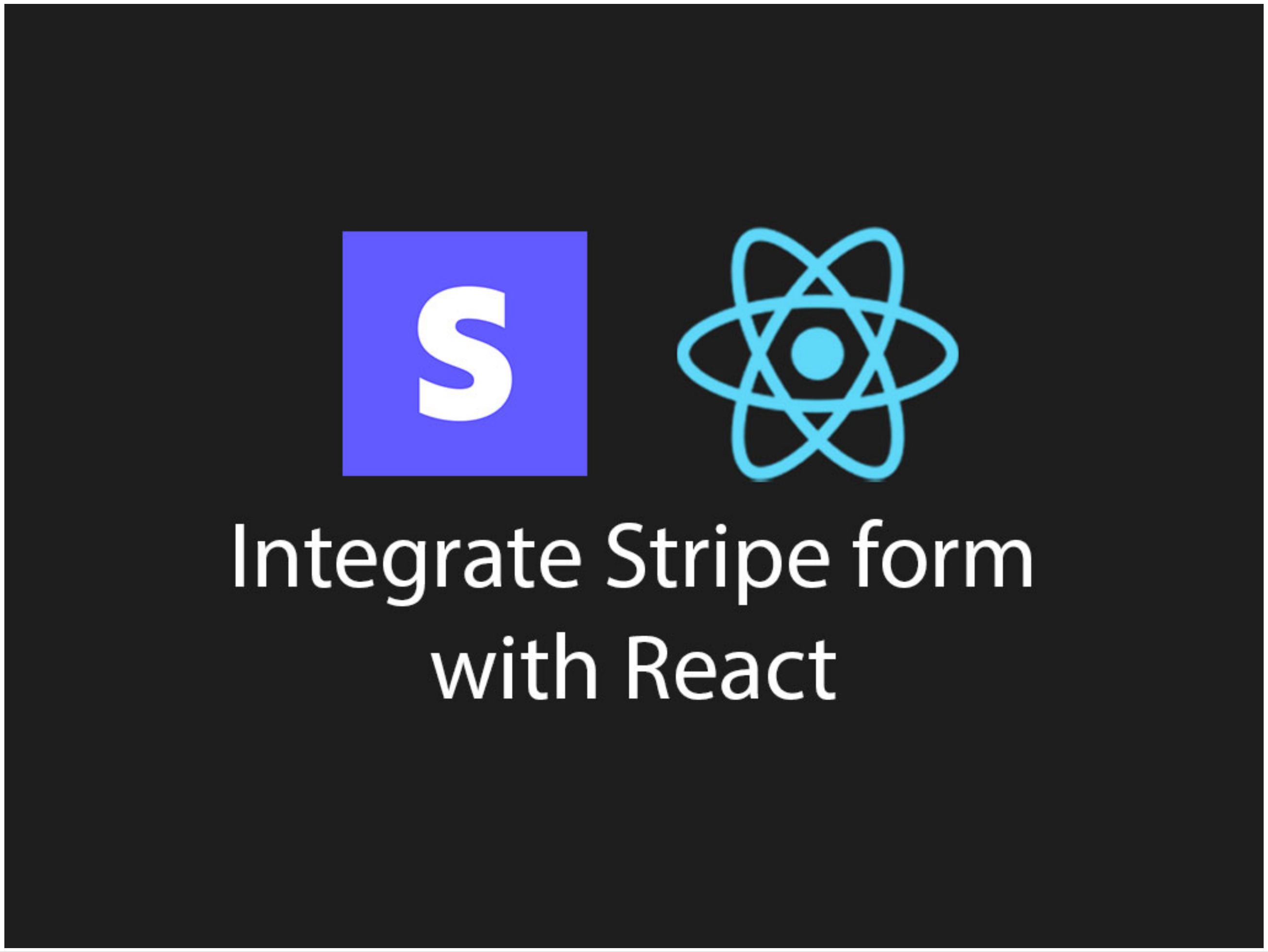
How to integrate a Stripe payment form with React in 4 easy steps



Ruben Leija

I launched this blog in 2019 and now I write to 130,000 monthly readers about JavaScript. Say hi to me at Twitter, @rleija_ (https://twitter.com/rleija_).

Using the Stripe.js library is a method to add a payment UI form to your website or application. But does Stripe integrate nicely with React?

Great news, it does! By using a combination of stripe-js and react-stripe-js libraries, you can easily create a Stripe payment form in your React application.

I will also give examples with React class components and functional components.

Install Stripe React libraries

Before we get started let's install the Stripe JS libraries onto your React application.

```
Terminal

npm install @stripe/react-stripe-js @stripe/stripe-js
```

`@stripe/stripe-js` – installs the core JavaScript utilities to make API calls to Stripe servers.

You can learn more about their stripe-js SDK here, "Stripe JavaScript SDK documentation (https://stripe.com/docs/js)".

`@stripe/react-stripe-js` – is a set of pre-built UI components to be able to make custom payment forms in your React app.

Each Stripe React component is mounted in a iframe element under the hood. This is done for security reasons.

Stripe React library core components

Now that the Stripe JavaScript libraries have been installed, let’s go over the core component that will be used.

Elements component

The `Elements component` allows you to access the Stripe JavaScript object that is coming from `@stripe/stripe-js`.

Any payment form component must nested inside `Elements`.

```
Stripe Elements component

<Elements stripe={stripePromise}>
  <StripeForm />
</Elements>
```

In the example above you may see a JavaScript variable called, `stripePromise`, being passed as a prop to the Elements component. I’ll go over that variable in step 1 of this guide.

ElementsConsumer component

In React, if you're building your payment form with a class component then you MUST use the `ElementsConsumer component`.

```
Stripe ElementsConsumer component

<Elements stripe={stripePromise}>
  <ElementsConsumer>
    {(ctx: any) => <StripeForm {...ctx} />}
  </ElementsConsumer>
</Elements>
```

When you use the `ElementsConsumer component`, the child component MUST be a function that returns your custom payment form component as shown above.

The child function will provide the Stripe and Elements JavaScript objects. You want to make sure to pass those objects down as props to your custom payment form component.

CardElement component

The `CardElement` component renders the UI for the card number, date expiration, CVC code, and the billing zip code fields.

Let's start writing some React code!

Step1: Load Stripe object

The first step to adding a payment form to your React app is to load the Stripe object.

First we start by importing the `loadStripe` utility function.

```
Load stripe object

import { loadStripe } from "@stripe/stripe-js";
```

`loadStripe()` is a function that returns a JavaScript promise.

It requires a string argument. That string value should be your publishable API key.

```
Load stripe object

const stripePromise = loadStripe("<pulishable_api_key>");
```

To access your publishable key, you can go to your API keys dashboard in Stripe (https://dashboard.stripe.com/account/apikeys).

I highly recommend that you don’t hard code your key in your project.

Even though the publishable key is safe to be viewed in the publics eye; try storing it as an environment variable.

Step 2: Add the Elements provider component

After you've initiated the `loadStripe()` call. Now you need to attach `stripePromise` to the prop attribute, `stripe`, found in the `Elements` provider component.

Class component example

```
Class component example

import {
  Elements,
  ElementsConsumer,
  CardElement,
} from "@stripe/react-stripe-js";

class StripePaymentForm extends React.Component {
  render() {
    return (
      <Elements stripe={stripePromise}>
      </Elements>
    );
  }
}
```

If you're going with the class component method, you will need to take this a step further by adding the `ElementsConsumer component`.

```
Class component example

class StripePaymentForm extends React.Component {
  render() {
    return (
      <Elements stripe={stripePromise}>
        <ElementsConsumer>
          {(ctx: any) => <PaymentForm {...ctx} />}
        </ElementsConsumer>
      </Elements>
    );
  }
}
```

Functional component example

```
Functional component example

import {
  Elements
} from "@stripe/react-stripe-js";

const StripePaymentForm = () => (
  <Elements stripe={stripePromise}>
    <PaymentForm {...ctx} />
  </Elements>
);
```

Step 3: Create payment form with Stripe CardElement

The next step is to create another React component that will render the payment form fields that uses the `CardElement` component.

Class component example

```
Class component example

class PaymentForm extends React.Component<any> {
  render() {
    return (
      <>
        <h1>stripe form</h1>
        <CardElement />
        <button onClick={this.handleSubmit}>Buy</button>
      </>
    );
  }
}
```

Functional component example

In the class component example we imported the `ElementsConsumer` component to get the Stripe API object, and the elements object.

In the functional component will use the hook utilities that Stripe provides.

Functional component example

```
import {
  Elements,
  CardElement,
  useElements,
  useStripe
} from "@stripe/react-stripe-js";

const PaymentForm = () => {
  const stripe = useStripe();
  const elements = useElements();

  return (
    <>
      <h1>stripe form</h1>
      <CardElement />
      <button onClick={handleSubmit(stripe, elements)}>Buy</button>
    </>
  );
}
```

`useStripe()` is a hook that allows a functional component to access the Stripe object.

`useElements()` is a hook that allows the developer to access the mounted `Elements`.

Step 4: Confirm card payment

This is the final stretch for the React portion of a Stripe payment form.

The objective here is to get the fields from the Stripe UI elements and create a payment form object that can be safely passed to a server API to process the payment intent portion.

Class component example

```
Class component example

class PaymentForm extends React.Component<any> {

  handleSubmit = async () => {
    const { elements, stripe } = this.props;
    const cardElement = elements.getElement(CardElement);

    const {error, paymentMethod} = await stripe.createPaymentMethod({
      type: 'card',
      card: cardElement,
    });

    if (error) {
      console.log('error:', error);
    } else {
      console.log('Payment method:', paymentMethod);
      // ... POST: /api/charge/user
    }
  };

  // ...Render method

}
```

Functional component example

```
Functional component example

const handleSubmit = (stripe, elements) => async () => {
  const cardElement = elements.getElement(CardElement);

  const {error, paymentMethod} = await stripe.createPaymentMethod({
    type: 'card',
    card: cardElement,
  });

  if (error) {
    console.log('[error]', error);
  } else {
    console.log('[PaymentMethod]', paymentMethod);
     // ... POST: /api/charge/user
  }
};

// ...PaymentForm component
```

Code breakdown

The first step in handler function, `handleSubmit`, is to get the reference card elements.

```
Get card element

const cardElement = elements.getElement(CardElement);
```

The next step is to call `stripe.createPaymentMethod()` to create a payment information object that can be safely passed to a server API.

```
stripe.createPaymentMethod

const {error, paymentMethod} = await stripe.createPaymentMethod({
  type: 'card',
  card: cardElement,
});
```

Here’s an example of a payment information object given from Stripe payment API:

stripe.createPaymentMethod response

```
{
  "id": "pm_1Ht3jfCqiD4E1v5r3l3m0kNG",
  "object": "payment_method",
  "billing_details": {
    "address": {
      "city": "Anytown",
      "country": "US",
      "line1": "1234 Main street",
      "line2": null,
      "postal_code": "123456",
      "state": null
    },
    "email": "jenny@example.com",
    "name": null,
    "phone": "+15555555555"
  },
  "card": {
    "brand": "visa",
    "checks": {
      "address_line1_check": null,
      "address_postal_code_check": null,
      "cvc_check": "pass"
    },
    "country": "US",
    "exp_month": 8,
    "exp_year": 2021,
    "fingerprint": "rHDKgLHVDcTqIKV7",
    "funding": "credit",
    "generated_from": null,
    "last4": "4242",
      "networks": {
        "available": [
          "visa"
        ],
        "preferred": null
      },
      "three_d_secure_usage": {
        "supported": true
      },
      "wallet": null
    },
    "created": 123456789,
    "customer": null,
    "livemode": false,
    "metadata": {
      "order_id": "123456789"
    },
    "type": "card"
  }
```

The response above is safe to pass to your API server and process the payment intent portion.

Full source code

Class component Stripe payment form

React Stripe payment form

```
import {
  Elements,
  ElementsConsumer,
  CardElement,
  useElements,
  useStripe
} from "@stripe/react-stripe-js";
import { loadStripe } from "@stripe/stripe-js";

const stripePromise = loadStripe("pk_test_JJ1eMdKN0Hp4UFJ6kWXWO4ix00jtX

class PaymentForm extends React.Component<any> {
  handleSubmit = async () => {
    const { elements, stripe } = this.props;
    const cardElement = elements.getElement(CardElement);

    const {error, paymentMethod} = await stripe.createPaymentMethod({
      type: 'card',
      card: cardElement,
    });

    if (error) {
      console.log('[error]', error);
    } else {
      console.log('[PaymentMethod]', paymentMethod);
      // ... SEND to your API server to process payment intent
    }
  };

  render() {
    return (
      <>
        <h1>stripe form</h1>
        <CardElement />
        <button onClick={this.handleSubmit}>Buy</button>
      </>
    );
  }
}

export class StripePaymentForm extends React.Component {
  render() {
    return (
      <Elements stripe={stripePromise}>
        <ElementsConsumer>
          {(ctx: any) => <PaymentForm {...ctx} />}
        </ElementsConsumer>
      </Elements>
    );
  }
}
```

Functional component Stripe payment form

React Stripe payment form

```
import {
  Elements,
  CardElement,
  useElements,
  useStripe
} from "@stripe/react-stripe-js";
import { loadStripe } from "@stripe/stripe-js";

const stripePromise = loadStripe("pk_test_JJ1eMdKN0Hp4UFJ6kWXWO4ix00jtX

const handleSubmit = (stripe, elements) => async () => {
  const cardElement = elements.getElement(CardElement);

  const {error, paymentMethod} = await stripe.createPaymentMethod({
    type: 'card',
    card: cardElement,
  });

  if (error) {
    console.log('[error]', error);
  } else {
    console.log('[PaymentMethod]', paymentMethod);
    // ... SEND to your API server to process payment intent
  }
};

const PaymentForm = () => {
  const stripe = useStripe();
  const elements = useElements();
  return (
    <>
      <h1>stripe form</h1>
      <CardElement />
      <button onClick={handleSubmit(stripe, elements)}>Buy</button>
    </>
  );
}

const StripePaymentForm  = () => (
  <Elements stripe={stripePromise}>
    <PaymentForm />
  </Elements>
);
```

I like to tweet about React and post helpful code snippets. Follow me there (https://twitter.com/rleija_) if you would like some too!

2F%2Flinguinecode.com%2Fpost%2Fintegrate-stripe-payment-form-with-

20Stripe%20payment%20form%20with%20React%20in%204%20easy%20steps&via=rleija_)

-with-react)