Переменные

Имя переменной чувствительно к регистру.

Имена переменных соответствуют тем же правилам, что и остальные наименования в PHP. Правильное имя переменной должно начинаться с буквы или символа подчеркивания и состоять из букв, цифр и символов подчеркивания в любом количестве. Это можно отобразить регулярным выражением: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

$this - это особая переменная, которой нельзя ничего присваивать.

По умолчанию, переменные всегда присваиваются по значению. То есть, когда вы присваиваете выражение переменной, все значение оригинального выражения копируется в эту переменную. Это означает, к примеру, что, после того как одной переменной присвоено значение другой, изменение одной из них не влияет на другую.

PHP также предлагает иной способ присвоения значений переменным: присвоение по ссылке. Это означает, что новая переменная просто ссылается (иначе говоря, "становится псевдонимом" или "указывает") на оригинальную переменную. Изменения в новой переменной отражаются на оригинале, и наоборот. Для присвоения по ссылке, просто добавьте амперсанд (&) к началу имени присваиваемой (исходной) переменной.

Важно отметить, что по ссылке могут быть присвоены только именованные переменные.

```php
<?php
    $var = 'Bob';
    $Var = 'Joe';
    echo "$var, $Var"; // Bob, Joe

    $bob = 'Bob';
    $john = $bob;
    $john = "My name is $bob";
    echo $john; // My name is Bob
    echo $bob; // Bob

    $foo = 'Bob';
    $bar = &$foo;
    $bar = "My name is $bar";
    echo $bar; // My name is Bob
    echo $foo; // My name is Bob
?>
```

Область видимости переменной

Область видимости переменной - это контекст, в котором эта переменная определена. В большинстве случаев все переменные PHP имеют только одну область видимости. Эта единая область видимости охватывает также включаемые (include) и требуемые (require) файлы. Например:

```
<?php
    $a = 1;
    include 'b.inc';
?>
```

Здесь переменная $a будет доступна внутри включенного скрипта b.inc.

Однако определение (тело) пользовательской функции задает локальную область видимости данной функции. Любая используемая внутри функции переменная по умолчанию ограничена локальной областью видимости функции.

```
<?php
    $a = 1; /* глобальная область видимости */

    function test() {
        echo $a; /* ссылка на переменную локальной области видимости */
    }

    test(); //
?>
```

Этот скрипт не сгенерирует никакого вывода, поскольку выражение echo указывает на локальную версию переменной $a, а в пределах этой области видимости ей не было присвоено значение.

Ключевое слово global

После определения $a и $b внутри функции как global все ссылки на любую из этих переменных будут указывать на их глобальную версию.

```
<?php
    $a = 1;
    $b = 2;

    function sum() {
        global $a, $b;

        $b = $a + $b;
    }
    echo $b; // 2
    sum();
    echo $b; // 3
?>
```

Второй способ доступа к переменным глобальной области видимости - использование специального, определяемого PHP массива $GLOBALS. Предыдущий пример может быть переписан так:

```php
<?php
    $a = 1;
    $b = 2;

    function sum() {
        $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
    }

    echo $b; // 2
    sum();
    echo $b; // 3
?>
```

$GLOBALS - это ассоциативный массив, ключом которого является имя, а значением - содержимое глобальной переменной. Обратите внимание, что $GLOBALS существует в любой области видимости, это объясняется тем, что $GLOBALS является суперглобальным.

Использование ключевого слова global вне функции не является ошибкой. Оно может быть использовано в файле, который включается внутрь функции.

Использование статических (static) переменных

```php
<?php
    function test() {
        $a = 0;
        echo $a;
        $a++;
    }

    test(); // 0
    test(); // 0
    test(); // 0

    /*
        Эта функция довольно бесполезна, поскольку при каждом вызове она устанавливает $a в 0 и выводит 0.
        Инкремент переменной $a++ здесь не играет роли, так как при выходе из функции переменная $a исчезает.
        Чтобы написать полезную считающую функцию, которая не будет терять текущего значения счетчика,
        переменная $a объявляется как static
    */

    function testStatic() {
        static $a = 0;
        echo $a;
        $a++;
    }

    testStatic(); // 0
    testStatic(); // 1
    testStatic(); // 2

    /*
        Теперь $a будет проинициализирована только при первом вызове функции,
        а каждый вызов функции test() будет выводить значение $a и инкрементировать его.
    */
?>
```

Важной особенностью области видимости переменной является статическая переменная. Статическая переменная существует только в локальной области видимости функции, но не теряет своего значения, когда выполнение программы выходит из этой области видимости.

Статические переменные также дают возможность работать с рекурсивными функциями. Рекурсивной является функция, вызывающая саму себя. При написании рекурсивной функции нужно быть внимательным, поскольку есть вероятность сделать рекурсию бесконечной. Вы должны убедиться, что существует адекватный способ завершения рекурсии. Следующая простая функция рекурсивно считает до 10, используя для определения момента остановки статическую переменную $count.

```php
<?php
    function test() {
        static $count = 0;

        $count++;
        echo $count;

        if ($count < 10) {
            test();
        }
    }

    test(); // 12345678910
?>
```

Переменные переменных

Иногда бывает удобно иметь переменными имена переменных. То есть, имя переменной, которое может быть определено и изменено динамически. Переменная переменной берет значение переменной и рассматривает его как имя переменной.

```php
<?php
    $a = 'Hello';
    $$a = 'World';

    echo "$a, ${$a}"; // Hello, World
    echo "$a, $Hello"; // Hello, World
?>
```

Для того чтобы использовать переменные переменных с массивами, вы должны решить проблему двусмысленности. То есть, если вы напишете $$a[1], обработчику необходимо знать, хотите ли вы использовать $a[1] в качестве переменной, либо вам нужна как переменная $$a, а затем ее индекс [1]. Синтаксис для разрешения этой двусмысленности таков: ${$a[1]} для первого случая и ${$a}[1] для второго.

Переменные извне PHP

HTML-формы (GET и POST)

Когда происходит отправка данных формы PHP-скрипту, информация из этой формы автоматически становится доступной ему.

Доступ к данным из простой HTML POST-формы:
echo $_POST['username'];
echo $_REQUEST['username'];

GET-форма используется аналогично, за исключением того, что вместо POST, вам нужно будет использовать соответствующую предопределенную переменную GET. GET относится также к QUERY_STRING (информация в URL после '?'). Так, например, http://www.example.com/test.php?id=3 содержит GET-данные, доступные как $_GET['id'].

Точки и пробелы в именах переменных преобразуется в знаки подчеркивания. Например, <input name="a.b" /> станет $_REQUEST["a_b"].

```
<?php
    if ($_POST) {
        echo '<pre>';
        echo htmlspecialchars(print_r($_POST, true));
        echo '</pre>';
    }
?>

<form action="" method="POST">
    Name: <input type="text" name="personal[name]" /><br />
    Email: <input type="text" name="personal[email]" /><br />
    City: <br />
    <select multiple name="city[]">
        <option value="baku">Baku</option>
        <option value="saatly">Saatly</option>
        <option value="sabirabad">Sabirabad</option>
    </select><br />
    <input type="submit" value="Send" />
</form>

Result:

    Array
    (
        [personal] => Array
            (
                [name] => Orkhan
                [email] => orkhanalyshov@gmail.com
            )

        [city] => Array
            (
                [0] => baku
                [1] => saatly
            )
    )
```

Имена переменных кнопки-изображения

При отправке формы вместо стандартной кнопки можно использовать изображение с помощью тега такого вида:

<input type="image" src="image.gif" name="sub" />

Когда пользователь щелкнет где-нибудь на изображении, соответствующая форма будет передана на сервер с двумя дополнительными переменными - sub_x и sub_y. Они содержат координаты нажатия пользователя на изображение. На самом деле имена переменных, отправленных браузером, содержат точку, а не подчеркивание, но PHP автоматически конвертирует точку в подчеркивание.

HTTP Cookies

Cookies - это механизм для хранения данных в удаленном браузере и отслеживания и идентификации таким образом вернувшихся пользователей. Вы можете установить cookies, используя функцию setcookie(). Cookies являются частью HTTP-заголовка, поэтому функция SetCookie должна вызываться до того, как браузеру будет отправлен какой бы то ни было вывод. Это ограничение аналогично ограничению функции header(). Данные, хранящиеся в cookie, доступны в соответствующих массивах данных cookie, таких как $_COOKIE и $_REQUEST.

Константы

Константы - это идентификаторы (имена) простых значений. Исходя из их названия, нетрудно понять, что их значение не может изменяться в ходе выполнения скрипта (исключения представляют "волшебные" константы, которые на самом деле не являются константами в полном смысле этого слова). Имена констант чувствительны к регистру. По принятому соглашению, имена констант всегда пишутся в верхнем регистре.

Как и superglobals, константы доступны из любой области видимости. Вы можете использовать константы в любом месте вашего скрипта, не обращая внимания на текущую область видимости.

Вы можете определить константу с помощью функции define() или с помощью ключевого слова const вне объявления класса. В то время как define() позволяет задать константу через выражение, конструкция const ограничена. Если использована конструкция const, константы могут содержать только скалярные данные (boolean, integer, float и string типов).

В отличие от определения констант с помощью функции define(), константы, объявленные с помощью ключевого слова const должны быть объявлены в самой верхней области видимости, потому что они определяются при компилировании скрипта. Это означает, что их нельзя объявлять внутри функций, циклов, выражений if и try/ catch блоков.

```php
<?php
    const SURNAME = 'Alyshov';
    define('NAME', 'Orkhan');

    echo NAME; // Orkhan
    echo SURNAME; // Alyshov
?>
```

"Волшебные" константы

PHP предоставляет большой список предопределенных констант для каждого выполняемого скрипта. Многие из этих констант определяются различными модулями и будут присутствовать только в том случае, если эти модули доступны в результате динамической загрузки или в результате статической сборки.

Есть восемь волшебных констант, которые меняют свое значение в зависимости от контекста, в котором они используются. Например, значение __LINE__ зависит от строки в скрипте, на которой эта константа указана.