Основы синтаксиса

В PHP существуют различные пары тегов, которые могут быть использованы для обозначения PHP-кода, в зависимости от того, как был сконфигурирован PHP. Из них только три <?php ?> , <?= ?> и <script language="php"> </script> доступны всегда. Другими являются теги в стиле ASP (*<% echo 'test'; %> ; <%= 'test'; %>*), которые могут быть включены или выключены в конфигурационном файле *php.ini* с помощью директивы *asp_tags*.

PHP требует окончания инструкций точкой запятой в конце каждой инструкции. Закрывающий тег блока PHP-кода автоматически применяет точку с запятой; т.е. нет необходимости ставить точку с запятой в конце последней строки блока с PHP-кодом.

Закрывающий тег PHP-блока в конце файла не является обязательным, и в некоторых случаях его опускание довольно полезно, например, при использовании *include* или *require*, так, что нежелательные пробелы не останутся в конце файла и вы все еще сможете добавить *HTTP-заголовки* после подключения к ответу сервера.

PHP поддерживает комментарии в стиле *C*, *C++* и оболочки *Unix*.

```php
<?php
    echo "Это тест"; // Это однострочный комментарий в стиле C++

    /* Это многострочный комментарий
    еще одна строка комментария */

    echo "Последний тест"; # Это комментарий в стиле оболочки Unix
?>
```

Типы

PHP поддерживает восемь простых типов.

Четыре скалярных типа:

boolean
integer
float (число с плавающей точкой, также известное как double)
string

Три смешанных типа:

array
object
callable

И, наконец, два специальных типа:

resource
NULL

```php
<?php
    $bool = TRUE;
    $str = 'foo';
    $int = 12;

    echo gettype($bool); // boolean
    echo gettype($str); // string
    echo gettype($int); // integer

    if (is_int($int)) {
        echo $int; // 12
    }
?>
```

Булев

Это простейший тип. boolean выражает истинность значения. Он может быть либо TRUE, либо FALSE. Обе они регистронезависимы.

Для явного преобразования в boolean, используйте (bool) или (boolean). Однако, в большинстве случаев приведение типа необязательно, так как значение будет автоматически преобразовано, если оператор, функция или управляющая конструкция требует boolean аргумент.

-1 рассматривается как TRUE, как и любое другое ненулевое (отрицательное или положительное) число!

Целые числа

Integer - это число из множества $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$.

Целые числа могут быть указаны в десятичной (основание 10), шестнадцатеричной (основание 16), восьмеричной (основание 8) или двоичной (основание 2) системе счисления, с необязательным предшествующим знаком (- или +).

Для записи в восьмеричной системе счисления, необходимо поставить пред числом 0 (ноль). Для записи в шестнадцатеричной системе счисления, необходимо поставить перед числом 0x. Для записи в двоичной системе счисления, необходимо поставить перед числом 0b.

```php
<?php
    echo 1234; // 1234 - десятичное число
    echo -123; // -123 - отрицательное число
    echo 0123; // 83 - восьмеричное число (эквивалентно 83 в десятичной системе)
    echo 0x1A; // 26 - шестнадцатеричное число (эквивалентно 26 в десятичной системе)
    echo 0b11111111; // 255 - двоичное число (эквивалентно 255 в десятичной системе)
?>
```

Для явного преобразования в integer, используйте приведение (int) или (integer). Однако, в большинстве случаев, в приведении типа нет необходимости, так как значение будет автоматически преобразовано, если оператор, функция или управляющая структура требует аргумент типа integer. Значение также может быть преобразовано в integer с помощью функции intval().

Числа с плавающей точкой

Числа с плавающей точкой (также известны как "float", "double", или "real").

```php
<?php
echo 1.234; // 1.234
echo 1.2e3; // 1200
echo 7E-10; // 7.0E-10
?>
```

NaN

Некоторые числовые операции могут возвращать значение, представляемое константой NAN. Данный результат означает неопределенное или непредставимое значение в операциях с плавающей точкой. Любое строгое или нестрогое сравнение данного значения с другим значением, включая его самого, возвратит FALSE.

Так как NAN представляет собой неограниченное количество различных значений, то NAN не следует сравнивать с другими значениями, включая ее саму. Вместо этого, для определения ее наличия необходимо использовать функцию is_nan().

Строки

Строка (тип string) - это набор символов, где символ - это то же самое, что и байт. Это значит, что PHP поддерживает ровно 256 различных символов, а также то, что в PHP нет встроенной поддержки Unicode.

Строка может быть определена четырьмя различными способами:

- одинарными кавычками
- двойными кавычками
- heredoc-синтаксисом
- nowdoc-синтаксисом

Одинарные кавычки

Простейший способ определить строку - это заключить ее в одинарные кавычки (символ ').

Чтобы использовать одинарную кавычку внутри строки, проэкранируйте ее обратным слешем (\). Если необходимо написать сам обратный слеш, продублируйте его (\\). Все остальные случаи применения обратного слеша будут интерпретированы как обычные символы: это означает, что если вы попытаетесь использовать другие управляющие последовательности, такие как \r или \n, они будут выведены как есть вместо какого-либо особого поведения.

В отличие от синтаксиса двойных кавычек и heredoc, переменные и управляющие последовательности для специальных символов, заключенных в одинарные кавычки, не обрабатываются.

Двойные кавычки

Самым важным свойством строк в двойных кавычках является обработка переменных.

Heredoc

Третий способ определения строк - это использование heredoc-синтаксиса: <<<. После этого оператора необходимо указать идентификатор, затем перевод строки. После этого идет сама строка, а потом этот же идентификатор, закрывающий вставку.

```php
<?php
$str = <<<EOD
    Lorem ipsum
    Another text here
    End
EOD;

    echo $str; // Lorem ipsum Another text here End
?>
```

Nowdoc

Nowdoc - это то же самое для строк в одинарных кавычках, что и heredoc для строк в двойных кавычках.

Nowdoc указывается той же последовательностью <<<, что используется в heredoc, но последующий за ней идентификатор заключается в одинарные кавычки, например, <<<'EOT'.

```php
<?php
$str = <<<'EOD'
    Lorem ipsum
    Another text here
    End
EOD;

    echo $str; // Lorem ipsum Another text here End
?>
```

Обработка переменных

Если строка указывается в двойных кавычках, либо при помощи heredoc, переменные внутри нее обрабатываются.

Существует два типа синтаксиса: простой и сложный. Простой синтаксис более легок и удобен. Он дает возможность обработки переменной, значения массива (array) или свойства объекта (object) с минимумом усилий.

Сложный синтаксис может быть определен по фигурным скобкам, окружающим выражение.

Простой синтаксис

Если интерпретатор встречает знак доллара ($), он захватывает так много символов, сколько возможно, чтобы сформировать правильное имя переменной. Если вы хотите точно определить конец имени, заключайте имя переменной в фигурные скобки.

```php
<?php
    $juice = "apple";

    echo "He drank some $juice juice."; // He drank some apple juice.
    echo "He drank some juice made of $juices."; // He drank some juice made of .
    echo "He drank some juice made of ${juice}s."; // He drank some juice made of apples.
?>
```

Сложный (фигурный) синтаксис

Он называется сложным не потому, что труден в понимании, а потому что позволяет использовать сложные выражения.

Любая скалярная переменная, элемент массива или свойство объекта, отображаемое в строку, может быть представлена в строке этим синтаксисом. Просто запишите выражение так же, как и вне строки, а затем заключите его в { и } . Поскольку { не может быть экранирован, этот синтаксис будет распознаваться только когда $ следует непосредственно за {. Используйте {\$, чтобы напечатать {$.

```php
<?php
    $world = 'World';
    echo "Hello, {$world}"; // Hello, World
?>
```

Массивы

Массив в PHP - это упорядоченное отображение, которое устанавливает соответствие между значением и ключом. Так как значением массива может быть другой массив PHP, можно также создавать деревья и многомерные массивы.

key может быть либо типа integer, либо типа string. value может быть любого типа.

Дополнительно с ключом key будут сделаны следующие преобразования:

- Строки, содержащие целое число будут преобразованы к типу integer. Например, ключ со значением "8" будет в действительности сохранен со значением 8. С другой стороны, значение "08" не будет преобразовано, так как оно не является корректным десятичным целым.
- Числа с плавающей точкой (тип float) также будут преобразованы к типу integer, т.е. дробная часть будет отброшена. Например, ключ со значением 8.7 будет в действительности сохранен со значением 8.
- Тип bool также преобразовываются к типу integer. Например, ключ со значением true будет сохранен со значением 1 и ключ со значением false будет сохранен со значением 0.
- Тип null будет преобразован к пустой строке. Например, ключ со значением null будет в действительности сохранен со значением "".
- Массивы (тип array) и объекты (тип object) не могут использоваться в качестве ключей. При подобном использовании будет генерироваться предупреждение: Недопустимый тип смещения (Illegal offset type).

Если несколько элементов в объявлении массива используют одинаковый ключ, то только последний будет использоваться, а все другие будут перезаписаны.

Массивы в PHP могут содержать ключи типов integer и string одновременно, так как PHP не делает различия между индексированными и ассоциативными массивами.

Параметр key является необязательным. Если он не указан, PHP будет использовать предыдущее наибольшее значение ключа типа integer, увеличенное на 1. Возможно указать ключ только для некоторых элементов и пропустить для других, при этом отчет начнется с указанного ключа.

Управляющая конструкция foreach существует специально для массивов. Она предоставляет возможность легко пройтись по массиву.

```php
<?php
    $array1 = array(
        "foo" => "bar",
        "bar" => "foo",
    );
    print_r($array1); // Array ( [foo] => bar [bar] => foo )

    // Начиная с PHP 5.4
    $array2 = [
        "foo" => "bar",
        "bar" => "foo",
    ];
    print_r($array2); // Array ( [foo] => bar [bar] => foo )

    $array3 = array(
        1    => "a",
        "1"  => "b",
        1.5  => "c",
        true => "d",
    );
    print_r($array3); // Array ( [1] => d )

    $array4 = array(
        "foo" => "bar",
        "bar" => "foo",
        100   => -100,
        -100  => 100,
    );
    print_r($array4); // Array ( [foo] => bar [bar] => foo [100] => -100 [-100] => 100 )

    $array5 = array("foo", "bar", "hello", "world");
    print_r($array5); // Array ( [0] => foo [1] => bar [2] => hello [3] => world )

    $array6 = array("foo", "bar", 6 => "hello", "world");
    print_r($array6); // Array ( [0] => foo [1] => bar [6] => hello [7] => world )
?>
```

Доступ к элементам массива может быть осуществлен с помощью синтаксиса array[key].

И квадратные и фигурные скобки можно взаимозаменяемо использовать для доступа к элементам массива (т.е. и $array[42] и $array{42} равнозначны).

```php
<?php
    $array = array(
        "foo" => "bar",
        42    => 24,
        "multi" => array(
            "dimensional" => array(
                "array" => "foo"
            )
        )
    );

    echo $array["foo"]; // bar
    echo $array[42]; // 24
    echo $array["multi"]["dimensional"]["array"]; // foo

    function getArray() {
        return array(1, 2, 3);
    }

    $secondElement = getArray()[1];
    echo $secondElement; // 2
?>
```

Попытка доступа к неопределенному ключу в массиве - это то же самое, что и попытка доступа к любой другой неопределенной переменной: будет сгенерирована ошибка уровня E_NOTICE, и результат будет NULL.

Существующий массив может быть изменен явной установкой значений в нем.

Это выполняется присвоением значений массиву array с указанием в скобках ключа. Кроме того, вы можете опустить ключ. В этом случае добавьте к имени переменной пустую пару скобок ([]).

Если массив $arr еще не существует, он будет создан. Таким образом, это еще один способ определить массив array. Однако такой способ применять не рекомендуется, так как если переменная $arr уже содержит некоторое значение (например, значение типа string из переменной запроса), то это значение останется на месте и [] может на самом деле означать доступ к символу в строке. Лучше инициализировать переменную путем явного присваивания значения.

Для изменения определенного значения просто присвойте новое значение элементу, используя его ключ. Если вы хотите удалить пару ключ/значение, вам необходимо использовать функцию unset().

```php
<?php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56; // В этом месте скрипта это то же самое, что и $arr[13] = 56
$arr["x"] = 42; // Добавляет к массиву новый элемент с ключом "x"
unset($arr[5]); // Удаляет элемент из массива
unset($arr); // Удаляет массив полностью
?>
```

Как уже говорилось выше, если ключ не был указан, то будет взят максимальный из существующих целочисленных (integer) индексов, и новым ключом будет это максимальное значение (в крайнем случае 0) плюс 1. Если целочисленных индексов еще нет, то ключом будет 0 (ноль).

```php
<?php
$array = array(1, 2, 3, 4, 5);
print_r($array); // Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )

// Удаляем каждый элемент, но сам массив оставляем нетронутым:
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array); // Array ( )

// Добавляем элемент (обратите внимание, что новым ключом будет 5, вместо 0)
$array[] = 6;
print_r($array); // Array ( [5] => 6 )

// Переиндексация
$array = array_values($array);
$array[] = 7;
print_r($array); // Array ( [0] => 6 [1] => 7 )
?>
```

Ресурс

Resource это специальная переменная, содержащая ссылку на внешний ресурс. Ресурсы создаются и используются специальными функциями.

Для определения того, является ли переменная ресурсом можно использовать функцию is_resource(), а функция get_resource_type() поможет получить тип данного ресурса.

Благодаря системе подсчета ссылок, определение отсутствия ссылок на ресурс происходит автоматически, после чего он освобождается сборщиком мусора. Поэтому, очень редко требуется освобождать память вручную.

Постоянные соединения с базами данных являются исключением из этого правила. Они не уничтожаются сборщиком мусора.

NULL

Специальное значение NULL представляет собой переменную без значения. NULL - это единственно возможное значение типа null.

Переменная считается null, если:

- ей была присвоена константа NULL
- ей еще не было присвоено никакого значения
- она была удалена с помощью unset()

Существует только одно значение типа null - регистронезависимая константа NULL.

Функции обратного вызова (callback-функции)

```php
<?php
    // Пример callback-функции
    function myCallbackFunction() {
        echo 'Hello, World!';
    }

    // Пример callback-метода
    class MyClass {
        function myCallbackMethod() {
            echo 'Hello, World!';
        }
    }

    // Type 1: Простой callback
    call_user_func('myCallbackFunction'); // Hello, World!

    // Type 2: Вызов метода класса
    call_user_func(array('MyClass', 'myCallbackMethod')); // Hello, World!
?>
```

Манипуляции с типами

PHP не требует (и не поддерживает) явного типа при определении переменной; тип переменной определяется по контексту, в котором она используется. То есть, если вы присвоите значение типа string переменной $var, то $var изменит тип на string. Если вы затем присвоите $var значение типа integer, она станет целым числом(integer).

Примером автоматического преобразования типа является оператор умножения '*'.

```php
<?php
    $foo = "1"; // $foo is string
    $foo *= 2; // $foo is now an integer
    $foo = $foo * 1.3; // $foo is now a float
    $foo = 5 * "10 Little Piggies"; // $foo is integer
?>
```

```php
<?php
    $a    = 'car'; // $a - это строка
    $a[0] = 'b';   // $a все еще строка
    echo $a;       // bar
?>
```

Приведение типов в PHP работает так же, как и в C: имя требуемого типа записывается в круглых скобках перед приводимой переменной.

Допускаются следующие приведения типов:

- (int), (integer) - приведение к integer
- (bool), (boolean) - приведение к boolean
- (float), (double), (real) - приведение к float
- (string) - приведение к string
- (array) - приведение к array
- (object) - приведение к object
- (unset) - приведение к NULL

Вместо использования приведения переменной к string, можно также заключить ее в двойные кавычки.

```php
<?php
    $foo = 10; // $foo это целое число
    var_dump($foo); // int(10)
    $bar = (boolean) $foo; // $bar это булев тип
    var_dump($bar); // bool(true)
?>
```