### Лекция 5.

### Тема:Переменные, константы и операторы

Важным элементом каждого языка являются *переменные*, *константы* и *операторы*, применяемые к этим *переменным* и *константам*. Рассмотрим, как выделяются и обрабатываются эти элементы в *PHP*.

#### Переменные

*Переменная* в PHP обозначается знаком доллара, за которым следует ее имя. Например:

$my\_var

Имя *переменной* *чувствительно к регистру*, т.е. *переменные* $my\_var и $My\_var различны.

Имена *переменных* соответствуют тем же правилам, что и остальные наименования в PHP: правильное имя *переменной* должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

В PHP 3 *переменные* всегда присваивались по значению. То есть когда вы присваиваете выражение *переменной*, все значения оригинального выражения копируются в эту *переменную*. Это означает, к примеру, что после присвоения одной *переменной* значения другой, изменение одной из них не влияет на значение другой.

<?php

$first = ' Text '; // Присваиваем $first

// значение

// ' Text '

$second = $first; // Присваиваем $second

// значение

// переменной $first

$first = ' New text '; // Изменяем

// значение

// $first

// на ' New text '

echo "Переменная с именем first равна " .$first <br>";

// выводим значение $first

echo "Переменная с именем second равна " .$second";

// выводим значение $second

?>

Пример 2.2. Присваивание по значению

Результат работы этого скрипта будет следующим:

Переменная с именем first равна New text

Переменная с именем second равна Text

PHP 4, кроме этого, предлагает еще один способ присвоения значений *переменным*: присвоение по ссылке. Для того, чтобы присвоить значение *переменной* по ссылке, это значение должно иметь имя, т.е. оно должно быть представлено какой-либо *переменной*. Чтобы указать, что значение одной *переменной* присваивается другой *переменной* по ссылке, нужно перед именем первой *переменной* поставить знак амперсанд &.

Рассмотрим тот же пример, что и выше, только будем присваивать значение *переменной* first *переменной* second по ссылке:

<?php

$first = ' Text '; // Присваиваем $first

// значение ' Text '

$second = &$first;

/\* Делаем ссылку на $first через $second.

Теперь значения этих переменных

будут всегда совпадать \*/

// Изменим значение $first

// на ' New text '

$first = ' New text ';

echo "Переменная с именем first " .

"равна .$first <br>";

// выведем значения обеих переменных

echo "Переменная с именем second " .

"равна $second";

?>

Пример 2.3. Присваивание по ссылке

Этот скрипт выведет следующее:

Переменная с именем first равна New text.

Переменная с именем second равна New text.

То есть вместе с *переменной* $first изменилась и *переменная* $second.

#### Константы

Для хранения постоянных величин, т.е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются ***константы***. Такими величинами могут быть математические *константы*, пароли, пути к файлам и т.п. Основное отличие *константы* от *переменной* состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у *константы* нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения. Как же тогда можно определить *константу*? Для этого существует специальная функция define() . Ее синтаксис таков:

define("Имя\_константы",

"Значение\_константы",

[Нечувствительность\_к\_регистру])

По умолчанию имена *констант* *чувствительны к регистру*. Для каждой *константы* это можно изменить, указав в качестве значения аргумента Нечувствительность\_к\_регистру значение True. Существует соглашение, по которому имена *констант* всегда пишутся в верхнем регистре.

Получить значение *константы* можно, указав ее имя. В отличие от *переменных*, не нужно предварять имя *константы* символом $. Кроме того, для получения значения *константы* можно использовать функцию *constant*() с именем *константы* в качестве параметра.

<?php

// определяем константу

// PASSWORD

define("PASSWORD","qwerty");

// определяем регистронезависимую

// константу PI со значением 3.14

define("PI","3.14", True);

// выведем значение константы PASSWORD,

// т.е. qwerty

echo (PASSWORD);

// тоже выведет qwerty

echo constant("PASSWORD");

echo (password);

/\* выведет password и предупреждение,

поскольку мы ввели регистрозависимую

константу PASSWORD \*/

echo pi;

// выведет 3.14, поскольку константа PI

// регистронезависима по определению

?>

Пример 2.4. Константы в PHP

Кроме *констант*, объявляемых пользователем, о которых мы только что рассказали, в PHP существует ряд *констант*, определяемых самим интерпретатором. Например, *константа* \_\_FILE\_\_ хранит имя файла программы (и путь к нему), которая выполняется в данный момент, \_\_FUNCTION\_\_ содержит имя функции, \_\_CLASS\_\_ – имя класса, PHP\_VERSION – версия интерпретатора PHP. Полный список предопределенных *констант* можно получить, прочитав руководство по PHP.

#### Операторы

***Операторы*** позволяют выполнять различные действия с *переменными*, *константами* и выражениями. Мы еще не упоминали о том, что такое выражение. Выражение можно определить как все, что угодно, что имеет значение. *Переменные* и *константы* – это основные и наиболее простые формы выражений. Существует множество операций (и соответствующих им *операторов* ), которые можно производить с выражениями. Рассмотрим некоторые из них подробнее.

|  |  |  |
| --- | --- | --- |
| Таблица 2.1. *Арифметические операторы* | | |
| **Обозначение** | **Название** | **Пример** |
| + | Сложение | $a + $b |
| - | Вычитание | $a - $b |
| \* | Умножение | $a \* $b |
| / | Деление | $a / $b |
| % | Остаток от деления | $a % $b |
| Таблица 2.2. Строковые операторы | | | | |
| **Обозначение** | **Название** | | | **Пример** |
| . | *Конкатенация* ( *сложение строк* ) | | | $c = $a . $b  (это строка, состоящая из $a и $b ) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 2.3. Операторы присваивания | | | | | |
| **Обозначение** | **Название** | **Описание** | | **Пример** | |
| = | Присваивание | *Переменной* слева от *оператора* будет присвоено значение, полученное в результате выполнения каких-либо операций или *переменной* / *константы* с правой стороны | | $a = ($b = 4) +5;  ( $a будет равна 9, $b будет равна 4 ) | |
| += |  | Сокращение. Прибавляет к *переменной* число и затем присваивает ей полученное значение | | $a += 5;  (эквивалентно $a = $a + 5; ) | |
| .= |  | Сокращенно обозначает комбинацию операций *конкатенации* и присваивания (сначала добавляется строка, потом полученная строка записывается в *переменную* ) | | $b = "Привет ";  $b .= "всем";  (эквивалентно $b = $b . "всем"; )  В результате: $b="Привет всем" | |
| Таблица 2.4. *Логические операторы* | | | | | |
| **Обозначение** | **Название** | | **Описание** | | **Пример** |
| and | И | | $a и $b истинны ( True ) | | $a and $b |
| && | И | |  | | $a && $b |
| or | Или | | Хотя бы одна из *переменных* $a или $b истинна (возможно, что и обе) | | $a or $b |
| || | Или | |  | | $a || $b |
| xor | Исключающее или | | Одна из *переменных* истинна. Случай, когда они обе истинны, исключается | | $a xor $b |
| ! | Инверсия (NOT) | | Если $a=True, то !$a=False и наоборот | | ! $a |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 2.5. Операторы сравнения | | | | | |
| **Обозначение** | **Название** | | **Описание** | | **Пример** |
| == | Равенство | | Значения *переменных* равны | | $a == $b |
| === | Эквивалентность | | Равны значения и *типы**переменных* | | $a === $b |
| != | Неравенство | | Значения *переменных* не равны | | $a != $b |
| <> | Неравенство | |  | | $a <> $b |
| !== | Неэквивалентность | | *Переменные* не эквивалентны | | $a !== $b |
| < | Меньше | |  | | $a < $b |
| > | Больше | |  | | $a > $b |
| <= | Меньше или равно | |  | | $a <= $b |
| >= | Больше или равно | |  | | $a >= $b |
| Таблица 2.6. Операторы инкремента и декремента | | | | | | |
| **Обозначение** | **Название** | **Описание** | | **Пример** | | |
| ++$a | Пре- *инкремент* | Увеличивает $a на единицу и возвращает $a | | <?  $a=4;  echo "Должно быть 4:" .$a++;  echo "Должно быть 5:" .++$a;  ?> | | |
| $a++ | Пост- *инкремент* | Возвращает $a, затем увеличивает $a на единицу | |  | | |
| --$a | Пре- *декремент* | Уменьшает $a на единицу и возвращает $a | |  | | |
| $a-- | Пост- *декремент* | Возвращает $a, затем уменьшает $a на единицу | |  | | |