PHP

заметки сайта know-online.com

10 июля 2017

Оглавление

Текст

Вывести текст

Вывести текст можно двумя командами: `echo` и `print`. Основное различие между этими командами в том, что функция `print` всегда возвращает `1`, а конструкция `echo` ничего не возвращает, она просто выводит текст.

```
echo 'PHP';  // PHP
print 'PHP'; // PHP
```

Можно использовать два вида кавычек: одиночные «'» и двойные «"». Разница между ними в том, что в двойных кавычках можно использовать переменные.

```
$lang = 'PHP';

echo "Язык: $lang"; // Язык: PHP
echo 'Язык: $lang'; // Язык: $lang
```

Чтобы вывести переменную вместе с одинарными кавычками, используют следующий код:

```
$lang = 'PHP';

echo 'Язык: '.$lang; // Язык: PHP
```

Рекомендуется использовать одинарные кавычки, т.к. это ускоряет выполнение PHP-кода (не происходит проверка на наличие переменных и специальных символов в строке).

PHP воспринимает строку как массив символов. Например:

```
$lang = 'PHP';

echo $lang[0]; // P
echo $lang[1]; // H
echo $lang[2]; // P
```

Регистр текста

Преобразовать в нижний или в верхний регистр можно через функции `mb_strtoupper()` и `mb_strtolower()`.

```
$text = 'Текст';
$text = mb_strtoupper($text, 'UTF-8'); // ТЕКСТ
$text = mb_strtolower($text, 'UTF-8'); // текст
```

Функции `strtoupper()` и `strtolower()` не рекомендуется использовать, т.к. они не понимают Юникод (т.е. не преобразует строку в указанный регистр).

Количество символов

Узнать количество символов в строке можно через функцию `mb_strlen()`.

```
mb_strlen('php', 'utf-8'); // 3
```

Если не указать `utf-8`, то функция вернёт количество байт, а не количество символов.

```
mb_strlen('»');          // 2 (хотя символ 1)
mb_strlen('»', 'utf-8'); // 1
```

Проверить наличие текста в строке

1-ый способ: поиск строки

Проверить наличие текста в абзаце можно через функцию `strpos()`, которая возвращает позицию указанного текста. Если текст не найден, то функция возвращает `-1`.

```
$text = 'PHP-script';
if (strpos($text, 'PHP') !== false) {
    echo 'строка найдена';
}
```

Также можно указать третий параметр, который указывает, откуда начинать искать строку.

2-ой способ: регулярное выражение

Проверить наличие текста с помощью регулярных выражений можно через функцию `preg_match()`.

```
$text = 'PHP-script';
if (preg_match('/PHP/', $text)) {
    echo 'строка найдена';
}
```

Если надо найти просто текст, как на примере выше, то из-за производительности лучше использовать функцию `strpos()`. Об этом также написано в официальной документации php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-notes.

Для более сложных поиска текста можно использовать `preg_match()`.

```
// Код проверяет наличие строк «PHP-script» и «Python-script»
$text = 'PHP-script';
if (preg_match('/(PHP|Python)-script/', $text)) {
    echo 'строка найдена';
}
```

Склонение во множественное число

1-ый способ: своя функция (рекомендуемый)

```php
function plural($n, $form1, $form2, $form5) {
    $n  = abs($n) % 100;
    $n1 = $n % 10;
    if ($n > 10 && $n < 20)      return $form5;
    else if ($n1 > 1 && $n1 < 5) return $form2;
    else if ($n1 == 1)           return $form1;

    return $form5;
}

$n = 3;
echo $n.' '.plural($n, 'комментарий', 'комментария', 'комментариев');
// $n == 3: 3 комментария
// $n == 101: 101 комментарий
// $n == 14: 14 комментариев
```

2-ой способ: ngettext()

```php
ngettext('комментарий', 'комментариев', $n);
// $n == 3: 3 комментариев
// $n == 101: 101 комментариев
// $n == 14: 14 комментариев
```

Закодировать строку как пароль

Пароли не следует хранить в открытом виде. Даже в базе данных.

1-ый способ: password_hash()

В PHP 5.5.0 появилась новая функция для генерации пароля — `password_hash()`.

```
$password = password_hash('пароль', PASSWORD_DEFAULT);
echo $password; // $2y$10$hAZy/dJkbOo6VwqJphP7A.Z1qfFbTBgGZMzRUe1gSlOYmhFyeVlQW
```

Проверить пароль можно через функцию `password_verify()`.

```
$user_password_from_db = '$2y$10$hAZy/dJkbOo6VwqJphP7A.Z1qfFbTBgGZMzRUe1gSlOYmhFyeVlQW';

if (password_verify('пароль', $user_password_from_db)) {
    echo 'Пароль введён правильно';
}
```

2-ой способ: crypt()

Рекомендуется использовать на сайтах, на которых PHP установлен меньше версии PHP 5.5.

```
$user_password = 'password';
$salt = '$1$adsyu&823ji$';

crypt($user_password, $salt); // $1$adsyu&82$cG6Jx1S.y69G7PZoVY4cP/
```

3-ий способ: md5()

Данный способ указан в качестве примера, использовать его не рекомендуется. Об этом также указано в официальной документации http://php.net/manual/ru/function.md5.php#refsect1-function.md5-notes.

```
md5('password'); // 5f4dcc3b5aa765d61d8327deb882cf99
```

Повторить строку несколько раз

Повторить строку несколько раз можно через функцию `str_repeat()`.

```
str_repeat('PHP ', 3); // PHP PHP PHP
```

Удаление символов

Удалить HTML

Удалить HTML можно через функцию `strip_tags()`, который также удаляет PHP-код в переданной строке.

```
strip_tags('<p>Пример кода</p>'); // Пример кода
strip_tags('<p>Пример <a href="//example.com/">кода</a></p>'); // Пример кода
strip_tags('<p>Пример <?="PHP"?> кода</p>'); // Пример  кода
```

Во втором параметре можно указать разрешённые теги:

```
strip_tags('<p>Пример <a href="//example.com/">кода</a></p>', '<p>'); // <p>Пример
кода</p>
```

Удалить указанные символы

1-ый способ

Удалить указанные символы можно через функцию `str_replace()`.

```
$chars = ['!','#']; // символы для удаления
str_replace($chars, '', 'PHP! #код#'); // PHP код
```

2-ой способ: в начале и в конце строки

Удалить указанные символы в начале и в конце строки можно через функцию `trim()`.

```
# удаление символов в начале и в конце строки

trim('#код#', '#');             // код
trim('/index.html', '/');       // index.html
trim('/page/about.html', '/'); // page/about.html
```

Также есть функция `ltrim()`, которая удаляет указанные символы в начале строки, и функция `rtrim()`, которая удаляет указанные символы в конце строки.

3-ий способ: регулярные выражения

Удалить символы через регулярные выражения можно через функцию `preg_replace()`.

```
$chars = '!#'; // символы для удаления
echo preg_replace('/['.$chars.']/', '', 'PHP! #код#'); // PHP код
```

Удалить строку до/после указанного символа

Удалить текст, который идёт до указанного символа можно через функцию `strstr()`.

```
$text = 'This is PHP script';
strstr($text, 'P'); // PHP script
```

Удалить текст, который идёт после указанного символа можно также через функцию `strstr()`, но добавив третий параметр `true`.

```
$text = 'This is PHP script';
strstr($text, 'P', true); // This is
```

Вариант через регулярное выражение (указан в качестве примера, рекомендуется использовать `strstr()`):

```
$text = 'This is PHP script';
preg_replace('/(?>P).*/', '$1', $text); // This is
```

Числа

Случайное число

Получить случайное число можно через функцию `mt_rand()`.

```
mt_rand(0, 100);
```

Функция `mt_rand()` работает в 4 раза быстрее, чем функция `rand()`.

Функция `mt_rand()` использует алгоритм «Вихрь Мерсенна».

Форматирование чисел (число с разделителями)

1-ый способ:

Форматировать число с разделителями можно через функцию `number_format()`.

```
// по-русски
number_format(1000000, 0, ',', ' ');  // 1 000 000
number_format(1000000, 2, ',', ' ');  // 1 000 000,00

// по-английски
number_format(1000000);     // 1,000,000
number_format(1000000, 2);  // 1,000,000.00
```

2-ой способ

Также можно использовать объект `NumberFormatter`, которая входит в библиотеку **intl**. Если её нет в PHP, то её надо установить командой (в Linux):

```
sudo apt-get install php5-intl
```

Объект `NumberFormatter` позволяет автоматически форматировать числа, в зависимости от указанной страны.

Пример использования:

```
# по-русски
$formatter = new NumberFormatter('ru_RU', NumberFormatter::DECIMAL);
echo $formatter->format(1234567.890); // 1 234 567,89

# по-английски
$formatter = new NumberFormatter('en_US', NumberFormatter::DECIMAL);
echo $formatter->format(1234567.890); // 1,234,567.89
```

Указать длину числа

Указать длину числа можно через функцию `str_pad()`.

```
str_pad(1,    3, 0,   STR_PAD_LEFT); // 001
str_pad(1,    3, '-', STR_PAD_LEFT); // --1
str_pad(10,   3, 0,   STR_PAD_LEFT); // 010
str_pad(100,  3, 0,   STR_PAD_LEFT); // 100
str_pad(1000, 3, 0,   STR_PAD_LEFT); // 1000
```

Возвести число в степень

Возвести число в степень можно через функцию `pow()`.

```
pow(2, 2); // 4
pow(2, 3); // 8
pow(2, 4); // 16
```

С версии PHP 5.6 можно использовать оператор `**`.

```
echo 2 ** 2; // 4
echo 2 ** 3; // 8
echo 2 ** 4; // 16
```

Массивы

Создание массива

Массив — переменная, которая может содержать несколько значений, доступные по индексу.

```php
$array = array('значение 1', 'значение 2');
echo $array['0']; // значение 1
```

С версии PHP 5.4 доступен сокращённый способ создания массива:

```php
$array = ['значение 1', 'значение 2'];
echo $array['0']; // значение 1
```

Ассоциативные массивы

Ассоциативные массивы — массивы, в которых индексы являются текстовыми, а не числовыми.

```php
$php = [
    'name' => 'php',
    'version' => 5.6,
];

// или
$php = [];
$php['name'] = 'php';
$php['version'] = 5.6;

echo $php['version']; // 5.6
```

Многомерный массив

Многомерный массив — массивы, в которых содержатся другие массивы.

```php
$lang = array(
    'php' => [],
    'python' => [],
);
```

Пример использования:

```php
$lang = array(
    'php' => [
        'name' => 'PHP',
        'version' => '5.6',
    ],
    'python' => [
        'name' => 'Python',
        'version' => '2.7',
    ],
);

echo $lang['php']['version']; // 5.6
echo $lang['python']['version']; // 2.7
```

Другое

Каждое значение в массиве называется **элемент массива**. Каждый элемент массива доступен через **индекс** (также называют **ключ**), который может быть числовым или строчным значением.

Ключи массива регистрозависимы. То есть `$php['name']` и `$php['NAME']` это два разных элемента массива.

Массивы нельзя сохранять в базу данных или в отдельные файлы как строка. Но можно сериализовать массив через функцию `serialize()`, и полученную строку хранить, например, в базе данных.

```php
$array = ['php', '5.6'];

serialize($array); // a:2:{i:0;s:3:"php";i:1;s:3:"5.6";}
unserialize('a:2:{i:0;s:3:"php";i:1;s:3:"5.6";}'); // ['php', '5.6']
```

Добавить элемент в массив

```php
# 1-й способ
$name[] = 'PHP';

# 2-ой способ (добавить элемент в конец массива)
array_push($name, 'PHP');

# 3-ий способ (добавить элемент в начало массива)
array_unshift($name, 'PHP');
```

Стоит отметить, что третий способ работает медленнее, чем первые два. Первые два способа просто добавляют элемент в конец массива, а для функции `array_unshift()` приходится ещё сдвигать индексы элементов.

`array_push()` и `array_unshift()` могут добавлять несколько элементов в массив, перечисленные через запятую.

```php
array_push($name, 'mysql', 'zend');
```

Выполнить функцию для каждого элемента массива

1-ый способ

Выполнить функцию для каждого элемента в массиве можно через функцию `array_walk()`.

```php
$array = array('php', 'mysql');

# преобразование элементов массива
function myFunc(&$arr) {
    $arr = '<strong>'.$arr.'</strong>';
}

array_walk($array, 'myFunc');
```

2-ой способ

Также можно использовать цикл `foreach` через символ `&`.

```php
$array = [2, 4, 6];
foreach ($array as &$arr) {
    $arr *= 2;
}

// array(4, 8, 12)
```

Разбить строку на массив

1-ый способ

Разбить строку на массив можно через функцию `explode()`.

```
$text = 'PHP MySQL Apache';
$array = explode(' ', $text);

$array; // Array([0] => PHP [1] => MySQL [2] => Apache)
```

2-ой способ: регулярные выражения

Разбить строку на массив через регулярные выражения можно через функцию `preg_split()`.

```
$text = 'PHP MySQL Apache';
$array = preg_split('/\s/', $text);

$array; // Array([0] => PHP [1] => MySQL [2] => Apache)
```

3-ий способ

Можно каждый символ строки разбить в массив через функцию `str_split()`.

```
$text = 'PHP';
$array = str_split($text);

$array; // Array([0] => P [1] => H [2] => P)
```

Сложить все числа в массиве

Сложить все числа в массиве можно через функцию `array_sum()`.

```
$chars = array(1,2,3,4,5);
array_sum($chars); // 15
```

Вывод элементов массива

Вывести элементы массива

1-ый способ

Вывести элементы массива можно через цикл `foreach()`.

```
$arrays = ['PHP', 'MySQL'];
$html = '';

foreach ($arrays as $array) {
    $html .= $array.', ';
}

echo $html; // PHP, MySQL
```

Стоит отметить, что вывод элементов массива происходит не внутри цикла, а внешне. Внутри цикла формируется только переменная с результатом. Это сделано для производительности, т.к. добавление элемента массива в переменную работает быстрее, чем сразу её вывод в каждой итерации цикла.

Если надо вывести элементы массива через запятую, как указано на примере выше, то можно сократить код:

```
$array = ['PHP', 'MySQL'];
echo implode(', ', $array); // PHP, MySQL
```

Функция `implode()` выводит элементы массива через указанный разделитель, в данном примере через разделитель «, ».

2-ой способ: Отладка

Когда надо временно вывести элементы массива, чтобы просмотреть её значения, то можно использовать функцию `print_r()` или `var_dump()`.

Также для этой цели можно установить **Xdebug**, который будет подсвечивать `var_dump()`, что визуально будет удобно для отладки массивов.

Вывести последний элемент массива

Вывести последний элемент массива можно через функцию `end()`.

```
$array = ['PHP', 'MySQL'];
end($array); // MySQL
```

Минимальное и максимальное значение элемента массива

Минимальное и максимальное значение элемента массивы можно узнать через функции `max()` и `min()`.

```
$arr = array(2,4,3,10,3);
max($arr); // 10
min($arr); // 2
```

Получить из массива только ИД

С версии PHP 5.5.0

В PHP 5.5.0 появилась функция `array_column()`, которая возвращает нужные данные из массива, например ИД.

```
$posts = [
    ['id' => 15, 'title' => 'Преимущества PHP'],
    ['id' => 23, 'title' => 'Разработка на PHP'],
    ['id' => 48, 'title' => 'Оптимизация PHP'],
];

$ids = array_column($posts, 'id');
$ids; // Array ( [0] => 15 [1] => 23 [2] => 48 )
```

До версии PHP 5.5.0

Можно через функцию `array_map()` «пройтись» по каждому элементу массива и вернуть нужный элемент.

```
$posts = [
    ['id' => 15, 'title' => 'Преимущества PHP'],
    ['id' => 23, 'title' => 'Разработка на PHP'],
    ['id' => 48, 'title' => 'Оптимизация PHP'],
];

$ids = array_map(function($el){return $el['id'];}, $posts);
$ids; // Array ( [0] => 15 [1] => 23 [2] => 48 )
```

Удаление элементов в массиве

Удалить элемент в массив

```
# 1-ый способ (удаление последнего элемента в массиве)
array_pop($array);

# 2-ой способ (удаление первого элемента в массиве)
array_shift($array);
```

Стоит отметить, что первый способ быстрее второго, т.к. функция `array_pop()` просто удаляет элемент в массиве, а функция `array_shift()`кроме удаления массива ещё сдвигает индексы элементов.

Удалить указанный элемент массива можно через функцию `unset()`.

```
$array = ['PHP', 'Zend', 'MySQL'];
unset($array[1]);

/* Array
(
  [0] => PHP
  [2] => MySQL
)
*/
```

Здесь стоит обратить внимание, что `unset()` только удаляет элемент массива, а индексы остаются прежними, как видно на примере выше. Сдвинуть индексы массива можно через функцию `array_values()`.

```
$array = ['PHP', 'Zend', 'MySQL'];
unset($array[1]);

// сдвинуть индексы
$array = array_values($array);

/* Array
(
  [0] => PHP
  [1] => MySQL
)
*/
```

Удалить диапозон элементов в массиве

Удалить указанный диапазон элементов в массиве можно через функцию `array_slice()`.

```
$array = array('PHP', 'MySQL', 'ZF', 'Apache', 'www');
array_slice($array, 0, 2); // ['PHP', 'MySQL']
array_slice($array, 0, 3); // ['PHP', 'MySQL', 'ZF']
array_slice($array, 1);    // ['MySQL', 'ZF', 'Apache', 'www']
```

Удалить дубликаты в массиве

Дубликаты в массиве удаляются через функцию `array_unique()`.

```
$array = array('PHP','MySQL','PHP','Zend');
$array = array_unique($array);

/* Array([0] => PHP [1] => MySQL [3] => Zend) */
```

Стоит обратить внимание, что `array_unique()` удаляет только дубликаты в массиве, а индексы остаются прежними, как видно на примере выше. Сдвинуть индексы массива можно через функцию `array_values()`.

```php
$array = array('PHP','MySQL','PHP','Zend');
$array = array_unique($array);

// сдвинуть индексы
$array = array_values($array);

/* Array([0] => PHP [1] => MySQL [2] => Zend) */
```

Функция `array_unique()` регистрозависима. То есть для `array_unique()` элементы «php» и «PHP» это два разных элемента.

Удалить пустые элементы в массиве

1-ый способ

Удалить пустые элементы в массиве можно через функцию `array_diff()`.

```php
$fruits = [
    'PHP',
    'MySQL',
    '',
    'Apache',
    null
];

array_diff($fruits, array('')); // Array( [0] => 'PHP', [1] => 'MySQL',[3] => 'Apache')
```

2-ой способ

Также удалить пустые элементы в массиве можно через функцию `array_filter()`.

```php
$fruits = [
    'PHP',
    'MySQL',
    '',
    'Apache',
    null
];

array_filter($fruits); // Array( [0] => 'PHP', [1] => 'MySQL',[3] => 'Apache')
```

Удалить указанный элемент в массиве

Удалить указанный элемент в массиве можно через функцию `array_diff()`.

```php
$apps = [
    'PHP',
    'MySQL',
    'Apache'
];

array_diff($apps, ['PHP']);
// Array( [1] => 'MySQL', [2] => 'Apache' )

array_diff($apps, ['PHP', 'MySQL']);
// Array( [2] => 'Apache' )
```

Если надо удалить элемент массива по индексу, то можно использовать следующий код:

```php
unset($array[1]);
```

Удалить элементы в массиве, значения которых меньше 5

Вернуть числа из массива, значения которых больше 5, можно через функцию `array_filter()`.

```php
$array = [1,7,34,5,2,12,6,4];
$array_filtered = array_filter($array, function ($v) { return $v > 5; });

$array_filtered; // Array( [1] => 7 [2] => 34 [5] => 12 [6] => 6 )
```

Сортировка массивов

Сортировать элементы в массиве

Сортировка элементов массива происходит через функцию `sort()`.

```
# строка
$array = ['PHP', 'MySQL', 'Zend'];
sort($array); // Array([0] => MySQL [1] => PHP [2] => Zend)

# числа
$arr = [1,5,4,7,43,21,5];
sort($arr, SORT_NUMERIC);
```

Функция `sort()` заново проставляет индексы, даже для ассоциативных массивов. Чтобы сортировать элементы массива вместе с индексами, надо использовать функцию `asort()`.

```
$array = [
    'name' => 'PHP',
    'version' => 5.6,
    'developer' => 'Zend',
];

sort($array);  // Array([0] => PHP [1] => Zend [2] => 5.6)
asort($array); // Array([name] => PHP [developer] => Zend [version] => 5.6)
```

Для сортировки элементов массива в обратном порядке используют функцию `array_reverse()`.

```
$array = ['PHP', 'MySQL', 'Zend'];
$array = array_reverse($array); // Array([0] => Zend [1] => MySQL [2] => PHP)
```

URL

Свойства URL

```
# https://know-online.com/php/url/svoistva-url?query=example

$_SERVER['SERVER_PROTOCOL']; // HTTP/1.1
$_SERVER['HTTP_HOST'];       // know-online.com
$_SERVER['SERVER_NAME'];     // know-online.com
$_SERVER['SERVER_PORT'];     // 80
$_SERVER['REQUEST_URI'];     // /php/url/svoistva-url?query=example
$_SERVER['QUERY_STRING'];    // query=example
$_SERVER['HTTPS'];           // on (появляется только на протоколе https)
```

Через PHP нельзя определить значение хеша (символа `#`).

Например http://know-online.com/**#php**.

Закодировать ссылку

Закодировать ссылку можно через функцию `urldecode()`.

```
urlencode('http://example.com/page/страница'); //
http%3A%2F%2Fmysite.com%2Fpage%2F%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0

urldecode('http%3A%2F%2Fexample.com%2Fpage%2F%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0
%B0'); // http://mysite.com/page/страница
```

Функция `urlencode()` не экранирует символы `-_.`. Пробел преобразуется в `+`.

Узнать IP сайта

Узнать IP сайта (домена) через функцию `gethostbyname()`.

```
gethostbyname('example.com'); // 37.140.192.183
```

Отправить POST-запрос

Отправить POST-запрос можно через cURL.

```php
// переменные для POST-запроса
$post = array(
    'lang' => 'PHP',
    'vers' => '5.4'
);

// Отправка POST-запроса
$ch = curl_init('http://example.com/');
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$data = curl_exec($ch);
```

В переменной `$data` содержится контент указанной страницы, загруженная с указанными POST-запросами.

Параметр `CURLOPT_RETURNTRANSFER` указывает, что ответ надо возвращать в переменную, а не выводить на странице.

Переменные

Присвоение переменных

Переменная в PHP начинается со знака `$`.

Обычное присвоение:

```
$a = 10;
$a; // 10
```

Множественное присвоение:

```
$a = $b = $c = 10;
$a; // 10
$b; // 10
$c; // 10
```

Тернарный оператор:

```
$a = (10 > 5) ? 1 : 0;
$a; // 1
```

Имена переменных регистрозависимы. Т.е. переменные `$var` и `$Var`, это две разные переменные.

Для каждой переменной выделяется место в оперативной памяти. Если переменная уже не нужна, её рекомендуется удалять через функцию `unset()`, чтобы освободить память для других операций.

```
unset($var_name);
```

Область видимости переменных

Область видимости переменных бывают двух видов: глобальные и локальные.

Глобальная переменная доступна в любом месте кода, а локальная только внутри функции, в которой она была определена.

```
$global_var = 'Глобальная переменная';

function custom() {
    $local_var = 'Локальная переменная';
}
custom();

echo $global_var; // Глобальная переменная
echo $local_var;  // Notice (8): Undefined variable: local_var
```

В коде выше, переменная `$local_var` доступна только внутри функции `custom()`. При этом, если уже существует переменная `$local_var`, то переменная `$local_var` внутри функции не переписывает существующую переменную.

```
$global_var = 'Глобальная переменная';
$local_var  = 'Глобальная переменная 2';

function custom() {
    $local_var = 'Локальная переменная';
    echo $local_var; // Локальная переменная
}
custom();

echo $global_var; // Глобальная переменная
echo $local_var;  // Глобальная переменная 2
```

Валидация переменных

Ссылка

Проверить на валидность ссылку можно через параметр `FILTER_VALIDATE_URL`.

```
$url = 'http://know-online.com/';
if (filter_var($url, FILTER_VALIDATE_URL)) {
    echo 'ссылка введена верно';
}
```

Дополнительные флаги:

- `FILTER_FLAG_PATH_REQUIRED` — в URL должна быть указана любая страница;
- `FILTER_FLAG_QUERY_REQUIRED` — в URL должна быть указана строка запроса.

`FILTER_VALIDATE_URL` не поддерживает Юникод в URL.

```
'http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0' // true
'http://ru.wikipedia.org/wiki/Программа' // false
```

Email

Проверить на валидность email-адреса можно через параметр `FILTER_VALIDATE_EMAIL`.

```
$email = 'mail@know-online.com';
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo 'email введён верно';
}
```

Число

Проверить на валидность числа можно через параметр `FILTER_VALIDATE_INT`.

```
$int = '55';
$config = array(
    'options' => array(
        'min_range' => 50,  // минимальное значение
        'max_range' => 100, // максимальное значение
    ),
);
if (filter_var($int, FILTER_VALIDATE_INT, $config)) {
    echo 'число введено верно';
}
```

IP-адрес

Проверить на валидность IP-адрес можно через параметр `FILTER_VALIDATE_IP`.

```
$ip = '127.0.0.1';
if (filter_var($ip, FILTER_VALIDATE_IP)) {
    echo 'ip введён верно';
}
```

Пространство имён (namespaces)

Пространство имён — механизм, позволяющий избежать конфликт между различными библиотеками.

Пространство имён создаётся через ключевое слово `namespace`. Весь последующий код будет относиться к только что созданному пространству имён.

```
namespace MyProject;

function foo(){}
foo(); // выполняется как функция MyProject\foo()
```

Можно обращаться к функции напрямую.

```
MyProject\foo();
```

Пространство имён должно создаваться в самом начале кода, т.е. весь код, в том числе HTML и даже пробелы, должны быть после вызова ключевого слова `namespace`. Единственное выражение, которое можно использовать до создания пространства имён, это `declare()`.

Вывести текущее пространство имени можно через константу `__NAMESPACE__`.

Полностью совместимые пространство имён и класс должны иметь следующую структуру `\<Наименование производителя>\(<Пространство имен>\)*<Название класса>`, например `\Symfony\Core\Request`.

Пространство имён появилось в PHP 5.3.

Как устроены переменные

Переменная PHP хранится в контейнере, называемом **zval** (сокращённо от «Zend value»).

Контейнер zval, помимо типа и значения переменной, также содержит два дополнительных элемента.

1. **is_ref** — указывает, является ли переменная частью «набора ссылок» или нет;
2. **refcount** — содержит количество имён переменных (символов), которые указывают на данный контейнер zval.

Для вывода информации о переменной используется функция `xdebug_debug_zval()`, для которой предварительно надо установить Xdebug.

```
$lang = 'PHP';
xdebug_debug_zval('lang'); // (refcount=1, is_ref=0), string 'PHP' (length=3)
```

Есть также функция `debug_zval_dump()`, но её не рекомендуется использовать, т.к. через неё сложно отслеживать значение refcount.

Благодаря свойству «is_ref», PHP знает как отличать обычные переменные от ссылок.

База данных

Подключение к базе данных

Подключение к базе данных происходит через метод `mysqli()`. Буква `i` означает «improved» (рус. улучшенный).

```
$mysqli = new mysqli('localhost', 'user', 'password', 'db_table');

if ($mysqli->connect_error) {
    exit('Не удалось подключиться. Ошибка №'.$mysqli->connect_errno.': '.$mysqli->connect_error);
}
```

Функция `mysql_query()` является устаревшей с версии PHP 5.5.0.

Выборка из базы данных

Выполнение SQL-кода происходит через метод `query()`.

```
$result = $mysqli->query('SELECT title,content FROM articles WHERE id = 10');

while ($article = $result->fetch_assoc()) {
    echo $article['title'];
    echo $article['content'];
}
$mysqli->free(); // рекомендуется использовать: освобождает память MySQL (не PHP)
```

Не рекомендуется делать выборку со всех полей (через символ `*`), т.к. это может уменьшить загрузку страниц из-за выборки лишних данных.

Нельзя передавать значения напрямую из `$_GET`, так как злоумышленник может получить контроль над базой данных, например удалить всю таблицу.

```
// так делать нельзя в целях безопасности сайта
$login = $_GET['login'];
$password = $_GET['password'];

$result = $mysqli->query('SELECT id,name,login FROM users WHERE login='.$login.' AND password='.$password);
```

В примере выше, посетитель вместо пароля может ввести `OR login='admin'`, и авторизуется под админом (данный вид авторизации называется SQL-инъекция).

Если надо передать значение из `$_GET`, то надо применить метод `real_escape_string()`.

```
// правильный и безопасный способ передачи параметров в SQL-запрос
$login = $mysqli->real_escape_string($_GET['login']);
$password = $mysqli->real_escape_string($_GET['password']);

$result = $mysqli->query('SELECT id,name,login FROM users WHERE login="'.$login.'" AND
password="'.$password.'"');
```

Количество выбранных записей из базы данных

Узнать количество выбранных записей из базы данных можно через свойство `num_rows`.

```
$result = $db->query('SELECT id FROM post');
$result->num_rows; // 24
```

Если при запросе в базу данных, надо узнать только количество записей, то рекомендуется использовать следующий код:

```
$result = $db->query('SELECT COUNT(id) FROM post');
$num_rows = $result->fetch_array();

echo $num_rows[0]; // 24
```

Если в базе данных миллионы записей, то в первом случае будет выбрана каждая запись, что сильно нагрузит веб-сервер. Второй код работает намного быстрее, т.к. не выбирает записи, а просто возвращает количество данных в указанной таблице.

Алгоритм постраничной навигации

```
$display_page = 10; // количество постов на одной странице

$res_qpost = $mysqli->query('SELECT COUNT(id) FROM post'); // количество выбранных записей
$qpost     = $res_qpost->fetch_array();
$qpagenav  = intval(($qpost[0] - 1) / $display_page) + 1;

// вывод постраничной навигации
for ($i = 1; $i <= $qpagenav; $i++) {
    echo '<span><a href="?page='.$i.'">'.$i.'</a></span>';
}
```

Выборка пользователей, которым 20 лет

Запрос ниже выбирает пользователей, которым 20 лет. Выборка происходит по полю с типом «Date».

```
$age = '20';

$date1 = date((date('Y') - $age - 1) . '-m-d');
$date2 = date((date('Y') - $age) . '-m-d');

$res = $mysqli->query('SELECT name FROM users WHERE date BETWEEN "'.$date1.'" AND
"'.$date2.'"');

while ($user = $mysqli->fetch_assoc($res)) {
    echo $user['name'];
}
```

Выборка пользователей, которым от 20 до 25 лет.

```
$age1 = 20;
$age2 = 25;

$date1 = date((date('Y') - $age2 - 1 ) . '-m-d');
$date2 = date((date('Y') - $age1) . '-m-d');

$res = $mysqli->query('SELECT name FROM users WHERE date BETWEEN "'.$date1.'" AND
"'.$date2.'"');
while ($user = $mysqli->fetch_assoc($res)) {
    echo $user['name'];
}
```

PDO

PDO позволяет работать с различными базами данных, использую один и тотже API. Например, при смене базы данных с MySQL на PostgreSQL, надо будет просто указать данные новой базы данных, не меняя сам код под новую базу данных.

```php
// Настройки подключения к базе данных
$dbc = [];
$dbc['user']     = 'user';
$dbc['password'] = 'pass';
$dbc['host']     = 'localhost';
$dbc['table']    = 'my_database';
$dbc['dsn']      = 'mysql:host='.$dbc['host'].';dbname='.$dbc['table'];

// Подключение через PDO
$db = new PDO($dbc['dsn'], $dbc['user'], $dbc['password']);
unset($dbc);

// Выполнение SQL-запроса
$rows = $db->query('SELECT data FROM dbtable');

while ($row = $rows->fetch(PDO::FETCH_ASSOC)) {
    echo $data;
}

// освобождением памяти
$rows->closeCursor();
unset($db);
```

ООП

ООП

Объекты в ООП состоят из **классов** (англ. class). Классы содержат **свойства** (переменные класса) и **методы** (функции класса).

Пример создания класса:

```php
class Lang
{
    // свойство класса lang
    public $verison;

    // метод show_version
    function show_version() {
        return $this->version;
    }
};

$php = new Lang();
$php->version = '5.3';
echo $php->show_version; // 5.3
```

Наследование

Наследовать класс можно через оператор `extends`.

```php
class myObj
{
    public function getPHPVersion(){
        return PHP_VERSION;
    }
};


class mySecondObj extends myObj {  }


$myobj = new mySecondObj();
echo $myobj->getPHPVersion(); // 5.4.3
```

В примере выше видно, что для объекта `mySecondObj` не было создано ни свойств ни методов. Но при этом он может использовать свойства и методы объекта `myObj`. Это и есть механизм «наследования».

Наследовать можно только один класс.

Запретить наследовать класс

Чтобы запретить наследовать класс, его надо объявить модификатором `final`.

```
final class myObj {}
```

Вызвать метод класса без создания экземпляра

Вызвать метод класса без создания экземпляра можно через модификатор `static`.

```
class myFunc
{
    public static function showPHPVersion() {
        return PHP_VERSION;
    }
};

echo myFunc::showPHPVersion(); // 5.3.4
```

В статических методах нельзя использовать свойство `$this`. Вместо свойство `$this` можно использовать следующее:

```
$t = new self();
return $t->vars;
```

Интерфейс

Интерфейс указывает классу, какой функционал он должен реализовать.

Интерфейс не может содержать код. Он может определять только имена методов. Наследуемые классы должны иметь теже методы, которые были объявлены в интерфейсе, иначе произойдёт ошибка.

Пример интерфейса:

```php
interface myTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}

class Template implements myTemplate
{
    public function setVariable($name, $var) {
        // код
    }

    public function getHtml($template) {
        // код
    }
}
```

Интерфейс указывает, какие методы и свойства должен включать класс. Т.е. если бы класс `Template` не содержал бы функцию `getHtml`, то отобразилась бы ошибка.

Вызов объекта как текст

Когда объект вызывается как строка, то PHP смотрит магический метод `__toString()`.

```php
class Test
{
    function __toString() {
        return $this->data;
    }
}

$test = new Test();
echo $test; // выполниться метод __toString()
```

Вызов объекта как функция

Когда объект вызывается как функция, то PHP смотрит магический метод `__invoke()`.

```php
class Doubles
{
    function __invoke($d) {
        return $d * 2;
    }
}

$double = new Doubles();
echo $double(2); // 4
echo $double(3); // 6
echo $double(4); // 8
```

Файлы

Создание файла

```php
$file = fopen('file.txt', 'w');
fwrite($file, 'контент для файла');
fclose($file);
```

Или сокращённо:

```php
file_put_contents('file.txt', 'контент для файла');
```

Добавить строку в файл

```php
$file = fopen('file.txt', 'a');

// блокировка записи в файл другим пользователям
flock($file, LOCK_EX);

// добавление контента в файл
fwrite($file, 'новая строка');

// снятие блокировки записи в файл
flock($file, LOCK_UN);

fclose($file);
```

Или сокращённо (добавив параметр `FILE_APPEND`, и параметр `LOCK_EX`, который предотвращает запись в файл другому пользователю во время выполнения текущей операции):

```php
file_put_contents('file.txt', 'новая строка', FILE_APPEND | LOCK_EX);
```

Вывести содержимое файла

```
// 1-й способ
file_get_contents('content.php');

// 2-й способ: html и php не отображаются (можно указать htmlspecialchars для их
отображения)
$file = file('content.php');
for ($i = 0; $i<sizeof($file); $i++) {
    echo $file[$i].'';
}

// 3-й способ: экранирует html и php + подсветка php-кода
highlight_file("content.php");
```

Также можно вывести любую строку из файла через функцию `file()`.

```
$file = file('content.php');
echo $file[0]; // 1-ая строка
echo $file[1]; // 2-ая строка
```

Проверить файл на существование

Проверить файл на существование можно через функцию `file_exists()`.

```
if (file_exists('myfile.php')) {
    // код
}
```

Также можно проверить, что файл является файлом или папкой.

```
if (is_file('file.txt')) {
    // код
}

if (is_dir($_SERVER['DOCUMENT_ROOT'].'/folder')) {
    // код
}
```

Изменить строку в файле

Пример изменения второй строки в текстовом файле.

```
$file = file('file.txt');
$file[1] = 'Новый текст для второй строки';
file_put_contents('file.txt', $file);
```

Удалить файл

Удалить файл можно через функцию `unlink()`.

```
unlink('file.txt');
```

Копировать файл

Копировать файл можно через функцию `copy()`.

```
copy('index.php', 'index_copy.php');
```

Вывести содержимое в папке

1-ый способ

Вывести содержимое папки (файлы и папки) можно через функцию `scandir()`, которая появилась в PHP 5.0.

```
$files = scandir('images/');
foreach ($files as $file) {
    echo $file;
}
```

2-ой способ: по шаблону

Вывести файлы по шаблону можно через функцию `glob()`.

```
$dir = glob('actions/*.php');
// actions/index.php
// actions/content.php
```

Также для примера выше можно указать несколько расширений:

```
$dir = glob("actions/*.{php,html}", GLOB_BRACE);
// actions/index.php
// actions/content.php
// actions/index.html
```

Параметр GLOB_BRACE разрешает подставлять несколько значений, например {php,html}.

3-ий способ: PHP 4

Если используется PHP 4, то вывести содержимое папки можно через функцию `opendir()`. Данная функция также доступна в PHP 5.

```
$dir = opendir('images/');
while (($file=readdir($dir)) !== false) {
    echo $file;
}
```

Загрузка файла на сервер

Шаг 1: Форма

Сначала надо добавить форму загрузки файла:

```
<form enctype="multipart/form-data" action="handler.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000">
    <p><input name="userfile" type="file"></p>
    <p><input type="submit" value="Send File"></p>
</form>
```

В этой форме надо обязательно указать атрибут `enctype="multipart/form-data"`, иначе файлы не будут передаваться через форму.

Также можно указать элемент формы с именем `MAX_FILE_SIZE` (обязательно перед элементом формы загрузки файла), где указывается допустимый размер загрузки файла. В первую очередь данный элемент нужен для того, чтобы сразу предупредить посетителя о превышении размера файла, не дожидаясь, пока сам файл не загрузится.

Шаг 2: Загрузка файла

Файл загружается через функцию `move_uploaded_file()`.

```
if (isset($_FILES['userfile'])) {

    // Папка для загрузки файлов
    $upload_dir = $_SERVER['DOCUMENT_ROOT'].'/uploads/';

    // Загрузка файла
    if (move_uploaded_file($_FILES['userfile']['tmp_name'],
$upload_dir.$_FILES['userfile']['name'])) {
        echo '<p>Файл успешно загружен</p>';
    }

}
```

Если файл не загружается, надо убедиться, что установлены правильные права на папку, куда загружаются файлы (в системе Linux).

Загрузить файл на FTP

Загрузить файл на FTP можно двумя способами. Через cURL и через FTP.

1-ый способ: cURL

```
$fp = fopen('/path/to/file.txt', 'r'); // файл для загрузки
$url = 'ftp://username:password@mysite.com:21/path/to/new/file.txt';

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_UPLOAD, 1);
curl_setopt($ch, CURLOPT_INFILE, $fp);
curl_setopt($ch, CURLOPT_INFILESIZE, filesize('/path/to/file.txt'));
curl_setopt($ch, CURLOPT_FTPASCII, 1);
```

2-ой способ: FTP

Загрузка файла происходит через функцию `ftp_put()`.

```
$conn_id = ftp_connect('ftp.site.ru');
$login_result = ftp_login($conn_id, 'ftp_user', 'ftp_pas');

ftp_put($conn_id, '/save/path/file.zip', 'file.zip', FTP_BINARY);
```

Скачать файл из интернета на сервер

1-ый способ:

Данный вариант будет работать, если в php.ini, параметр `allow_url_fopen` равен `1`.

```
$img_url = 'http://site.ru/images/image.jpg';

# 1-ый способ
copy($img_url, 'uploads/image.jpg');

# 2-ой способ (скачивание картинки)
$file = file_get_contents($img_url);
file_put_contents('path/to/save/image.jpg', $file);
```

2-ой способ: cURL

```
$fp = fopen('image.png', 'w'); // создание файла
$ch = curl_init('http://example.com/image.png');

curl_setopt($ch, CURLOPT_FILE, $fp); // запись в файл
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, false);

curl_exec($ch);
```

Соединить текстовые файлы в один файл

Код ниже выбирает все CSS-файлы и соединяет их в один текстовый файл.

```
// файл, где будет записываться вся информация
$save_file = $_SERVER['DOCUMENT_ROOT'].'/style.css';
$file_content = '';

// выборка CSS-файлов и получение их содержимого
foreach (glob($_SERVER['DOCUMENT_ROOT'].'/styles/*.css') as $file) {
    $file_content .= file_get_contents($file);
}

// сохранение полученной информации в файл
file_put_contents($save_file, $file_content, FILE_APPEND | LOCK_EX);
```

Вывод данных с Excel-файла

Чтение данных с Excel

Работать с файлом Excel можно через библиотеку **PHPExcel**. Данная библиотека понимает форматы «xlsx» и «xls».

Ссылка на библиотеку: https://github.com/PHPOffice/PHPExcel

Код ниже выводит содержимое Excel-файла:

```php
include 'PHPExcel/IOFactory.php'; // IOFactory отвечает за чтение Excel-файлов

$file = 'path/to/file.xlsx'; // файл для получения данных

$excel = PHPExcel_IOFactory::load($file); // подключить Excel-файл
$excel->setActiveSheetIndex(0); // получить данные из указанного листа

$sheet = $excel->getActiveSheet();

// формирование html-кода с данными
$html = '<table>';
foreach ($sheet->getRowIterator() as $row) {
    $html .= '<tr>';
    $cellIterator = $row->getCellIterator();
    foreach ($cellIterator as $cell) {

        // значение текущей ячейки
        $value = $cell->getCalculatedValue();

        // если дата, то преобразовать в формат PHP
        if (PHPExcel_Shared_Date::isDateTime($cell)) {
            $value = date('d.m.Y', PHPExcel_Shared_Date::ExcelToPHP($cell->getValue()));
        }

        $html .= '<td>'.$value.'</td>';
    }
    $html .= '<tr>';
}
$html .= '</table>';

// вывод данных
echo $html;
```

Стоит обратить внимание на метод `PHPExcel_Shared_Date::ExcelToPHP()`.

Excel хранит дату как число дней, прошедшего с 1 января 1900 года. А PHP хранит дату как число секунд, прошедшего с 1 января 1970. Для преобразование формат даты из Excel в PHP используется функция `PHPExcel_Shared_Date::ExcelToPHP()`.

Свойства файла

Узнать расширение файла

Узнать расширение файла можно через функцию `pathinfo()`.

```
$finfo = pathinfo('/index.php');
$finfo['extension']; // php

// другие свойства
$finfo['filename'];  // index
$finfo['dirname'];   // /
$finfo['basename'];  // index.php
```

Узнать MIME-файла

Узнать MIME файла можно через функцию `finfo_open()`.

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);
finfo_file($finfo, 'file.txt'); // text/plain
```

Функция `mime_content_type()` является устаревшей.

Размер файла

Узнать размер файла можно через функцию `filesize()`.

```
filesize('file.txt'); // 1856
```

Функция `filesize()` возвращает размер файла в байтах. Привести её в удобочитаемый вид можно через следующий код:

```
$size = filesize('file.txt'); // размер файла

$metrics[0] = 'Bite';
$metrics[1] = 'KB';
$metrics[2] = 'MB';
$metrics[3] = 'GB';
$metrics[4] = 'TB';
$metric = 0;

while(floor($size/1024) > 0) {
    ++$metric;
    $size /= 1024;
}

$ret = round($size,2)." ".(isset($metrics[$metric])?$metrics[$metric]:'??');

echo $ret; // 1.83 KB
```

Картинки

Создание картинки

Работа с изображениями в PHP происходит через библиотеку **GD Graphics Library** (встроена в PHP с версии PHP 4.3, до этого подключалась отдельно).

Пример создания картинки на PHP.

```
$im = imagecreate(125, 125); // создание картинки размером 125x125
$blue = imagecolorallocate($im,59,133,226); // фон картинки
imagepng($im, 'newpic.png'); // сохранение картинки
```

Сгенерированная картинка через код выше:

Создание миниатюры

1-ый способ: PHP 5.5

В PHP 5.5 появилась функция `imagescale()`, которая изменяет размеры картинки. До этого приходилось вручную узнавать размеры картинки для миниатюры (указано во втором примере).

```
function image_create($image_path) {

    $ext = pathinfo($image_path, PATHINFO_EXTENSION);
    switch($ext) {
        case 'gif':
            $im = imagecreatefromgif($image_path);
            break;
        case 'jpg':
            $im = imagecreatefromjpeg($image_path);
            break;
        case 'png':
            $im = imagecreatefrompng($image_path);
            break;
        default:
            throw new Exception('Неверный формат файла');
    }

    unset($ext);
    return $im;
}

function thumb($image_source, $save_as, $width) {

    // Проверка на наличие изображений
    if (!file_exists($image_source)) { throw new Exception('Изображение '.$image_source.'
не найдено'); }

    $image = image_create($image_source);
    $thumb = imagescale($image, $width);

    // сохранение прозрачности (для PNG и GIF)
    imagealphablending($thumb, false);
    imagesavealpha($thumb, true);

    // сохранение картинки
    imagepng($thumb, $save_as);

    // освобождение памяти
    imagedestroy($image);
    imagedestroy($thumb);
}

// создание миниатюры шириной в 200 пикселей
thumb('image.png', 'thumbname.png', 200);
```

2-ой способ: до PHP 5.5

```
function image_create($image_path) {

    $ext = pathinfo($image_path, PATHINFO_EXTENSION);
    switch($ext) {
        case 'gif':
            $im = imagecreatefromgif($image_path);
            break;
        case 'jpg':
            $im = imagecreatefromjpeg($image_path);
            break;
        case 'png':
            $im = imagecreatefrompng($image_path);
            break;
        default:
            throw new Exception('Неверный формат файла');
    }
    unset($ext);
    return $im;
}

function thumb($image_source, $save_as, $width) {

    // Проверка на наличие изображений
    if (!file_exists($image_source)) { throw new Exception('Изображение '.$image_source.'
не найдено'); }

    $image = image_create($image_source);
    list($im_size['width'], $im_size['height']) = getimagesize($image_source);
    $thumb_size['width'] = $width;
    $thumb_size['height'] = floor($im_size['height'] * ($thumb_size['width'] /
$im_size['width']));
    $thumb = imagecreatetruecolor($thumb_size['width'], $thumb_size['height']);

    // сохранение прозрачности (для PNG и GIF)
    imagealphablending($thumb, false);
    imagesavealpha($thumb, true);

    // создание миниатюры
    imagecopyresampled($thumb, $image, 0, 0, 0, 0, $thumb_size['width'],
$thumb_size['height'], $im_size['width'], $im_size['height']);

    // сохранение картинки
    imagepng($thumb, $save_as);

    // освобождение памяти
    imagedestroy($image);
    imagedestroy($thumb);
}

// создание миниатюры шириной в 200 пикселей
thumb('image.png', 'thumbname.png', 200);
```

Скриншот (снимок) сайта

Сделать скриншот (снимок) сайта можно через сервис mini.s-shot.ru.

```
<?php
$screenshot = [
    'url' => 'https://example.com/',
    'resolution' => '1024x768',
    'width' => 700,
];
?>

<img src="https://mini.s-
shot.ru/<?=$screenshot['resolution']?>/<?=$screenshot['width']?>/jpg/?<?=$screenshot['url'
]?>" alt="">
```

Создать Qr-код

1-ый способ: Google

Получить Qr-код можно через сервис Google.

chart.apis.google.com/chart?chs=200x200&cht=qr&chl=https://know-online.com/

Результат Qr-кода по ссылке выше:

С помощью PHP можно выполнить следующий код:

```
<?php
$url = 'https://example.com/page/3';
?>

<img src="http://chart.apis.google.com/chart?chs=200x200&cht=qr&chl=<?=$url?>">
```

2-ой способ: phpqrcode

Создать Qr-код можно через библиотеку phpqrcode.

Ссылка на библиотеку: https://github.com/t0k4rt/phpqrcode

```
include 'phpqrcode-master/qrlib.php';

QRcode::svg("Текст, который отобразится на при анализе Qr-кода");
```

Добавив второй параметр, картинка с Qr-кодом сохранится в указанный месте.

```
include 'phpqrcode-master/qrlib.php';

QRcode::png('https://example.com/', 'img/filename.png');
```

Вытащить картинки из HTML-кода в массив

Вытащить все картинки из HTML-кода можно через регулярные выражения,
с помощью функции `preg_match_all()`.

```
$content = 'text <img src="image.png"> text text <img alt="title" src="path/to/image.png">
text';
preg_match_all('/<img[^>]+src="(.*?)"[^>]*>/', $content, $images);
// массив $images хранит информацию об изображениях
```

В функции `preg_match_all()` задаётся третий параметр `$images`, в которой сохранится результат парсинга. На примере выше, массив `$images`будет хранить следующий информацию:

```
array (size=2)
  0 => array
      0 => '<img src="image.png">'
      1 => '<img alt="title" src="path/to/image.png">'
  1 => array
      0 => 'image.png'
      1 => 'path/to/image.png'
```

Почта

Отправить письмо на email

1-ый способ: стандартная функция mail()

Письмо на email отправляется через функцию `mail()`.

```
$header  = 'Content-Type: text/html; charset=utf-8'."\n";
$header .= 'From: "Admin" <noreply@example.com>';

mail('username@domain.ru', 'Тема письма', '<p>Сообщение</p>', $header);
```

2-ой способ: библиотека PHPMailer

```
require $_SERVER['DOCUMENT_ROOT'].'/PHPMailer/PHPMailerAutoload.php';

$mail = new PHPMailer;

$mail->From = 'no-reply@example.com';
$mail->FromName = 'Site name';
$mail->addAddress('username@domain.com', 'User Name');
$mail->isHTML(true);

$mail->Subject = 'Тема письма';
$mail->Body    = 'HTML-содержимое письма';

if (!$mail->send()) {
    echo 'Сообщение не было отправлено. Ошибка: ' . $mail->ErrorInfo;
} else {
    echo 'Сообщение успешно отправлено';
}
```

Ссылка на библиотеку: github.com/PHPMailer/PHPMailer

Отправить письмо со вложением

Отправлять письма с вложением рекомендуется через PHP-библиотеки, например «PHPMailer».

Вложение к письму добавляется через метод `addAttachment()`.

```php
require $_SERVER['DOCUMENT_ROOT'].'/PHPMailer/PHPMailerAutoload.php';

$mail = new PHPMailer;

$mail->From = 'no-reply@example.com';
$mail->FromName = 'Site name';
$mail->addAddress('username@domain.com', 'User Name');
$mail->isHTML(true);

$mail->Subject = 'Тема письма';
$mail->Body    = 'HTML-содержимое письма';

# прикрепление файла к письму
$mail->addAttachment($_SERVER['DOCUMENT_ROOT'].'/images/03.png', 'filename.jpg');

if (!$mail->send()) {
    echo 'Сообщение не было отправлено. Ошибка: ' . $mail->ErrorInfo;
} else {
    echo 'Сообщение успешно отправлено';
}
```

Ссылка на библиотеку: github.com/PHPMailer/PHPMailer

Время

Вывести дату

Вывести текущую дату можно через функцию `date()`.

```
# русский формат даты
date('d.m.Y');  // 16.03.2014
date('d F Y');  // 16 March 2014

# английский формат даты
date('F d, Y'); // March 16, 2014

# MySQL-формат
date('Y-m-d h:i:s'); // 2014-09-23 10:18:43
```

Временная зона

```
# php.ini
date.timezone = Europe/Moscow

# .htaccess
php_value date.timezone "Europe/Moscow"

# php-файл
date_default_timezone_set('Europe/Moscow'); // 1-ый способ
ini_set('date.timezone', 'Europe/Moscow');  // 2-ой способ
```

Если для сайта не будет указана временная зона, то PHP выдаст ошибку.

Также можно вручную указать время на основе GMT.

```
date_default_timezone_set('Etc/GMT-4');
```

Список поддерживаемых временных зон: php.net/manual/ru/timezones.php

Получить временную зону можно через следующий код:

```
date_default_timezone_get(); // 1-ый способ
ini_get('date.timezone');    // 2-ой способ

// Пример кода: Europe/Moscow
```

Дата из строки

```
strtotime('today');     // 17.05.2014
strtotime('now');       // 17.05.2014
strtotime('tomorrow');  // 18.05.2014
strtotime('yesterday'); // 16.05.2014

strtotime('+10 days');  // 27.05.2014
strtotime('+10 weeks'); // 26.07.2014

strtotime('next Thursday'); // 23.05.2014
strtotime('last Monday');   // 13.05.2014

strtotime('10 September 2000'); // 10.09.2000

strtotime('first day of next month'); // 01.06.2014
strtotime('last day of next month');  // 30.06.2014
```

Также можно указывать второй параметр, относительно которого будет меняться дата.

```
strtotime('+10 days', strtotime('10 September 2000')); // 20.09.2000
```

`strtotime()` возвращает количество секунд, прошедших с 1 января 00:00:00 UTC. Чтобы преобразовать дату в привычный вид, надо использовать функцию `date()`.

```
date('d.m.Y', strtotime('today')); // 17.05.2013
```

Дата рождения (возраст)

Узнать возраст человека по его дате рождения можно через метод `DateTime::diff()`.

```
$born = new DateTime('1990-11-05'); // дата рождения
$age = $born->diff(new DateTime)->format('%y');

echo $age; // 26
```

Подсчитать дни до указанной даты

Подсчитать дни до указанной даты можно через метод `DateTime::diff()`.

```php
$date = new DateTime('1990-11-95 14:02:45');
$days = $date->diff(new DateTime)->days;

echo $days; // 8774
```

ZIP

Создание ZIP

ZIP-архив создаётся через команду `$zip->open()`, с параметром `ZIPARCHIVE::CREATE` (данный параметр указывает, что если нет указанного архива, то его надо создать).

```
$zip = new ZipArchive();
$zip->open('archive.zip', ZIPARCHIVE::CREATE);
```

Добавляются файлы в архив через метод `addFile()`.

```
$zip->addFile('./file.txt', 'save_as_name.txt');
```

Для файлов с русским названием надо преобразовывать кодировку в cp866 (WinRar использует кодироку cp866 для чтения кириллицы).

```
$zip->addFile('./file.txt', iconv('utf-8', 'cp866', 'имя_файла.txt'));
```

Распаковать архив

Архив распаковывается через метод `extractTo()`.

```
function unzip($archive, $folder) {
    $zip = new ZipArchive();

    if (!file_exists($archive)) {
        throw new Exception('Архив не найден');
    }

    $zip->open($archive);
    $zip->extractTo($folder);
    $zip->close();
}

unzip('archive.zip', 'path/to/folder');
```

Количество файлов в ZIP

Узнать количество файлов можно через свойство `numFiles`.

```php
$zip = new ZipArchive();
$zip->open('archive.zip', ZIPARCHIVE::CREATE);

// узнать количество файлов
$zip->numFiles;
```

Проверить наличие файлов в ZIP

Проверить наличие файла в архиве можно через метод `locateName()`, который возвращает номер позиции файла.

```php
$zip = new ZipArchive();
$zip->open('archive.zip');

if ($zip->locateName('myfile.txt')) {
    echo 'Файл найден';
}
```

Настройки

Вывод ошибок

Вывод ошибок на сайте

Чтобы включить вывод ошибок, надо выполнить следующий PHP-код:

```
error_reporting(-1);
ini_set('display_errors', 1);
```

Значение `-1` указывает, что надо выводить все ошибки, например, устаревшие функции или несуществующие переменные.

На готовом сайте рекомендуется отключать вывод ошибок, и записывать все ошибки в файл.

Запись ошибок в файл

Рекомендуется отключать вывод ошибок на сайте, при этом сохраняя сами ошибки в файле. Данный способ помогает отслеживать ошибки в сложно допустимых местах, например перед редиректом.

Способы указания лога с ошибками PHP:

```
# .htaccess (рекомендуется)
php_value log_errors "On"
php_value error_log /var/log/example.com_php_errors.log

# .PHP-код
ini_set('log_errors', 'On');
ini_set('error_log', '/var/log/example.com_php_errors.log');

# php.ini
log_errors = On
error_log = /var/log/php_errors.log
```

Хранить файл с ошибками лучше всего вне директории сайта, чтобы его нельзя было открыть через браузер или HTTP-запрос.

Узнать версию PHP

1-ый способ: PHP-код

Узнать версию PHP можно через функцию `phpversion()` или через константу `PHP_VERSION`.

```
echo PHP_VERSION;  // PHP 5.5.9
echo phpversion(); // PHP 5.5.9
```

2-ой способ: консольная команда

Версию PHP можно узнать через консольную команду:

```
php -v
```

Пример ответа:

```
PHP 5.5.9 (cli) (built: Apr 17 2015 11:44:57)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
    with Zend OPcache v7.0.3, Copyright (c) 1999-2014, by Zend Technologies
```

Узнать установленные расширения

1-ый способ: PHP

Узнать все установленные расширения можно через команду `get_loaded_extensions()`, которая вернёт примерно следующий код:

```
Array
(
    [0] => Core
    [1] => date
    [2] => ereg
    [3] => libxml
    [4] => openssl
    [5] => pcre
    [6] => zlib
    ...
}
```

2-ой способ: консольная команда

Вывести список установленных модулей можно через консольную команду:

```
php -m
```

Выполнить PHP-код в консоли ОС

Выполнить PHP-код в консоли ОС можно через следующую команду:

```
php -a
```

Разное

Об PHP

PHP — язык программирования, применяемый для разработки сайтов и веб-приложений.

На PHP работают такие крупные сайты как wikipedia.org, vk.com, facebook.com и много других.

PHP является языком программирования с открытым исходным кодом. То есть при желании можно отредактировать исходные коды PHP под себя (требуется знания языка программирования C++).

Изначально PHP назывался «PHP/FI». Начиная с третьей версии, PHP/FI стал называться просто PHP.

Интерпретатор PHP 5 работает на движке Zend Engine 2. Интерпретатор PHP 7 будет работать на движке Zend Engine 3.

В случае обнаружения ошибки в PHP, об этом можно сообщить в специальном ресурсе разработчиков PHP — bugs.php.net.

Cookies

С помощью cookies можно сохранять информацию. Например, когда пользователь посещает страницу сайта, в cookies можно сделать запись «страница посещена». При следующем открытии страницы, можно выводить сообщение «страница просмотрена».

Создаются cookies через функцию `setcookie()`.

```
// создание cookie
setcookie("cookie_name", "Сохранённая информация", time() + 3600);
```

Все cookies доступны через массив `$_COOKIE`. Например, cookie, созданный выше, можно вывести через следующий код:

```
echo $_COOKIE['cookie_name']; // Сохранённая информация
```

Третий параметр функции `setcookie()` указывает время хранения cookies в секундах. В примере выше указано время хранения в 1 час (3600 секунд).

Cookies передаются клиенту вместе с HTTP заголовками, поэтому cookies должны быть определены до вывода HTML.

Нельзя создать cookies и тут же их использовать. Вывести cookies можно будет с обновлением страницы.

Через четвёртый параметр можно указывать URL, для которой будет сохраняться cookies. По умолчанию cookies сохраняется для текущего URL.

```
// Сохранение cookies для всех страниц
setcookie("name", "Сохранённая информация", time() + 3600, '/');

// Сохранение cookies для раздела "catalog/"
setcookie("name", "Сохранённая информация", time() + 3600, '/catalog/');

// Сохранение cookies для указанной страницы
setcookie("name", "Сохранённая информация", time() + 3600, '/about.html');
```

Создавать cookies рекомендуется с включённым параметром **httponly**, который запрещает доступ к cookies из JavaScript (в целях предотвращения XSS-атак).

```
# в .htaccess (рекомендуется)
<IfModule php5_module>
php_flag session.cookie_httponly on
</IfModule>

# при создании cookies
setcookie("name", "Сохранённая информация", time() + 3600, '/page.html', '', false, true);
```

Удаление cookies

Чтобы удалить cookies, надо для текущей cookies задать время хранения в прошедшем времени. Например, код ниже устанавливает cookies на час назад, т.е. удаляет его, т.к. истекло время хранения.

```
// Пример удаления cookies
setcookie("name", "Сохранённая информация", time() - 3600);
```

Обновление cookies

Cookies обновляются также, как и создаются, только устанавливается новое значение.

```
// Создание cookies
setcookie("name", "Сохранённая информация", time() + 3600);

// Обновление cookies
setcookie("name", "Новое значение", time() + 3600);
```

8 советов по использованию PHP

1. Использование PHP с версии 5.5

В качестве PHP рекомендуется выбирать версию PHP 5.5, т.к. с данной версии в ядро PHP встроен **OpCache**, что ускоряет работу PHP.

Чтобы убедиться, что OpCache работает, можно просто выполнить функцию `opcache_get_status()`.

2. Сокращённый вызов PHP-кода

PHP-код выполняется между тегами `<?php ?>`. Также возможно указывать сокращённый вид тегов `<? ?>`, но их не рекомендуется использовать (по умолчанию такое использование тегов запрещено).

Даже если запрещено использовать теги <? ?>, сокращённый код вывода переменной <?=$var?> работает в любом случае.

3. Закрывающий тег ?> в конце файла

В конце файла рекомендуется не использовать закрывающий тег `?>`, т.к. это помогает избежать добавления случайных символов пробела или перевода строки, которые могут послужить причиной нежелательных эффектов.

Например, после подключения файла через `include()` надо использовать `header()`, чтобы сделать перенаправление, и в этом случае PHP может вывести ошибку, т.к. в подключаемом файле страница уже начала вывод контента из-за закрывающего тега `?>` (который добавил перевод строки), а до использования `header()` не должно быть вывода содержимого страницы.

Об этом также написано в официальной документации: «*Закрывающий тег PHP-блока в конце файла не является обязательным, и в некоторых случаях его опускание довольно полезно, например, при использовании include ... нежелательные пробелы не останутся в конце файла и вы все еще сможете добавить http-заголовки*».

4. Использование регулярных выражений для поиска текста

Если надо найти строку в тексте, не используя шаблоны регулярных выражений, то рекомендуется использовать `strpos()` для производительности кода.

```php
$text = 'PHP-script';

# не правильно
if (preg_match('/PHP/', $text)) {
    echo 'строка найдена';
}

# правильно
if (strpos($text, 'PHP') !== false) {
    echo 'строка найдена';
}
```

5. Вывод ошибок на сайте

Вместо вывода ошибок на сайте рекомендуется записывать ошибки в файл. Также данный способ помогает отслеживать ошибки в сложно допустимых местах, например перед редиректом или в Ajax-запросах.

Подробнее о выводе ошибок можно прочитать 63 странице.

6. Длина массива в цикле for

В цикле `for` длину массива рекомендуется устанавливать заранее, иначе при каждой итерации цикла будет подсчитываться количество элементов в массиве.

```php
$arr = ['PHP', 'Zend', 'MySQL'];

// неправильно
for ($i = 0; $i < sizeof($arr); $i++) {
    // код
}

// правильно
for ($i = 0, $l = sizeof($arr); $i < $l; $i++) {
    // код
}
```

7. Использование httponly при создании cookies

Параметр **httponly** запрещает доступ к cookies из JavaScript, в целях предотвращения XSS-атак.

```
# в .htaccess (рекомендуется)
<IfModule php5_module>
php_flag session.cookie_httponly on
</IfModule>

# при создании cookies
setcookie("name", "Сохранённая информация", time() + 3600, '/page.html', '', false, true);
```

Подробнее о создании cookies можно прочитать на 67 странице.

8. Кэширование данных через Memcached

Для кэширования данных используйте Memcached, если сервер это позволяет, т.к. Memcached сохраняет данные в память сервера, что увеличивает скорость сохранения и получения закэшированных данных.

Zend Certification Exam

Zend Certification Exam — экзамен, который даёт возможность PHP-разработчику официально подтвердить свою квалификацию.

Сдача данного экзамена происходит через организацию **Pearson VUE**, которая имеет более 3 700 центров тестирования по всему миру.

Запись на экзамен начинается с сайта компании Zend (zend.com/en/services/certification/), где можно подать заявку на сдачу экзамена и оплатить **ваучер**, без которого сдача экзамена невозможна. Стоимость экзамена составит $195 + налоги + стоимость сдачи экзамена в центре тестирования.

На экзамене надо будет в течение 90 минут ответить на 70 вопросов. Сдача экзамена происходит на английском языке.

Вопросы бывают трёх типов:

1. несколько вариантов ответа с одним правильным;
2. несколько вариантов ответа с несколькими верными (необходимо указать все правильные);
3. открытые вопросы, ответы на которые нужно не выбирать из предложенных, а написать самим.

Все вопросы не выше среднего уровня сложности, однако, ориентированы на программистов с достаточным опытом работы и касаются только основных аспектов использования ядра PHP.