

Занятие #1.1

Темы

- Константы
- Типы данных
- Преобразование типов
- Операторы
- Функции

Константы

- **Константа** отличается от переменной тем, что, во-первых, ей нигде в программе **нельзя присвоить значение больше одного раза**, а во-вторых, ее **имя не предваряется знаком \$**, как это делается для переменных.

Константы

Предположим, определена константа PI, равная 3.1416

```
$a = 2.34 * sin(3 * PI / 8) +5;  
echo "Это число PI"; // Это число PI  
echo "Это число ".PI; // Это число  
3.1416
```

Константы



Предопределенные константы

`__FILE__`

Хранит имя файла, в котором расположен запущенный в настоящий момент код.

Пример (`echo __FILE__`)

`Z:\home\cko.t\www\les2.php`

Предопределенные константы

__LINE__

Содержит текущий номер строки, которую обрабатывает в текущий момент интерпретатор. Эта своеобразная "константа" каждый раз меняется по ходу исполнения программы.

Пример (echo **__LINE__**)

2

Предопределенные константы

`PHP_VERSION`

Версия интерпретатора PHP

Пример (`echo PHP_VERSION`)

5.3.13

Предопределенные константы

TRUE

Эта константа нам уже знакома и **содержит значение "истина"**.

FALSE

Эта константа нам уже знакома и **содержит значение "ложь"**.

Предопределенные константы

PHP_OS

Имя операционной системы, под управлением которой работает PHP.

Пример (echo PHP_OS)

WINNT

NULL

Содержит значение NULL.

Определенные константы

- Вы можете определить и свои собственные, новые константы. Делается это при помощи оператора `define()`, очень похожего на функцию.

```
void define(  
    string $name,  
    string $value,  
    bool $case_sen = true  
);
```

Определенные константы

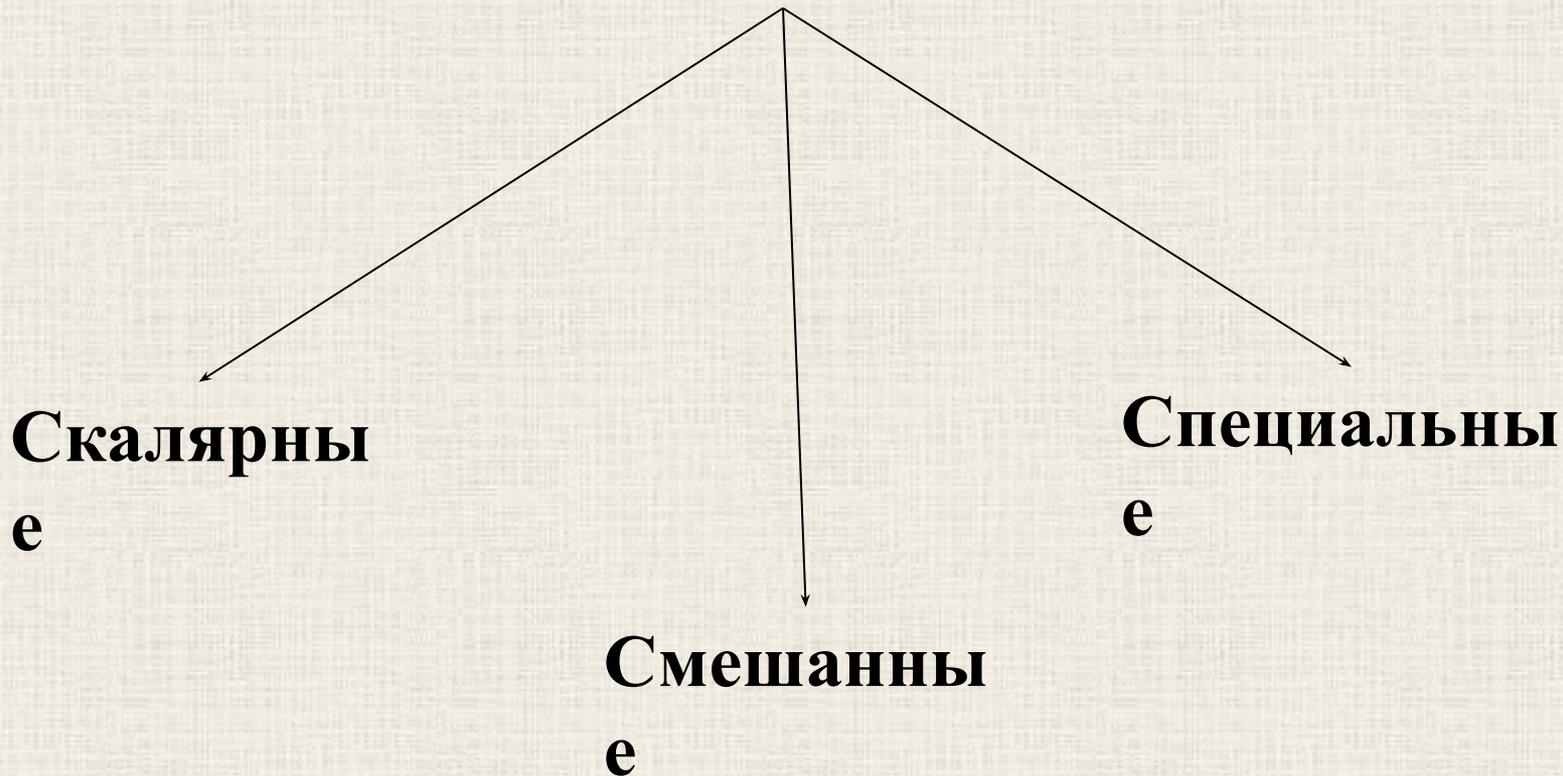
Примеры

- `define("pi", 3.14);`
- `define("str", "Test string");`
- `echo sin(pi / 4);`
- `echo str;`

Проверка существования КОНСТАНТЫ

- В PHP существует также функция, которая проверяет, существует ли (была ли определена ранее) константа с указанным именем.
- `bool defined(string $name)`
- Возвращает `true`, если константа с именем `$name` была ранее определена.

Типы данных



Скалярные типы данных

- boolean (true, false)
- integer (1, -20, 0, 1000000, ...)
- float (0.2, 12.3456, -1234.567, ...)
- string (“PHP”, “Hello WT-2”, ...)

Смешанные типы

- array
- object

Специальные типы

- resource
- NULL

Преобразование типов

Автоматическое преобразование при выполнении операций.

```
$a = 0 + "1"; // $a = 1  
if( '0' ) { ... }
```

Преобразование вручную

```
$a = (int) 3.72; // $a = 3
```

Операторы

Арифметические операторы

Арифметические операции

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 .

Операторы

Операторы присваивания

Простые

Комбинированные

Операторы

Присвоение значений переменных по ссылке (`$b = &$a`)

```
<?  
$a = 3;  
$b = &$a; // $b - это ссылка на $a  
  
print $a; // печатает 3  
print $b; // печатает 3  
  
$a = 4; // меняем $a  
  
print $a; // печатает 4  
print $b; // также печатает 4, так как $b является ссылкой на $a,  
           // а значение переменной $a успело измениться  
?>
```

Операторы сравнения

Операторы сравнения

Пример	Название	Результат
$a == b$	Равно	TRUE если a равно b после преобразования типов.
$a === b$	Тождественно равно	TRUE если a равно b и имеет тот же тип.
$a != b$	Не равно	TRUE если a не равно b после преобразования типов.
$a <> b$	Не равно	TRUE если a не равно b после преобразования типов.
$a !== b$	Тождественно не равно	TRUE если a не равно b или они разных типов.
$a < b$	Меньше	TRUE если a строго меньше b .
$a > b$	Больше	TRUE если a строго больше b .
$a <= b$	Меньше или равно	TRUE если a меньше или равно b .
$a >= b$	Больше или равно	TRUE если a больше или равно b .

Операторы инкремента и декремента

Операторы инкремента и декремента

Пример	Название	Действие
++\$a	Префиксный инкремент	Увеличивает a на единицу, затем возвращает значение a .
\$a++	Постфиксный инкремент	Возвращает значение a , затем увеличивает a на единицу.
--\$a	Префиксный декремент	Уменьшает a на единицу, затем возвращает значение a .
\$a--	Постфиксный декремент	Возвращает значение a , затем уменьшает a на единицу.

Логические операторы

Логические операторы

Пример	Название	Результат
<code>\$a and \$b</code>	И	TRUE если и <code>\$a</code> , и <code>\$b</code> TRUE.
<code>\$a or \$b</code>	Или	TRUE если или <code>\$a</code> , или <code>\$b</code> TRUE.
<code>\$a xor \$b</code>	Исключающее или	TRUE если <code>\$a</code> , или <code>\$b</code> TRUE, но не оба.
<code>! \$a</code>	Отрицание	TRUE если <code>\$a</code> не TRUE.
<code>\$a && \$b</code>	И	TRUE если и <code>\$a</code> , и <code>\$b</code> TRUE.
<code>\$a \$b</code>	Или	TRUE если или <code>\$a</code> , или <code>\$b</code> TRUE.

Строковые операторы

Конкатенация

$\$a = \text{“Привет”};$

$\$b = \text{“Иван”};$

$\$a = \$a . \$b; \Leftrightarrow \$a .= \$b;$

Строковые операторы

- Обращение к символам внутри строки
 - `$a = “Привет”;`
 - `echo $a[0];` // Выведет символ ‘П’
 - `echo $a[1];` // Выведет символ ‘р’
 - ...

Строковые операторы

Функция определения длины строки

```
int strlen ( string $string )
```

- \$a = “Привет”;
- echo strlen (\$a); // Выведет 6

Побитовые операторы

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 \wedge \b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $\$a$, либо только в $\$b$, но не в обоих одновременно.
$\sim \$a$	Отрицание	Устанавливаются те биты, которые не установлены в $\$a$, и наоборот.
$\$a \ll \b	Сдвиг влево	Все биты переменной $\$a$ сдвигаются на $\$b$ позиций влево (каждая позиция подразумевает "умножение на 2")
$\$a \gg \b	Сдвиг вправо	Все биты переменной $\$a$ сдвигаются на $\$b$ позиций вправо (каждая позиция подразумевает "деление на 2")

Операторы исполнения

<?

```
$output = `cmd`;
```

```
echo "<pre>$output</pre>";
```

?>

Microsoft Windows [Version 6.1.7601]

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 элементы.
$\$a === \b	Тождественно равно	TRUE в случае, если $\$a$ и $\$b$ содержат одни и те же элементы в том же самом порядке.
$\$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 $\$b$.

Оператор + присоединяет правый массив к массиву, размещенному слева НЕ перезаписывая элементы с дублирующимися ключами.

Оператор управления ошибками

- $@$ - оператор управления ошибками
- В случае, если $@$ предшествует какому-либо выражению в РНР-коде, любые сообщения об ошибках, генерируемые этим выражением, будут проигнорированы.

Пример

```
$a = @(5 / 0); /* Ошибка будет  
проигнорирована */
```

Порядок выполнения операторов

Ассоциативность	Оператор	Дополнительная информация
неассоциативна	clone new	clone и new
левая	[array()
правая	++ -- ~ (int) (float) (string) (array) (object) (bool) @	типы и increment/decrement
неассоциативна	instanceof	типы
правая	!	логические операторы
левая	* / %	арифметические операторы
левая	+ - .	арифметические операторы и строковые операторы
левая	<< >>	побитовые операторы
неассоциативна	< <= > >=	операторы сравнения
неассоциативна	== != === !== <>	операторы сравнения
левая	&	побитовые операторы и ссылки
левая	^	побитовые операторы
левая		побитовые операторы
левая	&&	логические операторы
левая		логические операторы
левая	? :	тернарный оператор
правая	= += -= *= /= .= %= &= = ^= <<= >>= =>	операторы присваивания
левая	and	логические операторы
левая	xor	логические операторы
левая	or	логические операторы
левая	,	множество применений

Приоритет операторов

У операторов с равным приоритетом левая ассоциативность подразумевает, что выражение вычисляется слева направо, правая ассоциативность, соответственно, подразумевает противоположный порядок.

ФУНКЦИИ

```
void var_dump ( mixed $exp [, mixed $... ] );
```

```
bool isset( mixed $var [, mixed $... ] );
```

```
string gettype( mixed $var );
```

```
bool is_{type}( mixed $var );
```

var_dump

Данная функция печатает не только значения переменных и массивов, но также и информацию об их типах.

```
array(2) {  
  [0]=>  
  int(1)  
  [1]=>  
  array(2) {  
    [0]=>  
    string(1) "a"  
    [1]=>  
    string(1) "b"  
  }  
}
```

isset

Можно проверить, существует ли (т. е. инициализирована ли) указанная переменная. Осуществляется это при помощи встроенного в PHP оператора `isset()`.

```
<?
    if (isset($myVar))
        echo "Такая переменная есть.
           Ее значение $myVar";
?>
```

gettype

```
$myVar = 50;  
echo gettype($myVar); // integer
```

is_{type}

- \$a = true;
- ...
- is_int(\$a);
- is_bool(\$a);
- ...
- is_array(\$a);

:)