

Lecture 2: Intro to PHP

- What is PHP?
- Language developed by Rasmus Lerdorf from the Apache Group
- Its primary use is for **server-side scripting**
 - Ex: To process HTML forms
 - Ex: To perform a DB query and pass on results
 - Ex: To dynamically generate HTML
- PHP scripts are often embedded within HTML documents
 - The server processes the HTML document, executing the PHP segments and substituting the output within the HTML document

Lecture 2: Intro to PHP

- The modified document is then sent to the client
- As mentioned previously, **the client never sees the PHP code**
 - The only reason the client even knows PHP is involved is due to the file extension → .php
 - But even this is not required if the server is configured correctly
 - The server can be configured to run PHP scripts even if the file does not have a .php extension
 - By default XAMPP will only execute PHP if the .php extension is provided
 - Can change this with an .htaccess file
 - See what happens if you have PHP code without the .php extension

Lecture 2: Intro to PHP

- PHP is a **HUGE** language
- It is a fully functional language
- It has an incredible amount of built-in features
 - Form processing
 - Output / generate various types of data (not just text)
 - Database access
 - Allows for various DBs and DB formats
 - Object-oriented features
 - Somewhat of a loose hybrid of C++ and Java
 - Huge function / class library

Lecture 2: Intro to PHP

- We will look at only a small part of PHP
- There are also many tools that are already pre-written in / for PHP
 - If you are building a substantial project you may want to use some of these
 - There are also **content management systems** written in PHP
 - Ex: <http://drupal.org/>
 - Ex: http://wordpress.org Ex: <http://wordpress.org/>
 - However, we may not be covering them here
 - We will focus on the straight PHP language

- **PHP Program Structure**
- Or really lack thereof
- PHP, as with many scripting languages, does not have nearly the same structural requirements as a language like Java
- A script can be just a few lines of code or a very large, structured program with classes and objects
 - The complexity depends on the task at hand
- However, there are some guidelines for incorporating PHP scripts into HTML files

Lecture 2: Intro. to PHP

- When a PHP file is requested, the PHP interpreter parses the entire file
 - Any content **within PHP delimiter tags is interpreted**, and the output substituted
 - Any **other content** (i.e. not within PHP delimiter tags) is simply **passed on unchanged**
 - This allows us to easily mix PHP and other content (ex: HTML)
 - See:
 - <http://us3.php.net/manual/en/language.basic-syntax.phptags.php>
 - <http://us3.php.net⁶/manual/en/language.basic->

Lecture 2: Intro to PHP

- Consider the following PHP file

```
<!DOCTYPE html>
```

```
<html>
```

HTML 5 Document

Root HTML Tag

```
<head>
```

```
<title>Simple PHP Example</title>
```

```
</head>
```

```
<body>
```

```
<?php echo "<p><h1>Output</h1>";  
        echo "<h2>Output</h2>";  
        echo "<h3>Output</h3></p>";
```

```
?>
```

```
<script language="PHP">
```

```
    echo "\n<b>More PHP Output</b>\n";  
    echo "New line in source but not rendered";  
    echo "<br/>";  
    echo "New line rendered but not in source";
```

```
</script>
```

```
</body>
```

```
</html>
```

Document Head

PHP Code

D
O
C
B
O
D
Y

Lecture 2: Intro. to PHP

- Now consider the resulting HTML

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Simple PHP Example</title>
```

```
  </head>
```

```
  <body>
```

```
    <p><h1>Output</h1><h2>Output</h2><h3>Output</h3></p>
```

```
  <b>More PHP Output</b>
```

```
New line in source but not rendered<br/>New line rendered but not in  
source </body>
```

```
</html>
```

- How will it look in the browser?
 - Look at it in the browser!
 - See ex2.php

Lecture 2: Intro to PHP

- If we prefer to separate a PHP code segment from the rest of our script, we can write it in another file and **include** it
 - Sometimes it is easier if we "separate" the PHP code from the straight html
 - We also may be using several different files, esp. if we are using classes
 - But we must tag it even if it is already within a PHP tagged segment
 - Included files are not interpreted by default
 - > Don't necessarily have to be PHP
 - > If we want PHP, include PHP tags within the included file look up more about includes!

See ex2.php

Lecture 2: Intro to PHP

- Simple types
 - See: <http://us3.php.net/manual/en/http://us3.php.net/manual/en/language.types.php>
- boolean
 - TRUE or FALSE
- integer
 - Platform dependent – size of one machine word
 - typically 32 or 64 bits
- float
 - Double precision
 - We could call it a double, but since we don't declare variables float works

Lecture 2: Intro to PHP

- string
 - We have **single-quoted** and **double-quoted** string literals
 - Double quoted allows for more escape sequences and allows variables to be **interpolated into the string**
 - What does that mean?
 - > Rather than outputting the name of the variable, we output its contents, even within a quote
 - > We'll see an example once we define variables
 - > Note that this is NOT done in Java
 - > See example
 - Length can be arbitrary
 - Grows as necessary

Lecture 2: Intro to PHP

- Easy conversion back and forth between strings and numbers
 - In Web applications these are mixed a lot, so PHP will **implicitly cast between the types when appropriate**
 - This is another clear difference between PHP and Java
 - > **Java requires explicit casting**
 - > **PHP allows explicit casting if desired**
 - See: <http://us3.php.net/manual/en/language.types.type-juggling.php>
- Can be indexed – the preferred way is using square brackets
 - `$mystring = "hello";`
 - `echo $mystring[1];`
 - Output here is 'e'

Lecture 2: Intro to PHP

- **PHP variables**
- All PHP variables begin with the \$
 - Variable names can begin with an underscore
 - Otherwise rules are similar to most other languages
- Variables are **dynamically typed**
 - No type declarations
 - Variables are *BOUND* or *UNBOUND*
 - > Unbound variables have the value NULL
 - Type information for a variable is obtained from the current bound value
 - Compare this to Java

Lecture 2: Intro. to PHP

- Implications of **dynamic typing**:
 - **No “type clash” errors like in Java**

```
int x = 3.5;    // oh no!  
String s = 100; // argh!
```


– **Instead we have in PHP**

```
$x = 3.5;      // no problem!  
$s = 100;     // a-ok!
```
 - **A variable’s type may change throughout program execution**

```
$x = 5;        // integer  
$x = $x + 1.5; // float  
$x = $x . " dollars"; // string
```

Lecture 2: Intro. to PHP

- Perhaps intentionally but perhaps by mistake
- We have to be careful and to test types during execution
 - `gettype()` function returns a string representation of variable's type

```
$x = 5; echo(gettype($x)); // integer
$x = $x + 1.5; echo (gettype($x)); // float
$x = $x . " dollars"; echo(gettype($x)); // string
```
 - `is_<type>()` function returns a boolean to test for a given <type>

```
$x = 5;      $check = is_int($x); // true
           $check = is_float($x); // false
```
 - Can use these tests to make decisions within a script

Lecture 2: Intro. to PHP

- Why is PHP dynamically typed?
- Allows for faster interpreting of the code
 - Compiled code will run faster than interpreted code, but compiling itself takes time
- Allows for easier / simpler templates / generic code
 - Think about generics in Java and how much syntax they require
 - Much of this is due to type checking
- For more information see:
 - http://en.wikipedia.org/wiki/Type_system

Lecture 2: Intro to PHP

- PHP programs have access to a large number of **predefined variables**
 - These variables allow the script access to server information, form parameters, environment information, etc.
 - Very helpful (as we will see) for determining and maintaining state information
 - Ex:
 - `$_SERVER` is an array containing much information about the server
 - `$_POST` is an array containing variables passed to a script via HTTP POST
 - `$_COOKIE` is an array containing cookies
 - See `ex5.php`