Internet en http

In deze les wordt kort de werking van "het internet" toegelicht, vanuit het oogpunt van onze browser.

De inhoud van dit vak gaat vooral over de client kant (lees : de browserkant) van het internet.

- Hoe er met de server kant gecommuniceerd wordt (deze les)
- Hoe we met Javascript de inhoud, visualisatie en gedrag van webpagina's kunnen aanpassen eenmaal ze door de browser zijn ingeladen.

Gebruikte software tools downloaden en installeren

We gaan er in deze cursus van uit dat je Chrome als browser gebruikt, al je oplossingen dienen op Chrome te werken.

- Download en installeer Google Chrome via https://www.google.com/intl/nl/chrome/

Verslag

In dit deel van de cursus staan verschillende vragen die je moet beantwoorden en opdrachten om iets te maken of uit te proberen. Het is belangrijk dat je alle opdrachten zorgvuldig uitvoert!

Documenteer je werk in een verslag document 'verslag internet en http.pdf' waarin je

- voor elke 'uitprobeer' opdracht een entry maakt met screenshots ter staving van wat je deed
- je antwoorden op de gestelde vragen neerschrijft

Indien nodig bewaar je oplossingen van grotere opdrachten (bv. met veel code) in een .zip file.

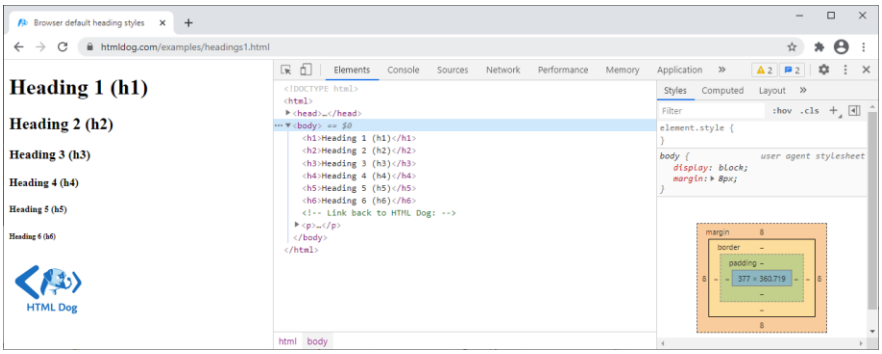
Chrome Developer Tools

Chrome biedt een heleboel informatie over een ingeladen webpagina en hoe die door de browser wordt verwerkt.

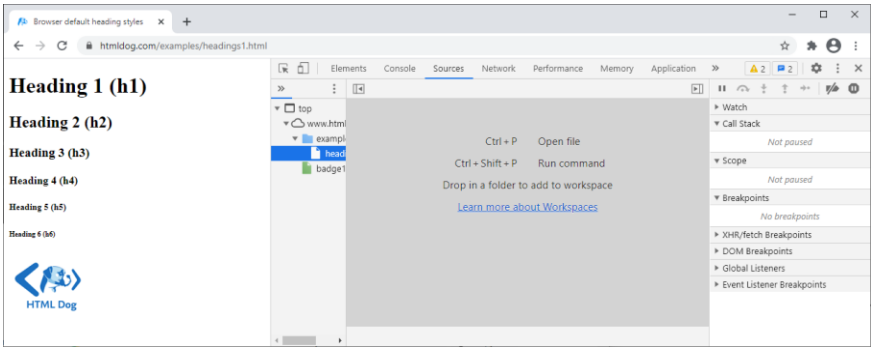
Dit gebeurt via de **Chrome developer tools**, die je kunt zichtbaar maken door op **F12** te drukken als de pagina is geopend in de browser.

Ga na wat de mogelijkheden zijn via [https://developer.chrome.com/devtools.](https://developer.chrome.com/devtools)

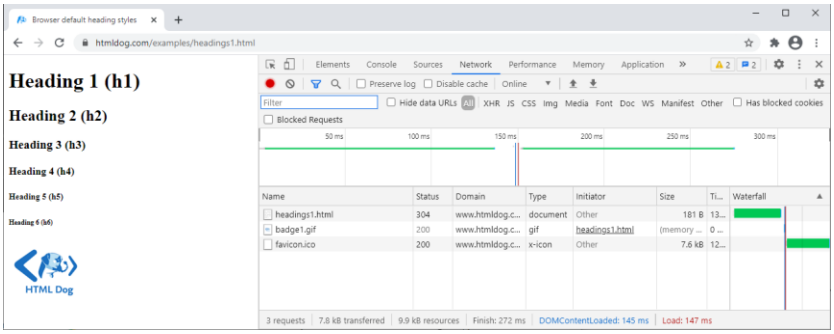
Wij zullen alleszins het 'Elements' tabblad gebruiken voor HTML & CSS



en het 'Sources' tabblad voor Javascript debugging.



In dit deel van de cursus gaan we echter dieper in op het 'Network' tabblad.



Dit tabblad toont ons namelijk welke netwerkcommunicatie de browser uitvoert bij het openen van een pagina.

HTTP – hypertext transport protocol

Telkens je met een browser naar een webpagina surft, wisselt de browser informatie uit met de webserver die de pagina moet aanleveren.

Het protocol waarmee deze beiden communiceren is het HTTP (hypertext transport protocol), waarbij de browser de rol speelt van HTTP-client en de webserver de HTTP-server is.

De communicatie tussen beiden bestaat uit een of meerdere **request** berichten (verstuurd door de browser) en met telkens (hopelijk) een bijbehorend **response** bericht van de server. Het HTTP legt vast welke soorten berichten er mogelijk zijn en wat ze betekenen. De berichten zijn tekstgebaseerd, we zullen ze verderop eens bekijken.

Opdracht 01

Zoek op het internet naar afbeeldingen op basis van de zoektermen **http client server**. Teken hieronder een schematische voorstelling van de communicatie tussen een client en een server. Voor computers gebruik je rechthoeken, voor berichten pijltjes en in het midden teken je een mysterieus wolkje dat 'het internet' voorstelt. Zorg ervoor dat de volgende termen erop aangeduid staan : client, server, request, response, internet, tekst en geef met getallen de volgorde van de berichten aan.

Elke request die de browser naar de server stuurt bevat een URL waarmee een resource geindentificeerd wordt (bv. een document, een video, een afbeelding, ...) waarin de browser geinteresseerd is. De respons wordt teruggestuurd op basis van het IP-adres van de client computer.

Voorbeeld

Een URL zou er zo kunnen uitzien

http://documents.example.com:8080/en/read?id=123&highlight=internet#section1

Deze URL bestaat uit de volgende onderdelen (de meesten zijn niet verplicht) :

- protocol : http
    - dit is het protocol dat gebruikt moet worden om de resource op te vragen. Een protocol beschrijft de spelregels van de communicatie en de toegelaten berichten. Het verkeer op het internet gebeurt op basis van veel verschillende protocollen, bv. SMB voor fileshares, IMAP voor email, HTTP om documenten uit te wisselen, ... Je browser ondersteunt veel verschillende protocollen, maar bij het dagdagelijkse browsen is dit doorgaans HTTP.
- host : example.com
    - de host naam is doorgaans een domeinnaam, maar kan bv. ook een IP-adres zijn.
    - je computer gebruikt het IP-adres van de server voor de eigenlijke communicatie, dit adres wordt opgezocht middels een DNS-lookup van de hostname.
- subdomain : documents
    - een subdomein is een techniek waarmee de beheerder van de domeinnaam, de inkomende berichten gericht naar bepaalde servers kan sturen
- port : 8080
    - een poortnummer identificeert voor welke service op de server het bericht bedoeld is. Een service is niks anders dan een programma dat op de server draait en luistert naar de inkomende trafiek voor een specifieke poort. Eenzelfde server kan immers vele rollen vervullen : fileserver voor network shares, een streaming server, een webserver, een databank server, etc.
- path : en/read
    - dit identificeert een bepaalde resource op de server. Vaak wordt hier een logische indeling gebruikt die verband houdt met iets uit 'de realiteit', bv.
      /nl/burgerzaken/identiteit/attest-bevolkingsregister/attest-van-hoofdverblijfplaats
      identificeert het document op de ‘stad.gent’ website met informatie over hoe je een ‘attest van hoofdverblijfplaats’ kunt bekomen van de dienst ‘burgerzaken’.
- parameters
    - parameters worden gebruikt om via de URL informatie mee te geven met de request (merk op dat er nog andere manieren zijn om dit te bewerkstelligen, zoals in de request body)
    - Uit de voorbeeld URL hierboven
        - parameter id heeft waarde 123
        - parameter highlight heeft waarde internet
- fragment : #section1
    - een fragment identificeert een onderdeel van de opgevraagde resource, bv. de sectie voor hoofdstuk 7 in een webpagina met de tekst van een boek. Je browser zal na het inladen van deze pagina automatisch naar hoofdstuk 7 scrollen.

Opdracht 02

Duid de verschillende onderdelen van de volgende URL :

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

Opdracht 03

Probeer zelf eens te achterhalen wat het IP adres is van de host die het 'www' subdomein van het vlaanderen.be domein voor z'n rekening neemt. Open hiervoor een een terminal venster en typ het volgende commando in :

ping www.vlaanderen.be

druk op enter om de opdracht uit te voeren. Wat is het gezochte IP-adres?

Probeer ook eens het commando

tracert southpark.com

uit, je krijgt dan een lijst te zien van de tussenliggende hosts langswaar je berichten voor southpark.com passeren. Al deze tussenstations maken deel uit van het internet wolkje.

Elk bericht dat de browser en server uitwisselen (zie bv. je afbeelding uit opdracht 1), is een tekst waarin we twee delen onderscheiden :

- body
    - dit is de eigenlijke informatie
        - bv. een request body kan de data bevatten die de gebruiker in een invulformulier invulde
        - bv. een response body kan de tekst van de gevraagde webpagina bevatten
- headers
    - dit is meta informatie (i.e. informatie over de informatie)
        - bv. in een request header kan staan in welk formaat de browser de resource kan ontvangen
        - bv. in een response header kan het tijdstip staan waarop de webpagina het laatst werd gewijzigd

Naast de url zal een HTTP-request dus nog veel meer informatie (lees : bijkomende tekst) bevatten.

De server ontvangt deze request en moet op basis van deze informatie beslissen wat zijn response zal zijn. Heel vaak bevat het response bericht ook de inhoud (lees : tekst) van een webpagina, nml. een HTML document.

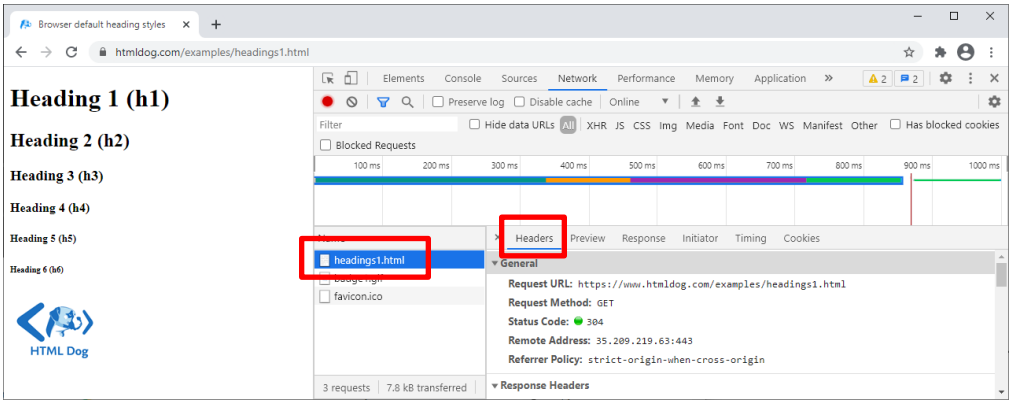
De browser ontvangt de response van de server en moet de bijbehorende informatie dan op een of andere manier presenteren aan de gebruiker in een tabblad.

Opdracht 04

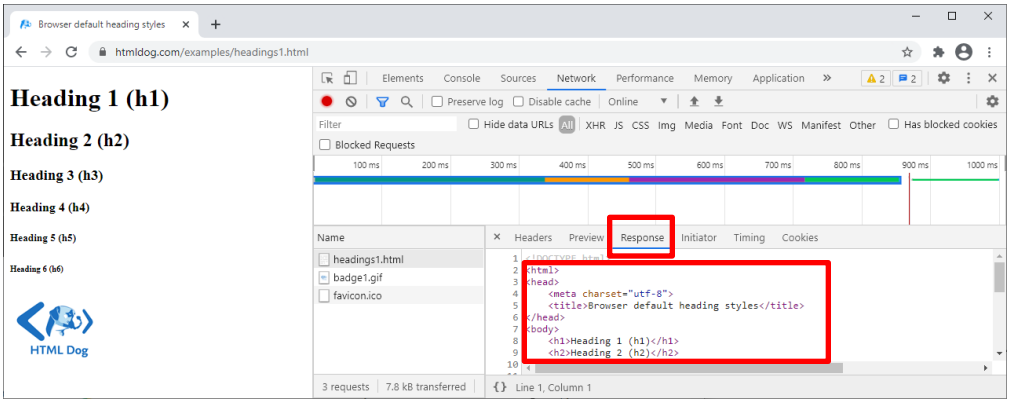
Open de Chrome developer tools (F12), ga naar het network tabblad en surf via de adresbalk naar

http://www.htmldog.com/examples/headings1.html

De browser verstuurt een request voor 'headings1.html' naar de server. Op het 'Headers' tabblad kun je de request en response headers bekijken (bv. Request URL en Status Code) :



Op het 'Response' tabblad kun je de reponse body bekijken :

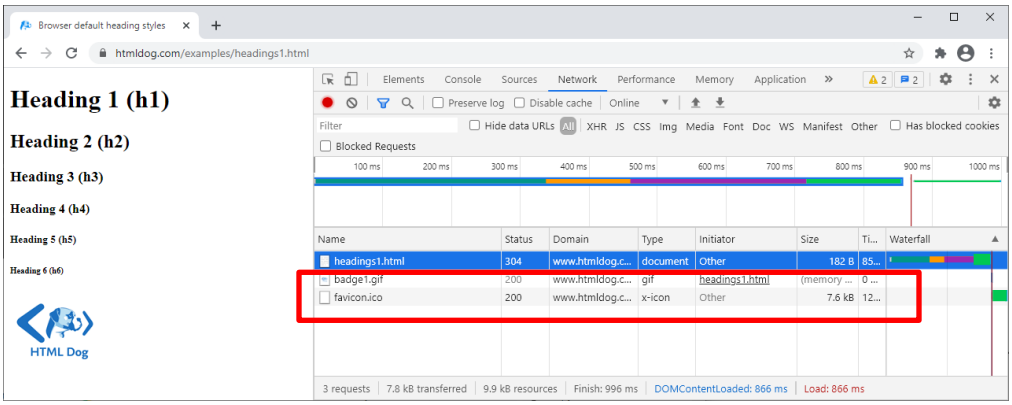


De response body bevat in dit geval de code voor een HTML document dat de inhoud en structuur van de opgevraagde headings1.html resource beschrijft.

Zo'n HTML document bevat naast de eigenlijke inhoud (bv. de tekst van een krantenartikel) heel vaak ook verwijzingen naar andere documenten

- bestanden met layout informatie (CSS bestanden)
- bestanden met uitvoerbare programma code (javascript bestanden)
- afbeeldingen
- andere resources (bv. links naar andere webpagina's)

Al deze bijkomende resources zal de browser via bijkomende requests opvragen :

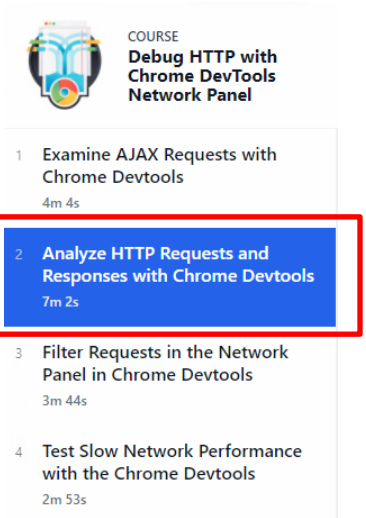


Opdracht 05

Bekijk onderstaande video die de werking van het 'Network' tabblad demonstreert :

https://egghead.io/lessons/tools-analyze-http-requests-and-responses-with-chrome-devtools

Dit is een onderdeel van een lessenreeks waarvan wij enkel hoofdstuk 2 bekijken (07:02 min) :



Opdracht 06

Bij het inladen van de pagina http://www.htmldog.com/examples/headings1.html in het eerdere voorbeeld, heeft de browser meer dan enkel een request voor de 'headings1.html' resource verstuurd.

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

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

(merk op dat browser plugins zoals anti-virus programma's en ad-blockers sommige requests kunnen tegenhouden en ook eigen requests kunnen versturen!)

Opdracht 07

Surf naar

http://www.vlaanderen.be

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

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 waneer de request verstuurd werd en wanneer de response werd ontvangen.

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?

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).

--

Opdracht 08

Open het network tabblad van de Chrome developer tools, surf naar je favoriete webmail client (bv. gmail) en log in.

Wacht tot 'alles' ingeladen is en zorg ervoor dat je niet met de muis over de eigenlijk webpagina gaat (beperk de bewegingen tot het 'Network' tabblad in de Chrome Developer Tools.

Afhankelijk van welke webmail provider je gebruikt, zul je zien dat er na verloop van tijd requests bijkomen ook al verandert er niks aan de webpagina.

Waarvoor zouden die 'spontane' requests dienen?

--

HTTP response status codes

Elke response die de server terugstuurt bevat een status code die aangeeft om wat voor soort response het gaat (alles ok, server overbelast, toegang tot resource verboden, etc).

Opdracht 09

Surf naar

http://www.vlaanderen.be/dezepaginabestaatniet.hml

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

Wat betekent de status code 404 in de response header?

--

Opdracht 10

Probeer nu eens

http://www.ditdomeinbestaatnogniet.be/blabla.html

Wat is het verschil met de vorige opdracht?

--

Opdracht 11

Zoek onderstaande HTTP status codes op (bv. wikipedia) die in een response kunnen voorkomen en schrijf hun betekenis op. Omcirkel deze die je al eens bent tegengekomen bij het surfen. Schrijf ook een gepaste hoofding boven de kolommen.

200	301	400	500
204	302	401	503
	303	404	

HTTP request methods

Elke client request is van een bepaalde soort die met de request method wordt aangeduid.

Opdracht 12

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

Waarvoor dienen de vaak gebruikte GET en POST methods?

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 4)?

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?

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

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?

Dit is trouwens geen hypothetisch geval, zie bv.

- http://thedailywtf.com/articles/The_Spider_of_Doom
- http://thedailywtf.com/articles/WellIntentioned-Destruction
- http://betanews.com/2005/05/06/google-web-accelerator-draws-concern/

--

Caching door de browser

Opdracht 13

Ga naar het network tabblad van de Chrome developer tools en surf naar

http://www.vlaanderen.be

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

--

Caching door de browser

Veel resources wijzigen niet zo vaak, denk bv. aan de afbeelding met het bedrijfslogo die op bijna alle pagina's van hun website voorkomt. Een browser kan veel trafiek vermijden door resources lokaal bij te houden voor toekomstig gebruik i.p.v. telkens dezelfde resource aan de server te vragen. Deze techniek heet 'caching' en is voor iedereen een goeie zaak : de gebruiker krijgt sneller de pagina te zien en spaart bandbreedte, de beheerder van de server spaart bandbreedte, geheugen en CPU cycles op de server.

Opdracht 14

Zorg dat de Chrome developer tools actief zijn. Open dan de vlaanderen.be pagina, rechtsklik op de refresh knop en kies **'Empty cache and hard reload'**.

Hiermee dwingen we de browser om bij het herladen van de pagina z'n cache te negeren en alle resources bij de server op te vragen.

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

Hoe lang duurde het vooraleer alle resources van de pagina waren ingeladen?

Klik nu gewoon op de refresh knop.

Kijk nogmaals hoeveel data er werd verstuurd. Waarom is dit zoveel minder? Laadde de pagina sneller?

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

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

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)

--

Caching door een web proxy

De responses op read-only requests (bv. GET requests) kunnen in principe op een tussenliggende host (ergens tussen jouw browser en de webserver) opgeslagen worden voor een toekomstige raadpleging.

Een dergelijk tussenstation noemt men een "caching proxy server", bv. het Traffic Server product van Apache.

Opdracht 15

Lees het stukje "Understanding HTTP Web Proxy Caching" uit hun documentatie :

https://docs.trafficserver.apache.org/en/latest/admin-guide/configuration/cache-basics.en.html

Beantwoord nu de volgende vragen :

- Is zo'n caching proxy server zinvol bij je thuis? Waarom wel/niet
- Welk nut zou een groot bedrijf of school er van hebben?
- Zou een ISP (Internet Service Provider, zoals bv. Telenet) er baat bij hebben?

Lees nu de stukjes "HTTP Object Freshness" en "Cache-Control header".

- Welke headers in een HTTP-request of response beinvloeden het cachen van een resource?
- Wat is het verschil van een cache-control header in een request en deze in een response?

--

Web analytics en tracking

Web analytics is de techniek om informatie over de websurfers te verzamelen die je site bezoeken, om deze informatie vervolgens te analyseren om bepaalde aspecten te optimaliseren (bv. meer verkopen).

Deze statistieken worden verzameld door javascript programma's die met de pagina worden meegeladen, de gewenste data capteren en daarna deze data naar een server doorsturen om opgeslagen te worden voor verdere analyse.

Men houdt bv. bij over welke browser het gaat, welke versie, welke schermgrootte, op welke pagina arriveerden de bezoekers op de website, welk pad doorheen de website hebben ze gevolgd, hoeveel bezoekers zijn terugkerende bezoekers, etc.).

Op basis van de verzamelde statistieken kunnen dan onderbouwde beslissingen worden genomen om het gekozen aspect (bv. meer verkopen) te optimaliseren.

Enkele bekende aanbieders zijn google analytics en statcounter.

http://www.google.com/analytics/
http://statcounter.com/

Deze aanbieders verzamelen statistieken over miljarden webpagina bezoekjes via de websites van hun hun klanten.

- statistieken over de bezochte website
  populairste landing pages, vaak gevolgde paden doorheen hun website, land van bezoeker, ...
- statistieken over de browsermogelijkheden van de bezoeker
  browser, versie, schermgrootte, ...

Elke klant krijgt natuurlijk enkel gedetailleerde informatie over hun eigen website bezoekers te zien.

Men stelt echter vaak ook globale statistieken ter beschikking over de browsermogelijkheden van bezoekers, bv. eens per maand. Deze zijn handig om te beslissen welke mogelijkheden je kunt gebruiken. Bv. is het nog nuttig om extra moeite te steken in work-arounds voor een bepaalde oude browser versie, op welke minimale schermbreedte je moet mikken met je website, ..

Een voorbeeld vind je op

http://gs.statcounter.com

Let er wel op dat het beter is om je eigen statistieken te tracken, want jouw doelpubliek is niet noodzakelijk hetzelfde als de gemiddelde websurfer.

Opdracht 16

Wat moet je aan je webpagina's veranderen om data te capteren met google analytics?

Wat moet je aan je webpagina's veranderen om data te capteren met statcounter?

Open de demo van statcounter.com, bekijk alle beschikbare pagina's met data (linkerkolom).

Welke data wordt gecapteerd voor onderstaande begrippen

- entry page, exit page
- visitor path
- came from
- keyword analysis

(mocht het je niet meteen duidelijk zijn, kun je deze begrippen ook opzoeken op hun 'features' pagina)

--

Opdracht 17

Zoek in de Chrome Web Store naar de Ghostery extension en installeer deze in Chrome. Bezoek nu de volgende webpagina's

- http://www.nieuwsblad.be
- http://www.cnn.com
- http://www.vlaanderen.be
- https://www.vrt.be/vrtnws/nl/

Schrijf bij elke webpagina hoeveel tracking scripts door Ghostery worden ontdekt.

Welke soorten tracking scripts je bent tegengekomen (beacon, advertising, analytics, etc.)?

Na afloop mag je deze extension weer verwijderen, we zullen ze dit semester niet meer gebruiken. Of je kunt ze eens een paar weken laten draaien en je verbazen over wat er allemaal wordt meegeladen met een doorsnee webpagina 😉

Opdracht 18

Zorg ervoor dat je browser venster gemaximaliseerd is en surf naar

http://www.starbucks.com

Maak nu je browser venster gaandeweg minder breed en let erop hoe de inhoud van de webpagina niet enkel opschuift maar ook daadwerkelijk verandert!

Deze techniek heet **responsive design** en gebruikt bv. media queries om bepaalde CSS-regels wel/niet toe te passen afhankelijk van o.a. de venster afmetingen.

Doorgaans verschuiven elementen en/of verkleinen ze, maar op sommige breedtes is er echter een abrupte overgang naar een andere layout. Neem bij elke abrupte overgang een screenshot (bv. via het Snipping Tool als je met Windows werkt).

Hoeveel verschillende layouts tel je?