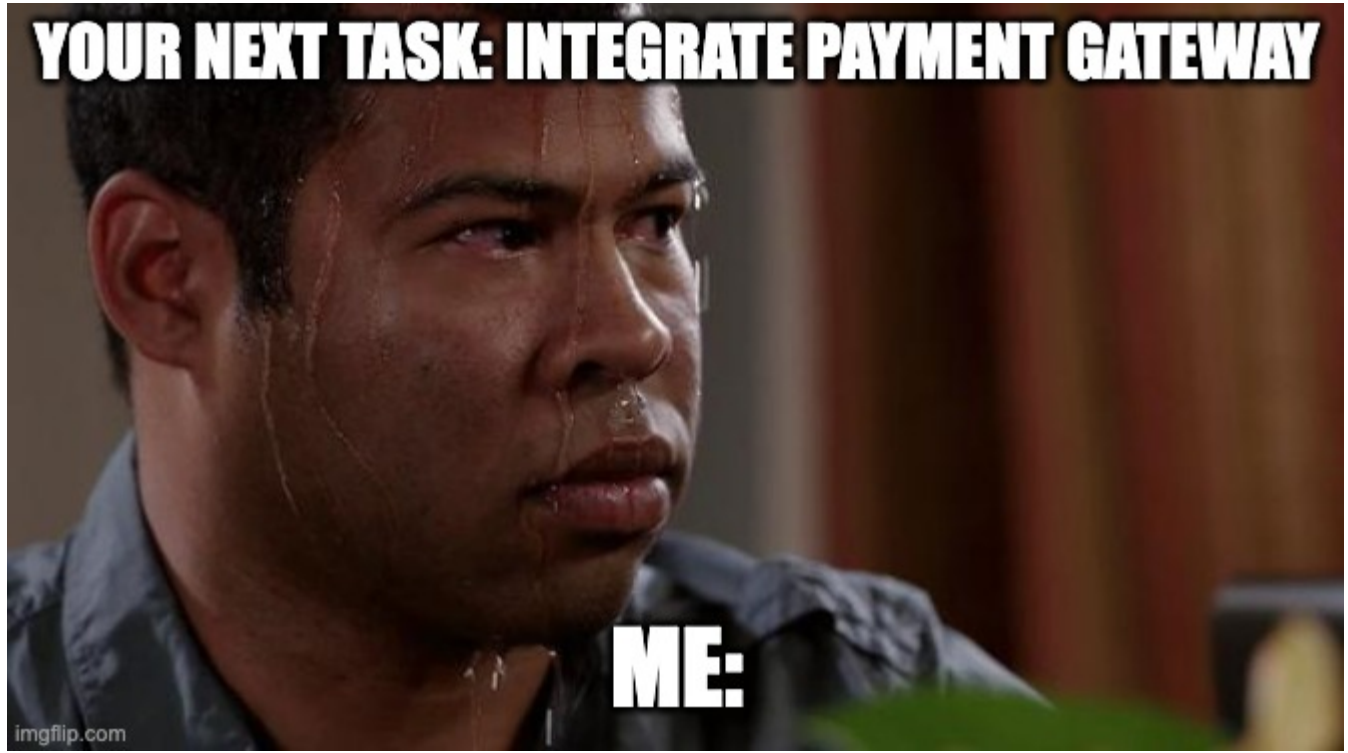
Welcome to Stripe Training

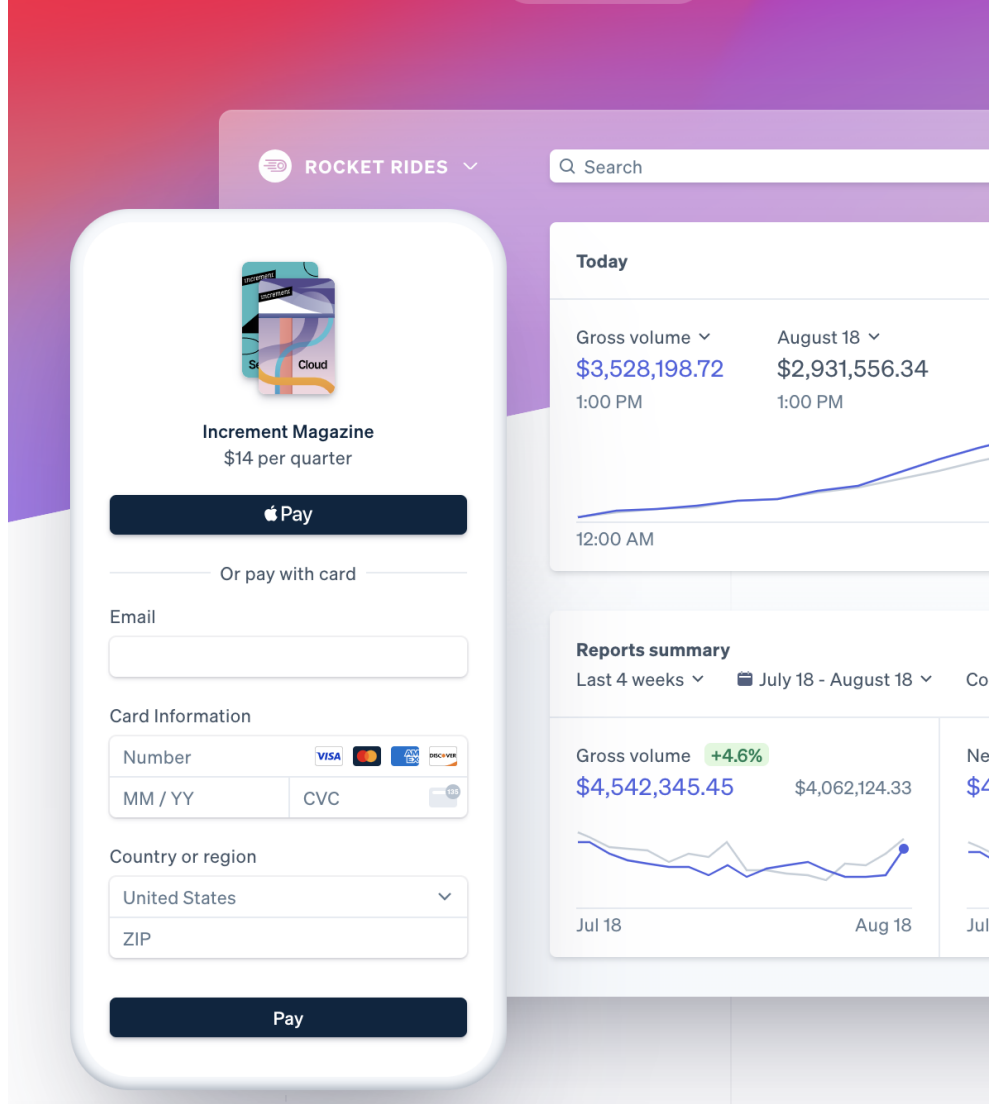
NUS Technology, 18 Nov 2023


YOUR NEXT TASK: INTEGRATE PAYMENT GATEWAY
ME:
imgflip.com

Introduction to Stripe Development

Stripe is an online payment processing platform that allows businesses to securely **accept payments** over the internet.

It provides a set of APIs (Application Programming Interfaces) and tools that enable businesses to handle payments, subscriptions, and manage their revenue streams.



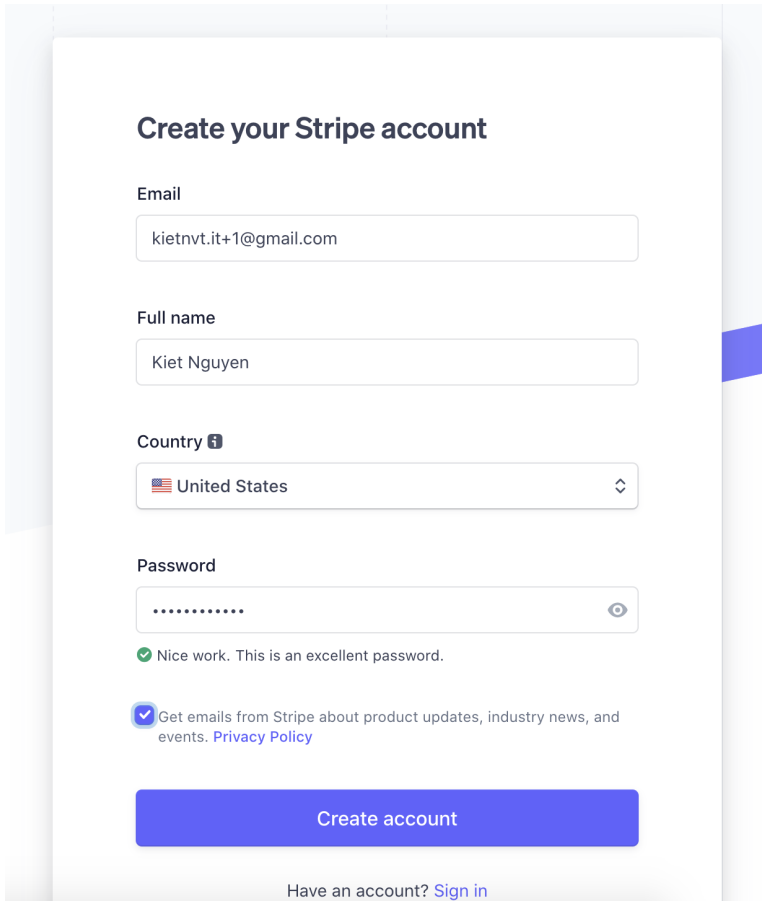
Benefits of Using Stripe for Development

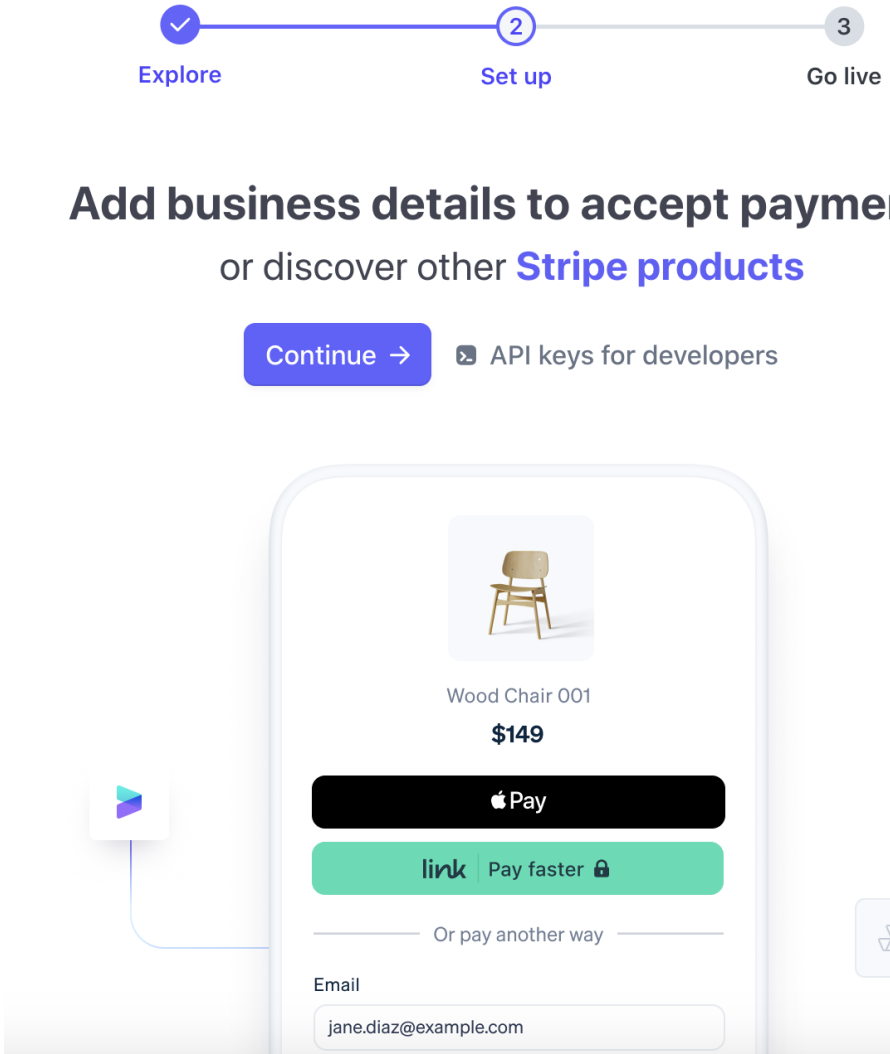
- **Easy Integration**: Stripe offers well-documented APIs and libraries
- **Security**: Stripe is PCI-DSS compliant, developers can offload much of the burden of securing payment data to Stripe's infrastructure.
- **Flexibility**: Stripe supports a wide range of payment methods, including credit cards, debit cards, Apple Pay, Google Pay, and various other digital wallets.
- **International Support**: Stripe is available in over 40 countries and supports payments in more than 135 currencies.
- **Subscription and Recurring Payments**
- **Invoicing and Billing**: Stripe offers features for generating and managing invoices
- **Customizable Checkout Flows**
- **Support and Community**: Stripe offers support through various channels, including email and chat.

Creating a Developer Account

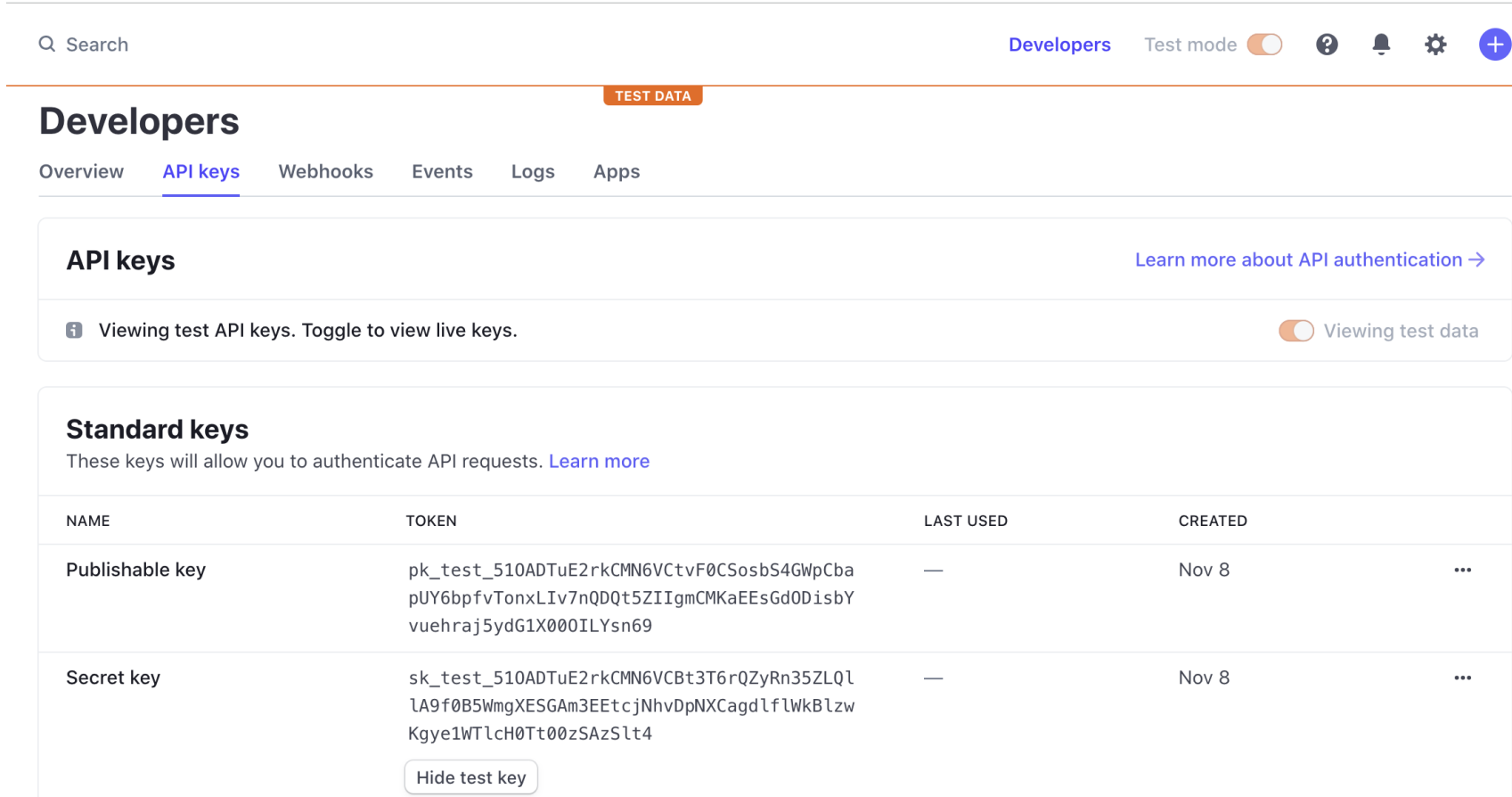
- Setting up a new Stripe Account
- Accessing the Developer Dashboard
- Generating and Managing API Keys
- Starting to made payment

Set up


Create your Stripe account
Email
kietnvt.it+1@gmail.com
Full name
Kiet Nguyen
Country
United States
Password
Nice work. This is an excellent password.
Get emails from Stripe about product updates, industry news, and events. Privacy Policy
Create account
Have an account? Sign in


Explore
2
Set up
3
Go live
Add business details to accept payments
or discover other Stripe products
Continue →
API keys for developers
Wood Chair 001
$149
Pay
link Pay faster
Or pay another way
Email
jane.diaz@example.com

Set up


Search
Developers
Test mode
TEST DATA
Developers
Overview
API keys
Webhooks
Events
Logs
Apps
API keys
Learn more about API authentication →
Viewing test API keys. Toggle to view live keys.
Viewing test data
Standard keys
These keys will allow you to authenticate API requests. Learn more
NAME
TOKEN
LAST USED
CREATED
Publishable key
pk_test_51OADTuE2rkCMN6VCtvF0CSosbS4GWpCba
pUY6bpfvTonxLIv7nQDQt5ZIIgmCMKaEEsGdODisbY
vuehraj5ydG1X00OILYsn69
—
Nov 8
Secret key
sk_test_51OADTuE2rkCMN6VCBt3T6rQZyRn35ZLQl
lA9f0B5WmgXESGAm3EEtcjNhvDpNXCagdlflWkBlzw
Kgye1WTlcH0Tt00zSAzSlt4
Hide test key
—
Nov 8

Start An Integration

Learn about Stripe's integration choices for accepting online payments.

1. Stripe Payment Links
2. Stripe Checkout
3. Stripe Elements
4. API Only
5. Web and mobile SDKs
6. Platforms: Freshbooks, Shopify, Squarespace

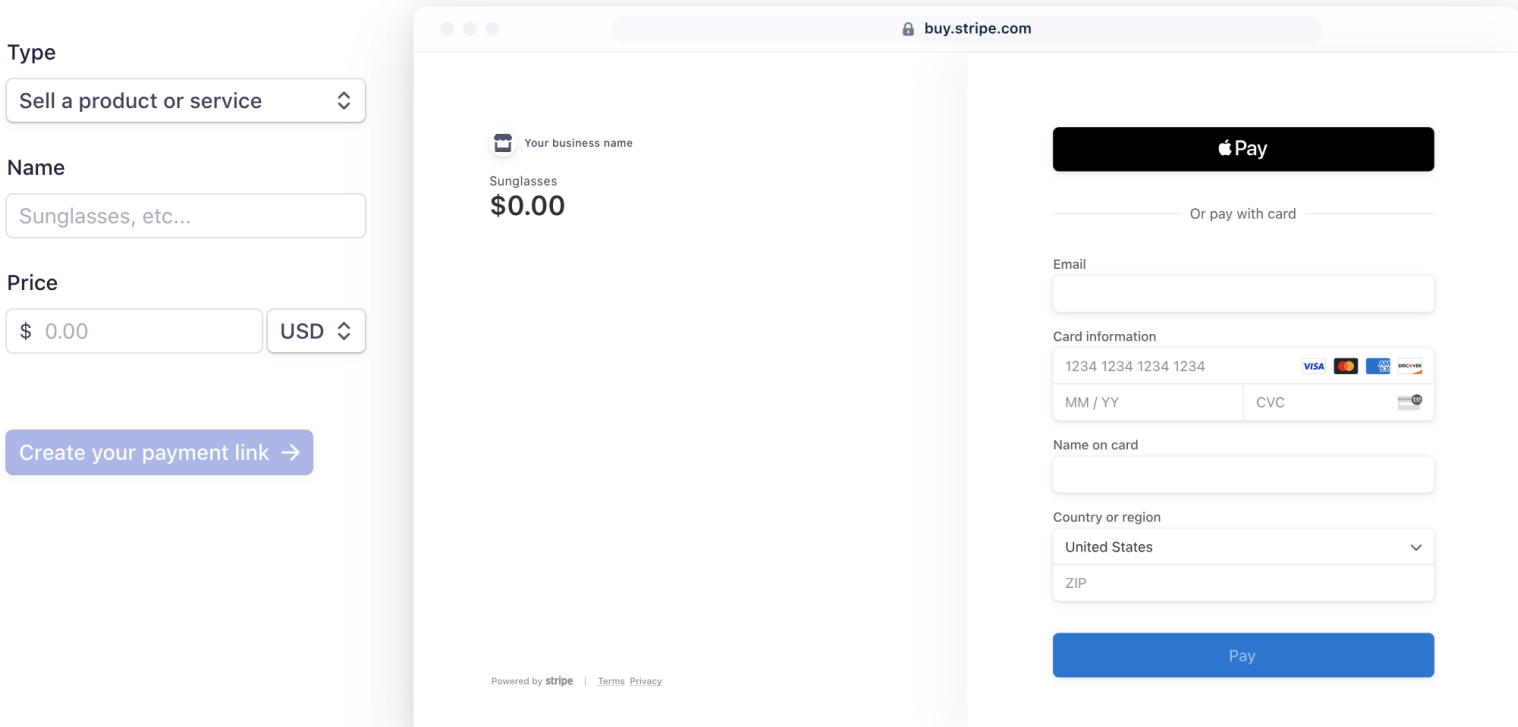
All integrations support one-time and recurring payments, fraud protection, and global payments

	PAYMENT LINKS	CHECKOUT	ELEMENTS	API ONLY
DESCRIPTION	Shareable link	Prebuilt payment form	UI components	No Stripe UI
INTEGRATION EFFORT	No coding	Low coding	More coding	Most coding
STRIPE-HOSTED PAGE	✔	✔	✕	✕
EMBED ON YOUR SITE	✕	✔	✔	✔
UI CUSTOM	Limited customization	Limited customization	Extensive customization with Appearance API	Customize full appearance and accept payments with your own UI

	PAYMENT LINKS	CHECKOUT	ELEMENTS	API ONLY
MOBILE SUPPORT	Responsive web	Responsive web	Responsive web and mobile native	Responsive web and mobile native
STRIPE TAX SUPPORT	Built-in	Built-in	Requires integration with Stripe Tax API	Requires integration with Stripe Tax API
RECURRING PAYMENTS	Built-in	Built-in	Requires integration with Subscriptions API	Requires integration with Subscriptions API

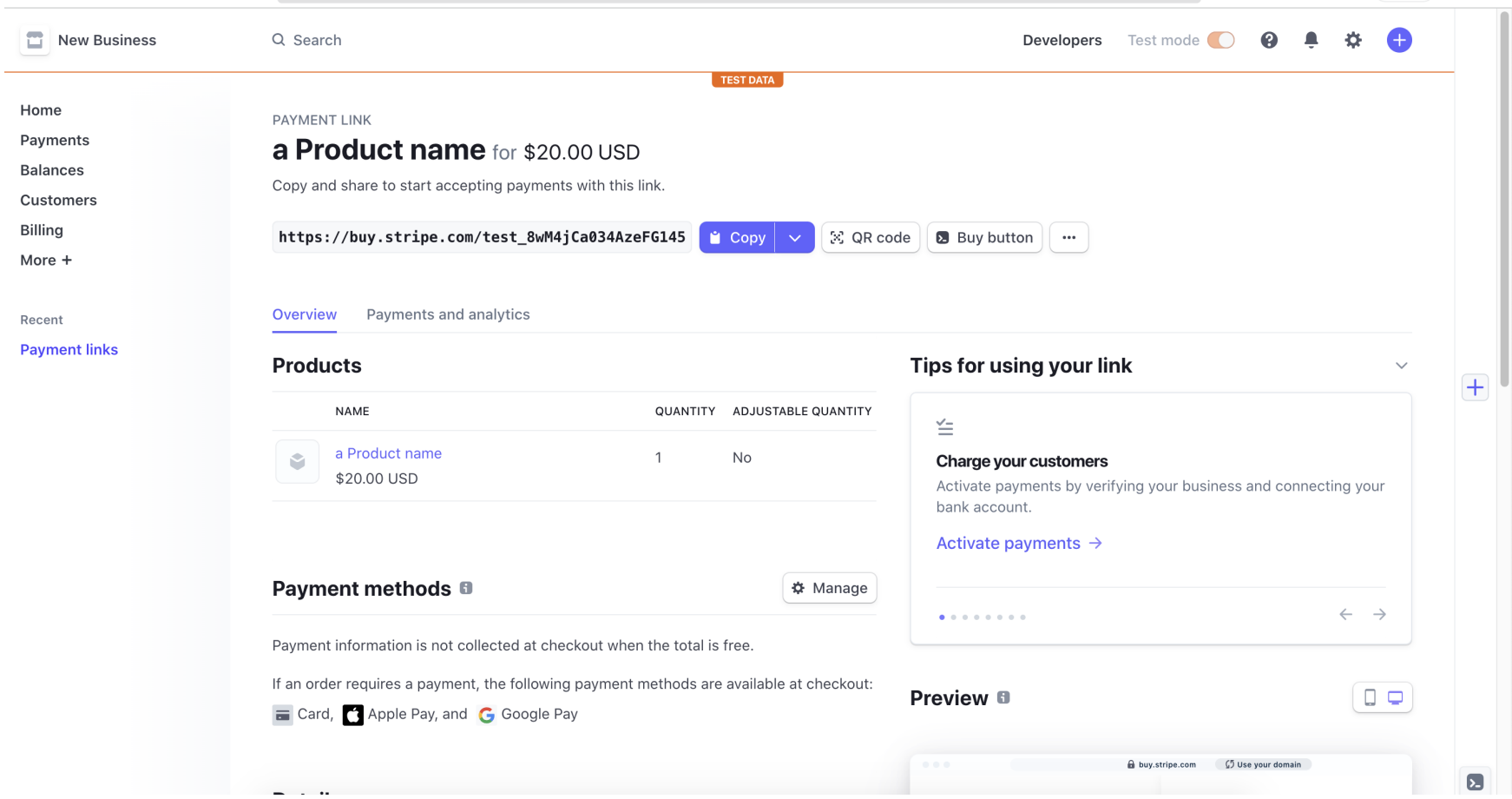
	PAYMENT LINKS	CHECKOUT	ELEMENTS	API ONLY
PAYMENT METHODS	▪ Dynamically display 40+ payment methods ▪ Manage in the Stripe Dashboard without coding	▪ Dynamically display 40+ payment methods ▪ Manage in the Stripe Dashboard without coding	▪ Dynamically display 40+ payment methods ▪ Manage in the Stripe Dashboard without coding ▪ Access to external payment methods	Integrate 50+ payment methods with code
FASTER CHECKOUT WITH LINK	Built-in	Built-in	Built-in	Unavailable
PCI COMPLIANCE HANDLING	Prefilled SAQ-A	Prefilled SAQ-A	Prefilled SAQ-A	SAQ-D required

Stripe Payment Links



Payment Links supports over 30 languages and over 20 payment methods.

Stripe Payment Links

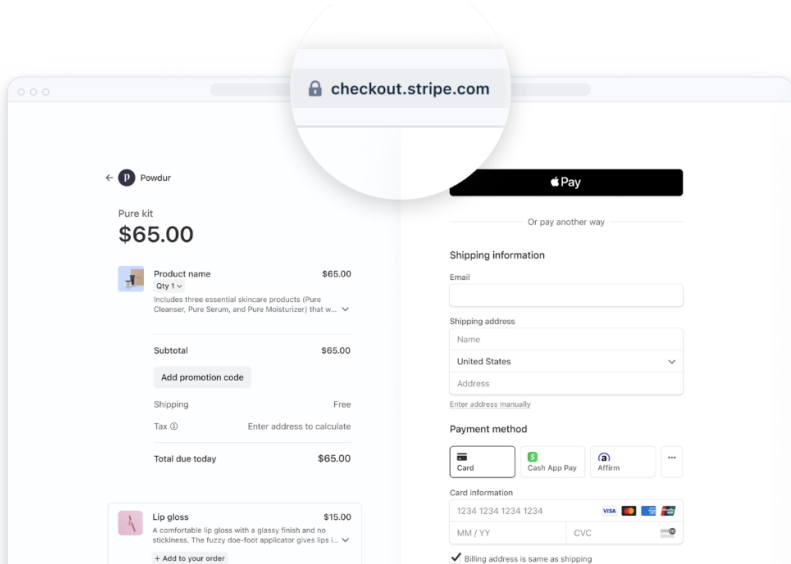


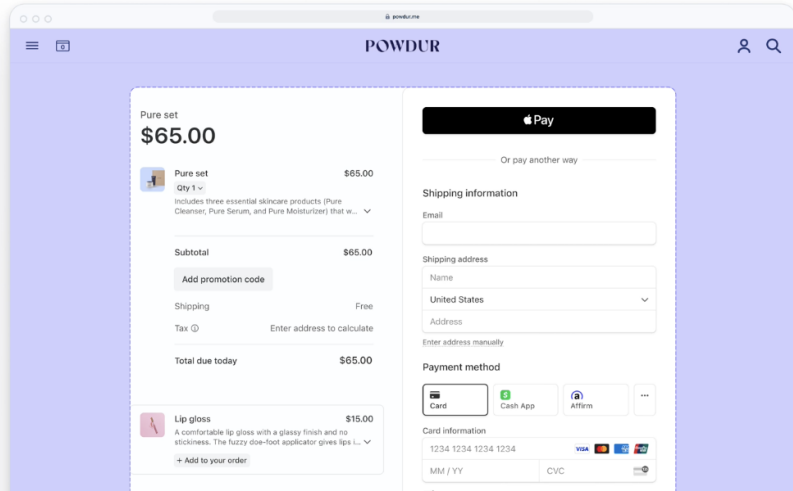
Stripe Payment Links

Demo payment link

Stripe Checkout

- Checkout is a low-code payment integration that creates a customizable form for collecting payments.
- You can embed Checkout directly in your website or redirect customers to a Stripe-hosted payment page. It supports one-time payments and subscriptions and accepts over 40 local payment methods.
- Built-in features: Support PayPal, Google Pay, Apple Pay, and Link, Responsive & Support Card Validation & Error message test

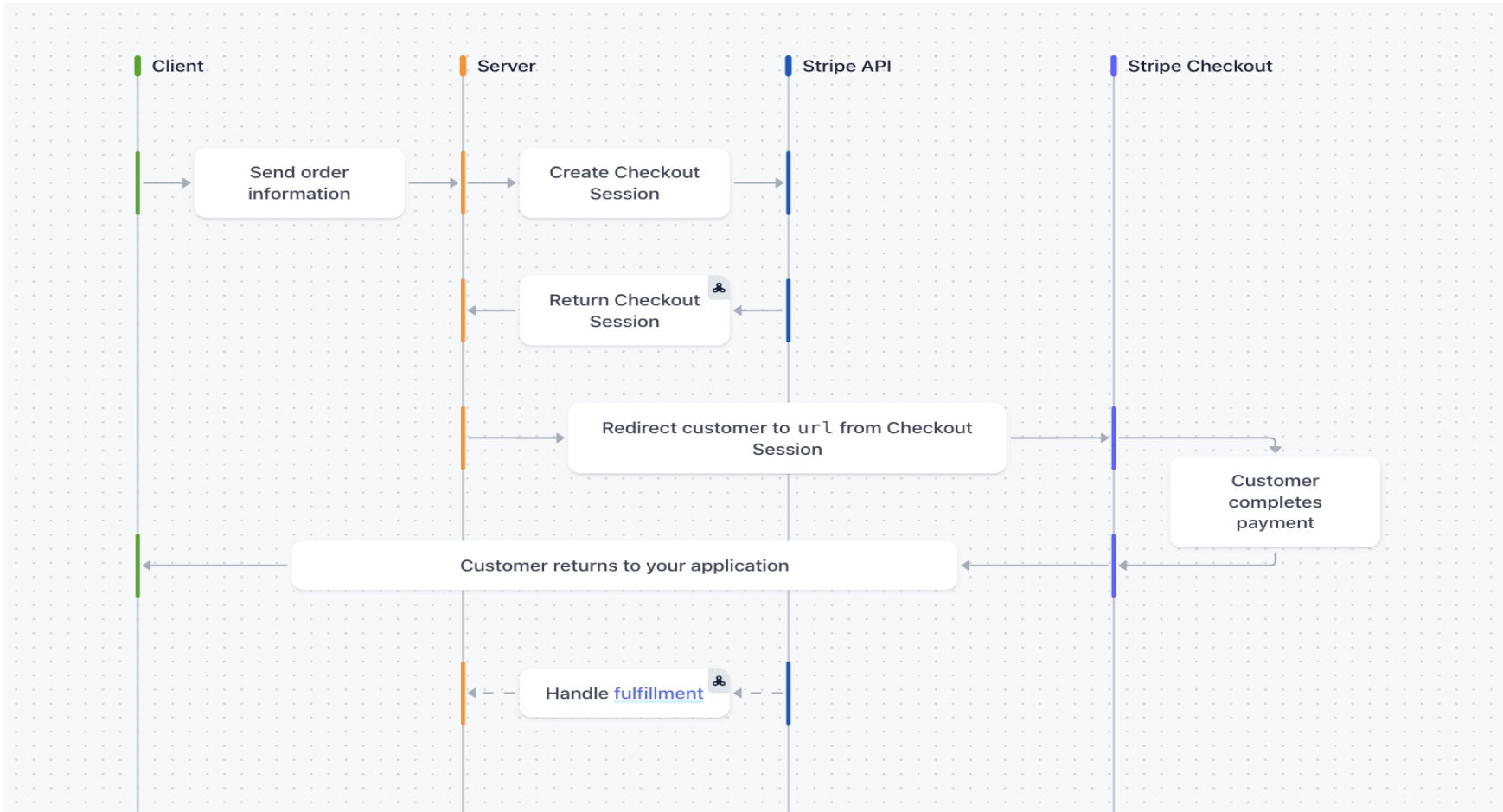




Stripe Checkout: Stripe-hosted page

1. When customers are ready to complete their purchase, your application creates a new Checkout Session.
2. The Checkout Session provides a URL that redirects customers to a Stripe-hosted payment page.
3. Customers enter their payment details on the payment page and complete the transaction.
4. After the transaction, a webhook fulfills the order using the checkout.session.completed event.

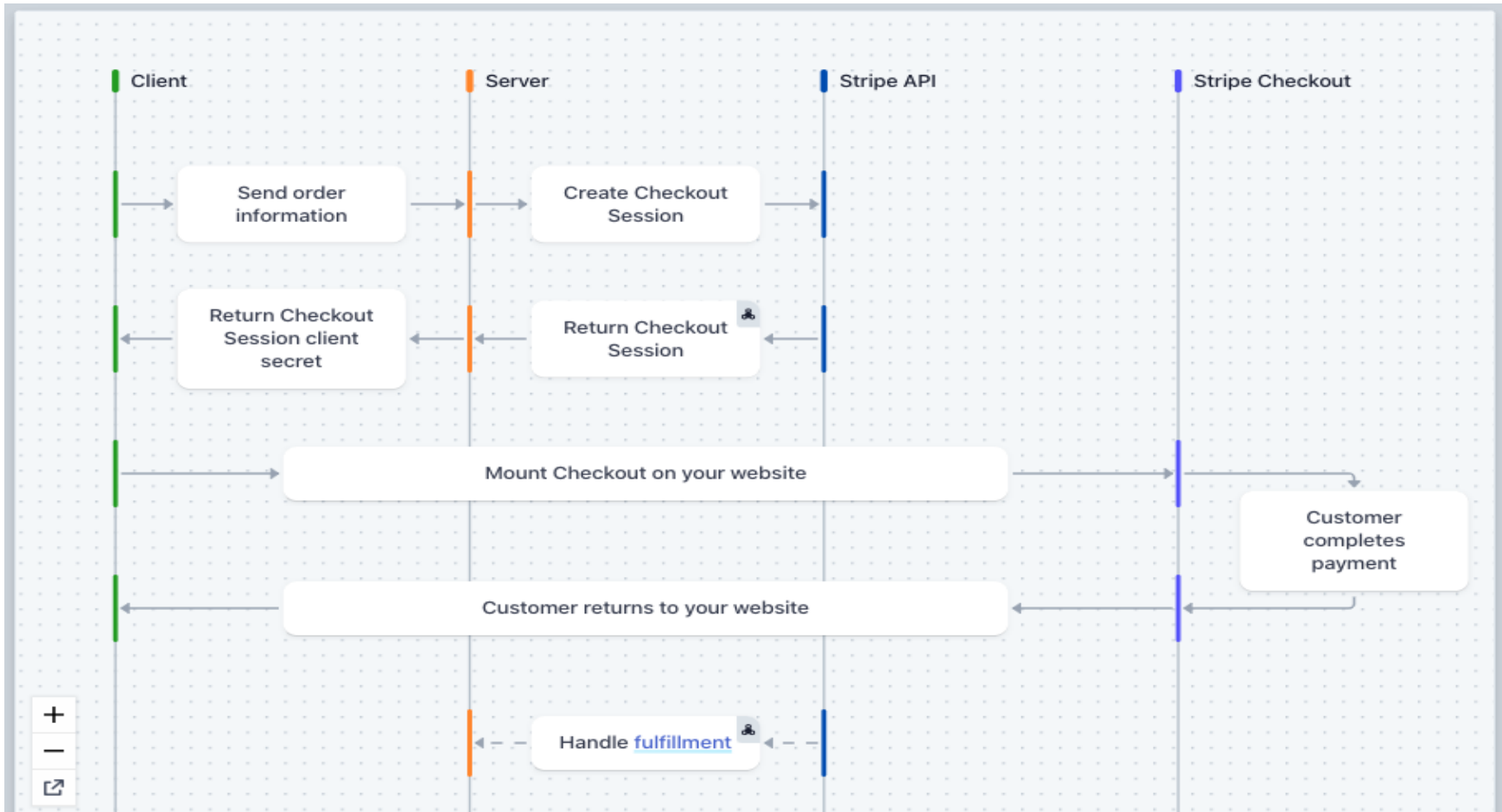
Stripe-hosted page flow



Stripe Checkout: Embedded form

1. When a customer is ready to complete their purchase, your application creates a new Checkout Session.
2. **You mount Checkout as an embeddable component on your website to show a payment form.**
3. Customers enter their payment details and complete the transaction.
4. After the transaction, the checkout.session.completed webhook event triggers the order fulfillment process.

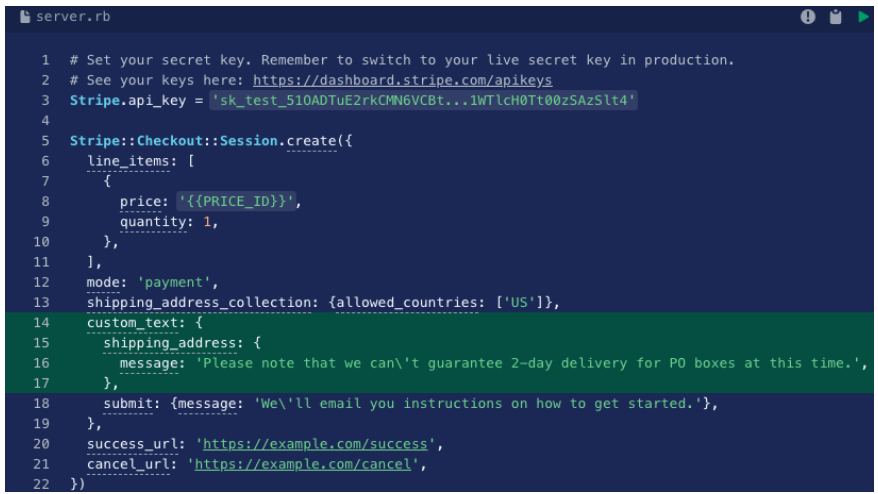
Stripe Checkout Embedded form flow



Stripe Checkout: Customize

1. Can customize branding: Logo, icon, background color, button color, font

2. Can customize text and policies

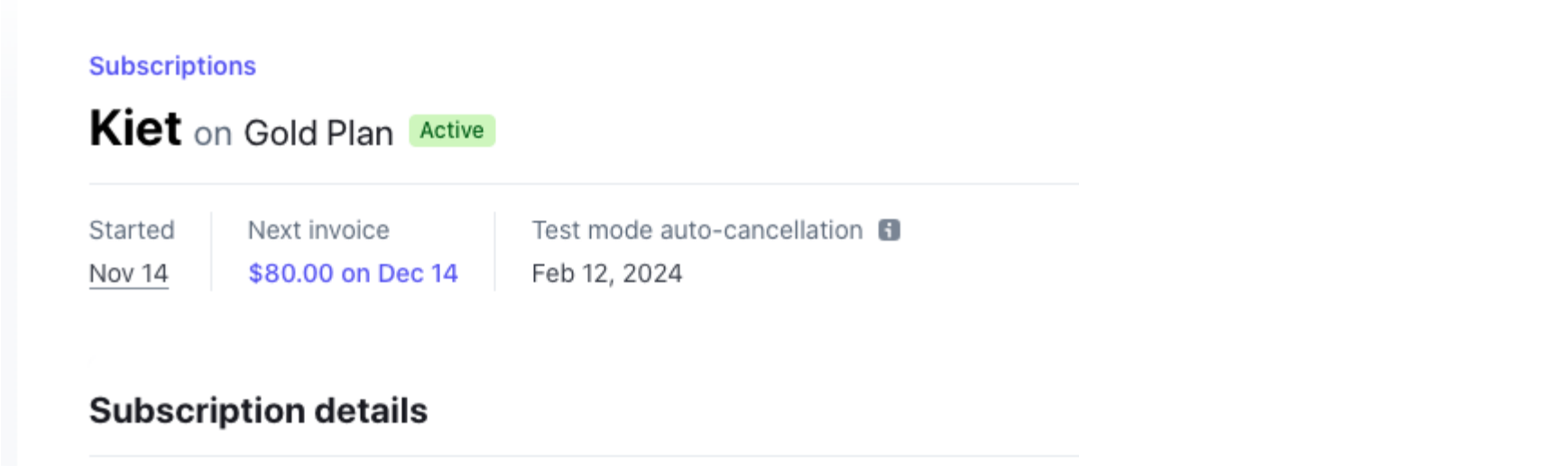


3. Can customize the Submit button

4. By default, Checkout detects the locale of the customer's browser and displays a translated version of the page in their language, if it is supported.

Stripe Checkout: Subscription overview & flow

```
session = Stripe::Checkout::Session.create({
  success_url: 'https://example.com/success.html?session_id={CHECKOUT_SESSION_ID}',
  cancel_url: 'https://example.com/canceled.html',
  mode: 'subscription',
  line_items: [{
    # For metered billing, do not pass quantity
    quantity: 1,
    price: price_id,
  }],
})
```



Stripe Checkout: Subscription overview & flow

After payment for subscription success, the customer returns to your success page, a `checkout.session.completed` webhooks will be call to your server.

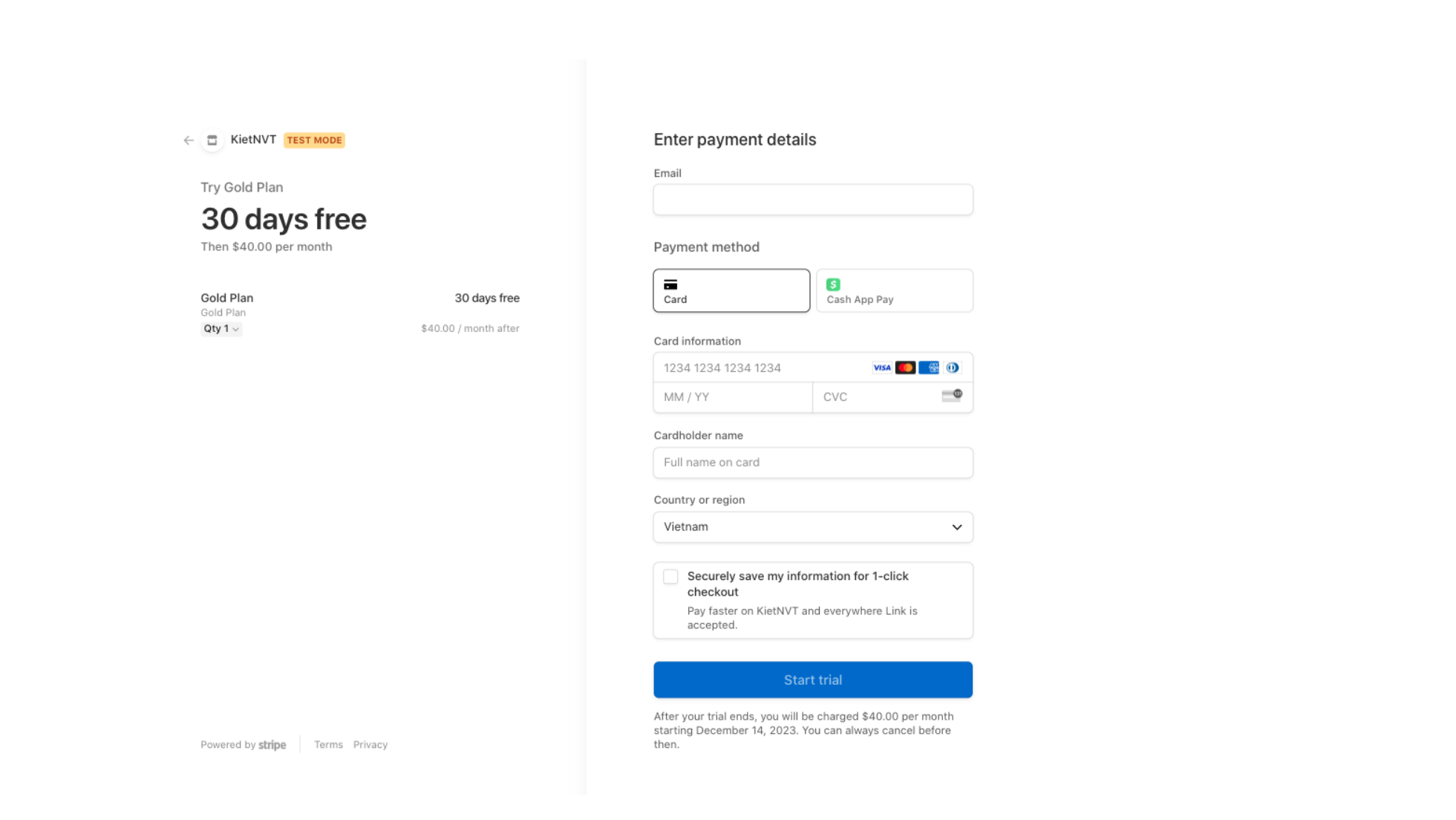
Each month (if billing monthly), `invoice.paid` webhook will be call to your server and `invoice.payment_failed` will be call when payment failed (card insufficient issue for example)

```
# Get the type of webhook event sent
event_type = event['type']
data = event['data']
data_object = data['object']

case event_type
when 'checkout.session.completed'
  # Payment is successful and the subscription is created.
  # You should provision the subscription and save the customer ID to your database.
when 'invoice.paid'
  # Continue to provision the subscription as payments continue to be made.
  # Store the status in your database and check when a user accesses your service.
  # This approach helps you avoid hitting rate limits.
when 'invoice.payment_failed'
  # The payment failed or the customer does not have a valid payment method.
  # The subscription becomes past_due. Notify your customer and send them to the
  # customer portal to update their payment information.
end
```

Stripe Checkout: Subscription overview & flow

Trials

```
Stripe::Checkout::Session.create({
  mode: 'subscription',
  success_url: 'https://example.com/success',
  cancel_url: 'https://example.com/cancel',
  line_items: [
    {
      price: '{{PRICE_ID}}',
      quantity: 1,
    },
  ],
  subscription_data: {
    trial_settings: {end_behavior: {missing_payment_method: 'cancel'}},
    trial_period_days: 30,
  },
  payment_method_collection: 'if_required',
})
```


KietNVT
TEST MODE
Try Gold Plan
30 days free
Then $40.00 per month
Gold Plan
Gold Plan
Qty 1
30 days free
$40.00 / month after
Powered by stripe
Terms
Privacy
Enter payment details
Email
Payment method
Card
Cash App Pay
Card information
1234 1234 1234 1234
MM / YY
CVC
Cardholder name
Full name on card
Country or region
Vietnam
Securely save my information for 1-click checkout
Pay faster on KietNVT and everywhere Link is accepted.
Start trial
After your trial ends, you will be charged $40.00 per month starting December 14, 2023. You can always cancel before then.

Stripe Checkout: Set the billing cycle date

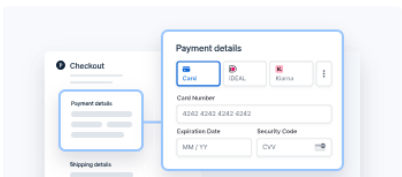
```
session = Stripe::Checkout::Session.create(
  line_items: [{
    price: '{{PRICE_ID}}',
    quantity: 1,
  }],
  mode: 'subscription',
  success_url: 'https://example.com/success?session_id={CHECKOUT_SESSION_ID}',
  cancel_url: 'https://example.com/cancel',
  subscription_data: { billing_cycle_anchor: 1701446593 }, # 01/12/2023}
)
```

Stripe Web Elements

Create your own checkout flows with prebuilt UI components.

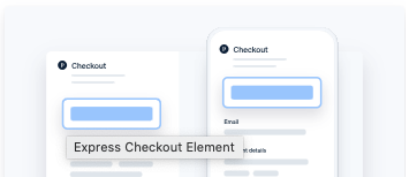
Stripe Elements is a set of prebuilt UI components for building your web checkout flow.

Stripe.js tokenizes sensitive payment details within an Element without ever having them touch your server.



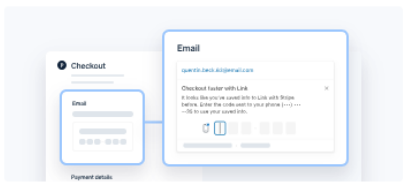
Payment Element RECOMMENDED

Accept a payment with one or multiple payment methods securely, including cards.



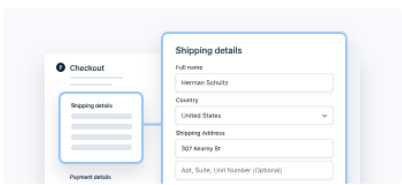
Express Checkout Element

Display popular Wallets like Apple Pay, Google Pay, and PayPal.



Link Authentication Element RECOMMENDED

Link auto-fills your customers' payment and shipping details to reduce friction and deliver an easy and secure checkout experience.



Address Element

Collect address information and display Link saved addresses.

Payment Element

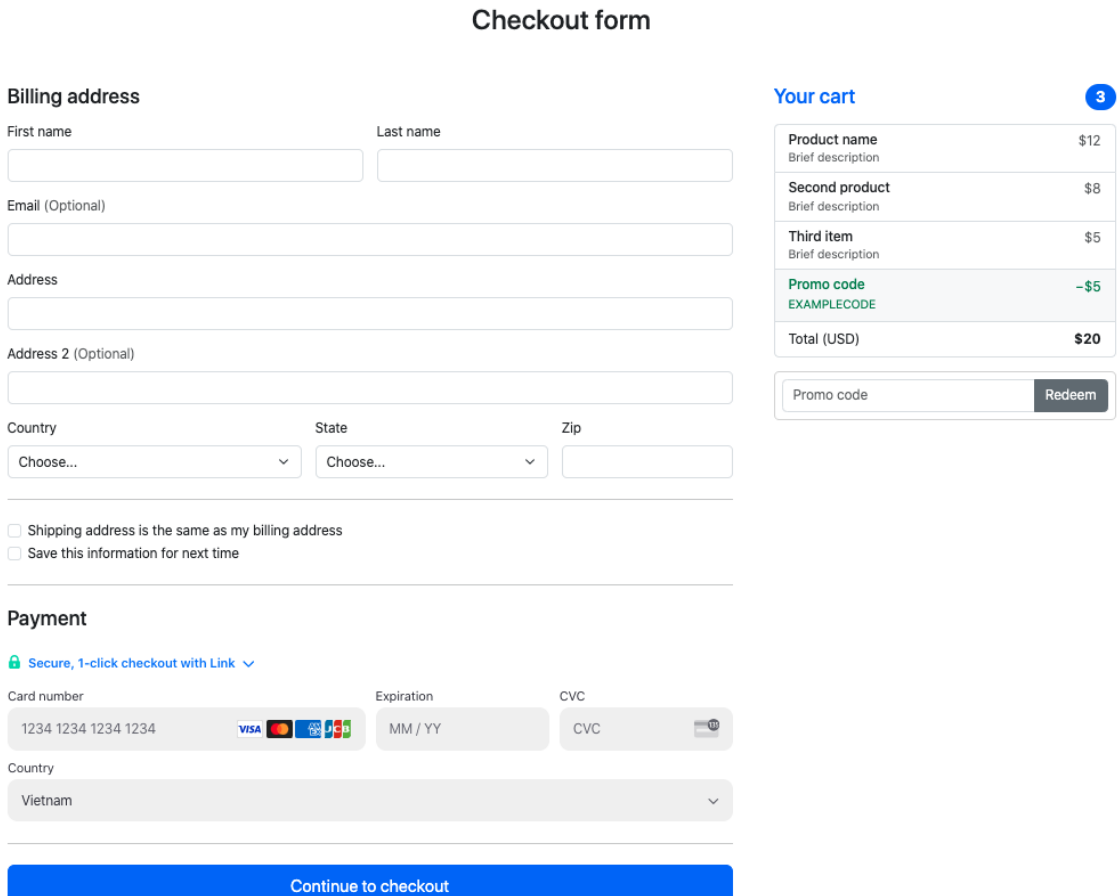
The Payment Element is a UI component for the web that accepts 40+ payment methods, validates input, and handles errors. Use it alone or with other elements in your web app's frontend.

Steps:

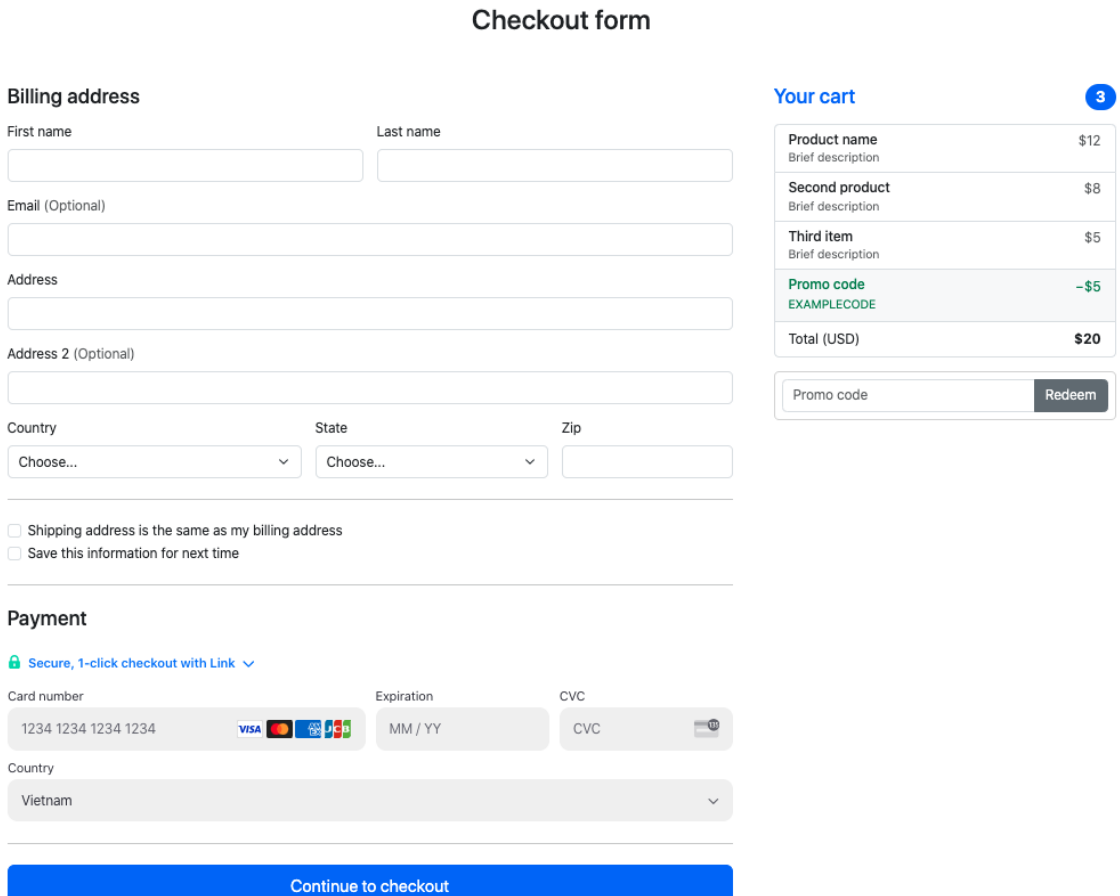
1. Create a Payment Element

```
const stripe = Stripe('pk_test_51OADTuE2rkCMN6VCtvF0CSosbS4GWpCbapUY6bpfvTonxLIv7nQDQt5ZIIgmCMKaEEsGdODisbYvuehraj5ydG1)

const appearance = { /* appearance */ };
const options = { /* options */ };
const elements = stripe.elements({ clientSecret, appearance });
const paymentElement = elements.create('payment', options);
paymentElement.mount('#payment-element');
```

Payment Element: Demo



Payment Element: Demo with other elements



Express Checkout Element

Show multiple one-click payment buttons with a single component.

The Express Checkout Element gives you a single integration for accepting payments through one-click payment buttons. Supported payment methods include Link, Apple Pay, Google Pay, and PayPal.

With this integration, you can:

- Dynamically sort payment buttons based on a customer’s location.
- Add payment buttons without any frontend changes.
- Integrate Elements seamlessly by reusing an existing Elements instance to save time.

Express Checkout Element: Demo

Prerequisites

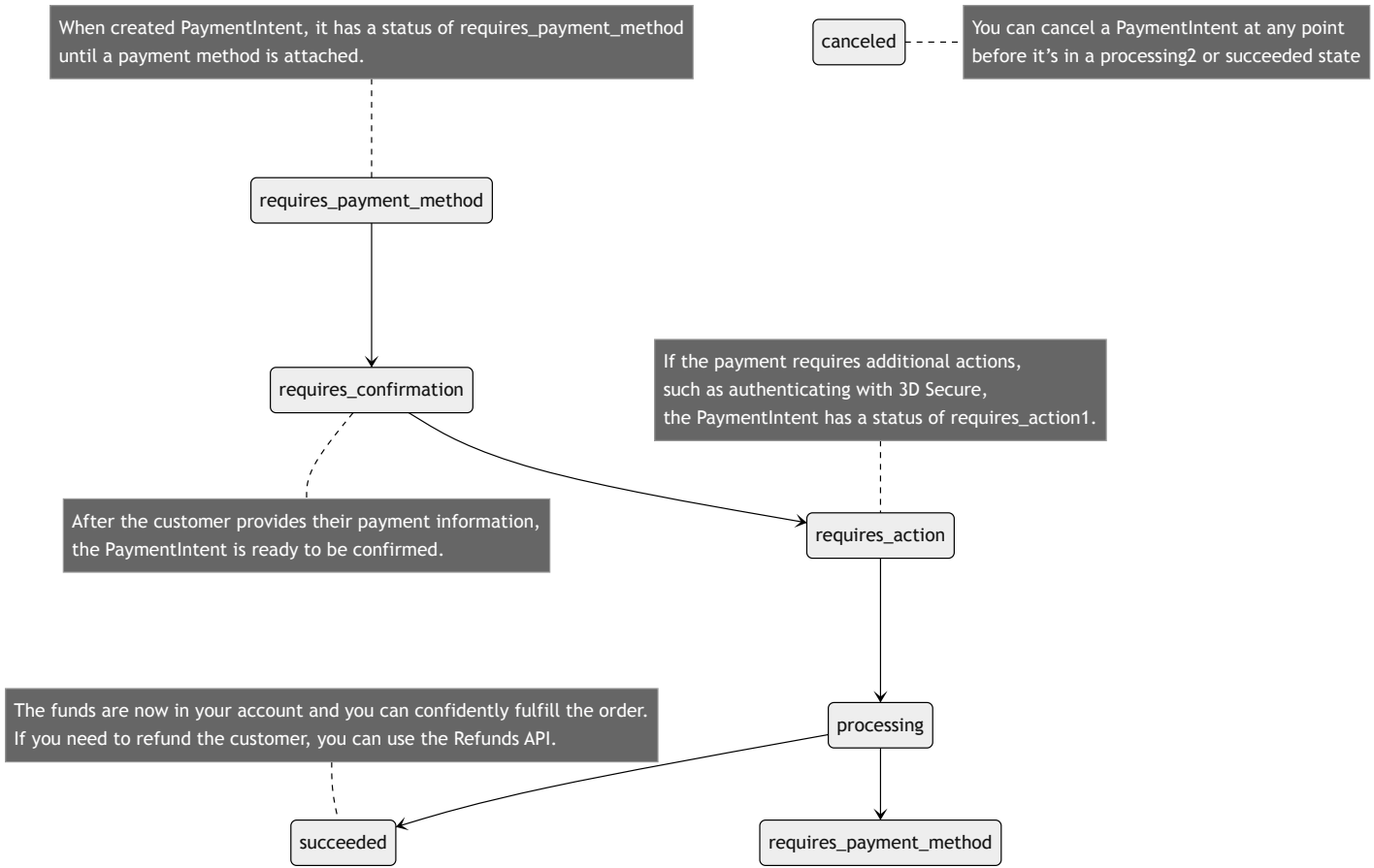
Before you start, you must:

- Add a payment method to your browser. For example, you can add a card to your Google Pay account or to your Wallet for Safari.
- Serve your application over HTTPS. This is required in development and in production. You can use a service such as ngrok.
- Register and verify your domain in both test mode and live mode.
- Create a PayPal Sandbox account to test your integration.

Payment Intents API

Use the Payment Intents API to build an integration that can handle complex payment flows with a status that changes over the PaymentIntent's lifecycle. It tracks a payment from creation through checkout, and triggers additional authentication steps when required.

- Automatic authentication handling
- Support for Strong Customer Authentication (SCA) and similar regulatory changes
- Use the Payment Intents API together with the Setup Intents and Payment Methods APIs. These APIs help you handle dynamic payments (for example, additional authentication like 3D Secure) and prepare you for expansion to other countries while allowing you to support new regulations and regional payment methods.


When created PaymentIntent, it has a status of requires_payment_method until a payment method is attached.
canceled
You can cancel a PaymentIntent at any point before it’s in a processing2 or succeeded state
requires_payment_method
requires_confirmation
If the payment requires additional actions, such as authenticating with 3D Secure, the PaymentIntent has a status of requires_action1.
After the customer provides their payment information, the PaymentIntent is ready to be confirmed.
requires_action
processing
The funds are now in your account and you can confidently fulfill the order. If you need to refund the customer, you can use the Refunds API.
succeeded
requires_payment_method

Integrate Steps: API Payment Intents

1. Creating a PaymentIntent

```
Stripe::PaymentIntent.create({
  amount: 1099,
  currency: 'usd',
  payment_method_types: ['card'],
})
```

2. Passing the client secret to the client side, init form payment => add payment method and confirm payment

```
get '/secret' do
  intent = # ... Create or retrieve the PaymentIntent
  {client_secret: intent.client_secret}.to_json
end
```

```
(async () => {
  const response = await fetch('/secret');
  const {client_secret: clientSecret} = await response.json();
  // Render the form using the clientSecret
})();
```

3. After the client confirms the payment, it is a best practice for your server to monitor webhooks to detect when the payment successfully completes or fails.

Card authentication and 3D Secure

For extra fraud protection, 3D Secure (3DS) requires customers to complete an additional verification step with the card issuer when paying.

It direct the customer to an authentication page on their bank’s website, and they enter a password associated with the card or a code sent to their phone

Card authentication and 3D Secure

T-shirt (Blue / M)

€2.00

Powered by stripe | Terms Privacy

Pay with card

Email

janedoe@stripe.com

Card information

5555 5555 5555 4444

01 / 25 | 123

Name on card

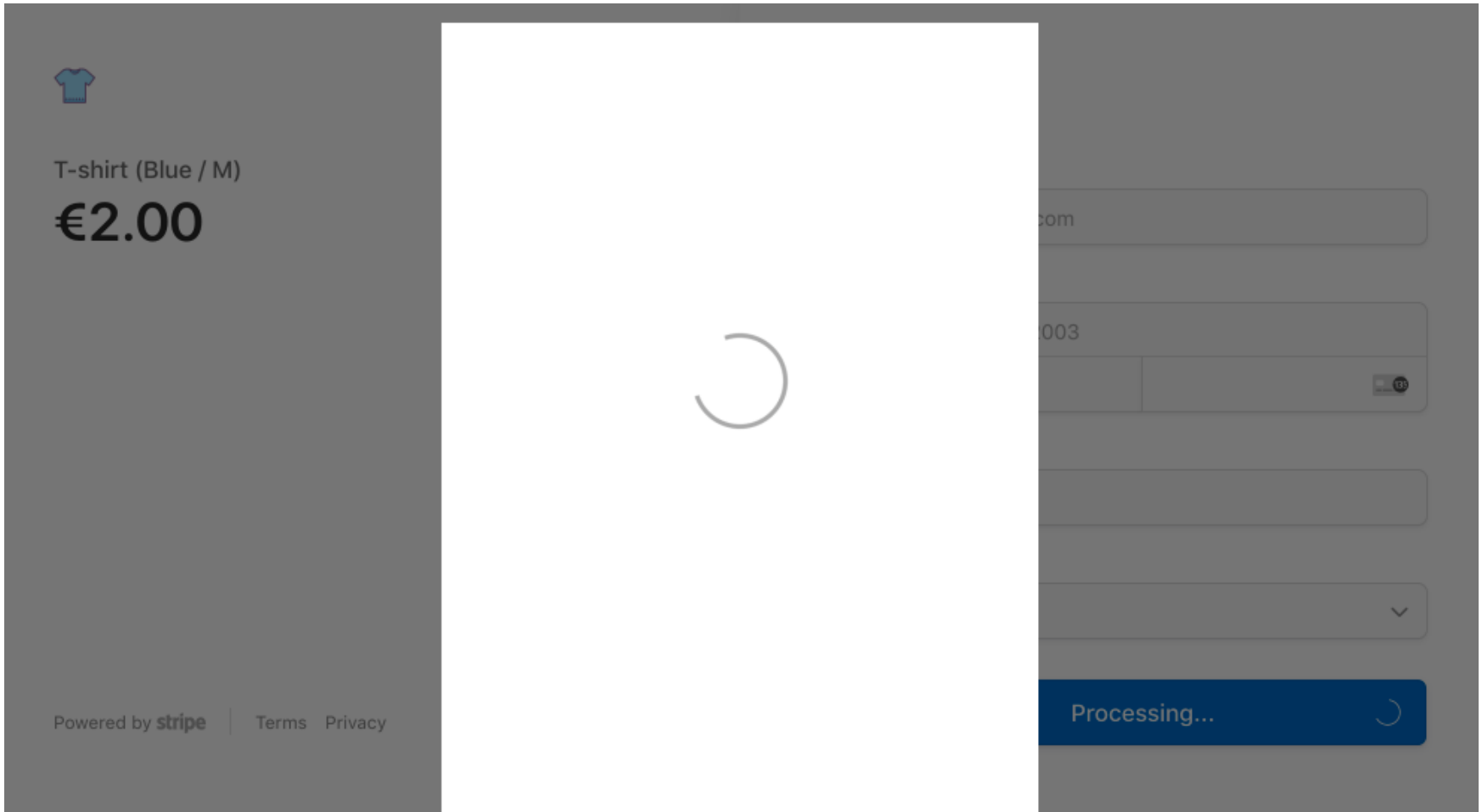
Jane Doe

Country or region

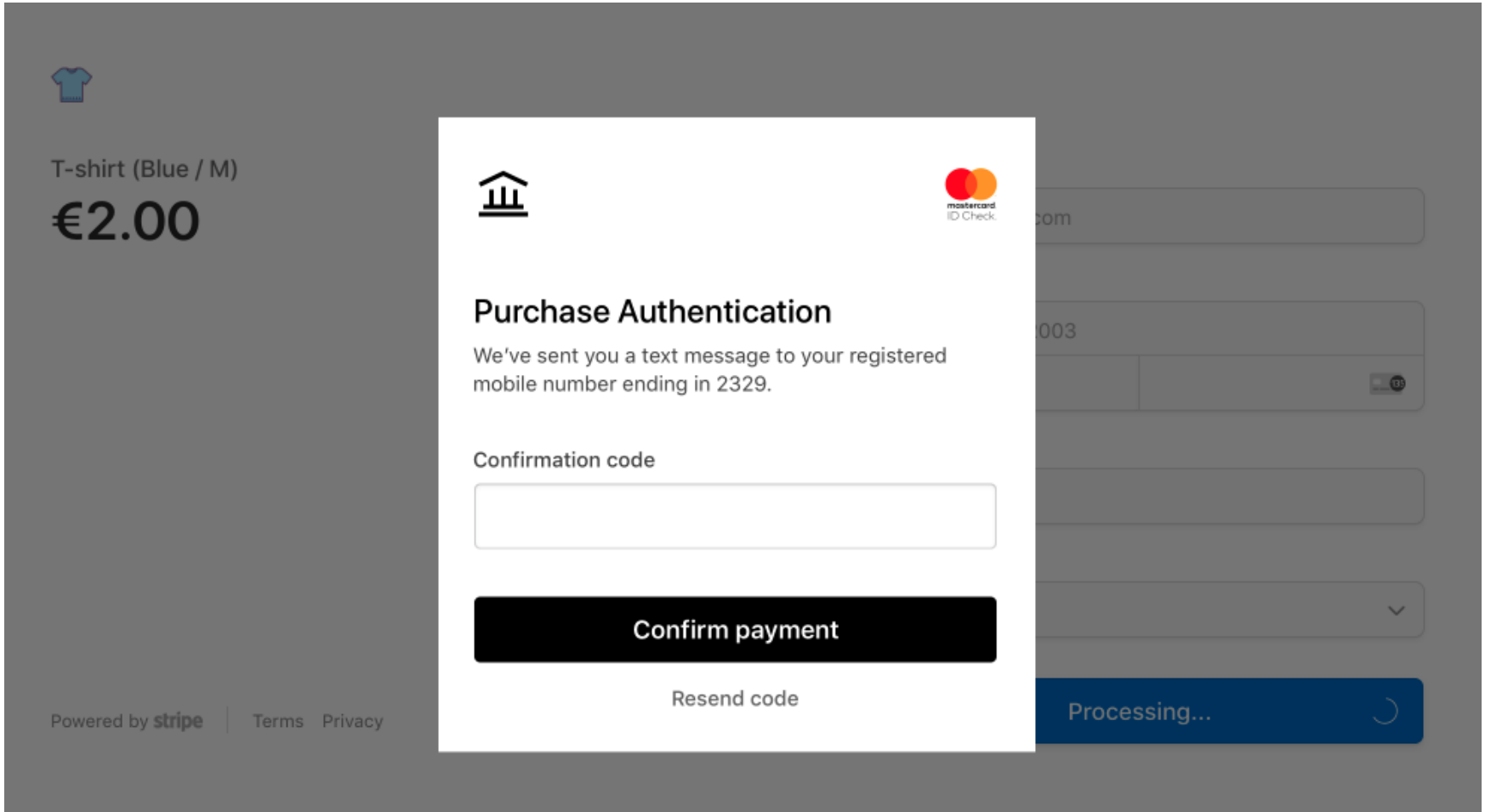
Ireland

Pay €2.00

Card authentication and 3D Secure



Card authentication and 3D Secure



Card authentication and 3D Secure

- The default method to trigger 3DS is using Radar to dynamically request 3D Secure based on risk level and other requirements.
- To trigger 3DS manually, set `payment_method_options[card][request_three_d_secure]` to any when creating or confirming a PaymentIntent or SetupIntent.
- When you set `request_three_d_secure` to `any`, Stripe requires your customer to perform authentication to complete the payment successfully if 3DS authentication is available for a card.
- If it’s not available for the given card, the payment proceeds normally.

Displaying the 3D Secure Flow

Stripe automatically displays the authentication UI in a pop-up modal when calling `confirmCardPayment` and `handleCardAction`. You can also redirect to the bank's website or use an iframe.

1. Redirect to the bank website

- To redirect your customer to the 3DS authentication page, pass a `return_url` to the `PaymentIntent` when confirming on the server or on the client.

```
stripe
  .confirmCardPayment('{PAYMENT_INTENT_CLIENT_SECRET}', {
    payment_method: {
      card: cardElement,
      billing_details: {
        name: 'Jenny Rosen',
      },
      return_url: 'https://mysite.com'
    },
  })
  .then(function(result) {
    // Handle result.error or result.paymentIntent
  });
```

Displaying the 3D Secure Flow

1. Redirect to the bank website

- After confirmation, if a PaymentIntent has a `requires_action` status, inspect the PaymentIntent's next_action. If it contains `redirect_to_url`, that means 3DS is required.

```
next_action: {
    type: 'redirect_to_url',
    redirect_to_url: {
      url: 'https://hooks.stripe.com/...',
      return_url: 'https://mysite.com'
    }
}
```

- In the browser, redirect the customer to the url in the `redirect_to_url` hash to complete authentication.

```
var action = intent.next_action;
  if (action && action.type === 'redirect_to_url') {
    window.location = action.redirect_to_url.url;
  }
```

When the customer finishes the authentication process, the redirect sends them back to the `return_url` you specified when you created or confirmed the PaymentIntent.

Displaying the 3D Secure Flow

2. Display in an iframe

- Made PaymentIntent Confirm with `return_url`
- Check PaymentIntent status, if `requires_action` status, An additional step like 3D Secure is required
- Render iframe

```
var iframe = document.createElement('iframe');
  iframe.src = paymentIntent.next_action.redirect_to_url.url;
  iframe.width = 600;
  iframe.height = 400;
  yourContainer.appendChild(iframe);
```

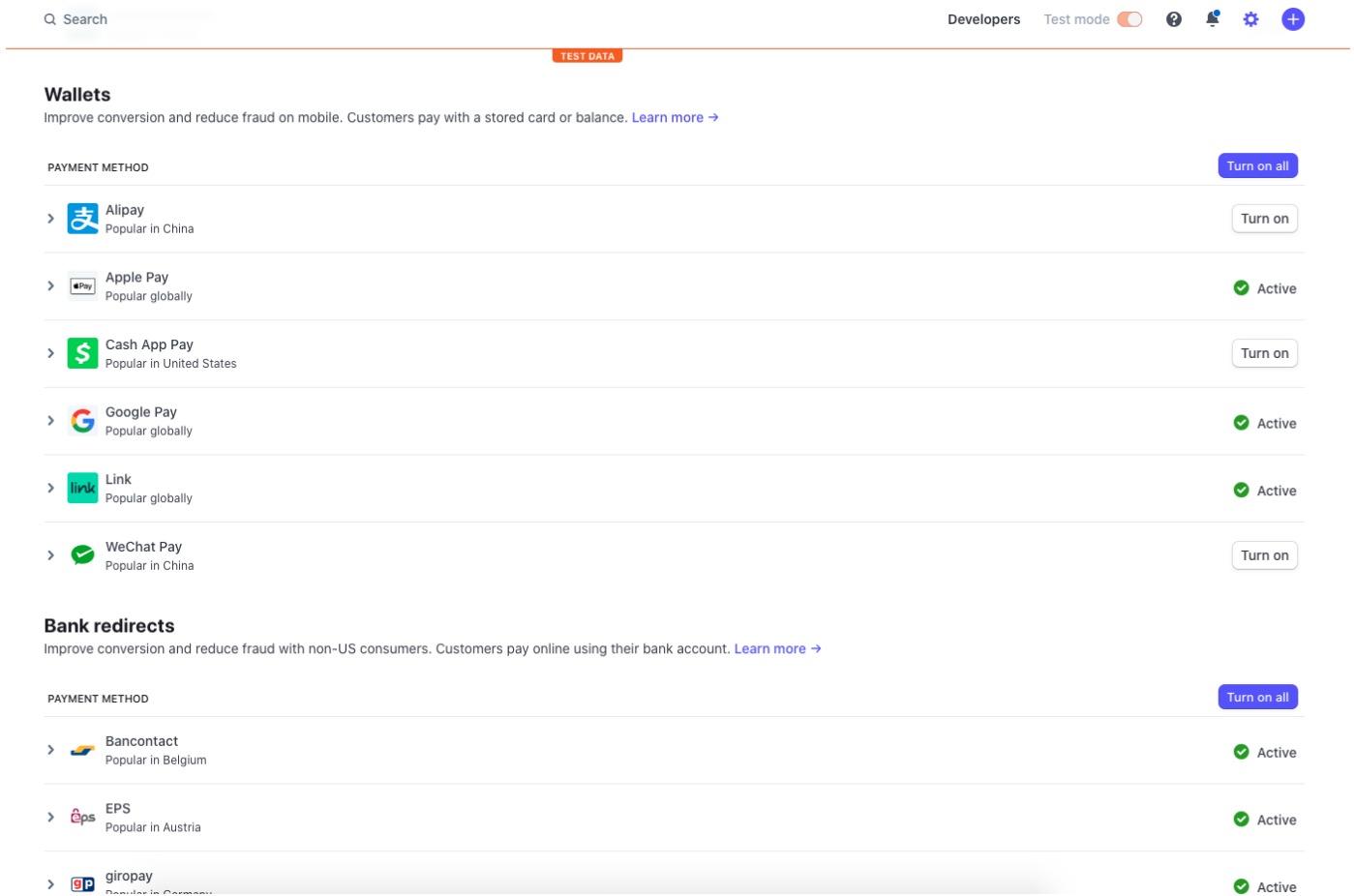
- The iframe redirects to the return_url you provided when confirming the PaymentIntent.
- On `return_url`, retrieve the updated PaymentIntent and check on the status of the payment.

Testing the 3D Secure flow

Test mode

```
| 4000000000003220 | Required  | The payment must complete 3DS2 authentication to be successful.

| 4000000000003063 | Required  | The payment must complete 3DS authentication to be successful.

| 4000008400001629 | Required  | 3DS authentication is required,
                                 but payments will be declined with a card_declined failure code after authentication.

| 4000000000003055 | Supported | 3DS authentication can still be performed, but isn’t required.

| 4242424242424242 | Supported | This card supports 3DS, but it isn’t enrolled in 3DS.
                                 This means that if your Radar rules request 3DS,
                                 the customer won’t go through additional authentication.
```

Payment Methods



Apple Pay

- Stripe users can accept Apple Pay in iOS applications in iOS 9 and above, and on the web in Safari starting with iOS 10 or macOS Sierra.
- Apple Pay is compatible with most Stripe products and features (for example, subscriptions)
- Stripe offers a variety of methods to add Apple Pay as a payment method. For integration details
- You can accept Apple Pay payments on the Web using Checkout or Elements
- No additional configuration is required to use Apple Pay in Checkout.
- Serve your application over HTTPS in development and production.
- Register and verify your domain.

checkout.stripe.com

Powdur TEST MODE

The Pure Set

US$65.00

Powered by stripe | Legal Returns Contact

Pay

Or pay another way

Email

Payment method

Card

Klarna

Card information

1234 1234 1234 1234

MM / YY

CVC

Name on card

Country or region

United States

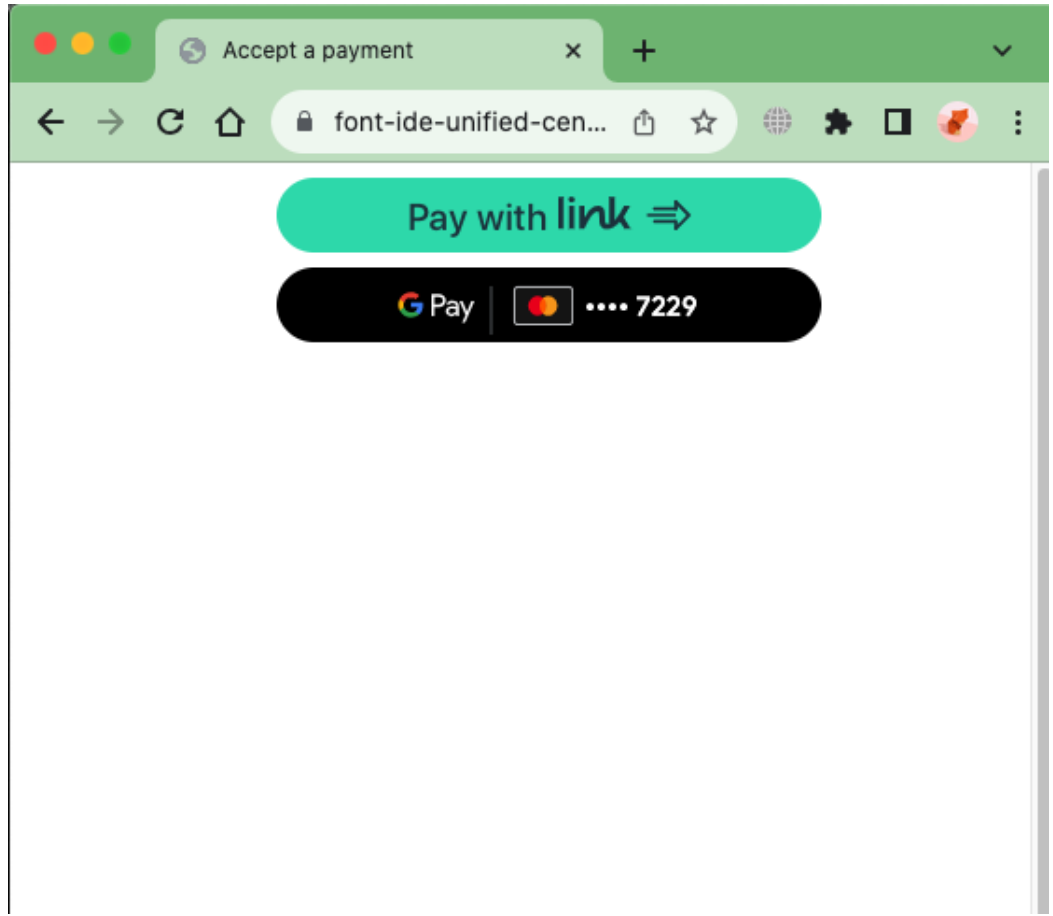
ZIP

Pay

Free returns and exchanges

Google Pay

- Google Pay allows customers to make payments in your app or website using any credit or debit card saved to their Google Account, including those from Google Play, YouTube, Chrome, or an Android device.
- You can accept Google Pay payments on the web using Checkout or Elements. Using Google Pay in Checkout requires no additional code implementation.
- Serve your application over HTTPS in development and production.
- Register and verify your domain.


Accept a payment
font-ide-unified-cen...
Pay with link
Pay
•••• 7229

Webhooks

A webhook is an HTTP endpoint that receives events from Stripe.

Webhooks allow you to be notified about payment events that happen in the real world outside of your payment flow such as:

- Successful payments (payment_intent.succeeded)
- Disputed payments (charge.dispute.created)
- Available balance in your Stripe account (balance.available)

Webhooks

Server create webhook endpoint to handle payload data from Stripe

```
class WebhookController < ActionController::Base
  skip_before_action :verify_authenticity_token

  def stripe
    data = params['data']['object']

    case params['type']
    when 'invoice.paid'
      if data['billing_reason'].freeze == 'subscription_cycle'
        StripeSubscription::Webhook::RenewalService.new(data).process
      end
    when 'customer.subscription.updated'
      StripeSubscription::Webhook::UpdatedService.new(data).process
    end

    render json: { status: 'ok' }, status: :ok
  end
end
```

Webhooks Verify events are sent from Stripe

```
# If you are testing your webhook locally with the Stripe CLI you
# can find the endpoint's secret by running `stripe listen`
# Otherwise, find your endpoint's secret in your webhook settings in
# the Developer Dashboard
endpoint_secret = 'whsec_...'
set :port, 4242

post '/my/webhook/url' do
  payload = request.body.read
  sig_header = request.env['HTTP_STRIPE_SIGNATURE']
  event = nil

  begin
    event = Stripe::Webhook.construct_event(
      payload, sig_header, endpoint_secret
    )
  rescue JSON::ParserError => e
    # Invalid payload
    puts "Error parsing payload: #{e.message}"
    status 400
    return
  rescue Stripe::SignatureVerificationError => e
    # Invalid signature
    puts "Error verifying webhook signature: #{e.message}"
    status 400
    return
  end
```


Thank you & QA