



# INNOVATIVE SOLUTION FOR IMPLEMENTING WEB FORM SUBMISSION IN STATIC WEBSITES

Jais Binoy

Research Associate, International Centre for Technological Innovations, Kerala.

## ABSTRACT

*In an age where web presence is paramount, static websites remain an elegant solution for many, offering simplicity and speed. However, static websites face inherent limitations when it comes to web form handling. This study delves into the challenges faced by static website hosting providers and presents an innovative approach to overcome these hurdles. By leveraging the power of JavaScript, Google Sheets, and Webhook extensions, this paper proposes a cost-effective solution that redefines web form handling for static websites. This approach not only addresses the shortcomings of server-side processing and data submission restrictions but also enhances data management. Furthermore, it explores the integration of Form Mule for efficient email notifications. This study presents a substantial and relevant solution for static website owners looking to optimize user interactions, implement web forms and data handling.*

**KEYWORDS:** *Static websites, Web form handling, JavaScript, Webhook, Google Sheets, Form submission*

## I. INTRODUCTION

Static websites have a substantial presence on the internet, catering to businesses, personal projects, and various organizations. Overcoming the limitations in web form handling is of paramount importance for these websites

### Research Question

How can the limitations and challenges associated with web form handling in static websites be effectively mitigated while maintaining cost-effectiveness?

### Research Objectives

- To identify and analyze the limitations of web form handling in static website hosting
- To propose an innovative solution for cost-effective web form handling in static websites using JavaScript, Google Sheets, and Webhook extensions.

This paper primarily focuses on addressing the limitations of web form handling in the context of static website hosting. It provides an analysis of the challenges faced and offers a comprehensive solution using readily available tools and technologies, making it accessible to a wide range of users. The study, however, does not delve into advanced server-side scripting or custom-built database solutions.

## II. METHODOLOGY

The research methodology for this paper presents a practical approach in establishing a dynamic form on a static website. The following methods were utilized:

- **Google Sheet Setup:** Created a Google Spreadsheet to serve as the data repository for form submissions.
- **Webhook Extension Installation & URL Generation:** Installed webhook extension, configured it to monitor the Google Spreadsheet, and generated a webhook URL.
- **HTML Form Integration with Data-Webhook-URL:** Developed the HTML form and embed it within the static website. Included the generated webhook URL.
- **JavaScript Data Extraction and Google Sheets Integration:** Utilized JavaScript to extract and process form data entered by users and seamlessly transmit it to the designated Google Spreadsheet, thereby facilitating efficient data collection.

## III. STATIC WEBSITE HOSTING SOLUTIONS

A static website consists of unchanging web pages created with HTML, CSS, and occasional JavaScript for simple features. Its content remains fixed until manually updated, lacking dynamic changes. As there are no back-end methods, client-server demands, nor database queries required in delivering a static website, it displays fast performance with its servers always ready with HTML outputs. Hosting such web pages is Static Web Hosting. In the present, a web hosting provider is necessary to host the website.



Below are some of the best platforms to host static webpages[1]:

Hosting Providers	Major Features					
	Storage	Bandwidth	Custom Domains	SSL	Build Minutes	Free Packages
Netlify	Unlimited	100 GB/month	Yes	Yes (Via Let's Encrypt)	300/month	Available
Vercel	Unlimited	Up to 100 GB/month	Yes	Yes	Up to 45 minutes of build time per deployment	Available
Cloudflare Pages	Unlimited	Unlimited	Yes	Yes	500 builds per month	Available
GitHub Pages	Unlimited	Unlimited	Yes	Automatic SSL	-	Available

#### IV. WEB FORM LIMITATIONS IN STATIC WEBSITES

One of the major drawbacks is web form handling within the context of web hosting providers, it's essential to be aware of the various limitations and drawbacks that can impact the way collecting and storing data through web forms. These factors can have a substantial effect on website's functionality and overall user experience.

One of the primary limitations is the often-restricted server-side processing capabilities that hosting providers offer. This limitation means there might be difficulties when trying to create highly dynamic and interactive web forms. These hosting providers might not support custom server-side scripts, which are essential for advanced functionalities like conditional logic or dynamic data retrieval.

Another significant drawback is the limitations imposed on aspects like data formats and the number of form submissions

that can be received. Hosting providers might restrict the number of form submissions, potentially resulting in lost data or frustrated users. The absence of SSL(Secure Sockets Layer) support, which ensures secure data transmission, can also be a significant concern, especially when handling sensitive information through web forms. Furthermore, without the database features organizing and managing the collected data can become a cumbersome task.

To overcome these limitations and create a dynamic web form handling, need a cost effective and simple way to handle web forms, secure data collection and have a user-friendly interaction.

#### V. CURRENT SOLUTIONS FOR FORM SUBMISSION IN STATIC PAGES

Some of the primary Form service providers available in the market and their feature comparison is done below[2]:

Features	Form Service Providers				
	Reform	Formspree	Static Forms	FormKeep	Getform
Price(Basic Plan)	\$15/month[3]	\$8/month[4]	-Nil-	\$4.99/month[5]	\$19/month[6]
Customizable Forms	✓	✓	✓	✓	✓
Email Notifications	✓	✓	✓	✓	✓
Autoresponders	✓	✓	✗	✓	✓
Analytics	✓	✗	✗	✓	✗
Integration	✓	✓	✓	✓	✓
Spam Prevention	✓	✓	✓	✓	✓
Storage(Max Avail)	Inbox	Custom(∞)	Inbox	Up to 40GB	Up to 10GB
Third-Party Integrations	✓	✗	✓	✓	✓
Serverless Functions	✗	✗	✗	✓	✗
HTML Forms Integration	✓	✓	✓	✗	✓
Custom Branding	✓	✓	✗	✓	✓
Email Templates	✓	✓	✗	✓	✓
GDPR Compliance	✗	✓	✗	✓	✓
Redirects	✗	✗	✗	✗	✓

## VI. PROPOSAL: WEBHOOK INTEGRATION WITH GOOGLE SHEETS USING JAVASCRIPT

The major limitation of a static hosting is Form submission and its related operations hence proposing an idea by using the possibilities of JavaScript, Google Sheets and Webhook extension to make Web form functionalities for a static website.

Google sheet will act as a database and the webhook extension generates a webhook URL, the JavaScript code helps in data submission to the sheets. It extracts the form data using the Form Data constructor and sends it as a POST request to a specified webhook URL using the Fetch API.

### Installing Webhook Extension

- ❖ **Open Google Sheets:** Go to Google Sheets by visiting (<https://sheets.google.com>) and sign in to Google account if not already logged in.

- ❖ **Create or Open a Sheet:** Either create a new Google Sheet or open an existing one to use the Webhook extension.
- ❖ **Access Add-ons:** Click on "Add-ons" in the Extensions menu of the Google Sheet's top menu bar.
- ❖ **Get Add-ons:** In the dropdown menu that appears, select "Get add-ons."
- ❖ **Search for Webhook:**
  - In the "Add-ons" store, there is a search bar in the upper-right corner.
  - Type "Webhook" into the search bar and press Enter.
- ❖ **Install Webhook Extension:**
  - The "Webhook" extension listed there. Click on it to open the details.
  - Click the "Install" button.



### Webhook Extension

- There will be prompts to grant permissions. Click through these prompts to install the extension.

### Generating a Webhook URL

- 1. Open Webhook Extension:**

After installing the Webhook extension, access it by going to "Add-ons" in the top menu bar and selecting "Webhook" or a similar option depending on the extension's name.
- 2. Create a New Webhook:**

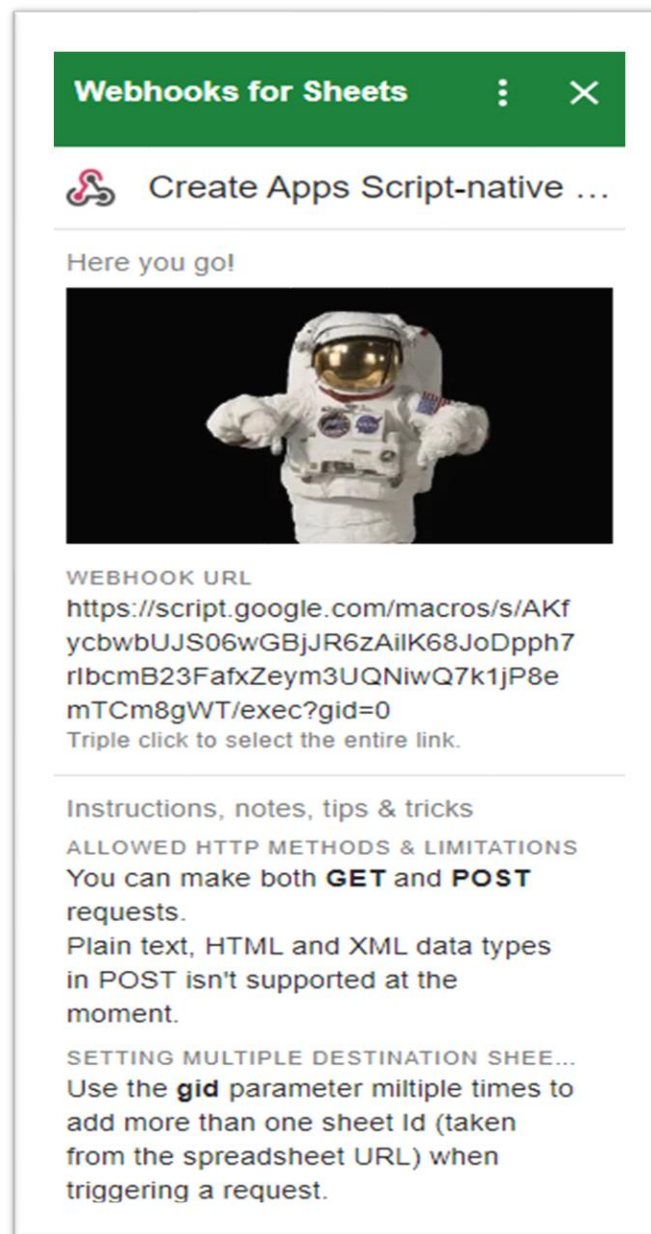
In the Webhook extension, find an option to create a new webhook. Click on it.

- 3. Configure the Webhook:**

Configure the webhook by providing information such as the event or trigger that should initiate the webhook, any authentication or headers required, and the destination URL where the webhook payload should be sent. Follow the prompts to set up the webhook.
- 4. Generate Webhook URL:**

As part of the setup process, there is an option to generate a Webhook URL. This URL will be used to receive data from from webhook.

### 5) Use the Webhook URL



The generated Webhook URL is used in applications, services, or other systems to send data to the Google Sheets.

**Webhook URL Syntax:** Protocol/Domain/Path/Script ID/exec?gid=

**Webhook URL Sample:** `https://script.google.com/macros/s/{Script ID}/exec?gid=' '`

**Protocol:** `https://`

**Domain:** `script.google.com`

**Path:** `/macros/s/` followed by the unique Script ID and `/exec`, which indicates the execution of a Google Apps Script web app.

**Script ID:** a unique identifier used as part of the URL to identify and access a specific Google Apps Script project.

**Query Parameter:** `?gid=0`

This URL is used to access and execute a specific Google Apps Script web app with the provided unique Script ID. The `?gid=0` query parameter can be used to access different sheets by changing the gid number.

**Form Syntax:**

```
<form id="myForm" class="jost-font" data-webhook-url="paste generated webhook url here">
  <!--FORM ELEMENTS HERE-->
</form>
<div id="successMessage" class="alert alert-success d-none mt-2">'Success Message Here'</div>
```



```
<div class="spinner-border text-primary d-none mt-2" role="status" id="spinner"></div>
```

### JavaScript Code

```
// Create error message container
```

```
const errorMessageDiv = document.createElement('div');  
errorMessageDiv.id = 'errorMessage';  
errorMessageDiv.className = 'alert alert-danger d-none mt-2';  
errorMessageDiv.textContent = 'There was an error submitting the form. Please try again.';  
document.getElementById('myForm').insertAdjacentElement('beforeend',  
errorMessageDiv);
```

```
document.getElementById('myForm').addEventListener('submit', function(event) {  
    event.preventDefault();  
    const webhookUrl = this.getAttribute('data-webhook-url');  
    const formData = new FormData(this);
```

```
// Disable submit button and show spinner
```

```
document.getElementById('submitBtn').setAttribute('disabled', true);  
document.getElementById('spinner').classList.remove('d-none');
```

```
// Clear previous messages
```

```
document.getElementById('successMessage').classList.add('d-none');  
errorMessageDiv.classList.add('d-none'); // Hide error message
```

```
fetch(webhookUrl, {  
    method: 'POST',  
    body: formData  
})
```

```
.then(response => {
```

```
    // Hide spinner
```

```
    document.getElementById('spinner').classList.add('d-none');
```

```
    if (response.ok) {
```

```
        // Show success message
```

```
        document.getElementById('successMessage').classList.remove('d-none');
```

```
    } else {
```

```
        // Show error message
```

```
        errorMessageDiv.classList.remove('d-none');
```

```
    }
```

```
})
```

```
.catch(error => {
```

```
    // Hide spinner
```

```
    document.getElementById('spinner').classList.add('d-none');
```

```
    // Show error message
```

```
    errorMessageDiv.classList.remove('d-none');
```

```
});
```

```
});
```

```
document.getElementById('resetBtn').addEventListener('click', function() {  
    document.getElementById('myForm').reset();  
    document.getElementById('submitBtn').removeAttribute('disabled');  
    document.getElementById('successMessage').classList.add('d-none');  
    errorMessageDiv.classList.add('d-none'); // Hide error message  
});
```

### Code Explanation

The code begins by creating an error message container called **errorMessageDiv**. This container is designed as a new `<div>` element, and it's configured with several attributes. These attributes include an 'id' of 'errorMessage' for easy identification, a 'className' that includes classes such as 'alert', 'alert-danger', 'd-none', and 'mt-2' for styling and visibility control, and a 'textContent' property that holds a default error message.

Subsequently, the error message container is inserted into the HTML form with the ID 'myForm', specifically placed just before the end of the form, so it can be used to display error messages.

Moving on, an event listener is added to the 'myForm' element. This event listener listens for the 'submit' event, which is triggered when the user attempts to submit the form. The event handler function is executed when this event occurs.

Inside the event handler function, the **event.preventDefault()** method is called to prevent the default form submission action from taking place. This allows the JavaScript code to handle the form submission process.

Next, the code retrieves the **data-webhook-url** attribute from the form element. This attribute is expected to contain the URL to which the form data will be sent.

The form data is then extracted using the **FormData** constructor, creating an object that holds the names and values of the form fields.

To provide feedback to the user during form submission, the code disables the submit button by setting its 'disabled' attribute to true. Additionally, it

displays a spinner element (with the ID 'spinner') to indicate that the form is being processed.

Any previous messages, whether they are success or error messages, are hidden. The success message is hidden by adding the 'd-none' class to it, and the error message container (**errorMessageDiv**) is also hidden to prepare for displaying new messages.

The code proceeds to send a POST request to the URL specified in the **webhookUrl** using the **fetch** API. The form data is included in the request body.

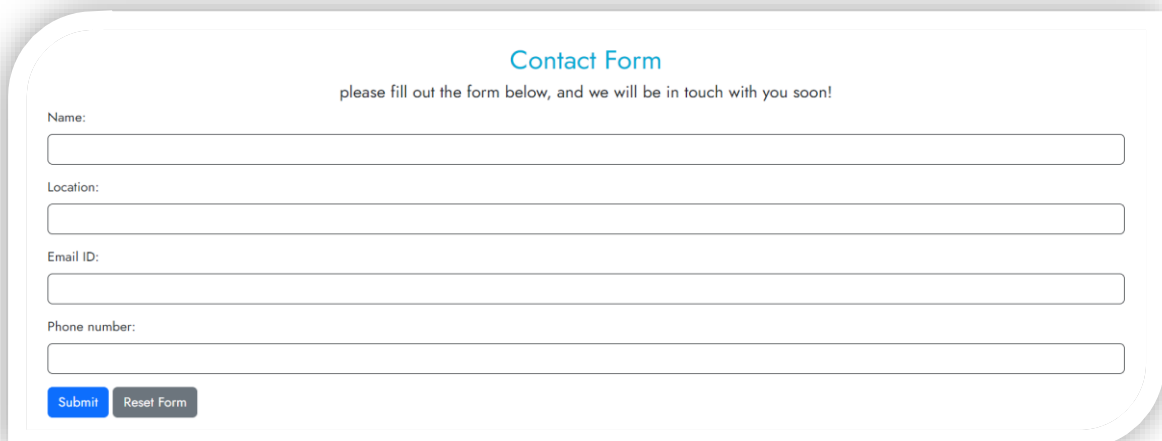
Upon completion of the request, a **then** block handles the response. If the response has an HTTP status code in the range of 200-299 (indicating a successful response), the success message is displayed. Conversely, if the response is not successful, the error message container is displayed.

In the event of an error during the request, such as a network error, a **catch** block is triggered. In this block, the spinner is hidden, and the error message container (**errorMessageDiv**) is shown to inform the user of the issue.

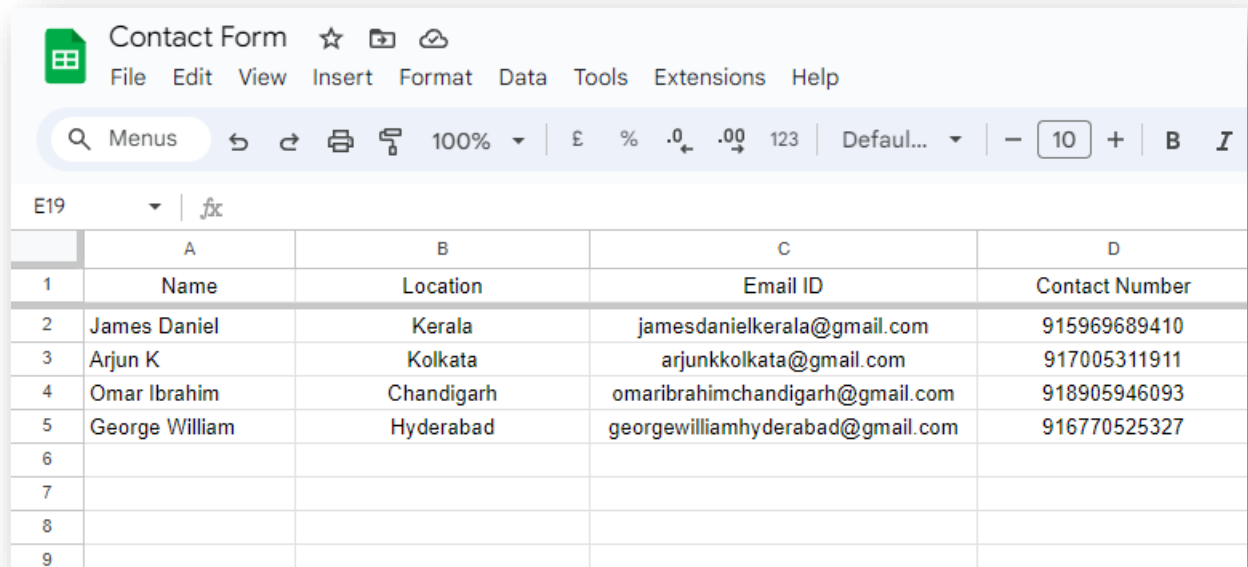
Lastly, the code adds an event listener to a button with the ID 'resetBtn'. When this button is clicked, it performs several actions. First, it resets the form with the ID 'myForm' by clearing all form fields. Then, it removes the 'disabled' attribute from the submit button, making it available for future submissions. It also hides the 'successMessage' by adding the 'd-none' class and hides the error message container (**errorMessageDiv**) to reset the user interface.

## VII. RESULTS

With the above specified integration, web form submission and data collection are possible for static websites. Below is a sample HTML form outlook with webhook integration.



The screenshot shows a contact form titled "Contact Form" with the instruction "please fill out the form below, and we will be in touch with you soon!". The form contains four input fields: "Name:", "Location:", "Email ID:", and "Phone number:". At the bottom left, there are two buttons: "Submit" (in blue) and "Reset Form" (in grey).



	A	B	C	D
1	Name	Location	Email ID	Contact Number
2	James Daniel	Kerala	jamesdanielkerala@gmail.com	915969689410
3	Arjun K	Kolkata	arjunkolkata@gmail.com	917005311911
4	Omar Ibrahim	Chandigarh	omaribrahimchandigarh@gmail.com	918905946093
5	George William	Hyderabad	georgewilliamhyderabad@gmail.com	916770525327
6				
7				
8				
9				

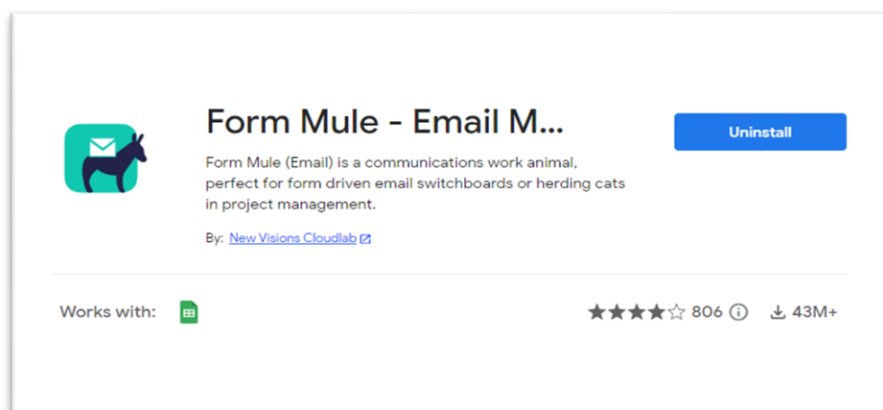
A demo with sample data generated using online tool[7].

### VIII. ADVANTAGES, DRAWBACKS & ADDITIONAL FEATURE

The major advantage of this implementation is that its cost free. The google sheets here acts as a database for data storage. There is no requirement of server-side functionalities which in turn save costs. Unlimited forms can be submitted. Multiple sheets can be used by changing the **gid** number. The google sheets has user friendly environment so it's easy for handling, sorting and analysing data. It offers robust data import and export capabilities. Doesn't require software installation, making it accessible from various operating systems. It can handle large

datasets and is suitable for both personal and enterprise-level data management and also provides automatic backups, robust security features, and compliance with industry standards, making it a secure option for data storage.

Even though there are many advantages there exists some drawbacks for this method, out of which one is that the users won't receive email notification after they submit the form. To overcome this, there is an existing feature of sheets , an additional extension named Form Mule which allows to set time triggered mailing facility. The mail content can be written as per our choice.



### Form Mule Extension

### VIII. CONCLUSION

In conclusion, this study has examined the limitations of web form handling within the static web hosting. These limitations encompass restricted server-side processing capabilities, data submission restrictions, and security concerns. To address these challenges, proposed an innovative approach that leverages

Google Sheets as a database for web form data. By using JavaScript and Webhook extensions, web forms can be seamlessly integrated with Google Sheets, allowing for data submission and secure storage. This method is not only cost-effective but also user-friendly, offering an alternative to traditional server-side processing. Additionally, introduced



Form Mule as a solution for email notifications. The integration of these solutions enhances the functionality of static websites, making them more efficient and user-oriented. Future directions may include further refinements to the integration process and exploring additional extensions to meet specific web form needs.

## VIII. REFERENCES

1. Mehedi Sharif,(2023), " 10 Best Free Static Website Hosting Providers In 2023", Gethugothemes, <https://gethugothemes.com/best-free-static-website-hosting>.
2. "8 Tools to Use Forms on a Static Site", (2021), simplystatic.com, <https://simplystatic.com/tutorials/forms-on-a-static-site/>.
3. Bjorn Lindholm, Peter Suhm,(2021), "Add Reform to your tool belt", Reform, <https://www.reform.app/pricing/>.
4. "Formspree Plans", (2023), Formspree Inc. <https://formspree.io/plans>.
5. David,(2023), "Features & Pricing", FormKeep Inc. <https://formkeep.com/features/pricing>.
6. "The best plan for you", (2023), Getform, <https://getform.io/pricing>.
7. "Free Fake Phone Number Generator Tool", (2023), Texttr Inc, <https://texttrapp.com/free-tools/texttr-free-fake-phone-number-generator-tool>.