YRKESHÖGSKOLAN
NOVIA

Tillgänglighet och användarvänlighet inom webbutveckling

Sebastian Renkonen

Examensarbete för ingenjör (YH)-examen

El- och automation

Vasa 2024

EXAMENSARBETE

Författare:	Sebastian Renkonen
Utbildning och ort:	El- och automationsteknik, Vasa
Inriktning:	Informationsteknik
Handledare:	Kaj Wikman

Titel: Tillgänglighet och användarvänlighet inom webbutveckling

---

Datum: 30.4.2024 Sidantal: 41

Abstrakt

Arbetet har utmynnat i en webbsida som byggts med primärt fokus på tillgänglighet och användarvänlighet. Syftet med detta var att sidan ska kunna nå ut till bredast möjliga publik och att användare oavsett eventuell funktionsnedsättning ska kunna ta del av den. För att följa den europeiska rättsakten om tillgänglighet beaktas riktlinjer ur Web Content Accessibility Guidelines (WCAG); en samling riktlinjer framtagna av World Wide Web Consortium (W3C).

För ändamålet har ramverk och bibliotek med öppen källkod använts som aktivt utvecklas av dess användare och som har grundläggande funktioner för tillgänglighet färdigt inbyggda. Öppen källkod prioriterades för att själv kunna utöka dess funktionalitet vid behov.

Front-end utvecklades med metaramverket Nuxt och använde sig av TailwindCSS samt komponentbiblioteket Shadcn/ui. Back-end använde även Nuxt för att kommunicera med den molnbaserade databasen Supabase. Samtliga teknologier har övervägts mot motsvarande alternativ och valts med beaktande av tillgänglighetsfunktioner, prestanda, digital integritet, och utvecklarupplevelse (DX).

Webbsidan uppnår kraven för AA-standarden angiven i WCAG och bibehåller även en visuell karaktär som urskiljer sig från motsvarande webbsidor. Typsnitten som används för brödtext är utvecklade för att vara lättlästa. Färgkontrasten mellan text och bakgrund är tillräcklig för att kunna läsas av användare med nedsatt syn. Sidan kan navigeras med antingen pekskärm, mus, eller tangentbord, och fungerar på de flesta moderna webbläsarna oavsett skärmstorlek.

Språk: svenska
Nyckelord: tillgänglighet, tillgänglighetsfunktion, användarupplevelse, användargränssnitt, WCAG, front-end

BACHELOR'S THESIS

Author:	Sebastian Renkonen
Degree Programme:	Electrical Engineering and Automation
Specialisation:	Information Technology
Supervisor(s):	Kaj Wikman

Title: Accessibility and user-friendliness in web development

---

Date: 30.4.2024 Number of pages:41

Abstract

The work has resulted in a website that has been built with accessibility and user-friendliness as its primary focus. The purpose of this was that the site would be able to reach out to as wide an audience as possible and that all users, no matter potential disabilities, could partake in it. To conform to the European accessibility act, the Web Content Accessibility Guidelines (WCAG) have been followed; a collection of guidelines developed by the World Wide Web Consortium (W3C). For this purpose, the frameworks and libraries used all have open-source code that is actively developed by its users, and which have basic functionality for accessibility already built in. Open source was prioritized to enable the expansion of functionalities as needed.

The frontend was developed with the meta-framework Nuxt and used TailwindCSS along with the component library Shadcn/ui. Backend also used Nuxt for communicating with the cloud-based Supabase database. All technologies have been compared to equivalent alternatives and chosen based on accessibility features, performance, digital privacy, and developer experience (DX).

The website meets the criteria for the AA-standard set in the WCAG while also maintaining a visual flair that distinguishes it from similar websites. The fonts used for body text are developed to be easy to read. The color contrast between text and background is sufficient to be read by users with impaired vision. The site can be navigated using a touch screen, mouse, or keyboard, and works on most modern web browsers regardless of screen size.

Language: Swedish
Key words: accessibility, accessibility features, user experience, user interfaces, WCAG, frontend

Innehållsförteckning

Begrepp

HTML	HyperText Markup Language, kod som definierar webbsidans struktur.
CSS	Cascading Style Sheets, kod som definierar webbsidans utseende.
JavaScript	Programmeringsspråk som används för att hantera webbsidans funktioner och beteende.
Frontend	Den del av en webbplats som slutanvändaren ser och interagerar med.
Backend	Den del av en webbplats där data lagras och skickas ut till användare. Databaser och servrar.
Full stack	Avser både frontend och backend tillsammans.
W3C	World Wide Web Consortium, industrikonsortium grundat av Tim Berners-Lee, som även grundat World Wide Web. Samarbetar med industrier, regeringar och forskare för att utveckla standarder och protokoll som främjar tillgänglighet, internationalisering, personlig integritet och säkerhet på webben.
WAI	Web Accessibility Initiative, Den grupp inom W3C som har hand om tillgänglighetsfrågor.
WCAG	Web Content Accessibility Guidelines, standardiserade riktlinjer för användarvänlighet på nätet, framtaget av W3C i samarbete med experter ur olika ledande företag och organisationer.

ARIA	Accessible Rich Internet Applications, teknisk specifikation framtagen av W3C för att främja tillgänglighet på webben.
Ramverk	En samling med användbar kod och funktioner som fungerar som verktyg inom ett programmeringsspråk.
API	Application Programming Interface, gränssnitt för program att kommunicera sinsemellan.
Pakethanterare	Laddar ned färdigskrivna program och bibliotek med öppen källkod.
SFC	Single-File Component, filer med ändelsen .vue där HTML, CSS och JavaScript hanteras i en och samma fil.
SEO	Search Engine Optimization, sökmotoroptimering är en praxis som förbättrar webbsidans chanser att dyka upp tidigt när relevanta sökningar görs med sökmotorer.

1 Inledning

Arbetet har gått ut på att från grunden utveckla en webbaserad kokbok. Webbplatsen ska uppfylla följande kriterier:

- Tillgänglighet på AA-nivå enligt senaste specifikationer från WCAG.
- Minimalt disk/server-utrymme och optimerad prestanda.
- Respektera användarnas integritet.
- Vara intuitiv att använda.
- Bibehålla en visuell karaktär som skiljer den från andra motsvarande websidor.

WCAG är en internationell standard för tillgänglighet på webben. Det finns tre olika nivåer:

- A som är miniminivån.
- AA som inkluderar allt från A-nivån och AA-nivån. Många organisationer strävar efter att möta kriterierna för denna nivå.
- AAA inkluderar samtliga krav från alla nivåer.

(Web Accessibility Initiative (WAI), 2020).

Eftersom AAA-nivån är striktare och sätter mer begränsningar på en webbsida så lämpar den sig för exempelvis statliga tjänster och banker, eller andra tjänster som kan vara obligatoriska att använda.

1.1 Uppdragsgivare

Arbetet har skett på beställning av en privat person som ämnar att driva sidan framöver. Personen i fråga har restaurangutbildning och kockar även på fritiden för vänner och bekanta. Eftersom hen ofta får förfrågningar om recept så beställdes en receptsida där recepten kan samlas upp och göras tillgängliga för alla.

1.2 Bakgrund

Anledningen till att så stor fokus läggs på tillgänglighet är, förutom personliga humanistiska värderingar, att över en fjärdedel av invånarna i EU som fyllt 16 år hade år 2023 någon form av funktionsnedsättning (Eurostat, 2024). Att anpassa webbsidan även för dem gör att sidan har potential att nå ut till fler användare. En annan orsak är Europaparlamentets direktiv från 2019 om tillgänglighetskrav för produkter och tjänster, för vilket lösningar bör ha implementerats av alla berörda senast 28 juni 2025 (Europaparlamentet, 2019).

2 Teknologier

För att säkerställa att de teknologier som används har stöd även i framtiden så används etablerade ramverk med öppen källkod som funnits över ett år, som aktivt uppdateras månatligen, och som laddas ner och används av andra utvecklare över tusentals gånger per månad. Enda undantaget är Wysimark som inte hade något lämpligt avgiftsfritt alternativ. Med öppen källkod kan utvecklingen av teknologierna fortsättas av vilken användare som helst även om personen eller organisationen bakom skulle sluta stöda dem. Öppen källkod ger också en trygghet att man inte använder sig av oönskad telemetri eftersom koden är fritt tillgänglig och kan granskas av allmänheten. Den teknologi som används har valts baserat på att den är utvecklad med bästa praxis för bland annat tillgänglighet och prestanda färdigt inbyggt.

Tanken bakom valen av teknologier är att vara tidseffektiv och inte spendera tid på att utveckla redan fritt tillgängliga funktioner, att undvika buggar och mänskliga fel så långt som möjligt, samt att ha en aktiv internetgemenskap som går att vända sig till med frågor och återkoppling.

2.1 Nuxt

Webbplatsen är byggd med Nuxt 3; ett metaramverk baserat på Vue som har öppen källkod och är skrivet i TypeScript. Vue är främst gjort för frontend-utveckling men Nuxt möjliggör full stack-utveckling med ett och samma ramverk.

2.1.1 SFC

Nuxt använder sig av Single-File Components, SFC, som är komponenter med filändelsen .vue där HTML, CSS och JavaScript kan skrivas i en och samma fil.

Kodexempel 1. SFC-struktur med data som kallas in till HTML.

```
<script setup>
import { ref } from 'vue'
const greeting = ref('Hello World!')
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```

Inom script-taggen skrivs logiken för komponenten, antingen i JavaScript eller TypeScript. Här importeras utomstående komponenter, antingen de som finns inom projektet eller moduler installerade via pakethanterare. Här definieras även konstanter, variabler och funktioner. Attributet *setup* som finns inom öppningstaggen gör att Nuxt automatiskt tar hand om så kallad boilerplate-kod som är repetitiv kod som annars måste skrivas för hand vilket ger mer utrymme för misstag att hända. Eftersom koden då blir mindre blir den också lättare att läsa. Setup-attributet gör även att koden optimeras under kompileringen genom att hoppa över mellansteg.

Inom template så skriver man den HTML som ska vara inom body-taggen på vanliga HTML-dokument. Man kommer åt JavaScript-variabler genom att lägga variabelns namn inom dubbla klammerparenteser eller genom att använda sig av v-bind inom HTML-taggen.

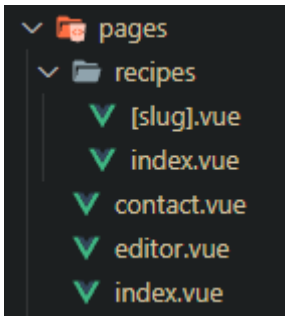
Kodexempel 2. v-for loopar igenom en lista och visar dynamiskt innehållet inom klammerparentes.

```
<SelectItem v-for="language in languages" :key="language" :value="language">
    {{ language }}
</SelectItem>
```

Inom style-taggen definieras CSS. Genom att lägga till ett attribut *scoped* inom öppningstaggen så tillämpas den CSS som följer enbart för den SFC-fil som den ligger i.

Med en SFC kan en hel sida representeras i en och samma fil. Man kan även använda en SFC för att kapsla in återkommande element, till exempel en knapp, så att så att elementet och alla tillbehör finns på en och samma plats. Sådana komponenter läggs i mappen components och kan därmed automatiskt importeras av andra .vue-filer genom att använda en tagg inom template-taggen vars namn motsvarar komponentens namn. Följaktligen blir en sida också lättare att läsa, såvida komponenterna har logiska och beskrivande namn. Man kan se vilka element en sida består av och gå in på detaljer och bakomliggande logik för varje element genom att utforska deras .vue-fil enskilt.

I Nuxt är sidorna och rutterna representerade av en fil- och mappstruktur, där varje sida har en egen fil av typen .vue. Routning mellan sidorna tas automatiskt om hand av Nuxt för alla filer i mappen Pages.



Figur 1. Mappstrukturen för sidor i Nuxt.

Den fil som ligger direkt under mappen pages och som döps till index.vue blir automatiskt första sidan för webbplatsen. Övriga sidor kommer man till i webbläsaren

Förutom sidor så finns en mapp för komponenter där man kan förvara .vue-filer med sådana element som kommer återanvändas flera gånger på sidan, alternativt filer som kräver så mycket kod att koden blir svårläst så man kan kapsla in den i sin egen fil.

Komponenter ur components-mappen importeras automatiskt av Nuxt i de sidor där de används.

2.1.2 Ekosystem

Nuxt är ett modulärt ramverk som har ett stort ekosystem med moduler för olika ändamål som kan installeras efter behov. På så vis hålls projektets storlek mindre än om alla verktyg skulle komma med från början vid varje nytt projekt vare sig de används eller ej. Nuxts ekosystem går dessutom att komplettera med moduler ur Vues ekosystem. Moduler som utvecklats och som underhålls av Nuxt-teamet inkluderar:

- NuxtImg för bildoptimering.
- NuxtLink för länkoptimering.
- Nuxt UI för färdigbyggda komponenter till användargränssnitt.
- Nuxt Content för innehållshantering och Markdown-översättning.
- Nuxt DevTools för utvecklarverktyg (Nuxt, 2024).

2.1.3 Rendering

Nuxt kan rendera webbsidor på olika sätt inom samma applikation:

- Static Rendering: Sidans innehåll genereras lokalt före den läggs upp på nätet. Lämpar sig för enkla sidor som inte har dynamiskt innehåll.
- Server Side Rendering, SSR: JavaScript körs på servern och sidan renderas före den skickas till klienten. Detta sätt valdes för projektet.
- Client Side Rendering, CSR: JavaScript skickas till klienten och sidan renderas när koden körs på deras maskin.
- Hybrid rendering: utvecklare sätter själva regler för varje rutt hur den ska renderas.
- Universal rendering: standard för Nuxt. Renderar första sidan på servern och använder sedan JavaScript som finns på sidan för att avgöra hur följande rutter ska renderas (Nuxt, 2024).

2.2 PNPM

PNPM, kort för Performant Node Package Manager, är en pakethanterare för JavaScript som bygger på NPM. En pakethanterare laddar ner moduler med programkod från en databas. NPM laddar ner dessa i en skild mapp på datorn och kopierar över dem till mappen node_modules som finns i roten av projektet. PNPM optimerar processen genom att skapa länkar för de nedladdade modulerna och i stället spara dessa i node_modules-mappen. Eftersom det inte finns fler än en kopia av enskilda moduler på hårddisken så sparas lagringsutrymme och processen snabbas upp avsevärt.

Eftersom PNPM använder länkar i mappen node_modules så kan den också frångå den platta mappstrukturen som NPM tillämpat, där alla moduler som laddas ner är överst i node_modules, även moduler som man inte explicit laddat ner utan som laddats ner eftersom de behövs av en annan modul. Problemet med den platta mappstrukturen är att man har tillgång till kod och funktioner som man kanske inte är medveten om och det kan orsaka buggar och problem, som nu undviks med PNPM (PNPM, 2024).

2.3 TypeScript

Programmeringsspråk som bygger på JavaScript och lägger till statisk typkontroll. TypeScript skrivs med samma syntax som JavaScript men med specificerade datatyper, till skillnad från JavaScript som gissar datatypen utifrån innehållet av data. Med statisk typkontroll får man under programmeringen felmeddelanden när data används på ett sätt som är felaktigt för datatypen i stället för att en bugg uppstår under körningen. TypeScript översätts vid kompilation till JavaScript så att webbläsare kan köra koden.

2.4 TailwindCSS

TailwindCSS, eller bara Tailwind, är ett CSS-ramverk där CSS skrivs in direkt i HTML som en klass inom den tagg som man vill ska påverkas. Med Tailwind så ser man direkt vilken CSS som påverkar ett element genom att se på själva elementet. När sidan kompileras minimerar Tailwind all CSS så att endast det som används kommer med i produktion.

Tailwind underlättar följsam design eftersom man kan använda ett prefix framför olika klasser för att markera vid vilken skärmstorlek som värdena för klassen ska börja gälla. Prefixet *sm* gör att klassen börjar gälla vid en skärmbredd av minst 640 pixlar, *md* börjar gälla vid 768 pixlar, och så vidare. Det finns även ett motsvarande prefix för om klassen ska gälla enbart för mörkt läge:

```
<div class="w-16 sm:w-20 md:w-28 bg-white dark:bg-black">
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolor
</div>
```

Eftersom Tailwinds klasser är mer begränsade än vanlig CSS så är det mindre krävande att skapa en enhetlig sida med jämna avstånd och färger som fungerar ihop. Vill man ändå använda något utöver de befintliga klasserna så kan man använda så kallade arbiträra värden där man skriver det önskade värdet inom hakparentes:

```
<div class="text-[#18c241]">
  Specifik grön text
</div>
```

Om mellanslag ingår i det arbiträra värdet så ersätts de med golvstreck. Man kan antingen ange arbiträra värden för en del av en klass, såsom färgen på en text, eller så kan hela klassen vara arbiträr.

2.5 Shadcn/ui

Shadcn/ui är ett bibliotek med olika UI-element som ofta används på webben. Biblioteken har öppen källkod och laddas direkt ner i projektet under mappen components. Man har därför explicit kontroll över varje element själv och kan göra vilka ändringar man vill. Alla komponenter är byggda med tillgänglighet i åtanke och stöder därför direkt tangentbordsnavigering och aria-funktioner. Det snabbar upp utvecklandet avsevärt eftersom UI-element kan vara tidskrävande och repetitiva att konstruera och kan kräva många iterationer för att få till exempel ARIA-teknologi att fungera korrekt.

2.6 Supabase

Supabase är en online databastjänst som bygger på PostgreSQL; en avancerad databashanterare med fler funktioner än motsvarande SQL-databaser. Supabase tillhandahåller ett smidigt användargränssnitt för att hantera databasen, samt en API för att kommunicera med den. Supabase har öppen källkod och är avgiftsfritt under vissa datamängder.

2.7 PostCSS

PostCSS är en CSS-kompilator som sköter om autoprefixering, minifiering och polyfills för CSS. Autoprefixering är processen att lägga till de prefix som behövs för olika klasser så att de ska fungera i olika webbläsare.

Minifiering är processen att krympa CSS-filer så att det tar så lite utrymme som möjligt. Onödiga mellanslag, tabbar och radbrytningar raderas, namn byts ut till förkortningar och dupliceringar slås ihop. Även färgvärden kan komprimeras. Minifieringen sköts av modulen CSSNano som installeras skilt för PostCSS.

Polyfills är också till för att CSS ska fungera i olika webbläsare, men är till för att få nyare CSS-funktioner att fungera på webbläsare som ännu inte stöder dessa. Detta sker genom att kompilera koden till motsvarande äldre CSS.

2.8 Wysimark

Wysimark är en WYSIWYG-redigerare för Markdown-text. WYSIWYG är kort för ‘what you see is what you get’, en benämning på textredigerare där man ser ändringarna man gör i realtid. Markdown är ett lätt sidbeskrivningsspråk för att formatera text. Det har stöd för rubriker, listor, textegenskaper, länkar och bilder. En text i Markdown kan tolkas med hjälp av Nuxt-modulen Nuxt Content för att visas som HTML.

2.9 Vercel

Vercel är ett webbhotell med Git-integration. När man pushar sin kod till git så kompileras den och webbsidan uppdateras direkt. Tjänsten är gratis för webbsidor med litet trafik och när gränsen överskrids får man ett meddelande. Om man väljer att inte betala så kan man pausa sidan tillfälligt till nästa månad.

3 Metoder

Webbsidans olika beståndsdelar gås igenom turvis och systematiskt. Först görs en snabb marknadsanalys för att se vilka webbläsare som används mest och som måste tas i beaktande. Sedan väljs typsnitt för sidans brödtext med stor vikt på lättlästhet, och ett mer utsmyckat typsnitt väljs för rubriker. Sedan väljs sidans huvudsakliga färgpalett så att text och bakgrund har tillräckligt hög kontrastskillnad. Efter det ordnas sidans layout så att den avstånden mellan olika element är jämna och så att tillhörande element grupperas med varandra. Layouten görs även följsam så att den fungerar på samtliga skärmstorlekar. Webbsidan görs navigerbar för tangentbord med hjälp av semantiskt korrekt HTML och tydliga markörer för markerade element. Sist så analyseras webbsidan för eventuella brister som korrigeras.

3.1 Stöd för webbläsare

När nya teknologier utvecklas för webben så behöver webbläsarna implementera stöd för funktionerna för att de ska kunna dras nytta av. Stödet för webbläsare utvecklas i efterhand och kräver att användarna sedan laddar ner uppdateringarna. Denna diskrepans betyder att man som webbutvecklare inte omedelbart kan ta i bruk alla teknologier utan att först se till att ens huvudsakliga målgrupp har möjlighet att använda sig av dem.

Statcounter erhåller en tjänst som visar grafer över webbläsarnas användarstatistik. Där kan statistik fås över vilka webbläsare som för tillfället är de mest använda i världen.

Figur 2. Statistik över olika webbläsares användning globalt (StatCounter, 2024).

Genom att jämföra med användarstatistiken för Finland så ser man att Firefox är betydligt mer populär här och bör också tas i beaktande under utvecklingen.

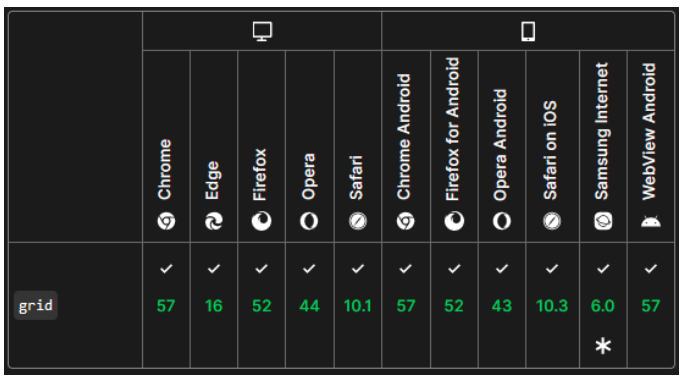
Figur 3. Statistik över olika webbläsares användning begränsat till Finland (StatCounter, 2024).

Med Chrome och Safari i topp kan man sedan jämföra användningsgraden av olika versioner.

Figur 4. Statistik över mest använda versioner av webbläsare globalt (StatCounter, 2024).

De flesta användarna finns på mobila plattformar, vilket betyder att en stor del av marknaden faller bort ifall man inte utvecklar för mobila enheter. 2,15 % av användarna har ännu version 120 som är en äldre version av Chrome. Den andelen av potentiella kunder faller bort ifall man använder teknologier som fått stöd först i 121. Idealet är att samtliga läsare stöds men i praktiken har man inte alltid resurser för att underhålla det och bör då vara medveten om vilka plattformar man önskar prioritera.

På sidan MDN Web Docs, som är Mozilla's öppna dokumentation för webstandarder, kan man se vilka som är de tidigaste versionerna av olika webbläsare som stöder specifika funktioner.



Figur 5. Lista från MDN över tidigaste webbläsarversionerna som stöder CSS-funktionen grid (Mozilla, 2024).

Även caniuse.com listar olika versioners stöd med en angiven procent av användare som direkt kan ta i bruk teknologin, samt för vilka tidigare versioner av webbläsare som funktioner stöds. Sidan använder sig av statistik från ovannämnda Statcounter.

Figur 6. Globalt stöd för bildformatet AVIF (Deveria & Schoors, 2024).

PostCSS är ett JavaScript-bibliotek som kompilerar all CSS före produktion så att den omvandlas till sådan CSS som även äldre webbläsare klarar av. Med hjälp av det kan man som utvecklare använda sig av nya teknologier som underlättar programmeringen utan att oroa sig för att arbeta kring webbläsarstödet och i värsta fall behöva implementera flera lösningar för samma problem. PostCSS använder sig för *polyfills* som betyder att den extra kod som krävs för att göra funktioner tillgängliga över flera webbläsare fylls i automatiskt vid kompilering. Genom att använda PostCSS behöver man endast se till att PostCSS är uppdaterat, i stället för alla andras webbläsare.

3.2 Text

Olika typsnitt har olika funktion och lämpar sig för olika sammanhang. För att sidan ska ha ett enhetligt utseende väljs ändå så få typsnitt som möjligt. De olika användningsområdena för typsnitten begränsades till rubrik, brödtext, siffror, och effekt. Rubrikerna har typsnittet Soria som är ett typsnitt i lätt jugendstil med seriffer. Typsnittets utsmyckningar ger sidan mer karaktär. För brödtexten används typsnittet Lexend eftersom det utvecklats för att vara lättläst. För siffror och taggar används typsnittet Sono, vars tecken har samma bredd. Det gör att bredden för elementet som innehåller tecknen är densamma oavsett vilket tecken som visas, och därmed påverkar det inte heller sidans layout. Typsnittet Sriracha har en handskriven karaktär och används för att skapa effekten av att namnet på ett recept har skrivits med penna på ett fotografi av maträtten.

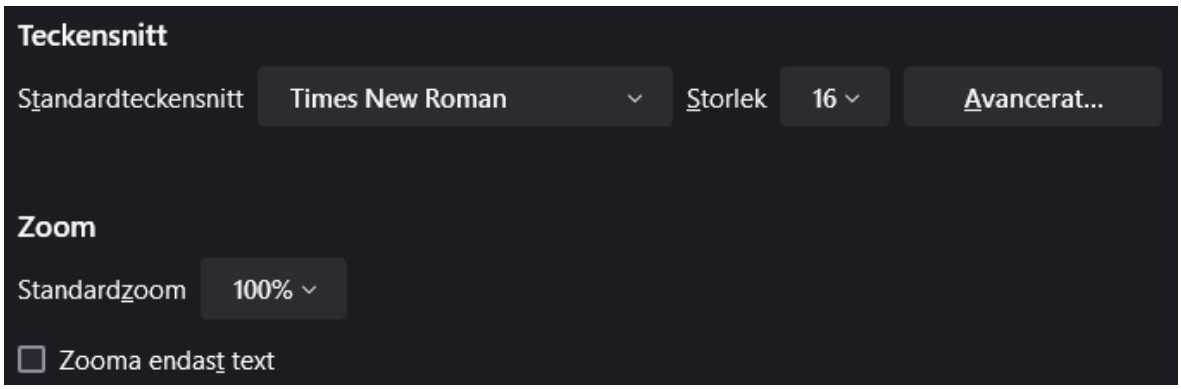
Rubrik	Soria
Brödtext	Lexend
Siffror och taggar	Sono
Effekt	Sriracha

Figur 7. Alla de typsnitt som används för sidan.

3.2.1 Enheter

Det finns inga krav i WCAG på hur stor text måste vara så länge den är läsbar. Standardstorleken för text i Tailwind är 1 rem. Genom att använda enheten rem (kort för *root element*) så kan storleken av texten ändras i webbläsarens inställningar enligt användarens egna preferenser. 1 rem motsvarar 16 pixlars höjd på en webbläsare med standardinställningar. Användningen av rem gör även att texten förstoras om man zoomar in. Om pixlar skulle användas som enhet skulle texten förbli den angivna höjden i pixlar oavsett preferens eller zoom. Viktigt är även att ingen särskild textstorlek ställs in för hela dokumentet, eftersom det skulle påverka samtliga element och den angivna storleken i rem skulle kunna betyda något annat än vad användaren definierat.

Med denna teknik så kan användare ändra storlek på text även utan utomstående hjälpmedel, samt zooma in upp till 200 %, och således uppfylls kriterierna för WCAG 1.4.4 (Accessibility Guidelines Working Group, 2023d).



Figur 8. Standardinställningar för typsnitt och zoom i webbläsaren Firefox.

3.2.2 Textalternativ

Allt innehåll som inte är text bör ha ett textalternativ så att blinda användare kan få samma information genom textläsare. Om innehållet i fråga endast används som dekoration så ska det i stället kunna ignoreras av skärmläsare. Samtliga fotografier kommer med en alt-text, vilket är en text som läses upp för att beskriva en bild när skärmläsare kommer över den. Om ingen alt-text anges så läses i stället URL för bilden upp, vilket kan ta relativt länge utan att säga så mycket. Dekorativa element är sådana som skulle kunna tas bort utan att påverka användarens förståelse av sidan. För att skärmläsare ska hoppa över dessa kan alt-texten deklareras som tom.

Figur 9. Bakgrunden för denna sektion är en dekorativ bild.

Bakgrunden bakom de senaste recepten skulle kunna beskrivas med *fönster* eller *stjärnhimmel* som alt-text, men det skulle inte tillföra användaren något när de ska laga mat, så alt-texten lämnas tom. Fönsterkarmen och gardinstången är en skild bild som används som ram för att kunna anpassas för skärmstorlek utan att stjärnorna på himmeln ska dras ut till fel proportioner. Bilder som definieras för ramar läses inte upp av skärmläsare eftersom de är angivna för CSS och inte för HTML. Inga åtgärder krävs därför för ramen. Bilderna av maträtterna kan beskrivas av namnet för rätten. Dock så läses namnet för en och samma rätt upp två gånger om man bara lägger namnet som alt-text, och därför användes i det här fallet attributet *aria-labelledby* inom bildens HTML-tagg. Med aria-labelledby kan man referera till ett annat element som ska användas som textbeskrivning, i det här fallet det element som innehåller namnet på rätten:

Kodexempel 3. Bilden för ett recept får sin alternativa text av elementet recipe_name.

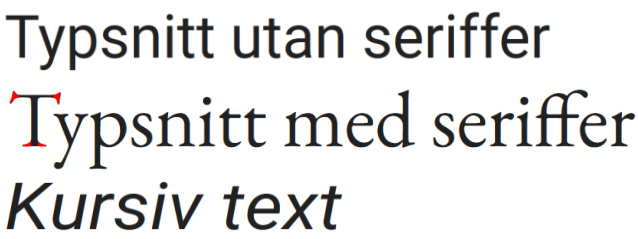
```
<NuxtImg :src="recipe.thumbnail" aria-labelledby="recipe_name" width="210" height="210"
```

Kodexempel 4. Elementet där receptets namn visas ges ett id med värdet recipe_name.

```
<CardTitle id="recipe_name">
  {{ recipe.name }}
</CardTitle>
```

3.2.3 Dyslexi

Vissa typsnitt lämpar sig bättre för dyslektiker än andra. Generellt så rekommenderas sans-seriff-typsnitt som har gott med mellanrum mellan tecken så att de inte ska maskera varandra eller flyta ihop. Kursiv text är svårare att läsa och bör användas sparsamt, till exempel för att understryka enskilda ord. (Luz Rello, 2013).



Figur 10. Exempel på sans-serif, seriff, och kursiv. Seriffer markerade i rött.

Till först övervägdes typsnittet OpenDyslexic för brödtext, ett typsnitt som enkom tagits fram för dyslektiker. OpenDyslexic har tjockare tecken nertill för att förhindra att läsaren ska rotera dem, och varje bokstav är unik eftersom tecken som brukar vara spegelvända motsvarigheter, som b och d, inte ska kunna misstas för varandra.

Figur 11. Exempel på typsnittet OpenDyslexic.

Anledningen till att OpenDyslexic inte användes är för att senare studier visat att skillnaden i läshastighet inte påverkats avsevärt av typsnittet i sig, utan att motsvarande effekt kunde fås med kontrolltypsnittet om det kompenserades så att tecken- och ordavståndet var det samma (Joseph & Powell, 2022).

I stället valdes typsnittet Lexend för brödtext. Lexend har utvecklats för allmän läshastighet och uppfyller de ovannämnda huvudsakliga rekommendationerna att tecken ska vara utan seriff och ha tillräckligt med avstånd från varandra (Shaver-Troup, 2018). Lexend kommer i flera olika varianter där tecknens bredd och avstånd är större för att underlätta läsandet. Sist och slutligen användes standardversionen av Lexend men det går i ett senare skede att ge användaren själv möjlighet att bestämma typsnitt och ändå bevara utseendet.

Lexend

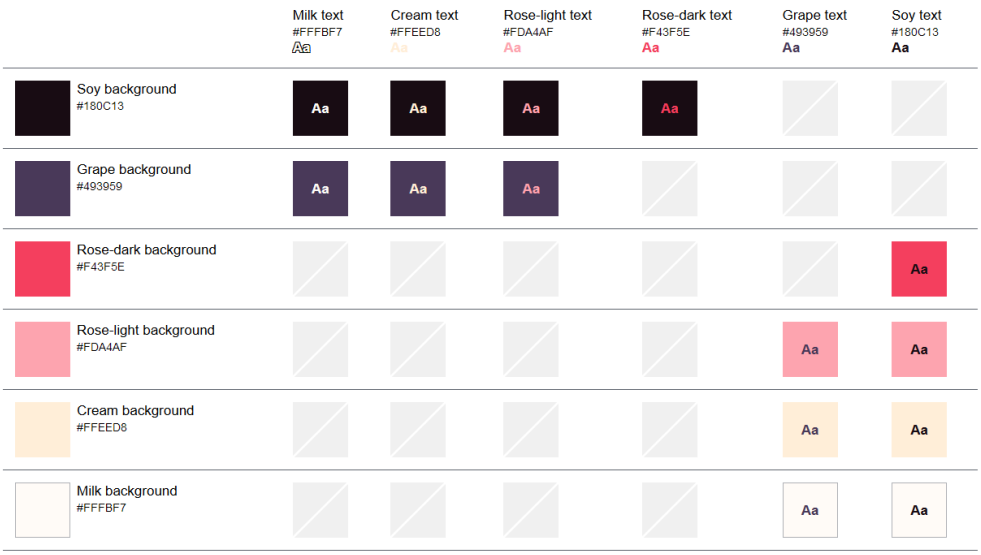
Lexend Giga

Figur 12. Exempel på typsnittet Lexend och varianten Giga.

3.3 Färg

Då kontrastskillnaden mäts mellan texten och bakgrunden som texten ligger på så bör den aldrig vara under 7:1, förutom om textens höjd är större än eller lika med 24 pixlar, för vilket kontrasten inte bör vara under 4.5:1 (Accessibility Guidelines Working Group, 2023a).

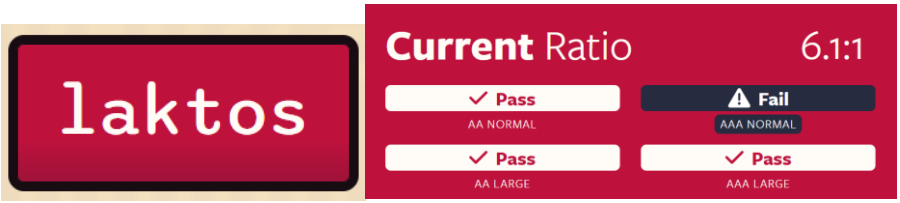
Genom att skapa en matris över alla valda färger och jämföra kontrasten så kan man lättare se vilka färger som lämpar sig för vad. I matrisen visas samtliga kombinationer med lägre kontrast än 4.5:1 som en grå ruta med streck över. Färgerna *soy* och *grape* lämpar sig som bakgrundsfärg mot *milk, cream* och *rose-light* som förgrundsfärg, eller vice versa. Enda kombinationen som inte lämpar sig är *grape* och *rose-dark*.



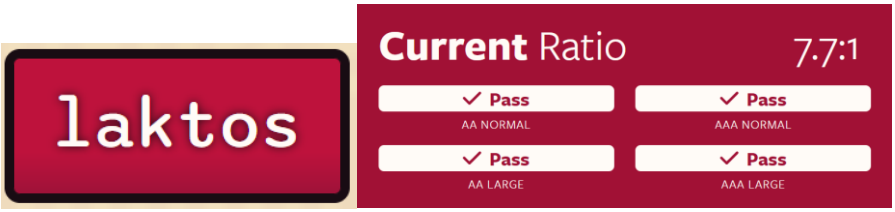
Figur 13. Matris över färgkombinationer med över 4.5:1 kontrast (Toolness, 2024).

Denna matris visade enbart kontraster över 4.5:1 som är kravet för AA-standarden. För att säkert uppnå AAA-standarden med kontrast över 7:1 så jämfördes kontraster enskilt.

För att uppnå kravet på 7:1 kan man använda sig av skuggning eller konturer kring texten för att höja kontrasten (Institute for Disability Research, Policy and Practice, 2021). Det här användes för rutorna med allergener så att de skulle kunna vara röda som är en varningsfärg, med vit text för att urskilja sig från kategoritaggarna som har svart text. Skillnaden kan verka minimal men den är avgörande.

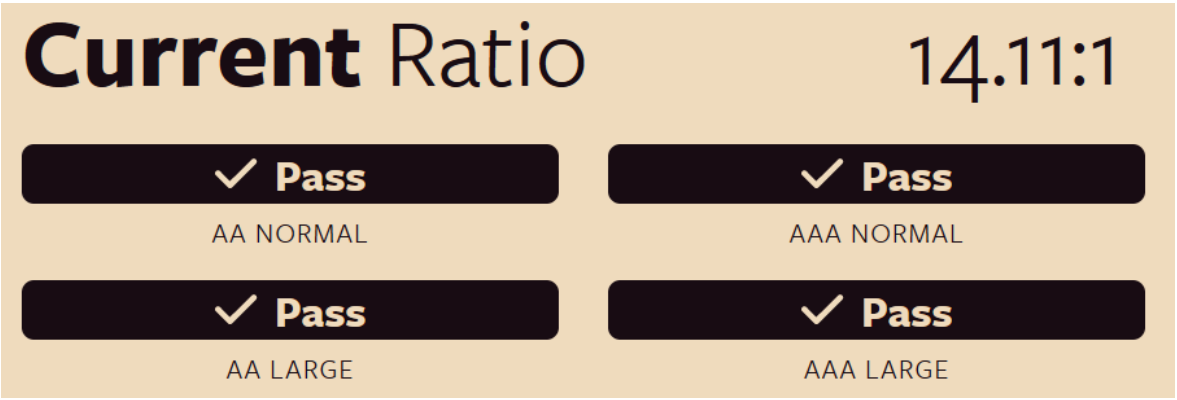


Figur 14. Kontrasten där den var som lägst mellan vit text och röd bakgrund var otillräcklig.



Figur 15. Med en svart skugga kring texten så var kontrasten som lägst 7.7:1, alltså godkänt.

Bakgrundsmönstret som förekommer på varje sida består av två färger, och om man jämför den mörkare av de två med förgrundsfärgen så är kontrasten långt över tillräcklig.



Figur 16. Textfärg över bakgrundsfärg (OddBird, 2024).

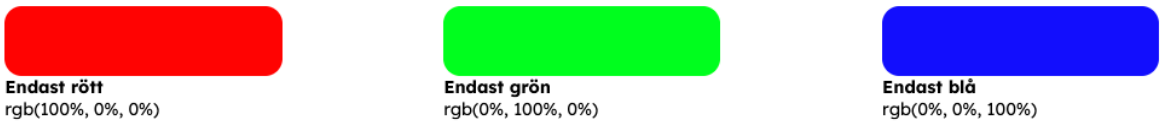
3.3.1 Taggar med dynamisk bakgrundsfärg

Recepten har taggar som kategoriserar dem. Taggarna ger en överblick av receptet och man kan söka på recept enligt taggar. Varje tagg har olika bakgrundsfärg som ska kunna ställas in när taggen skapas. För att hålla kontrastkraven på över 7:1 mellan namnet och bakgrunden för taggen så behöver användarens input begränsas.

De vanligaste systemen för att representera färg inom webbutveckling är hexadecimal, RGB, och HSL. I ett hexadecimalt färgsystem representeras mängden rött, grönt och blått i en given färg med ett 8-bitars värde representerat av två hexadecimala karaktärer från 0 till f, där 0 är minst och f är högst. En färg skrivs ut med ett inledande nummertecken följt av sex karaktärer, två för var färg. #000000 är då svart och #ffffff är vitt.

Figur 17. Grundfärger med hexadecimala värden.

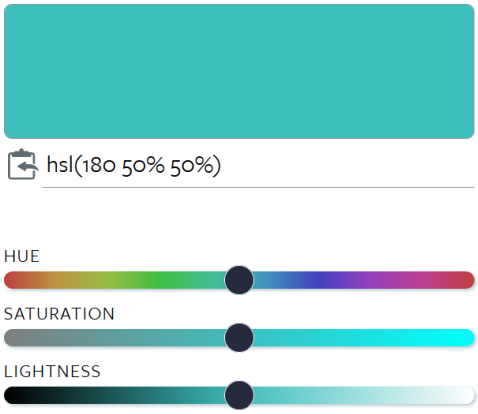
RGB står för ‘Red Green Blue’, namngett efter värdena som används. I ett RGB-färgsystem fungerar det i princip likadant men den hexadecimala representationen är utbytt mot procentenheter. RGB(0%, 0%, 0%) är då svart och RGB(100%, 100%, 100%) är vitt.



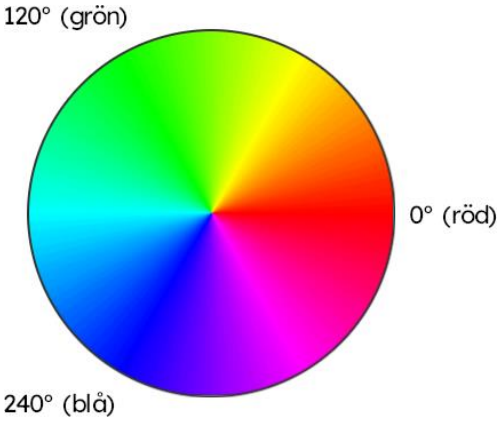
Figur 18. Grundfärger med RGB-notering.

Ingen av dessa system lämpar sig för ändamålet eftersom inget enskilt värde används för att direkt kontrollera ljusheten eller kulörtonen.

HSL-färgsystemet däremot avgör färgen baserat på värden för kulörton, mättnad och ljushet (Hue, Saturation, Lightness) Kulörtonen ställs in som grader mellan 0° och 360° där 0° är rött, 120° grönt, 240° blått och 360° är rött igen. Graderna brukar illustreras med ett färghjul. Mättnaden är hur svag eller djup färgen är på en skala från 0–100 %, där 0 % är grått och 100 % är stark färg. Ljushet är också på en skala från 0–100 %, där 0 % är helt svart och 100 % är helt vitt.

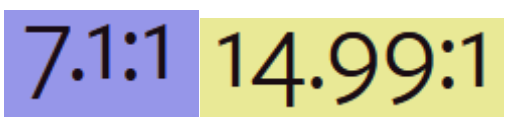


Figur 19. HSL-färg med medianvärdet av samtliga parametrar (OddBird, 2024).



Figur 20. Ett färghjul som förklarar kulörtonens rotation.

Genom att lägga en text överst med den textfärg vi avser använda så kan kontrasten kontrolleras för samtliga 360 grader av kulörtoner genom att svepa igenom dem. Med en textfärg som har värdet hsl(326.24 34.47% 7.1218%) så har de starkaste färgerna som uppnår kontrastkraven en mättnad på 65 % och ljushet på 75 % och då är kontrasten mellan 7.1:1 och 14.99:1. Resultatet blir dock enbart pastellfärger



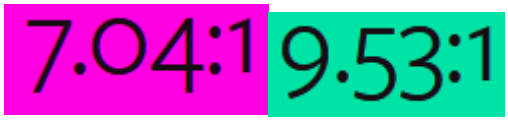
Figur 21. Lägst och högst kontrast med HSL-färger (OddBird, 2024).

Ett relativt nytt färgsystem som heter OKLCH baserar i stället sina färger på ljushet, krominans och kulörton (Lightness, Chroma, Hue). Principen är ungefär samma som för HSL men färgsystemet baserar sig på hur människan, snarare än datorer, uppfattar färg. Det

innebär färre fluktueringar i kontrast och OKLCH lämpar sig därmed bättre i tillgänglighetssyfte. Det har även en utvidgad färgrymd jämfört med tidigare nämnda färgsystem, vilket innebär att man med nyare skärmar kan se fler färger om OKLCH används.

Med OKLCH mot samma textfärg fås djupare färger med en ljushet på 74 % och krominans på 0.325. Eftersom ljusare färger också går att använda så fås ett större färgspektrum med både djupa färger och pastellfärger. Kontrasten mot textfärgen blir som lägst 7.04 och som högst 9.53:1 med nuvarande inställningar. Färgerna blir då lättare att tämja.

Figur 22. Motsvarande färger i OKLCH är djupare (OddBird, 2024).



Figur 23. Lägst och högst kontrast med OKLCH-färger (OddBird, 2024).

Figur 24. En ny tagg kan lätt skapas med starka OKLCH-färger.

3.4 Layout

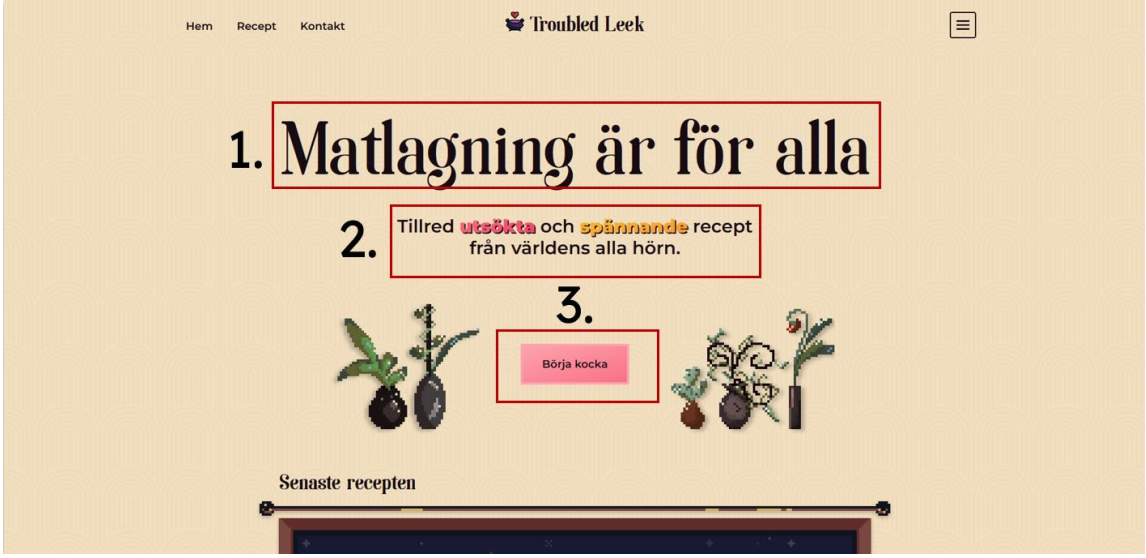
Elementens form och placering på sidan inverkar på hur de uppfattas av användare. Viktigare element får ta upp mer utrymme och knappar och länkar bör vara tillräckligt stora för att kunna tryckas på av mobila användare. Elementen bör ha tillräckligt med mellanrum för att kunna urskiljas från varandra, men grupperade element bör också vara tillräckligt nära varandra för att visa att de hör samman. Detta bör implementeras så att det fungerar oavsett skärmstorlek.

3.4.1 Visuell hierarki

För att leda användare av sidan igenom innehållet i den ordning som det är tänkt att läsas så kan man använda sig av visuell hierarki. Det finns få studier om visuell hierarki men det har sina rötter i gestaltpsykologin och är en etablerad praxis inom visuell design. Bland andra Google förespråkar visuell hierarki i sitt designsystem Material Design eftersom det underlättar för skärmläsare (Accessible design, 2021).

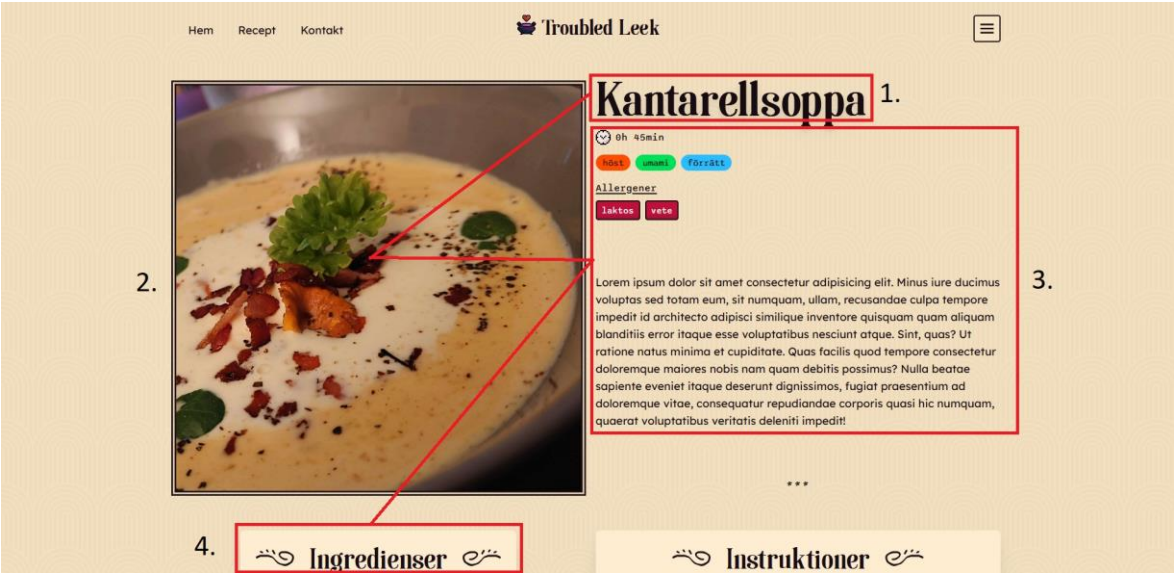
Canva, känt för sitt webbaserade grafiska designverktyg, tillhandahåller en guide för visuell hierarki och beskriver det där som ”ordningen i vilken människor tar in information på en sida” och att det är ”ett system för att prioritera element så att de är lättförstådda.” Man beskriver i guiden de två vanligast förekommande mönstren för visuell hierarki som **F-mönster**, där läsaren läser från vänster till höger och kan skanna dokumentets innehåll för rubriker och dylikt längs med vänstra kanten, samt **Z-mönster** för hierarkier med mindre tät text. (Marshall, 2020).

På första sidan är avsikten att läsaren först ska se rubriken med tanken bakom sidan, sedan en underrubrik som kort berättar vad sidan kan användas för, och sist ska blicken landa på en knapp som uppmanar användaren att ta i bruk sidan.



Figur 25. Hierarkin för textelementen på första sidan.

För att leda blicken till första punkten i figuren ovan används ett mer dekorativt typsnitt med större teckenstorlek än någon annanstans på sidan och texten har hög kontrast emot bakgrunden. Följande punkt har mindre teckenstorlek med ett mer avskalat typsnitt utan seriffer. Sist är knappen som har lägre kontrast jämfört med bakgrunden och som tar upp mindre utrymme på skärmen än föregående element.



Figur 26 . Hierarkin i sidan för ett recept följer ett omvänt Z-mönster.

Sidan för ett recept har en hierarki som följer ett omvänt Z-mönster. Först ser läsaren namnet på receptet, eftersom det stort och har hög kontrast mot bakgrunden. Till näst

landar blicken på bilden av receptet, som inte kommer först trots att det tar upp halva skärmen. Detta eftersom den inbördes kontrasten mellan olika föremål i bilden är lägre än för receptets namn. Den inbördes kontrasten kommer att variera från bild till bild, men kommer generellt sett inte komma i närheten av så olika färger som svart mot ljusbeige.

Sedan dras blicken till de färgglada taggarna som ännu inte lästs eftersom texten jämförelsevis är så liten. All text för sådan information är samma storlek eftersom allt är lika viktigt. De kan ändå urskiljas från varandra genom färg och form. Tillredningstiden visas bredvid en ikon av en klocka. Taggarna med kategorier är pillerformade med vit text och varierande bakgrundsfärg. Allergenerna är fyrkantiga med rundade hörn och har alla röd bakgrundsfärg med svart kontur och vit text. Sektionen för tilläggsinformation följer internt ett F-mönster. Detta är också anledningen till att just ett *omvänt* Z-mönster används.

Längst ner på sidan sticker rubrikerna för det huvudsakliga receptet upp så att läsaren ska se att det går att bläddra ner för att läsa vidare. Detta så att läsaren inte ska överväldigas av all text på samma gång. Både sektionerna för ingredienser och instruktioner följer ett F-mönster.

Figur 27. Hierarkin för ingredienser och instruktioner följer ett F-mönster.

Ingredienser och instruktioner i indelade i skilda rutor och har båda en rubrik överst som beskriver rutans innehåll. De mindre rubrikerna inom ingrediensrutan beskriver för vilken

komponent av maträtten som ingredienserna tillhör och rubrikerna inom instruktionsrutan beskriver för vilket moment de följande instruktionerna tillhör.

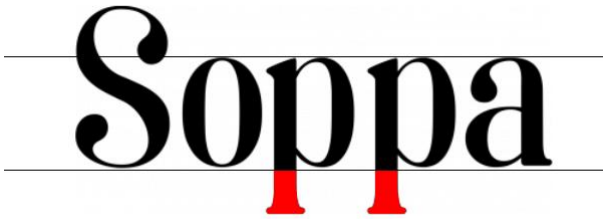
3.4.2 Avstånd

För att layouten ska vara enhetlig så separeras olika element med jämna avstånd. Detta uppnås med CSS-funktionerna grid och flexbox. Genom att lägga samtliga element som tillhör main-taggen inom flexbox så kan man ange avståndet mellan elementen. Avståndet blir då det samma mellan samtliga grupperade element. En grupp med element inom en flexbox eller ett grid kan även skilt för sig grupperas inom en egen flexbox för att inbördes få jämna avstånd.

Figur 28. På första sidan används både flexbox och grid för att få jämna avstånd.

I figuren ovan används flexbox för att separera elementen med ett avstånd som motsvarar rubrikens x-höjd, det vill säga hur hög den gemena bokstaven x är som har varken överhäng eller underhäng. Avståndet mellan rubriken och underrubriken specificeras skilt för att avståndet skall mätas från underrubrikens övre kant till rubrikens x-höjds nedre kant, alltså utan beaktande för underhäng. Eftersom få bokstäver har

underhäng, i detta fall enbart två g:n, så ser avståndet för långt ut om det räknas från underhänget.



Figur 29. Exempel med x-höjd inom horisontala linjer och underhäng markerat med rött.

CSS-funktionen grid används för att lägga de dekorativa bilderna och knappen i ett rutnät där samtliga rutor är lika stora. Detta eftersom elementens storlek varierar. Elementen centreras horisontalt inom sina rutor och knappen centreras även vertikalt. Genom denna teknik så är tyngdpunkten för elementen ungefär på jämnt avstånd. Knappens tyngdpunkt är något högre upp än bildernas eftersom dess funktion är viktigare och ska upptäckas först.

Sidan för ett recept har jämna avstånd mellan alla element i rutan för tilläggsinformation, förutom förordet som är vertikalt centrerat inom sin ruta för fylla ut tomrum och skilja den från den icke-flytande texten ovan så att läsaren förstår att de är oberoende av varandra.

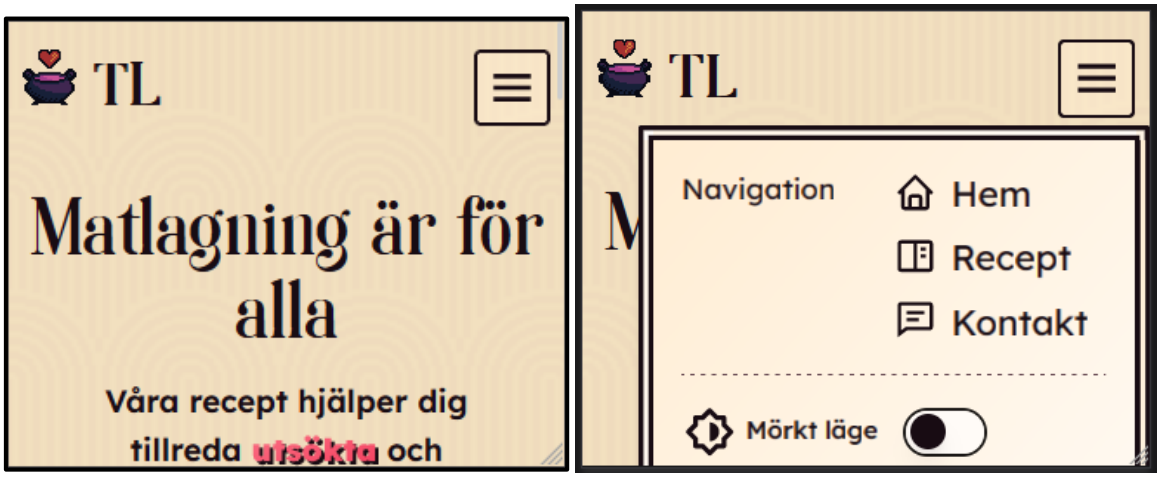
Figur 30. Jämna avstånd mellan informationsgrupperna i receptet.

3.4.3 Följsam layout

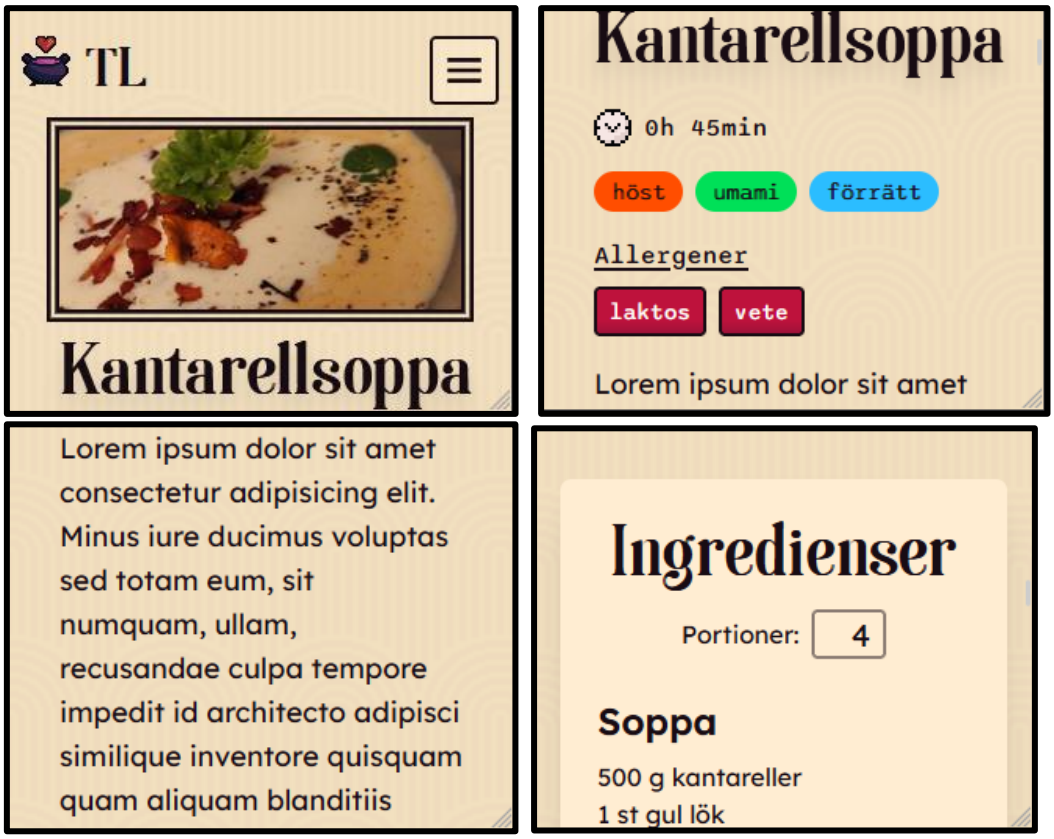
En följsam layout innebär att sidans layout automatiskt anpassas till skärmens storlek. Elementen bör varken vara för stora för att rymmas med eller för små för att se. Samtliga element bör vara med inom ramarna för sidan utan att oavsiktligt täckas av något annat element. (Accessibility Guidelines Working Group, 2023c).

Fasta värden används endast som undantag för att avgöra storleken för ett element. Fasta värden används endast för pixelbilderna på sidan så att dimensionen för en pixel ska förbli kvadratisk. Annars ser bilden inte korrekt ut. Storleken för dessa justeras då stegvis genom att halvera eller fördubbla höjd och bredd.

Enligt WCAG ska man inte behöva bläddra i mer än en riktning förutom i vissa undantagsfall. Minsta skärmdimensionerna som detta kriterium gäller för är 320 pixlars bredd och 256 pixlars höjd. För att programmera mot dessa kriterier så användes webbläsarens utvecklarverktyg för följsamt läge med dimensionerna inställda och det kontrollerades att sidan fungerade även för dessa dimensioner.



Figur 31. Första sidan och menyn med skärmdimensionerna 320x256.



Figur 32. Receptet med skärmdimensionerna 320x256.

För att åstadkomma en följsam layout används Tailwinds brytpunkter som definierar vid vilka skärmstorlekar som specifika CSS-regler ska börja gälla. För rubriken för ett recept anges textens storlek med följande formel i Tailwind:

Kodexempel 5. Formel för dynamisk kalkylering av rubrikens storlek.

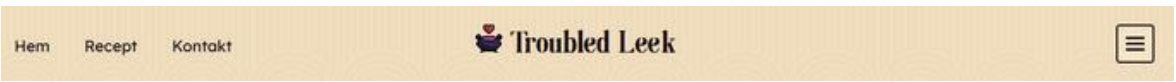
```
<h1
  class="font-soria text-foreground self-start
  [font-size:_clamp(3rem,_0.5rem_+8vw,_5rem)]">
  {{ recipe.name }}
</h1>
```

Formeln använder Tailwinds funktion för att ange arbiträra värden inom hakparentes. Där används CSS-funktionen clamp() som tar in parametrar för minsta möjliga storlek, önskad storlek däremellan, och största möjliga storlek. Minsta storleken är 3 rem, största är 5 rem, och allt däremellan räknas ut genom att ta en halv px och addera 8 % av skärmens bredd. Formeln användes eftersom rubriken ibland stack utanför skärmen för vissa skärmstorlekar om man endast ändrade den med brytpunkter. Värdena har tagits fram genom att testa för hand.

Navigationsmenyn ändras också enligt skärmbredd. När skärmen är som minst så är den indelad i två kolumner: en för logon och en för menyknappen. När skärmbredden kommer över brytpunkten 1024 px som anges med prefixet ’lg’ så blir kommer en till kolumn med som rymmer länkarna.

Kodexempel 6. Följsam layout för navigationsmenyn med Tailwind.

```
<header class="grid grid-cols-2 lg:grid-cols-3 sm:container
mx-auto mt-14 mb-4 sm:mb-16 px-2">
    <nav>
        <ul class="flex flex-row items-center justify-start">
```



Figur 33. Navigationsmenyn vid full skärmstorlek.

Figur 34. Navigationsmenyn vid minst skärmstorlek.

3.4.4 Kumulativt layoutskift

Kumulativt layoutskift, CLS, är ett oönskat beteende som gör att sidans layout ändras vartefter att element laddas in. Fenomenet uppstår när dimensionerna för text eller bild inte är förutbestämt utan uppfattas av webbläsaren först när elementet i fråga laddats in. Effekten av layoutskift blir mer märkbar desto fler element som bidrar till problemet och märks av längre desto långsammare internetuppkoppling man har. Beteendet kan störa användare och påverkar sidans sökmotoroptimering, SEO, negativt (Understanding Core Web Vitals and Google search results, 2024).

För att förbättra CLS så används modulen NuxtImg som fyller ut utrymmet för en bild före den laddats genom att man specificerar bildens bredd och höjd på förhand. Med NuxtImg får man även följsamma storlekar på bilderna och kan ställa brytpunkter för olika skärmstorlekar med samma syntax som TailwindCSS.

Eftersom de typsnitt som används inte kommer förinstallerade på datorer så kommer dessa att behöva laddas för användare när de först kommer till sidan. Under tiden används förinstallerade typsnitt, men eftersom storlek och avstånd varierar mellan typsnitt så kommer layouten att påverkas när det nya typsnittet med andra storlekar och avstånd laddas in. För att undvika detta beteende så kan man ändra proportionerna för reservtypsnitten så att de motsvarar det önskade typsnittet. Fontaine är en JavaScript-modul som genererar CSS för reservtypsnitt så att deras storlek och avstånd motsvarar de typsnitt som ännu inte laddats in. Fontaine behöver inte konfigureras utan bara installeras för att generera den CSS som behövs.

3.4.5 Knappar

Knappar, länkar, inputs, och allt annat som ska gå att trycka på i mobilen bör ha en minimistorlek på 44 pixlars bredd och höjd enligt WCAG. Det underlättar för användare med pekskärmar och gör det allmänt lättare att träffa knapparna. (Accessibility Guidelines Working Group, 2023e).

Eftersom elementen är som minst när skärmen är som minst så kan man utgå från de minsta skärmdimensionerna och gå igenom alla knappar där och sedan sätta in Tailwind-klasser för att specificera en minimibredd eller -höjd på 44 pixlar för de element som är under kraven.

Kodexempel 7. Minimibredd för en länk deklarerad med Tailwind.

```
<NuxtLink to="/"
    class="min-w-[44px] min-h-[44px]"
    aria-label="Home">
```

3.5 Navigering

Hela sidan bör vara navigerbar med endast tangentbord för att underlätta för användare med rörelsenedsättning. Man bör även kunna navigera sidan utan hinder med hjälp av skärmläsare. (Accessibility Guidelines Working Group, 2023b). Båda dessa krav kan uppnås med hjälp av korrekt semantisk HTML.

3.5.1 Semantisk HTML

HTML är uppbyggt av taggar som var och en har betydelse. Taggar har antingen en starttagg <a> och en sluttagg eller så är de självslutande <img />. Självslutande taggar används när inga andra element kan finnas inom elementet. En detalj som bör uppmärksammas för att undvika oväntat beteende är att snedstrecket i sig inte sluter en tagg, utan det är taggen själv som antingen är självslutande eller inte. Snedstrecket är endast där för läsaren ska se att taggen är självslutande, och kan till och med vara vilseledande ifall den används för taggar som inte sluter sig själva.

Man kan specificera olika egenskaper inom starttaggen genom att skriva namnet på egenskapen följt av ett likhetstecken och värdet inom citattecken. När man skriver semantiskt korrekt HTML så använder man rätt tagg för rätt ändamål och tar dess befintliga egenskaper i beaktande. Till exempel så bör knappar använda sig av taggen <button>. Om man använder fel tagg för en knapp så kräver det mer arbete, processorkraft och utrymme för att få motsvarande funktion.

Genom att skriva korrekt semantisk HTML så underlättas läsandet av koden, SEO förbättras eftersom Google gör mätningar som berör detta och baserar placeringen i sökresultat på mätningarnas resultat. Man kan se en del av dessa mätningar i verktyget Lighthouse som tas upp i ett senare skede (se kapitel 3.7.2). Semantisk HTML är även bättre ur ett tillgänglighetsperspektiv, framför allt för skärmläsare.

Ett HTML-dokument börjar alltid med en html-tagg som sluter sig över resten av dokumentet. På följande nivå kan enbart finnas en head-tagg och en body-tagg. I head-taggen lägger man metadata som inte visas på själva sidan men som innehåller viktig information, såsom beskrivning för sidan som visas på sökmotorer, länk till adressikonen, eller favicon, och länk till CSS-filer för sidan. I body-taggen kommer allt som kommer att synas när man går in på sidan.

Kodexempel 8. head- och body-taggar inom html-tagg.

```
<html>
    <head>

    </head>
    <body>

    </body>
</html>
```

I Nuxt ersätts body-taggen av template-taggen och man använder funktionen useHead() för att specificera vad som ska komma inom head-taggen. När det kompileras så följer sidan i slutändan ändå en standard HTML-struktur.

Kodexempel 9. Exempel på en semantiskt korrekt HTML-struktur.

```
<html>
  <body>
    <header>
      <nav>
        <ul>
          <li>
            <a href="#">link 1</a>
          </li>
          <li>
            <a href="#">link 2</a>
          </li>
          <li>
            <a href="#">link 3</a>
          </li>
        </ul>
      </nav>
    </header>
    <main>
      <article>
        <header>
          <h1>Header</h1>
          <h2>published 2024</h2>
        </header>
        <p>paragraph containing longer text</p>
      </article>
      <section>
        <p>section containing another paragraph unrelated to
        the first one</p>
      </section>
    </main>
    <footer>footer</footer>
  </body>
</html>
```

Exemplet ovan visar den struktur som sidorna i projektet följer. Dokumentet läses uppifrån och ner både vid tangentbordsnavigering och av skärmläsare. Överst är en header-tagg som omfattar en navigationsmeny inom nav-taggen. En logo kan visas inom header-taggen skilt från nav-taggen men i det här fallet fungerar logon även som länk till första sidan och listas därför under navigationsmenyn. Eftersom navigationsmenyn listar olika länkar används en oordnad lista <ul> med listelement som innehåller länkar <a>. Under header-taggen kommer main-taggen och det är här det huvudsakliga innehållet för sidan kommer. Inom main-taggen finns en artikel som i sin tur har en ny header-tagg för att beskriva artikelns innehåll. Taggarna för rubrikerna <h1> och <h2> följer en numerisk ordning utan att hoppa över någon siffra. Sedan kommer en ny sektion för ny text som är avskild från artikeln, och sist utanför main-taggen kommer en tagg <footer> för fotnoter.

3.5.2 Navigering med tangentbord

Man kan navigera sidan genom att trycka på Tab, varpå HTML-dokumentet gås igenom uppifrån och ner och varje element som går att klicka på markeras. Man kan då klicka med Enter, kryssa för med mellanslag, eller avbryta med Esc.

Högst upp på varje sida finns en *hopplänk* som är ett osynligt element som länkar till huvudinnehållet, vilket är definierat inom main-taggen. Användaren kan på så vis undvika att gå igenom navigationen varje gång sidan byts och presenteras i stället med möjligheten att skippa direkt till sidans huvudsakliga innehåll efter första Tab-tryckningen.

Kodexempel 10. Hopplänk.

```
<div class="-mt-10 mb-10 focus:mt-0">
    <a href="#main">Hoppa över navigation</a>
</div>
```

Runt varje länk och knapp kommer en streckad linje i svart som förtydligar att just det elementet är i fokus. Linjen som tillkommer har samma krav på färgkontrast, alltså minst 7:1 jämfört med bakgrunden. Om en knapp är en annan färg än bakgrunden så mäts kontrasten både mot bakgrunden och knappens färg.

Figur 35. Fokuserat element vid tangentbordsnavigation.

Användaren kan kryssa av de moment som redan är avklarade för att hålla koll på var i receptet man befinner sig. Texten streckas då över och blir en mattare färg. Det går att klicka igen för att återställa texten. Med tangentbord kryssar man av genom att trycka på mellanslag.

Figur 36. Man kan kryssa för avklarade moment.

3.6 Språk

Då språk går att välja ska metoden finnas lättillgänglig överst på sidan. Flaggor bör undvikas i den här kontexten eftersom de representerar geografiska platser där flera språk kan förekomma samtidigt. Flaggor är dessutom politiskt laddade och felanvändning kan vara kontroversiellt och stötande. Till exempel så är det missvisande att använda svenska flaggan för finlandssvenska språket, eftersom man fortfarande befinner sig i Finland. I stället för flaggor kan man använda en ikon för glob eller annan lämplig ikon, och språkens namn bör visas på eget språk för att kännas igen av de som talar det språket. (Söderman & Gestrin, 2024).

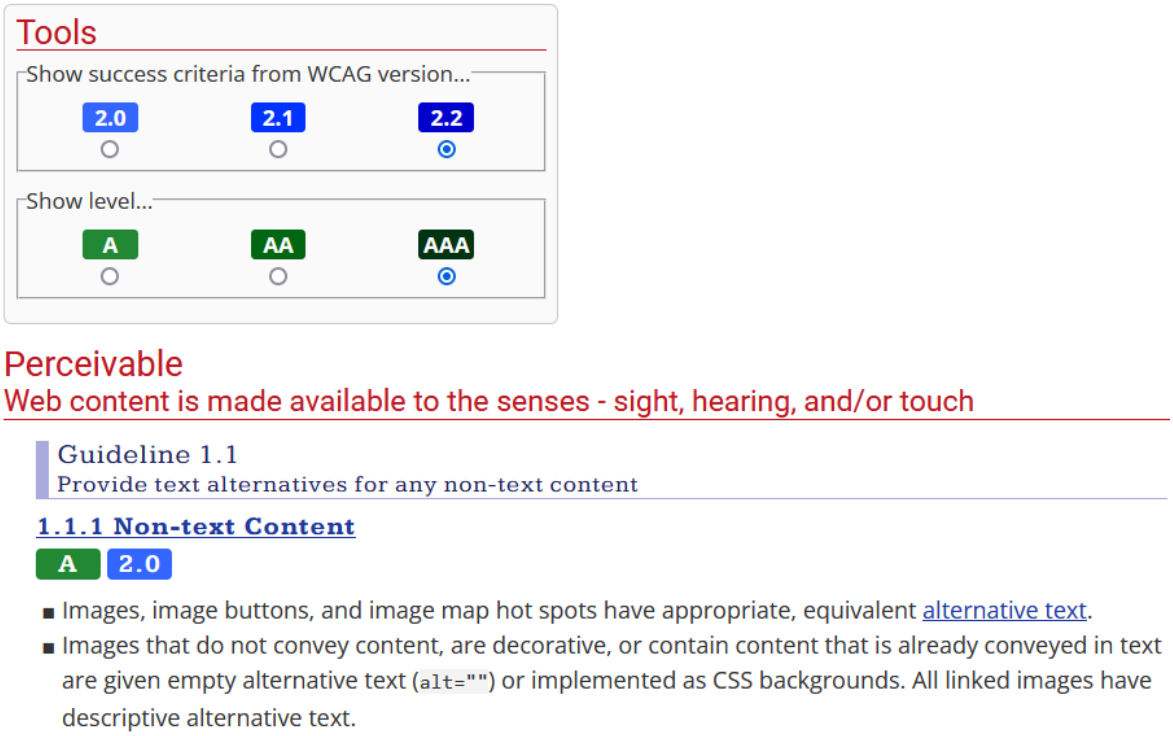
Figur 37. Språkinställning finns i menyn högst upp på sidan, och markeras av en glob-ikon och beskrivande text.

3.7 Kodanalys

För att säkerställa att man uppfyller alla krav är det viktigt att analysera koden och den befintliga hemsidan. Det kan vara svårt att själv memorera alla punkter i WCAG så det underlättar om man använder sig av andra verktyg för att kontrollera sin produkt. Verktygen som användes mest förutom extensiv forskning på nätet är en checklista för WCAGs krav som finns tillgänglig via WebAIM, utvecklad av Utah State University, och Google Lighthouse som finns i Chromium-baserade webbläsare och analyserar den kod som körs och gör mätningar på den.

3.7.1 WebAIMs checklista för WCAG

WebAIM, kort för Web Accessibility In Mind, är en webbsida som utvecklas av Utah State University Institute for Disability Research, Policy, and Practice. Förutom mer koncisa och konkreta förklaringar för vad kriterierna i WCAG innebär så har de en checklista som man kan gå igenom för sin webbsida. Man kan filtrera enligt de tre olika A-nivåerna och enligt de olika versionerna av WCAG 2. Den har även länkar till mer ingående förklaringar för varje kriterium. (Institute for Disability Research, Policy and Practice, 2023).



Figur 38. WebAIM's WCAG 2 Checklist (Institute for Disability Research, Policy and Practice, 2023).

3.7.2 Google Lighthouse

Lighthouse är ett verktyg utvecklat av Google som är inbyggt i Chromium-baserade webbläsare. Verktyget analyserar en webbsida för vanliga brister och ger en utvärdering för olika kategorier med ett betyg from 0 till 100 och förslag på förbättringar.

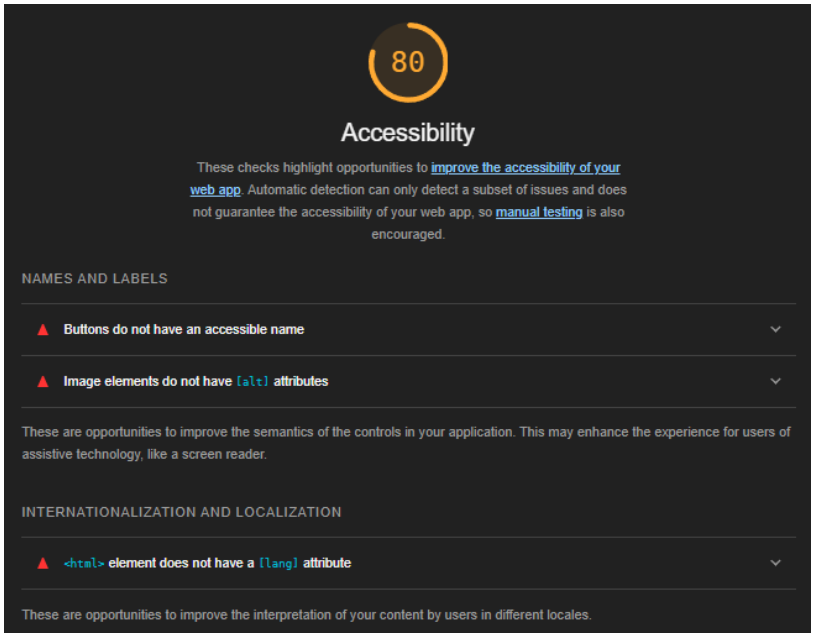
När sidan först mättes så fick tillgängligheten tre anmärkningar: sidans språk hade inte specificerats i HTML, alternativ text fattades från vissa bilder, och menyknappen hade inget namn för skärmläsare.

Kodexempel 11. Språket för HTML-dokumentet ställs in med funktionen useHead().

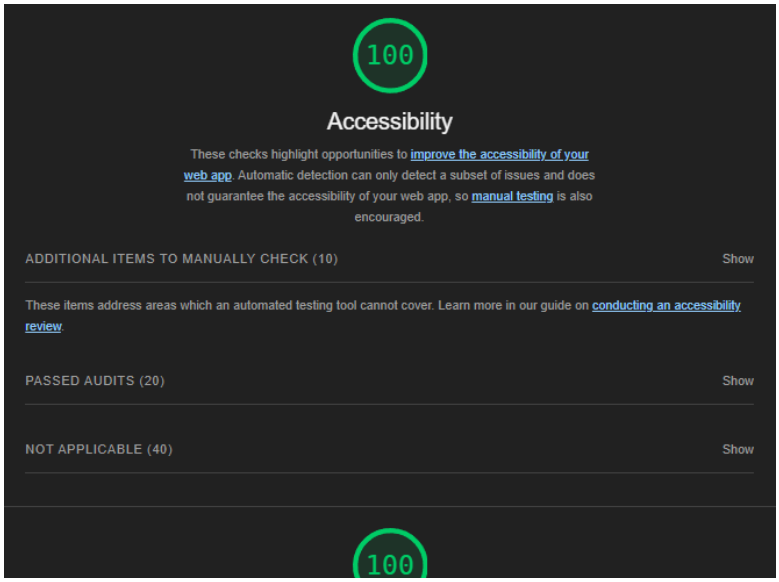
```
<script>
useHead({
  htmlAttrs: {
    lang: 'sv'
  },
});
</script>
```

Funktionen useHead() används för att lägga till metadata i head-taggen. Här användes den för att ändra dokumentets attribut för språk svenska. Alternativ text, eller alt-text, har

redan diskuterats mer ingående (se kapitel 3.2.2), men problemet löstes med att ange en tom alt-text för dekorativa element och med attributet aria-labelledby för bilder på recept så att deras alt-text bestäms av receptets namn för att undvika dupliceringar. Sist och slutligen gavs ett namn åt menyknappen med hjälp av attributet aria-label. När analysen nu körs igen så hittas inga problem.



Figur 39. Rapport för sidans tillgänglighet genererad av Google Lighthouse.



Figur 40. Lighthouse-rapporten efter problemen åtgärdats.

4 Resultat

Resultatet av arbetet är en webbplats som fyller kraven för AA-standarden framtagen av WCAG. Alla sidor har en följsam design som fungerar på alla skärmar från med dimensionerna 320x256 och högre. Layouten har en visuell hierarki som leder läsarens blick genom innehållet i lämplig ordning. Gränssnittet går att navigera med enbart tangentbord och repeterande element såsom navigationsmenyn går att hoppa över. Typsnittet för brödtexten är tydligare och går snabbare att läsa än övriga typsnitt som är förinstallerade, och det finns studier som understryker detta. Det går att välja färger för taggar oberoende kulörton utan att kontrasten mellan taggens text och bakgrundsfärgen riskerar bli för låg. Användare av sidan får en överblick av hur länge ett recept kommer ta att genomföra, vilka allergier som recept inte lämpar sig för, stegvis instruktioner för hur receptet ska utföras, samt möjligheten att strecka över de moment man redan utfört för att minnas var i receptet man befinner sig under matlagningen. Webbplatsen lyckas fylla kraven och ändå bibehålla ett unikt utseende med en distinkt färgpalett och karaktäristisk pixelkonst.

5 Diskussion

Tillgänglighet har varit ett intressant ämne att forska om och fastän kraven kan verka restriktiva så har jag börjat se möjligheter hur olika sidor kan förbättras, och fått en ny uppskattning för välgjorda webbsidor. Jag känner mig bekväm med verktygen som jag använt i projektet och kommer antagligen använda de flesta av dem även framöver om jag får välja. Särskilt givande har det varit att få konkretisera vad som gör en bra webbsida, med allt från layouter, text och färg till interaktivitet och tillgänglighet. Det har även lett till en djupare förståelse för funktionsvariationer som dyslexi och synnedsättningar, men också hur även dessa målgrupper kan ta del av webben.

5.1 Utvecklingsmöjligheter

Trots att sidan redan följer föreskrifterna i WCAG finns det ännu utvecklingsmöjligheter. Som nämnts i kapitel 3.3.3 så skulle en möjlighet vara att låta användare själva bestämma vilken variant av Lexend-typsnittet de vill använda. Man kan redan använda utomstående

teknologi för att justera avstånd mellan ord och tecken, men genom att ändra typsnittsvariant så skulle även själva tecknen anpassas för lättare läsning med större avstånd.

En intressant ny teknologi som utvecklas för Vue är signaler. Signaler gör att när något av elementen på sidan uppdateras med dynamiska data så behöver inte hela sidan uppdateras utan enbart det element som berörs, vilket ger ökad prestanda. Konceptet är taget från ramverket Solid.

Även om knappar för olika språk fanns med nu för att demonstrera så skulle sidan givetvis på riktigt kunna översättas till engelska i framtiden. Databasen är redan uppbyggd på ett sätt som ska underlätta översättning i framtiden. Ett problem med översättning är att det antingen måste göras för hand eller automatiseras, både med sina för- och nackdelar. Att översätta för hand tar längre tid och innehållet kan skilja något språken emellan. Å andra sidan så kan automatiserad översättning skötas av till exempel AI men måste då kontrolleras så att det verkligen fungerar som det ska.

Kommentarsfält skulle göra sidan mer interaktiv men då behövs antingen ett inloggningssystem för alla som kommenterar eller så måste skydd implementeras mot robotattacker. Det finns tjänster som Disqus som tillhandahåller kommentarsfält som en tjänst, men de begär en månadsavgift om man vill vara utan reklam och man måste förlita sig på att en extern tjänst hålls i gång.

Källförteckning

Accessibility Guidelines Working Group. (20.6.2023a). *Contrast (Enhanced) (Level AAA).* Hämtat från WCAG 2.1 Understanding Docs: https://www.w3.org/WAI/WCAG21/Understanding/contrast-enhanced.html

Accessibility Guidelines Working Group. (20.6.2023b). *Keyboard Accessible.* Hämtat från WCAG 2.1 Understanding Docs: https://www.w3.org/WAI/WCAG21/Understanding/keyboard-accessible

Accessibility Guidelines Working Group. (20.6.2023c). *Reflow (Level AA).* Hämtat från WCAG 2.1 Understanding Docs: https://www.w3.org/WAI/WCAG21/Understanding/reflow.html

Accessibility Guidelines Working Group. (20.6.2023d). *Resize Text (Level AA)*. Hämtat från WCAG 2.1 Understanding Docs: https://www.w3.org/WAI/WCAG21/Understanding/resize-text.html

Accessibility Guidelines Working Group. (20.6.2023e). *Target Size (Level AAA).* Hämtat från WCAG 2.1 Understanding Docs: https://www.w3.org/WAI/WCAG21/Understanding/target-size.html

Accessible design. (2021). Hämtat från Material Design 3: https://m3.material.io/foundations/accessible-design/design-to-implementation

Deveria, A., & Schoors, L. (April 2024). *AVIF image format*. Hämtat från caniuse.com: https://caniuse.com/?search=avif

Europaparlamentet. (10.9.2019). *Produkters och tjänsters tillgänglighet.* Hämtat från https://eur-lex.europa.eu/legal-content/SV/TXT/HTML/?uri=LEGISSUM:4403933

Eurostat. (26.4.2024). *Level of disability (activity limitation) by sex, age and income quintile*. Hämtat från https://ec.europa.eu/eurostat/databrowser/view/hlth_silc_12/default/table?lang=en

Institute for Disability Research, Policy and Practice. (9.1.2021). *Contrast and Color Accessibility.* Hämtat från WebAIM: https://webaim.org/articles/contrast/#ratio

Institute for Disability Research, Policy and Practice. (12.10.2023). *WebAIM's WCAG 2 Checklist.* Hämtat från WebAIM: https://webaim.org/standards/wcag/checklist

Joseph, H., & Powell, D. (2022). Does a specialist typeface affect how fluently children with and without dyslexia process letters, words, and passages? *Dyslexia (Chichester, England)*, 28. Hämtat från https://doi.org/10.1002/dys.1727

Luz Rello, R. B.-Y. (2013). *Good Font for Dyslexia.* Barcelona, Spanien. Hämtat från https://dyslexiahelp.umich.edu/sites/default/files/good_fonts_for_dyslexia_study.pdf

Marshall, S. (2020). *The ultimate guide to visual hierarchy.* Hämtat från Canva: https://www.canva.com/learn/visual-hierarchy/

Microsoft. (2024). *TypeScript*. Hämtat från https://www.typescriptlang.org/

Mozilla. (2024). *grid*. Hämtat från MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS/grid

Nuxt. (2024). *Nuxt Docs*. Hämtat från https://nuxt.com/docs/getting-started/introduction

OddBird. (2024). *OddContrast*. Hämtat från OddContrast: https://www.oddcontrast.com/

PNPM. (2024). *Motivation*. Hämtat från https://pnpm.io/motivation

Shaver-Troup, B. (2018). *The Effect of Typographical Factors on Reading Performance of Second Grade Students.* Hämtat från https://www.proquest.com/docview/2112748829

StatCounter. (Mars 2024). *Browser Market Share Worldwide*. Hämtat från https://gs.statcounter.com/browser-market-share/

Söderman, M., & Gestrin, C. (2024). *Sidan hittades ej - den digitala tvåspråkigheten i Finland.* Helsingfors, Finland: Tankesmedjan Agenda. Hämtat från https://agenda.fi/publikation/sidan-hittades-ej-den-digitala-tvasprakigheten-i-finland/

Toolness. (2024). *Accessible color palette builder*. Hämtat från https://toolness.github.io/accessible-color-matrix/

Understanding Core Web Vitals and Google search results. (18.3.2024). Hämtat från Google Search Central: https://developers.google.com/search/docs/appearance/core-web-vitals

Web Accessibility Initiative (WAI). (13.7.2020). *Web Content Accessibility Guidelines (WCAG) 2 Level AAA Conformance*. Retrieved from https://www.w3.org/WAI/WCAG2AAA-Conformance