Integrating Checkout

Stripe's embedded payment form, Checkout, simplifies and secures online payment processing. Quickly integrate Checkout into your site to provide your users with a streamlined, mobile-ready payment experience that is constantly improving. If you need help after reading this, check out our answers to common questionsor chat live with other developers in #stripe on freenode.

The easiest way to integrate Stripe is via Checkout, an embedded tool that takes care of building an HTML form, validating user input, and securing your customers' card data. Using Checkout, sensitive credit card information is sent directly to Stripe, and does not touch your server. Stripe returns to your site a *token*representation of the card, and this token can then be used in a charge request.

Complete requirements

Checkout alone does not create a charge. Checkout fulfills the first half of the payment process, but server-side code is also necessary.

Checkout is a fairly high-level solution. For a more custom approach, you can instead use Stripe with your own payment form via Stripe.js.

To see Checkout in action, click the button below, filling in the resulting form with:

- Any random, syntactically valid email address (the more random, the better)
- 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**

Pay with Card

Embedding Checkout in your site

To get started, add the following code to your payment page, making sure that the form submits to your own server-side code within the **action** attribute:

```
<form action="/your-server-side-code" method="POST">
  <script
```

```
      src="https://checkout.stripe.com/checkout.js" class="stripe-button"
      data-key="pk_test_6pRNASCoBOKtIshFeQd4XMUh"
      data-amount="999"
      data-name="Stripe.com"
      data-description="Widget"
      data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
      data-locale="auto"
      data-zip-code="true">
    </script>
  </form>
```

The most important line is the `data-key` attribute added to the script tag. This key identifies your account when communicating with Stripe. 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.

The above configuration also accepts the user's postal code, when applicable, and passes this to Stripe. Although optional, using address and postal code verifications is highly recommended as they'll help reduce fraud.

An alternative to the blue button demonstrated above is to implement a custom Checkout integration. The custom approach allows you to use any HTML element or JavaScript event to open Checkout, as well as be able to specify dynamic arguments, such as custom amounts.

Generating and using tokens

With Stripe, sensitive cardholder data does not hit your server, greatly minimizing your PCI compliance burden. Stripe takes care of the hardest parts of PCI compliance, like redacting logs and encrypting card details. Just enable HTTPS on your checkout page, and we'll take over from there.

Here's the whole workflow:

The customer arrives at your payment page that includes the Checkout code, loaded over HTTPS.

The customer clicks the payment button (e.g., **Pay with Card**), completes the payment form, and clicks **Pay $9.99** within the Checkout window (or whatever your Checkout pay button is).

Example integrations

Read a Checkout integration guide to get up and running as quickly as possible:

- Sinatra
- Rails
- Flask
- PHP

Checkout sends the payment details directly to Stripe from the customer's browser, assuming the details pass basic validation.

Stripe returns a token to Checkout, or an error message if the card-network validation fails.

Checkout takes the returned token and stores it in the page's primary form—the one surrounding the script tag above, in a hidden element named `stripeToken`.

Checkout submits the form to your server.

Your server uses the posted token to charge the card.

If need be, revisit the Checkout demo at the top of the page to see these first four steps in action.

Checkout, running in the browser, securely accepts the payment information but does not initiate a payment attempt. The actual charge request is triggered from your server.

Next steps

Once the form, with the stored payment token, is submitted to your server, you'll want to use the payment details just collected. Usually this means one of three actions:

Charging the customer immediately

Saving the credit card details for later

Signing your customer up to a subscription

Checkout Reference

For a more advanced Checkout integration, learn how to dynamically display a user-supplied charge amount. You can also learn how to use Checkout to handle customer card updates on your website.

Questions?