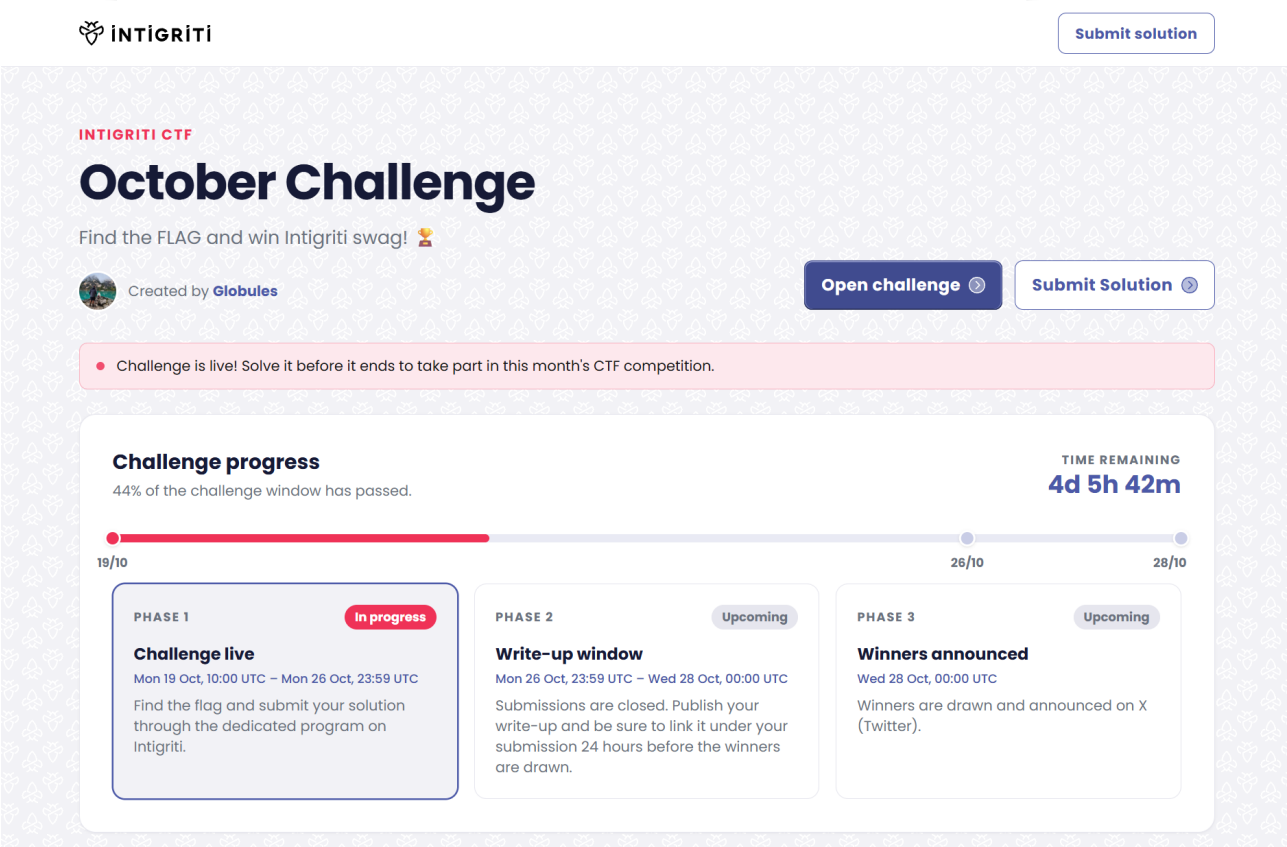
Intigriti October 2026 Challenge: CTF Challenge 1026 by Globules

In October 2026 ethical hacking platform Intigriti (https://www.intigriti.com/) launched a new Capture The Flag challenge. The challenge itself was created by community member Globules.



Rules of the challenge

- Leverage a vulnerability on the challenge page.
- Find the flag in the format `INTIGRITI{.*}`
- Shouldn't be self-XSS or related to MiTM attacks.

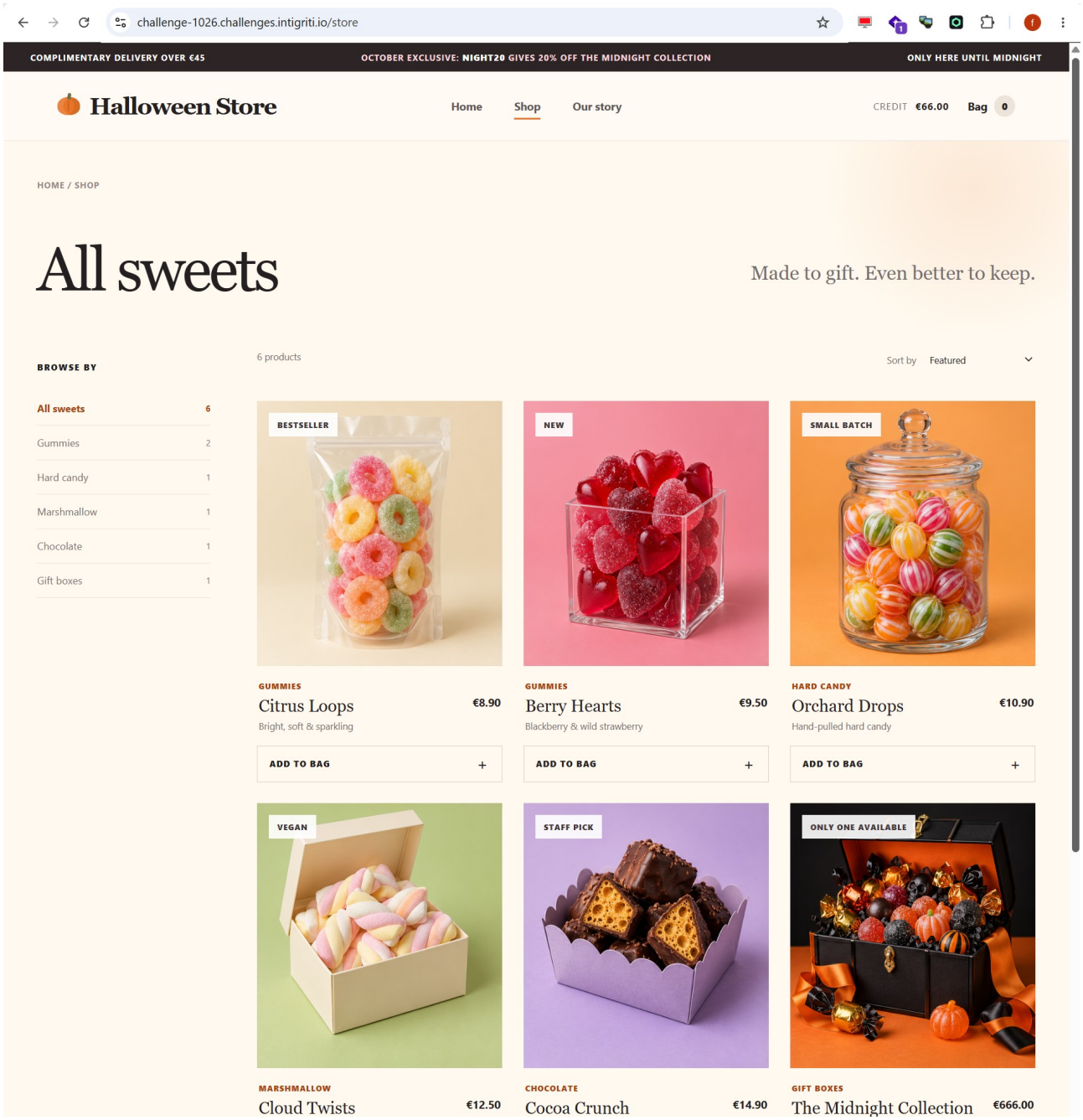
Challenge

The rules indicate a flag needs to be retrieved from the web application by finding a vulnerability. Once working on the challenge it became clear this was not the usual blind XSS, RCE, SQLi... challenge that requires technical knowledge or a decent AI setup to retrieve the flag. No this challenge could probably be solved with a bit of knowledge but mostly a sharp mind checking for Business logic flaws.

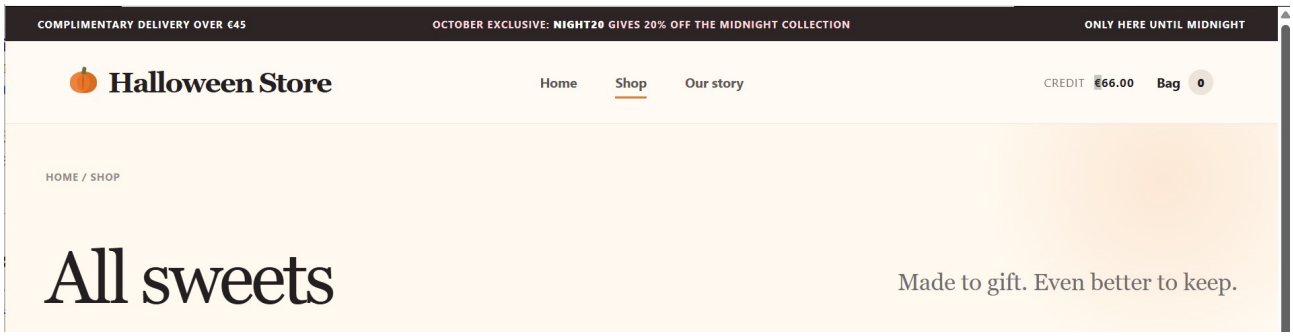
Steps taken to solve the challenge

Step 1: High level recon

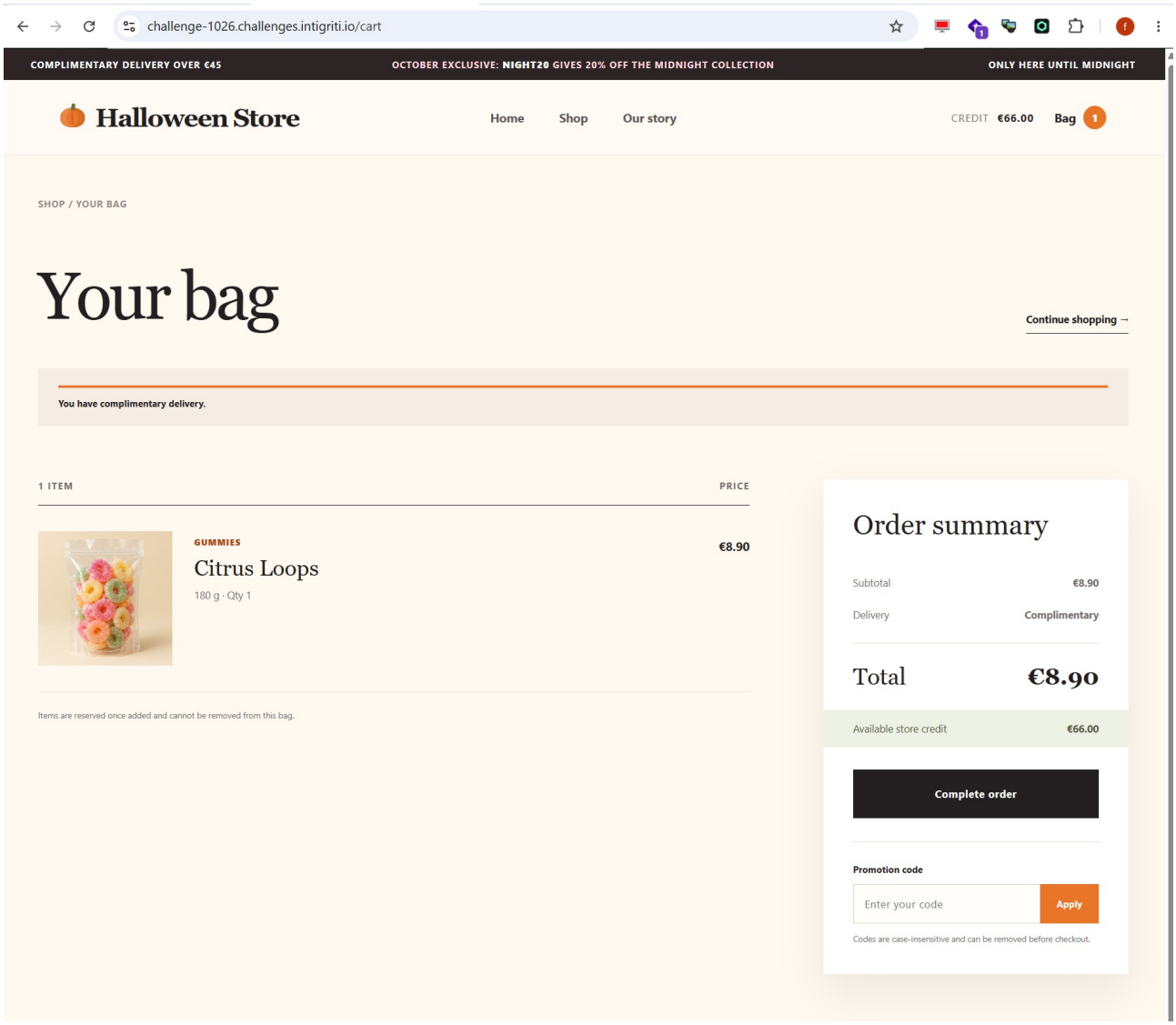
First thing first and that is using the application as intended. The challenge is a web shop that can be found at: https://challenge-1026.challenges.intigriti.io/store



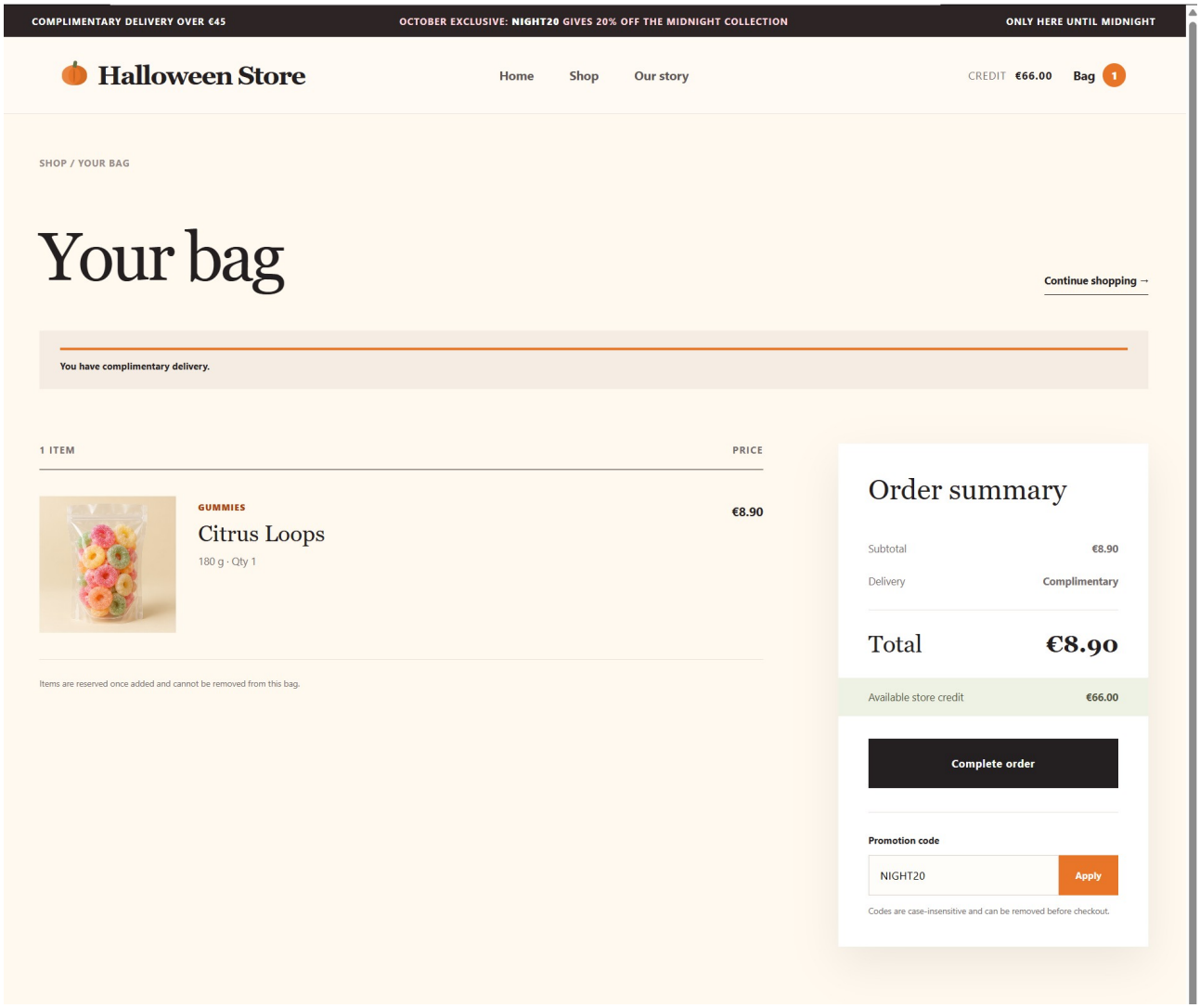
Clicking around in the shop has the expected functionalities where we can buy Halloween candy. We have € 66.00 to spend.

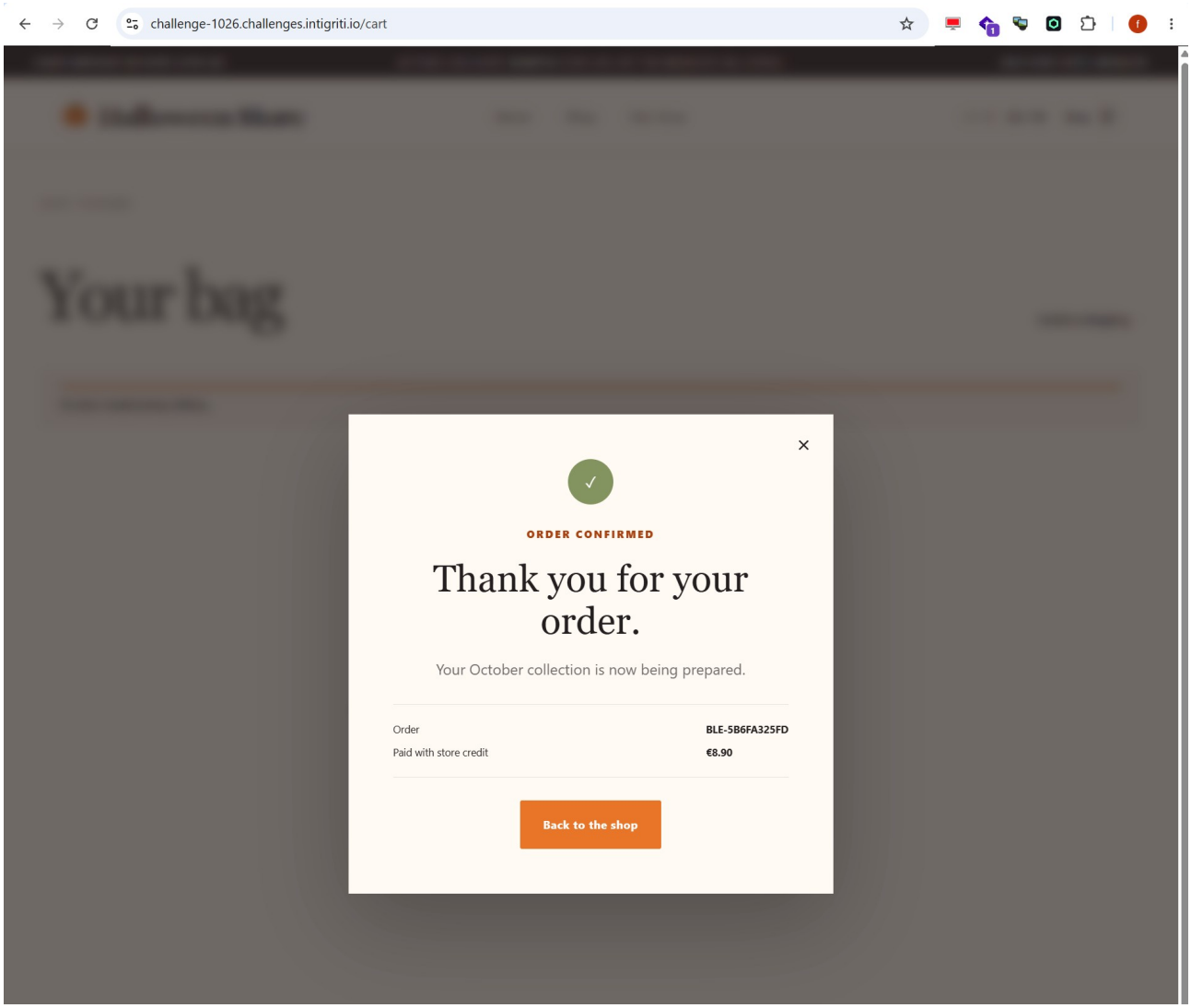


We can add anything into our bag ready for purchasing.

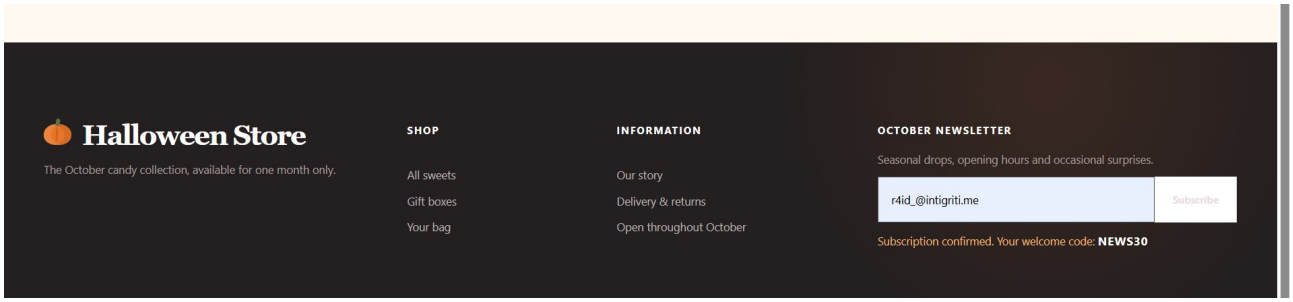


We can use coupon code like “NIGHT20” which we find on top as an “October exclusive” to get discounts.





At the bottom we have the option to subscribe to the newsletter which gives us another coupon



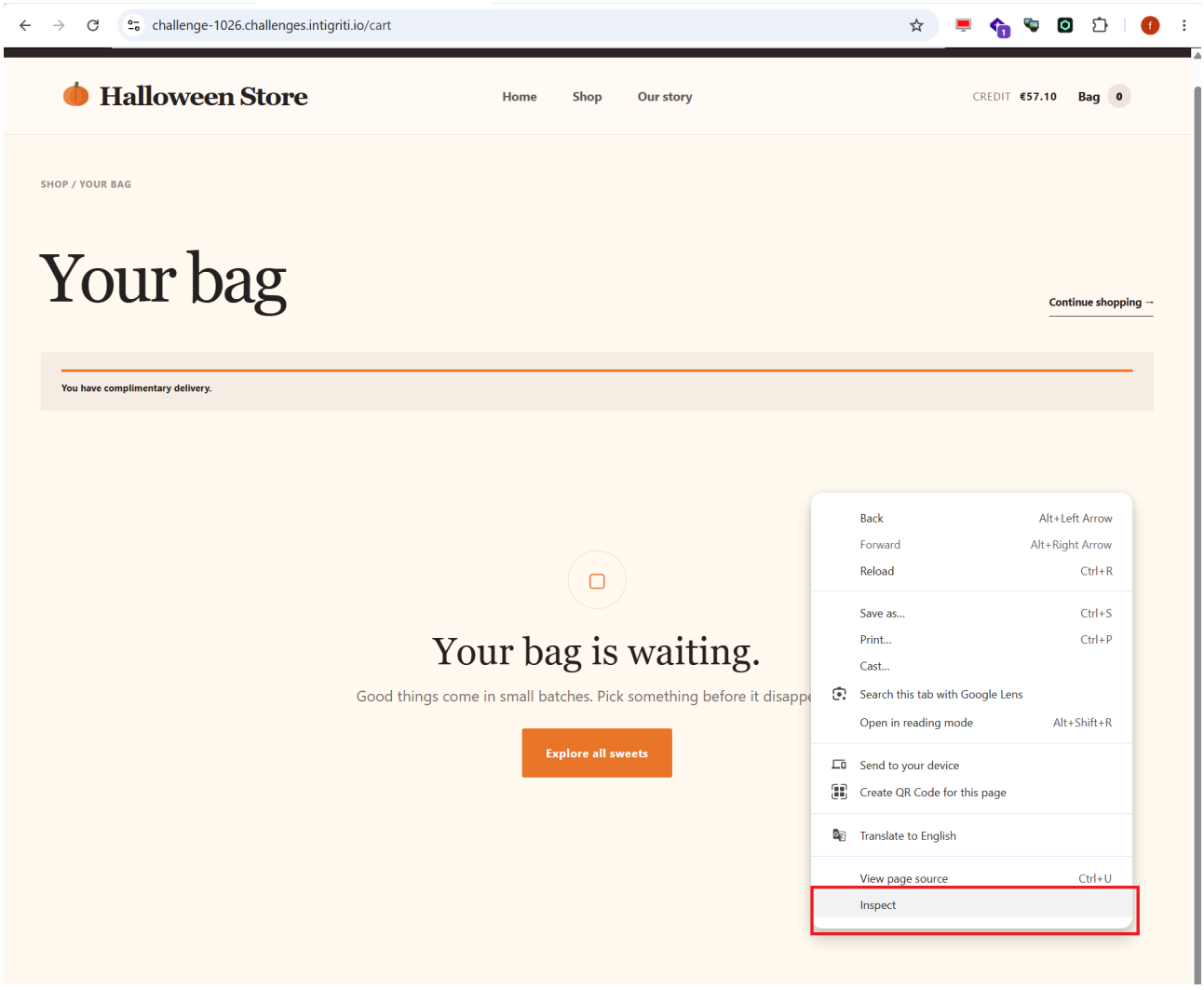
This shows a bit all the functionality the web shop has. Nothing strange at this first sight but technically we already have everything to solve the challenge at this moment. We will dive now a bit deeper to see how you could find a clue where the flag was hidden.

Step 2: Client side JavaScript recon

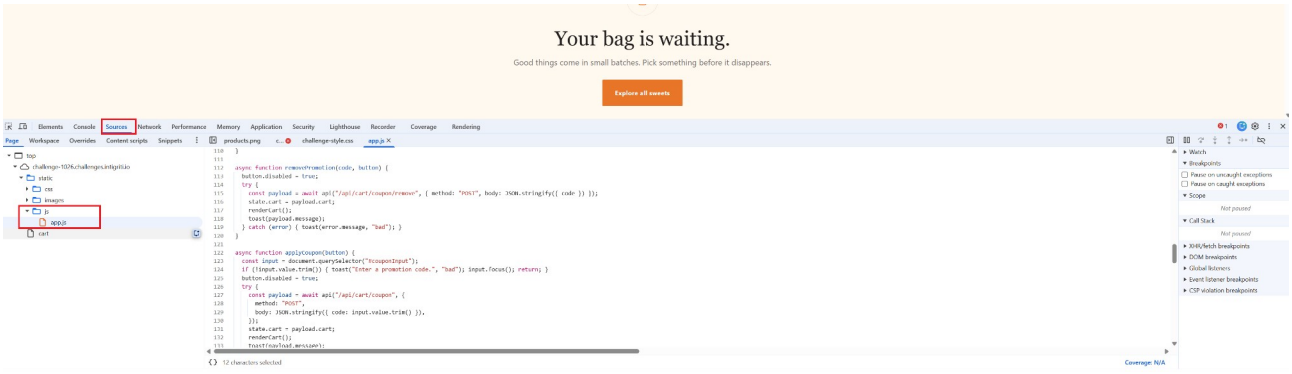
This part is a bit more technical but not really needed to be able to solve this challenge. The client side JavaScript just has a clue to where we can find the flag.

JavaScript is running client side which means your browser can show the JavaScript code behind a web page. In this case the shop.

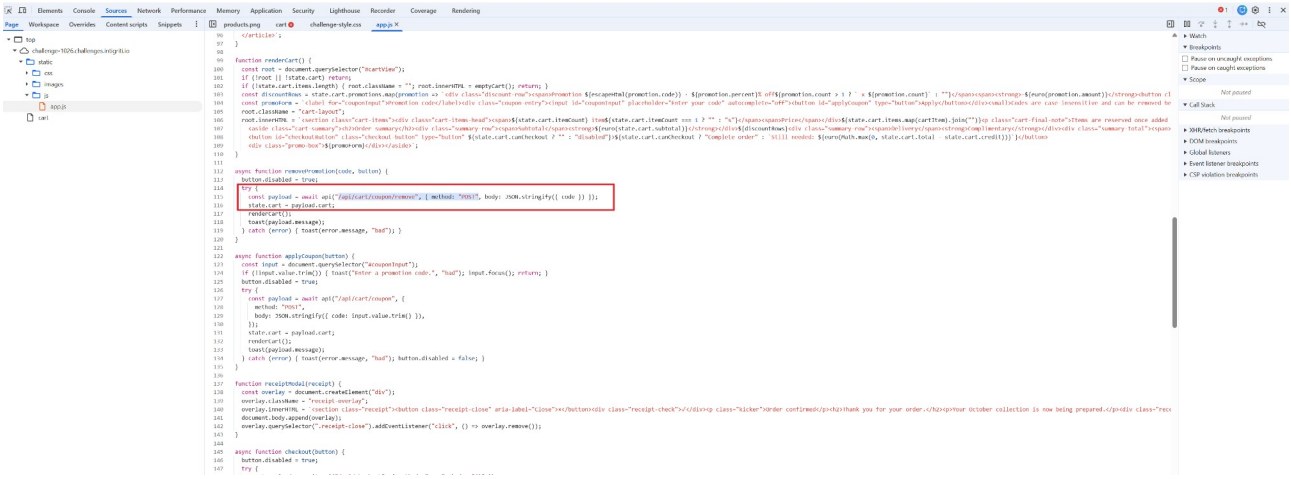
Right click anywhere on the web shop page and choose “Inspect”



This opens the browser developer tools. Choose the “Sources” tab and inside the “js” folder we can find “app.js” which is the client side source code for our web shop.



On a real target I would look for leaked secrets, API endpoints, unknown HTML pages… inside the JavaScript like the API path shown in the screenshot below. This shows the POST web request made to the API back-end to remove a coupon for example.

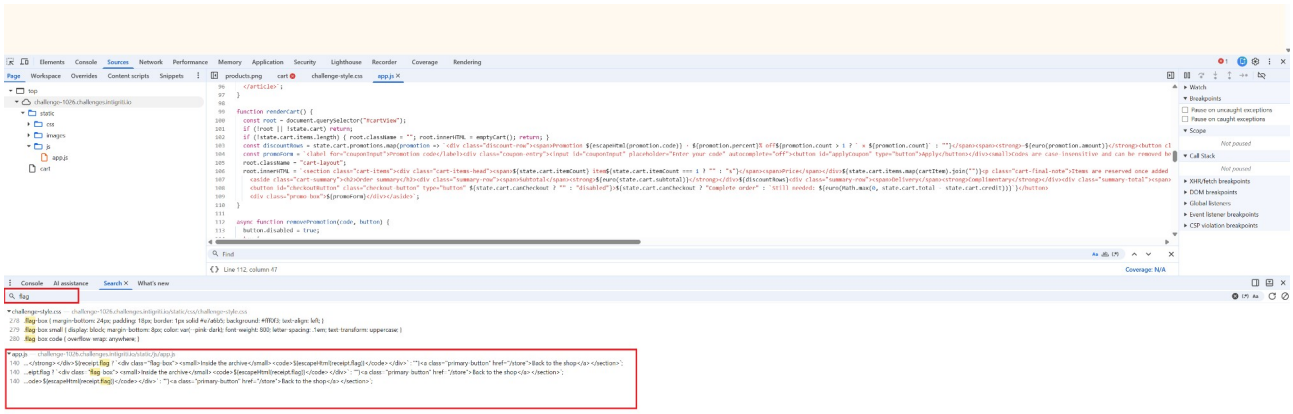


For a real target we would use tooling or manually go through such JavaScript file to find such API endpoints and dig for parameters to be tested for any possible attack vector.

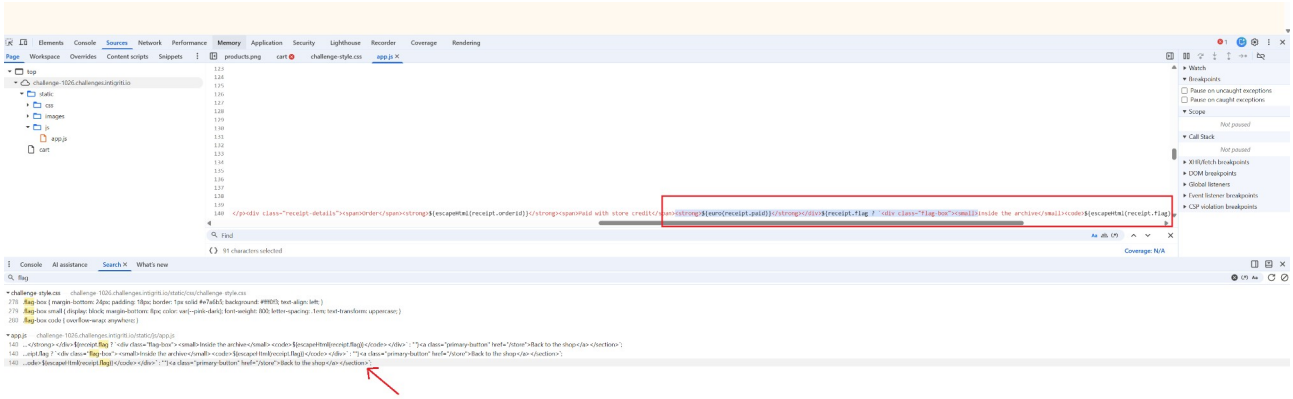
This challenge is far more simple. You could also search in the JavaScript file for the word “flag” for example. Right click in the left part of the developer tools on the web URL of the shop.
Choose “Search in all files”



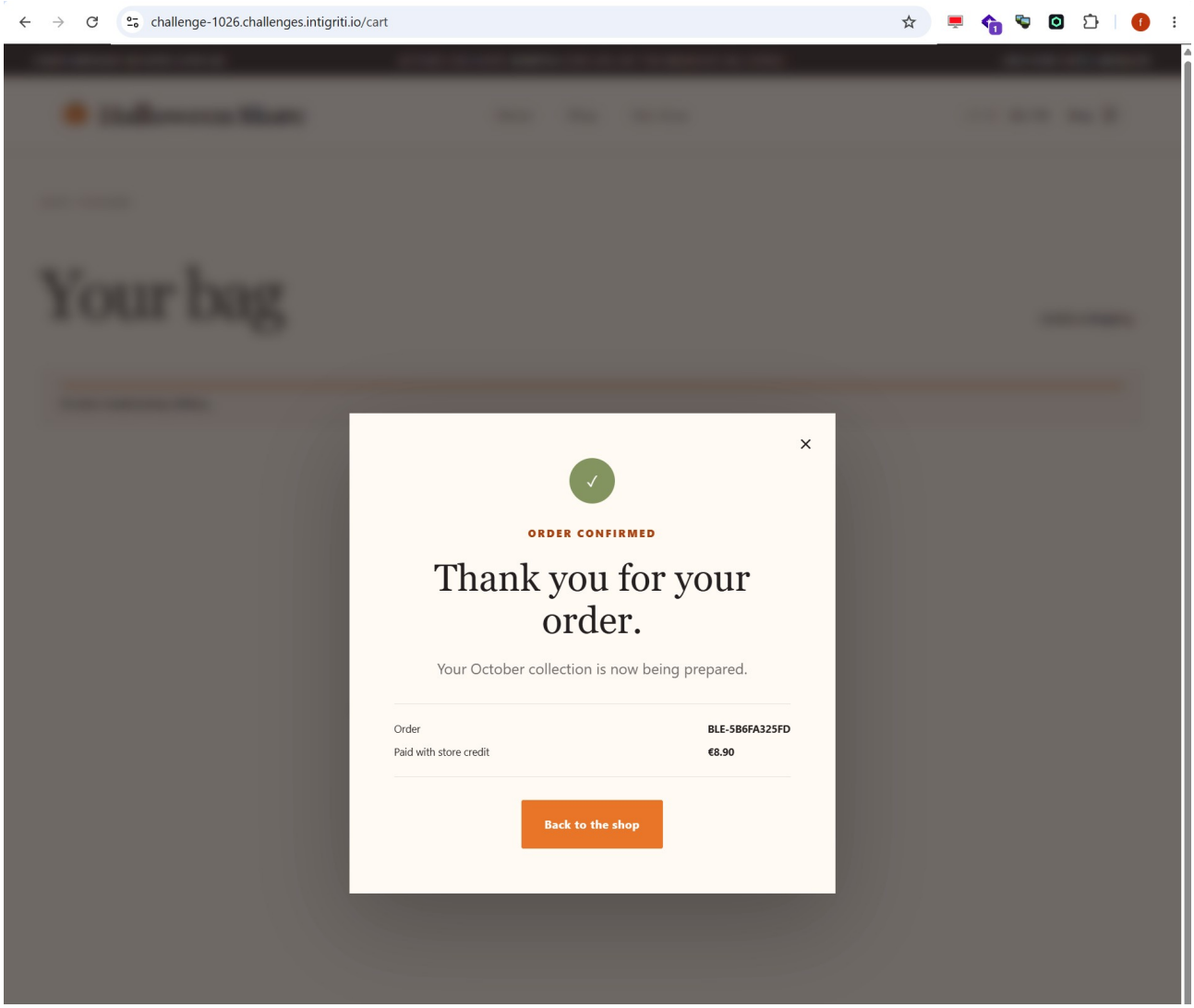
If we now search for the word “flag” we get several lines inside the client side JavaScript showing the flag is used in the web shop HTML and JavaScript code.



Clicking on the search results brings us to the exact line inside the JavaScript code and this reveals a big hint. The flag is hidden in the web shop receipt. This means the popup shown when we purchase something from the web shop should show the flag.



Remember during our high level recon we made a purchase but the popup never showed the flag so we need to find a flaw in the web shop to show the flag.



Take-aways from recon:

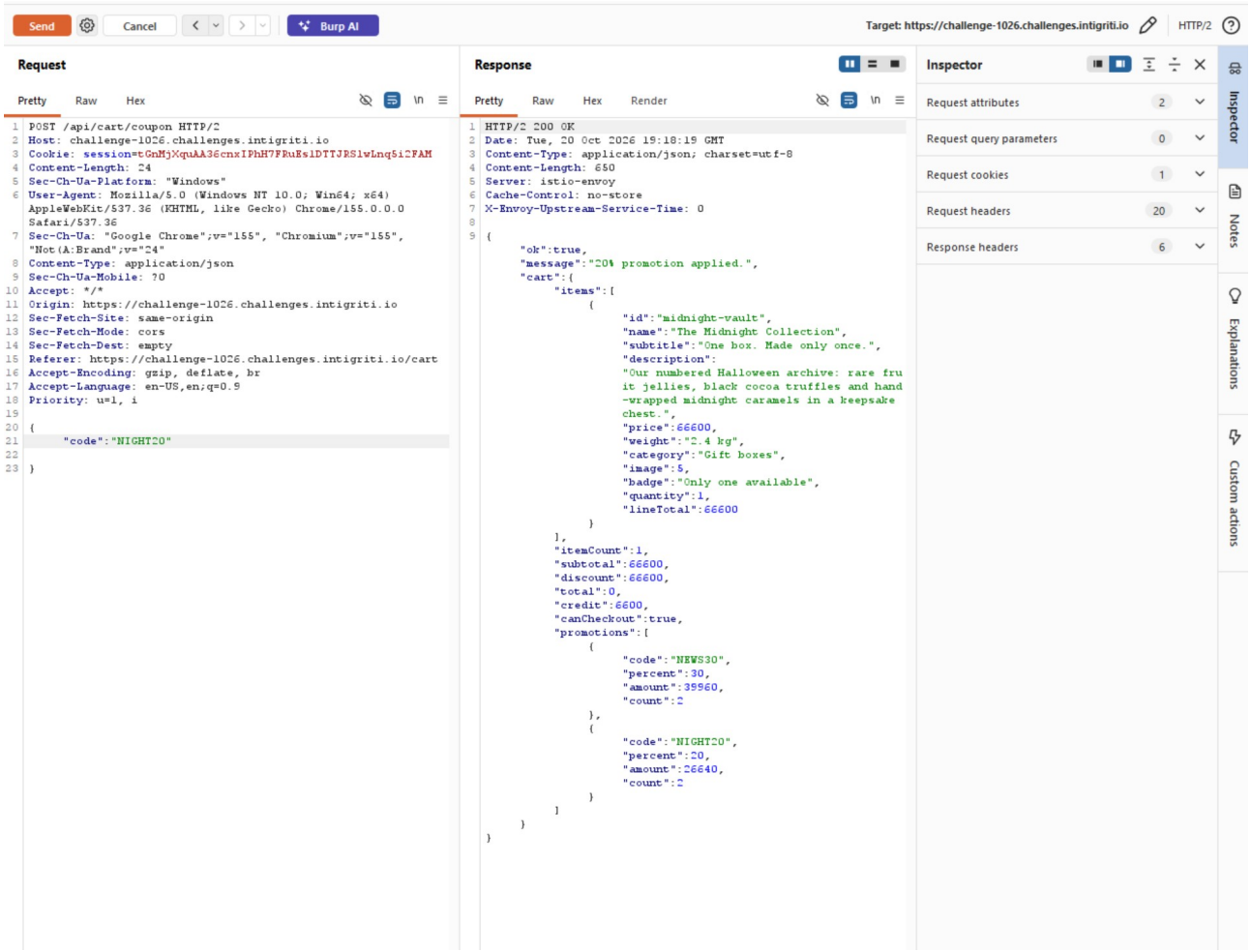
- The focus should be on the web shop receipt as this hides the flag.
- The receipt has an HTML element to show the flag so we we are probably not looking for blind XSS, RCE, SQLi… or that kind of attacks but more a business logic flaw.

Step 3: Breaking commercial rules via a business logic flaw

A first idea would be to search inside the client side JavaScript code for the flaw and uncover the flag but that would be to easy and probably any AI tool would solve that in seconds or minutes.
Most web applications can have client side code like JavaScript but also server side code we can under normal circumstances never see as a visitor.

This web shop also has code running on the server side. The flaw to uncover the flag is hidden there but we got a big hint we need to play with an order receipt.

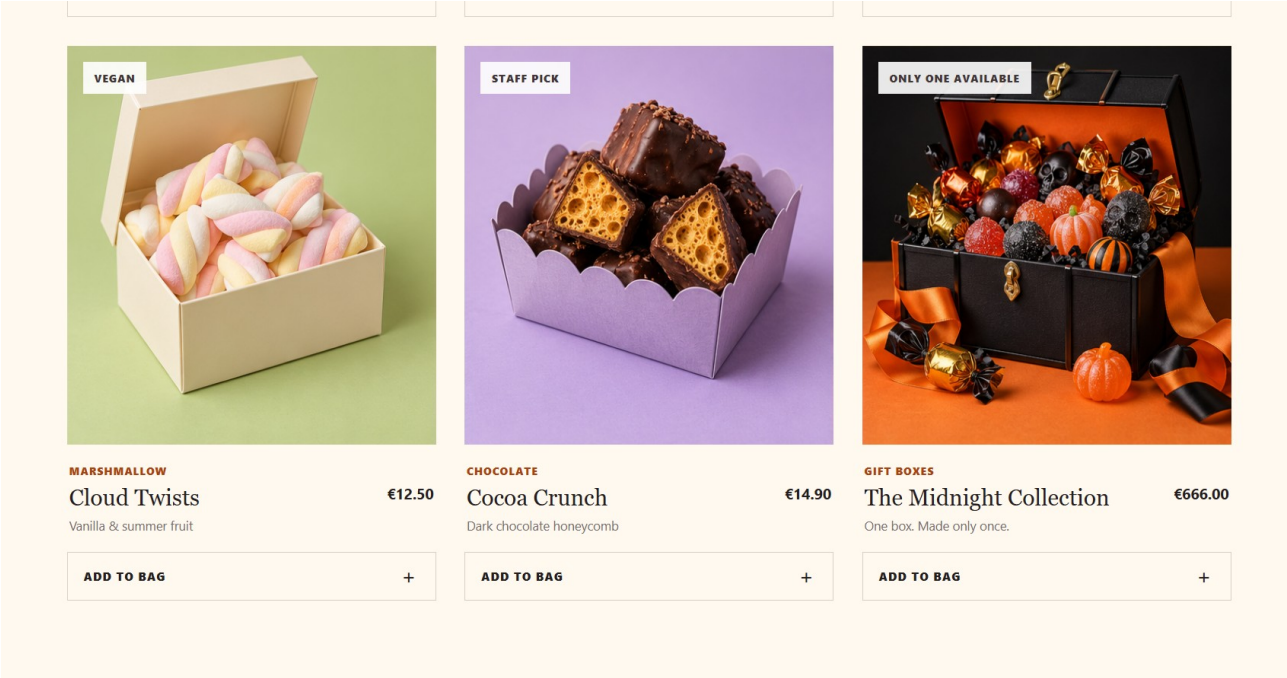
With a proxy tool like BURP suite it is possible to see the API call leaving your webbrowser towards the server side code. Here an example applying coupon code “NIGHT20” to get 20% discount. On any real target I would test for SQLi, RCE, JSON… flaws here. But this challenge is less technical.



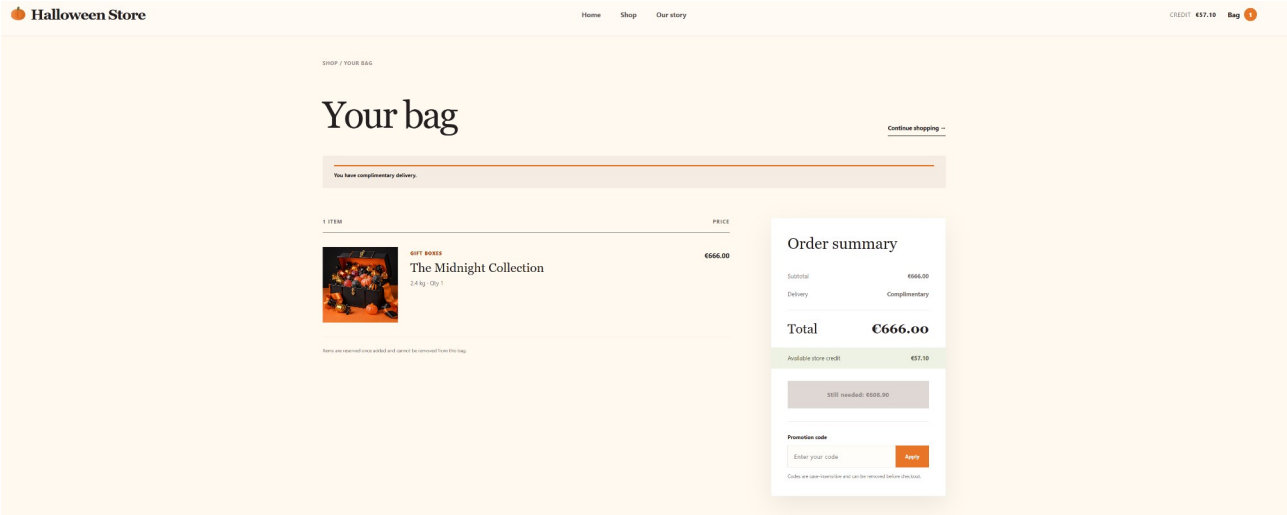
The order receipt consist of 2 things:

- Products we buy for a certain price.
- Coupon codes that give us discounts.

The shop gives us € 66.00 euro to buy anything. If you look closely to the items available in the shop one is very expensive and costs € 666 which is far more then the budget we have.

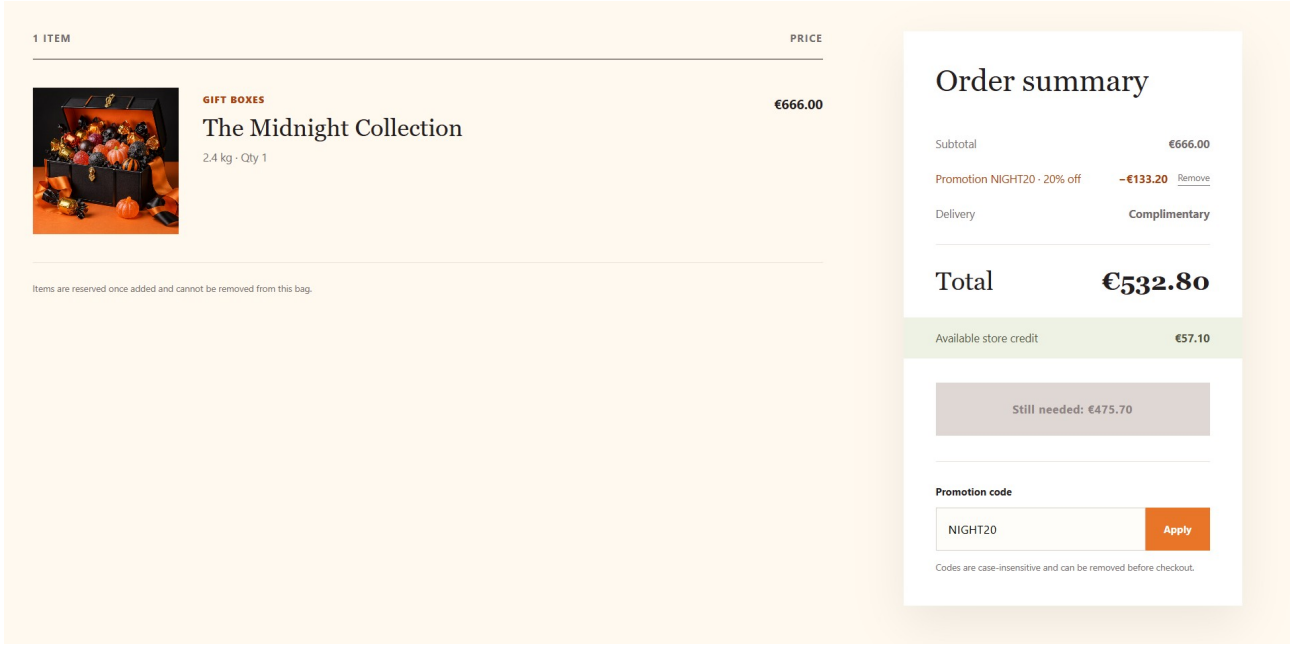


It would be nice to get that one for free. Adding it to our bag it shows we still need extra money to buy it.

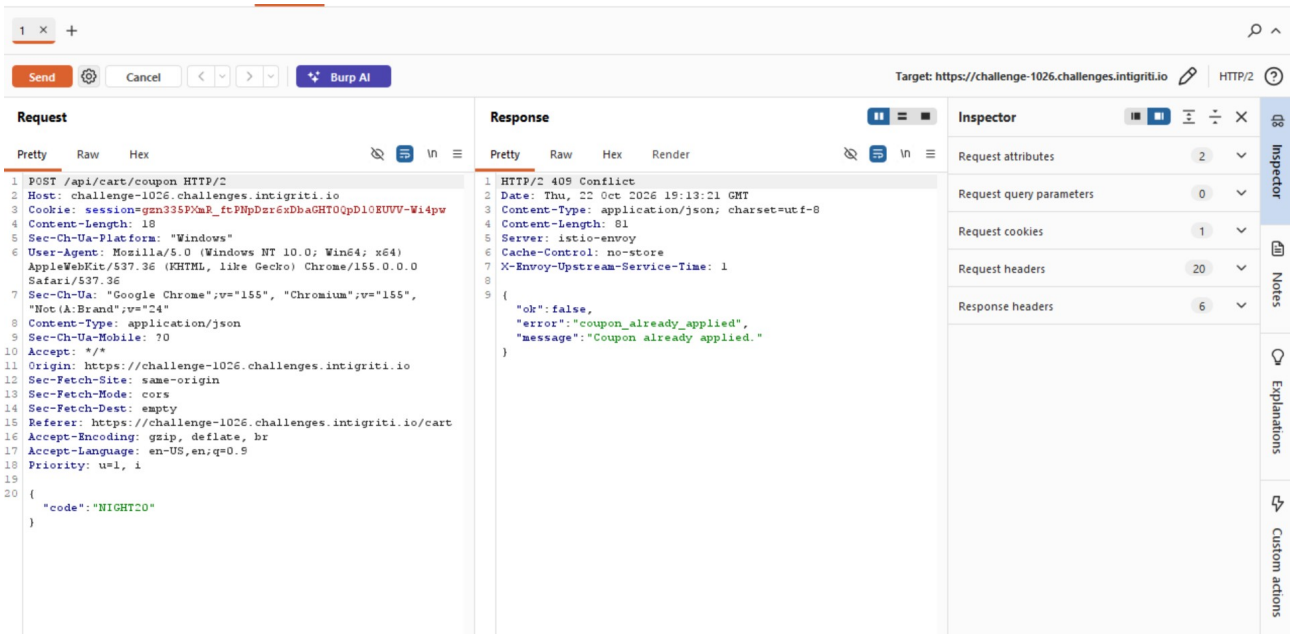


But we have coupon codes. “NIGHT20” gives us 20% and once subscribing to the newsletter gave us “NEWS30” for another 30%

Trying to add “NIGHT20” multiple times on our bag wont work:



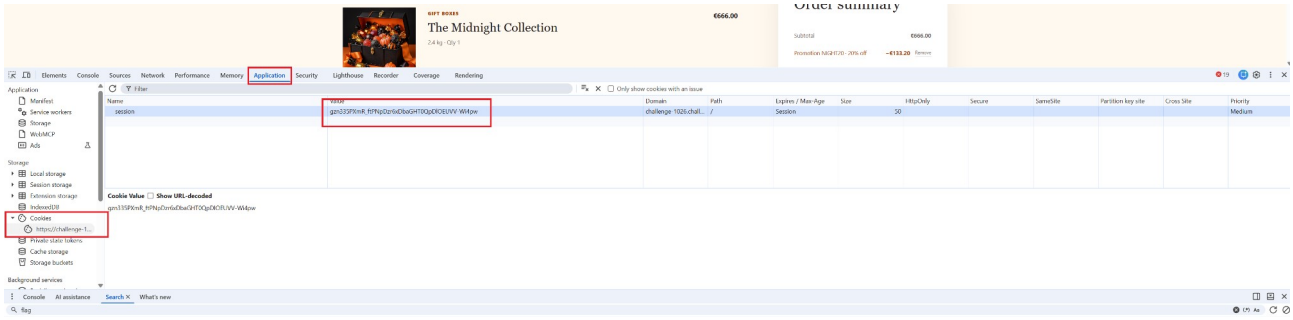
Looking at the API call in a proxy tool like BURP suite shows why. The coupon is already applied.

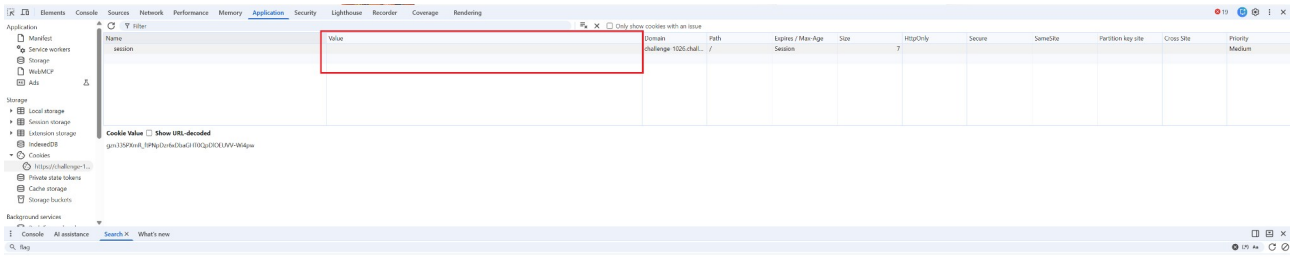


The web shop at the server side correctly checks if we try to apply a coupon multiple times.

Coupon code “NEWS30” is only valid on the first order being placed and if you subscribed to the newsletter. In case you already bought something to use coupon “NEWS30” use another browser or delete your session cookie.

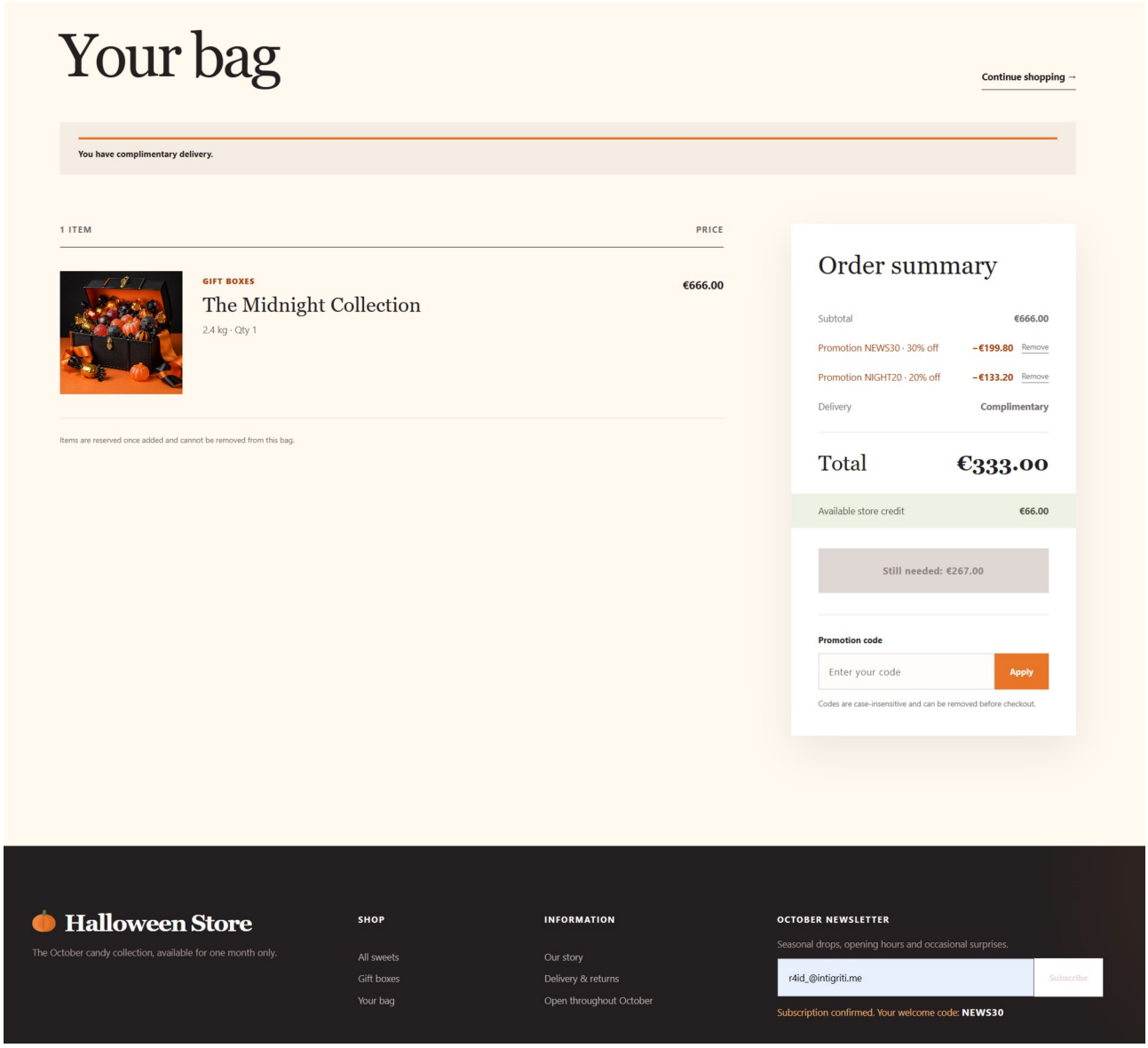
Delete the session cookie and refresh the web shop page.





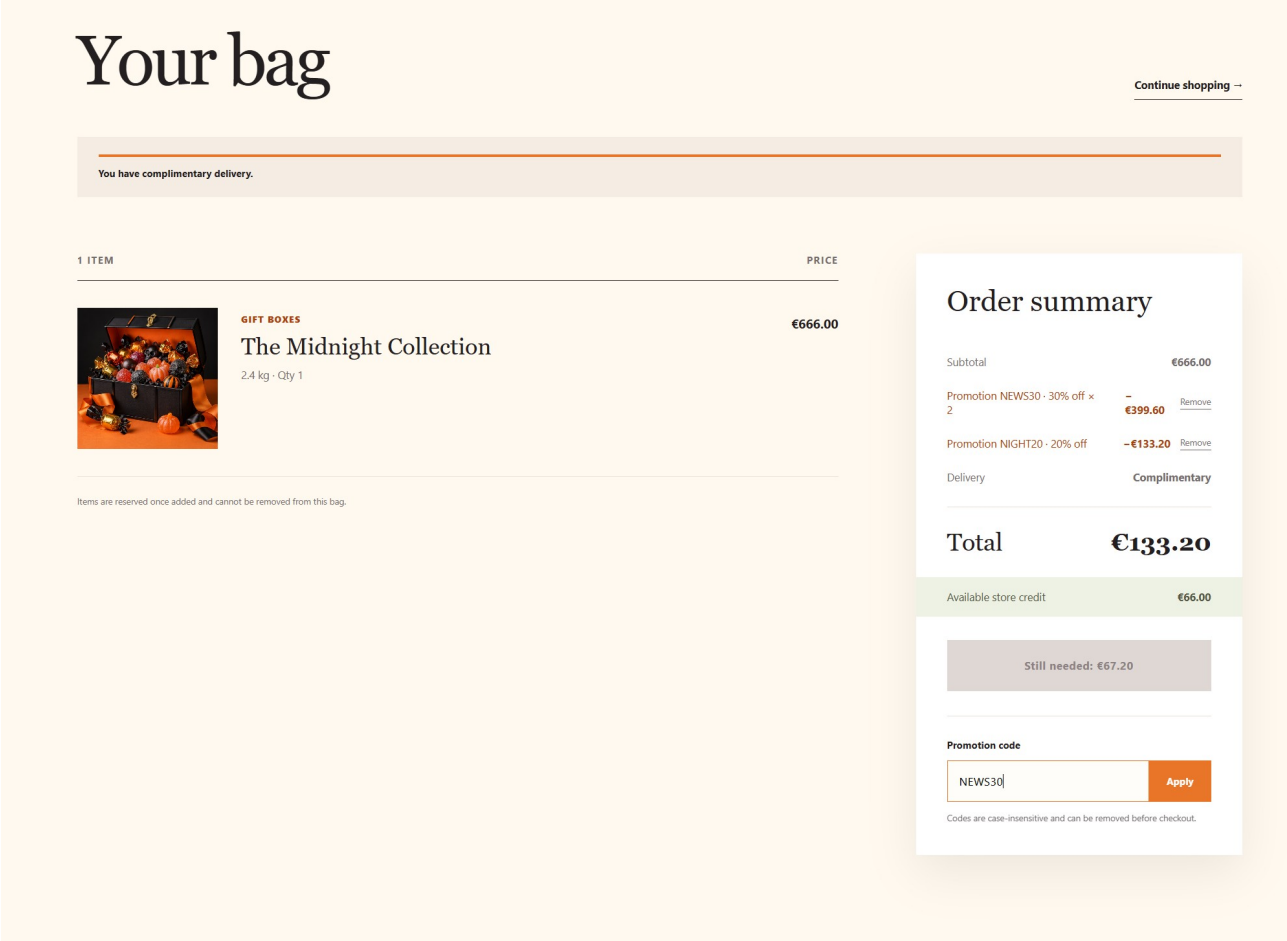
Add the € 666 “The Midnight Collection” to your bag as the first item to be bought. We have € 66 available again.

Subscribe to the newsletter to activate coupon “NEWS30” and use it. Add the other discount coupon “NIGHT20”



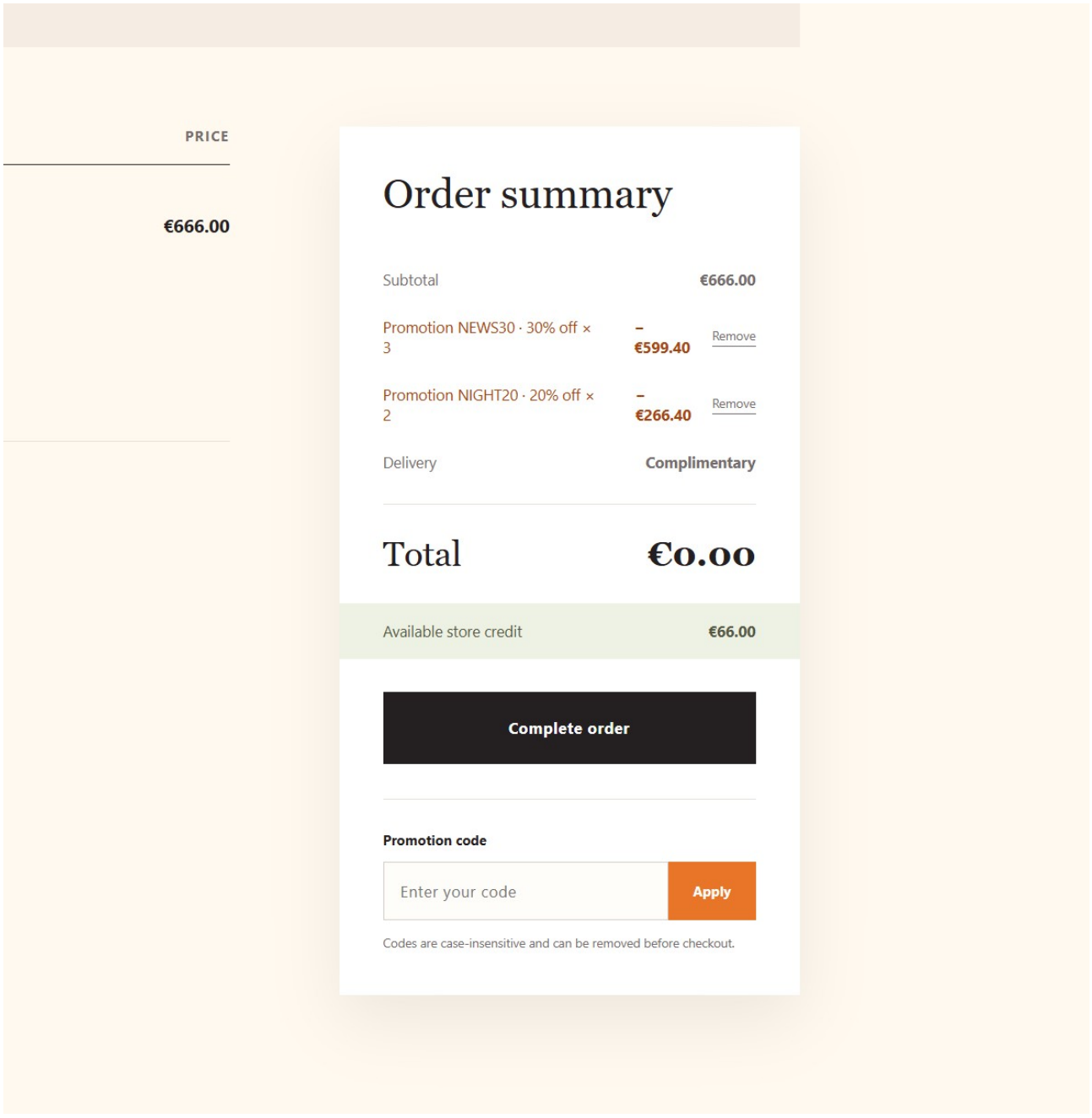
Coupon stacking seems not forbidden. We can add multiple different coupon codes. Using 2 times the same coupon code right after each other is not allowed but stacking them is working.

We first added coupon “NEWS30” then we added coupon “NIGHT20”. If we now try again to add “NEWS30” this works like a charm:



By each time swapping between the 2 coupon codes we can stack them to get the “The Midnight Collection” for free. The back-end server code clearly has a business logic flaw not checking for stacking different coupon codes.

We are not allowed to use the same coupon code 2 times right behind each other but on server side nothing is checked when we swap between coupon codes.



The total prices drops to € 0.00 so we are ready to complete our order. The receipt now shows the flag we are looking for as we were able to purchase a product for free via a business logic flaw.

