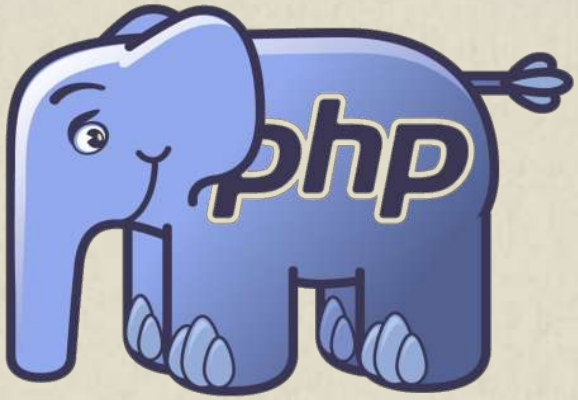


МАССИВЫ

МАССИВЫ

- Массив– это несколько значений, хранящихся, как одно целое. Например:

```
$ages = [42, 45, 33, 18];
```

- Для задания массива используются квадратные скобки
  Иногда можно встретить старый синтаксис, примерно так: `array(1, 2, 3)`*. Не используйте его!*

- Массив в PHP может содержать в себе любое количество элементов любого типа:

```
$foo = [1, 13, ‘bar’, ‘baz’, 42]
```

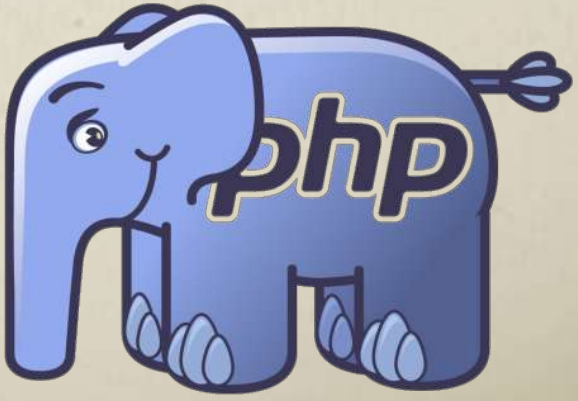
- В массив можно в любой момент добавить еще один элемент:

```
$foo = [1, 2, 3];
$foo[] = 5;
```

- Для быстрого просмотра массива существует конструкция var_dump (вы уже знаете о ней):

```
$foo = [1, 2, 3];
var_dump($foo);
```

МАССИВЫ



ИНДЕКСЫ МАССИВОВ

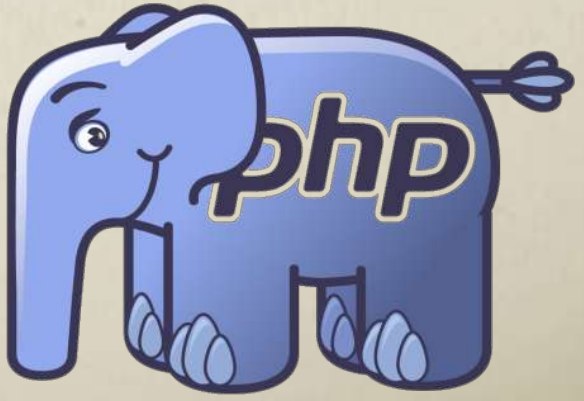
- Все элементы массива, даже если это явно не указано, имеют «номер», который называется «индекс». Если индекс не указан, счет идет от 0.
  К любому элементу можно обратиться по его индексу:

```
$arr = [1, 2, 3];
echo $arr[0]; // 1
$arr[1] = 42;
var_dump($arr); // 1, 42, 3
```

- Индекс можно указать явно:
  `$arr = [1=>‘январь’, 2=>’февраль’]`
- Индекс может быть и числовым и строковым:

```
$arr = [
  ‘jan’=>‘январь’, ‘feb’=>‘февраль’
];
echo $arr[‘jan’]; // январь
```

МАССИВЫ



МНОГОМЕРНЫЕ МАССИВЫ

- Массив в качестве одного из своих элементов может иметь другой массив. Это так называемые многомерные массивы:

```
$arr = [];
$arr[1] = [10, 20, 30];
$arr[2] = [100, 200, 300];
```

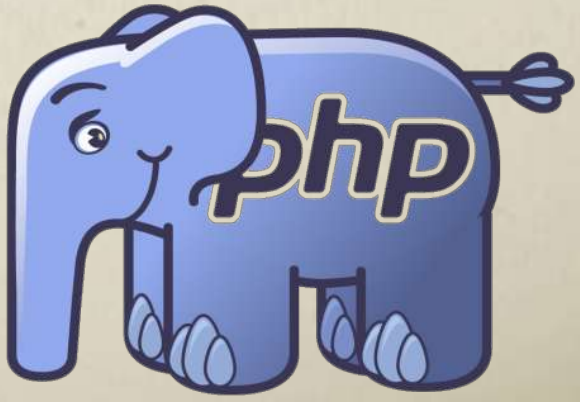
- Вложенность массивов друг в друга неограничена
- Для обращения к элементам многомерного массива используются несколько квадратных скобок:

```
$arr[1]; // это массив [10, 20, 30]
$arr[1][0]; // это число 10
```

- Индексы также могут быть и числовым и строковым:

```
echo $arr['jan'][1]; // понедельник
```

МАССИВЫ



ЦИКЛ foreach

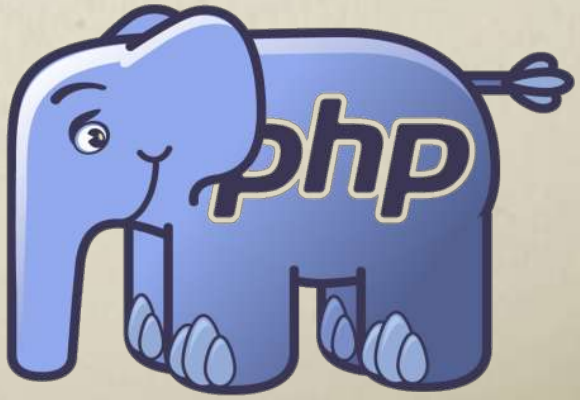
Специальная конструкция языка, позволяющая пройтись по всем элементам массива:

```
$arr = [1, 2, 3];

foreach ($arr as $value) {
  echo $value;
}
```

- Переменная **`$value`** на каждом шаге цикла получает значение следующего элемента массива
- Цикл закончится тогда, когда закончится обход всех элементов
- Можно сразу получать и индексы, в еще одну переменную:

```
foreach ($arr as $index => $value) {
  ...
}
```

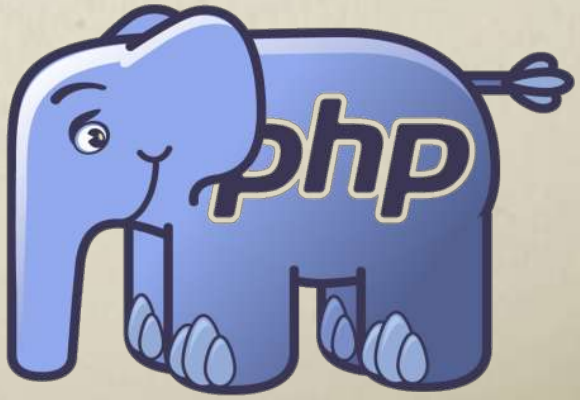
FOREACH

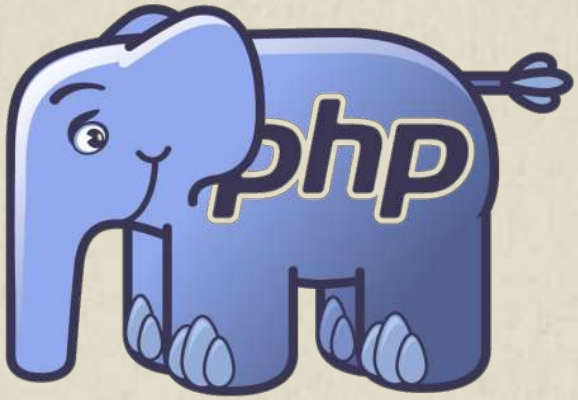


ФУНКЦИИ для работы с массивами

- `$res = in_array($el, $arr);`
  Возвращает булево значение, ответ на вопрос «есть ли в массиве такой элемент?»
- `$res = array_merge($a1, $a2);`
  Слияние двух массивов в один, операция «объединение»
- `$res = array_intersect($a1, $a2);`
  Возвращает совпадающие элементы в массивах, операция «пересечение»
- `$res = implode(‘,’, $arr);`
  Превращает массив в строку, перечисляя его элементы через заданный разделитель
- `$res = explode(‘,’, $str);`
  Превращает строку в массив, разбивая ее по заданному разделителю

ФУНКЦИИ





ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ

Метод GET протокола HTTP

- Это **самый простой** метод для передачи данных от клиента на сервер. Его суть очень проста:
  - К ссылке (любой) можно прикрепить параметры и их значения:

```
<a href="/news.php?id=12">...</a>
```

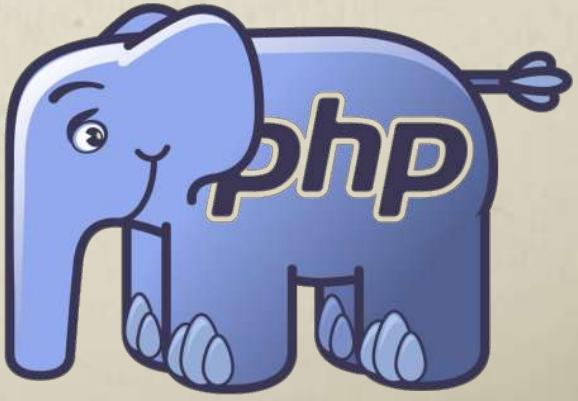
  - В данном случае имя параметра "`id`", а значение "`12`"
  - В PHP вам сразу доступны эти параметры через **суперглобальный массив** `$_GET`

```
echo $_GET['id']; // 12!
```

- Массив `$_GET` называется «суперглобальным» потому что доступен всегда и в любом месте вашей программы
- Передать можно и несколько параметров сразу, разделяя их знаком «&»:

```
/news.php?topic=12&from=2016-01-01
```

МЕТОД GET



Метод POST протокола HTTP

- Это специальный метод для передачи данных от клиента на сервер

- В отличие от GET данные идут «невидимо» для пользователя и не отображаются в адресной строке (однако, можно посмотреть в отладчике)

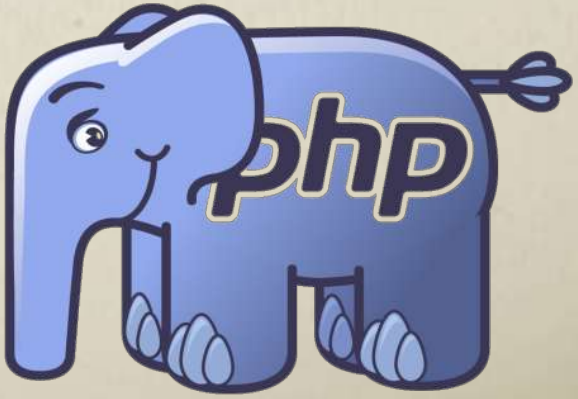
- Самое частое применение, это передача данных из формы:

```
<form action="foo.php" method="post">
  <input type="text" name="bar">
  <input type="submit">
</form>
```

- В этом случае в PHP данные будут доступны в массиве $_POST:

```
echo $_POST['bar'];
```

МЕТОД POST



Суперглобальный массив $_ POST

- Доступен всегда, в любом месте вашей программы
- Содержит в себе данные, переданные в текущем запросе от пользователя на сервер
- Индексы массива – совпадают с названиями полей формы
- Интересные особенности:
  - Точки в именах полей преобразуются в подчеркивания в индексах массива $_POST;
  - Можно передавать массивы, вот пример:

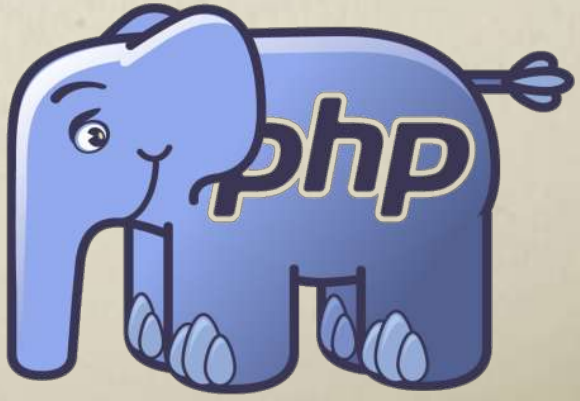
HTML:

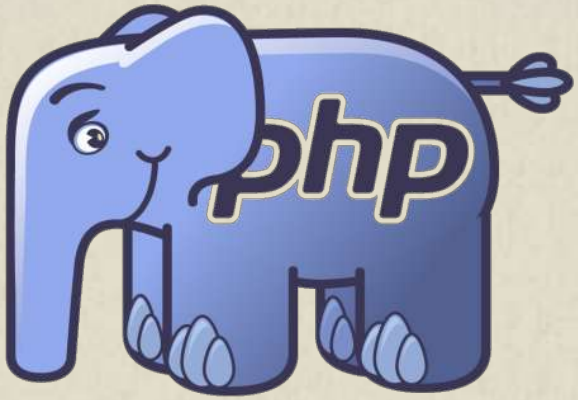
```
<input type="text" name="foo[1]">
<input type="text" name="foo[2]">
```

PHP:

```
echo $_POST['foo'][1];
```

МАССИВ $_POST





ФАЙЛЫ НА СЕРВЕРЕ

Еще один цикл – «ПОКА»

- Для начала нам нужно понять, что такое цикл WHILE («пока»)

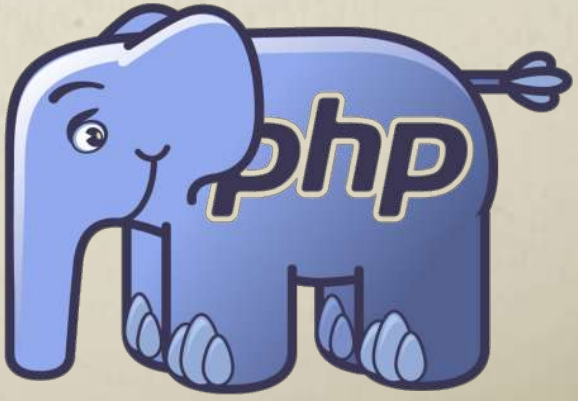
```
// while (условие) { тело цикла }
// цикл будет выполняться,
// ПОКА условие - ИСТИНА

$i = 1;
while ($i <= 10) {
  echo $i;
  $i++;
}
```

- Возможна (но редко применяется) и обратная форма записи:

```
$i = 0;
do {
  echo $i;
  $i++;
} while ($i <= 10)
```

- Разница в том, что цикл с пост-условием всегда выполнится хотя бы один раз

НЕМНОГО ТЕОРИИ



ЧТЕНИЕ файлов – способы прочитать данные из файла в свою программу

- Для начала нам нужно открыть файл. При этом мы получим «ресурс» – ссылку на открытый файл, с которой потом сможем работать:

```
$res = fopen(ПУТЬКФАЙЛУ, 'r');
```

- А затем в цикле читать строки из файла:

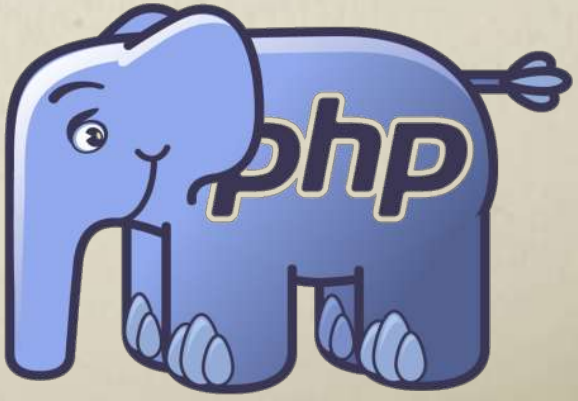
```
while ( !feof($res) ) {
  $line = fgets($res, 1024)
}
```

- И закрыть файл:

```
fclose($res);
```

- Функции, которые вам нужно знать:
  - `fopen()`
  - `fclose()`
  - `fread()`
  - `fgets()`

ЧТЕНИЕ ИЗ ФАЙЛОВ



ЧТЕНИЕ файлов – способы прочитать данные из файла в свою программу

Всё это прекрасно, но нельзя ли как-то проще?

Конечно можно, это же PHP! ☺

- `$lines = file(ПУТЬКФАЙЛУ);`

  Чтение целиком файла в массив. Каждый элемент массива – строка.

- `$str = file_get_contents(ПУТЬКФАЙЛУ);`

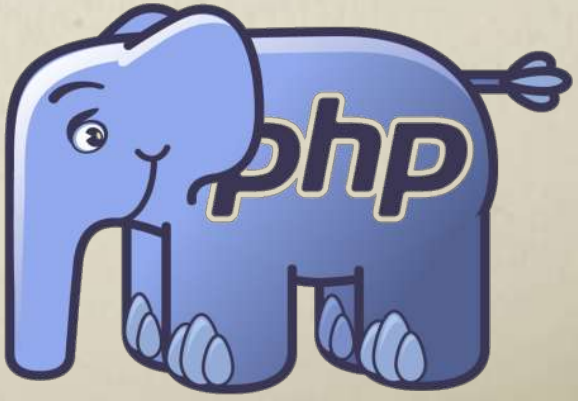
  Чтение целиком файла в строку. Самый оптимальный по производительности вариант

И еще функции, которые вам нужно знать:

- `readfile()`
- `file_exists()`
- `is_readable()`

И, конечно, константа `__DIR__` !

ЧТЕНИЕ ИЗ ФАЙЛОВ



ЗАПИСЬ в файл – способы из программы записать данные в файл

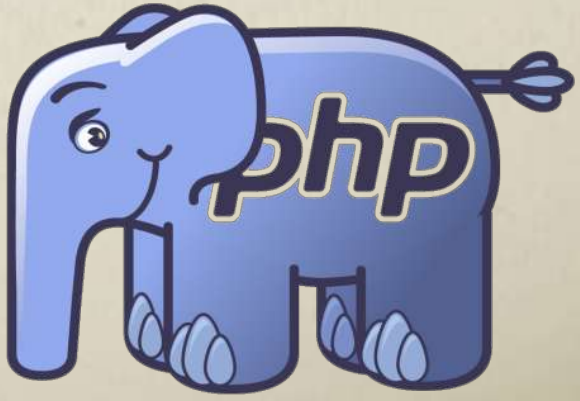
Способ первый, трудоемкий, но зато всё под контролем:

```
$res = fopen(ПУТЬКФАЙЛУ, 'w');
fwrite($res, $data); // string!
fclose($res);
```

Настала пора подробно поговорить о режимах открытия файлов:

- `r` – чтение
- `r+` - чтение и запись
- `w` – запись. файл будет создан, если не существовал или «обнулен»
- `w+` – запись и чтение. файл будет создан, если не существовал или «обнулен»
- `a` – запись. файл будет создан, если не существовал. запись в конец файла

ЗАПИСЬ В ФАЙЛЫ



ЗАПИСЬ в файл – способы из программы записать данные в файл

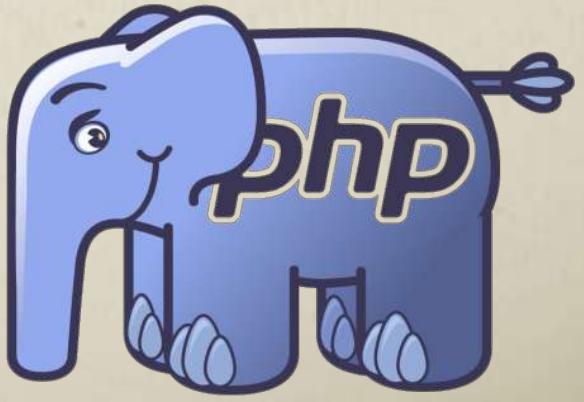
Способ второй, простой, **потому что это PHP**:

```
file_put_contents(ПУТЬКФАЙЛУ, $data);
```

УПРАЖНЕНИЕ

Самостоятельно исследуйте, существуют ли другие варианты

ЗАПИСЬ В ФАЙЛЫ



INCLUDE не так прост, как вы думаете

Это же близко к «чтению из файлов», не так ли?

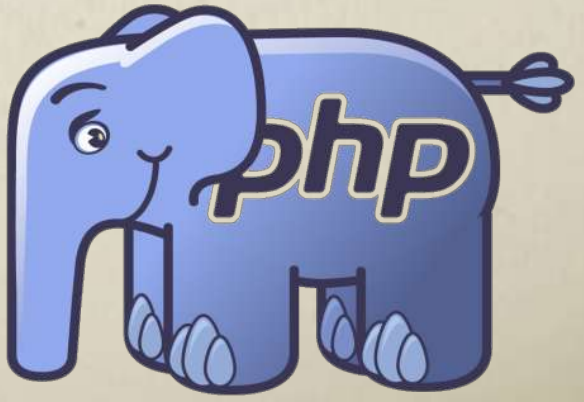
```
// foo.php

<?php
$res = 2 + 2;
return $res;

// index.php

<?php
$foo = include(__DIR__ . '/foo.php');
echo $foo;
```

Таким образом файлы в PHP, как и функции, могут возвращать значения с помощью оператора `return`.

Чтобы это значение получить – достаточно получить то, что вернет конструкция `include`

Что в JavaScript возвращает функция, если в ней не указано явно возвращаемое значение?

КСТАТИ...

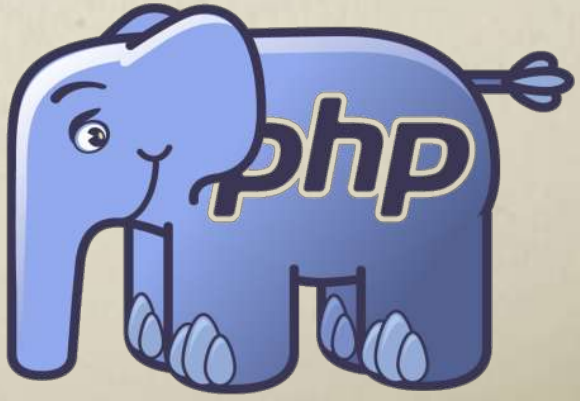


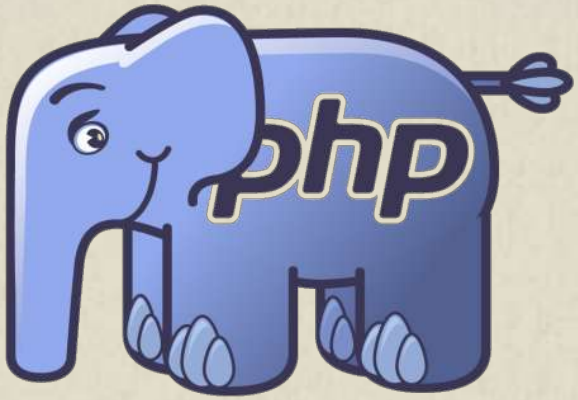
А еще есть средства для работы не только с отдельными файлами, но с файловой системой в целом

Например, имеются следующие функции:

- `scandir($path)`
  Возвращает массив имён всех файлов, содержащихся в папке `$path`
- `dirname($path)`
  Имя родительской папки для указанного файла (или папки)
- `pathinfo($path)`
  Возвращает массив с частями пути (путь, имя файла, расширение)
- `filesize($path)`
  Размер файла `$path` в байтах
- `realpath($path)`
  Возвращает канонический абсолютный путь для указанного. Раскрывает все «.», «..», ссылки и так далее

КСТАТИ...



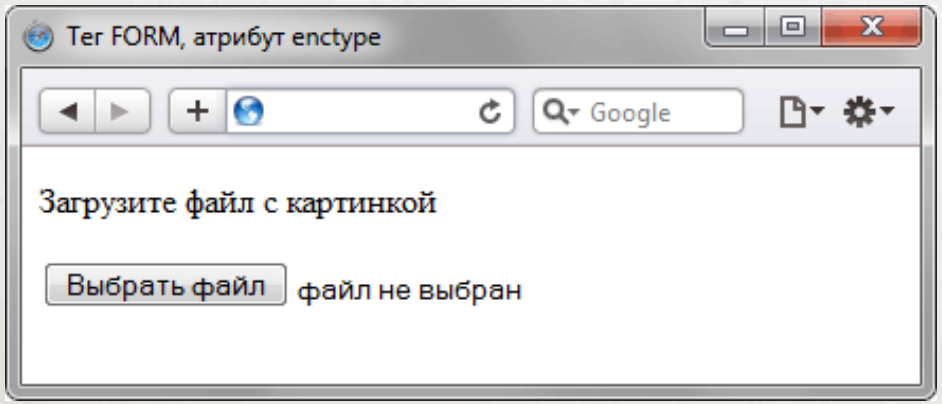


ЗАГРУЗКА ФАЙЛОВ

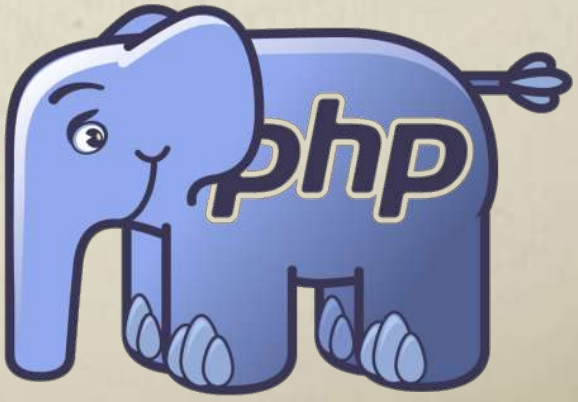
Для того, чтобы иметь возможность загрузить файлы от клиента (из браузера) на сервер, нужно:

- Специальным образом построить форму загрузки:

```
<form
  action="/upload.php"
  method="post"
  enctype="multipart/form-data"
>

  <input type="file" name="myimg">

</form>
```



ФОРМА ДЛЯ ЗАГРУЗКИ



Для того, чтобы иметь возможность загрузить файлы от клиента (из браузера) на сервер, нужно:

- Прочитать данные из суперглобального массива `$_FILES`:

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

 if (0 == $_FILES['myimg']['error']) {

    $res = move_uploaded_file(
      $_FILES['myimg']['tmp_name'],
      ПОЛНЫЙПУТЬНОВОЕИМЯНАСЕРВЕРЕ
    );

  }
}
```

Важно:

- Обращать внимание на размер загружаемого файла, на него могут быть установлены ограничения на сервере
- Использовать корректные пути!

ПРИЕМ ФАЙЛА НА СЕРВЕРЕ