Oppgave 11 - Prosjekt Redegjørelse

Filer i prosjektet

HTML

arbeidsledighet.html
favoritt.html
index.html
lekeplasser.html
toalettsok.html

CSS

stilark.css

JS

arbeidsledighet.js
favoritt.js
felles.js
lekeplasser.js
sok.js
toaletter.js

Media

bergensbakgrunn.jpg
bergenkommune.png
header_long.png
header_short.png
hordaland.png
lekeikon.png
wc_female.png
wc_male.png

Hvordan ressursene er organisert

CSS

Vi har valg å ha en stor CSS-fil for håndtering av grafiske elementer på nettsiden. Ved å ha en fil er det lettere å holde kontroll på responsiviteten samt at det minsker sjansen for duplisering av kode.

JavaScript

JavaScriptfilene er organisert slik at det er et felles JavaScript-dokument for kode som skal benyttes på mer enn en side. Sidespesfikk kode ligger derimot i egne JavaScript-dokumenter slik at det er lettere å finne frem tilhørende kode. På denne måten kan man også kjøre sidespesifikke funksjonskall ved oppstarten av siden.

Hvor og hvordan de ulike programmeringsoppgavene er løst

Oppgave 1 - Responsiv forside og undersider

Oppgave 1
Filene til denne oppgaven er alle html-dokumentene samt stilark.css. Noe av designløsningene, som menyknappen, har også javascript-funksjon i felles.js.

I oppgave 1 har vi gjort designvalg og tatt hensyn som gjenspeiles i hele oppgaven. For å gjøre nettsiden «responsive», har vi anvendt ulike funksjoner og metoder i CSS. Først og fremst har vi valgt å benytte flex-box for å håndtere hva som skjer med nettsiden dersom arealet endres. Flex-box har gjort det enkelt å kunne styre hvordan elementene i nettsiden oppfører seg ved endring av proporsjonene. I tråd med progressiv forbedring har vi valgt å designe siden «mobile-first» (Gremillion,2018). Dette på bakgrunn av at et økende antall bruker mobilen som sin plattform for tilgang til internett. Allerede i 2012 overtok smart-phone salget pc-salget (Xia, 2017). Vi ønsker derfor et fokus på mobilversjonen av vår nettside, og bygger siden med utgangspunkt i den.

Designet er valgt for at det skal være enkelt og intuitivt å bruke nettsiden for å tilegne seg informasjon. I tråd med dette har vi valgt to forskjellige utforminger av designet, avhengig av om du er på mobil eller desktop.

Unikt for mobilversjonen
Mobilversjonen er primært satt opp slik at elementer tar opp hele bredden av skjermen. Dette løses blant annet med å sette flex: 100% samt width og height: 100%, slik at elementene fyller det området som er tilgjengelig. Dette er gjort primært for å sikre brukervennlighet, da elementer med uendret størrelse relativt til viewporten ville vært svært små og vanskelig å samhandle med. Menyen, som det primære navigasjonsverktøyet, er spesielt tilpasset og lagd for mobilversjonen av siden. Menyen benytter en «onclick»-funksjon slik at den må trykkes på. Dette er valgt over en tradisjonell hover-funksjon, da touch-skjermer ikke benytter en mus som hovrer. Bakgrunnsbildet i headeren er satt til background-repeat: repeat-x;, for å gi mobilversjonen et særpreg. Logoen er laget for oppgaven, da ingen logo vi var fornøyd med fantes på nettet.

Unikt for desktopversjonen

I desktopversjonen av nettsiden har vi fokusert på oversikt og tilgjengelighet. Vi har også ønsket et ryddig utseende, og forsøkt på dette ved å begrense hvor mye som vises på x-aksen til en hver tid. Ved flex-box kunne vi oprettet kolonner som wrappet oppover og utover x-asken når skjermen blir bredere, men i mange tilfeller har vi latt være å gjøre dette da det kan fort «over-populate» nettsiden. I desktopversjonen er headeren satt til no-repeat, og menyknappene er nå lagt utover headeren. Dette er gjort for å gjøre navigasjonen lettere.

Media-query

For de ulike oppløsningene (breddene) i media-queryet har vi valgt å følge bootstraps definisjoner av devices (Mark Otto,2018). Slik at små devices (mindre tablets) begynner på 768px. Medium devices (mindre desktop) begynner på 992px;. Større devices begynner på 1200px og opp. Vi har på sidene med kart og tabell også valgt en enda høyere oppløsning, for bedre plassbenyttelse. Vi har ikke definert noen media-query under 768px, da vi designer sidene mobile first.

Forsiden (index)

Index deler header og meny-mulighetene med de andre sidene. På forsiden har vi valgt et flex-box oppsett som tilpasser seg skjermstørrelsen og vil wrappe elementene utover (fra mobile first). Responsiviteten utgjøres av media-query som vil justere flexen (til flex: 1 auto;) for å sørge for at sidefyllene kommer opp på sidene når skjermbredden er stor nok. Justering av order i media-query sørger for at divene kommer opp slik vi ønsker. Utenom headeren er siden organsiert (fra desktop perspektiv) i en slags grid, med en topp, et senter-element, to side-elementer og en footer. Hele denne gridden er responsiv, og fungerer godt som en template for resten av oppgaven. Dette oppsettet er brukt på index, toalettsøk og favoritt.

Toalettsøk

Toalettsøksiden er organisert på mye av den samme måten som index. Det blir brukt <aside> for å ordne innholdet på linje i desktop. Ulikt er her at den midterste diven inneholder søkefunksjonen i stedet for bilde. Oppsettet gir en intuitiv og ryddig overgang mellom mobil og desktop.

Lekeplasser

Denne siden har et annerledes oppsett for responsive design. Dette på bakgrunn av at vi er kun har to elementer, og tanken var at dette oppsettet er bedre egnet med så få elementer. Her er det benyttet inline-flex med row-wrap for å lage to kolonner med diver. Dette medfører at når man går fra mobil til desktop, vil kartet komme på samme akse som listen.

Favoritter

Denne siden benytter samme template som index og toalettsøk. Her har vi en funksjon på midten som beregner avstand fra din favoritt. Denne layouten er valgt ettersom vi kun hadde ett element på denne siden, og følte det ble litt tomt uten noe annet.

Arbeidsledighet
Organisert på en litt annerledes måte. Her har vi først og fremst tatt hensyn til canvasen som grafen vises på. Normalt ville vi ha tilpasset grafen for mobil-versjonen, men denne blir for liten til å gi mening på et så lite format. Derfor har vi valgt at grafen er scrollbar i mobilversjonen, slik at den er leselig dersom den får mye data. Her har vi også valgt å skru ned skriftstørrelsen ved desktopversjonen, for å gi en ryddigere layout.

Stilarket
Vi har valgt å legge all CSS i samme stilark, da koden kan brukes på mange av sidene samtidig. Dette er spesielt viktig når det gjelder media-query. Stilarket inneholder notasjoner og en inndeling som viser hvilke kode som gjelder for hvilke oppgaver/sider.

Oppgave 2 - Uthenting av JSON-objekt fra ekstern nettside

Oppgave 2 er løst i både i arbeidsledighet.html, favoritt.html,lekeplasser.html og toalettsok.html. Js-filer innvolvert: arbeidsledighet.js, favoritt.js, lekeplasser.js, sok.js, toaletter.js

For å hente ut JSON-objekt fra en URL har vi benyttet en *XMLHttpRequest*. Forespørselen muliggjør tilkobling og uthenting av data fra en ekstern kilde. For å kunne benytte denne funksjonen til ulike URLer og datakilder har vi valgt å benytte en Callback-funksjon. På denne måten kan vi gjenbruke mest mulig kode. Som følge av at funksjonen har en callbackfunksjon kan den benyttes på en rekke ulike scenarier.

Oppgave 3 og 4 - Toaletter i Bergen

Filene til denne oppgaven finner du her:
toaletter.js, toalettsok.html, stilark.css

I denne oppgaven har vi opprettet markører på kartet som viser et infovindu med toalett nummer og navn på plassering som søkefunksjonen returnerer.

Tre tabeller blir opprettet, en for markører, en for info vinduer og en som lagrer informasjon som skal vises i info vinduet.

Kartet blir først sentrert og zoomet inn på nivå 15 for at alle markørene skal få plass på kartet.

En for-løkke itererer gjennom elementene returnert av søkefunksjonen,
toalett nummer og navn på plassering blir satt lik variablene toalettNr og plass som blir hentet fra tabellene idInnhold og plasseringInnhold fra søkefunksjonen.

Videre blir variabelen pos initialisert lik koordinatene longitude og latitude.
Longitude og latitude blir hentet fra de dynamiske tabellene latitudeInnhold
og longtitudeInnhold fra søkefunksjonen.

Man må konvertere fra datatypen String til datatypen Float. Dette må man gjøre fordi elementer hentet fra listen toaletter inneholder String objekter.
Det samme må man gjøre i tilfelle for toalett nummer, hvor man må konvertere streng objekter til datatypen int.

Variabelen marker setter hver markør lik en angitt posisjon på kartet
og markørene blir lagt til i markers tabellen.

Variabelen vinduInnhold holder på informasjon om toalett nummer, navn på plassering og herre/dame ikon.
Videre blir hvert vindu innhold element lagt til i innholdArray med en for-løkke.

Neste for-løkke oppretter infowindow objekter med elementer fra innholdArray tabellen og legger til alle elementene i tabellen infoVinduArray.

Den siste for-løkken sørger for at når man trykker på en markør så blir info vindu med innehold vist over hver markør med informasjon over toalett nummer, navn på plassering og ikon for dame og herre. Dette skjer med funksjonen addListener.

På slutten sjekker If-setningen om nåværende vindu over markøren er åpen,
hvis den er det så blir den lukket med close() funksjonen.

Tabellene infoVinduArray, innholdArray og markers blir nullstilt hver gang man kaller på metoden initmap() for å unngå overlapping fra forrige søk.

Oppgave 5 - Søkefunksjon

Redegjørelse for oppgave 5

Filene brukt til å løse denne oppgaven finner du her:

sok.js, toalettsok.html, stilark.css

Datasettet over toalettene i Bergen tas inn som en variabel i JSON format. I begynnelsen av søkefunksjonen bruker funksjonen hentForm() som "oversetter" inputen fra checkboxene til et format som er mer håndterlig videre i den avanserte søkefunksjonen.
Den avanserte søkefunksjonen bruker 2 lister. En statisk liste som inneholder 8 "true" elementer og en dynamisk liste som for hver "entry" i variabelen toaletter sjekker de 8 vilkårene vi tar inn via søkefelt og checkboxer. Den itererer over alle entriene og sender true eller false til den dynamiske listen avhengig av om kriteriene fra checkboxer og søkefelt stemmer overens med det aktuelle toalettet. Om alle 8 kriterier stemmer overens, dvs de 8 første feltene i begge listene er lik, sender vi entriens data som vi er interessert i videre i diverse lister. Disse brukes til å oppdatere markørene på kartet samt html koder som oppdaterer listen på selve nettsiden over alle toalettene. Alle de ulike kriteriene blir filtrert i hovedsak via if statements som sjekker om de ulike vilkårene er oppfylt.

Funksjonen enkelt søk sammenligner alle bokstavene vi har slått inn i enkelt søk feltet med tilsvarende bokstaver i listen toaletter under "entries", henholdsvis "plassering" og "adresse", dvs vi kan søke på både adresse og toalett navn. Om det vi har slått inn stemmer overens med enten hver bokstav i "plassering" eller "adresse" viser vi toalettet i kartet og oppdaterer listen vi viser på nettsiden. Vi bruker 2 "for løkker" for å få til dette. En for "adresse" og "plassering" og en for hver bokstav vi har slått inn i søkefeltet. Dvs for løkken for hver bokstav er inne den andre "for løkken".

Hurtigsøk funksjonen tar inn alle søkekriteriene våre via ulike regex'er og setter de inn i de ulike søkefeltene og checkboxene til den avanserte søkefunksjonen. Videre kjører vi den avanserte søkefunksjonen ut fra disse kriteriene, etter at søkefunksjonen er kjørt nullstiller vi alle søkefelt til den avanserte søkefunksjonen. Denne løsningen gjør at vi kan gjenbruke kode fremfor å lage duplikater av metoder som er svært like.

funksjonaliteten til hurtigsøk funksjonen er som følger:

slå inn alle ønskede søkekriterier og trykk på knappen, den fungerer dermed helt som den avanserte søkefunksjonen bare at søkekriteriene blir slått inn i et tekstfelt.

for mann, slå inn “mann”
for kvinne slå inn “kvinne”
for rullestoltilgang slå inn “rullestoltilgang”
for stellerom slå inn “stellerom”
for gratis slå inn “gratis”
for åpen nå slå inn “åpen nå”
for maksprис slå inn “pris xx” eks: “pris 12” eller “pris 9”
for åpent klokkeslett slå inn “åpent xx” eks “åpent 12” eller “åpent 23.15”

Oppgave 6 - Kart over lekeplasser

Filene brukt for å løse denne oppgaven er:
lekeplasser.html, lekeplasser.js, stilark.css

I denne oppgaven ble mye av koden fra oppgave 4 gjenbrukt.

Datasettet over lekeplassene blir hentet inn som JSON format.

Først blir kartet sentrert med koordinater som tilsvarer midtpunktet av lekeplassene og zoomet inn på nivå 12 for å få plass til alle markørene på kartet.

Lekeplass nummer blir konvertert fra streng objekter til datatypen int fra tabellen for å angi det i info vinduet.

Koordinat posisjonene latitude og longitude blir konvertert fra streng objekter til datatypen Float for å angi det i kart funksjonen som desimaltall.

Variabelen vinduInnhold holder på informasjon om navn på lekeplass, nummer på lekeplass og lekeplass ikon. Denne variabelen blir lagt til tabellen innholdArray med en for-løkke som setter hvert element i tabellen.

Neste for-løkke legger infoWindow objektene med vindu innhold fra variabelen vinduInnhold til hvert info vindu element i tabellen infoVinduArray.

Til slutt sørger den siste for-løkken for at funksjonen addListener viser informasjon over hver markør med informasjon om lekeplass nummer, lekeplass navn og lekeplass ikon.

If-setningen på slutten tar en sjekk om hvis et info vindu allerede er åpen over en markør
og man trykker på en ny markør på kartet, så blir forrige info vindu lukket.
Dette skjer ved at variabelen currentInfoWindow setter tilstanden lik null hvis et info vindu er åpen.

Oppgave 7 - Distanse mellom to koordinater

Filer brukt til å løse denne oppgaven er:
favoritt.html, favoritt.js

For å løse denne oppgaven benyttet vi oss av informasjon vi fant på nettet for å løse det matematiske. Vi fant ut at Havershine-formelen var effektiv i så måte ("Haversine formula - Rosetta Code", 2018). Denne formelen tar med hensyn til at jorden er kurvet i beregningen. Til tross for at oppgaven forteller at man kan ta utgangspunkt i at jorden er helt flat, valgte vi å ta i bruk nettopp denne formelen for økt realisme.

For å presentere distansen har vi laget en if-statement som gir brukeren relevant måleenhetsstørrelse i forhold til størrelsen på distansen. Korte distanser blir presentert i meter, distanser over 1000 meter blir presentert i kilometer og lengre distanser på mer enn 10 kilometer blir presentert i mil.

Oppgave 8 og 9 - Velg “Min favorittlekeplass” og finn nærmeste toalett

Filene brukt til å løse denne oppgaven er:
favoritt.html, favoritt.js, stilark.css

I denne oppgaven genereres DOM-elementet av typen *select* med tilhørende *option*-elementer i JavaScript. Videre fylles listen med data, og størrelsen settes til 10 elementer hvis ikke datasettet er mindre. *Option*-elementene i listen får en onclick-funksjon som finner nærmeste toalett og tilhørende distanse (benytter funksjon fra Oppgave 7). Til slutt legges *select*-elementet inn i en som allerede ligger på HTML-siden.

Funksjonen som returnerer nærmeste toalett viser plasseringen på nærmeste toalett og distansen fra lekeplassen til toalettet i luftlinje. Vi har valgt å ikke vise noe

ytterligere info om favoritt lekeplassen en navnet som er i listen da det ellers ikke er mye data som er relevant å visualisere på nettsiden.

Oppgave 10 - Valgfritt datasett

Filene til denne oppgaven finner du her:
arbeidsledighet.html, arbeidsledighet.js, stilark.css

I oppgave 10 hentes datasettet inn som en variabel, dvs klippet ut og limt inn som JSON fra http://data.ssb.no/api/v0/dataset/ under “Arbeidsledige, sesongjustert (AKU). Siste 13 mnd”. Datasettet presenteres via et interaktivt interface som består av en måned fra og en måned til dropdown meny. I tillegg til noen checkboxer som lar deg velge mellom prosent og antall. Metodene exlusive1() og exclusive2() sørger her for at man kun kan velge en av knappene “prosent” og “antall” i tillegg til at en av knappene alltid er på.

Det eksisterer helt enkle onclick funksjoner for alle månedene til drop down menyen, de har som funksjon å fylle måneds feltet med den aktuelle måneden samt lukke menyen ved registrert “klikk”.

Generer chart er metoden som brukes til å generere hele grafen som presenteres. Mellom hver gang vi trykker på knappen for å generere søylediagrammene så kalles metoden resetCanvas som fjerner det gamle Canvas elementet og legger til et nytt. Dette for at vi kan oppdatere den nye informasjonen til søylediagrammet uten innblanding fra det gamle søylediagrammet. I praksis bruker den bare metodene fillrect() og filltext() som lager rektangler (søyler) med labels innad i en “for løkke” basert på de relevante dataene fra valueCreator1 og labelCreator1.

For gjennomsnitt og median så tar funksjonen label creator å sender videre de relevante tallene i systemet, som senere brukes for å regne ut median og gjennomsnitt. For medianen bruker vi en egen funksjon for å regne ut dette mens for gjennomsnitt så deles bare totalscoren på antall. Dette gjøres i valueCreator1 metoden.

I praksis er det valueCreator1() og labelCreator1() som tar inn infoen fra aktuelle knapper og felt og lager innhold og lister som genererChart bruker. Ved at alle disse metodene er selvstendige metoder er det svært lett for eksempel å bytte ut metoden for å generere søylediagrammer med en annen metode som for eksempel kan lage pie charts. Dvs vi får en bra “decoupling” med dette oppsettet for endringer og vedlikehold av kode.

Kritikk

Denne oppgaven er løst med utgangspunkt i mye bruk av JavaScript. Brukere som har skrudd av JavaScript i sin nettleser vil derfor. Dette ønsker vi å påpeke at er noe uheldig og kan påvirke brukeropplevelsen

Referanser

Gremillion, B. (2018). A Hands-On Guide to Mobile-First Design. Hentet fra https://www.uxpin.com/studio/blog/a-hands-on-guide-to-mobile-first-design/

Haversine formula - Rosetta Code. (2018). Hentet fra http://rosettacode.org/wiki/Haversine_formula

Mark Otto, a. (2018). CSS · Bootstrap. Hentet fra https://getbootstrap.com/docs/3.3/css/

Xia, V. (2017). What is Mobile First Design? Why It's Important & How To Make It?. Retrieved from https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00