ОБРАБОТКА ФОРМ

ТИПЫ ЗАПРОСОВ HTTP

- Get – используется при наборе адреса сайта в строке браузера или перехода по ссылке.
- Post – служит для отправки формы, например при регистрации на сайте, добавления комментария к статье.

URL

- http://example.com:80/path/to/document.html?parametrs
- Протокол – часть url, указывает браузеру, какой протокол нужно использовать для обмена данными с Web-сервером.
- Имя хоста – это либо доменное имя хоста, либо его IP-адрес.
- Порт – (80 – стандартный для http, по умолчанию), порт идентифицирует постоянно работающую программу на сервере.
- Путь к файлу – страница может быть виртуальной или реальной.

URL И ПАРАМЕТРЫ ЗАПРОСОВ

- http://хост/путь?параметры,
- Параметры – это набор пар вида: имя=значение
- Параметры – параметры, которые передаются на сервер, для передачи нескольких параметров используется символ амперсанда &.

Способ посылки параметров сценарию, когда данные помещаются в командную строку, называется методом Get.

ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ

- Переменные окружения представляют собой именованные значения параметров, которые операционная система (процесс-родитель) передает запущенной программе.
- Программа может с помощью специальных функций получить значение любой установленной переменной окружения, указав её имя.
- Так может поступить и CGI сценарий, когда захочет узнать значение того или иного заголовка запроса.
- Набор передаваемых сценарию переменных окружения ограничен стандартами.

ЗАГОЛОВКИ ЗАПРОСА И МЕТОД GET

- Что происходит, когда мы набираем в браузере строку и нажимаем Enter?
- Браузер анализирует строку, выделяет из нее имя сервера и порт (+протокол), устанавливает соединение с Web-сервером по адресу **сервер: порт** и посылает ему запрос, например, следующего вида:

GET somestring HTTP/1.0\n
••• информация •••
\n\n

- \n\n – маркер окончания запроса (два обязательных символа новой строки).
- После Get-строки могут следовать и другие строки с информацией (**заголовки headers**), обычно их формирует браузер.

GET,
ФОРМАТ GET СЦЕНАРИЙ?ПАРАМЕТРЫ HTTP/1.0

- Переменные окружения
  - REQUEST_URI – uri-адрес
  - QUERY_STRING – в данной строке сохраняются параметры
  - REQUEST_METHOD – метод передачи GET

POST
POST **СЦЕНАРИЙ?ПАРАМЕТРЫ HTTP/1.0**

- Переменные окружения
  - REQUEST_URI – uri-адрес
  - QUERY_STRING – в данной строке сохраняются параметры
  - REQUEST_METHOD – метод передачи POST
- Метод отличается от GET, тем что данные можно передавать не только через командную строку, но и в конце заголовков.

CONTENT-TYPE

CONTENT-TYPE: APPLICATION/X-WWW-FORM-URLENCODED

- Переменная окружения CONTENT_TYPE
- Данный заголовок идентифицирует **тип передаваемых данных.**
  - Application/x-www-form-urlencoded - этот формат используется методами GET и POST.
  - Распространен еще формат multipart/form-data (для загрузки данных на сервер).

HOST
HOST: ИМЯ ХОСТА

- Переменная окружения: HTTP_ HOST.
- В соответствии с http/1.1 в Интернете на каждом узле может располагаться сразу несколько хостов.

Существует одно важное отличие между переменными HTTP_HOST и SERVER_NAME. Дело в том, что SERVER_NAME никогда не включает номер порта, к которому подключился браузер (обычно 80), в то время как HTTP_HOST, наоборот, содержит значение вида *хост*:*порт*, когда порт отличен от 80.

USER-AGENT

MOZILLA/5.0 (WINDOWS NT 6.3; WOW64; RV:41.0) GECKO/20100101 FIREFOX/41.0

- Переменная окружения HTTP_USER_AGENT.
- Через данный заголовок клиент сообщает серверу сведения о себе, не всегда правдивые.

REFERER

REFERER: *URL_АДРЕС*

- Переменная окружения HTTP_ REFERER.
- Заголовок формируется браузером и содержит URL страницы, с которой осуществился переход на текущую страницу по гиперссылке.

CONTENT-LENGTH
CONTENT-LENGTH: ДЛИНА

- Переменная окружения - CONTENT_ LENGTH.
- Заголовок содержит строку, являющуюся десятичным представлением длины данных в байтах, передаваемых методом POST. Если используется метод Get, то данная переменная окружения не устанавливается.

COOKIE
COOKIE: ЗНАЧЕНИЕ*COOKIES*

- Переменная окружения - HTTP_ COOKIE.
- В данной переменной хранятся все coolies в URL-кодировке.

ACCEPT

ACCEPT: TEXT/HTML, TEXT/PLAIN, IMAGE/GIF, IMAGE/JPEG

- Переменная окружения – HTTP_ACCEPT.
- В этом заголовке браузер перечисляет, какие типы документов он «понимает». Браузеры могут отправить в этом заголовке значение */*.

ПРИМЕР – ВЫПОЛНЕНИЕ ЗАПРОСА ЧЕРЕЗ TELNET

- В командной строке введем
- telnet localhost 80 (Enter)
- get index.html http/1.1 (Enter)
- host: localhost (Enter)
- (Enter) (Enter)

С помощью заголовков реализуются такие механизмы, как контроль кодировок, cookies, метод post и т.д.

ВЗАИМОДЕЙСТВИЕ С СЕРВЕРОМ - ФОРМЫ

- Одним из наиболее распространённых способов, с помощью которого пользователь взаимодействует с сервером, является использование HTML-форм.
- Данные на форме отправляются на сервер с помощью одного из вариантов, например, с помощью кнопки Submit.
- Язык PHP изначально был создан для облегчения получения данных из пользовательских форм.

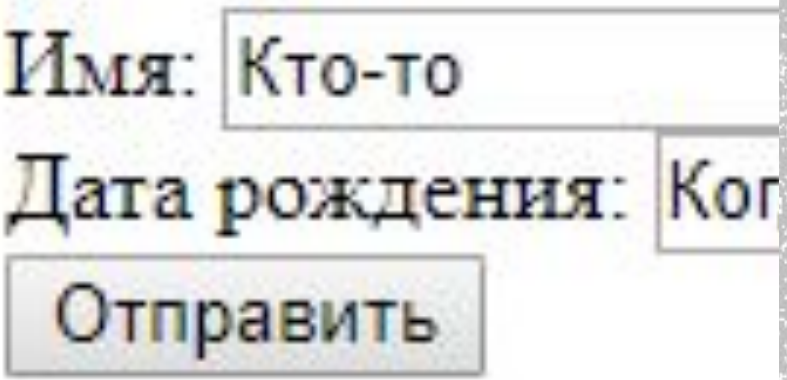
ПЕРЕДАЧА ПАРАМЕТРОВ ВРУЧНУЮ

- http://example.com/scriptcgi?name=Thomas&bom=1962-03-11

```
<!DOCTYPE html>
<html lang="ru">
<body>
Привет, name! Я знаю, Вы родились born!
</body>
</html>
```

```
<html><body>
Привет, Thomas! Я знаю, Вы родились 1962-03-11!
</body></html>
```

```
<html lang="ru">
<head>
<title>Форма</title>
<meta charset='utf-8'>
</head>
<body>
<form action="script.cgi" method="GET"
Имя:
<input type="text" name="name" value="
Дата рождения:
<input type="text" name="born" value="
<input type="submit" value="Отправить"
</form>
</body>
</html>
```



ПЕРЕДАЧА ПАРАМЕТРОВ МЕТОД GET

- После нажатия на кнопку **Отправить** скрипт script.cgi передаст через ? Все параметры, которые помещены внутрь тегов input в форме, отделяя их амперсандами (&).

МЕТОД POST

- Если передаются данные, то используется заголовок content-length, для того, чтобы сервер понял, когда все данные будут переданы.
- Сервер никак не интерпретирует post-данные, а посылает их непосредственно сценарию.
- Post используют при загрузке файлов через Web, при обработке больших форм.
- При применении Get url становится длинным, а при методе post данные доставляются в url-кодированном виде (все интернет-сервисы, начиная с e-mail и заканчивая Web, применяют способ перекодировки символов в диапазонах 0-32, 128-256, пробел представляется символом +, например 9E - %9E, все буквы кириллицы увеличиваются примерно в 3 раза).

МЕТОД POST И ФОРМЫ

- Для отправки данных методом POST указывается данный метод.
- Пользователь всегда будет иметь дело только с полями ввода, переключателями и кнопками формы, гиперссылками.
- Необходимо выполнять проверку, переданы ли параметры на сервер (была ли нажата кнопка), если нет, то выдается форма, если да – результат ее работы.

СВЕРХГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

- **Глобальные переменные –** это любая переменная, объявленная на самом верхнем уровне сценария, т.е. вне функции.
- **Сверхглобальные переменные** – это массивы, встроенные в PHP. Они автоматически заполняются при запуске сценария и доступны из любой части сценария. Доступ к ним можно получить внутри функции или метода.

СВЕРХГЛОБАЛЬНЫЕ МАССИВЫ PHP

Массив	Описание
$_COOKIE	Содержит ключи и значения параметров cookies, полученных от браузера
$_ENV	Содержит ключи и значения переменных сценария
$_FILES	Содержит информацию о полученных файлах
$_GET	Содержит ключи и значения, переданные сценарию с использованием метода get протокола HTTP
$_POST	Содержит ключи и значения, переданные сценарию с использованием метода post протокола HTTP
$_REQUEST	Объединённый массив, содержащий значения, полученных из сверхглобальных массивов $_get, $_post и $_cookies
$_SERVER	Переменные, созданные сервером
$GLOBALS	Содержит все глобальные переменные, связанные с текущим сценарием

ПРИМЕР ПРОСМОТРА СВЕРХГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ $_SERVER

```
<?php
foreach ($_SERVER as $key => $value){
print "\$_SERVER[\"$key\"] == $value <br />";    }
?>
```

$_SERVER["HTTP_HOST"] == localhost
$_SERVER["HTTP_USER_AGENT"] == Mozilla/5.0 (Windows NT 6.1; rv:25.0) Gecko/20100101 Firefox/25.0
$_SERVER["HTTP_ACCEPT"] == text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
$_SERVER["HTTP_ACCEPT_LANGUAGE"] == ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
$_SERVER["HTTP_ACCEPT_ENCODING"] == gzip, deflate
$_SERVER["HTTP_REFERER"] == http://localhost/php/
$_SERVER["HTTP_COOKIE"] == vc=1
$_SERVER["HTTP_CONNECTION"] == keep-alive
$_SERVER["PATH"] == \usr\local\ImageMagick;\usr\local\php5;C:\Program Files\CodeGear\RAD Studio\6.0\bin;C:\Users\Public\Documents\RAD Studio\6.0\Bpl;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Common Files\Ulead Systems\MPEG;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\QuickTime\QTSystem\
$_SERVER["SystemRoot"] == C:\Windows
$_SERVER["COMSPEC"] == C:\Windows\system32\cmd.exe
$_SERVER["PATHEXT"] == .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
$_SERVER["WINDIR"] == C:\Windows
$_SERVER["SERVER_SIGNATURE"] ==
Apache/2.2.4 (Win32) mod_ssl/2.2.4 OpenSSL/0.9.8k PHP/5.2.12 Server at localhost Port 80

$_SERVER["SERVER_SOFTWARE"] == Apache/2.2.4 (Win32) mod_ssl/2.2.4 OpenSSL/0.9.8k PHP/5.2.12
$_SERVER["SERVER_NAME"] == localhost
$_SERVER["SERVER_ADDR"] == 127.0.0.1
$_SERVER["SERVER_PORT"] == 80
$_SERVER["REMOTE_ADDR"] == 127.0.0.1

РАБОТА С ФОРМАМИ

- PHP позволяет обрабатывать данные, которые пользователь ввел в поля формы.
- После активации кнопки **submit** данные отправляются на страницу – обработчик, указанную в поле **action** элемента <form>.
- На странице – обработчике располагается PHP скрипт, который выполняет определенные операции над полученными данными, например формирует и отсылает письмо по указанным пользователем реквизитам.

ПЕРЕДАЧА ДАННЫХ ОБРАБОТЧИКУ

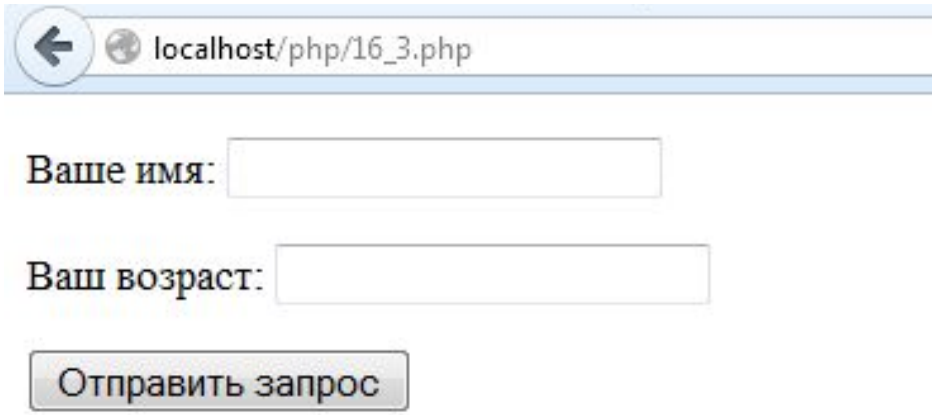
- Данные из формы передаются на сервер как последовательность пар имя/значение.
- Это значит, что имя каждого элемента формы (появляющееся в атрибуте NAME тега) связывается со значением этого элемента (введённым или выбранным пользователем).
- Формат имя/значение, используемый для передачи, имеет вид имя=значение.

ДАННЫЕ ИЗ ФОРМЫ

- Все данные, передаваемые из формы в программу–обработчик располагаются в следующих суперглобальных массивах: $_GET, $_POST, и $_REQUEST.
  - **$_GET[]** — содержит все значения, передаваемые методом GET.
  - **$_POST[]** –содержит все значения, передаваемые методом POST.
  - **$_REQUEST[]** –содержит все значения, передаваемые методами POST и GET.

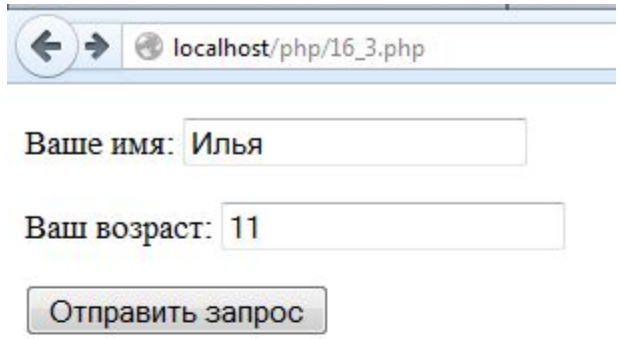
ПРИМЕР ФОРМЫ HTML

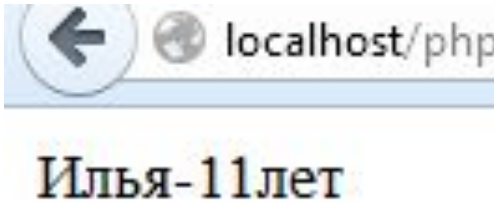
```
<form action="action.php" method="post">
 <p>Ваше имя: <input type="text" name="name" /></p>
 <p>Ваш возраст: <input type="text" name="age" /></p>
 <p><input type="submit" /></p>
</form>
```



ВЫВОДИМ ДАННЫЕ ФОРМЫ

```
<?php
echo htmlspecialchars($_POST['name']);
echo "-";
echo (int)$_POST['age'];
echo "лет";
?>
```

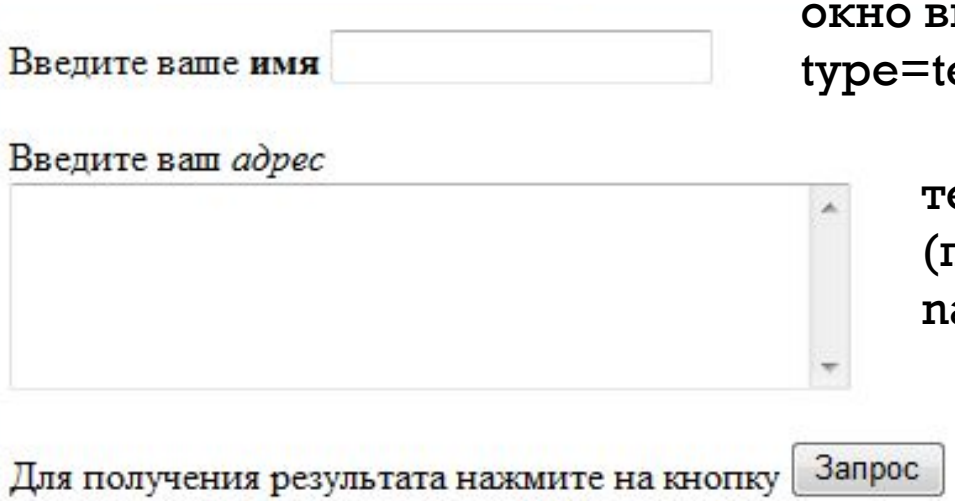




ПЕРЕМЕННАЯ *$_POST["NAME1"]*

- это переменная php-интерпретатора и после нажатия кнопки «ввести» содержит данные, введенные в поле под названием name1-атрибут name тега input.
- строка *value="<?php echo $_POST["name2"]?>* означает, что при загрузке формы после нажатия на кнопку «ввести» данные, введенные пользователем, не будут потеряны, а будут снова отображены в этом поле.

ПРИМЕР GET



окно ввода данных text (параметры: type=text, name=”Пользователь ”)

текстовая область textarea (параметры: rows=5, cols=40, name=”Адрес ”)

кнопка (параметры: type=submit, label =Запрос, метод get)

Для чтения данных из формы сценарий:

```
<?php
print "Добро пожаловать, <b>". $_GET['Пользователь']."</b><br />
\n\n";
print "Ваш адрес: <br /> <b>". $_GET['Адрес']."</b>";
?>
```

МАССИВ $_REQUEST

- Использование суперглобального массива $_Request очень удобно, особенно когда не известно, каким методом были переданы данные.
- Благодаря циклу foreach можно перебрать значения массива $_Request.

```
<?php
foreach($_REQUEST as $key => $value)
{
echo $key;
echo ": ".$value;
echo "<br/>";
}
?>
```

ПРОВЕРКА И ОТЛАДКА ФОРМ

- При работе с формами часто бывает необходимо выполнять проверку введенных пользователем данных. Для этих целей PHP имеет ряд функций:
  - ***is_string()*** – позволяет проверить, является ли переменная строкой.
  - ***is_int()*** – позволяет определить, является ли переменная целым числом.
  - ***is_numeric()*** – позволяет определить, является ли переменная числом.
  - ***is_numeric()*** – позволяет определить, является ли переменная числом с плавающей точкой.
  - ***strlen(string)*** – позволяет определить длинну строки.
  - ***strtolower()*** –преобразует все символы строки в нижний регистр.
  - ***strtoupper(string)*** —преобразует все символы строки в верхний регистр

ПРИМЕР

```
<?php
if ($_POST['submitB'] ==
"Submit")
{
$valid_form = true;
if ($_POST['name'] == "")
{
echo "Введите свое имя";
$valid_form = false;
}
else
{
$name = $_POST['name'];
}
if ($_POST['sname'] == "")
{
echo "Введите фамилию ";
$valid_form = false;
}
else
{ $sname = $_POST['sname']; }
if ($_POST['pass'] == "")
{ echo "Введите пароль";
$valid_form = false;
}
elseif (strlen($_POST['pass']) < 6)
{ echo "Пароль должен содержать не
менее 6 символов";
$valid_form = false; }
else
{ $password = $_POST['pass']; }
if($valid_form == true)
{ echo "Все поля формы заполненны
корректно. Приветствуем вас $name
$sname <br>
Вы авторизовались под паролем
$password<br><br>"; }
}
?>
```

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

- **Регулярное выражение** - это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Т.о. регулярное выражение представляет собой набор символов, описывающих правило поиска подстроки.
- Алгоритм поиска с использованием регулярных выражений был впервые разработан одним из создателей UNIX Кеном Томпсоном.

СИНТАКСИС РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

- Простейшее регулярное выражение: "abc". Это выражение соответствует любой строке, которая содержит подстроку "abc".

ВЫРАЖЕНИЕ В КВАДРАТНЫХ СКОБКАХ

1. Квадратные скобки ограничивают поиск теми символами, которые в них заключены.
   a. "[abc]". Этому регулярному выражению соответствует любая строка, содержащая abc либо вместе, либо каждый из них в отдельности.
   b. Регулярное выражение, соответствующее всем буквам русского алфавита - "[а-Я]" или "[а-яА-Я]".
   c. Регулярные выражения, соответствующие числам: "[0-9]" или "[0123456789]".

ГРУППОВОЙ СИМВОЛ ТОЧКИ "."

1. объединяет два одиночных символа, за исключением символа \n.
   a. .ок - выражение, в частности соответствует строкам "кок", "док", "ток".
   b. "x.[0-9]" – соответствует строке, содержащей символ x, за которым идет любой другой символ и цифры от 0 до 9. Этому критерию, к примеру, удовлетворяют строки "xy1", "xz2" и т. д.

ВЕТВИ

1. разделяются символом |, действующим как оператор OR (ИЛИ). Т. е., если в выражении используются ветви, то для соответствия регулярного выражения какой-либо строке, достаточно, чтобы только одна из ветвей соответствовала этой строке. Ветвление удобно применять при проверке расширений и имен файлов, зон доменных имен и т. д.
   a. "abc|абв" – этому регулярному выражению соответствует любая строка, содержащая подстроки "abc" или "абв".
   b. "ru|com|net" – регулярное выражение проверяет, содержатся ли в строке подстроки "ru", "com" или "net".

ИСКЛЮЧАЮЩЕЕ ВЫРАЖЕНИЕ

1. Для исключения последовательности символов из поиска перед этой ней ставится символ "^".
   a. "[^а-я]" – регулярное выражение отвечает любому символу, не содержащемуся в диапазоне а-я. символ ^ находится внутри квадратных скобок, так как только в этом случае он имеет значение "не". При использовании символа ^ вне квадратных скобок, он имеет совсем иное значение.

КВАЛИФИКАТОРЫ

1. символы +, ?, *. Квалификаторы говорят о том, сколько раз последовательность символов может встретиться в строке и указываются непосредственно после той части выражения, к которой они применяются.
   a. "a+" - хотя бы один a (строки "абв" и "абва" соответствуют этому выражению, а строка "укр" - нет);
   b. "a?" - ноль или один a (строки "абв" и "укр" соответствуют этому выражению, а строка "абва" - нет);
   c. "a*" - ноль или более a (строки "абв" и "абва" и "укр" соответствуют этому выражению).

ГРАНИЦЫ

1. числа в фигурных скобках, указывающие количество вхождений в строку фрагмента выражения, непосредственно предшествующего границе. Для указания количества вхождений не одного символа, а их последовательности, используются круглые скобки.
   a. "xy{2}" соответствует строке, в которой за x следует два y;
   b. "xy{2,}" соответствует строке, в которой за x следует не менее двух y (может быть и больше);
   c. "xy{2,6}" соответствует строке, в которой за x следует от двух до шести y;
   d. "x(yz){2,6}" соответствует строке, в которой за x следует от двух до шести последовательностей yz;
   e. "x(yz)*" соответствует строке, в которой за x следует ноль и более последовательностей yz;

ПОДВЫРАЖЕНИЯ

1. иногда бывает удобно создавать регулярное выражение таким образом, чтобы можно было, к примеру, сказать, что, по крайней мере, за одной из строк "морская", следует точно строка "волна". Для этого регулярное выражение разбивают на **подвыражения** с помощью круглых скобок.
    a. (морская)*волна – это выражение соответствует строкам "волна", "морская волна", "морская морская волна" и т.д.

СООТВЕТСТВИЕ НАЧАЛУ И КОНЦУ СТРОКИ

1. В регулярном выражении можно указать, должно ли конкретное подвыражение встречаться в начале, в конце строки или и в начале и в конце строки. В этом случае символ ^ ставится за пределами выражения в скобках. Знак доллара $ соответствует концу строки.
   a. Символ ^ соответствует началу строки: "^xy". Такое выражение соответствует любой строке, начинающейся с xy.
   b. **"^[a-z]"**

- **"xy$" –** это регулярное выражение соответствует любой строке, заканчивающейся на xy.

СОПОСТАВЛЕНИЕ СО СПЕЦИАЛЬНЫМИ СИМВОЛАМИ

- В тех случаях, когда нужно сопоставить выражение строке, в которой встречаются спецсимволы, такие как $, ^, { и т. д., перед ними ставится символ обратной косой черты (\).
- Например, для того, чтобы найти в строке символ $, в регулярном выражении нужно написать "\$". То же самое относится и к самому символу обратной косой черты.
- Если нужно провести сопоставление с символом обратной косой черты, то в этом случае ставится две обратных косых черты, т. е. \\.

КЛАССЫ СИМВОЛОВ

1. называются сокращенные обозначения для предопределенных символов.
   a. Класс **[[:alnum:]]** - буквенно-цифровые символы
   b. Класс **[[:digit:]]** - десятичные цифровые символы
   c. Класс **[[:xdigit:]]** - шестнадцатеричные цифровые символы
   d. Класс **[[:alpha:]]** - буквенные символы
   e. Класс **[[:upper:]]** - прописные буквенные символы
   f. Класс **[[:lower:]]** - строчные буквенные символы
   g. Класс **[[:punct:]]** - знаки пунктуации
   h. Класс **[[:space:]]** - символы пробела
   i. Класс **[[:blanc:]]** - символы табуляции и пробела
   j. Класс **[[:print:]]** - печатные символы
   k. Класс **[[:cntrl:]]** - управляющие символы
   l. Класс **[[:graph:]]** - печатные символы, за исключением пробельные

- Пример: Эквивалентом выражения "[a-zA-Z_0-9]" является выражение "**[[:alnum:]]**"; Выражению "[0-9]" эквивалентно выражение "**[[:digit:]]**". Выражению "[a-Z]" эквивалентно регулярное выражение "**[[:alpha:]]**".

ФУНКЦИЯ **PREG_MATCH**

- выполняет проверку на соответствие регулярному выражению. Функция возвращает три возможных значения:
- 0 - если совпадений не найдено,
- 1 - если codпадение найдено (после нахождения первого совпадения работа функции прекращается),
- false - если произошла ошибка.

АРГУМЕНТЫ ФУНКЦИИ

- **pattern**
- Обязательный аргумент. Регулярное выражение (шаблон поиска)
- **string_name**
- Обязательный аргумент. Строка, сравниваемая с регулярным выражением.
- **matches**
- Необязательный аргумент. Если совпадение есть, то массив $matches будет заполнен значениями. В $matches[0] будут помещена часть строки полностью соответствующая шаблону, $matches[1] будет содержать текст соответствующей первой маске, $matches[2] текст второй маски и так далее.
- **flags**
- Необязательный аргумент. Данный аргумент может содержать только одно значение PREG_OFFSET_CAPTURE.
- Если этот флаг указан, то в массиве matches будет возвращен массив массивов, где под индексом "0" в первом вложенном массиве будет находиться совпавшая строка, а под индексом "1" ее смещение от начала строки.
- **offset**
- Необязательный аргумент. Аргумент offset указывает позицию в байтах с которой необходимо начать поиск.

ПРОВЕРКА РАБОТЫ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ

Например, строка: «Карл у Клары украл кораллы, а Клара у Карла украла кларнет».

Если применить к этой известной скороговорке регулярное выражение вида:

/(Клар.*?)\s/i,
то мы получим три слова соответствующих данному шаблону: “Клары”, “Клара”, и “кларнет”.

В РЕЗУЛЬТАТЕ РАБОТЫ ФУНКЦИИ ДОЛЖЕН ВЕРНУТЬСЯ МАССИВ:

```
ARRAY (
0 => 'КЛАРЫ',
1 => 'КЛАРА',
2 => 'КЛАРНЕТ',
)
```

```
function TestRegularFirst(){
    $text='Карл у Клары украл кораллы, а Клара у Карла украла кларнет .';
    echo '<strong>Пример:</strong> '.htmlspecialchars($text);
      preg_match_all("/(Клар.*?)\s/i",$text,$result);
    echo '<br/><strong>Результат:</strong> <pre>'.var_export($result[1],true).'</pre>';
  }
  TestRegularFirst();
```

ПРИМЕР РЕГУЛЯРНОГО ВЫРАЖЕНИЯ В PHP

- **function email_check($email) {**
  if
  (!preg_match("/^(?:[a-z0-9]+(?:[-_.]?[a-z0-9]+)?@[a-z0-9_.-]+(?:\.?[a-z0-9]+)?\.[a-z]{2,5})$/i",trim($email)))
  {
  return false;
  }
  else return true;
  }