**Lecture 354**

**How payments work**

Let's first of all start with the payment process, how does this work? Well it's a pretty long process, we start by collecting the payment method, so the credit card data typically. We then have to verify that, is the credit card data correct, is it expired, is the number correct? We then have to charge it and after we charge it, we have to manage payments, so that includes things like fraud protection, also managing disputes and so on. And last but not least, we of course have to process the order in our app, so in our server side code, for example that we store it in the database there. Now all the payment related things here are pretty complex tasks, both from a legal and a technical side and therefore typically you outsource them and by the way, even very big companies outsource this. Stripe is a very popular company offering payment services, it offers a great integration and it's super easy to add to any application as you will see in this module. How does it work? Well we have our client and our server, so that is all the code we own and we wrote and in the client, we'll collect credit card data. We'll do that with the help of stripe and we'll send it to the stripe servers which are not owned by us but by that company to validate that input. Once it is valid, stripe will return a token to us which basically encodes or which includes that credit card data and the confirmation that it is correct. We send that token to our server, so to our code and in our code, we create a charge or we charge this payment method then with the help of stripe again. So we create a payment, an object, a charge object, we send that to stripe with that token and with our price included and stripe will then do the actual charging, do the actual managing and we will get a response once this is done and then we can also continue with our code and edit this or store this in the database and so on. So this is generally how that will work, now let me walk you through that step by step.

**Lecture 355**

**Adding a checkout page**

To add payment to our application, let's work on that checkout.ejs file with which we never work thus far. I want to go there before I actually place an order when I click on order now in my cart, so I'll go to this page first. For that, I'll copy the content of cart.ejs add it in checkout.ejs to make sure that I have my default head. I'll get rid of that stylesheet though, I don't need it here, I want to include the navigation, the main content will be totally different though, I will keep my body however, so or my end here. So with that, I have everything set up to have a basic checkout page, now let's first of all make sure we reach that before we start adding stripe. For that we need a new route of course in a shop.js file, here we have our create order route and in the end what I need is a new route, a get route which we can name checkout and there, I want to be authenticated because we can only buy something if we are logged in let's say and then I'll use my shop controller and there, get checkout which is an action I still have to create. And in that get checkout page then, we'll have the possibility to create an order and so on. So for now, let's just add this and let's work on our controller then, on our shop controller and there I need a get checkout method. So maybe here before post order because logically, it comes prior to that, I'll have get checkout with request response and next and in there, I want to render my checkout page. So we can of course grab one of our other render functions here, render that, here I will render shop/checkout, the path can be checkout, it doesn't really matter, we don't have a naviagtion item we want to highlight, here we can checkout as a text and of course we need some data for the checkout. Now what do I want to return here? Well in the end, I will need the same data as I have in my cart, so I need my products here which I do have in the cart, so I can actually copy that entire code from get cart I guess and add this here and now just make sure we use shop checkout, path checkout and here, checkout but besides that I also need to calculate the total let's say. So total sum or whatever you want to name it and that total sum of course can be retrieved from our product data because in the cart or in the products here we have in the cart, you have to keep in mind that products is in an array of products where we have the quantity and then a product ID field with detail data about the products because we populate this product ID field with the detailed product data. So therefore, the total sum can be generated by looping through all products, so here I can do products forEach and create a new variable, total which is zero initially and then there, total is equal plus equal this shortcut to add something to the existing total, so total plus the product and then there is a quantity field times the product product ID price and then it's this total which I pass to my view. So that's all the data I should need in checkout view. So now back in my checkout.ejs file, now I can indeed grab that unordered list for rendering a list of items like this, I'll go through all my products. I'll remove the delete button though, it's too late at this point let's say, we can of course always cancel the checkout process if we want to though, actually I'll delete this entire form therefore, this is my unordered list and below that, we can add a div with a h2 tag of total and then here I output total sum. Now let's have a look at that and to have a look, I need to make sure that on cart.ejs when you click order now, I don't submit this form here, so I will actually comment this out, we don't need that anymore but instead I want to have a normal anchor tag here where I say order now where I simply point at /checkout, so this new route we added and I'll just give this a class of button to look a bit like a button. With all that added, let's go back to our page, let's go to products here maybe, add this to the cart and click order now and this is now my, well a page here. Now it's not looking super beautiful I'd say, I need to include that cart.css file again because since I'm reusing the rest of the code here and we could create our own stylesheet but I want to keep this relatively short, since I'm using all that, I of course also want to use the same styles. So now if I reload this checkout page here, it's looking ok, the total could be placed in the middle maybe, we can do that by giving this div a class of centered which is a utility class I added earlier in the course and now we have this checkout page. Again we can always work on the visuals but it's not so much about the visuals here. So now we have this total and now I want to add stripe and for that, let's first of all set up an account with them.

**Lecture 356**

**Using Stripe in your app**

You can simply google for stripe and you should find their page and they have a really great documentation by the way in case you want to use it. So definitely dive into developers documentation but we can simply click Create Account here and then create an account really quick. Now once you got your account created you can already get started. First of all you need to validate your account to make sure you click on that verify email link you get in an extra email. And once you did that you're ready to get started. No 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 life 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. And with that we can go back to home and there click on grow your online business with payments and read the docs. Now this takes us to does stripe documentation. It turns out that you have various different ways of implementing payments with stripe and of course you can check out their entire documentation to learn all about the different ways of collecting payments. Now here we can go to Web under Bill your own and click on integrated stripe J.S.. This simply allows us to implement a javascript library on the front end. So in our view is to make that whole payment process very very smooth and straight forward. So what we'll do is we'll pick this script here and go back to our code and there in the checkout. E J ust file. Let's go to that div where we output our sum and let's actually add a new div below it which also has that center class. And in this new div we can paste in that script but not just as a script let's also add a button here with an idea of order dash B10 and a class of button for the styling where we can say order just like this. Now let's add another script tag here where we now will write some inline script and that's all front end javascript so not running with node j asked on our server but instead executing in the browser off our users. Now here we can first of all call stripe with a capital S like this and insert your local testing key. Now you'll find this here in documentation already pre filled. This is the key. You can also see in the developer part of your home screen. So put in other words here under developers api keys. This is the key I'm talking about. This is the same key see here. So we can grab that key and entered is as a string as arguments here for this strive function call this scribe function is available because we're running this scripture we're importing this script and thereafter let's get access to our order button by using document get element by Adi That's Dom API which we can use in JavaScript which runs in the browser to get access to element on the page and here I will get access to that button by its ideas by simply passing the ideas here. 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. So in node J.S. so let's wait for it is to finish old and restart my server and now here at the top we can import stripe by requiring stripe like this but then does actually gives us a function which we need to execute to which we now need to pass our private stripe key. That's the key. You have to reveal here and by the way of course I'll change those keys after this recording session so you can use mine. Copy that key here and enter it is here. Now always keep this key private. So only use it in your note J.S. code never expose it in one of your views because they're your users could see it. And this is a key you should always keep private now with stripe imported like this. Let's go back to get checkout so to the controller where we prepared a checkout page and in there of course you want to gather all the product stayed up that's all fine. However I will adjust as a bit and create a variable products here right at the beginning of the function. Also a total which is Syria initially and change this year to not create a new constant or a new variable here but instead use these variables here which now simply are available everywhere dysfunction in any nested function as well. Whereas before does only was available inside of this function and dad will just not be enough. Once we made the changes we're about to make leave did code as it is otherwise. But instead of rendering immediately here we now need to do something different here. We need to do something else here in this promise and this then block we should return stripe. Check out dot sessions create. Remember we needed such a session key in our template. Now here we're going to create such a session which ultimately gives us such a key to create you parse an object where you configure that session. Now one thing we can already do is we can grab a render here cut debt and add a new. Then block their offer which is why I had to outsource some data into global or into function wide variables and in this then block you'll get the stripe session eventually we're not done configuring it but you will get it there and in here you can then rendered a checkout page and all the past let's say a session I.D. key to the template which holds a session dot I.D.. So this session you're getting here it will have I.D. field which is that drive session. Now we'll use that in a view in a second. Before we do so let's configure that session because there are some things which we do need to configure for when we need to add a payment underscore method underscore types key here that holds an array and there we want to add card as a type which means we accept credit card payments next let's add a line underscore items key and there we need to specify which items will be checked out. So here I want to use my products which I basically created here and mapped as a little bit because each product needs to look a bit different. So every product will be transformed with help of the built in map method. Javascript offers on arrays and they are the new object I return for every product the new products array which will therefore be part of that newly returned products array we passed line items will have a name which is p dot product I.D. dot title description which is p dot product I.D. dot description in case you're wondering why we can access title and description on product I.D.. Keep in mind that we populate it that product I.D. field so it's not just a product I.D. It's a complete product data which is why we can access the title description for example. The amount is p dot product I.D. dot price times 100 because we need to specify this in sense the currency is USD for a U.S. dollar and the quantity is Pitot quantity. That is data which stripe needs India. And that's the format stripe needs. It needs an array of objects which have currency quantity and amount and then all the year our extra name and description fields. And this needs to be named name for example. So with this re formatted data we also give stripe to data needs to process the payment but we're still not done there is more we need to configure on this sessions specifically we need to add a success underscore you are L and A cancel underscore a you are L these are you are ls stripe will redirect a user to once the transaction was completed or failed here I want to dynamically derive d you are L and domain off to server this node script is running on so that it is valid both here in development where we are on local host as well as later once we deployed is on the page with any IP or domain. So here I will use request protocol which is a property again I can get from this request object express gives us that is simply age 2. He or H CBS plus colon forward slash forward slash so this will build us something like h to be colon forward slash forward slash plus request get host. This will give us our host address. So for example local hosts three thousand during development or later once we deployed it the IP address or domain of the host deployed it on to. So now it is will give us that start of the redirection you are ill thereafter I want to go to slash checkout slash success here and now I will copy that and added here as well and redirect to slash cancel so these are in the end routes which stripe will redirect us to once we ever confirmed the payment or cancelled it with that we're creating such a session for forwarding the session I.D. to the view. So back in checkout E.J. us we now need to output the session ideas here now we do so with each. Yes we can also inject something into javascript code which is pretty neat. We can inject our session ideas here with the familiar E.J. s syntax just inside of javascript code in this case instead of a string of our javascript code was that we can be redirected to stripe hopefully and then back but the pages were redirected to once the transaction succeeded or failed. Checkout success or checkout cancel well these routes might not exist yet. Indeed if we check our routes file here to shop file we have no checkout success and checkout cancel routes there. So let's add them maybe here below checkout. We need to add queue get routes slash checkout slash success and then a number one slash checkout slash cancel. Now when we cancel I will in the end execute the get checkout controller again. I simply want to redirect the user back to the checkout page. If we cancel. If we succeed I need a new controller though and there I want to do the same thing I did before in post order so we can simply redirect the user here to post order. Now we already use post order here on slash create order. That is a route we don't need anymore though because we will now replace that thanks to our new checkout flow. So we only need checkout success at checkout cancel and for checkout success we go to post order. We're actually here I will trade a new controller get checkout success but you could have used post order as well. Now let's go to shop J.S. and there. Let's copy post order and simply trade a new controller function named get checkout success though. As I said you could have all just kept post order. I'm just doing it here. Tools have to get word at the beginning because in the end a get request will reach this controller here. Now in this function you can leave everything as it is. We're gathering all the products from the cart and then we're trading on order storing them in a database and then we redirect the user to orders. If we now saved it let's give it a try. Let's go back to our page here and there to the cart where I have two items in there to click on order now and I got an error. Now the reason for that error is actually related to stripe. You could debug it by going to capture us and logging the error we're handling here and yet need to add a name here. Max to be able to use this. So after naming this year I reload does looks better. Now here I have to order button and if I click that you should be redirected to stripes page. Now here you can enter any e-mail address and then some dummy card data like 4 2 4 2 4 2 A lot of 4 2s some date which is in the future here and then any CVC code of your choice and simply any name on the card click pay. And there's no process as this should succeed and redirects you to the orders page yet because it goes to the checkered success page where it is is created and indeed you should see that the card is empty now. 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. As I mentioned there is way more you can do a stripe and of course the official docs are the place to go if you really want to build online shop and use all the cool features stripe offers. But this lecture hopefully got you started with stripe.