flutterwave

HTML checkout

Our HTML checkout option is like Flutterwave Inline, but done entirely in HTML. You create a regular HTML form containing the payment details. When the customer submits the form, they'll be redirected to our payment page where they can complete the payment.

An example

Here's an example of how you'd implement HTML checkout:

HTML CSS Result EDIT ON

```html
<form method="POST" action="https://checkout.flutterwave.com/v3/hosted/pay">
  <div>
    Your order is N2,000
  </div>
  <input type="hidden" name="public_key" value="FLWPUBK_TEST-
02b9b5fc6406bd4a41c3ff141cc45e93-X" />
  <input type="hidden" name="customer[email]" value="test@mailnator.com" />
  <input type="hidden" name="customer[name]" value="Ayomide Jimi-Oni" />
  <input type="hidden" name="tx_ref" value="txref-81123" />
  <input type="hidden" name="amount" value="2000" />
  <input type="hidden" name="currency" value="NGN" />
  <input type="hidden" name="meta[source]" value="docs-html-test" />
  <br>
  <button type="submit" id="start-payment-button">Pay Now</button>
</form>
```

Resources

Try it out 😀

You can actually try out the payment above. Use the card number `4187427415564246` with CVV `828` and expiry `09/32`, or grab a test card or bank account from our Testing helpers page and try making the payment. It works!

Walkthrough

Let's take a closer look at what this code is doing.

First, we create a regular HTML form. The form must have the `method` as `POST`, and the `action` pointing to Flutterwave's checkout page.

```
<form method="POST" action="https://checkout.flutterwave.com/v3/hosted/pay">
```

Next up is the payment button. This is the button the customer clicks after they've reviewed their order and are ready to pay you. Make sure it's inside the form and set to `type="submit"` so the form submits when it's clicked.

```
<form method="POST" action="https://checkout.flutterwave.com/v3/hosted/pay">
  <button type="submit">Pay Now</button>
</form>
```

Finally, we add **hidden** input fields to the form containing the payment options. These payment options are the same values used in the Inline and Standard flows, converted into form fields. Object fields are referenced with square brackets.

For example, a payload like this:

```
{
  "public_key": "FLWPUBK_TEST-SANDBOXDEMOKEY-X",
  "tx_ref": "bitethtx-019203",
  "amount": 3400,
  "currency": "NGN",
  "payment_options": "card, ussd",
    "redirect_url": "https://demoredirect.localhost.me/",
    "meta": {
      "token": 54
    },
    "customer": {
      "name": "Jesse Pinkman",
      "email": "jessepinkman@walterwhite.org"
    }
  }
```

becomes this:

```
<input type="hidden" name="public_key" value="FLWPUBK_TEST-SANDBOXDEMOKEY-X" /
<input type="hidden" name="tx_ref" value="bitethtx-019203" />
<input type="hidden" name="amount" value="3400" />
<input type="hidden" name="currency" value="NGN" />
<input type="hidden" name="redirect_url" value="https://demoredirect.localhost.me/"
<input type="hidden" name="meta[token]" value="54" />
<input type="hidden" name="customer[name]" value="Jesse Pinkman" />
<input type="hidden" name="customer[email]" value="jessepinkman@walterwhite.org" />
```

That's it for HTML checkout. When the customer submits the form, they'll be redirected to the payment page.

After the payment

Four things will happen when payment is done (successful or cancelled):

1. We'll redirect to your `redirect_url` with `status` and `tx_ref` query parameters after payment is complete.
2. We'll send you a webhook if you have that enabled.
3. We'll send an email receipt to your customer if the payment was successful (unless you've disabled that).
4. We'll send you an email notification.

When we redirect to you, the `status` query parameter will be one of the following:

- `"cancelled"`: If the customer clicked "Cancel Payment".
- `"successful"`: If the payment was successful. There'll also be a `transaction_id` query parameter containing the Flutterwave transaction ID.

On your server, you should handle the redirect and **always** verify the final state of the transaction.

Transaction verification must be server-side

Transaction verification should always be done on the server, as it makes use of your secret key, which should never be exposed publicly.

Here's what transaction verification could look like in a Node.js app with our backend SDK:

Node.js

```
app.get('/payment-callback', async (req, res) => {
    if (req.query.status === 'successful') {
        const transactionDetails = await Transaction.find({ref: req.query.tx_ref});
        const response = await flw.Transaction.verify({id: req.query.transaction_id});
        if (
            response.data.status === "successful"
            && response.data.amount === transactionDetails.amount
            && response.data.currency === "NGN") {
            // Success! Confirm the customer's payment
        } else {
            // Inform the customer their payment was unsuccessful
        }
    }
);
```

What if the payment fails?

If the payment attempt fails (for instance, due to insufficient funds), you don't need to do anything. We'll keep the payment page open, so the customer can try again until the payment succeeds or they choose to cancel.

If you have webhooks enabled, we'll send you a notification for each failed payment attempt. This is useful in case you want to later reach out to customers who had issues paying.

All done! If you get stuck, you can always ask for help on our developer forum.

And if HTML checkout isn't your cup of tea, we've got you covered. Check out our Collecting Payments overview to know your options.