PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP Data Types

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP Data Types

PHP Integer

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

Rules for integers:

- 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: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Data Types

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Data Types

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Data Types

PHP Array

An array stores multiple values in one single variable.
In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Data Types

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

PHP Data Types

PHP Object

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

PHP Data Types

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.
We will not talk about the resource type here, since it is an advanced topic.

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

strlen() - Return the Length of a String
The PHP strlen() function returns the length of a string.

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

PHP Strings

str_word_count() - Count Words in a String

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

PHP Strings

strrev() - Reverse a String
The PHP strrev() function reverses a string.
Example
Reverse the string "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

PHP Strings

strpos() - Search For a Text Within a String

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

PHP Strings

str_replace() - Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Strings

Complete PHP String Reference

For a complete reference of all string functions, go to our complete PHP String Reference.

The PHP string reference contains description and example of use, for each function!

PHP Numbers

in this chapter we will look in depth into Integers, Floats, and Number Strings.
PHP Numbers
One thing to notice about PHP is that it provides automatic data type conversion.
So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
This automatic conversion can sometimes break your code.

PHP Numbers

PHP Integers

An integer is a number without any decimal part.

2, 256, -256, 10358, -179567 are all integers. While 7.56, 10.0, 150.67 are floats.

So, an integer data type is a non-decimal number between -2147483648 and 2147483647. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.

Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

PHP Numbers

Here are some rules for integers:
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 has the following functions to check if the type of a variable is integer:
is_int()
is_integer() - alias of is_int()
is_long() - alias of is_int()
Example
Check if the type of a variable is integer:

```
<?php
$x = 5985;
var_dump(is_int($x));
$x = 59.85;
var_dump(is_int($x));
?>
```

PHP Numbers

PHP Floats

A float is a number with a decimal point or a number in exponential form.

2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.

The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following functions to check if the type of a variable is float:

is_float()

is_double() - alias of is_float()

Example

Check if the type of a variable is float:

```
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

PHP Numbers

PHP Infinity

A numeric value that is larger than PHP_FLOAT_MAX is considered infinite.

PHP has the following functions to check if a numeric value is finite or infinite:

is_finite()

is_infinite()

However, the PHP var_dump() function returns the data type and value:

Example

Check if a numeric value is finite or infinite:

```
<?php
$x = 1.9e411;
var_dump($x);
?>
```

PHP Numbers

PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

is_nan()

However, the PHP var_dump() function returns the data type and value:

Example

Invalid calculation will return a NaN value:

```
<?php
$x = acos(8);
var_dump($x);
?>
```

PHP Numbers

PHP Numerical Strings

The PHP is_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

Check if the variable is numeric:

```
<?php
$x = 5985;
var_dump(is_numeric($x));
$x = "5985";
var_dump(is_numeric($x));
$x = "59.85" + 100;
var_dump(is_numeric($x));
$x = "Hello";
var_dump(is_numeric($x));
?>
```

PHP Numbers

PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), or intval() function are often used to convert a value to an integer.

Cast float and string to integer:

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;
echo "<br>";
// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

PHP Constants

Create a PHP Constant

To create a constant, use the define() function.

Syntax

define(*name*, *value*, *case-insensitive*)

Parameters:

name: Specifies the name of the constant

value: Specifies the value of the constant

case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

Example

Create a constant with a **case-sensitive** name:

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

PHP Constants

PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

Create an Array constant:

```
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
```

PHP Constants

Constants are Global

Constants are automatically global and can be used across the entire script.

This example uses a constant inside a function, even if it is defined outside the function:

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
function myTest() {
  echo GREETING;
}
myTest();
?>
```