Semesteroppgave
INFO134
Våren 2018

Prosjektet vårt består av disse HTML-sidene:

- ➔ index.html
  - ◆ Dette er hovedsiden vår og den heter Bergen i menyen som blir vist på nettsiden.
  - ◆ Denne siden, og alle andre sider i prosjektet, er responsive og tilpasser seg enheten som leser dokumentet.
  - ◆ Her bruker vi JS-filen felles.js for å få hamburgermenyen til å fungere.
- ➔ toalett.html
  - ◆ På denne siden laster vi opp en liste av toaletter og et kart med markører ved hjelp av JavaScript-filene felles.js og toalett.js.
- ➔ lekeplasser.html
  - ◆ På denne siden laster vi opp en tabell med oversikt over lekeplasser i Bergen ved hjelp av JavaScript-filene felles.js og lekeplasser.js.
  - ◆ Her kan vi også regne ut avstanden mellom to lekeplasser.
- ➔ minFavLekeplass.html
  - ◆ På denne siden laster vi også opp en tabell med oversikt over lekeplasser i Bergen, men her har vi også med en rad med radio-buttons som gjør det mulig å velge en favoritt-lekeplass. Når man velger en lekeplass får man opp informasjon om nærmeste andre lekeplass og nærmeste toalett. For å gjøre dette bruker vi JavaScript-filene felles.js og favoritt.js.
- ➔ norgeRundt.html
  - ◆ På denne siden laster vi opp data om Norge Rundt-episoder som har foregått i Bergen. Et utvalg av disse dataene presenterer vi i en tabell, hvor vi kan filtrere etter hovedrollens kjønn, og søkeord.

JavaScript-filen felles.js blir brukt av flere html-sider. Denne filen inneholder funksjoner som er felles for alle sidene, eller brukes av flere enn en av sidene. Dette har vi gjort for å slippe å skrive samme kode i flere dokumenter. Utenom dette har vi valgt å ha separate JS-filer til de forskjellige html-dokumentene. Dette er fordi det da er lettere å ha oversikt over funksjonaliteten til de forskjellige sidene, og det gjør det lettere å finne feil. JavaScript-filene er skreddersydd til hver enkelt side, slik at de forskjellige knappene har forskjellige event-listeners og funksjoner.

Vi har valgt å bare ha et CSS-dokument. Dette er fordi vi da slipper å skrive mye kode flere ganger. Dette dokumentet har vi skilt i flere deler ved bruk av kommentarer. Øverst har vi en fellesdel hvor funksjonalitet som blir brukt av mer enn en side er. Videre har vi delt dokumentet inn etter hver html-side. Media queryene for forskjellige skjermstørrelser har vi lagt nederst i dokumentet da disse er felles for flere av sidene.

Slik har vi løst de ulike programmeringsoppgavene:

Oppgave 1:

Vi har laget en meny som er synlig i alle html-dokumentene. Denne gir enkel tilgang til alle andre sider i dokumentet. Vi har brukt media queries og flexbox til å gjøre siden responsiv, slik at den tilpasser seg enheten den vises på.

Oppgave 2:

Vi har definert funksjonen getURL i JS-filen felles.js. Denne tar et parameter:url og laster ned ressursen URL-en refererer til. Funksjonen bruker et XMLHttpRequest-objekt for å foreta forespørselen, og returnerer et løfte.

Oppgave 3:

Vi har definert flere JavaScript-funksjoner som benytter funksjonen vi lagde i forrige oppgave. De ligger i toalett.js, lekeplasser.js og favoritt.js. Vi lastet ned og tolket datasettet som beskriver toalettene i Bergen i toaletter.js.

Oppgave 4:

Vi opprettet filen toalett.html. Vi lagde funksjoner i filene toalett.js og felles.js for å generere og legge inn en nummerert liste med navnet på toalettene i datasettet. i toalett.js brukte vi også Google Map-APIet til å vise lokasjonen til toalettene med markører på kartet. Hver markør skal inneholde et tall som korresponderer til tallet i den nummererte listen, slik at disse kan brukes som referanse.

Oppgave 5:

Oversikten over toalettene støtter både hurtigsøk og avansert søk. Vi genererte et søkeobjekt i toalett.js. Når man bruker hurtigsøk eller avansert søk, så oppdateres søkeobjektet, slik at det som er valgt i avansert-søk skjemaet eller eksplisitt skrevet i hurtigsøket blir representert. Etter endt søk bruker vi dette søkeobjektet til å oppdatere listen. Vi bruker listen til å oppdatere kartet, slik at markørene får riktig tall i forhold til endringene i listen som vises. Man kan søke etter kjønn, rullestoltilgang, toaletter som er åpne nå, toaletter som er åpent på et gitt tidspunkt (nærmeste time), stellerom, maks pris, gratis og man kan skrive fritekst som sammenlignes med toalettets navn og adresse.

Ved avansert søk søker man ut i fra et skjema. Ved hurtigsøk skriver man direkte inn i hurtigsøk-boksen hva man søker etter. For eksempel “rullestol max:10”, hvis du vil søke etter toaletter med rullestoltilgang som ikke koster mer enn 10 kr, eller “kaien stellerom kjønn:kvinne” hvis du søker etter toaletter som har navn eller har adresse som inneholder “kaien”, har stellerom og har dametoalett.

Oppgave 6:

Vi opprettet html-dokumentet lekeplasser.html. Vi gjenbrukte funksjonen getURL fra felles.js, og opprettet lekeplasser.js for å laste opp datasettet om lekeplasser i Bergen på nettsiden vår. Vi valgte å vise det som en tabell, slik at mer informasjon blir synlig på en oversiktlig og strukturert måte.

Oppgave 7:

I filen felles.js definerte vi funksjonen sjekkAvstand. Den tar to objekter som parametre. Disse objektene må ha egenskapene latitude og longitude for at funksjonen skal fungere. Denne funksjonen regner ut avstanden mellom objektene.

Oppgave 8:

Vi definerte en funksjonen generateDOM, som legger en tabell inn i en div vi allerede har laget i minFavLekeplass (oppgave 9). Den kjører funskjonen loadFile som ligger i felles.js. Den itererer gjennom datasettet og kjører filen createElement i favoritt.js for hvert element. createElement legger til en rekke, som representerer en lekeplass, i tabellen vi lagde i generateDOM. Hver rekke som blir laget ved hjelp av denne funksjonen har også en radiobutton, som gjør at man kan velge denne lekeplassen som favoritt.

Oppgave 9:

Vi opprettet et HTML-dokument: minFavLekeplass. Her tar vi utgangspunkt i datasettet om lekeplasser. Dokumentet lister opp alle lekeplassene i et tabell-element som vi genererte i oppgave 8. Tabellen har radiobuttons, slik at man kan velge en av de som favoritt. Når brukeren har valgt et favorittelement, vises nærmeste andre lekeplass og nærmeste toalett. Dette gjør vi ved å iterere gjennom datasettene og regne ut avstanden ved bruke funksjonen sjekkAvstand vi lagde i oppgave 7. Vi lagrer elementet og avstanden dersom det er den korteste avstanden til nå, helt til vi har iterert gjennom hele datasettet. Da sitter vi igjen med det elementet som er nærmeste.

Vi synes oppgaveteksten var uklar på hvordan oppgave 8 og 9 skulle løses med tanke på hvilken siden man skulle velge og vise favorittlekeplass på. Vi valgte å vise lekeplass og avstandsmåler på en side (lekeplasser) og vise en liste med lekeplasser og radiobuttons for å velge favoritt på en annen side (favoritt). Vi har valgt å løse oppgaven etter hva vi synes var den mest logiske og oversiktlige måten å vise informasjonen på.

Oppgave 10:

Vi valgte å bruke et datasett om Norge Rundt som man kan finne på denne linken: https://data.norge.no/data/norsk-rikskringkasting-nrk/norge-rundt-statistikkmoro.
Vi opprettet norgeRundt.html og norgerundt.js for å presentere datasettet. Vi satte opp en tabell for å vise dataene. Dette datasettet er veldig stort, 104 turtle filer med 100 elementer i hver fil, så vi valgte å bare laste inn episoder som er fra Bergen, slik at nettsiden ikke skal bruke for lang tid å laste inn. På nettsidene til Data Norge kan man filtrere datasettet. Ved å filtrere på kommune fikk vi en URL som vi kunne bruke til å hente episoder som er fra Bergen. Selv om vi begrenset oss til episoder i Bergen, så er datasettet fremdeles stort, 8 sider. Vi laster derfor bare inn én side av datasettet om gangen (dvs. 100 elementer). Presentasjonen vår gir brukeren mulighet til å bla mellom sidene i datasettet. I praksis vil det si at vi laster inn et

del-datasett hver gang man blar. Slik kan man i ved å bla, se alle de over 700 episodene som er fra Bergen. Vi kan filtrere tabellen vi har satt opp basert på brukerens valg; kjønn på hovedperson, årstall episoden ble sendt på tv og fritekst på tema og antrekk. I datasettet som hentes er det mye mer informasjon om hver episode, men vi har valgt å bare fokusere på og vise tittel, årstall, tema og antrekk da dette var den informasjonen vi synes var mest interessant.Vi har valgt å kunne filtrere på kjønn til hovedskuespiller, men vi synes ikke denne informasjonen var interessant å vise i tabellen. Vi har valgt å også inkludere link til hver episode. Linken er til nrk sine nettsider og åpnes i nytt vindu.

Vi har valgt å bruke to google API fonter og en font (Lato) som vi har lastet ned i en fil. Dette er kode vi ikke har skrevet selv. Dette har vi bare gjort for estetikkens del, og det har ikke noen innvirkning på resten av koden vi har skrevet.

Slik har vi arbeidet sammen:

Under arbeidet med denne semesteroppgaven har vi brukt Teletype i Atom når vi har jobbet sammen på skolen. På den måten kan vi enkelt jobbe sammen i realtime og samarbeide om de tingene vi synes er vanskelig. Ved å laste ned Atom og Teletype på datamaskinen på labben og laste ned prosjektet med GitHub har vi alle kunne jobbet på egen pc mens vi har kunnet vise nettsiden på en stor skjerm slik at ikke en må vise fremgangen og endringen på sin egen pc. Vi har brukt GitHub til versjonskontroll og pushet til git hver gang vi har jobbet på lab eller hver for oss. I noen tilfeller når vi har eksperimentert med nye ting har vi opprettet brancher for å ikke ødelegge for nettsiden og for å lettere kunne gå tilbake til master hvis noe gikk veldig galt.

Alle hører ikke til på samme labgruppe, så vi har jobbet en del hver for oss. Ved å bruke denne kombinasjonen av GitHub og Teletype har vi hatt god kontroll på prosjektets fremgang og dermed fått mange av fordelene GitHub gir oss, samtidig som vi slipper merge-konflikter når vi jobber i samme dokument i realtime. Samarbeidet oss imellom har fungert veldig bra, og vi tror at vi har bidratt omtrent like mye hver. Det er vanskelig å si hvem som har gjort hva i oppgaven, da alle har jobbet litt med alt. Dette har fungert godt for oss, da har alle fått god forståelse og oversikt over koden og hva de forskjellige delene av koden gjør. Vi leverer en samlet oppgave hvor alle har god forståelse og oversikt over oppgaven, hva som er gjort og hvordan oppgavene er løst.