



PHP



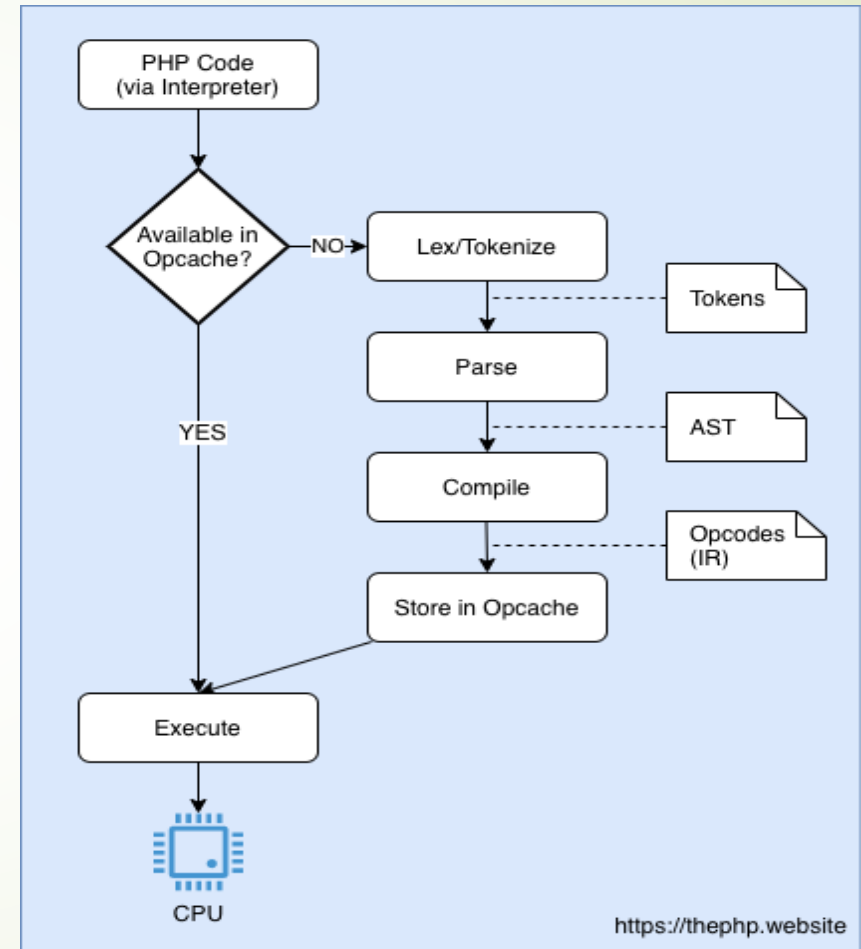
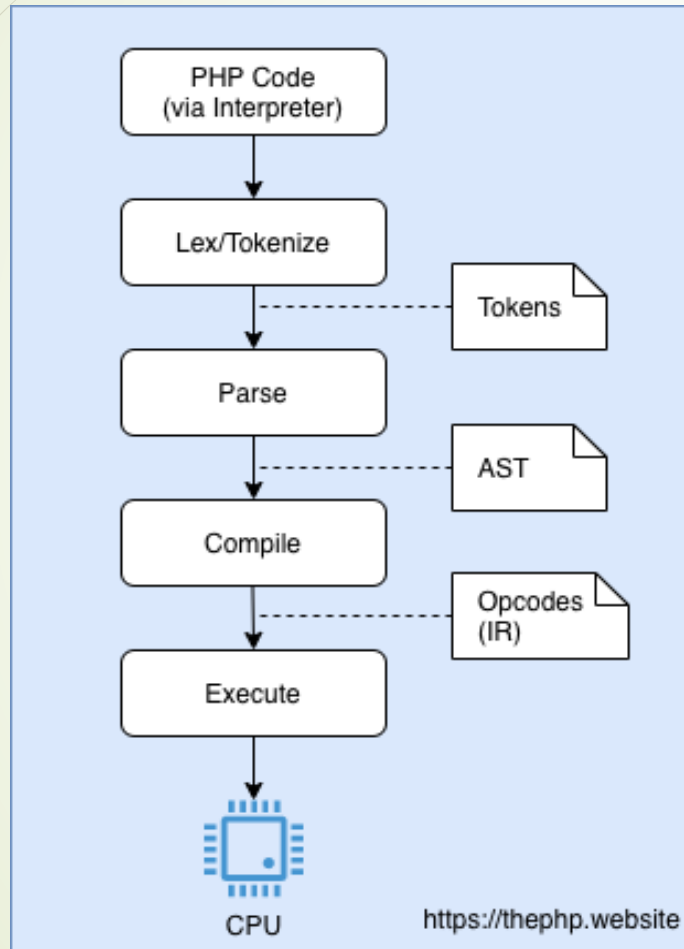
# Статические и динамические сайты

- **Статические сайты** состоят из неизменяемых страниц. Это значит, что сайт имеет один и тот же внешний вид, а также одно и то же наполнение для всех посетителей. При запросе такого сайта в браузере сервер сразу предоставляет готовый HTML-документ, CSS и JS файлы.
- **Динамические сайты**, в свою очередь, имеют изменяемые страницы, адаптирующиеся под конкретного пользователя. Такие страницы не размещены на сервере в готовом виде, а собираются заново по каждому новому запросу.

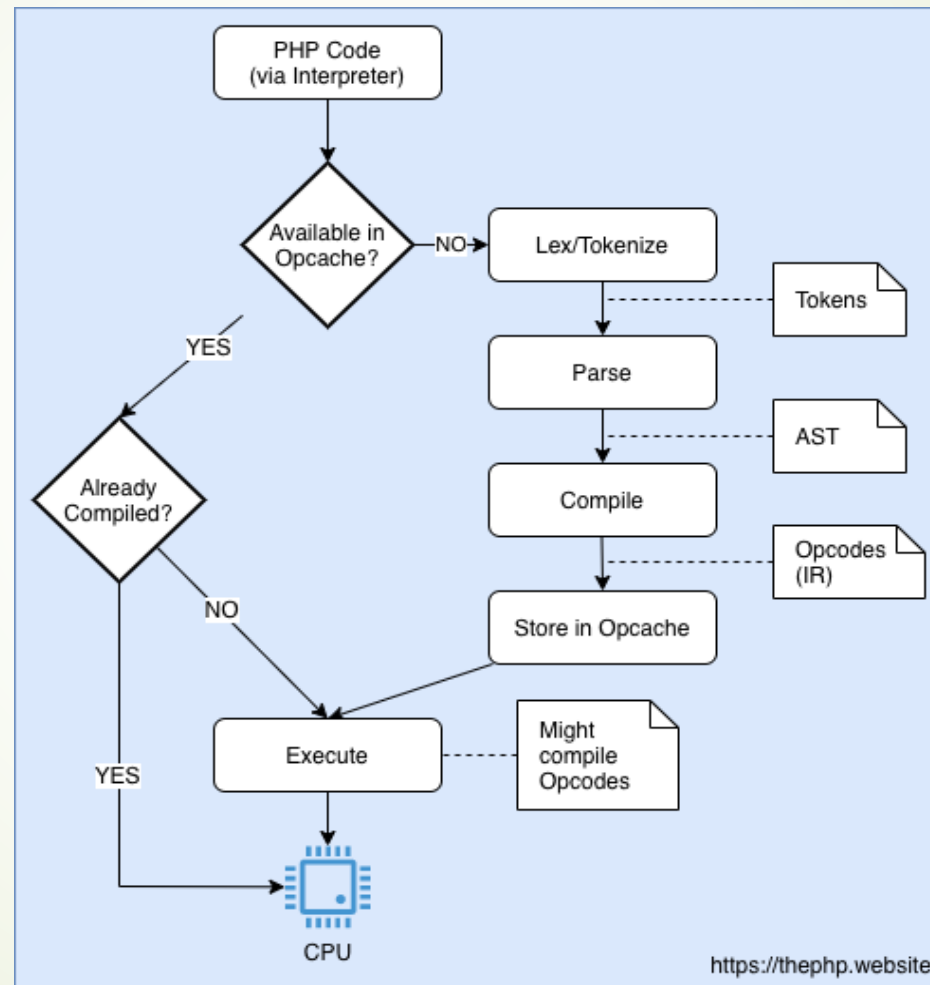
# PHP

- **PHP** (англ. PHP: Hypertext Preprocessor) — скриптовый язык программирования общего назначения, применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-сайтов.





# JIT-КОМПИЛЯЦИЯ





# Пример скрипта

```
<html>  
  <head>  
    <title>My First Page</title>  
  </head>  
  <body>  
    <?php  
      echo "Hello world!";  
    ?>  
  </body>  
</html>
```



# Переменные и константы

```
<?php
    $capital = "Paris";
    echo "The capital of France is", $capital,"<br />";
    $text = "news"
    echo "This is {$text}paper";
```

```
?>
```

```
<?php
    $apples = 5;
    $fruit = "apples";
    echo "Число яблок - ", $$fruit;
    echo "Число яблок - ${$fruit}";
```

```
?>
```

```
<?php
    define("pi", 3.14);
```

```
?>
```





# Типы данных



- **Boolean**. Это логический тип, который содержит значение true или false.
- **Integer**. Содержит значения целого числа (Например: 4 или 10 или другое целое число).
- **String**. Содержит значение текста произвольной длины (Например: Олег, Киев, Австрия).
- **Float**. Вещественное число (Например: 1.2, 3.14, 8.5498777).
- **Object**. Объект.
- **Array**. Массив.
- **Resource**. Ресурс (Например: файл).
- **NULL**. Значение NULL.



# Оператор исполнения

```
➤ <?php  
    $d = `dir d:\\`;   
    echo $d;  
?>
```

# Строковые операции

chr	Возвращает символ по его коду ASCII
chunk_split	Разбивает строку на подстроки заданной длины
crypt	Зашифровывает строку с использованием одного из алгоритмов
<u>echo</u>	Выводит одну или несколько строк
explode	Разбивает строку на подстроки, ограниченные заданным разделителем, и форматирует из них массив
html_entity_decode	Декодирует все HTML-представления в соответствующие символы. Функция обратна по отношению к htmlentities
htmlentities	Кодирует все специальные символы в их HTML-представление
htmlspecialchars	Кодирует все символы в их HTML-представление
implode	Формирует строку из элементов массива
ltrim	Удаляет начальные пробелы из строки
rtrim	Удаляет конечные пробелы из строки
number_format	Представляет число в виде строки в различных форматах
ord	Возвращает ASCII-код символа
parse_str	Разбивает строку URL и присваивает значение переменным
print	Выводит строку
printf	Выводит строку с форматированием
sprintf	Возвращает строку с форматированием
setlocale	Устанавливает информацию о кодовой странице
similar_text	Вычисляет степень похожести двух строк



# foreach

```
➡ <?php
    $array = array ("Apple", "Limon", "Chery", "Oranges");

    foreach ($array as $value)
    {
        echo "Вы выбрали фрукт - $value <br>";
    }
?>
```

# Создание массивов

```
➤ <?php
    $arr = array("php", "html", "css");
?>

<?php
    $arr = array(1 => "php", "html", "css");
?>

<?php
    $arr = ["php", "laravel", "yii", "zend", "cakephp"];
?>
```

# Удаление элемента из массива

```
➡ <?php
    $arr[0] = "PHP";
    $arr[1] = "HTML";
    $arr[2] = "CSS";
    unset($arr[1]);
    print_r($arr);
?>
```

# Функции для работы с массивами

array_chunk	Разбивает массив на несколько меньших массивов заданного размера
array_combine	Создает массив из двух заданных массивов - массива индексов элементов и массива значений
array_count_values	Формирует массив, индексами которого являются значения заданного массива, а значениями - число повторений соответствующего значения в заданном массиве
array_diff	Формирует массив из тех элементов первого заданного массива, которые отсутствуют в остальных заданных в качестве аргументов функции массива
array_fill	Заполняет массив заданным значением
array_intersect	Формирует массив из элементов, которые присутствуют во всех заданных массивах
array_key_exists	Проверяет наличие заданного индекса в массиве
array_keys	Возвращает массив из индексов заданного массива
array_merge	Объединяет несколько массивов в один
array_multisort	Выполняет сортировку многомерного массива или нескольких одномерных массивов
array_pad	Дополняет массив до заданного количества элементов заданным значением
array_pop	Возвращает последний элемент массива, одновременно удаляя элемент из массива
array_push	Добавляет заданные элементы в конец массива
array_rand	Выбирает один или несколько случайно взятых элементов из массива

# Преобразования



<?php

```
$arr[0] = "PHP";  
$arr[1] = "HTML";  
$arr[2] = "CSS";
```

```
$string = implode(" ", $arr);  
echo $string;
```

?>



<?php

```
$string = "PHP, HTML, CSS";  
$arr = explode(" ", $string);  
print_r($arr);
```

?>



# Extract

```
■ <?php
    $arr["one"] = "PHP";
    $arr["two"] = "HTML";
    $arr["three"] = "CSS";

    extract($arr);

    echo "\$one = $one <br>";
    echo "\$two = $two <br>";
    echo "\$three = $three <br>";
?>
```



# Compact

➤ <?php

```
$one = "PHP";
```

```
$two = "HTML";
```

```
$three = "CSS";
```

```
$arr = compact("one", "two", "three");
```

```
print_r($arr);
```

```
?>
```

# Соединение массивов

```
➤ <?php
    $arr[1] = "PHP";
    $arr[2] = "HTML";
    $arr[3] = "CSS";

    $arr2[1] = "PHOTOSHOP";
    $arr2[2] = "PAINT.NET";
    $arr2[3] = "DREAMWEAVER";

    $new_arr = array_merge($arr, $arr2);
    print_r($new_arr);
?>
```

# Вычитание

```
➤ <?php
    $arr[1] = "PHP";
    $arr[2] = "HTML";
    $arr[3] = "CSS";

    $arr2[1] = "PHP";
    $arr2[2] = "PAINT.NET";
    $arr2[3] = "DREAMWEAVER";

    $diff = array_diff($arr, $arr2);
    print_r($diff);
?>
```

# Пересечение

```
➤ <?php
    $arr[1] = "PHP";
    $arr[2] = "HTML";
    $arr[3] = "CSS";

    $arr2[1] = "PHP";
    $arr2[2] = "PAINT.NET";
    $arr2[3] = "DREAMWEAVER";

    $diff = array_intersect($arr, $arr2);
    print_r($diff);
➤ ?>
```

# Уникальные значения

➤ <?php

```
$arr = array(30, 44, 97, 30);
```

```
print_r($arr);
```

```
$new_arr = array_unique($arr);
```

```
print_r($new_arr);
```

?>

# Многомерные массивы

➤ <?php

```
$companies["Microsoft"][1] = "Programmer";  
$companies["Microsoft"][2] = "PR";  
$companies["Microsoft"][3] = "Office Manager";
```

```
$companies["Google"][1] = "IT";  
$companies["Google"][2] = "Web-design";
```

```
$companies["Mozilla"][1] = "PR";  
$companies["Mozilla"][2] = "C++ Programmer";
```

```
print_r($companies);
```

?>





# Операторы



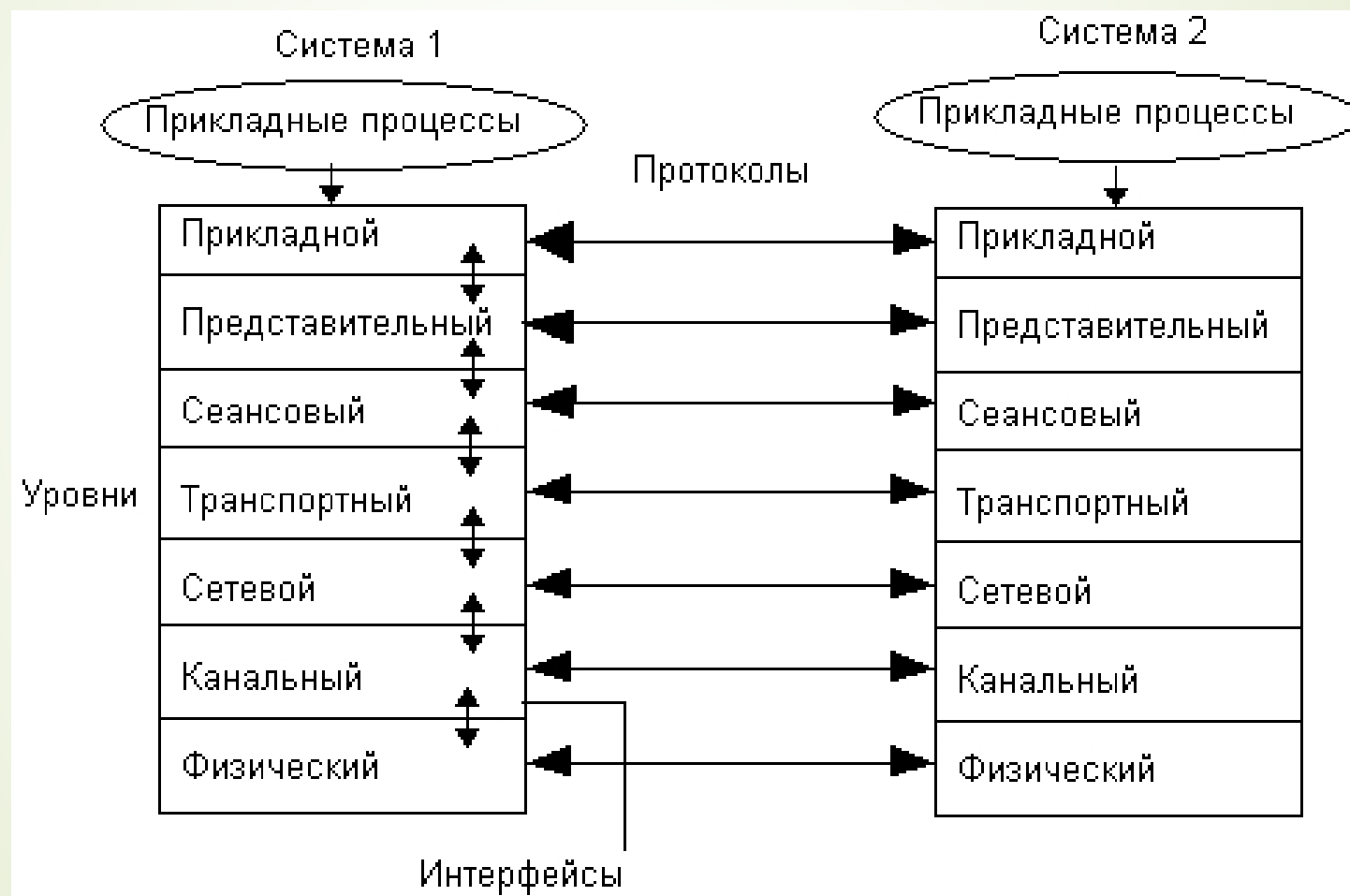
<code>\$a + \$b</code>	Объединение
<code>\$a == \$b</code>	Равно
<code>\$a === \$b</code>	Тождественно равно
<code>\$a != \$b</code>	Не равно
<code>\$a &lt;&gt; \$b</code>	Не равно
<code>\$a !== \$b</code>	Тождественно не равно



# Функции

- ```
function name([argument_list...])  
{  
    [statements;]  
    [return value;]  
}
```
- ```
function hello($text = "Привет") {  
    echo $text;  
}
```

# Модель OSI





# HTTP-запрос



- HTTP-метод, обычно глагол подобно GET, POST или существительное, как OPTIONS или HEAD, определяющее операцию, которую клиент хочет выполнить. Обычно, клиент хочет получить ресурс (используя GET) или передать значения HTML-формы (используя POST).
- Путь к ресурсу
- Версию HTTP-протокола.
- Заголовки (опционально), предоставляющие дополнительную информацию для сервера.
- Тело, для некоторых методов, таких как POST, которое содержит отправленный ресурс.

# Типы запросов

GET – запрос ресурса, данных

HEAD – аналогичен GET, но возвращает только заголовки без самого контента

POST – «отправка» данных» на сервер

PUT – применяется для загрузки содержимого запроса на указанный в запросе URI.

Если по заданному URI не существует ресурс, то сервер создаёт его и возвращает статус 201 (Created).

DELETE – удаление ресурса

CONNECT – запускает двустороннюю связь с запрошенным ресурсом. Метод можно использовать для открытия туннеля.

OPTIONS – используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.

TRACE – возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

PATCH – аналогично PUT, но применяется только к фрагменту ресурса



# КОДЫ СОСТОЯНИЯ

- Код ответа (состояния) HTTP показывает, был ли успешно выполнен определённый HTTP запрос. Коды сгруппированы в 5 классов:
  1. Информационные 100 - 199
  2. Успешные 200 - 299
  3. Перенаправления 300 - 399
  4. Клиентские ошибки 400 - 499
  5. Серверные ошибки 500 - 599

# Заголовки клиента

Поля HTTP заголовка	Описание поля HTTP заголовка	Пример
Accept	Поле заголовка запроса Accept используется, чтобы определить тип информации, который должен содержаться в ответе <u>HTTP сервера</u>	Accept: text/plain;
Accept-Charset	Поле заголовка запроса Accept-Charset указывает на <u>кодировку</u> , которая должна быть в ответе сервера. Другими словами: данное поле указывает на то, какие наборы символов приемлемы для ответов сервера	Accept-Charset: utf-8
Accept-Encoding	Поле заголовка запроса Accept-Encoding указывает серверу на то, какие способы кодирования приемлемы для ответа.	Accept-Encoding: compress, gzip
Accept-Language	Поле заголовка запроса Accept-Language указывает серверу приемлемые языки (естественные языки: русский, китайский, английский и пр.)	Accept-Language: da, en-gb; en;
Host	Поле заголовка запроса Host используется для указания доменного имени и порта запрашиваемого ресурса.	Host: zametkinapolyah.ru
User-Agent	Поле заголовка запроса User-Agent содержит в себе полную информацию о клиенте пользователя, например, о браузере.	User-Agent: CERN-LineMode/2.15 libwww/2.17b3



# Заголовки ответа сервера

Allow	Поле заголовка ответа Allow передает клиенту методы, которые тот может использовать	Allow: GET, HEAD, PUT
Content-Encoding	Поле заголовка ответа Content-Encoding указывает на дополнительный способ кодирования тела HTTP объекта с целью сжатия	Content-Encoding: gzip
Content-Language	Поле заголовка ответа Content-Language указывает клиенту на каком языке информация, находящаяся в теле объекта.	Content-Language: mi, en
Content-Length	Поле заголовка ответа Content-Length указывает необходимую длину тела сообщения в байтах	Content-Length: 3495
Content-Location	Поле заголовка ответа Content-Location используется для идентификации исходного местоположения объекта на сервере.	«Content-Location» «:»absoluteURI   relativeURI )
Content-MD5	Поле заголовка ответа Content-MD5 используется для проверки целостности объектов сообщений	Content-MD5 : 8c2d46911f3f5a326455f0ed7a8ed3b3

# Общие заголовки

Поля HTTP заголовка	Описание поля HTTP заголовка	Пример
Cache-Control	Общее поле HTTP заголовка Cache-Control определяет директивы для управления кэшем, которым должны следовать все кэширующие механизмы	Cache-Control: no-cache Cache-Control: no-store  Cache-Control: max-age=3600
Connection	Общее поле HTTP заголовка Connection позволяет управлять HTTP соединением	Connection: close
Date	Общее поле HTTP заголовка хранит дату и время создания HTTP сообщения	Date: Tue, 15 Nov 1994 08:12:31 GMT
MIME-Version	Общее поле HTTP заголовка MIME-Version содержит версия протокола MIME, по которому было сформировано сообщение.	MIME-Version = «MIME-Version» «:» 1*DIGIT «.» 1*DIGIT
Pragma	Общее поле HTTP заголовка Pragma используется для включения особых директив, которые применяются к любому получателю HTTP сообщения.	Pragma: no-cache
Trailer	Общее поле HTTP заголовка Trailer хранит в себе список полей, которые имеют отношение к кодированию сообщения и кодированию передачи.	Trailer :field-name
Transfer-Encoding	Общее поле HTTP заголовка Transfer-Encoding служит для передачи списка методов кодирования передачи	Transfer-Encoding: chunked

# MIME

- MIME (Multipurpose Internet Mail Extension, Многоцелевые расширения почты Интернета) — спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео, архивов и др.

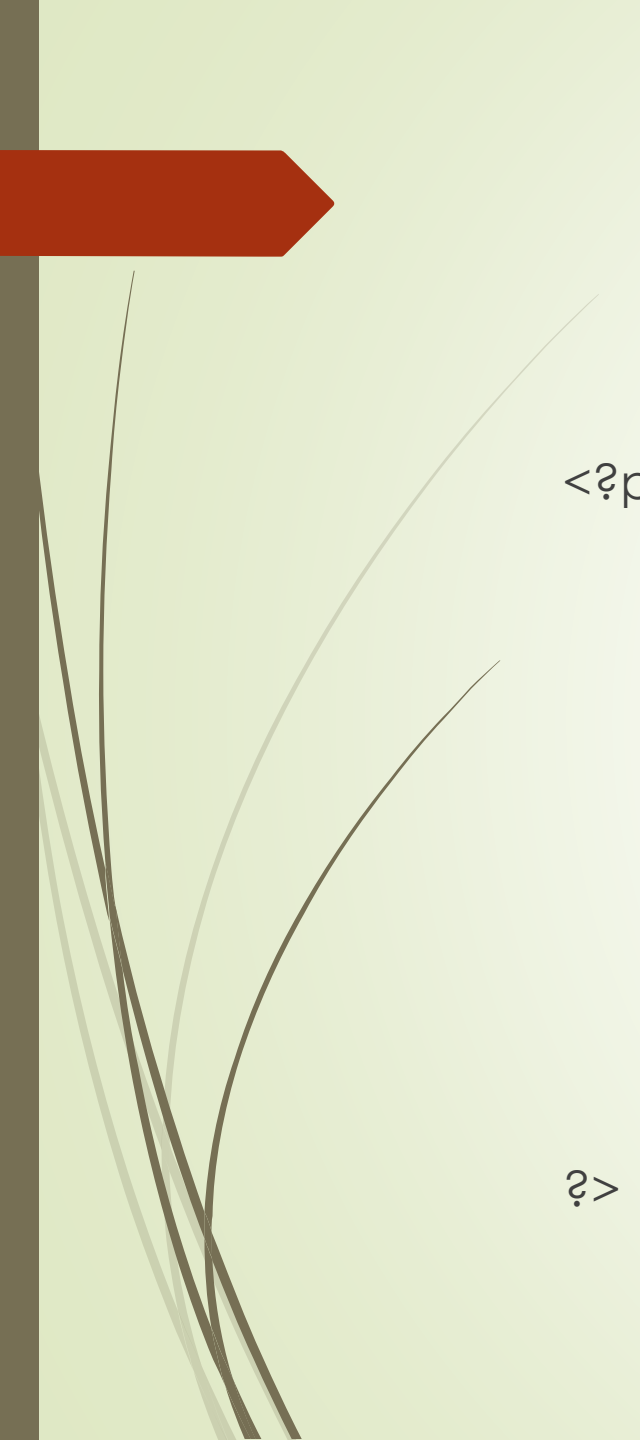
Расширение файла	Тип данных
avi	video/avi
avi	video/msvideo
bmp	image/bmp
bmp	image/x-windows-bmp
book	application/book
c	text/x-c
class	application/java
cpp	text/x-c
cpt	application/mac-compactpro
cpt	application/x-compactpro
cpt	application/x-cpt
css	text/css



# Обработка форм



```
<?php header("Content-Type: text/html; charset=utf-8");?>
    <form action="app/check.php" method="post">
        <p>Имя: <input name="name" type="text"></p>
        <p>Фамилия: <input name="surname" type="text"></p>
        <p>E-mail: <input name="email" type="text"></p>
        <p>Сообщение: <br /><textarea name="message" cols="30"
            rows="5"></textarea></p>
        <p><input type='submit' value='Отправить'></p>
    </form>
<?php>
```



```
<?php
```

```
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
```

```
        $name = $_POST['name'];
```

```
        $surname = $_POST['surname'];
```

```
        $email = $_POST['email'];
```

```
        $message = $_POST['message'];
```

```
        echo "name: $name <br/> surname: $surname <br/> email: $email
```

```
        <br/> message: $message";
```

```
    }
```

```
?>
```



# ООП

```
➤ class FirstClass  
    {  
        ...  
    }
```

```
<?php  
    $instance = new FirstClass();  
    // Это можно сделать и с помощью переменной:  
    $className = 'FirstClass';  
    $instance = new $className(); // FirstClass()  
?>
```

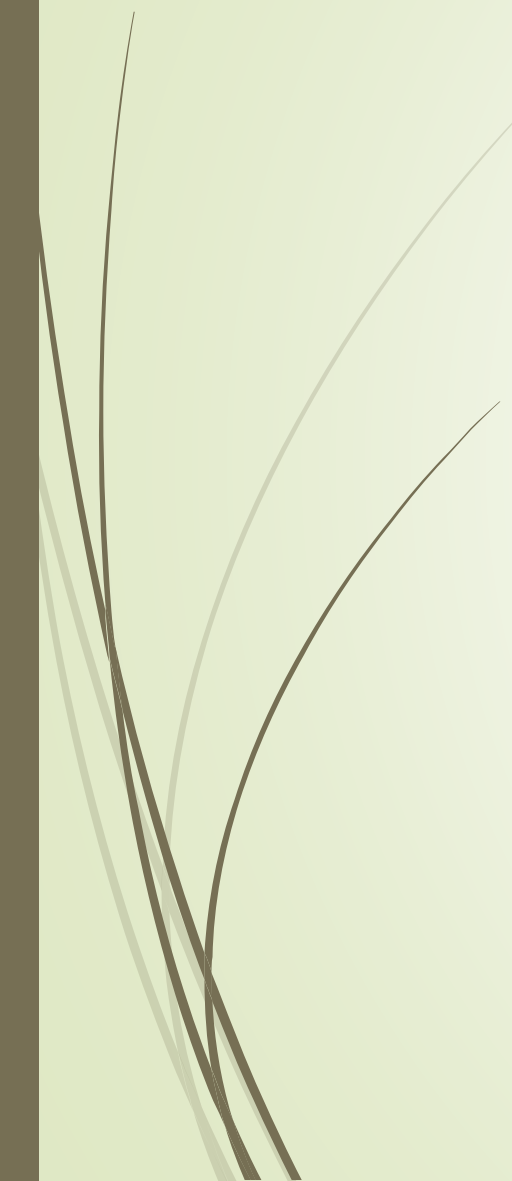



# Конструктор и деструктор

```
function __construct(mixed $args = "") { // можно передавать параметры
    ...
}
```

```
function __destruct() {
    echo "Вызов деструктора";
}
```





```
abstract class AbstClass {  
    abstract protected function getValue();  
    public function printValue() {  
        print $this->getValue() . "\n";  
    }  
}  
class FirstClass extends AbstClass  
{  
    protected function getValue() {  
        return "FirstClass";  
    }  
}
```



# Static

```
class FirstClass
{
    public static $var1 = "var1";

    public function staticValue() {
        return self::$var1;
    }
}
```

# Интерфейсы

```
interface CarTemplate
{
    public function getId(); // получить id автомобиля
    public function getName(); // получить название
    public function add(); // добавить новый автомобиль
}
?>
```

- Для реализации интерфейса используется оператор **implements**
- Если нужно, то классы могут реализовывать более одного интерфейса, реализуемые интерфейсы должны разделяться запятой.

# Перегрузка свойств



- Обращения к свойствам объекта могут быть перегружены с использованием методов `__get` и `__set`. Эти методы будут срабатывать в том случае, если объект не содержит свойства, к которому осуществляется доступ. Синтаксис:

```
<?php
    public void __set (string $name , mixed $value)
    public mixed __get (string $name)
?>
```

# Перегрузка методов

- Вызовы методов могут быть перегружены с использованием методов `__call`. Эти методы будут срабатывать в том случае, если объект не содержит метода, к которому осуществляется доступ. Синтаксис:

```
<?php
    public mixed __call (string $name , array $arguments)
?>
```

- 
- 
- Метод `__toString()` будет срабатывать при попытке преобразования класса в строку.
  - Метод `__invoke()` вызывается, когда объект пытаются вызвать как функцию.



# Пространства имен

```
<?php
    namespace App\Main;
    class MyClass {
        function hello() {
            return "hello";
        }
    }
}
```

?>

```
<?php
    namespace App\Main;
    require_once "App\Main\MyClass.php";
    $obj = new \App\Main\MyClass;
    echo $obj->hello(); // hello
```

?>






# Псевдонимы

- use App/Core/Controller as CoreController;

# Трейты

```
➤ trait MyTransliterater {  
    private $letters = array(  
        'а' => 'a',      'б' => 'b',   'в' => 'v',  
        'г' => 'g',      'д' => 'd',   'е' => 'e',  
        'ё' => 'e',      'ж' => 'zh',  'з' => 'z',  
        'и' => 'i',      'й' => 'y',   'к' => 'k',  
        ...  
    );  
  
    public function translate($str) {  
        return strtr(trim($str), $this->letters);  
    }  
}
```



```
class MyClass {  
    use MyTransliterato;  
    private $data;  
    ...  
    public function getPreparedData() {  
        $this->data['url'] = strtolower($this->translate($this->data['title']));  
  
        return $this->data;  
    }  
}
```

# Открытие файлов

- `r` – открытие файла только для чтения.
- `r+` - открытие файла одновременно на чтение и запись.
- `w` – создание нового пустого файла. Если на момент вызова уже существует такой файл, то он уничтожается.
- `w+` - аналогичен `r+`, только если на момент вызова файл такой существует, его содержимое удаляется.
- `a` – открывает существующий файл в режиме записи, при этом указатель сдвигается на последний байт файла (на конец файла).
- `a+` - открывает файл в режиме чтения и записи при этом указатель сдвигается на последний байт файла (на конец файла). Содержимое файла не удаляется.


```
<?php
    $fp = fopen('counter.txt', 'r'); // Бинарный режим
    $fp = fopen('counter.txt', 'rt'); // Текстовый режим
    $fp = fopen("http://www.yandex.ru", "r");
    $fp = fopen("ftp://user:password@example.ru", 'w');
?>
```

# Чтение файла

```
?php
$fd = fopen("form.php", 'r') or die("не удалось открыть файл");
while(!feof($fd))
{
    $str = htmlentities(fgets($fd));
    echo $str;
}
fclose($fd);
?>
```

```
<?php
    $str = htmlentities(file_get_contents("form.php"));
    echo $str;
?>
```

```
<?php
$fd = fopen("form.php", 'r') or die("не удалось открыть файл");
while(!feof($fd))
{
    $str = htmlentities(fread($fd, 600));
    echo $str;
}
fclose($fd);
?>
```



# Запись в файл

```
<?php
```

```
$fd = fopen("hello.txt", 'w') or die("не удалось создать файл");  
$str = "Привет мир";  
fwrite($fd, $str);  
fclose($fd);
```

```
?>
```