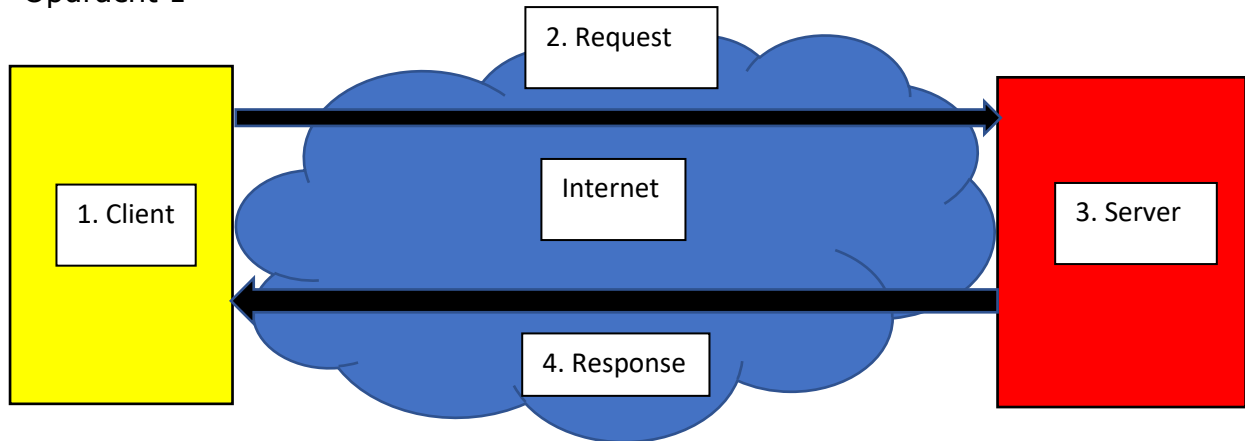


Opdracht 1



Opdracht 2

https://www.bol.com/nl/p/hoe-werkt-dat-nou/9200000057347012/?country=BE&suggestionType=browse#product_alternatives

protocol

subdomein

host

path

parameters

fragment

Opdracht 3

Network tab

The screenshot shows a web browser window with the address bar displaying `htmldog.com/examples/headings1.html`. The page content consists of a list of headings: **Heading 1 (h1)**, **Heading 2 (h2)**, **Heading 3 (h3)**, **Heading 4 (h4)**, **Heading 5 (h5)**, and **Heading 6 (h6)**. Below the headings is a small image of a dog, labeled **HTML Dog**.

The browser's developer tools are open to the **Network** tab. The top section shows a timeline of network requests. The bottom section is a table of requests:

Name	Status	Type	Initiator	Size	Time	Waterfall
headings1.html	200	document	Other	(disk cache)	6 ms	
badge1.gif	(failed)		headings1.html	0 B	191 ms	

Response body

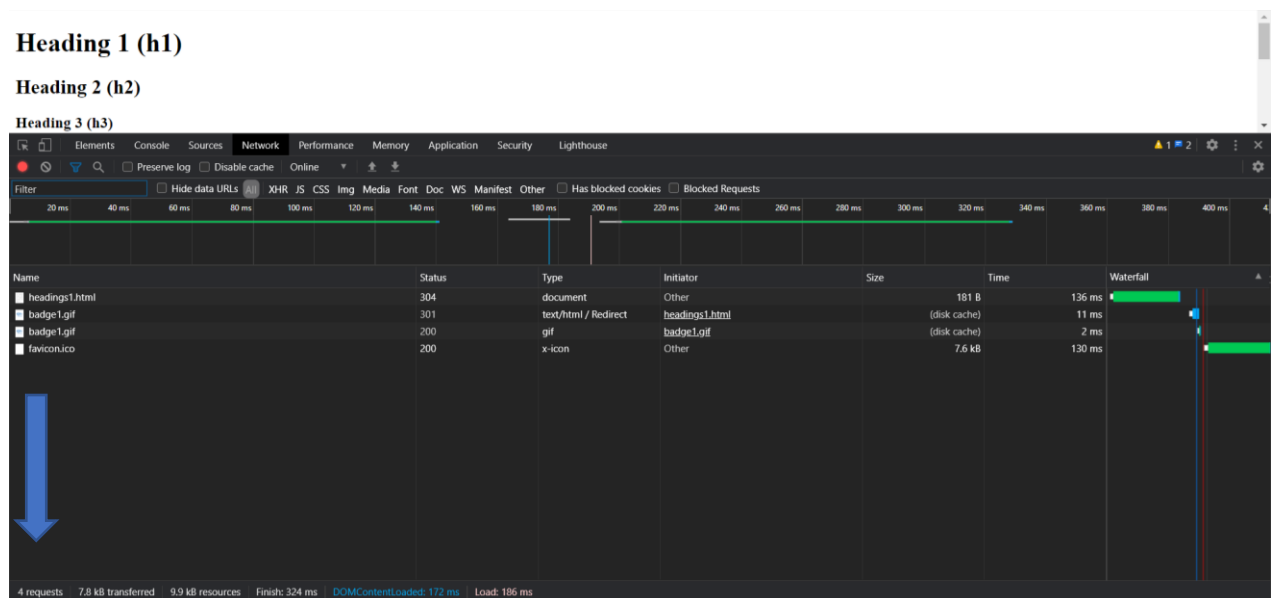
```
Name
headers1.html
  badge1.gif

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Browser default heading styles</title>
6 </head>
7 <body>
8   <h1>Heading 1 (h1)</h1>
9   <h2>Heading 2 (h2)</h2>
10  <h3>Heading 3 (h3)</h3>
11  <h4>Heading 4 (h4)</h4>
12  <h5>Heading 5 (h5)</h5>
13  <h6>Heading 6 (h6)</h6>
14
15  <!-- Link back to HTML Dog: -->
16  <p><a href="http://www.html5dog.com/examples/"></a></p>
17 </body>
18 </html>
```

Network code

Status Code: 200 (from disk cache)

Opdracht 4



Welke resources heeft je browser nog meer opgevraagd? Hoe zie je dit?

Gifs and icons

Kun je in het HTML document in de response body terugvinden waarom net die resources werden opgevraagd?

Omdat in de html een href is naar een img, deze moet dus opgehaald worden.


Opdracht 5

Welke andere soorten resources worden opgevraagd door het inladen van deze pagina?

Documenten, stylesheets, jpeg's, png's, scripts, gif's, font's, fetch, vnd.microsoft.icon en xhr's

Werden alle requests naar dezelfde server verstuurd? Om dit te zien kun je best een 'domain' kolom toevoegen aan de tabel. Rechtsklik hiervoor op de hoofding van de tabel en vink 'Domain' aan. Rechts naast elke request zie je de timing informatie die weergeeft wanneer de request verstuurd werd en wanneer de response werd ontvangen.

Neen,

Name	Status	Domain
 honeypot.css?qg0nsj	200	www.vives.be
 normalize.css?qg0nsj	200	www.vives.be
 mothership.css?qg0nsj	200	www.vives.be
 style.min.css?qg0nsj	200	www.vives.be
 support.css?qg0nsj	200	www.vives.be
 reject.css?qg0nsj	200	www.vives.be
 googleanalytics.js?qg0nsj	200	www.vives.be
 gtm.js?id=GTM-KSFD3NG	200	www.googletagmanager.com
 hotjar-561704.js?sv=5	200	static.hotjar.com
 fbevents.js	200	connect.facebook.net
 hotjar-561704.js?sv=6	200	static.hotjar.com
 flowbox.js	200	connect.getflowbox.com
 beurs.html	200	www.vives.be
 OpenSans-Regular.woff	200	www.vives.be
 iconfont.woff	200	www.vives.be
 OpenSans-Bold.woff	200	www.vives.be
 OpenSans-ExtraBold.woff	200	www.vives.be
 OpenSans-Light.woff	200	www.vives.be

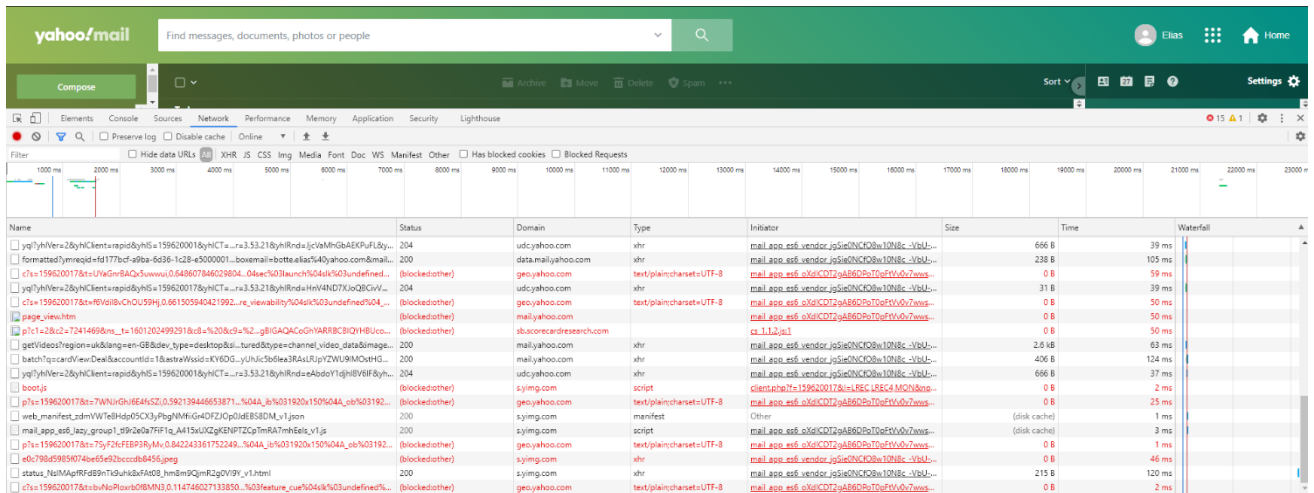
Het opvragen van 2 resources van een webpagina kan normaliter onafhankelijk van elkaar gebeuren, de browser kan dus een pagina sneller kunnen inladen door een volgende request te versturen nog voor de response op een vorige request werd ontvangen. **Hoe kun je dit uit de timing informatie afleiden?**

De requests gebeuren praktisch op hetzelfde moment, de responses niet. Sommige resources hebben meer tijd nodig om te downloaden.

Lijkt het erop dat het aantal gelijktijdige 'onafgewerkte' requests beperkt is? (onafgewerkt, in de zin dat de request verstuurd is maar nog geen response werd ontvangen).

Nee, de requests gebeuren bijna instant, maar volgens een prioriteitranking.

Opdracht 6



De requests dienen om de mailbox te refreshen en mogelijke nieuwe mail op te halen.

Opdracht 7

En merk op dat er wel degelijk een response teruggestuurd wordt alhoewel de pagina niet bestaat. Hoe komt dit?

Omdat het de foutpagina template moet ophalen.

Name	Status	Domain	Type	Initiator	Size	Time	Waterfall
deezepaginabestaatniet.html	404	www.vivres.be	document	Other	6.8 kB	48 ms	
system.base.css?tgq0nj	200	www.vivres.be	stylesheet	deezepaginabestaatniet.html	(memory cache)	0 ms	
date.css?tgq0nj	200	www.vivres.be	stylesheet	deezepaginabestaatniet.html	(memory cache)	0 ms	
ccdoc.css?tgq0nj	200	www.vivres.be	stylesheet	deezepaginabestaatniet.html	(memory cache)	0 ms	
reset.html5.css?tgq0nj	200	www.vivres.be	stylesheet	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery-1.8.2.min.js?v=1.8.2	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery-extend-3.4.0.js?v=1.4.4	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery.html-generate-3.5.0-backport.js?v=1.4.4	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery.once.js?v=1.2	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery.abusecss.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
dispal.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
viver_content.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery.cookie-1.4.1.min.js?v=1.4.1	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
google_analytics_reports.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
nl_at2652-bu5f34e8etv700w8b_vMh5w7ELbNlx7gw.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
googleanalytics.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
modernizr.min.js	200	cdn.jsdelivr.net	script	deezepaginabestaatniet.html	(memory cache)	0 ms	
jquery.placeholder.js?tgq0nj	200	www.vivres.be	script	deezepaginabestaatniet.html	(memory cache)	0 ms	

Wat betekent de status code 404 in de response header?

De browser kon communiceren met een bepaalde server, maar deze kon niet vinden wat wij zochten. Vandaar dat aangegeven wordt dat de pagina niet gevonden kon worden.

Name	Status
deezepaginabestaatniet.html	404

Opdracht 8

Wat is het verschil met de vorige opdracht?

We krijgen hier zelfs geen statuscode

Name	Status	Domain	Type	Initiator	Size	Time	Waterfall
data:image/png;base64...	200		png	chrome-error://chrome-error/#:5238	(memory cache)	0 ms	
data:image/png;base64...	200		png	chrome-error://chrome-error/#:5238	(memory cache)	0 ms	
data:image/png;base64...	200		png	chrome-error://chrome-error/#:5238	(memory cache)	0 ms	

Opdracht 9

Succescodes	Omleidingscodes	Client errors	Server errors
200=OK, succesvolle HTTP request.	301=Moved Permanently, deze en alle toekomstige aanvragen moeten verwezen worden naar de gegeven URI	400=Bad request, de server kan of wil niet de aanvraag volbrengen door een client error. (Bv te groot bestand)	500= Internal server error, de standaard error code wanneer een onbekende fout is opgetreden en er is geen geschikte specifiekere code.
204=No Content, De aanvraag was succesvol maar geeft geen content weer.	302=Found, de client moet naar een andere URL surfen of kijken	401=Unauthorized, er is een authenticatie vereist die nog niet is ingegeven of is mislukt.	503=Service Unavailable, de server kan de aanvraag niet behandelen (overbelast of down for maintenance)
	303=See Other, de response op de request kan gevonden worden onder een andere URI met de GET methode.	404=Not Found, De gevraagde source kon niet gevonden worden, maar is misschien beschikbaar in de toekomst.	

Opdracht 10

Zoek op het internet welke HTTP request methods er bestaan en schrijf ze neer.

GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

Waarvoor dienen de vaak gebruikte GET en POST methods?


















GET: vraagt een representatie van de specifieke resource aan, aanvragen met GET zouden enkel data moeten ontvangen.

POST: Deze methode is gebruikt om een entity aan te geven aan de gespecificeerde resource, dit veroorzaakt vaak een verandering aan de staat of neveneffecten aan de server.

Waar in een request staat aangegeven om welke request method het gaat, en hoe vind je dit

terug in de Chrome developer tools (zie uitleg bij opdracht 3)?

In de method tab.

Name	Method
 Methods	GET
 ZillaSlab-Regular.subset.bbc33fb47cf6.woff2	GET
 ZillaSlab-Bold.subset.e96c15f68c68.woff2	GET
 analytics.js	GET
 lux.js?id=108906238	GET
 perf.654b849a6fd9.js	GET
 google-analytics_analytics.js?secret=tpqf8q	GET
 clock.661d537f6023.svg	GET
 web-docs-sprite.22a6a085cf14.svg	GET
 react-main.c74b5c815269.js	GET
 mathml.3cb4c04c0706.js	GET
 auth-modal.119e5d70465f.js	GET
 react-bcd-signal.cf0fc711cfa2.js	GET
 whoami	GET
 favicon32.7f3da72dcea1.png	GET
 icon-survey.3c3e291c45a3.svg	GET
 curve.7d36cb7fbf85.svg	GET

Als je een url in de adresbalk van je browser typt en op enter drukt, wat voor request method gebruikt de browser dan om die resource op te vragen bij de server?

GET

Als je in een webpagina op een gewone hyperlink klikt, welke request method wordt er dan gebruikt?

GET

Josh Breckman worked for a company that landed a contract to develop a content management system for a fairly large government website. Much of the project involved developing a content management system so that employees would be able to build and maintain the ever-changing content for their site.

Because they already had an existing website with a lot of content, the customer wanted to take the opportunity to reorganize and upload all the content into the new site before it went live. As you might imagine, this was a fairly time consuming process. But after a few months, they had finally put all the content into the system and opened it up to the Internet.

Things went pretty well for a few days after going live. But, on day six, things went not-so-well: all of the content on the website had completely vanished and all pages led to the default "please enter content" page. Whoops.

Josh was called in to investigate and noticed that one particularly troublesome external IP had gone in and deleted *all* of the content on the system. The IP didn't belong to some overseas hacker bent on destroying helpful government information. It resolved to googlebot.com, Google's very own web crawling spider. Whoops.

After quite a bit of research (and scrambling around to find a non-corrupt backup), Josh found the problem. A user copied and pasted some content from one page to another, including an "edit" hyperlink to edit the content on the page. Normally, this wouldn't be an issue, since an outside user would need to enter a name and password. But, the CMS authentication subsystem didn't take into account the sophisticated hacking techniques of Google's spider. Whoops.

As it turns out, Google's spider doesn't use cookies, which means that it can easily bypass a check for the "isLoggedIn" cookie to be "false". It also doesn't pay attention to Javascript, which would normally prompt and redirect users who are not logged on. It does, however, follow every hyperlink on every page it finds, including those with "Delete Page" in the title. Whoops.

After all was said and done, Josh was able to restore a fairly older version of the site from backups. He brought up the root cause -- that security could be beaten by disabling cookies and javascript -- but management didn't quite see what was wrong with that. Instead, they told the client to NEVER copy paste content from other pages.



Alex Papadimoulis
Founder, The Daily WTF

Stel, iemand schrijft een webapplicatie om producten te beheren en realiseert de "wis productgegevens" functionaliteit door middel van een gewone hyperlink met 'wis' opschrift. Om een product te wissen moet een gebruiker dus gewoon op de 'wis' link klikken bij dit product.

Op een bepaald moment komt bv. de google-bot langs die (programmatorisch) alle links uitprobeert om te zien wat dit oplevert aan nieuwe pagina's om te indexeren. Of stel dat de browser een accelerator plugin bevat die proactief gelinkte pagina's inlaadt zodat de gebruiker niet hoeft te wachten bij het klikken op een link.

Wat zou er dan gebeuren met de productgegevens?

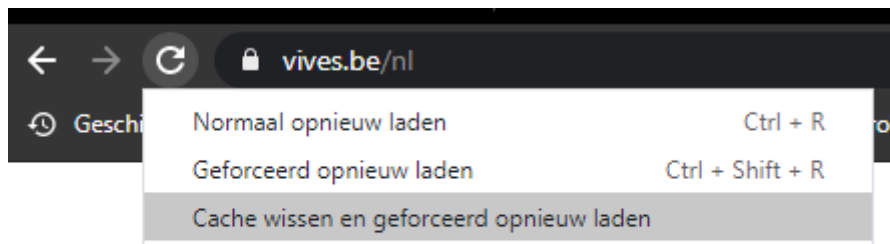
Ze zouden allemaal verwijderd worden door de bot.

Opdracht 11

Bekijk de vele requests die het inladen van die ene pagina heeft veroorzaakt. Hoeveel request waren er in totaal?

132+

Opdracht 12



Hoeveel kilobytes of megabytes aan data werd er verstuurd om alle nodige resources in te laden?

3.8mb

Hoelang duurde het vooraleer alle resources van de pagina waren ingeladen?

2,03s

Klik nu gewoon op de refresh knop. Kijk nogmaals hoeveel data er werd verstuurd. Waarom is dit zoveel minder? Laadde de pagina sneller?

63,9kb, we vragen niet alle resources op, waardoor de pagina sneller laad (788ms)

Waar kun je zien welke documenten daadwerkelijk verstuurd werden en welke niet?

Als ze een response body hebben, dan zijn ze verstuurd geweest.

Waar vindt de browser dan de inhoud van de documenten die niet bij de server werden

opgevraagd?

In de memory cache.

Hoe weet de browser welke documenten best opgevraagd (moeten) worden en welke niet? M.a.w. hoe lang mag de browser een bepaalde resource als 'vers' te beschouwen? (Hint: kijk eens naar de response headers)

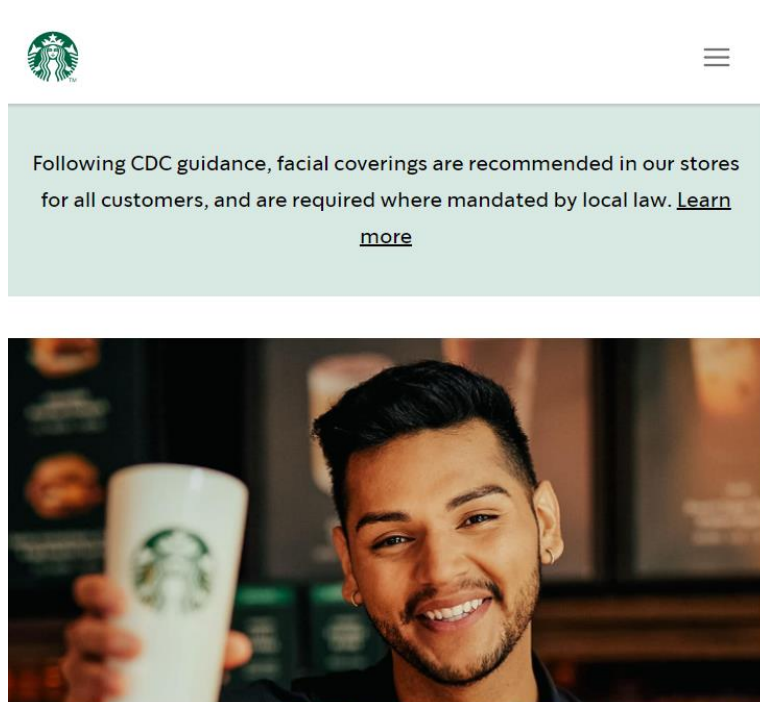
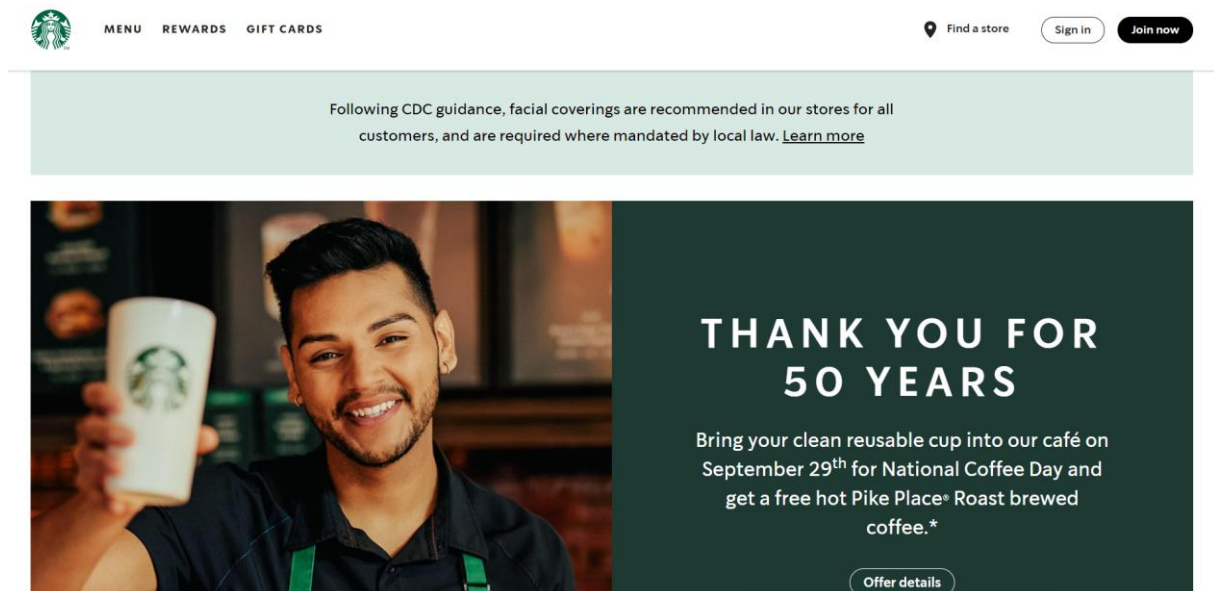
De documenten die een bepaalde functie bevatten krijgen voorrang

Opdracht 13

- ☐ <https://www.nieuwsblad.be/> 8
- ☐ <https://www.cnn.com> 11
- ☐ <https://www.vives.be> 7
- ☐ <https://www.vrt.be/vrtnws/nl/> 6

Advertising, essentieel, sociale media, website analytics, onbekend

Opdracht 14



Hoeveel verschillende layouts tel je?

2 verschillende layouts