Введение в PHP.

Все файлы скриптов на PHP имеют вид “название_файла.php”.

Теги PHP

PHP код встраивается в html с помощью специальных тегов:

Вариант 1 - в стиле XML:

```
<?php
...
?>
```

Вариант 2 - в стиле ASP:

```
<%
...
%>
```

Вариант 3 - в стиле сценариев HTML:

```
<script language="php">
...
</script>
```

Комментарии в PHP

```
/* Многострочный комментарий
в стиле классического Си */
// Однострочный комментарий в стиле C++
#  Однострочный комментарий в стиле Perl
```

Создание страницы на php

Самый простой скрипт который выводит информацию о PHP выглядит следующим образом:

```
<?php
   phpinfo();
?>
```

Данный скрипт следует сохранить в файл с именем index.php в корне вашего сервера и перейти в браузере по ссылке http://имя_хоста, после чего вы должны увидеть информацию о PHP.

Составной оператор в PHP

В PHP составной оператор выглядит также как и в языке C++, то есть заключается в фигурные скобки { и }. Нужен он для того, чтобы выполнять какую-то группу операторов. Наиболее часто это используется при использовании условий if и циклов for, while, repeat (подробное описание этих операторов разберем в следующих уроках).

```
<?php
{
echo "Этот текст выводится посредством PHP<br>";
echo 10*5;
}
?>
```

Основы языка PHP

Константы

Константы определяются в PHP-программе с помощью функции **define( )**. Например:

```
define("PI", 3.1415927);
```

После определения константа не может быть изменена. В имени константы обычно используются только заглавные буквы.
PHP имеет ряд предопределенных констант. Например:

- **_FILE_** содержит имя файла, который в данный момент читает PHP;
- **_LINE_** содержит номер строки этого файла.

Переменные

Имя любой переменной в PHP начинается со знака **$**. Имена переменных *чувствительны к регистру* символов.

Тип переменной не требуется задавать специально. Конкретный тип переменной устанавливается и меняется в ходе выполнения программы.
PHP поддерживает восемь типов данных:

- логический (принимает значения true или false);
- целое число;

- вещественное число с плавающей точкой;
- строка;
- объект;
- массив;
- ресурс (специальный тип);
- null (специальный тип).

Тип переменной можно *проверить* с помощью функции **gettype()**.

Пример 1:

```
<?php
$t = "0";
echo "t = $t -- ".gettype($t)."<br>";
$t += 2;
echo "t = $t -- ".gettype($t)."<br>";
$t = $t + 3.5;
echo "t = $t -- ".gettype($t)."<br>";
$t = 5 + "5 поросят";
echo "t = $t -- ".gettype($t)."<br>";
$t = 5.0 + "5 поросят";
echo "t = $t -- ".gettype($t);
?>
```

Результат примера 1:

```
t = 0 -- string
t = 2 -- integer
t = 5.5 -- double
t = 10 -- integer
t = 10 -- double
```

Тип переменной можно *изменить* с помощью функции **settype()**.

Пример 2:

```
<?php
$t = 3.14;
echo "t = $t -- ".gettype($t)."<br>";
settype($t,"string");
echo "t = $t -- ".gettype($t)."<br>";
settype($t,"integer");
echo "t = $t -- ".gettype($t)."<br>";
settype($t,"double");
echo "t = $t -- ".gettype($t)."<br>";
settype($t,"boolean");
echo "t = $t -- ".gettype($t);
?>
```

Результат примера 2:

```
t = 3.14 -- double
t = 3.14 -- string
t = 3 -- integer
t = 3 -- double
t = 1 -- boolean
```

Приведение типа переменной осуществляется с помощью операторов:

- (bool) - к логическому типу;
- (int) - к целому числу;
- (double) - к вещественному числу;
- (string) - к строке;
- (array) - к массиву;
- (object) - к объекту.

Отличие от изменения типа с помощью функции settype() состоит в том, что оператор приведения создает временную копию нового типа, оставляя саму переменную без изменений.

Пример 3:

```
<?php
$t = 3.14;
$tmp = (string) $t;
echo "tmp = $tmp -- ".gettype($tmp)."<br>";
$tmp = (int) $t;
echo "tmp = $tmp -- ".gettype($tmp)."<br>";
$tmp = (double) $t;
echo "tmp = $tmp -- ".gettype($tmp)."<br>";
$tmp = (bool) $t;
echo "tmp = $tmp -- ".gettype($tmp)."<br>";
echo "t = $t -- ".gettype($t);
?>
```

Результат примера 3:

```
tmp = 3.14 -- string
tmp = 3 -- integer
tmp = 3.14 -- double
tmp = 1 -- boolean
t = 3.14 -- double
```

Иногда для упрощения логики программы удобнее использовать переменные имена переменных. PHP предоставляет такую возможность в виде динамических переменных. Динамической называют переменную, имя которой хранится в ней самой.

Пример 4:

```
<?php
$t = "Всем";          // переменной t присваиваем значение "Всем"
$$t="привет!";// Переменной "Всем" присваиваем значение "привет!"
echo "$$t ".$$t."<br>";
echo "$t ${$t}<br>";
echo "$t $Всем";
?>
```

Результат примера 4:

```
$Всем привет!
Всем привет!
Всем привет!
```

В PHP возможно обращение к одной и той же переменной с использованием различных имен. Для реализации этого используются **ссылки**. Ссылки позволяют двум или большему количеству переменных ссылаться на одну и ту же область памяти.

Пример 5:

```
<?php
$t1 = 96;
$t2 = &$t1;
echo "t1 = $t1; t2 = $t2<br>";
$t1 = 315;
echo "t1 = $t1; t2 = $t2";
?>
```

Результат примера 5:

```
t1 = 96; t2 = 96
t1 = 315; t2 = 315
```

Операторы

Операторы PHP напоминают общеизвестные операторы языка Си.

Унарные операторы

-	Изменение знака на противоположный
!	Дополнение. Используется для реверсирования значения логических переменных
++	Увеличение значения переменной. Может применяться и как префикс, и как суффикс
--	Уменьшение значения переменной. Может применяться и как префикс, и как суффикс

Арифметические операторы

-	Вычитание
+	Сложение
*	Умножение
/	Деление
%	Остаток от деления

Оператор конкатенации

Оператор конкатенации "." присоединяет правую строку к левой.

Пример 6:

```
<?php
$a = "Всем";
$b = $a." привет!";
echo $b;
?>
```

Результат примера 6:

Всем привет!

Оператор конкатенации обрабатывает операнды любых типов как строки. Результат его выполнения всегда является строкой.

Операторы присваивания

=	Присваивание
+=	Сложение ($n += 777; аналогично $n = $n + 777;)
-=	Вычитание ($n -= 777; аналогично $n = $n - 777;)
*=	Умножение
/=	Деление
%=	Остаток от деления
.=	Конкатенация ($n .= "777"; аналогично $n = $n."777";)

Битовые операторы

Битовые операторы позволяют изменять отдельные биты целых чисел.

&	И
|	ИЛИ
^	Исключающее ИЛИ
~	Инверсия
>>	Сдвиг вправо
<<	Сдвиг влево

Операторы сравнения

>	Больше (Больше ли первое значение, чем второе?)
>=	Больше или равно (Верно ли, что значение-1 не меньше второго?)
<	Меньше (Меньше ли первое значение, чем второе?)
<=	Меньше или равно (Верно ли, что значение-1 не больше второго?)
==	Равно (Равнозначны ли значения двух переменных?)
===	Идентично (Одинаковы ли как значения, так и типы двух переменных?)
!= , <>	Не равно (Не равны ли значения двух переменных?)

!==	Не идентично (Не одинаковы ли значения или типы данных двух переменных?)

Логические операторы

Логические операторы отличаются от битовых тем, что работают не с числами, а с логическими значениями: TRUE и FALSE.

and	И
or	ИЛИ
xor	Исключающее ИЛИ
!	Инверсия
>>	Сдвиг вправо
<<	Сдвиг влево
&&	И
||	ИЛИ

Логические операторы "И" и "ИЛИ" имеют два формата. Это не синонимы. Дело в том, что оператор **or** имеет приоритет ниже, чем **||**, а **and** - ниже, чем **&&**. Таким образом, при построении сложных условных выражений можно обойтись без скобок. Однако, в таком способе указания порядка проще и запутаться.

Проверка содержимого переменной

Иногда необходимо проверить, существует ли переменная или какое она имеет значение. Ниже приведены функции, позволяющие выполнить такие действия.

isset($имя_переменной) #Истина, если переменная объявлена даже без присваивания значения.

empty($имя_переменной) #Истина, если значение переменной равно нулю или пустой строке, либо переменная не объявлена.

PHP также позволяет проверить тип переменной. Например, для того чтобы проверить, является ли переменная целочисленной, следует воспользоваться функцией

is_int($number) Результатом выполнения этой функции является TRUE, если переменная $number имеет тип integer. Рассмотрим подобные функции.

is_array ($var2) проверяет, является ли переменная $var2 массивом.

is_float ($number) проверяет, является ли переменная $number числом с плавающей точкой.

is_null ($varl) проверяет, равно ли значение переменной $varl нулю

is_numeric($string) проверяет, является ли переменная $string числовой строкой.

is_string ($string) проверяет, является ли переменная $string строкой.

Для проверки обратных условий следует воспользоваться символом восклицания (!). Например, при обработке следующего выражения будет получено значение TRUE, если переменная не объявлена: ! isset($имя_переменной)