# Projektopgave del 2

Url: https://github.com/larsk7cdk/vejrportalen
Branch: del 2

Vejrportalen url: http://vejrportalen.k7c.dk/

Til del 2 af projektopgaven har jeg arbejdet med følgende

- HTML tags
- Bootstrap
- Navigering
- Github
- *Webpack*

Følgende vil jeg beskrive hvad jeg har arbejdet med indenfor de enkelte punkter

## HTML Tags

Det overordnede formål med denne uges projektopgave, har været at få en forståelse for opbygningen af struktur og betydningen af HTML5 tags og WAI-ARIA role's.

Generelt er siden blevet opdateret med HTML5 tags og nogle yderligere med en role attribut. HTML5 standarden er kommet med en række nye tags, som semantisk beskriver siden i forhold til f.eks. skærm oplæsere. Det er også muligt at sætte en role attribut på elementet. En role er W3C's guideline for web standarder som beskriver et element. En role kan også bruges til at overstyre et element. For eksempel kan et anchor tags opføre sig som en button ved at sætte en role på

```
<a href="#" role="button" aria-label="Delete item 1">Delete</a>
```

## Ændringer i løsningen

På index.html siden er der lavet følgende

I <head> sektionen er der tilføjet et <meta> tag med description af hvad sitet indeholder, til brug for søgeoptimering.

Under <body> sektionens <header> er der tilføjet en "role=banner". Det er muligt at have flere header tags, men kun et af dem må have en banner role.

Side navigationen er placeret i et <nav> tag med en "role=navigation". Ligesom med header må der godt være flere nav tags, men kun et med navigation role.

Det dynamiske indhold på siden, her varslinger og vejrudsigten, er indeholdt i et <article> tags. En article's indhold skal kunne "tages ud af siden" og stadig give en mening. Hvis en article skal opdeles yderligere, benyttes <section>. En section benyttes til at opdele indhold i grupperinger.

Varslinger er placeret i et <aside> tag med en "role=complementary". Et aside tag relaterer sit indhold til f.eks. en article eller main tag. Derfor er dette valgt, da jeg ser varslingen som en relation til vejrudsigten.

Vejrudsigten og de kommende siders indhold er placeret i et <main> tag, med en ”role=main”. Det er her hovedindholdet på sitet vil være placeret.

Som beskrevet senere er der nu tilføjet et <footer> tag, med en ”role=complementary”. Footeren er ment til at kunne indeholde kontaktoplysninger, firma, copyright information osv.

## Bootstrap

I del 1 af projektopgaven havde jeg taget bootstrap i brug. Her fik jeg lavet selve layoutet af siden. Mit fokus i forbindelse med del 2, har været på at få gjort siden mere responsiv, så den skalerer bedre i forbindelse med mobil visning. Mine rettelser til del 2 omfatter

Siden som viser vejrudsigten for de næste 5 døgn, var i del 1 lavet som en responsiv tabel. I del 2 er dette ændret til at benytte bootstrap d-flex, som er bootstrap’s implementering af CSS Flexbox. Jeg er derfor gået væk fra kolonne visning.

Visningen af hver enkelt døgn er bygget som et bootstrap card. De er placeret i en flexbox række og vil når der ikke er plads nok ombryde og ”hoppe” ned på en ny linie. Dette er opnået ved hjælp af flex-wrap. Ved bootstrap breakpoint lg, sættes egenskaben justify-content-lg-around for at nå opnå en ensartet afstand. Når vi kommer op i denne visning, vil cards’ene ikke ombryde og derfor stå på en række.

Der er oprettet en footer med information om mail, firma og hvornår siden sidst er opdateret. Footeren er fixed til bunden. For at få baggrunden på siden til at gå helt ned til footeren, har jeg valgt at lave højden på containeren der indeholder varslinger og ”sider” til 100vh. Dette giver den ulempe at jeg får en vertical scrollbar. Denne fejl har jeg ikke nået at løse.

Mail’en har jeg lavet som et anchor tag, så når man trykker på linket, åbner den default mail programmet.

## Navigering

Jeg har oprettet 2 nye sider i løsningen, som i første omgang kun indeholder en overskrift og en ledetekst. Til at starte med lavede jeg de 3 sider identiske, forstået på den måde at navigeringen, varslinger CDN’s osv. var indeholdt i alle 3 sider. Dette bryder i mine øjne mod DRY princippet. Jeg valgte derfor at kigge på hvordan dette kunne løses, for at undgå repeterende kode.

Min løsning er nu lavet ved brug af jQuery. jQuery har jeg ikke tidligere haft benyttet, da jeg normalt koder i Angular. På index.html siden er der lavet 2 div tags med hver deres unikke ID. ID’et benyttes til jQuery’s selector, så jeg ved brug af jQuery load funktion, asynkront kan rendere html på index siden.

Dette har flere fordele. Index.html siden loades kun en gang og ved sideskift, har man ikke oplevelsen af en refresh af siden. Når der sker et sideskift, har jeg lavet en animation på opacity. Dette giver en mere glidende overgang, når der navigeres til en ny side.

## Github

Min kode er nu lagt på Github. Jeg har benyttet Github i min dagligdag, men der har været et setup med branches og repositories som kunne tilgås.

Jeg har derfor sat mig dybere ind i muligheden for at lave branches, fetch, merge, stash osv.

Øverst i dokument er et link til koden på Github.

## Webpack

For at kunne understøtte ES2015, har jeg kigget på at få lavet et webpack setup. Webpack er en kode bundler, som b.la. kan transpile ES5 til ES2015 så man får mulighed for at benytte moduler, klasser osv. Dette tog hurtigt noget tid og jeg valgte derfor at lægge det fra mig igen. Dette med begrundelsen om, jeg ikke ved om det giver værdi i forhold til det videre forløb.

Endvidere benyttes der CDN'er i koden som den er lige nu, hvilket også gør at webpack's mulighed for minificering af kode ikke er helt så nødvendig.