Операторы

Операторы можно сгруппировать по количеству принимаемых ими значений. Унарные операторы принимают только одно значение, например, ! (оператор логического отрицания) или ++ (инкремент). Бинарные операторы принимают два значения; это, например, знакомые всем арифметические операторы + (плюс) и - (минус), большинство поддерживаемых в PHP операторов входят именно в эту категорию. Ну и, наконец, есть всего один тернарный оператор, ? :, принимающий три значения, обычно его так и называют -- "тернарный оператор" (хотя, возможно, более точным названием было бы "условный оператор").

Приоритет оператора

Приоритет оператора определяет, насколько "тесно" он связывает между собой два выражения. Например, выражение 1 + 5 * 3 вычисляется как 16, а не 18, поскольку оператор умножения ("*") имеет более высокий приоритет, чем оператор сложения ("+"). Круглые скобки могут использоваться для принудительного указания порядка выполнения операторов. Например, выражение (1 + 5) * 3 вычисляется как 18.

Если операторы имеют равный приоритет, то будут ли они выполняться справа налево или слева направо определяется их ассоциативностью. К примеру, "-" является лево-ассоциативным оператором. Следовательно 1 - 2 - 3 сгруппируется как (1 - 2) - 3 и пересчитается в -4. С другой стороны "=" - право-ассоциативный оператор, так что $a = $b = $c сгруппируется как $a = ($b = $c).

Неассоциативные операторы с одинаковым приоритетом не могут использоваться совместно. К примеру 1 < 2 > 1 не будет работать в PHP. Выражение 1 <= 1 == 1, с другой стороны, будет, поскольку == имеет более низкий приоритет чем <=.

```php
<?php
    echo 3 * 3 % 5; // (3 * 3) % 5 = 4
    echo true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

    $a = 1;
    $b = 2;
    $a = $b += 3; // $a = ($b += 3)
    echo $a; // 5
    echo $b; // 5

    $c = 1;
    echo $c + $c++; // 3
    echo $c; // 2

    $i = 1;
    $array[$i] = $i++;
    echo $array[2]; // 1
?>
```

+, - и . имеют одинаковый приоритет

```php
<?php
    $x = 4;
    echo "x - 1 = ". $x-1 .", text here."; // -1, text here.
    echo (("x - 1 = ". $x) - 1) . " text here."; // -1 text here
    echo "$x - 1 = ". ($x-1); // x - 1 = 3
?>
```

Арифметические операторы

Арифметические операции

Пример	Название	Результат
+$a	Идентичность	Конвертация *$a* в int или float, что более подходит.
-$a	Отрицание	Смена знака *$a*.
$a + $b	Сложение	Сумма *$a* и *$b*.
$a - $b	Вычитание	Разность *$a* и *$b*.
$a * $b	Умножение	Произведение *$a* и *$b*.
$a / $b	Деление	Частное от деления *$a* на *$b*.
$a % $b	Деление по модулю	Целочисленный остаток от деления *$a* на *$b*.

Результат операции остатка от деления % будет иметь тот же знак, что и делимое — то есть, результат $a % $b будет иметь тот же знак, что и $a.

```php
<?php
    echo 5 % 3; // 2
    echo 5 % -3; // 2
    echo -5 % 3; // -2
    echo -5 % -3; // -2
?>
```

Оператор присваивания

Базовый оператор присваивания обозначается как "=". На первый взгляд может показаться, что это оператор "равно". На самом деле это не так. В действительности, оператор присваивания означает, что левый операнд получает значение правого выражения, (т.е. устанавливается значением).

```php
<?php
    $a = ($b = 4) + 5; // $a теперь равно 9, а $b было присвоено 4.

    $a = 3;
    $a += 5; // устанавливает $a в 8, как если бы мы написали: $a = $a + 5;

    $b = "Hello ";
    $b .= "There!"; // устанавливает $b в "Hello There!", как и $b = $b . "There!";
?>
```

Присваивание по ссылке

Присваивание по ссылке также поддерживается, для него используется синтаксис $var = &$othervar;. 'Присваивание по ссылке' означает, что обе переменные указывают на одни и те же данные и никакого копирования не происходит.

```php
<?php
    $a = 3;
    $b = &$a; // $b - это ссылка на $a

    echo "$a"; // 3
    echo "$b"; // 3

    $a = 4; // меняем $a

    echo "$a"; // 4
    echo "$b"; // также печатает 4, так как $b является ссылкой на $a
?>
```

Побитовые операторы

Побитовые операторы позволяют считывать и устанавливать конкретные биты целых чисел.

Побитовые операторы

Пример	Название	Результат
$a & $b	И	Устанавливаются только те биты, которые установлены и в *$a*, и в *$b*.
$a | $b	Или	Устанавливаются те биты, которые установлены в *$a* или в *$b*.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в *$a*, либо только в *$b*, но не в обоих одновременно.
~ $a	Отрицание	Устанавливаются те биты, которые не установлены в *$a*, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной *$a* сдвигаются на *$b* позиций влево (каждая позиция подразумевает "умножение на 2")
$a >> $b	Сдвиг вправо	Все биты переменной *$a* сдвигаются на *$b* позиций вправо (каждая позиция подразумевает "деление на 2")

Операторы сравнения

В случае, если вы сравниваете число со строкой или две строки, содержащие числа, каждая строка будет преобразована в число, и сравниваться они будут как числа. Эти правила также распространяются на оператор switch. Преобразование типов не происходит при использовании === или !== так как в этом случае кроме самих значений сравниваются еще и типы.

```php
<?php
    var_dump(0 == "a"); // 0 == 0 -> true
    var_dump("1" == "01"); // 1 == 1 -> true
    var_dump("10" == "1e1"); // 10 == 10 -> true
    var_dump(100 == "1e2"); // 100 == 100 -> true
?>
```

Операторы сравнения

Пример	Название	Результат
$a == $b	Равно	**TRUE** если *$a* равно *$b* после преобразования типов.
$a === $b	Тождественно равно	**TRUE** если *$a* равно *$b* и имеет тот же тип.
$a != $b	Не равно	**TRUE** если *$a* не равно *$b* после преобразования типов.
$a <> $b	Не равно	**TRUE** если *$a* не равно *$b* после преобразования типов.
$a !== $b	Тождественно не равно	**TRUE** если *$a* не равно *$b* или они разных типов.
$a < $b	Меньше	**TRUE** если *$a* строго меньше *$b*.
$a > $b	Больше	**TRUE** если *$a* строго больше *$b*.
$a <= $b	Меньше или равно	**TRUE** если *$a* меньше или равно *$b*.
$a >= $b	Больше или равно	**TRUE** если *$a* больше или равно *$b*.

Из-за особого внутреннего представления float, не нужно проверять на равенство два float-числа.

Выражение (expr1) ? (expr2) : (expr3) интерпретируется как expr2, если expr1 имеет значение TRUE, или как expr3 если expr1 имеет значение FALSE.

Выражение expr1 ? : expr3 возвращает expr1 если expr1 имеет значение TRUE, и expr3 в другом случае.

```php
<?php
    $a = 5;

    echo $a == 5 ? : 4; // 1
    echo $a == 5 ? 1 : 0; // 1
?>
```

Сравнение различных типов		
Тип операнда 1	**Тип операнда 2**	**Результат**
null или string	string	**NULL** преобразуется в "", числовое или лексическое сравнение
bool или null	что угодно	Преобразуется в bool, **FALSE** < **TRUE**
object	object	Встроенные классы могут определять свои собственные правила сравнения, объекты разных классов не сравниваются, объекты одного класса - сравниваются свойства тем же способом, что и в массивах (PHP 4), в PHP 5 есть свое собственное объяснение
string, resource или number	string, resource или number	Строки и ресурсы переводятся в числа, обычная математика
array	array	Массивы с меньшим числом элементов считаются меньше, если ключ из первого операнда не найден во втором операнде - массивы не могут сравниваться, иначе идет сравнение соответствующих значений (смотри пример ниже)
array	что угодно	array всегда больше
object	что угодно	object всегда больше

Оператор управления ошибками

PHP поддерживает один оператор управления ошибками: знак (@). В случае, если он предшествует какому-либо выражению в PHP-коде, любые сообщения об ошибках, генерируемые этим выражением, будут проигнорированы.

Оператор @ работает только с выражениями. Есть простое правило: если что-то возвращает значение, значит вы можете использовать перед ним оператор @. Например, вы можете использовать @ перед именем переменной, произвольной функцией или вызовом include, константой и так далее. В то же время вы не можете использовать этот оператор перед определением функции или класса, условными конструкциями, такими как if, foreach и т.д.

Операторы исполнения

PHP поддерживает один оператор исполнения: обратные кавычки (``). Обратите внимание, что это не одинарные кавычки! PHP попытается выполнить строку, заключенную в обратные кавычки, как консольную команду, и вернет полученный вывод (т.е. он не просто выводится на экран, а, например, может быть присвоен переменной).

Обратные кавычки недоступны, в случае, если включен безопасный режим или отключена функция shell_exec().

```php
<?php
    $output = `ls -al`;
    echo "<pre>$output</pre>";

    /*
    total 3
    drwxr-xr-x    3 HP Pavil Администст        0 Oct 10 23:44 .
    drwxr-xr-x    1 HP Pavil Администст     4096 Nov  9 23:08 ..
    -rw-r--r--    1 HP Pavil Администст       60 Nov 13 19:32 index.php
    */
?>
```

Операторы инкремента и декремента

PHP поддерживает префиксные и постфиксные операторы инкремента и декремента в стиле C.

Операторы инкремента/декремента не влияют на булевы значения. Декремент NULL также не даст никакого эффекта, однако инкремент даст значение 1.

Операторы инкремента и декремента

Пример	Название	Действие
++$a	Префиксный инкремент	Увеличивает *$a* на единицу, затем возвращает значение *$a*.
$a++	Постфиксный инкремент	Возвращает значение *$a*, затем увеличивает *$a* на единицу.
--$a	Префиксный декремент	Уменьшает *$a* на единицу, затем возвращает значение *$a*.
$a--	Постфиксный декремент	Возвращает значение *$a*, затем уменьшает *$a* на единицу.

Инкрементирование или декрементирование булевых переменных не приводит ни к какому результату.

Логические операторы

Логические операторы		
Пример	**Название**	**Результат**
$a and $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a or $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.
$a xor $b	Исключающее или	**TRUE** если *$a*, или *$b* **TRUE**, но не оба.
! $a	Отрицание	**TRUE** если *$a* не **TRUE**.
$a && $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a || $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.

Строковые операторы

В PHP есть два оператора для работы со строками (string). Первый - оператор конкатенации ('.'), который возвращает строку, представляющую собой соединение левого и правого аргумента. Второй - оператор присваивания с конкатенацией ('.='), который присоединяет правый аргумент к левому.

Операторы, работающие с массивами

Операторы, работающие с массивами		
Пример	**Название**	**Результат**
$a + $b	Объединение	Объединение массива *$a* и массива *$b*.
$a == $b	Равно	**TRUE** в случае, если *$a* и *$b* содержат одни и те же пары ключ/значение.
$a === $b	Тождественно равно	**TRUE** в случае, если *$a* и *$b* содержат одни и те же пары ключ/значение в том же самом порядке и того же типа.
$a != $b	Не равно	**TRUE**, если массив *$a* не равен массиву *$b*.
$a <> $b	Не равно	**TRUE**, если массив *$a* не равен массиву *$b*.
$a !== $b	Тождественно не равно	**TRUE**, если массив *$a* не равен тождественно массиву *$b*.

Оператор проверки типа

Оператор instanceof используется для определения того, является ли текущий объект экземпляром указанного класса.