PHP 5.x : MySQL

PHP: Hypertext Preprocessor

### PHP Code Parsed

In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

### PHP embedded with HTML

/index.php

|  |
| --- |
| <html>   <head>  <title>Hello World</title>  </head>   <body>  <?php **echo** "Hello, World!";?>  </body> </html> |

### View PHP/HTML in Browser

Hello, World!

(check page view source)

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

|  |
| --- |
| <?php PHP code goes here ?>  <? PHP code goes here ?>  <script language = "php"> PHP code goes here </script> |

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

From the next chapter we will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

#### Data Types

* The main way to store information in the middle of a PHP program is by using a variable.
* Here are the most important things to know about variables in PHP.
* All variables in PHP are denoted with a leading dollar sign ($).
* The value of a variable is the value of its most recent assignment.
* Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
* Variables can, but do not need, to be declared before assignment.
* Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
* Variables used before they are assigned have default values.
* PHP does a good job of automatically converting types from one to another when necessary.

PHP has a total of eight data types which we use to construct our variables −

* **Integers** − are whole numbers, without a decimal point, like 4195.
* **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
* **Booleans** − have only two possible values either true or false.
* **NULL** − is a special type that only has one value: NULL.
* **Strings** − are sequences of characters, like 'PHP supports string operations.'
* **Arrays** − are named and indexed collections of other values.
* **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
* **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

### 

### Integer

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so −

$a = 12345;

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

|  |
| --- |
| <?php  $x = 5985; var\_dump($x); ?> |

### Double

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code −

|  |
| --- |
| <?php  $many = 2.2888800;  $many\_2 = 2.2111200;  $few = $many + $many\_2;    **print**("$many + $many\_2 = $few <br>"); ?> |

### 

### Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

|  |
| --- |
| **if** (**TRUE**)  **print**("This will always print<br>"); **else**  **print**("This will never print<br>"); |

**Interpreting other types as Booleans**

Here are the rules for determine the "truth" of any value not already of the Boolean type −

* If the value is a number, it is false if exactly equal to zero and true otherwise.
* If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
* Values of type NULL are always false.
* If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
* Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

### NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this −

|  |
| --- |
| $my\_var = **NULL**; |

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed −

|  |
| --- |
| $my\_var = **null**; |

A variable that has been assigned NULL has the following properties −

* It evaluates to FALSE in a Boolean context.
* It returns FALSE when tested with IsSet() function.

### 

### Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

* $string\_1 = "This is a string in double quotes";
* $string\_2 = 'This is a somewhat longer, singly quoted string';
* $string\_39 = "This string has thirty-nine characters";
* $string\_0 = ""; // a string with zero characters

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

|  |
| --- |
| <?php  $variable = "name";  $literally = 'My $variable will not print!';    **print**($literally);  **print** "<br>";    $literally = "My $variable will print!";  **print**($literally); ?> |

This will produce following result −

|  |
| --- |
| My $variable will not **print**! My name will **print** |

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

Certain character sequences beginning with backslash (\) are replaced with special characters

Variable names (starting with $) are replaced with string representations of their values.

**The escape-sequence replacements are −**

* \n is replaced by the newline character
* \r is replaced by the carriage-return character
* \t is replaced by the tab character
* \$ is replaced by the dollar sign itself ($)
* \" is replaced by a single double-quote (")
* \\ is replaced by a single backslash (\)

#### Operators

**What is Operator?** Simple answer can be given using expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

### Arithmetic Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiply both operands | A \* B will give 200 |
| / | Divide numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | Increment operator, increases integer value by one | A++ will give 11 |
| -- | Decrement operator, decreases integer value by one | A-- will give 9 |

### Comparison Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the value of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

### Logical (or Relational) Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and | Called Logical AND operator. If both the operands are true then condition becomes true. | (A and B) is true. |
| or | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A or B) is true. |
| && | Called Logical AND operator. If both the operands are non zero then condition becomes true. | (A && B) is true. |
| || | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is false. |

### Assignment Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator, Assigns values from right side operands to left side operand | C = A + B will assign value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand | C -= A is equivalent to C = C - A |
| \*= | Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand | C \*= A is equivalent to C = C \* A |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand | C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand | C %= A is equivalent to C = C % A |

### Conditional (or ternary) Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| ? : | Conditional Expression | If Condition is true ? Then value X : Otherwise value Y |

### PHP variables & Scope : Statics and global variables

### Comments in PHP