Kanditatnummer:

- 179
- 275
- 314
- 184

Index.html:
Index.html består av de fire div-elementene som vises på skjermen. Det vises kun en div til enhver tid. Index filen består også av header og footer.

myjs.js:
Scritpfilen er til for å hente ut og bearbeide data. Dataen som manipuleres blir satt inn i de ulike divene i index-filen. Scriptfilen består av de tre konstruktørene, sysselsatte, utdanning og befolknings_data og flere hjelpemetoder.

stilark.css:
Stilarket strukturer dataen slik at den vises på en fin måte i nettleser og mobiltelefon. Stilarket setter farger, størrelse på font, posisjon og det er her vi endrer retning på tabellene.

I rapporten skal dere også besvare følgende spørsmål:

1. **Lastes datasettene ned samtidig eller etter hverandre av deres program? Begrunn svaret ditt. Henvis gjerne til koden og forklar når de tre forespørslene blir sendt. (Du trenger ikke å rettferdiggjøre hvorfor deres program laster inn dataene på denne måten.)**

   Oversikt over befolkning lastes ned når nettsiden lastes inn (body onload). Dette gjorde vi slik at datasettet ikke skulle lastes ned hver gang brukeren trykket på oversikt, slik den gjorde når vi lastet det onclick. Utdanning lastes kun ned hvis brukeren skriver inn et kommunenummer i detaljer. Mens sysselsetting lastes ned hvis brukeren skriver kommunenummer i sammenligning eller i detaljer.

2. **Hvordan vet programmet deres når det tredje (siste) datasettet er lastet ned. Begrunn svaret deres. (Henvis gjerne til en variabel, eller et sted i koden der dette er sikkert.)**

```
//load-funksjon som laster inn datasettet. Kjøres når objektet Utdannign opprettes.
Utdanning.prototype.load = function() {
  var ajax = new XMLHttpRequest();

  ajax.open("GET", this.url, false);
  ajax.send();
  if (ajax.readyState == 4) {
    this.obj = JSON.parse(ajax.responseText);
  } else {
    console.log("GREIDE IKKE Å LASTE DATASETT");
  }
};
```

Hvilket datasett som lastes ned sist, er avhengig av hvordan man bruker nettsiden. Hvis man trykker på sammenligning sist, vil sysselsetting lastes inn sist, men om man trykker på detaljer sist, vil utdanning lastes ned til slutt. Eksempel over er fra utdanning, men sysselsetting ser lik ut.

Vi har kanskje gjort det på en litt annerledes måte, fordi vi hadde store problemer med å returnere objekter med asynkron XMLHttpRequest(); Vi måtte derfor sette spørringen til synkron. Dvs. at javascript-tråden ikke kan brukes til noe annet mens objektet lastes ned. Dette er i grunn dårlig praksis, fordi brukeren kan oppleve at siden fryses. Ettersom det går ganske fort å laste inn datasettene, valgte vi å gjøre det på denne måten, fordi det var lettere å jobbe med objektene. Vi trengte derfor ikke en variabel som indikerte når datasettet var lastet inn, men vi sjekket likevel når readyState var lik 4.

3. **På små skjermer skal de historiske dataene presenteres vertikalt. På store skjermer skal de presentereshorisontalt. Forklar hvordan dere har løst dette. (Henvis gjerne til CSS-koden deres.)**

```
@media only screen and (max-width: 1250px)  {

  tr { display: block; float: left; }
  th, td { display: block; }

}
```

Bruker en media Query i css-filen. Når man er på en skjerm som er mindre enn 1250px, vil tabellen bli representert vertikalt istedenfor horisontalt. Det som tidligere var ”To right”, tr, blir nå satt til float left, og blir displayet som block. Disse elementene vil nå ligge under hverandre.

4. **Har alle tre datasett nøyaktig de samme kommunene? Forklar kort hvordan dere fant dette svaret. Dere trenger ikke å legge ved ekstra kode hvis dere har skrevet kode for å svare på dette spørsmålet, men bare forklare fremgangsmåten deres.**

```
function test(){
  var oversikt = new Befolknings_Data(befolkning_url);
  var sysselsatte = new Sysselsatte(sysselsatte_url);
  var utdanning = new Utdanning(utdanning_url);

  var id_befolkng = oversikt.getIDs();
  var id_sysselsatte = sysselsatte.getIDs();
  var id_utdanning = utdanning.getIDs();

  //returns true
  console.log(checkEqual(id_befolkng, id_sysselsatte));

  //returns false
  console.log(checkEqual(id_sysselsatte, id_utdanning));

}

function checkEqual(id1, id2){

  if(id1.length != id2.length){
    return false;
  }

  id1.sort();
  id2.sort();

  for(var i = 0; i < id1.length; i++){
    if(id1[i] != id2[i]){
      return false;
    }
  }
  return true;
}
```

Først oppretter vi de tre datasettene og henter ut tre lister med alle kommunenummer. Vi lager en checkEqual metode som tar inn to kommunenummer-lister. Hvis lengden på de to listene er ulik, returnerer metoden ”false”, med en gang. Deretter sorterer vi begge listene og går gjennom listen og sjekker at alle elementene er de samme. Hvis den går igjennom hele loopen uten og finne to ulike elementer, returneres true.

Først sjekker vi at befolkning er lik sysselsetting, her returneres true, deretter sjekker vi om sysselsetting er lik utdanning, her returneres false. Dermed er befolkning heller ikke lik utdanning. De tre datasettene er altså ikke nøyaktig lik hverandre.