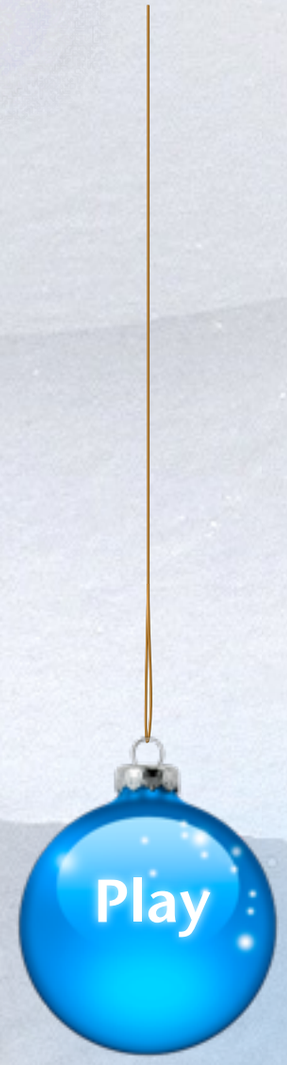
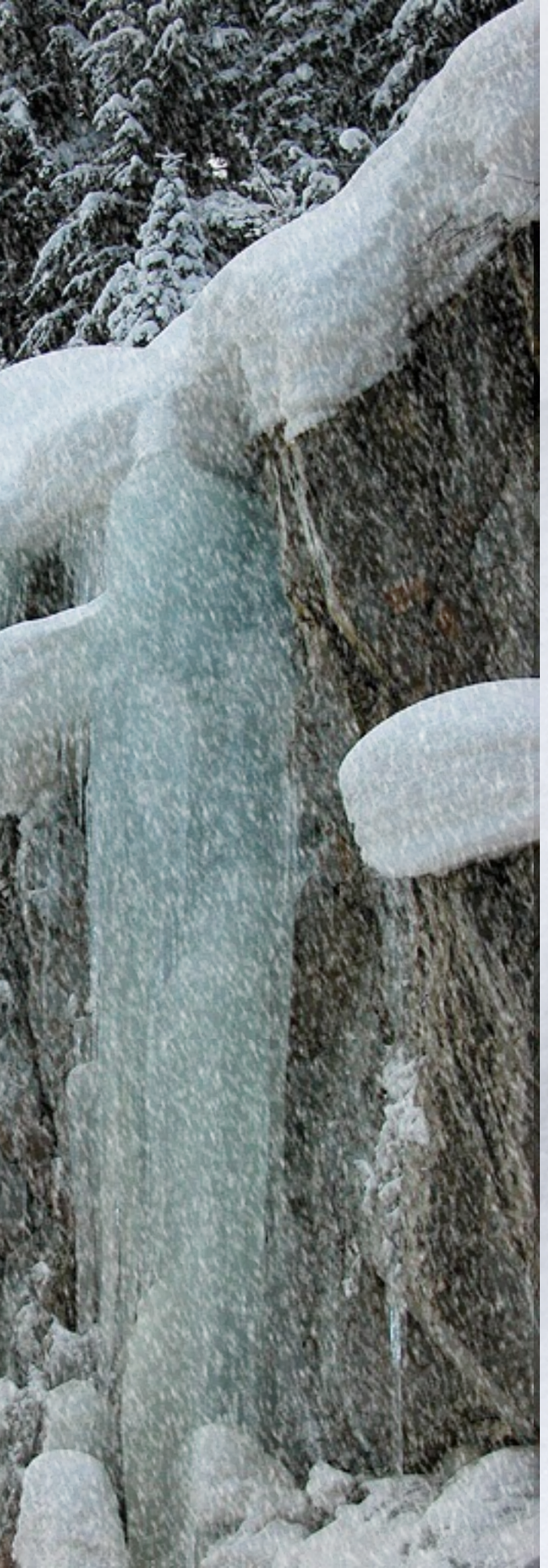
Lecture 7

Basic PHP for Server Side Programming





Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP

Dynamic Vs. Static

- Static Page
  - Client/Consumer's Viewpoint: an url refers to an identical html file
  - Server/Producer's Viewpoint: a file stored within or sub-within the root folder of a Web Server
  - it is a html ...
  - Can be display directly at a browser

- Dynamic Page
  - Client/Consumer's Viewpoint: an url refers to a dynamic html (maybe vary each time requested)
  - Server/Producer's Viewpoint: a program/script produces html
  - it is NOT a html, but a program producing html(s)
  - Can't be display directly at a browser

- Dynamic Web Page, Dynamic HTML(DHTML), what's the difference?

Server-Side Web Programming

- server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl
- the web server contains software that allows it to run those programs and send back their output as responses to web requests
- each language/framework has its pros and cons
  - we use PHP for server-side programming in this textbook

The Options for Webserver Programming

- SSI: Simple interface for basic dynamic content.
  - Non-standard - read your server docs.
- CGI: The standardized, portable general-purpose API, not limited to working with HTML pages.
- Enhanced CGI-like: Typically gain efficiency but lose portability compared to standard CGI.
- ASP/ASP.net
- JSP/Servlets
- PHP/Python/Perl
- Ruby on Rails(RoR)

SSI – Server Side Includes

- Instructions to the server embedded in XHTML
- Server must be SSI enabled
- File extension of .shtml is usually used
- SSI instructions in HTML Comments

  <!-- This is a comment -->

- SSI instructions preceded by hash (#)

  <!--#include virtual=myfile.html-->

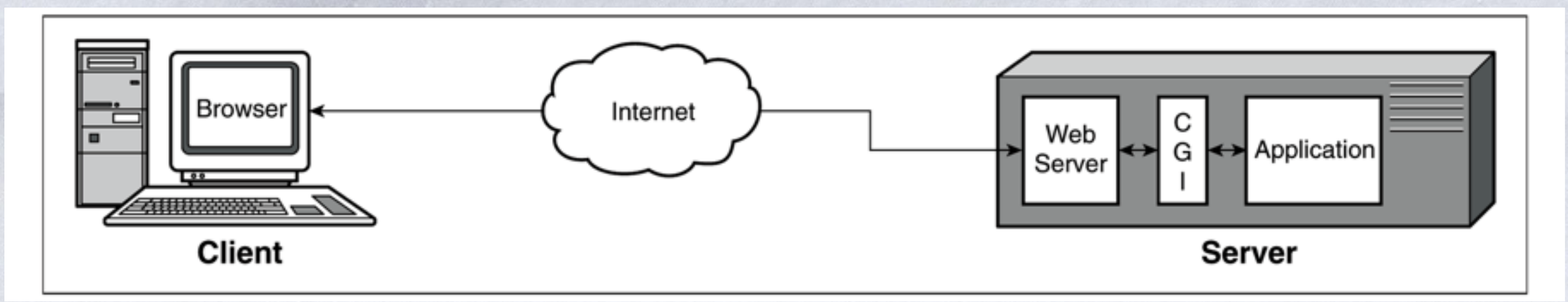
  - This includes the file “myfile.html” in the main XHTML file

Demo SSI code

```
<h2> Server Side Includes </h2>

<p> This is the main file (demoSSI.shtml) - everything between
the  two horizontal lines is held in an external file (table.html).

<p>

<hr />

<!--#include virtual="table.html"-->

<hr />

<p> This is the main file again </p>
```

Common Gateway Interface (CGI)

- CGI allows browsers to request the execution of server-resident software
- An HTTP request to run a CGI program specifies a program, rather than a document
  - Servers can recognize such requests in two ways:
    - By the location of the requested file (special subdirectories for such files)
  - A server can be configured to recognize executable files by their file name extensions



CGI

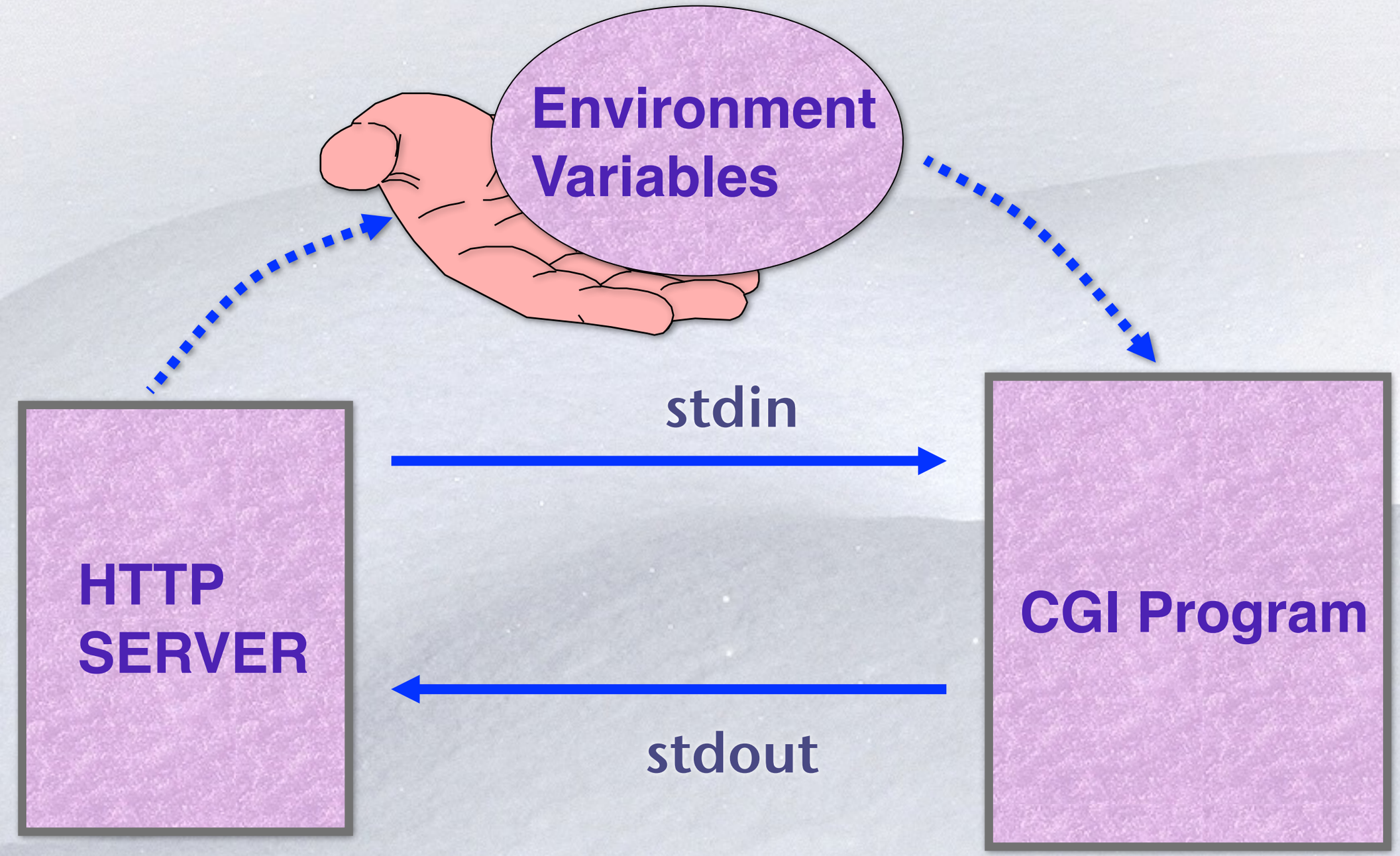
- A CGI program can produce a complete HTTP response, or just the URL of an existing document
- CGI programs often are stored in a directory named cgi-bin
- CGI programs may be written in any language
  - Most major languages have CGI libraries
  - PERL is the most commonly used language for CGI

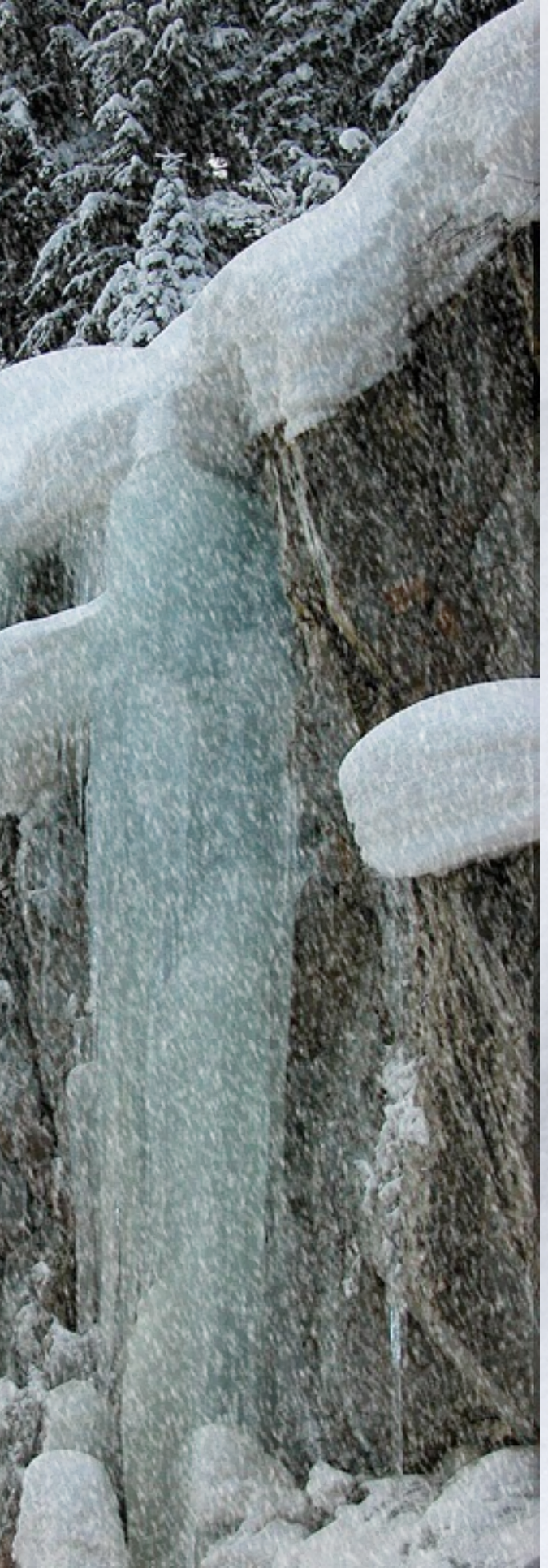
Request ⟶ CGI program

- The web server sets some environment variables with information about the request.
- The web server fork()s and the child process exec()s the CGI program.
- The CGI program gets information about the request from environment variables.

Important CGI Environment Variables

- REQUEST_METHOD
- QUERY_STRING
- CONTENT_LENGTH


Environment
Variables
stdin
HTTP
SERVER
CGI Program
stdout



Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP

What is PHP?

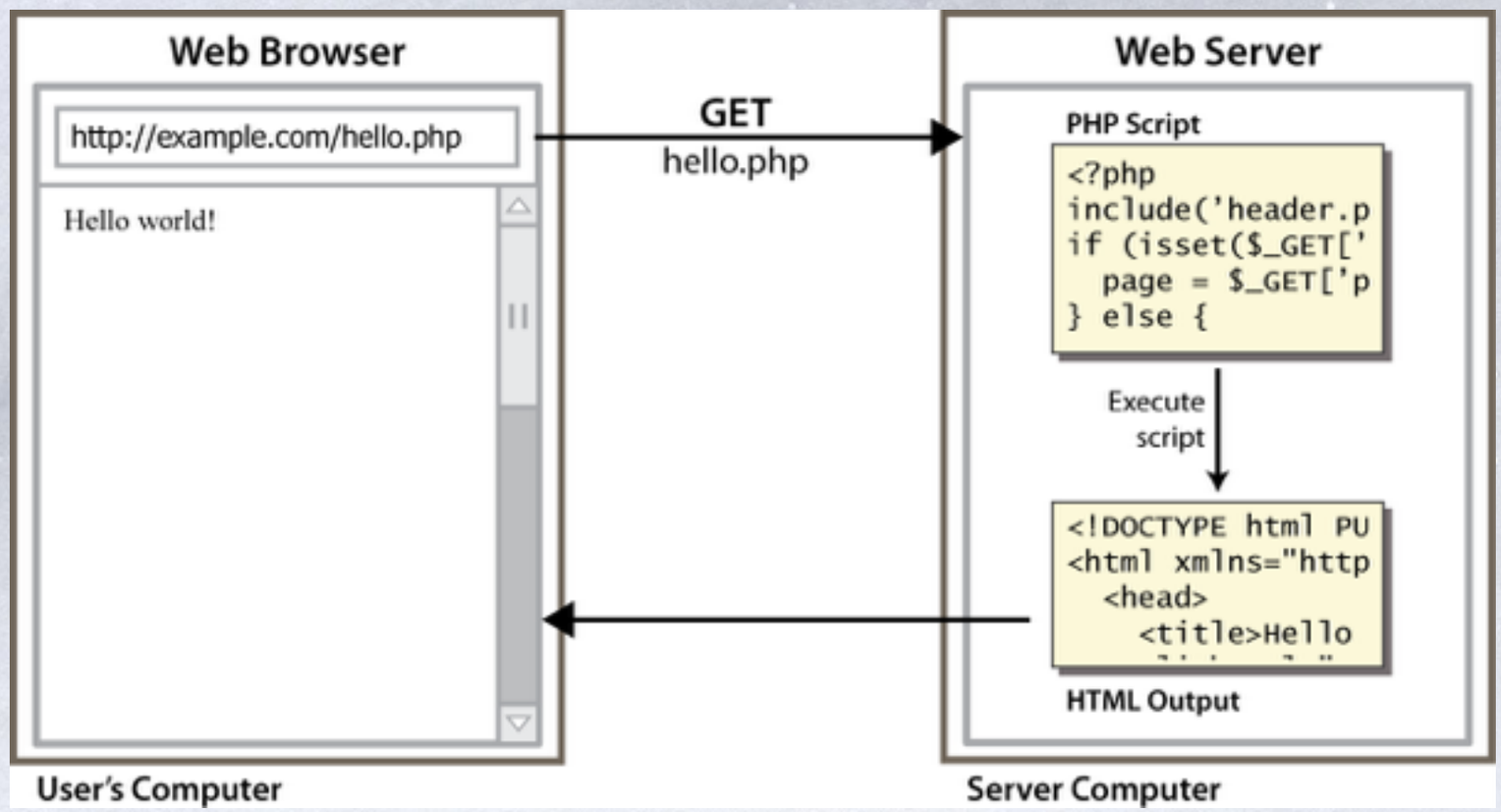
- PHP stands for "PHP Hypertext Preprocessor"
- a server-side scripting language
- used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc
  - authenticate users
  - process form information
- PHP code can be embedded in XHTML code

PHP 7.0.0 RC 3

- Improved performance: PHP 7 is up to twice as fast as PHP 5.6
- Consistent 64-bit support
- Many fatal errors are now Exceptions
- Removal of old and unsupported SAPIs and extensions
- The null coalescing operator (??)
- Combined comparison Operator (<=>)
- Return Type Declarations
- Scalar Type Declarations
- Anonymous Classes

Lifecyle of PHP Web request

- browser requests a .html file (static content): server just sends that file
- browser requests a .php file (dynamic content): server reads it, runs any script code inside it, then sends result across the network
  - script produces output that becomes the response sent back



Why PHP?

- There are many other options for server-side languages: Ruby on Rails, JSP, ASP.NET, etc. Why choose PHP?
  - free and open source: anyone can run a PHP-enabled server free of charge
  - compatible: supported by most popular web servers
  - simple: lots of built-in functionality; familiar syntax
  - available: installed on our servers and most commercial web hosts

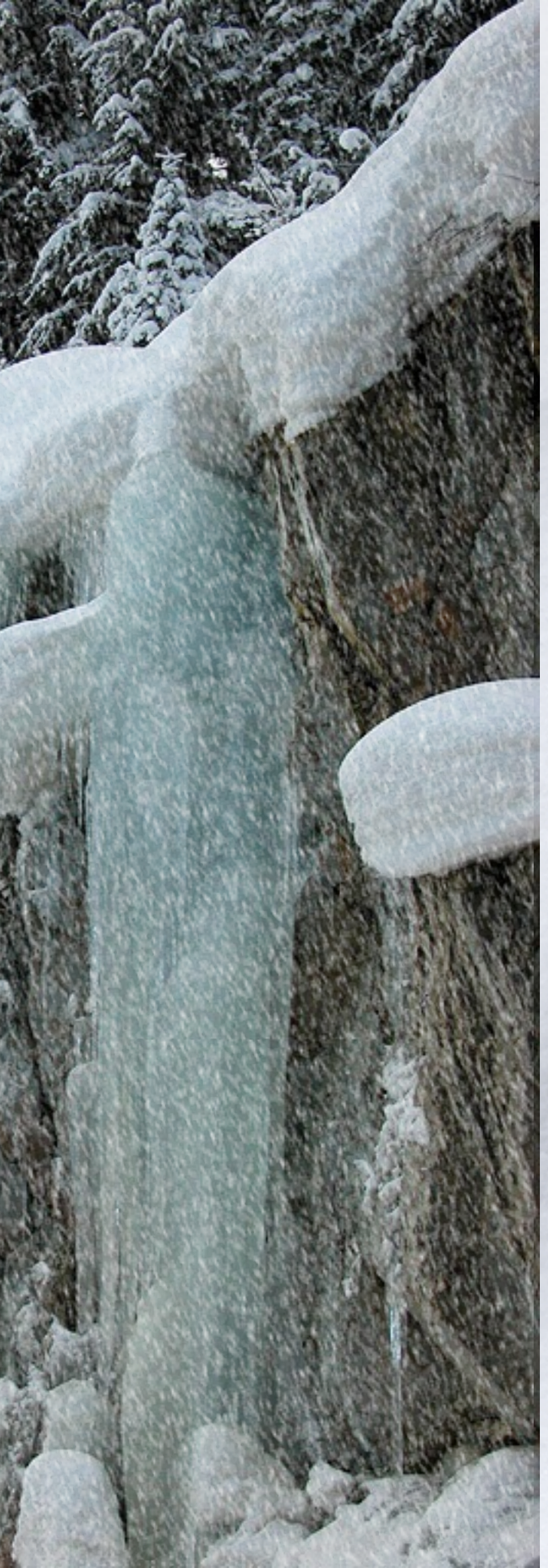
Server-side Programming Languages

Most popular server-side programming languages

© W3Techs.com	usage	change since 1 August 2015
1. PHP	81.4%	+0.1%
2. ASP.NET	16.5%	-0.2%
3. Java	3.0%	
4. static files	1.6%	
5. ColdFusion	0.7%	

percentages of sites

http://w3techs.com



Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Comments in PHP

- like Java, but # is also allowed
- a lot of PHP code uses # comments instead of //

```
<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 1;
echo $x;
?>
```

PHP Case Sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- all variable names are case-sensitive.

PHP echo and print Statements

- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

```
echo "This ", "string ", "was ", "made ", "with multiple parameters.”;
print "Hello world!<br>";
```

Variables

- Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.
- a loosely typed language (like JavaScript or Python)
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case-sensitive ($age and $AGE are two different variables)

Variables Scope

- variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
    - A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function
  - global
    - A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.
    - The global keyword is used to access a global variable from within a function.
    - PHP also stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
  - static
    - Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when you first declare the variable

Examples

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
  //$GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
  static $xy = 0;
}

myTest();
echo $y; // outputs 15
```

```php
<?php
  $x = 5; // global scope

  function myTest() {
        $y = 5; // local scope
         // using x inside this function will generate an error
         echo "<p>Variable x inside function is: $x</p>”;
        echo "<p>Variable y inside function is: $y</p>";
        }
        myTest();

        echo "<p>Variable x outside function is: $x</p>”;
        // using y outside the function will generate an error
        echo "<p>Variable y outside function is: $y</p>";
?>
```

Data Types

- Basic types: Integer, Float, Boolean, String, Array, Object, NULL, Resource
  - test what type a variable is with is_type functions, e.g. is_string
  - gettype function returns a variable's type as a string (not often needed)
- PHP converts between types automatically in many cases:
  - string → int auto-conversion on +
  - int → float auto-conversion on /
- type-cast with (type): $age = (int)"21";

Integer,Float and Boolean

- An integer is a whole number (without decimals)
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- A float (floating point number) is a number with a decimal point or a number in exponential form.
- A Boolean represents two possible states: TRUE or FALSE
  - the following values are considered to be FALSE(all others are TRUE):
    - 0 and 0.0 (but NOT 0.00 or 0.000)
    - "", "0", and NULL(includes unset variables)
    - arrays with 0 elements

Arithmetic operators

- + -* / % . ++ --= += -= *= /= %= .=
- many operators auto-convert types:
  - 5 + "7" is 12

NULL

- A variable is NULL if
  - it has not been set to any value (undefined variables)
  - it has been assigned the constant NULL
  - it has been deleted using the unset function
- Can test if a variable is NULL using the isset function
- NULL prints as an empty string (no output)

```php
$name = "Victoria";
$name = NULL;
if (isset($name)) {
  print "This line isn't going to be reached.\n";
}
```

String type

- zero-based indexing using bracket notation
- string concatenation operator is .(period), not +
  - 5 + "2 turtle doves" === 7
  - 5 . "2 turtle doves" === "52 turtle doves"
- can be specified with " "or ' '

```php
$favorite_food = "Ethiopian";
print $favorite_food[2];                # h
```

Part of String Functions

str_replace()	Replaces some characters in a string (case-sensitive)
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string

Interpreted strings

- strings inside " " are interpreted variables that appear inside them will have their values inserted into the string

```php
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n";     # You are 16 years old.
```

- strings inside ' ' are not interpreted:

```php
print 'You are $age years old.\n';     # You are $age years old.\n
```

Arrays

- element type is not specified; can mix types
- there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index
  - Associative arrays - Arrays with named keys
  - Multidimensional arrays - Arrays containing one or more arrays

```
$array = array();
$animals = array("Monkey", "Lion", "Turtle", “Horse");
$animals = array(2 => "Monkey", "Lion", "Turtle", “Horse");
$namesAndAges = array("John Doe" => 45, "Jane Doe" => 33, "Dog Doe" => 11);
```

Multidimensional arrays

```
$cars = array
  (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
  );
```

```
<?php
  echo $cars[0][0].": 库存：".$cars[0][1].", 销量：".$cars[0][2].".<br>";
  echo $cars[1][0].": 库存：".$cars[1][1].", 销量：".$cars[1][2].".<br>";
  echo $cars[2][0].": 库存：".$cars[2][1].", 销量：".$cars[2][2].".<br>";
  echo $cars[3][0].": 库存：".$cars[3][1].", 销量：".$cars[3][2].".<br>";
  ?>
```

Array functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Array function example

- the array in PHP replaces many other collections in Java list, stack, queue, set, map, ...

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                    # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);         # ("bh", "kk", "hm", "jp")
array_pop($tas);                     # ("bh", "kk", "hm")
array_push($tas, "ms");              # ("bh", "kk", "hm", "ms")
array_reverse($tas);                 # ("ms", "hm", "kk", "bh")
sort($tas);                          # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2);     # ("hm", "kk")
```

for/foreach Loop

Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
foreach ($array as $value) {
    code to be executed;
}
```

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

if/else statement

- if statement - executes some code only if a specified condition is true
- if...else statement - executes some code if a condition is true and another code if the condition is false
- if...elseif....else statement - specifies a new condition to test, if the first condition is false
- switch statement - selects one of many blocks of code to be executed

while loop (same as C)

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

- Break and continue keywords also behave as in Java and c

Functions

- Two types of functions
  - library functions: such as array_push are part of the PHP library and can be used by anyone
  - user functions: you may write your own functions and use them across your code.
- parameter types and return types are not written
- a function with no return statements implicitly returns NULL

```
function name(parameterName, ..., parameterName) {
  statements;
}
```
PHP

```
function quadratic($a, $b, $c) {
  return -$b + sqrt($b * $b - 4 * $a * $c) / (2 * $a);
}
```
PHP

Calling functions

- if the wrong number of parameters are passed, there's a warning

```
<?php
   function printVars($a, $b)
   {
      printf("%d%d\n", $a, $b);
   }
   printVars(1,2,3);
   printf("-----------\n");
   printVars(1);
?>
```

输出为：

```
12
-----------
PHP Warning:  Missing argument 2 for printVars(), called in moreArgs.php on line 10 and defined in moreArgs.php on line 3
PHP Notice:  Undefined variable: b in moreArgs.php on line 5
10
```

Default parameters values

- if no value is passed, the default will be used (defaults must come last)

```
function name(parameterName, ..., parameterName) {
  statements;
}
```
PHP

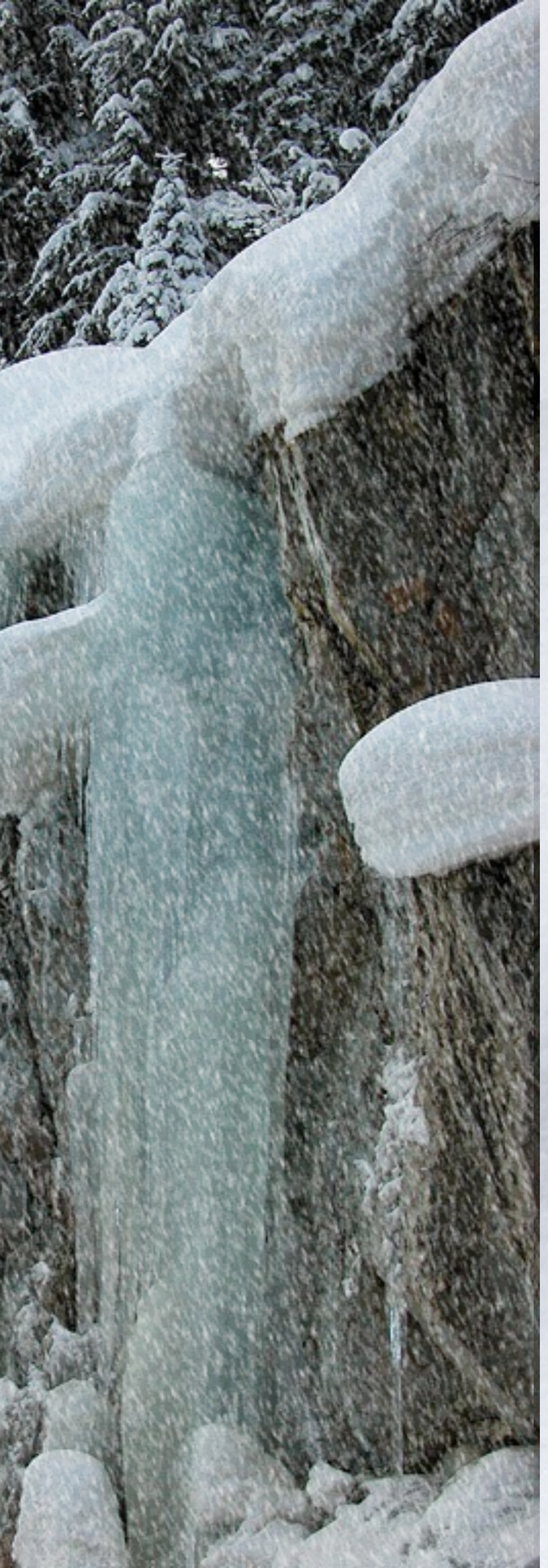
```
function print_separated($str, $separator = ", ") {
  if (strlen($str) > 0) {
    print $str[0];
    for ($i = 1; $i < strlen($str); $i++) {
      print $separator . $str[$i];
    }
  }
}
```
PHP

```
print_separated("hello");          # h, e, l, l, o
print_separated("hello", "-");     # h-e-l-l-o
```
PHP

PHP syntax template

- any contents of a .php file between <?php and ?>are executed as PHP code
- all other contents are output as pure HTML
- can switch back and forth between HTML and PHP "modes"

```
HTML content

   <?php
      PHP code
   ?>

HTML content

   <?php
      PHP code
   ?>

HTML content   . . .
```
PHP



Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP

Don’t print HTML tag in PHP

- printing HTML tags with print statements is bad style and error-prone:
  - must quote the HTML and escape special characters, e.g. \"
  - best PHP style is to minimize print/echo statements in embedded PHP code
- but without print, how do we insert dynamic content into the page?

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p> I can count to $i! </p>\n";
}
?>
```

Including files: include, require

- inserts the entire contents of the given file into the PHP script's output page
- encourages modularity
- useful for defining reused functions needed by multiple pages
- require is almost same as include, but different when the target was not found
  - the script with include will continue run with a warning message dumped to the output
  - the script with require will stop running and dump an error to the output

```php
include("filename");
```

```php
include("header.php");
```

PHP expression blocks

- PHP expression block: a small piece of PHP that evaluates and embeds an expression's value into HTML
  - <?=expression?>is equivalent to:

```
<?php print expression; ?>                                    PHP
```

- useful for embedding a small amount of PHP (a variable's or expression's value) in a large block of HTML without having to switch to "PHP-mode"

```
<?= expression ?>                                             PHP
```

```
<h2> The answer is <?= 6 * 7 ?> </h2>                         PHP
```

The answer is 42

output

Expression block example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: Embedded PHP</title></head>
  <body>
    <?php
    for ($i = 99; $i >= 1; $i--) {
      ?>
      <p> <?= $i ?> bottles of beer on the wall, <br />
          <?= $i ?> bottles of beer. <br />
          Take one down, pass it around, <br />
          <?= $i - 1 ?> bottles of beer on the wall. </p>
      <?php
    }
    ?>
  </body>
</html>
```

PHP

Common errors

```
...
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <? $i ?>
    </p>
  </body>
</html>
```

PHP

- </body> and </html> above are inside the for loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected $end'
- if you forget = in <?=, the expression does not produce any output

Complex expression blocks

- expression blocks can even go inside HTML tags and attributes

```php
...
  <body>
    <?php
    for ($i = 1; $i <= 3; $i++) {
      ?>
      <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
      <?php
    }
    ?>
  </body>
```

PHP

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

output

References

- www.w3schools.com

Thanks!!!