Functional Perimeter

Accept payment methods from all over the world with an integrated and secure user interface component.

The Payment Element is a user interface component for the Web that accepts more than one payment method, validates input, and manages errors. Use it alone or with other front-end elements of your Web application.

How It Works

Installation

Installation via our PantoJS library

PantoJS is a wrapper that simplifies loading Panto on your website. It ensures that it is only loaded once and offers TypeScript autocomplete.

Installation of Panto.js

Use npm (for example) to install the Panto.js module:

```
npm install @panto/panto-js
```

Usage

loadPanto

This function returns a promise that resolves with a newly created Panto object once Panto.js has been loaded. It takes the same parameters as those passed during the direct initialization of a Panto instance. If necessary, it will load Panto.js for you by inserting the Panto.js script tag. If you call loadPanto in a server environment, it will resolve to null.

```
import { loadPanto } from "@panto/panto-js";

const panto = await loadPanto({
  publicApiKey: "public_0b38ef2f95c09ad0aa08",
  env: "sandbox",
  sandboxInstance: "my-sandbox-instance",
});
```

- **publicApiKey**: We have placed a random API key in this example. Replace it with your actual publishable API keys to test this code via your Panto account. You can find your API keys in your

Panto Dashboard.

- **env**: The environment in which to load Panto.js. Can be sandbox or production.
- **sandboxInstance**: The name of your sandbox instance. Only required if you are using the sandbox environment.

Manual Installation

Manually add the Panto.js script tag to the of each page on your site. If an existing script tag is already present, this module will not insert a new one. When you call loadPanto, it will use the existing script tag.

```
<!-- Somewhere in your site's <head> -->
<script src="https://js.panto.io/v1" async></script>
```

1) Creating a paymentIntent (Backend)

On the server side, proceed to create a paymentIntent on the route https://api.panto.io/payment-intents

Example in PHP:

```
function createPaymentIntent($items) {
    $username = 'Your private key';
    $password = "";

    // Calculate your product totals on the back-end to avoid any manipulation
of object in front
    $calculateOrderAmount = function() use ($items) {
        $itemTotal = 0;
        foreach ($items as $item) {
            $itemTotal += $item['quantity'] * $item['price'];
        }
        return $itemTotal;
    };

    $paymentIntentBody = [
        'amount' => $calculateOrderAmount(),
        'currency' => "EUR",
        'return_url' => "https://www.my-shop.com/order/cart/checkout-onshop",
        'transfer_data' => [
            [
                'destination' => 'Your accountID',
                'amount' => $calculateOrderAmount(),
                'application_fee' => 0,
                'availability_type' => "IMMEDIATE"
            ],
        ],
    ];

    $ch = curl_init("https://api.panto.io/payment_intents");
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($paymentIntentBody));
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        "Content-Type: application/json",
        "Authorization: Basic " . base64_encode($username . ":" . $password)
    ]);

    $response = curl_exec($ch);
    curl_close($ch);

    $paymentIntent = json_decode($response, true);
    return $paymentIntent['secret_key'];
}
```

With $items:

```
// Replace with your backend product price retrieval logic

$items = [
    [
        'quantity' => 2, // Quantity of item 1
        'price' => 10.00 // Price of item 1
    ],
    [
        'quantity' => 1, // Quantity of item 2
        'price' => 20.00 // Price of item 2
    ],
        ...
];
```

2) **Create a Payment Element (Frontend)**

This code creates a Payment Element and mounts it on the DOM:

```
const panto = await loadPanto("your public key");

const appearance = {
  /* appearance */
};
const options = {
  /* options */
};
const elements = panto.elements({ clientSecret, appearance }); // Replace
clientSecret with your secret returned by the previously created paymentIntent
const paymentElement = await elements.create("payment", options);
paymentElement.mount("#payment-element");
```

(Optional) Appearance

Use the Appearance API to control the style of all elements. Choose a theme or update specific information.

For example, choose the "flat" theme and replace the main text color.

```
jsx
const panto = await loadPanto('your public key');

const appearance = {
  theme: 'panto',
  variables: {
      colorPrimary: '#587a9d',
      colorBackground: '#ff0000',
      colorText: '#515581',
    },
};
const options = { /* options */ };
const elements = panto.elements({ clientSecret, appearance }); // Replace
clientSecret with your secret returned by the previously created paymentIntent
const paymentElement = await elements.create('payment', options);
paymentElement.mount('#payment-element');
```

3) Making the Payment on the Client Side

When your client clicks on the payment button, call `confirmPayment()` with the PaymentElement and pass a return_url to indicate the page to which Panto should redirect the user once the payment is made. For payments requiring authentication, Panto directly displays the 3DSecure authentication request or redirects the client to an authentication page, depending on the payment method. Once the authentication process is finalized, the client is redirected to the `return_url`.

```
document
        .querySelector("#payment-element")!
        .addEventListener("submit", handleSubmit);

      async function handleSubmit(e: any) {
        const { error } = await panto.confirmPayment({
          elements,
          confirmParams: {
            return_url: `https://www.my-shop.com/order/success`,
          },
        });
        // Error handling
        if (error.type === "card_error" or error.type === "validation_error") {
          console.error(error.message);
        } else {
```

```
            console.error("An unexpected error occurred.");
          }
        }
```

Error Handling

In the event of an immediate error (e.g., the card is declined), Panto.js returns an error. Display this error directly to your client so they can make another payment attempt.

Displaying a Message on the Payment Status

When Panto redirects the client to the return_url, the query parameter payment_intent_client_secret is added by Panto.js. Use it to retrieve the PaymentIntent in order to determine the data to display to your client.