email js

**ChatGPT đã nói:**

Could you clarify what you mean by **"email js"**? There are a few possible interpretations:

1. **EmailJS (the service)**: A third-party service that lets you send emails directly from JavaScript without needing a backend. Useful for contact forms on static websites.
2. **Sending email using JavaScript in general**: Might involve using SMTP.js, EmailJS, or connecting to a backend that sends emails.
3. **Writing JavaScript code to send an email**: Could involve Node.js and libraries like nodemailer.

If you're referring to **EmailJS (emailjs.com)**, here’s a basic example of how it works:

**✅ How to use EmailJS (Frontend only – no backend needed)**

**1. Include the EmailJS SDK:**

html

Sao chépChỉnh sửa

<script type="text/javascript" src="https://cdn.emailjs.com/dist/email.min.js"></script>

<script type="text/javascript">

(function(){

emailjs.init("YOUR\_USER\_ID"); // Get from EmailJS dashboard

})();

</script>

**2. Create an HTML form:**

html

Sao chépChỉnh sửa

<form id="contact-form">

<input type="text" name="from\_name" placeholder="Your Name" required />

<input type="email" name="email" placeholder="Your Email" required />

<textarea name="message" placeholder="Your Message" required></textarea>

<button type="submit">Send</button>

</form>

**3. Add JavaScript to handle the form submission:**

html

Sao chépChỉnh sửa

<script>

document.getElementById('contact-form').addEventListener('submit', function(e) {

e.preventDefault();

emailjs.sendForm('YOUR\_SERVICE\_ID', 'YOUR\_TEMPLATE\_ID', this)

.then(function(response) {

console.log('SUCCESS!', response.status, response.text);

alert("Message sent!");

}, function(error) {

console.log('FAILED...', error);

alert("Failed to send message.");

});

});

</script>

Let me know if you’re using **Node.js**, **SMTP.js**, or something else, and I can tailor the answer to your setup.

**Bạn đã nói:**

how to get the userid ?

**ChatGPT đã nói:**

To get your **User ID** (or more accurately, your **Public Key**) from **EmailJS**, follow these steps:

**🔑 How to Get Your EmailJS User ID (Public Key):**

1. **Go to** [**EmailJS.com**](https://www.emailjs.com)  
   Log in to your account, or create one if you haven’t yet.
2. **Open the Dashboard**  
   Once logged in, go to your **Dashboard**.
3. **Find Your Public Key**:
   * Click on your **Account** name in the top right corner.
   * Go to **Account > API Keys**.
   * Copy the **Public Key** (it will look like something like this: C6a7dXXXXXXXaVSeX).

This is the key you'll use in:

js

Sao chépChỉnh sửa

emailjs.init("YOUR\_PUBLIC\_KEY");

**⚠️ Important:**

* The **Public Key** is safe to use in client-side code.
* Do **not** expose your **Private Key** in frontend code.

Let me know if you also need help setting up the **Service ID** and **Template ID** — both are required to send emails through EmailJS.

**Bạn đã nói:**

give me exmaple mail content

**ChatGPT đã nói:**

Sure! Here's an example of a **basic email content** you might send using **EmailJS**, such as for a contact form on a website.

**✅ Email Template Example (Content)**

**1. Variables in your EmailJS Template (you define these on the EmailJS website):**

* from\_name
* email
* message

**2. Email Content Template (on EmailJS site):**

css

Sao chépChỉnh sửa

📩 New Contact Form Submission

Name: {{from\_name}}

Email: {{email}}

Message:

{{message}}

Sent via your website contact form.

This template will get filled dynamically when you send the form.

**📋 Sample Input from User (HTML Form)**

html

Sao chépChỉnh sửa

<form id="contact-form">

<input type="text" name="from\_name" placeholder="Your Name" required />

<input type="email" name="email" placeholder="Your Email" required />

<textarea name="message" placeholder="Your Message" required></textarea>

<button type="submit">Send</button>

</form>

**🧠 What the Email Will Look Like (example):**

vbnet

Sao chépChỉnh sửa

📩 New Contact Form Submission

Name: Jane Doe

Email: jane@example.com

Message:

Hi, I love your website. Can you help me with pricing?

Sent via your website contact form.

Let me know if you want a **HTML version** of the email or a different style (e.g., feedback, support, newsletter subscription).

**Bạn đã nói:**

for added to the event

**ChatGPT đã nói:**

If you want to send an email for an **event registration** or someone being added to an event, here’s an example of how to structure the email content for that.

**✅ Example Email Content for Event Registration**

**1. Variables for Event Registration (on EmailJS Template):**

* name (Name of the person registering)
* email (Email of the person registering)
* event\_name (Name of the event)
* event\_date (Date of the event)
* event\_time (Time of the event)

**2. Email Content Template (on EmailJS site):**

css

Sao chépChỉnh sửa

🎉 You Have Successfully Registered for the Event!

Dear {{name}},

Thank you for registering for the event: \*\*{{event\_name}}\*\*.

Event Details:

- Date: {{event\_date}}

- Time: {{event\_time}}

We are excited to have you join us!

Best regards,

The Event Team

This template is designed to notify both you and the user that they have been added to an event.

**📋 Example Input (HTML Form)**

html

Sao chépChỉnh sửa

<form id="event-registration-form">

<input type="text" name="name" placeholder="Your Name" required />

<input type="email" name="email" placeholder="Your Email" required />

<input type="text" name="event\_name" placeholder="Event Name" required />

<input type="date" name="event\_date" required />

<input type="time" name="event\_time" required />

<button type="submit">Register</button>

</form>

**🧠 Example of the Email Received:**

**For the person registering:**

vbnet

Sao chépChỉnh sửa

🎉 You Have Successfully Registered for the Event!

Dear John Doe,

Thank you for registering for the event: \*\*Tech Conference 2025\*\*.

Event Details:

- Date: 2025-06-15

- Time: 10:00 AM

We are excited to have you join us!

Best regards,

The Event Team

**🧑‍💻 JavaScript Code to Handle Submission:**

html

Sao chépChỉnh sửa

<script>

document.getElementById('event-registration-form').addEventListener('submit', function(e) {

e.preventDefault();

emailjs.sendForm('YOUR\_SERVICE\_ID', 'YOUR\_TEMPLATE\_ID', this)

.then(function(response) {

console.log('SUCCESS!', response.status, response.text);

alert("Registration successful! Check your email for confirmation.");

}, function(error) {

console.log('FAILED...', error);

alert("Failed to register.");

});

});

</script>

Let me know if you'd like further customization for this email, or if you need to add more details to the registration process!

**Bạn đã nói:**

explain the js code, why only submit the form and then the emailjs can get all element ?

**ChatGPT đã nói:**

Great question! The key to understanding how the **JavaScript code** works with **EmailJS** lies in how **form data** is handled when you submit the form and how **EmailJS** is able to access all form elements without manually extracting them one by one.

Let's break down the code and explain step by step:

**Step-by-Step Explanation of the Code:**

javascript

Sao chépChỉnh sửa

document.getElementById('event-registration-form').addEventListener('submit', function(e) {

e.preventDefault(); // Prevents the default form submission behavior

emailjs.sendForm('YOUR\_SERVICE\_ID', 'YOUR\_TEMPLATE\_ID', this) // Sends the form data to EmailJS

.then(function(response) {

console.log('SUCCESS!', response.status, response.text);

alert("Registration successful! Check your email for confirmation.");

}, function(error) {

console.log('FAILED...', error);

alert("Failed to register.");

});

});

**Key Elements in the Code:**

1. **Form Element:**

html

Sao chépChỉnh sửa

<form id="event-registration-form">

<!-- Form fields like name, email, event details -->

</form>

This is the form where the user will input their data.

1. **Event Listener (JavaScript):**

javascript

Sao chépChỉnh sửa

document.getElementById('event-registration-form').addEventListener('submit', function(e) {

e.preventDefault();

});

This attaches an **event listener** to the form that listens for the submit event. The **submit event** is triggered when the user clicks the "Register" button or presses enter in any form field.

1. **Prevent Default Submission (e.preventDefault()):**

javascript

Sao chépChỉnh sửa

e.preventDefault();

Normally, when a form is submitted, it would reload the page or redirect to another URL. **e.preventDefault()** prevents this default behavior so that the form submission doesn't reload the page. Instead, the JavaScript code inside this function gets executed, which in this case sends the form data to EmailJS.

1. **Sending Form Data to EmailJS (emailjs.sendForm)**:

javascript

Sao chépChỉnh sửa

emailjs.sendForm('YOUR\_SERVICE\_ID', 'YOUR\_TEMPLATE\_ID', this);

Here’s the magic part:

* + **emailjs.sendForm** is a built-in method that sends the form data to the **EmailJS** service.
  + The **first argument** ('YOUR\_SERVICE\_ID') is the **Service ID** you configured on EmailJS, which identifies the service you're using (like Gmail, Outlook, etc.).
  + The **second argument** ('YOUR\_TEMPLATE\_ID') is the **Template ID** of the email template you created on EmailJS.
  + The **third argument** (this) refers to the **form element** itself — the form that the user just submitted. It’s the **entire form element** that contains all the input fields.

**Why does this work?**  
When you pass this (the form) to emailjs.sendForm, the method automatically collects all the input fields (like name, email, event\_name, etc.) and their corresponding values. You don’t need to manually extract them one by one.

**How does it know what to send?**  
The emailjs.sendForm function looks at the **name attributes** of the input fields in the form. These are used as variables in the template you created on EmailJS. For example, if you have a field like:

html

Sao chépChỉnh sửa

<input type="text" name="name" />

EmailJS automatically associates the value entered by the user with the variable {{name}} in the email template you created.

When you use emailjs.sendForm(this), it sends all the form data to EmailJS without needing to manually reference each form field.

1. **Then, Handling the Response:**

javascript

Sao chépChỉnh sửa

.then(function(response) {

console.log('SUCCESS!', response.status, response.text);

alert("Registration successful! Check your email for confirmation.");

}, function(error) {

console.log('FAILED...', error);

alert("Failed to register.");

});

* + If the email is sent successfully, the then() function is triggered, which logs a success message and shows an alert to the user.
  + If there's an error (e.g., network issue, invalid API keys), the catch() block is triggered, logging the error and alerting the user that the registration failed.

**Summary:**

* **emailjs.sendForm** can automatically collect all form elements and their values because you're passing the **form element (this)** directly to the function. It looks for the **name attributes** in the form fields and uses those names as placeholders in your **EmailJS template**.
* The e.preventDefault() stops the default form submission so you can handle it with JavaScript and send the form data to **EmailJS** for email sending.

By using **sendForm**, you're essentially sending the entire form's data in one go, without needing to manually gather each input field's value. EmailJS handles the data extraction from the form and populates the corresponding template placeholders automatically.

**Bạn đã nói:**

ah i see, so in the html, just define the name as attributes which defined on the emailjs dashboard and then it auto get it ?

**ChatGPT đã nói:**

Exactly! You’ve got it! 🎉

**How It Works:**

* **In your HTML form**, you define the **name attribute** for each input field. These name attributes will match the variable names that you define in your **EmailJS template** on the dashboard.
* **When the form is submitted**, EmailJS looks at the **name attributes** of the form fields and uses them to fill in the placeholders in the template.

**Step-by-Step Example:**

1. **HTML Form:**  
   In your HTML, define the form fields with **name attributes**:

html

Sao chépChỉnh sửa

<form id="event-registration-form">

<input type="text" name="name" placeholder="Your Name" required />

<input type="email" name="email" placeholder="Your Email" required />

<input type="text" name="event\_name" placeholder="Event Name" required />

<input type="date" name="event\_date" required />

<input type="time" name="event\_time" required />

<button type="submit">Register</button>

</form>

1. **EmailJS Template:**  
   On the **EmailJS dashboard**, you create a template, and inside the template, you define placeholders that match the **name attributes** of your form fields. For example:

css

Sao chépChỉnh sửa

🎉 You Have Successfully Registered for the Event!

Dear {{name}},

Thank you for registering for the event: \*\*{{event\_name}}\*\*.

Event Details:

- Date: {{event\_date}}

- Time: {{event\_time}}

We are excited to have you join us!

Best regards,

The Event Team

* + **{{name}}**, **{{email}}**, **{{event\_name}}**, **{{event\_date}}**, and **{{event\_time}}** in the template are placeholders that correspond to the **name attributes** of your form fields (name, email, event\_name, etc.).

1. **EmailJS Will Automatically Fill in the Data:**  
   When the user submits the form, **EmailJS automatically matches the form data with the placeholders** in the template based on the **name attributes**.

So, if the user enters:

* + **Name:** John Doe
  + **Email:** john@example.com
  + **Event Name:** Tech Conference 2025
  + **Event Date:** 2025-06-15
  + **Event Time:** 10:00 AM

The email sent through EmailJS will look like:

vbnet

Sao chépChỉnh sửa

🎉 You Have Successfully Registered for the Event!

Dear John Doe,

Thank you for registering for the event: \*\*Tech Conference 2025\*\*.

Event Details:

- Date: 2025-06-15

- Time: 10:00 AM

We are excited to have you join us!

Best regards,

The Event Team

**The Magic: Form Field name Attribute Matches Template Variables**

* The **name attribute** in the HTML form field corresponds directly to the **placeholder variable** in the EmailJS template.
* When you call emailjs.sendForm(), EmailJS looks for the name attributes in the form, automatically maps the values to the placeholders in your template, and sends the email.

**Why This Works:**

* The name attribute in HTML is used to associate form data with the variable names in your email template.
* **EmailJS** uses the **form’s name attributes** to dynamically populate the email template, so you don't have to manually handle form field data in JavaScript.

**Key Takeaway:**

* **Just define the name attribute** in your form elements (HTML) to match the placeholders in the EmailJS template.
* **EmailJS will automatically collect the form data** based on these name attributes and replace the placeholders in the email template with the actual values.

Feel free to let me know if you have more questions or if anything is unclear! 😊