PHP :

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
* PHP is forgiving: PHP language tries to be as forgiving as possible.
* PHP Syntax is C-Like.
* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

Example (1):

## "Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML

<html>

<head>

<title>Hello World</title>

</head>

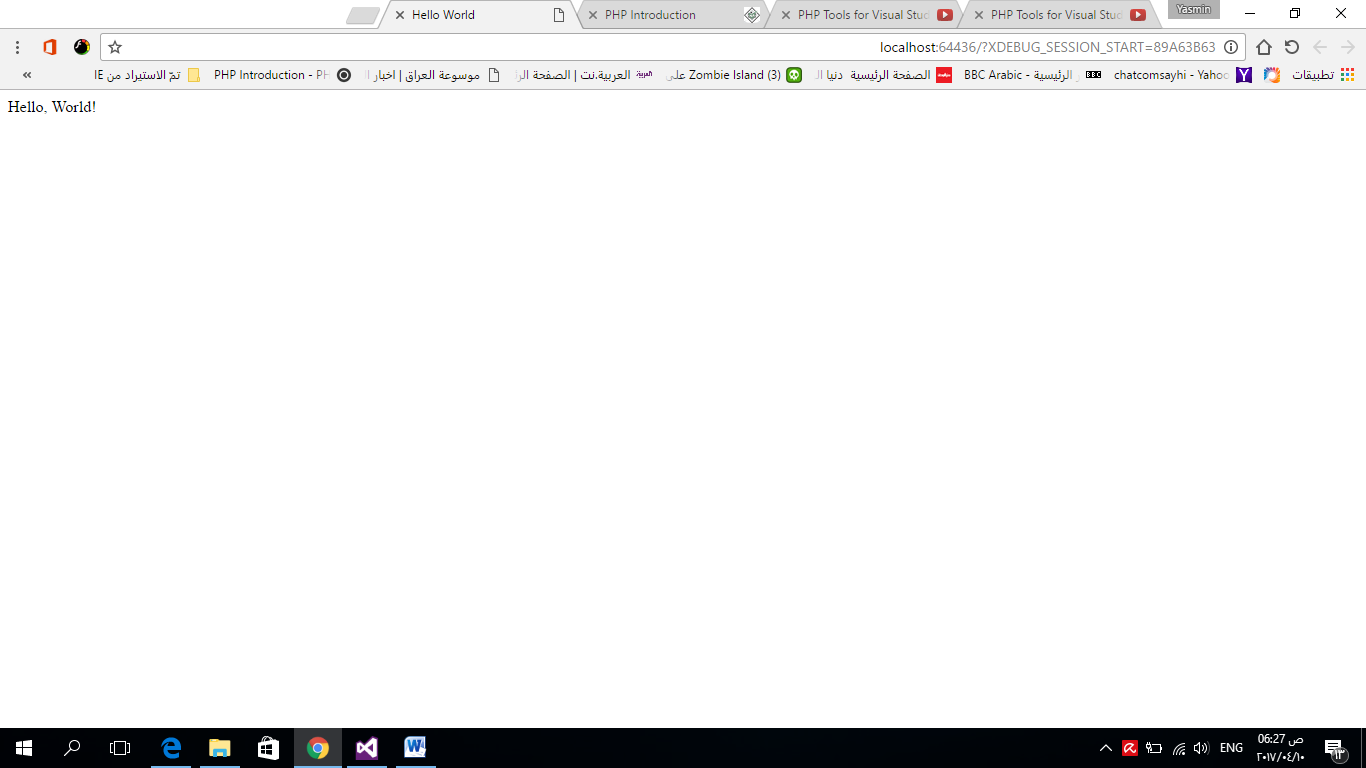
<body>

<?php echo "Hello, World!";?>

</body>

</html>

Output :



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

A most common tag is the <?php...?>

# PHP - Environment Setup

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

* **Web Server** − PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here − <https://httpd.apache.org/download.cgi>
* **Database** − PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here − <https://www.mysql.com/downloads/>
* **PHP Parser** − 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 - Syntax Overview

* The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP'. There are four ways to do this −

### Canonical PHP tags

* The most universally effective PHP tag style is −
* <?php...?>
* If you use this style, you can be positive that your tags will always be correctly interpreted.

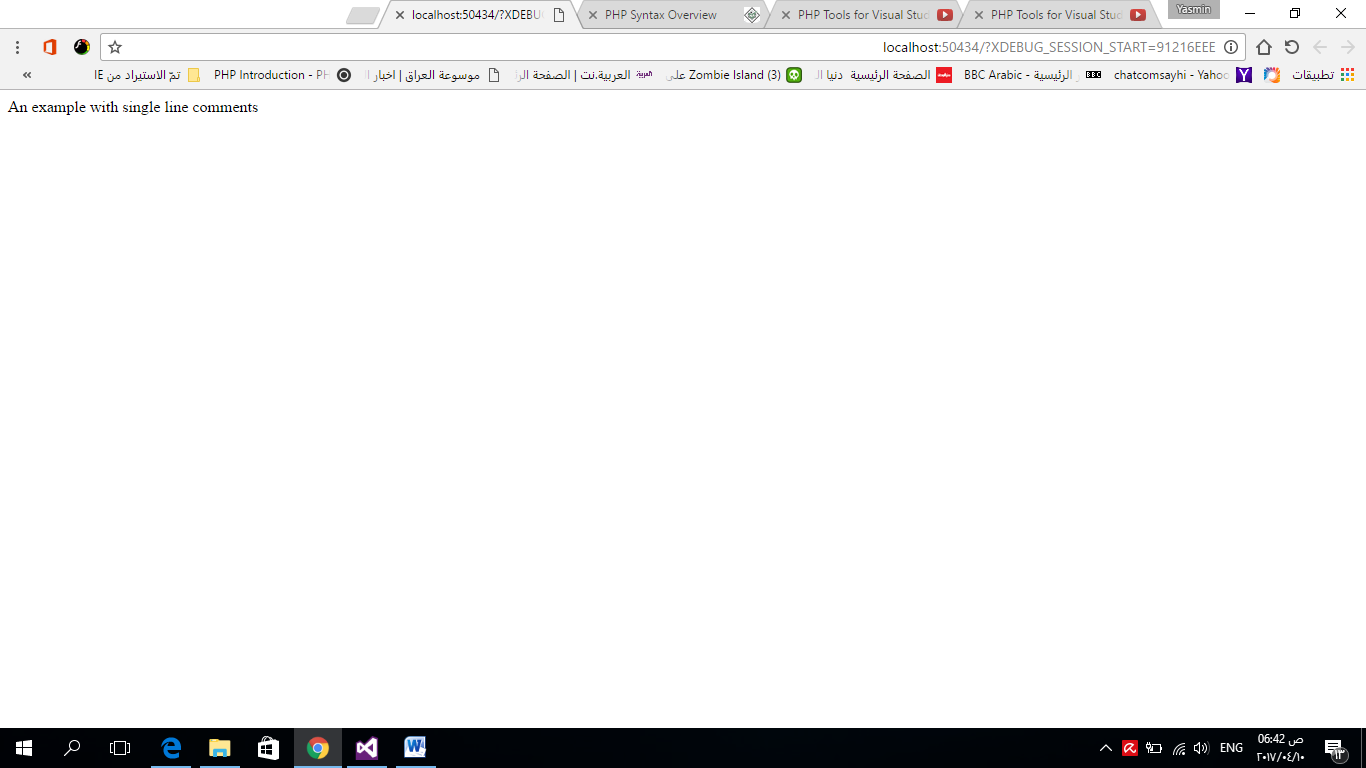
### Commenting PHP Code

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP −

**Single-line comments** − They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

|  |
| --- |
| <?php  # This is a comment, and  # This is the second line of the comment  // This is a comment too. Each style comments  only  print "An example with single line comments";  ?> |

**Output :**



**Multi-lines printing** − Here are the examples to print multiple lines in a single print statement −

|  |
| --- |
| <?php    print "This spans  multiple lines.  The newlines will be  output as well";  ?> |

**Multi-lines comments** − They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here are the example of multi lines comments.

|  |
| --- |
| <?php  /\* This is a comment with multiline  Purpose: Multiline Comments Demo  Subject: PHP  \*/  print "An example with multi line comments";  ?> |

## Output :

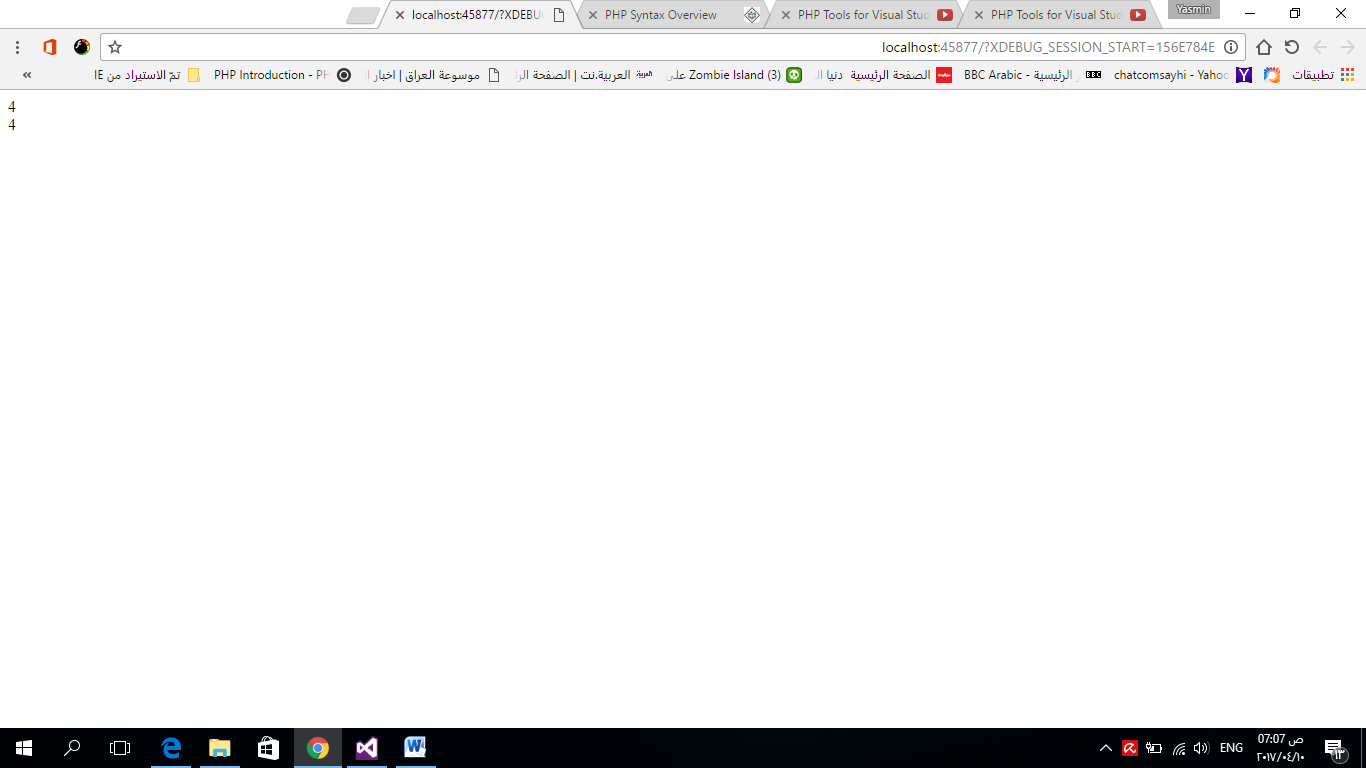
## 

## PHP is whitespace insensitive

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

|  |
| --- |
| <?php  $four = 2 + 2; // single spaces  print $four;  $four=  2+  2; // multiple lines  print" <br>";  print $four;  ?> |

Output :

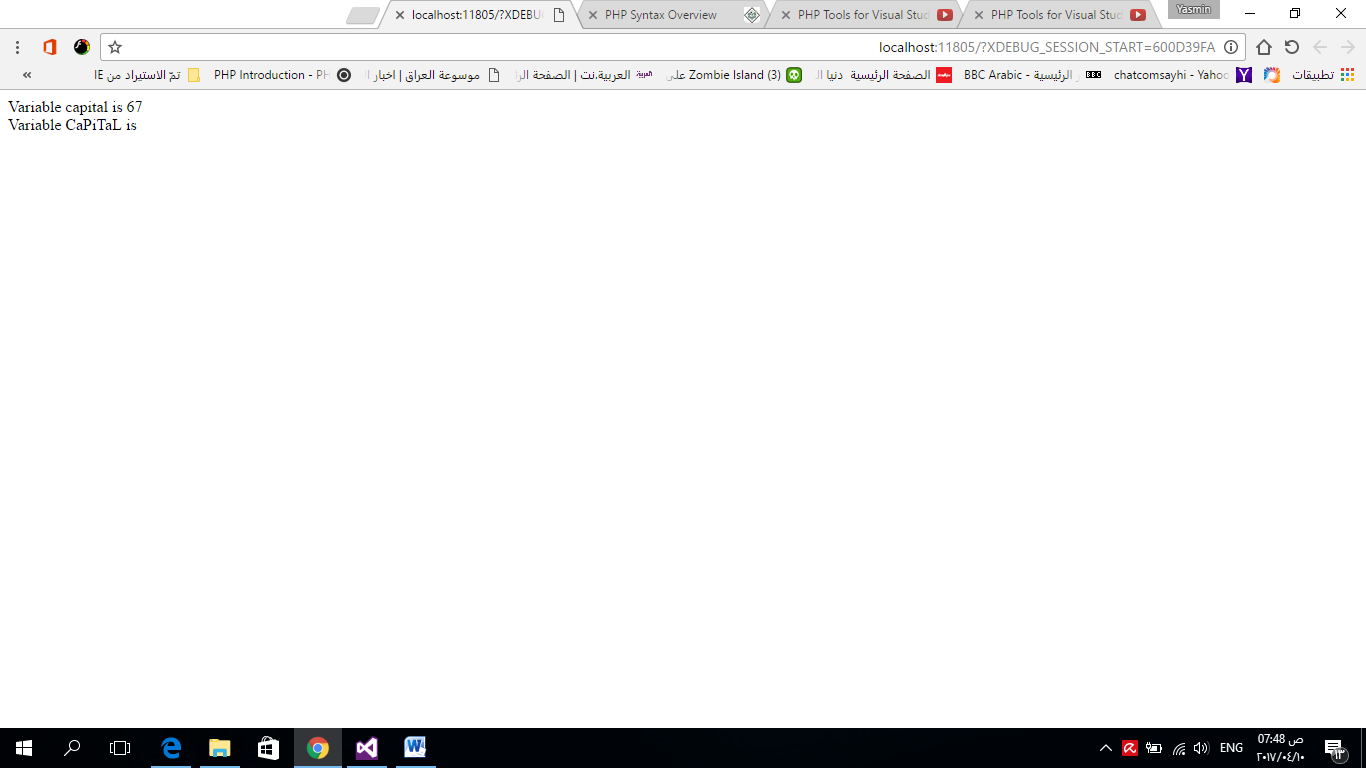


## PHP is case sensitive

Yeah it is true that PHP is a case sensitive language. Try out following example −

|  |
| --- |
| <?php  $capital = 67;  print("Variable capital is **$capital**<br>");  print("Variable CaPiTaL is **$CaPiTaL**<br>");  ?> |

Output :



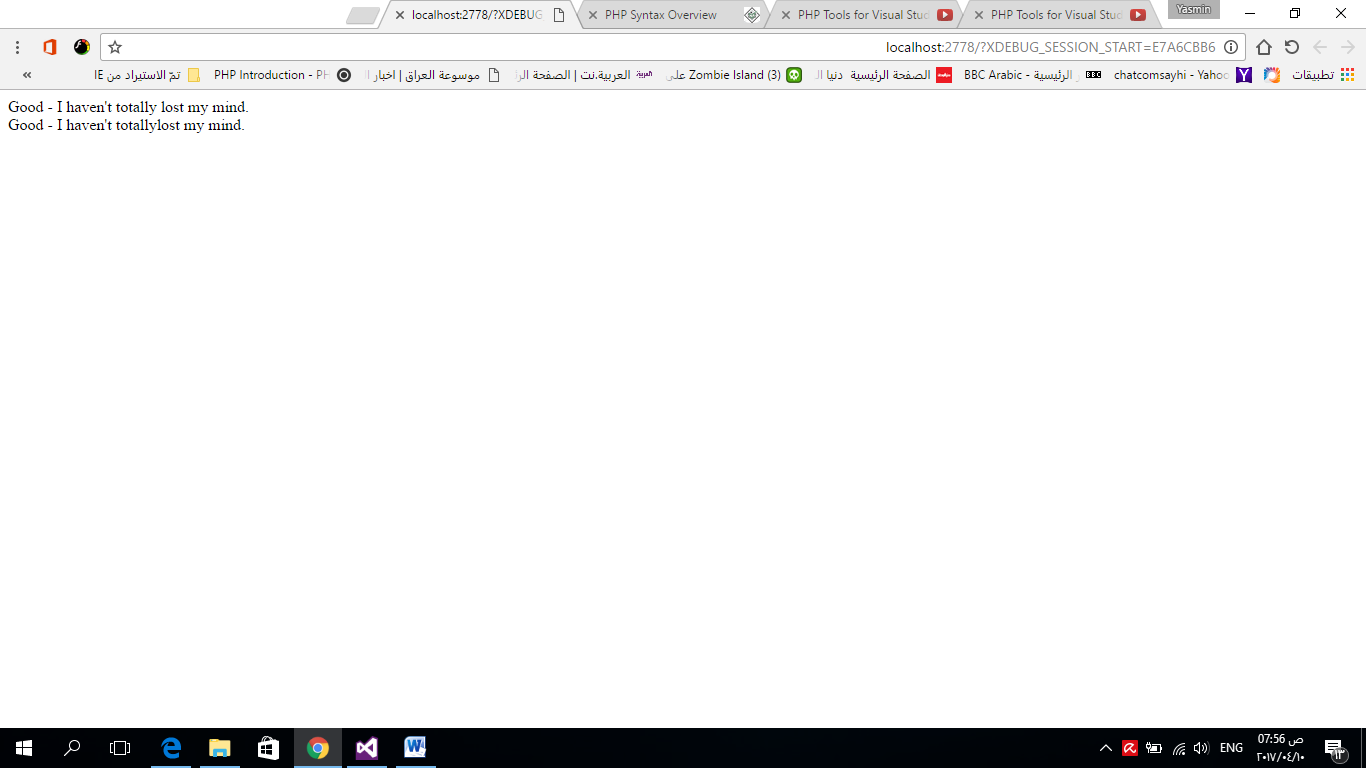
## Braces make blocks

Although statements cannot be combined like expressions, you can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

Here both statements are equivalent −

|  |
| --- |
| <?php  if (3 == 2 + 1)  print("Good - I haven't totally lost my mind.<br>");  if (3 == 2 + 1) {  print("Good - I haven't totally");  print("lost my mind.<br>");  }  ?> |

Output :



# PHP - Variable 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 variables are Perl-like.

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).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

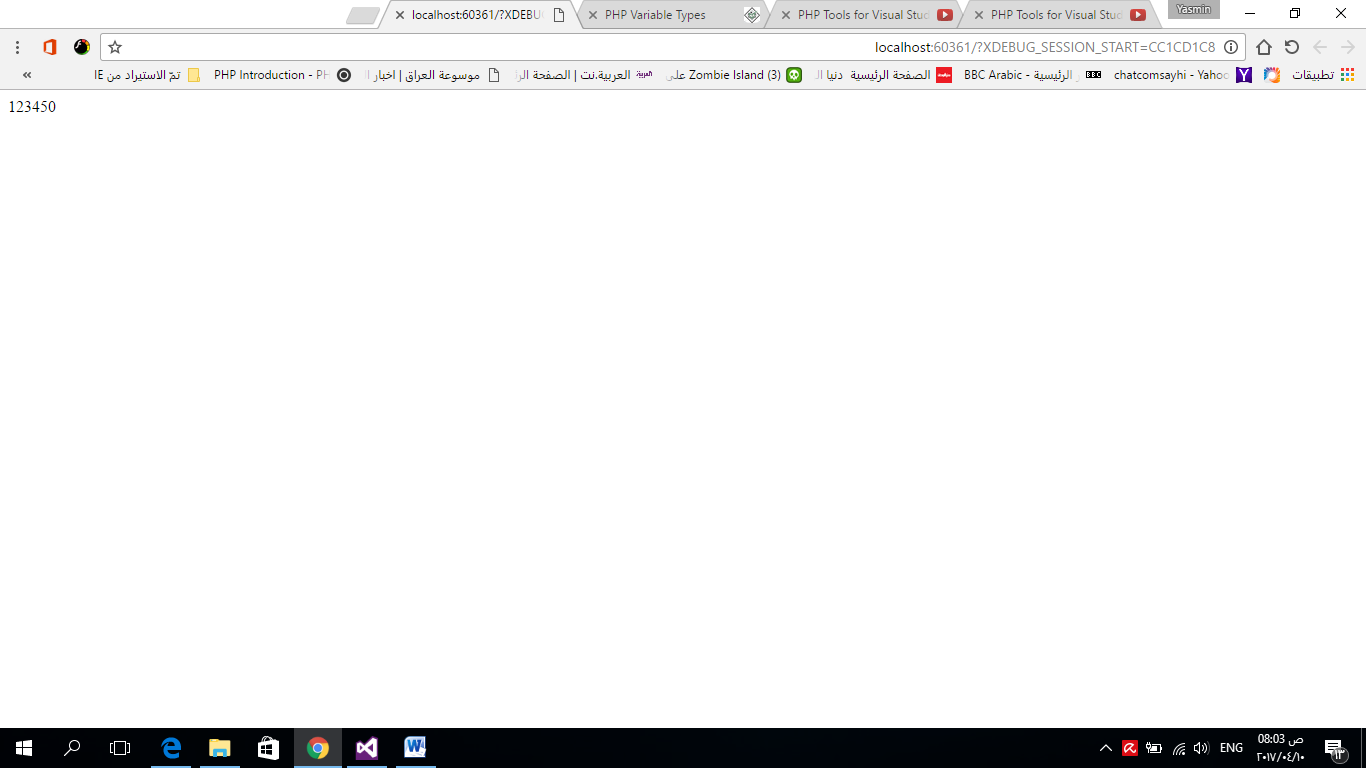
We will explain only simple data type in this chapters. Array and Objects will be explained separately.

## Integers

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 −

|  |
| --- |
| <?php  $int\_var = 12345;  $another\_int = -12345 + 12345;  echo $int\_var;  print $another\_int;  ?>; |

Output "

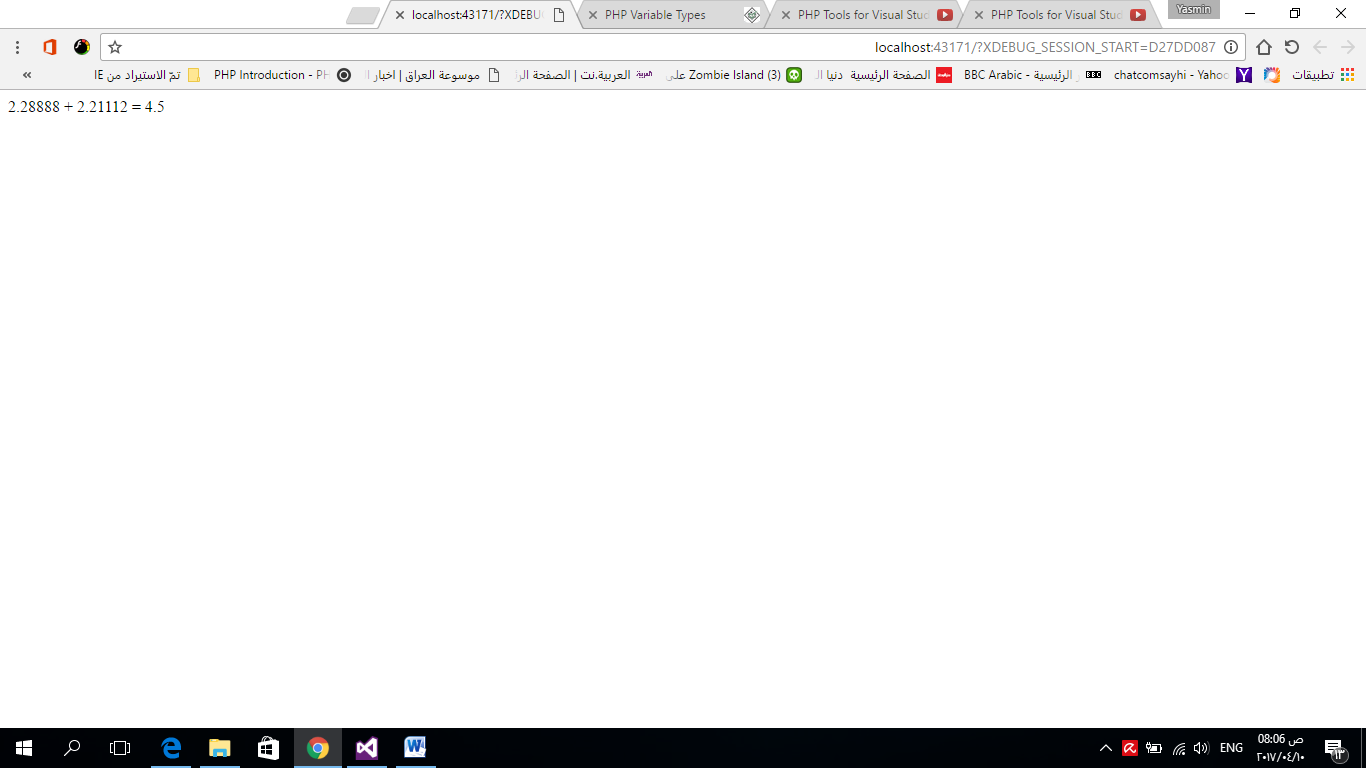


## Doubles

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>");  ?> |

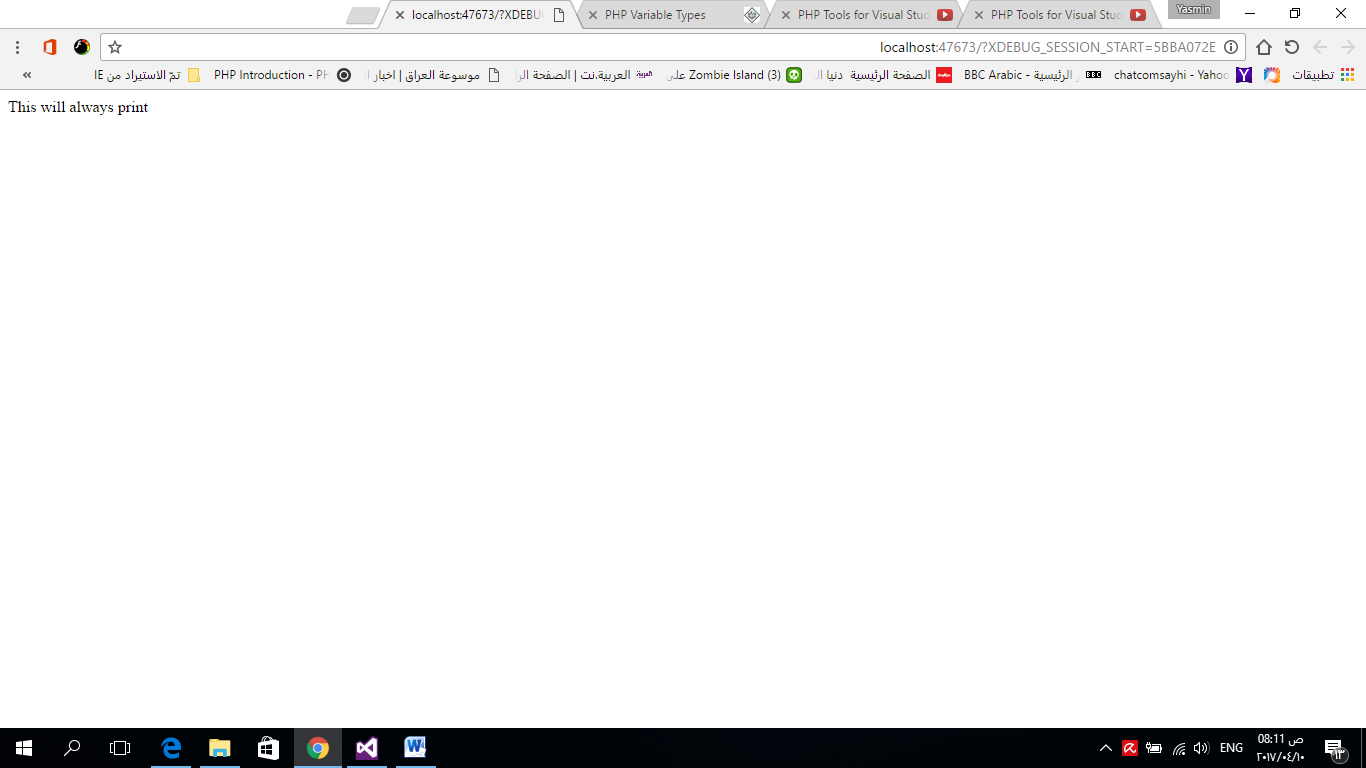
Output :



## 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 −

|  |
| --- |
| <?php  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).
* Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

$true\_num = 3 + 0.14159;

$true\_str = "Tried and true"

$true\_array[49] = "An array element";

$false\_array = array();

$false\_null = NULL;

$false\_num = 999 - 999;

$false\_str = "";

## 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';

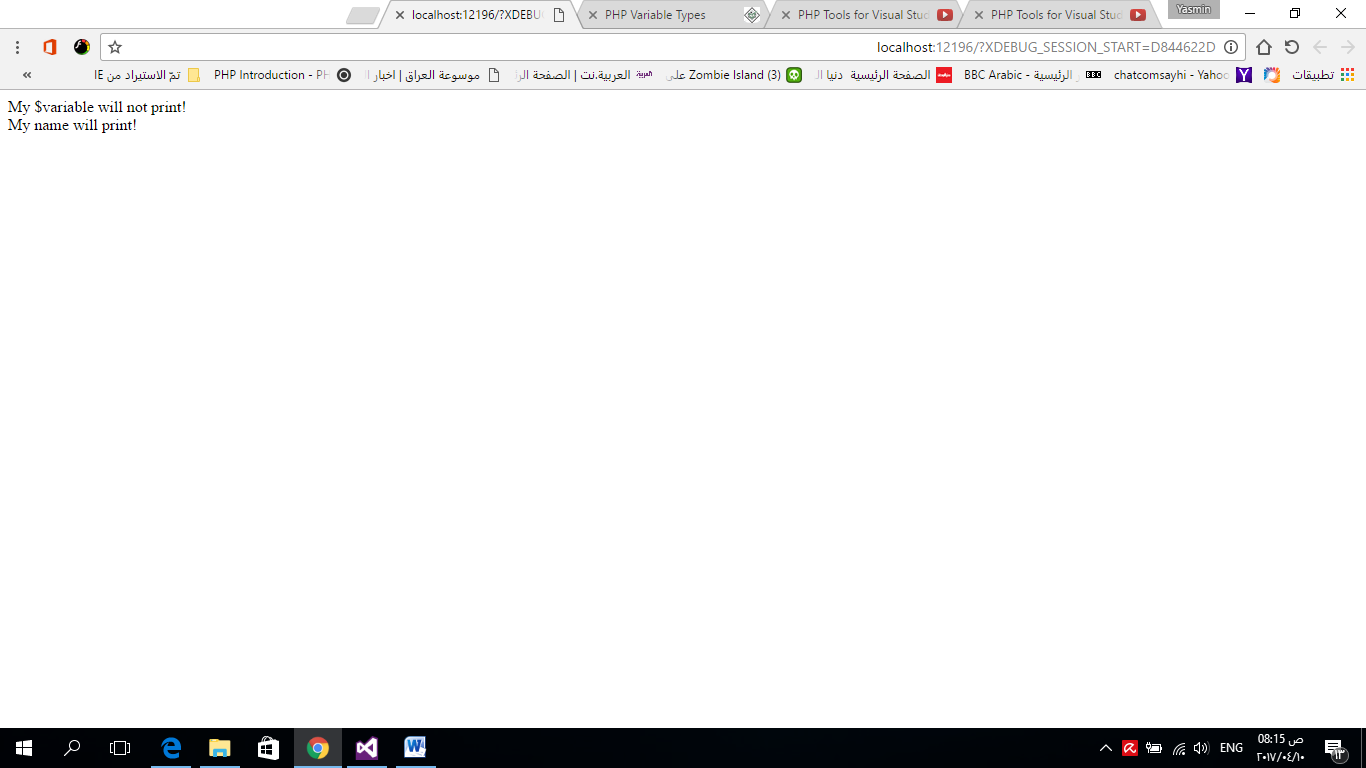
$string\_39 = "This string has thirty-nine characters";

$string\_0 = ""; // a string with zero characters

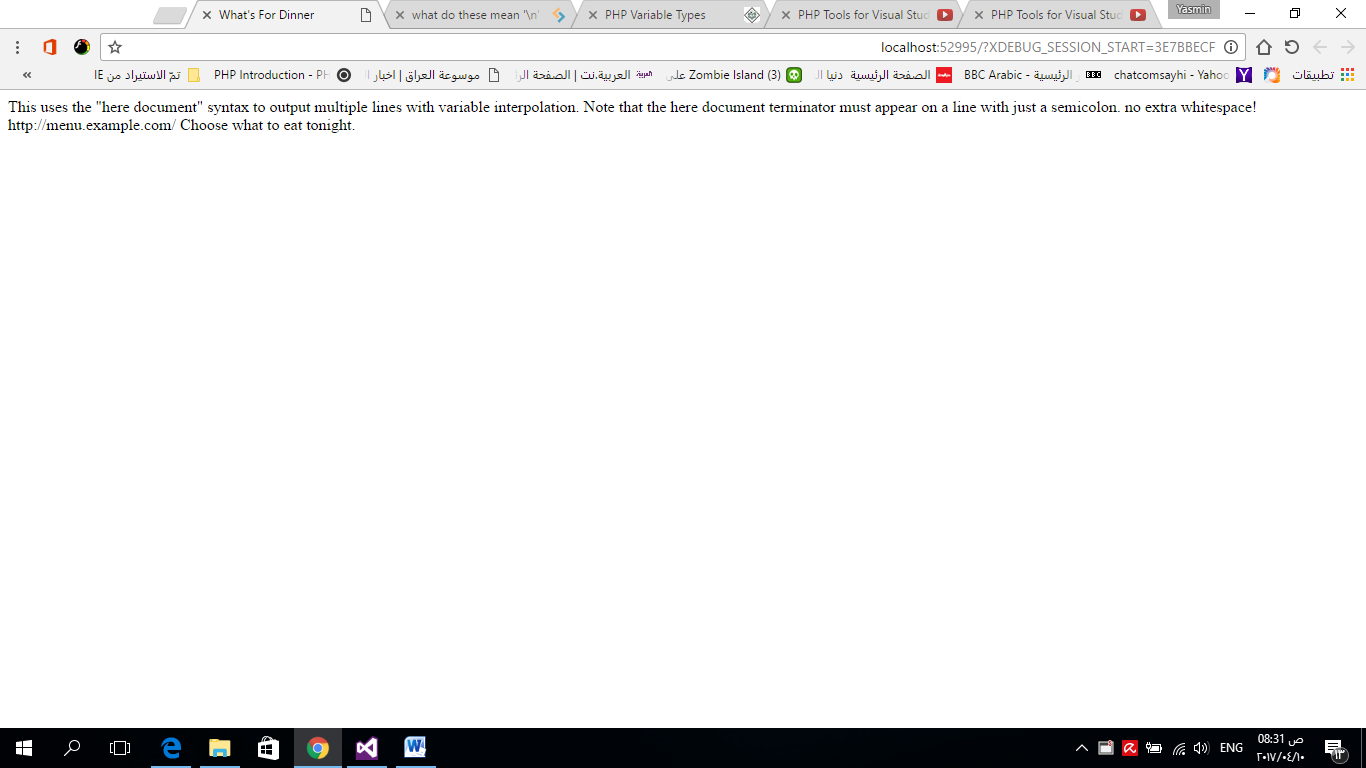
Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

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

Output :



|  |
| --- |
| <?php  $channel =<<<\_XML\_  <channel>  <title>What's For Dinner</title>  <link>http://menu.example.com/ </link>  <description>Choose what to eat tonight.</description>  </channel>  \_XML\_;  echo <<<END  This uses the "here document" syntax to output multiple lines with variable  interpolation. Note that the here document terminator must appear on a line with  just a semicolon. no extra whitespace!  END;  print $channel;  ?> |



Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types −

* Local variables
* Function parameters
* Global variables
* Static variables.

## Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function −

<?php

$x = 4;

function assignx () {

$x = 0;

print "\$x inside function is $x. <br />";

}

assignx();

print "\$x outside of function is $x. <br />";

?>

This will produce the following result −

$x inside function is 0.

$x outside of function is 4.

## Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be −

<?php

// multiply a value by 10 and return it to the caller

function multiply ($value) {

$value = $value \* 10;

return $value;

}

$retval = multiply (10);

Print "Return value is $retval\n";

?>

This will produce the following result −

Return value is 100

## Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example –

|  |
| --- |
| <?php  $somevar = 15;  function addit() {  GLOBAL $somevar;  $somevar++;  print "Somevar is **$somevar**";  }  addit();  ?> |

This will produce the following result −

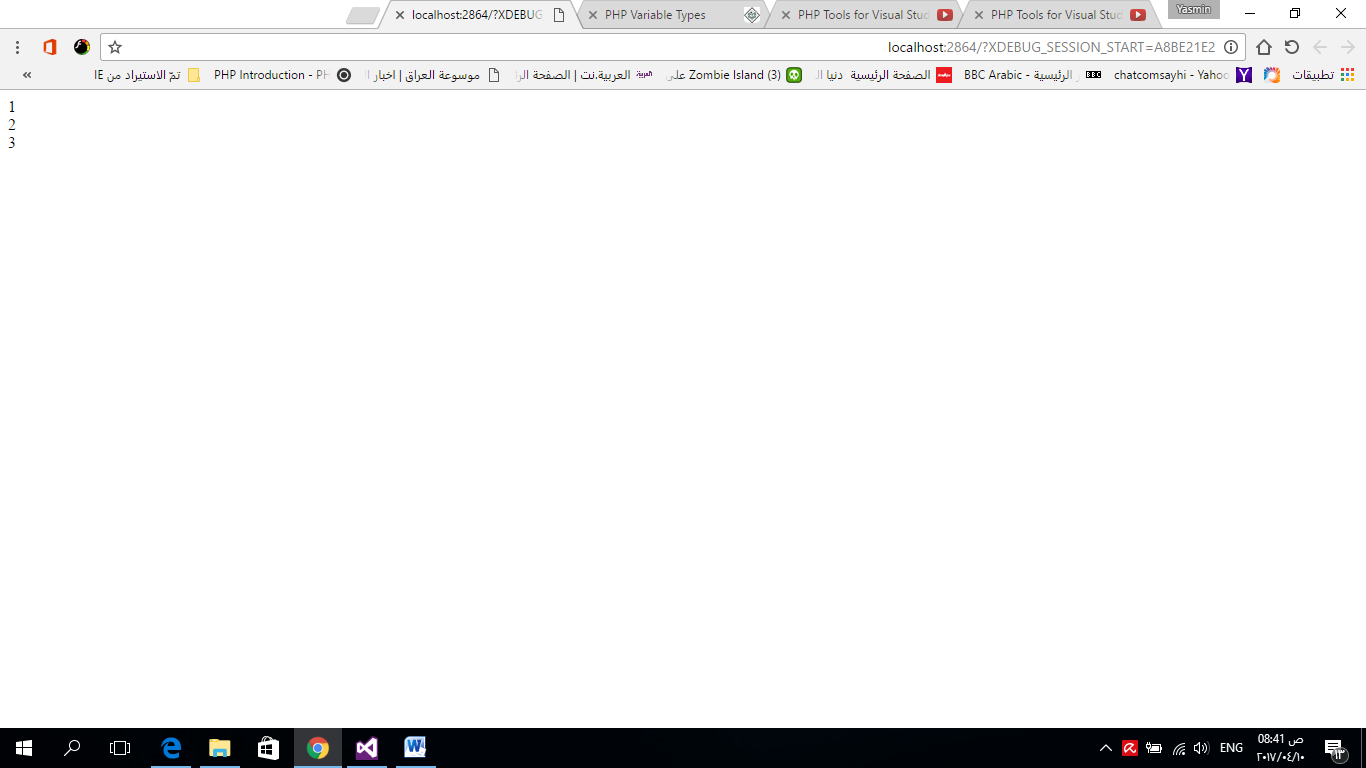
Somevar is 16

## Static Variables

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

|  |
| --- |
| <?php  function keep\_track() {  STATIC $count = 0;  $count++;  print $count;  print "<br />";  }  keep\_track();  keep\_track();  keep\_track();  ?> |



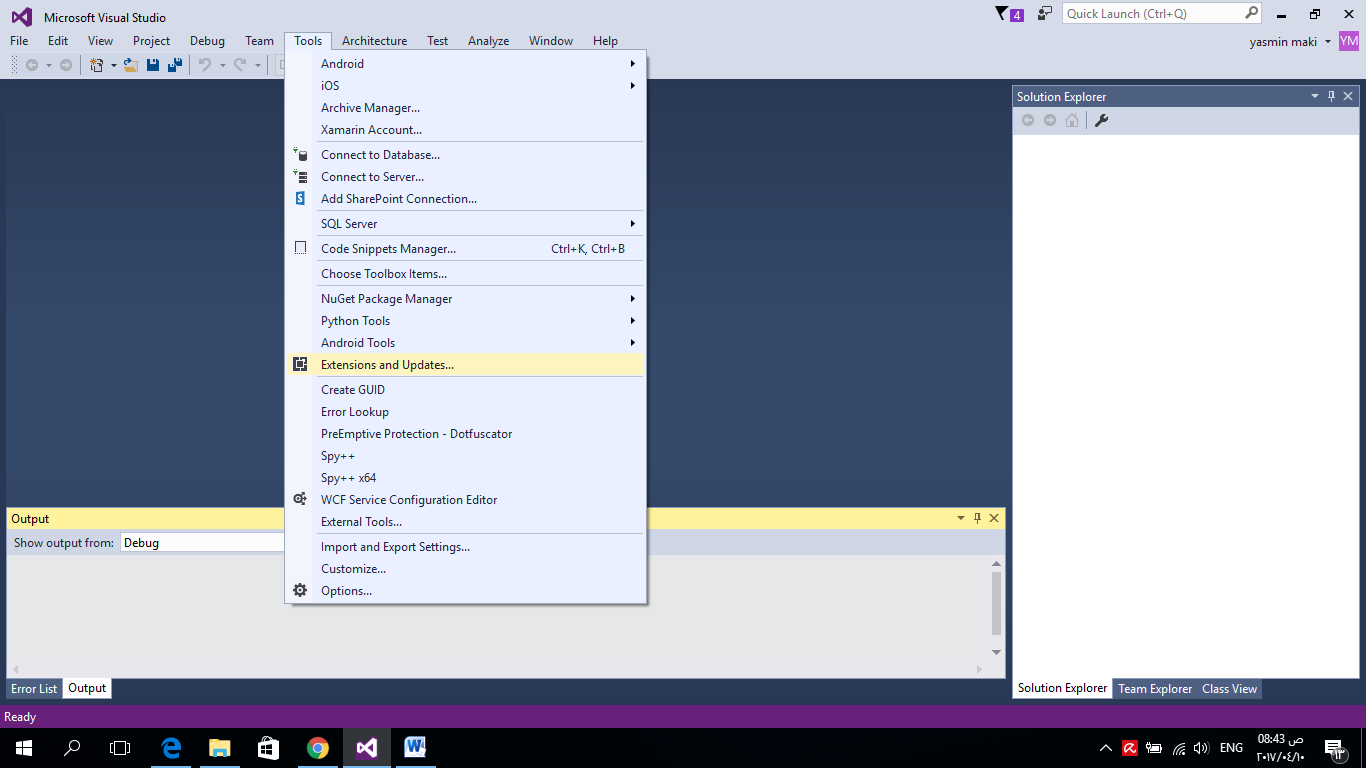
## Variable Naming

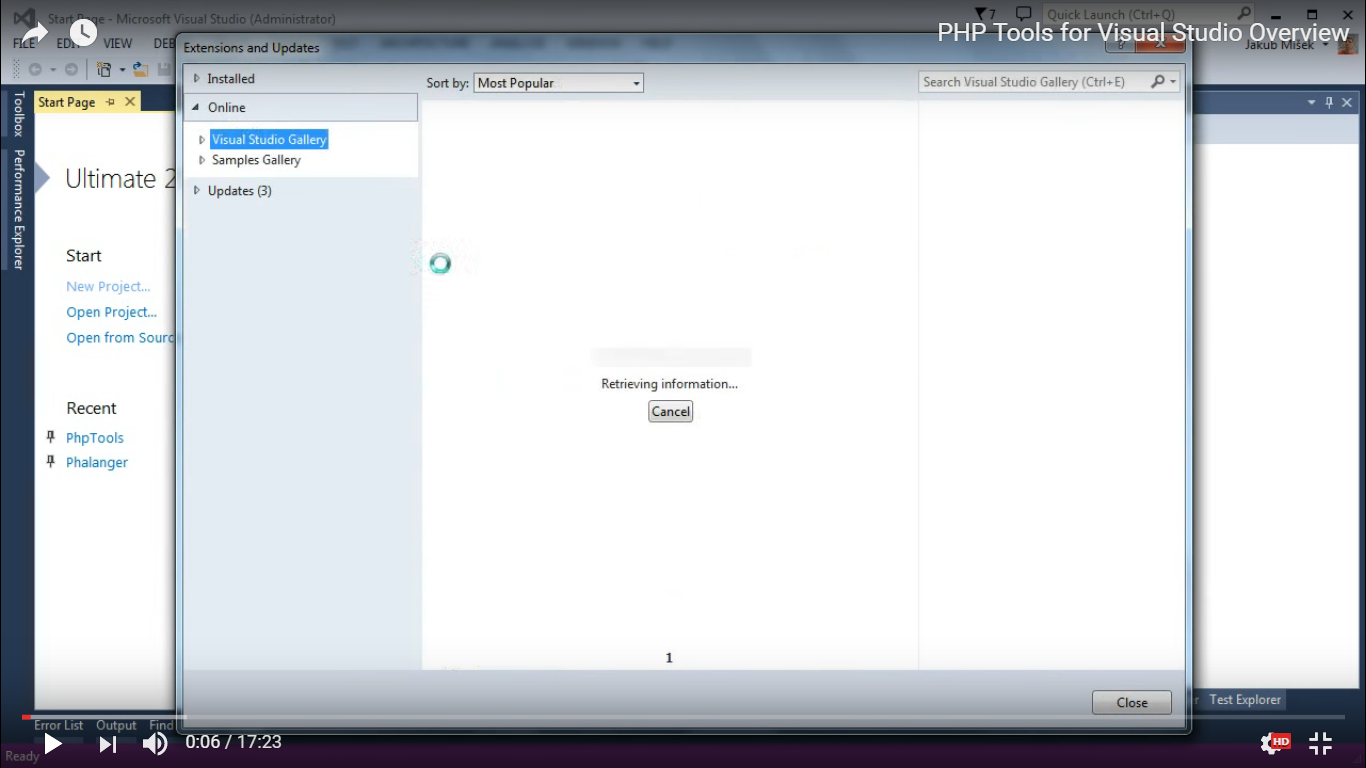
Rules for naming a variable is −

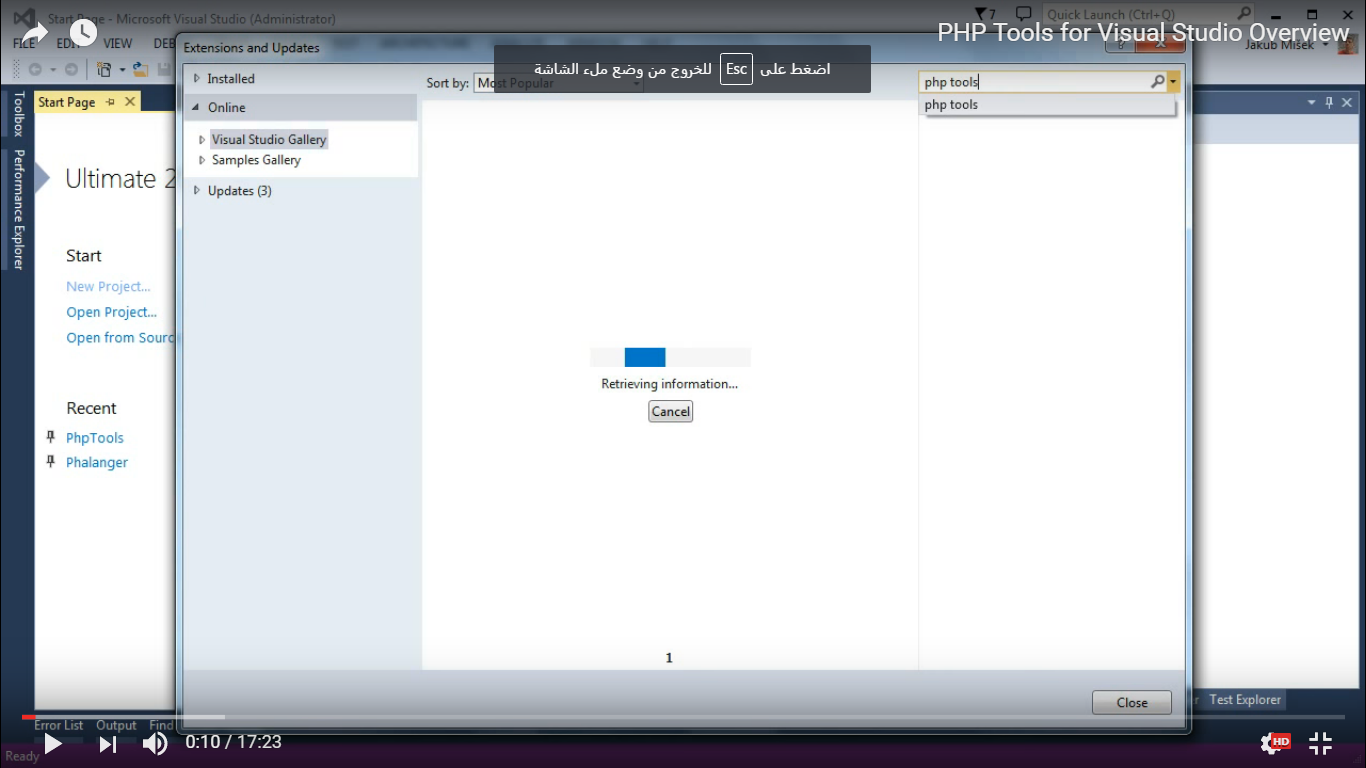
* Variable names must begin with a letter or underscore character.
* A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc

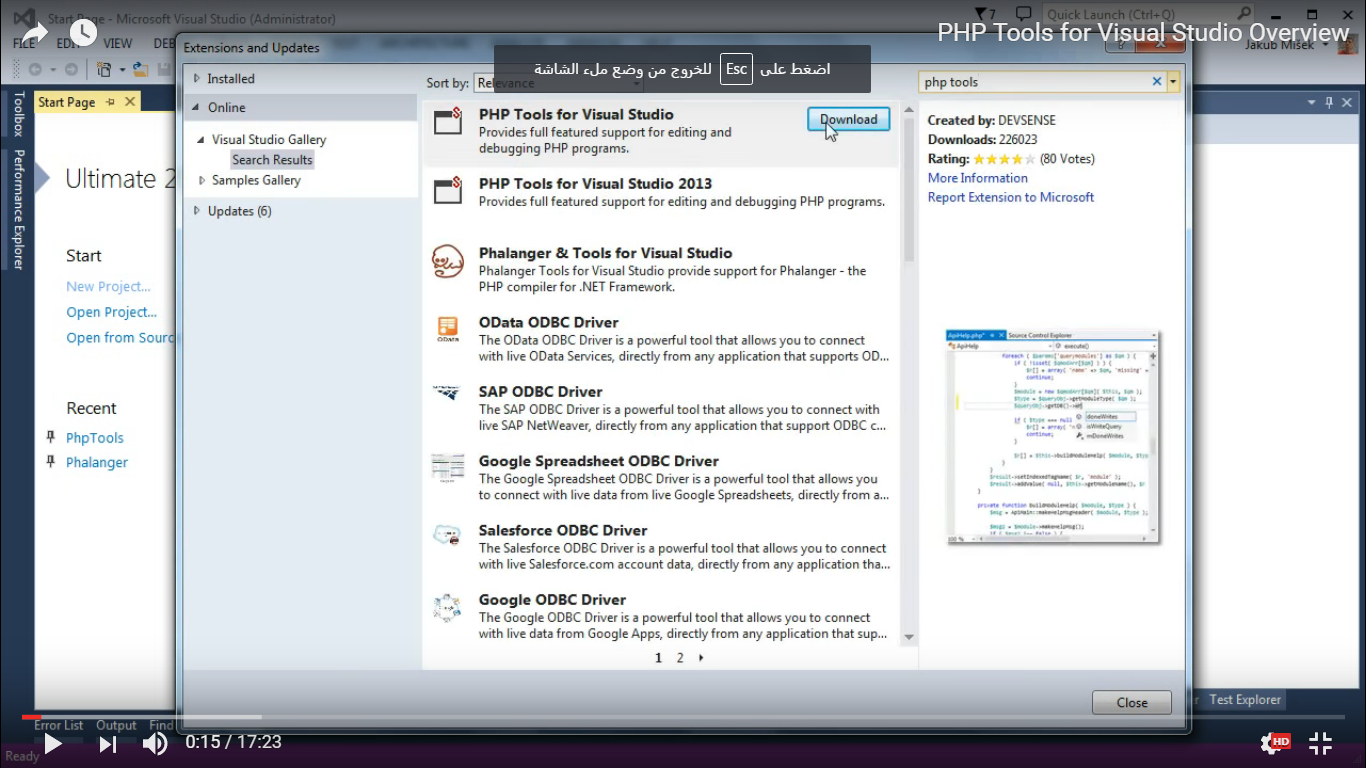
There is no size limit for variables.

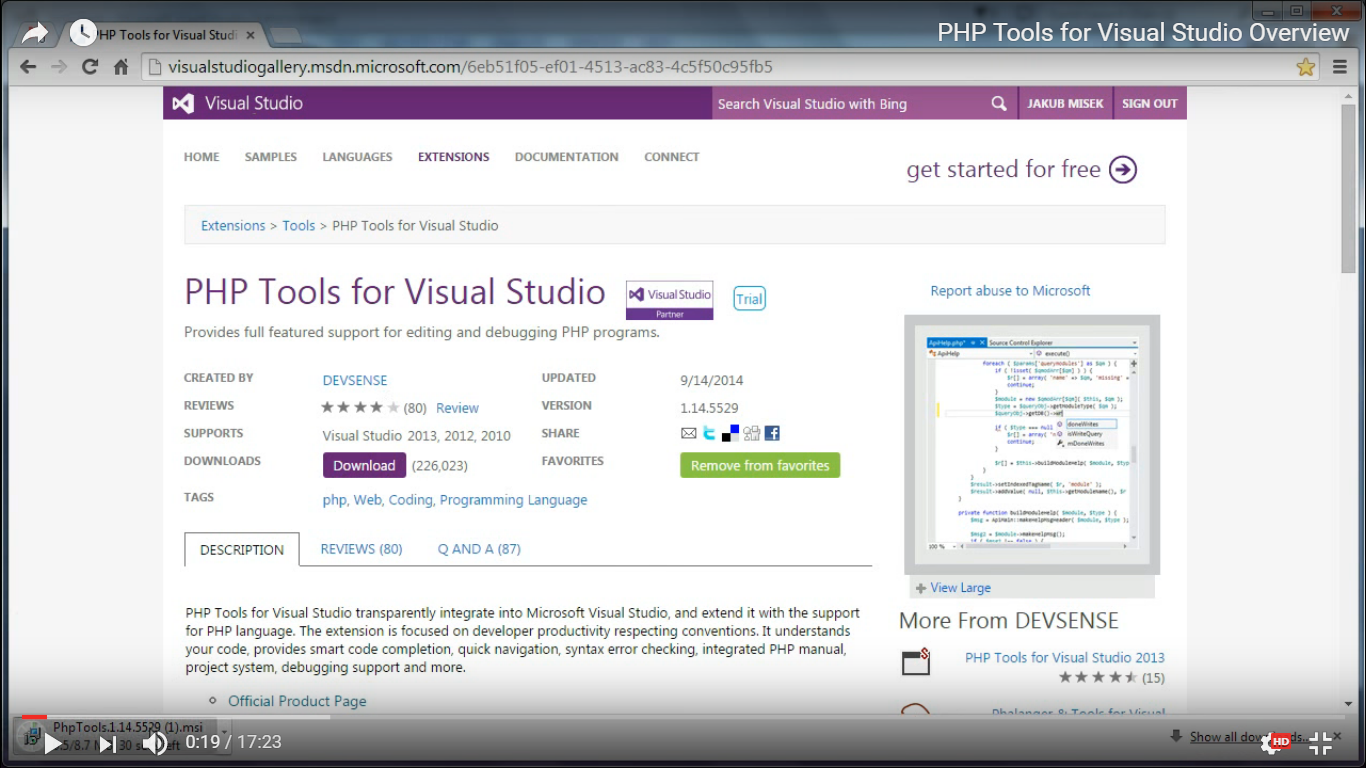
Installing php tool in visual studio :

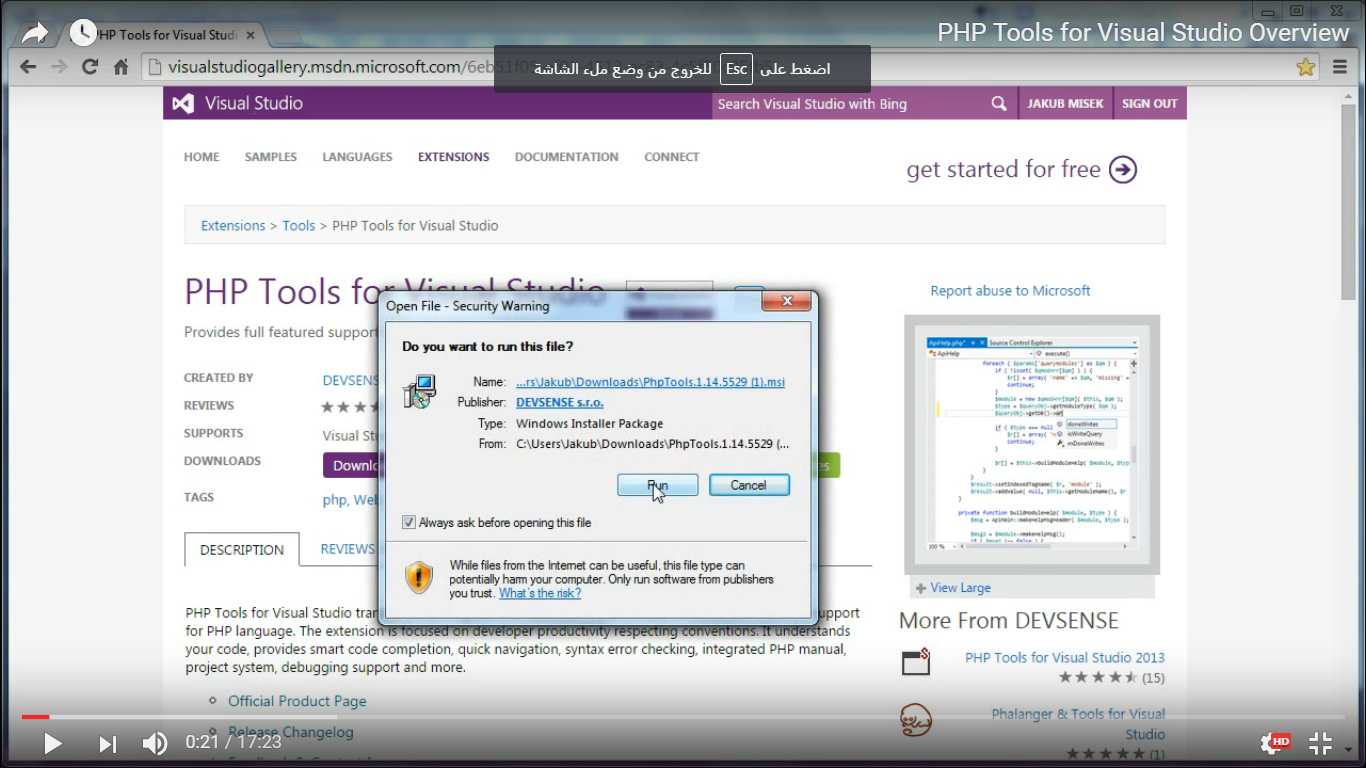


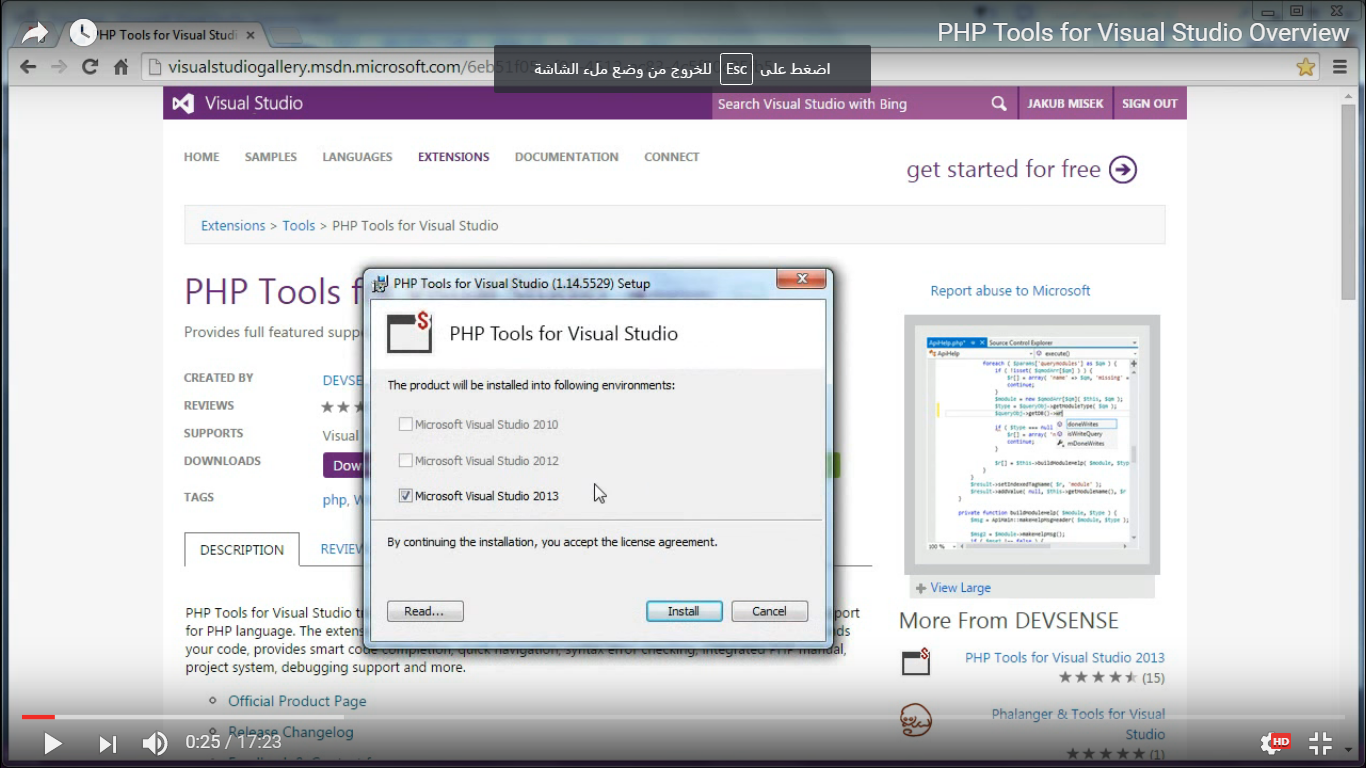












Close visual studio and check the current version of php

