Тема 3.1
«Переменные и типы данных PHP»

Вступление

PHP – нестроготипизированный язык. Это значит, что переменные в нём могут менять свой тип в процессе выполнения программы.

Также PHP не накладывает строгих ограничений на участие в выражениях переменных разных типов.

Ещё одним отличием PHP от многих других языков является то, что переменные не нужно объявлять отдельно. Как правило, они «объявляются» через инициализацию при первом использовании.

Рассмотрим подробнее…

Программы на PHP

Программы PHP могут выполняться двумя способами:

как сценарное приложение web-сервером;

как консольные программы.

Поскольку, нашей задачей является программирование веб-приложений, мы преимущественно будем рассматривать первый способ.

PHP, как правило, используется сугубо для программирования приложений, связанных с Internet. Однако, PHP можно еще использовать в качестве интерпретатора из командной строки для автоматизации решения некоторых задач под Windows и Linux.

Рассмотрим процесс выполнения php-сценария при обращении браузера к серверу:

- браузер запрашивает страницу с расширением .php
- веб-сервер запускает PHP и передаёт ему необходимые данные;
- PHP возвращает результат веб-серверу;
- веб-сервер возвращает результат браузеру.

Команды PHP

Команды PHP заключаются в специальные теги, которые бывают четырёх видов:

<?php ?>

<? ?>

<script language = "php"></script>

<% %> (отключён по умолчанию в настройках)

Мы будем использовать наиболее классический вариант:

```
<?
    ...
?>
```

Команды PHP

Внутри какого-либо блока кода можно «выйти из PHP», при условии, что дальше мы «войдем» в него снова и закончим код, т.е., возможна следующая конструкция:

```
<?
  if($i<3)
   {
    echo("<p>Hello, world!<p>");
?>
  <p>Hello!</p>
  // эта строка не интерпретируется как код PHP
  // и выводится только если  блок кода выполняется
<?
   } // if
?>
```

Команда (конструкция, если точнее) echo в PHP применяется для вывода фактически всего, что встречается на web-страницах (текст, разметку HTML, числа).

Комментарии в PHP

PHP предоставляет несколько способов вставки комментариев. Проще всего пользоваться двойным слешем в стиле языка C (//), после чего PHP игнорирует всё, что расположено до конца строки.

Также можно пользоваться многострочными комментариями в стиле C (/*…*/). Для однострочных комментариев можно также пользоваться символом решетки (#) (комментарий скриптовых языков UNIX).

```
<?
  echo("<p>Hello</p>"); // комментарий
  echo("<p>Hello</p>"); # комментарий
  /*
    и это тоже комментарий
  */
?>
```

Комментарии в PHP

Следует помнить о том, что «комментарии PHP» действуют только внутри «тегов PHP». Если PHP встретит эти символы комментариев вне «тегов PHP», то они, как и любой текст, будут помещены на html-страницу.

```
<?
  echo("<p>Hello</p>"); // нормальный комментарий
?>
// а вот этот комментарий отобразится браузером.
<!-- Комментарий  HTML.
Будет виден в исходном коде HTML, но не в браузере  -->
```

Заметим, что комментарии можно вставлять не только после конца оператора, а, например, и вот так:

```
<?
  $a = "Hello, world";
  echo strstr($a,"H");
  // комментарий
?>
```

Переменные в PHP

В PHP переменные начинаются со знака доллара ($). За этим знаком может следовать любое количество буквенно-цифровых символов и символов подчёркивания, но первый символ не может быть цифрой.

Следует также помнить, что имена переменных в PHP **чувствительны** к регистру, в отличие от ключевых слов.

При объявлении переменных в PHP **не требуется** явно указывать тип переменной, при этом одна и та же переменная может иметь на протяжении программы **разные типы**.

Переменная инициализируется в момент присваивания ей значения и существует до тех пор, пока выполняется программа. Т.е., в случае веб-страницы это означает, что переменная существует до тех пор, пока обрабатывается текущий запрос.

Константы в PHP

Константы объявляются в PHP при помощи функции define():

```
define(CONSTANT, value)
```

Первый параметр этой функции – имя константы, второй – её значение. При использовании константы на неё ссылаются по имени:

```
<?
  define (CONSTANT1,15);
  echo CONSTANT1;
?>
```

Константы в PHP

По традиции имена констант пишут буквами верхнего регистра.

Существует функция defined(), которая проверяет, определена ли константа:

```
<?
   define(CONSTANT,"Hello");
   if(defined("CONSTANT"))
   {
      echo("<p>CONSTANT is defined</p>");
   }
?>
```

Типы данных в PHP

PHP поддерживает восемь типов данных:

Четыре скалярных типа:
integer – целые числа;
float (double) – дроби;
boolean – логический тип;
string – строки.

Два смешанных типа:
array – массив;
object – объект;

Два специальных типа:
resource – ресурс;
NULL – «отсутствие значения».

Типы данных в PHP

Для удобства понимания в документации по PHP используются также следующие **псевдо-типы**:

mixed – «любой тип» или «возможны варианты»;
number – любое целое или дробное число;
callback – имя функции, которая должна быть вызвана.

Типы данных в PHP: integer

integer – целое число, не имеет дробной части и представляется последовательностью из одной или нескольких цифр.

Восьмеричные числа начинаются с цифры 0, после которой следует серия цифр от 0 до 7.

Шестнадцатеричные целые числа имеют префикс 0x или 0X и могут состоять из цифр от 0 до 9 и букв от a (A) до f (F).

```
$a=10; // десятеричная система счисления
$b=05; // восьмеричная система счисления
$c=0xFF; // шестнадцатеричная система счисления
```

Типы данных в PHP: integer

Задача

Объявить три целочисленные переменные в разных системах счисления, сложить их и результат присвоить четвёртой переменной.

Решение

```
$a=10; // десятеричная система счисления
$b=05; // восьмеричная система счисления
$c=0xFF; // шестнадцатеричная система счисления
$d=$a+$b+$c;
```

Типы данных в PHP: double (float)

double – числа с плавающей точкой (они же числа двойной точности или действительные числа) могут быть определены при помощи любого из следующих способов:

```
$a = 1.234; // обычная десятичная дробь
$b = 1.2e3; // экспоненциальная запись
$c = 7E-10; // экспоненциальная запись
```

Типы данных в PHP: boolean

boolean – логический тип данных, который принимает только два значения: истинное (true, TRUE) и ложное (false, FALSE).

Логические величины создаются двумя способами: при проверке условий и в виде значений переменных.

```
$d=TRUE;
```

Типы данных в PHP: string

string – строка: последовательность символов, которая рассматривается как единое целое, но при этом обеспечивает доступ к отдельным символам.

Обратите внимание: в PHP не поддерживается символьный тип данных (char).

Строки делятся на две категории в зависимости от типа ограничителя – они могут ограничиваться парой кавычек (" ") или апострофов (' ').

Между этими категориями существуют два принципиальных отличия.

Типы данных в PHP: string

Отличие первое

Имена переменных в строках, заключенных в кавычки (“ “), заменяются соответствующими значениями, а строки в апострофах (‘ ’) интерпретируются буквально, даже если в них присутствуют имена переменных,

Два следующих объявления дают одинаковый результат:

```
$food = "apple";
$food = 'apple';
```

Однако результаты следующих объявлений сильно различаются:

```
$sentence = "My favorite food is $food";
$sentence2 = 'My favorite food is $food';
```

Переменной $sentence присваивается строка

My favorite food is apple.

Обратите внимание: переменная $food автоматически интерпретируется. С другой стороны, переменной $sentence2 присваивается строка

My favorite food is $food.

В отличие от переменной $sentence, в $sentence2 осталась не интерпретированная переменная $food.

Типы данных в PHP: string

Прежде чем рассматривать второе фундаментальное отличие строк, заключенных в апострофы и в кавычки, необходимо познакомиться со служебными символами, используемыми в строках PHP.

В PHP, как и в большинстве современных языков программирования, строки могут содержать служебные символы (например, символы табуляции или новой строки).

\n – «новая строка»
\r – «возврат каретки»
\t – «табуляция»
\\ – «бэкслэш» (обратная косая черта)
\$ – «доллар»
\" – «кавычка»

Типы данных в PHP: string

Отличие второе

Второе отличие заключается в том, что в строках, заключённых в кавычки, распознаются все существующие служебные символы, а в строках, заключенных в апострофы, – только служебные символы "\\" и "\".

Следующий пример наглядно демонстрирует различия между присваиванием строк, заключенных в кавычки и апострофы:

```
$double_list = "item1\nitem2\nitem2";
$single_list = 'item1\nitem2\nitem2';
```

Если вывести обе строки в браузере, окажется, что строка в кавычках содержит внутренние символы новой строки, а в строке в апострофах последовательность \n выводится как обычные символы.

Хотя многие служебные символы в браузерах несущественны, при форматировании для других условий они играют очень важную роль.

Типы данных в PHP: string

К отдельным символам строки можно обращаться как к элементам массива с последовательной нумерацией:

```
$sequence_number = "04efgh";
$letter = $sequence_number[4];
```

Переменной $letter будет присвоено значение g. Нумерация элементов строк и массивов начинается с 0.

Соответственно, выражение $sequence_number[1] будет равно 4.

Внимание! Начиная с 4-й версии PHP синтаксис доступа к элементам строки изменён (старый по-прежнему работает, но его не рекомендуется использовать), т.о. вышеприведённое выражение лучше записать так:

```
$letter = $sequence_number{4};
```

Т.е. с использованием фигурных { } скобок вместо квадратных [ ].

Типы данных в PHP: array

array – массив. Массив в PHP – это «упорядоченное отображение, которое устанавливает соответствие между значением и ключом».

Этот тип оптимизирован в нескольких направлениях, поэтому вы можете использовать его как собственно массив, список (вектор), хэш-таблицу (являющуюся реализацией карты), словарь, коллекцию, стек, очередь или, возможно, как что-то еще.

Поскольку значением элемента массива может быть другой массив, появляется возможность эмулировать с помощью массивов деревья.

Более подробно мы рассмотрим массивы в соответствующем разделе курса, поскольку они требуют особого внимания.

Типы данных в PHP: object

object – объект (экземпляр класса). Используется при написании программ в методологоо объектно-ориентированного программирования (ООП).

Для инициализации объекта используется выражение new, создающее в переменной экземпляр класса:

```
class c
{
 function sum_($a,$b)
 {
  echo $a+$b;
 }
}

$ex = new c;
$ex->sum(5,7);
```

Подробнее мы поговорим об объектах в разделе курса, посвящённом ООП в PHP.

Типы данных в PHP: resource

resource – ресурс: специальная переменная, содержащая ссылку на внешний ресурс.

Ресурсы создаются и используются специальными функциями.

Полный перечень этих функций и соответствующих типов ресурсов следует искать в документации по PHP (список огромный).

В качестве примера ресурсов можно назвать: соединение с СУБД, дескриптор каталога, дескриптор файла и т.п.

Типы данных в PHP: NULL

NULL – специальное «нулевое значение».

Это значение говорит о том, что переменная не имеет значения.

NULL – это единственно возможное значение типа NULL.

Переменная считается «равной NULL» если

- ей была присвоена константа NULL;
- ей ещё не было присвоено какое-либо значение;
- она была удалена с помощью unset().

```
$var = NULL;
```

«Переменные переменных»

PHP поддерживает т.н. «двойное разыменование» или «переменные переменных».

Так если:

```
$a="Hello";
```

и

```
$b="a";
```

то

```
echo $$b
```

выведет

```
Hello
```

Двойной знак доллара ($$) означает, что нужно взять значение указанной за ним переменной и рассмотреть его как имя «настоящей переменной».

Иногда это явление можно использовать для сложных случаев обработки данных, но следует быть очень осторожным.

Преобразование и переключение типов

PHP поддерживает два действия над типами данных:

- преобразование (приведение);
- переключение.

В случае **преобразования (приведения)** тип переменной меняется, а её значение модифицируется согласно определённому набору правил, которые мы рассмотрим чуть позже. Результат операции сохраняется до следующего преобразования типа.

В случае переключения тип переменной меняется (а значение модифицируется согласно определённому набору правил) только на время участия переменной в том или ином выражении, а затем – восстанавливается (и тип, и значение).

Переключение типов

Рассмотрим переключение типов на примере:

```
$a=10;
$b=15.7;
$c="Minsk";
$d=$a+$b+$c;
```

PHP анализирует выражение и определяет, что оно – арифметическое. Затем PHP анализирует операнды и приходит к выводу, что результат будет дробным числом. После чего происходит переключение типов переменных $a и $c к дробному числу, а их значения на момент участия в выражении становятся 10.0 и 0.0 соответственно. Затем происходит сложение: 10.0+15.7+0.0 = 25.7 и результат записывается в переменную $d. Сами же переменные $a и $c «не пострадали», т.е. их тип и значение остались неизменными.

Преобразование типов

Как уже было сказано, в PHP можно «навсегда» поменять тип переменной. PHP поддерживает для этого два способа:

- через функции gettype($var) и settype($var,$type);
- в стиле языка C.

Использование функции gettype() позволяет определить тип переменной.

Например:

```
$a="Minsk";
$b=gettype($a); // значение $b == string
```

Преобразование типов

Функция gettype() может возвращать следующие значения:

boolean (bool с версии PHP 4.2.0);

integer (int с версии PHP 4.2.0);

double (для любой дробной переменной, включая неявно существующие real и float);

string;

array;

object;

resource;

NULL;

unknown type.

Преобразование типов

Для определения типа или, если точнее, проверки того, что переменная является переменной того или иного типа, существует набор функций, возвращающих TRUE и FALSE в случае, если переменная является переменной «ожидаемого типа» и если не является – соответственно.

is_bool()
is_numeric()
is_float()
is_int()
is_string()
is_object()
is_array()
is_null()

Преобразование типов

Функция settype() позволяет установить новый тип переменной:

```
$a=5;
settype($a,”double”); // тип $a стал double, значение == 5.0
```

В качестве второго аргумента функция settype() принимает следующие значения:

boolean (bool с версии PHP 4.2.0);

integer (int с версии PHP 4.2.0);

float (double для версий PHP до 4.2.0);

string;

array;

object;

NULL.

Преобразование типов

Преобразование типов в стиле языка C реализуется с помощью синтаксиса круглых скобок:

(int), (integer) - приведение к целому числу;

(bool), (boolean) - приведение к логическому типу;

(float), (double), (real) - приведение к числу с плавающей точкой;

(string) - приведение к строке;

(array) - приведение к массиву;

(object) - приведение к объекту.

Пример:

```
$a=10;
$a=(string)$a; // $a стала строкой со значением «10»
```

Преобразование типов

Сейчас мы рассмотрим правила, согласно которым PHP преобразует значения переменных при преобразовании и переключении типов.

Здесь следует уяснить два важных момента:

1) PHP при любом преобразовании или переключении типа старается сохранить значение или хотя бы часть значения переменной настолько, насколько это возможно.

2) Несмотря на все усилия PHP иногда могут возникнуть неочевидные ситуации, а потому (если сомневаетесь в результате) лучше явно привести тип переменной к тому, который вам нужен.

Преобразование к boolean

При преобразовании в логический тип, следующие значения рассматриваются как FALSE:

сам boolean FALSE;

целое 0 (ноль);

число с плавающей точкой 0.0 (ноль);

пустая строка «» и строка «0»;

пустой массив;

объект без атрибутов (только в PHP 4);

специальный тип NULL (включая неустановленные переменные);

Все остальные значения рассматриваются как TRUE (включая любой ресурс).

Преобразование к integer

Из логического типа

FALSE преобразуется в 0 (ноль)

TRUE преобразуется в 1 (единицу).

Из чисел с плавающей запятой

Число будет округлено в меньшую сторону.

Если число превышает пределы целого (как правило, это +/- 2.15e+9 = 2^31), результат будет неопределенным.

Внимание! Никогда не приводите неизвестную дробь к целому, так как это может иногда дать неожиданные результаты:

```
echo (int) ( (0.1+0.7) * 10 ); // выводит 7!
```

Из строк

Рассмотрим чуть позже в разделе «преобразование строк в числа»

Из других типов

Для других типов поведение преобразования в целое не определено. В настоящее время поведение такое же, как если бы значение сперва было преобразовано в булев тип. Однако не полагайтесь на это поведение, так как оно может измениться в следующих версиях PHP

Преобразование к double (float)

Когда и как строки преобразуются в числа с плавающей запятой мы рассмотрим в разделе «Преобразование строк в числа».

Для значений других типов преобразование будет таким же, как если бы значение сначала было преобразовано в целое, а затем в число с плавающей точкой.

Преобразование к string

Помимо преобразования в строку с помощью функции settype() и синтаксиса круглых скобок можно использовать функцию strval(), которая возвращает строковое значение переменной.

В выражениях, где необходима строка, преобразование происходит автоматически. Это происходит при использовании функции echo() или print(), либо при сравнении значения переменной со строкой.

Логический тип

TRUE преобразуется в строку «1».

FALSE представляется как «» (пустая строка).

Целое (integer) или дробь (float)

Преобразуются в строку, представленную числом, состоящим из его цифр (включая показатель степени для чисел с плавающей запятой).

Преобразование к string

Массивы

Всегда преобразуются в строку "Array", поэтому нельзя отобразить содержимое массива, используя echo() или print().

Чтобы просмотреть один элемент массива, нужно использовать echo $array_name[“element_index”].

Объекты

Всегда преобразуются в строку "Object". Если необходимо вывести значение переменной-члена объекта с целью отладки, нужно использовать echo $class->var_name.

Ресурсы

Всегда преобразуются в строки со структурой "Resource id #X", где X – уникальный номер ресурса, присвоенный ему PHP во время выполнения.

NULL

Всегда преобразуется в пустую строку «».

Преобразование к string

Как следует из вышеизложенного, вывод массивов, объектов или ресурсов в виде строк не предоставляет никакой полезной информации о значениях таких переменных.

Более подходящий способ вывода значений для отладки – использовать функции print_r() и var_dump().

Также можно преобразовать значения любых типов PHP в строки для постоянного хранения. Этот метод называется **сериализацией** и может быть выполнен при помощи функции serialize().

Кроме того, если в PHP включена поддержка WDDX, можно сериализовать значения PHP в структуры XML.

Преобразование строк в числа

Если строка распознается как числовое значение, результирующее значение и тип определяется, как показано далее.

Строка будет распознана как float, если она содержит любой из символов «.», «e», или «E». Иначе она будет определена как целое.

Значение определяется по начальной части строки. Если строка начинается с верного числового значения, будет использовано это значение. Иначе значением будет 0 (ноль).

Верное числовое значение – это одна или более цифр (в т.ч. содержащие десятичную точку), предварённых или непредварённых знаком, с последующим необязательным показателем степени.

Показатель степени – это «e» или «E» с последующими одной или более цифрами.

Преобразование строк в числа

Рассмотрим на примерах:

$a = 1 + "10.5"; // **float (11.5)**

$a = 1 + "-1.3e3"; // **float (-1299)**

$a = 1 + "bob-1.3e3"; **integer (1)**

$a = 1 + "bob3"; // **integer (1)**

$a = 1 + "10 Something"; // **integer (11)**

$a = 4 + "10.2 Something"; // **float (14.2)**

$a = "10.0 cats " + 1; // **float (11)**

$a = "10.0 cats " + 1.0; // **float (11)**

Преобразование строк в числа

Задача 1:

$a=“10 cats”;

$b=“5$a dogs”;

$c=$a/$b;

Чему равно $c ?

$c == 10/510 == 1/51;

Задача 2:

$a=“10 cats”;

$b=‘5$a dogs’;

$c=$a/$b;

Чему равно $c ?

$c == 10/5 == 2;

Преобразование к array

integer, float, string, boolean и resource

Для любого из типов: integer, float, string, boolean и resource преобразование в массив даёт массив с одним элементом (с индексом 0), являющимся скалярным значением исходной переменной.

object

Преобразование в массив объекта даёт в качестве элементов массива свойства (переменные-члены) этого объекта. Ключами будут имена переменных-членов.

NULL

Преобразование в массив значения NULL даёт пустой массив.

Преобразование к object

Если объект преобразуется в объект, он не изменяется.

Если в объект преобразуется значение любого иного типа, создаётся новый экземпляр встроенного класса stdClass. Если значение было пустым, новый экземпляр также будет пустым. При любом другом значении оно будет содержатся в переменной-члене scalar.

```
$obj = (object) “Minsk”;
echo $obj->scalar;  // выведет Minsk
```

Преобразование к NULL

Не существует.

Фактически, любую переменную можно «преобразовать к NULL», удалив её с помощью функции unset().

Таблицы сравнения типов

Сейчас мы рассмотрим таблицы, иллюстрирующие то, как PHP сравнивает данные различных типов. Но для начала нужно подчеркнуть два момента:

1) HTML-формы НИКОГДА не передают тип переменной, они ВСЕГДА передают СТРОКИ (или массивы строк). Для проверки того, является ли строка числом, используйте функцию is_numeric(), а ещё лучше – регулярное выражение.

Трюк: если вы знаете, что в переменной должно прийти, например, целое ненулевое значение, можно использовать код:

```
$var=(int)$var++;
$var--;
if ($var===0) {echo “Кыш, мерзкий хакер!”; die(); }
```

2) Использование if ($x …) пока переменная $x не определена сгенерирует ошибку E_NOTICE. Вместо этого используйте функцию empty() или isset() и/или инициализируйте переменную.

Сравнение типов и функции…

Сравнение типов *$x* и результатов функций PHP, связанных с типами.

Выражение	gettype()	empty()	is_null()	isset()	Логическое: if($x)
$x = "";	string	TRUE	FALSE	TRUE	FALSE
$x = NULL	NULL	TRUE	TRUE	FALSE	FALSE
var $x;	NULL	TRUE	TRUE	FALSE	FALSE
$x не определена	NULL	TRUE	TRUE	FALSE	FALSE
$x = array();	array	TRUE	FALSE	TRUE	FALSE
$x = false;	bool	TRUE	FALSE	TRUE	FALSE
$x = true;	bool	FALSE	FALSE	TRUE	TRUE
$x = 1;	integer	FALSE	FALSE	TRUE	TRUE
$x = 42;	integer	FALSE	FALSE	TRUE	TRUE
$x = 0;	integer	TRUE	FALSE	TRUE	FALSE
$x =- 1;	integer	FALSE	FALSE	TRUE	TRUE
$x = "1";	string	FALSE	FALSE	TRUE	TRUE
$x = "0";	string	TRUE	FALSE	TRUE	FALSE
$x = "- 1";	string	FALSE	FALSE	TRUE	TRUE
$x = "php";	string	FALSE	FALSE	TRUE	TRUE
$x = "true";	string	FALSE	FALSE	TRUE	TRUE
$x = "false";	string	FALSE	FALSE	TRUE	TRUE

Сравнение с помощью ==

Гибкое сравнение с помощью ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Сравнение с помощью ===

Жёсткое сравнение с помощью ===

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE