Nyhedsapplikation

Eksamensopgave efterår 2022

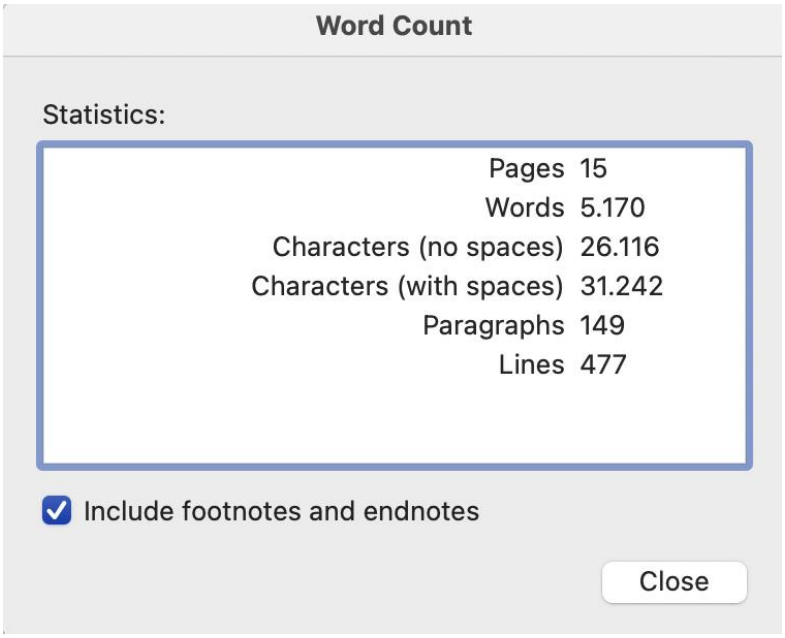
Fag: Programmering og Udvikling af små Systemer samt Databaser, BINTO1064.LA_E22

Studienummer: 160365

Antal anslag: 31.242

Antal normalsider, inkl indholdsfortegnelse: 18

Antal sider i alt: 27



Indholdsfortegnelse

Oversigt over krav

Opfyldt?	Nr.	Krav
Ja	1a	App'en skal tillade brugeren at logge ind.
Ja	1b	App'en skal gøre det muligt for en bruger at logge ud.
Ja	2a	App'en skal tillade en bruger at oprette en profil.
Ja	2b	App'en skal tillade en bruger at slette sin egen profil.
Ja	2c	App'en skal tillade en bruger at opdaterer sin egen profil.
Ja (til dels)	2d	App'en skal tillade at en bruger kan sætte sine yndlingsnyhedskategorier.
Nej	2e	App'en skal tillade at en bruger kan tilføje en eller flere nyhedsartikler til favoritter.
Ja (til dels)	2f	App'en skal tracke hvilke nyhedsartikler en bruger har læst.
Ja	3a	App'en skal vise en liste af de seneste nyheder til brugeren.
Ja	3b	App'en skal kunne vise en enkelt nyhed med links og billede.
Ja (til dels)	3c	App'en skal gøre det muligt for en bruger at søge på mindst to forskellige parametre for nyheder.
Ja	4a	App'en skal vise den nuværende tid og dato
Ja	4b	App'en skal vise vejrudsigten for de næste 7 dage.
Ja	4c	App'en skal vise den nuværende temperatur.
Ja	4d	App'en skal vise solopgang og solnedgangs tidspunkt.
Ja	4e	App'en skal bruge vejrdata for lokationen København
Ja	5	Krav til webapplikationens udseende

Introduktion

Til dette års efterårseksamen på HA(it.) 1. semester i Programmering og Udvikling af små Systemer og Databaser har jeg udviklet en nyheds aggregator hjemmeside. Min nyhedsapplikation giver brugeren de seneste nyheder, vejrudsigten, mulighed for at oprette en profil og meget mere. Applikationen gør brug af bl.a. HTML, CSS, JavasScript, DOM manipulation, API kald og Browser API'er.

Jeg vil i denne rapport gennemgå mine løsninger samt løsningsovervejelser til samtlige krav, som eksaminator har forelagt i eksamensopgavebeskrivelsen. Jeg vil gennemgå disse systematisk, og har undervejs inddraget relevante og specifikke bilag og figurer samt procesevalueringer. For hvert krav vil jeg forklare hvordan og hvilke metoder jeg har brugt for at opfylde kravet. Der er flere krav hvori jeg har brugt samme metode, og jeg har nogle steder valgt kun at forklare koden mere uddybende ét af stederne, således at jeg ikke gentager mig selv unødvendigt. Jeg har dog gjort det tydeligt, hvilken metode eller teknik som er blevet brugt de specifikke steder.

Arkitektur og design

Tilgå nyhedsapplikationen

Når min applikation skal tilgås, gøres dette ved at åbne filen index.html. Denne åbnes med Live Server, således at nyheds API'et bliver kaldt. Man vil ikke have adgang til nogle af funktionaliteterne ud over nyheder og vejrudsigten før man har oprettet en profil eller efterfølgende logget ind. Når man har oprettet en profil eller er logget ind, vil man have kunne benytte samtlige funktionaliteter.

Opbygning og filer

For at programmere min applikation har jeg brugt kildekode-editoren Visual Studio Code. Her har jeg tre typer filer, 1) HTML; 2) JavaScript og 3) CSS. Jeg har valgt at opbygge min applikation således, at man har adgang til vejrudsigten samt de seneste nyheder uden at have en bruger og være logget ind. Der kommer dog flere funktionaliteter og sider hvis man opretter en profil og er logget ind. Jeg kunne have valgt at give adgang uden at have en profil og være logget ind, men jeg vurderede at dette vil være mere omfattende end min nuværende løsning,

og derfor kræve længere tid. Jeg har desuden brugt Visual Studio Code udvidelsen 'Live Server', som opdaterer siden live, mens jeg redigerer i den. Live Serveren, er desuden dét der gør det muligt at hente mit nyheds API, da der ellers ikke ville være en server til at fetche denne.

Flere af de forelagte krav kræver at jeg gemmer data. Dette har jeg, på baggrund af oplysningerne i opgavebeskrivelsen, valgt at gøre ved at gemme dataen som flade filer på klientens maskine. Her gør jeg brug af JSON-filer i localStorage, illustreret i figur 1. Her er jeg dog opmærksom på, at der ved brug af localStorage til at gemme brugeres oplysninger, bør tages nogle sikkerhedsmæssige tiltag. Data gemt i localStorage kan nemt blive tilgået af andre med JavaScript, som vil få adgang til eventuelt følsomme data. For at gøre dette mere sikkert, kan en løsning være at kryptere følsomme data, såsom adgangskode og brugernavn. Men til denne opgavetype og dens formål, har jeg ikke implementeret dette.

Figur 1: Profil gemt som JSON i localStorage.

For at løse flere af kravene, har jeg brugt forskellige API'er, herunder bl.a. et nyheds API og vejr API. For at have kunne hente disse API'er har jeg en API-key, som er min personlige kode til at verificerer min adgang til API'erne. Denne kode ville kunne blive misbrugt, og da man eksempelvis ofte betaler for antallet af gange man har kaldt på API'en, ville andre kunne udnytte min API-key, og jeg vil sidde tilbage med en stor opkrævning. Der er flere måder hvorpå man kan gemme sin API-key, eksempelvis ved at gemme API-key'en i andre filer[1]. Ligesom ved brug af localStorage, vil jeg med denne opgavetype og dens formål, ikke implementere dette.

Krav om forsidens udseende

Min forside, homePage.html, lever op til kravet for forsidens udseende. Generelt går det overordnede udseende igen under alle mine sider. Dette har jeg valgt at gøre, således at der er

[1] Gagne, 2018

større sammenhæng og flow mellem de forskellige sider. Jeg har desuden designet de forskellige elementer på mine sider således at deres længder følger med vinduets størrelse. Til dette har jeg sat diverse elementers 'width' i procenter i min CSS. Dette fungerer i høj grad, kun med nogle få undtagelser. Jeg har desuden brugt flex-wrap således at mine artikler flytter sig i takt med, at vinduet bliver større eller mindre.

Jeg har brugt flere ikoner forskellige steder på min applikation. Alle er fundet hos Font Awsome[2]. Da jeg ønsker at opfylde design kravet, har jeg udvalgt de ikoner som jeg mener passer bedst. Nogle af disse ikoner er dog 'Pro', og kræver brugerbetaling hos Font Awsome, og jeg har i stedet for valgt, at tage screenshots af disse. Den bedste løsning ville dog være, at betale for medlemskab, således at jeg ville have fri adgang til alle ikoner. Til denne opgaves type og formål, har jeg dog vurderet, at denne løsning kan gå an. Var applikationens formål anderledes, ville jeg dog mene, at et betalt medlemskab, ville være den korrekte og bedste løsning.

Brugerens navn bliver desuden vist på mine forskellige sider. For at gøre dette har jeg oprettet flere funktioner, som parser den string med profil information som ligger i localStorage (bliver uddybet senere) og tager den specifikke data. Der bliver vha. JavaScript oprettet p-elementer, hvor property'et .innerHTML ændrer teksten i elementet til dataen fra localStorage. For at indsætte det oprettede p-element det rigtige sted, har jeg brugt metoden .appendChild(), (se bilag 1). Hvis jeg havde alle funktioner til hver side i samme JavaScript, ville de overskrive hinanden, og kun en af dem ville virke. Derfor har jeg valgt at oprette separate JavaScript filer til hver side, for at være sikker på at funktionerne ikke overskriver noget.

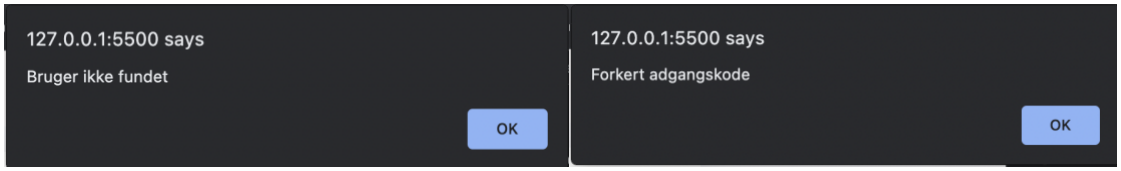
Kravspecifikationer

Jeg har opfyldt 16 ud af 17 krav, og for at vise det endelige resultat på klientsiden, har jeg i bilag 2 linket til en video hvor jeg gennemgår disse. I denne video bliver localStorage og diverse pop-up beskeder ikke vist, og jeg har derfor valgt at vedlægge disse i bilag 3. I det kommende afsnit vil jeg uddybe mine løsninger, præsentere mine løsningsovervejelser, procesevalueringer samt refleksioner.

[2] Font Awsome

Krav 1a og 1b – Logge ind og ud

Det første brugeren møder når de tilgår min applikation, er min forside. Her kan de se nyheder og vejrudsigten, men hvis de vil benytte funktionaliteterne på min side, må de logge ind eller oprette en profil. For at logge ind skal man have oprettet en profil, således at brugerens information, herunder brugernavn og adgangskode, ligger i localStorage. Dette vil blive uddybet i afsnittet om oprettelse. Når man trykker 'Log ind' kalder dette på en funktion, med et else-if statement (se bilag 4). Her tjekker jeg om det indtastede brugernavn og adgangskode findes i localStorage. For at gøre dette tager jeg først fat i min string med profilinfo, som jeg har oprettet under oprettelse af brugeren. For at jeg kan arbejde videre med denne, bruger jeg metoden JSON.parse(). I min else-if statement, tjekker jeg vha. '==='-operatøren først om det indtastede brugernavn og adgangskode begge ligger i localStorage. Er det indtastede brugernavn lig med brugernavnet i profilinfoen, og er kodeordet lige med kodeordet i profilinfoen, vil brugeren blive logget ind og omdirigeret til forsiden, homePage.html, hvor de vil have adgang til funktionaliteterne. Hvis det indtastede brugernavn ikke findes i localStorage, vil brugeren få en alert, som informerer dem om, at brugeren ikke er fundet. Hvis brugernavnet derimod er korrekt, men koden er forkert, vil brugeren få en alert, som informerer om, at adgangskoden er forkert (se figur 2).



Figur 2: Alert ved forkert brugernavn eller kode.

Jeg oplevede flere problemer med mit log in. I første omgang oprettede jeg inputfelterne, samt 'log ind' knappen i et form-element, ligesom ved oprettelse af bruger. Man kunne dog få adgang til forsiden (homePage.html) selvom brugernavn eller adgangskode ikke var korrekt. Jeg måtte derfor undlade form-elementet, hvorfor en bruger kan forsøge at logge ind uden at udfylde brugernavn eller adgangskode. I disse tilfælde vil brugeren dog få besked om, at brugeren ikke er fundet eller at adgangskoden er forkert (se figur 2), hvorfor de alligevel ikke vil kunne få adgang.

Når brugeren er logget ind, har de mulighed for at logge ud. Når de trykker på log ud-ikonet i øverste højre hjørne, kalder denne på en confirm, som viser en dialog box. I denne dialog box skal brugeren tage stilling til, om de er sikre på at de vil logge ud. Hvis de trykker 'ja', vil de blive omdirigeret til forsiden uden funktionaliteter, index.html (se figur 3). Nu vil de ikke have adgang før de logger ind igen.

```
// Her laver jeg mit log Ud
function logOut () {
    if(confirm("Er du sikker på at du vil logge ud?")) {
        window.location.replace('index.html'); // omdirgerer til index.html
    } else {}
}
```

Figur 3: Funktion til log ud.

Ved denne løsningsmetode har jeg taget udgangspunkt i, at der kun er en bruger pr. enhed. Forsøger man på nuværende tidspunkt at oprette flere brugere, vil de overskrive hinanden, og kun den nyeste bruger vil være gældende. Ved at videreudvikle, ville jeg for hver bruger oprette et nyt objekt i arrayet i key'en 'profileInfo'. Ved min logIn funktion, ville jeg derfor skulle lave et for-loop som løber igennem arrayet, og tjekker om det indtastede brugernavn og adgangskode skulle befinde sig i et af profilerne.

Krav 2a og 2b – oprette og slette profil

Når man åbner index.html, kommer man ind på en side, hvor man får to muligheder. Du kan oprette dig, og du kan logge ind, men for at logge ind, skal man have en profil. Til oprettelsen har jeg brugt et form-element, således at når man har udfyldt fornavn, efternavn, brugernavn og adgangskode, og trykker opret profil, vil de indtastede data blive gemt i locaStorage (se figur 1). Når man opretter sig, skal man udfylde alle felterne. Hvis man undlader et felt og trykker 'Opret profil', vil man få en besked om, at man skal udfylde de manglende felter (se figur 4). På denne måde sikrer jeg at al data som jeg senere skal benytte, er indtastet. Når oprettelsen har været succesfuld, og oplysningerne er gemt i lcoaStorage, vil man blive omdirigeret til sin profil. Herfra vil man kunne tilgå nyhederne, favoritter og søgning. Man kan altså ikke tilgå førnævnte uden at have en profil, og være logget ind. For at gemme oplysningerne i localStorage, har jeg brugt interfacet localStorage og metoden .setItem() for at

indsætte værdierne i inputfelterne i localStorage. Jeg har desuden brugt metoden JSON.stringify() for at omforme den indtastede data til en JSON string.

Figur 4: Besked ved manglende felt

Under brugerens profil har man mulighed for at slette sin profil. Når man trykker på denne knap, vil der komme en pop-up besked, hvor man skal tage stilling til hvorvidt man er sikker på at man vil slette sin profil. Dette har jeg valgt, så man ikke ved en fejl kommer til at slette sin profil. Hvis man trykker 'cancel' kommer man blot tilbage til profilsiden, men hvis man trykker 'ok', kalder dette på en funktion, som sletter localStorage og omdirigerer til index.html. På denne måde er profiloplysningerne fjernet fra localStorage, og man kan ikke logge ind, da oplysningerne ikke længere er tilgængelig (jf. log ind).

For at have ekstra sikkerhed, kunne jeg tilføje at passwordet minimum skal have et tal, have en minimumlængde m.v. Jeg ville også kunne kryptere den personlige info gemt i localStorage, da localStorage generelt ikke er særlig sikkert med hensyn til sådanne oplysninger. Desuden vil en oprettelse i localStorage kun gælde på den enhed hvorpå brugeren har oprettet sig. Skulle jeg videreudvikle applikationen, ville jeg kunne gemme profilinformationerne på serveren, således at de kunne tilgås fra flere enheder mv., samt implementere det ekstra lag af sikkerhed.

Krav 2c – opdater profil

Min applikation gør det muligt for en bruger at ændre sine oplysninger. For at dette kan lade sig gøre, har jeg lavet fire form-elementer med er input-felt og en 'Opdater' knap til hhv. fornavn, efternavn, brugernavn og adgangskode. Når man trykker 'opdater', vil dette kalde på en funktion, som parser stringen med personlig info, således at jeg kan gå ind og ændre i denne.

Jeg sætter herefter det indtastede ind, og bruger igen metoden .stringify() til at indsætte den i localStorage. Siden vil afslutningsvis reloade, således at eksempelvis navnet bliver opdateret på hele siden (se bilag 5). Jeg har desuden tilføjet en alert, som informerer brugeren om, at eksempelvis brugernavnet er succesfuldt ændret.

Krav 2d – Yndlingsnyhedskategorier

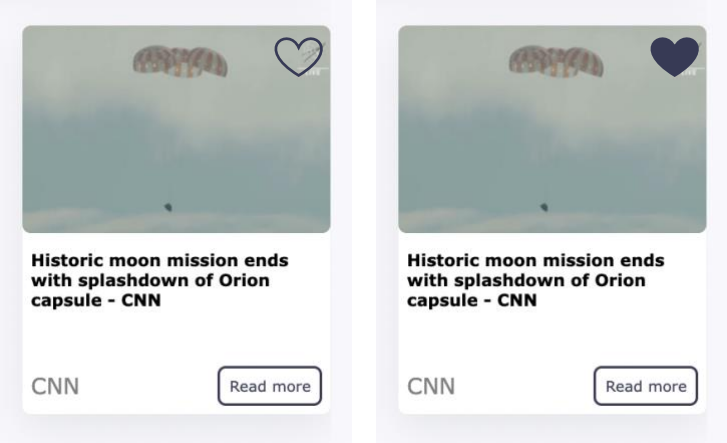
For at opfylde kravet om, at brugeren skal kunne sætte sine nyhedskategorier, har jeg brugt localStorage. Her har brugeren via profilsiden, mulighed for at tilføje sine yndlingskategorier, dog i skrivende stund, blot én. Jeg har her lavet et input felt, hvor brugeren kan indtaste sin yndlingsnyhedskategori, og tilføje denne til sin side ved at trykke 'tilføj'. Ligeledes kan brugeren fjerne kategorien ved at trykke på 'fjern' (se bilag 6). Dette har jeg gjort ved at tage værdien af inputfeltet og metoden .setItem() fra localStorage interfacet. For at få kategorien vist på profilsiden, har jeg først brugt metoden JSON.parse() til at parse den string hvori kategorien er. Jeg bruger herefter metoden .getItem() for at hente det specifikke objekt i localStorage, .innerHTML for at ændre værdien i det givne li-element og .appendChild() for at indsætte det.

I en videreudvikling ville jeg gøre det muligt for brugeren at oprette flere yndlingskategorier. Dette ville jeg gøre ved at lave en funktion, der bliver kaldt når brugeren trykker 'tilføj', som opretter flere key's i local storage. 'Fjern' knappen vil kunne fjerne hver enkelte kategori, ved at slette en given key. På denne måde ville brugeren kunne oprette uendeligt mange kategorier samt slette specifikke. Lige nu er der desuden heller ingen funktionalitet i yndlingskategorierne, men jeg kunne forestille mig, at jeg på sigt ville kunne sorterer i artiklerne på forsiden, således at det er de mest relevante artikler det bliver vist. Her ville jeg evt. kunne bruge samme metode som jeg har brugt ved krav 3c, hvor brugeren kan søge efter artikler.

Krav 2e – Tilføje en eller flere nyhedsartikel til favoritter

Kravet om at en bruger skal kunne tilføje en eller flere nyhedsartikler til favoritter har jeg ikke opfyldt. Jeg har en side til favoritter, som lige nu viser tilfældige artikler. På trods af at jeg ikke har opfyldt kravet, vil jeg dog alligevel komme med to mulige løsningsforslag.

Først og fremmest skal brugeren have mulighed for at markere en artikel som favorit. Dette kunne eksempelvis gøres, ved at klikke på et ikke udfyldt hjerte på den pågældende artikel. Jeg ville på dette ikon have en 'onclick' funktion, som ændrer ikonet til eksempelvis et udfyldt hjerte (se figur 5). Jeg ville nu kunne lave en funktion, som først og fremmest fetcher nyheds API'en på samme måde som ved nyhederne på forsiden og ved søgning (vil blive uddybet nærmere senere). I samme funktion ville jeg dernæst have et if-statement, som tjekker ikonet. Hvis ikonet er det udfyldte hjerte, vil funktionen tage det pågældende artikels index og tilføje denne til et nyt array. For at vise artiklen på favoritsiden, vil jeg loope igennem det nye array i et for-loop, og indsætte hhv. billede, titel, kilde og url til siden. Metoden til at få vist artiklerne vil blive uddybet i afsnittet om krav 3a og 3b.



Figur 5: Mock-up af favoritknap på artikel.

Ovenstående metode tager dog ikke højde for, om en bruger er logget ind eller ud, og jeg vil derfor skulle bruge localStorage. Favoritartiklerne bør tilføjes til en key i localStorage, og for-loopet bør gå igennem denne key. Igen løber jeg ind i problemer, da artiklerne vil være i localStorage på trods af om en bruger er logget ind eller ud. I den nuværende applikation har jeg, som tidligere nævnt, taget udgangspunkt i, at der kun er en bruger pr. enhed, og da man ikke har adgang til nogle funktionaliteter uden at være logget ind, vil man derfor heller ikke have adgang til favoritter. Denne metode vil derfor løse problemet til en vis grad.

En anden løsning ville være at have en variabel, loggedIn, som enten er true eller false. I funktionen som tjekker favoritikonet, vil der være et if-else statement. Hvis loggedIn er true vil for-loopet køre, hvis loggedIn er false vil der ikke blive vist nogle artikler, selvom de ligger i localStorage. Da loogedIn er en boolean, vil jeg kunne bruge 'logical not'operatoren ! til at

ændre loggedIn værdien. Dette ville desuden nemt kunne implementeres i funktionen logIn(), som bliver kaldt når en bruger logger ind, samt i funktionen logOut(), som bliver kaldt når en bruger logger ud. Altså vil loggedIn være true når brugeren er logget ind, men vil blive ændret til false, når brugeren logger ud, og omvendt. På trods af at jeg ikke har opfyldt kravet om at en bruger kan tilføje en eller flere artikler til favoritter, vil jeg mene at jeg er godt på vej, taget mine overvejelser til mulige løsninger i betragtning.

Krav 2f – Tracke læste nyhedsartikler

Min app kan tracke hvilke nyheder brugeren har læst. Dette har jeg gjort ved hjælp af HTML a-elementets href attribut og CSS regler. Jeg har linket til de enkelte artiklers ophav i hyperlinket i 'Read more' a-elementet, og stylet disse. For at markere at artiklen er læst, har jeg brugt selektoren :visited, for at ændre 'Read more' elementets udseende i CSS (se bilag 7). En anden løsning ville være at ændre 'Read more' knappens .innerHTML til 'Read again', således at det er mere tydeligt at artiklen er blevet læst. Ved at videreudvikle, ville jeg kunne bruge localStorage og i højere grad opfylde kravet. Her ville jeg kunne oprette en key i localStorage, med den unikke url til artiklen som brugeren har læst, som value. Hver gang en bruger trykker på et link, vil der således oprettes en ny key, hvor værdien er url'en til den specifikke artikel.

Der er flere begrænsninger ved min nuværende løsning. Da selektoren :visited bruger data fra computerens historik, vil den ikke være begrænset af hvorvidt en bruger er logget ind eller ud. Da man har adgang til artikler uden at være logget ind, vil man stadig kunne se hvilke artikler brugeren har tilgået, uden at være logget ind. Her kunne jeg have ændret i mine filers opbygning, således at logIn.html ville være index.html, og at man derfor ikke ville have adgang til min applikation, uden at være logget ind. Et andet problem er dog også, at hvis
en anden bruger logger ind på samme computer, vil linket som den første bruger har tilgået, fortsat være markeret som læst hos den anden bruger, samt vil url'en fortsat være i localStorage. Da denne applikation er lavet med udgangspunkt i at der kun er én bruger pr. enhed, er dette dog ikke ligeså relevant for nuværende.

Krav 3a og 3b – En liste med seneste nyheder og enkelt nyhed med links og billede

På forsiden af min webapplikation ses syv af de seneste nyheder, hvoraf den seneste af disse er fremhævet. Disse nyheder er hentet fra nyheds API'et som blev givet i opgaveformuleringen. For at hente API'et har jeg oprettet en funktion til at fetche dataen og gemme denne som en JSON fil. I samme funktion har jeg to for-loops der går gennem det array, som jeg får af API'et. Jeg har begrænset antallet af nyheder til syv, således at hjemmesiden ikke fortsætter ud ad frame. Mit for-loop til de små nyhedsartikler starter derfor ved index 1 og slutter ved index 6 (i < 7), og mit for-loop til den store nyhed starter ved index 0 og slutter ved index 0 (i < 1). Jeg kunne have valgt at indsætte alle nyhederne fra nyheds-API'et ved at jeg lade mit for-loop loope igennem hele arrayets længde.

I mine to for-loops har jeg samme opbygning, som fremgår af bilag 8. Jeg starter med at definere at der for hver artikel skal laves en div, samt at denne skal have hhv. class 'smallArticles' og 'mainArticle'. Ved hjælp af JavaScript opretter jeg img-, p-, div- og a-elementer hvorefter jeg bruger .src, .textContent og .href, til at indsætte hhv. urlToImg, title, source og url i de oprettede HTML-elementer. Efterfølgende benytter jeg Node metoden .appendChild() til at rykke de oprettede HTML-elementer ind i de oprettede div'er 'smallArticles' og 'mainArticle'. Afslutningsvis rykker jeg vha. .appendChild() ydereligere 'smallArticles' ind i min div med 'otherArticles', som er oprettet inde i min HTML. Jeg oplever dog at nogle billeder fra API'en ikke virker, da der ikke er noget reference link i 'urlToImg'. For at komme udenom dette billede, ville jeg kunne lave en funktion, der indsætter et vilkårligt billede, i tilfælde hvor 'urlToImg' = null.

Med den brugerplan som jeg er oprettet med, har jeg et loft for, hvor mange gange jeg kan kalde på API'en inden for en bestemt tidsperiode. Det hænder derfor at nyhederne ikke bliver vist. For at undgå dette problem, ville jeg kunne indgå i en brugerbetaling, hvor jeg vil betale for antallet af gange, som jeg kalder på API'en. Til denne opgaves opgavetype og formål, er dette dog ikke nødvendigt.

Kravet om at app'en skal kunne vise én enkelt nyhed med billede og links har jeg fortolket som, at hver enkelte nyhed skal separeres, således at det er tydeligt hvilket billede, titel og kilde som hører sammen. Jeg har desuden valgt at indsætte URL'en til hver enkelt artikel i de tilhørende 'Read more' knapper. Altså vil man blive ført til den oprindelige artikel når man

trykker på denne knap, og man vil på denne måde kunne læse mere om den enkelte nyhed. Ved at videreudvikle, ville jeg kunne lave nye sider til hver enkelte nyhed, således at man ikke blev omdirigeret til en anden webside, men forblev på min app, når man trykker 'Read more'.

Krav 3c – søge efter nyheder

Min applikation gør det muligt for en bruger at søge efter artikler med specifikke emner eller kilder. Dette har jeg løst ved at bruge samme struktur som ved mit nyheds API kald. Her har jeg dog ændret url'en, således at værdien af input-elementet i søgebaren, bliver indsat i url'en (se bilag 9)[3]. En bruger har hermed mulighed for at søger efter kilder, såsom 'BBC' eller 'Fox', samt emneord, såsom 'bitcoin' og 'soccer'.

På samme måde som ved artiklerne på forsiden, bruger jeg et for loop for at indsætte dem på siden. Jeg har her valgt at begrænse antallet til 6, så min side ikke drukner i artikler. Her kunne jeg vælge at lade mit for loop køre til i < data.length, for at få alle søgeresultaterne. Hvis man søger på et nyt emneord lige efter man har søgt, vil de første resultater fortsat være vist. For at undgå dette, bør siden reloades vha. window.location.reload(), før de nye resultater vises.

Med min løsning er man dog begrænset, da nyheds API'et ikke har mange danske nyheder, og man vil derfor ikke få mange resultater ved at søge på danske ord. Ligeledes kan man ikke søge på andre søgekriterier, såsom dato. Jeg ved dog at der er flere måder hvorpå man kan gøre dette, hvor jeg kan bruge samme metode som den nuværende, men i stedet fetche andre url'er, hvor inputtet er et tal, ligesom ved en dato[4].

Krav 4a – Tid og dato

Kravet om at app'en skal vise nuværende tid og sted har jeg opfyldt ved brug af objektet Date. Når jeg burger dette objekt som en constructor, returnerer denne en string med nuværende tid og dato, baseret på klientens enhed. For at opdele tid og dato i henhold til kravet om applikationens udseende, har jeg lavet én div til tid og én div til dato, samt brugt følgende indbyggede JavaScript metoder, Date.getHours(), Date.getMinutes(), Date.getSeconds(), Date.getDate(), Date.getDay(), Date.getMonth() og Date.getFullYear()[5].

[3] "Get started", n.d.
[4] "Everything", n.d.
[5] "Date", 5/12-22

Da .getDay() og .getMonth() returnerer nummeret på dagen i ugen og nummeret på måneden i året, har jeg lavet else if statements, som kan ses i bilag 10. I dette statement siger jeg, at hvis todaysWeekday === 1, så skal innerHTML i det givne h-element ændres til 'Mandag', osv. Ligeledes har jeg gjort dette med måned. Jeg er dog opmærksom på, at .getDay() returnerer et tal mellem 0 og 6, samt at .getMonth() returnerer et tal mellem 0 og 11. Her er søndag og januar begge 0. Dette har jeg løst på to måder. Som det ses i bilag 10, har jeg blot sagt, at hvis todaysWeekday === 0, skulle innerHTML i det givne h-element ændres til 'Søndag'. Da jeg skulle løse problemet i måned, mente jeg, at det var nemmest at adderer 1 på det returnerede nummer på måned, således at januar blev måned 1 og december måned 12, hvilket kan ses i figur 6.

```
const todaysMonth = todaysDate.getMonth()+1;
```

Figur 6: Adderer 1 til måned, så måned 0 bliver januar osv.

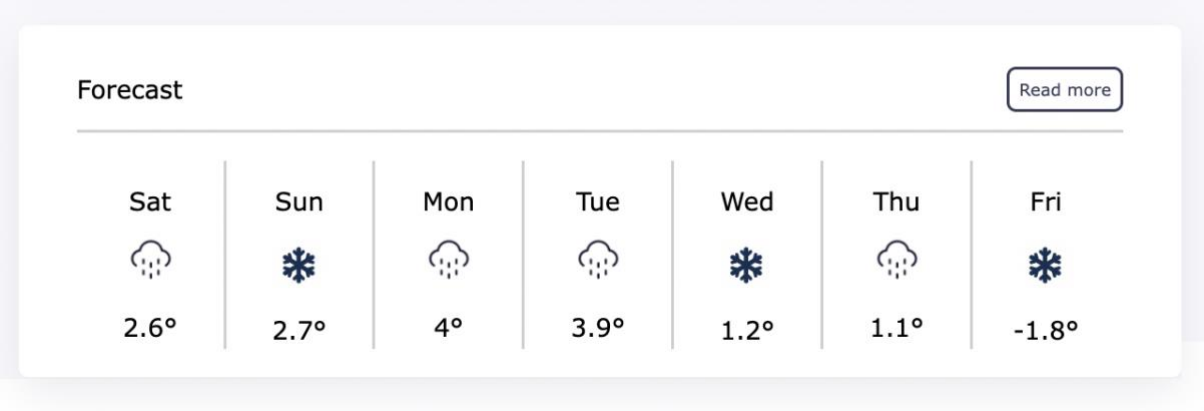
Der er to måder at opfylde kravet om at vise den nuværende tid. Date() constructor'en har også metoder for timer, minutter og sekunder. Her kunne jeg bruge samme metode som ved dato og årstal. Dog viste uret med denne metode kun tiden hvor klienten tilgik hjemmesiden. Da jeg gerne ville have uret til at opdatere hvert sekund, har jeg i stedet lavet en funktion, der ved hjælp af metoden setInterval() bliver kaldt hvert millisekund. Den er derfor så præcis som mulig, da den ved 1000 millisekunder, kunne risikere at være 999 millisekunder bagud ift. computerens tid[6].

Krav 4b, 4c og 4e - Vejrudsigt for de næste 7 dage og nuværende temperatur, solopgang og solnedgang for København

For at opfylde kravet om, at applikationen skal kunne vise vejrudsigten for de næste 7 dage samt nuværende temperatur og solopgang og solnedgang, har jeg brugt en API fra Visual Crossing[7]. Jeg har her oprettet mig og hentet min API key, og ved brug af deres query builder[8] har jeg valgt den præcise data som jeg skal bruge (se bilag 11). For at kunne hente API'en, har jeg oprettet en funktion, hvor jeg benytter metoden fetch(). Overordnet har jeg brugt samme metode som ved artiklerne (se bilag 8). Jeg har brugt JavaScript til at oprette elementer, brugt diverse Node Js metoder og .appendChild() til at flytte de oprettede elementer til de rigtige

[6] WilliamG, 2016
[7] Visual crossing
[8] "Query builder", n.d.

steder, således at min applikation lever op til kravet for udseende. Skulle man have lyst til at finde endnu flere detaljer om vejret, har jeg valgt at linke til DMI's vejrudsigt for København i 'Read more' knappen, illustreret i figur 7.



Figur 7: Vejrudsigten for de næste syv dage samt 'read more' knap der fører til DMI's verjudsigt.

For at indsætte nuværende vejrdata, har jeg hentet data fra API arrayets objekt 'currentConditions', og hentet nuværende temperatur og lokation, og dernæst brugt .appendChild() og indsat dem i de givne div'er. Ligeledes har jeg gjort for at få tidspunkterne for solopgang og solnedgang.

For at lave vejrudsigten, har jeg først og fremmest hentet max temperatur for de givne dage. I arrayet vil days[0] være nuværende dag, så den første dag i vejrudsigten er derfor days[1] osv. Her har jeg ligesom før brugt JavaScript til at oprette div'er, og .textContent til at ændre div'ens indhold til forecast.days[i].tempmax, og appended dem ind i de givne div'er. Således bliver de forskellige dages max temperatur indsat de korrekte steder.

For at indsætte vejrikonerne for både nuværende vejr, vejrudsigten samt solopgang og solnedgang, har jeg taget screenshots af de ikoner som er brugt i mockup'et i opgavebeskrivelsen. Mine andre ikoner har jeg fundet via Font Awsome, men da man ved de fleste af deres vejrikoner skal have betalt medlemskab, har jeg set mig nødsaget til at tage screenshots af dem. For at indsætte vejrikonerne har jeg i første omgang lavet funktioner for hvert ikon, der opretter et img-element, og benytter .src til at ændre billedet til det givne billede i min billede-mappe (se figur 8). For at indsætte det korrekte billede ift. nuværende vejrtype, har jeg for hver dag oprettet et else-if statement som kigger på værdien i days[i].icon, og appender de tilhørende billede (se bilag 12). Hvis dette eksempelvis er === "rain" skal den givne div' appende regn-ikonet. For at være sikker på at alle mulige værdier i dette 'icon'

objekt har jeg eksempelvis indsat sol og sky-ikonet ved både ”partly-cloudy-day” og ”partly-cloudy-night”[9].

```
//Regn ikon
function createElementRain(tag, className) {
    let rainElement = document.createElement('img');
    rainElement.classList.add('rainIcon');
    rainElement.src = "../Billeder/rain.png"
    return rainElement;
}
```

Figur 8: Funktion til at indsætte vejrikon

Konklusion

Jeg har i denne rapport gennemgået mine løsninger samt løsningsovervejelser til de krav, som har været forelagt i eksamensopgavebeskrivelsen. Jeg har opfyldt 16 ud af 17 krav, hvor jeg bl.a. har brugt metoder såsom Date(), JSON.parse(), appendChild(), if-else statements, fetch() og mange andre. Jeg har udviklet min applikation, således at en bruger har mulighed for at oprette, ændre og slette sin profil, sætte sin yndlingsnyhedskategori, samt logge ind og ud. På forsiden får man vist de seneste nyheder, præcis tid, nuværende vejr, tidspunkt for solopgang og solnedgang, se vejrudsigten for de næste syv dage samt mulighed for at søge efter nyheder. Jeg har brugt localStorage for at lagre brugeres oplysninger, og har reflekteret over andre funktionaliteter hvor jeg kunne benytte localStorage. I rapporten har jeg desuden reflekteret over sikkerheden forbundet med at benytte localStorage til lagring af brugeres oplysninger.

Der har undervejs i applikationens udvikling været flere hændelser, hvor mine metoder ikke har fungeret. Eksempelvis har jeg flere steder i applikationen benyttet form-elementer, men ved mit log in, kunne jeg ikke benytte samme metode, samt er der en begrænsning på hvor mange gange man kan kalde på nyheds API’en uden brugerbetaling. Jeg har dog fundet løsninger på de hændelser som skulle være opstået undervejs. For at opfylde kravene stillet af eksaminator i eksamensopgavebeskrivelsen, har jeg fundet både simple og mere komplekse

[9] Visual Crossing, 2022

løsninger. Ved enhver løsning vil der være begrænsninger, og disse har jeg reflekteret over og kommet med løsningsforslag på, hvis jeg skulle videreudvikle min nyhedsapplikation.

Bilag

Bilag 1

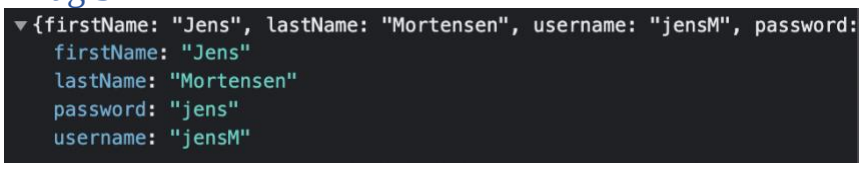
```
// Her indsætter jeg brugerens navn på homePage
function getUserInfoHomePage (){
    var profileInfoArr = JSON.parse(window.localStorage.getItem('profileInfo')); //Jeg parser profileInfo så jeg kan
rette i den

    let infoFirstNameParagraph = document.createElement('p'); // Opretter p-element
    infoFirstNameParagraph.innerHTML = profileInfoArr.firstName; // indsætter fornavn i p-elementet

    let infoLastNameParagraph = document.createElement('p'); // opretter p-element
    infoLastNameParagraph.innerHTML = profileInfoArr.lastName; // indsætter efternavnet i p-elementet

    let firstNameElement = document.getElementById('frontPageProfileInfoFirstName');
    firstNameElement.appendChild(infoFirstNameParagraph); // indsætter fornavnet i den rigtige div

    let lastNameElement = document.getElementById('FrontPageProfileInfoLastName');
    lastNameElement.appendChild(infoLastNameParagraph) // indsætter efternavnet i den rigtige div
}
getUserInfoHomePage()
```

Bilag 2

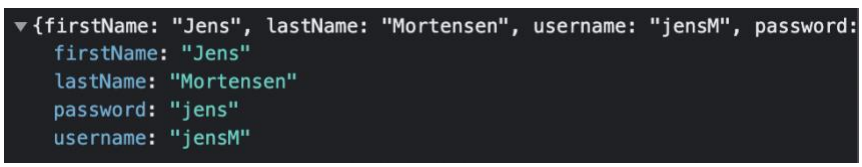
Link til videogennemgang af min applikation

https://www.loom.com/share/cd78b10549d94a23bc44fff6d3f85933

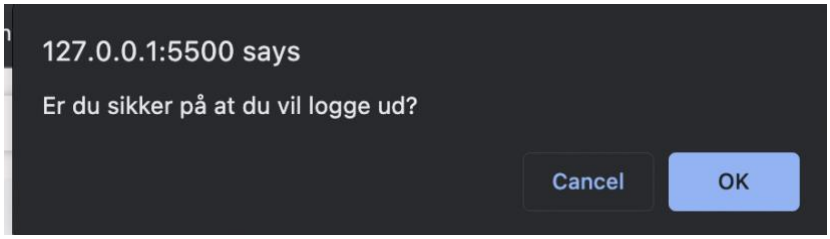
Bilag 3



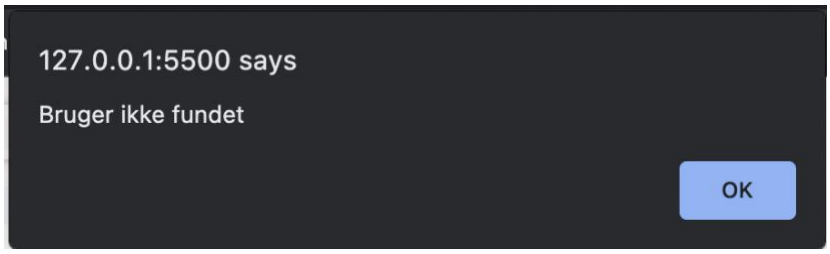
LocalStorage ved oprettelse.



LocalStorage efter opdatering.

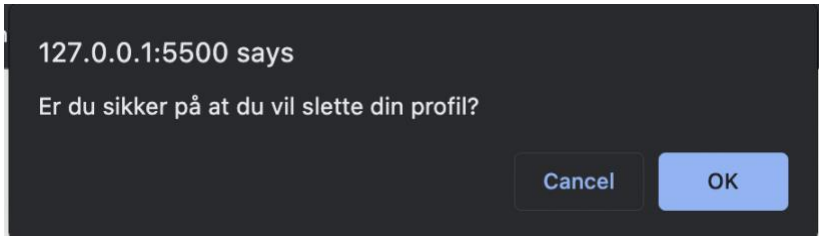


Pop-up ved log ud.



Pop-up ved bruger ikke fundet.

Pop-up ved forkert adgangskode.



Pop-up ved slet profil.

Bilag 4

```
// Her laver jeg mit log in
function logIn () {
    var profileInfoArr = JSON.parse(window.localStorage.getItem('profileInfo')); // parser så jeg kan læse
    var usernameField = document.getElementById('usernameLogin').value; // tager indtastet brugernavn
    var passwordField = document.getElementById('passwordLogin').value; // tager indtastet adgangskode
    var userUsername = profileInfoArr.username; // brugernavn fra localStorage
    var userPassword = profileInfoArr.password; // adgangskode fra localStorage

    if (usernameField === userUsername && passwordField === userPassword) { // Hvis brugernavn og
adgangskode er ens med det som er i localStorage
        window.location.replace('homePage.html') // omdirigeret til homepage.html
```

```
    } else if (usernameField !== userUsername) { // viser fejl
        alert("Bruger ikke fundet")
    } else if (usernameField === userUsername && passwordField !== userPassword) { // viser fejl
        alert("Forkert adgangskode")
    }
}
```

Bilag 5

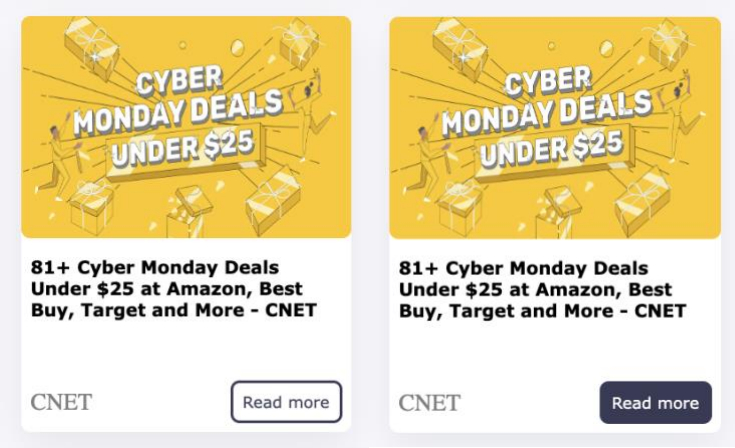
```
// Her opdaterer jeg brugerens fornavn
function updateFirstName () {
    let newFirstName = document.getElementById('UpdateProfileInputFirstName').value; // tager unput

    var profileInfoArr = JSON.parse(window.localStorage.getItem('profileInfo')); // parser så jeg kan ændre

    profileInfoArr.firstName = newFirstName; // sæter firstname til input

    window.localStorage.setItem('profileInfo', JSON.stringify(profileInfoArr)); // laver string til localStorage

    window.location.reload() // reloader så de steder hvor nanvnet er displayet bliver opdateret
    alert("Fornavn er succesfuldt opdateret")
}
```

Bilag 6

```
// Her lader jeg brugeren sætte sine yndlingskategorier PT 1
function addFavoriteCategory () {
    let category = document.getElementById('favoritesInput').value; // tager input

    const favoriteCategories = {
        category: category, // indsætter indput
    }

    window.localStorage.setItem('category', JSON.stringify(favoriteCategories)); // laver til en string
    window.location.reload() //siden reloader således at categorien kan vises på siden
}

// Her displayer jeg yndlingskategorien
function showFavCat () {
```

```
    var categoryArr = JSON.parse(window.localStorage.getItem('category'));

    let categoryList = document.createElement('li');
    categoryList.innerHTML = categoryArr.category;

    let element = document.getElementById('favoriteCategoryOne');
    element.appendChild(categoryList);
}
showFavCat()

// Her sletter jeg yndlingskategorien
function removeFavoriteCategory () {
    localStorage.removeItem('category') // tager fat i key 'category'
    window.location.reload() //siden reloader således at categorien slettes fra siden
}
```

Bilag 7



Ikke læst Læst

```
/* Styler read more knap hvis læst */
.smallArticleReadMoreButton:visited {
    background-color: #373B55;
    color: #FFFFFF;
}
```

Bilag 8

```
  // Her laver jeg et forloop, som går gennem artiklerne og henter nedenstående. Jeg har sat i < 6, da jeg som udgangspunkt kun vil have 6 artikler.
    for (let i = 1; i < 7; i++) {
```

```
    // Her siger jeg at der ved small articles skal laves en div for hver artikel
    let smallArticles = document.createElement('div');
    smallArticles.classList.add('smallArticles');


        // Her henter jeg billede af API arrayet og appender det i smallArticles
        let articlePicture = document.createElement('img');
        articlePicture.classList.add("smallArticlePicture");
        articlePicture.src = data.articles[i].urlToImage;
        smallArticles.appendChild(articlePicture);

        // Her henter jeg artiklens titel fra API arrayet og appender det i smallArticles
        let articleTitle = document.createElement('p');
        articleTitle.classList.add("smallArticleText");
        articleTitle.textContent = data.articles[i].title;
        smallArticles.appendChild(articleTitle);

        // Her henter jeg artiklens source fra API arrayet og appender denne i smallArticle
        let articleSource = document.createElement('div');
        articleSource.classList.add("smallArticleSource");
        articleSource.textContent = data.articles[i].source.name;
        smallArticles.appendChild(articleSource);

        // Her henter jeg url'en til artiklen fra API arrayet og appender denne i smallArticleSource
        let articleReadMore = document.createElement('a');
        articleReadMore.classList.add("smallArticleReadMoreButton");
        let readMoreText = document.createTextNode("Read more")
        articleReadMore.appendChild(readMoreText)
        articleReadMore.href = data.articles[i].url;
        articleSource.appendChild(articleReadMore);

    // Her sætter jeg de små artikler ind
    let otherArticles = document.getElementById('otherArticles');
    otherArticles.appendChild(smallArticles)
}
```

Bilag 9

```
// Her har jeg min search funktion som henter api'en med det søgeord som brugeren har indtastet
```

```
const search = async() => {
    let searchkeyWord = document.getElementById('searchBarFieldInput').value; // henter indtastet søgeord

    let otherArticlesSearch = document.getElementById('otherArticlesSearch');
    let apiKeySearch = '7accb7cc82db4eaaa46f779aba484c45'
    const searchUrl = `https://newsapi.org/v2/everything?q=${searchkeyWord}&from=2022-12-
08&sortBy=popularity&apiKey=${apiKeySearch}`


    const headersSearch = {
    'Authorization': `Bearer ${apiKeySearch}`
};

    const  responseSearch = await fetch(searchUrl,
        {
            method: 'GET',
            headers: headersSearch,
        });

    const data = await responseSearch.json();
    console.log(data);
```

Bilag 10

```
// indsætter ugedagen
const todaysWeekday = todaysDate.getDay();
    if (todaysWeekday === 1) {
        document.getElementById('dateWeekday').innerHTML='Mandag'
    } else if (todaysWeekday === 2) {
        document.getElementById('dateWeekday').innerHTML='Tirsdag'
    } else if (todaysWeekday === 3) {
        document.getElementById('dateWeekday').innerHTML='Onsdag'
    } else if (todaysWeekday === 4) {
        document.getElementById('dateWeekday').innerHTML='Torsdag'
    } else if (todaysWeekday === 5) {
        document.getElementById('dateWeekday').innerHTML='Fredag'
    } else if (todaysWeekday === 6) {
        document.getElementById('dateWeekday').innerHTML='Lørdag'
    } else if (todaysWeekday === 0) {
        document.getElementById('dateWeekday').innerHTML='Søndag'
```

```
}
```

Bilag 11

```
▼Object i
   address: "copenhagen, dk"
  ▼currentConditions:
     icon: "partly-cloudy-day"
     sunrise: "08:09:26"
     sunset: "15:45:31"
     temp: 6.8
   ▶[[Prototype]]: Object
  ▼days: Array(8)
   ▼0:
      icon: "cloudy"
      sunrise: "08:09:26"
      sunset: "15:45:31"
      temp: 6.5
      tempmax: 7.1
    ▶[[Prototype]]: Object
   ▶1: {tempmax: 5.7, temp: 5, sunrise: '08:11:10', sunset: '15:44:30', icon: 'rain'}
   ▶2: {tempmax: 5.1, temp: 4.5, sunrise: '08:12:51', sunset: '15:43:32', icon: 'rain'}
   ▶3: {tempmax: 4.5, temp: 3.9, sunrise: '08:14:31', sunset: '15:42:38', icon: 'rain'}
   ▶4: {tempmax: 3.8, temp: 3.5, sunrise: '08:16:08', sunset: '15:41:48', icon: 'rain'}
   ▶5: {tempmax: 3.5, temp: 3, sunrise: '08:17:43', sunset: '15:41:01', icon: 'rain'}
   ▶6: {tempmax: 2.2, temp: 1.4, sunrise: '08:19:15', sunset: '15:40:18', icon: 'rain'}
   ▶7: {tempmax: 1.4, temp: 0.5, sunrise: '08:20:45', sunset: '15:39:39', icon: 'rain'}
```

Bilag 12

```
// Her appender jeg det korrekte ikon ift array'et
// Dag 1
if (forecastData.days[1].icon === "rain") {
    forecastOneIconDiv.appendChild(createElementRain('img', 'rainIcon'))
} else if (forecastData.days[1].icon === "snow") {
    forecastOneIconDiv.appendChild(createElementSnow('img', 'snowIcon'))
} else if (forecastData.days[1].icon === "fog") {
    forecastOneIconDiv.appendChild(createElementFog('img', 'fogIcon'))
} else if (forecastData.days[1].icon === "wind") {
    forecastOneIconDiv.appendChild(createElementWind('img', 'windIcon'))
} else if (forecastData.days[1].icon === "cloudy") {
    forecastOneIconDiv.appendChild(createElemenCloud('img', 'cloudIcon'))
} else if (forecastData.days[1].icon === "partly-cloudy-day") {
    forecastOneIconDiv.appendChild(createElementSunAndCloud('img', 'partlyCloudyIcon'))
} else if (forecastData.days[1].icon === "partly-cloudy-night") {
    forecastOneIconDiv.appendChild(createElementSunAndCloud('img', 'partlyCloudyIcon'))
} else if (forecastData.days[1].icon === "clear-day") {
```

```
    forecastOneIconDiv.appendChild(createElementSun('img', 'sunIcon'))
} else if (forecastData.days[1].icon === "clear-night") {
    forecastOneIconDiv.appendChild(createElementSun('img', 'sunIcon'))
};
```

Litteraturliste

- Barasa, M. (2021, Januar 11). How to Use Local Storage with JavaScript. *The Engineering Education*. https://www.section.io/engineering-education/how-to-use-localstorage-with-javascript/

- *Date*. (5/12-22). Developer Morzilla. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

- *Everything*. (n.d.). newsAPI. https://newsapi.org/docs/endpoints/everything

- Font Awsome. https://fontawesome.com/. Senest besøgt 9/12-22

- Gagne, M. (2018, Marts 14). A Very Expensive AWS Mistake. *Medium*. https://medium.com/@morgannegagne/a-very-expensive-aws-mistake-56a3334ed9ad

- *Get started.* (n.d.). newsAPI. https://newsapi.org/docs/get-started#search

- Micheal, N. (2021, November 1). Storing and retrieving JavaScript obejcts in localStorage. Log Rocket. https://blog.logrocket.com/storing-retrieving-javascript-objects-localstorage/

- *Query Builder*. (n.d.). Visual Crossing. https://www.visualcrossing.com/weather/weather-data-services

- *Storage*. (10/10-22). Developer Morzilla. https://developer.mozilla.org/en-US/docs/Web/API/Storage

- Visual Crossing. (2022). Defining the icon set paramenter in the Weather API. *Visual Crossing*. https://www.visualcrossing.com/resources/documentation/weather-api/defining-icon-set-in-the-weather-api/

- WilliamG (2016, September 9). Making a live clock in javascript. *Stack Overflow*. https://stackoverflow.com/questions/39418405/making-a-live-clock-in-javascript

- *Window.confirm()*. (13/9-22). Developer Morzilla. https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm