Самоучитель

Максим Кузнецов, Игорь Симдянов

PHP 5/6

Третье издание

Объектно-ориентированное программирование

Работа с базами данных

Защита приложений от взлома

Актуальные примеры

Новое в PHP 5/6

bhv

УДК 681.3.06
ББК 32.973.26-018.2
К89

Кузнецов, М. В.

К89 Самоучитель PHP 5/6 / М. В. Кузнецов, И. В. Симдянов —
3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 672 с.: ил.

ISBN 978-5-9775-0409-6

Описаны самые последние версии языка разработки серверных сценариев PHP — 5.3 и 6.0. Рассмотрены основы языка, вопросы объектно-ориентированного программирования на PHP, обработки исключительных ситуаций, взаимодействия с MySQL, регулярные выражения, работа с электронной почтой. Книга содержит множество примеров, взятых из реальной практики разработки динамических Web-сайтов.

Третье издание книги, ранее выходившей под названием "Самоучитель PHP 5", существенно переработано, дополнено и будет интересно не только программистам, впервые знакомящимся с языком, но и читателям предыдущих изданий книги и профессионалам.

Для программистов и Web-разработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 03.12.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 54,18.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.002108.02.07
от 28.02.2007 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0409-6

Оглавление

Глава 2

Быстрый старт

Несмотря на то, что PHP является универсальным языком программирования и может применяться для разработки практически любого программного обеспечения, основная его специализация — Web-разработка. Именно с этой целью он и проектировался, поэтому содержит множество инструментов для работы в Интернете.

> ***Замечание***
>
> В данной и последующих главах будем исходить из того, что интерпретатор PHP и Web-сервер установлены у вас на локальной машине и связаны таким образом, что скрипты, помещенные в файл с расширение php, обрабатываются интерпретатором, и по запросу к Web-серверу в браузере выводится результат работы. Если у вас не настроена связка Web-сервера и PHP, вы можете обратиться к *приложениям 1* и *2*, в которых детально описывается процесс развертывания Web-среды в операционных системах Windows и UNIX.

2.1. Скрипты

Язык программирования PHP считается скриптовым языком, поэтому программы, написанные на нем, называют *скриптами*. Главное отличие традиционных программ от скриптов заключается в том, что скрипты работают только в определенной среде и используют ресурсы данной среды. Программы, будь то компилируемые или интерпретируемые, не зависят от какого-то стороннего программного обеспечения. Например, скриптовый язык программирования JavaScript работает только в Web-браузерах, Visual Basic for Applications только в среде Microsoft Office. С использованием этих языков программирования невозможно создать программу, работающую без соответствующей среды. В случае PHP в качестве такой среды выступает Web-окружение (Web-сервер, сервер базы данных, почтовый сервер и т. п.).

Замечание

Впрочем, язык программирования PHP допускает создание программ, работающих независимо от Web-сервера, однако в такой форме он не получил сколько бы то ни было широкого распространения.

Одной из главных особенностей языка программирования PHP является тот факт, что его код может располагаться вперемешку с HTML-кодом. Для того чтобы интерпретатор PHP различал HTML- и PHP-код, последний заключается в специальные теги `<?php` и `?>`, между которыми располагаются конструкции и операторы языка программирования PHP. В листинге 2.1 приводится классический пример, выводящий в окно браузера при помощи конструкции `echo` фразу "Hello world!" ("Привет мир!"). Содержимое листинга 2.1 следует поместить в файл с расширением php, например, в файл index.php. По умолчанию файлы с другими расширениями не обрабатываются PHP-интерпретатором, и PHP-код остается необработанным. Впрочем, такое поведение Web-сервера можно изменить *(см. приложение 1)*.

Замечание

`echo` — это конструкция языка программирования, позволяющая вывести одну или несколько строк в окно браузера. Более подробно конструкция `echo` обсуждается в *разд. 5.2*, посвященном строковым функциям.

Листинг 2.1. Простейший PHP-скрипт, index.php

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?php
       echo "Hello world!";
    ?>
  </body>
</html>
```

В результате работы скрипта из листинга 2.1 в окно браузера будет выведена фраза "Hello world!" (рис. 2.1).

При работе с серверными языками программирования, такими как PHP, следует помнить, что скрипты, расположенные между тегами `<?php` и `?>`, выполняются на сервере. Клиенту приходит лишь результат работы PHP-кода, в чем можно легко убедиться, просмотрев исходный код HTML-страницы. В листинге 2.2 приводится результат работы PHP-скрипта из листинга 2.1, который браузер получает от сервера.

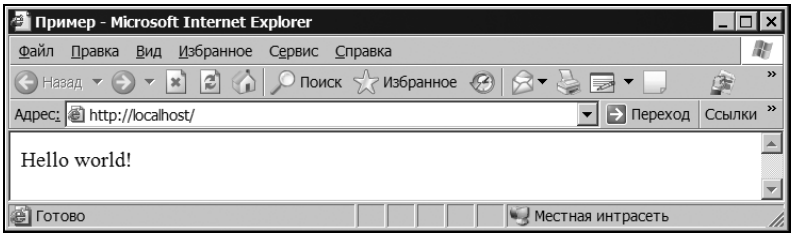

Рис. 2.1. Результат работы скрипта из листинга 2.1

Замечание

Для просмотра исходного текста необходимо щелкнуть правой кнопкой мыши по странице и выбрать в контекстном меню соответствующий пункт, название которого зависит от браузера. В Internet Explorer необходимо выбрать пункт **Просмотр HTML-кода**, в Opera — **Исходный текст**, в FireFox — **View Page Source**.

Листинг 2.2. Исходный код HTML-страницы, которую получает браузер

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    Hello world!  </body>
</html>
```

Если связка PHP и Web-сервера настроена неправильно, или PHP-скрипт был размещен в файле с расширением, отличным от php, например, в файле index.html, то в исходном коде HTML-страницы вместо содержимого листинга 2.2 можно увидеть содержимое листинга 2.1. В этом случае все, что расположено между тегами `<?php` и `?>`, будет рассматриваться браузером как один большой неизвестный ему HTML-тег, а окно браузера будет пустым (рис. 2.2).

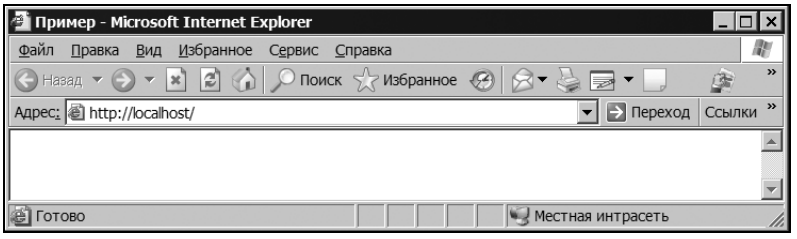

Рис. 2.2. PHP-код не подвергся интерпретации

2.2. Начальные и конечные теги

Как было указано в предыдущем разделе, PHP-скрипт должен быть размещен между начальным тегом `<?php` и конечным тегом `?>` для того, чтобы интерпретатор мог разделить HTML- и PHP-коды. Даже если HTML-код не используется, указание PHP-тегов является обязательным, в противном случае PHP-код будет выведен в окно браузера как есть, без интерпретации. Помимо тегов `<?php` и `?>`, PHP поддерживает еще ряд тегов, представленных в табл. 2.1.

Замечание

До версии PHP 6.0 поддерживались начальные и конечные теги, характерные для ASP-скриптов, `<%` и `%>`, включить которые можно было в конфигурационном файле php.ini, назначив директиве `asp_tags` значение `On`. В версии PHP 6.0 данный тип тегов был исключен.

Таблица 2.1. *Варианты PHP-тегов*

Вариант тегов	Описание
`<?php ... ?>`	Классический вариант PHP-тегов, именно его рекомендуется использовать в повседневной практике
`<? ... ?>`	Краткий вариант PHP-тегов, работает только в том случае, если в конфигурационном файле php.ini включена (т. е. принимает значение `On`) директива `short_open_tag`. По умолчанию эта директива включена, однако все-таки лучше ориентироваться на полный вариант PHP-тегов `<?php ... ?>`. Последнее особенно актуально при использовании PHP совместно с XML-кодом, который использует теги `<?xml ... ?>`. При использовании коротких тегов может возникнуть неоднозначность в интерпретации

`<script language="php">`
`...`
`</script>`
	Расширенный вариант PHP-тегов, так же как и вариант `<?php ... ?>`, доступен в любой момент без дополнительных настроек конфигурационного файла php.ini. Данный вариант несколько громоздок и распознается не всеми PHP-редакторами, тем не менее, следует быть готовыми встретить и такой вариант в PHP-скриптах

`<?= ... ?>`	Специальный вид тегов, предназначенный для вывода простого выражения в окно браузера. Конструкция `<?= ... ?>` эквивалентна `<?php echo ... ?>`

Отдельно следует отметить конструкцию `<?= ... ?>`, которую удобно использовать, когда скрипт состоит из одного выражения.

Например, скрипт из листинга 2.1 можно было бы переписать так, как это продемонстрировано в листинге 2.3.

Листинг 2.3. Использование тегов <?= ... ?>

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?= "Hello world!" ?>
  </body>
</html>
```

Следует отметить, что HTML-страница может содержать более чем одну PHP-вставку. В листинге 2.4 приводится пример, который содержит две вставки: одна задает название страницы (в HTML-теге `<title>`), а вторая определяет содержимое страницы (в HTML-теге `<body>`).

> ***ЗАМЕЧАНИЕ***
>
> В листинге 2.4 используется функция `date()`, которая возвращает текущую дату и время. Более подробно эта функция обсуждается в *главе 11*, посвященной функциям даты и времени.

Листинг 2.4. Допускается несколько PHP-вставок в HTML-код

```
<html>
  <head>
    <title><?php echo "Вывод текущей даты" ?></title>
  </head>
  <body>
    <?php
      echo "Текущая дата:<br>";
      echo date(DATE_RSS);
    ?>
  </body>
</html>
```

Результат работы скрипта из листинга 2.4 представлен на рис. 2.3.

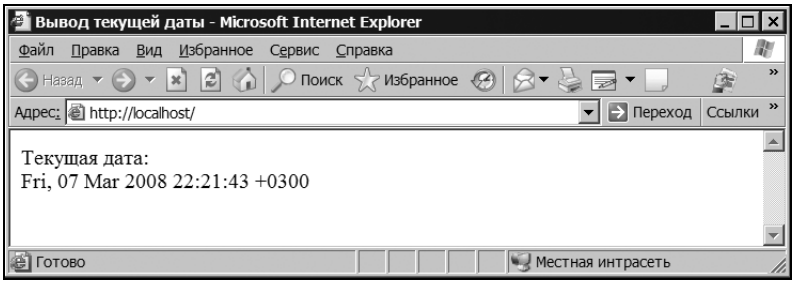

Рис. 2.3. Результат работы скрипта из листинга 2.4

2.3. Использование точки с запятой

Совокупность конструкций языка программирования, завершающуюся точкой с запятой, будем называть *выражением*.

Как видно из листинга 2.4, после строки "Вывод текущей даты" не указывается точка с запятой. Выражение одно, и надобность отделять его от других выражений отсутствует. Однако, как можно видеть во второй вставке, в конце каждой из конструкций `echo` имеется точка с запятой. Если забыть указать этот разделитель, интерпретатор языка программирования PHP посчитает выражение на новой строке продолжением предыдущего и не сможет корректно разобрать скрипт. В результате будет сгенерировано сообщение об ошибке "Parse error: syntax error, unexpected T_ECHO, expecting ',' or ';'" ("Ошибка разбора: синтаксическая ошибка, неожиданно встречена конструкция echo, ожидается либо запятая ',', либо точка с запятой ';'").

Последнее выражение перед завершающим тегом `?>` может не снабжаться точкой с запятой не только в тегах `<?= ... ?>`, но и во всех остальных вариантах, рассмотренных в табл. 2.1. Например, в листинге 2.4 после выражения `echo date(DATE_RSS)` точку с запятой можно не указывать. Однако настоятельно рекомендуется не пользоваться этой особенностью и помещать точки с запятой после каждого выражения, т. к. добавление новых операторов может привести к появлению трудноулавливаемых ошибок.

Переводы строк никак не влияют на интерпретацию скрипта, выражение может быть разбито на несколько строк — интерпретатор PHP будет считать, что выражение закончено лишь после того, как обнаружит точку с запятой или завершающий тег `?>`. В листингах 2.5 и 2.6 представлены два скрипта, аналогичные по своей функциональности.

Листинг 2.5. Использование точки с запятой

```
<?php
  echo 5 + 5;
```

```
  echo 5 - 2;
  echo "Hello world!";
?>
```

Листинг 2.6. Альтернативная запись скрипта из листинга 2.5

```
<?php
echo(5
     +
     5); echo(5-
     2); echo("Hello world!"
     );
?>
```

ЗАМЕЧАНИЕ

Следует избегать конструкций, подобных той, которая приведена в листинге 2.6. Понятно написанный код — залог успеха, т. к. такой код намного проще и быстрее отлаживать. А искусство программиста, как известно, состоит не в безошибочном написании кода (ошибки делают все), а в умении его отлаживать.

2.4. Составные выражения. Фигурные скобки

Фигурные скобки позволяют объединить несколько выражений в группу, которую обычно называют *составным выражением* (листинг 2.7).

Листинг 2.7. Составное выражение

```
<?php
  {
    echo 5 + 5;
    echo 5 - 2;
    echo "Hello world!";
  }
?>
```

Само по себе составное выражение практически никогда не используется, основное его предназначение — работа совместно с условными операторами, операторами цикла и т. п., которые подробно рассматриваются в *главе 5*.

Составное выражение может быть расположено в нескольких PHP-вставках. В листинге 2.8 приводится пример двух составных выражений, которые разбиты несколькими PHP-вставками. Задача скрипта сводится к случайному выводу в окно браузера либо зеленого слова "Истина", либо красного слова "Ложь". Без использования фигурных скобок оператор `if` распространял бы свое действие только на одно выражение, использование составного выражения позволяет распространить его действие на несколько простых выражений.

Замечание

Логический оператор `if` рассматривается в *главе 5*, а функция `rand()`, генерирующая случайное число из заданного интервала, обсуждается в *главе 12*.

Листинг 2.8. Составное выражение может располагаться в нескольких PHP-вставках

```
<?php
  if(rand(0,1))
  {
    ?>
    <div style='color:green'>
    <?php
      echo "Истина";
    ?>
    </div>
    <?php
  }
  else
  {
    ?>
    <div style='color:red'>
    <?php
      echo "Ложь";
    ?>
    </div>
    <?php
  }
?>
```

Как видно из листинга 2.8, составное выражение в любой момент может быть прервано тегами `<?php` и `?>`, а затем продолжено. Впрочем, существуют исключения, например, составное выражение, применяемое для формирования класса, нельзя

разбивать тегами `<?php` и `?>`. Возможно, в следующих версиях языка это ограничение будет устранено.

2.5. Комментарии

Код современных языков программирования является достаточно удобным для восприятия человеком по сравнению с машинными кодами, ассемблером или первыми языками программирования высокого уровня. Тем не менее, конструкции языка продиктованы архитектурой компьютера, и, создавая программы, разработчик волей не волей использует компьютерную, а не человеческую логику. Это, зачастую приводит к созданию достаточно сложных построений, которые нуждаются в объяснении на обычном языке. Для вставки таких пояснений в код предназначены *комментарии*.

PHP предоставляет несколько способов для вставки комментариев, варианты которых представлены в табл. 2.2.

***Таблица 2.2.** Варианты комментариев*

Комментарий	Описание
`// ...`	Комментарий в стиле языка C++, начинающийся с символа двух слэшей (`//`) и заканчивающийся переводом строки
`# ...`	Комментарий в стиле скриптовых языков UNIX, начинающийся с символа диеза (`#`) и заканчивающийся переводом строки
`/* ... */`	Если два предыдущих комментария ограничены лишь одной строкой, то комментарий в стиле языка C `/* ... */` является многострочным

В листинге 2.9 демонстрируется использование всех трех видов комментариев из табл. 2.2.

Листинг 2.9. Комментарии

```
<?php
  echo "Hello"; // это комментарий
  echo "Hello"; # и это комментарий
  /*
    а это многострочный
    комментарий
  */
?>
```

Естественно, что комментарии PHP действуют только внутри тегов-ограничителей `<?php ... ?>`. То есть если символы комментариев будут находиться вне тегов-ограничителей, то они, как и любой текст, будут отображены браузером (листинг 2.10).

Листинг 2.10. Комментарии действуют только внутри тегов <?php и ?>

```
<?php
  echo "Hello<br>"; // нормальный комментарий
?>
// этот текст отобразится браузером.
<!-- Этот текст не будет отображен браузером, поскольку заключен между симво-
лами, являющимися комментариями HTML. Однако он может быть просмотрен в исход-
ном коде HTML-страницы -->
```

Результат работы скрипта из листинга 2.10 представлен на рис. 2.4.

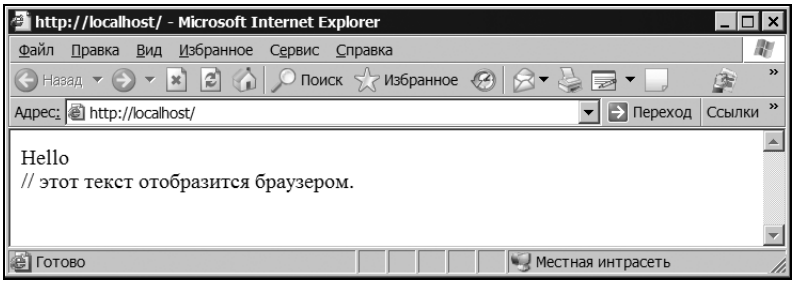

Рис. 2.4. Результат работы скрипта из листинга 2.10

Комментарии можно вставлять не только после выражения после точки с запятой, но и посередине. Так, скрипт, представленный в листинге 2.11, является вполне корректным.

Листинг 2.11. Комментарий в списке аргументов функции

```
<?php
  echo strstr( // эту функцию мы рассмотрим позднее
              "Hello, world", "H");
?>
```

Глава 3

Переменные и типы данных

Ключевым объектом практически любого языка программирования является переменная. Под *переменной* в общем случае понимается именованная область памяти. В этой области может храниться либо строка, либо число, которыми можно манипулировать при помощи имени переменной (его мы далее для простоты будем называть просто переменной). То, что хранится в области памяти, будем называть *значением переменной*.

3.1. Объявление переменной. Оператор =

В PHP переменные начинаются со знака доллара (`$`), за которым может следовать любое количество буквенно-цифровых символов и символов подчеркивания, но первый символ не может быть цифрой. Таким образом, допустимы следующие имена переменных: `$n`, `$n1`, `$user_func_5` и т. д. В отличие от ключевых слов, имена переменных в PHP *чувствительны к регистру*, т. е. переменные `$user`, `$User` и `$USER` являются различными (листинг 3.1).

Листинг 3.1. Зависимость переменных от регистра

```
<?php
  $user = "Владимир";
  $User = "Дмитрий";
  $USER = "Юрий";
  echo $user; // Владимир
  echo $User; // Дмитрий
  echo $USER; // Юрий
?>
```

Как видно из листинга 3.1, для присвоения значения переменной необходимо воспользоваться оператором присваивания =, который позволяет инициализировать переменную.

При объявлении числовых значений в качестве разделителя целого значения и дробной части выступает точка (листинг 3.2).

Листинг 3.2. Объявление чисел

```
<?php
  $number = 1;
  $var = 3.14;
?>
```

Допускается инициализация одним значением сразу нескольких переменных за счет того, что оператор = возвращает результат присвоения. В листинге 3.3 переменным `$num`, `$number` и `$var` присваивается значение 1 в одну строку за счет использования оператора = в цепочке.

Листинг 3.3. Инициализация трех переменных одним значением

```
<?php
  $num = $number = $var = 1;
?>
```

3.2. Типы данных

Язык PHP является слаботипизированным, т. е. переменные языка не требуют строгого задания типа при их объявлении, а в ходе выполнения программы тип переменной может быть практически всегда изменен неявным образом без специальных преобразований. Например, переменная, объявленная строкой, может использоваться далее в арифметических операциях, выступать как логическая переменная, а в конце ей в качестве значения может быть присвоен объект. Все это позволяет разработчику практически не задумываться о типах данных. Тем не менее, некоторые ограничения на типы переменных все-таки накладываются, поэтому о типах переменных все же следует иметь представление. В табл. 3.1 представлены типы данных, которые поддерживаются PHP.

ЗАМЕЧАНИЕ

Тип `int64` планируется ввести в PHP 6.0.

Таблица 3.1. *Типы данных PHP*

Тип данных	Описание
`boolean`	Логический тип, способный принимать лишь два значения: `TRUE` (истина) и `FALSE` (ложь). Более подробно данный тип обсуждается в *главе 5*
`integer`	Целое число, разрядность которого зависит от архитектуры компьютера. Если разрядность операционной системы составляет 32 бита, то число может принимать значения от –2 147 483 648 до 2 147 483 647. Если разрядность составляет 64 бита, то число может принимать значение от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
`int64`	Целое число, разрядность которого не зависит от архитектуры компьютера и всегда равна 64 битам. Таким образом, переменные данного типа могут принимать значения от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
`double` (или `float`)	Вещественное число, минимально возможное значение которого составляет $\pm1.7\times10^{-308}$, а максимально возможное $\pm1.7\times10^{308}$
`string`	Строковый тип, может хранить строку произвольного объема
`array`	Массив — это объединение нескольких однотипных переменных под одним именем. Обращаться к отдельным переменным можно при помощи индекса массива. Более подробно массивы обсуждаются в *главе 6*
`object`	Объект. Это конструкция, объединяющая несколько разнотипных переменных и методы их обработки. Более подробно объекты обсуждаются в *главе 18*
`resource`	Дескриптор, позволяющий оперировать тем или иным ресурсом, доступ к которому осуществляется при помощи библиотечных функций. Дескрипторы применяются при работе с файлами, базами данных, динамическими изображениями и т. д. Более подробно дескрипторы будут рассмотрены в соответствующих главах
`NULL`	Специальный тип, который сигнализирует о том, что переменная не была инициализирована

3.3. Целые числа

Целые числа являются наиболее распространенными в программировании. Это связано с тем, что большинство прикладных задач носит арифметический характер,

а также с тем, что это наиболее быстродействующий тип данных. Если для обработки чисел с плавающей точкой используется математический сопроцессор с достаточно сложными алгоритмами их обработки, то оперировать целыми числами процессор может напрямую.

Замечание

В отличие от других языков программирования переполнения (т. е. выхода значения за допустимые границы) в PHP не бывает, если полученное значение не убирается в `integer`, для него автоматически выбирается больший тип данных (`int64` или `double`).

Язык PHP является C-подобным, поэтому целочисленная переменная может быть объявлена несколькими способами (листинг 3.4).

Листинг 3.4. Объявление целочисленных переменных

```
<?php
  $num = 1234; // десятичное число
  $num = +123; // десятичное число
  $num = -123; // отрицательное число
  $num = 0123; // восьмеричное число (эквивалентно 83)
  $num = 0x1A; // шестнадцатеричное число (эквивалентно 26)
?>
```

Целое положительное число объявляется, как правило, без указания плюса перед ним (впрочем, указанием символа плюса + не приводит к ошибке). Для объявления отрицательного числа перед ним размещается символ минуса -.

По умолчанию считается, что числа задаются в десятичной систем исчисления, однако PHP позволяет объявлять переменные в восьмеричной и шестнадцатеричной системах счисления. В восьмеричной системе счисления основанием служит число 8, а для выражения всех чисел используются цифры от 0 до 7. Число 8 в восьмеричной системе счисления имеет то же значение, что число 10 в десятичной. Шестнадцатеричная система счисления (основанием служит число 16) использует цифры от 0 до 9 и буквы английского алфавита от A до F, означающие шестнадцатеричные "цифры" 10, 11, 12, 13, 14 и 15. Числа, объявленные в восьмеричной системе счисления, предваряются префиксом 0, в шестнадцатеричной системе — префиксом 0x.

Замечание

Может показаться, что восьмеричные и шестнадцатеричные числа являются избыточным наследием языка C, однако, как будет показано дальше, они применяются и в Web-разработке, например, при задании прав доступа к файлам и папкам.