

## Раздаточный материал к шестой лекции

### Строки

#### Пример: Heredoc синтаксис определения строки

```
<?php
$str = <<<EOD
Пример строки,
охватывающей несколько строчек,
с использованием heredoc-синтаксиса.
EOD;
?>
```

#### Особенности строк

- К строкам не применим оператор “+” для конкатенации, для слияния нескольких строк используется оператор точка “.”.
- Массивы всегда преобразуются в строку **"Array"**.
- Объекты всегда преобразуются в строку **"Object"**. Если вы хотите получить имя класса требуемого объекта, используйте функцию `get_class()`.
- Ресурсы всегда преобразуются в строки со структурой **"Resource id #1"**, где **1** - это уникальный номер ресурса (resource), присвоенный ему PHP во время выполнения. Если вы хотите получить тип ресурса, используйте `get_resource_type()`.
- Интересной особенностью PHP является возможность сериализации экземпляра класса или массива в строку и десериализации из строки (Функции `serialize()` и `unserialize()`!).
- Не ожидайте получить код символа, преобразовав его в целое (как вы могли бы сделать, например, в C). Для преобразования символов в их коды и обратно используйте функции `ord()` и `chr()`.

#### Основные функции для работы со строками

<b>addslashes</b>	Экранирует спецсимволы в стиле языка C
<b>addslashes</b>	Экранирует спецсимволы в строке
<b>bin2hex</b>	Преобразует бинарные данные в шестнадцатичное представление
<b>crc32</b>	Вычисляет CRC32 для строки
<b>crypt</b>	Необратимое шифрование (хэширование)
<b>echo</b>	Выводит одну или более строк
<b>explode</b>	Разбивает строку на подстроки
<b>fprintf</b>	Записывает отформатированную строку в поток
<b>html_entity_decode</b>	Преобразует HTML сущности в соответствующие символы
<b>htmlentities</b>	Преобразует символы в соответствующие HTML сущности.

<b>htmlspecialchars_decode</b>	<b>Преобразует специальные HTML-сущности обратно в соответствующие символы</b>
<b>htmlspecialchars</b>	<b>Преобразует специальные символы в HTML сущности</b>
<b>implode</b>	<b>Объединяет элементы массива в строку</b>
<b>ltrim</b>	<b>Удаляет пробелы из начала строки</b>
<b>md5</b>	<b>Возвращает MD5 хэш строки</b>
<b>money_format</b>	<b>Форматирует число как денежную величину</b>
<b>nl2br</b>	<b>Вставляет HTML код разрыва строки перед каждым переводом строки</b>
<b>parse_str</b>	<b>Разбирает строку в переменные</b>
<b>similar_text</b>	<b>Вычисляет степень похожести двух строк</b>
<b>str_replace</b>	<b>Заменяет строку поиска на строку замены</b>
<b>str_shuffle</b>	<b>Переставляет символы в строке</b>
<b>str_split</b>	<b>Преобразует строку в массив</b>
<b>strip_tags</b>	<b>Удаляет HTML и PHP тэги из строки</b>
<b>strlen</b>	<b>Возвращает длину строки</b>
<b>strpbrk</b>	<b>Ищет в строке любой символ из заданного набора</b>
<b>substr_replace</b>	<b>Заменяет часть строки</b>
<b>substr</b>	<b>Возвращает подстроку</b>

## Работа с файлами и директориями

### Основные функции для работы с файлами

<b>chmod</b>	Изменяет режим доступа к файлу или каталогу
<b>copy</b>	Копирует файл
<b>dirname</b>	Возвращает имя каталога из указанного пути
<b>disk_free_space</b>	Получить размер доступного пространства в каталоге
<b>fclose</b>	Закрывает дескриптор файла
<b>feof</b>	Проверяет, достигнут ли конец файла
<b>fflush</b>	Сброс буфера вывода в файл
<b>fgetc</b>	Считывает символ из файла
<b>fgetcsv</b>	Читает строку из файла и производит разбор данных CSV
<b>fgets</b>	Читает строку из файла
<b>fgetss</b>	Прочитать строку из файла и отбросить HTML-теги
<b>file_exists</b>	Проверить наличие указанного файла или каталога
<b>file_get_contents</b>	Получить содержимое файла в виде одной строки
<b>file_put_contents</b>	Записать строку в файл
<b>file</b>	Читает содержимое файла и помещает его в массив
<b>filemtime</b>	Получить время последнего доступа к файлу
<b>filesize</b>	Получить размер файла
<b>filetype</b>	Получить тип файла
<b>flock</b>	Портируемое рекомендательное записывание файлов
<b>fopen</b>	Открывает файл или URL
<b>fseek</b>	Устанавливает смещение в файловом указателе
<b>is_dir</b>	Определяет, является ли файл директорией
<b>is_file</b>	Определяет, является ли файл обычным файлом
<b>rename</b>	Переименовывает файл или директорию
<b>rewind</b>	Сбрасывает курсор у файлового указателя
<b>rmdir</b>	Удаляет директорию
<b>tempnam</b>	Создаёт файл с уникальным именем
<b>touch</b>	Устанавливает время доступа и модификации файла
<b>unlink</b>	Удаляет файл

**Пример:** Простая запись в файл:

```
$outStr      = «Some string to write»;
$outFile     = fopen("data.js", "w+");
fwrite($outFile, $outStr);
fclose($outFile);
```

## is\_dir()

Функция определяет, является ли файл директорией.

```
<?
var_dump(is_dir('a_file.txt')) . "\n";
var_dump(is_dir('bogus_dir/abc')) . "\n";

var_dump(is_dir('..')); //на одну директорию выше
?>
```

## Основные используемые при работе с каталогами функции

<b>chdir</b>	Сменить каталог
<b>chroot</b>	Сменить корневой каталог
<b>dir</b>	Класс directory
<b>closedir</b>	Освободить дескриптор каталога
<b>getcwd</b>	Получить имя текущего рабочего каталога
<b>opendir</b>	Открыть каталог
<b>readdir</b>	Получить элемент каталога по его дескриптору
<b>rewinddir</b>	Сбросить дескриптор каталога
<b>scandir</b>	Получить список файлов и каталогов, расположенных по указанному пути

## Пример: Получение списка всех файлов в директории

```
<?php
// Обратите внимание, что оператор !== не существовал до версии 4.0.0-RC2

if ($handle = opendir('/path/to/files')) {
    echo "Дескриптор каталога: $handle\n";
    echo "Файлы:\n";

    /* Именно этот способ чтения элементов каталога является правильным. */
    while (false !== ($file = readdir($handle))) {
        echo "$file\n";
    }

    /* Этот способ НЕВЕЕРЕН. */
    while ($file = readdir($handle)) {
        echo "$file\n";
    }

    closedir($handle);
}
?>
```

**Замечание:** readdir() также возвращает элементы с именами . и ... Если они не нужны, то их нужно отбрасывать.

## Работа с заголовком HTML документа

Согласно [RFC 2616](#) заголовок гипертекстового документа может содержать множество разнообразной информации.

### Примеры:

Простой редирект, однако это не простое перемещение по новому адресу, но также и возвращается статусный код 302 (Redirect).

```
<?php
    header("Location: http://www.tsi.lv/");
    exit();
?>
```

### Пример вызова требования аутентификации:

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', false);
?>
```

Также модификация заголовка используется для того, чтобы указать, что происходит вывод документа в отличном от HTML формате.

К примеру, если скрипт динамически генерирует PDF документ, то его скорее всего нужно выдать пользователю вместе с диалогом сохранения файла, для этого можно воспользоваться следующим кодом:

```
<?php
//Указание типа выводимого документа
header('Content-type: application/pdf');

//Имя документа по умолчанию
header('Content-Disposition: attachment; filename="downloaded.pdf"');

//Выдача содержимого документа
readfile('original.pdf');
?>
```

### Пример управления кэшированием документа

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Date in the past
?>
```

Важно понимать, что заголовки всегда отсылаются клиентскому браузеру, а функция **header()** позволяет переписать некоторые их значения.