>  
window.location = 'http://www.ryerson.ca/'  
</script>";  
$cleaninput = str_replace("script", "skript", $userinput);  
echo $cleaninput;
```





end of lesson