Leerdoelen week 3

Leerdoelen:

* **Ik kan uitleggen wat een server is.**  
  Een server is een machine die is aangesloten op het internet, met daarin IP adressen
* **Ik kan uitleggen wat requests en responses zijn.**

Request = aanvraag vanuit browser met IP adres aan server

Response = terugkoppeling van server met data aan browser (is zowel een file als een   
 contenttype =) nu weet browser hoe het de data die het terug krijgt moet behandelen  
 bv als: html of css of javascript

* **Ik kan uitleggen wat een IP adres is.**

Mijn servers IP adres is de postcode (IP adres) die ik op mijn computer heb en hiermee

* **Ik kan uitleggen wat DNS is.**

Domein Name Service: wanneer ik een website invoer op google dan gaat de aanvraag eerst naar de DNS en vraagt wat is het IP adres en vervolgens vraag hij het IP adres op bij de server

Het Domain Name System (**DNS**) is het systeem en netwerkprotocol dat op het Internet gebruikt wordt om namen van computers naar numerieke adressen (IP- adressen) te vertalen en omgekeerd. Hoewel dit "vertalen" genoemd wordt gaat het gewoon om opzoeken in tabellen, waarin namen aan nummers gekoppeld zijn.

* **Ik kan (grofweg) de stappen uitleggen tussen het opvragen van een HTML pagina totdat deze in je browser te zien is.**

1 ik tik in google naam site  
2 dan DNS (domein name server lookup), hey we moeten in de server bij dit IP adres gaan zoeken  
3 terugkoppeling van server aan browser, browser weet nu hoe het de data moet behandelen als; htms of css of javascript

4 dan loopt de browser alles na van boven naar onder: = parsing (zie tekening onder)  
5 wanneer het een **asset** tegenkomt, dat is bv een css file , javascript file, image dan wordt er opnieuw een request gedaan

vb  
css file: – vertel de browser een request te maken voor main.css (zie tekening) aan   
 server

– de server antwoord (response) met css file  
– en de browser zet the content to text/css

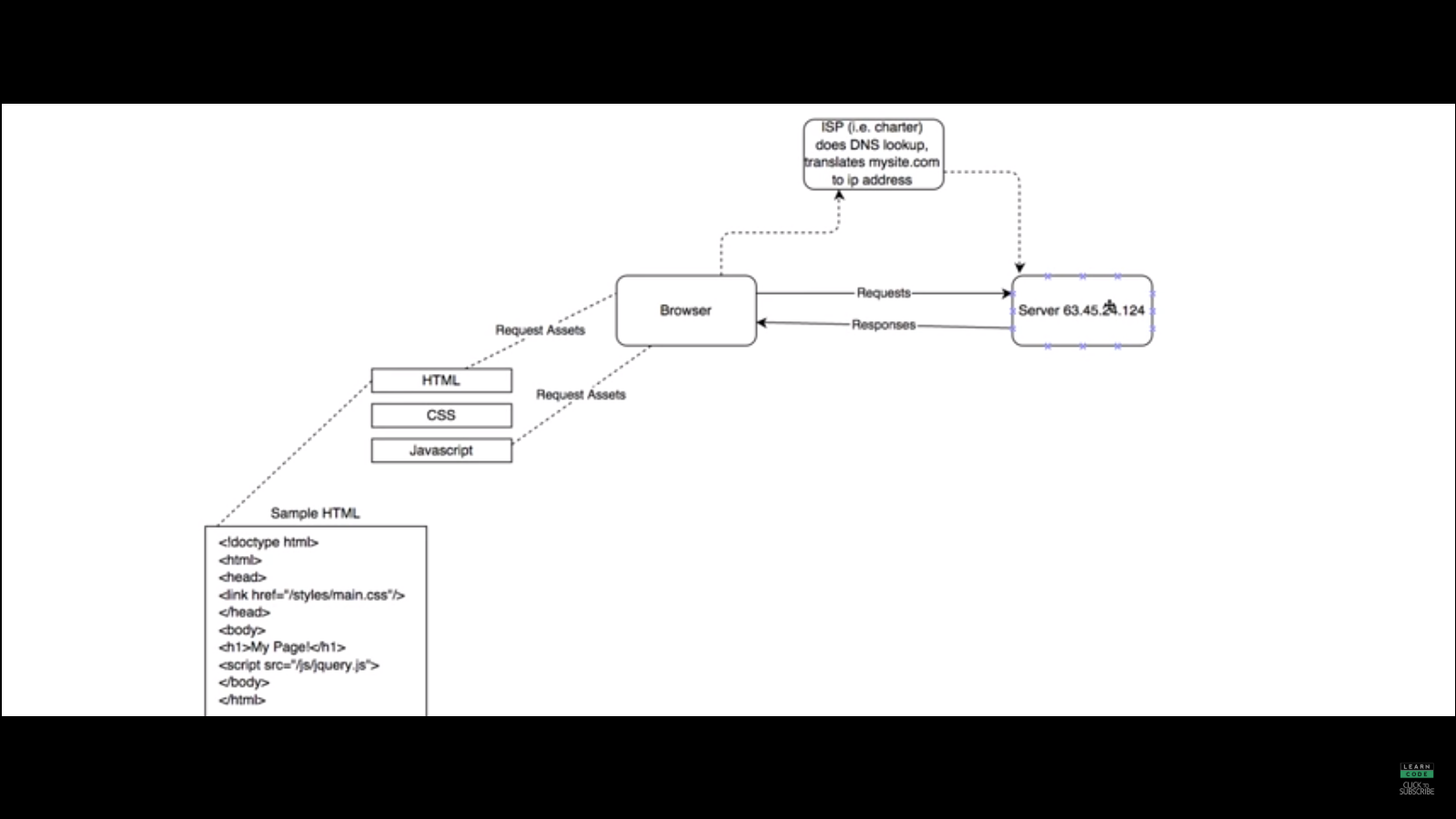
– nu weten we (zien we) dat het een css file is

* **Javascript assets**je kan ook assets request maken met behulp van javascript, als bv op een klinkt klikt dan zegt javascript meteen   
  – maak een request (bij server) voor bv de nieuwste 5 tweets  
  – de response komt terug en ipv naar de html pagina te gaan gaat de response direct naar de   
  javascript file en javascript gaat daar dan iets mee doen.  
    
  Javascript kan requests maken voor;

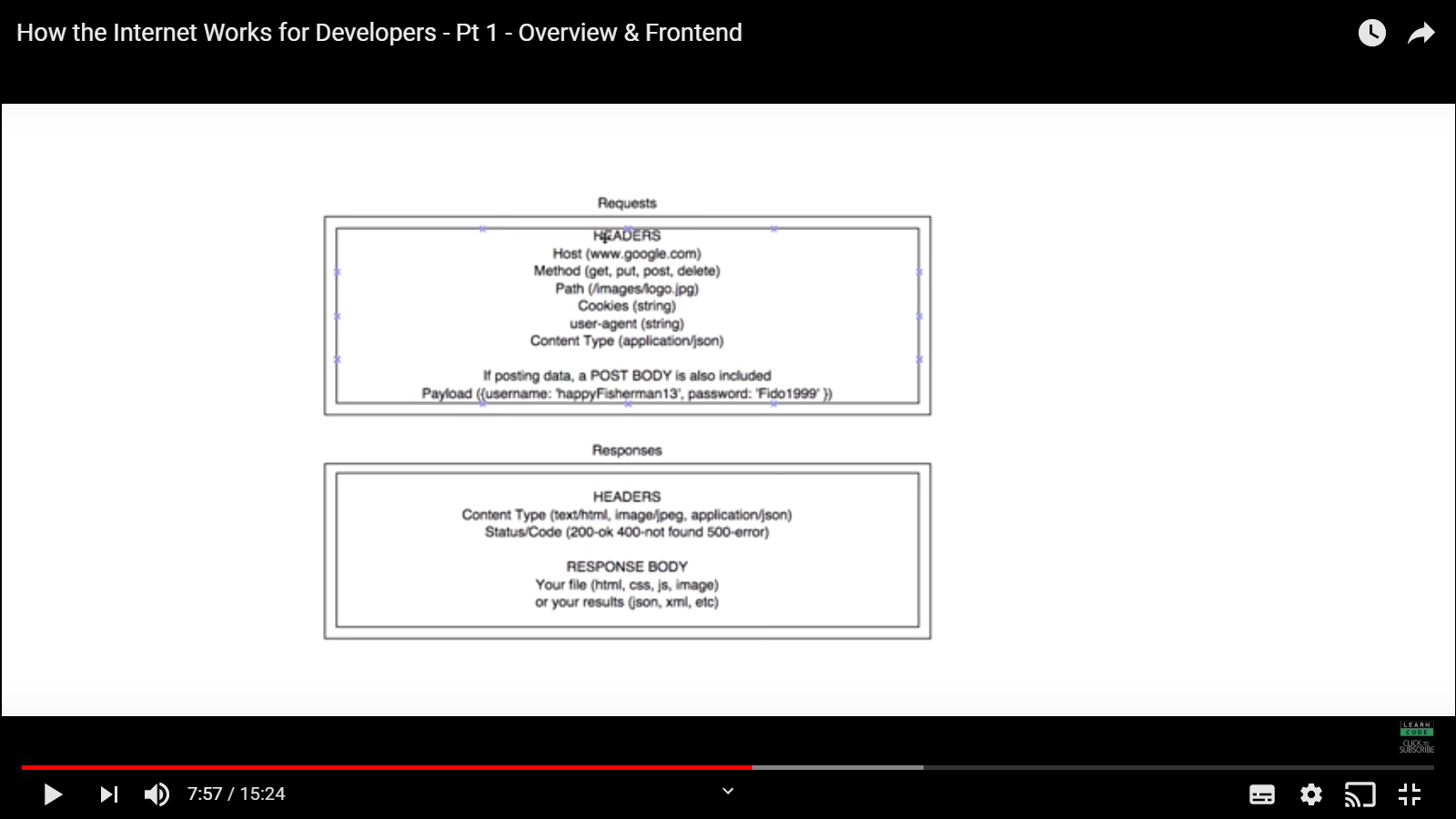
- images  
-voor css files

-voor bijna alles

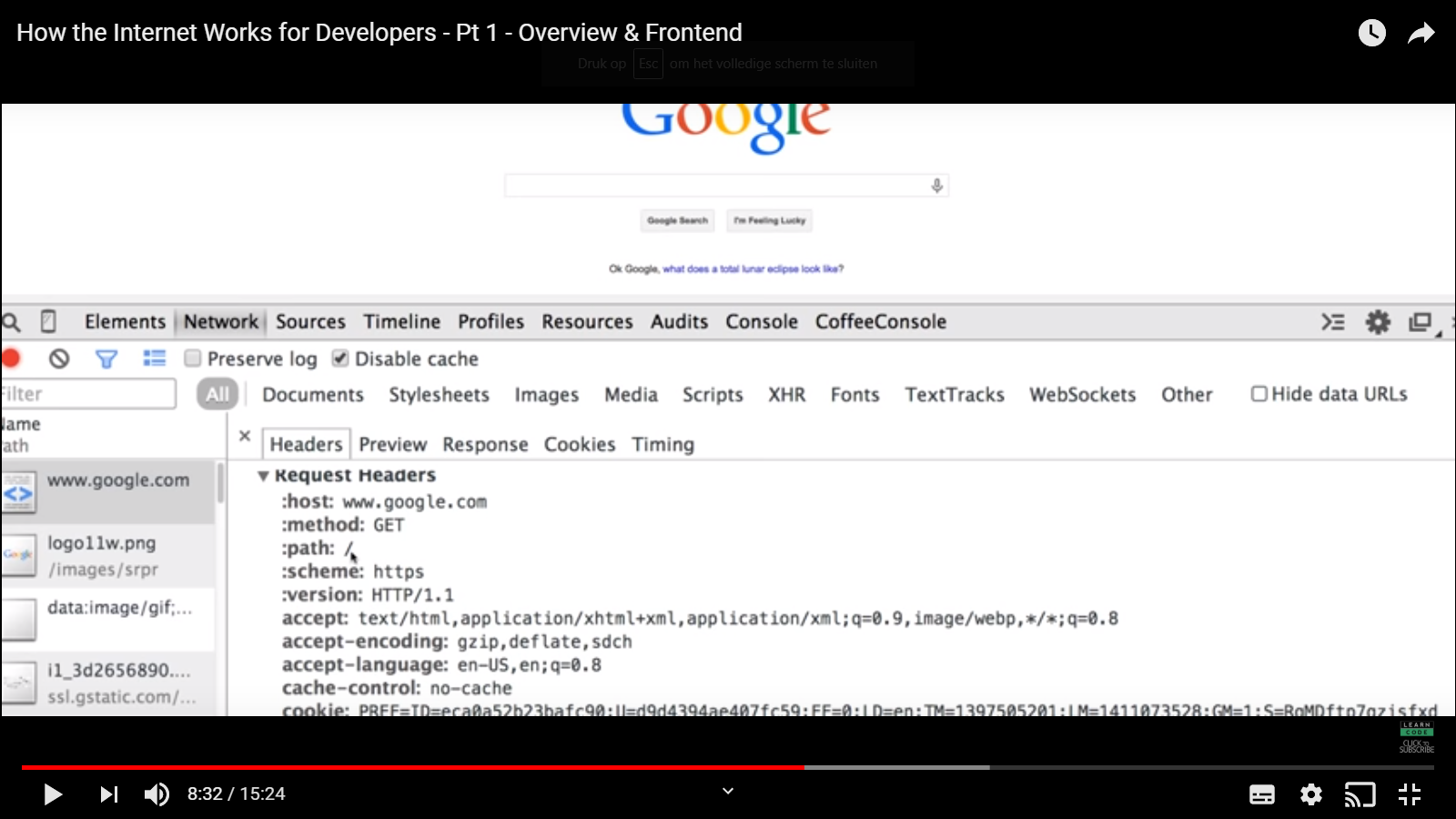
!!want een request is een verzoek aan de server en deze bepaald wat je terug krijgt aan de hand van the content type



* **Een request bevat 2 dingen**



**1 Headers en eventueel een POST body:**



Request Headers

host: [www.google.com](http://www.google.com/)

method: GET  
path: niks, maar bv [www.google.com/**blablabla**](http://www.google.com/blablabla)

cookies: informatie dat de website opslaat op jouw computer

user-agent string: informatie over mijn webbrowser = chrome 37 (zie boven)

Content type: alleen bij een POST body

Wanneer ik bijvoorbeeld een log-in doe dan;  
Method: POST

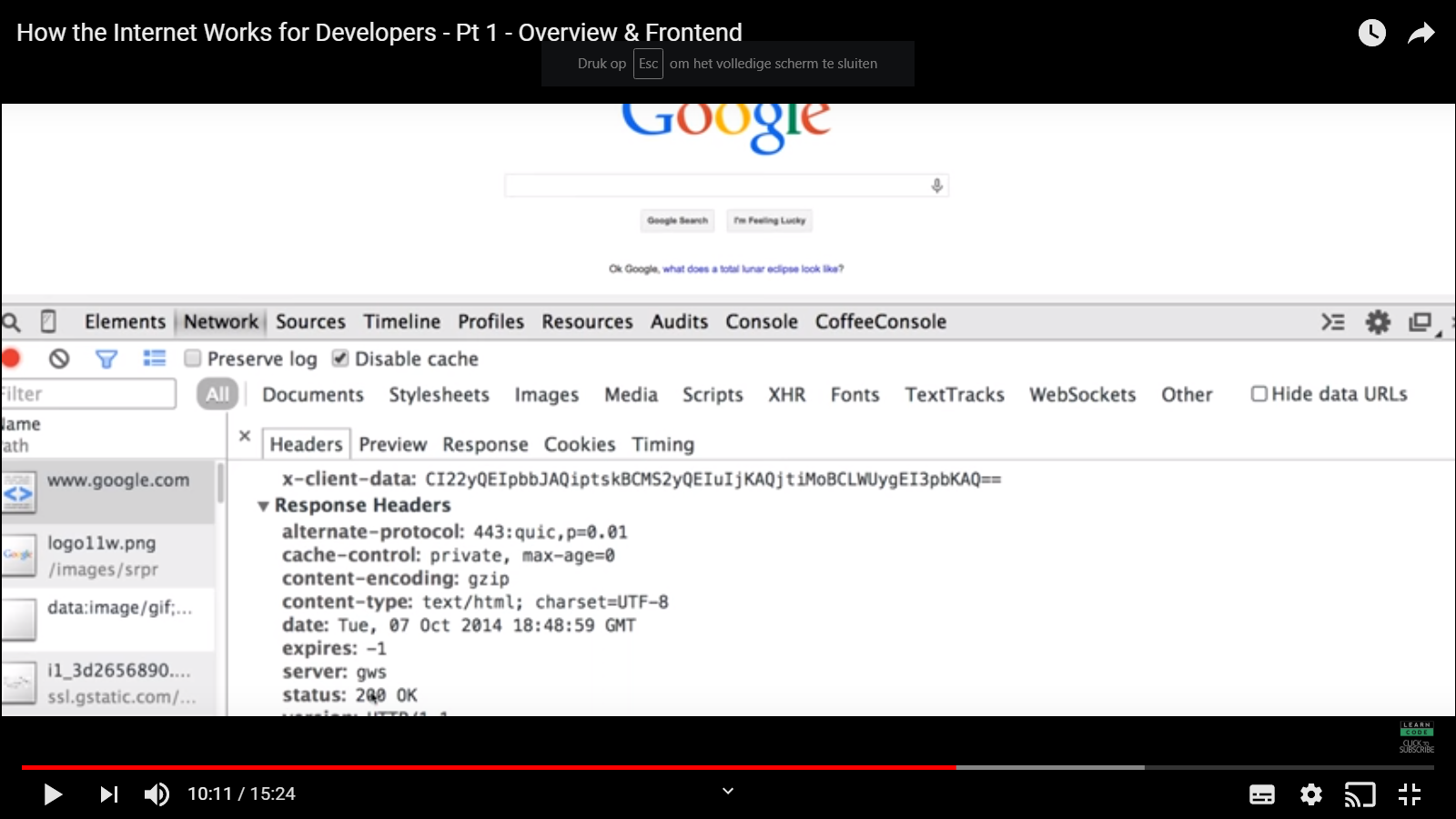
methods kunnen zijn:   
GET: give me a recourse

PUT: added someting / update something

POST: creating something new / log in

DELETE: deleting something

**2 Response Headers**



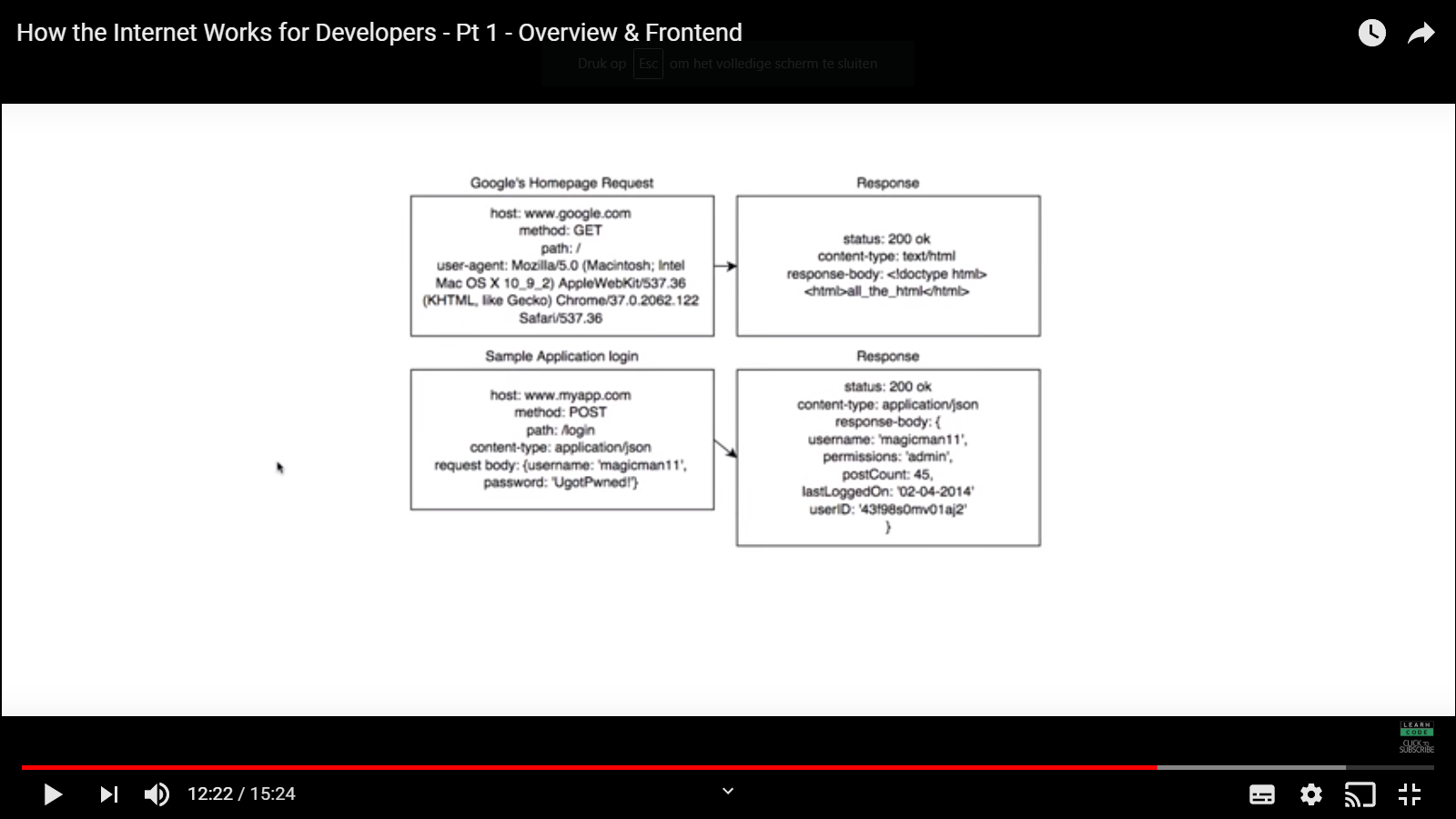
Response Headers

-content-type: = text/html (zie vb boven)

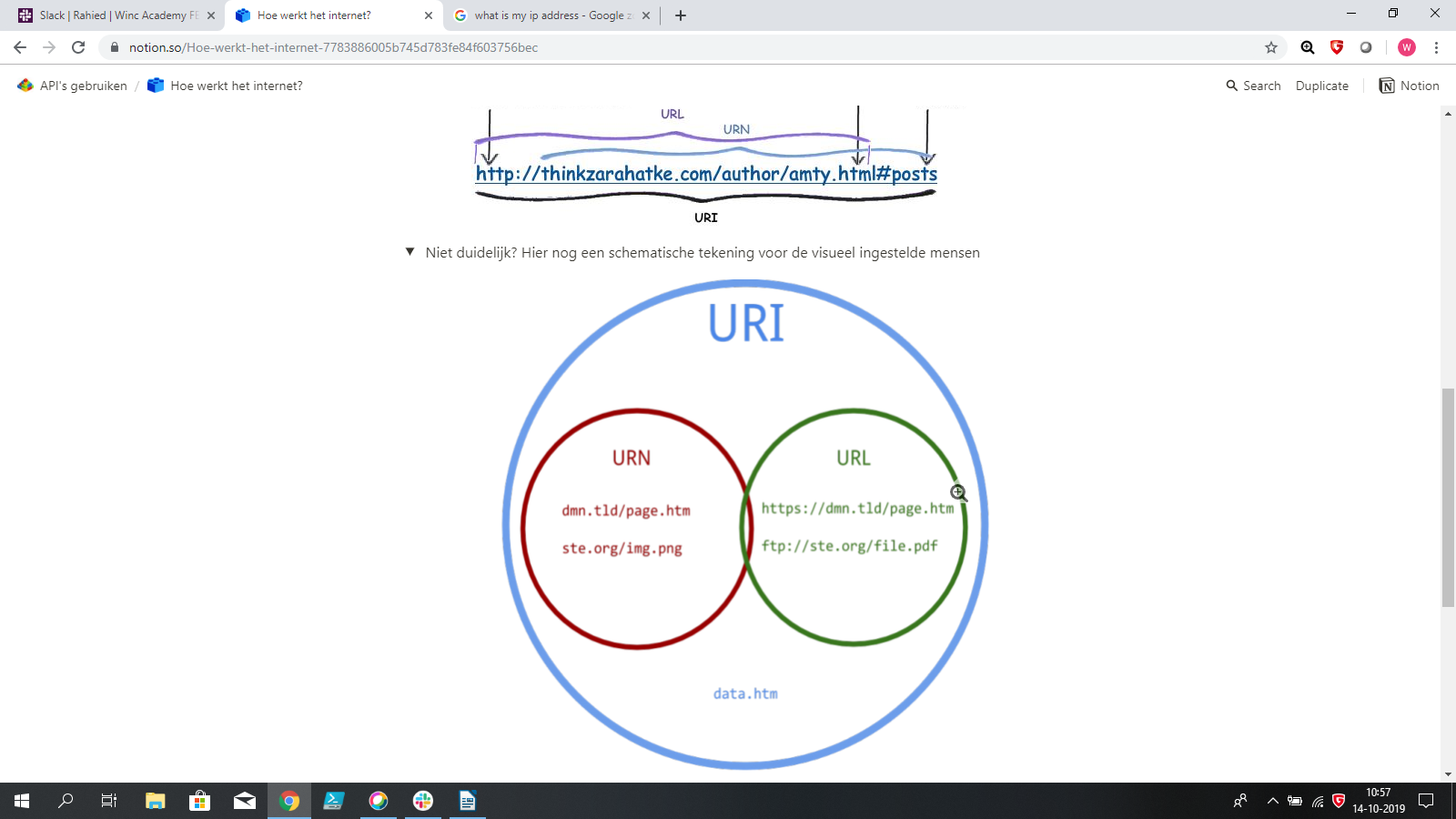
-Status code;

200 = goed  
 300 = redirected  
 400 = error  
 500 = server broke up/blown-up

**Status code erg belangrijk wanneer je iets met javascript doet**

****

* **Ik begrijp wat het verschil is tussen URI, URN & URL**

****

**URI =** Een uniform resource identifier (**URI**), is een internet-protocolelement.   
Het is een unieke benaming van een "bron", een stuk informatie, data of dergelijke

**URL =** Een **Uniform Resource Locator** (afgekort **URL**) is een gestructureerde naam die verwijst naar een stuk data. Voorbeelden zijn het unieke adres waarmee de locatie van een webpagina op internet wordt aangegeven of een emailadres . In de naam is alle informatie opgenomen over de benodigde techniek om de betreffende gegevens te bereiken. De URL is een bijzondere vorm van de URI.

**URN = Uniform Resource Name (URN)** is een naam van een computerstandaard op het World Wide Web.

is bedoeld om als persistent, locatie-onafhankelijke identifier te dienen. Het is een combinatie van een identifier (een toegekende code of NID) en een aanduiding van de naamruimte (namespace of NSS). Binnen de naamruimte zijn de identifiers uniek en in een gespecificeerd formaat. De autoriteit die verantwoordelijk is voor de naamruimte ziet daar op toe. Voor URN is het een vereiste dat elke naamruimte-aanduiding wereldwijd uniek is, waarmee ook iedere URN uniek is.

URI, URL en URN lijken op elkaar maar zijn niet gelijk

Leerdoelen**:**

* **Ik begrijp dat een HTTP-statuscode mij informatie geeft over mijn aanvraag (request).**

Yes

* **Ik kan een HTTP-statuscode opzoeken en begrijpen.**

Yes, zie lijst

* **Ik kan uitleggen waar HTTP voor staat.**

Hypertext Transfer Protocol (**HTTP**) is het protocol voor de communicatie tussen een webclient (meestal een webbrowser of een app) en een webserver. Dit protocol wordt niet alleen veel op het wereldwijde web gebruikt, maar ook op lokale netwerken (we spreken dan van een intranet).

Leerdoelen:

* **lk kan uitleggen wat een API is.**

Application Programming Interface, pakt n boodschap van mij brengt deze naar de server, de server antwoord en API brengt dat antwoord weer naar mij

* **Ik kan uitleggen waar de afkorting API voor staat.**

Application = applicatie =  
 Programming = programming

Interface = omgeving

* **Ik kan uitleggen wat de volgende HTTP Methods doen:** 
  + GET =give me a recourse
  + POST = added someting / update something
  + PUT = creating something new / log in
  + DELETE = deleting something
* **Ik begrijp waarom sommige APIs authentication nodig hebben.**

Ter controle, zodat het programma weet of jij wel of geen toestamming hebt voor dat  
 wat je wilt doen

* **Ik begrijp dat er verschillende manieren van authenticatie zijn voor een API.**   
  Ja, veel

**POSTMAN**

Postman is een simpel programma om op een eenvoudige manier HTTP requests uit te voeren op (RESTful) APIs. Postman wordt gebruikt door developers om te kijken of een server reageert zoals verwacht.

Leerdoelen**:**

* Ik begrijp wat Postman doet en hoe het werkt   
  create HTTP request met request builder

done

* Ik begrijp waarom developers met Postman werken

testing

done

* Ik begrijp hoe ik Postman moet instellen om een HTTP request te kunnen doen

bijna

done