Лекция 2. Основы языка PHP.

2.1. Назначение и история языка PHP

Язык PHP предназначен для написания динамических HTML-страниц. Программа на PHP встраивается в текст (код) HTML-страницы в виде одного или нескольких блоков, начинающихся строкой *<?php* и заканчивающихся строкой ?>. Схематично структуру HTML-документа со скриптом на PHP можно изобразить так:

```
<HTML>
. . .
. . .

<?php
  //Начало скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
?>

. . .
. . .

<?php
  //Продолжение скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
?>

. . .
. . .

<?php
  //Конец скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
?>

. . .
. . .
</html>
```

PHP относится к императивным языкам и удобен для обработки текстовой информации. Для этого в нём есть следующие средства:

- тип данных *строка*;
- операция *конкатенация*;
- операция *интерполяция*;
- регулярные *выражения*.

Распространено мнение, что PHP основывается на языке C. Это не совсем так. PHP разрабатывался как альтернатива языку Perl, который также предназначен для обработки текстовой информации. Оба языка относятся к языкам с контекстно зависимым типом данных, в отличие от СИ с его статическим, принудительным назначением типов. Разработчик языка *Perl* Ларри Уолл постарался использовать всё лучшее, что было накоплено в языках программирования на момент появления в 1987 году этого языка, в том числе и многие черты языка C.

К сожалению, многое в PHP не только не лучше, чем в Perl, но и хуже. Вот пример оператора foreach, который служит для перебора всех элементов массива:

```
   Perl                                  PHP
 foreach $element (@Arr)              foreach ($Arr as
$element )
 {                       }            {
}
```

При выполнении обоих операторов выполняются одни и те же действия: перебираются все элементы массива и над каждым элементом производятся операции, указанные в блоке. Оператор, написанный на Perl, читается естественно: "для каждого элемента $element массива @Arr ...". А вот как прочитать тот же оператор на PHP? Не понятно, зачем было переделывать удобный оператор. К сожалению ещё хуже обстоит дело с регулярными выражениями в PHP по сравнению с Perl.

PHP начал разрабатываться в 1994 году. Первая версия (релиз) появилась в 1995 году. Вторая - в 1997. Это были очень слабые версии. Тем не менее, PHP быстро завоевал популярность среди разработчиков, благодаря тому что его транслятор был встроен в самый распространённый веб-сервер Apache. Широко распространился набор LAMP (Linux, Apache, MySQL, PHP). Следующие версии PHP вышли:

- PHP 3.0 - 1998 г.
- PHP 4.0 - 2000 г.
- PHP 5.0 - 2004 г.
- релиз PHP 7.0 - 3.12.2015 г.

Шестая версия разрабатывалась с 2006 года, но в марте 2010 была признана бесперсперктивной. В настоящее время PHP имеет достаточно качественные трансляторы и большое количество модулей, содержащих функции для разработки веб-приложений.

2.2. Основные элементы

Алфавит языка PHP состоит из латинских букв, арабских цифр и специальных знаков.

Константы в PHP объявляются следующим образом:

```
define("имя_константы", значение);
```

Например

```
define("pi", 3.1415926);
$R = 5; //раддиус
$L = 2*pi*$R; // длина окружности
```

Переменные могут быть трёх типов:

- скаляры,
- массивы,
- объекты.

Имя переменной состоит из префикса "$" и идентификатора. Идентификатор может состоять из букв, цифр и знака подчёркивания. Идентификатор не может начинаться с цифры. Пример:

```
имя переменной $alfa
$ -    префикс
alfa - идентификатор
```

Скаляр в зависимости от контекста может принимать один из *внутренних типов*

- число с фиксированной запятой (целое число);
- число с плавающей запятой;
- строка;
- логический.

Внутренний тип логично было бы назвать подтипом, но, к сожалению, такой термин не используется.

Примеры целых скаляров.

```
$a = 1234; //десятичное число
$x = 020;  //восьмеричное = 16₁₀
$y = 0x20;  // шестнадцатеричное = 32₁₀
```

Числа с плавающей запятой имеют 13-14 значащих разрядов. Одна и та же скалярная переменная меняет внутренний тип в зависимости от контекста.

Примеры контекстной зависимости внутреннего типа

```
$a = 3; //целае скалярная переменная
$a *= 4;//целае $a=12
$a /=9; //с плавающей запятой $a=1.33 333 333 333
$a = $a.' метра'; //строковае $a="1.33333333333
метра"
//В следующем операторе $a в качестве условия
//принимает логическое значение "истина" ($a =
1),
//а при печати $a="1.33333333333 метра"
if($a)print "a=$a<BR>";
```

Несколько неожиданно, что явно объявить переменную заданного типа нельзя, а привести к заданному типу можно. Допускаются следующие приведения типов:

- (int), (integer) - приведение к целому числу;
- (bool), (boolean) - приведение к булеву типу;
- (float), (double), (real) - приведение к числу с плавающей точкой;
- (string) - приведение к строке;
- (array) - приведение к массиву;
- (object) - приведение к объекту

Примеры приведения типов

```
$x = 2.0;     //float
 //Есть специальная функция определения
встроенного типа gettype($x)
 echo "\$x==$x, тип: " . gettype($x) . "<br>\n";
//напечатается "$x==2, тип: double"
 $y = (int)$x; //integer $y=2
 $y = '3.14 - это пи'; //строковое
 $z = (float)$y; //$z=3.14
//если строка начинается не с цифры, то
преобразование в число даёт нуль
$x='red';
print 'int '.(int)$x.', float
'.(float)$x.'<BR>';//int 0, float 0
```

Примеры округления

```
$a = 2.3;  $b = 6.7;
$e = floor($a);  //Отброшена дробная часть $e=2
$f = floor($b);  //$f = 6
$e = round($a);  //арифметическое округление $e=2
$f = round($b);  //$f=7
```

```
$e = ceil($a);  //округление до ближайшего
большего или равного целого
$f = ceil($b);  // $e=3, $f=7
```

Пример форматирования переменной для печати или вставки в формируемую строку

```
$a=2/3;
print "a=$a<BR>";//a=0.666666666667
print strlen($a)."<BR>"; //14 знаков
$a=sprintf("%4.2f",$a);
print "a=$a<BR>";//a=0.67
```

Примеры присвоения переменным логических значений

```
$a=true; $b = false;//true соответсевует 1, false - 0
или "пусто"
print "a=$a b=$b <BR>";//a=1 b=    (пусто)
```

2.3. Операции над скалярами

Арифметические операции

\+ – сложение,

\- – вычитание,

* – умножение,

/ – деление,

% – деление по модулю.

Возведение в степень делается с помощь функции

```
pow(основание, показатель_степени)
print '2<sup>10<sup>='.pow(2,10);
//Напечатается 2^10=1024
```

Строковая операция - *конкатенация* (слияние двух строк) обозначается точкой.

Пример конкатенации

```
$a = 'Петер';
$b = 'бург';
$gor = $a.$b; //$gor='Петербург'
```

Одиночные и двойные кавычки. Интерполяция

В PHP, так же как и в Perl, большое внимание уделено способам формирования строк. Очень часто длинная строка составляется из коротких строк, представленых либо как значения переменных, либо в виде констант. Ниже приведён пример составления предложения из заданных слов двумя способами:

- с помощью конкатенации,
- с помощью интерполяции.

Пример

```
//Даны переменные
 $Fam = 'Петров';
 $Imja = 'Сергей';
 $Otch = 'Юрьевич';
 $den = '08';
 $mes = '11';
 $god = '1993';
 //Составим из них с помощью операции конкатенации
предложение
 //"Сергей Юрьевич Петров родился 08-11-1993."
 $R = $Imja.' '.$Otch.' '.$Fam.' родился '.$den.'-
'.$mes.'-'.$god.'.';
 //При формировании строки $R использовалось 11
операций конкатенации
 //и 12 кавычек. При таком способе очень легко
допустить ошибку.
 //Применим интерполяцию
 $R = "$Imja $Otch $Fam родился $den-$mes-$god.";
 print "$R<BR>";
 //Напечатается тот же результат
 // Сергей Юрьевич Петров родился 08-11-1993.
```

Операция интерполяции применяется только к строке, заключённой в **двойные кавычки**, и состоит в замене имён переменных их значениями. В строке, заключённой в одиночные кавычки, замена имён переменных их значениями не производится:

```
$R = '$Imja $Otch $Fam родился $den-$mes-$god.';
print "$R<BR>";
//Напечатается
//$Imja $Otch $Fam родился $den-$mes-$god.
```

Операции сравнения

Обозначение	Название	Пример	Результат
==	Равно	$a == $b	**TRUE** если $a равно $b.
===	Тождественно равно	$a === $b	**TRUE** если $a равно $b и имеет тот же тип. (Добавлено в PHP 4)
!=	Не равно	$a != $b	**TRUE** если $a не равно $b.
<>	Не равно	$a <> $b	**TRUE** если $a не равно $b.

!==	Тождественно не равно	$a !== $b	**TRUE** если $a не равно $b или в случае, если они разных типов (Добавлено в PHP 4)
<	Меньше	$a < $b	**TRUE** если $a строго меньше $b.
>	Больше	$a > $b	**TRUE** если $a строго больше $b.
<=	Меньше или равно	$a <= $b	**TRUE** если $a is меньше или равно $b.
>=	Больше или равно	$a >= $b	**TRUE** если $a больше или равно $b.

Операции === и !== свидетельствуют о том, что в PHP не до конца решена задача определения типа переменной из контекста.

Идея контекстной зависимости состоит в том, что в любом месте программы автоматически выбирается исходя из контекста тип скалярной переменной. Эта проблема решена в языке Perl, появившемся в конце 1987 года (PHP - 1995). Здесь можно посмотреть, как определяется тип скаляров в контексте операций сравнения в языке Perl.

Примеры операций сравнения.

```
$x=2.0;
$y=2;
if($x==$y) print "true<BR>";//true
if($x===$y)print "true<BR>";
else       print "false<BR>";    //false
$y='2 дня';
if($x==$y) print "true<BR>";//true
if($x===$y)print "true<BR>";
else       print "false<BR>";    //false
```

Операции сравнения и типы данных При выполнении арифметических и строковых операций тип операндов выбирается автоматически из контекста. Если операция арифметическая, то тип числовой, в строковой операции переменные рассматриваются как строки символов. Например

```
$a = '2';
$b = 3.1;
$c = $a + $b; //$a и $b - числа, результат $c = 5.1
$str = $a.$b;  //$a и $b - строки символов, результат
$str = '23.1'
```

При одной и той же операции *сравнения* операнд может принимать разные типы. Рассмотрим зависимость результатов сравнения от типов операций на примере операции " > " (больше).
1. Пусть переменные $a и $b - строковые.

```
$a = '2';
$b = '2a';
$c = $b - $a;
$L = $b > $a;
```

```
/*Результат
$c = 0      в операции сложения произошло изменение
типа на "целое"
$L = true   в операции сравнения обе переменные
считаются строковыми
*/
```

2. Переменные $a и $b разных типов. $a - целая, $b - строковая

```
$a = 2;
$b = '2a';
$c = $b - $a;
$L = $b > $a;
/*Результат
$c = 0        в операции сложения произошло изменение
типа $b на "целое"
$L = false  в операции сравнения обе переменные
считаются целыми
($a = $b = 2).
*/
```

Таким образом, чтобы предсказать результат операции сравнения, программист должен знать типы сравниваемых переменных, то есть нарушается принцип автоматического выбора типа данных в зависимости от контекста. Чтобы узнать тип переменной, нужно воспользоваться функцией *gettype()*.

```
$a=2;
$T= gettype($a); //$T =integer
$b='2a';
$T= gettype($b); //$T=string
```

Логические операции

Обозначение	Название	Пример	Результат
and	Логическое 'и'	$a and $b	TRUE если и $a, и $b TRUE.
or	Логическое 'или'	$a or $b	TRUE если или $a, или $b TRUE.
xor	Исключающее 'или'	$a xor $b	TRUE если $a, или $b TRUE, но не оба.
!	Отрицание	! $a	TRUE если $a не TRUE.
&&	Логическое 'и'	$a && $b	TRUE если и $a, и $b TRUE.
||	Логическое 'или'	$a || $b	TRUE если или $a, или $b TRUE.

Операции and, or и xor имеют приоритеты ниже, чем && и ||.

Примеры логических операций

```
$a =5; $b = 2;
```

```
$L = $a && $b; //$L=true
$L1 = $a and $b; //$L1=5
```

Инкремент и декремент

```
$a = 3;
Префиксный инкремент
$b = ++$a; //$a=4; $b=4

$a = 3;
суффиксный инкремент
$b = $a++; //$a=4; $b=3

 $a = 3;
Префиксный декремент
$b = --a; //$a=2; $b=2

$a = 3;
суффиксный декремент
$b = $a--; //$a=2; $b=3
```