INTRODUCTION TO PHP

Chapter 9

Note

- Examples for this chapter are at

https://swe.umbc.edu/~zzaidi1/is448/chap9-examples

- PHP programs, are server-side programs that cannot be seen in the browser by doing a 'View Source' in the browser.
- All PHP programs discussed in this class are zipped up in the examples folder. Download the zip file, php1.zip to see the actual PHP code

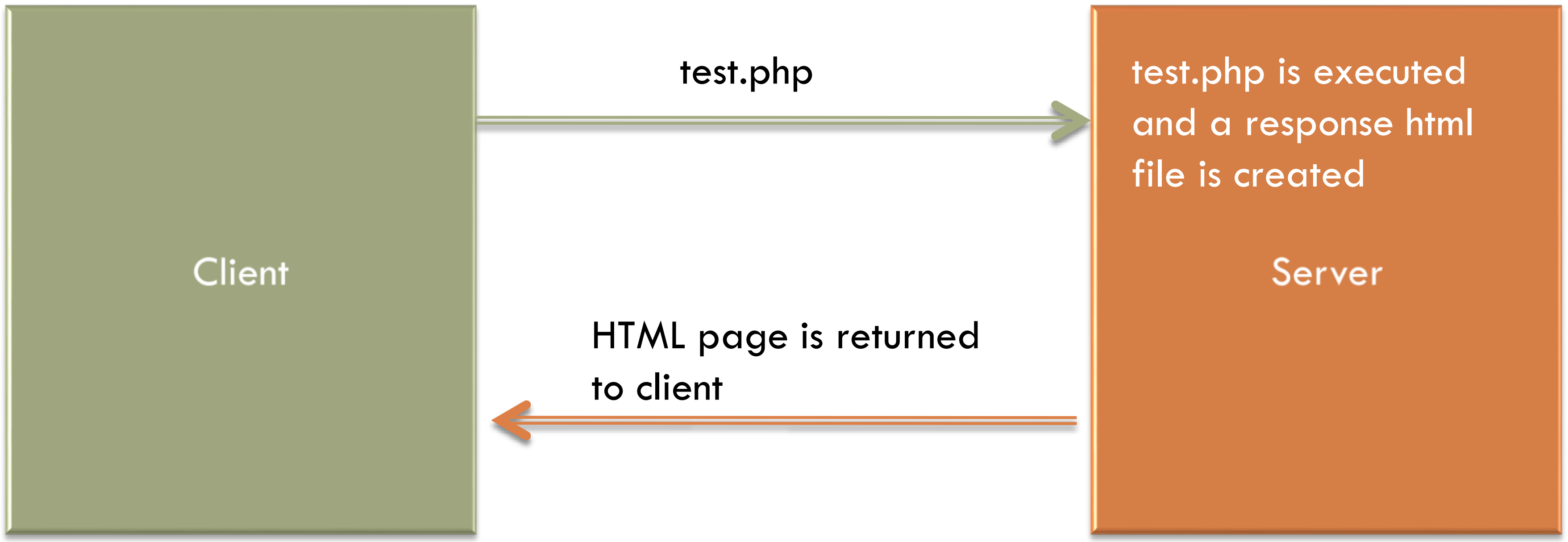
Origin and Uses of PHP

- Developed by Rasmus Lerdorf in 1994
- PHP
  - is a server-side scripting language, embedded in XHTML pages
  - has good support for form processing
  - can interface with a wide variety of databases
  - stands for PHP Hypertext Processor

Why server-side programming?

- JavaScript already allows us to create dynamic, programmable web pages. Why use a server-side language instead of JavaScript?
- **security**:
  - server-side code has access to server's important and/or private data
  - client can't see your source code
- **compatibility**: avoids browser JavaScript compatibility issues
- **efficiency**: faster for users
  - don't have to run a script to view each page
  - don't have to send entire data set from server to user's browser
- **power**: fewer restrictions (can write to files, open connections to other servers, connect to databases, ...)

Overview of PHP


test.php
test.php is executed and a response html file is created
Client
Server
HTML page is returned to client

- When a PHP document is requested of a server, the server will send the document first to a PHP processor
- The result of the processing is the response to the request

Overview of PHP

- PHP processor has two modes of operation
  - **Copy mode** in which plain HTML in the input file is copied to the output file
  - **Interpret mode** in which PHP code in the input file is interpreted and the output from that code sent to output file
    - Means that Output of PHP script must be XHTML or embedded client script
  - This new output file is sent to the requesting browser

- The client never sees PHP code, the client (i.e., the user) only sees the output produced by the PHP code

Overview of PHP

- PHP has typical scripting language characteristics
  - Dynamic typing, untyped variables
  - Associative arrays
  - Pattern matching
  - Extensive libraries

General Syntactic Characteristics

- PHP code is contained between the tags
  <?php and **?>**
- Code can be included with the PHP include
  `include("table2.inc");`
- When a file is included, the PHP interpreter reverts to copy mode
  - Thus, PHP code in an include file must also be in <?php and ?> tags

General Syntactic Characteristics

- A basic PHP scripting block

```
<?php
    your PHP code
?>
```

- Example: See helloworld.php
  - A simple PHP program that sends the string “Hello World” to the browser window,

Output

- Two basic output statements
  - print
  - echo
- Both functions
  - take string parameters
  - used to send data to output
- You can optionally surround each string in parenthesis

```
<?php
    print “text”;
    echo “hello world”;
    print “<a href=\“test.html\”>Test <\/a>”;
?>
```

```
<?php
    print (“text”);
    echo (“hello world”);
?>
```

PHP Syntax

- PHP statements are terminated with semicolons
- Curly braces are used to create compound statements
- One line comments can begin with # or // and continue to the end of the line
- Multi-line comments can begin with /* and end with */

Variables

- All *variable* names in PHP begin with the dollar symbol $
- Case sensitivity
  - Variable names are case sensitive
  - Keywords and function names are not case sensitive
- Always implicitly declared by assignment
  - Type not specified
- User-defined variables
- Special reserved variables
- Example: See variables.php

```
$username = “John”;
$age = 14;
$driving_age = $age +  2;
```

11.4 Primitives, Operations, Expressions

- Four scalar data types
  - boolean, integer, double, string
- Two compound data types
  - array, object
- Test what type a variable is with *is_type* functions, e.g. **is_string**
- PHP converts between types automatically in many cases:
  - string $\longrightarrow$ int auto-conversion on +
  - int $\longrightarrow$ float auto-conversion on /
- Can also type-cast with (*type*): **$age = *(int)*"21";**

11.4 Numeric Types

- PHP distinguishes between integer and floating point numeric types
- Integer is equivalent to long in C, that is, usually 32 bits
- Double literals can include decimal point, exponent or both

11.4 String Type

- String literals are enclosed in single or double quotes
  - Double quoted strings have escape sequences interpreted and variables interpolated
  - Single quoted strings have neither escape sequence interpretation nor variable interpolation
  - A literal $ sign in a double quoted string must be escaped with a backslash, \
- Double-quoted strings can cover multiple lines, the included end of line characters are part of the string value
- Example: See strings.php

Strings

- Double quotes vs. single quotes

```
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n"; # this line prints: You are 16 years old.
print 'You are $age years old.\n'; # this line prints: You are $age years old.\n`
```

11.4 String Operations

- String catenation is indicated with a period (.)
- Characters are accessed in a string with a subscript enclosed in curly braces
- Many useful string functions are provided
  - **strlen** gives the length of a string
  - **strcmp** compares two strings as strings
  - **chop** removes whitespace from the end of a string
- Example: See strings.php

```
<?php
$str1 = 'This ';
$str2 = 'is ';
$str3 = 'IS448';

$full = $str1.$str2.$str3;

echo $full;
?>
```

11.4 Boolean Type

- The boolean type has two values :TRUE and FALSE

- Other type values are coerced as needed by context, for example, in control expressions
  - The integer value 0, the empty string and the literal string "0" all count as false
  - NULL counts as false
  - The double value 0.0 counts as false. Beware, however, that double calculations rarely result in the exact value 0.0

11.4 Scalar Type Conversions

- Implicit type conversions as demanded by the context in which an expression appears
  - A string is converted to an integer if a numeric value is required and the string has only a sign followed by digits
  - A string is converted to a double if a numeric value is required and the string is a valid double literal (including either a period or e or E)
- Type conversions can be forced in three ways
  - `(int)$sum` in the C style
  - `intval($sum)` using several conversion functions
  - `settype($x, “integer”)`
- Type can be determined with the `gettype` function and with the `is_int` function and similar functions for other types

11.4 Arithmetic Operators and Expressions

- PHP supports the usual operators supported by the C/C++/Java family
- *Integer* divided by *integer* results in *integer* value if there is no remainder but results in *double* value if there is a remainder
  - 12/6 is 2
  - 12/5 is 2.4
- A variety of numeric functions available: floor, ceil, round, srand, abs, min, max

11.4 Assignment Operators

- The assignment operators used in C/C++/Java are supported in PHP

  =

- And compound operators

  e.g., +=, =+, -=, =-

11.6 Relational Operators

- PHP has the usual comparison operators: >, < <=, >=, == and !=
- PHP also has the identity operator ===
  - This operator does not force coercion
- The regular comparisons will force conversion of values as needed
  - Comparing a string with a number (other than with ===) will result in the string converting to a number if it can be. Otherwise the number is converted to a string
  - If two strings are compared (other than with ===) and the strings can both be converted to numeric values, the conversion will be done and the converted values compared
  - Use `strcmp` on the strings if the latter feature is a problem

11.6 Boolean Operators

- PHP supports
  - and, or, &&, ||, !, xor

- and and or are lower-precedence than && and || provided

Example using pre-defined functions

- PHP has an online manual
  - http://php.net/manual/en/langref.php
- PHP has an extensive collection of pre-defined functions
  - http://us2.php.net/manual/en/funcref.php
- Example of using a pre-defined function:
  - See **`today.php`**
  - uses the date function to dynamically generate a page with the current date
  - Other parameters to use with date function are described here:

    http://php.net/manual/en/function.date.php

11.6 Selection Statements

- PHP provides an `if` statement with almost the same syntax as C/C++/Java
  - The only difference is the `elseif`
- Example: See ifstmt.php

11.6 Selection Statements

- The `switch` statement is provided with syntax and semantics similar to C/C++/Java
  - *case* expression may be any expression that evaluates to a simple type, that is, integer or floating-point numbers and strings
  - `break` is necessary to prevent execution from flowing from one case to the next
- Example: See switch.php

Example

- Change HTML title and content based on value of a PHP variable and an if-statement
- Example: See **`change_title.php`**

11.6 Loop Statements

- PHP provides the **`while`** and **`for`** and **`do-while`** as in JavaScript
- The **`for loop`** is illustrated in the example **`powers.php`**
- This example also illustrates a number of mathematical functions available in PHP

Lab

- Start with the file lab1.php from the zip file for this class (php1.zip)
- Add PHP code to lab1.php to change the background of this page based on the day of the week
  - Hint 1: Use if-else statements and the date PHP function
  - Hint 2: Refer to change_title.php to see how to dynamically change HTML page content with PHP
  - Hint 3: you can make your PHP code change a CSS property also.