Лекция 2

Комментарии, переменные, операторы, константы.

Типы данных

- Разделение инструкций.
- Комментарии.

ОСНОВНОЙ СИНТАКСИС

Изолирование от HTML

```
<p>Это будет проигнорировано PHP и отображено браузером.</p>
<?php echo 'А это будет обработано.'; ?>
<p>Это тоже будет проигнорировано PHP и отображено браузером.</p>
```

```
<?php if ($expression == true): ?>
  Это будет отображено, если выражение истинно.
<?php else: ?>
  В ином случае будет отображено это.
<?php endif; ?>
```

Продвинутое изолированием с использованием условий.

Изолирование от HTML

```
1.  <?php echo 'если вы хотите работать с документами XHTML
или XML, делайте так'; ?>

2.  <script language="php">
        echo 'некоторые редакторы (например, FrontPage) не
любят инструкции                    обработки';
    </script>

3.  <? echo 'это простейшая инструкция обработки SGML'; ?>
    <?= выражение ?> Это синоним для "<? echo выражение ?>"

4.  <% echo 'Вы можете по выбору использовать теги в стиле A
SP'; %>
    <%= $variable; # Это синоним для "<% echo . . ." %>
```

Изолирование от HTML

Замечания:

- если вы намереваетесь вставлять PHP-код в XML или XHTML, чтобы соответствовать XML стандартам, вам следует использовать форму <?php ?>;
- короткие теги доступны, только когда они включены с помощью директивы short_open_tag в конфигурационном файле **php.ini**, либо если PHP был скомпилирован с опцией --enable-short-tags ;
- теги в стиле ASP (четвертый пример) доступны, только когда они включены с помощью директивы asp_tags в конфигурационном файле **php.ini**;
- Следует избегать использования коротких тегов при разработке приложений или библиотек, предназначенных для распространения или размещения на PHP-серверах, не находящихся под вашим контролем, так как короткие теги могут не поддерживаться на целевом сервере. Для создания переносимого, совместимого кода, не используйте короткие теги;
- Начиная с PHP 5.4 короткий тег echo <?= всегда распознается и действует, несмотря на значение опции short_open_tag.

Разделение инструкций

```php
<?php
    echo 'Это тест';
?>

<?php echo 'Это тест' ?>

<?php echo 'Мы опустили последний закрывающий тег';
```

Замечания:

- Закрывающий тег PHP-блока в конце файла не является обязательным, и в некоторых случаях его опускание довольно полезно, например, при использовании include Закрывающий тег PHP-блока в конце файла не является обязательным, и в некоторых случаях его опускание довольно полезно, например, при использовании include или require.
- Это также удобно при использовании буферизации вывода, где нежелательно иметь пробелы в конце частей ответа, сгенерированного подключаемыми файлами.

Комментарии

```php
<?php
    echo "Это тест"; // Это однострочный комментарий в стиле c++
    /* Это многострочный комментарий
       еще одна строка комментария */
    echo "Это еще один тест";
    echo "Последний тест"; # Это комментарий в стиле оболочки Unix
?>
```

```
<h1>Это <?php # echo "простой";?> пример</h1>
<p>Заголовок вверху выведет 'Это пример'.</p>
```

HTML-код после ***// ... ?>*** или *# ... ?>* БУДЕТ напечатан:

?> завершает режим PHP и возвращает режим HTML,

а *//* или *#* не могут повлиять на это.

Если включена директива asp_tags, то аналогичное поведение будет с *// %>* и *# %>*.

Однако, тег ***</script>*** не завершает режим PHP в однострочном комментарии.

ПЕРЕМЕННЫЕ, КОНСТАНТЫ И ОПЕРАТОРЫ.

Переменные

- Переменные в PHP обозначаются знаком $, за которым следует ее имя.
- Правильное имя переменной должно начинаться с буквы или символа подчеркивания и состоять из букв, цифр и символов подчеркивания в любом количестве.
- Имя переменной чувствительно к регистру
- *$this* - это особая переменная, которой нельзя ничего присваивать

```
<?php
$var = 'Bob';
$Var = 'Joe';
echo "$var, $Var";      // выведет "Bob, Joe"

$4site = 'not yet';     // неверно; начинается с цифры
$_4site = 'not yet';    // верно; начинается с символа подчеркивания
$täyte = 'mansikka';    // верно; 'ä' это (Расширенный) ASCII 228.
?>
```

Присваивание по значению

```php
<?php
  // Присваиваем $first значение ' Text '
  $first = ' Text ';
  // Присваиваем $second значение переменной $first
  $second = $first;
  // Изменяем значение $first на ' New text '
  $first = ' New text ';
  // выводим значение $first
  echo "Переменная с именем first "."равна $first<br>";
  // выводим значение $second
  echo "Переменная с именем second "."равна $second";
?>
```

Результат:

Переменная с именем first равна New text
Переменная с именем second равна Text

Присваивание по ссылке

```php
<?php
$foo = 'Боб';              // Присваивает $foo значение 'Боб'
$bar = &$foo;              // Ссылка на $foo через $bar.
$bar = "Меня зовут $bar"; // Изменение $bar...
echo $bar;
echo $foo;                 // меняет и $foo.
?>
```

Результат

Меня зовут Боб
Меня зовут Боб

```php
<?php
$foo = 25;
$bar = &$foo;      // Это верное присвоение.
$bar = &(24 * 7);  // Неверно; ссылка на неименованное выражение.

function test()
{
   return 25;
}

$bar = &test();    // Неверно.
?>
```

По ссылке могут быть присвоены только именованные переменные:

Присваивание по умолчанию

В PHP и нет необходимости инициализировать переменные,
Но это считается очень хорошей практикой.

Неинициализированные переменные принимают значение по умолчанию в зависимости от их типа, который определяется из контекста их первого использования:

- булевы принимают значение **FALSE**,
- целые и числа с плавающей точкой - ноль,
- строки (например, при использовании в echo) - пустую строку,
- массивы становятся пустыми массивами.

Для обнаружения инициализации переменной используйте функцию isset().

Предопределенные переменные

- Суперглобальные переменные — это встроенные переменные, которые всегда доступны во всех областях видимости:
  - $GLOBALS — Ссылки на все переменные глобальной области видимости
    - $_SERVER — Информация о сервере и среде исполнения
    - $_GET — GET-переменные HTTP
    - $_POST — HTTP POST variables
    - $_FILES — Переменные файлов, загруженных по HTTP
    - $_REQUEST — Переменные HTTP-запроса
    - $_SESSION — Переменные сессии
    - $_ENV — Переменные окружения
    - $_COOKIE — HTTP Куки

- $php_errormsg — Предыдущее сообщение об ошибке
- $HTTP_RAW_POST_DATA — Необработанные POST-данные
- $http_response_header — Заголовки ответов HTTP
- $argc — Количество аргументов переданных скрипту
- $argv — Массив переданных скрипту аргументов

Область видимости переменной

```php
<?php
$a = 1;
include 'b.inc';
?>
```

```php
<?php
$a = 1; /* глобальная область видимости */

function test()
{
    echo $a; /* ссылка на переменную локальной области видимости */
}

test();
?>
```

Область видимости переменной

```php
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

```php
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS['b'] =
     $GLOBALS['a'] + $GLOBALS['b'];
}

Sum();
echo $b;
?>
```

Использование статических переменных

```php
<?php
function test()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

```php
<?php
function test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

```php
<?php
function foo(){
    static $int = 0;          // верно
    static $int = 1+2;        // неверно  (поскольку это выражение)
    static $int = sqrt(121);  // неверно  (поскольку это тоже выражение)

    $int++;
    echo $int;
}
?>
```

Функции работы с переменными

- boolval — возвращает двоичное значение переменной
- empty — проверяет, пуста ли переменная
- floatval — возвращает значение переменной в виде числа с плавающей точкой
- get_defined_vars — возвращает массив всех определенных переменных
- gettype — возвращает тип переменной
- intval — взвращает целое значение переменной
- is_array — определяет, является ли переменная массивом
- is_boolis_bool — проверяет, является ли переменная булевой (is_floatis_bool — проверяет, является ли переменная булевой (is_float, is_intis_bool — проверяет, является ли переменная булевой (is_float, is_int, is_numericis_bool — проверяет, является ли переменная булевой (is_float, is_int, is_numeric, is_nullis_bool — проверяет, является ли переменная булевой (is_float, is_int, is_numeric, is_null, is_scalaris_bool — проверяет, является ли переменная булевой (is_float, is_int, is_numeric, is_null, is_scalar, is_string, и т.д.)
- isset — определяет, была ли установлена переменная значением отличным от NULL
- print_r — выводит удобочитаемую информацию о переменной
- settype — присваивает переменной новый тип

Константы

```
define("Имя_константы", "Значение_константы", [Нечувствительность_к_регистру])
```

```php
<?php
  // определяем константу PASSWORD
  define("PASSWORD","qwerty");
  // определяем регистронезависимую константу PI со значением 3.14
  define("PI","3.14", True);
  // выведет значение константы PASSWORD
  echo (PASSWORD);
  // тоже выведет значение константы PASSWORD
  echo constant("PASSWORD");
  // выведет password и предупреждение,
  // поскольку мы ввели регистрозависимую константу
  echo (password);
  // выведет 3.14,
  // поскольку константа PI регистронезависима
  echo pi;
?>
```

Предопределенные константы

- __FILE__ хранит имя файла программы (и путь к нему), которая выполняется в данный момент,
- __FUNCTION__ содержит имя функции
- __CLASS__ – имя класса
- PHP_VERSION – версия интерпретатора PHP.

Полный список предопределенных констант можно получить, прочитав руководство по PHP.

- Приоритет оператора
- Арифметические операторы
- Оператор присваивания
- Побитовые операторы
- Операторы сравнения
- Оператор управления ошибками
- Операторы исполнения
- Операторы инкремента и декремента
- Логические операторы
- Строковые операторы
- Операторы, работающие с массивами
- Оператор проверки типа

ОПЕРАТОРЫ

Арифметические операторы

Пример	Название	Результат
-$a	Отрицание	Смена знака *$a*.
$a + $b	Сложение	Сумма *$a* и *$b*.
$a - $b	Вычитание	Разность *$a* и *$b*.
$a * $b	Умножение	Произведение *$a* и *$b*.
$a / $b	Деление	Частное от деления *$a* на *$b*.
$a % $b	Деление по модулю	Целочисленный остаток от деления *$a* на *$b*.

```php
<?php

echo (5 % 3)."\n";          // выводит 2
echo (5 % -3)."\n";         // выводит 2
echo (-5 % 3)."\n";         // выводит -2
echo (-5 % -3)."\n";        // выводит -2

?>
```

Строковые операторы

Обозначение	Название	
.	Конкатенация	Возвращает строку, представляющую собой соединение левого и правого аргумента
.=	Присваивание с конкатенацией	Присоединяет правый аргумент к левому

```php
<?php
$a = "Hello ";
$b = $a . "World!"; // $b теперь содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!";     // $a теперь содержит строку "Hello World!"
?>
```

Операторы присваивания

Обозначение	Название	Описание	Пример
=	Присваивание	Переменной слева от оператора будет присвоено значение, полученное в результате выполнения каких-либо операций или переменной/константы с правой стороны	$a = ($b = 4) + 5;
+=	Присваивание и сумма	Сокращение. Прибавляет к переменной число и затем присваивает ей полученное значение	$a += 5;
–=	Присваивание и разность	Сокращение. Вычитает из переменной число и затем присваивает ей полученное значение	$a –= 5;
.=	Присваивание с конкатенацией	Сокращенно обозначает комбинацию операций конкатенации и присваивания (сначала добавляется строка, потом полученная строка записывается в переменную)	$b = "Привет "; $b .= "всем";

Логические операторы

Пример	Название	Результат
$a and $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a or $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.
$a xor $b	Исключающее или	**TRUE** если *$a*, или *$b* **TRUE**, но не оба.
! $a	Отрицание	**TRUE** если *$a* не **TRUE**.
$a && $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a || $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.

Смысл двух разных вариантов для операторов "and" и "or" в том, что они работают с различными приоритетами.

Логические операторы

```php
<?php
// "||" имеет больший приоритет, чем "or"

// Результат выражения (false || true) присваивается переменной $e
// Действует как: ($e = (false || true))
$e = false || true;

// Константа false присваивается $f, а затем значение true игнорируется
// Действует как: (($f = false) or true)
$f = false or true;

var_dump($e, $f);

// "&&" имеет больший приоритет, чем "and"

// Результат выражения (true && false) присваивается переменной $g
// Действует как: ($g = (true && false))
$g = true && false;

// Константа true присваивается $h, а затем значение false игнорируется
// Действует как: (($h = true) and false)
$h = true and false;

var_dump($g, $h);
?>
```

```
bool(true)
bool(false)
bool(false)
bool(true)
```

Операторы сравнения

Обозначение	Название	Описание	Пример
==	Равенство	Значения переменных равны	$a == $b
===	Эквивалентность	Равны значения и типы переменных	$a === $b
!=	Неравенство	Значения переменных не равны	$a != $b
<>	Неравенство		$a <> $b
!==	Неэквивалентность	Переменные не эквивалентны	$a !== $b
<	Меньше		$a < $b
>	Больше		$a > $b
<=	Меньше или равно		$a <= $b
>=	Больше или равно		$a >= $b

```
<?php
  $a = 5;      // 5 как целое число (integer)
  ($a == 5);          // Сравниваются значения; Вернёт true
  ($a == '5');   // Сравниваются значения (игнорируя типы); Вернёт true
  ($a === 5);  // Сравниваются типы и значения (integer vs. integer); Вернёт true
  ($a === '5'); // Сравниваются типы и значения (integer vs. string); Вернёт false
  /* Строгое сравнение */
  if (strpos('testing', 'test')) {              // 'test' находится в 0 позиции, результатом будет 'false'
       // Ваш код... }
  if (strpos('testing', 'test') !== false) { // Результатом будет 'true', т.к. тут строгое сравнение (0 !== false)
       // Ваш код... } ?>
```

Операторы инкремента и

Обозначение	Название	Описание
++$a	Пре-инкремент	Увеличивает $a на единицу и возвращает $a
$a++	Пост-инкремент	Возвращает $a, затем увеличивает $a на единицу
--$a	Пре-декремент	Уменьшает $a на единицу и возвращает $a
$a--	Пост-декремент	Возвращает $a, затем уменьшает $a на единицу

```php
<?php
echo "<h3>Постфиксный инкремент</h3>";
$a = 5;
echo "Должно быть 5: " . $a++ . "<br />\n";
echo "Должно быть 6: " . $a . "<br />\n";

echo "<h3>Префиксный инкремент</h3>";
$a = 5;
echo "Должно быть 6: " . ++$a . "<br />\n";
echo "Должно быть 6: " . $a . "<br />\n";

echo "<h3>Постфиксный декремент</h3>";
$a = 5;
echo "Должно быть 5: " . $a-- . "<br />\n";
echo "Должно быть 4: " . $a . "<br />\n";

echo "<h3>Префиксный декремент</h3>";
$a = 5;
echo "Должно быть 4: " . --$a . "<br />\n";
echo "Должно быть 4: " . $a . "<br />\n";
?>
```

Побитовые операторы

Пример	Название	Результат
$a & $b	И	Устанавливаются только те биты, которые установлены и в $a, и в $b.
$a | $b	Или	Устанавливаются те биты, которые установлены в $a или в $b.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b, но не в обоих одновременно.
~ $a	Отрицание	Устанавливаются те биты, которые не установлены в $a, 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, работающие с массивами

Пример	Название	Результат
$a + $b	Объединение	Объединение массива *$a* и массива *$b*.
$a == $b	Равно	**TRUE** в случае, если *$a* и *$b* содержат одни и те же пары ключ/значение.
$a === $b	Тождественно равно	**TRUE** в случае, если *$a* и *$b* содержат одни и те же паты ключ/значение в том же самом порядке и того же типа.
$a != $b	Не равно	**TRUE**, если массив *$a* не равен массиву *$b*.
$a <> $b	Не равно	**TRUE**, если массив *$a* не равен массиву *$b*.
$a !== $b	Тождественно не равно	**TRUE**, если массив *$a* не равен тождественно массиву *$b*.

Операторы, работающие с массивами

```php
<?php
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry",
           "c" => "cherry");

$c = $a + $b; // Объединение $a и $b
echo "Union of \$a and \$b: \n";
var_dump($c);

$c = $b + $a; // Объединение $b и $a
echo "Union of \$b and \$a: \n";
var_dump($c);
?>
```

```
Union of $a and $b:
array(3) {
  ["a"]=>
  string(5) "apple"
  ["b"]=>
  string(6) "banana"
  ["c"]=>
  string(6) "cherry"
}
Union of $b and $a:
array(3) {
  ["a"]=>
  string(4) "pear"
  ["b"]=>
  string(10) "strawberry"
  ["c"]=>
  string(6) "cherry"
}
```