PHP. Основы

Обозначение PHP кода

- <?php echo("если вы хотите работать с документами XHTML или XML, делайте так\n"); ?>
- <script language="php">
  echo ("некоторые редакторы (например, FrontPage) не любят инструкции обработки");
  </script>
- <? echo ("это простейшая инструкция обработки SGML\n"); ?> Способ доступен, если использовать short_tags() (только в PHP 3), включив установку short_open_tag в конфигурационном файле PHP, либо скомпилировав PHP с параметром --enable-short-tags для configure.
- <% echo ("Вы можете по выбору использовать теги в стиле ASP"); %> Способ доступен, если теги в стиле ASP были включены, используя конфигурационную установку asp_tags.

Комментарии

- // Это однострочный комментарий в стиле c++
- /* Это многострочный комментарий
  еще одна строка комментария */
- # Это комментарий в стиле оболочки Unix

Переменные

```
$a_1 = 4;
$isVar; //принято в JS
$is_var; //принято в PHP
echo $a_1. '<br>';
var_dump($user); // выводится: string(14) " фраза "
```

isset - определяет, установлена ли переменная.

```
echo isset ($a); // TRUE
```

unset - разустанавливает данную переменную.

```
unset ($a);
```

Типы данных

PHP поддерживает следующие типы данных:

1. String – Строки. Строка может быть любым текстом внутри кавычек. Можно использовать одинарные или двойные кавычки.
   ```
   echo "Фраза". '<br>' .'Фраза2,' . '<br>';
   $user = 'Souron';
   echo $user[0]. $user[5];
   ```
2. Integer - Целые числа. Целое число может быть положительным или отрицательным. Целые числа могут быть указаны в десятичной (основание 10), шестнадцатеричной (основание 16), восьмеричной (основание 8) или двоичной (основание 2) системе счисления.
   PHP имеет следующие функции, чтобы проверить, является ли тип переменной целочисленным (integer):
   is_int()
   is_integer() - псевдоним is_int()
   is_long() - псевдоним is_int()

   echo strlen("Hello world!"); // выведет 12 – **длина строки**
   PHP функция str_word_count(str) считает количество слов в строке.
   PHP функция strrev(str) переворачивает строку.
   PHP функция strpos(str, text) ищет определенный текст в строке. Если совпадение найдено, функция возвращает позицию символа первого совпадения. Если совпадений не найдено, возвращается FALSE.
   ```
   echo strpos("Hello world!", "world"); // выведет 6
   ```
   str_replace (mixed search, mixed replace, mixed subject [, int &count]). Эта функция возвращает строку или массив subject, в котором все вхождения search заменены на replace.

3. Float (числа с плавающей запятой - также называемые double).
   ```
   echo 42.6;
   ```
4. Boolean – Логические.

```
$x = true;
```

5.

```
Array – Массив                  // ассоциативные ассивы
$a = ['пн', 'вт', 5];           $a = ['Вася'=>300, 'girls'=>['Маша','Даша','Лена']];
echo $a; // Array               echo $a['Вася'];
var_dump($a);                   echo $a['girls'];
echo $a[0];                     echo $a['girls'][1];  //300ArrayДаша
```

6. Object – Объект. Классы и объекты - два основных аспекта объектно-ориентированного программирования. **Класс** - это шаблон для объектов, а **объект** - это экземпляр класса.

7. NULL – Ноль. Специальное значение NULL говорит о том, что эта переменная не имеет значения. Если переменная создается без значения, ей автоматически присваивается значение NULL. Переменные также могут быть очищены путем установки значения в NULL.
8. Resource – Ресурс. Это хранение ссылки на функции и ресурсы, внешние по отношению к PHP.

Константы

Именованная величина, которая не изменяется в процессе выполнения программы. В PHP константы определяются функцией define(). Эта функция имеет следующий формат:
define ($name, $value, $case_sen), где:
$name - имя константы;
$value - значение константы;
$case_sen - необязательный параметр логического типа,указывающий, следует ли учитывать регистр букв (true) или нет (false).

```
define("pi",3.14,true);
```

Константы могут быть определены и доступны в любом месте без учета области видимости. Константы могут иметь только скалярные значения.

Для проверки существования константы можно использовать функцию defined(). Данная функция возвращает true, если константа объявлена. Приведем пример:

```
define("pi",3.14,true);
if (defined("pi")==true) echo "Константа pi объявлена!";
```

Математические функции

В PHP есть набор математических функций, которые позволяют выполнять математические задачи с числами.
echo(pi()); - возвращает значение ПИ // вернёт 3.1415926535898
echo(min(0, 150, 30, 20, -8, -200)); - возвращает минимум // вернёт -200
echo(max(0, 150, 30, 20, -8, -200)); - возвращает максимум // вернёт 150
echo(abs(-6.7)); - возвращает абсолютное (положительное) значение числа // вернёт 6.7
echo(sqrt(64)); - возвращает квадратный корень числа // вернёт 8
echo(round(0.49)); - округляет число с плавающей запятой до ближайшего целого числа // вернёт 0
echo(rand(min, max)); – генерирует случайное число в промежутке между min и max включительно.

Операторы

```
echo 2 + 3;
echo 2 - 3;
echo 2 * 5;
echo 10 / 2;
echo $a % $b;
echo $a ** $b;
-$a – смена знака
$a += 4;
$user = 'имя '; $user .= 'фамилия';
```

Инкремент, декремент

```
echo $a++ . '<br>';
echo ++$a . '<br>';
```

Побитовые операторы

$a & $b - Побитовое 'и'. Устанавливаются только те биты, которые установлены и в $a, и в $b.
$a | $b - Побитовое 'или'. Устанавливаются те биты, которые установлены либо в $a, либо в $b.
$a ^ $b - Исключающее или. Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b
~ $a - Отрицание. Устанавливаются те биты, которые в $a не установлены, и наоборот.
$a << $b - Сдвиг влево. Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает 'умножение на 2')
$a >> $b - Сдвиг вправо. Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает 'деление на 2')

операторы сравнения - все аналогично JS.

логические операторы

$a and $b, $a && $b - Логическое 'и'
$a or $b, $a || $b - Логическое 'или'
$a xor $b - Исключающее 'или'. TRUE если $a, или $b TRUE, но не оба.
! $a – Отрицание

типизация переменных

```
echo 3 + '4'; // строка + число = число
```

Операторы Array

Оператор	Имя	Пример	Результат
+	Объединение	$x + $y	Объединение $x и $y
==	Равенство	$x == $y	Возвращает true если $x и $y имеют одинаковые пары ключ/значение
===	Идентичность	$x === $y	озвращает true, если $x и $y имеют одинаковые пары ключ/значение в одном и том же порядке и одинаковых типов
!=	Неравенство	$x != $y	Возвращает true если $x не равен $y
<>	Неравенство	$x <> $y	Возвращает true если $x не равен $y
!==	Неидентичность	$x !== $y	Возвращает true если $x не идентичный $y

Операторы условного присваивания

Оператор	Имя	Пример	Результат
?:	Тройной	$x = *expr1* ? *expr2* : *expr3*	Возвращает значение $x. Значение $x равно *expr2*, если *expr1* = TRUE. Значение $x равно *expr3*, если *expr1* = FALSE
??	Нулевое слияние	$x = *expr1* ?? *expr2*	Возвращает значение $x. Значение $x равно *expr1*, если *expr1* существует и не равно NULL. Если *expr1* не существует или имеет значение NULL, значение $x равно *expr2*. Введено в PHP 7

if else

```
if ($a > $b) {
    echo "a больше, чем b";
} else {
    echo "a НЕ больше, чем b";
}

$a = null; // или $a = '';
if (empty($a)) echo 'пусто'; else echo 'не пусто'; // сокращенный вариант
```

Конструкция elseif

```
if ($a > $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";
}
```

Конструкция if-else имеет еще один альтернативный синтаксис:

```
if (логическое_выражение):
команды;
elseif (другое_логическое_выражение):
другие_команды;
else:
иначе_команды;
endif
```

ЦИКЛЫ

```
for ($i=0; $i < 3; $i++) {
  echo 'Название поста' . ' ';
}
```

```
foreach (массив as $значение)
команды;
$days = ['пн','вт','ср','чт','пт','сб','вс'];
foreach ($days as $value) {
 echo $value . ' ';
}

foreach (массив as $ключ=>$значение)
команды;
$posts = [
  'Налоги для фрилансера' => [
    'date' => '23 сентября',
    'author' => 'Иван'
  ],
  'Как найти заказы' => [
    'date' => '26 июля',
    'author' => 'Григорий'
  ],
];
```

```
$posts = 0;
while ($posts <= 10) {
  echo 'Пост' . '<br>';
  $posts++;
}

$x = 1;
do {
    echo $x;
} while ($x++<10);
```

```
foreach ($posts as $post => $value) {
 ?>
 <p><?php echo $post . ' - ' . $value['date']; ?></p>
 <?php
};
```

switch

```
$lang = 'ru';
switch ($lang) {
 case 'ru':
  echo 'русский';
  break;
 default:
  echo 'язык не опознан';
  break;
}
```

функции

```
function get_sum($a, $b) {
 echo $a + $b;
}
get_sum(10, 5);
```

Особенности пользовательских функций в PHP:

- Доступны параметры по умолчанию. Есть возможность вызывать одну и ту же функцию с переменным числом параметров;
- Пользовательские функции могут возвращать любой тип;
- Область видимости переменных внутри функции является иерархической (древовидной);
- Есть возможность изменять переменные, переданные в качестве аргумента (если они переданы по ссылке).
- При вызове функции funct() нужно указать все передаваемые параметры, поскольку они являются обязательными.

Ссылки

Жесткая ссылка представляет собой просто переменную, которая является синонимом другой переменной. Чтобы создать жесткую ссылку, нужно использовать оператор & (амперсанд). Жесткие ссылки удобно применять при передаче параметров пользовательской функции и возврате значения из нее.

```
$b = &$a;
```

Символическая ссылка — это всего лишь строковая переменная, хранящая имя другой переменной (переменная переменная). Чтобы добраться до значения переменной, на которую ссылается символическая ссылка, необходимо применить дополнительный знак $ перед именем ссылки. Рассмотрим пример:

```
$a=10;
$p="a"; // (присваиваем $p имя другой переменной)
echo $$p; // выводит переменную, на которую ссылается $p, т. е. $a
$$p=100; // присваивает $a значение 100
```

При **удалении ссылки**, просто разрывается связь имени и содержимого переменной. Это не означает, что содержимое переменной будет разрушено. Например:

```
$a = 1;
$b =& $a;
unset($a);
Этот код не сбросит $b, а только $a.
```

Включения

require имя_файла;

При запуске (именно при запуске, а не при исполнении!) программы интерпретатор просто заменит инструкцию на содержимое файла имя_файла (этот файл может также содержать сценарий на PHP, обрамленный, как обычно, тэгами <? и ?>). Причем сделает он это непосредственно **перед запуском программы** (в отличие от include, который рассматривается ниже).

include имя_файла;

В отличие от конструкции require конструкция include позволяет включать файлы в код PHP скрипта **во время выполнения сценария**.
require_once и include_once – однократное включение - перед включением файла интерпрететор проверяет, включен ли указанный файл ранее или нет. Если да, то файл не будет включен вновь.