Oliver Östman

Verkkosivuston responsiivisuuden ja saavutettavuuden parantaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

13.11.2023

Tiivistelmä

Tekijä:	Oliver Östman
Otsikko:	Verkkosivuston responsiivisuuden ja saavutettavuuden parantaminen
Sivumäärä:	34 sivua
Aika:	13.11.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaajat:	Projektipäällikkö Päivi Laine Lehtori Ulla Sederlöf

---

Insinöörityön tarkoituksena oli parantaa Keikkakaveri-verkkosivuston responsiivisuutta ja lisätä sivuston saavutettavuutta. Tavoitteena oli, että verkkosivusto saavuttaisi saavutettavuuden kannalta kansainvälisen WCAG-saavutettavuusohjeistuksen (Web Content Accessibility Guidelines) A- tai AA-tason ja toimisi responsiivisesti mobiililaitteilla ja tableteilla.

Työssä sivusto käytiin läpi ja katsottiin, mistä löytyy responsiivisuusongelmia mobiililaitteilla ja tableteilla. Sivuilla, joilta löytyi responsiivisuuden kanssa ongelmia, responsiivisuutta parannettiin, kunnes ongelmia ei enää ollut. Työssä käytettiin responsiivisuuden kanssa keskeytyskohtia, suhteellisia yksiköitä ja muita yleisiä responsiivisuustekniikoita. Lopuksi tutustuttiin vielä siihen, miten responsiivisuuden tarkistusta voisi automatisoida tulevaisuudessa.

Työssä käytiin verkkosivut läpi myös käyttäen kolmea eri saavutettavuuden tarkistustyökalua, jotka tarkistivat, mitä ongelmia sivuilla oli saavutettavuuden kannalta. Saavutettavuuden virheet ja ongelmat, jotka tarkistustyökalut löysivät, korjattiin. Joihinkin ongelmiin ei kuitenkaan ehditty työn aikana koskea, joten ne kirjoitettiin muistiin ja ilmoitettiin hankkeelle.

Työn tuloksena sivustosta ei enää löytynyt responsiivisuusongelmia ja suurin osa saavutettavuusvirheistä saatiin ratkaistua. Työn aikana selvisi, että responsiivisuus ja saavutettavuus kannattaa ottaa jo verkkosivun suunnitteluvaiheessa huomioon.

Avainsanat: responsiivisuus, saavutettavuus, React, WCAG

Abstract

Author:	Oliver Östman
Title:	Improving website responsiveness and accessibility
Number of Pages:	34 pages
Date:	13 November 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and communication technology
Professional Major:	Media engineering
Supervisors:	Päivi Laine, Project Manager
	Ulla Sederlöf, Senior Lecturer

The purpose of the final year project was to improve the responsiveness and increase the accessibility of the Keikkakaveri website. The goal was to get the website working responsively on mobile devices and tablets and to achieve the WCAG (Web Content Accessibility Guidelines) A or AA level in terms of accessibility.

The website was reviewed for problems on responsiveness on mobile devices and tablets. When problems were found with responsiveness they were fixed. Breakpoints, relative units, and other responsiveness techniques were used when improving the responsiveness. The website was also reviewed using three different accessibility checking programs. Accessibility errors and issues found by the tools were fixed. However, there was not enough time to investigate some issues, so they were documented and reported. Also, how responsiveness checking could be automated in the future was studied in the project.

Upon completing the project there were no responsiveness problems with mobile devices and tablets. Most of the accessibility errors were solved as well. During the project, it became clear that responsiveness and accessibility should already be considered in the design phases of websites.

Keywords: responsiveness, accessibility, React, WCAG

Sisällys

Lyhenteet

WCAG:	Web Content Accessibility Guidelines.
TTT:	Työterveys, -hyvinvointi ja -turvallisuus.
MUI:	Material UI.
CSS:	Cascading style sheets.
PX:	Pixel.
UI:	User interface.
UX:	User experience.
HTML:	Hypertext markup language.
DOM:	Document object model.
EU:	Euroopan unioni.
ARIA:	Accessible rich internet applications.

1 Johdanto

Insinöörityö tehtiin Kestävä Keikkatyö -hankkeelle, ja sen tarkoituksena oli parantaa responsiivisuutta ja lisätä saavutettavuutta Keikkakaveri-sivustolle. Tavoitteena oli saada sivusto toimimaan mobiililaitteilla ja tableteilla ja saavutettavuus ainakin WCAG-ohjeistuksen A-tasolle.

Responsiivisuus on nykyään aika lailla perusedellytys verkkosivustoilla. Tästä syystä yleensä sivustoilla ei nykyään ilmene responsiivisuuden kanssa juurikaan ongelmia. Monet komponenttikirjastot, joita käytetään sivujen luomisessa, vielä lisäävät responsiivisuutta automaattisesti, kun sivuja ollaan tekemässä. Saavutettavuus taas on uudempi ja ehkä hieman tuntemattomampi asia verkossa, mutta se on nopeasti lisääntymässä. Vuonna 1999 julkaistiin verkkosisällön saavutettavuudesta ensimmäinen ohjeistus, jonka nimeksi tuli WCAG (Web Content Accessibility Guidelines), ja vuonna 2008 julkaistiin WCAG 2.0. Nykyään WCAG 2.0:aa käytetään monien maiden lainsäädännön saavutettavuusvaatimusten perustana. [Tietoa WCAG ohjeistuksesta: 23.] Saavutettavuus on hyvin tärkeää verkossa esimerkiksi näkövammaisille, ja sitä pitäisi olla varsinkin välttämättömillä sivustoilla, kuten valtioneuvoston tai pankin sivuilla.

Kestävä Keikkatyö -hankkeen tarkoitus oli työturvallisuuden, -terveyden ja -hyvinvoinnin parantaminen, ja Keikkakaveri-verkkosivusto oli vain yksi osa tämän tavoitteen saavuttamiseen. Sivustolla ei ollut otettu saavutettavuutta vielä huomioon, ja responsiivisuuden kanssa alettiin huomata ongelmia sivustolla. Keikkakaveri-sivuston tarkoitus oli olla web-sovellus Kestävä Keikkatyö -hankkeelle, ja sivusto tarjoaa alustan työtehtävien hallintaan, viestintään ja työlupien sekä dokumenttien säilyttämiseen.

Insinöörityössä tutkittiin myös, mitä saavutettavuus on ja miten sitä pystytään lisäämään sivustolle, muuten kuin vain saavutettavuuden virheitä korjaillen. Saavutettavuusvirheiden etsimiseen käytettiin kolmea eri saavutettavuuden testaustyökalua. Responsiivisuuden ongelmia sivustolla tutkittiin enemmän käsin käyttäen Chrome-selaimen sisäänrakennettuja DevTools-työkaluja.

2 Kestävä Keikkatyö -hanke

2.1 Hankkeen tausta

Kestävä Keikkatyö on Euroopan sosiaalirahaston ja sosiaali- ja terveysministeriön rahoittama hanke. Hanketta toteutetaan yhteistyössä Metropolia Ammattikorkeakoulun, Tampereen yliopiston, Oulun yliopiston ja Satakunnan Ammattikorkeakoulun kanssa.

Kestävä Keikkatyö -hankkeen päämääränä on työturvallisuuden, -terveyden ja -hyvinvoinnin (TTT) parantaminen. Hankkeessa lähdetään siitä, että näitä parantamalla tuottavuus ja kilpailukyky vahvistuisivat keikkatyötä hyödyntävissä pienissä ja keskisuurissa yrityksissä sekä keikkatyötä välittävissä yrityksissä. Hankkeen tavoitteena on myös luoda kulttuuria, jossa vastuu olisi kaikilla osapuolilla ja työturvallisuudesta ja hyvinvoinnista pidettäisiin yhdessä huolta. Samalla olisi mahdollista vahvistaa johtamisosaamista ja selkeyttää keikkatyötä hyödyntävien ja välittävien yritysten, keikkatyöläisten ja työterveyshuollon yhteistyötä, sopimuskäytäntöjä ja rooleja. Lisäksi hankkeen päämääränä on työllisyyden edistäminen, työurien pidentäminen sekä työvoimatarpeen ja työntekijöiden nykyistä parempi kohtaaminen. [Kestävä Keikkatyö -hanke: 13.]

Kuvassa 1 näkyy Kestävä Keikkatyö -posteri, joka esittää monia eri hankkeeseen kuuluvia osia.

Kuva 1. Kestävä Keikkatyö -hankkeen posteri [Kestävä Keikkatyö Posteri].

Näiden periaatteiden pohjalta on tarkoitus luoda tuloksiin perustuva Kestävä Keikkatyö -malli. Mallin avulla yritykset voisivat tunnistaa helpommin TTT-tekijöihin ja niiden johtamiseen liittyviä kehittämistarpeita sekä tuottaa niihin ratkaisuja erityisesti keikkatyöntekijöiden näkökulmasta.

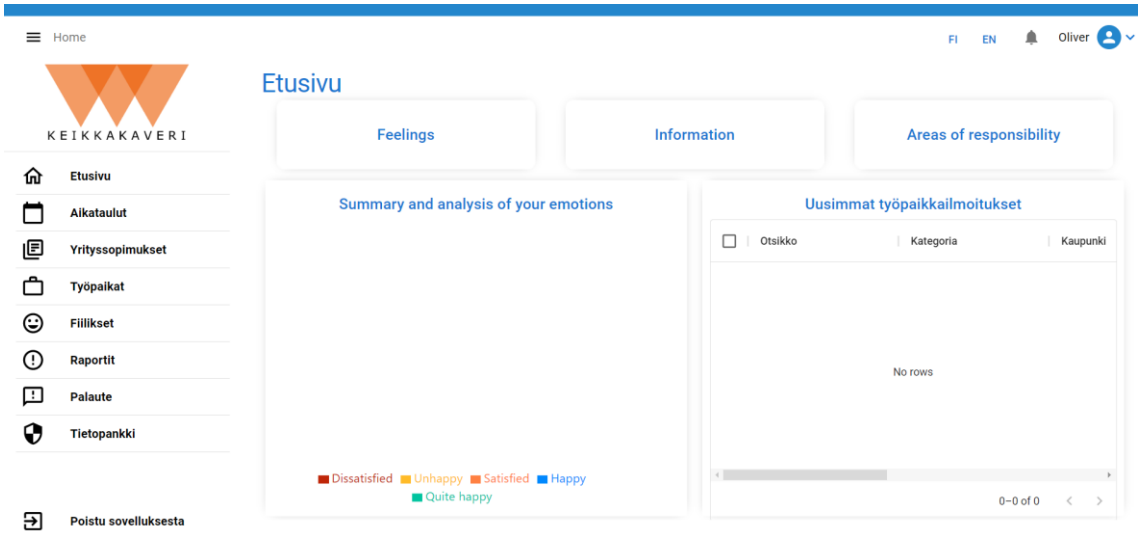
2.2 Keikkakaveri-sovellus

Keikkakaveri on verkkopohjainen sovellus vuokratyöntekijöille, -välittäjille ja yrityksille, jotka haluavat käyttää vuokratyövoimaa. Se tarjoaa alustan arkipäivän työtehtävien hallintaan, viestintään ja työlupien sekä dokumenttien säilyttämiseen sähköisesti. Keikkakaveri-sivustoa oli työstetty Metropolia Ammattikorkeakoulussa jo parilla innovaatiokurssilla ja oli suunnitelmissa, että vielä ainakin yksi innovaatiokurssi olisi tulossa tekemään sitä lisää.

Keikkakaveri-sivusto koostuu sen etusivusta, käyttäjäsivuista ja tietopankista. Etusivulla on lyhyt selitys, mikä Keikkakaveri on, kuten kuvassa 2 näkyy, ja myös lyhyesti työntekijöiden, yritysten ja välittäjien vastuualueista ja työn elinkaaresta. Etusivun tarkoitus on antaa jonkinlainen idea käyttäjälle, mikä Keikkakaveri on, ja antaa pääsy kirjautumaan. Kirjautumisen jälkeen pääsee käsiksi sivuston tärkeimpiin osiin: Käyttäjäsivuihin.

Kuva 2. Keikkakaveri sivuston pääsivu insinöörityöprojektin lopussa [Keikkakaveri-sivusto].

Tehdessään käyttäjätilin käyttäjillä on mahdollisuus valita kolmesta eri käyttäjäroolista: työntekijä, henkilöstöpalvelu tai käyttäjäyritys. Käyttäjä- ja henkilöstöpalveluyritys -käyttäjäroolit voivat vielä myös tarkemmin kohdentaa käyttäjärooliaan antamalla yritykselleen kategorian. Käyttäjien sivuilla on paljon eri toimintoja, kuten näkyy kuvassa 3, jossa on esillä työntekijä-käyttäjäroolin etusivu. Monet käyttäjien sivuista ovat aika itsestäänselviä. Jokaisella käyttäjäroolilla on melkein samanlainen etusivu kirjautuneena, mutta sivun sisältö vaihtuu riippuen valitusta käyttäjäroolista. Käyttäjäsivuilla on sivupalkki, jonka avulla pääsee kaikille käyttäjän sivuille ja toimintoihin. Kaikki käyttäjät pääsevät myös omaan profiiliinsa ja asetuksiinsa oikeasta yläkulmasta klikkaamalla omaa käyttäjänimeä.

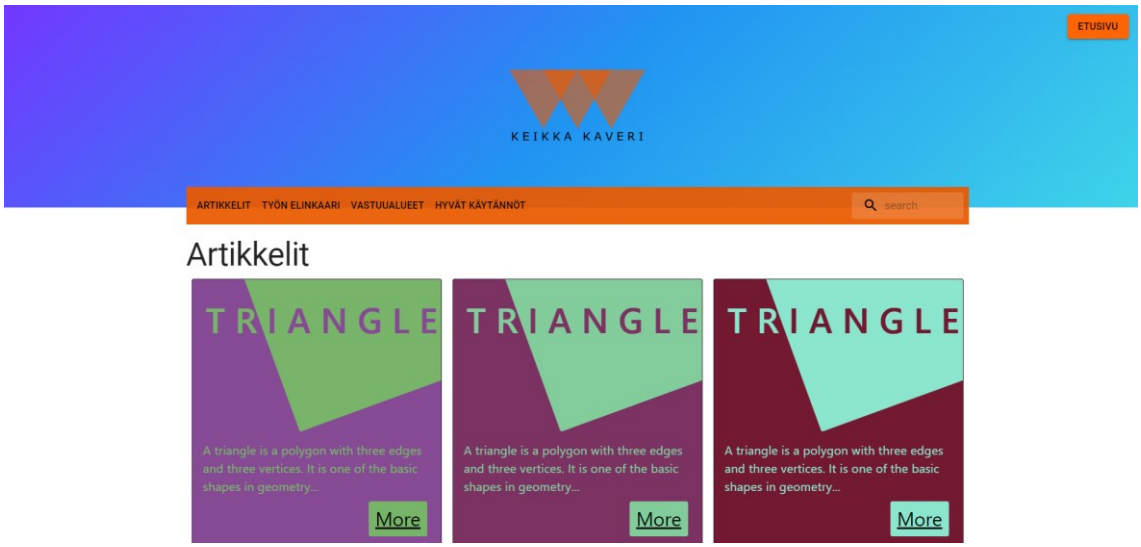


Kuva 3. Keikkakaveri-sivuston työntekijä-käyttäjäroolin etusivu [Keikkakaveri-sivusto].

Tietopankkiin oli tarkoitus tulla kokoelma ohjeita ja yleistietoa vuokratyöstä. Tietopankki ei kuitenkaan ollut valmis vielä, kun tämä insinööriprojekti alkoi, eikä se myöskään valmistunut insinöörityöprojektin aikana. Tietopankista löytyy

- tietoa vuokratyön yhteistyön vaiheista
- tietoa eri roolien vastuista vuokratyössä
- ohjeita, miten eri roolit voivat huolehtia vuokratyöntekijöiden turvallisuudesta, työterveydestä ja -hyvinvoinnista
- tietoa hyvistä käytännöistä vuokratyössä.

Tietopankkia ei insinöörityössä käyty juurikaan läpi, koska se ei ollut vielä valmis. Korjattavaa sivulla oli kuitenkin paljon, kuten sivun artikkelit neliöissä, joissa kontrasti oli huono eivätkä ne sopineet kunnolla mobiilinäytölle. Kuvassa 4 näkyy vielä projektin aikana keskeneräinen tietopankin Artikkelit-sivu.



Kuva 4. Keikkakaveri-sivuston keskeneräinen tietopankkisivu.

Keikkakaveri oli tehty käyttäen React-kirjastoa ja MUI-komponenttikirjastoa. Sivuston loppupää toimii insinöörityöprojektin aikana ilmaisella MongoDB Cloud -pilvipalvelimella. Valmis sivusto oli tarkoituksena siirtää Amazon-palvelimelle, jossa oli jo vanha versio sivustosta. Sivustossa ei vielä ollut otettu yhtään huomioon saavutettavuutta. Sivustolla alkoi myös esiintyä ongelmia mobiililaitteilla ja tableteilla responsiivisuuden kanssa. Insinöörityöprojektissa oli tarkoitus parantaa saavutettavuutta ja korjata responsiivisuutta sivustolla.

3 Responsiivisuus

Responsiivisella suunnittelulla tarkoitetaan verkkosivujen suunnittelua tavalla, jossa sivut mukautuvat aina käyttäjäystävälliseen muotoon käyttäjän päätelaitteen mukaisesti. Sivut näyttävät silloin hyvältä ja toimivat moitteettomasti tietokoneella, tableteilla ja eri mobiililaitteilla. Responsiivinen suunnittelu koostuu joustavista ruuduista ja sijoittelusta, kuvista ja älykkäästä CSS-mediakyselyjen

(media query) käytöstä [Friedman 2018: 9]. Kun käyttäjä vaihtaa esimerkiksi kannettavasta tietokoneestaan älypuhelimeensa, verkkosivuston pitäisi automaattisesti muuttaa resoluutio ja kuvakoko laitteelle sopivaksi.

3.1 Responsiivisuuden tekniikat

Responsiivisuus on nykyään erittäin tärkeää, kun suuri osa käyttäjistä tulee verkkosivuille mobiililaitteilla. Nykyään responsiivisuus on aika lailla uuden verkkosivun perusedellytys. Responsiivisuuden tarve kasvoi, kun älypuhelinten ja tablettien osuus internetissä kasvoi räjähdysmäisesti. Kun älypuhelimet tulivat markkinoille, jotkin verkkosivustot tekivät myös omat sivut älypuhelimille. Ne olivat aina erilliset sivut, jotka oli toteutettu mobiililaitteille ottaen huomioon niiden pienemmät näyttökoot. Nykyään on kuitenkin tapahtunut selkeä muutos, eikä enää yleensä tehdä erikseen mobiilisivuja, vaan vain yksi responsiivinen sivusto, joka mukautuu kaikille laitteille.

Mobiililiikenne kattaa jo yli puolet liikenteestä internetissä. Statistan mukaan vuonna 2023 mobiililaitteet (tabletteja lukuun ottamatta) muodostivat 58,33 % maailman internetliikenteestä [Bianchi 2023: 2]. Responsiivisuus auttaa myös hakukoneiden näkyvyydessä. Google suosii hakutuloksissa responsiivisia sivuja erityisesti älypuhelin- ja tablettikäyttäjille, mikä sitten luonnollisesti pudottaa hakutuloksissa sivustoja, jotka eivät ole mobiiliystävällisiä. Verkkosivujen ylläpidon näkökulmasta responsiivinen verkkosuunnittelu säästää aikaa ja vähentää kustannuksia. Kaikki muutokset voi tehdä samaan paikkaan sen sijaan, että tekisi kahta eri sivustoa, toisin kuin jos ylläpitää erillistä mobiili- ja työpöytäversiota.

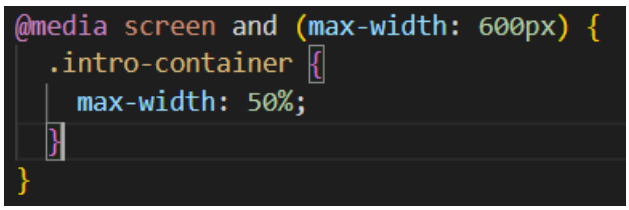
Suhteelliset yksiköt

Responsiivisuuden kannalta on hyvä käyttää suhteellisia yksiköitä, kuten prosentteja ja em/rem-yksiköitä elementeille ja tekstille. Suhteelliset yksiköt ovat responsiivisuuden kannalta parempia kuin absoluuttiset yksiköt, kuten pikseliyksikkö (px). Pikseli-mittayksiköllä elementti pysyy aina sen kokoisena, kuin isoksi px-yksikkö on sille laitettu näkymästä riippumatta, kun taas prosentti- ja

em/rem-yksiköiden avulla sivun elementti mukautuu eri näkymien kokoon. Esimerkiksi jos pöytäkoneella kohteelle antaa leveydeksi 800 px ja vaikka se näyttäisi hyvältä, se ei välttämättä näytä hyvältä puhelimen näytöllä, joka on 400 px leveä. Prosenttiyksikön avulla taas voisi kohteelle antaa 100 % leveydelle, jolloin kohde vie sen ympäröivän elementin verran eli yleensä koko ruudun leveyden. Em-yksikkö skaalautuu suhteessa elementin fonttikokoon, eli esimerkiksi 2 em tarkoittaa kaksi kertaa elementin nykyisen fontin kokoa. Rem-yksikkö on sama mutta vain suhteessa juurielementin fonttikokoon. [CSS Units: 3.]

Breakpointit

Sivustolle on myös hyvä laittaa keskeytyskohtia eli englanniksi breakpointeja. Breakpointit ovat kohtia, joissa sivuston rakenne muuttuu riippuen näytön koosta. Media query tai suomeksi mediakysely on CSS-tekniikka, jonka avulla voi ottaa käyttöön valittuja CSS-ominaisuuksia vain, jos tietty ehto on totta. Kuvassa 5 näkyy hyvin yksinkertainen versio mediakyselystä. Yleensä mediakyselyn ehdoksi valitaan ruudun leveys. Mediakyselyjen avulla voidaan sivustoon lisätä breakpointeja. Niiden avulla voidaan esimerkiksi vierekkäiset kuvat, jotka ruudun koon pienentyessä tulevat liian pieniksi, laittaa allekkain. Breakpointien avulla sivustoa voidaan skaalata paremmin sopimaan näytön resoluutiolle. Sivustolla olisi yleensä hyvä olla ainakin kolme breakpointia: pöytäkoneelle, kannettavalle tietokoneelle ja mobiililaitteille, mutta niitä voi tehdä niin monta kuin haluaa tai tarvitsee.



Kuva 5. Yksinkertainen mediakysely, kun ruudun laajuus on 600 px tai vähemmän.

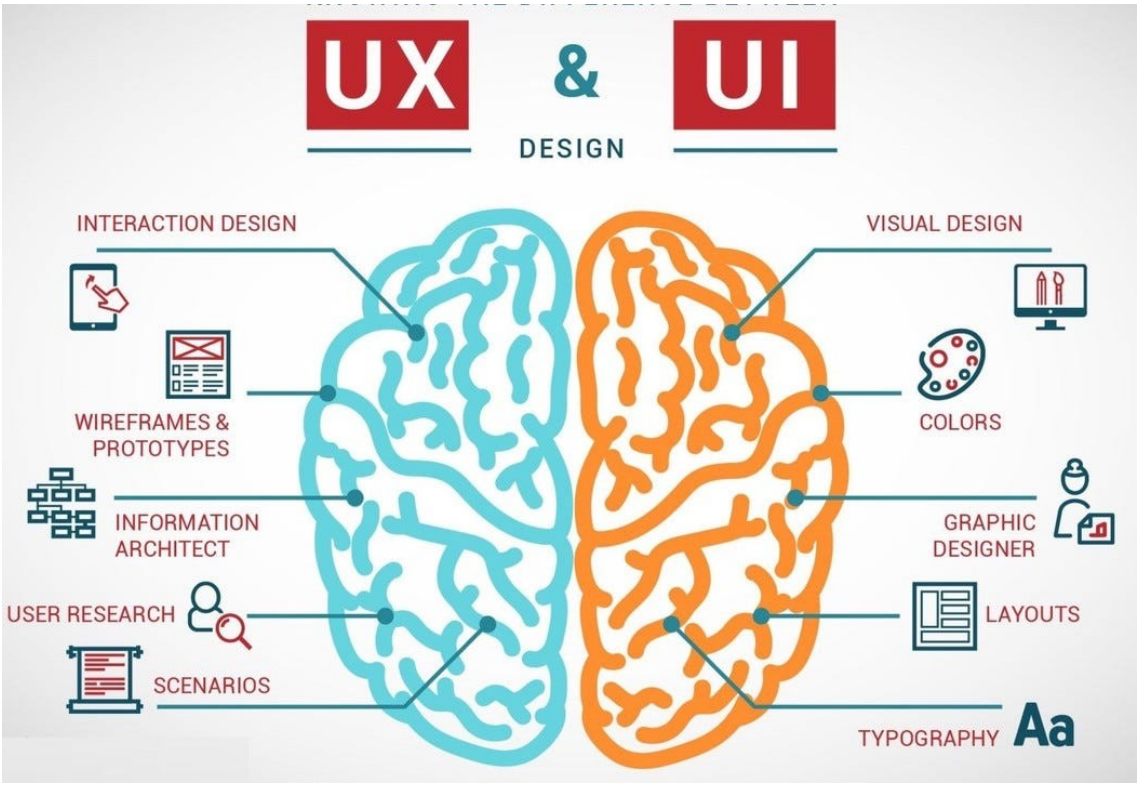
Yksi responsiivisten verkkosivujen parhaista puolista on se, että ne antavat hyvän käyttäjäkokemuksen laitteesta ja näytön koosta riippumatta. Hyvä käyttäjäkokemus myös saa ihmiset tulemaan takaisin sivustolle.

Responsiivinen media

Yksi suuri ongelma, joka pitää ratkaista responsiivisessa web-suunnittelussa, on kuvien kanssa työskentely. Yleensä jokainen kuva latautuu alkuperäiseen kokoonsa, ellei kuvaportti tule kapeammaksi kuin kuvan alkuperäinen leveys tai kun mikään muu leveyteen perustuva kuvan tyyli ei ohita tätä sääntöä. Toinen ongelma kuvissa on myös niiden koko levyllä. Kuvien kanssa pitää ottaa huomioon se, että käyttäjän pitää ladata ne, jotta ne näkyvät sivustolla. Tämä tarkoittaa, että jos kuvan levytila on liian suuri tai käyttäjällä ei ole hyvin nopeaa internetyhteyttä tai nopeutta, tulee sivun lataamisesta tuskallisen hidasta. Vaikka käyttäjällä olisi hyvä internetnopeus, olisi kuitenkin hyvä pitää kuvien koko levyllä mahdollisimman pienenä. Olisi siis hyvä antaa mobiililaitteille pienempi kuva ladattavaksi, eli olisi hyvä olla kaksi eri kuvaa, mobiililaitteille ja pöytäkoneelle.

3.2 Käyttöliittymäsuunnittelu

Käyttöliittymäsuunnittelu tai englanniksi UI-design (user interface design) keskittyy siihen, miten käyttäjä käyttää käyttöliittymää. Käyttöliittymä on siis ohjelman tai sivuston näkyvä osa, jota käyttäjä käyttää ohjelman tai sivuston käyttämiseen. UX-design (user experience design) eli käyttökokemuksen suunnittelu taas keskittyy siihen, miten käyttäjän kokemusta voidaan parantaa. Käyttöliittymäsuunnittelu koskee ainoastaan digitaalisia tuotteita tai palveluita, kun taas käyttökokemuksen suunnittelussa voidaan ottaa myös fyysiset tuotteet tai palvelut huomioon. [UX- ja UI-design – mitä sinunkin tulisi tietää web design -jargonista: 24.] Kuvassa 6 näkyy käyttöliittymä- ja käyttökokemussuunnittelun eroja.



Kuva 6. Käyttöliittymä- ja käyttökokemussuunnittelun eroavaisuuksia [Duckmanton 2019: 7].

Hyvä käyttöliittymä täyttää sivuston tai palvelun tavoitteet, ja se on suunniteltu käyttäjän tarpeisiin. Tärkein osa käyttöliittymästä on se, että sillä saa tehtyä tarvittavat toiminnot. Hienoimmastakaan käyttöliittymästä ei ole hyötyä, jos käyttäjä ei saa sen avulla tehtyä sivuston tai palvelun tarkoitettuja toimintoja. Käyttöliittymän pitäisi olla selkeä ja looginen, ja sitä pitäisi pystyä ennakoimaan, käyttäjän pitäisi ymmärtää, miten sivustoa tai palvelua navigoidaan intuitiivisesti. Käyttöliittymän pitäisi myös olla responsiivinen ja saavutettava. Responsiivisuutta nykyään odotetaan jo itsestäänselvänä sivustolla, kun erilaisia päätelaitteita käytetään paljon.

Käyttöliittymän pitäisi myös olla johdonmukainen varsinkin fonttien ja värien kannalta. Fonttien pitäisi olla luettavia, ja niiden koko ei saisi paljon vaihtua sivustoa selatessa. Värit pitäisi myös olla helppo ymmärtää tai olla itsestäänselviä, eli esimerkiksi hyväksy-napin väri ei saisi olla punainen, koska yleensä peruuta-nappeja merkitään punaisella värillä. Kontrasti on myös hyvä ottaa

huomioon värien ja varsinkin tekstin kanssa. [UX- ja UI-design – mitä sinunkin tulisi tietää web design -jargonista: 24.]

Saavutettavuus on myös tärkeä osa käyttöliittymää, ja käyttöliittymän pitäisi olla kaikkien saavutettavissa. Tästä syystä pitäisi miettiä, mitä kaikkea käyttäjän tarvitsee tehdä käyttääkseen hyvin käyttöliittymää ja miten jotkut käyttäjät eivät välttämättä pysty käyttämään sitä.

3.3 React-JavaScript-kirjasto

React on Metan (ennen Facebook) tekemä ilmainen ja avoimeen lähdekoodiin perustuva JavaScript-kirjasto käyttöliittymäkomponentteihin perustuvien käyttöliittymien rakentamiseen. Reactia käytetään interaktiivisten käyttöliittymien ja verkkosovellusten rakentamiseen nopeasti ja tehokkaasti, huomattavasti vähemmällä koodilla kuin perus-JavaScriptillä. [Herbert 2022: 10.] Monet kehittäjät käyttävät Reactia rakentaakseen skaalautuvia, yksinkertaisia ja nopeita käyttöliittymiä yksisivuisille tai monisivuisille verkkosovelluksille. Reactin avulla pystyy myös tekemään virheenkorjauksia helposti. [Patadiya 2022: 16.]

React julkaistiin vuonna 2013, ja se on nykyään yksi yleisimmin käytetyistä käyttöliittymäkirjastoista verkkokehitykseen. React on JavaScript-kirjasto, joten sen käyttämistä varten tarvitaan ainakin perusosaaminen JavaScript-kielestä. Reactia on kuitenkin helppo oppia, sillä se yhdistää HTML:n ja JavaScriptin peruskäsitteitä muutamilla hyödyllisillä lisäyksillä. Reactissa on myös mahdollista kirjoittaa HTML-rakenteita samaan tiedostoon JavaScript-koodin kanssa. [Deshpande 2023: 5.]

React käyttää virtuaalista DOM:a (Document Object Model). Kun objektin tila muuttuu React-sovelluksessa, virtuaalinen DOM päivitetään, ja se sitten vertaa aiempaa tilaansa ja päivittää vain ne objektit, jotka muuttuivat ”oikeaan” DOM:iin sen sijasta, että päivittäisi kaikki objektit. Todellisen DOM:n muokkaaminen on huomattavasti hitaampaa kuin virtuaalisen DOM:n muokkaaminen. Reactin avulla pystyy rakentamaan niin sanottuja yksisivuisia verkkosovelluksia.

Yksisivuinen verkkosovellus lataa vain yhden HTML-dokumentin ensimmäisellä pyynnöllä. Sitten se päivittää verkkosivun tietyn osan, sisällön tai tekstiosan, joka pitää päivittää käyttäen JavaScriptiä. Palvelimen ei tarvitse ladata koko verkkosivua uudelleen saadakseen uuden sivun aina kun käyttäjä tekee uuden pyynnön. [Deshpande 2023: 5.]

React erottaa käyttöliittymän useisiin itsenäisiin ja käyttökelpoisiin osiin, jotka voidaan käsitellä erikseen. Reactissa sovelluksia kehitetään luomalla uudelleenkäytettäviä komponentteja. Komponentit ovat React-sovellusten rakennuspalikat, ja yksi sovellus koostuu yleensä useista komponenteista. [Deshpande 2023: 5.] Kuvassa 7 näkyy hyvin yksinkertainen React-komponentti. Sen sijaan, että käsittelisi koko käyttöliittymää yhtenä kokonaisuutena, React rohkaisee kehittäjiä erottamaan nämä monimutkaiset käyttöliittymät yksittäisiksi uudelleen käytettäviksi komponenteiksi, jotka muodostavat käyttöliittymän rakennuspalikat. Näillä komponenteilla on logiikkansa ja ohjauksensa, ja niitä voidaan käyttää uudelleen kaikkialla sovelluksessa, mikä puolestaan lyhentää sovelluksen kehitysaikaa paljon.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Kuva 7. Yksinkertainen React-komponentti.

Reactia käytetään sekä verkko- että mobiilisovellusten kehittämiseen. Siitä johdettu React Native -kehys on myös erittäin suosittu, ja sitä käytetään mobiilisovellusten luomiseen. React Nativen avulla pystyy tekemään mobiilisovelluksia Android- ja IOS-laitteille.

Material UI tai MUI on komponenttikirjasto Reactille, ja se on yksi suosituimmista komponenttikirjastoista. MUI on käytössä monilla isoilla sivustoilla, kuten Amazon, Netflix ja Spotify. MUI tarjoaa kattavan valikoiman

käyttöliittymäkomponentteja ja valmiiksi tehtyjä komponentteja, joita on helppo lisätä sivustoon. [Move faster with intuitive React UI tools: 15.] Kuvassa 8 näkyy MUI:n ”container”-komponentti. Saavutettavuus on myös jo otettu huomioon komponenteissa, kuten esimerkiksi näppäimistönavigointi on jo sisään rakennettuna elementeissä, joten siitä ei itse tarvitse huolehtia. Esimerkiksi ruudunlukulaitteet tunnistavat, mikä komponentti on kyseessä ilman ARIA:n erikseen lisäystä. Komponenttien päälle on helppo luoda omia komponentteja tai tyylejä, mikä tekee MUI:sta hyvin mukautettavan.

```
export default function SimpleContainer() {
  return (
    <React.Fragment>
      <CssBaseline />
      <Container maxWidth="sm">
        <Box sx={{ bgcolor: '#cfe8fc', height: '100vh' }} />
      </Container>
    </React.Fragment>
  );
}
```

Kuva 8. MUI:n ”container”-komponentti.

4 Saavutettavuus

Termejä saavutettavuus ja esteettömyys käytetään usein toistensa synonyymeina. Saavutettavuutta käytetään silloin, kun kyseessä on ”aineeton” ympäristö, kuten verkkopalvelut ja verkkosivut, ja esteettömyys taas on käytössä silloin, kun kyseessä on fyysinen ympäristö. Saavutettavuus tarkoittaa sitä, että mahdollisimman moni ihminen voi käyttää verkkopalvelua, verkkosivua ja mobiilisovellusta mahdollisimman helposti. Tämä tarkoittaa sitä, että verkkosisältö on ymmärrettävissä ja sitä voi myös käyttää erilaisten avustavien teknologioiden kanssa. Verkkopalvelussa on myös huomioitava, että sisältö on selkeää ja ymmärrettävää ja se on luettavissa ruudunlukuapuvälineillä. [Verkkosisältöjen saavutettavuus: 25.]

4.1 Saavutettavuuden periaatteet

Saavutettavuus parantaa käyttäjäkokemusta kaikille ihmisille ja on tarpeellinen ominaisuus käyttäjille, joilla on esimerkiksi näköongelmia, niin että he pystyvät käyttämään sivustoa tai palvelua kunnolla. Saavutettavuudessa on oleellista, että erilaisten käyttäjien tarpeet otetaan mahdollisimman hyvin huomioon, kun suunnitellaan ja toteutetaan verkkosivuja tai palveluja. Saavutettavan verkkosivuston suunnittelussa ja toteutuksessa olisi syytä huomioida kolme osa-aluetta:

- tekninen toteutus
- helppokäyttöisyys ja
- sisältöjen selkeys ja ymmärrettävyys.

Tekninen saavutettavuus on se, että sivuston lähdekoodi on loogista ja tehty virheettömästi. Sivustossa on noudatettu standardeja ja saavutettavuuden ohjeistuksia. Sivuston tulisi myös toimia eri päätelaitteilla ja toimia eri apuohjelmien kanssa, kuten ruudunlukuohjelmien kanssa.

Helppokäyttöisyys on sitä, että sivustoa on helppo hahmottaa ja navigoida ja toiminnot ja sisältö löytyvät vaivattomasti. Navigaatio ei saisi olla liian monitasoinen, ja sivujen nimien pitäisi olla kuvaavia ja selkeitä. Pääsisällön pitäisi erottua selkeästi sivun muista elementeistä. Palvelussa pitäisi olla vaivatonta suorittaa haluttu toiminto.

Ymmärrettävyys taas on sitä, että sivustolla käytetään selkeää ja ymmärrettävää kieltä. Sivuston teksti on jäsennelty helppolukuiseksi, ja sivustolla on käytetty kuvaavia väliotsikoita, kuvatekstiä ja linkkitekstiä hyvin. Sivuston sisältö saisi olla myös enemmän monikanavaista eli sivustolla voisi olla enemmän videoita, kuvia ja ääntä tekstin lisäksi. [Yleistä saavutettavuudesta: 30.]

Suomessa aluehallintovirasto valvoo saavutettavuutta, mutta se on kuitenkin vain yksi monesta tehtävästä, joita virasto tekee. Aluehallintovirasto on valvova valtion viranomainen, jonka tehtävänkuvaan kuuluu erilaisia oikeusturvaan, perusoikeuksiin ja turvalliseen ympäristöön liittyviä valvontatehtäviä.

Aluehallintovirastoja on yhteensä seitsemän, ja yleensä jokainen niistä valvoo vain omaa toiminta-aluettaan. Joitain valvontatehtäviä kuitenkin valvoo vain yksi virasto, kuten saavutettavuutta valvoo vain Etelä-Suomen aluehallintovirasto. Saavutettavuudessa viraston tehtäviin kuuluu

- yleisen ohjauksen ja neuvonnan anto saavutettavuusvaatimuksista ja saavutettavuuteen liittyvien oikeuksien toteuttamisesta
- kantelujen ja selvityspyyntöjen käsittely digitaalisten palvelujen saavutettavuuspuitteista
- vuosittaista valvontaa EU-komission ehtojen mukaisesti
- valvonnan tulosten raportointi EU-komissiolle
- kansainvälisten saavutettavuusohjeiden valmistelun osallistuminen.

Valvonnan tavoitteena on selvittää, täyttävätkö eri verkkosivustot ja mobiilisovellukset digipalvelulain saavutettavuusvaatimukset. [Digipalvelulain vaatimukset: 6.]

4.2 WCAG-ohjeistus

WCAG (Web Content Accessibility Guidelines) on kansainvälisen World Wide Web -konsortion (W3C) kehittämä ja ylläpitämä ohjeistus verkkosisältöjen saavutettavuudesta. WCAG-ohjeistuksen ensimmäinen versio julkaistiin vuonna 1999. WCAG 2.0 -versio julkaistiin vuonna 2008, ja sitä käytetään monien maiden lainsäädännön saavutettavuusvaatimusten perustana. Suomessa ja EU-maissa saavutettavuusvaatimuksissa edellytetään kesäkuussa 2018 hyväksytyn WCAG 2.1 -version noudattamista. [Tietoa WCAG-ohjeistuksesta: 23.]

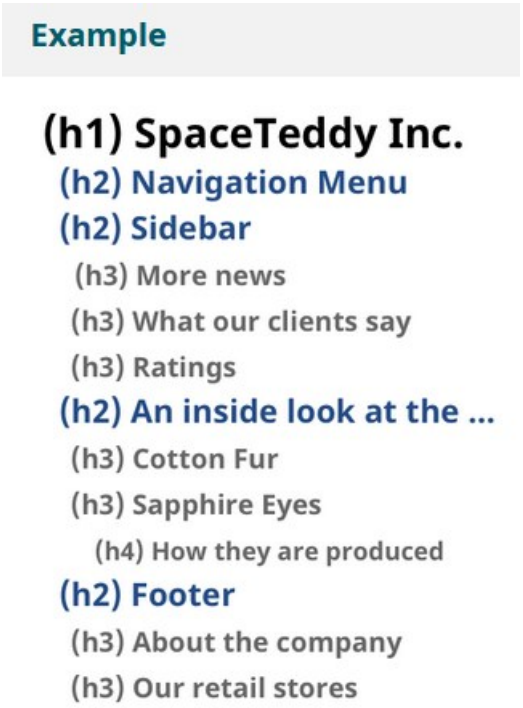
WCAG-kriteerit on jaettu kolmeen tasoon: A-, AA- ja AAA-tasoon. Verkkopalvelu, joka täyttää AAA-tason kriteerit, varmistaa saavutettavuuden mahdollisimman laajalle joukolle henkilöitä. AAA-tasoa koskevat siis tiukimmat vaatimukset. Julkisia toimijoita velvoitetaan digitaalisten palvelujen laissa toteuttamaan verkkopalvelut niin, että WCAG 2.1 -standardin kriteerit täyttyvät A- ja AA-tasoilla. [WCAG: 27.]

Verkkopalvelun saavutettavuutta parantaa WCAG-ohjeistuksen noudattaminen, etenkin tekniseltä kannalta. Verkkosisältöjen ymmärrettävyyteen ja verkkopalvelun käytettävyyteen WCAG-ohjeistus ei juurikaan ota kantaa. [WCAG: 27.] WCAG-ohjeistusta noudattamalle ei vielä takaa, että sivusto olisi kaikille käyttäjille saavutettava tai helppokäyttöinen vaikka se on tärkeä ohjeistus saavutettavuuden kehittämiseksi. WCAG-ohjeistuksen A-tason kriteerien noudattamisesta voikin sanoa, että se varmistaa saavutettavuuden minimitason. [Tietoa WCAG-ohjeistuksesta: 23.]

A-taso on perustaso, jonka kriteerit parantavat saavutettavuutta osalle käyttäjistä, varsinkin jos käyttäjillä on erityisiä haasteita verkkopalvelujen käytössä. Videoiden tekstitysvaatimus on esimerkiksi A-tason kriteeri. AA-tason kriteerit parantavat saavutettavuutta vielä entistä laajemmalle joukolle. Kuvailutulkkauksen tai ääniselitteen tarjoaminen videoille on esimerkiksi AA-tason kriteeri. AAA-tason kriteerit parantavat saavutettavuutta vielä useammalle: sisältöjen tarjoaminen viittomakielisinä videoina ja tekstin ymmärrettävyyden parantaminen ovat esimerkiksi AAA-tason kriteerejä. WCAG-ohjeistus on kaiken kaikkiaan valtava tietokokonaisuus, joka sisältäisi noin tuhat A4-sivua tulostettuna. [Tietoa WCAG-ohjeistuksesta: 23.]

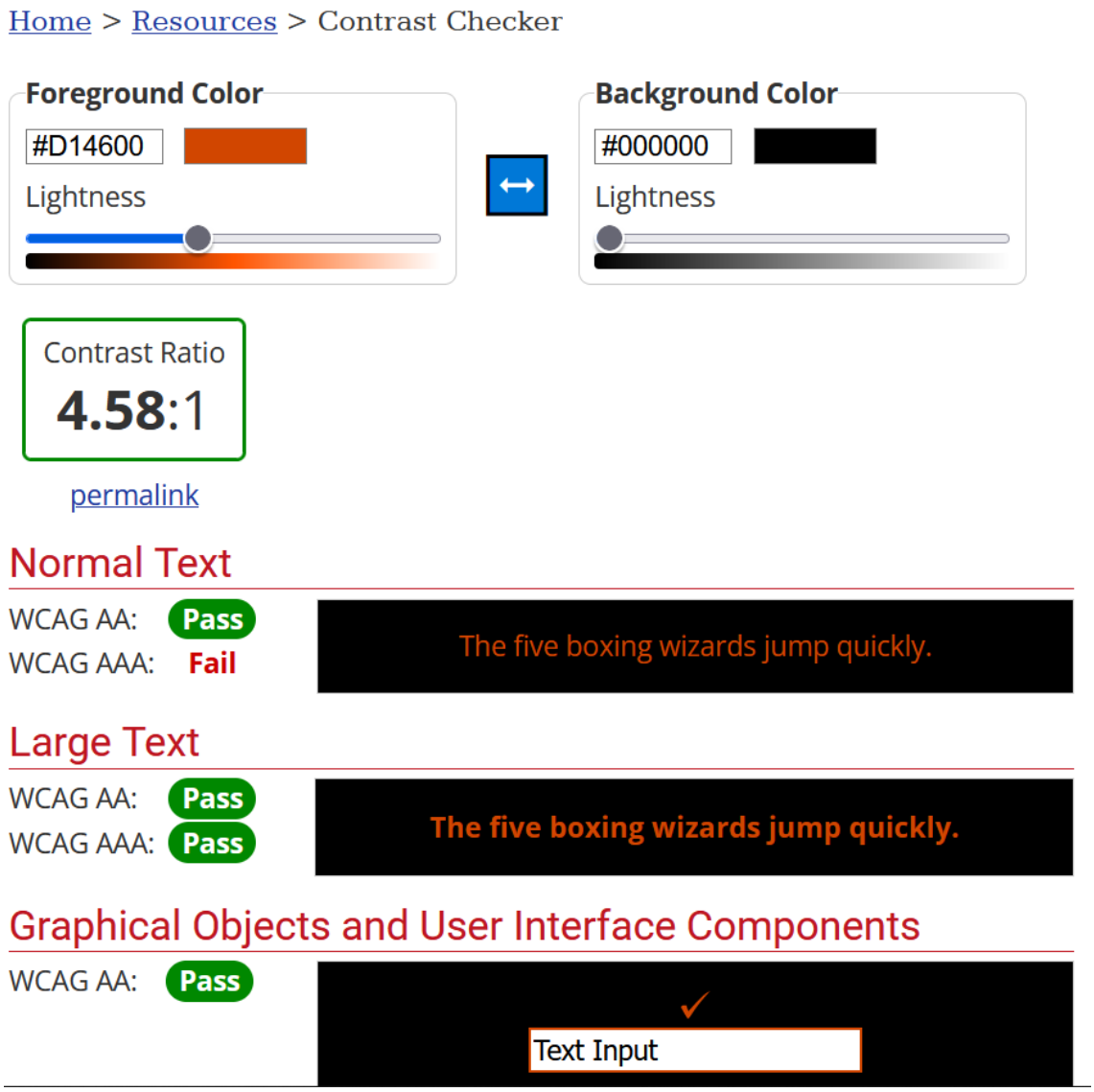
4.3 Yleisimpiä saavutettavuuden ongelmia

Yleisimmät ongelmat, joita verkkosivujen saavutettavuudessa löytyy, ovat otsikoissa, kontrastissa, etiketeissä (label-elementti), linkeissä ja kuvien vaihtoehtoisessa tekstissä. Saavutettavuuden ongelmana otsikoissa on yleensä se, että verkkosivujen otsikot eivät aina ala pääotsikolla (h1) ja otsikot saattavat ohittaa tasoja sivuilla. [The WebAIM million 2022: 22.] Ruudunlukulaitteet yleensä käyttävät otsikoita sivun navigointiin, ja siksi sivuja, joilla ei ole hyvää otsikkohierarkiaa, on vaikeampi käydä ruudunlukulaitteilla läpi. Kuvassa 9 näkyy saavutettavasti tehty järjestys otsikoille.



Kuva 9. Saavutettava otsikkojärjestys [Headings: 11].

Yleisin saavutettavuuden ongelma sivuilla on tekstin ja taustan kontrasti. WebAIM arvioi, että 83,6 prosentilla kotisivuista on kontrastiongelmia [The WebAIM million 2022: 22]. WCAG 2.1 -ohjeistuksen [Web Content Accessibility Guidelines (WCAG) 2.1: 28] mukaan värien kontrastisuhde 4,5:1 on verkkosivujen saavutettavuuden kultainen standardi tyypillisessä tekstissä. Kuvassa 10 näkyy WebAIM-sivuston kontrastin tarkistustyökalu kontrastisuhteella 4,58:1. Kontrasti on kuitenkin yleensä helppo ongelma korjata: verkossa on paljon ilmaisia kontrastintarkastus-sivuja ja lisäosia, kuten WebAIM. Kontrasti on hyvä ottaa huomioon aikaisin projektissa, kun sivujen väriteemaa ja ulkonäköä mietitään.



Kuva 10. WebAIM-sivuston kontrastintarkistustyökalu [Contrast Checker: 4].

Label-elementti edustaa nimikkeen kuvatekstiä käyttöliittymässä. Yleisin saavutettavuuden ongelma label-elementtien kanssa on se, että ne on jätetty tyhjiksi. Ongelmia tulee varsinkin ruudunlukulaitteita käyttäville, kun nämä laitteet eivät aina ymmärrä esimerkiksi lomakekenttiä, joissa ei ole label-elementtiä. Label-elementtien kanssa voi kuitenkin tulla helposti saavutettavuusongelmia, joten niin niitä olisi aina hyvä käydä läpi saavutettavuudentarkistusohjelmalla.

Linkkien kanssa olisi hyvä, jos ne olisivat helposti tunnistettavia, niitä ei olisi liian paljon ja ne olisivat tarpeeksi kuvailevia. Ruudunlukulaitteilla on vaikea

käydä sivustoa läpi, jos näitä seikkoja ei oteta huomioon linkeissä. Ruudunlukulaitteet lukevat sivuston sellaisena kuin se on, joten linkeissä olisi hyvä käyttää muuta kuin ”klikkaa tästä” -tekstiä ja sen sijaan kertoa lyhyesti, mihin linkki vie. Ruudunlukulaitteet eivät myöskään välttämättä huomaa linkkejä tai ne saattavat lukevat navigointilinkit joka kerta, kun sivu latautuu. Yksi tapa, jolla tämän voisi saada korjattua on määrittää oikeat ARIA-roolit linkeille.

Kuvien vaihtoehtoinen teksti on myös hyvin yleinen saavutettavuuden ongelma-alue. WebAIM arvioi, että kuvien vaihtoehtoinen teksti puuttuu 58,2 prosentissa verkkosivustoista [The WebAIM million 2022: 22]. Varsinkin ruudunlukulaitteita käyttäville on vaihtoehtoinen teksti tärkeä, kun ei laite pysty itse kääntämään kuvaa tekstiksi. Samoin kuin linkeissä, vaihtoehtoisen tekstin pitäisi olla lyhyt ja kuvaileva. Yksi tapa, jolla kuville voi lisätä vaihtoehtoisen tekstin on ARIA. ARIA (Accessible Rich Internet Applications) on joukko rooleja ja ominaisuuksia, joiden avulla on helpompi tehdä saavutettavaa verkkosisältöä. Kuvassa 11 näkyy esimerkki ARIA-roolista.

HTML

```
<svg role="img" aria-label="Description of your SVG image">
  <!-- contents of the SVG image -->
</svg>
```

Kuva 11. ARIA-kuvarooli svg-elementissä.

Yleensä ARIA:n kanssa kuitenkin suositellaan, että jos tavallisella HTML-elementillä pystyy tekemään saman, minkä ARIA-elementti voisi tehdä, olisi parempi käyttää tavallista HTML-elementtiä. ARIA ei myöskään itsestään tee sivustoa saavutettavammaksi: jos sitä käyttää väärin, siitä voi olla enemmän haittaa kuin hyötyä.

5 Keikkakaveri-sivuston työstäminen

Insinöörityöprojektissa käytettiin Visual Studio Code -koodieditoria, GitHubia ja MongoDB Cloud -tietokantaohjelmaa. Käytössä olivat myös React ja komponenttikirjasto MUI, koska Keikkakaveri oli tehty niitä käyttäen. Enimmäkseen responsiivisuudessa kuitenkin pysyttiin vain HTML:ssä ja CSS:ssä. Saavutettavuuden testauksessa käytettiin kolmea eri saavutettavuuden testauksen työkalua: WAVE, Tota11y ja Axe. Aluksi tutkittiin myös, mitä saavutettavuus on ja mikä Kestävä Keikkatyö -hanke on. Projektin jälkeen tutustuttiin myös hieman pariin web-sovellusten automatisoituun testaustyökaluun.

GitHub on monelle koodin kirjoittajalle tuttu työkalu. Se on koodin isännöintialusta version hallintaan ja yhteistyötä varten, ja se mahdollistaa ryhmien yhdessä työskentelyn projekteissa missä tahansa ryhmät olisivatkaan. GitHub tekee Gitin käytön versionhallintaa varten todella helpoksi. Git on ilmainen ja avoimen lähdekoodin hajautettu versionhallintajärjestelmä, jota GitHub käyttää. GitHub on myös ilmainen, ja kuka tahansa voi rekisteröityä ja alkaa isännöidä julkista koodivarastoa, mikä tekee GitHubista erityisen suositun avoimen lähdekoodin projekteissa. [What is GitHub? A Beginner’s Introduction to GitHub 2022: 29.]

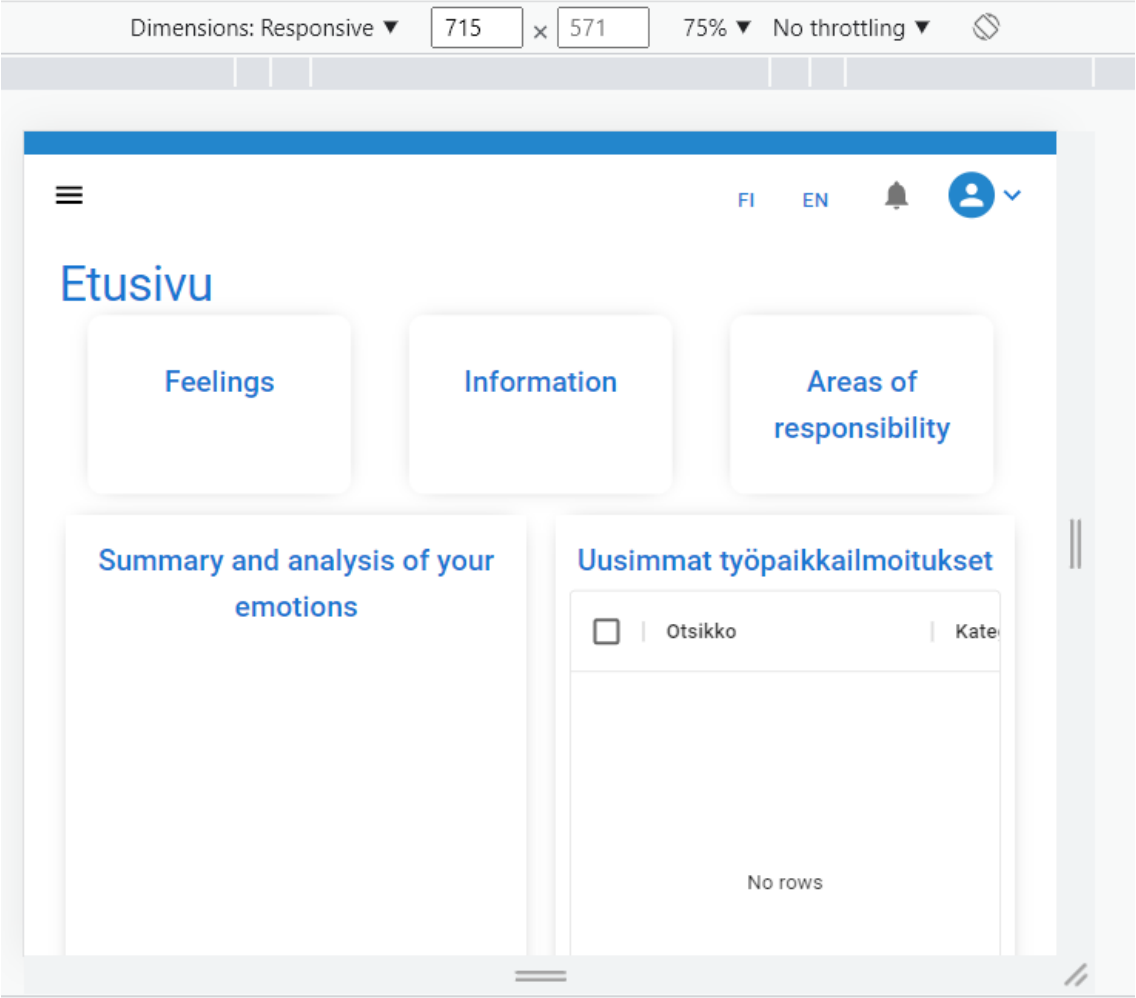
MongoDB on MongoDB Inc:n kehittämä dokumenttiorientoitunut tietokantaohjelmisto. Projektissa käytettiin MongoDB Cloudia projektin tietokantana silloin, kun insinöörityötä tehtiin projektissa. MongoDB:hen piti tehdä käyttäjätili ja oma tietokanta, jotta pystyi tekemään käyttäjätilejä projektiin, ja niiden avulla pääsi käymään kaikilla projektin sivuilla. [For the next generation of intelligent applications: 8.]

Työn alussa projekti ladattiin GitHubista ja tehtiin pienet muokkaukset ”env”-tiedostoon, jotta saatiin projekti pyörimään omalla tietokoneella MongoDB:llä. MongoDB oli tärkeä, koska sen avulla sai tehtyä käyttäjätilejä ja niiden avulla pääsi käsiksi kaikkiin Keikkakaverin sivuihin. Kun projektin sai pyörimään

omalla koneella ja sai yhdistettyä MongoDB-tietokantaan, pystyi alkamaan käydä sivustoa läpi.

5.1 Responsiivisuuden parantaminen

Responsiivisuuden parantamisessa käytettiin Chrome-selaimen sisäänrakennettuja DevTools-työkaluja. DevTools-työkaluihin sisältyy ”Device mode”, joka on kokoelma ominaisuuksia, jotka auttavat simuloimaan mobiililaitteita ja tabletteja tietokoneella. Kuvassa 12 näkyy Device mode käytettynä Keikkakaveri-sivuston käyttäjäroolin etusivulla. Tällä sai helposti katsottua, miltä sivut näyttävät tabletissa tai puhelimessa. Sivuja käytiin läpi samalla, kun vaihdettiin resoluutiota eri laitteiden resoluutioihin ja katsottiin, tuliko missään ongelmia. Kun ongelmia esiintyi, yritettiin selvittää, miksi ja mistä niitä tuli, minkä jälkeen ne korjattiin.

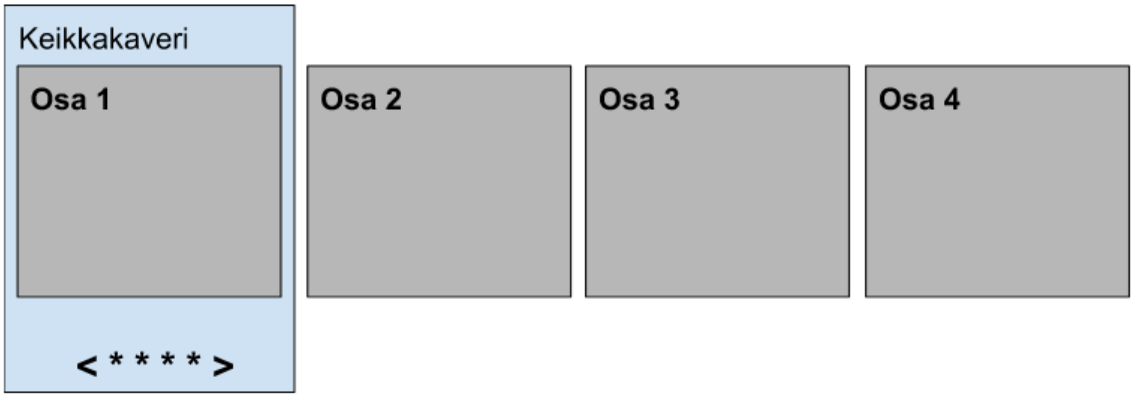


Kuva 12. Keikkakaverin työntekijä-käyttäjäroolin etusivu katsottuna DevToolsin Device Modea käyttäen.

Sivusto oli enimmäkseen jo aika hyvin tehty responsiivisuuden kannalta, vain joillain sivuilla tuli ongelmia. MUI-kirjastoa oli käytetty melkein kaikilla sivuilla paitsi kotisivulla, jossa meni eniten aikaa ongelmien korjaamisessa. Suurimmaksi osaksi sivut olivat jo olleet hyvin saavutettavia ja responsiivisia MUI-kirjaston takia. Joillain sivuilla kuitenkin tuli vielä muita ongelmia, esimerkiksi taulukoissa.

Kotisivu oli jaettu neljään eri horisontaaliseen osaan, kuten kuvassa 13 näkyy. Kotisivulla siis näkyi vain yksi osa kerrallaan, ja sivulla piti klikata sivun pohjassa olevaa nuolta tai palloa siirtyäkseen seuraavaan osaan. Tässä tuli

ongelmia varsinkin mobiililaitteiden ja tablettien kannalta, kun niillä oli vaikea huomata, että sivussa oli enemmän osia.



Kuva 13. Keikkakaveri-sivuston etusivu jaettu neljään osaan.

Mobiililaitteille ja tabletille vaihdettiin kotisivun tavalliseksi vertikaaliseksi vierittäväksi sivuksi, kuten kuvassa 14 näkyy. Tämän saavuttamiseksi piti yrittää yhdistää kotisivun neljä eri osaa yhdeksi jatkuvaksi sivuksi. Tässä rupesi tulemaan ongelmia. Ensimmäisen osan tausta piti saada pysymään vain ensimmäisessä osassa, kun se tuli muidenkin sivujen taustaksi. Ensimmäisen osan kuvien kanssa tuli myös ongelmia, kun kaikki kuvat eivät enää kaikkien muutoksien jälkeen mahtuneet kunnolla sivulle. Muita ongelmia kotisivuilla oli sivun neljännen osan ”footer”-tekstin kanssa, jossa sivun teksti ei pienentynyt näytön koon mukaan.

Kuva 14. Keikkakaverin kotisivu tablettiresoluutiolla.

Jokaisen käyttäjäroolin sivut piti myös erikseen tarkistaa, että ne olivat responsiivisia. Responsiivisuusongelmia löytyi myös käyttäjärooli-sivujen taulukoissa, jotka eivät skaalautuneet mobiili- tai tablettikokoon. Taulukoissa piti vaihtaa taulukoille annettu ruutujen koko niiden minimiko'oiksi ja antaa flex-ominaisuus ruuduille, jotta taulukot tulivat skaalautuviksi. Taulukoille itsessään piti myös antaa koko, jotta ne pysyivät ruudun kokoisina. Esimerkkikoodissa 1 näkyy kahden ruudun koodi muutoksien jälkeen.

```
{
     field: “email”,
     headerName: (i18next.t(“list_email”)),
     minWidth: 200,
     flex: 1
},
{
     field: “city”,
     headerName: (i18next.t(“list_email”)),
     minWidth: 125,
     flex: 1
},
```

Esimerkkikoodi 1. Taulukon kahden ruudun koodi responsiivisuuspäivityksien jälkeen.

Joillain sivuilla piti myös vaihtaa sivujen pituutta tai sivujen elementtien pituutta pois pikseliko’oista. Pikseliyksikkö ei ole responsiivisuuden kannalta aina hyvä pituuden yksikkö, koska pikselit ovat absoluuttisia yksiköitä ja absoluuttiset yksiköt pysyvät aina samankokoisina näytöstä riippumatta. Tämän takia pikselit vaihdettiin vh-yksiköiksi, joissa 1 vh vastaa 1 %:a ruudun korkeudesta eli 100 vh olisi aina 100 %:a ruudun korkeudesta. Tämä tarkoittaa sitä, että kun sivustoa katsotaan pöytäkoneella ja vaihdetaan mobiilille, niin sivuston rakenne pysyy samanlaisena, ellei sitä ole erikseen määritelty vaihtuvaksi ruudun koon mukaan.

5.2 Saavutettavuuden parantaminen

Saavutettavuuden tarkistuksessa käytettiin kolmea eri saavutettavuudentarkistusohjelmaa: WAVE, Tota11y ja Axe. Nämä ohjelmat ovat ilmaisia, helposti ladattavissa ja niitä pystyi käyttämään tekemättä käyttäjätiliä.

WAVE

WAVE on Utahin osavaltion yliopistossa tehty internetin saavutettavuuden arviointityökalu, jota WebAim kehittää. WAVE on sarja arviointityökaluja, jotka auttavat tekijöitä tekemään verkkosisällöstä helpommin käytettävää. WAVE voi tunnistaa monia saavutettavuus- ja WCAG-virheitä, mutta myös helpottaa verkkosisällön inhimillistä arviointia. WAVE:lla on ilmainen selaimen laajennus, joka on saatavilla Chromelle, Firefoxille ja Edgelle. Laajennus mahdollistaa

verkkosisällön esteettömyysongelmien arvioinnin suoraan selaimessa. [Wave Web Accessibility Evaluation Tools: 26.] Laajennuksessa oli ongelmana vain, että se avautui sivun rinnalle, jolloin sivun tila pieneni. WAVE siis helposti hajotti sivuston responsiivisuuden kannalta, ja sen kanssa piti hieman temppuilla joka kerta, kun sen avasi. WAVE myös näyttää runsaasti eri merkkejä siihen, mitkä elementit ovat rikki ja mitkä ovat hyviä. Nämä merkit myös vaikeuttivat sivun käyttöä, kun ne välillä siirsivät muita sivun elementtejä. Ne sai kuitenkin muutettua WAVE:ssa, mutta kesti jonkin aikaa, ennen kuin selvisi, miten asian pystyi korjaamaan.

Tota11y

Tota11y on Khan Academyn luoma saavutettavuuden visualisoinnin työkalu, jonka tavoitteena on helpottaa saavutettavuuden testausta. Tota11yn tavoitteena on helpottaa saavutettavuuden korjausta tarjoamalla hauska ja interaktiivinen tapa nähdä saavutettavuusongelmat. Sen sijaan, että käytäisiin läpi pitkiä tarkastusraportteja, joita voi olla vaikea hahmottaa, Tota11yn tekijöiden mielestä olisi parempi tarjota yksinkertaisia visualisointeja selaimessa. [Scales 2015: 19.]

Tota11y itsessään on JavaScript-tiedosto, joka lisää pienen painikkeen verkkosivun alakulmaan. Kuvassa 15 näkyy Tota11y avattuna Keikkakaverin kotisivulla. Tota11y toimii kirjanmerkkinä, ja sitä voidaan käyttää missä tahansa tietokoneen selaimessa. Selaimessa täytyy vain klikata kirjanmerkkiä, niin Tota11y lähtee käyntiin sivulla. Tota11y tekee helpoksi joidenkin yleisimpien saavutettavuusongelmien havaitsemisen. Virheiden osoittamisen lisäksi Tota11y ehdottaa myös tapoja korjata nämä rikkomukset.

Kuva 15. Tota11y avattuna Keikkakaverin kotisivulla.

Tota11yn työkalupalkki sisältää monia toimintoja. Monet niistä merkitsevät sivun elementtejä, yleensä näyttääkseen, että elementit löytyvät tai kun jokin on vialla sivulla. Tota11y sisältää seuraavia ominaisuuksia:

- Tunnistaa kuvia, joissa on vaihtoehtoista tekstiä ja ilman olevia.
- Merkitsee tekstin, jossa on kontrastirikkomuksia ja ehdottaa sopivia väriyhdistelmiä sen korjaamiseen.
- Hahmottaa asiakirjan otsikkorakenteen ja osoittaa mahdolliset virheet siinä.
- Korostaa syöttökenttiä, joissa ei ole asianmukaisia tunnisteita ja ehdottaa korjauksia kontekstin perusteella.
- Merkitsee kaikki sivulla olevat ARIA-merkit.
- Havaitsee epäselvän linkkitekstin.

Axe

Axe on Dequen kehittämä sarja työkaluja saavutettavuuden testaamista varten. Se koostuu kolmesta eri ohjelmasta: DevTools, Auditor ja Monitor. Ne kaikki toimivat Dequen rakentamalla Axe-core-saavutettavuussääntöjen moottorilla, joka perustuu avoimeen lähdekoodiin. Axe-coressa on erityyppisiä sääntöjä WCAG 2.0:lle ja 2.1:lle tasoille A ja AA sekä useita parhaita käytäntöjä, jotka auttavat tunnistamaan yleisiä saavutettavuuskäytäntöjä. [Axe Accessibility Testing Tools

are the Best on the Planet: 1.] DevTools oli ainoa Axen ohjelmista, jota työssä käytettiin. Axea käytettiin vähiten kolmesta valitusta saavutettavuudentarkistusohjelmasta. DevToolsista on ilmainen selainlaajennus saavutettavuuden testausta varten. Tässä ilmaisessa versiossa ei ole kaikkia ominaisuuksia, vaan jos haluaa enemmän toimintoja kuten yksittäisten komponenttien testauksen, pitää DevToolsista maksaa kuukausittainen maksu.

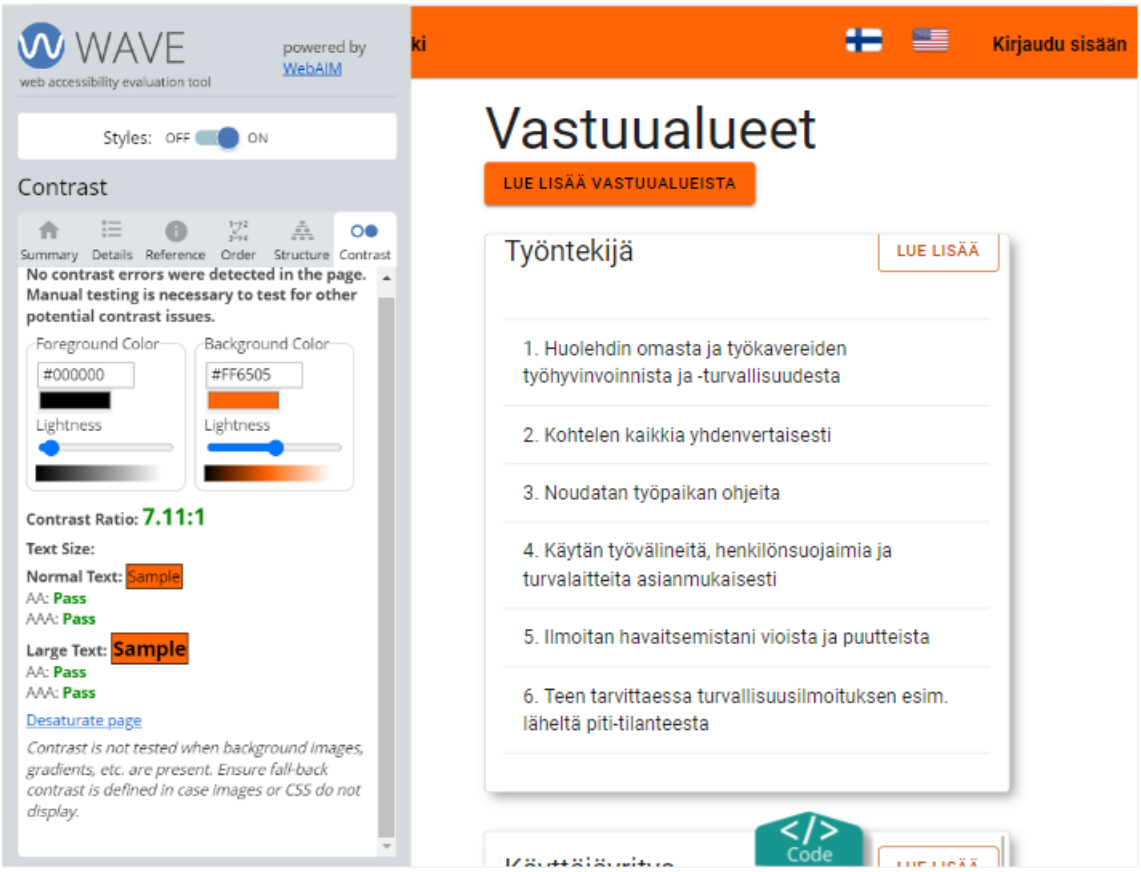
Saavutettavuuden tarkistus työkalujen avulla

Saavutettavuudentarkistusohjelmat löysivät paljon saavutettavuusvirheitä ja ongelmia, mutta monet niistä oli suhteellisen helppo korjata. Virheitä löytyi enimmäkseen otsikoissa, kontrastissa, label-elementeissä ja ARIA:ssa. Kaikki ohjelmat antoivat melkein samoja virheitä tai ongelmia, mutta välillä yksi ohjelma saattoi antaa virhettä, jota muut ohjelmat eivät antaneet. Ohjelmat eivät siis olleet yksimielisiä siinä, missä kaikkialla oli saavutettavuusvirheitä. Jotkin ohjelmat myös antoivat enemmän tai vaikeampia virheitä, kuten Axe, jonka antamat virheet vain kirjattiin ja annettiin projektin lopussa hankkeelle.

Otsikot kertovat sivun sisällön järjestyksen. Verkkoselaimet, laajennukset ja avustavat tekniikat pystyvät käyttämään niitä sivun sisäiseen navigointiin. Saavutettavuuden kannalta olisi hyvä, että sivun teksti alkaa <h1>-otsikolla. Tämä myös auttaa tekstistä puheeksi -ohjelmia löytämään, mistä teksti alkaa. Otsikkojen ohittaminen voi olla hämmentävää, ja sitä tulisi välttää, jos mahdollista. Esimerkiksi olisi hyvä välttää, että <h2>-otsikon jälkeen ei tule suoraan <h4>-otsikkoa. Projektin sivuilla oli käytetty erilaisia otsikoita sen mukaan, minkä koko näytti hyvältä, sen sijaan, että olisi seurattu saavutettavaa otsikkojärjestystä. Kuvassa 16 näkyy Keikkakaverin kotisivun otsikkojärjestys.

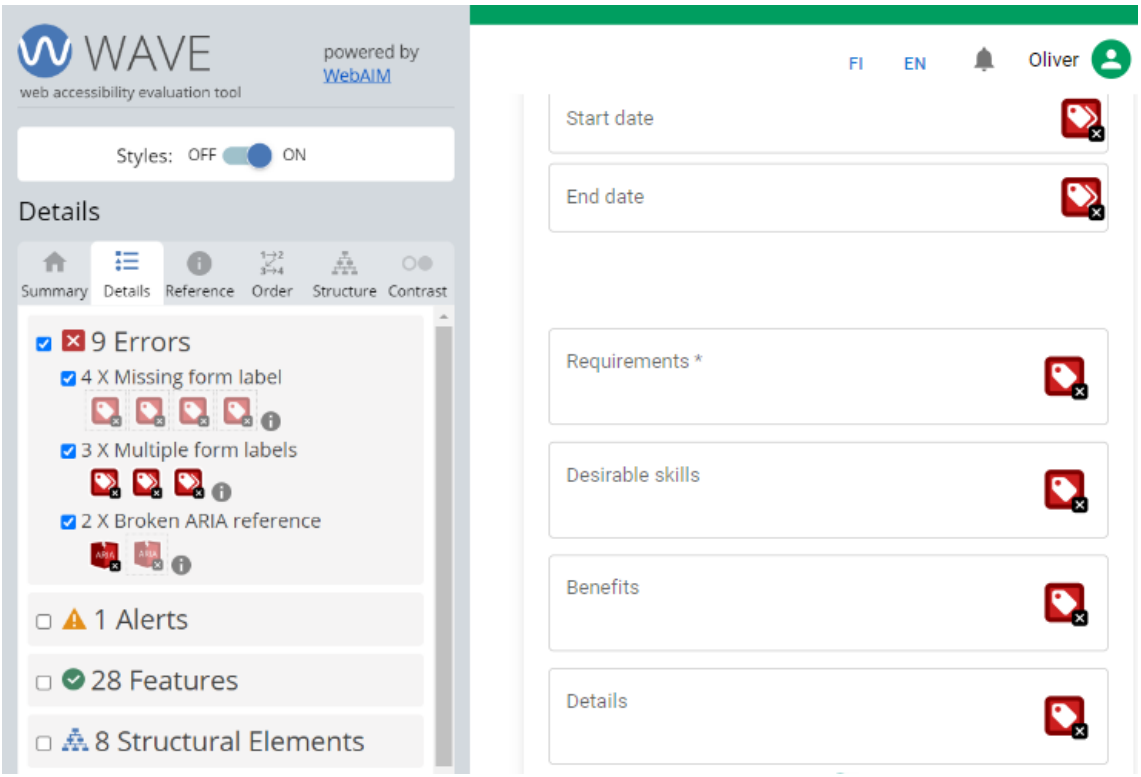
Kuva 16. Tota11y näyttää Keikkakaverin kotisivun otsikoiden järjestyksen.

Sivustolla ei ollut oikein otettu huomioon kontrastia saavutettavuuden kannalta, mutta sivujen kontrastivirheet oli suhteellisen helppo korjata. Kontrastia tarkasteltiin enimmäkseen WAVE-työkalulla. Keikkakaverin sivujen väriteemaa ei lähdetty muuttamaan, vaan vain tummennettiin monien sivujen värejä. Kontrasti pyrittiin saamaan ainakin AA-tason läpi. Kuvassa 17 näkyy WAVE tarkistamassa kontrastia Keikkakaverin kotisivulla. Kontrastivirheitä esiintyi enimmäkseen napeissa, joissa käytettiin oranssia taustaa. Muuten sivuston tausta on enimmäkseen valkoinen, minkä takia kontrastivirheitä ei oikein tullut.



Kuva 17. WAVE tarkistaa, pääseekö Keikkakaveri-sivuston kontrasti AA- ja AAA-tason läpi.

Sivustolla oli myös paljon label-elementti- ja ARIA-virheitä. Sivustolla oli kuitenkin käytetty MUI-kirjastoa, joka toi automaattisesti saavutettavuutta ARIAn ja label-elementtien kannalta. Enimmäkseen ongelmia label-elementtien kanssa tuli siitä, että niitä ei ollut tai ne oli jätetty tyhjiksi. Kuvassa 18 näkyy label-elementtien virheitä, joita WAVE löysi. ARIAsta tuli myös jonkin verran virheitä, eikä ollut täysin varmaa, oliko ARIAt tehty oikein. Yleensä ARIAn kanssa on parempi jättää se pois kokonaan kuin lisätä virheellistä ARIAa. Virheellistä ARIAa on vaikeampi löytää, kun tarkistusohjelmat yleensä vain tarkistavat, onko jotain tekstiä ARIAssa, eikä sitä, onko se juuri oikeaa siihen kohtaan. Suurimmasta osasta ARIA-virheistä ja joitain label-virheitä ei yritetty korjata vaan vain kirjattiin ne muistiin.



Kuva 18. WAVE näyttää label-elementtivirheitä sivulla.

Vaikeimmat saavutettavuuden virheet kirjattiin ja ilmoitettiin hankkeelle projektin loppuosassa. Vaikeimpiin virheisiin listattiin myös virheet, joissa sivujen ulkonäköä olisi pitänyt alkaa vaihtaa. Seuraavassa on esimerkkejä vaikeimmista saavutettavuusvirheistä, joita ohjelmilta saatiin:

- Varmista, että maamerkit ovat ainutlaatuisia.
- Dokumentissa tulee olla yksi päämaamerkki.
- Dokumentissa saa olla enintään yksi bannerin maamerkki.
- Vieritettävällä alueella on oltava pääsy näppäimistöön.
- - ja -elementit saavat sisältää suoraan vain -elementtejä.
- Sivun tulee sisältää ensimmäisen tason otsikko (<h1>).

Näihin virheisiin ei projektin aikana ehditty tutustua tai ei teknisistä syistä pystytty korjaamaan, minkä takia ne jäivät tekemättä.

5.3 Responsiivisuuden tarkistuksen automatisointi

Tulevaisuudessa olisi hyvä aina ottaa saavutettavuus ja responsiivisuus huomioon heti projektien alussa. Ne pitäisi myös ottaa huomioon projektin jokaisessa vaiheessa, jotta projektin lopussa ei tule yllätyksiä. Tässä projektissa yritettiin lisätä responsiivisuutta vasta projektin loppuvaiheessa, ja se ei välttämättä mennyt hyvin.

Responsiivisuuden testaus on tärkeä ja paljon aikaa vievä osa responsiivisuuden suunnittelussa, varsinkin jos responsiivisuutta ollaan lisäämässä sivustolle, jossa sitä ei ole vielä ollut. Tämä on kyllä nykyään harvinaista, koska monet komponenttikirjastot ottavat responsiivisuuden hyvin huomioon. Responsiivisuutta kannattaisi tarkistaa melkein jokaisen muutoksen jälkeen, samoin saavutettavuutta.

Automatisoimalla responsiivisuuden tarkistus ei tarvitsisi itse tarkistaa sivuston jokaista kohtaa läpi, vaan voisi vain jättää ohjelman tarkistamaan vaikka yön yli ja seuraava päivänä katsoa, mistä on löytynyt vikoja. Automaattiselle responsiivisuuden tarkastajalle tarvitsee vain tehdä skripti tai ohjeet ohjelmalle, joita automaatio-ohjelma seuraa. Automaatio on hyvä varsinkin, jos koko ajan tarvitsee tarkistaa tiettyjä osia sivustossa. Tällöin tarvitsee tehdä vain kerran ohjeet tarkistusohjelmalle ja sen jälkeen tarvitsee vain tehdä pieniä muutoksia ohjeisiin. Automaatio taas ei välttämättä ole paras vaihtoehto, jos koko ajan pitää tarkistaa jotain uutta osaa sivustosta, jota ei ole ennen ollut. Tämä kuitenkin vaihtelee riippuen siitä, kuinka iso tämä uusi osa on ja mitä tarkistusohjelmaa käyttää. Jotkin tarkistusohjelmat tarvitsevat vain sivun url:n, minkä jälkeen ne katsovat, mitä ne löytävät sivulta. Url:n lisäksi voi myös antaa tarkempia ohjeita, mitä tarkistusohjelma voi testata tai tarkistaa sivulla. Vaikka tarkistusohjelma ei löytäisi kaikkia vikoja sivulla, se kuitenkin vähentää työtä, jota pitää testauksessa tehdä.

Insinöörityöprojektissa ei käytetty automaattisia responsiivisuudentarkistusohjelmia, vaan vasta projektin jälkeen tutustuttiin niihin. Nämä tarkistusohjelmat ovat

kuitenkin hyvin yleisiä isommissa hankkeissa, ja niistä tutustuttiin kahteen, Seleniumiin ja Cypressiin.

Seleniumilla on kolme eri automaatio-ohjelmaa: WebDriver, IDE ja Grid. WebDriver ja Grid on tarkoitettu raskaampaan testaukseen. Gridiä käytetään rinnakkaisten testien suorittamiseen useilla koneilla, ja WebDriver täytyy asentaa projektiin ja siinä skriptit täytyy kirjoittaa itse. Näistä syistä valittiin Seleniumin IDE-ohjelma. Selenium IDE on tallennuksen ja toiston testausautomaatio-ohjelma, joka toimii verkkoselaimen lisäosana Firefoxiin, Chromeen tai Edgeen [Selenium automates browsers: 20]. IDE:n avulla pystyy tekemään nopeita virheidentoistoskriptejä tai sitä voi käyttää auttamaan virheiden etsinnässä verkkosivulla. IDE:n käytössä ei edes tarvinnut osata koodata, vaan skriptin pystyi tekemään vain laittamalla IDE:n nauhoittamaan, mitä itse teki sivulla, minkä jälkeen se pystyi toistamaan nämä toiminnot. IDE:ssä ei vain ollut paljon komentoja, ja esimerkiksi kuvien ottamiseen tarvitsi ladata lisäosa. IDE:stä ei löytynyt työkaluja saavutettavuuden testausta varten, joten ei tullut varmuutta siitä, voiko sitä tehdä IDE:tä käyttäen.

Cypress on aika samantyylinen kuin Seleniumin WebDriver: näitä kahta myös usein vertaillaan toisiinsa [Test. Automate. Accelerate: 21]. Cypress kuten WebDriver pitää asentaa projektiin, ja siihen pitää myös itse kirjoittaa skriptit. Cypressille pystyi luomaan skriptin, joka laittoi sen käymään Keikkakaveri-sivuston sivut läpi vaihtaen näytön kokoa jokaisella sivulla ja ottaen kuvan koko sivusta jokaisella eri valitulla näytön koolla. Tämän avulla pystyi ottamaan kuvia automaattisesti jokaisesta sivuston sivusta eri näytön ko’oilla sen sijaan, että olisi itse tarvinnut tehdä tämä ja katsoa, esiintyikö virheitä sivulla. Cypressillä on myös kokeellinen Cypress Studio, joka mahdollistaa skriptien luomisen nauhoittamalla sen, mitä itse tekee sivulla. Tätä ei kuitenkaan ehditty kokeilla. Saavutettavuuden testaustyökaluja ei suoraan löytynyt Cypressistä, mutta kun siihen kirjoitetaan itse skriptit, sillä varmaankin pystyy jotenkin testaamaan saavutettavuutta.

6 Yhteenveto

Insinöörityön tavoitteena oli Keikkakaveri-sivuston responsiivisuuden ja saavutettavuuden lisääminen. Projekti meni suhteellisen hyvin: responsiivisuutta ja saavutettavuutta saatiin parannettua sivustolla ja tehtiin pari muuta pientä lisäystä, jotka pyydettiin lisäämään. Alun perin saavutettavuuden ja responsiivisuuden parantamisen ja lisäämisen oli tarkoitus alkaa vasta projektin loppuosassa, mutta aikataulu muuttui yllättäen. Lähdettiin sitten parantamaan saavutettavuutta ja responsiivisuutta Keikkakaveri-sivustolle, kun sivusto ei ollut vielä kokonaan valmis. Jälkeenpäin katsottuna tämä ei ollut parasta Keikkakaverin kannalta, kun sivustolle oli tulossa vielä uutta sisältöä, rakenteen päivityksiä ja uusi ryhmä oli vielä tulossa tekemään sivustoa. Paljon kuitenkin saatiin tehtyä, ja toivottavasti työstä ja muistiin kirjatuista virheistä oli hyötyä projektille ja myöhemmille ryhmille.

Responsiivisuutta saatiin parannettua sivustolla varsinkin sivuston etusivulla, joka ei aluksi mahtunut mobiililaitteen tai tabletin näytölle. Käyttäjäsivujen taulukot saatiin myös tehtyä responsiivisemmiksi, muuten sivustolla ei ollut paljon responsiivisuusongelmia. Saavutettavuutta saatiin parannettua sivustolla suhteellisen hyvin: kaikkea ei saatu korjattua mutta hyvä osa kuitenkin. Saavutettavuuden tarkistusohjelmat löysivät paljon kaikenlaisia ongelmia, joista suurin osa saatiin korjattua. Loput ongelmista, jotka sivustolta löytyivät, merkittiin muistiin hankkeelle, jotta joku voi myöhemmin korjata ne käymättä koko sivustoa uudelleen läpi.

Hieman epävarmaksi kuitenkin jäi, mitä kaikkea olisi pitänyt saavuttaa tai yrittää saavuttaa projektin aikana. Tästä syystä ei yritetty tehdä suuria muutoksia sivuille vaan vain korjata virheitä, jotka oli helppo korjata tai jotka eivät tarvinneet isoja muutoksia sivuilla. Saavutettavuus ei myöskään ollut kovin tuttu alue projektin alussa, mutta ajan myötä siihen ehdittiin enemmän ja opittiin paljon uutta saavutettavuudesta. Responsiivisuudesta opittiin myös lisää, vaikka se oli jo tutumpi aihe.

Lähteet

Axe accessibility testing tools are the best on the planet. Verkkoaineisto. Deque. <https://www.deque.com/axe/>. Luettu 7.2022.

Bianchi, Tiago. 2023. Share of global mobile traffic 2015–2022. Verkkoaineisto. Statista. <https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/>. Luettu 9.2023.

CSS Units. Verkkoaineisto. W3Schools. <https://www.w3schools.com/cssref/css_units.php>. Luettu 6.2022.

Contrast Checker. Verkkoaineisto. Webaim. <https://webaim.org/resources/contrastchecker/>. Luettu 6.2022.

Deshpande, Chinmayee. 2023. The best guide to know what is react. Verkkoaineisto. Simplilearn. <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs>. Päivitetty 7.2.2023. Luettu 6.2022.

Digipalvelulain vaatimukset. Verkkoaineisto. Aluehallintovirasto. <https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/>. Luettu 6.2022.

Duckmanton, Daryl. 2019. Why UX and UI should remain separate. Verkkoaineisto. Uxdesign. <https://uxdesign.cc/why-ux-and-ui-should-remain-separate-7d6e3addb46f>. Luettu 9.2023.

For the next generation of intelligent applications. Verkkoaineisto. MongoDB. <https://www.mongodb.com/>. Luettu 7.2022.

Friedman, Vitaly. 2018. Responsive Web Design: What is it and how to use it. Verkkoaineisto. Smashingmagazine <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>. Luettu 6.2022.

Herbert, David. 2022. What is React.js? (Uses, Examples & More). Verkkoaineisto. Hubspot. <https://blog.hubspot.com/website/react-js>. Luettu 6.2022.

Headings. 2017. Verkkoaineisto. W3. <https://www.w3.org/WAI/tutorials/page-structure/headings/>. Päivitetty 4.5.2017. Luettu 4.2023.

HTML Responsive Web Design. Verkkoaineisto. W3Schools. <https://www.w3schools.com/html/html_responsive.asp>. Luettu 6.2022.

Kestävä Keikkatyö -hanke. Verkkoaineisto. Metropolia Ammattikorkeakoulu. <https://www.metropolia.fi/fi/tutkimus-kehitys-ja-innovaatiot/hankkeet/kestava-keikkatyo>. Luettu 6.2022.

Kestävä Keikkatyö Posteri. Verkkoaineisto. Tampereen korkeakouluyhteisö. <https://projects.tuni.fi/uploads/2022/09/147556fe-keke-posteri.pdf>. Luettu 9.2023.

Move faster with intuitive React UI tools. Verkkoaineisto. MUI. <https://mui.com/>. Luettu 7.2022.

Patadiya, Jaydeep. 2022. React JS – is it really a charismatic frontend maker. Verkkoaineisto. Radixweb. <https://radixweb.com/blog/react-js-for-frontend-development-features-benefits>. Päivitetty 25.8.2022. Luettu 6.2022.

React. Verkkoaineisto. Reactjs. <https://legacy.reactjs.org/>. Luettu 6.2022.

Saavutettavuus. Verkkoaineisto. Invalidiliitto. <https://www.invalidiliitto.fi/esteettomyys/saavutettavuus>. Luettu 6.2022.

Scales, Jordan. 2015. Tota11y – an accessibility visualization toolkit. Verkkoaineisto. Khanacademy. <https://blog.khanacademy.org/tota11y-an-accessibility-visualization-toolkit/>. Luettu 7.2022.

Selenium automates browsers. Verkkoaineisto. Selenium. <https://www.selenium.dev/>. Luettu 6.2023.

Test, Automate, Accelerate. Verkkoaineisto. Cypress. <https://www.cypress.io/>. Luettu 6.2023.

The WebAIM million. 2023. Verkkoaineisto. Webaim. <https://webaim.org/projects/million/>. Päivitetty 29.3.2023. Luettu 6.2022.

Tietoa WCAG-ohjeistuksesta. Verkkoaineisto. Saavutettavuusvaatimukset. <https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/>. Luettu 6.2022.

UX- ja UI-design – mitä sinunkin tulisi tietää web design -jargonista. 2019. Verkkoaineisto. Sininen härkä. <https://sininenharka.fi/ux-ja-ui-design-mita-sinunkin-tulisi-tietaa-web-design-jargonista/>. Luettu 8.2022.

Verkkosisältöjen saavutettavuus. Verkkoaineisto. Saavutettavasti. <https://www.saavutettavasti.fi/verkkosisaltojen-saavutettavuus/>. Luettu 6.2022.

Wave Web Accessibility Tools. Verkkoaineisto. Wave. <https://wave.webaim.org>. Luettu 6.2022.

WCAG. Verkkoaineisto. Saavutettavasti. <https://www.saavutettavasti.fi/verkko-sisaltojen-saavutettavuus/wcag/>. Luettu 6.2022.

Web Content Accessibility Guidelines (WCAG) 2.1. Verkkoaineisto. W3. <https://www.w3.org/Translations/WCAG21-fi/>. Luettu 10.2023.

What is GitHub? A beginners introduction to GitHub. 2022. Verkkoaineisto. Kinsta. <https://kinsta.com/knowledgebase/what-is-github/>. Päivitetty 13.12.2022. Luettu 6.2022.

Yleistä saavutettavuudesta. Verkkoaineisto. Saavutettavuusvaatimukset. <https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/>. Luettu 6.2022.