Cool HTML & CSS Forms & How To Hook Them Up To Google Sheets

One thing that really helped me, not only to bring my website to life, but to collect data is… Using HTML forms, styling them with CSS, and using Javascript to help me capture the data from the form and send it to a Google Sheet.

Yes, it seems complicated and doing it for the first time was kind of frustrating, but… When you get it working, correctly, it's worth collecting the data from the forms on your website in a spreadsheet!

Why is that important? It's because autoresponder services can increase their price and since it is THEIR platform, they can BAN you, if they choose to. What would happen? You'd lose your email list. So why not collect it yourself?

As an authentic affiliate marketer (who won't promote CRAP) you'll likely not get banned, but some autoresponder services don't even let you use the autoresponder for affiliate marketing at all.

So you can see why **TRULY OWNING YOUR EMAIL LIST** really, really, really matters!

Yes, the pro autoresponder services (that people pay a subscription fee to use) do have affiliate programs, but they also have limitations.

After the form "stuff" (getting it working), we can dive into making your own autoresponder.

It's worth it to say that I use Wordpress for my websites. Wordpress has a plugin (page builder) called Elementor. I use Elementor to add my HTML, CSS, and Javascript with the HTML element.

If you want an "in depth" explanation of how I set up a WordPress Website for affiliate marketing, I wrote a PDF called:

👉 **"How To Set Up A Wordpress Website For Affiliate Marketing & Other Cool Stuff"**

You can find it on my download page (where you probably got this one).

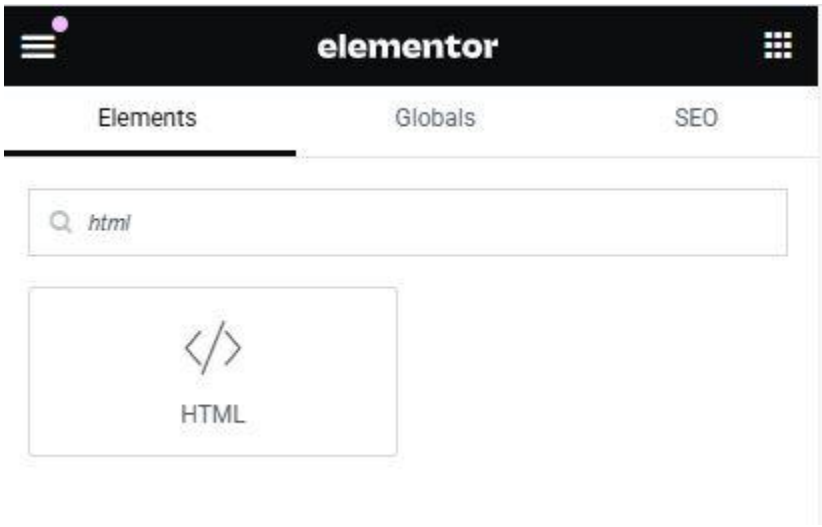
It's over 90 pages with screenshots. Everything from how I get domains, how I get web hosting, how I connect the domains to the web hosting, how to install Wordpress on your own hosting, and much more!

Let’s do this!

👉 First, you’ll need a new (or existing) page to put your html form onto.

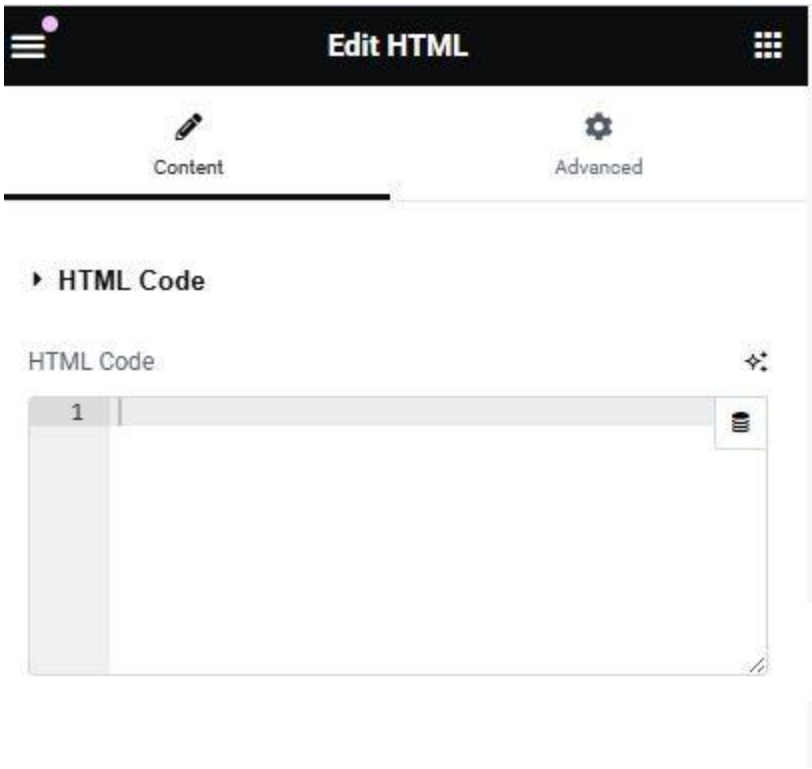
Got one? Cool. Let’s continue…

👉 To find the HTML element in Elementor, just search html in the search bar and it’ll pop up.



👉 Now, we’ll just drag the HTML element onto the page (if you’re using WordPress like I do).

Inside the HTML Element’s text editor, we’ll be adding some HTML (What else? Right?)



Don’t worry if you’ve never used HTML or CSS before. I’ll give you the code and you can copy and paste it into that box (in the image above), sounds good? Cool. Let’s do it.

Let’s talk a bit about HTML. If you’re new to it, it’s a computer language called Hyper-Text Markup Language.

HTML creates the “structure” or the “bones” of a page. Everything you see, online, is created with HTML, styled with CSS (Cascading Style Sheets) and the functionality is mostly scripts written in JavaScript.

That’s as simple of an explanation as I’ve got. If you want to learn more about coding, I learned everything I know about coding on Youtube. It’s free and you’d be amazed at what you can find (and learn - if you actually work on it).

👉 Here’s the code I promised you:

👉 (See next page).

👉 HTML:

```
<form id="myForm" action="YOUR_APP_URL" method="POST">
  <input type="text" id="name" name="name" placeholder="Your Name" required />
  <input type="email" id="email" name="email" placeholder="Your Email" required />
  <button type="submit">Join Now</button>
  <div id="form-message" style="margin-top: 10px;"></div>
</form>
```

Want to break it down with me? Okay!

We have the form tags… <form></form>

Inside the opening form tag, we have the ID of the form, an ACTION, and a METHOD.
The ID of the form communicates about the form, itself.
The ACTION tells the form what to do.
So does the METHOD.

The action directs the form to the app script that sends the data from the form to the Google Sheet you're connecting to the form.

The method tells the form how to do that, to POST it to the spreadsheet.

👉 Between the form tags, we have two inputs and a button.

The inputs are for a user to put their name and their email address into the form. The inputs have placeholders so that the user knows what to enter into the inputs.

👉 The placeholders "disappear" when the user types into the form.
I prefer to use placeholders in my forms instead of labels because it makes for a "cleaner" form, (in my opinion).

Other than placeholders, the inputs have a type for each. The name input's type is text, the email input's type is email.

The inputs also have names.

The button's type is submit.

👉 In between the button tags is where you type the text you want to appear on the button.

Now comes the fun stuff! Styling the form!

Don’t worry if you’ve never used CSS before, we can add it right into the HTML element (with the HTML code I just gave you).

👉 CSS:

```
<style>

  #form-message {
  font-weight: bold;
  transition: all 0.3s ease;
  margin: 10px auto;
  padding: 12px 16px;

}

  form {
    max-width: 400px;
    margin: 20px auto;
    padding: 20px;
    background: #fff;
    border-radius: 12px;
    box-shadow:
  0 0 0px rgba(0,0,0,0.1),       /* base shadow for depth */
  0 0 900px rgba(52, 152, 217, 0.4), /* outer blue glow */
  0 0 5px rgba(255, 255, 255, 0.2); /* subtle inner light pop */
  }

  input[type="text"],
  input[type="email"] {
    width: 100%;
    padding: 12px 16px;
    margin-bottom: 16px;
    border: 2px solid #ccc;
    border-radius: 8px;
    font-size: 16px;
    transition: all 0.3s ease;
  }


  button {
    width: 100%;
```

```
    padding: 12px 16px;
    background: crimson !important;
    color: #fff !important;
    font-size: 18px;
    font-weight: 700 !important;
    border: 2px solid #000 !important;
    border-radius: 8px;
    cursor: pointer;
    transition: background 0.2s ease;
    box-shadow: 0 4px 16px rgba(0,0,0,0.1);

  }

  button:hover {
    background: #ffd500 !important;

  }
</style>
```

Ready to break it down?

👉 First, we're using the style tags because we're adding the CSS code into the HTML text editor.
👉 Everything's between the style tags: <style></style>.

In the code I gave you, we're styling the form, itself, the inputs, and the button.

Form:

The form has a max-width: 400px;
The form has a margin: 20px auto; (auto centers the form).
The form has a background of white: #fff;

You can get the guide of hex colors:

👉 **"How I Find Hex Colors & Make Money Promoting Coolors.co"**
(Also on my download page).

We also gave the form some padding so that the inputs aren't right up against the edges of the form.

The form has a border-radius to decrease it's sharp corners. (You don't have to have it if you don't want it on there. You can change anything in the code that you want to).

Other than the border-radius, the form has a box-shadow to make the form "pop off the page." It adds some "depth" and attracts attention to the form.

Inputs:

The inputs got a width, too. We set their width at 100%; because we want it to be across the form.

We gave the inputs a margin, too, because we want them to have a space away from the edge of the form and we want the inputs to have a space that separates them.

We gave the inputs a border: 2px solid #ccc;

There are a few types of borders you can use. In this example we went with a solid border.
The inputs also have border-radius to slightly round their corners.

We gave the inputs a font-size of 16px. You can give the inputs a font-size of whatever you'd like, but we went with 16px because that's not too big (and not too small).

The last thing we added to the inputs is a transition because we added a placeholder. The transition is for the placeholder to transition (to ease away to add the text and email into the inputs). It doesn't need much of a transition, a second would be too long so we went for 0.3s (seconds) and added ease to ease it out.

Great! Now let's talk about what we did with the button (buttons are the best if they function right and they are fun to style).

Button:

First, we gave it a width of 100%.
The button has a padding on the left and right, then the top and bottom.
The button has a background color of bright yellow: #ffdd57;
The button has a text a color of black: #000.
The button has a font-size of 18px. Slightly bigger than the font-size for the inputs.
We gave the button a border of none; (but you can give it a border if you want to).

Then, we gave the button a border-radius to slightly round the corners of the button.
We gave the button a cursor of pointer so that the cursor changes to a pointer for the button

from an arrow.

We added a transition on the button, too, because we added a hover effect for the button. When the user hovers over the button, the background of the button changes so we added a transition for that.

Put the CSS code above the form's HTML code, okay?

You can style your form however you want to. I ask chatgpt for help styling my forms and other things related to my sites. I'm sure if you ask it (nicely), it'll help you out (just like it helps me).

(Chatgpt helped me with the code I'm using in this PDF).

The next part's the most complicated because we're going to use a script and we're going to connect it to a Google Sheet.

Let's dive into that! Ready? Let's go!

👉 First, we have to make a new Google Sheet. Go to Google Sheets.

Create a new sheet and call it whatever you want to.

✅ Set Headers in the First Row

1. **Open your Google Sheet**.

👉 In **cell A1**, type:

```
Timestamp
```

👉 In **cell B1**, type:

```
Name
```

👉 In **cell C1**, type:

```
Email
```

Press **Enter** after each one so they're saved.

🔍 These are your column headers — your script will look for these when it writes data. They **must** be in the very **first row** (row 1), and spelled exactly like: `Timestamp Name` and `Email`.

🎯 Quick Tip:

Don't put **"A"** or **"B"** as the header text — those are just column labels that Google Sheets uses automatically. Instead, 👉 your **first row** (Row 1) should have the actual labels:
Timestamp | **Name** | **Email**

👉 Now let's add the script to the form to connect them.

In the HTML editor that we wrote (or added) our code (the code I gave you), we'll add some more code. It'll be a script and goes between script tags <script></script>.

👉 Here's the script:

```
<script>
  document.addEventListener("DOMContentLoaded", function () {
    const form = document.getElementById("myForm");
    const messageDiv = document.getElementById("form-message");

    form.addEventListener("submit", function (e) {
      e.preventDefault(); // Stops the form from reloading the page

      const name = document.getElementById("name").value.trim();
      const email = document.getElementById("email").value.trim();

      if (!name || !email) {
        messageDiv.textContent = "Please fill in both fields.";
        messageDiv.style.color = "red";
        return;
      }

      fetch("YOUR_WEB_APP_URL", {
        method: "POST",
```

```
      headers: {
        "Content-Type": "application/x-www-form-urlencoded",
      },
      body: `name=${encodeURIComponent(name)}&email=${encodeURIComponent(email)}`,
    })
      .then(() => {
        messageDiv.textContent = "🎉 You're in! Check your inbox for the goodies.";
        messageDiv.style.color = "green";
        form.reset();
      })
      .catch((error) => {
        console.error("Form submission error:", error);
        messageDiv.textContent = "❌ Something went wrong. Try again later.";
        messageDiv.style.color = "red";
      });
  });
});
</script>
```

Replace YOUR_WEB_APP_URL with the actual link from your Apps Script deployment.

What do I mean about the actual link from your Apps Script deployment?

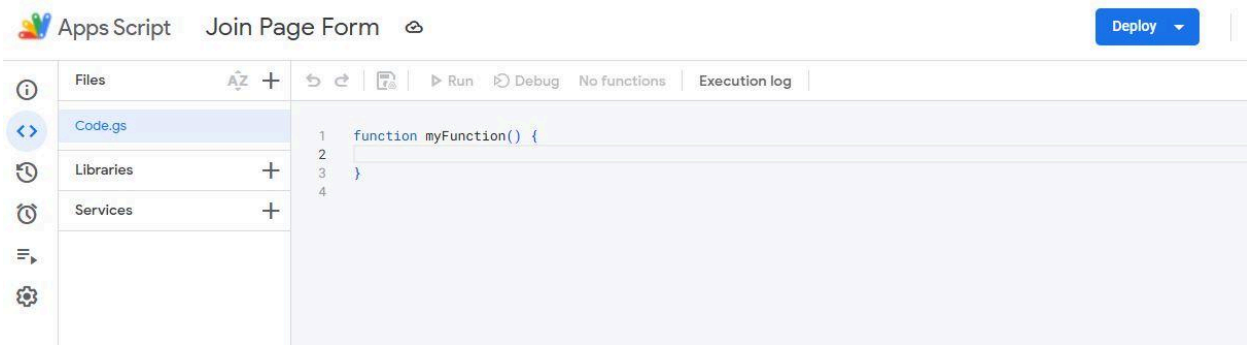
When you connect your form to a Google Sheet, you'll use a small Google Apps Script to handle the form submission. Once we've written that script, you'll need to **deploy** it as a web app — and that deployment gives you a special **URL** (link). That URL is what your form sends the data to.

Put the JavaScript I gave you **underneath the form tags in your HTML editor** where we put the code for the form in Elementor.

👉 Getting the Link from Your Apps Script

1. **Go to your Google Sheet.**

    👉 Click **Extensions** → **Apps Script**.



2. 👉 **Delete any starter code** and paste in the script I give you:

```
function doPost(e) {
  // Log the full event object to see what we are getting
  Logger.log("Received event: " + JSON.stringify(e));

  // Check if event object is null or undefined
  if (!e) {
    Logger.log("No event object received.");
    return ContentService.createTextOutput("No event object received").setMimeType(ContentService.MimeType.TEXT);
  }

  // Check if 'e.parameter' exists and log if missing
  if (!e.parameter) {
    Logger.log("No parameters received.");
    return ContentService.createTextOutput("No parameters received").setMimeType(ContentService.MimeType.TEXT);
  }

  // Log the individual parameters to verify they're coming through
  var name = e.parameter.name || "No name provided";
  var email = e.parameter.email || "No email provided";
  var timestamp = new Date();
```

```
22.
23.   Logger.log("Name: " + name);
24.   Logger.log("Email: " + email);
25.   Logger.log("Timestamp: " + timestamp);
26.
27.   // Add data to Google Sheets
28.   try {
29.     var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
30.     sheet.appendRow([timestamp, name, email]);
31.     Logger.log("Data added to sheet.");
32.   } catch (err) {
33.     Logger.log("Error while adding data to sheet: " + err);
34.     return ContentService.createTextOutput("Error adding data to
    sheet").setMimeType(ContentService.MimeType.TEXT);
35.   }
36.
37.   // Return a success response
38.   return ContentService.createTextOutput(
39.     JSON.stringify({ result: "success" })
40.   ).setMimeType(ContentService.MimeType.JSON);
41. }
42.
43.
```

📌 Important Notes:

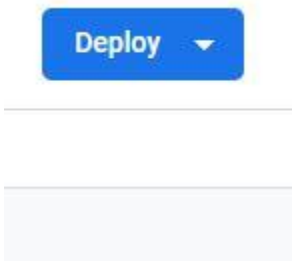
- Make sure your sheet's name is the same as in the script.
- This script grabs the values from the form (`name` and `email`) and adds a timestamp automatically.

👉 Okay! Let's deploy the script and get the URL to connect your form!

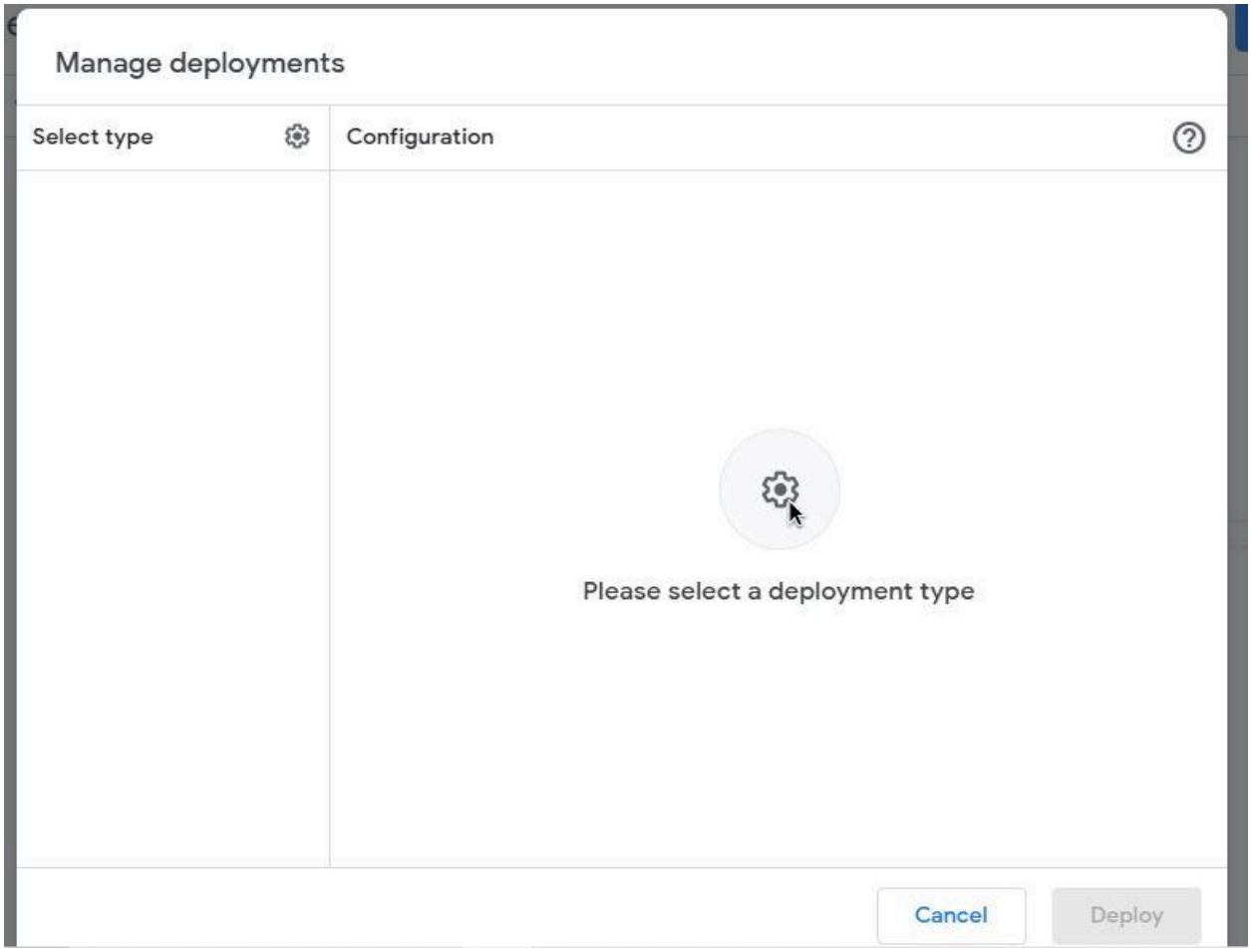
✅ Ready for Deployment?

Once you pasted in the script I just gave you:

👉 Click Deploy → Manage Deployments → + New Deployment



👉 Click the gear icon ⚙️ next to "Select type" → Choose Web app



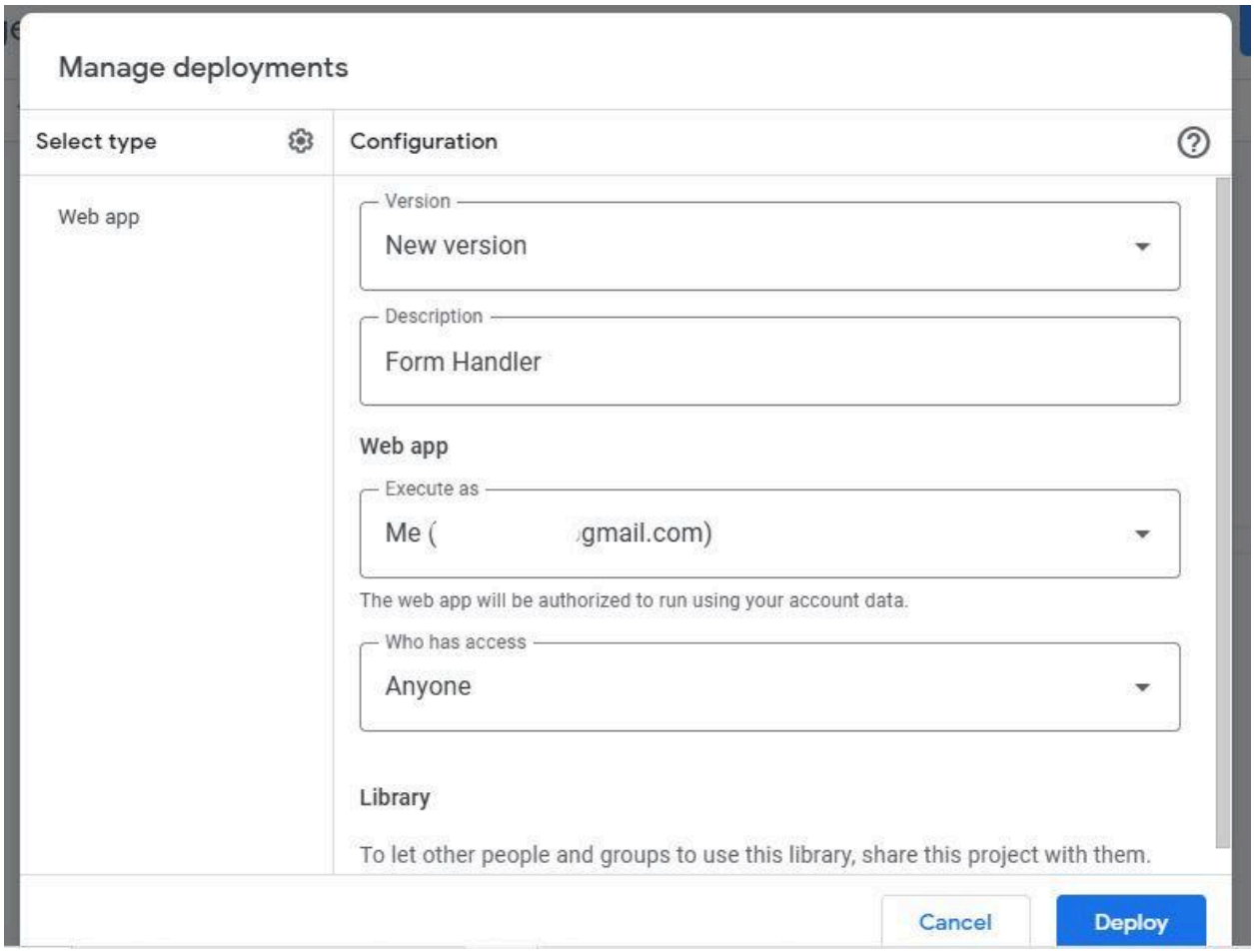
👉 Fill out:

Description: Form Handler

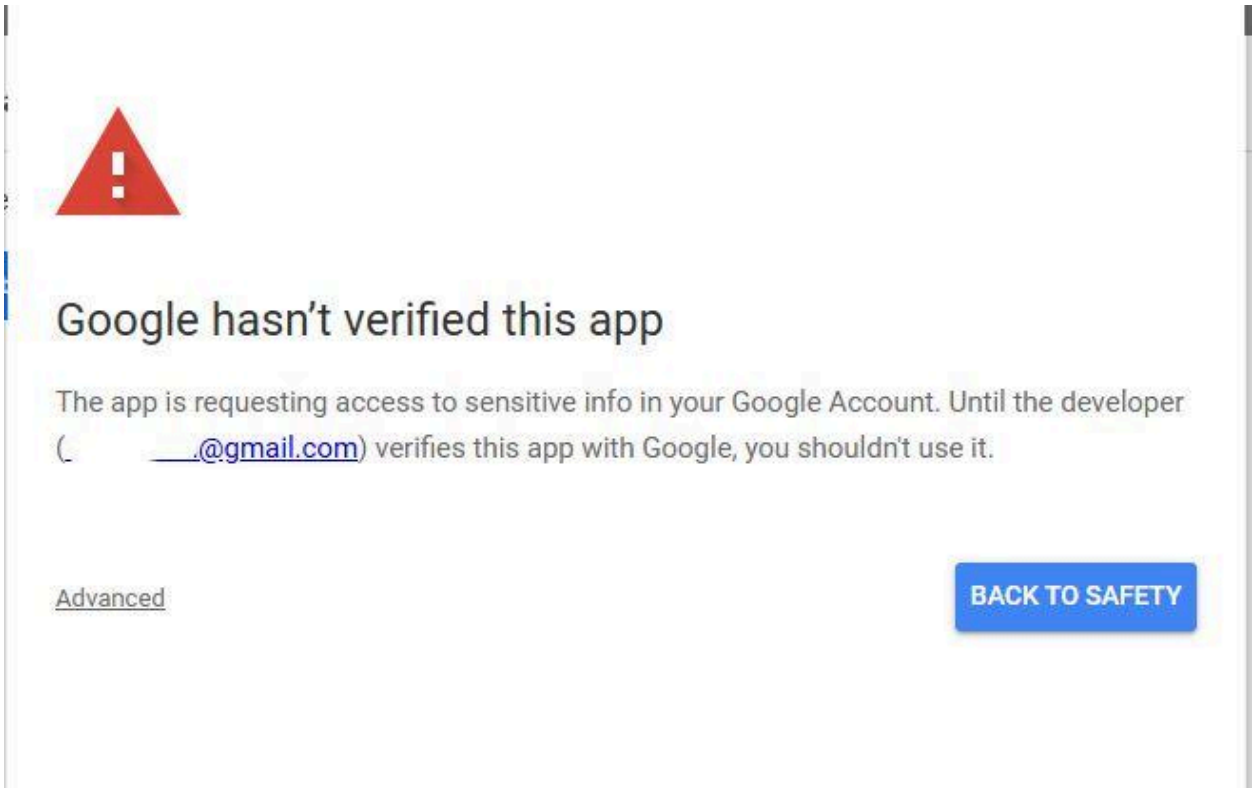
Execute as: Me

Who has access: Anyone

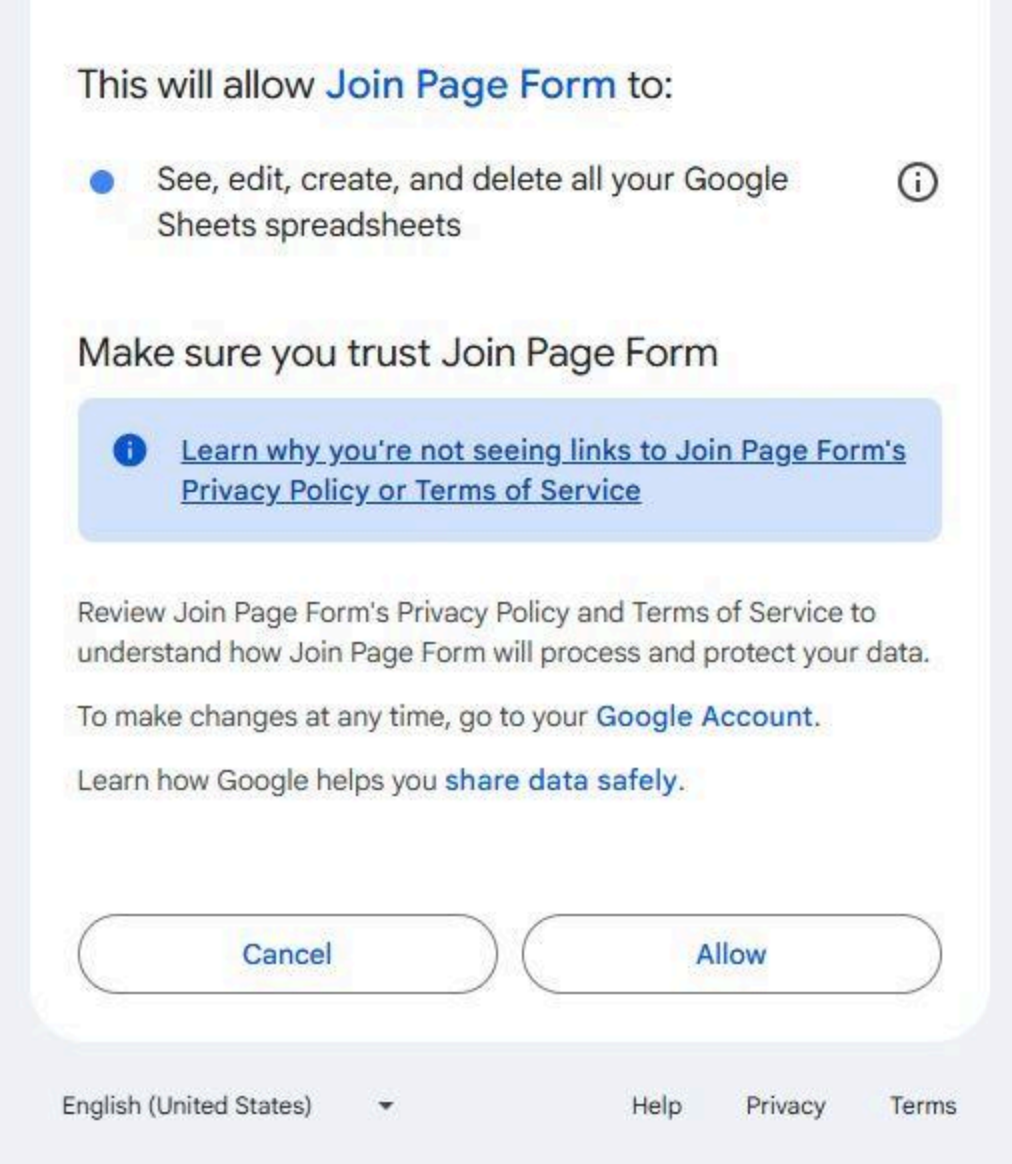
👉 Click Deploy. Authorize it when prompted.



If you get an error that Google hasn’t verified this app, don’t worry about that.
There’s a way to fix it!

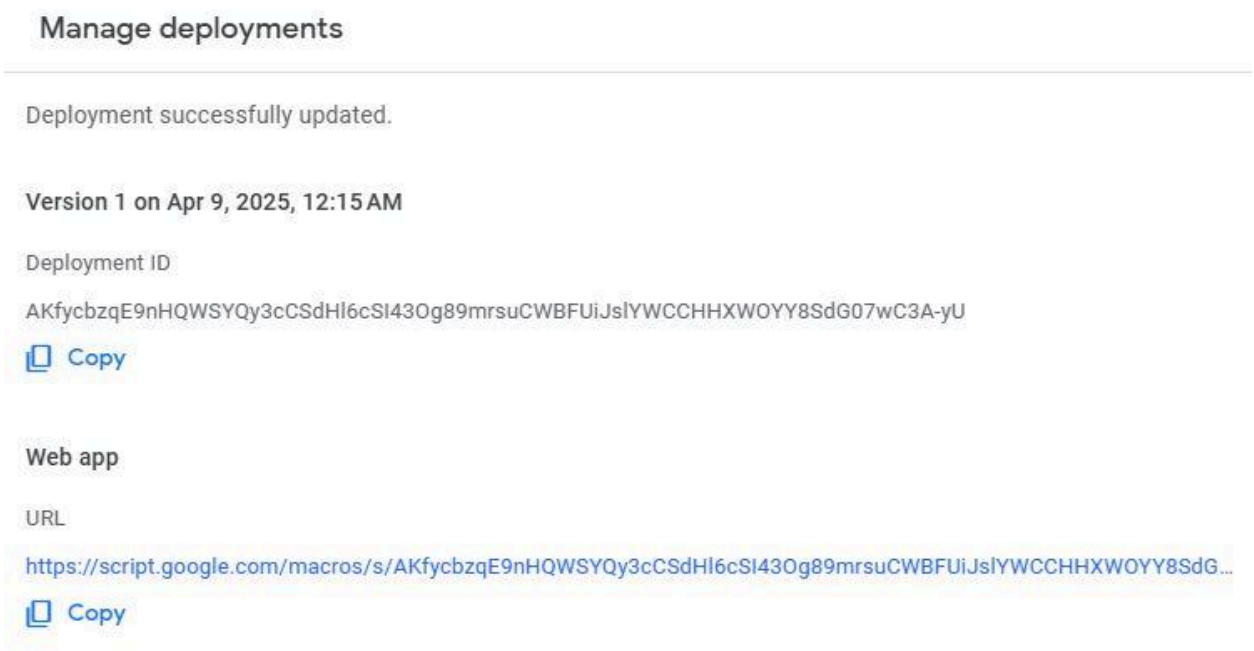


Just click the link that says Advanced on the left.



👉 Click on Allow.

Then you should get to this page **that has the app URL that you need to copy and paste into your HTML & JavaScript** in your HTML text editor (in Elementor).



The bottom link is the URL for the web app that joins the html form (we built in this PDF into Elementor's HTML text editor) to your Google Sheet.

The URL that you paste into your JavaScript code in Elementor's HTML text editor. Where it says:

"YOUR_WEB_APP_URL"

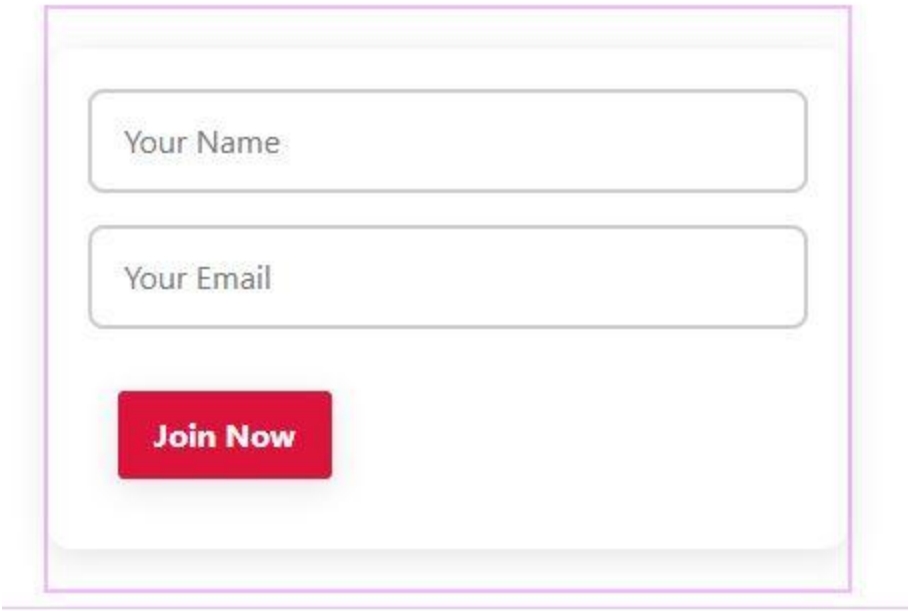
Copy the web app URL – this is what you'll need to plug into your HTML/JS!

Between the quotation marks is where you paste the URL for the web app.
The quotation marks are very important. Without them the code won't work.
If the lines of code in the CSS don't end with a ; the code won't work.

The code should work, copied from this PDF. I tested it myself, to make sure.
I had to test it up to 10 times while just building it. It took me 2 days to figure it out. I'm sharing it with you so you can get it up, connected, faster than I did.

(These are guides I wish I had - and didn't).

Now, you should have a form that looks something like this:



Cool! Test it out to make sure it posts the data to your Google Sheet!

👉 **Look for the next PDF called: “How To Set Up Your Own Autoresponder.”**