Refleksjonsnotat

Arbeidsoppgave 1 – Skrevet av Arthur Leonard Thomassen

Dokumentet viser hvordan jeg har jobbet med oppgaven, hvilke vurderinger jeg har gjort rundt utseendet, hvordan jeg har løst koden, hva jeg har lært siden studiestart og hvordan jeg selv opplever at prosessen har gått så langt

Innhold

Hvordan jeg har jobbet med oppgaven

Metoder

- **Samarbeid med skolen**: Gode råd fra Amina og andre elever
- **Sparringspartner hjemme**: Gode råd fra samboer som er fullstack
- **God støtte**: Kaffe og Netflix eller Spotify på siden

Verktøy

- **JetBrains IntelliJ IDEA Ultimate**: Programmet som ble benyttet for å programmere
- **GitHub**: Brukt til versjonskontroll og lagring av prosjektet
- **Prettier**: Formatering av kode
- **StyleLint**: Brukt til å kontrollere kvalitet og struktur i CSS
- **StyleLint-order**: Ble benyttet for å sortere i alfabetisk rekkefølge innenfor CSS class selector
- **Notion**: Jeg er veldig glad i å skrive om studien og generelt og programmering

---

Forklaring

Jeg er vandt til å bruke IntelliJ, GitHub og hjelpeverkøy som jeg installerer i package.json i root folder. Det fine med å installere pakker som for eksempel Prettier i repositoriet med en Prettier konfigurasjonsfil, er at alle som koder i samme repository, vil benytte seg av de samme instillingene.

Versjonskontroll via GitHub gir oss god kontroll på koden, de ulike versjonene, endringer og masse mer. Jeg tror at Microsoft vil videre gjør en god med GitHub. Mange er skeptiske, men føler at de har gjort en god jobb med open source siden de begynte med å lage TypeScript helt tilbake til 2012.

Kilder jeg har lest under oppgaven

- **Mmdn - Basic HTML syntax**: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Basic_HTML_syntax
- **First rule of ARIA – do not use ARIA**: https://cerovac.com/a11y/2020/09/first-rule-of-aria-do-not-use-aria-and-why-it-is-so-important/
- **What are Self Closing Tags in HTML?:** https://www.scaler.com/topics/self-closing-tags-in-html/?utm_source=chatgpt.com
- **Stylelint configuring**: https://stylelint.io/user-guide/configure/
- **Flexbox**: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Hvilke vurderinger jeg har gjort rundt utseendet

Verktøy

- **Figma**: Link
- **html.to.design**: En Google Chrome extension for å generere enn nettside til Figma fil

---

Forklaring

Min glede med programmering, er å lage funksjonelle koder med kreativitet. Ønske mitt er å jobbe i en stor organisasjon (NAV, DNB, Sparebank1, Digipost osv.) hvor jeg kan jobbe med et team på å vedlikeholde og utvikle et produkt. En tanke kunne vært Fire Expert av Schneider Electric, hvor de har laget et nettbasert programmerinsverktøy for teknikkere, for at de kan programmere brannalarmsentraler. Det er litt kreativt hvor kombinasjon av hardware produkt møter frontend (og backend).

Når det kommer til design, har jeg ikke så mye sterke meninger annet enn at det må være bra og forutsigbart for brukere. Når jeg trenger inspirasjons liker jeg å bruke Aksel som kilde, de har mye god dokumentasjon og forklaring på hva som er bra og ikke så bra. Designet ser moderne, men samtidig veldig enkelt for brukere.

Når jeg programmerte nettsiden, valgte å kode først. Så brukte jeg html.to.design for å få nettsiden min over til Figma. Jeg lærte mye av elevene i løpet av uken på egenstudie om hvordan vi bruker Figma. Hvorfor vi kan lage komponenter og variabler, det ga en fin innføring til Figma for meg.

Aksel - inspirasjon

- **Aksel:** https://aksel.nav.no
- **Aksel designsystem:** https://aksel.nav.no/designsystemet

Lisensbruk

- **Lisens for bruk av SVG:** https://www.svgrepo.com/page/licensing/

Hvordan jeg har løst koden

Parametere

- **Total maks bredde nettside:** 1200px
- **Total maks bredde innhold:** 750px
- **font-family:** system-ui, sans-serif;

Verktøy

- **JetBrains IntelliJ IDEA Ultimate**: Programmet som ble benyttet for å programmere
- **GitHub**: Brukt til versjonskontroll og lagring av prosjektet
- **Prettier**: Formatering av kode
- **StyleLint**: Brukt til å kontrollere kvalitet og struktur i CSS
- **StyleLint-order**: Ble benyttet for å sortere i alfabetisk rekkefølge innenfor CSS class selector

---

Forklaring

Kobling av CSS fil

For å benytte en ekstern CSS fil, må den kobles til HTML filen som skal ha stylingen. Ved å bruke <link href="", vil man kunne definere hvilken fil som skal brukes.

Responsivt design

Med @media (width <= 1040px) i CSS gir det mulighet for å kunne gi tilpasset CSS når nettleserens vindu er på en bredde lik eller mindre enn 1040px.

Farger

Kombinasjon 1: Kort og seksjoner

Bakgrunn: mørk blågrønn (#004E52)
Tekst: lys grå (#E3E3E3)
Kontrast: 7,39:1 – oppfyller kravene
WCAG: Består AA og AAA for både normal og stor tekst, samt UI-komponenter

Kombinasjon 2: Bakgrunn og signatur

Bakgrunn: mørk blå (#070D17)
Tekst: grønn signatur (#09B94F)
Kontrast: 7,47:1 – oppfyller kravene
WCAG: Består AA og AAA for både normal og stor tekst, samt UI-komponenter

Tilgjengelighet

- Bilder: Ble lagt til med beskrivende alt-tekst `alt`.
- Lenker: Ble lagt til med `aria-label` der det er relevant. Alle lenker ble skrevet med `href=""`.

Bruk av rem, px og variabler

Variabler har blitt benyttet alle steder hvor en verdi må benyttes flere enn en gang.

- rem: Benyttes for skriftstørrelser for skalerbarhet, brukerer kan justere tekststørrelsen i nettleserens instillinger.
- px: Alt annet (marg, padding, bredde, høyde osv.) for presis kontroll.

Bruk av variabler i CSS filen

For å enklere vedlikehold og konsistens, brukes CSS-variabler der verdier brukes mer enn en gang.

```css
:root {
  --color-signature: #09b94f;
  --spacing-medium: 30px;
}

header {
  display: flex;
  color: var(--color-signature);
  font-size: var(--font-size-medium);
}
```

JavaScript

Dokumentasjon

- **querySelectorAll**: https://developer.mozilla.org/en-US/docs/Web/API/Element/querySelectorAll
- **mouseenter:** https://developer.mozilla.org/en-US/docs/Web/API/Element/mouseenter_event
- **mouseleave**: https://developer.mozilla.org/en-US/docs/Web/API/Element/mouseleave_event

cardContainer - Forklaring av kode

Koden finner først alle seksjoner som har klassen cardContainer og sender dem inn til funksjonen applySectionHover. Inne i denne funksjonen hentes alle elementer med klassen .card fra den aktuelle seksjonen, og for hvert kort legges det til addEvenListener for mouseenter og mouseleave. Når musepekeren holdes over ett kort, går koden gjennom de andre kortene i samme seksjon og legger på CSS klassen cardOtherHovered for å markere at de ikke er i fokus. Når musepekeren forlater kortet, fjernes denne klassen igjen fra alle kortene, slik at de går tilbake til den vanlige CSS koden. Effekten er at hover-oppførselen fungerer separat for hver seksjon, siden logikken begrenses til kortene inne i den spesifikke containeren som ble sendt inn.

```javascript
document.addEventListener('DOMContentLoaded', function () {
  document.querySelectorAll('.cardContainer').forEach((section) => {
    applySectionHover(section);
  });
  function applySectionHover(section) {  Show usages
    const items = section.querySelectorAll('.card');
    items.forEach((item) => {
      item.addEventListener('mouseenter', () => {
        items.forEach((other) => {
          if (other !== item) {
            other.classList.add('cardOtherHovered');
          }
        });
      });
      item.addEventListener('mouseleave', () => {
        items.forEach((other) => {
          other.classList.remove('cardOtherHovered');
        });
      });
    });
  }
});
```

menu - Forklaring av kode

Koden venter først til hele siden er ferdig lastet før den kjører. På den måten slipper vi å få feil hvis vi prøver å bruke elementer som ikke har rukket å lastes inn enda.

Deretter henter den ut to ting vi trenger: selve checkboxen og hamburger-icon som du klikker på. Disse to jobber sammen for å vise eller skjule menyen.

Det finnes en liten funksjon som styrer om man kan scrolle på siden eller ikke. Når menyen er åpen låses scrollen (overflow), så du ikke kan skrolle i bakgrunnen. Når menyen lukkes åpnes scrollingen igjen som normalt.

Koden tar også hensyn til de som bruker tastatur. Hvis du står på hamburger-ikonet og trykker Enter, så åpnes eller lukkes menyen på samme måte som når du klikker med musen. Det er lagt til støtte for gamle nettlesere ved å ha (|| menuEnter.keyCode === 13)

Til slutt passer koden på å oppdatere scrollefunksjonen hver gang menyen endres. Dermed oppfører alt seg riktig uansett om du klikker, trykker Enter eller på annen måte åpner og lukker menyen.

```
173  document.addEventListener('DOMContentLoaded', function () {
174    const menuToggle = document.getElementById('menu-toggle');
175    const hamburgerIcon = document.querySelector('.hamburger-icon');
176
177    function updateBodyOverflow() {  Show usages  new *
178      document.body.style.overflow = menuToggle.checked ? 'hidden' : 'auto';
179    }
180    hamburgerIcon.addEventListener('keydown', function (menuEnter) {
181      if (menuEnter.key === 'Enter' || menuEnter.keyCode === 13) {
182        menuToggle.checked = !menuToggle.checked;
183        updateBodyOverflow();
184      }
185    });
186    menuToggle.addEventListener('change', updateBodyOverflow);
187  });
```

Hva har lært siden studiestart og opplevelse så langt

Første undervisning - Velkommen og Intro til HTML
20.08.2025

Semantikk
Semantisk HTML gir struktur og mening til innholdet på en nettside.

I stedet for å bare style tekst og bokser, forteller du nettlesere og søkemotorer hva innholdet faktisk er. For eksempel gjør <h1> og <h2> det klart at noe er en overskrift, mens <nav> viser at en del av siden er en meny. Dette gjør siden mer tilgjengelig, lettere å forstå og bedre for søk. Uten semantikk ser alt bare ut som tilfeldige plasserte bokser, men med semantikk får innholdet kontekst og hirerarki.

<meta charset="UTF-8">
Linjen forteller nettleseren hvilket tegnesett nettsiden bruker.

Kort fortalt: det sikrer at bokstaver, tall og symboler (som æ, ø, å eller emoji) vises riktig uten det kan du risikere at spesialtegn ser ut som rare symboler eller spørsmålstegn.

<title>Min første nettside</title>
Definerer tittelen på nettsiden, den teksten som vises i fanen i nettleseren og som søkemotoren ofte bruker som overskrift i søkeresultater.

**<b> og **
<b> gir kun en visuell effekt og gjør teksten fet uten å tilføre mening. Skjermleseren ignorerer denne markeringen.

 derimot signaliserer at teksten er viktig og blir derfor fremhevet av skjermlesere. Men om man bruker overalt bare for å få fet skrift, mister det verdi og kan skape en støyete lytteopplevelse. Poenget er å bruke når noe faktisk har betydning, og la ren visuell styling løses med css eller <b> når det bare handler om utseende.

Grunnleggende HTML + Semantikk
27.08.2025

Viktig å bruke semantikk. Litt repetisjon fra forrige undervisning.

Linker jeg fant under forelesningen
- https://ndla.no/r/konseptutvikling-og-programmering-im-ikm-vg1/semantisk-html/0d94e80dec
- https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements

Semantiske tagger
<header> <footer> <nav> <main> <section> <article> <aside>

Ikke semantiske tagger

aria-labelledby
aria-labelledby="cards-heading" på <section> forteller skjermlesere at denne seksjonen skal identifiseres ved innholdet i elementet med id="cards-heading".
Dokumentasjon: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Reference/Attributes/aria-labelledby

Oppsummering

Jeg har lært veldig mye siden studiestart. Skulle jeg laget en ny nettside nå, ville vurderingene og koden i starten ha en helt annen start fra begynnelsen. Jeg har innsett hvor mye rammeverk gjør for oss, og det at jeg bedre forstår fundamentet og det grunnleggende vil gjøre sånn at jeg vil levere bedre kvalitet fremover.