

Специфика синтаксиса

Глобальные переменные:

```
define();
```

Переменные переменных

```
$foo = "bar";  
$$foo = 'baz';
```

После выполнения второго оператора у переменной `$bar` будет значение `"baz"`. Значение переменной `foo` рассматривается как имя переменной.

Переменные-ссылки

```
$black = &$white
```

Области видимости переменной:

- Локальная область: переменные, объявленные в функции, локальны для функции. Но созданная внутри цикла - все равно функция.
- Глобальная: перед переменной поставить `global`

```
function updateCounter()  
{  
    global $counter;  
    $counter++;  
}  
$counter = 10;  
updateCounter();  
echo $counter;
```

В массиве `$GLOBALS` хранятся глобальные переменные, доступные напрямую

```
function updateCounter()  
{  
    $GLOBALS[counter]++;  
}
```

```
}
```

- Статическая переменная сохраняет значение между вызовами функции.
Объявляется через `static`

`isset()` проверяет, существует ли переменная, `unset()` уничтожает переменную.

Операторы

Приор.	Оператор	Описание
21	<code>clone, new</code>	создание нового объекта
20	<code>[</code>	индекс массива
19	<code>~</code>	побитное отрицание
	<code>++</code>	инкремент
	<code>--</code>	декремент
	<code>(int), (bool), (float), (string), (array), (object), (unset)</code>	Приведение типов <div>\$b = (int) \$a;</div>
	<code>@</code>	подавление ошибок
18	<code>instanceof</code>	проверка типа
17	<code>!</code>	Логическое отрицание
16	<code>* / %</code>	Умножение, деление, остаток от деления
15	<code>+ - .</code>	Сложение, вычитание, конкатенация строки
14	<code><< >></code>	Побитный сдвиг влево, побитный сдвиг вправо
13	<code>< <= > >=</code>	Меньше, меньше или равно, больше, больше или равно
12	<code>== != <> === !==</code>	Равно, не равно, тип и значение равны, тип и значение не равны
11	<code>&</code>	Побитное и
10	<code>^</code>	Побитное исключающее или
9	<code> </code>	Побитное или
8	<code>&&</code>	Логическое и
7	<code> </code>	Логическое или
6	<code>?:</code>	Условный оператор
5	<code>= += -=</code> и т.д.	Присваивание

Приор.	Оператор	Описание
4	and	Логическое и
3	xor	Логическое исключающее или
2	or	Логическое или
1	,	разделитель списка

Префиксный и постфиксный способ записи

```
$m = ++$var;//увеличит значение на 1 и вернет значение
$m = $var++;//вернет значение и увеличит значение на 1
```

Выполнение команд оболочки

Обратные кавычки.

```
$listing = `ls -al /tmp`;
```

Условные операторы

Стандартный синтаксис:

```
if (условие)
    {действия}
else
    {действия}
```

Тенарный синтаксис:

```
if($active) { echo "да";} else { echo "нет";}
```

Альтернативный синтаксис:

```
<? if ($user_validated) :?>
<table>
<tr>
<td>Имя:</td><td>Sophia</td>
</tr>
<tr>
<td>Фамилия:</td><td>lee</td>
</tr>
</table>
```

```
<? else: ?>
```

Пожалуйста войдите.

```
<? endif ?>
```

Switch

```
switch ($name) {  
  case 'one':  
    //do something  
    break;  
  case 'two':  
    //do something  
    break;  
}
```

```
switch ($name):  
  case 'one':  
    //do something  
    break;  
  case 'two':  
    //do something  
    break;  
endswitch;
```

Циклы

```
while (условие) {  
  //операторы  
}
```

```
while (условие):  
  //операторы  
endwhile;
```

continue / break - следующая итерация / выход из текущего цикла. break 2 - выход из 2 текущих циклов.

```
do  
  оператор  
while (условие);
```

```
for ($counter=0; $counter<10; $counter++){  
    оператор  
}
```

Счетчик по массиву

```
foreach ($mass as $elem) {  
    оператор  
}
```

```
foreach ($mass as $key => $val) {  
    оператор  
}
```

Альтернативный способ:

```
foreach ($mass as $key => $val):  
    оператор  
endforeach;
```

Обработка ошибок

```
function inverse($x) {  
    if (!$x) {  
        throw new Exception('Деление на ноль.');    }  
    return 1 / $x;  
}  
  
try {  
    echo inverse(5) . "\n";  
    echo inverse(0) . "\n";  
} catch (Exception $e) {  
    echo 'PHP перехватил исключение: ', $e->getMessage(), "\n";  
}
```

Включение кода

include - включение статического кода

```
<?php include "header.html"; ?>  
содержимое страницы  
<?php include "footer.html"; ?>
```

require - включение динамического кода

```
require "codelib.php";  
mysub(); //определено в codelib.php
```

Лучше:

```
<?php require "design.php";  
header(); ?>  
content  
<?php footer(); ?>
```

Подавление ошибок:

```
@include
```

Если в файле конфигурации PHP (php.ini) включена опция `allow_url_fopen`, вы можете включать файлы, находящиеся на удаленных узлах, указав URL вместо пути к файлу

```
include "http://www.exomple.com/codelib.php";
```

`include_once` и `require_once` - однократная загрузка в пределах скрипта

Область видимости включаемых файлов = области видимости точки включения.

Revision #8

Created 3 July 2024 05:20:21 by Admin

Updated 5 July 2024 16:19:24 by Admin