# Sending data to Google docs from your web page, using your own form and AJAX!

Added by Ernesto Rivera, last edited by Joaquín Arellano on Jun 10, 2013

**About**
Google docs easily integrates in a web page if embedding the code for calling their forms, in that way we only have to worry about placing it in the right place and Google makes the rest. Nevertheless, Google forms have a standard view which could break your site design, moreover, you'll probably want to manipulate the Form DOM through JQuery. As long as forms are embedded in an iFrame, you will not be able to do DOM Manipulation. The solution: Build your own form and send data to Google. But how do I do that? In this article you'll find the answer to that question.

Google Docs/Drive is a widely used service for data storage and documents sharing.

A distinctive functionality from this platform is that it allows to embed forms and / or save data from a web page to a spreadsheet (or other types of documents).

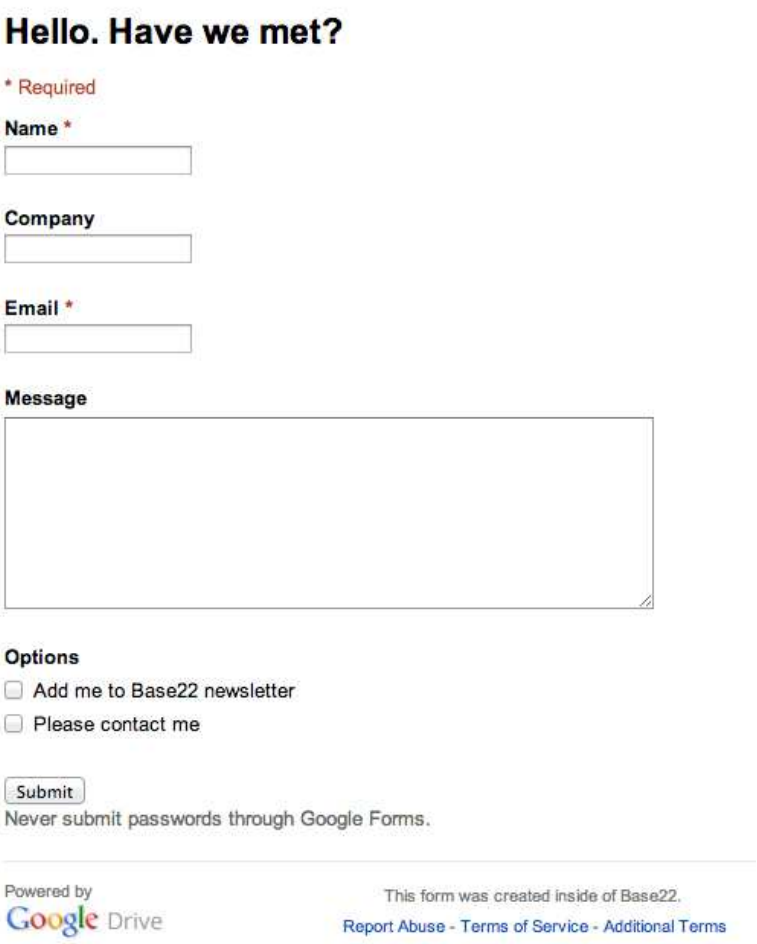
In this scenario, we will focus to spreadsheets.

There are two ways to upload information to Google.

1. Thorugh the embeded Google Forms.
2. Through an ajax request for posting data to Google.

### Embeded Google Forms

The first case scenario is the easier way, but it has as inconvenient that you will not be able to modify this form. Nevertheless its great advantage consists on its easy embedding thourgh a single line of Code. Google will make the rest. When you use the embeded form, you will be able to render in your web page a form that misteriously saves data to your spreadsheet. An example is shown next:




As you can easily see, it is a common form with Google legends and you can change really few things on it. Nevertheless, there is a way to create our own form, give it our own styles to make it match with our site and finally send our data to Google Docs thourgh a call to Ajax with the proper parameters, which will be explained in the next section.

In order to get the Google Form as you can see in the last picture, you need, with your spreadsheet created, create a form through the asistant in File > New > Form. Follow the assistant and you will get the html tag to embed in your code. If you already have your form

you just need to retrieve the embedding code from Form Menu > Embed form. Put the code in you html in the place you want the form to be rendered.

Nevertheless, this form can't be changed and it will be difficult to keep control over it. In order to have full and easy control over it, there is the second option:

### Recording data to Google Docs through Ajax.

This option was used in a scenario where we allow users to provide their data for contact and for newsletter suscriptions. This is specifically saved data to Google Spreadsheets.

*Scenarios*

We can find basically two scenarios: What we could call "a version" and "b version". It is quite simple to differentiate between both of them, simply by looking at the forms, we will find that the form for the a version contains on its footer the legend "Powered by Google Docs"  while the b version will contain the legend "Powered by Google Drive"  . Another option to differentiate them is through the destination (or update) address, being https://spreadsheets1.google.com/formResponse for the a version and https://docs.google.com/ for the b version.

### Requirements

In order to reach our spreadsheet, we need to know the associated form key from Google (also we can get the embed address where we can find it). We can obtain it by opening our spreadsheet, click on "Form" menu > "Embed form in a web page". When the form loads in a new browser tab, we can reach the form key. In scenario 1, a modal window will be opened, showing the iframe code to embed, you need to look for the parameter "formkey" and copy that key. In scenario 2, the form will be opened in a new tab, you will get the direct URL access to the form. The form key can be found as the number/letter sequence after "forms/d/" and before "/viewform".

### The Ajax request

The Ajax request is basically the same with a few differences, which will be explained later. It must contain the next settings:

*url*: the url address for the google service, it can include or not the form key according to the update scenario.

(case 1) "https://spreadsheets1.google.com/formResponse". Note: apparently, it also works without the number one in the "spreadsheets" part of the URL.

(case 2) "https://docs.google.com/a/SOMEDOMAIN.com/forms/d/XXXXXXX/formResponse", where XXXXXXX must be replaced by the form key.

Examples:

- url: "https://spreadsheets1.google.com/formResponse"
- url: "https://docs.google.com/a/YOURDOMAIN.com/forms/d/15QFO2VE44-9gAwcJeTPPWvxAX7v_1Ye9qmjdX2VzLBw/formResponse"

*data*: a JSON array containing the information we want to store in our spreadsheet.

For scenario 1, we need also to provide the form key along with the data. The JSON keys will identify the column where the information will be stored in our spreadsheet. The adequate syntax is shown below:

data: {formkey: "xxxx", "entry.0.single": var1, "entry.1.single": var2, "entry.2.single": "yyyy", "entry.3.single": var3}

Please note that information keys have the form "entry.x.single", where x corresponds to the column where the data will be stored.

For scenario 2, we only need to provide the data (be careful, do not provide the formkey in this scenario). The adequate syntax is:

data: {"entry.1" : var1, "entry.2" : var2, "entry.3": "A String"}

In this scenario 2, you will only provide the keys under the form "entry.x" where x corresponds to the column where the data will be stored.

For both scenarios, if you have required fields, they must be passed in the data, and must match the entry number with the column number in the spreadsheet.

type: "POST"

dataType: "xml",

statusCode: where we will provide a status code and an anonymous function that describes what should be done in case a determined code is retrieved as response. In scenario 1, we will get a statusCode = 200 when the request is properly handled, while in scenario 2, we will get a statusCode = 0.

### Coding examples

With these requirements gathered, we can have an HTML form such as:

**contactus.html**

```
<div id="form" class="contact-us-form">
    <div class="title">
        <strong>Have any questions?</strong>
    </div>
    <div class="subtitle">
        <strong>Drop us a line</strong>
    </div>
    <form id="callus" target="_self" onsubmit="" action="javascript:
postContactToGoogle()">
        <fieldset>
            <label for="name">What's your name? *</label>
            <input id="name" type="text" name="name">
        </fieldset>
        <fieldset>
            <label for="email">What's your email? *</label>
            <input id="email" type="text" name="email">
        </fieldset>
        <fieldset>
            <label for="feed">Questions or Feedback?*</label>
            <textarea id="feed" name="feed"></textarea>
        </fieldset>
        <div style="text-align: right; padding-bottom: 15px;">* Required</div>
        <div style="width: 100%; display: block; float: right;">
            <button id="send" type="submit">
                Contact Us
            </button>
        </div>
        <div style="width: 100%; display: block; float: right; padding-top: 15px;">
            <div class="requestSubmited" style="display:none; text-align: center;">Your
request has been sent!</div>
        </div>
    </form>
</div>
```

Please note that the form action calls a Javascript function which after validating the fields, will post the data to Google. Fields names can have the name we want just as the form. We need to include a Javascript tag containing the already mentioned functions, as we show next for the b version:

**contactus.html**

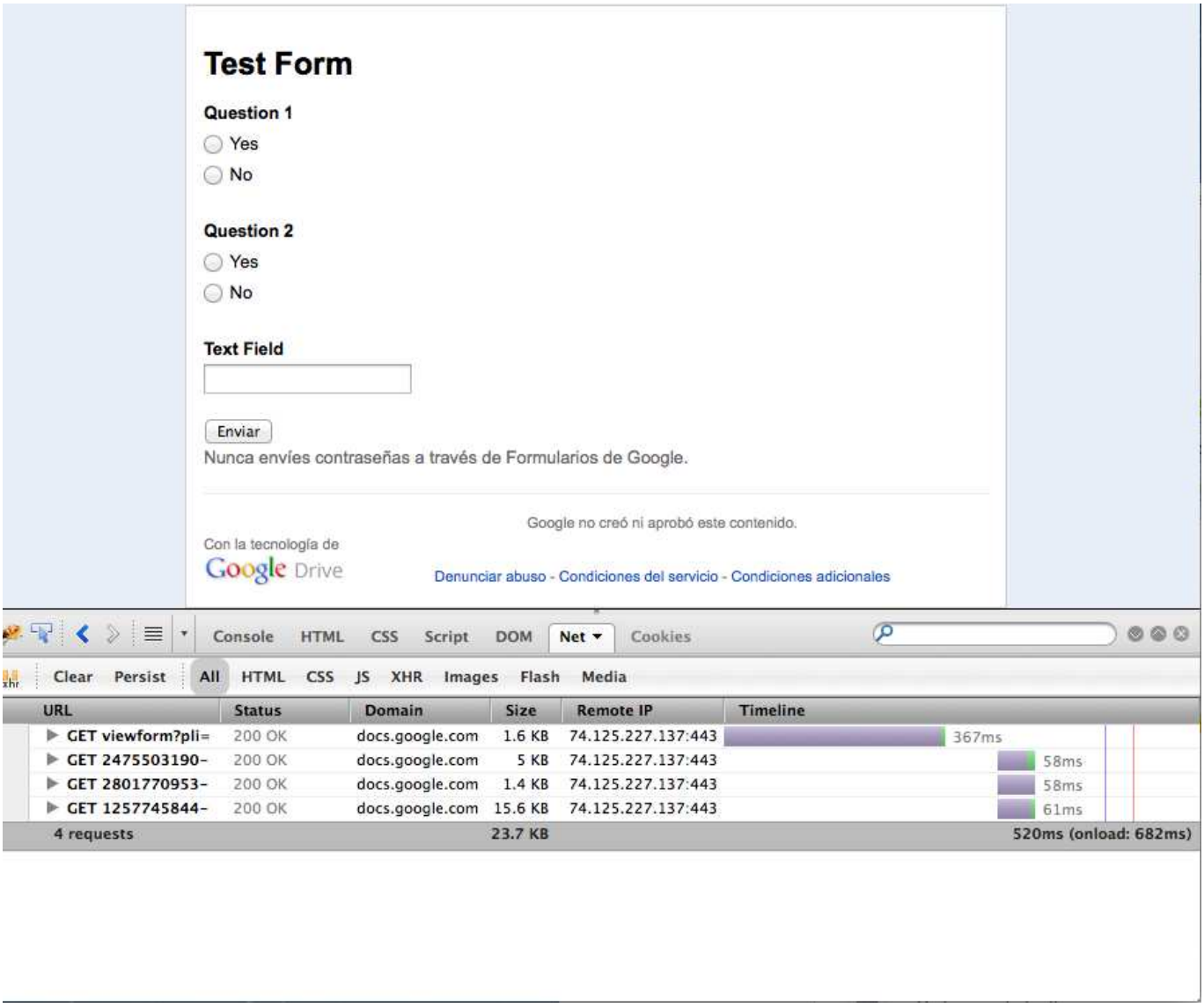
```
<script type="text/javascript">
    function validateEmail(email) {
        var re = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\
[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]
{2,}))$/;
        return re.test(email);
    }"
    function postContactToGoogle(){
        var name = $j('#name').val();
        var email = $j('#email').val();
        var feed = $j('#feed').val();
        if ((name !== "") && (email !== "") && ((feed !== "") &&
(validateEmail(email)))) {
            $j.ajax({
                url: "https://docs.google.com/yourFormURL/formResponse",
                data: {"entry.1" : name, "entry.3" : email, "entry.4": feed},
                type: "POST",
                dataType: "xml",
                statusCode: {
                    0: function (){

                        $j('#name').val("");
                        $j('#email').val("");
                        $j('#feed').val("");
                        //Success message
                    },
                    200: function (){
                        $j('#name').val("");
                        $j('#email').val("");
                        $j('#feed').val("");
                        //Success Message
                    }
                }
            });
        }
        else {
            //Error message
        }
    }
</script>
```

The validateEmail function will check if the parameter email is a valid string for email address. This function will be called next to the corresponding verification of fields not empty; if fields have values and email is a valid email address then we call our ajax function, containing the url (explained in requirements "yourFormURL" will be replaced by the text explained in that section), data, type, dataType and status code described. If ajax returns the status code number 200, it means we succeeded Google also can return an empty status, with a 0 for status code number, in this case we have also succeeded In these two cases we just need to clean up our form (if necessary) and show a success message. If the user left an empty field or the email is not a valid address, we show the corresponding error message. Remember that url and data will be built according to the scenario we face, described on section "Requirements".

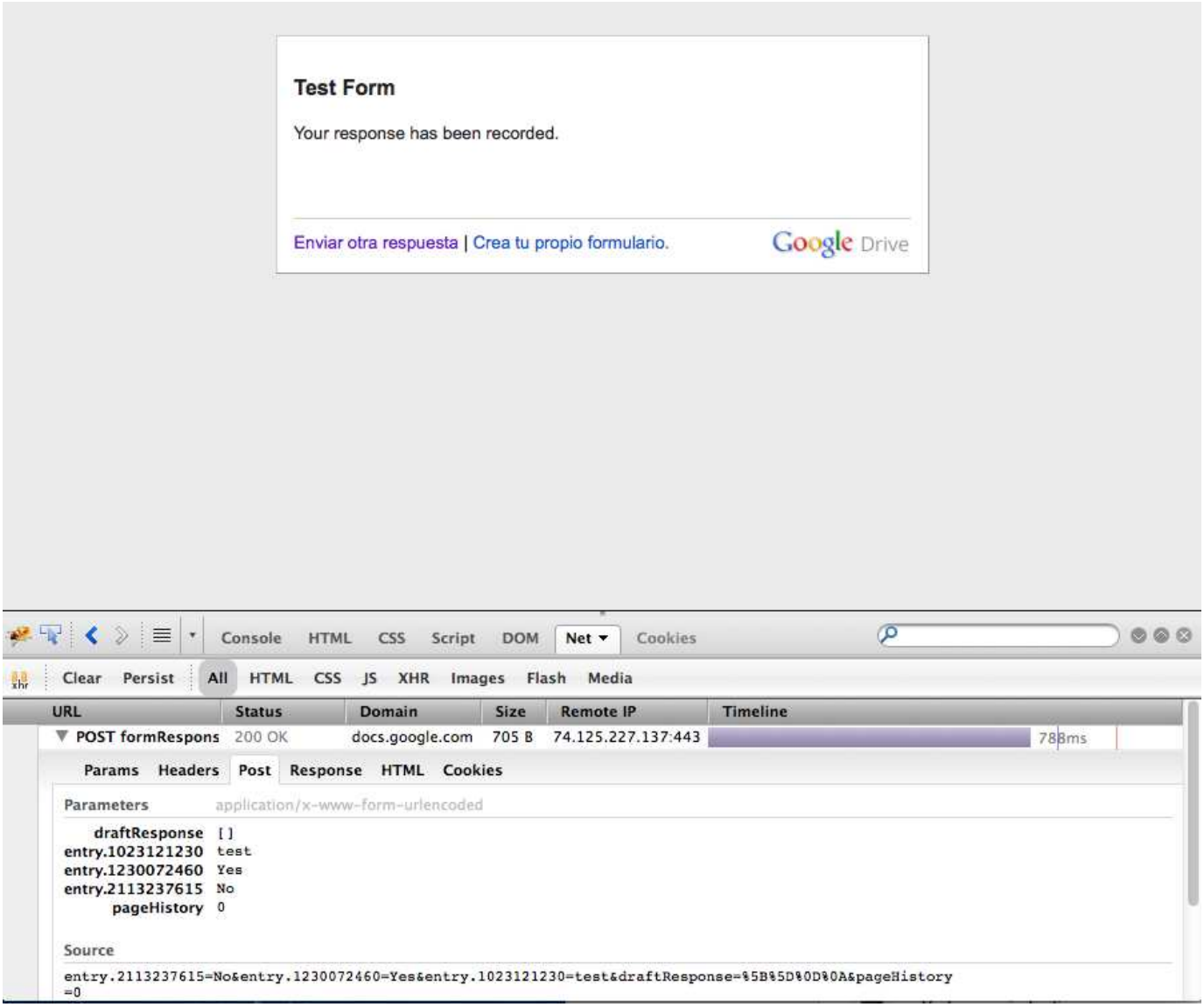
**UPDATE:** Thanks to Rahul, Daniel, Joe Kale for their valuable feedback, because of that we found some interesting issues that could arise when you experiment on sending data to Google docs through jQuery Ajax. Also thanks to Joaquín for the help to find the solution to Joe's scenario. If your customized form is reaching Google servers, but it is not recording values you can have one of the next possible problems:

1. It is possible that you need to check the entry identifiers, by using the Google form (via a code inspector, Net tab). Sometimes (we don't know the reason) Google uses specific entry ids, different to the consecutive one's we are using in our example, because of this reason, you need to use your Google Form to insert a record in your spreadsheet, and then using the Net Tab of your favorite code inspector, look for the entry names in the form's post call. You will notice if there is a different name when you find something like **"entry.1023121230"**. In this case, you need to place this name just as used by the Google Form, instead of entry1, entry2, etc. (See next images).

   Google Form before sending Data:




After data has been sent, note the entry names in Firebug's Net tab:




2. If you are on the second scenario from above, please verify the URL parameter for your Ajax call. In this scenario, when you copy the form key (see requirements), be careful to only copy that key, because when you form the url for your Ajax call, you need it ends with **"/formResponse"**, as described in our second block of code.
3. Finally, sometimes Google inserts the values in a different sequence to the one provided. When this happens, we just need to arrange the order in which we send data in the Ajax call. Refer to the code next for more reference.

**Replicating Joe Kale's scenario:**

**Demo code for non-sequential entry ids**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Google Forms</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js">
</script>
    </head>
    <body>
        <div>
            <div>
                <strong>Test Google Form</strong>
            </div>
            <form id="form" target="_self" onsubmit="" action="">
                <fieldset>
                    <label>Question 1</label>
                    <input id="qs1_op_1" type="radio" value="Yes" name="qs1">
                    <input id="qs1_op_2" type="radio" value="No" name="qs1">
```

```
                </fieldset>
                <fieldset>
                    <label>Question 2</label>
                    <input id="qs2_op_1" type="radio" value="Yes" name="qs2">
                    <input id="qs2_op_2" type="radio" value="No" name="qs2">
                </fieldset>
                <fieldset>
                    <label>Text</label>
                    <textarea id="feed" name="feed"></textarea>
                </fieldset>
                <div style="width: 100%; display: block; float: right;">
                    <button id="send" type="submit">
                        Send
                    </button>
                </div>
            </form>
        </div>
        <script type="text/javascript">
            function postToGoogle() {
                var field1 = $("input[type='radio'][name='qs1']:checked").val();
                var field2 = $("input[type='radio'][name='qs2']:checked").val();
                var field3 = $('#feed').val();

                $.ajax({
                    url: "https://docs.google.com/forms/d/FORM_KEY/formResponse",
                    data: {"entry.1023121230": field3, "entry.1230072460": field1,
"entry.2113237615": field2},
                    type: "POST",
                    dataType: "xml",
                    statusCode: {
                        0: function() {
                            //Success message
                        },
                        200: function() {
                            //Success Message
                        }
                    }
                });
            }

            $(document).ready(function(){
                $('#form').submit(function() {
                    postToGoogle();
                    return false;
                });
            });
        </script>
    </body>
</html>
```

In this example, we have two radio buttons and a text box, whose values we want to record to Google Docs. Note that entries names now are formatted as **entry.1023121230**. In our Google Spreadsheet and in our HTML form we have placed the optional values as first and second columns (HTML controls) and the text as third column (HTML control), while in our Ajax call we send first the text value and then the option values.

**UPDATE: Our teammate Ben Shoemate found another approach that solve the cross domain error, but just works in the case 2 (the current way Google forms works), we have tested that approach and works!!**

**Feel free to check it too: Tip: Adding a submission form to your blog**

3 people like this