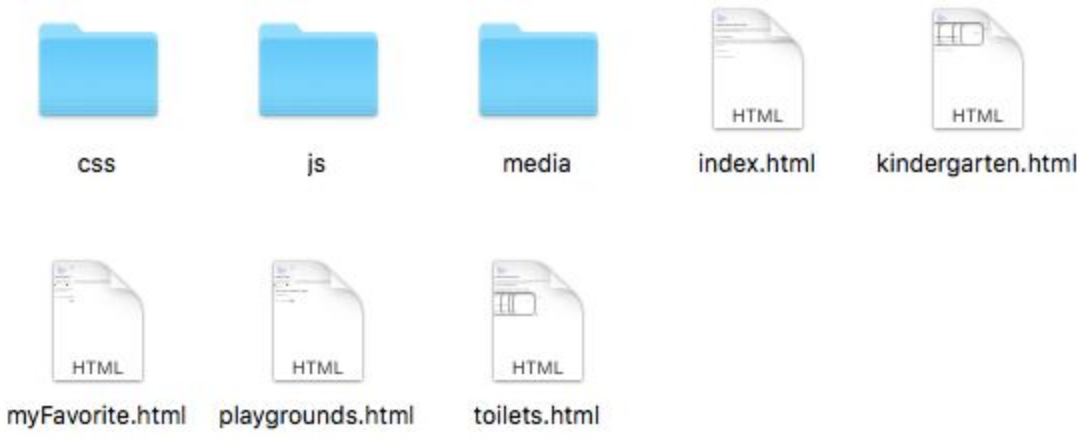
Oppgave 11

Dokumenter vedlagt i oppgaven

Under følger en redegjørelse av alle dokumenter som inngår i prosjektet. HTML dokumenter ligger i ytterste nivå, Javascript i mappen js, CSS i mappen css og mediefiler i mappen media.



Struktur og organisering av kode

Filene er strukturert slik at hvert HTML-dokument har et tilhørende javascript-dokument som inneholder scriptet som benyttes på siden. loadFile.js er plassert i et eget dokument for å redusere duplisering av kode, ettersom alle sidene implenterer dette scriptet. Det er likevel noe duplisering av kode i de ulike javascript-dokumentene. Dette gjelder spesielt funksjoner som oppretter kart og markers, plasserer informasjon fra JSON-dokumentene i lister og funksjoner knyttet til søkefunksjonalitet. Dette er i hovedsak gjort for å ta hensyn til ulik oppbygning av JSON-filer, hvor informasjon om navn, koordinater og liknende ikke alltid er å finne på samme nivå. For toaletter og lekeplasser kunne en dog endret noen av funksjonene til å ta de ulike datasettet som parameter, og benyttet samme funksjon i flere dokumenter. Dette er likevel ikke gjort fordi prosjektet er utviklet over en viss tidsperiode, og vi ønsket å tilrettelegge for ulik struktur i de tilhørende JSON-filene.

Inndelingen av javascript-dokumenter er i stor grad motivert av behovet for å ha god struktur og oversikt når flere personer arbeider med det samme prosjektet. Vi opplevde at det var enklere å kommunisere og samarbeide når det var klart og tydelig definert hvilket script som tilhørte hvilket HTML-dokument. Det er også gjort for å gjøre det enklere for personer som ikke har vært med på utviklingen av siden å navigere i prosjektet. I neste seksjon av besvarelsen vil det redegjøres kort for løsningen av hver enkelt oppgave, og eventuelle videre kommentarer til struktur vil fremgå der.

All CSS ligger i en fil da dette gjør det lettere med generell styling, og mer oversiktlig. Mange aspekter på de forskjellige HTML-dokumentene har samme styling, og det er derfor best praksis å holde alt samlet i denne sammenheng. Man slipper også ekstra "http requests", som gjør at lasting av siden går tregere. I større prosjekter enn dette kan én CSS-fil potensielt bli uoversiktlig, men her fungerer det fint.
Siden er stylet med prinsippet "mobile first", og tilpasset større skjermer ved hjelp av media queries. Mobil-menyen kunne blitt laget med javascript, men med tanke på nettlesere som ikke støtter javascript er det bedre å lage den i CSS, slik at alle har mulighet til å navigere seg rundt på siden.

Prosjektets oppgaver

Oppgave 1

Forsiden gir tilgang på prosjektets andre sider, og er som alle andre sider responsive med bruk av flexbox og media queries.

Løsningnen på denne oppgaven ligger i filene index.html, kindergarten.html, myFavorite.html, playgrounds.html, toilets.html, style3.css.

Oppgave 2

Løsning på oppgave 2 finnes i loadFile.js, med funksjonen “loadFile(url)”. Funksjonen tar et parameter url, og returnerer et løfte som benyttes i samtlige andre javascript-dokumenter. Funksjonen oppretter et XMLHttpRequest-objekt og setter responstype til JSON, slik at funksjonen kun svarer med et objekt hvis URL-ressursen er et JSON-dokument, og med null hvis URL-en viser til noe annet.

Oppgave 3

Funksjonen loadToilets() implementerer loadFile(url) fra oppgave 2. Denne plasserer informasjonen fra JSON-dokumentet i en egendefinert variabel (toiletDataset) som benyttes videre i scriptet. URL til datasettet er gitt som variabel øverst i scriptet. Ettersom videre bruk av variabelen avhenger av om denne er lastet inn eller ikke, er også loadToilets() implementert som et løfte. Å plassere resultatet fra loadFile i en egendefinert variabel er i grunnen ikke nødvendig, ettersom man kunne benyttet “fromResult” som returneres av loadFile(). Det er likevel gjort for å gjøre det lettere å lese og forstå innholdet i koden. Liknende løsninger er implementer på alle prosjektets script-dokumenter.

Oppgave 4

Denne oppgaven er koblet opp mot oppgave 2 og 3. For å legge til et google map, måtte vi først finne plass til den i html filen, legge inn API nøkkelen, og deklarert hvor den skal være og hvor stor den er i CSS filen. For å legge til markers, måtte vi først hente ut datasettet fra loadToilets(), og spesifisere at vi ville ha latitude og longitude fra datasettet, som er koordinatene vi bruker for hvor vi vil ha marksene våres.

Løsningen på denne oppgaven finnes i toilets.html og toilet.js

Oppgave 5

I denne oppgaven skulle vi lage to typer søk: hurtigsøk, og avansert søk. Løsningen ligger i filene toilet.js og toilets.html

Hurtigsøk er fri tekst som du kan skrive inn i den øverste inputen. Avansert søk er checkboxes, og to inputer for tid og pris. Det er mulig å kombinere hurtigsøk med avansert søk.

For å utføre hurtigsøk, er det disse søkekriteriene som er støttet:
Herre, Mann, dame, lady, f.eks Kjønn:Mann, rullestol, Rullestoltilgang, stellerom, skifterom, open, åpen, gratis.

Hvis du skal søke på pris, er det viktig at en først skriver "pris:", før du skriver inn f.eks 12 (pris:12). Grunnen for dette var at jeg måtte skille pris og klokkeslett i regex uttrykkene. Hadde dette skillet ikke eksistert, og du skulle ha søkt på klokkeslett, ville regex uttrykket for pris også slått inn på søket. Da ville søkefunksjonen ha søkt på både pris og klokkeslett og en ville ikke fått returnert det en egentlig søkte på.

Skal du søke på tid er det bare til å skrive inn klokkeslettet f.eks slik: 12.00, eller, 12:00. Det er veldig viktig at ".", eller, ":" skrives inn. Dette er pga. I tids søk vil regex uttrykket til tid bli splittet på disse.

Skal du søke på adresse eller plassering er du nødt til å skrive det fulle navnet. f.eks:
Nonneseter terminal, sør, C. sundtsgt, Bergen kommune, innbyggerservice.
Alle ".", eller, "," må være med. Med andre ord, det må være likt som det står i datasettet. Dette er noe som kan forbedres ved en senere anledning, slik at man bare kan skrive deler av navnet.

For å tilbakestille kartet, må du refreshe siden.

Oppgave 6
Denne oppgaven er løst i dokumentene playgrounds.html og playgrounds.js. Informasjonen fra datasettet presenteres med navnet på lekeplassen i en nummerert liste, og lokasjonen til lekeplassen vist på et google.maps. Nummerering på markersene i google maps samsvarer med

nummerering i listen. Ettersom datasettet kun inneholdt id, navn og koordinater vurderte vi dette som en hensiktsmessig måte å presentere informasjonen på.

Siden implementerer en frisøk-funksjon, hvor man kan søke på hele eller deler av navnet på lekeplassen. Markers og listen med navn oppdaterer seg ved søk.

Oppgave 7

Ettersom denne funksjonen skulle benyttes i "favoritt"-oppgaven, finnes løsningen i filen myFavoritePlayground.js med navn "function findDistance(lat1, long1, lat2, long2)". Funksjonen benytter pytagoras formel for å finne kortest mulig avstand mellom to punkter. Punketne gis som parameter lat1, long1, lat2, long2. Funksjonen tar ikke hensyn til jordens krumming, og benyttes for løsning av oppgave ni i funksjonen "loadFavorite(clickedOption)" i samme dokument.

Oppgave 8

Oppgave åtte er løst i dokumentet myFavoritePlayground.js, i "function placeDropdownElements()". Denne funksjonen tar elementene fra JSON-dokumentet og oppretter et select-element med flere klikkbare option-elementer. Når man trykker på en lekeplass fra listen utløses search-funksjonen som oppdaterer informasjonen på siden.

Oppgaven er løst med select- og option-elementer på bakgrunn av forslag i oppgaveteksten. Vi opplevde likevel en del utfordringer med denne løsningen da den viste seg å ha begrenset funksjonalitet i andre nettlesere enn Mozilla FireFox og på mobile enheter. Ettersom oppgaveteksten spesifiserte at scriptet kun måtte fungere optimalt på nyere FireFox-versjoner, har vi valgt å beholde denne løsningen til tross for nevnte utfordringer.

Oppgave 9

Oppgave ni er løst i filen myFavoritePlayground.js, som implementeres av myFavorite.html. Oppgaven anvender både løsningen fra oppgave 8 og oppgave 7. Når brukeren velger et

favoritt-element vises nærmeste toalett og informasjon om dette. I tillegg vises begge elementer på et google maps. For praktisk implementasjon se kommentarer i koden.

Oppgave 10

Velg et vilkårlig annet åpent datasett fra en norsk institusjon. Implementer en presentasjon av dette datasettet. Presentasjonen må ha minst én ikke-triviell interaksjon. En knapp som viser/skjulerdataene er triviell; en knapp som sorterer en liste er triviell. Mulighet til å sortere en liste på mer enn en måte er ikke-triviell; mulighet til å filtrere en liste basert på brukerens valg er ikke-triviell.

Oppgave 10 er løst kindergarten.js, med tilhørende HTML-fil Kindergarten.html. Datasettet brukt i denne oppgaven er hentet fra http://www.barnehagefakta.no/swagger/ui/index#/. Ettersom datasettet omfattet hele norge, la vi inn en begrensning under *Location -> GET /api/Location/radius/{lat}/{lng}/{radius},* slik at kun barnehager men en viss raduis fra Bergen sentrum ble inkludert. Informasjonen fra datasettet presenteres i en tabell på siden, og lokasjonen til barnehagene vises på et kart.

Barnehagene som vises i tabellen og på kartet kan filtreres etter aldersgruppe, eierform og antall barn. Det er mulig å filtreres på flere egenskaper samtidig, da velger man bare alternativer fra mer enn en dropdown meny