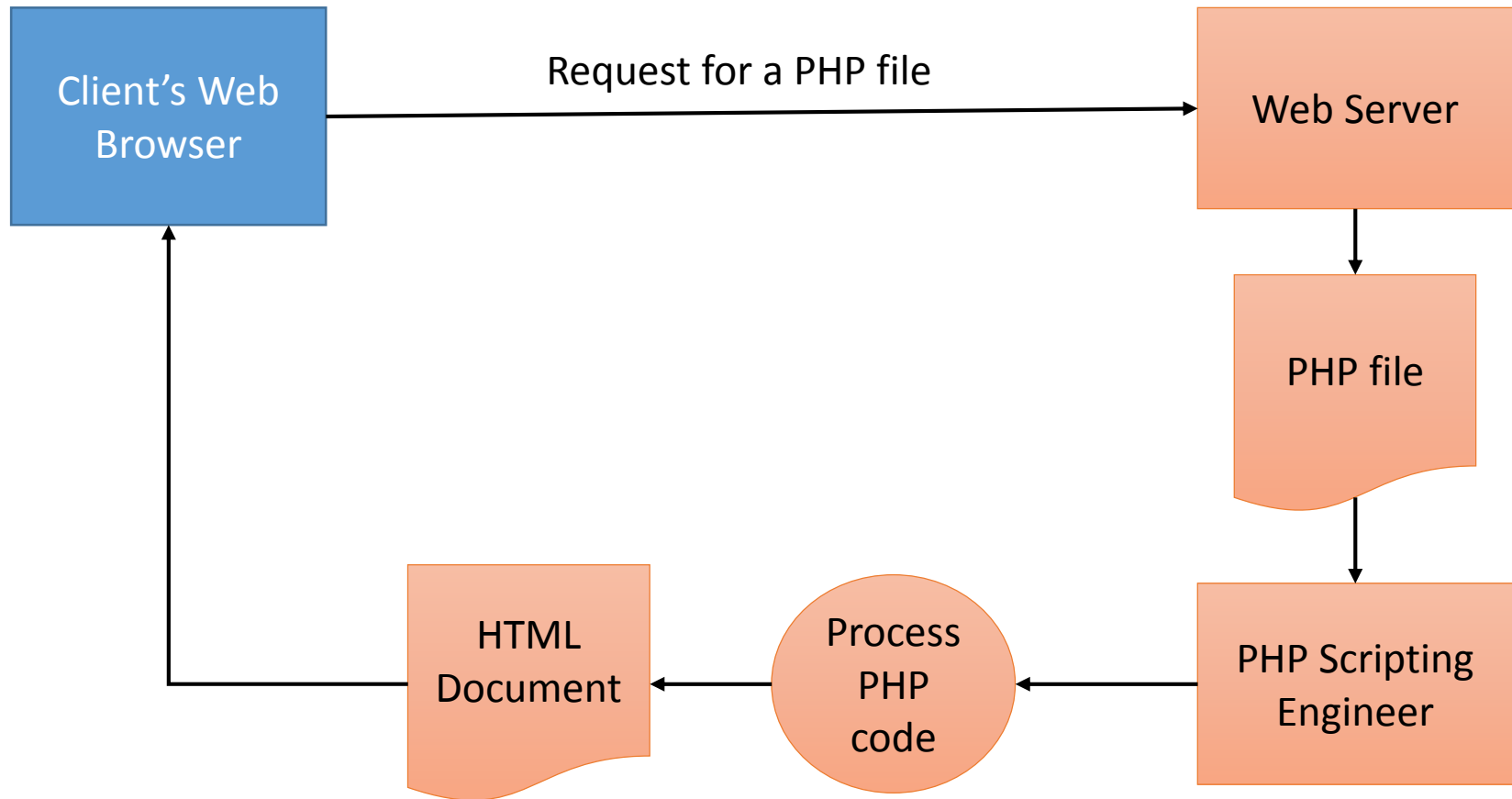


Introduction to PHP

PHP

- PHP, Hypertext Preprocessor, is an open-source, server-side programming language.
- It fills the gap between static html pages and fully dynamic pages.
- It is embedded directly in the HTML.
- You can directly type PHP code into a web page as separate section. And the web page containing PHP code must have an extension of .php.
- Whenever a request is made for a document with an extension of .php, the Web server sends the file to the scripting engine for processing.
- The scripting engine ignores any non-PHP code and only processes the PHP code it finds within PHP code blocks. The Web server then returns the results of the PHP script and any HTML elements found in the PHP file to the client,

Client's Request for a PHP file



The generated HTML returned to client

Creating PHP Code Blocks

- You write PHP scripts within code declaration blocks. which are separate sections on a Web page that are interpreted by the scripting engine.
- Four types of code declarations.
 - Standard PHP script Delimiters
 - The `<script>` element
 - Short PHP script Delimiters
 - ASP-style script delimiters

Standard PHP script delimiters

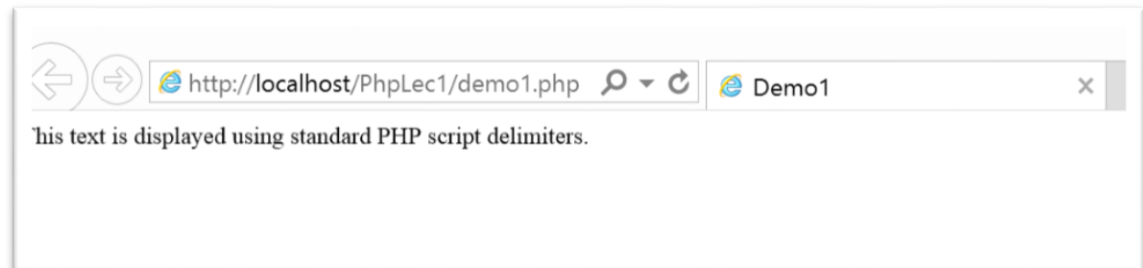
- The standard method of writing PHP code declaration blocks is to use the `<?php` and `?>` script delimiters.

```
<?php  
    statements;  
?>
```

- The PHP Group officially recommends that you use standard PHP script delimiters to write PHP code declaration blocks. One reason is that standard PHP script delimiters are guaranteed to be available on any Web server that supports PHP.

Example of Standard PHP script delimiters

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo1</title>
  </head>
  <body>
    <?php
      echo "This text is displayed using standard PHP script
delimiters. ";
    ?>
  </body>
</html>
```



The <script> Element

- When the <script> element is used with PHP, you must assign a value of “php” to the language attribute of the <script> element to identify the code block as PHP.

```
<script language="php">
```

```
    statements;
```

```
</script>
```

- Example,

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <title>Demo2</title>
```

```
    </head>
```

```
    <body>
```

```
        <script language="php">
```

```
            echo "This text is displayed using a PHP script section.";
```

```
        </script>
```

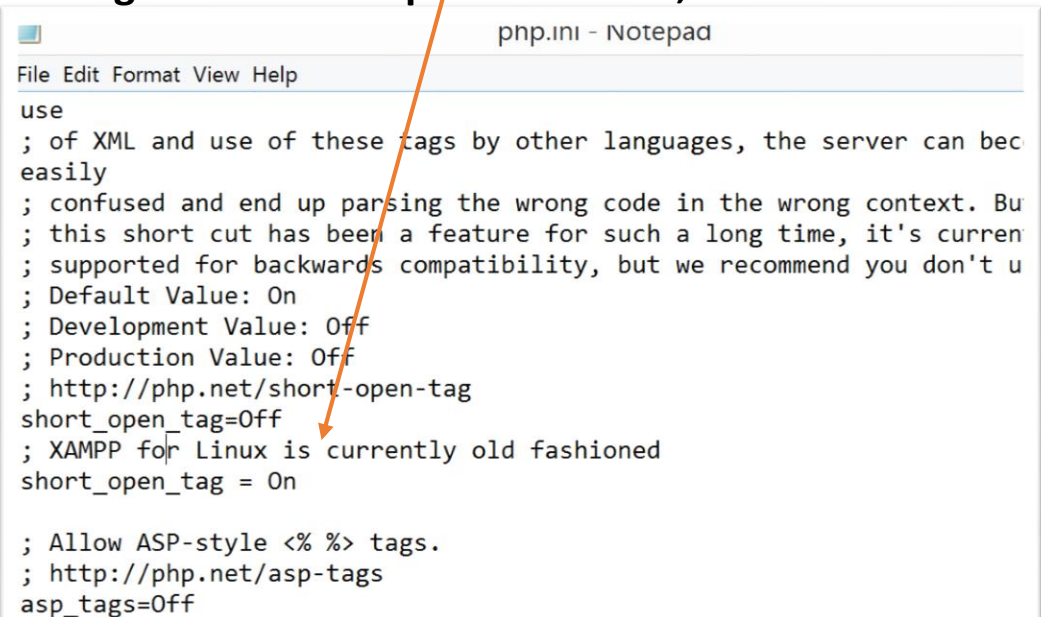
```
    </body>
```

```
</html>
```

Short PHP Script Delimiters

- A simplified method of writing PHP code declaration blocks is to use the short `<? and ?>` script delimiters.
- The `<? and ?>` can be disabled in a Web Server's **php.ini** configuration file, so the PHP Group discourages the use of short delimiters, especially when developing scripts that will be redistributed and used by other Web developers
- Example,

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo3</title>
  </head>
  <body>
    <?
      echo "This text is displayed using short PHP script delimiters.";
    ?>
  </body>
</html>
```



```
File Edit Format View Help
use
; of XML and use of these tags by other languages, the server can be
easily
; confused and end up parsing the wrong code in the wrong context. Bu
; this short cut has been a feature for such a long time, it's curren
; supported for backwards compatibility, but we recommend you don't u
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/short-open-tag
short_open_tag=Off
; XAMPP for Linux is currently old fashioned
short_open_tag = On

; Allow ASP-style <% %> tags.
; http://php.net/asp-tags
asp_tags=Off
```


ASP-style Script Delimiters

- Some Web developers prefer to use the ASP-style script delimiters of `<%` and `%>` to develop PHP scripts. The syntax for ASP-style script delimiters is similar to that of short PHP script delimiters, as follows:

`<% statements; %>`

- Example,
`<!DOCTYPE html>`
`<html>`
 `<head>`
 `<title>Demo4</title>`
 `</head>`
 `<body>`
 `<%`
 `echo "This text is displayed using ASP-style script`
 `delimiters."`
 `%>`
 `</body>`
`</html>`

Functions

- The term function refers to a subroutine (or individual statements grouped into a logical unit) that performs a specific task. PHP includes numerous built-in functions that perform various types of tasks
- To execute a function, you create a statement to call it in your script and that statement is referred to as a **function call**. It consists of the function name followed by parentheses. The data inside the parentheses are called **arguments or actual parameters**.
- Sending data to a called function is called **passing arguments**.
- Many functions generate, or return, some sort of a value that you can use in your script.

Functions

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Demo5</title>
```

```
  </head>
```

```
  <body>
```

```
    <?php
```

```
      echo round(7.556)."<br>";
```

```
      echo round(7.556, 2);
```

```
      phpinfo();
```

```
    ?>
```

```
  </body>
```

```
</html>
```

The screenshot shows a web browser window with the address bar displaying `http://localhost/PhpLe` and a tab titled "Demo5". The page content shows the output of a PHP script: `8` followed by `7.56` on a new line. Below this is a blue banner for "PHP Version 5.4.27" with the PHP logo. The rest of the page displays the output of the `phpinfo()` function, which is a table of system and configuration details.

System	Windows NT KEN-PC 6.2 build 9200 (Unknown Windows version Home Premium Edition i586)
Build Date	Apr 3 2014 00:52:41
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension	API220100525,TS,VC9

The echo and print statements

- The echo and print statements are language constructs of the PHP programming language.
 - Both create new text on a Web page that is returned as a response to a client.
 - The print statement returns a value of 1 if it is successful or a value of 0 if it is not successful, while the echo statement does not return a value.
 - Echo supports multiple arguments. If you want to pass multiple arguments to the echo statements, separate them with commas,
- A programming language construct refers to a built-in feature of a programming language.

Example of the echo and print statements

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo5</title>
  </head>
  <body>
    <?php
      echo "<p>Using the echo statement: <br />",
        "PHP Programming <br />",
        "Web Services <br />",
        "Restful Services </p>";
      print"<p>Using the print statement: <br />".
        "PHP Programming <br />".
        "Web Services <br />".
        "Restful Services </p>";
    ?>
  </body>
</html>
```



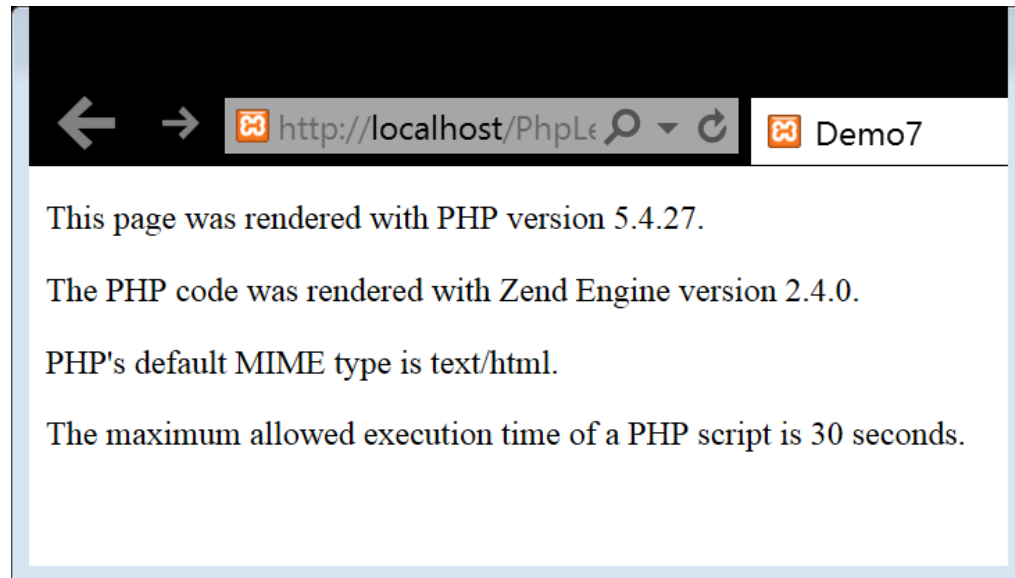
Creating Multiple Code Declaration Blocks

- You can include as many PHP script sections as you want within a document. However, when you include multiple script sections in a document, you must include a separate code declaration block for each section.

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php
echo "<p>Output from the first script section.</p>";
?>
<h2>Second Script Section</h2>
<?php
echo "<p>Output from the second script section.</p>";
?>
</body>
</html>
```

Example of Creating Multiple Code Declaration Blocks

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo7</title>
  </head>
  <body>
    <p>This page was rendered with PHP version
      <?php echo phpversion(); ?>.
    </p>
    <p>The PHP code was rendered with Zend Engine version
      <?php echo zend_version(); ?>.
    </p>
    <p>PHP's default MIME type is
      <?php echo ini_get("default_mimetype"); ?>.
    </p>
    <p>The maximum allowed execution time of a PHP script is
      <?php echo ini_get("max_execution_time"); ?> seconds.
    </p>
  </body>
</html>
```



Case Sensitivity in PHP

- PHP are mostly case insensitive, although there are some exceptions. This means that you can use any of the following versions of the echo statement without receiving an error message:

```
<?php
```

```
echo "<p>Good <strong> students</strong>, <br />";
```

```
Echo "<strong>Good students</strong>, <br />";
```

```
ECHO " and <strong> teachers</strong>!</p>"; ?>
```

- Though you use whatever case you want, be sure to use it consistently to avoid any potential problems and to make it easier to debug your scripts.
- Note that variable and constant names are case sensitive.

Variables

- Variables are memory locations that are used to store values. The values of variables can be changed.
- To use a variable in a program, you first have to write a statement that creates the variable and assigns it a name.
- The name you assign to a variable is called an identifier. You must observe the following rules:
 - Identifiers must begin with a dollar sign (\$).
 - Identifiers may contain uppercase and lowercase letters, numbers, or underscores (_). The first character after the dollar sign must be a letter.
 - Identifiers cannot contain spaces.
 - Identifiers are case sensitive.

Declaring and initializing variables

- Before you can use a variable in your code, you have to create it.
- The process of specifying and creating a variable name is called declaring the variable.
- The process of assigning a first value to a variable is called initializing the variable.
- In PHP, you must declare and initialize a variable in the same statement, using the following syntax:
`$variable_name = value;`

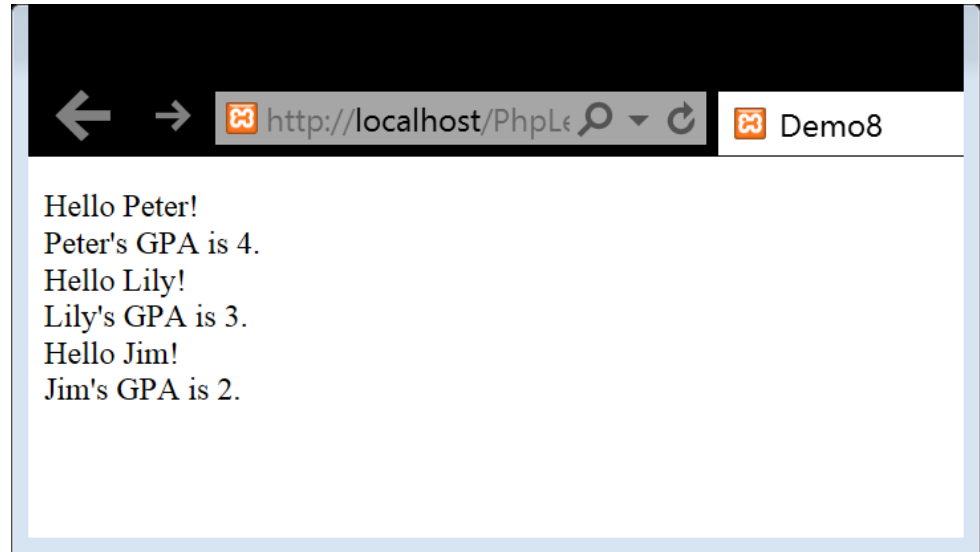
Displaying Variables

- If you want to display text strings and variables in the same statement,
- You can pass them to the echo statement as individual arguments, separated by commas. `echo "<p>The number of books is ", $numBooks, ".</ p>";`
- You can also include variable names inside a text string,
 - If you use double quotation marks, the value assigned to the variable will appear.
`echo "<p>The number of books is $numBooks.</p>";`
 - if you use single quotation marks, the name of the variable will appear.
`echo '<p>The number of books is $numBooks.</p>';`

Example of Variables

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo8</title>
  </head>
  <body>
    <?php
      $student1 = "Peter";
      $student2 = "Lily";
      $student3 = "Jim";
      $gpa1 = 4.0;
      $gpa2 = 3.0;
      $gpa3 = 2.0;

      echo "<p>Hello $student1!<br />";
      echo "$student1's GPA is $gpa1.<br />";
      echo "Hello ", $student2, "!<br />";
      echo "$student2's GPA is $gpa2.<br />";
      echo "Hello ", $student3, "!<br />";
      echo "$student3's GPA is $gpa3.<br />";
    ?>
  </body>
</html>
```



Defining Constants

- A constant contains information that does not change during the course of program execution.
- Unlike variable names, constant names do not begin with a dollar sign (\$). In addition, it is common practice to use all uppercase letters for constant names.
- You use the `define()` function to create a constant. The syntax for the `define()` function is as follows:
`define("CONSTANT_NAME", value);`
- Examples,

```
define("TOP_SCHOOL", "Happy Valley College");  
define("MAX_SIZE ", 200);
```
- When you refer to a constant in code, and do not include a dollar sign.
- You can pass a constant name to the `echo` statement. E.g. `echo "<p>The maximum size is ", MAX_SIZE, "</p>";`
- Unlike variables, you cannot include the constant name within the quotation marks that surround a text string.

Data Types

- Variables can contain many different kinds of values, for example, the time of day, a dollar amount, or a person's name. A data type is the specific category of information that a variable contains.
- Data types that can be assigned only a single value are called primitive types. PHP supports the five primitive data types:
 - **Integer** numbers that is the set of all positive and negative numbers and zero, with no decimal places
 - **Floating-point** numbers that are positive or negative numbers with decimal places or numbers written using exponential notation.
 - **Boolean** that is a logical value of “true” or “false”
 - **String** that is a text such as “Hello World”
 - **NULL** that is an empty value, also referred to as a NULL value

Reference types

- The PHP language also supports reference (or composite), data types, which can contain multiple values or complex types of information,
- The two reference data types supported by the PHP language are arrays and objects.

Loosely typed programming languages.

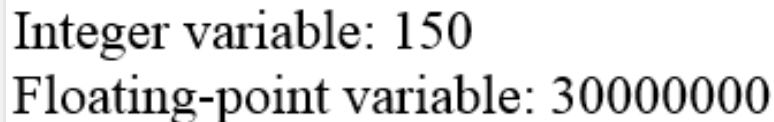
- PHP is a loosely typed programming language (or dynamic typing)
- In PHP, you are not required to declare the data type of variables. Because the PHP scripting engine automatically determines what type of data is stored in a variable and assigns the variable's data type accordingly.

- Example,

```
$ChangingVariable = "Hello World"; // String
$ChangingVariable = 8;           // Integer number
$ChangingVariable = 5.367;       // Floating-point number
$ChangingVariable = TRUE;        // Boolean
$ChangingVariable = NULL;        // NULL
```

- Example,

```
$IntegerVar = 150;
$FloatingPointVar = 3.0e7; // floating-point number 30000000
echo "<p>Integer variable: $IntegerVar<br />";
echo "Floating-point variable: $FloatingPointVar</p>";
```

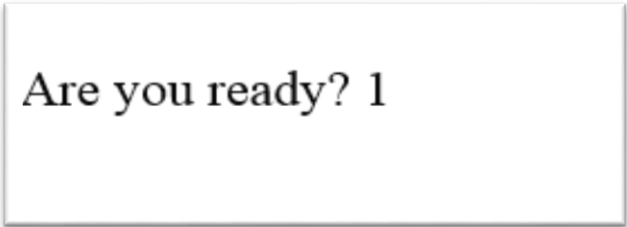


```
Integer variable: 150
Floating-point variable: 30000000
```


Boolean Values

- A Boolean value is a value of “true” or “false”.
- In PHP programming, however, you can only use the words TRUE or FALSE to indicate Boolean values. PHP then converts the values TRUE and FALSE to the integers 1 and 0.

```
<?php
    $isReady = TRUE;
    echo "<p>Are you ready? $isReady</p>";
?>
```



Are you ready? 1

Arrays

- An array is a set of variables represented by a single variable name.
- In PHP, you can create numerically indexed arrays and associative arrays.
- An element refers to a single piece of data that is stored within an array. By default, the numbering of elements within a PHP array starts with an index number of zero (0).
- You create an array using the `array()` construct or by using the array name and brackets. The `array()` construct uses the following syntax: `$array_name = array(values);`
`$cities= array("San Francisco", "San Jose", "Fremont", "New York",
"Boston ", "Los Angles ", "Chicago");`
- To add more elements to an existing array using statements that include the array name and brackets:
`$cities[] = "Mountain View";`
`$cities[] = "Santa Cruz";`

Arrays

- In PHP, the values assigned to different elements of the same array can be of different data types

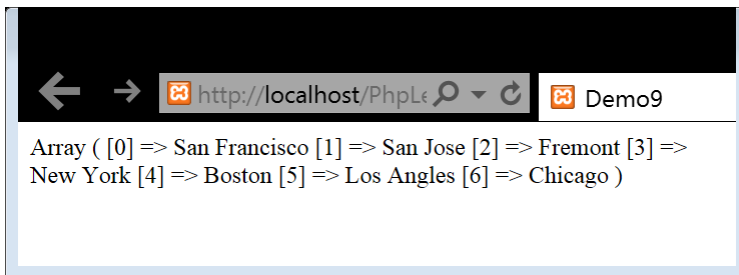
```
$StudentRecord= array(  
    "Peter Lee", // student name (string)  
    25,          // age(integer)  
    4.0,         // GPA (floating-point)  
    true);       // graduate student (Boolean)
```

- To access an element's value, you include brackets and the element index.
- To find the total number of elements in an array, use the `count()` function. You pass to the `count()` function.

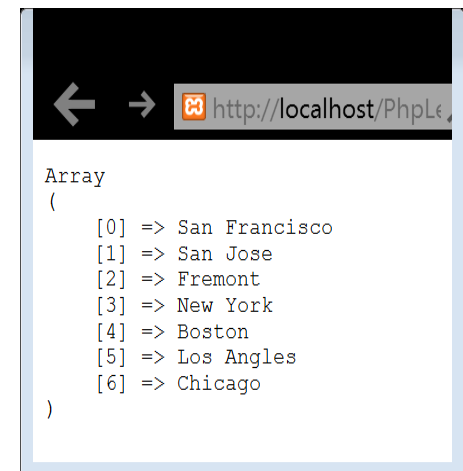
```
echo "<p>The array cities has ", count($cities), " cities.</p>";
```
- PHP includes the `print_r()`, `var_export()`, and `var_dump()` functions, which you can use to display or return information about variables.

Example of Arrays

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo9</title>
  </head>
  <body>
    <?php
      $cities = array(
        "San Francisco",
        "San Jose",
        "Fremont",
        "New York",
        "Boston ",
        "Los Angles ",
        "Chicago");
      print_r($cities);
    ?>
  </body>
</html>
```



```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo9</title>
  </head>
  <body>
    <?php
      $cities = array(
        "San Francisco",
        "San Jose",
        "Fremont",
        "New York",
        "Boston ",
        "Los Angles ",
        "Chicago");
      echo "<pre>";
      print_r($cities);
      echo "</pre>"
    ?>
  </body>
</html>
```

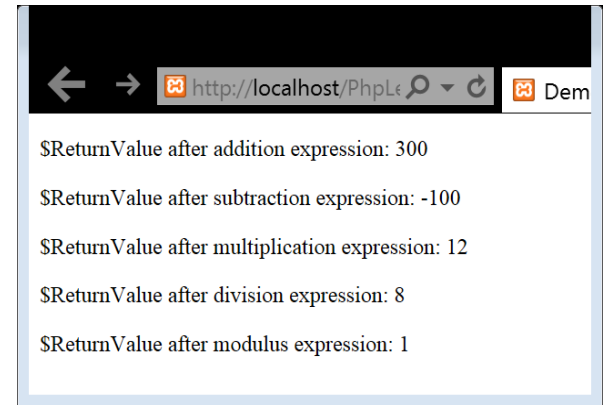


Expressions

- An expression is a literal value or variable (or a combination of literal values, variables, operators, and other expressions) that can be evaluated by the PHP scripting engine to produce a result.

Example of Arithmetic Operators

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo9</title>
  </head>
  <body>
    <?php
      // ADDITION
      $x = 100;
      $y = 200;
      $ReturnValue = $x + $y; //$ ReturnValue is assigned the value 300
      echo '<p>$ReturnValue after addition expression: ', $ReturnValue, "</p>";
      // SUBTRACTION $x = 10; $y = 7;
      $ReturnValue = $x - $y; // $ReturnValue changes to 3
      echo '<p>$ReturnValue after subtraction expression: ', $ReturnValue, "</p>";
      // MULTIPLICATION
      $x = 2;
      $y = 6;
      $ReturnValue = $x * $y; // $ReturnValue changes to 12
      echo '<p>$ReturnValue after multiplication expression: ', $ReturnValue, "</p>";
      // DIVISION
      $x = 24;
      $y = 3;
      $ReturnValue = $x / $y; // $ReturnValue changes to 8
      echo '<p>$ReturnValue after division expression: ', $ReturnValue, "</p>";
      // MODULUS
      $x = 3;
      $y = 2;
      $ReturnValue = $x % $y; // $ReturnValue changes to 1
      echo '<p>$ReturnValue after modulus expression: ', $ReturnValue, "</p>";
    ?>
  </body>
</html>
```



Comparison Operators

- **Comparison** operators are used to determine how one operand compares to another. A Boolean value of TRUE or FALSE is returned after two operands are compared.
- Symbol Operation Description

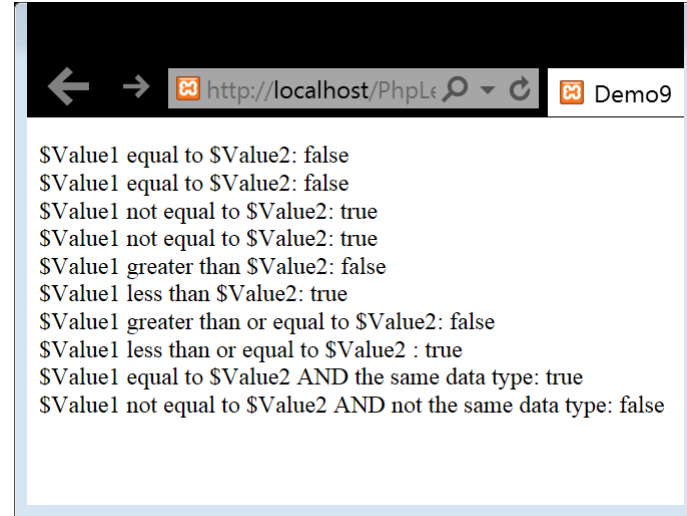
Symbol	Meaning
==	Equal - Returns TRUE if the operands are equal
===	Strict equal - Returns TRUE if the operands are equal and of the same data type
!= or <>	Not equal - Returns TRUE if the operands are not equal
!==	Strict not equal - Returns TRUE if the operands are not equal or not of the same data type
>	Greater than - Returns TRUE if the left operand is greater than the right operand
<	Less than - Returns TRUE if the left operand is less than the right operand
>=	Greater than or equal to - Returns TRUE if the left operand is greater than or equal to the right operand
<=	Less than or equal to - Returns TRUE if the left operand is less than or equal to the right operand

Comparing Operands in PHP

- When two operands are numeric values, the PHP scripting engine compares them numerically.
- When two non-numeric values are used as operands, the PHP scripting engine compares them in alphabetical order.
 - `$ReturnValue = "b" > "a";` returns TRUE
- When one operand is a number and the other is a string, the PHP scripting engine attempts to convert the string value to a number. If the string value cannot be converted to a number, a value of FALSE is returned.
 - `$ReturnValue = 10 == "ten";` returns a value of FALSE
 - `$ReturnValue = 10 == "10";` returns a value of TRUE

Example of Comparison Operators

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo9</title>
  </head>
  <body>
    <?php
      $Value1 = "first text string";
      $Value2 = "second text string";
      $ReturnValue = ($Value1 == $Value2 ? "true" : "false");
      echo '<p>$Value1 equal to $Value2: ', $ReturnValue, "<br />";
      $Value1 = 50; $Value2 = 75;
      $ReturnValue = ($Value1 == $Value2 ? "true" : "false");
      echo '$Value1 equal to $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 != $Value2 ? "true" : "false");
      echo '$Value1 not equal to $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 <> $Value2 ? "true" : "false");
      echo '$Value1 not equal to $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 > $Value2 ? "true" : "false");
      echo '$Value1 greater than $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 < $Value2 ? "true" : "false");
      echo '$Value1 less than $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 >= $Value2 ? "true" : "false");
      echo '$Value1 greater than or equal to $Value2: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 <= $Value2 ? "true" : "false");
      echo '$Value1 less than or equal to $Value2: ', $ReturnValue, "<br />";
      $Value1 = 25;
      $Value2 = 25;
      $ReturnValue = ($Value1 === $Value2 ? "true" : "false");
      echo '$Value1 equal to $Value2 AND the same data type: ', $ReturnValue, "<br />";
      $ReturnValue = ($Value1 !== $Value2 ? "true" : "false");
      echo '$Value1 not equal to $Value2 AND not the same data type: ', $ReturnValue, "</p>";
    ?>
  </body>
</html>jj
```

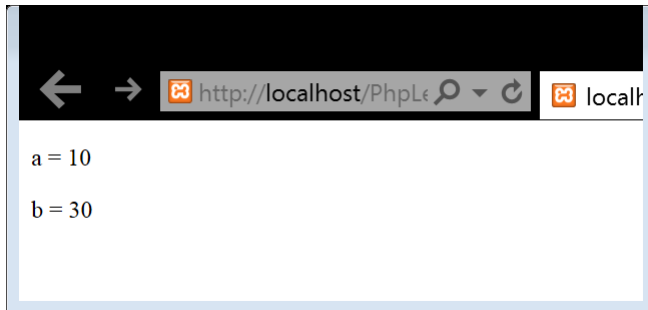


Logical Operators

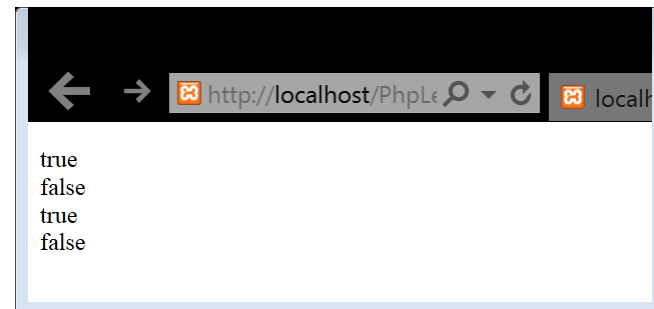
- Logical operators are used for comparing two Boolean operands for equality. Boolean operands are operands that are limited to the values TRUE or FALSE.
 - && or AND
 - Logical And - Returns TRUE if both the left operand and right operand return a value of TRUE; otherwise, it returns a value of FALSE
 - || or OR
 - Logical Or - Returns TRUE if either the left operand or right operand returns a value of TRUE; otherwise, it returns a value of FALSE
 - XOR
 - Logical Exclusive Or - Returns TRUE if only one of the left operand or right operand returns a value of TRUE; otherwise it returns a value of FALSE
 - !
 - Logical Not Returns TRUE if an expression is FALSE and returns FALSE if an expression is TRUE

Example of Logical Operators

```
<?php
$a = 10;
$b = 20;
if ($a > 10 || $b = 30)
    echo "<p>a = $a</p>";
echo "<p>b = $b</p>";
?>
```



```
<?php
$TrueValue = true;
$FalseValue = false;
$ReturnValue = ($TrueValue ? "true" : "false");
echo "<p>$ReturnValue<br />";
$ReturnValue = ($FalseValue ? "true" : "false");
echo "$ReturnValue<br />";
$ReturnValue = ($TrueValue || $FalseValue ? "true" : "false");
echo "$ReturnValue<br />";
$ReturnValue = ($TrueValue && $FalseValue ? "true" : "false");
echo "$ReturnValue<br />";
echo "</p>";
?>
```



Type Casting

- Even though PHP automatically assigns the data type of a variable, sometimes you want to ensure that a variable is of the data type expected by your script. One way to ensure this is through **casting**, or **type casting**, which copies the value contained in a variable of one data type into a variable of another data type.
 - `$NewVariable = (new_type) $OldVariable;`
 - Example

```
$SpeedLimitMiles = "55.8 mph";  
$SpeedLimitKilometers = (int) $SpeedLimitMiles * 1.6;  
echo "$SpeedLimitMiles is equal to $SpeedLimitKilometers kph";
```

The gettype() function

- Instead of just guessing data types, you can view a variable's type by using the gettype() function, which returns one of the following strings, depending on the data type:
 - Boolean
 - Integer
 - Double
 - String
 - Array
 - Object
 - Resource
 - NULL
 - Unknown type
- Example.

```
$MortgageRate = .0575;  
echo gettype($MortgageRate); // double
```
- Another way is to use one of the 15 is_*() functions that test for various kinds of data types. E.g. is_numeric(), is_string(), is_int() etc.
- Example,

```
$MortgageRate = .0575;  
$Result = ((is_double($MortgageRate)) ?  
    "The variable contains a decimal number." :  
    "The variable does not contain a decimal number.");  
echo $Result;
```

Exercises

- Question 1:
 - We can use the `round()`, `ceil()`, and `floor()` functions to round a fraction up or down to the nearest integer. The `round()` function rounds a fraction to the nearest integer, the `ceil()` function rounds a fraction up to the nearest integer, and the `floor()` function rounds a fraction down to the nearest integer. Write a PHP script that demonstrates the use of these functions. Save the document as `RoundedNumbers.php`.
- Question 2:
 - Write a PHP script that uses a comparison operator to determine whether a variable contains a number and whether the number is even. You need to use the `is_numeric()` function and the comparison operator. For floating-point numbers, you need to use the `round()` function to convert the value to the nearest whole number. Save the document as `FindEvenNumbers.php`.
- Question 3:
 - Write a PHP script that assigns the names of months to an array named `$Months[]`. Use output statements to display “The names of the months are: ” along with the values in the `$Months[]` array. Following the output statements, reassign the values in the `$Months[]` array with the month number. January is 1, February is 2, March is 3, and so on. Then use output statements to display “The Month numbers are: ” along with the values in the `$Months[]` array. Save the document as `MonthsArray.php`.