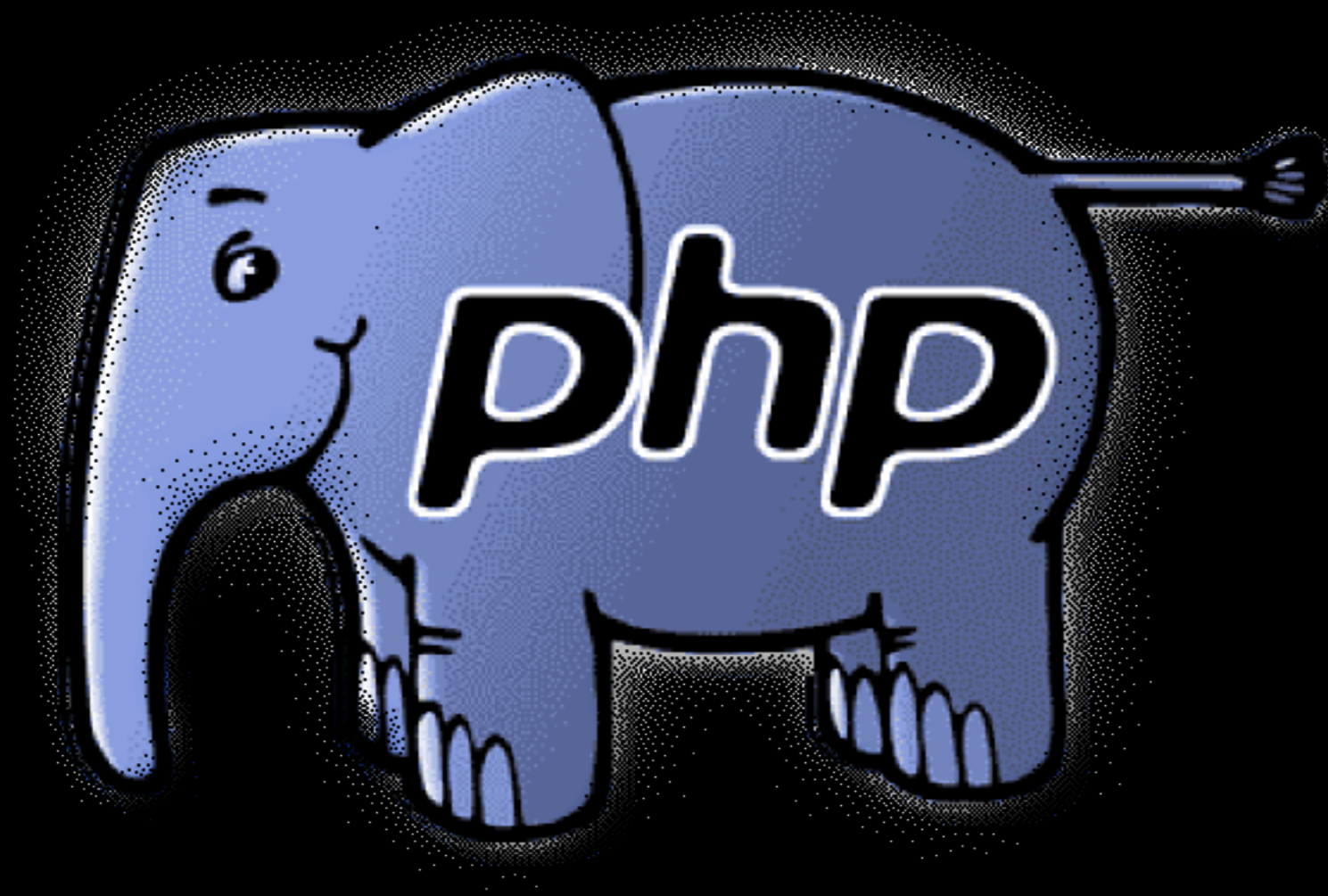


Intro to PHP

Multimedia Design 2
Week 8

Intro to PHP

1. What is it?
2. What does it look like?
3. Variables & strings
4. Arrays
5. Accepting input



1. What is it?

What is PHP?

- PHP: Hypertext Preprocessor (yes, the acronym PHP includes the word “PHP”)
- A server-side scripting language
- Widely available
- Conceptually simple

The Hello World

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

Opening tag

<?php

Start the PHP code block

```
echo "Hello, world!";
```

?>

Closing tag

```
<?php
```

```
echo "Hello, world!";
```

```
?>
```

End PHP code block

PHP code block

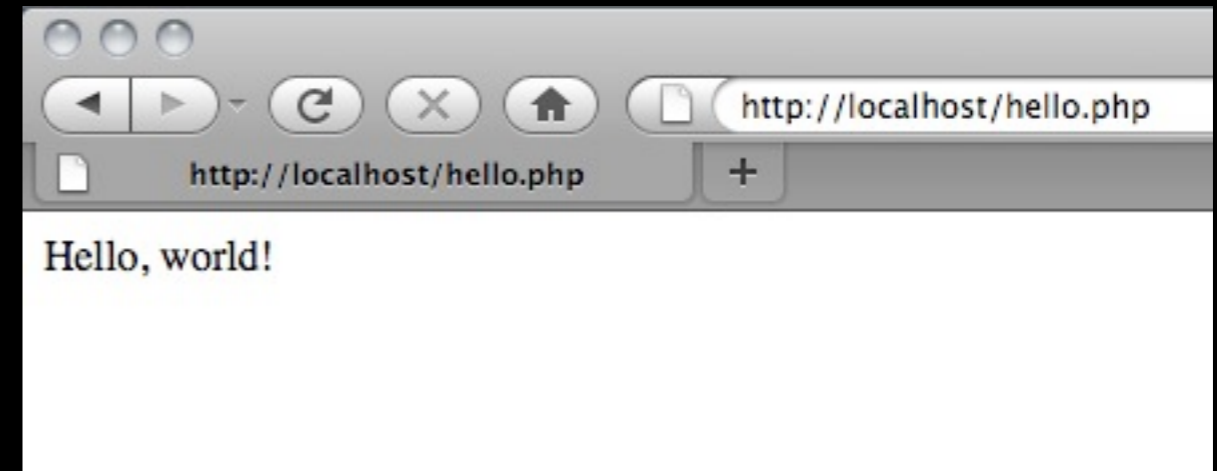
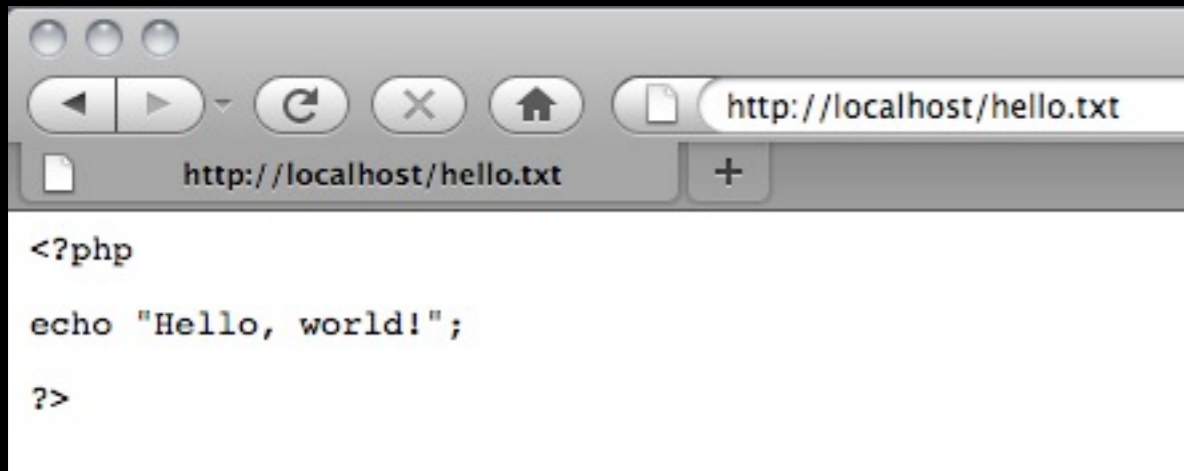
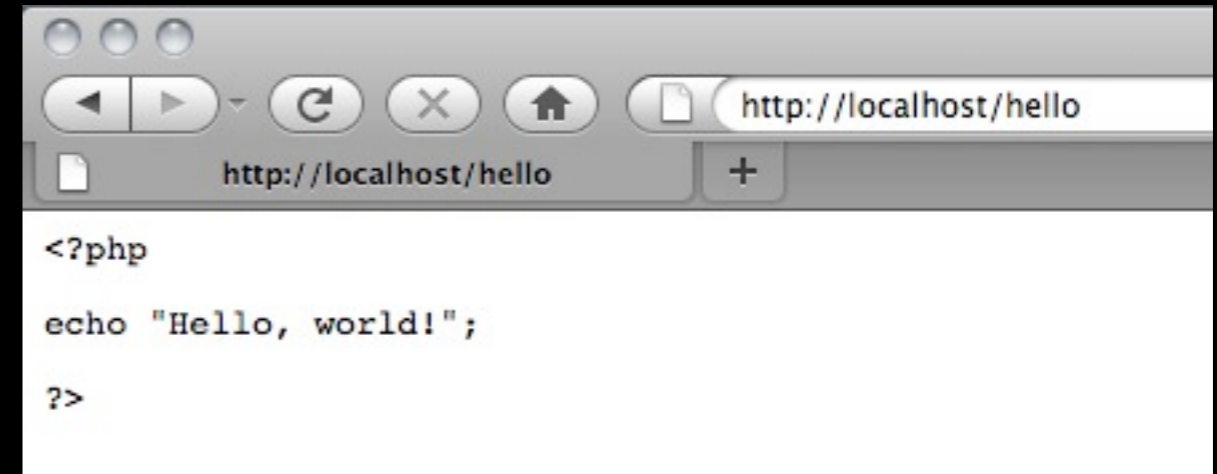
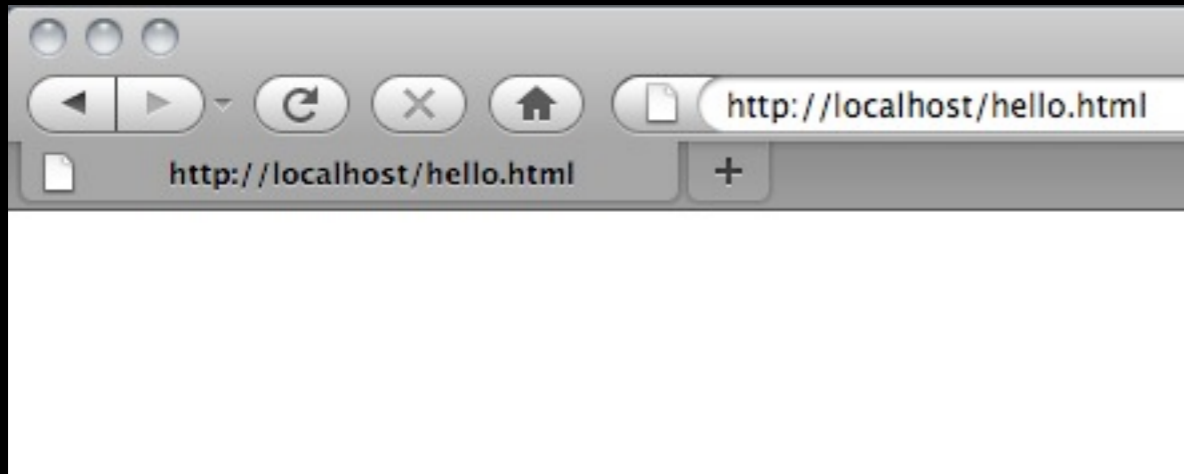
```
<?php
```

```
echo "Hello, world!";
```

```
?>
```

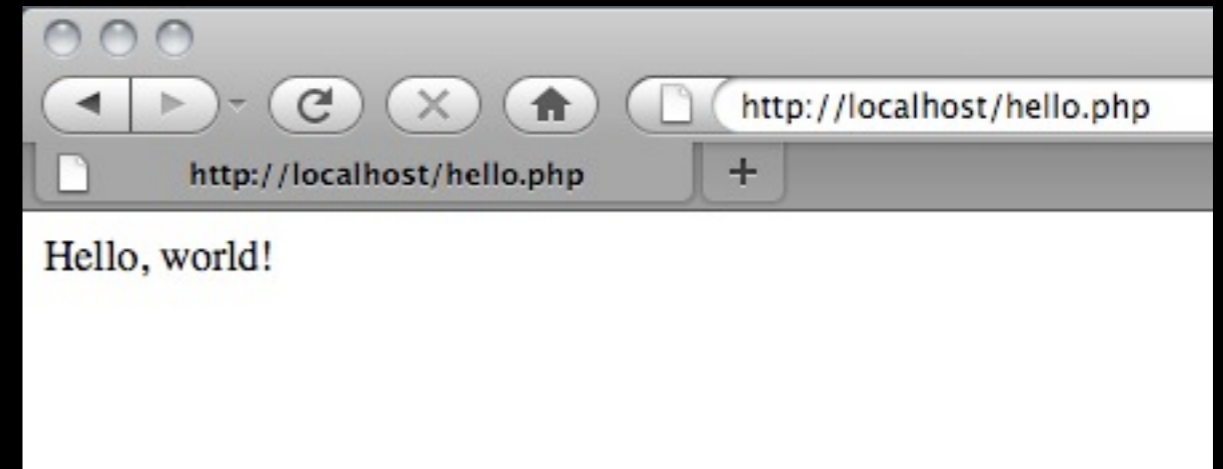
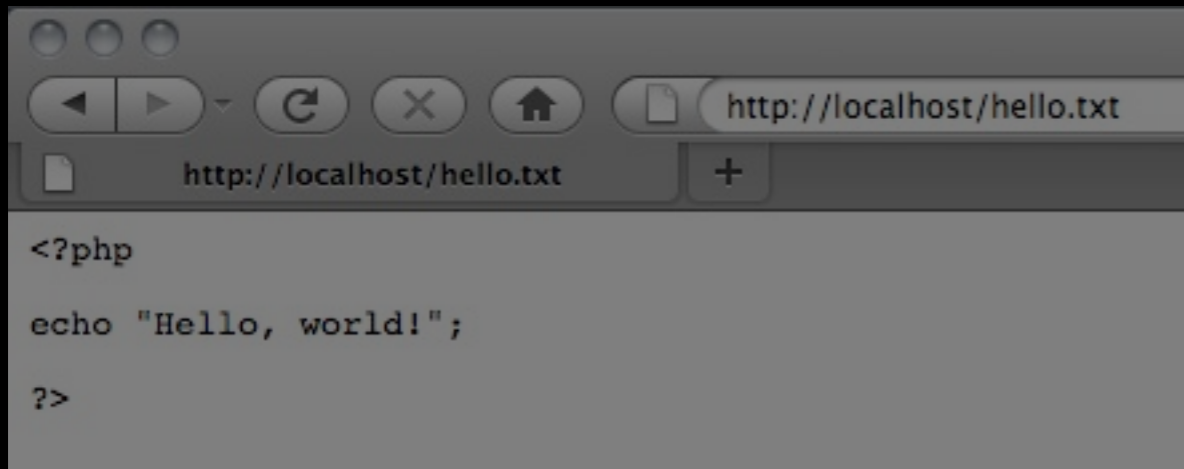
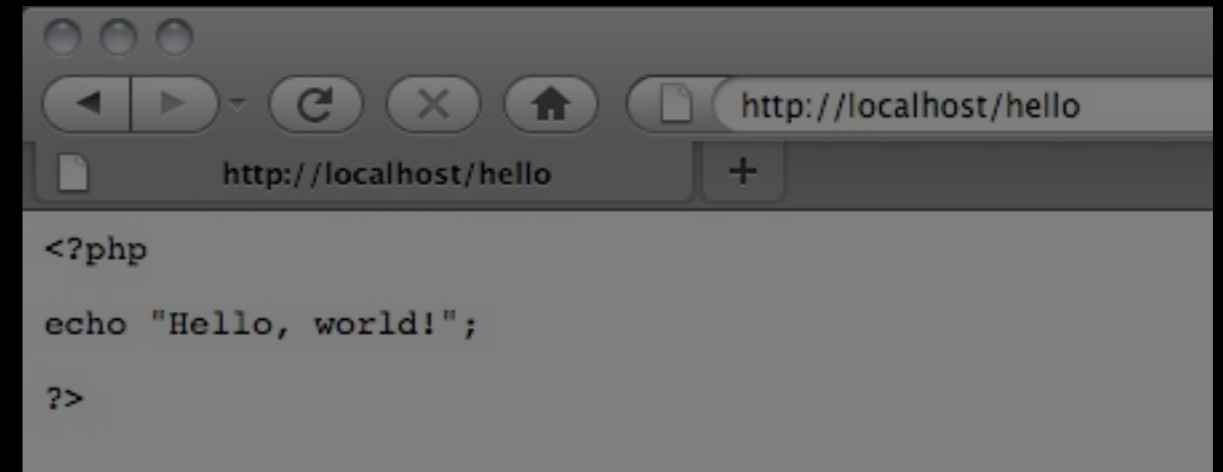
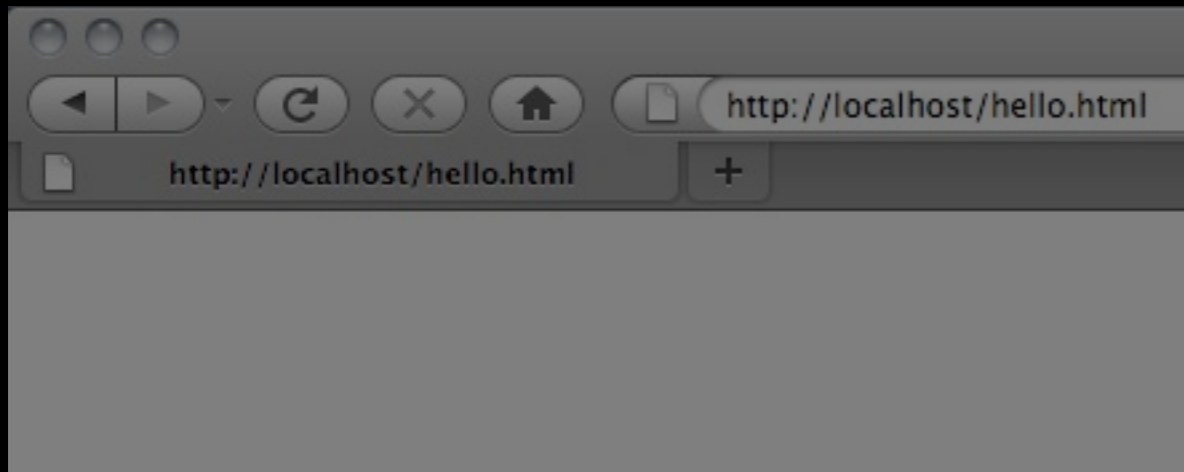
PHP code goes here

File names matter



The same code, named with 4 different file extensions

File names matter



The same code, named with 4 different file extensions

To properly execute a PHP script, you must name its file ending with .php



2. What's it look like?

PHP can be mixed in with HTML

```
<div class="fields column-ab">
  <?php
  .
  global $response;
  if (!empty($response)) {
    echo "<p class=\"feedback\"><strong>$response</strong></p>";
  }
  .
  ?>
  <label for="first_name">
    <?php l('first_name'); ?>
  </label>
  <input type="text" name="first_name"<?php v('first_name'); ?> id="first_name" class="text" />
  <br class="clear" />
  <label for="last_name">
    <?php l('last_name'); ?>
  </label>
  <input type="text" name="last_name"<?php v('last_name'); ?> id="last_name" class="text" />
  <br class="clear" />
```

PHP can also execute independently

```
<?php
.
if (loginValidates()) {
    $_SESSION['authenticated'] = true;
    header('Location: admin.php');
} else {
    header('Location: admin.php?login=failed');
}
.
function loginValidates() {
    if (empty($_POST['username'])) {
        return false;
    } else if ($_POST['username'] != USERNAME) {
        return false;
    } else if (empty($_POST['password'])) {
        return false;
    } else if ($_POST['password'] != PASSWORD) {
        return false;
    }
    return true;
}
.
?>
```

Look familiar?

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

```
// Variables in PHP  
$foo = 'bar';
```

```
// Arrays in PHP  
$baz = array(1, 2, 3, 4);
```

```
// Loops in PHP  
for ($i = 0; $i < 5; $i++) {  
    // ...  
}
```

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```

Look familiar?

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

```
// Variables in PHP  
$foo = 'bar';
```

```
// Arrays in PHP  
$baz = array(1, 2, 3, 4);
```

```
// Loops in PHP  
for ($i = 0; $i < 5; $i++) {  
    // ...  
}
```

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```

Look familiar?

Variables are prefixed with \$ signs

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

```
// Variables in PHP  
$foo = 'bar';
```

```
// Arrays in PHP  
$baz = array(1, 2, 3, 4);
```

```
// Loops in PHP  
for ($i = 0; $i < 5; $i++) {  
    // ...  
}
```

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```

Look familiar?

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

Arrays use parentheses and are prefixed with the word **array**

```
// Arrays in PHP  
$baz = array(1, 2, 3, 4);
```

```
// Loops in PHP  
for ($i = 0; $i < 5; $i++) {  
    // ...  
}
```

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```

Look familiar?

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

```
// Variables in PHP  
$foo = 'bar';
```

Loops and conditionals use a very similar syntax, differences are minor

```
// Loops in PHP  
for ($i = 0; $i < 5; $i++) {  
    // ...  
}
```

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```

Look familiar?

```
// Variables in JavaScript  
var foo = 'bar';
```

```
// Arrays in JS  
var baz = [1, 2, 3, 4];
```

```
// Loops in JS  
for (var i = 0; i < 5; i++) {  
    // ...  
}
```

```
// Functions in JS  
function shorten(str) {  
    return str.substr(0, 10);  
}
```

```
// Variables in PHP  
$foo = 'bar';
```

```
// Arrays in PHP  
$baz = array(1, 2, 3, 4);
```

```
// Loops in PHP
```

PHP is (traditionally) less object-oriented in its design

```
// Functions in PHP  
function shorten($str) {  
    substr($str, 0, 10);  
}
```



3. Variables & Strings

Variables

```
// Strings
$nickname = "Johnny Appleseed";
$realname = "John Chapman";
$epitaph = "He lived for others";

// Numbers
$born = 1774;
$died = 1845;
```

Single quotes

```
echo '<h1>' . $nickname . '</h1>';  
echo '<h2>' . $realname . '</h2>';  
echo '$epitaph<br />$born-$died';
```

Use the `.` symbol to concatenate strings together

Double quotes

Strings can span multiple lines (not so with single quoted strings)

```
echo "  
  <h1>\ "$nickname\" </h1>  
  <h2>$realname</h2>  
  $epitaph<br />$born-$died  
";
```

To **escape** the " symbol in a double quote string, use \"

You can **inject variables** into your string (not so with single quotes)

Heredocs

Heredocs allow you to avoid escaping " symbols

```
echo <<<END
  <h1 class="name">$name</h1>
  <h2>a.k.a. $nickname</h2>
  $epitaph<br />$born-$died
END;
```

You can use any token for END as long as the last line begins with it

Inspecting variables

```
// Read a file with the U.S. states  
$states = file('states.txt');
```

```
// Outputs a string value  
echo "Number of states: " . count($states);
```

```
// Allows us to inspect the contents  
print_r($states);
```

```
// Gives more details about datatypes  
var_dump($states);
```



4. Arrays

Arrays

```
// An array
$fishes = array(
    'anchovy',
    'bass',
    'cod'
);

// Get the last item in the array
echo $fishes[2];

// Add an item to the array
$fishes[] = 'dogfish';
```

Associative arrays

```
// An associative array
$cast = array(
    'Gilligan' => 'Bob Denver',
    'Skipper' => 'Alan Hale, Jr.',
    'Ginger' => 'Tina Louise'
);

// Get an item from the associative array
echo $cast['Gilligan'];

// Add an item to the associative array
$cast['Mary Ann'] = 'Dawn Wells';
```

Nested arrays

```
// Arrays nested inside an associative array
$mlb = array(
    'National League' => array(
        'National League East',
        'National League Central',
        'National League West'
    ),
    'American League' => array(
        'American League East',
        'American League Central',
        'American League West'
    )
);
```

Arrays in strings

```
// Arrays
echo "$fishes[3]<br />";

// Associative arrays
echo "{$cast['gilligan']}<br />";

// Nested arrays
echo "{$mlb['National League'][0]}<br />";
```

You must encapsulate associative or nested arrays with the { and } symbols

Iterating

```
$mlb = array(
    'National League' => array(...)
    'American League' => array(...)
);

// foreach loop
foreach ($mlb as $league => $divisions) {
    echo "<h1>$league</h1>";
    // Nested for loop
    for ($i = 0; $i < count($divisions); $i++) {
        echo "$divisions[$i]<br />";
    }
}
```



5. Accepting input

Superglobals

- Global variables that are pre-set by PHP before your script executes
- They are easy to spot— all-caps and prefixed by an underscore
- `$_GET` and `$_POST` are superglobals that store content submitted by the user

\$_POST

```
<form action="post.php" method="post">
  <input type="text" name="foo" />
  <input type="submit" value="Go" />
</form>
<?php

if (!empty($_POST['foo'])) {
  echo "You entered: {$_POST['foo']}";
}

?>
```

\$_GET

```
<form action="get.php">
  <input type="text" name="foo" />
  <input type="submit" value="Go" />
</form>
<?php

if (!empty($_GET['foo'])) {
  echo "You entered: {$_GET['foo']}";
}

?>
```

\$_GET example

```
<?php

// Use input to determine which page to show
if ($_GET['p'] == 'portfolio') {
    $page = 'portfolio.php';
} else if ($_GET['p'] == 'photos') {
    $page = 'photos.php';
} else {
    $page = 'home.php';
}
include $page;

?>
```

\$_POST example

```
<?php

$username = 'admin';
$password = 'secret';

if ($_POST['username'] == $username &&
    $_POST['password'] == $password) {
    include 'restricted.php';
} else {
    $response = 'Sorry, login failed.';
    include 'login.php';
}

?>
```

Magic quotes

```
// All you need to know is they are evil
function antimagic(&$value) {
    $value = is_array($value) ?
        array_map('antimagic', $value) :
        stripslashes($value);
    return $value;
}
```

```
antimagic($_GET);
antimagic($_POST);
```

Next week

```
<?php
```

```
include 'template.php';
```

```
?>
```