PHP

Основа

Включение кода в html

- XML style

```
<?php  ?>
```

- SGML style

```
<? ?>
```

Общая информация:

- Динамическая типизация,
- Функции не чувствительны, переменные чувствительны к регистру
- Завершающая ;
- Операторные скобки { }, после закрывающей скобки ; не нужна
- Комментарии:

```
// # /**/
```

- Переменные начинаются с $
- Имена функций и классы строки без пробелов

Целые числа

- Десятичные: без первого 0, аналог long
- Восьмеричные: первый 0
- Шестнадцатеричные: 0x
- Двоичные: 0b

Строки

- "" - разименование переменной, Escape-последовательности, '' - как есть
- \ - экранирование

Булевы

- Ключевое слово false
- Целое число 0
- Число с плавающей запятой 0.0
- Пустая строка("") и строка, содержащая "0"

- Массив с О элементов
- Объект без значений и функций
- Значение NULL

Массивы

- Обычные

```
$person = array("Эдисон",  "Ванкель",  "Крэппер");
$person[O] = "Эдисон";
$person = ("Эдисон", "Ванкель", "Крэппер");
```

- Ассоциативные

```
$creator = array('Лампочку' => "Эдисон",
'Роторный двигатель'=> "Ванкель",
'Туалет'=> "Крэппер");
$creator['Роторный двигатель'] = "Ванкель";
```

Ресурсы

- Ресурс (resource) - это специальная переменная, содержащая ссылку (дескриптор) на внешний ресурс.

Сравнение и преобразование типов

- is_float(), is_string(), is_bool(), is_object(), is_resource(), is_null()
- intval() вещественное в целое
- Для преобразования строки в число первым символом д б число. Иначе 0. Будет преобразовывать, пока не повстречает логическую ошибку и на этом завершит работу и вернет значение.

Операторы

- присвоение =
- сравнение ==, <, >,

Специфика синтаксиса

Глобальные переменные:

```
define();
```

Переменные переменных

```
$foo = "bar";
$$foo = 'baz";
```

После выполнение второго оператора у переменной $bar будет значение "baz". Значение переменной foo рассматривается как имя переменной.

Переменные-ссылки

```
$black = &$white
```

Области видимости переменной:

- Локальная область: переменные, объявленные в функции, локальны для функции. Но созданная внутри цикла - все равно функция.
- Глобальная: перед переменной поставить global

```
function updateCounler()
{
  global $counter;
  $counler++;
}
$counter = 10;
updateCounter();
echo $counter;
```

В массиве $GLOBALS хранятся глобальные переменные, доступные напрямую

```
function updateCounter()
{
$GLOBALS[counter]++;
```

```
}
```

- Статическая переменная сохраняет значение между вызовами функции. Объявляется через static

isset() проверяет, существует ли переменная, unset() уничтожает переменную.

Операторы

Приор.	Оператор	Описание
21	clone, new	создание нового объекта
20	[	индекс массива
19	~	побитное отрицание
	++	инкремент
	--	декремент
	(int), (bool), (float), (string), (array), (object), (unset)	Приведение типов `$b = (int) $a;`
	@	подавление ошибок
18	isinstanceof	проверка типа
17	!	Логическое отрицание
16	* / %	Умножение, деление, остаток от деления
15	+ - .	Сложение, вычитание, конкатенация строки
14	<< >>	Побитный сдвиг влево, побитный сдвиг вправо
13	< <= > >=	Меньше, меньше или равно, больше, больше или равно
12	== != <> === !==	Равно, не равно, тип и значение равны, тип и значение не равны
11	&	Побитное и
10	^	Побитное исключающее или
9	|	Побитное или
8	&&	Логическое и
7	||	Логическое или
6	?:	Условный оператор
5	= += -= и т д	Присваивание

Приор.	Оператор	Описание
4	and	Логическое и
3	xor	Логическое исключающее или
2	or	Логическое или
1	,	разделитель списка

Префиксный и постфиксный способ записи

```
$m = ++$var;//увеличит значение на 1 и вернет значение
$m = $var++;//вернет значение и увеличит значение на 1
```

Выполнение команд оболочки

Обратные кавычки.

```
$listing =  `ls -al /tmp`;
```

Условные операторы

Стандартный синтаксис:

```
if (условие)
   {действия}
else
   {действия}
```

Тенарный синтаксис:

```
 if($active) { echo "да";} else { echo "нет";}
```

Альтернативный синтаксис:

```
<? if ($user_validated) :?>
<table>
<tr>
<td>Имя:</td><td>Sophia</td>
</tr>
<tr>
<td>Фамилия:</td><td>lee</td>
</tr>
</table>
```

```
<? else: ?>
Пожалуйста войдите.
<? endif ?>
```

Switch

```
switch ($name) {
case 'one':
    //do something
    break;
case 'two':
    //do something
    break;
}
```

```
switch ($name):
case 'one':
    //do something
    break;
case 'two':
    //do something
    break;
endswitch;
```

Циклы

```
while (условие) {
  //операторы
}
```

```
while (условие):
  //операторы
endwhile;
```

continue / break - следующая итерация / выход из текущего цикла. break 2 - выход из 2 текущих циклов.

```
do
  оператор
while (условие);
```

```
for ($counter=0; $counter<10; $counter++){
  оператор
}
```

Счетчик по массиву

```
foreach ($mass as $elem) {
  оператор
}
```

```
foreach ($mass as $key => $val) {
  оператор
}
```

Альтернативный способ:

```
foreach ($mass as $key => $val):
  оператор
endforeach;
```

Обработка ошибок

```
function inverse($x) {
    if (!$x) {
        throw new Exception('Деление на ноль.');
    }
    return 1 / $x;
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'PHP перехватил исключение: ',  $e->getMessage(), "\n";
}
```

Включение кода

include - включение статического кода

```
<?php include "header.html"; ?>
содержимое страницы
<?php include "footer.html"; ?>
```

require - включение динамического кода

```
require "codelib.php";
mysub();  //определено в codelib.php
```

Лучше:

```
<?php require "design.php";
heoder(); ?>
content
<?php footer(); ?>
```

Подавление ошибок:

```
@include
```

Если в файле конфигурации PHP (php.ini) включена опция allow_url_fopen, вы можете включать файлы, находящиеся на удаленных узлах, указав URL вместо пути к файлу

```
include "http://www.exomple.com/codelib.php";
```

include_once и require_once - однократная загрузка в пределах скрипта

Область видимости включаемых файлов = области видимости точки включения.

Функции

Синтаксис:

```
function [&] имя_функции([parameter,...]) {


  return $perem;
}
```

& - передача по ссылке

Параметры по умолчанию

```
function getPreferences($whichPreference = 'all')
```

Функция, возвращающая текст

```
<?php function column() {
 ?>
 </td><td>
 <?php }
```

Вложенные объявления:

- Не ограничивают видимость внутренней функции, которая может быть вызвана в любом месте вашей программы.
- Автоматически не получает параметры внешней функции.
- Внутренняя функция не может быть вызвана, пока не была вызвана внешняя функция,
- Нельзя вызвать из кода, обработанного после внешней функции

Функция с динамическим количеством переменных

```
function getPreferences()
```

- func_get_args() - массив всех переданных параметров функции.
- func_num_args() - количество параметров, переданных функции.
- func_get_arg($num) - параметр с определенным номером.

При отсутствии аргумента будет выведено предупреждение.

Контроль типа есть, но только относительно классов, массивов или функций. Примитивы нельзя.

Если после переменной добавить (), то будет вычислено значение переменной, произойдет поиск функции с именем равным значению и выполнение этой функции.

function_exist($fname) определяет наличие функции с данным именем

Анонимная функция (замыкание)

```
usort($array, function($a, $b){
  return strlen($a) - strlen($b)
})
```

Для передачи внутрь переменных используется use

```
usort($array, function($a, $b) use ($Someperem){
  if ($Someperem == 'random') {return rand(0,2) - 1;}
  else {
    return strlen($a) - strlen($b)
    }
})
```

Строки

Двойные кавычки разименовывают переменную. Если нет пробелов со следующим текстом, то нужно в фигурныех скобках:

```
$n = 12;
echo "You are {$n}th number";
```

Одинарные - как есть.

Формат heredoc

```
$clerihew = <<EndOIOuote
Съешь ещё этих мягких
французских булок,
да выпей чаю.
EndOIOuote;
echo $clerihew;
```

Вывод строк:

```
echo "one", "two";
print("some");
```

Есть еще printf, print_r, var_dump

Функции:

strlen($string);	длина строки
trim(), ltrim(), rtrim()	обрезание лишних пробелов слева и справа, второй необ. параметр - список символов, удаляемых вместе с пробелами
strtolower() strtoupper()	всю строку
ucfirst()	одну первую букву всей строки
ucwords()	первую букву каждого слова
htmlentities() htmlspecialchars{}	Преобразование сырой строки в html-безопасную строку
strip_tags()	Удаление html тегов

get_meta_tags()	возвращает массив META-тегов HTML страницы, указанный в виде URL или локального файла
rawurlencode() rawurldecode{)	кодирование/декодирование строки согласно URL-соглашению RFC 3986
addslashes() stripslashes()	экранировать/убрать обратный слеш перед: одинарные кавычки, двойные кавычки, NUL-байты и обратные слеши
strrev()	порядок символов на обратный
str_repeat()	Повтор строки n раз
str_pad()	заполняет одну строку другой строкой
explode(separator , string [,limit]); implode(separator, array);	разбитие / сборка

Сравнение и поиск строк:

soundex()	Сравнение степени похожести звучания (не написания) двух строк
metaphone()	Более мягкое сравнение степени похожести звучания (не написания) двух строк
similar_text()	возвращает число одинаковых символов, которые есть в двух переданных строках. В третий параметр, если он задан, заносится степень похожести двух строк, выраженная в процентах
levenshtein()	Вычисляет расстояние Левенштейна между двумя строками
substr(string, start [, length]);	
subStr_count(большая_строка, маленькая_строка)	сколько раз меньшая строка встречается в большей строке
substr_replace(original, new, start [, length]);	заменяет часть строки original начинающуюся с символа с порядковым номером start и длиной length строкой new и возвращает результат. Если не задан четвертый аргумент, substr_replace() удаляет текст, начиная с позиции start до конца строки. length в 0 для вставки без удаления "" для удаления без вставки Если start - отрицательное число, замена начинается с символа с порядковым номером start, считая от конца строки Отрицательное значение аргумента length определяет количество символов от конца строки, на котором заканчивается замена
strpos() strrpos()	первая / последняя позиция

strspn(), strcspn()	длина участка. в начале строки, полностью соответствующего маске
parse_url()	массив составляющих URL parse_url("http:/ /me:secret@example.com/ cgi-bln/board?user=fred"); ([scheme] => http [host] => example.com [user] => me [pass] =>· secrel [path] => /cgi-bln/board [query] => user=fred)

Массивы

Инициализация:

```
 $addresses = array("spam@cyberpromo.net", "abuse@example.com", "root@example.com" );
 $price = array(
'прокладка'=> 15.29,
'диск'=> 75.25,
'шина' => 50.00
);
```

```
$addresses[0] = "spam@cyberpromo.net";
$price['прокладка'] = 15.29;
```

Добавление в конец:

```
$family[] = "Павел";
```

range(1, 50); создает массив последовательных чисел или символов между двумя заданными значениями.

count($family); кол-во элементов

Многомерные массивы

```
$row0 = array( 1, 2, 3 );
$row1 = array(4,  5, 6);
$row2 = array(7, 8, 9);
$multi = array($row0, $row1, $row2);
```

Копирование нескольких значений и вырезка:

```
list ($var1, ... ) = $array;
```

Оставшиеся значения игнорируются. Если в массиве меньше - NULL.

```
$subset = array_slice(array, offset,length);
```

Если массив ассоциативный, то ключи заменяются на цифры.

```
$nums = range(1, 7);
$rows = array_chunk($nums, 3);
```

rows[0] = (1,2,3); rows[1] = (4,5,6); rows[3] = (7);

Обработка всего массива

array_sum(array) Возвращает сумму значений

array_merge(arr1, arr2, ...) Объединение массивов

array_diff(arr1, arr2) Разница

array_filter(arr, callback) Каждое значение элемента передается в callback, в возвращаемом - только элементы, для которых callback вернул true. Ключи массива сохраняются.

Использование в виде множеств

```
function arrayUnion($a, $b){
  $union = array_merge($a, $b);
  $union = array_unique($union);
  return $union;
  }
```

Стек

При помощи array_push, array_pop

Разное

array_keys(array); - массив ключей

array_key_exists(key, аггау) - проверка существования ключа в массиве

in_array(to_find, аггау [, strict]) - поиск значений

$removed = array_splice(array, start [, length [, replacement]]); - удаление/вставка элементов в массив

extroct{) - создание переменных из массива (ключи - имена)

compact() - из переменных массив

array_walk{array, callable); - применение к каждому элементу функции, третий параметр - параметр функции (для нескольких - массив)

array_search(элем, масс) - возвращает индекс найденного элемента

in_array() - есть или нет в массиве элемент

array_reverse() - обратный порядок массива

array_flip() - массив, в котором ключи стали значениями и наоборот

	По возрастанию	**По убыванию**	**Пользовательская**
Сортирует по значениям, затем обновляет индексы	sort()	rsort()	usort()
По значениям	asort()	arsort()	ausort()
По ключам	ksort()	krsort()	kusort()

Пользовательская сортировка требует предоставления функции, которая принимает два значения и возвращает значение, которое определяет порядок двух переданных значений в отсортированном массиве. Функция должна вернуть 1, если первое значение больше, чем второе, -1, если первое значение меньше, чем _второе и 0, если значения равны с точки зрения вашей функции сортировки.

Объекты

Создание класса:

```
class SimpleClass
{
    public $var = 'значение по умолчанию';
    private $privatevar = 57;
    protected $protperem = '8';
    static $mystat = 55; //изнутри обращаться через self
    static function someFunc() {
      echo "Hello!";
    }
    // Объявление метода
    public function displayVar() {
        echo $this->var;
    }
    private function displayPrivate() {
        echo $this->privatevar;
    }
}
```

Наследование:

```
class Child extends Person {}
```

Вызов методов родительского и дочернего класса:

```
parent::birthday();
self::birthday();
```

Создание объекта:

```
$newobj = new ClassName($arg1, $arg2, ...);


$clname = "ClassName";
```

```
$newobj2 = new $clname;
```

Доступ к свойствам и методам:

```
$some = $obj -> prop;
$some = $obj -> meth([]);
$some = $obj::methstat([]);
```

Клонирование объекта:

```
$b = clone $f;
```

Запрет переопределения методов

```
final function lastFunc(){}
```