

# Суперглобальные переменные

- Введение
- Суперглобальные массивы
- Кодирование URL-строк
- Заключение

## Введение

Некоторые предопределённые переменные в PHP являются **суперглобальными**, это означает, что они доступны в любом месте скрипта. Вы можете получить к ним доступ из любой функции, класса или файла без необходимости делать что-либо особенное.

**Важно.** Нет необходимости использовать синтаксис **global \$var** для доступа к ним в функциях и методах.

Суперглобальными переменными являются:

- **`$_GET`**
- **`$_POST`**
- **`$_FILES`**
- **`$_REQUEST`**
- **`$_GLOBALS`**
- **`$_SERVER`**
- **`$_COOKIE`**
- **`$_SESSION`**
- **`$_ENV`**

**Примечание.** С некоторыми массивами мы уже работали. Массивы **`$_COOKIE`** и **`$_SESSION`** будут рассматриваться в отдельных .

## **\$\_GET**

**\$\_GET** — переменные **HTTP GET**.

**\$\_GET** — **ассоциативный массив переменных**, переданных скрипту через **параметры URL** (известные также как **строка запроса**).

Массив **\$\_GET** может быть использован для **сбора пользовательских данных** после отправки HTML формы с атрибутом **method="get"**.

Обратите внимание, массив заполняется не только для GET-запросов формы, а скорее для всех запросов, в которых присутствует **строка запроса**.

**Еще раз.** В массив **\$\_GET** попадают данные, отправленные в строке запроса (URL).

### **example\_1. index.php**

```
<form action="server.php" method="get">

    Логин: <input type="text" name="login" value="master"><p>

    Е-mail: <input type="email" name="email"
        value="master@ya.ru"><p>

    Пароль: <input type="password" name="pwd" value="12345"><p>

    <p>
        <input type="reset" value="Очистить">
        <button>Отправить</button>
    </p>
</form>
```

### **example\_1. server.php**

```
<?php
```

```
// выводим данные массива $_GET  
  
echo "<pre>";  
  
print_r($_GET);  
  
echo "</pre>";  
  
?>
```

Массив **\$\_GET** подробно рассмотрен в предыдущих .

## **\$\_POST**

**\$\_POST** — переменные **HTTP POST**.

Массив **\$\_POST** — широко используется для **сбора пользовательских данных** после отправки формы HTML с атрибутом **method="post"**.

**\$\_POST** представляет собой **ассоциативный массив переменных**, переданных скрипту методом POST при использовании свойства **enctype**:

- **application/x-www-form-urlencoded** – используется по умолчанию
- **multipart/form-data** – используется при отправке файлов

в заголовке Content-Type запроса HTTP.

Когда пользователь отправляет данные, нажав кнопку **Submit**, данные формы отправляются в файл (обработчик формы), указанный в атрибуте действия (**action**) тега **form**.

**\$\_POST** также широко используется для **передачи значений переменных** как в обычных, так и в скрытых полях. Совместно с форматом **JSON** очень часто применяется при организации многостраничных форм.

**Важно.** Атрибут **method="post"** не отменяет возможности добавления

данных в строку запроса (URL).

### example\_2. index.php

```
<form action="server.php?course=programmirovaniye-na-yazyke-
php&topic=otpravka-dannyh-na-server&lesson=zagruzka-fajlov"
method="post">

    Логин: <input type="text" name="login" value="master"><p>

    Е-mail: <input type="email" name="email"
    value="master@ya.ru"><p>

    Пароль: <input type="password" name="pwd" value="12345"><p>

    <p>
        <input type="reset" value="Очистить">
        <button>Отправить</button>
    </p>
</form>
```

### example\_2. server.php

```
<h2>Суперглобальный массив $_POST</h2>

<?php
    // выводим данные массива $_POST
    echo "<pre>";
    print_r($_POST);
    echo "</pre>";
?>

<h2>Суперглобальный массив $_GET</h2>

<?php
    // выводим данные массива $_GET
    echo "<pre>";
```

```
print_r($_GET);

echo "</pre>";

?>
```

Массив `$_POST` подробно рассмотрен в предыдущих .

## **`$_FILES`**

`$_FILES` — переменные файлов, загруженных по **HTTP**. Массив `$_FILES` – глобальная переменная в PHP наподобие `$_POST` или `$_GET`.

`$_FILES` представляет собой **ассоциативный массив переменных**, загруженных в текущий скрипт через метод HTTP POST. Используется для отправки пользовательских файлов на сервер.

**Важно.** Данные попадут в массив `$_FILES` только при установке свойства формы `enctype` в значение `multipart/form-data`.

В следующем демонстрационном примере рассмотрим форму одновременной передачи данных нескольких массивов.

### **example\_3. index.php**

```
<form action="server.php?course=programmirovaniye-na-yazyke-
php&topic=otpravka-danniy-na-server&lesson=zagruzka-fajlov"
method="post" enctype="multipart/form-data">

    Логин: <input type="text" name="login" value="master"><p>

    E-mail: <input type="email" name="email"
    value="master@ya.ru"><p>

    Пароль: <input type="password" name="pwd" value="12345"><p>

    Выберите файл: <input type="file" name="file"><p>

    <p>
```

```
<input type="reset" value="Очистить">  
<button>Отправить</button>  
</form>
```

### example\_3. server.php

```
<h2>Суперглобальный массив $_FILES</h2>  
  
<?php  
  
    // выводим данные массива $_FILES  
  
    echo "<pre>";  
  
    print_r($_FILES);  
  
    echo "</pre>";  
  
?>  
  
<h2>Суперглобальный массив $_POST</h2>  
  
<?php  
  
    // выводим данные массива $_POST  
  
    echo "<pre>";  
  
    print_r($_POST);  
  
    echo "</pre>";  
  
?>  
  
<h2>Суперглобальный массив $_GET</h2>  
  
<?php  
  
    // выводим данные массива $_GET  
  
    echo "<pre>";  
  
    print_r($_GET);  
  
    echo "</pre>";  
  
?>
```

Массив **\$\_FILES** подробно рассмотрен в предыдущих .

## **\$\_REQUEST**

**\$\_REQUEST** — переменные **HTTP-запроса**.

**\$\_REQUEST** — **ассоциативный массив**, который по умолчанию содержит данные переменных **\$\_GET**, **\$\_POST** и **\$\_COOKIE**.

В примере **example\_4** рассмотрим, как это работает.

### **example\_4. index.php**

```
<form action="server.php?course=programmirovaniye-na-yazyke-
php&topic=otpravka-dannyh-na-server&lesson=zagruzka-fajlov"
method="post">

    Логин: <input type="text" name="login" value="master"><p>

    E-mail: <input type="email" name="email"
    value="master@ya.ru"><p>

    Пароль: <input type="password" name="pwd" value="12345"><p>

    <p>
        <input type="reset" value="Очистить">
        <button>Отправить</button>
    </p>
</form>
```

В обработчике формы можно не делить полученные данные на массивы **\$\_GET** и **\$\_POST**, а воспользоваться суперглобальным массивом **\$\_REQUEST**.

### **example\_4. server.php**

```
<?php
    // выводим данные массива $_REQUEST
    echo "<pre>";
    print_r($_REQUEST);
    echo "</pre>";
```

```
echo '<h2>Суперглобальный массив $_GET</h2>' ;  
  
// выводим данные массива $_GET  
  
echo "<pre>";  
  
print_r($_GET);  
  
echo "</pre>";  
  
echo '<h2>Суперглобальный массив $_POST</h2>' ;  
  
// выводим данные массива $_POST  
  
echo "<pre>";  
  
print_r($_POST);  
  
echo "</pre>";  
  
?>
```

**Замечание.** Спорный вариант обработки данных форм. Я предпочитаю из **контекста кода** определять, с данными какого происхождения имею дело.

**Важно.** В случае совпадения имен ключей массивов **\$\_POST** и **\$\_GET** в массив **\$\_REQUEST** попадет только один из них (что логично).

## \$GLOBALS

Массив **\$GLOBALS** — суперглобальная переменная PHP, которая используется для доступа к глобальным переменным из любого места в PHP скрипте (внутри функций или методах).

**Ассоциативный массив \$GLOBALS**, содержит ссылки на все переменные, определённые в данный момент в глобальной области

видимости скрипта. **Имена переменных являются ключами массива.**

**Важно.** Массив **\$GLOBALS** содержит переменные, определенные только в **глобальной** области видимости.

В демонстрационном примере **example\_5** я рассмотрел три основные возможности передачи переменных в пользовательскую функцию. Каждый из способов имеет право на существование, но я бы выбрал переменную-параметр, классический способ работы с функциями.

### **example\_5. index.php**

```
<?php

    // инициируем переменные глобальной области видимости

    $name = "Eugene";
    $login = "daugava";
    $email = "pechora_pro@mail.ru";
    $role = "SuperBombaAdmin";

    // определение функции с параметром

    function testScopeVariables ($login="NoNickName") {

        // глобальная переменная

        global $name;

        // вывод глобальной переменной

        echo 'Имя: <b>' . $name . '</b><br>';
        // вывод переменной-параметра

        echo 'Логин: <b>' . $login . '</b><br>';
        // вывод значений глобального массива $GLOBALS

        echo "Email: <b>" . $_GLOBALS['email'] . "</b><br>";
        echo "Роль: <b>" . $_GLOBALS['role'] . "</b><br>";
```

```
}

// вызов функции

testScopeVariables ($login);

?>
```

Если вы не разобрались с переменными, которые будут попадать в массив, следующий пример расставит все по местам.

### example\_6. index.php

```
<?php

// определение функции

function testScopeVariables () {

    // объявляем переменные локальной области видимости

    $name = "Eugene";

    $login = "daugava";

    $email = "pechora_pro@mail.ru";

    $role = "SuperBombaAdmin";

}

// вызов функции

testScopeVariables ();

// вывод данных глобального массива

echo "name: ", $GLOBALS['name'], "<br>";

echo "login: ", $GLOBALS['login'], "<br>";

echo "email: ", $GLOBALS['email'], "<br>";

echo "role: ", $GLOBALS['role'], "<br>";

?>
```

Так как переменные объявлены в локальной области видимости функции, массив **\$GLOBALS** перечисленных ключей не содержит.

С другой стороны, если в локальной области записать данные в глобальный массив напрямую, все работает.

### **example\_7. index.php**

```
<?php

    // определение функции

    function testScopeVariables () {

        // записываем данные в глобальный массив

        $GLOBALS ['name'] = "Eugene";

        $GLOBALS ['login'] = "daugava";

        $GLOBALS ['email'] = "pechora_pro@mail.ru";

        $GLOBALS ['role'] = "SuperBombaAdmin";

    }

    // вызов функции

    testScopeVariables ();

    // вывод данных глобального массива

    echo "name: ", $GLOBALS ['name'], "<br>";

    echo "login: ", $GLOBALS ['login'], "<br>";

    echo "email: ", $GLOBALS ['email'], "<br>";

    echo "role: ", $GLOBALS ['role'], "<br>";

?>
```

При необходимости создания более сложных структур можно использовать двумерный массив.

```
// определение функции

function fnTestGlobals () {

    // записываем данные в двумерный глобальный массив

    $GLOBALS ['user'] [] = "Eugene";
```

```
$GLOBALS['user'][] = "daugava";  
$GLOBALS['user'][] = "pechora_pro@mail.ru";  
$GLOBALS['user'][] = "SuperBombaAdmin";  
}
```

В предыдущих разделах я писал, что рассматриваемые массивы являются переменными **HTTP-запроса**. Это значит, что они формируются автоматически из запроса и присваивать значения им напрямую не имеет смысла.

**Примечание.** Значения присвоить можно, но сервер перезапишет их при первом же HTTP-запросе.

Таким образом, сервер воспримет такие массивы как обычные **пользовательские массивы** с именами, совпадающими с именами зарезервированных глобальных массивов.

### example\_8. index.php

```
<?php  
  
    // присваиваем массивам значения  
  
    // массивы названы именами, совпадающими с глобальными  
    // массивами  
  
    $_GET['course'] = "programmirovaniye-na-yazyke-php";  
    $_POST['topic'] = "otpravka-dannyh-na-server";  
    $_REQUEST['lesson'] = "zagruzka-fajlov";  
  
    $GLOBALS['email'] = "pechora_pro@mail.ru";  
  
    // в массиве то, что ему присвоили (не более того)  
  
    echo "<hr>";  
  
    echo "<pre>";  
  
    print_r($_REQUEST);
```

```
echo "</pre>";

?>

<form action="server.php?name=den&role=sa" method="post">

    Логин: <input type="text" name="login" value="master"><p>

    Пароль: <input type="password" name="pwd" value="12345"><p>

    <p>

    <input type="reset" value="Очистить">

    <button>Отправить</button>

</form>
```

Кроме того, созданные глобальные массивы не окажут никакого влияния на массивы, созданные формой. Они существуют в разные моменты времени, в разных файлах, в разных сценариях.

### **example\_8. server.php**

```
<?php

    // все данные формы в одном массиве

    echo "<pre>";

    print_r($_REQUEST);

    echo "</pre>";

    echo "<hr>";

?>
```

На рисунке представлен скриншот HTML-разметки файла, загруженного пользователю. Никаких массивов здесь нет.

```
<!DOCTYPE html>
<html lang="ru">
... ▶ <head> ... </head> == $0
▼ <body>
  <h1>Безопасность приложений</h1>
  <hr>
  <pre>Array ( [lesson] => загрузка-файл ) </pre>
▶ <form action="server.php?name=den&role=sa" method="post"> ... </form>
▶ <footer align="center"> ... </footer>
</body>
</html>
```

Переменные массивов будут сформированы при отправке формы на обработчик **server.php**.

**Примечание.** Сказанное выше просто, но зачастую именно здесь возникает недопонимание процессов у начинающих разработчиков.

В примере **example\_9** классический способ использования глобальной переменной внутри пользовательской функции.

### **example\_9. index.php**

```
<?php

// подключаем файл, получаем данные массива $person

require 'person.php';

// определение функции поиска актуальных преподавателей

function fnOutActualPerson() {

    // поиск и вывод актуальных преподавателей

    foreach ($GLOBALS['person'] as $item) {

        if ($item['actual'] == 1) {

            echo <<<HERE

            <h3>{$item['surname']} {$item['name']}
            {$item['patronymic']}</h3>

            Должность: {$item['post']}
```

```

        Категория: {$item['category']}</br>
        Уровень образования: {$item['level_edu']}</br>
        Трудовой стаж: {$item['experience_total']}</br>
        Стаж работы в ОУ:
        {$item['experience_college']}</br>
        e-mail: {$item['email']}</br>
        Сайт: {$item['site']}
        <hr>
        HERE;
    }
}
}

// вызов функции
fnOutActualPerson();
?
```

## Массивы \$\_GET и \$\_POST в \$GLOBALS

Ассоциативный массив **\$GLOBALS**, содержит ссылки на все переменные, определённые в глобальной области видимости. А это значит, что массивы **\$\_GET** и **\$\_POST** также **доступны в \$GLOBALS**.

### example\_10. index.php

```

<form action="server.php?course=programmirovaniye-na-yazyke-
php&topic=otpravka-dannyh-na-server&lesson=zagruzka-fajlov"
method="post">

    Логин: <input type="text" name="login" value="master"><p>
    E-mail: <input type="email" name="email"
    value="master@ya.ru"><p>
```

```
Пароль: <input type="password" name="pwd" value="12345"><p>
<p>
<input type="reset" value="Очистить">
<button>Отправить</button>
</form>
```

### example\_10. server.php

```
<?php
    // выводим данные массива $GLOBALS
    echo "<pre>";
    print_r($GLOBALS);
    echo "</pre>";
    echo '<h2>Данные массива $_GET в $GLOBALS</h2>';
    // выводим данные массива $_GET
    echo "<pre>";
    print_r($_GET);
    echo "</pre>";
    echo '<h2>Данные массива $_POST в $GLOBALS</h2>';
    // выводим данные массива $_POST
    echo "<pre>";
    print_r($_POST);
    echo "</pre>";
?>
```

Обратите внимание, что массивы **\$\_FILES** и **\$\_COOKIE** также входят в **\$GLOBALS**.

## **\$\_SERVER**

Массив **`$_SERVER`** — информация о сервере и среде исполнения.

**`$_SERVER`** — **ассоциативный массив**, который содержит сведения о заголовках, пути и расположении сценариев.

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.

Далее перечислю **наиболее используемые элементы**, которые могут находиться внутри массива **`$_SERVER`**:

**`$_SERVER['SERVER_NAME']`** — имя хоста, на котором выполняется текущий скрипт.

**`$_SERVER['PHP_SELF']`** — имя файла скрипта, который сейчас выполняется, относительно корня документов.

**`$_SERVER['REQUEST_METHOD']`** — какой метод был использован для запроса страницы; к примеру 'GET', 'HEAD', 'POST', 'PUT'.

**`$_SERVER['QUERY_STRING']`** — строка запроса, если есть, через которую была открыта страница.

**`$_SERVER['SCRIPT_FILENAME']`** — абсолютный путь к исполняемому скрипту.

**`$_SERVER['DOCUMENT_ROOT']`** — директория корня документов, в которой выполняется текущий скрипт, в точности та, которая указана в конфигурационном файле сервера.

**`$_SERVER['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`. Доверять этому заголовку нельзя.

**`$_SERVER['REQUEST_URI']`** – URI, который был предоставлен для доступа к этой странице. Например, '/index.html'.

### **example\_11. index.php**

```
<?php

    // имя хоста, на котором выполняется текущий скрипт

    echo '$_SERVER[\\' SERVER_NAME\'] => ',  

        $_SERVER['SERVER_NAME'];

    echo "<p>";

    // имя файла скрипта, который сейчас выполняется,  

    // относительно корня документов

    echo '$_SERVER[\\' PHP_SELF\'] => ', $_SERVER['PHP_SELF'];

    echo "<p>";

    // какой метод был использован для запроса страницы: GET,  

    HEAD, POST, PUT

    echo '$_SERVER[\\' REQUEST_METHOD\'] => ',  

        $_SERVER['REQUEST_METHOD'];

    echo "<p>";

    // строка запроса, если есть, через которую была открыта  

    // страница

    echo '$_SERVER[\\' QUERY_STRING\'] => ',  

        $_SERVER['QUERY_STRING'];

    echo "<p>";

    // абсолютный путь к исполняемому скрипту

    echo '$_SERVER[\\' SCRIPT_FILENAME\'] => ',  

        $_SERVER['SCRIPT_FILENAME'];

    echo "<p>";

    // директория корня документов, в которой выполняется текущий  

    // скрипт, в точности та, которая указана в конфигурационном  

    // файле сервера
```

```

echo '$_SERVER[\'DOCUMENT_ROOT\'] => ',  

$_SERVER['DOCUMENT_ROOT'];  
  

echo "<p>";  
  

// адрес страницы (если есть), с которой браузер пользователя  

перешёл на эту страницу. Устанавливается браузером  

пользователя, устанавливается не всеми браузерами  
  

echo '$_SERVER[\'HTTP_REFERER\'] => ',  

$_SERVER['HTTP_REFERER'];  
  

echo "<p>";  
  

// URI для доступа к текущей странице  
  

echo '$_SERVER[\'REQUEST_URI\'] => ',  

$_SERVER['REQUEST_URI'];  
  

?>  
  

<p>  

Просмотр глобальных переменных в  

сценарии  

server.php  

</p>

```

Сценарий файла **server.php** повторяет сценарий файла **index.php**, а вот значения некоторых глобальных переменных массива будут отличаться.

Из всего набора переменных массива **\$\_SERVER** наиболее часто в веб-разработке будут использоваться (скорее всего) переменные следующих ключей массива:

- **QUERY\_STRING**
- **REQUEST\_URI**

#### **example\_12. index.php**

```

<?php  

// тестируем ключ QUERY_STRING

```

```

echo '$_SERVER[ \'QUERY_STRING\' ] => ',  

$_SERVER[ 'QUERY_STRING' ];  

  

echo "<p>";  

// тестируем ключ REQUEST_URI  

  

echo '$_SERVER[ \'REQUEST_URI\' ] => ',  

$_SERVER[ 'REQUEST_URI' ];  

  

?>

```

### **example\_12. server.php**

```

<?php  

  

// тестируем ключ QUERY_STRING  

  

echo '$_SERVER[ \'QUERY_STRING\' ] => ',  

$_SERVER[ 'QUERY_STRING' ];  

  

echo "<p>";  

  

// тестируем ключ REQUEST_URI  

  

echo '$_SERVER[ \'REQUEST_URI\' ] => ',  

$_SERVER[ 'REQUEST_URI' ];  

  

echo '<h2>Суперглобальный массив $_GET</h2>';  

  

echo "<pre>";  

  

print_r($_GET);  

  

echo "</pre>";  

  

echo '<h2>URI страницы ($_SERVER[ \'REQUEST_URI\' ])</h2>';  

  

print($_SERVER[ 'REQUEST_URI' ]);  

  

?>

```

**Еще раз.** Записи суперглобального массива **\$\_SERVER** создаются веб-сервером и нет гарантии, что все веб-серверы предоставят любую из

этих записей.

Однако, многие из этих переменных присутствуют в спецификации CGI/1.1, поэтому разработчик вправе ожидать их поддержку сервером.

В некотором смысле заменой глобальному массиву `$_SERVER` может стать функция **parse\_url()** и если сервер не предоставил какую-либо информацию, ее можно получить используя эту функцию.

## **parse\_url**

**parse\_url ()** — разбирает URL и возвращает его компоненты.

### **Описание**

```
parse_url($url, $component = -1);
```

Эта функция разбирает **URL** и возвращает **ассоциативный массив**, содержащий все компоненты URL, которые в нём присутствуют. Элементы массива не будут декодированы как URL.

Эта функция не предназначена для проверки на корректность данного URL, она только разбивает его на нижеперечисленные части. Частичные и недопустимые URL также принимаются, **parse\_url()** пытается сделать всё возможное, чтобы разобрать их корректно.

### **Параметр**

- **url** – URL для разбора.
- **component** – укажите одну из констант PHP\_URL\_SCHEME, PHP\_URL\_HOST, PHP\_URL\_PORT, PHP\_URL\_USER, PHP\_URL\_PASS, PHP\_URL\_PATH, PHP\_URL\_QUERY или PHP\_URL\_FRAGMENT, чтобы получить только конкретный компонент URL в виде **строки** (string). Исключением является указание PHP\_URL\_PORT, в этом случае возвращаемое значение будет числовым (int).

### **Возвращаемые значения**

При разборе значительно некорректных URL-адресов **parse\_url()** может вернуть **false**.

Если параметр **component** будет опущен, функция возвратит **ассоциативный массив** (array). В массиве будет находиться по крайней мере один элемент. Возможные ключи в этом массиве:

- **scheme** (например, http)
- **host**
- **port**
- **user**
- **pass**
- **path**
- **query** (после знака вопроса ?)
- **fragment** (после знака решётки #)

**Важно.** **Фрагмент** (fragment) **не отсылается** браузером на сервер.

Более подробно рассмотрим в следующих примерах.

Если параметр **component** определён, функция **parse\_url()** вернёт строку (string) (или число (int), в случае PHP\_URL\_PORT) вместо массива (array). Если запрошенный компонент не существует в данном URL, будет возвращён **null**.

Функция **parse\_url()** **различает отсутствующие и пустые запросы и фрагменты**.

В демонстрационном примере **example\_13** рассмотрим возможности функции **parse\_url()**. Обратите внимание на передачу **фрагмента** строки запроса.

### **example\_13. index.php**

```
<h2>Функция parse_url()</h2>
<h3>Перейдите по ссылке для анализа работы функции</h3>
```

```

<!--

    обратите внимание, строка запроса ссылки содержит фрагмент
    (fragment) - #fragment

-->

<p><a href="server.php?login=master&role=admin&topic=otpravka-
dannyh-na-server&lesson=5#fragment">Просмотреть переменные на
странице server.php</a></p>

```

### **example\_13. server.php**

```

<?php

    // вариант 1

    // анализируем результат работы функции parse_url

$arr = parse_url($_SERVER['REQUEST_URI']);

    // вариант 2

    // искусственно прицепим к строке фрагмент

    // $arr = parse_url($_SERVER['REQUEST_URI'] . "#id-11");

    // рассмотри ключи ассоциативного массива $arr

echo 'path = ' . $arr['path'] . '<p>';

echo 'query = ' . $arr['query'] . '<p>';

echo 'fragment = ';

if (empty($arr['fragment'])) {

    echo 'Нет данных';

} else

    echo $arr['fragment'];

    // выведем содержимое массива

    // echo "<pre>";

    // print_r($arr);

```

```
// echo "</pre>";  
?>
```

## Хеш-ссылки

Фрагмент, о котором я говорю, это так называемая **хеш-ссылка**.

**Хеш-ссылками** называются ссылки, на конце которых стоит знак диеза (или хеш-символ, от англ. hash) **#** с определенной последовательностью символов после него.

**Хеш-ссылки** позволяют организовать **навигацию** внутри одной страницы, когда на ней представлен большой структурированный материал, и часто используются при размещении содержаний статей.

Например, с помощью ссылки:

[http://pechora\\_pro.ru/about/#contacts](http://pechora_pro.ru/about/#contacts)

вы перейдете не только на страницу "Об авторе", но также браузер **автоматически прокрутит** окно до нужного раздела, отмеченного идентификатором **contacts** (`id='contacts'`).

Поисковые системы при анализе ссылок на странице автоматически отбрасывают всё, что находится после символа хеша, **с ним работает только браузер**. Поэтому, следующие ссылки, с точки зрения поисковых систем, будут совершенно **одинаковыми**:

- [http://pechora\\_pro.ru/about/#contacts](http://pechora_pro.ru/about/#contacts)
- [http://pechora\\_pro.ru/about](http://pechora_pro.ru/about)

## **\$\_COOKIE**

**\$\_COOKIE** (куки) представляют небольшие наборы данных, с помощью которых веб-сайт может сохранить на компьютере пользователя любую информацию.

**\$\_COOKIE ассоциативный массив переменных**, переданных скрипту через HTTP COOKIES.

С помощью файлов куки можно отслеживать активность пользователя на сайте: зарегистрирован пользователь на сайте или нет, отслеживать историю его визитов и т.д.

**Примечание.** Массиву **\$\_COOKIE** посвящены отдельные темы.

## **\$\_SESSION**

**\$\_SESSION** — переменные сессии.

**\$\_SESSION – ассоциативный массив**, содержащий переменные сессии, которые доступны для текущего скрипта.

**Примечание.** Массиву **\$\_SESSION** посвящены отдельные темы.

**Примечание.** Массивы **\$\_COOKIE** и **\$\_SESSION** лежат в основе любой **аутентификации пользователя** и являются важнейшим материалом курса.

## **Кодирование URL-строк**

По историческим причинам URL-адреса можно отправлять только **с помощью набора символов ASCII**. Поскольку URL-адреса часто

содержат символы вне набора ASCII, адрес должен быть преобразован в **правильный формат ASCII**.

В любом языке программирования существуют функции, которые можно использовать для URL-кодирования/декодирования строк.

**Примечание.** В предыдущих для передачи данных в строке запроса я использовал, как правило, набор символов английского алфавита. Но так бывает далеко не всегда и использовать **символы русского алфавита** – обычная практика.

Кодирование URL заменяет опасные символы ASCII на "%", за которыми следуют две шестнадцатеричные цифры.

URL-адреса не могут содержать пробелы. Кодирование URL обычно заменяет пробел знаком плюс (+) или %20.

## **urlencode**

**urlencode ()** – URL-кодирование строки.

### **Описание**

**urlencode (\$string) ;**

Эта функция удобна, когда закодированная строка будет использоваться в запросе, как часть URL, в качестве удобного способа передачи переменных на следующую страницу.

### **Параметры**

- **string** – строка, которая должна быть закодирована.

### **Возвращаемые значения**

Возвращает строку, в которой все не цифробуквенные символы (кроме -\_. ) должны быть заменены знаком процента (%), за которым следует два

шестнадцатеричных числа, а пробелы закодированы как знак сложения (+).

Строка кодируется тем же способом, что и POST-данные веб-формы, то есть по типу контента **application/x-www-form-urlencoded**.

## urlencode

**urlencode ()** — декодирование URL-кодированной строки.

### Описание

**urlencode** (string \$string) ;

Декодирует любые кодированные последовательности %## в данной строке. Символ "плюс" ('+') декодируется в символ пробела.

### Параметры

- **string** – строка, которая должна быть декодирована.

### Возвращаемые значения

Возвращает декодированную строку.

Кодирование имеет некоторые особенности. Разбираем следующий демонстрационный пример.

### example\_14. index.php

```
<h2>Кодирование URL-строк</h2>

<?php

    echo '<h3>Переменная $par1:</h3>' ;
    echo $par1 = 'course=' . urlencode('Программирование на языке
    PHP') ;
    echo '<h3>Переменная $par2:</h3>' ;
    echo $par2 = urlencode('course=Программирование на языке')
```

```
    PHP' );
?>

<h3>Просмотреть значения переменных после декодирования:</h3>
<a href="server.php?<?= $par1; ?>">Просмотреть данные переменной
$par1</a><p>
<a href="server.php?<?= $par2; ?>">Просмотреть данные переменной
$par2</a><p>
```

#### **example\_14. server.php**

```
<?php
    // выводим данные массива $_GLOBALS['_GET']
    echo "<pre>";
    print_r($_GLOBALS['_GET']);
    echo "</pre>";
?>
```

## **Заключение**

---

сложн , но рассматриваемые вопросы являются базовыми при работе с суперглобальными переменными. В следующих – подробное изучение массивов **\$\_COOKIE** и **\$\_SESSION** без которых невозможна безопасная работа современных веб-приложений.

В заключении приведу небольшой пример, суммирующий практические навыки текуще .

#### **example\_15. index.php**

```
<?php
```

```

include "person.php";

include "function.php";

$id = 2;

$person = fnSearchPerson();

$json_person = json_encode($person, JSON_UNESCAPED_UNICODE);

$encode_json_person = urlencode($json_person);

echo '<h3>$person:</h3>';

echo '<pre>';

print_r($person);

echo '</pre>';

echo '<h3>$json_person:</h3>';

echo $json_person;

echo '<p>';

echo '<h3>$encode_json_person:</h3>';

echo $encode_json_person;

// добавляем кодированную json-строку к ссылке

echo "<p><a href='server.php?" . $encode_json_person .

"!>Переход для просмотра</a></p>";

?>

```

### **example\_15. function.php**

```

<?php

function fnSearchPerson () {

    // перебор переменной глобального массива

    foreach ($GLOBALS['person'] as $item) {

        if ($item['id_personnel'] == $GLOBALS['id']) {

            return $item;
        }
    }
}

```

```
    }  
}  
}
```

### **example\_15. server.php**

```
<?php  
  
    // выполняем обратные преобразования  
  
    $json = urldecode($_SERVER['QUERY_STRING']);  
  
    $person = json_decode($json, 1);  
  
    echo "<pre>";  
  
    print_r($person);  
  
    echo "</pre>";  
  
?>
```

