Creating a Custom Payment Form with Stripe.js

This tutorial helps you build your first payment form with Stripe. If you need help after reading this, check out our answers to common questions or chat live with other developers in #stripe on freenode.

Processing payments with Stripe has two components:

- Securely collecting the payment details
- Using the collected payment method in a charge request

This tutorial explains how to use HTML and Stripe.js, our foundational JavaScript library, to implement the first component. There are three steps:

- Collecting credit card information with Stripe.js
- Converting those payment details to a single-use token
- Submitting that token, with the rest of your form, to your server

Other options

If you'd rather not build your own payment form, consider Checkout, our embedded, optimized payment tool. Or avoid coding altogether with Stripe's third-party integrations. In addition, you may wish to add Apple Pay to your website to provide a better experience for customers on iOS devices.

The above steps assume you have a typical HTML payment form:

```
<form action="/your-charge-code" method="POST" id="payment-form">
  <span class="payment-errors"></span>

  <div class="form-row">
    <label>
      <span>Card Number</span>
      <input type="text" size="20" data-stripe="number">
    </label>
  </div>

  <div class="form-row">
    <label>
      <span>Expiration (MM/YY)</span>
      <input type="text" size="2" data-stripe="exp_month">
    </label>
    <span> / </span>
    <input type="text" size="2" data-stripe="exp_year">
  </div>

  <div class="form-row">
    <label>
      <span>CVC</span>
```

```
        <input type="text" size="4" data-stripe="cvc">
      </label>
    </div>

    <div class="form-row">
      <label>
        <span>Billing Zip</span>
        <input type="text" size="6" data-stripe="address_zip">
      </label>
    </div>

    <input type="submit" class="submit" value="Submit Payment">
  </form>
```

The HTML is fairly standard, but note how every input for sensitive data—number, CVC, expiration, and zip code—omits the `name` attribute. By omitting a `name`, the user-supplied data in those fields won't be passed to your server when the form is submitted. Each element also includes a `data-stripe` attribute, to be discussed later.

With Stripe.js, you never have to handle sensitive card data. It's automatically converted to a representative *token* that you can safely send to your servers and use to charge your customers.

> **Do not give names to the payment details form elements!** This prevents that data from touching your server, which means you no longer need to worry about redacting logs, encrypting cardholder details, or other burdens of PCI compliance.

Step 1: Collecting credit card information

To securely collect the credit card information, first include Stripe.js in your page:

```
<script type="text/javascript" src="https://js.stripe.com/v2/"></script>
```

To prevent problems with some older browsers, put the script tag in the page `<head>` or as the final line in the `<body>` at the end of your code.

> **Note**: Stripe.js should be loaded directly from **https://js.stripe.com/v2/**.

In a separate script tag, after the above, set your publishable key:

```
<script type="text/javascript">
  Stripe.setPublishableKey('pk_test_6pRNASCoBOKtIshFeQd4XMUh');
</script>
```

Your publishable API key identifies your website to Stripe during communications. We've placed a random API key in the code. Replace it with your actual publishable API key to test this code through your Stripe account.

You will need to replace the test key with your live key for production uses. When you're ready, learn more about how the keys play into test and live modes.

Step 2: Create a single use token

The second step is to convert the payment details into a single-use representative token. This is done by interrupting the form's submission and passing the payment details directly to Stripe (from the user's browser).

Library agnostic

While this example uses jQuery, Stripe.js has no external dependencies. You can use vanilla JavaScript if you'd rather.

After the code above, create an event handler that handles the `submit` event on the form. The handler should send the form data to Stripe for tokenization and prevent the form's submission. (The form will be submitted by JavaScript later.)

```
$(function() {
  var $form = $('#payment-form');
  $form.submit(function(event) {
    // Disable the submit button to prevent repeated clicks:
    $form.find('.submit').prop('disabled', true);

    // Request a token from Stripe:
    Stripe.card.createToken($form, stripeResponseHandler);

    // Prevent the form from being submitted:
    return false;
  });
});
```

The most important line is the call to `Stripe.card.createToken`. The first argument is the payment details. This can either be provided as individual members of a generic object, or by passing along a reference to the entire form. In the latter case, as in the above, the relevant values are fetched from their associated inputs using the `data-stripe` attribute specified in the HTML code example at the top of the page.

Although optional, using address and postal code verifications is highly recommended as they'll help reduce fraud. The complete list of fields you can provide is available in the Stripe.js reference.

The second argument to `Stripe.card.createToken` — `stripeResponseHandler` —is a callback you provide to handle the response from Stripe. The handler is written to accept two arguments:

- `status` is one of the status codes described in the API docs
- `response` is of the following form:

```
{
  id: "tok_u5dg20Gra", // Token identifier
  card: {...}, // Dictionary of the card used to create the token
  created: 1478099077, // Timestamp of when token was created
  currency: "usd", // Currency that the token was created in
  livemode: false, // Whether this token was created with a live API key
  object: "token", // Type of object, always "token"
  used: false // Whether this token has been used
}
```

Understand that `Stripe.card.createToken` is an *asynchronous* call. The function returns immediately and calls `stripeResponseHandler` when it receives a response from Stripe's servers. This whole process only takes a few moments, and your customer never leaves your site or even the payment form page during that time.

Step 3: Sending the form to your server

The third and final step to securely collect your customer's payment details is to submit the received token to your server for final use. When your script receives the response from Stripe's servers, the `stripeResponseHandler` function is called:

- If the card information entered by the user returned an error, the error gets displayed on the page
- If no errors were returned—a single-use token was created successfully, add the returned token ID to the form and submit the form to your server

Your server

Stripe.js and your payment form alone do not create a charge. Combined, Stripe.js and the HTML form fulfill the first half of the payment process, but server-side code is also necessary to complete the workflow.

```
function stripeResponseHandler(status, response) {
  // Grab the form:
  var $form = $('#payment-form');

  if (response.error) { // Problem!

    // Show the errors on the form:
    $form.find('.payment-errors').text(response.error.message);
    $form.find('.submit').prop('disabled', false); // Re-enable submission

  } else { // Token was created!

    // Get the token ID:
    var token = response.id;

    // Insert the token ID into the form so it gets submitted to the server:
    $form.append($('<input type="hidden" name="stripeToken">').val(token));

    // Submit the form:
    $form.get(0).submit();
  }
};
```

To get the token to your server, it's stored in a new hidden input. Its value is the received token's ID.

After adding the token ID, the form is submitted using JavaScript. (As a reminder, the form's submission was prevented earlier so the script could wait for Stripe to tokenize the credit card details.) All of the form's data will be sent as a POST request to the URL in the form's `action`. If you have other form elements, such as the user's email address, that will be submitted per usual.

Demo

This live demonstration uses a cosmetic variation on the HTML and JavaScript explained above. It also includes basic validation before making the token request of Stripe. We've written a general purpose library called [jQuery.payment] to help with client-side input validation and formatting card numbers.

Number	4242 4242 4242 4242		
Expiry	12 / 17	**CVC**	123
Zip Code	12345		

The form has been pre-filled with valid test values, but feel free to change any of them, using:

- One of Stripe's [test card numbers], such as **4242 4242 4242 4242**
- Any three digit CVC code
- Any expiration date in the future

Any billing postal code, such as **12345**

Next steps

Once you've sent your form to your server, you're going to want to do something with the payment details you just collected (in the form of a token). This is usually one of two things:

- Charge your user immediately
- Sign them up for a subscription

The token is single-use only and has a short life. Use it in an API call immediately. If you wish to charge a credit card multiple times or just at a later time, you should use your token to create a Customer object.

Questions?

We're always happy to help with code or other questions you might have! Search our site for more information or send us an email!