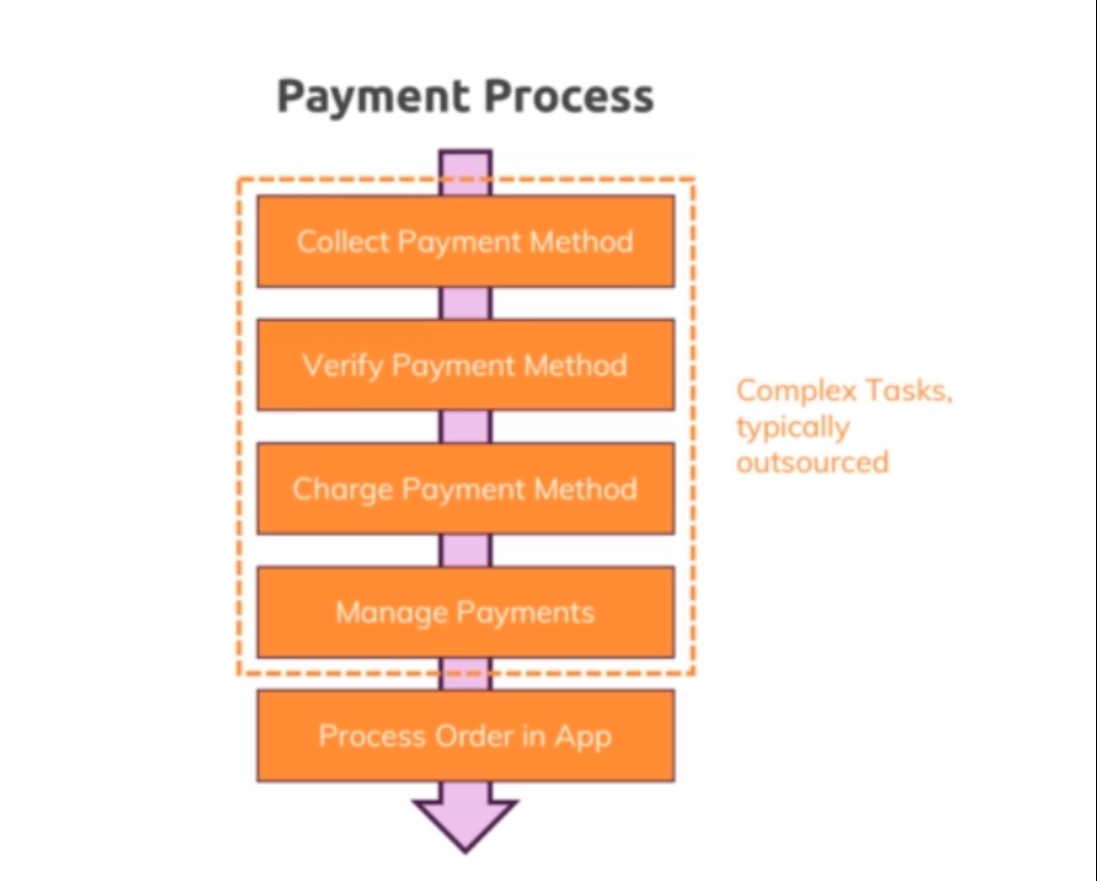
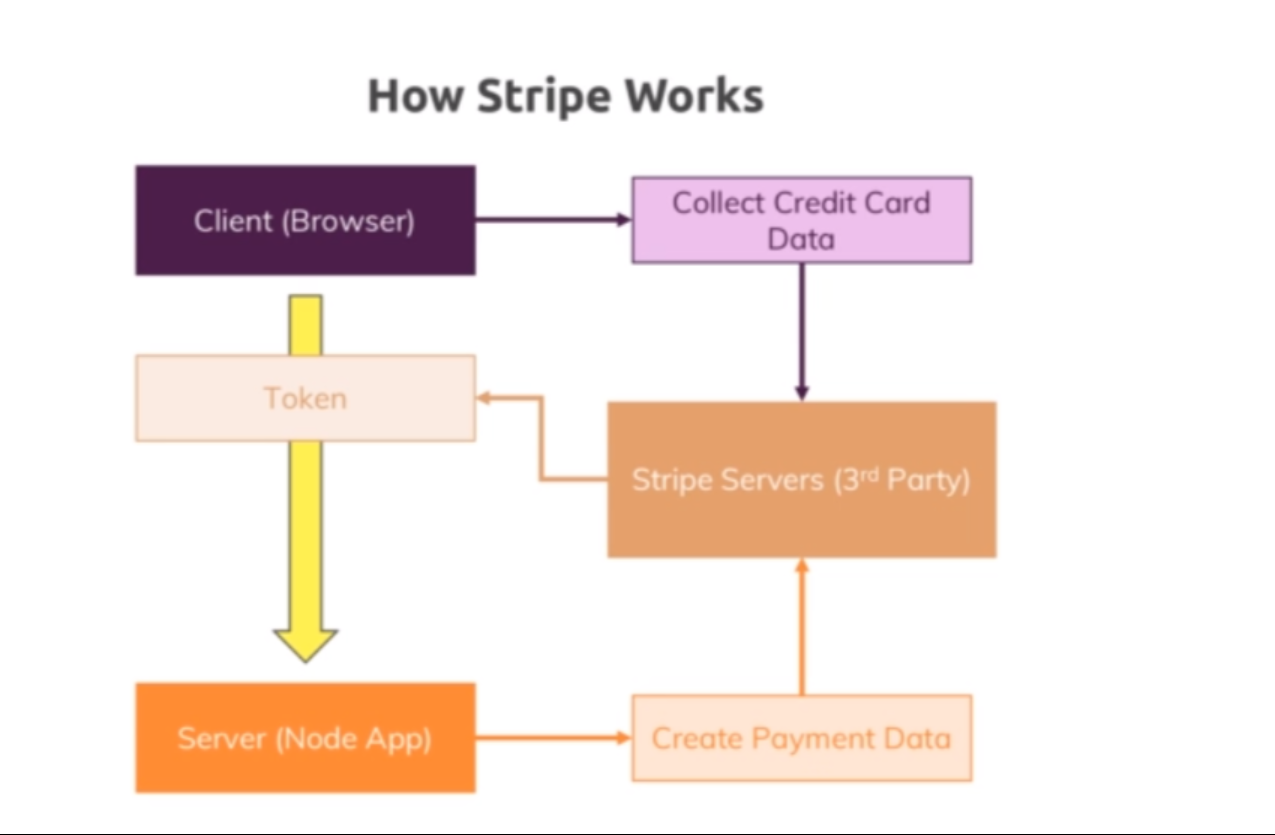
**Lecture 354**

**How payments work**

* Refer notes
* 
* 

**Lecture 355**

**Adding a checkout page**

* Refer code 01-added-checkout-page
* Checkout.ejs -🡪 added checkout page
* Shop.js routes 🡪 added route for checkout
* Shop.js controller 🡪 added getCheckout method
* Cart.ejs 🡪 removed order now form , added href link for checkout

**Lecture 356**

**Using Stripe in your app**

* Refer code 02-using-stripe
* Created stripe account
* Add a name for the account after logging in , from the top left corner.
* Npm install –save stripe
* Checkout.ejs -- > linking stripe js file
* Shop.js controller 🡪 getCheckout method, added getCheckoutSuccess method with same logic as in postOrder method

Importing stripe and using the secret key

* Shop.js routes 🡪 adding routes for /checkout/success and /checkout/cancel
* Now important under developers you'll find a bunch of API keys which you will need to add stripe and we are seeing special testing data here. Which is fine for our development if you want to build a real application and want to push it to production you would switch to your live data here. For this you have to act with your account though will not do that here will work with the test data to get started.
* So now we simply have access to the button where I now want to listen for a click so on the order button we can add an event listener a click event listener and then pass a function which should execute when that button gets clicked. Now here will not send the user to our own back into our own roots which we registered and said here will let stripe do some magic. We'll use that stripe object which we created up there and call redirect to checkout written like this now redirect to checkout takes a javascript object where we can configured is what this will do in the end is it will redirect the user to some of stripes pages where did user then enters credit card data ends on and once all of this is done and the payment is confirmed there the user is redirected back to us. Now here we have to provide a session I.D. And that's the interesting thing that should be a string but at the moment we have no such session I.D. So what do we do there. What how do we get that session I.D. over death we have to go to the controller where we in the end rendered checkout Ajax and that of course is here in shop checks that get checkout controller here here in get checkout. We now have to adjust our code a little bit because besides rendering this checkout page we have to prepare such a stripe session in the end. Now to prepare that we have to install a new package so I'll quit my development server and go ahead and install it. But running npm install dash dash save stripe stripe is the package name which we need to install. This is now a package which we can use on our server side code.
* Everything now works fine , but problem is that if we manualy enter the /checkout/success we goto the orders page , without going through stripe. We need to prevent this.
* Now that all works but does approach has a flaw currently in the end we confirm that an order was successful by simply running the logic and get checkout success that we can always trigger that if we just manually route to this page. If I add a product to the card again and now I don't order it but I simply go to slash checkout slash success here does all this exceeds my card is empty and I placed the order without paying for it. You can't always see your orders here in stripe though if you go there you will see your past orders and then of course you only see one order and that's the order we processed fruit stripe. So the order I hacked here with manually entering you are L of course does not show up here. Only the orders that re went through stripes form can be seen here. And of course you can look into the payments you received here and for example see the day. The email address the amount paid. The payment method the name of the user. And so on. Now the advantages with that you can actually compare that to the orders you see in your database and check if there are any fraudulent orders in there. So you should always do that when using this approach. But of course for a large scale shop manually comparing orders is not really an ideal solution. And indeed this is a weakness that is also listed here in the docs of stripe. If you go to stripe checkout one time payments you basically learn about the approach we just set up and dare you see that you should not rely on the success you are L alone. Instead you have to fulfill a payment which means make sure that stripe tells you when a payment happened instead of a you are L telling you now. Actually here you see that you can manually use the dashboard to check if the order really was placed so Dad would be our solution here as your application grows. Web hooks here would actually be the preferred solution. The idea here is that you can configure stripe such that it sends a request to a U or L of your choice which you would have to manage here in your application with routing and death. Dan tells you that the order succeeded because a stripe sends you that request behind the scenes. It does not to send the request to a user ELO of your page. Anyone can enter. Instead it will be a request validated by stripe. That's not as easy to fake setting up Web hoaxes a bit more complex though but the documentation here is really great. If you want to do it. The biggest problem we have at the moment is we couldn't really test the web hooks here because for web hooks to work stripe needs to be able to send the request behind the scenes to your web page and therefore your web page needs to be exposed to truly real Internet. And that's not the case for us here during development. It's only running when our local machine which is why redirecting the user works but sending such a behind the scenes request would not work. Hence if you need that automated process the stripe dogs are the way to go. For the moment using the dashboard to validate orders and to make sure that you're really only shipping goods to users who placed a valid order is the way to go and that is how you can implement payment with stripe.