



# PHP РАБОТА С ФАЙЛАМИ JSON. CSV



**АЛЕКСЕЙ КУЛАГИН** / ШТРИХПУНКТИР



# АЛЕКСЕЙ КУЛАГИН

Технический руководитель, системный архитектор, разработчик  
в «Штрихпунктир»



[al.qlogin@gmail.com](mailto:al.qlogin@gmail.com)



[alex\\_kulagin](https://t.me/alex_kulagin)



# ПЛАН ЗАНЯТИЯ

1. Форматы DSV и JSON
2. Удаленные ресурсы
3. Работа с файловой системой



# ФОРМАТЫ DSV И JSON



# DSV

**DSV (delimiter separated values)** — текстовый формат, в котором данные разделены специальными символами. Данный формат предназначен для представления табличных данных.

Каждая строка — отдельная запись в таблице.

В зависимости от разделителя существуют разные форматы *CSV* и *TSV* с разделителем запятая (comma) и табуляция соответственно.

В DSV первой строкой может идти перечисление названий столбцов.


Все значения хранятся как строки и тип значения определяется непосредственно в коде.



# JSON

**JSON (JavaScript Object Notation)** — текстовый формат, в котором данные описаны примитивами языка JS:

- null
- булево (true / false)
- число
- строка
- массив примитивов
- объект с ключами строками и значениями примитивами




Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования.

Для многих языков, в том числе PHP, существует готовый код для создания и обработки данных в формате JSON.



# УДАЛЕННЫЕ РЕСУРСЫ





Содержимое страниц в интернете также является текстовыми данными и к ним тоже можно обращаться. Например, вызов удаленного api <https://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22> для получения погоды с сайта openweather вернет json.

Отличие таких ресурсов, что в них нельзя записать данные как на файловую систему.



# РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ



## ПРОСТЕЙШИЕ ФУНКЦИИ

Во всех функциях (если не оговорено иначе) первый аргумент — путь к файлу.

# ФУНКЦИЯ FILE

Функция `file()` читает содержимое файла и помещает его в массив.

Каждый элемент массива соответствует строке файла, с символами новой строки включительно.

В случае ошибки `file()` возвращает FALSE.

Вторым параметром можно указать специальные флаги:

- `FILE_USE_INCLUDE_PATH` – ищет файл в `include_path`
- `FILE_IGNORE_NEW_LINES` – пропускать новую строку в конце каждого элемента массива
- `FILE_SKIP_EMPTY_LINES` – пропускать пустые строки

```
// Используем необязательный параметр flags (начиная с PHP 5)
```

```
$trimmed = file('somefile.txt', FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
```

# ФУНКЦИЯ FILE\_GET\_CONTENTS

Функция `file_get_contents()` читает содержимое файла в строку.

Использование функции `file_get_contents()` наиболее предпочтительно в случае необходимости получить содержимое файла целиком, поскольку для улучшения производительности функция использует технику отображения файла в память (memory mapping), если она поддерживается вашей операционной системой.

В случае неудачи функция `file_get_contents()` вернёт `FALSE`.

```
$homepage = file_get_contents('http://www.example.com/');
```

# ФУНКЦИЯ FILE\_PUT\_CONTENTS

Функция `file_put_contents()` записывает данные в файл.

Если файл не существует, то он будет создан. Иначе существующий файл будет перезаписан.

Второй аргумент — данные для записи — могут представлять собой строку, массив и поток. В случае массива все элементы будут склеены без разделителя. Если данные являются потоковым ресурсом (stream), оставшийся буфер этого потока будет скопирован в указанный файл.



Третьим аргументом можно указать специальные флаги:

- `FILE_USE_INCLUDE_PATH` – ищет `filename` в подключаемых директориях
- `FILE_APPEND` – если файл `filename` уже существует, данные будут дописаны в конец файла вместо того, чтобы его перезаписать
- `LOCK_EX` – получить эксклюзивную блокировку на файл на время записи. Другими словами, между вызовами `fopen()` и `fwrite()` произойдет вызов функции `flock()`



### *Простой пример с использованием `file_get_contents()`*

```
1 $file = 'people.txt';
2 // Открываем файл для получения существующего содержимого
3 $current = file_get_contents($file);
4 // Добавляем нового человека в файл
5 $current .= "John Smith\n";
6 // Пишем содержимое обратно в файл
7 file_put_contents($file, $current);
```

### *Пример с использованием флагов*

```
1 $file = 'people.txt';
2 // Новый человек, которого нужно добавить в файл
3 $person = "John Smith\n";
4 // Пишем содержимое в файл, используя флаг FILE_APPEND для дописывания
5 // содержимого в конец файла и флаг LOCK_EX для предотвращения записи
6 // данного файла кем-нибудь другим в данное время
7 file_put_contents($file, $person, FILE_APPEND | LOCK_EX);
```



## РЕСУРСЫ ДЛЯ РАБОТЫ С ФАЙЛОМ

# ФУНКЦИЯ `fopen`

Функция `fopen()` создает именованный ресурс для работы с файлом

Вторым параметром нужно указать режим работы с файлом. Самыми распространенными являются:

- **r** — Открывает файл только для чтения; помещает указатель в начало файла.
- **r+** — Открывает файл для чтения и записи; помещает указатель в начало файла.
- **w** — Открывает файл только для записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует — пробует его создать.
- **w+** — Открывает файл для чтения и записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует — пытается его создать.
- **a** — Открывает файл только для записи; помещает указатель в конец файла. Если файл не существует - пытается его создать.
- **a+** — Открывает файл для чтения и записи; помещает указатель в конец файла. Если файл не существует — пытается его создать.

---

Windows предлагает флаг режима текстовой трансляции (**t**), который автоматически переводит `\n` в `\r\n` во время работы с файлом.

И наоборот — вы также можете использовать **b**, чтобы принудительно включить бинарный режим, в котором ваши данные не будут преобразовываться. Чтобы использовать эти режимы, укажите 'b' или 't' последней буквой параметра.

```
$handle = fopen("/home/netology/data.txt", "ab");
```

# ФУНКЦИЯ FCLOSE

Функция `fclose()` закрывает ресурс, связанный с файлом через `fopen`.

## ФУНКЦИЯ FREAD

Функция `fread()` предназначена для бинарно-безопасного чтения из ресурса, открытого с помощью `fopen`. Причем файл должен быть открыт с флагом `b` на системах, которые различают текстовые и бинарные форматы файлов (например Windows).

Первый аргумент — ресурс.

Вторым аргументом указывается количество байт, которые нужно считать.

Чтение останавливается как только было достигнуто одно из следующих условий:

- было прочитано length байт
- достигнут EOF (конец файла)
- стал доступен пакет или произошел тайм-аут сокета (для сетевых потоков)
- если читаемый поток является буферизованным и не представляет собой обычный файл, то за один раз максимум читается количество байт, равное размеру одной порции данных (обычно это 8192), однако, в зависимости от ранее буферизованных данных, размер возвращаемых данных может быть больше размера одной порции данных

```
1 $filename = "/usr/local/something.txt";  
2 $handle = fopen($filename, "r");  
3 $contents = fread($handle, filesize($filename));  
4 fclose($handle);
```

## ФУНКЦИЯ FWRITE

Функция `fwrite()` предназначена для бинарно-безопасной записи в ресурс, открытый с помощью `fopen`. Причем файл должен быть открыт с флагом `b` на системах, которые различают текстовые и бинарные форматы файлов (например Windows).

Первый аргумент — ресурс.

Второй аргумент — строка.



Третьим аргументом можно передать количество данных, которые должны быть записаны. Если строка содержит меньше данных, чем указано, то запись закончится, когда вся строка будет записана.

При повторной записи в файловый ресурс, данные будут добавлены в конец содержимого файла

```
1 $fp = fopen('data.txt', 'w');  
2 fwrite($fp, '1');  
3 fwrite($fp, '23'); // данные будут дописаны  
4 fclose($fp);  
5  
6 // содержимое 'data.txt' теперь 123, а не 23!
```



# РАБОТА С CSV

# ФУНКЦИЯ FGETCSV

Функция `fgetcsv()` читает строку из файла, производит разбор данных CSV и возвращает индексированный массив с прочитанными полями.

Параметры функции:

- `handle` — корректный файловый указатель на файл, успешно открытый при помощи `fopen()`
- `length` — должен быть больше самой длинной строки (в символах), найденной в CSV-файле (включая завершающий символ конца строки). В противном случае, строка будет разбита на куски длиной в `length` символов, если только место разрыва не будет внутри ограничителей полей (enclosure). Отсутствие этого параметра (или установка его в 0 в PHP 5.1.0 и выше) приведет к тому, что длина строки будет неограничена. Это может сказаться на скорости выполнения
- `delimiter` — необязательный параметр `delimiter` устанавливает разделитель поля (только один символ)

```
1 $row = 1;
2 $handle = fopen("test.csv", "r");
3 if ($handle !== FALSE) {
4     $data = fgetcsv($handle, 1000, ",", "");
5     while ($data !== FALSE) {
6         $num = count($data);
7         echo "<p> $num полей в строке $row:</p>\n";
8         $row++;
9         for ($c=0; $c < $num; $c++) {
10             echo $data[$c] . "<br />\n";
11         }
12     }
13     fclose($handle);
14 }
```

## ФУНКЦИЯ `FPUTCSV`

Функция `fputcsv()` форматирует строку (переданную в виде массива `fields`) в виде CSV и записывает её (заканчивая переводом строки) в указанный файл.

Параметры функции:

- `handle` — указатель на файл должен быть корректным и указывать на файл, успешно открытый функциями `fopen()`
- `fields` — массив строк (`string`)
- `delimiter` — дополнительный параметр `delimiter` устанавливает разделитель полей (только один символ)

```
1 $list = array (  
2     array('aaa', 'bbb', 'ccc', 'dddd'),  
3     array('123', '456', '789'),  
4     array('"aaa"', '"bbb"')  
5 );  
6  
7 $fp = fopen('file.csv', 'w');  
8  
9 foreach ($list as $fields) {  
10     fputcsv($fp, $fields);  
11 }  
12  
13 fclose($fp);  
14 /* Содержимое файла file.csv  
15 aaa,bbb,ccc,dddd  
16 123,456,789  
17 """"aaa""", """"bbb"""  
18 */
```



## **ФУНКЦИИ РАБОТЫ С JSON**

# ФУНКЦИЯ `JSON_DECODE`

Функция `json_decode()` декодирует строку JSON.

Данная функция применяется для разбора строки в php-переменную. По умолчанию объекты json преобразуются в `stdClass`.

```
1  $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
2  $parsed = json_decode($json);
3  var_dump($parsed);
4  /*
5  object(stdClass)#1 (5) {
6      ["a"] => int(1)
7      ["b"] => int(2)
8      ["c"] => int(3)
9      ["d"] => int(4)
10     ["e"] => int(5)
11 }
12 */
```



Если вторым аргументом передать true, то объекты будут преобразованы в ассоциативные массивы

```
1  $parsed = json_decode($json, true);
2  var_dump($parsed);
3  /*
4  array(5) {
5      ["a"] => int(1)
6      ["b"] => int(2)
7      ["c"] => int(3)
8      ["d"] => int(4)
9      ["e"] => int(5)
10 }
11 */
```

# ФУНКЦИЯ JSON\_ENCODE

Функция `json_encode()` преобразует значение переменной php в JSON строку.

Пример сохранения в файл

```
1 $arr = array('a' => 1, 'b' => 2, 'c' => '3' );  
2 file_put_contents("./numbers.json", json_encode($arr));  
3 // в файле numbers.json будет {"a": 1, "b": 2, "c": "3"}
```

## ФУНКЦИЯ `JSON_LAST_ERROR_MSG`

Функция `json_last_error_msg()` возвращает сообщение об ошибке в случае успешного выполнения, `"No error"`, если ошибки не произошло, или `FALSE` в случае возникновения ошибки после вызова `json_encode()` или `json_decode()`.

# ФУНКЦИЯ `JSON_LAST_ERROR`

Функция `json_last_error()` возвращает целочисленное значение, которое может быть одной из следующих констант:

- `JSON_ERROR_NONE` — Ошибок нет
- `JSON_ERROR_DEPTH` — Достигнута максимальная глубина стека
- `JSON_ERROR_STATE_MISMATCH` — Неверный или некорректный JSON
- `JSON_ERROR_CTRL_CHAR` — Ошибка управляющего символа, возможно неверная кодировка
- `JSON_ERROR_SYNTAX` — Синтаксическая ошибка
- `JSON_ERROR_UTF8` — Некорректные символы UTF-8, возможно неверная кодировка
- `JSON_ERROR_RECURSION` — Одна или несколько зацикленных ссылок в кодируемом значении
- `JSON_ERROR_INF_OR_NAN` — Одно или несколько значений NAN или INF в кодируемом значении
- `JSON_ERROR_UNSUPPORTED_TYPE` — Передано значение с неподдерживаемым типом
- `JSON_ERROR_INVALID_PROPERTY_NAME` — Имя свойства не может быть закодировано
- `JSON_ERROR_UTF16` — Некорректный символ UTF-16, возможно некорректно закодирован

### Пример использования:

```
1 // Верная json-строка
2 $json[] = '{"Organization": "PHP Documentation Team"}';
3 // Неверная json-строка, которая вызовет синтаксическую ошибку,
4 // здесь в качестве кавычек мы используем ' вместо "
5 $json[] = "'Organization': 'PHP Documentation Team'";
6 foreach ($json as $string) {
7     echo 'Декодируем: ' . $string;
8     json_decode($string);
9
10    switch (json_last_error()) {
11        case JSON_ERROR_NONE:
12            echo ' - Ошибок нет';
13            break;
14        case JSON_ERROR_DEPTH:
15            echo ' - Достигнута максимальная глубина стека';
16            break;
17        case JSON_ERROR_STATE_MISMATCH:
18            echo ' - Некорректные разряды или несоответствие режимов';
19            break;
```

```
1 case JSON_ERROR_CTRL_CHAR:
2     echo ' – Некорректный управляющий символ';
3     break;
4 case JSON_ERROR_SYNTAX:
5     echo ' – Синтаксическая ошибка, некорректный JSON';
6     break;
7 case JSON_ERROR_UTF8:
8     echo ' – Некорректные символы UTF-8, возможно неверно закодирован';
9     break;
10 default:
11     echo ' – Известная ошибка';
12     break;
13 }
14
15 echo PHP_EOL;
16 }
```

### *Результат выполнения:*

Декодируем: `{"Organization": "PHP Documentation Team"}` – Ошибок нет

Декодируем: `{'Organization': 'PHP Documentation Team'}` – Синтаксическая ошибка, некорректный JSON

Для чтения и записи json в файлы используются стандартные функции, рассмотренные выше.



**Задавайте вопросы и напишите отзыв о лекции!**

**АЛЕКСЕЙ КУЛАГИН**



[al.klugin@gmail.com](mailto:al.klugin@gmail.com)



[alex\\_kulagin](https://www.telegram.me/alex_kulagin)