Løsningsforslag Eksamen Høst 2016

I dette løsningsforslaget brukes CSS-dokumentet `brukerinput.css` som du også finner på elevnettstedet under kapittel 10 («Brukerinput med HTML og CSS»). Dokumentet inneholder CSS-kodene som blir gjennomgått i kapitlet.

Nedenfor finner du kommentarer til hvordan vi har valgt å løse de tre oppgavene. Du finner også utfyllende kommentarer i hver av kodefilene.

Oppgave 1

Kommentarer til løsningen:

- På elevnettstedet kan du finne oppskrifter til video- og lydredigerings-programmer.
- Vi brukte først et videoredigeringsprogram (VideoPad) til å sette sammen videoklippene til en 25 sekunder lang film.
- Deretter ble lydklippene satt sammen til et 25 sekunder langt lydklipp i programmet Audacity. Vi brukte de siste 25 sekundene av trekkspillklippet (og tonet inn starten). Måkeskrikene ble lagt på i tillegg.
- Til slutt ble lydklippet lagt til over videoen i videoredigeringsprogrammet.

Filmen skulle også lagres i to ulike formater. Vi er litt usikre på hva som menes med denne delen av oppgaven. Det finnes tre filformater som støttes av `<video>`-elementet i HTML5, og av dem er det formatet MP4 som støttes av flest nettlesere. Det er derfor som oftest riktig å bruke formatet MP4. Det kan også hende at de ønsker filmer med ulik oppløsning, slik at de kan tilpasses mobiltelefoner, TV-er, ulike PC-skjermer osv. Vi har valgt å bare lage en MP4-fil i løsningsforslaget, men oppskriften på elevnettstedet viser hvordan vi kan velge andre filformater og oppløsninger.

Alle filene som ble brukt i prosjektet finnes i løsningsforslaget. I mappen «Eksporterte filer» ligger den eksporterte MP4-filen.

Oppgave 2

Denne oppgaven er en ganske standard «kalkulator-oppgave», der brukeren skal angi litt informasjon. Applikasjonen skal videre undersøke informasjonen fra brukeren og skrive ut en feilmelding eller en bekreftelse.

I løsningen bruker vi egenskapen `className` som lar oss redigere CSS-klassene til et element. Vi kan da legge til ulike bakgrunnsfarger i meldingen til brukeren. Hvis brukeren har angitt feil informasjon, blir bakgrunnsfargen rød, og hvis brukeren har angitt riktig informasjon, blir bakgrunnsfargen grønn.

Oppgaven ber om at det skal vises fram bilder som illustrerer hvor mange middager som er bestilt. I en eksamenssituasjon må eleven bruke en av de vedlagte bildene, men vi har valgt å legge til en mer passende illustrasjon. Framgangsmåten er den samme. Det er også et alternativ å lage to ulike bilder, ett for to middager og ett for tre middager, for så å avgjøre hvilket av de to bildene som vises fram.

For å registrere antall personer, har vi her valgt å bruke et tallfelt. I oppgave 3 bruker vi en nedtrekksmeny i samme situasjon. Vi har gjort det for å vise begge alternativene. Nedtrekksmenyen tvinger brukeren til å velge riktig, men den forutsetter at alle tillatte verdier er tilgjengelige.

Oppgave 3

Rutinen i deloppgave a) er omtrent helt lik den vi laget i oppgave 2. Vi har løst den litt annerledes enn i oppgave 2, med nedtrekksmenyer i stedet for radioknapper og tallfelt, for å vise at slike oppgaver kan løses på ulike måter.

I deloppgave b) dreier utfordringen seg om å slå sammen informasjonen som vi finner i de to tabellene i oppgaveteksten. For å få til det må vi ha en form for kobling mellom de to. Vi har løst det ved å lage to datasett:

- Ett som er identisk med bestillingstabellen i oppgaveteksten, der hver rad i tabellen representeres av ett objekt i en array.
- Ett som ligner på råvaretabellen i oppgaveteksten, men med to nye egenskaper lagt til: `uke`, som angir hvilken uke råvaren skal brukes i, og `middag3`, som er `true` hvis råvaren er med i den tredje middagen, og false hvis råvaren ikke er med i den tredje middagen.

Da har vi alt vi trenger for å koble sammen informasjonen fra de to tabellene. Vi kan da skrive en pseudokode for koden som skal beregne totalbehovet for hvert av fiskeslagene:

```
FOR alle raavarer
  IF raavarer[i].uke === 26
    FOR alle bestillinger
      IF bestillinger[j].uke === 26
        IF raavarer[i].middag3
          IF bestillinger[j].antallMiddager === 3
            // Øker raavarer[i].totalGram med antall bestillinger[j]
            // for henholdsvis barn, ungdom og voksne og ganger hver av
            // antallene med tilsvarende antall gram
        ELSE
          // Øker raavarer[i].totalGram med antall bestillinger[j] for
          // henholdsvis barn, ungdom og voksne og ganger hver av
          // antallene med tilsvarende antall gram
```

Denne pseudokoden kan oppsummeres slik:

- Lag en `for`-løkke som går gjennom samtlige råvarer
- Hvis en råvare er i en middag i uke 26, skal vi fortsette
- Lag en annen `for`-løkke som går gjennom samtlige bestillinger
- Hvis en bestilling er for uke 26, skal vi fortsette
- Hvis gjeldende råvare (`i`) brukes i den tredje middagen (`middag3`), sjekker vi om gjeldende bestilling (`j`) har bestilt tre middager før vi øker totalgram
- Hvis gjeldende råvare (`i`) ikke brukes i den tredje middagen (`else`), øker vi totalgram fordi alle skal ha middag 1 og middag 2.