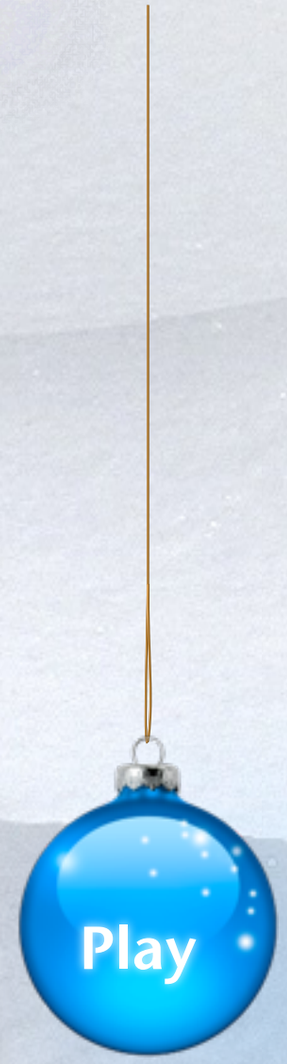
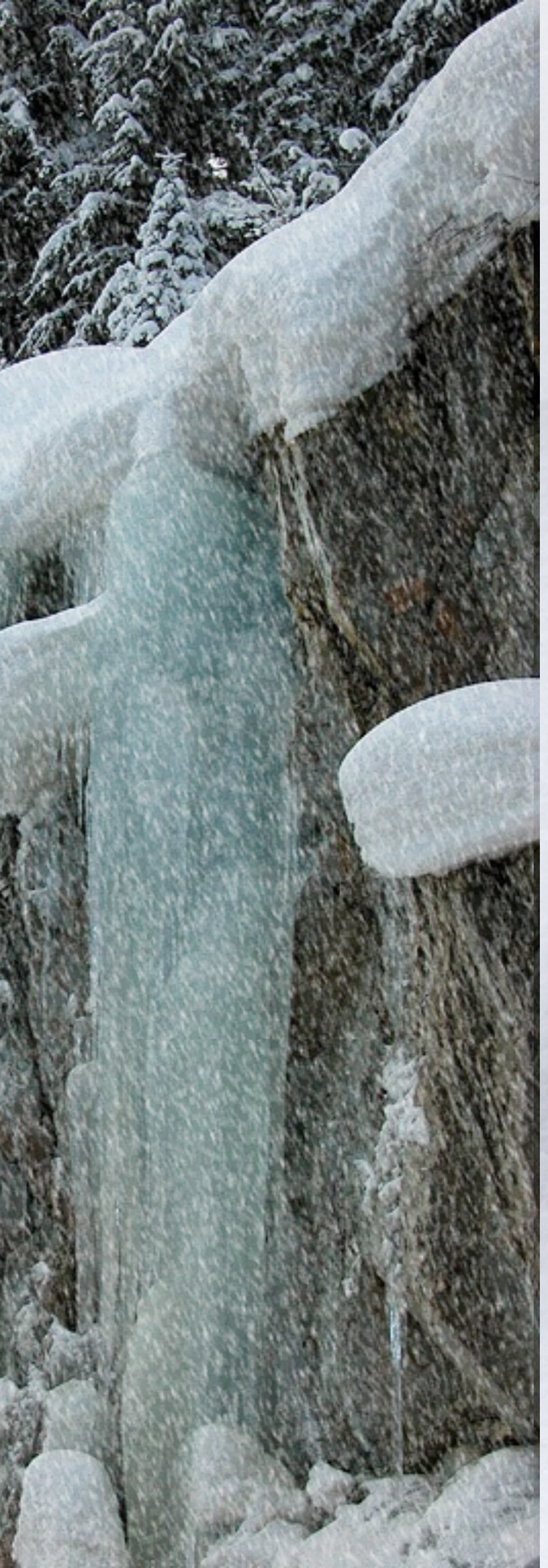
Lecture 7

Basic PHP for Server Side Programming





Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP
- Function & Scope

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-->

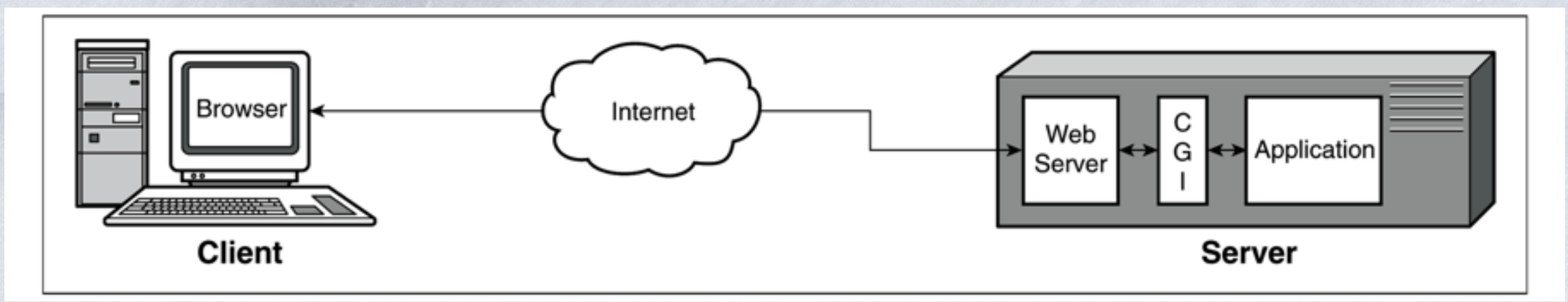
  - This includes the file “myfile.html” in the main XHTML file

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

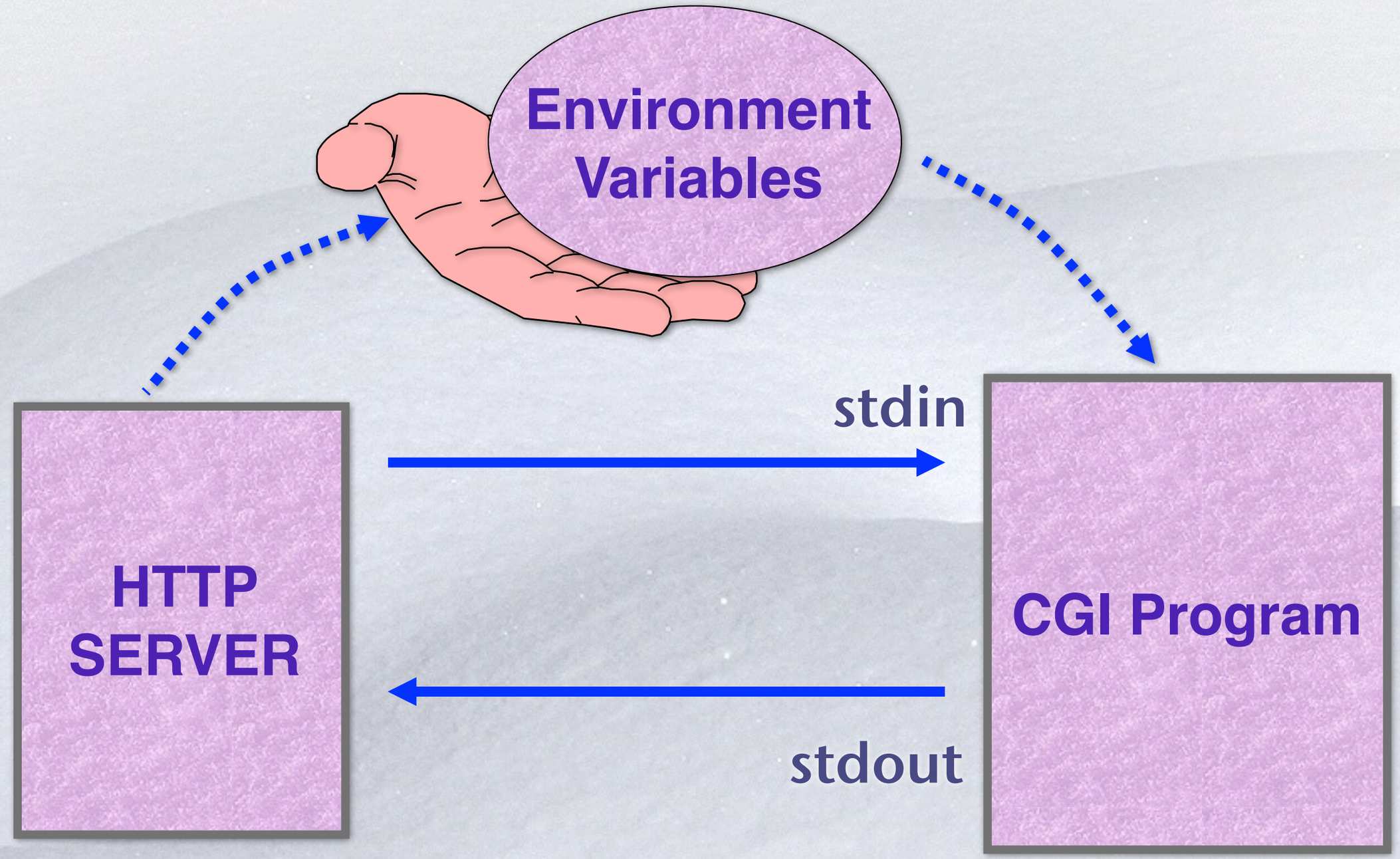
- A CGI program can produce a complete HTTP response, or just the URL of an existing document
- CGI programs often are stored in a directory named cgi-bin
- CGI programs may be written in any language
  - Most major languages have CGI libraries
  - PERL is the most commonly used language for CGI

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

JSP Big Picture

GET /hello.jsp

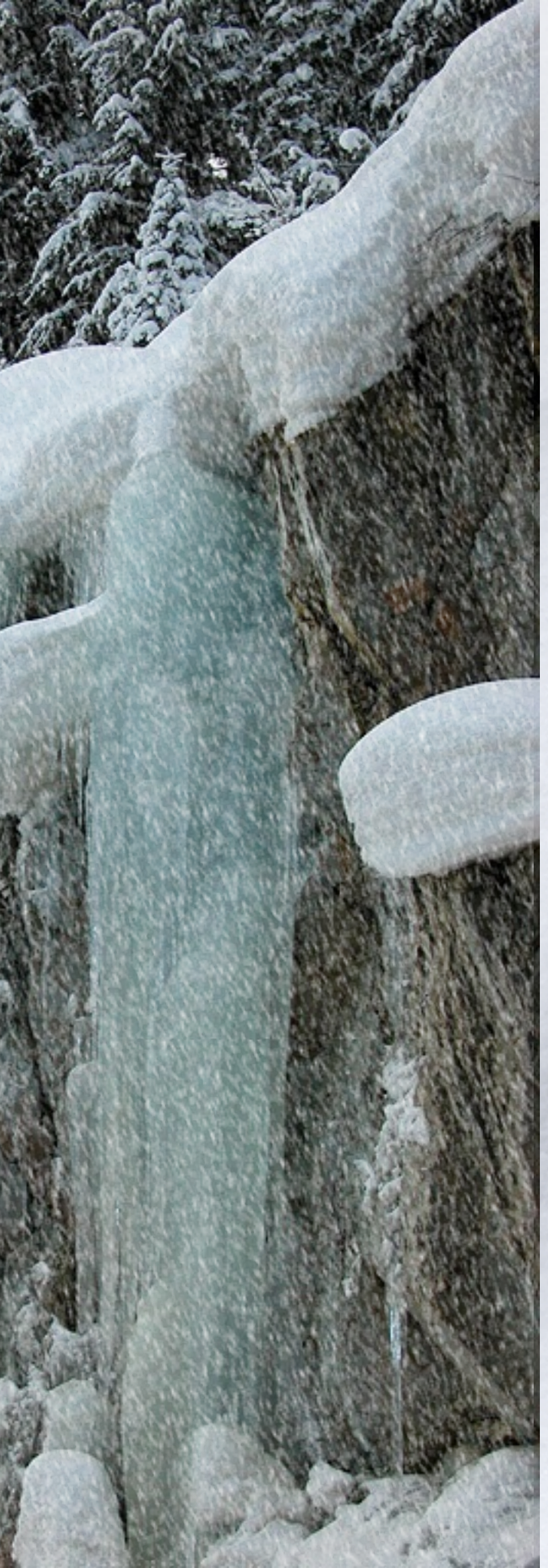
<html>Hello
!</html>

Server w/
JSP Container

Hello.jsp

HelloServlet.java

HelloServlet.class



Outline

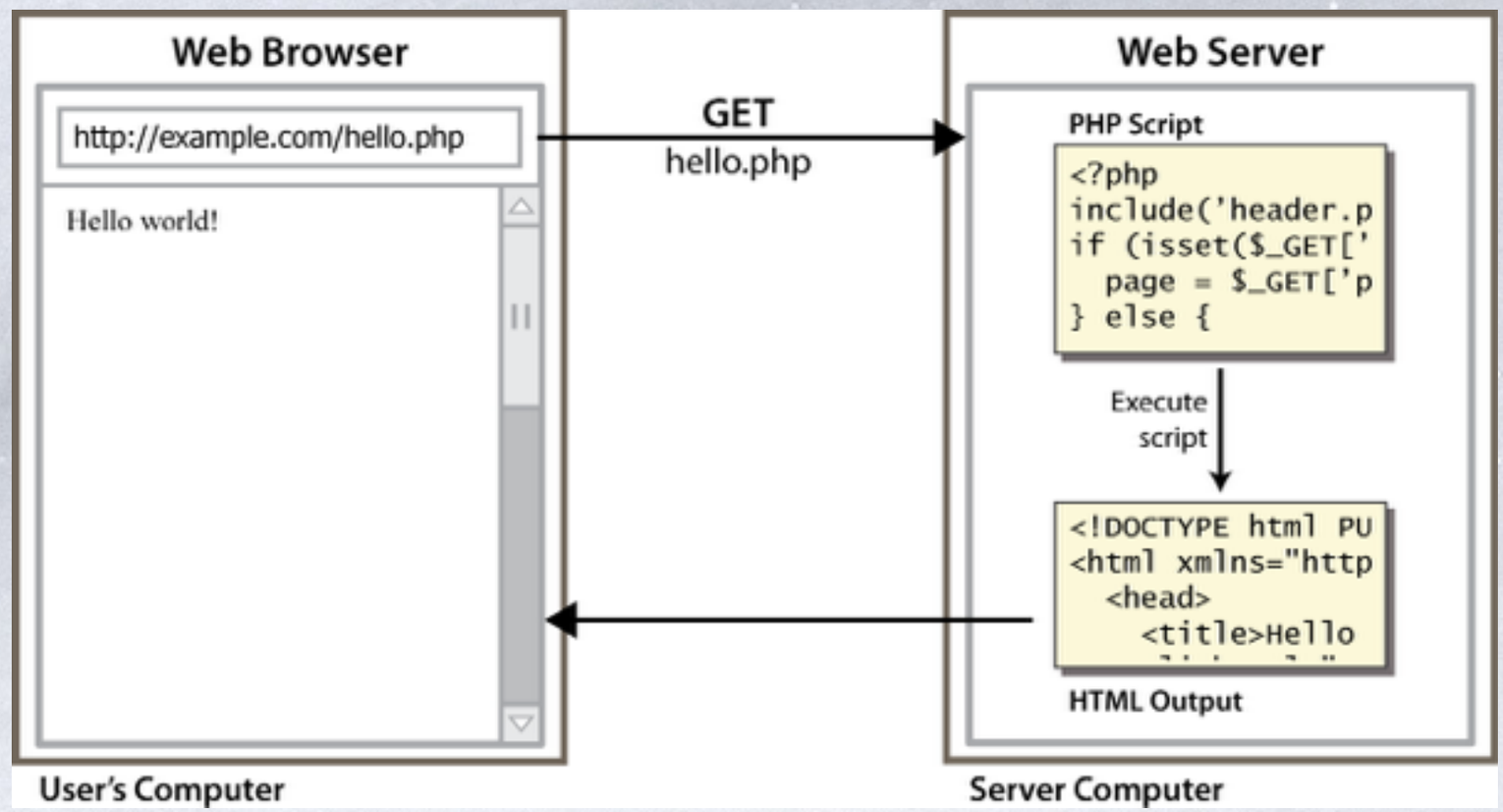
- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP
- Function & Scope

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

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

Hello, World!

- The following contents could go into a file hello.php:

```php
<?php
print "Hello, world!";
?>
```

PHP

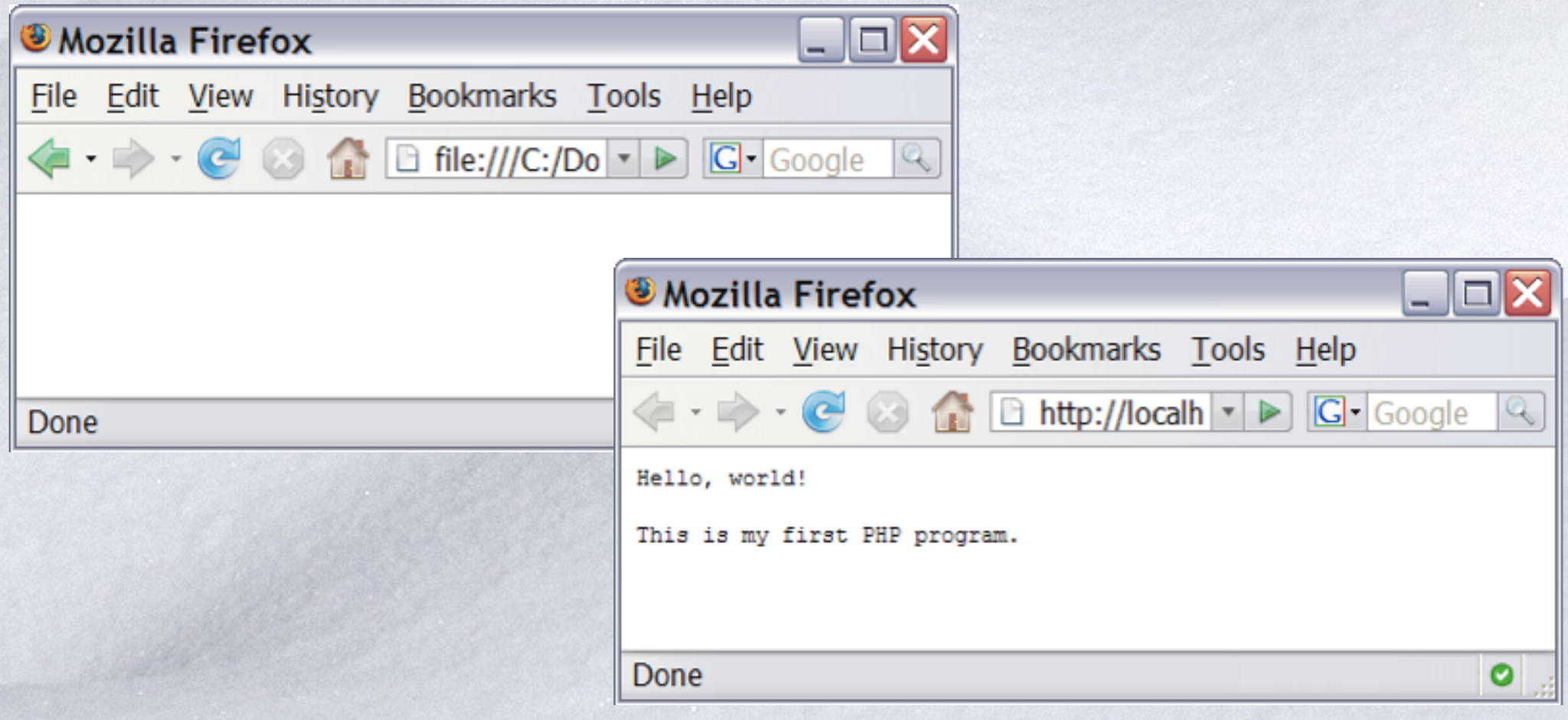
```
Hello, world!
```

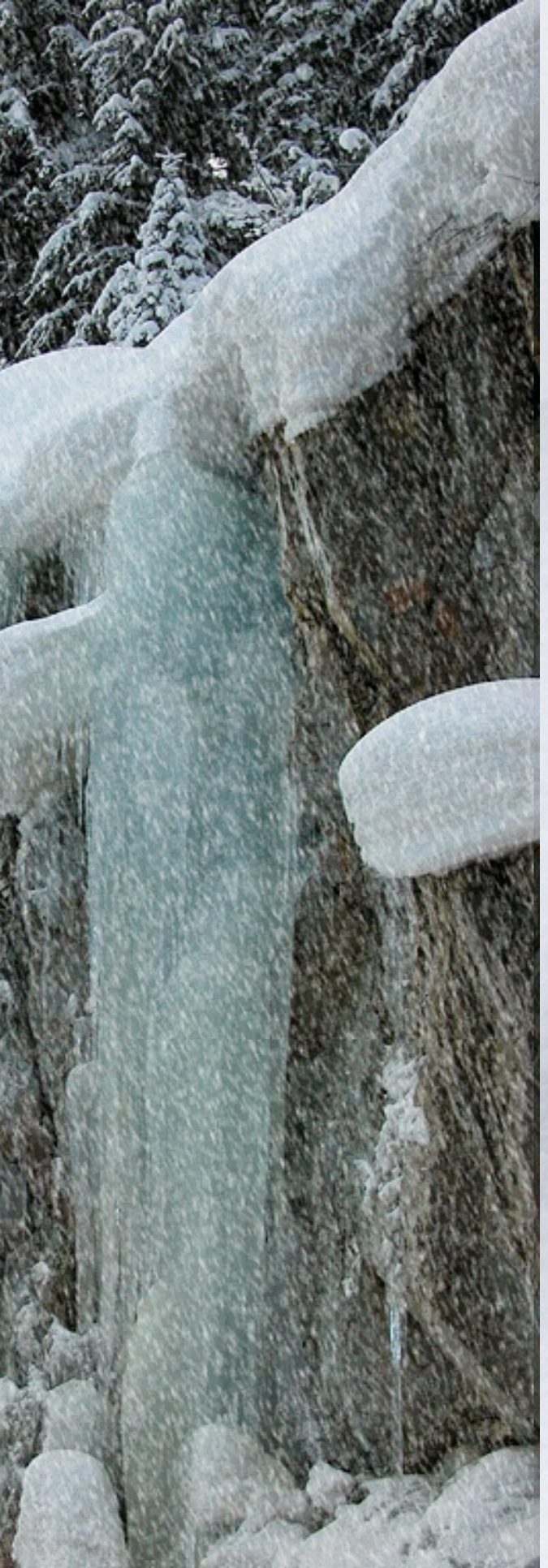
output

- a block or file of PHP code begins with <?php and ends with ?>
- PHP statements, function declarations, etc. appear between these endpoints

Viewing PHP output

- Your PHP code must be run/executed first, before it reaches a browser!





Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP
- Function & Scope

Comments

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

```php
#  single-line comment

//  single-line comment

/*
multi-line comment
*/
```

Console output: print

- some PHP programmers use the equivalent echo instead of print
  - arguments of echo vs. print

```php
print "text";
```

```php
print "Hello, World!\n";
print "Escape \"chars\" are the SAME as in Java!\n";

print "You can have
line breaks in a string.";

print 'A string can use "single-quotes".  It\'s cool!';
```

```
Hello, World! Escape "chars" are the SAME as in Java! You can have line breaks in a string. A string can use "single-quotes". It's cool!
```

output

Variables

- names are case sensitive; separate multiple words with _
- names always begin with $, on both declaration and usage
- always implicitly declared by assignment (type is not written)
- a loosely typed language (like JavaScript or Python)

```
$name = expression;                                                    PHP
```

```
$user_name = "PinkHeartLuvr78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;                                              PHP
```

Types

- basic types: int, 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";

int and float types

- int for integers and float for reals
- division between two int values can produce a float

```php
$a = 7 / 2;                    # float: 3.5
$b = (int) $a;                 # int: 3
$c = round($a);                # float: 4.0
$d = "123";                    # string: "123"
$e = (int) $d;                 # int: 123
```

Arithmetic operators

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

Bool (Boolean) type

- 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
- can cast to boolean using (bool) or (boolean)
- FALSE prints as an empty string (no output); TRUE prints as a 1

```php
$feels_like_summer = FALSE;
$php_is_rad = TRUE;

$student_count = 217;
$nonzero = (bool) $student_count;      # TRUE
```

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
```

```php
# index  0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);                # 16
$cmp = strcmp($name, "Brian Le");       # > 0
$index = strpos($name, "e");            # 2
$first = substr($name, 9, 5);           # "Hatch"
$name = strtoupper($name);              # "STEFANIE HATCHER"
```

Name	Java Equivalent
strlen	**length**
strpos	**indexOf**
substr	**substring**
strtolower, strtoupper	**toLowerCase, toUpperCase**
trim	**trim**
explode, implode	**split, join**
strcmp	**compareTo**

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

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types

```php
$name = array();                                    # create
$name = array(value0, value1, ..., valueN);

$name[index]                                        # get element value
$name[index] = value;                               # set element value
$name[] = value;                                    # append
```

```php
$a = array();       # empty array (length 0)
$a[0] = 23;         # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";     # add string to end (at index 5)
```

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 loop (same as C)

```
for (initialization; condition; update) {
  statements;
}
```
PHP

```
for ($i = 0; $i < 10; $i++) {
  print "$i squared is " . $i * $i . ".\n";
}
```
PHP

if/else statement

- NOTE: although elseif keyword is much more common, else if is also supported

```php
if (condition) {
   statements;
} elseif (condition) {
   statements;
} else {
   statements;
}
```

while loop (same as C)

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

```php
while (condition) {
  statements;
}
```

```php
do {
  statements;
} while (condition);
```

The foreach loop

- a convenient way to loop over each element of an array without indexes

```php
foreach ($array as $variableName) {
  ...
}
```

```php
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
  print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```

Math operations

```php
$a = 3;
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));
```

abs	ceil	cos	floor	log	log10	max
min	pow	rand	round	sin	sqrt	tan

math functions

M_PI	M_E	M_LN2

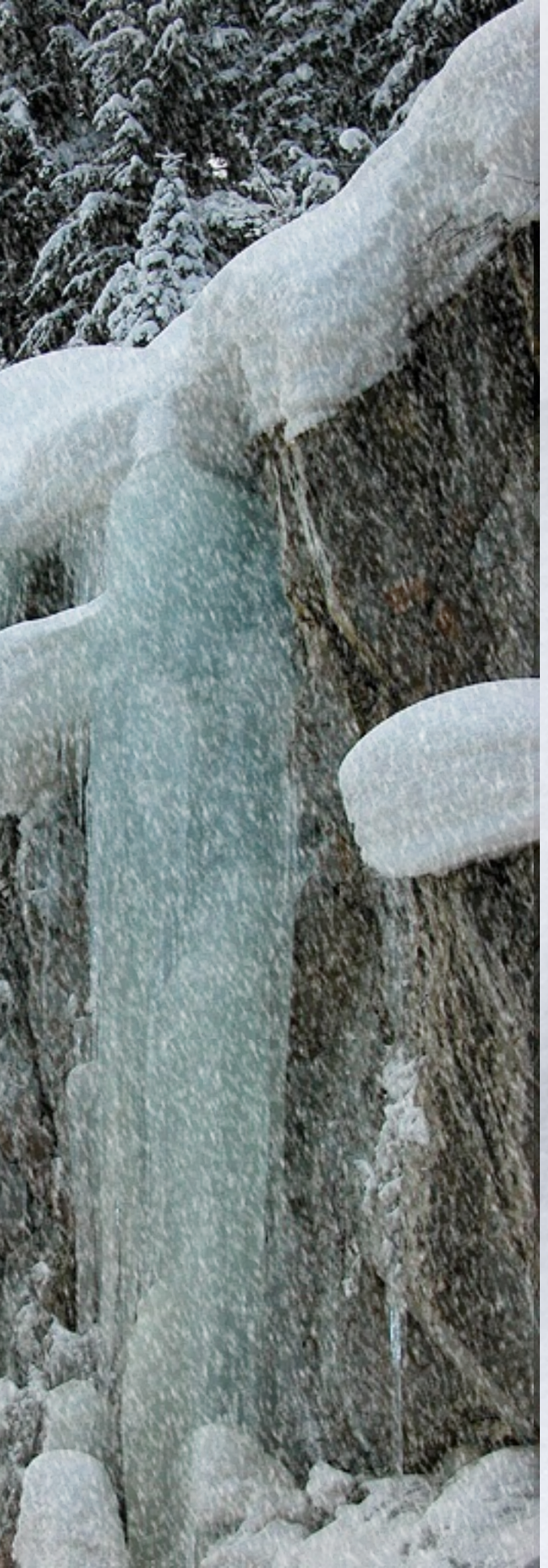
math constants

- the syntax for method calls, parameters, returns is the same as Java and C

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   . . .
```



Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP
- Function & Scope

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

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

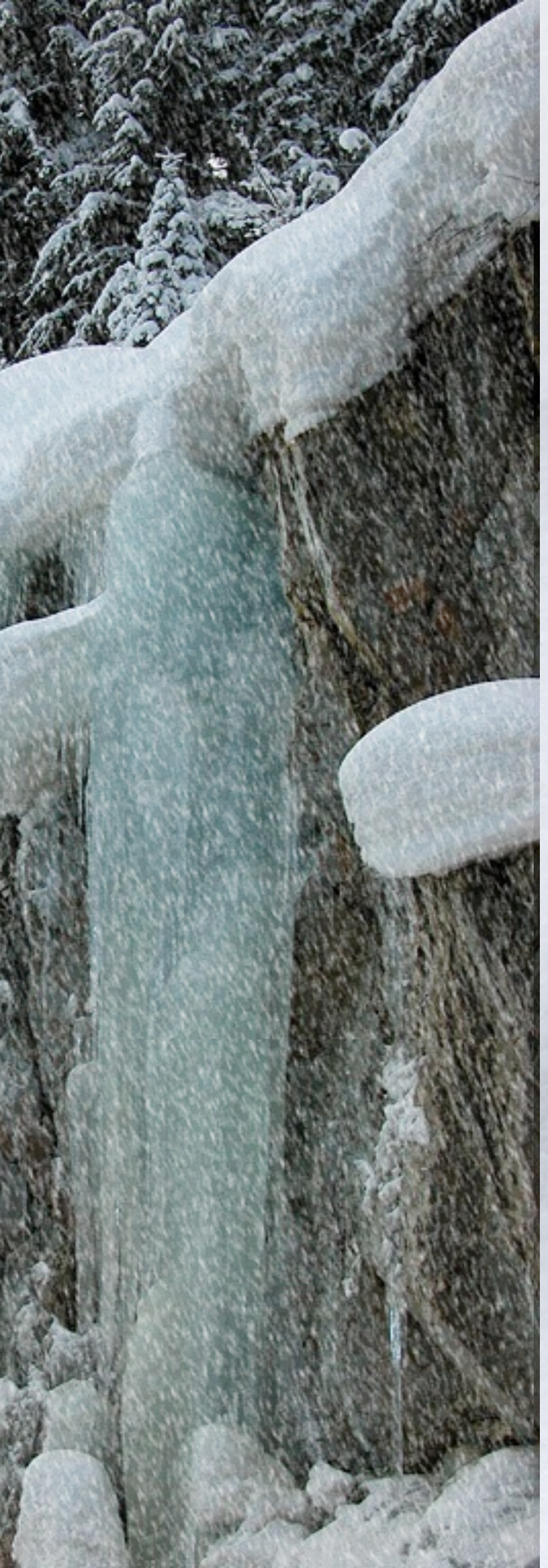
PHP

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

output



Outline

- Server-Side Basics
- Introduction to PHP
- PHP Basic Syntax
- Embedded PHP
- Function & Scope

Functions

- 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

Variable scope

- variables declared in a function are local to that function
- variables not declared in a function are global
- if a function wants to use a global variable, it must have a global statement

```php
$school = "UW";                    # global
...

function downgrade() {
  global $school;
  $suffix = "Tacoma";              # local

  $school = "$school $suffix";
  print "$school\n";
}
```

Thanks!!!