ЛЕКЦИЯ 4. ТИПЫ ДАННЫХ

Общие сведения о типах

В языке PHP предусмотрено всего восемь типов:

1. **целые числа,**
2. **числа с плавающей точкой двойной точности,**
3. **логические значения,**
4. **строковые значения,**
5. **массивы,**
6. **объекты,**
7. **значения NULL**
8. **ресурсы.**

Целочисленные значения представляют собой целые числа без десятичной точки, такие как 495.

Числа с плавающей точкой двойной точности — это числа с плавающей точкой наподобие 3.14159 или 49.0, имеющие наибольший возможный формат представления.

Логические значения принимают только два возможных значения: TRUE и FALSE.

Значение NULL — это специальный тип, который имеет только одно значение: NULL.

Строки — это последовательности символов, например 'PHP поддерживает работу со строками'.

Массивы представляют собой именованные и индексированные коллекции других значений.

Объекты — это экземпляры определяемых программистом классов, которые могут включать не только значения других типов, но и функции, относящиеся к соответствующему классу.

Ресурсами называют специальные переменные, которые хранят ссылки на ресурсы, внешние по отношению к интерпретатору PHP (такие как соединения с базой данных).

Первые пять из этих типов являются **простыми типами**, а следующие два типа (массивы и объекты) **— составными**. Различия между простыми и составными типами заключаются в том, что составные типы могут применяться для представления произвольных значений произвольных типов, а простые типы этого не позволяют. В этой статье кратко рассматриваются только простые типы.

Отсутствие объявлений типов переменных

Тип переменной **не требуется объявлять заранее**. Вместо этого программист может перейти непосредственно к выполнению операции присваивания и дать возможность интерпретатору PHP заняться определением типа присваиваемого значения выражения, как показано в следующем примере:

```
$pi = 3.14159;
```

```
$str = 'PHP поддерживает работу со строками';
```

Автоматическое преобразование типов

В языке PHP предусмотрены удобные способы автоматического преобразования типов в случае необходимости. Как и в большинстве других современных языков программирования, в языке PHP выполняются правильные действия, например, при вычислении математических выражений со смешанными числовыми типами. В частности, результатом выполнения выражения:

```
$pi = 3 + 0.14159;
```

является число с плавающей точкой (двойной точности), причем целое число 3 неявно преобразуется в число с плавающей точкой перед выполнением операции сложения.

Простые типы

Целые числа

Целочисленные значения относятся к самому простому типу — соответствуют простым целым числам, как положительным, так и отрицательным. Целочисленные значения могут присваиваться переменным или использоваться в выражениях, как показано в следующем примере:

```
$int_var = 12345;
$another_int = -12345 + 12345;      // Результат будет равен нулю
```

Числа с плавающей точкой двойной точности

Примеры чисел с плавающей точкой двойной точности приведены ниже:

```
$first_double = 123.456;
$second_double = 0.456;
$even_double = 2.0;
```

Следует отметить, что присваивание переменной $even_double числового значения без значащих цифр после точки не влечет за собой то, что вместо числа с

плавающей точкой присваивается целое число. Целые числа и числа с плавающей точкой двойной точности записываются в память с помощью разных форматов представления, поэтому результатом выполнения следующего примера становится присваивание числа с плавающей точкой, а не целого числа, даже несмотря на то, что это значение выводится как 5:

```
$five = $even_double +3;
```

Но почти в любых ситуациях пользователь может применять в математических выражениях произвольные сочетания чисел с плавающей точкой двойной точности и целых чисел и предоставлять возможность решать проблемы, связанные с определением типов для соответствующих значений, интерпретатору PHP.

Логические значения

Логическими называют истинностные значения, которые используются в таких конструкциях, позволяющих управлять ходом выполнения программы, как часть оператора if, в которой происходит проверка условия. Логические значения могут комбинироваться с применением логических операций для получения более сложных логических выражений.

Логические константы

В языке PHP предусмотрена пара констант, специально предназначенных для использования в качестве логических констант. Это — константы **TRUE** и **FALSE**, которые могут применяться примерно следующим образом:

```
if (TRUE)
      print("Этот текст всегда будет печататься<br>");
else
      print("Этот текст никогда не будет напечатан<br>");
```

Интерпретация значений других типов как логических значений

Ниже приведены правила определения истинности любого значения, которое еще не является логическим:

- Если значение является числовым и точно равно нулю, то рассматривается как ложное, все прочие числовые значения считаются истинными.
- Если значение является строковым, то рассматривается как ложное, если строка пуста (имеет длину, равную нулю) или представляет собой строку "0", и как истинное в противном случае.
- Значения NULL всегда являются ложными.
- Если значение имеет составной тип (представляет собой массив или объект), то рассматривается как ложное, если не содержит других значений, и как истинное в противном случае. Объект считается содержащим значение, если в его состав входит переменная экземпляра, которой было присвоено значение.

Ниже приведены переменные, в именах которых указано истинностное значение (true — истина, false — ложь), принимаемое этими переменными при использовании в логическом контексте:

```
$true_num = 3 + 0.14159;
$true_str = "Строка true.";
$true_array[49] = "Элемент массива";

$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = ""; // Строка нулевой длины
```

Безусловно, из первого приведенного выше правила следует, что число с плавающей точкой двойной точности (0.0) преобразуется в ложное логическое значение. Но необходимо учитывать, что выражения с плавающей точкой опасно использовать в качестве логических выражений из-за возможных ошибок округления.

Значение null

На первый взгляд мир логических значений может показаться малым, поскольку для переменных логического типа предусмотрены только два возможных значения. А тип NULL сужает пространство значений до крайнего предела: данные типа NULL могут иметь только одно возможное значение,

которым является значение NULL. Чтобы присвоить переменной значение NULL, достаточно применить к переменной примерно такую операцию присваивания:

```
$my_var = NULL;
```

Специальная **константа NULL** записывается прописными буквами в соответствии с общепринятым соглашением, но фактически имя этой константы не чувствительно к регистру, приведенный выше оператор вполне можно было бы записать следующим образом:

```
$my_var = null;
```

Так чем же отличаются значения NULL? Значение NULL представляет отсутствие значения (значение NULL можно рассматривать как недостающее или незаданное значение). Переменная, которой присвоено значение NULL, почти неотличима от переменной, которой вообще не было присвоено значение. В частности, переменная, которой присвоено значение NULL, обладает описанными ниже свойствами:

- В логическом контексте вычисления, проведенные с этой переменной, приводят к получению значения FALSE.
- Применение к этой переменной проверки с помощью функции IsSet() приводит к получению значения FALSE. (Таким свойством не обладают переменные каких-либо других типов.)

Интерпретатор PHP не выводит предупреждающее сообщение при передаче такой переменной в функцию и возврате из функции, тогда как передача переменной, значение которой еще не было присвоено, иногда приводит к активизации предупреждающих сообщений.

Переменные со значением NULL в основном предназначены для использования в таких ситуациях, когда требуется преднамеренно применить переменную, не имеющую значения, и разработчик хочет, чтобы об этом узнал и читатель созданного им кода, и интерпретатор PHP. Последнее соображение становится особенно важным, если приходится передавать значения переменных в функции.

Строки

Как показано ниже, строки представляют собой символьные последовательности:

```
$string_1 = "Это строка в двойных кавычках.";
$string_2 = 'Это строка в одинарных кавычках';
$string_26 = "В этой строке 26 символов.";
$string_0 = "";     // Строка, количество символов в которой равно нулю
```

Строки могут быть заключены в одинарные или двойные кавычки. От применяемых ограничительных символов зависит то, как будут обрабатываться строки во время чтения интерпретатором. Строки в одинарных кавычках остаются почти неизменными. С другой стороны, строки в двойных кавычках подвергаются более интенсивной обработке: переменные, заданные в таких строках, заменяются своими значениями, а некоторые символьные последовательности интерпретируются особым образом.

Строки в одинарных кавычках

Если не учитывать, что происходит интерпретация особым образом нескольких символьных последовательностей, чтение и сохранение в памяти строк в одинарных кавычках осуществляется без каких-либо изменений. Например, обработка интерпретатором следующего кода:

```
$var = 12345;
$literally = 'Переменная var = $var.\\nOk';
print($literally);
```

приводит к получению в окне браузера следующего вывода:

Обработка строки с одинарными кавычками

Строки в одинарных кавычках подчиняются также такому общему правилу, что кавычки другого типа не приводят к разрыву строки, заключенной в кавычки. Например, следующий оператор является допустимым:

```
    $singly_quoted = 'В этой переменной двойные кавычки: " не приводят к нарушению в работе';
```

Чтобы ввести одинарную кавычку (применяемую в качестве апострофа) в строку, заключенную в одинарные кавычки, необходимо обозначить одинарную кавычку с помощью знака переключения (на другой режим обработки символов) в виде обратной косой черты, как показано ниже:

```
$singly_quoted = 'Вводим \'одинарную кавычку\' в строку';
```

Хотя в большинстве контекстов символы обратной косой черты в строках с одинарными кавычками интерпретируются буквально, можно также использовать два символа обратной косой черты (в качестве управляющей последовательности для обозначения одного символа обратной косой черты, не применяемого в составе управляющей последовательности). Это удобно, если символ обратной косой черты должен быть задан в качестве последнего символа в строке, как в следующем примере:

```
// Получается строка 'C:\Windows\System32\'
$win_path = ' C:\\Windows\\System32\\';
```

Подводя итог, отметим, что строки в одинарных кавычках сохраняются в памяти в неизменном виде и единственными исключениями являются две указанные управляющие последовательности (\\ и \').

Строки в двойных кавычках

Строки, заключенные в двойные кавычки (например, "this"), подвергаются предварительной обработке интерпретатором PHP двумя приведенными ниже способами:

- Некоторые символьные последовательности, начинающиеся с обратной косой черты (\), заменяются специальными символами.
- Имена переменных (начинающиеся со знака $) заменяются строковыми представлениями их значений.

- Ниже описано, какие замены происходят при обработке некоторых управляющих последовательностей:
- Последовательность \n заменяется символом обозначения конца строки.
- Последовательность \r заменяется символом возврата каретки (переноса).
- Последовательность \t заменяется символом табуляции.
- Последовательность \$ заменяется самим знаком доллара ($).
- Последовательность \" заменяется одной двойной кавычкой (").
- Последовательность \\ заменяется одной наклонной чертой влево (\).

Первые три из этих замен позволяют упростить задачу по включению в строку некоторых пробельных символов, предоставляя возможность визуально контролировать расстановку таких символов. Последовательность \$ позволяет включить в строку символ $, если он должен появиться в непосредственном виде, а не интерпретироваться как начало имени переменной. Последовательность \" предназначена для того, чтобы можно было включить в строку в двойных кавычках символ двойной кавычки, не разрывая эту строку. Наконец, поскольку символ \ обозначает начало всех этих последовательностей, то необходим способ включения данного символа в непосредственном виде, а не в качестве начального символа управляющей последовательности; для этого перед символом обратной косой черты достаточно поставить еще один такой символ.

Как и в случае строк в одинарных кавычках, в строки в двойных кавычках можно свободно включать кавычки другого типа, без символа управляющей последовательности:

```
$has_apostrophe = "There's no problem here";
```

Подстановка значений переменных

После обнаружения в строке, заключенной в двойные кавычки, каждого символа $, не обозначенного как относящийся к управляющей последовательности, интерпретатор PHP пытается трактовать текст, который следует за этим символом, как имя переменной и подставить текущее значение соответствующей переменной в строку. При этом замена имени переменной значением этой переменной может осуществляться с помощью различных

описанных ниже способов, в зависимости от того, какое значение присвоено этой переменной:

- Если переменной в настоящее время присвоено строковое значение, то оно заменяет в строке с двойными кавычками имя соответствующей переменной (иными словами, выполняется подстановка значения переменной вместо имени переменной).
- Если переменной в настоящее время присвоено значение, отличное от строкового, то данное значение преобразуется в строковое, а затем осуществляется замена имени переменной строковым значением.
- Если переменной в настоящее время не присвоено значение, то интерпретатор PHP заменяет имя переменной пустым значением (или, если применить эквивалентную формулировку, интерпретатор PHP подставляет вместо имени переменной пустую строку).

Проверка типа

Во время выполнения программы иногда возникает необходимость узнать, к какому типу относится то или иное значение. Такая задача становится особенно важной в связи с тем, что тип переменной может быть изменен в результате присваивания этой переменной другого значения. В языке PHP, во-первых, предусмотрена общая функция проверки типа **gettype(),** а во-вторых, имеются отдельные булевы функции для каждого из пяти основных типов. Итоговые сведения об этих функциях приведены в таблице ниже. Следует учитывать, что некоторые из этих функций имеют альтернативные имена:

Функции проверки типа

Функция	Описание
gettype(arg)	Возвращает строку, представляющую тип arg: integer (целое число), float (число с плавающей точкой), string (строковое значение), array (массив), object (объект) или unknown (неизвестный тип)
is_int(arg), is_integer(arg), is_long(arg)	Возвращает значение true, если arg - целое число, в противном случае - false

is_double(arg), is_float(arg), is_real(arg)	Возвращает значение true, если arg - число с плавающей точкой, в противном случае - false
is_bool(arg)	Возвращает значение true, если arg - булево значение (true или false), в противном случае - false
is_null(arg)	Возвращает значение true, если arg имеет тип null, в противном случае - false
is_string(arg)	Возвращает значение true, если arg - строка, в противном случае - false

Присваивание и приведение типа

Как уже было сказано, интерпретатор PHP часто автоматически преобразует данные из одного типа в другой, если этого требует контекст. Кроме того, в случае необходимости такие преобразования могут быть выполнены принудительно, по указанию программиста, разрабатывающего программу на языке PHP. Но программист должен знать, какие действия происходят в той и другой ситуации.

Правила преобразования типов

Ниже приведены некоторые общие правила, по которым выполняется преобразование данных из одного типа в другой интерпретатором PHP.

- Целое число — в число с плавающей точкой: создается число с плавающей точкой, полностью соответствующее исходному (например, int 4 становится равным float 4.0).
- Число с плавающей точкой — в целое число: дробная часть отбрасывается, т.е. число округляется в сторону нуля.
- Число — в булево значение: FALSE, если число точно равно 0, в противном случае TRUE.
- Число — в строку: создается строка, которая выглядит точно так же, как выглядело бы число после вывода его значения. Вывод целых чисел осуществляется в виде последовательности цифр, а числа с плавающей точкой выводятся с минимально необходимой точностью. Достаточно высокие или низкие значения с плавающей точкой преобразуются в экспоненциальный формат.
- Булево значение — в число: 1, если TRUE; 0, если FALSE.

- Булево значение — в строку: '1', если TRUE, пустая строка, если FALSE.
- Значение NULL — в число: 0.
- Значение NULL — в булево значение: FALSE.
- Строка — в число: преобразование эквивалентно чтению числа из строки с последующим преобразованием в требуемый тип. Если число не может быть считано успешно, то полученное значение является нулевым. Для того чтобы операция чтения считалась выполненной успешно, не обязательно требуется считывать всю строку.
- Строка — в булево значение: FALSE, если это — пустая строка или строка, равная '0', TRUE в противном случае.
- Простой тип (число или строка) — в массив: преобразование эквивалентно созданию нового массива с одним элементом, имеющим индекс нуль.

В приведенном выше списке было отмечено, что значения некоторых типов после преобразования в числовые значения имеют неопределенный результат. В данном контексте под словом "неопределенный" подразумевается, что разработчики PHP еще не пришли к согласию в отношении того, по каким правилам должно осуществляться такое преобразование в текущей и будущих версиях PHP, поэтому в коде не следует осуществлять преобразование подобного рода. Может оказаться, что в какой-то конкретной версии PHP подобные преобразования значений указанных типов в числовые значения осуществляются в программе успешно, но не следует рассчитывать на то, что в следующей версии соответствующее преобразование приведет к получению полезного результата.

Явные преобразования

В языке PHP предусмотрены три способа, с помощью которых программист может манипулировать данными, имеющими разные типы: применение функций преобразования, операторов приведения типа (подобные используемым в языке C) и вызов функции **settype()** с параметрами в виде переменных. Эти способы кратко описаны ниже:

Функции **intval(), floatval () и strval()** преобразуют свои параметры соответственно в целые числа, числа с плавающей точкой или строковые значения.

Любому выражению может предшествовать оператор приведения типа (имя типа в круглых скобках), который преобразует результат выражения в значение требуемого типа.

В качестве первого параметра функции **settype()** можно задать любую переменную, а эта функция преобразует значение переменной из текущего типа в тот тип, который задан в виде второго строкового параметра.

В следующем примере продемонстрированы все 3 версии приведения типов:

```
$str = "101 далматинец";

// Преобразуем строку в целое число
$count = intval($str);
echo $count;                // 101

// Преобразуем строку в булево значение
$bool = (bool)$str;
echo ($bool) ? "TRUE" : "FALSE";      // 'TRUE', т.к. строка не пустая и
не '0'

// Преобразуем строку в число с плавающей точкой
settype($str, "float");
echo gettype($str);      // 'double'
```

Ключевые слова с именами шести основных типов (**integer, float, boolean, string, array и object**) являются допустимыми компонентами в операторах приведения типа, а также допустимыми строковыми параметрами функции settype(). Кроме того, в операторах приведения типа допускается использование некоторых альтернативных ключевых слов, обозначающих имена типов: (**int**) вместо (**integer**), (**double**) или (**real**) вместо (**float**) и (**bool**) вместо **(boolean**).