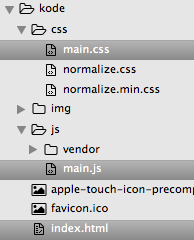
# Lag ditt første webkart

KODE og dokumentasjon: http://goo.gl/aFECjB

## Hva består en webside av?

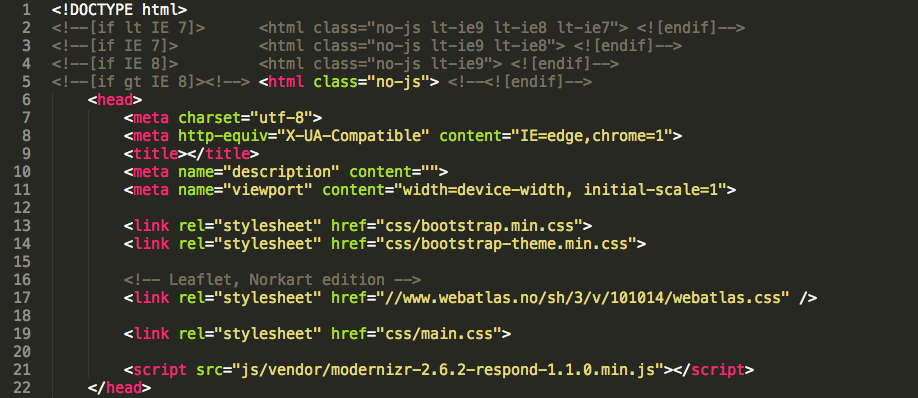
En webside består generelt av 3 komponenter:

1. HTML som beskriver komponentene til websiden
2. CSS som beskriver stilen til HTML-komponentene
3. Javascript som er programmeringsscript som kjøres og kan gjøre websidene interaktive og dynamiske



Filen *index.html* er ofte utgangspunktet til en webside. Denne består av:

«header». Toppen av filen. Her laster vi inn stilfiler og setter opp de øverste HTML-komponentene. Det er lurt å bruke en mal på oppsettet av et webprosjekt. Vi bruker HTML5 Boilerplate med Bootstrap i dette tilfellet. (<https://html5boilerplate.com/>)



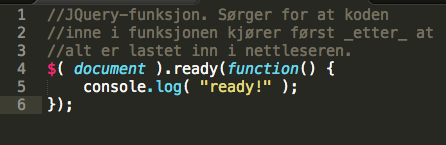
«body», eller «kjøttet» i HTML’en. Her ligger selve sidestrukturen. Vi lager et DIV-element som vi skal bruke senere til å plassere kartet i. Merk deg ID’en til DIV-elementet – det skal vi bruke senere.



«footer» er slutten av HTML-filen. Her er det lurt å laste inn Javascript-filene. Dette sørger for litt raskere sidelasting. Merk at vi laster inn kartbiblioteket (Webatlas.js) fra Norkart sine servere, mens de andre kommer fra lokale filer.



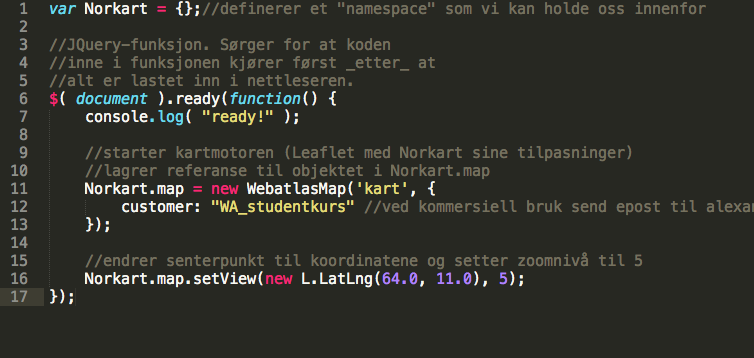
Under mappen js/ legger vi alle de lokale Javascript-filene våre. Her ligger main.js som vi skal bruke for å skrive koden vår til å sette opp kartet. Merk at vi pakker inn all kode i en JQuery-kode på denne måten:

**

Dette er en metode fra JQuery som gjør at vi venter med å kjøre koden våre til alle kodebibliotekene er lastet inn i nettleseren og faktisk klare til å kjøres.

## Kart på en-to-ferdig.

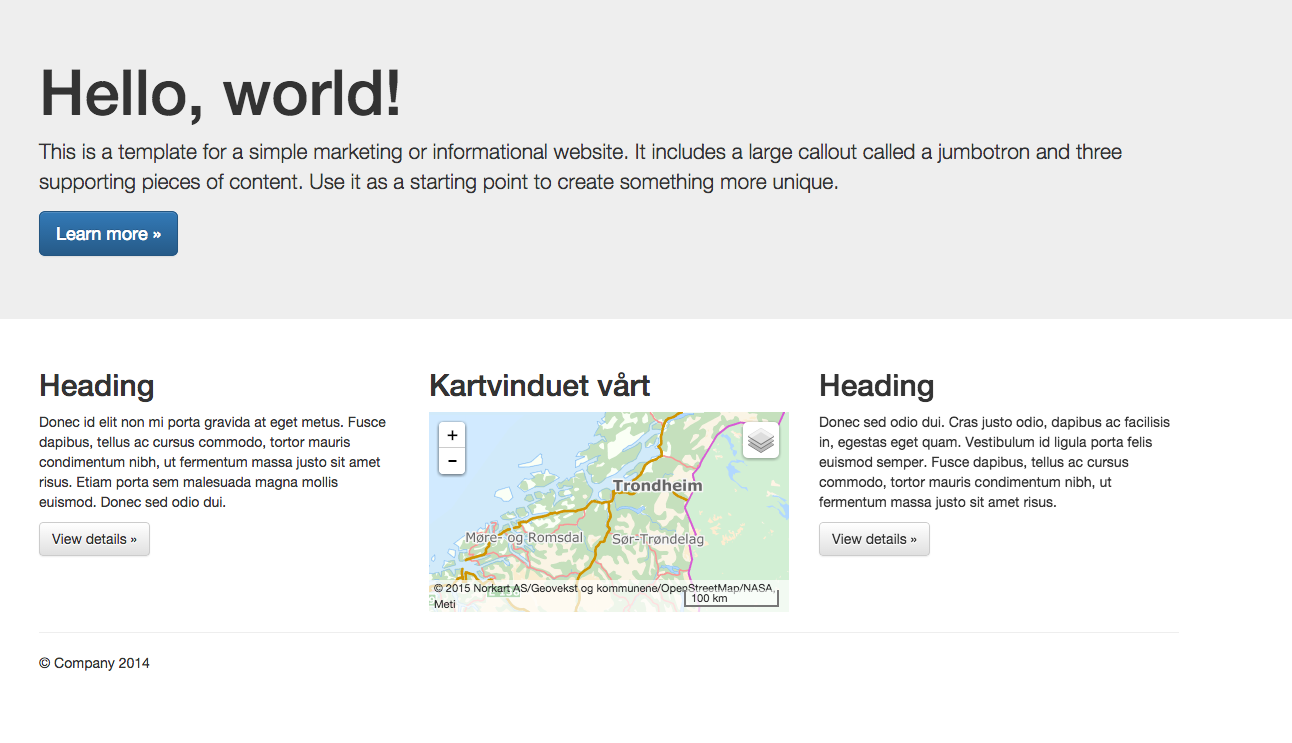
Kartet starter vi med et par linjer kode:



Kartvinduet henger seg på det DIV-elementet med samme id som vi spesifiserte. Det er ingenting i veien for å ha flere kartvinduer i samme nettside hvis man vil det.

Bredden og høyden på kartvinduet er definert i stilfilen (css’en). Den finner vi under /css/main.css. Her kan vi legge inn stiler for å endre på bredde/høyde på kartet og utseende til alle andre elementer på siden.



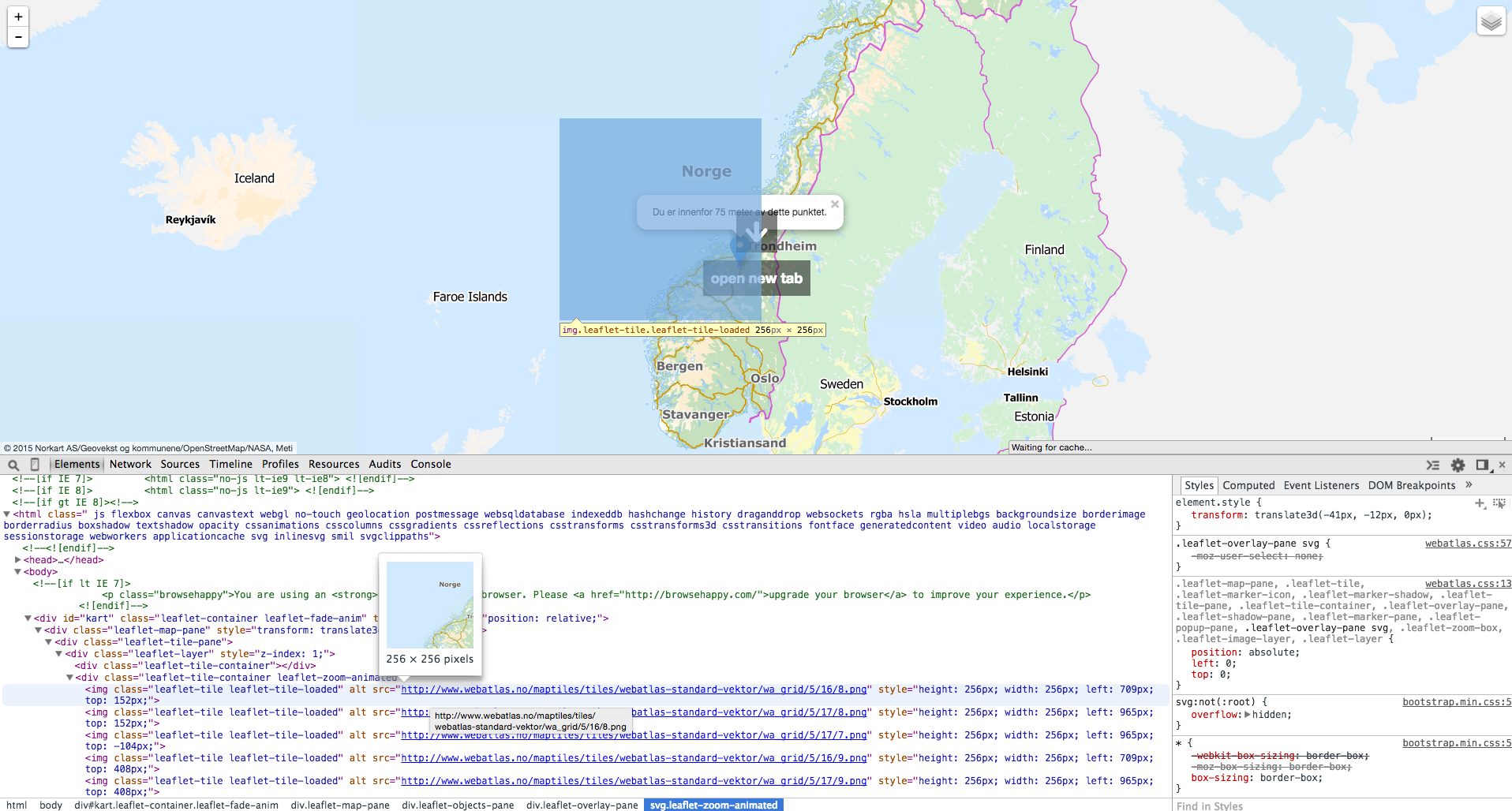


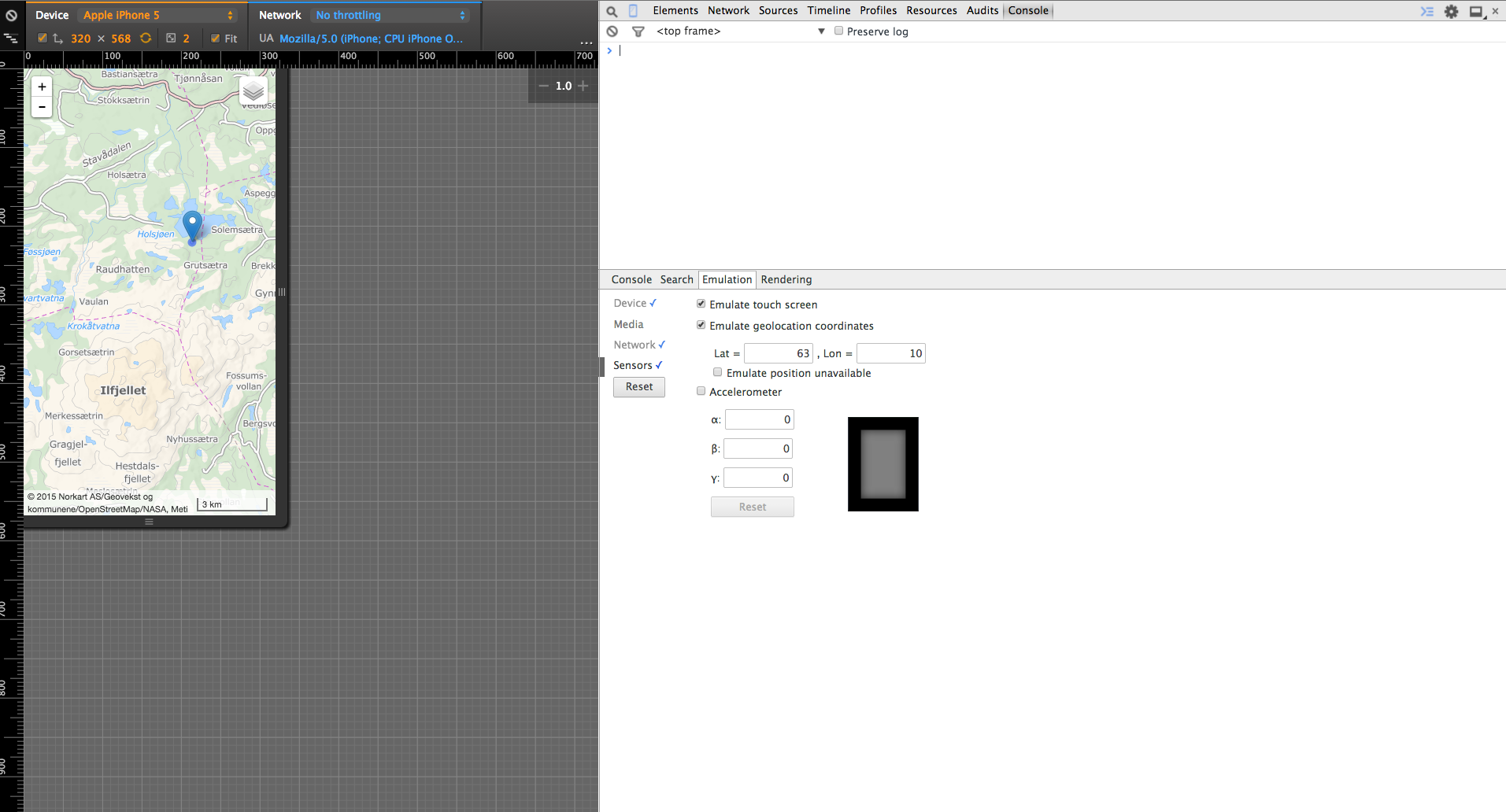
## Debugging og mobilsimulering i Chrome

F12 åpner ”developer tools” i de fleste nettlesere. Chrome tilbyr en ganske avansert form for utviklerverktøy som gjør det enklere å drive med webutvikling. I tillegg til å inspisere elementer på nettsiden, endre html og css ”live” og se på console for Javascript, kan den også simulere mobilenheter. Det kan være veldig nyttig, spesielt for å simulere posisjonssensor, men også andre sensorer og skjermstørrelser.

I Javascript kan vi skrive til console ved å skrive koden: *console.log(”teksten din”);*

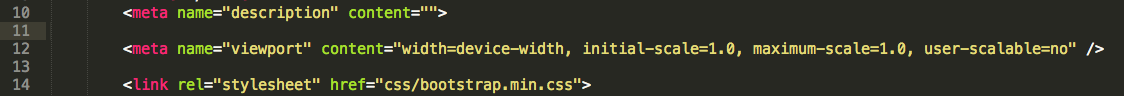
Du kan også debugge direkte i Javascript-koden. Prøv deg frem selv!

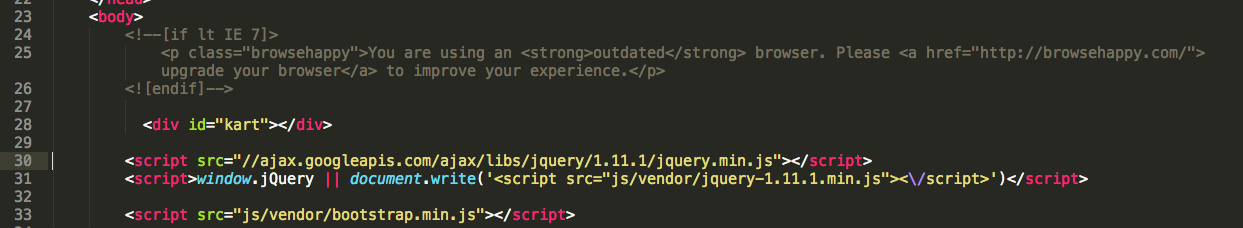




## Mobiltilpasset

Leaflet.js og dermed Webatlas.js er designet for å være responsive. Dette betyr at det tilpasser seg rimelig bra ulike skjermstørrelser og enheter. For å lage et mobiltilpasset kart trenger vi kun å lage et ”fullskjermskart”. Vi fjerner en god del HTML som vi ikke trenger lengre og endre på litt css.



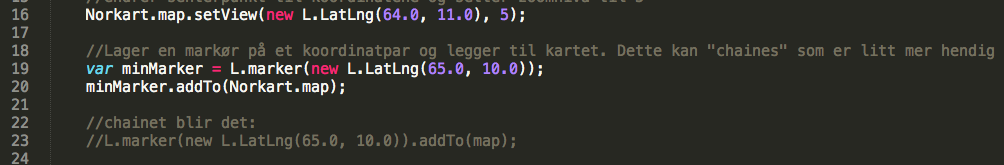




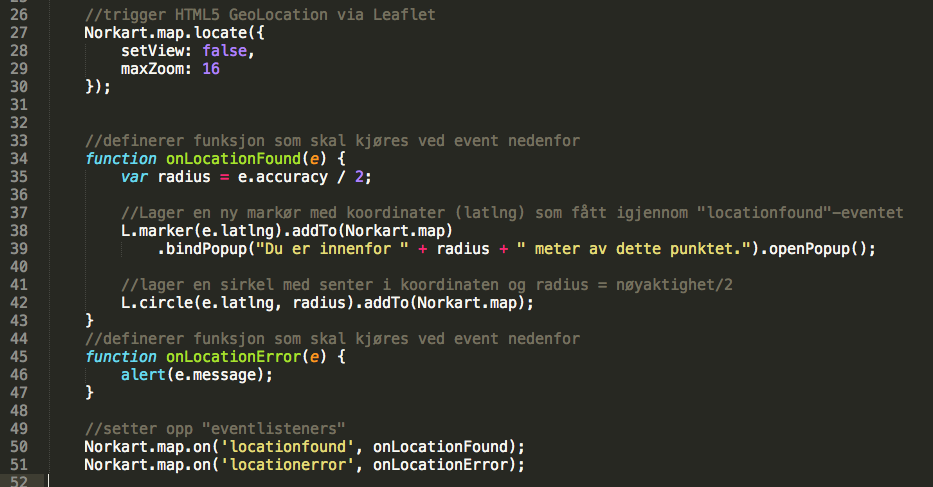


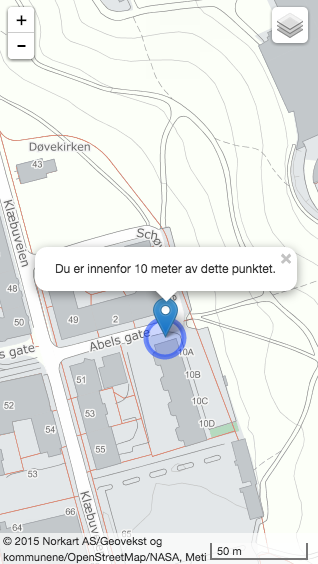
## Hente posisjon fra HTML5 GeoLocation og vise markør i kartet

Vi starter med å lære hvordan lage en enkel markør i kartet på en gitt posisjon:



Vi kan bruke HTML5 GeoLocation for å finne posisjonen til brukeren. Merk at GeoLocation API’et er avhengig av nettleserstøtte og støtte på enheten. Hvis GPS ikke er tilgjengelig brukes WiFi-posisjon eller nettverksposisjon via Google sine tjenester. GeoLocation har også en metode som ”lytter” på oppdateringer. Denne kan brukes til å lage en GPS-tracker. Se her for mer informasjon: <http://leafletjs.com/reference.html#map-locate-options>





## OPPGAVE:

Slå på ”watch” på locate og lytt på endringer på posisjonen.

Lag en ny markør for hver endring.

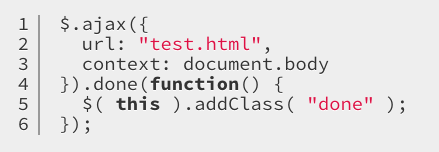
**Mer avansert**:

Flytt den eksisterende markøren dynamisk etter endringene og tegn en strek mellom punktene.

## Last inn data asynkront

AJAX er en metode for å hente inn data asynkront med sidelastingen. Dette er metoden som sørger for at man slipper å ”refreshe” en side for å hente ny data (for eksempel nye meldinger på Facebook). JQuery har enkle metoder som skjuler mye kompleksitet i dette. Den som er mest avansert er $.ajax() hvor dokumentasjonen er her: <http://api.jquery.com/jQuery.ajax/>

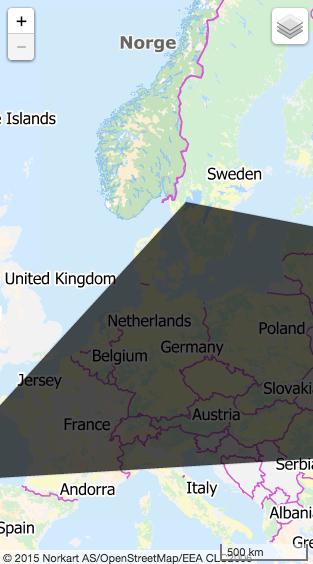
Da kan et kall se slik ut:



I tillegg finnes det andre ”shorthands” som $.get og $.getJSON som er hendige å bruke



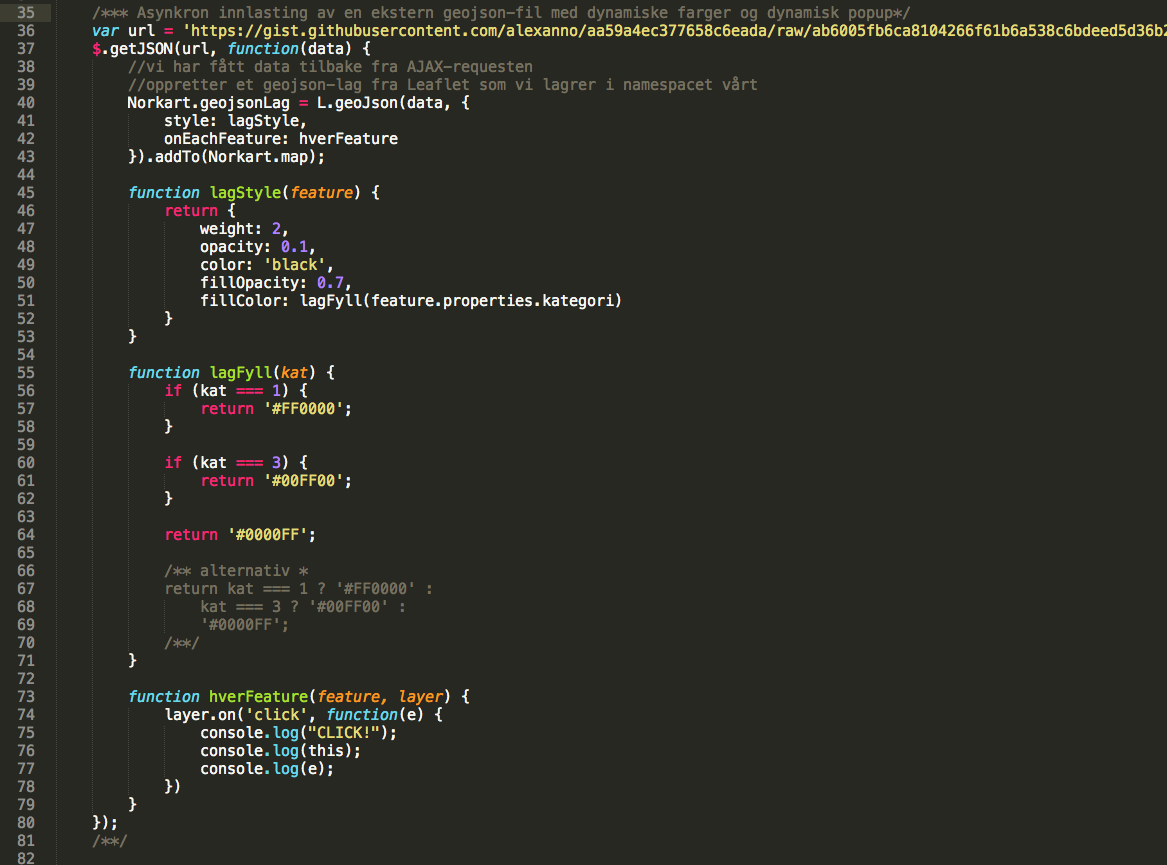
Vi skal laste inn en GeoJSON-fil som er lagd ved hjelp av <http://geojson.io/> og lagret lokalt som /test.geojson



Urlen som det spørres etter kan fint være fra en ekstern kilde – og kan være en tjeneste som returnerer data direkte fra fks en database. Merk at det i utgangspunktet ikke er lovlig å sende AJAX-requests på tvers av domener – unntatt hvis de støtter CORS. Dette er det flere og flere som gjør. Alternativt kan man bruke en CORSproxy, fks: [www.corsproxy.com/](http://www.corsproxy.com/)

## Dynamisk styling og litt interaksjon

Styling av objekter kan gjøres dynamisk ved å bruke funksjoner. I tillegg kan man også hekte på en funksjon som kjøres for hver feature. Prøv å klikk på polygonene og se i console hva som skjer.



Oppgave:

Lag en popup for hver feature og åpne popupen ved klikk.

Avansert:

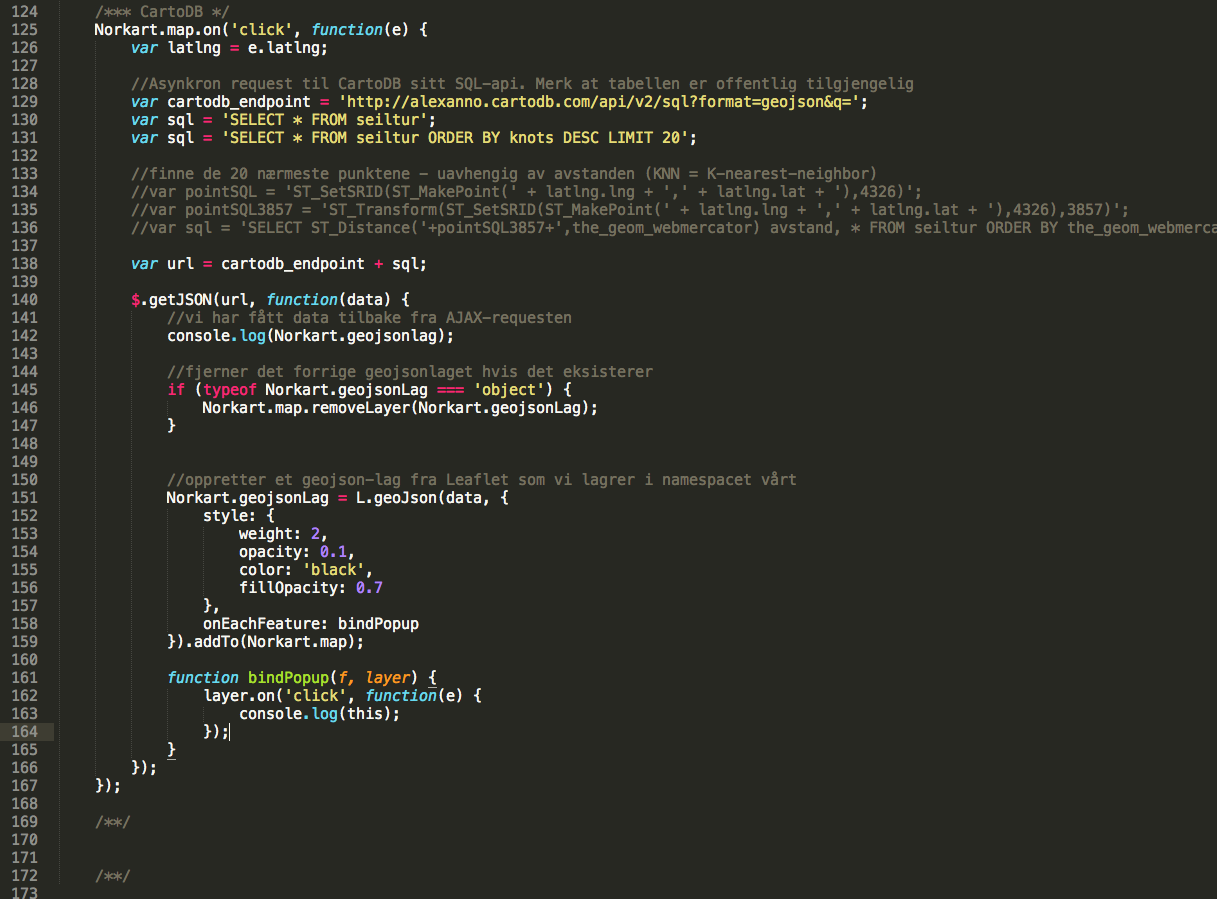
Zoom kartet til ”bounds” av det objektet som er trykket på. Gjør det samme når musen ”hover” over objektene.

Prøv også å lytt på ”mousemove” og lag en markør der hvor det ble klikket med popup som inneholder koordinatene. (tips: <http://leafletjs.com/reference.html#events> )

## CartoDB (postgis)

CartoDB er en skytjeneste som i all hovedsak tilbyr lagring i PostGIS med visualiseringsverktøy på toppen. Dette betyr at vi enkelt kan leke oss med funksjonalitet i PostGIS uten å drifte vår egen server. PostGIS er den kraftigste romlige databasen som bygger på PostgreSQL. Begge deler er OpenSource og tilbyr svært kraftige funksjoner. CartoDB tilbyr enkel integrasjon med Leaflet og dermed også Webatlas.js via cartodb.js (<http://docs.cartodb.com/cartodb-platform/cartodb-js.html>)

Her er et eksempel på hvordan vi kan hente ut data basert på en spørring direkte fra interaksjon i kartet. Mer om SQL her: <http://postgis.net/docs/>



Prøv deg frem med ulike spørringer og last opp dine egne datasett på CartoDB.

Noen tips hvis du vil ta ut data fra OpenStreetMap basert på ogr2ogr:

ogr2ogr -f "geojson" pubs\_restaurant\_norway.geojson norway-latest.osm.pbf -sql "select \* from points where IN ('pub', 'restaurant', ‘brewery’)" -progress

Lykke til!

Alexander Salveson Nossum ([alexander.nossum@norkart.no](mailto:alexander.nossum@norkart.no) | @alexanno)