ПЕРЕМЕННЫЕ И КОНСТАНТЫ

Переменные

Основной способ сохранения информации в ходе выполнения программы PHP состоит в использовании переменных; этот способ предусматривает выбор некоторого имени переменной и привязку к нему того значения, которое потребуется в процессе дальнейшего выполнения программы.

Ниже приведены наиболее важные сведения о переменных в языке PHP, которые следует знать:

- Все переменные в языке PHP обозначаются префиксом в виде знака доллара (**$**).
- Значением переменной является значение, присвоенное ей **в последней по времени операции присваивания**.
- Присваивание значений переменным осуществляется с помощью операции присваивания; при этом переменная должна находиться слева от знака операции присваивания (=), а вычисляемое выражение — справа.
- Переменные могут быть объявлены перед присваиванием им значений, но такое требование не является обязательным.
- **Переменные не имеют связанного с ними типа**, отличного от типа своего текущего значения.
- Переменные, используемые до того, как им будет присвоено значение, имеют значения, заданные по умолчанию.

Имена всех переменных в языке PHP начинаются с префикса в виде знака $. Остальная часть имен переменных, находящаяся за начальным префиксом $, должна состоять из букв в коде ASCII (прописных и строчных), цифр (0-9) и символов подчеркивания (_). Кроме того, первым символом после знака $ не должна быть цифра.

Присваивание значений переменным

Операция присваивания значения переменной очень проста — достаточно лишь написать имя переменной, ввести после него один знак равенства (=), затем ввести выражение, значение которого требуется присвоить переменной:

```
$pi = 3 + 0.14159;    // Приближенное значение числа "пи"
```

Следует учитывать, что переменной присваивается результат выражения, а не само выражение. Например, после присваивания значения с помощью приведенного выше оператора невозможно определить, что значение, присвоенное переменной $pi, было получено путем сложения двух чисел.

Вполне возможно, что в программе фактически потребуется вывести на внешнее устройство приведенное математическое выражение, а не вычислять его значение. Программист может вынудить интерпретатор PHP рассматривать оператор присваивания переменной результата математического вычисления как оператор присваивания строкового значения, заключив это выражение в кавычки:

```
$pi = "3 + 0.14159";
```

Во многих языках программирования попытка использовать переменную до того, как ей будет присвоено значение, рассматривается как ошибка. В некоторых других языках такая возможность предоставляется, но может оказаться, что чтение значения этой переменной равносильно чтению сформированного случайным образом содержимого какой-то области памяти. В языке PHP по умолчанию применяется параметр конфигурации, касающийся активизации сообщений об ошибках, который позволяет использовать переменные без присвоенных им значений, не вызывая активизации ошибок, а интерпретатор PHP гарантирует, что такие переменные будут иметь вполне приемлемые значения, заданные по умолчанию.

Переменные в языке PHP не имеют связанных с ними типов, поэтому в отношении переменной, приведенной в тексте программы, нельзя узнать заранее, будет ли она использоваться для хранения числа или строки символов. Как же в этом случае можно определить, какое заданное по умолчанию значение должна иметь переменная, если ей еще не присвоено значение?

Ответ на этот вопрос состоит в том, что тип переменной с не присвоенным значением (как и в случае переменной с присвоенным значением) интерпретируется в зависимости от контекста, в котором используется эта переменная. В той ситуации, когда ожидается появление числа, вырабатывается число; аналогичный принцип применяется в отношении символьных строк. В любом контексте, в котором переменная рассматривается как число, переменная с не присвоенным значением рассматривается как имеющая значение 0, а в любом контексте, в котором ожидается строковое значение, переменная с не присвоенным значением рассматривается как имеющая в качестве значения пустую строку (строку длиной нуль символов).

Поскольку язык PHP не предъявляет такого требования, чтобы переменной было обязательно присвоено значение перед ее использованием, в некоторых ситуациях можно фактически изменять код выполнения программы, избирательно присваивая или не присваивая значение переменной! В языке PHP предусмотрена функция IsSet, которая проверяет переменную для определения того, было ли этой переменной присвоено значение.

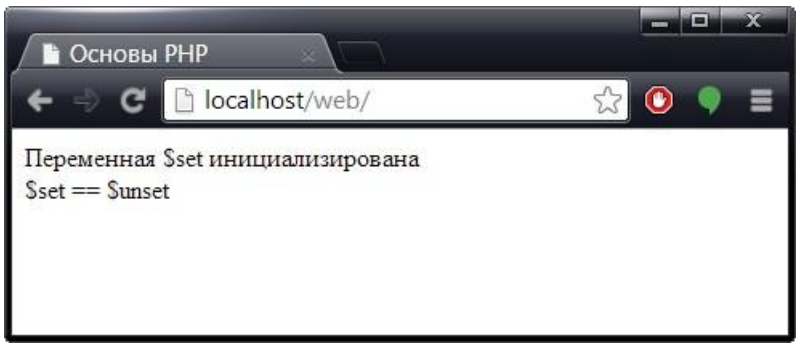
Как показано в приведенном ниже коде, переменную с неприсвоенным значением можно отличить даже от переменной, которой было присвоено значение, предусмотренное по умолчанию:

```
<?php
       error_reporting(0);     // Отключим сообщения об ошибках
       $set = 0;       // Переменной $set присвоено значение
       // $unset; - эта переменная не задана

       if (isset($set)) echo 'Переменная $set инициализирована<br>';
       if (isset($unset)) echo 'Переменная $unset инициализирована<br>';

       if ($set == $unset) echo '$set == $unset';
?>
```

Безусловно, на первый взгляд это может показаться странным, но приведенный выше код вырабатывает следующий результат:



Переменной $unset никогда еще не присваивалось значение, поэтому она вырабатывает нулевое значение, когда ожидается число (как в проверке со сравнением на равенство, которое показывает, что две рассматриваемые переменные имеют одинаковое значение). Тем не менее функция IsSet позволяет провести различие между переменными $set и $unset.

Область определения переменной

Область определения — это формальный термин, используемый при формулировке правил, касающихся того, в каких ситуациях некоторое имя (скажем, переменной или функции) имеет один и тот же смысл в разных местах и в каких ситуациях два полностью совпадающих имени фактически ссылаются на разные объекты.

Любая переменная PHP, не заданная в функции, имеет глобальную область определения, и действие переменной распространяется за пределы данного конкретного "потока" выполнения. Иными словами, если какой-то переменной присваивается значение в начале файла сценария PHP, то переменная с этим именем сохраняет одно и то же значение в ходе обработки остальной части файла сценария, и если переменной не присваивается какое-то другое значение, то она сохраняет прежнее значение до конца сеанса выполнения данного кода (исключением являются переменные с теми же именами, находящиеся в теле функций).

Ниже показан расширенный пример использования глобальных и локальных переменных, показывающий их различия между собой:

```
<!DOCTYPE HTML>
<html>
```

```
<head>
<meta charset="utf-8">
<title>Основы PHP</title>
<?php
	$number = 25;	// Это глобальная переменная, ее можно использовать в последующих сценариях
?>
</head>
<body>
<?php
	echo $number;
	function func1() {
		if (isset($number)) {	// В контексте функции переменной $number не существует
			echo '<br>Переменная $number существует<br>';
		}
		else {
			echo ('<br>Переменная $number не существует<br>');
		}
		// Внешнюю переменную можно сделать доступной внутри тела функции только
		// с использованием ключевого слова global
		global $number;
		if (isset($number)) {
			echo 'А теперь переменная $number существует и она равна '.$number.'<br>';
		}
		$local = 11;	// Это локальная переменная, ее можно использовать только в теле функции
	}
	func1();
?>
</body>
</html>
```

Глобальные переменные внутри функций следует использовать в редких случаях, поскольку легко изменить значение переменной по ошибке, не предусмотрев последствий. Ошибки такого типа бывает очень сложно обнаружить.

Статические переменные – это переменные, которые не исчезают после завершения функции. Значение статической переменной можно снова использовать при следующем вызове функции – она по-прежнему будет иметь то же значение, которое получила при последнем вызове функции. Проще всего считать такую переменную глобальной, но доступной только для данной функции. Статическая переменная объявляется с помощью ключевого слова **static**:

```
<?php
	// Определение функции
	function birthday(){
		// Определить переменную age как статическую
		static $age = 0;
		// Увеличить значение переменной age на 1
		$age = $age + 1;
		// Вывести значение статической переменной age
		echo "Вызов функции birhday: $age<br />";
```

```
        }
        // Присвоить переменной age значение 30
        $age = 30;
        // Вызвать функцию дважды
        birthday();
        birthday();
        // Вывести значение переменной age
        echo "Возраст: $age<br />";
?>
```

В результате исполнения этого фрагмента будет выведено:

Для предоставления информации об окружении, в котором работает PHP-сценарий, PHP использует специальные переменные, которые называются **суперглобальными (super globals)**. Эти переменные не нужно объявлять, как глобальные, они автоматически становятся общедоступными и содержат важные сведения об окружении сценария, например, данные, полученные от пользователя.

Начиная с версии PHP 4.0.1 суперглобальные переменные определены как массивы, их можно вызывать как внутри функций, так и внутри остального контекста программы. Старые суперглобальные переменные, носящие имена с префиксом $HTTP_* и расположенные не в массивах, по-прежнему существуют, но их не рекомендуется использовать из соображений безопасности. В таблице ниже приведен перечень суперглобальных массивов, появившихся в версии PHP 4.0.1:

Суперглобальные массивы в PHP

Имя переменной массива	Описание
$GLOBALS	Содержит все глобальные переменные, доступные локальному сценарию. Имена переменных используются как индексы массива
$_SERVER	Содержит информацию об окружении веб-сервера

$_GET	Содержит информацию о запросах GET (при отправке форм). Эти значения следует обязательно проверять перед использованием
$_POST	Содержит информацию о запросах POST (другой тип отправки форм). Эти значения следует обязательно проверять перед использованием
$_COOKIE	Содержит информацию о cookies HTTP
$_FILES	Содержит информацию о файлах, загружаемых методом POST
$_ENV	Содержит информацию об окружении сценариев
$_REQUEST	Содержит информацию о пользовательском вводе. Эти значения следует обязательно проверять пред использованием. Вместо этого массива следует использовать $_GET или $_POST, т.к. они более специализированные
$_SESSION	Содержит информацию из всех переменных, зарегистрированных в рамках сессии (сеанса пользователя)

Примером суперглобальной переменной может служить $_SERVER["PHP_SELF"]. Эта переменная содержит имя исполняемого сценария и входит в состав массива $_SERVER:

Использование переменной PHP_SELF

```
<?php
	// Выведет расположение текущего файла
	echo $_SERVER["PHP_SELF"];
?>
```

Суперглобальные переменные обеспечивают удобный способ доступа к информации об окружении сценария – от настроек сервера до введенных пользователем данных.

Константы

В программе вы можете определять константы. Значение константы, как следует из ее названия, не может изменяться во время исполнения программы. Константы определяют с помощью функции **define()**, которой в первом аргументе передается имя константы, а во втором – ее значение. Константы имеют глобальную область видимости и могут принимать значение любого элементарного (скалярного) типа данных, например, строки или числа.

Чтобы получить значение константы, достаточно просто обратиться к ее имени или воспользоваться функцией **constant**. В отличие от переменных, перед именем константы знак доллара ($) не ставится.

Если имя константы хранится в переменной или возвращается функцией, то чтобы получить значение константы, необходимо воспользоваться функцией **constant(имя_константы)**. Эта функция получает имя константы в качестве аргумента и возвращает ее значение. Кроме того, с помощью функции **get_defined_constants()** можно получить список (в виде массива) всех определенных вами констант.

Отличия констант и переменных:

- в именах констант принято использовать только заглавные буквы;
- имена констант не начинаются со знака доллара ($);
- определить константу можно только с помощью функции **define**, а не простым оператором присваивания;
- константы определяются и доступны глобально;
- после объявления константу нельзя переопределить или отменить;
- константы могут иметь только скалярные значения (числа и строки).

В примере ниже показан порядок использования констант:

Использование констант в программе

```
<?php
	define("HELLO", "Привет, мир!");
	echo HELLO;
	$constant_name = "HELLO";
	echo constant($constant_name);
?>
```

Этот код дважды выведет значение константы HELLO. В константе полезно хранить значение, которое должно оставаться неизменным, например, путь к файлу настроек.

PHP предоставляет несколько предопределенных констант, похожих на суперглобальные переменные. Примеры таких констант: константа **__FILE__** – возвращает имя исполняющегося PHP-файла, константа **__LINE__** – возвращает номер строки этого файла. Видно, что имя предопределенной константы начинается и заканчивается двумя символами подчеркивания. Эти константы удобно использовать для вывода сообщений об ошибках, поскольку с их помощью можно указать, при исполнении какой строки возникла ошибка.