### Лекция 6

### Тема: Типы данных

*PHP* поддерживает восемь простых *типов данных*.

Четыре скалярных *типа*:

* *boolean (логический)* ;
* *integer (целый)* ;
* *float (с плавающей точкой)* ;
* *string (строковый)*.

Два смешанных *типа*:

* *array (массив)* ;
* *object (объект)*.

И два специальных *типа*:

* *resource (ресурс)* ;
* *NULL*.

В *PHP* не принято явное объявление *типов**переменных*. Предпочтительнее, чтобы это делал сам *интерпретатор* во *время выполнения* программы в зависимости от контекста, в котором используется *переменная*. Рассмотрим по порядку все перечисленные *типы данных*.

#### Тип boolean (булев или логический тип)

Этот простейший *тип* выражает истинность значения, то есть *переменная* этого *типа* может иметь только два значения – истина TRUE или ложь FALSE .

Чтобы определить булев *тип*, используют ключевое слово TRUE или FALSE. Оба регистронезависимы.

<?php

$test = True;

?>

Пример 2.5. Логический тип

Логические *переменные* используются в различных *управляющих конструкциях* (циклах, условиях и т.п., более подробно речь о них пойдет в одной из следующих лекций). Иметь *логический тип*, т.е. принимать только два значения, истину или ложь, могут также и некоторые *операторы* (например, *оператор* равенства). Они также используются в *управляющих конструкциях* для проверки каких-либо условий. Например, в условной конструкции проверяется истинность значения *оператора* или *переменной* и в зависимости от результата проверки выполняются те или иные действия. Здесь условие может быть истинно или ложно, что как раз и отражает *переменная* и *оператор* *логического типа*.

<?php

// Оператор '==' проверяет равенство

// и возвращает

// булево значение

if ($know == False) { // если $know

// имеет значение

// false

echo "Изучай PHP!";

}

if (!$know) { // то же самое, что

// и выше, т.е. проверка

// имеет ли $know значение

// false

echo "Изучай PHP!";

}

/\* оператор == проверяет, совпадает ли

значение переменной $know со строкой

"Изучай PHP". Если совпадает, то

возвращает true, иначе – false.

Если возвращено true, то выполняется

то, что внутри фигурных скобок \*/

if ($know == "Изучай PHP")

{ echo "Начал изучать"; }

?>

Пример 2.6. Использование логического типа

#### Тип integer (целые)

Этот *тип* задает число из множества целых чисел Z = {..., -2, -1, 0, 1, 2, ...}. Целые могут быть указаны в десятичной, шестнадцатеричной или *восьмеричной системе счисления*, по желанию с предшествующим знаком " - " или " + ".

Если вы используете восьмеричную систему счисления, вы должны предварить число 0 (нулем), для использования шестнадцатеричной системы нужно поставить перед числом 0x.

<?php

# десятичное число

$a = 1234;

# отрицательное число

$a = -123;

# восьмеричное число (эквивалентно

# 83 в десятичной системе)

$a = 0123;

# шестнадцатеричное число (эквивалентно

# 26 в десятичной системе)

$a = 0x1A;

?>

Размер *целого* зависит от платформы, хотя, как правило, максимальное значение около двух миллиардов (это 32-битное знаковое). Беззнаковые *целые* PHP не поддерживает.

Если вы определите число, превышающее пределы *целого типа*, оно будет интерпретировано как *число с плавающей точкой*. Также если вы используете *оператор*, результатом работы которого будет число, превышающее пределы *целого*, вместо него будет возвращено *число с плавающей точкой*.

В PHP не существует *оператора* деления *целых*. Результатом 1/2 будет *число с плавающей точкой* 0.5. Вы можете привести значение к *целому*, что всегда округляет его в меньшую сторону, либо использовать функцию round(), округляющую значение по стандартным правилам. Для преобразования *переменной* к конкретному *типу* нужно перед *переменной* указать в скобках нужный *тип*. Например, для преобразования *переменной* $a=0.5 к *целому типу* необходимо написать (integer)(0.5) или (integer) $a или использовать сокращенную запись (int)(0.5). Возможность явного приведения *типов* по такому принципу существует для всех *типов данных* (конечно, не всегда значение одного *типа* можно перевести в другой *тип* ). Мы не будем углубляться во все тонкости приведения *типов*, поскольку PHP делает это автоматически в зависимости от контекста.

#### Тип float (числа с плавающей точкой)

*Числа с плавающей точкой* (они же числа двойной точности или *действительные числа*) могут быть определены при помощи любого из следующих синтаксисов:

<?php

$a = 1.234;

$b = 1.2e3;

$c = 7E-10;

?>

Размер *числа с плавающей точкой* зависит от платформы, хотя максимум, как правило, ~1.8e308 с точностью около 14 десятичных цифр.

#### Тип string (строки)

***Строка*** – это набор символов. В PHP символ – это то же самое, что байт, это значит, что существует ровно 256 различных символов. Это также означает, что PHP не имеет встроенной поддержки Unicode. В PHP практически не существует ограничений на размер *строк*, поэтому нет абсолютно никаких причин беспокоиться об их длине.

*Строка* в PHP может быть определена тремя различными *способами*:

* с помощью *одинарных кавычек* ;
* с помощью *двойных кавычек* ;
* *heredoc-синтаксисом*.

##### Одинарные кавычки

Простейший *способ* определить *строку* – это заключить ее в *одинарные кавычки* " ' ". Чтобы использовать *одинарную кавычку* внутри *строки*, как и во многих других языках, перед ней необходимо поставить символ обратной косой черты " \ ", т. е. экранировать ее. Если обратная косая черта должна идти перед *одинарной кавычкой* либо быть в конце *строки*, необходимо продублировать ее " \\' ".

Если внутри *строки*, заключенной в *одинарные кавычки*, обратный слэш " \ " встречается перед любым другим символом (отличным от " \ " и " ' " ), то он рассматривается как обычный символ и выводится, как и все остальные. Поэтому обратную косую черту необходимо экранировать, только если она находится в конце *строки*, перед закрывающей кавычкой.

В PHP существует ряд комбинаций символов, начинающихся с символа обратной косой черты. Их называют ***управляющими последовательностями***, и они имеют специальные значения, о которых мы расскажем немного позднее. Так вот, в отличие от двух других синтаксисов, *переменные* и *управляющие последовательности* для специальных символов, встречающиеся в *строках*, заключенных в *одинарные кавычки*, не *обрабатываются* .

<?php

echo 'Также вы можете вставлять в строки

символ новой строки таким образом,

поскольку это нормально';

// Выведет: Чтобы вывести ' надо

// перед ней поставить \

echo 'Чтобы вывести \' надо перед ' .

'ней поставить \\';

// Выведет: Вы хотите удалить C:\\*.\*?

echo 'Вы хотите удалить C:\\\*.\*?';

// Выведет: Это не вставит: \n новую строку

echo 'Это не вставит: \n новую строку';

// Выведет: Переменные $expand также

// $either не подставляются

echo 'Переменные $expand также $either' .

'не подставляются';

?>

Пример 2.7. Использование управляющих последовательностей

##### Двойные кавычки

Если *строка* заключена в *двойные кавычки* " " ", PHP распознает большее количество *управляющих последовательностей* для специальных символов. Некоторые из них приведены в [таблице 2.7](https://intuit.ru/studies/courses/42/42/lecture/27177?page=5#table.2.7).

|  |  |
| --- | --- |
| Таблица 2.7. *Управляющие последовательности* | |
| **Последовательность** | **Значение** |
| \n | Новая *строка* ( LF или 0x0A (10) в ASCII) |
| \r | Возврат каретки ( CR или 0x0D (13) в ASCII) |
| \t | Горизонтальная табуляция ( HT или 0x09 (9) в ASCII) |
| \\ | Обратная косая черта |
| \$ | Знак доллара |
| \" | Двойная кавычка |

Повторяем, если вы захотите экранировать любой другой символ, обратная косая черта также будет напечатана!

Самым важным свойством *строк* в *двойных кавычках* является *обработка переменных*.

##### Heredoc

Другой *способ определения строк* – это использование ***heredoc-синтаксиса***. В этом случае *строка* должна начинаться с символа <<<, после которого идет идентификатор. Заканчивается *строка* этим же идентификатором. Закрывающий идентификатор должен начинаться в первом столбце *строки*. Кроме того, идентификатор должен соответствовать тем же правилам именования, что и все остальные метки в PHP: содержать только буквенно-цифровые символы и знак подчеркивания и начинаться не с цифры или знака подчеркивания.

*Heredoc* -текст ведет себя так же, как и *строка* в *двойных кавычках*, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в *heredoc*, но вы по-прежнему можете использовать перечисленные выше *управляющие последовательности*. *Переменные* внутри *heredoc* тоже *обрабатываются*.

<?php

$str = <<<EOD

Пример строки, охватывающей несколько

строчек, с использованием

heredoc-синтаксиса

EOD;

// Здесь идентификатор – EOD. Ниже

// идентификатор EOD

$name = 'Вася';

echo <<<EOD

Меня зовут "$name".

EOD;

// это выведет: Меня зовут "Вася".

?>

Пример 2.8. Использование heredoc-синтаксиса

Замечание: Поддержка *heredoc* была добавлена в PHP 4.