Лабораторная работа № 7 «Создание формы обратной связи»

Одной из наиболее часто встречающихся на практике задач является реализация формы обратной связи. Тобишь написание ее HTML кода, оформление ее на CSS, создание PHP скрипта, который бы обрабатывал полученные от пользователя данные и отправлял их на нашу почту, написание JS скрипта, который бы проверял форму на адекватность вводимых данных, защита нашего детища от спама, чтобы наш почтовый ящик не обвалился от атак ботов.

HTML

В первую очередь мы пишем HTML код, в нем задаются поля, которые будет заполнять пользователь. Они же в дальнейшем будут оформляться. Код формы выглядит следующим образом:

<form method="post" action="mail.php">

<div **class**="left">

<label for="name">Имя:</label>

<input maxlength="30" type="text" name="name" />

<label for="phone">Телефон:</label>

<input maxlength="30" type="text" name="phone" />

<label for="mail">E-mail:</label>

<input maxlength="30" type="text" name="mail" />

</div>

<div **class**="right">

<label for="message">Сообщение:</label>

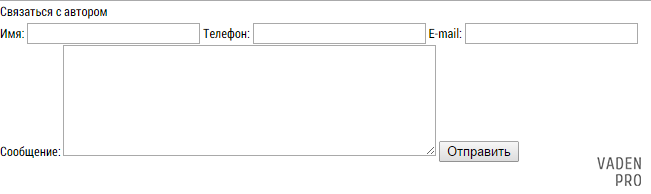
<textarea rows="7" cols="50" name="message"></textarea>

<input type="submit" value="Отправить" />

</div>

</form>

И визуально она выглядит сейчас следующим образом:

[](https://vaden-pro.ru/sites/default/files/forma-html.png)

Согласен, пока все некрасиво и ничего не понятно, но мы только начали.

Рассмотрим приведенный выше код подробно:

* <form method="post" action="mail.php"> … </form>

для того, чтобы создать форму необходимо использовать тег form. Именно он определяет начало и конец формы для интерпретатора кода. У него, как и у любого тега, есть целый набор атрибутов, но обязательных для работы формы всего два, это method (метод отправки запроса на сервер, для форм стандартно используют post) и action (указывает путь к файлу-обработчику формы, именно в этом файле будет содержаться скрипт на PHP, который после будет отправлять введенные пользователем значения нам на почту, в нашем случае мы видим, что это файл называется mail.php и лежит он в том же директории сайта, что и рассматриваемая нами страница).

* <input maxlength="30" type="text" name="name" />

Далее у нас следуют инпуты. Это собственно сами поля формы в которые пользователи будут вводить необходимую нам информацию (type="text" говорит о том, что это будет текст). Атрибут maxlength указывает сколько символов может ввести пользователь в данное поле формы. Самый важный атрибут это name – он задает имя конкретного поля. Именно по этим именам в дальнейшем PHP скрипт будет обрабатывать поступающую в него информацию. При желании можно еще задать атрибут placeholder, который выводит внутри поля текст исчезающий при установке курсора внутри нее. Одной из проблем плейсхолдера является то, что он не поддерживается некоторыми старыми браузерами.

* <label for="name">Имя:</label>

Используется в случае если мы отказались от плейсхолдеров. Обычная подпись поля, атрибут for сообщает к какому конкретно полю относится данная подпись. Значением указывается name интересующего нас поля.

* <textarea rows="7" cols="50" name="message"></textarea>

Также как и инпут предназначен для введения пользователем информации, только на этот раз поле заточено для длинных сообщений. Rows указывает размер поля в строках, cols в символах. В целом они задают высоту и ширину нашего поля.

* <input type="submit" value="Отправить" />

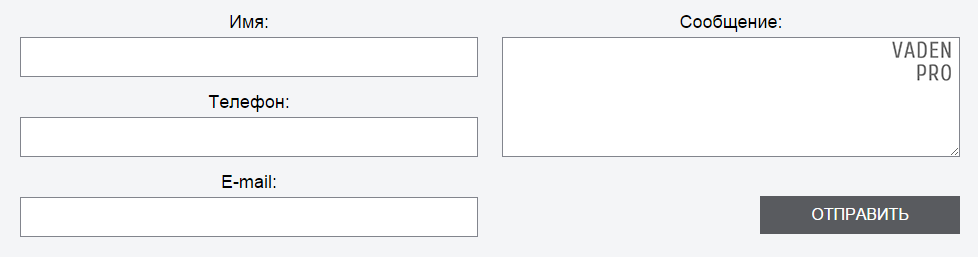
О том, что это кнопка для отправки формы нам сообщает type="submit", а value задает текст, который будет внутри этой кнопки.

* <div **class**="right"> </div>

использованы только для дальнейшего визуального оформления формы.

CSS

Для того, чтобы наша форма обратной связи выглядела презентабельно ее нужно оформить. Для получения следующего результата:

[](https://vaden-pro.ru/sites/default/files/oformlenie-formi.png)

Мы использовали данный код:

form {

background: *#f4f5f7;*

padding: 20px;

}

form .left, form .right {

display: inline-block;

vertical-align: top;

width: 458px;

}

form .right {

padding-left: 20px;

}

label {

display: block;

font-size: 18px;

text-align: center;

margin: 10px 0px 0px 0px;

}

input, textarea {

border: 1px solid *#82858D;*

padding: 10px;

font-size: 16px;

width: 436px;

}

textarea {

height: 98px;

margin-bottom: 32px;

}

input[type="submit"] {

width: 200px;

float: right;

border: none;

background: *#595B5F;*

color: *#fff;*

text-transform: uppercase;

}

Подробно расписывать CSS я не вижу смысла, обращу Ваше внимание лишь на ключевые моменты:

1. Не стоит писать оформление под каждый тег в форме. Старайтесь строить свои селекторы так, чтобы парой строк кода оформлять все необходимые Вам элементы.
2. Не используйте для переноса строк и создания отступов лишние теги по типу <br>,<p> и тд, с этими задачами прекрасно справляется CSS со свойством display: block и margin с padding. Больше о том, почему не стоит пользоваться <br> в верстке вообще можете почитать в статье [Тэг br, а так ли он нужен?](https://vaden-pro.ru/teg-br-tak-li-nuzhen).
3. Не стоит пользоваться табличной версткой для форм. Это противоречит семантике этого тега, а поисковики любят семантичный код. Для того, чтобы формировать визуальную структуру документа нам достаточно тегов div, и заданных им в CSS свойств display: inline-block (выстраивает блоки в ряд) и vertical-align: top (не дает им разбежаться по экрану), задаем им необходимую высоту и вуаля, ничего лишнего и все расположено так, как нам нужно.

Для желающих экономить свое время на оформлении сайтов могу посоветовать пользоваться CSS фреймворками при создании сайтов, особенно самописных. Мой выбор в этом плане- [Twitter Bootstrap](https://vaden-pro.ru/twitter-bootstrap-chto-eto" \o "Twitter Bootstrap). Урок по оформлению форм с его использованием можно посмотреть [тут](https://vaden-pro.ru/oformlyaem-formy-s-bootstrap).

PHP

Ну вот и пришло время сделать нашу форму работоспособной.

Заходим в наш корневой каталог сайта и создаем там файл mail.php, к которому мы ранее указывали путь в атрибуте action тега form.

В конечном итоге его код будет выглядеть следующим образом:

<!doctype html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

<title>Ваше сообщение успешно отправлено</title>

</head>

<body>

**<?php**

$back = "<p><a href=**\"**javascript: history.back()**\"**>Вернуться назад</a></p>";

if(!empty($\_POST['name']) and !empty($\_POST['phone']) and !empty($\_POST['mail'])

and !empty($\_POST['message'])){

$name = trim(strip\_tags($\_POST['name']));

$phone = trim(strip\_tags($\_POST['phone']));

$mail = trim(strip\_tags($\_POST['mail']));

$message = trim(strip\_tags($\_POST['message']));

mail('почта\_для\_получения\_сообщений@gmail.com', 'Письмо с адрес\_вашего\_сайта',

'Вам написал: '.$name.'<br />Его номер: '.$phone.'<br />Его почта: '.$mail.'<br />

Его сообщение: '.$message,"Content-type:text/html;charset=windows-1251");

echo "Ваше сообщение успешно отправлено!<Br> Вы получите ответ в

ближайшее время<Br> **$back**";

exit;

}

else {

echo "Для отправки сообщения заполните все поля! **$back**";

exit;

}

**?>**

</body>

</html>

Обсуждение HTML и CSS части этого документа можно пропустить. По своей сути это обычная страница сайта, которую Вы можете оформить по своему желанию и необходимости. Рассмотрим же важнейшую ее часть – PHP скрипт обработки формы:

$back = "<p><a href=**\"**javascript: history.back()**\"**>Вернуться назад</a></p>";

Этой строкой мы создаем ссылку для возвращения на предыдущую страницу. Поскольку мы заранее не знаем с какой страницы пользователь попадет на эту, то делается это при помощи маленькой JS функции. В дальнейшем мы просто будем обращаться к этой переменной для вывода ее в нужных нам местах.

if(!empty($\_POST['name']) and !empty($\_POST['phone'])

and !empty($\_POST['mail']) and !empty($\_POST['message'])){

*//внутрення часть обработчика*

}

else {

echo "Для отправки сообщения заполните все поля! **$back**";

exit;

}

Тут мы прикручиваем проверку формы на наполненность полей. Как вы догадались, в части $\_POST['name'] в кавычках мы пишем значение атрибута name наших инпутов.

Если все поля заполнены, то скрипт начнет обрабатывать данные в своей внутренней части, если же хоть одно поле не было заполнено, то на экран пользователя выведется сообщение с требованием заполнить все поля формы echo "Для отправки сообщения заполните все поля! $back" и ссылкой для возврата на предыдущую страницу, которую мы создали самой первой строкой.

Дале вставляем во внутреннюю часть обработчика формы:

$name = trim(strip\_tags($\_POST['name']));

$phone = trim(strip\_tags($\_POST['phone']));

$mail = trim(strip\_tags($\_POST['mail']));

$message = trim(strip\_tags($\_POST['message']));

Таким образом мы очистили вводимые пользователем данные от html тегов и лишних пробелов. Это позволяет нам обезопасить себя от получения вредоносного кода в высылаемых нам сообщениях.

Проверки можно и усложнить, но это уже по вашему желанию. Минимальную защиту на серверной стороне мы уже поставили. Дальнейшее мы сделаем на стороне клиента используя JS.

Полностью отказываться от защиты формы на серверной стороне в пользу JS не рекомендую, поскольку хоть и крайне редко, но встречаются уникумы с отключенным в браузере JS.

После чистки тегов добавляем отправку сообщения:

mail('почта\_для\_получения\_сообщений@gmail.com', 'Письмо с адрес\_вашего\_сайта',

'Вам написал: '.$name.'<br />Его номер: '.$phone.'<br />Его почта: '.$mail.'<br />

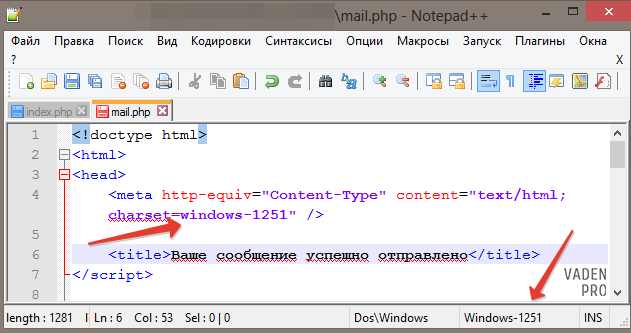
Его сообщение: '.$message,"Content-type:text/html;charset=windows-1251");

Именно эта строка и занимается формированием и отправкой сообщения к нам. Заполняется она следующим образом:

1. 'почта\_для\_получения\_сообщений@gmail.com' – сюда между кавычек вставляете свою почту
2. 'Письмо с адрес\_вашего\_сайта' – это тема сообщения, которое будет приходить на почту. Можно написать сюда что угодно.
3. 'Вам написал: '.$name.'<br />Его номер: '.$phone.'<br />Его почта: '.$mail.'<br />Его сообщение: '.$message – формируем сам текст сообщения. $name – вставляем информацию заполненную пользователем через обращение к полям из предыдущего шага, в кавычках описываем что значит это поле, тегом <br /> делаем перенос строки, чтобы сообщение в целом было читабельно.
4. Content-type:text/html;charset=windows-1251 - в конце идет явное указание типа данных передаваемого в сообщении и его кодировки.

**ВАЖНО!**

Кодировка указанная в «голове» документа (<meta http-equiv="Content-Type"content="text/html; charset=windows-1251" />), кодировка из сообщения Content-type:text/html;charset=windows-1251 и в целом кодировка файла PHP должны совпадать иначе в получаемых на почту сообщениях вместо русских или английских букв будут выводиться «кракозябры».

[](https://vaden-pro.ru/sites/default/files/kodirovka-faila.png)

Многие не указывают кодировку отправляемого сообщения явно, но на некоторых почтовых клиентах из-за этого в дальнейшем могут возникнуть проблемы (на почту приходят нечитабельные письма), потому я рекомендую ее все-же указывать.

Проверка формы на адекватность вводимых данных

Чтобы пользователи по невнимательности не промахивались полями и заполняли все корректно стоит поставить проверку вводимых данных.

Это можно сделать как на PHP на стороне сервера, так и на JS на стороне клиента. Я использую второй вариант, поскольку так человек сразу может узнать что он сделал не верно и исправить ошибку не делая дополнительных переходов по страницам.

Код скрипта вставляем в том же файле, где у нас располагается HTML часть формы. Для нашего случая он будет выглядеть так:

<script>

**function** checkForm(form) {

**var** name = form.name.value;

**var** n = name.match(/^[A-Za-zА-Яа-я ]\*[A-Za-zА-Яа-я ]+$/);

if (!n) {

alert("Имя введено неверно, пожалуйста исправьте ошибку");

return **false**;

}

**var** phone = form.phone.value;

**var** p = phone.match(/^[0-9+][0-9- ]\*[0-9- ]+$/);

if (!p) {

alert("Телефон введен неверно");

return **false**;

}

**var** mail = form.mail.value;

**var** m = mail.match(/^[A-Za-z0-9][A-Za-z0-9\.\_-]\*[A-Za-z0-9\_]\*@([A-Za-z0-9]+([A-Za-z0-9-]\*

[A-Za-z0-9]+)\*\.)+[A-Za-z]+$/);

if (!m) {

alert("E-mail введен неверно, пожалуйста исправьте ошибку");

return **false**;

}

return **true**;

}

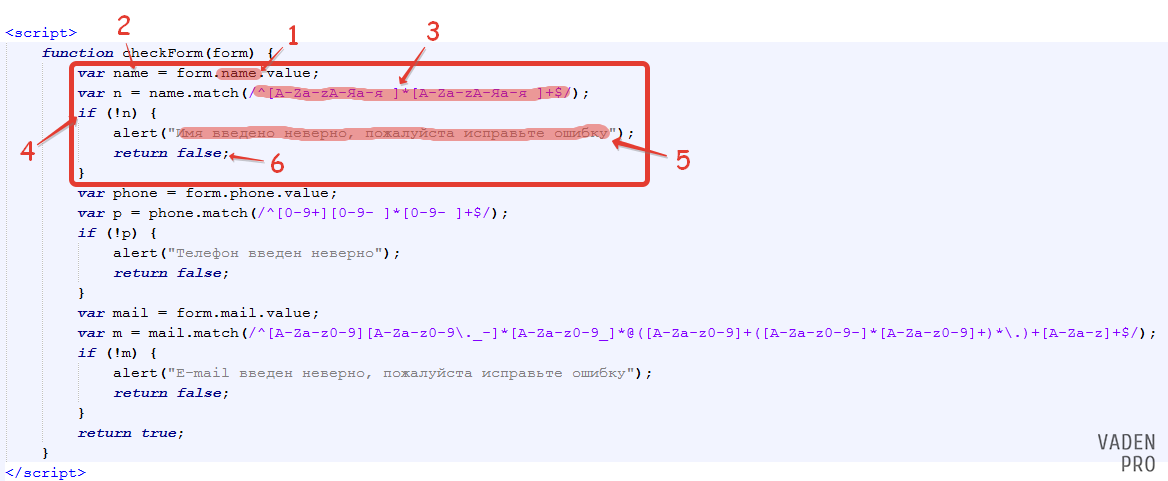
**</script>**

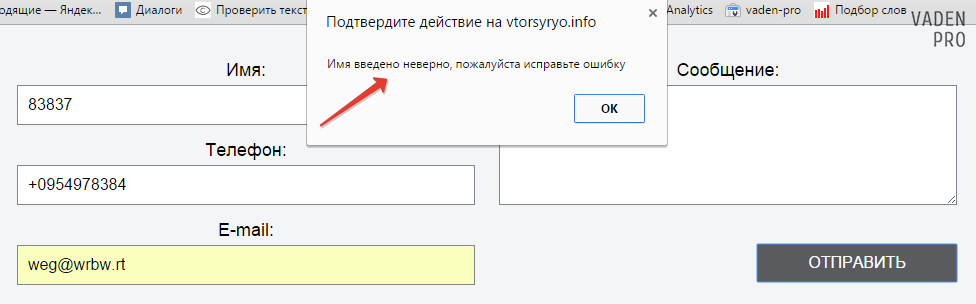
Ну а теперь обычный разбор:

Для того, **чтобы при нажатии на кнопку отправки формы, у нас происходила ее проверка** вешаем запуск нашего скрипта на тег form:

<form method="post" action="mail.php" onSubmit="return checkForm(this)">

Теперь по пунктам забираем состав проверки:

[](https://vaden-pro.ru/sites/default/files/proverka-formi.png)

1. Обращаемся к нашей форме и по значению атрибута name выбираем интересующий нас инпут.
2. Записываем его содержимое в переменную
3. Собственно сама проверка формы на совпадение с заданными нами символами. (/^[A-Za-zА-Яа-я ]\*[A-Za-zА-Яа-я ]+$/ - могут использоваться только буквы русского и английского алфавитов; /^[0-9+][0-9- ]\*[0-9- ]+$/ - первый символ это «+» или цифра, дальше могут быть только цифры; /^[A-Za-z0-9][A-Za-z0-9\.\_-]\*[A-Za-z0-9\_]\*@([A-Za-z0-9]+([A-Za-z0-9-]\*[A-Za-z0-9]+)\*\.)+[A-Za-z]+$/ - первые несколько символов это английские буквы или точки, потом знак @, потом снова английские буквы, потом точка и потом еще несколько английских букв).
4. Если все верно, то проверка ничего не выводит и идет к следующему блоку кода, если же проверка на символы выдала ошибку, то выполняется часть скрипта с действиями 5 и 6.
5. Пользователю выводится это сообщение о ошибке.  
   [](https://vaden-pro.ru/sites/default/files/soobchenie-o-oshibke.png)
6. Происходит выход из скрипта проверки.

Как можете заметить такая мини проверка пишется для каждого нашего поля. Проверку на одно поле я выделил на скриншоте красным квадратом, у других полей она имеет аналогичную структуру и если возникнет необходимость добавить поле или убрать его, Вы теперь с легкостью сможете это сделать.