

Content

- **Php Comments**
- **Variables**
- **Echo**
- **Arithmetic Operation**
- **Data Types**

PHP syntax template

HTML content

```
<?php
```

PHP code

```
?>
```

HTML content

```
<?php
```

PHP code

```
?>
```

HTML content ...

PHP

- any contents of a .php file between `<?php` and `?>` are executed as PHP code
- all other contents are output as pure HTML

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 `//`
 - we recommend `#` and will use it in our examples

Comments in PHP

Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment  
  
# Shell-style comments  
  
/* C-style comments  
    These can span multiple lines */
```

Variables in PHP

- PHP variables must begin with a "\$" sign
- Global and locally-scoped variables
 - Global variables can be used anywhere
 - Local variables restricted to a function or class
- Certain variable names reserved by PHP
 - Form variables (\$_POST, \$_GET)
 - Server variables (\$_SERVER)
 - Etc.

Variables

```
$name = expression;
```

PHP

```
$user_name = "PinkHeartLuvr78";
```

```
$age = 16;
```

```
$drinking_age = $age + 5;
```

```
$this_class_rocks = TRUE;
```

PHP

- names are case sensitive; Case-sensitive (\$Foo != \$foo != \$fOo)
- Separate multiple words with _
- names always begin with \$, on both declaration and usage
- implicitly declared by assignment (type is not written; a "loosely typed" language)

Variable usage

```
<?php
$foo = 25;          // Numerical variable
$bar = "Hello";    // String variable

$foo = ($foo * 7);  // Multiplies foo by 7
$bar = ($bar * 7);  // Invalid expression
?>
```

Echo

The PHP command '**echo**' is used to output the parameters passed to it

- The typical usage for this is to send data to the client's web-browser

Syntax

- void **echo** (string **arg1** [, string **arg n** ...])
- In practice, arguments are not passed in parentheses since **echo** is a language construct rather than an actual function

Concatenation

Use a period to join strings into one.

```
<?php
$string1="Hello";
$string2="PHP";
$string3=$string1 . " " . $string2;
Print $string3;
?>
```

```
Hello PHP
```

Echo example

```
<?php
$foo = 25;          // Numerical variable
$bar = "Hello";     // String variable

echo $bar;          // Outputs Hello
echo $foo,$bar;     // Outputs 25Hello
echo "5x5=", $foo;  // Outputs 5x5=25
echo "5x5=$foo";    // Outputs 5x5=25
echo `5x5=$foo`;    // Outputs 5x5=$foo
?>
```

Notice how echo ``5x5=$foo`` outputs `$foo` rather than replacing it with 25

Strings in single quotes (`` ``) are not interpreted or evaluated by PHP

This is true for both variables and character escape-sequences (such as `"\n"` or `"\\"`)

Arithmetic Operations

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

Arithmetic Operations

```
<?php
    $a=15;
    $b=30;
    $total=$a+$b;
    Print $total;
    Print "<p><h1>$total</h1>";
    // total is 45
?>
```

\$a - \$b // subtraction

\$a * \$b // multiplication

\$a / \$b // division

\$a += 5 // \$a = \$a+5 Also works for *= and /=

bool (Boolean) type

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

PHP

- the following values are considered to be FALSE (all others are TRUE):
 - 0 and 0.0
 - "", "0", and NULL (includes unset variables)
 - arrays with 0 elements
- can cast to boolean using (bool)
- FALSE prints as an empty string (no output); TRUE prints as a 1

Math operations

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

PHP

<u>abs</u>	<u>ceil</u>	<u>cos</u>	<u>floor</u>	<u>log</u>	<u>log10</u>	<u>max</u>
<u>min</u>	<u>pow</u>	<u>rand</u>	<u>round</u>	<u>sin</u>	<u>sqrt</u>	<u>tan</u>

math functions

M_PI	M_E	M_LN2
------	-----	-------

math constants

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

NULL

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

- 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)

Printing HTML tags in PHP = bad style

```
<?php
print "<!DOCTYPE html>\n";
print "<html>\n";
print "    <head>\n";
print "        <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
    print "<p class=\"count\"> I can count to $i! </p>\n";
}
?>
```

PHP

- printing HTML tags with print statements is bad style and error-prone:
 - must quote the HTML and escape special characters, e.g. \"
- but without print, how do we insert dynamic content into the page?

PHP expression blocks

<code><?= expression ?></code>	PHP
<code><h2> The answer is <?= 6 * 7 ?> </h2></code>	PHP
The answer is 42	output

- **PHP expression block:** evaluates and embeds an expression's value into HTML
- `<?= expr ?>` is equivalent to `<?php print expr; ?>`

Types

- basic types: int, float, boolean, string, array, object, NULL
 - 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` \rightarrow `int` auto-conversion on `+` (`"1" + 1 == 2`)
 - `int` \rightarrow `float` auto-conversion on `/` (`3 / 2 == 1.5`)
- type-cast with (*type*):
 - `$age = (int) "21";`