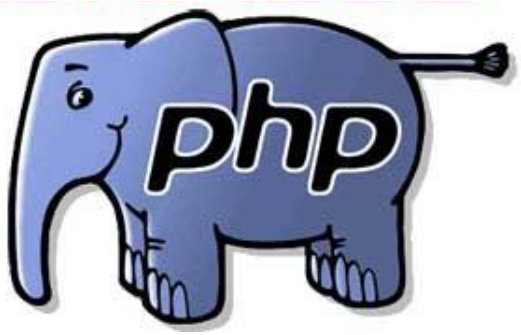
Introduction to PHP

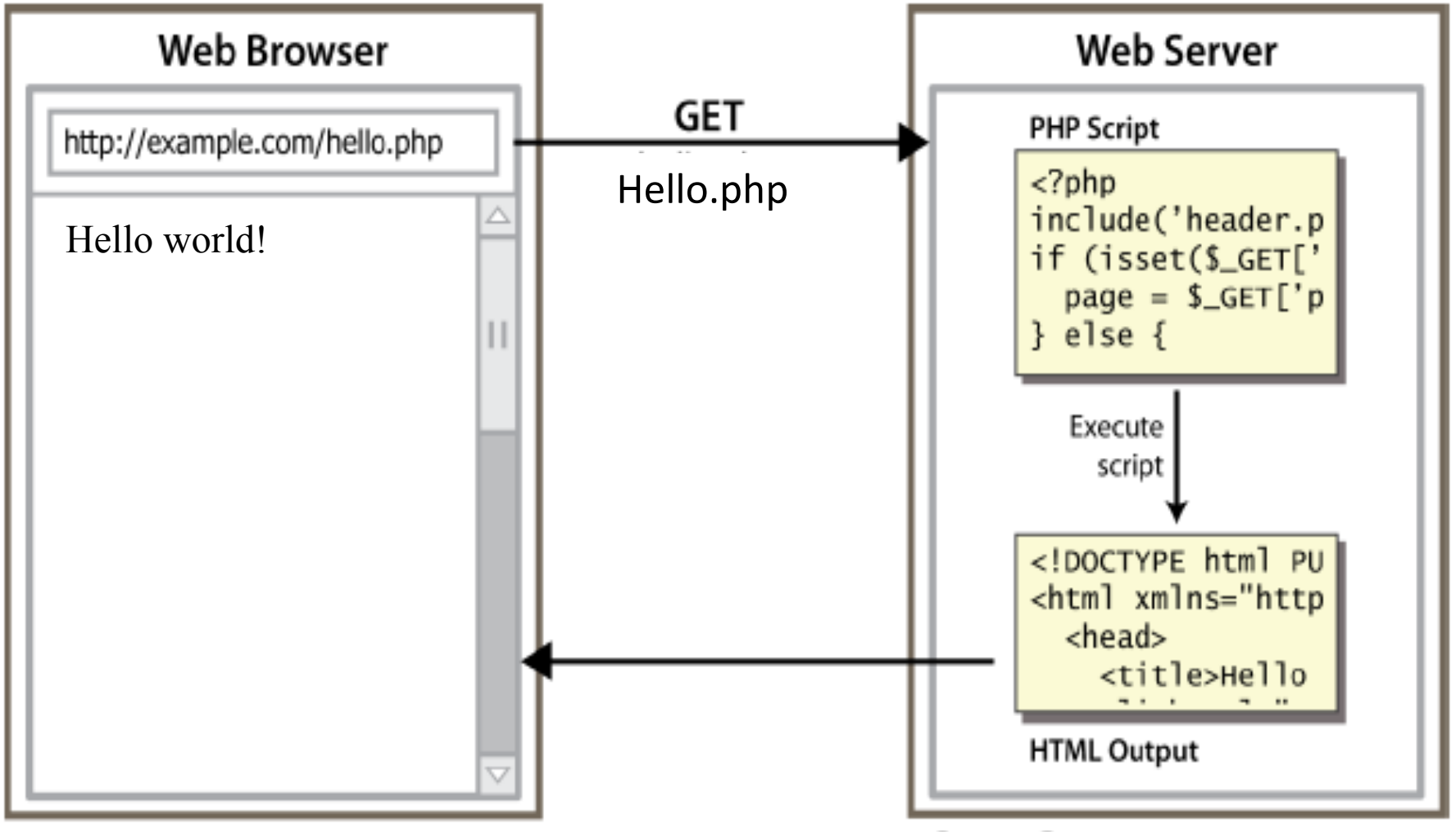
Lecture 3

What is PHP?

- PHP stands for "PHP Hypertext Preprocessor"
- 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



Lifecycle of a PHP web request



Why PHP?

- Free and open source
- Compatible
  - as of November 2006, there were more than 19 million websites (domain names) using PHP.
- Simple

Hello World!

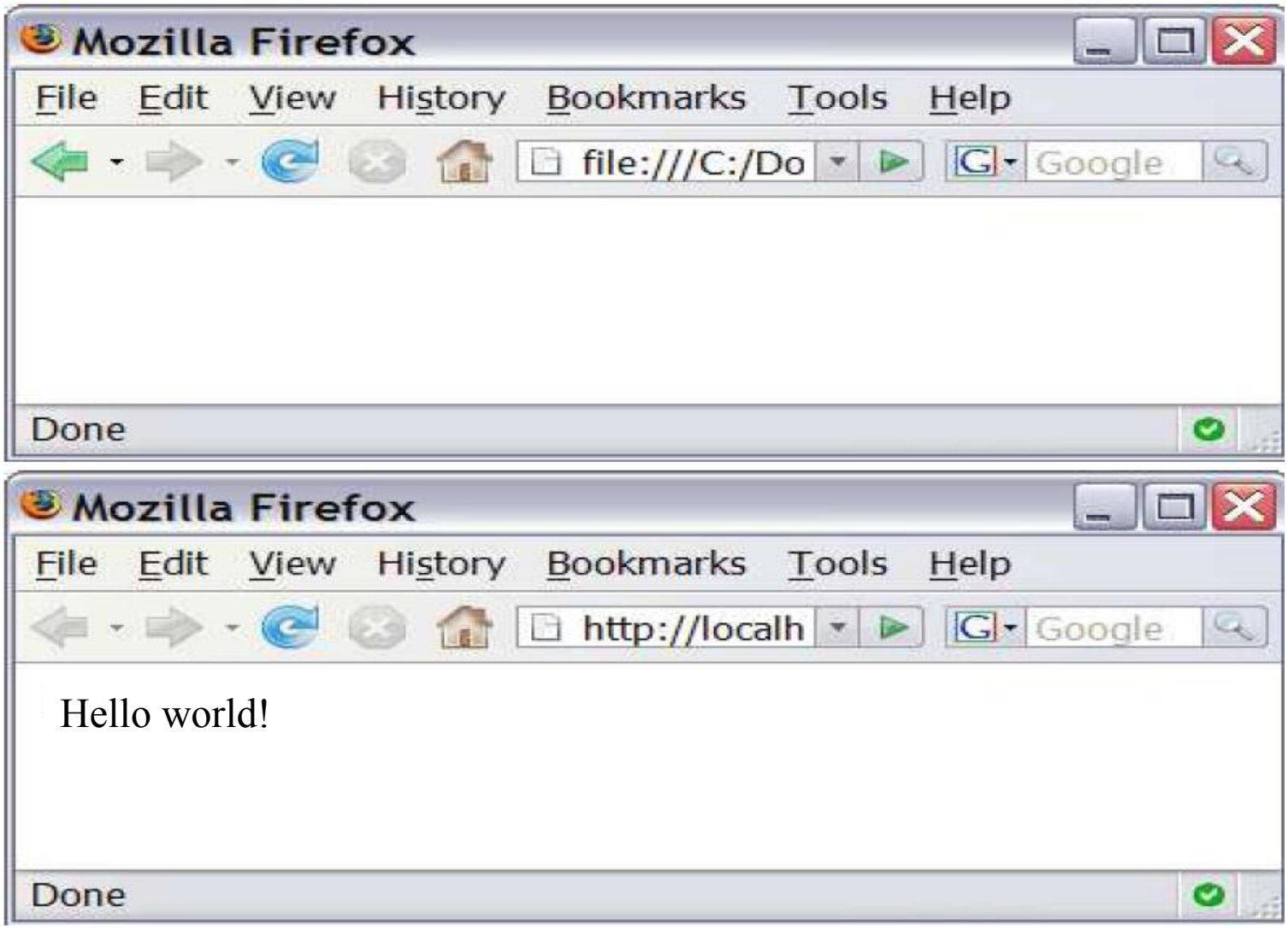
```php
<?php
print "Hello, world!";
?>
```

PHP

```
Hello world!
```

output

Viewing PHP output



PHP BASIC SYNTAX

PHP syntax template

```
HTML content
<?php
PHP code
?>
HTML content
<?php
PHP code
?>
HTML content ...
```

PHP

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

Console output: `print`

```
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!';
```
PHP

```
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

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

```
$user_name = "mundruid78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;
```
PHP

- names are case sensitive
- 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)

Variables

- basic types: *int, float, boolean, string, array, object, NULL*
  - test type of variable with `is_type` functions, e.g. `is_string`
  - `gettype` function returns a variable's type as a string
- 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";`

Arithmetic operators

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

Comments

```
# single-line comment
// single-line comment
/*
multi-line comment
*/
```
PHP

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

`String` Type

```php
$favorite_food = "Ethiopian";
print $favorite_food[2];
$favorite_food = $favorite_food . " cuisine";
print $favorite_food;
```

PHP

- zero-based indexing using bracket notation
- there is no char type; each letter is itself a String
- string concatenation operator is . (period), not +
  - `5 + "2 turtle doves" === 7`
  - `5 . "2 turtle doves" === "52 turtle doves"`
- can be specified with "" or ''

`String` Functions

```
# index 0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);
$cmp = strcmp($name, "Brian Le"); #compares
$index = strpos($name, "e"); #
$first = substr($name, 9, 5);
$name = strtoupper($name);
```

PHP

More: http://www.w3schools.com/php/php_ref_string.asp

`String` Functions (cont.)

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

Interpreted Strings

```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 interpreted
  - variables that appear inside them will have their values inserted into the string
- strings inside ' ' are not interpreted:

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

Interpreted Strings (cont.)

```
print "Today is your $ageth birthday.\n"; # $ageth not
found
print "Today is your {$age}th birthday.\n";
```

PHP

- if necessary to avoid ambiguity, can enclose variable in {}

Interpreted Strings (cont.)

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

- 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)

`for` loop (same as Java)

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

PHP

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

PHP

`bool` (Boolean) type

```php
$feels_like_summer = FALSE;
$php_is_great = TRUE;
$student_count = 7;
$nonzero = (bool) $student_count; # TRUE
```

PHP

- 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
- FALSE prints as an empty string (no output); TRUE prints as a 1

`if/else` statement

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

PHP

`while` loop (same as Java)

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

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

Math operations

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

PHP

math functions

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

math constants

M_PI	M_E	M_LN2

`Int` and `Float` Types

```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
```

PHP

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

Arrays

```php
$name = array();          # create
$name = array(value0, value1, ..., valueN);
$name[index]              # get element value
$name[index] = value;     # set element value
$name[] = value;          # append
```

PHP

```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)
```

PHP

- Append: use bracket notation without specifying an index
- Element type is not specified; can mix types

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

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

PHP

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

foreach loop

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

```
$fellowship = array("Frodo", "Sam", "Gandalf",
"Strider", "Gimli", "Legolas", "Boromir");
print "The fellowship of the ring members are: \n";
for ($i = 0; $i < count($fellowship); $i++) {
      print "{$fellowship[$i]}\n";
}
print "The fellowship of the ring members are: \n";

foreach ($fellowship as $fellow) {
      print "$fellow\n";
}
```
PHP

Multidimensional Arrays

```php
<?php $AmazonProducts = array( array("BOOK", "Books", 50),
                               array("DVDs", "Movies", 15),
                               array("CDs", "Music", 20)
                             );
for ($row = 0; $row < 3; $row++) {
      for ($column = 0; $column < 3; $column++) { ?>
            <p> | <?= $AmazonProducts[$row][$column] ?>
      <?php } ?>
      </p>
<?php } ?>
```

PHP

Multidimensional Arrays (cont.)

```php
<?php $AmazonProducts = array( array("Code" =>"BOOK",
"Description" => "Books", "Price" => 50),
                                array("Code" => "DVDs",
"Description" => "Movies", "Price" => 15),
                                array("Code" => "CDs",
"Description" => "Music", "Price" => 20)
                              );
for ($row = 0; $row < 3; $row++) { ?>
      <p> | <?= $AmazonProducts[$row]["Code"] ?> | <?=
$AmazonProducts[$row]["Description"] ?> | <?=
$AmazonProducts[$row]["Price"] ?>
     </p>
<?php } ?>
```

PHP

`String` compare functions

Name	Function
strcmp	compareTo
strstr, strchr	find string/char within a string
strpos	find numerical position of string
str_replace, substr_replace	replace string

- Comparison can be:
  - Partial matches
  - Others
- Variations with non case sensitive functions
  - strcasecmp

`String` compare functions examples

```php
$offensive = array( offensive word1, offensive
word2);
$feedback = str_replace($offcolor, “%!@*”,
$feedback);
```

PHP

```php
$test = “Hello World! \n”;
print strpos($test, “o”);
print strpos($test, “o”, 5);
```

PHP

```php
$toaddress = “feedback@example.com”;
if(strstr($feedback, “shop”)
      $toaddress = “shop@example.com”;
else if(strstr($feedback, “delivery”)
      $toaddress = “fulfillment@example.com”;
```

PHP

Regular expressions

```
[a-z]at              #cat, rat, bat…
[aeiou]
[a-zA-Z]
[^a-z]               #not a-z
[[:alnum:]]+         #at least one alphanumeric char
(very) *large        #large, very very very large…
(very){1, 3}               #counting “very” up to 3
^bob                 #bob at the beginning
com$                 #com at the end
```

PHPRegExp

- Regular expression: a pattern in a piece of text

EMBEDDED PHP

Printing HTML tags in PHP = bad style

```
<?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";
}
?>
```

HTML

- 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 expression blocks

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

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

```
The answer is 42
```
output

- 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

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>
```

Common errors: unclosed braces, missing = sign

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

PHP

- 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

```
...
<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

Functions

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

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

- parameter types and return types are not written
- a function with no return statements implicitly returns NULL

Default Parameter Values

```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

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

PHP

- if no value is passed, the default will be used

PHP Include File

- Insert the content of one PHP file into another PHP file before the server executes it
- Use the
  - `include()` generates a warning, but the script will continue execution
  - `require()` generates a fatal error, and the script will stop

include() example

```php
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/contact.php">Contact Us</a>
```

PHP

```php
<html>
<body>

<div class="leftmenu">
<?php include("menu.php"); ?>
</div>

<h1>Welcome to my home page.</h1>
<p>I have a great menu here.</p>

</body>
</html>
```

PHP

PHP FILE INPUT/OUTPUT

PHP file I/O functions

function name(s)	category
file, file_get_contents, file_put_contents	reading/writing entire files
basename, file_exists, filesize, fileperms, filemtime, is_dir, is_readable, is_writable, disk_free_space	asking for information
copy, rename, unlink, chmod, chgrp, chown, mkdir, rmdir	manipulating files and directories
glob, scandir	reading directories

Reading/writing files

contents of foo.txt	file("foo.txt")	file_get_contents("foo.txt")
Hello how are you? I'm fine	array("Hello\n", #0 "how are\n", #1 "you?\n", #2 "\n", #3 "I'm fine\n" #4)	"Hello\n how are\n you?\n \n I'm fine\n"

- `file` returns lines of a file as an array
- `file_get_contents` returns entire contents of a file as a string

Reading/writing an entire file

```php
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

PHP

- `file_get_contents` returns entire contents of a file as a string
- `file_put_contents` writes a string into a file, replacing any prior contents

Appending to a file

```php
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text,
FILE_APPEND);
```

PHP

old contents

Roses are red,
Violets are blue.
All my base,
Are belong to you.

new contents

Roses are red,
Violets are blue.
All my base,
Are belong to you.
P.S. ILY, GTG TTYL!~

The `file` function

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {
      ?>
      <li> <?= $line ?> </li>
<?php
}
?>
```
PHP

- file returns the lines of a file as an array of strings
  - each string ends with \n
  - to strip the \n off each line, use optional second parameter:

```
$lines = file("todolist.txt",FILE_IGNORE_NEW_LINES);
```
PHP

Unpacking an array: `list`

```
list($var1, ..., $varN) = array;
```
PHP

```
$values = array("mundruid", "18", "f", "96");
...
list($username, $age, $gender, $iq) = $values;
```
PHP

- the list function accepts a comma-separated list of variable names as parameters
- use this to quickly "unpack" an array's contents into several variables

Fixed-length files, `file` and `list`

```
Xenia Mountrouidou
(919)685-2181
570-86-7326
```

contents of file personal.txt

```
list($name, $phone, $ssn) = file("personal.txt");
```

PHP

- reads the file into an array of lines and unpacks the lines into variables
- Need to know a file's exact length/format

Splitting/joining strings

```php
$array = explode(delimiter, string);
$string = implode(delimiter, array);
```

PHP

```php
$class = "CS 380 01";
$class1 = explode(" ", $s); # ("CS", "380", "01")
$class2 = implode("...", $a); # "CSE...380...01"
```

PHP

- `explode` and `implode` convert between strings and arrays

Example `explode`

```
Harry Potter, J.K. Rowling
The Lord of the Rings, J.R.R. Tolkien
Dune, Frank Herbert
```

contents of input file books.txt

```
<?php foreach (file("books.txt") as $book) {
      list($title, $author) = explode(",", $book);
      ?>
      <p> Book title: <?= $title ?>, Author: <?=
$author ?> </p>
<?php
}
?>
```

PHP

Reading directories

function	description
scandir	returns an array of all file names in a given directory (returns just the file names, such as "`myfile.txt`")
glob	returns an array of all file names that match a given pattern (returns a file path and name, such as "`foo/bar/myfile.txt`")

Example for `glob`

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
      $text = file_get_contents($poemfile);
      file_put_contents($poemfile, strrev($text));
      print "I just reversed " .
basename($poemfile);
}
```

PHP

- glob can match a "wildcard" path with the * character
- the basename function strips any leading directory from a file path

Example for `glob`

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
      $text = file_get_contents($poemfile);
      file_put_contents($poemfile, strrev($text));
      print "I just reversed " .
basename($poemfile);
}
```

PHP

- glob can match a "wildcard" path with the * character
- the basename function strips any leading directory from a file path

Example for `scandir`

```
<ul>
<?php
$folder = "taxes/old";
foreach (scandir($folder) as $filename) {
      ?>
      <li> <?= $filename ?> </li>
<?php
}
?>
</ul>
```

PHP

- .
- ..
- 2009_w2.pdf
- 2007_1099.doc

output