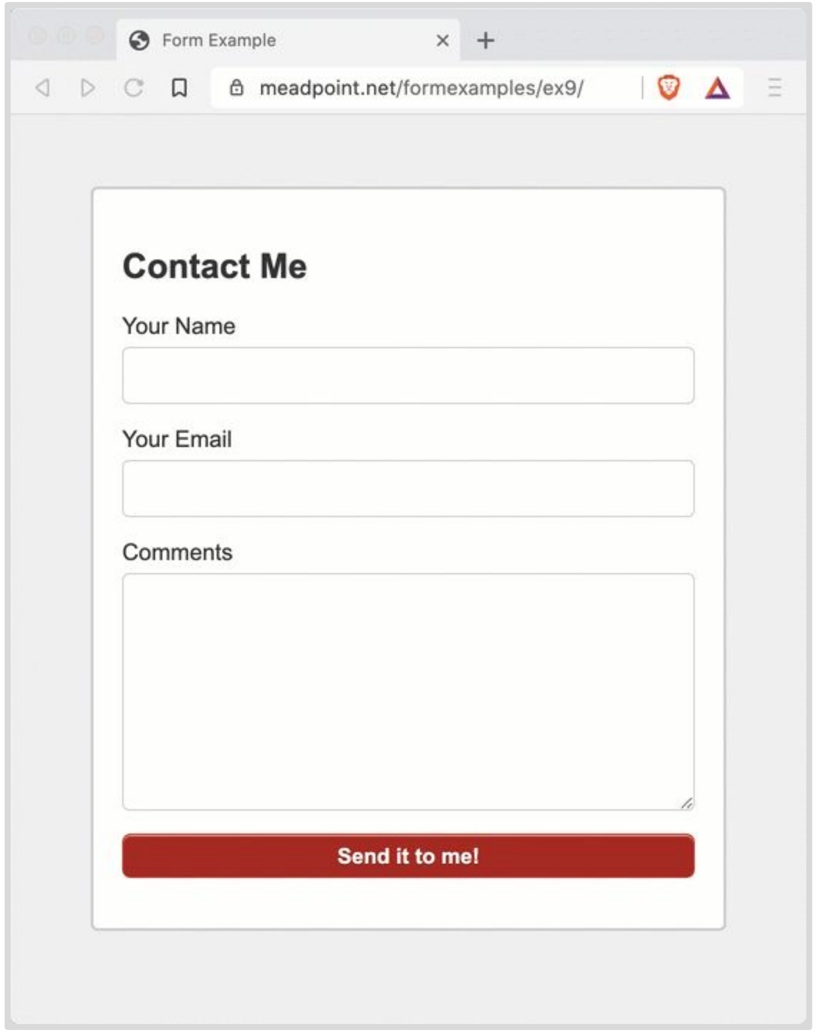
Async Sending Data

With JavaScript

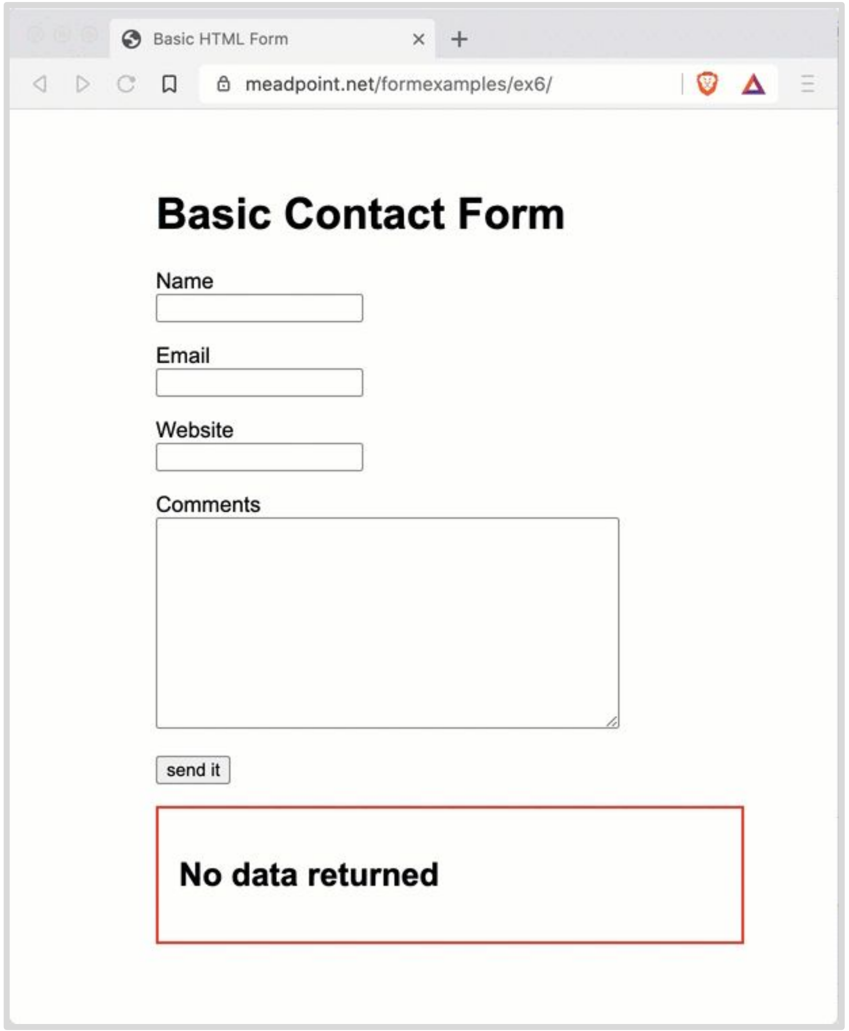
End Result

For this project, you will explore ways of sending data asynchronously to a web server and as well as getting data back from the server.



Basic jQuery Version

First start with the basic version using jQuery. This version will use the validator jQuery plugin to validate the data, then take the data, if it passes, and send it to the server to be processed.



Basic Contact Form

Name

Email

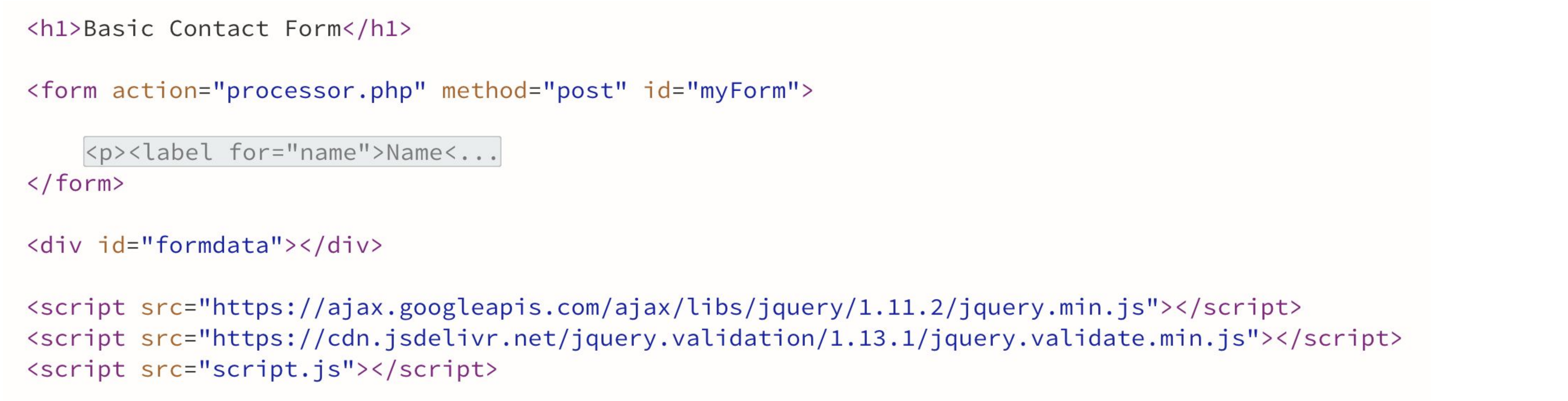
Website

Comments

send it

No data returned

HTML File



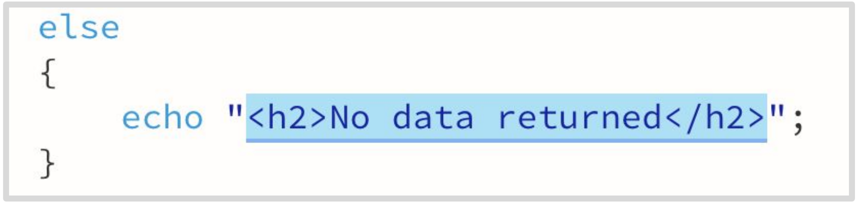
```
<h1>Basic Contact Form</h1>

<form action="processor.php" method="post" id="myForm">

    <p><label for="name">Name<...
</form>

<div id="formdata"></div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/jquery.validation/1.13.1/jquery.validate.min.js"></script>
<script src="script.js"></script>
```

The HTML file includes a form, plus links to jQuery, the jQuery validator plugin and your script file.

There is a little basic CSS at the top of the page for styling purposes.

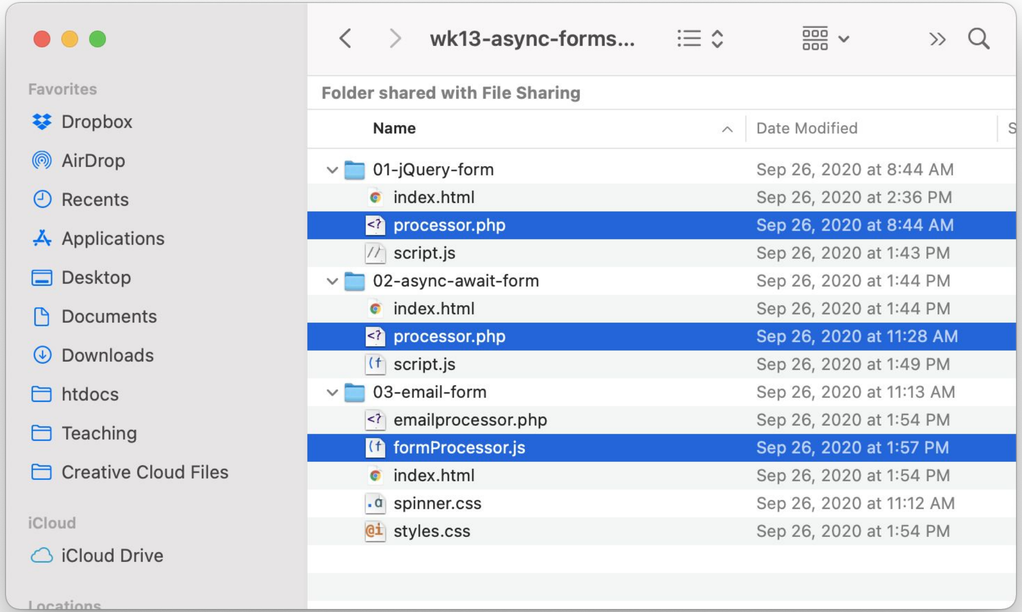
The PHP File

If you look at the PHP file, you will see that it takes data from the form, makes sure the fields aren't empty, and that the email address is actually an email address, then simply echos back the data.


```
else
{
        echo "<h2>No data returned</h2>";
}
```

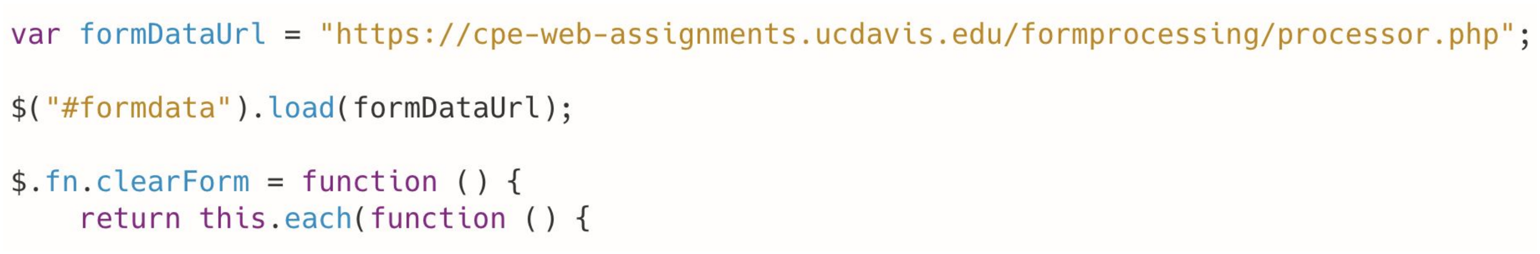
On the Server

PHP has to run on a server, so even though the PHP files are included in this project, you won't actually need to use them. They are there for reference.



Name	Date Modified
01-jQuery-form	Sep 26, 2020 at 8:44 AM
index.html	Sep 26, 2020 at 2:36 PM
processor.php	Sep 26, 2020 at 8:44 AM
script.js	Sep 26, 2020 at 1:43 PM
02-async-await-form	Sep 26, 2020 at 1:44 PM
index.html	Sep 26, 2020 at 1:44 PM
processor.php	Sep 26, 2020 at 11:28 AM
script.js	Sep 26, 2020 at 1:49 PM
03-email-form	Sep 26, 2020 at 11:13 AM
emailprocessor.php	Sep 26, 2020 at 1:54 PM
formProcessor.js	Sep 26, 2020 at 1:57 PM
index.html	Sep 26, 2020 at 1:54 PM
spinner.css	Sep 26, 2020 at 11:12 AM
styles.css	Sep 26, 2020 at 1:54 PM

Loading the PHP file



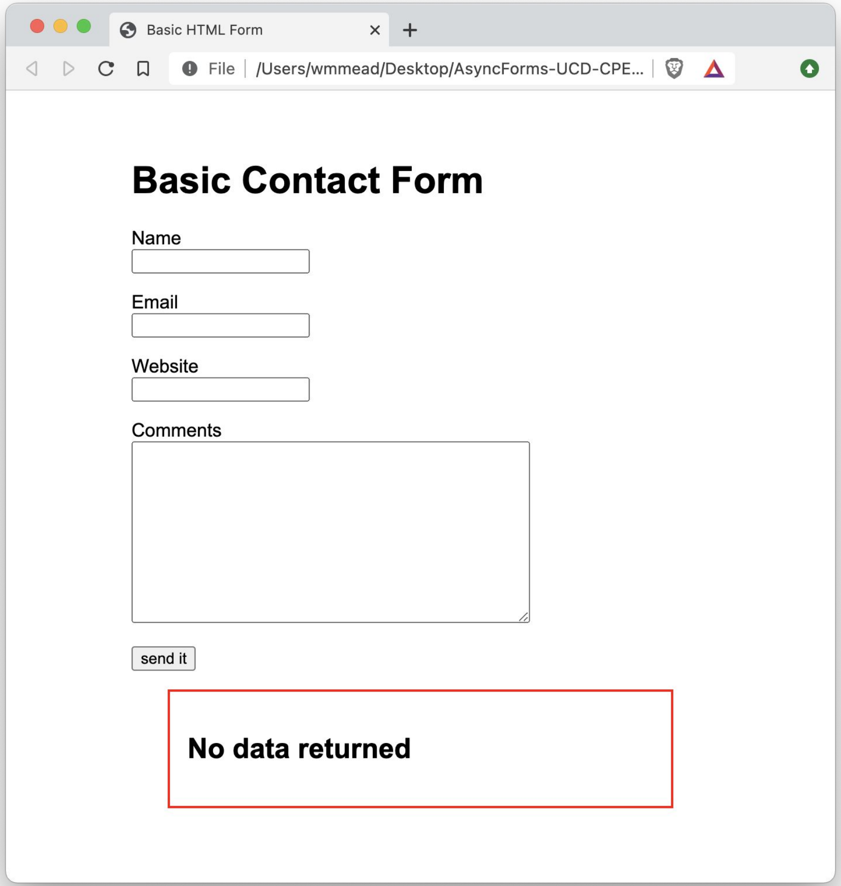
```
var formDataUrl = "https://cpe-web-assignments.ucdavis.edu/formprocessing/processor.php";

$("#formdata").load(formDataUrl);

$.fn.clearForm = function () {
    return this.each(function () {
```

On the script.js file, you will see there is a function for clearing the form.

You can ignore that for now and just add the first two lines, which will use the jQuery.load() function, to load the PHP file into the <div id=”formdata”>

Getting the Page

Even running this file right in your browser should go to the PHP file on the server and bring back "No data returned"



Basic Contact Form

Name

Email

Website

Comments

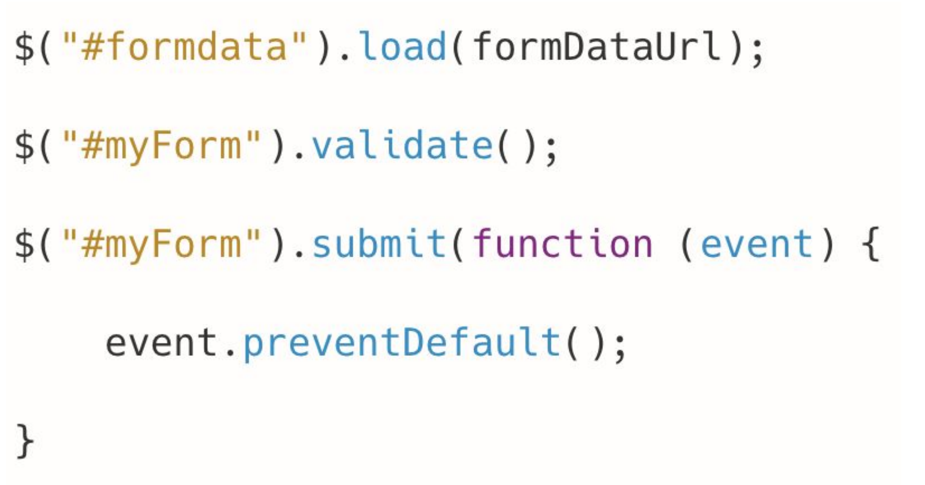
send it

No data returned

Validate the Form and Catch Submission

After you load data from the PHP file, add a line to validate the form. You are just using the very basic version of the validator plugin for this project.

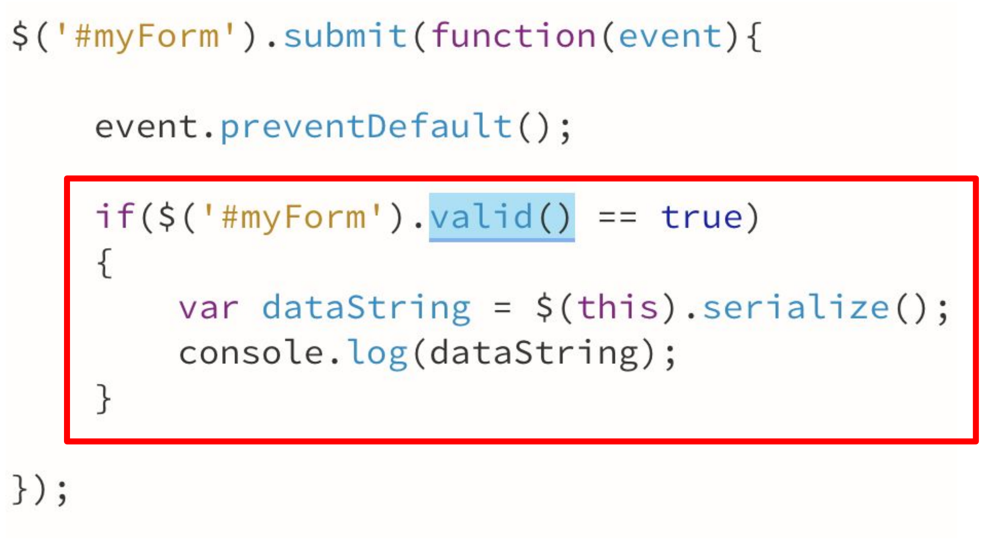
Then add an event handler for when the form is submitted. Pass in the special event object and use it to prevent the default behavior.



```
$("#formdata").load(formDataUrl);

$("#myForm").validate();

$("#myForm").submit(function (event) {

    event.preventDefault();

}
```

If the form is valid

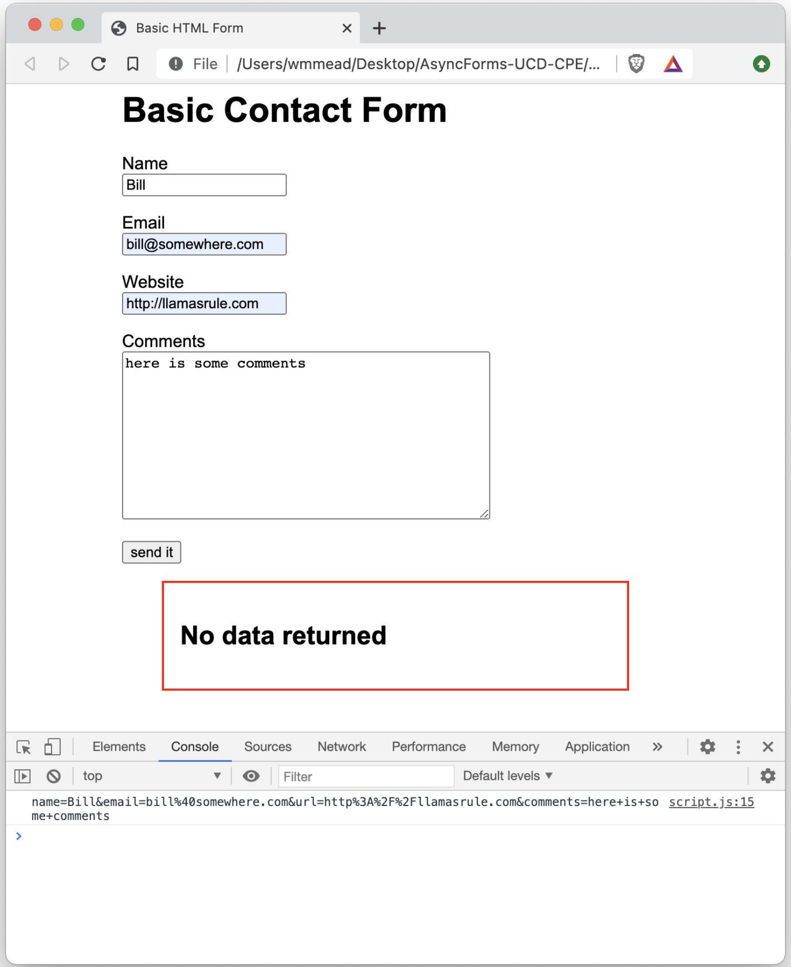
Inside the function that runs when the form is submitted, add an if statement that checks if the form is valid.



```
$('#myForm').submit(function(event){

    event.preventDefault();

    if($('#myForm').valid() == true)
    {
        var dataString = $(this).serialize();
        console.log(dataString);
    }

});
```

Seeing the Data

Test the file and check the console. You should see the data from the form, if you fill it out and click the send button.



Basic Contact Form

Name: Bill

Email: bill@somewhere.com

Website: http://llamasrule.com

Comments: here is some comments

send it

No data returned

```
name=Bill&email=bill%40somewhere.com&url=http%3A%2F%2Fllamasrule.com&comments=here+is+so  script.js:15
me+comments
```

Sending the Data

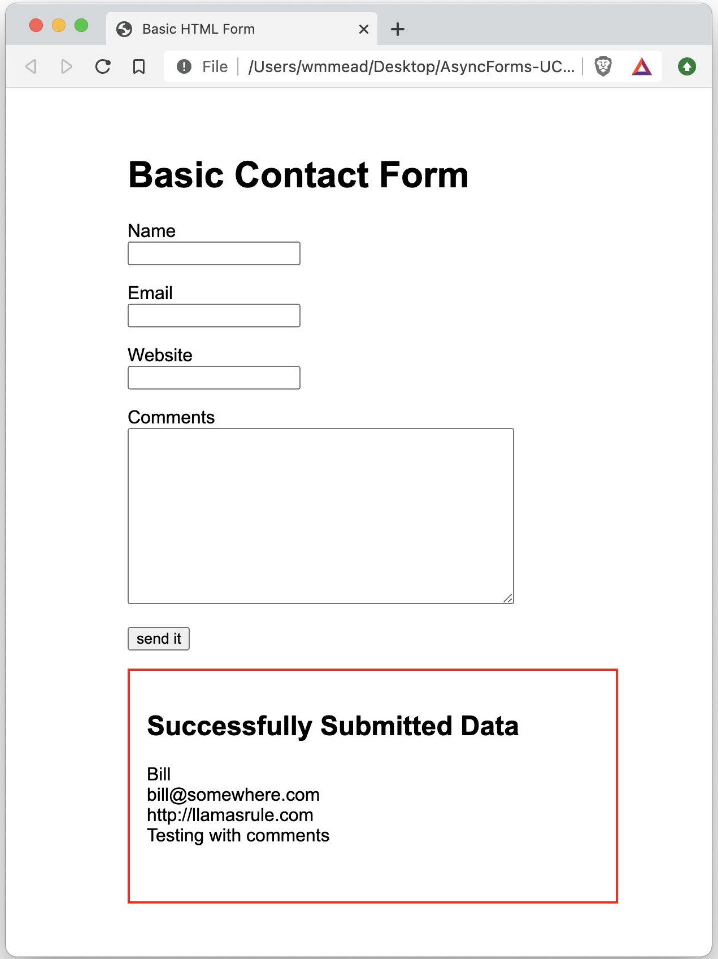
```
if ($('#myForm').valid() == true) {
    var dataString = $(this).serialize();
    //console.log(dataString);
    $.ajax({
        type: "POST",
        url: formDataUrl,
        data: dataString,
        success: function (data) { $("#formdata").html(data); $('#myForm').clearForm(); }
    });
}
```

Finally, inside the if statement, add the call to the jQuery ajax method. This method takes an object as an argument. The type of data is POST, the url is the variable that holds the link to the processor.php page.

The data it is sending is dataString, then upon success, run a function that passes in the data form the server. Put that data in the div and clear out the form.

End Result

If you fill out the form and submit it, it should work.



Basic Contact Form

Name

Email

Website

Comments

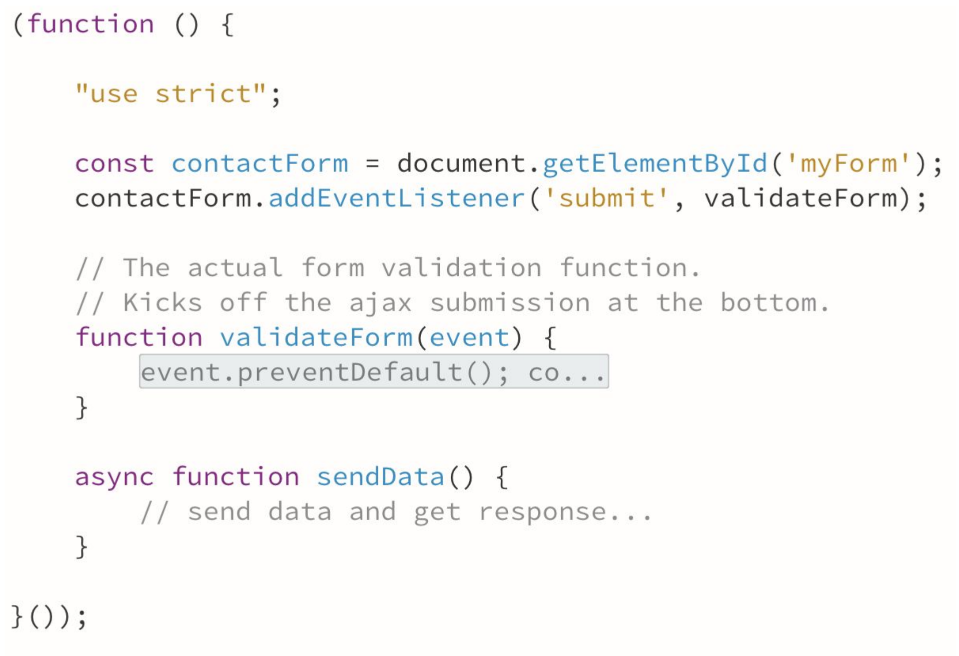
send it

Successfully Submitted Data

Bill
bill@somewhere.com
http://llamasrule.com
Testing with comments

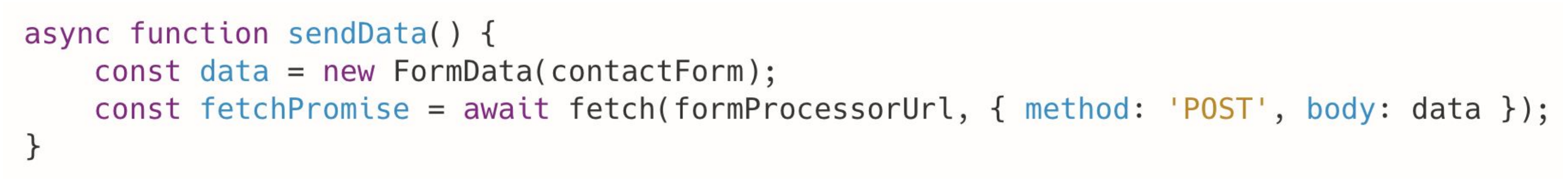
Async / Await Version

Next you will do the same thing without jQuery. We will skip the vanilla JS XMLHttpRequest version and just look at the Async / Await version, since that is the future.



```
(function () {

    "use strict";

    const contactForm = document.getElementById('myForm');
    contactForm.addEventListener('submit', validateForm);

    // The actual form validation function.
    // Kicks off the ajax submission at the bottom.
    function validateForm(event) {
        event.preventDefault(); co...
    }

    async function sendData() {
        // send data and get response...
    }

}());
```

Using FormData and Fetch



```
async function sendData() {
    const data = new FormData(contactForm);
    const fetchPromise = await fetch(formProcessorUrl, { method: 'POST', body: data });
}
```

Add these two lines. The first line uses the FormData API to take data from a form and do what the .serialize() jQuery function does.

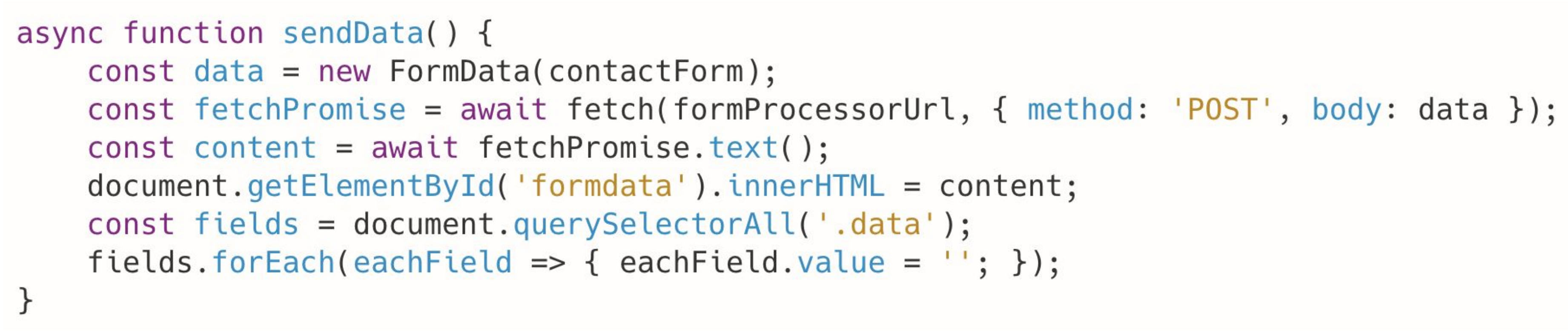
The variable, contactForm, was defined at the top of the script to hold our form. We know the data coming from the form is good, because it passed validation.

Getting the Data

```
async function sendData() {
    const data = new FormData(contactForm);
    const fetchPromise = await fetch(formProcessorUrl, { method: 'POST', body: data });
    const content = await fetchPromise.text();
    document.getElementById('formdata').innerHTML = content;
}
```

The next two lines take the text from the processor.php file, which in this case is just our data again, and stick it into the formdata div.

Getting the Data

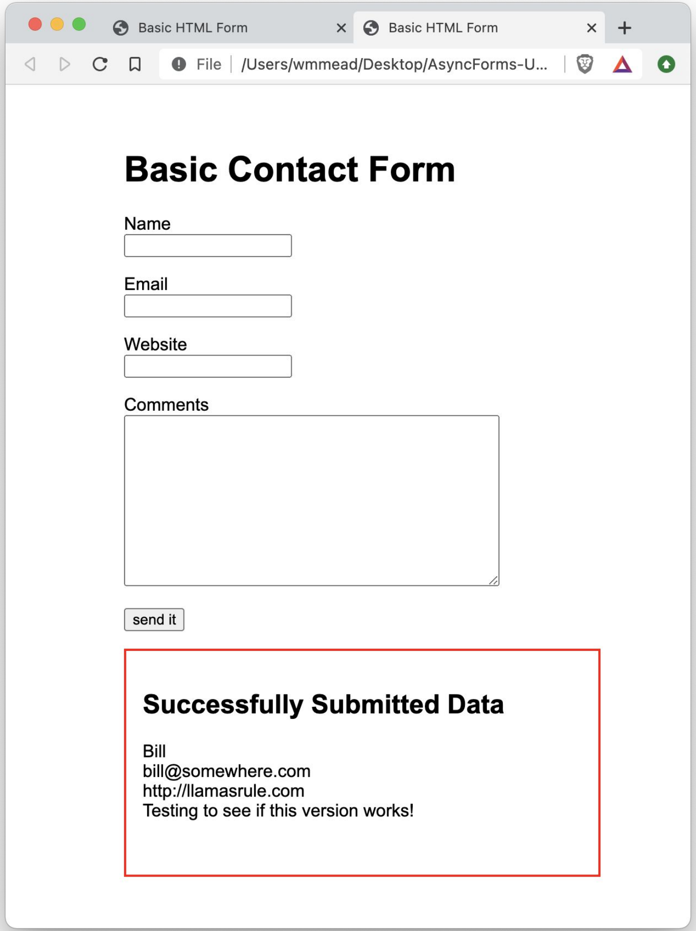


```
async function sendData() {
    const data = new FormData(contactForm);
    const fetchPromise = await fetch(formProcessorUrl, { method: 'POST', body: data });
    const content = await fetchPromise.text();
    document.getElementById('formdata').innerHTML = content;
    const fields = document.querySelectorAll('.data');
    fields.forEach(eachField => { eachField.value = ''; });
}
```

The last two lines simply get the form fields and clear them out. This should work. Test it out!

Testing Async Await

This version should work…



Basic Contact Form

Name

Email

Website

Comments

send it

Successfully Submitted Data

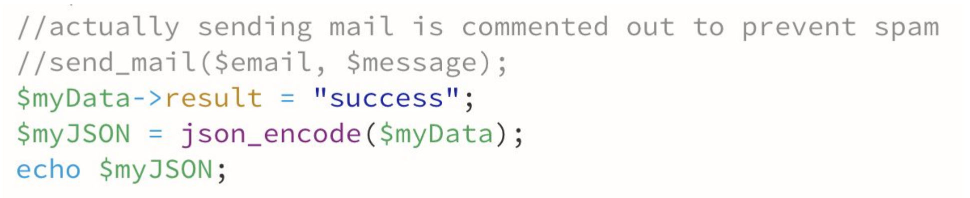
Bill
bill@somewhere.com
http://llamasrule.com
Testing to see if this version works!

A Useful Example

A typical use for this type of functionality, is for a contact form. The PHP file included in this example is something you could use to send email.

On the PHP file, the function that actually sends email is commented out. However, it will return a JSON string that looks like this:

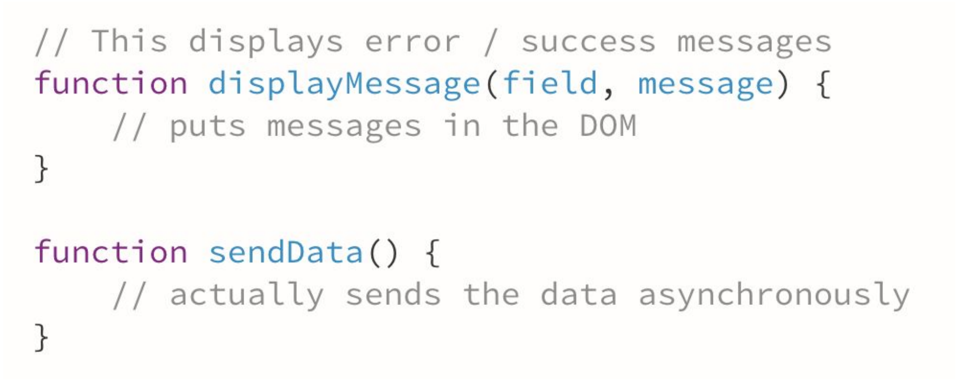
{‘result’ : ‘success’}



```php
//actually sending mail is commented out to prevent spam
//send_mail($email, $message);
$myData->result = "success";
$myJSON = json_encode($myData);
echo $myJSON;
```

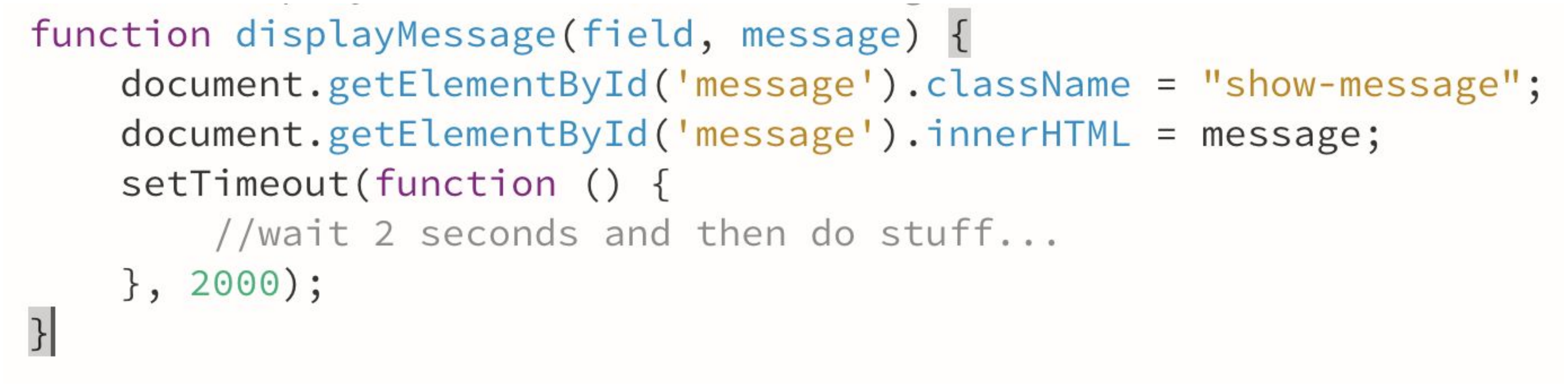
Working with Form Data

The function that does the form validation is already provided. But this version is doing some nicer interface effects if there are errors or if it is successful.



```
// This displays error / success messages
function displayMessage(field, message) {
    // puts messages in the DOM
}

function sendData() {
    // actually sends the data asynchronously
}
```

Working with Error Messages

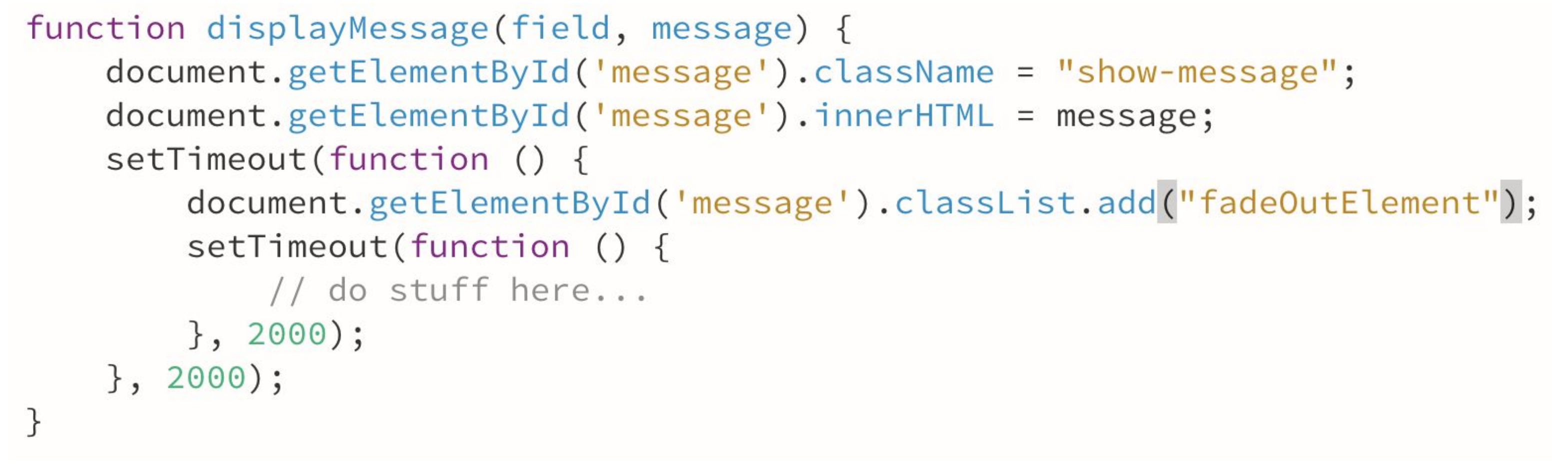


```
function displayMessage(field, message) {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = message;
    setTimeout(function () {
        //wait 2 seconds and then do stuff...
    }, 2000);
}
```

If someone puts bad data into a field, or leaves it blank, this function runs. The validateForm function above passes in the offending field and a message from the message array at the top of the page.

Set the message div to class “show-message”, and put the correct message in that div. Then wait two seconds before we continue.

Working with Error Messages



```
function displayMessage(field, message) {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = message;
    setTimeout(function () {
        document.getElementById('message').classList.add("fadeOutElement");
        setTimeout(function () {
            // do stuff here...
        }, 2000);
    }, 2000);
}
```

After two seconds, add the class “fadeOutElement” to the message div. This will fade the message out over two seconds.

After that, hide the message and put the cursor in the field that was not filled in properly.

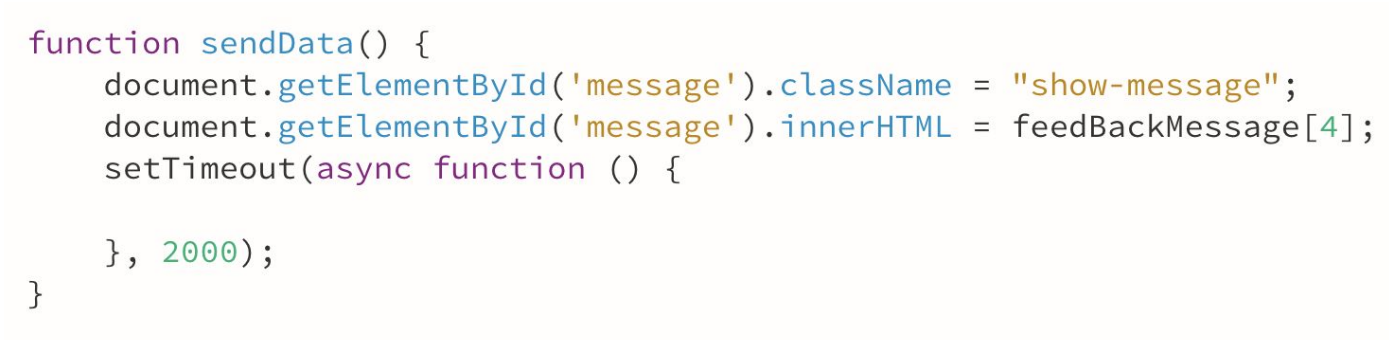
Errors Handled

```
// This displays error / success messages
function displayMessage(field, message) {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = message;
    setTimeout(function () {
        document.getElementById('message').classList.add("fadeOutElement");
        setTimeout(function () {
            document.getElementById('message').className = "hide-message";
            document.getElementById(field.id).focus();
        }, 2000);
    }, 2000);
}
```

That will handle errors. Test it and see if it works.

You will adjust this function a little to handle the success message as well.

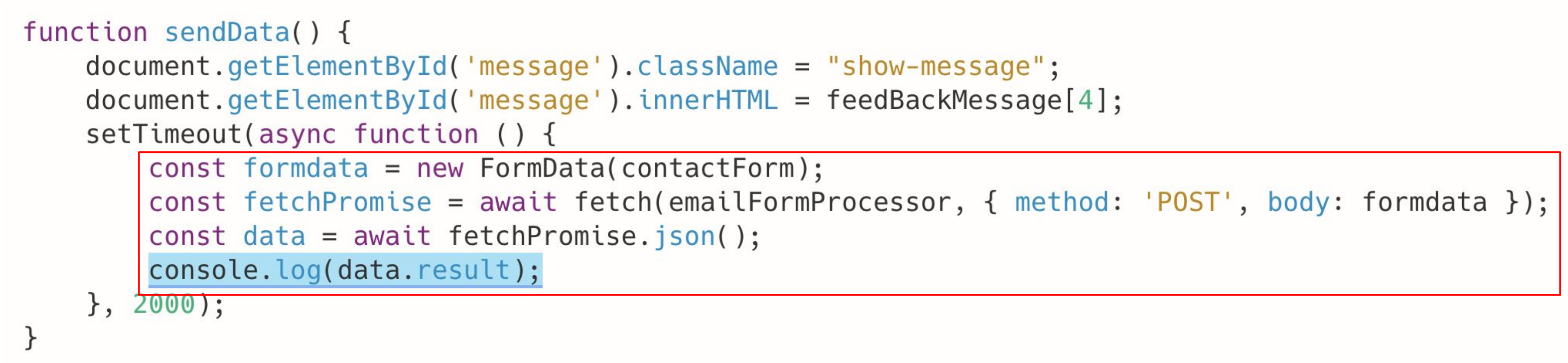
The sendData() Function



```
function sendData() {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = feedBackMessage[4];
    setTimeout(async function () {

    }, 2000);
}
```

This function is only a little different from the last async/await form example.

The sendData() Function

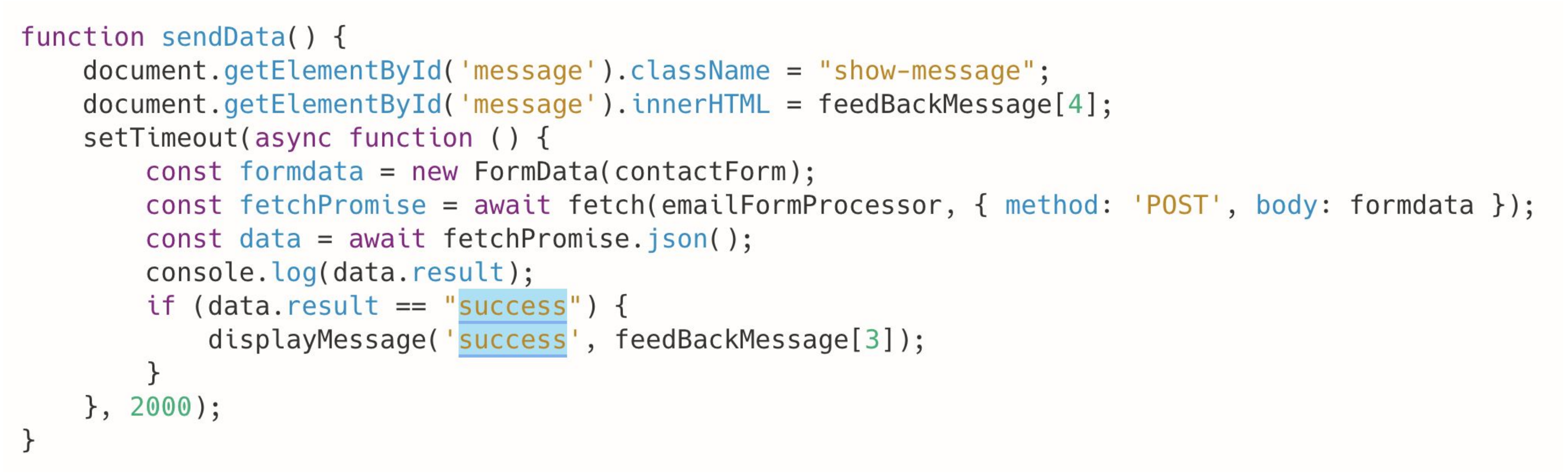


```
function sendData() {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = feedBackMessage[4];
    setTimeout(async function () {
        const formdata = new FormData(contactForm);
        const fetchPromise = await fetch(emailFormProcessor, { method: 'POST', body: formdata });
        const data = await fetchPromise.json();
        console.log(data.result);
    }, 2000);
}
```

Now do the same tasks you did in the previous async/await example. Get the form data, send it to the PHP file. The only difference is that this time, that file is returning a JSON string.

Optionally, you can console.log out that string and you should get “success” in the console log.

The sendData() Function



```
function sendData() {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = feedBackMessage[4];
    setTimeout(async function () {
        const formdata = new FormData(contactForm);
        const fetchPromise = await fetch(emailFormProcessor, { method: 'POST', body: formdata });
        const data = await fetchPromise.json();
        console.log(data.result);
        if (data.result == "success") {
            displayMessage('success', feedBackMessage[3]);
        }
    }, 2000);
}
```

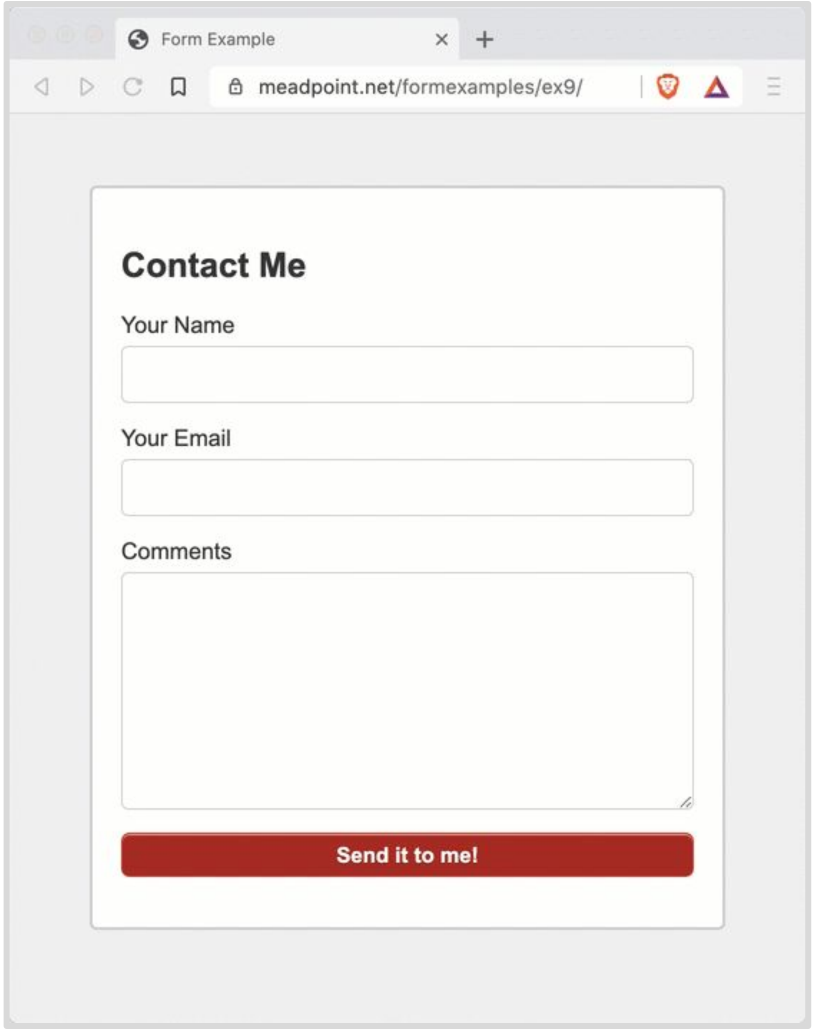
If data.result is success, then display the success message.

Updated displayMessage() Function

```
function displayMessage(field, message) {
    document.getElementById('message').className = "show-message";
    document.getElementById('message').innerHTML = message;
    setTimeout(function () {
        document.getElementById('message').classList.add("fadeOutElement");
        setTimeout(function () {
            if (field != 'success') {
                document.getElementById('message').className = "hide-message";
                document.getElementById(field.id).focus();
            }
            else {
                document.getElementById('message').setAttribute("class", "hide-message");
                document.getElementById('name').value = '';
                document.getElementById('email').value = '';
                document.getElementById('comment').value = '';
            }
        }, 2000);
    }, 2000);
}
```

Summary

This is a great element that can be included on any website and used for contact forms. You could expand on it further, or modify the styling however you like!



Contact Me

Your Name

Your Email

Comments