**1) require**

(PHP 4, PHP 5, PHP 7)

require аналогично [include](https://www.php.net/manual/ru/function.include.php), за исключением того, что при ошибке он также выдаст фатальную ошибку уровня **E\_COMPILE\_ERROR**. Другими словами, он остановит выполнение скрипта, тогда как [include](https://www.php.net/manual/ru/function.include.php) только выдал бы предупреждение **E\_WARNING**, которое позволило бы скрипту продолжить выполнение.

Пример 1.1

require\_once 'application/lib/Dev.php';

Выражение require\_once аналогично [require](https://www.php.net/manual/ru/function.require.php) за исключением того, что PHP проверит, включался ли уже данный файл, и если да, не будет включать его еще раз.

**2) spl\_autoload\_register**

(PHP 5 >= 5.1.0, PHP 7)

*spl\_autoload\_register* — Регистрирует заданную функцию в качестве реализации метода \_\_autoload()

**Описание**

spl\_autoload\_register ([ [callable](https://www.php.net/manual/ru/language.types.callable.php) $autoload\_function [, bool $throw = TRUE [, bool $prepend = FALSE ]]] ) : bool

Регистрирует функцию \_\_autoload в предоставляемой SPL очереди. В результате очередь будет активирована, даже если ранее она была выключена.

Если в вашем скрипте реализована функция [\_\_autoload()](https://www.php.net/manual/ru/function.autoload.php), ее необходимо явно зарегистрировать в очереди \_\_autoload. Это необходимо, т.к. spl\_autoload\_register() полностью заменяет механизм кеширования [\_\_autoload()](https://www.php.net/manual/ru/function.autoload.php) функциями [spl\_autoload()](https://www.php.net/manual/ru/function.spl-autoload.php) и [spl\_autoload\_call()](https://www.php.net/manual/ru/function.spl-autoload-call.php).

spl\_autoload\_register() позволяет задать несколько реализаций метода автозагрузки. Она создает очередь из функций автозагрузки в порядке их определения в скрипте, тогда как встроенная функция [\_\_autoload()](https://www.php.net/manual/ru/function.autoload.php) может иметь только одну реализацию.

**Список параметров**

*autoload\_function*

Имя функции, реализующей метод [spl\_autoload()](https://www.php.net/manual/ru/function.spl-autoload.php). Если аргумент не задан, будет зарегистрирована реализация по умолчанию.

*throw*

Этот параметр определяет, должна ли spl\_autoload\_register() выбрасывать исключение, если зарегистрировать autoload\_function оказалось невозможным.

*prepend*

Если передано значение true, spl\_autoload\_register() поместит указанную функцию в начало очереди вместо добавления в конец.

Возвращаемые значения

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Пример 2.1

    spl\_autoload\_register(function ($class) {

        include $class . '.php';

    });

Пример 2.2

    spl\_autoload\_register(function($class){

        $path=str\_replace('\\','/',$class.'.php');

        if(file\_exists($path)) {

            require $path;

        }

    });

**3) str\_replace**

(PHP 4, PHP 5, PHP 7)

str\_replace — Заменяет все вхождения строки поиска на строку замены

**Описание**

str\_replace ( [mixed](https://www.php.net/manual/ru/language.pseudo-types.php#language.types.mixed) $search , [mixed](https://www.php.net/manual/ru/language.pseudo-types.php#language.types.mixed) $replace , [mixed](https://www.php.net/manual/ru/language.pseudo-types.php#language.types.mixed) $subject [, int &$count ] ) : [mixed](https://www.php.net/manual/ru/language.pseudo-types.php#language.types.mixed)

Эта функция возвращает строку или массив, в котором все вхождения search в subject заменены на replace.

Если не нужны сложные правила поиска/замены (например, регулярные выражения), использование этой функции предпочтительнее [preg\_replace()](https://www.php.net/manual/ru/function.preg-replace.php).

**Список параметров**

Если search и replace - массивы, то str\_replace() использует каждое значение из соответствующего массива для поиска и замены в subject. Если в массиве replace меньше элементов, чем в search, в качестве строки замены для оставшихся значений будет использована пустая строка. Если search - массив, а replace - строка, то эта строка замены будет использована для каждого элемента массива search. Обратный случай смысла не имеет.

Если search или replace являются массивами, их элементы будут обработаны от первого к последнему.

*search*

Искомое значение, также известное как needle (иголка). Для множества искомых значений можно использовать массив.

*replace*

Значение замены, будет использовано для замены искомых значений search. Для множества значений можно использовать массив.

*subject*

Строка или массив, в котором производится поиск и замена, также известный как haystack (стог сена).

Если subject является массивом, то поиск с заменой будет осуществляться над каждым элементом subject, а результатом функции также будет являться массив.

*count*

Если передан, то будет установлен в количество произведенных замен.

**Возвращаемые значения**

Эта функция возвращает строку или массив с замененными значениями.

Пример 3.1

// присваивает <body text='black'>  
$bodytag = str\_replace("%body%", "black", "<body text='%body%'>");

**4) trim**

(PHP 4, PHP 5, PHP 7)

trim — Удаляет пробелы (или другие символы) из начала и конца строки

**Описание**

**trim** ( string $str [, string $character\_mask = " \t\n\r\0\x0B" ] ) : string

Эта функция возвращает строку str с удаленными из начала и конца строки пробелами. Если второй параметр не передан, **trim()** удаляет следующие символы:

* " " (ASCII *32* (*0x20*)), обычный пробел.
* "\t" (ASCII *9* (*0x09*)), символ табуляции.
* "\n" (ASCII *10* (*0x0A*)), символ перевода строки.
* "\r" (ASCII *13* (*0x0D*)), символ возврата каретки.
* "\0" (ASCII *0* (*0x00*)), *NUL*-байт.
* "\x0B" (ASCII *11* (*0x0B*)), вертикальная табуляция.

**Список параметров**

*str*

Обрезаемая строка ([string](https://www.php.net/manual/ru/language.types.string.php)).

*character\_mask*

Можно также задать список символов для удаления с помощью необязательного аргумента character\_mask. Просто перечислите все символы, которые вы хотите удалить. Можно указать конструкцию *..* для обозначения диапазона символов.

**Возвращаемые значения**

Обрезаемая строка.

Пример 4.1

        //Удалим из url стоящий с переди слэш "/", с помощью функции trim()

        $url=trim($\_SERVER['REQUEST\_URI'],'/');

**5) $\_SERVER**

**$HTTP\_SERVER\_VARS [удалено]**

(PHP 4 >= 4.1.0, PHP 5, PHP 7)

$\_SERVER -- $HTTP\_SERVER\_VARS [удалено] — Информация о сервере и среде исполнения

**Описание**

Переменная *$\_SERVER* - это массив, содержащий информацию, такую как заголовки, пути и местоположения скриптов. Записи в этом массиве создаются веб-сервером. Нет гарантии, что каждый веб-сервер предоставит любую из них; сервер может опустить некоторые из них или предоставить другие, не указанные здесь. Тем не менее, многие эти переменные присутствуют в [» спецификации CGI/1.1](http://www.faqs.org/rfcs/rfc3875), так что вы можете ожидать их наличие.

**Замечание**: До PHP 5.4.0, *$HTTP\_SERVER\_VARS* содержит ту же начальную информацию, но она не [суперглобальная](https://www.php.net/manual/ru/language.variables.superglobals.php). (Заметьте, что *$HTTP\_SERVER\_VARS* и *$\_SERVER* являются разными переменными, так что PHP обрабатывает их соответственно).

**Индексы**

Вы можете найти (а можете и не найти) любой из следующих элементов в массиве *$\_SERVER*. Обратите внимание, что немногие элементы, если они есть, будут доступны (или даже будут иметь какое-нибудь значение), если PHP запущен в [командной строке](https://www.php.net/manual/ru/features.commandline.php).

'*PHP\_SELF*'

Имя файла скрипта, который сейчас выполняется, относительно корня документов. Например, *$\_SERVER['PHP\_SELF']* в скрипте по адресу *http://example.com/foo/bar.php* будет */foo/bar.php*. Константа [\_\_FILE\_\_](https://www.php.net/manual/ru/language.constants.predefined.php) содержит полный путь и имя файла текущего (то есть подключенного) файла. Если PHP запущен в командной строке, эта переменная содержит имя скрипта, начиная с PHP 4.3.0. Раньше она была недоступна.

'[argv](https://www.php.net/manual/ru/reserved.variables.argv.php)'

Массив агрументов, переданных скрипту. Когда скрипт запущен в командой строке, это дает C-подобный доступ к параметрам командной строки. Когда вызывается через метод GET, этот массив будет содержать строку запроса.

'[argc](https://www.php.net/manual/ru/reserved.variables.argc.php)'

Содержит количество параметров, переданных скрипту (если запуск произведен в командной строке).

'*GATEWAY\_INTERFACE*'

Содержит используемую сервером версию спецификации CGI; к примеру '*CGI/1.1*'.

'*SERVER\_ADDR*'

IP-адрес сервера, на котором выполняется текущий скрипт.

'*SERVER\_NAME*'

Имя хоста, на котором выполняется текущий скрипт. Если скрипт выполняется на виртуальном хосте, здесь будет содержатся имя, определенное для этого виртуального хоста.

**Замечание**: В Apache 2 вы должны установить *UseCanonicalName = On* и *ServerName*. В противном случае это значение отразит имя хоста, предоставленное клиентом, которое может быть подделано. Небезопасно полагаться на это значение в контексте, требующем безопасности.

'*SERVER\_SOFTWARE*'

Строка идентификации сервера, указанная в заголовках, когда происходит ответ на запрос.

'*SERVER\_PROTOCOL*'

Имя и версия информационного протокола, через который была запрошена страница; к примеру '*HTTP/1.0*';

'*REQUEST\_METHOD*'

Какой метод был использован для запроса страницы; к примеру '*GET*', '*HEAD*', '*POST*', '*PUT*'.

**Замечание**: PHP-скрипт завершается после отправки заголовков (то есть после того, как осуществляется любой вывод без буферизации вывода), если метод запроса был *HEAD*.

'*REQUEST\_TIME*'

Временная метка начала запроса. Доступно с PHP 5.1.0.

'*REQUEST\_TIME\_FLOAT*'

Временная метка начала запроса с точностью до микросекунд. Доступно с PHP 5.4.0.

'*QUERY\_STRING*'

Строка запроса, если есть, через которую была открыта страница.

'*DOCUMENT\_ROOT*'

Директория корня документов, в которой выполняется текущий скрипт, в точности та, которая указана в конфигурационном файле сервера.

'*HTTP\_ACCEPT*'

Содержимое заголовка *Accept:* из текущего запроса, если он есть.

'*HTTP\_ACCEPT\_CHARSET*'

Содержимое заголовка *Accept-Charset:* из текущего запроса, если он есть. Например: '*iso-8859-1,\*,utf-8*'.

'*HTTP\_ACCEPT\_ENCODING*'

Содержимое заголовка *Accept-Encoding:* из текущего запроса, если он есть. Например: '*gzip*'.

'*HTTP\_ACCEPT\_LANGUAGE*'

Содержимое заголовка *Accept-Language:* из текущего запроса, если он есть. Например: '*en*'.

'*HTTP\_CONNECTION*'

Содержимое заголовка *Connection:* из текущего запроса, если он есть. Например: '*Keep-Alive*'.

'*HTTP\_HOST*'

Содержимое заголовка *Host:* из текущего запроса, если он есть.

'*HTTP\_REFERER*'

Адрес страницы (если есть), с которой браузер пользователя перешёл на эту страницу. Этот заголовок устанавливается браузером пользователя. Не все браузеры устанавливают его, а некоторые в качестве дополнительной возможности позволяют изменять содержимое заголовка *HTTP\_REFERER*. Одним словом, нельзя доверять этому заголовку.

'*HTTP\_USER\_AGENT*'

Содержимое заголовка *User-Agent:* из текущего запроса, если он есть. Эта строка, обозначающая браузер, который открыл данную страницу. Типичным примером является строка: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Помимо всего прочего, вы можете использовать это значение с функцией [get\_browser()](https://www.php.net/manual/ru/function.get-browser.php), чтобы адаптировать вывод вашей страницы к возможностям браузера пользователя.

'*HTTPS*'

Принимает непустое значение, если запрос был произведен через протокол HTTPS.

**Замечание**: Обратите внимание, что при использовании ISAPI с IIS значение будет *off*, если запрос не был произведен через протокол HTTPS.

'*REMOTE\_ADDR*'

IP-адрес, с которого пользователь просматривает текущую страницу.

'*REMOTE\_HOST*'

Удаленный хост, с которого пользователь просматривает текущую страницу. Обратный поиск DNS основан на значении переменной *REMOTE\_ADDR*.

**Замечание**: Ваш сервер должен быть настроен, чтобы создавать эту переменную. Для примера, в Apache вам необходимо присутствие директивы *HostnameLookups On* в файле *httpd.conf*, чтобы эта переменная создавалась. См. также [gethostbyaddr()](https://www.php.net/manual/ru/function.gethostbyaddr.php).

'*REMOTE\_PORT*'

Порт на удаленной машине, который используется для связи с сервером.

'*REMOTE\_USER*'

Аутентифицированный пользователь.

'*REDIRECT\_REMOTE\_USER*'

Аутентифицированный пользователь, если запрос был перенаправлен изнутри.

'*SCRIPT\_FILENAME*'

Абсолютный путь к исполняемому скрипту.

**Замечание**: Если скрипт запускается в командной строке (CLI), используя относительный путь, такой как *file.php* или *../file.php*, переменная *$\_SERVER['SCRIPT\_FILENAME']* будет содержать относительный путь, указанный пользователем.

'*SERVER\_ADMIN*'

Эта переменная получает свое значение (для Apache) из директивы конфигурационного файла сервера. Если скрипт запущен на виртуальном хосте, это будет значение, определенное для данного виртуального хоста.

'*SERVER\_PORT*'

Порт на компьютере сервера, используемый сервером для соединения. Для установок по умолчанию, значение будет '*80*'; используя SSL, например, это значение будет таким, какое сконфигурировано для соединений безопасного HTTP.

**Замечание**: Чтобы получить физический (реальный) порт в Apache 2, необходимо установить *UseCanonicalName = On* и *UseCanonicalPhysicalPort = On*, иначе это значение может быть подменено и не вернуть реальной значение физического порта. Полагаться на это значение небезопасно в контексте приложений, требующих усиленной безопасности.

'*SERVER\_SIGNATURE*'

Строка, содержащая версию сервера и имя виртуального хоста, которые добавляются к генерируемым сервером страницам, если включено.

'*PATH\_TRANSLATED*'

Путь файловой системы (не document root) к текущему скрипту, после того как сервер выполнил отображение virtual-to-real.

**Замечание**: Начиная с PHP 4.3.2, переменная *PATH\_TRANSLATED* больше не устанавливается неявно в Apache 2 SAPI, по сравнению с Apache версии 1, где она устанавливается в то же самое значение, что и переменная *SCRIPT\_FILENAME*, когда она не используется Apache. Это изменение было сделано для соответствия спецификации CGI, где переменная *PATH\_TRANSLATED* должна существовать только тогда, когда *PATH\_INFO* определена. Пользователи Apache 2 могут использовать директиву *AcceptPathInfo = On* в конфигурационном файле *httpd.conf* для задания переменной *PATH\_INFO*.

'*SCRIPT\_NAME*'

Содержит путь к текущему исполняемому скрипту. Это полезно для страниц, которые должны указывать на самих себя. Константа [\_\_FILE\_\_](https://www.php.net/manual/ru/language.constants.predefined.php) содержит полный путь и имя текущего (то есть включенного) файла.

'*REQUEST\_URI*'

URI, который был предоставлен для доступа к этой странице. Например, '*/index.html*'.

'*PHP\_AUTH\_DIGEST*'

При выполнении аутентификации HTTP Digest этой переменной присваивается заголовок 'Authorization', отправленный клиентом (который затем следует использовать для проведения соответствующей проверки).

'*PHP\_AUTH\_USER*'

При выполнении HTTP-аутентификации этой переменной присваивается имя пользователя, предоставленное пользователем.

'*PHP\_AUTH\_PW*'

При выполнении HTTP-аутентификации этой переменной присваивается пароль, предоставленный пользователем.

'*AUTH\_TYPE*'

При выполнении HTTP-аутентификации этой переменной присваивается тип аутентификации, который используется.

'*PATH\_INFO*'

Содержит любой предоставленный пользователем путь, содержащийся после имени скрипта, но до строки запроса, если она есть. Например, если текущий скрипт запрошен по URL *http://www.example.com/php/path\_info.php/some/stuff?foo=bar*, то переменная *$\_SERVER['PATH\_INFO']* будет содержать */some/stuff*.

'*ORIG\_PATH\_INFO*'

Исходное значение переменной '*PATH\_INFO*' перед обработкой PHP.

Пример 5.1

        //Определим url который был предоставлен для доступа к этой странице

        $url=$\_SERVER['REQUEST\_URI'];

6) **preg\_match**

(PHP 4, PHP 5, PHP 7)

preg\_match — Выполняет проверку на соответствие регулярному выражению

**Описание**

**preg\_match** ( string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0 ]]] ) : int

Ищет в заданном тексте subject совпадения с шаблоном pattern.

**Список параметров**

*pattern*

Искомый шаблон в виде строки.

*subject*

Входная строка.

*matches*

В случае, если указан дополнительный параметр matches, он будет заполнен результатами поиска. Элемент *$matches[0]* будет содержать часть строки, соответствующую вхождению всего шаблона, *$matches[1]* - часть строки, соответствующую первой подмаске и так далее.

*flags*

flags может быть комбинацией следующих флагов:

PREG\_OFFSET\_CAPTURE

В случае, если этот флаг указан, для каждой найденной подстроки будет указана ее позиция (в байтах) в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемого массива matches в массив, каждый элемент которого содержит массив, содержащий в индексе с номером *0* найденную подстроку, а смещение этой подстроки в параметре subject - в индексе *1*.

PREG\_UNMATCHED\_AS\_NULL

Если этот флаг передан, несовпадающие подмаски будут представлены значениями NULL; в противном случае они отображаются в виде пустых строк ([string](https://www.php.net/manual/ru/language.types.string.php)).

*offset*

Обычно поиск осуществляется слева направо, с начала строки. Можно использовать дополнительный параметр offset для указания альтернативной начальной позиции для поиска (в байтах).

### Возвращаемые значения

**preg\_match()** возвращает 1, если параметр pattern соответствует переданному параметру subject, 0 если нет, или **FALSE** в случае ошибки.

**8) ucfirst**

(PHP 4, PHP 5, PHP 7)

ucfirst — Преобразует первый символ строки в верхний регистр

**Описание**

**ucfirst** ( string $str ) : string

Возвращает строку str, в которой первый символ переведен в верхний регистр, если этот символ является буквой.

Принадлежность того или иного символа к буквенным определяется с учетом текущей локали. Это означает, что, например, в используемой по умолчанию локали "C", символ ä не будет преобразован.

**Список параметров**

str

Входная строка.

**Возвращаемые значения**

Возвращает результирующую строку.

Пример 8.1

<?php  
$foo = 'hello world!';  
$foo = ucfirst($foo);             // Hello world!  
  
$bar = 'HELLO WORLD!';  
$bar = ucfirst($bar);             // HELLO WORLD!  
$bar = ucfirst(strtolower($bar)); // Hello world!  
?>

Пример 8.2

//Заносим переменную $path формируемый динамически класс. Функция ucfirst() делает заглавным первый символ аргумента

 $path='application\controllers\\'.ucfirst($this->params['controller']).'Controller';