

Working Bootstrap Contact form with PHP and AJAX

Tutorial by Ondrej Svestka

Bootstrapious.com

Today I would like to show you how to easily build a **working contact form** using **Bootstrap** framework and **AJAX** with **PHP**. You will need some basic knowledge of HTML, CSS and Bootstrap CSS framework.

In the tutorial, I will walk you through the parts that will show you **how to code the contact form in HTML**, style it a bit and add real-time validation to the required fields.

Then I will show you **how the data is handled and email sent in the PHP script**.

As last, you will learn how little **JavaScript** magic (using jQuery) is needed to **submit the form via AJAX without reloading the page itself**. This will be really handy, when you have the contact form on one page web and you don't want to reload whole page.

The result of this tutorial will be working responsive contact form with fields validation and with some basic CSS styling. It will look like on the picture on the next page.

Contact form Tutorial from [Bootstrapious.com](https://bootstrapious.com)

This is a demo for our tutorial dedicated to crafting working Bootstrap contact form with PHP and AJAX background.

Contact form succesfully submitted. Thank you, I will get back to you soon!



Firstname *

Please enter your firstname *

Lastname *

Please enter your lastname *

Email *

Please enter your email *

Phone

Please enter your phone

Message *

Message for me *

SEND MESSAGE

* These fields are required.

HTML template

So let's start with the **main HTML layout**.

There **should not be anything tricky for you**, so just few words about it:

- In the **head** we include Bootstrap stylesheet, *Lato* from Google fonts, and local **custom.css** stylesheet.
- before the **<body>** closing tag we include *jQuery*, *Bootstrap scripts* and local *contact.js* file which will handle AJAX sending of the form
- for Bootstrap, jQuery and a Font I used their **CDN versions**. If you will test the scripts without internet connection, don't forget to include their **local versions**

The form - HTML code

As we have the main layout ready, we can replace the `<form></form>` in our HTML code with our fancy **Bootstrap Contact form** itself.

At the end, our form should look like this:

```
<form id="contact-form" method="post" action="contact.php" role="form">
  <div class="messages"></div>
  <div class="controls">
    <div class="row">
      <div class="col-md-6">
        <div class="form-group">
          <label for="form_name">Firstname *</label>
          <input id="form_name" type="text" name="name" class="form-control" placeholder="Please
enter your firstname *" required="required" data-error="Firstname is required.">
          <div class="help-block with-errors"></div>
        </div>
      </div>
      <div class="col-md-6">
        <div class="form-group">
          <label for="form_lastname">Lastname *</label>
          <input id="form_lastname" type="text" name="surname" class="form-control"
placeholder="Please
enter your lastname *" required="required" data-error="Lastname is required.">
          <div class="help-block with-errors"></div>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-6">
        <div class="form-group">
          <label for="form_email">Email *</label>
          <input id="form_email" type="email" name="email" class="form-control" placeholder="Please
enter your email *" required="required" data-error="Valid email is required.">
          <div class="help-block with-errors"></div>
        </div>
      </div>
      <div class="col-md-6">
        <div class="form-group">
          <label for="form_phone">Phone</label>
          <input id="form_phone" type="tel" name="phone" class="form-control" placeholder="Please
enter your phone">
          <div class="help-block with-errors"></div>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-12">
        <div class="form-group">
```

```

        <label for="form_message">Message *</label>
        <textarea id="form_message" name="message" class="form-control" placeholder="Message for
me *" rows="4" required="required" data-error="Please,leave us a message."></textarea>
        <div class="help-block with-errors"></div>
    </div>
</div>
<div class="col-md-12">
    <input type="submit" class="btn btn-success btn-send" value="Send message">
</div>
</div>
<div class="row">
    <div class="col-md-12">
        <p class="text-muted"><strong>*</strong> These fields are required. Contact form template by
<a href="http://bootstrapious.com" target="_blank">Bootstrapious</a>.</p>
    </div>
</div>
</div>
</form>

```

Well, this looks a bit more interesting already, so let's have a look at all the pieces of the puzzle now:

- We will **send the contact form values via POST to the PHP script** called `contact.php`. As there could be more forms on the page (search etc.), we mark our form with `#contact-form` id to address it correctly. There is also empty div `.messages` that will serve us to display the success or error message after sending the form via AJAX.
- Standard **Bootstrap forms markup** is used - rows, columns, form groups.
- For the **validation of the fields**, we will be using great **Bootstrap validator**. Validation rules are specified on form inputs via the standard HTML5 attributes, in our case these are `required` and `type="email"`.
- We will be using **custom error messages** for each field, passed to the script in the `data-error` attribute.
- To display the possible errors, there is a `<div class="help-block with-errors"></div>` block added to each `form-group`
- Also note `type="email"` and `type="tel"` inputs that will **enhance the user experience**, especially if using the mobile device.

This should basically do for the HTML part of the form and we can move on to the PHP script.

PHP script

The **PHP script** that will handle the **email sending** is located in the `contact.php` file.

In the first part of the script, we **configure the basic variables** we will need. These are:

- `$from` - email address that will be in the From field of the email.
Important: To avoid being marked as spam, use email on your domain: so if you will be using the form on mygreatsite.com, use `'info@mygreatsite.com'` in this variable.
- `$sendTo` - email address that will receive the email with the output of the form. Can be your personal address or can be same as the address as in `$from` variable. Make sure this email exists.
- `$subject` - subject of the email
- `$fields` - array of form control names and their english counterparts. If we have have input called name `<input name="name">`, we can call it in our email e.g. *Customer Name* like this: `'name' => 'Customer Name'`
- `$okMessage` - the message text displayed on the webpage, when message is succesfully sent
- `$errorMessage` - the text of the message displayed in case of an error

We will wrap the whole code block in `try/catch` block which will catch all the possible errors.

The code works as follows:

1. we **start building the email message** content in `$emailText`
2. we **iterate through the `$_POST`** (array containing all the values sent through the POST request)
3. if we find out that the key of the item from `$_POST` array also exists in our `$fields` array, we **include it to the text of the message** in `$emailText`
4. we **send the email** via PHP internal `mail()` function
5. we create `$responseArray` variable to be sent as **JSON response** back to our `index.html` to be handled by our JavaScript function
6. if the request came via AJAX (you check this in PHP using the condition `if(!empty($_SERVER['HTTP_X_REQUESTED_WITH']) && strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest')`) we **send the JSON response**, if not we simply display the message (this should be rare case - e.g. for users with disabled JavaScript)

```

<?php

// configure
$from = 'Demo contact form <demo@domain.com>';
$sendTo = 'Demo contact form <demo@domain.com>';
$subject = 'New message from contact form';
$fields = array('name' => 'Name', 'surname' => 'Surname', 'phone' => 'Phone', 'email' =>
'Email', 'message' => 'Message'); // array variable name => Text to appear in email
$okMessage = 'Contact form successfully submitted. Thank you, I will get back to you
soon!';
$errorMessage = 'There was an error while submitting the form. Please try again later';

// let's do the sending

try
{
    $emailText = "You have new message from contact
form\n=====\\n";

    foreach ($_POST as $key => $value) {

        if (isset($fields[$key])) {
            $emailText .= "$fields[$key]: $value\\n";
        }
    }

    mail($sendTo, $subject, $emailText, "From: " . $from);

    $responseArray = array('type' => 'success', 'message' => $okMessage);
}
catch (\Exception $e)
{
    $responseArray = array('type' => 'danger', 'message' => $errorMessage);
}

if (!empty($_SERVER['HTTP_X_REQUESTED_WITH']) &&
strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest') {
    $encoded = json_encode($responseArray);

    header('Content-Type: application/json');

    echo $encoded;
}
else {
    echo $responseArray['message'];
}

```

Technical requirements for the PHP are:

- PHP >= 5.3 and email server setup

A little bit of Javascript

The JavaScript part of this tutorial will handle the validation of the form and its sending via AJAX. We will save it to `contact.js`.

First, we will run the validator script on our contact form.

Then, we will add some **JavaScript** that will help us with the **submitting of the form via AJAX** request.

It works like this:

1. When the form with the `#contact-form` id is submitted, we make the **POST request** to `contact.php` script.
2. On request's success we work with the **JSON object** that is **returned by the PHP script**. The object has only two properties - *type* and *message*
3. We use type and message to **construct the message visible for the user** - in case of error we display `alert-danger`, in case of success we display `alert-success`
4. We **display the message**, reset form inputs and `return false`; to prevent the usual form submitting

```
$(function () {  
    $('#contact-form').validator();  
    $('#contact-form').on('submit', function (e) {  
        if (!e.isDefaultPrevented()) {  
            var url = "contact.php";  
            $.ajax({  
                type: "POST",  
                url: url,  
                data: $(this).serialize(),  
                success: function (data)  
                {  
                    var messageAlert = 'alert-' + data.type;  
                    var messageText = data.message;  
                    var alertBox = '<div class="alert ' + messageAlert + ' alert-dismissable"><button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>' + messageText + '</div>';
```

```
        if (messageAlert && messageText) {  
            $('#contact-form').find('.messages').html(alertBox);  
            $('#contact-form')[0].reset();  
        }  
    }  
});  
return false;  
}  
})  
});
```

Conclusion

So, that's all for today.

Now you should have a **great contact form working** and to be **ready to be implemented** on your website.

As always there could be further modifications done as e.g. Captcha - I will try to include some of these in one of the next updates of this tutorial.

If you liked the article - let your friends know about it - remember: *sharing is caring* :)

Thanks!

If you haven't done that already, have a look at my [free Bootstrap themes](#).

Ondrej

Last updated on 16th May, 2016

Development on your computer

If you want to **develop** the form on your computer, you will need local server with **PHP support**. One of the most common solutions is e.g. **XAMPP** (I use it myself), but there are many more.

The localhost solution, be it XAMPP or similar, **will most probably not have any working mail server included**. To be able to debug/develop the emails, there are **utilities, that simulate the mail server behaviour** and instead of sending the email, they save the output/source code of the email to your harddrive. If you are using XAMPP, it comes with the utility called mailtodisk and emails will be saved in xamppfolder/mailoutput. One of the similar solutions can be e.g. **MailCatcher**, but I don't have any personal experience with it.

To be able to send real email messages, you will need to put the script/page to the internet. To order great web hosting, have a look at **iPage** - one of the most reliable web hosts on the market. If you are new to them, to will also give you 80% off + many extras.



Tools to Grow Your Website's Traffic

And yes, they work on any website.