Web-разработка на PHP-технологиях

Курс лекций и семинаров для студентов,
желающих научиться основам Web-разработки на PHP

Осень-Зима 2014

Лекция №3

PHP — Синтаксис языка
Переменные, константы, типы данных

Автор: Дмитрий Левин, Senior PHP Developer, Sibers

Sibers®

Переменные и константы

Общие понятия о переменных в PHP

Как и в любом другом языке программирования, в PHP существует такое понятие, как переменная.

Переменная — это область оперативной памяти, доступ к которой осуществляется по имени.

- Все данные, с которыми работает программа, хранятся в виде переменных (исключение — константа, которая, впрочем, может содержать только число или строку
- Такого понятия, как указатель (как в Си), в PHP не существует — при присвоении переменная копируется один-в-один, какую бы сложную структуру она ни имела.
- Имена всех переменных в PHP должны начинаться со знака $ — так интерпретатору значительно легче «понять» и отличить их, например, в строках.
- Имена переменных чувствительны к регистру букв:
  например, $var — не то же самое, что $Var или $VAR

Общие понятия о переменных в PHP

```php
<?php

    $var = "Bob";
    $Var = "Joe";
    echo "$var, $Var";      // выведет "Bob, Joe"
    echo '$var, $Var';      // выведет "$var, $Var"

    $4site = 'not yet';     // неверно; начинается с цифры
    $_4site = 'not yet';    // верно; начинается с символа подчеркивания
```

Общие понятия о константах в PHP

Встречаются случаи, когда переменные довольно неудобно использовать для постоянного хранения каких либо определенных значений, которые не меняются в течение работы программы. Такими значениями могут быть математические константы, пути к файлам, разнообразные пароли и т. д. Как раз для этих целей в PHP предусмотрена такая конструкция, как константа.

Константой называется именованная величина, которая не изменяется в процессе выполнения программы (скрипта).

Особенности констант:

- вы не можете изменять значения констант, которые были им присвоены при их объявлении.
- константы удобно использовать для хранения значений, которые не должны изменяться во время работы программы.
- константы могут содержать только скалярные данные (логического, целого, плавающего и строкового типов).

Общие понятия о константах в PHP

Различия между константами и переменными:

- У констант нет приставки в виде знака доллара ($);
- Константы можно определить только с помощью функции define(), а не присваиванием значения;
- Константы могут быть определены и доступны в любом месте без учета области видимости;
- Константы не могут быть определены или аннулированы после первоначального объявления;
- Константы могут иметь только скалярные значения.

Общие понятия о константах в PHP

```php
<?php

    define('DB_USER', 'root');
    define('DB_PASS', 'qwerty');
    define('DB_NAME', 'test');

    var_dump(DB_USER, DB_PASS, DB_NAME);

//  string(4) "root"
//  string(6) "qwerty"
//  string(4) "test"
```

Типы данных

Типы данных

- boolean (двоичные данные)
- integer (целые числа)
- float (число с плавающей точкой или 'double')
- string (строки)
- array (массивы)
- object (объекты)
- resource (ресурсы)
- NULL (пустой тип)
- mixed (смешанный тип)
- number (числа)
- callback (обратного вызова)

Типы данных: **Boolean**

Тип **Boolean** (двоичные данные)
Это простейший тип. Он выражает истинность значения — это может быть либо **TRUE**, либо **FALSE**.

```
<?php
        $x = true; // присвоить $x значение TRUE
  if($x == true){
        $x = false;
        }
```

Типы данных: **Integer**

Тип integer (целые числа)

Целые могут быть указаны в десятичной, шестнадцатеричной или восьмеричной системе исчисления, по желанию, с предшествующим знаком (- или +).

```
$a = 1234; // десятичное число
$a = -123; // отрицательное число
$a = 0123; // восьмеричное число (эквивалентно 83 в десятичной системе)
$a = 0x1A; // шестнадцатеричное число (эквивалентно 26 в десятичной системе)
```

Типы данных: **Float**

Тип **float** (числа с плавающей точкой)

Числа с плавающей точкой (они же числа двойной точности или действительные числа)

```
$a=1.234;
$b=1.2e3;
$c=7E-10;
```

Никогда не доверяйте точности чисел с плавающей точкой до последней цифры, и не проверяйте напрямую их равенство

Типы данных: **String**

Тип string (строки)

Строка в PHP — это набор символов любой длины. В отличие от Си, строки могут содержать в себе также и нулевые символы, что никак не повлияет на программу.

Иными словами, строки можно использовать для хранения бинарных данных. Длина строки ограничена только размером свободой оперативной памяти.

$a = "Это просто текст, записанный в строковую переменную";

Типы данных: **Array**

Тип array (массивы)

Массив в PHP — это упорядоченный набор данных, в котором установлено соответствие между значением и ключом.

Индекс (ключ) служит для однозначной идентификации элемента внутри массива. В одном массиве не может быть двух элементов с одинаковыми индексами.

Типы данных: **Array**

Простой массив (список)
Массивы, индексами которых являются числа, начинающиеся с нуля - это списки:

```php
<?php

    // Простой способ инициализации массива
    $names[0]   =   "Апельсин";
    $names[1]   =   "Банан";
    $names[2]   =   "Груша";
    $names[3]   =   "Помидор";
    // Здесь: names - имя массива, а 0, 1, 2, 3 - индексы массива

    // Альтернативный и более верный способ
    $names = array(
        "Апельсин",
        "Банан",
        "Груша",
        "Помидор"
    );
```

Типы данных: **Array**

Ассоциативные массивы

В PHP индексом массива может быть не только число, но и строка.

Причем на строку не накладываются ограничения: она может содержать пробелы, специальные символы и быть любой длины.

Массивы, индексами которых являются строки, называются ассоциативными массивами. Индексы ассоциативных массивов называются ключами.

Пример ассоциативного массива:

```php
<?php

    // Ассоциативный массив
    $names["Иванов"]    =   "Иван";
    $names["Сидоров"]   =   "Николай";
    $names["Петров"]    =   "Петр";
    // В данном примере: фамилии - ключи ассоциативного массива
```

Типы данных: **Array**

Многомерные массивы

```php
<?php

    // Многомерный массив
    $Array["Ivanov"]    = array(
        "name"  =>  "Иванов И.И.",
        "age"   =>  "25",
        "email" =>  "ivanov@mail.ru"
    );
    $Array["Petrov"]    = array(
        "name"  =>  "Петров П.П.",
        "age"   =>  "34",
        "email" =>  "petrov@mail.ru"
    );
    $Array["Sidorov"]   = array(
        "name"  =>  "Сидоров С.С.",
        "age"=>"47",
        "email"=>"sidorov@mail.ru"
    );
```

```php
$array = array(
    "peoples" => array(
        "ivanov"  => array(
            "name"  =>  "Иванов И.И.",
            "age"   =>  "25",
            "email" =>  "ivanov@mail.ru"
        ),
        "petrov" => ...
    )
);
```

Типы данных: **Object**

Тип object (объекты)

Объект является одним из базовых понятий объектно-ориентированного программирования.

Для доступа к отдельным элементам и функциям используется оператор **->**, а не квадратные скобки.

Для инициализации объекта используется выражение **new**, создающее в переменной экземпляр объекта.

Типы данных: **Object**

```php
<?php

    class ExampleObject {

        function test(){
            echo 'Text from the test function';
        }

    }

    $object = new ExampleObject();
    $object->test();
```

Типы данных: **NULL**

Тип NULL (пустой тип)

Специальное значение NULL говорит о том, что эта переменная не имеет значения. NULL — это единственно возможное значение типа NULL (пустой тип).

Переменная считается NULL если:

- ей была присвоена константа NULL;
- ей еще не было присвоено какое-либо значение;
- она была удалена с помощью unset().

```php
<?php

    $var = null;

    if(is_null($var)){
        echo '$var is empty';
    }
```

```php
<?php

    $var  = null;
    $bool = false;

    if ($var == $bool){
        var_dump('equal');
    }

    if ($var === $bool){
        var_dump('equal');
    }
```

Типы данных: **Mixed**

Псевдотип mixed (смешанный тип)

mixed говорит о том, что параметр может принимать множество (но не обязательно все) типов

```php
<?php

/**
 * Render list
 * @param mixed $value (string or array)
 */
function listToString($value){
    if(is_array($value)){
        return implode(',', $value);
    }
    return $value;
}

echo listToString('Cars');
// вывод : Cars

echo listToString(array('Cars', 'Airplanes', 'Trains'));
// вывод Cars,Airplans,Trains
```

Типы данных: Callback

Псевдотип callback (обратного вызова)

Некоторые функции, такие как call_user_func() или usort() принимают в качестве параметра определенные пользователем callback-функции.

Callback-функции могут быть не только простыми функциями, но также методами объектов, включая статические методы классов.

PHP-функция передается просто как строка ее имени

```php
<?php
    // простой пример callback
    function my_callback_function() {
        echo 'hello world!';
    }
    call_user_func('my_callback_function');

    // примеры callback-метода
    class MyClass {
        function myCallbackMethod() {
            echo 'Hello World!';
        }
    }
    // вызов метода статического класса без создания объекта
    call_user_func(array('MyClass', 'myCallbackMethod'));
```

Sibers

Область видимости переменной

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
<?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.

Область видимости переменной

```php
<?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.

Область видимости переменной

```php
<?php

    $a = 1;
    $b = 2;

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

        $b = $a + $b;
    }

    Sum();
    echo $b;
```

Вышеприведенный скрипт выведет 3. После определения $a и $b внутри функции как global все ссылки на любую из этих переменных будут указывать на их глобальную версию. Не существует никаких ограничений на количество глобальных переменных, которые могут обрабатываться функцией.

Область видимости переменной

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'];
    }

    Sum();
    echo $b;
```

$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.

Заключение:

- в третьей лекции мы рассмотрели общие понятия о переменных и константах в PHP
- разобрали 11 типов данных
- обсудили область видимости переменной

В следующей лекции:

PHP — Синтаксис языка
Операторы

Полезные ссылки:

http://www.php.su/learnphp/vars/?basic
http://www.php.su/learnphp/datatypes/
http://php.net/manual/ru/language.variables.scope.php
http://php.net/manual/ru/reserved.variables.globals.php