Saavutettavuus verkkosivustoilla

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

kevät 2022

Jenni Lehtonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Jenni Lehtonen Vuosi 2022
Työn nimi Saavutettavuus verkkosivustoilla
Ohjaaja Lasse Seppänen

Internetin käyttö lisääntyy päivä päivältä, minkä vuoksi saavutettavuus on erittäin tärkeää. Tämän opinnäytetyön tarkoituksena oli tutkia saavutettavuutta sisällöntuotannon ja ohjelmistokehityksen näkökulmasta. Opinnäytetyö tarjoaa tietoa myös siitä, miten saavutettavuus otetaan huomioon ohjelmoitaessa ReactJS:llä.

Opinnäytetyö on jaettu saavutettavuutta käsittelevään teoriaosuuteen ja verkkosivustoprojektiin. Opinnäytetyön aikana tehdyssä projektissa kehitettiin yksinkertainen verkkosivusto, jonka avulla havainnollistettiin, miten verkkosivustoista käytännössä tehdään saavutettavia. Kehitetylle verkkosivustolle suoritetaan myös saavutettavuustestausta. Opinnäytetyö on toiminnallinen.

Teoriaosuudessa määritellään, mitä saavutettavuus tarkoittaa ja miksi se on tärkeää. Tämän jälkeen tutkitaan saavutettavuuteen liittyvää lainsäädäntöä ja WCAG-ohjeistusta. Seuraavaksi teoriaosuudessa siirrytään tutkimaan verkkosivustojen kehitystä ja katsotaan, mitä asioita täytyy saavutettavuuden kannalta ottaa huomioon. Opinnäytetyön teoriaosuus on kirjoitettu saavutettavuuteen liittyvän kirjallisuuden ja internetistä löytyvän materiaalin pohjalta. Opinnäytetyön käytännönosassa on myös hyödynnetty omaa tietoa.

Opinnäytetyössä havaittiin, että saavutettavuus on erittäin laaja aihe ja sen huomioiminen verkkosivustoilla kiitettävällä tasolla vaatii monialaisen kehitystiimin. Opinnäytetyön aikana syntyi pohdintaa siitä, huomioidaanko saavutettavuutta kuitenkaan tarpeeksi tällä hetkellä, sillä internetissä on useita verkkosivustoja, joilla on saavutettavuusongelmia.

Avainsanat Saavutettavuus, WCAG, ReactJS, web-kehitys, saavutettavuustestaus, verkkosivusto
Sivut 53 sivua ja liitteitä 2 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Author Jenni Lehtonen
Subject Accessibility on websites
Supervisors Lasse Seppänen

Abstract
Year 2022

As Internet is being used more and more in our day to day lives, accessibility is very important. The purpose of this thesis was to investigate accessibility on the websites from the point of view of content production and software development. The thesis also provides information on how to take accessibility into account in programming with ReactJS.

The thesis is divided into two sections. First, the central concepts related to accessibility are explained. The thesis proceeds by discussing the development project. A simple website was built in the development project during the thesis and the purpose of it was to demonstrate how to make a website accessible. The accessibility of the website was also tested. This thesis is practical.

In the theory section, the meaning of accessibility and its importance are explained. After that, the law regarding the accessibility and Web Content Accessibility Guidelines are examined. The theory section proceeds by investigating web development and things to consider regarding the accessibility on the websites. The theory is based on the literature of accessibility and the materials found on the internet. The practical section of the thesis is also based on the writer's own knowledge.

The conclusion was that accessibility is a very wide subject and in order to a website to be accessible, the development team must have professionals coming from different fields. The thesis process revealed multiple issues related to accessibility on websites. This leads to the question whether the accessibility issues are not being solved properly in current website development.

Keywords Accessibility, WCAG, ReactJS, web development, accessibility testing, website

Pages 53 pages and appendices 2 pages

Sanasto

DOM	Document Object Model. Kuvaa verkkosivuston rakenteen puuna.
CSS	Cascading Style Sheets. CSS on tekniikka, jonka avulla määritellään verkkosivuston tyylejä.
HTML	HyperText Markup Language. Verkkosivustoilla käytetty merkintäkieli, jonka avulla rakennetaan verkkosivuston runko.
JavaScript	JavaScript on ohjelmointikieli, jolla ohjelmoidaan toimintoja verkkosivustoille.
JSX	JavaScript XML. JSX-koodia käytetään ReactJS:llä rakennetuilla verkkosivustoilla.
Kirjasto	Kokoelma valmiita koodeja, jotka sisältävät hyödyllisiä funktioita erilaisten toimintojen suorittamiseen.
Ohjelmistokehys	Koodiperusta, jonka päälle voi rakentaa verkkosivuston.
ReactJS	JavaScript-kirjasto, jolla voi rakentaa verkkosivustoja.
Responsiivisuus	Responsiivinen verkkosivusto mukautuu käyttäjän päätelaitteelle sopivaksi.
Syntaksi	Tapa kirjoittaa koodia jollakin ohjelmointikielellä.
Tapahtumakäsittelijä	Käynnistää jonkin toiminnon esimerkiksi hiiren klikkauksen perusteella.
WAI-ARIA	Web Accessibility Initiative – Accessible Rich Internet Applications Suite. WAI-ARIA on W3C:n kirjoittama spesifikaatio, jonka avulla verkkosivuston elementeistä voi tehdä saavutettavia.
WCAG	Web Content Accessibility Guidelines. Verkkosivustojen saavutettavuusohjeistus.
Yarn	Paketinhallintajärjestelmä, jonka avulla projektiin voi asentaa erilaisia riippuvuuksia, esimerkiksi React Routerin reititystä varten.

Sisällys

Kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Saavutettavuus on erittäin tärkeä ja ajankohtainen asia, sillä ihmiset ympäri maailmaa käyttävät internetiä joka päivä joko töissä tai vapaa-ajalla. Yhteiskuntamme on muuttumassa monelta osin digitaaliseksi ja monia palveluita on vaikea käyttää, jos verkkosivustoja ei ole kehitetty saavutettavuus huomioon ottaen. Selovuon Saavutettavuusoppaan (2019, s. 15) mukaan saavutettavalla sisällöllä voi auttaa 1,5–2,5 miljoonaa ihmistä pelkästään Suomessa. Koko maailman mittakaavassa saavutettavuudesta hyötyviä henkilöitä on moninkertainen määrä. Tämän vuoksi on tärkeää, että kaikilla on tasavertainen mahdollisuus käyttää internetiä ja selata verkkosivustoja. Saavutettavuuden avulla edistetään ihmisten välistä tasa-arvoa ja yhdenvertaisuutta.

Tässä opinnäytetyössä tutkitaan saavutettavuutta kokonaisuutena ja tavoitteena on luoda lukijalle hyvä yleiskäsitys saaavutettavuuteen liittyvistä asioista. Teoriaosuudessa lukijalle selitetään, mitä saavutettavuus tarkoittaa ja ketkä siitä hyötyvät. Tämän lisäksi tutustutaan saavutettavuuteen liittyvään lainsäädäntöön sekä WCAG-ohjeistukseen, jonka noudattamista laki vaatii. Teoriaosuuden lopussa tutkitaan, mitä verkkosivustojen kehityksessä täytyy huomioida sisällöntuotannon ja ohjelmistokehityksen näkökulmista sekä miten saavutettavuutta voidaan testata.

Käytännön osuudessa tehdyn verkkosivustoprojektin avulla pohditaan saavutettavuutta ohjelmistokehityksen näkökulmasta: Kuinka helppoa on täyttää kaikki saavutettavuuslain vaatimukset ja kuinka paljon aikaa ominaisuuksien toteuttaminen saavutettavasti vie? Projektin aikana kehitetyn verkkosivuston saavutettavuutta myös testataan.

Tässä opinnäytetyössä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

- Mitä saavutettavuus tarkoittaa ja miksi se on tärkeää?
- Mitä vaatimuksia saavutettavalla verkkosivustolla on?
- Miten ReactJS:llä voi rakentaa saavutettavan verkkosivuston?
- Miten verkkosivuston saavutettavuutta testataan?

2 Saavutettavuus

Tässä luvussa tutustutaan ensin saavutettavuuteen käsitteenä. Sen jälkeen tutkitaan, minkälaisia toimintarajoitteita käyttäjillä saattaa olla. Toimintarajoitteiden kohdalla tutkitaan myös, minkälaisia avustavia teknologioita käyttäjillä on apunaan.

2.1 Saavutettavuuden merkitys

Saavutettavuus on digitaalisen julkaisun ominaisuus. Digitaalinen julkaisu voi olla esimerkiksi verkkosivusto, PowerPoint-esitys tai PDF-tiedosto. Kun julkaisu on kehitetty saavutettavuus huomioon ottaen, kaikki pystyvät käyttämään sitä ja ymmärtämään sen sisällön riippumatta siitä, onko heillä esimerkiksi huono näkö tai jokin sairaus, jonka vuoksi käyttäminen voisi muutoin olla vaikeaa tai jopa mahdotonta. Saavutettavuuteen vaikuttaa moni eri asia, esimerkiksi taustan ja tekstin välinen kontrasti, sisällön esittämistapa sekä sisällön kieli. (Selovuo, 2019, s. 13–14)

Saavutettavuus on tärkeää ja se vaikuttaa kaikkien ihmisten elämään. Monet palvelut ovat siirtyneet digitaalisiksi, minkä vuoksi on tärkeää tehdä niistä saavutettavia, jotta ne olisivat kaikkien käytettävissä. Timo Övermarkin mukaan ei ainoastaan riitä, että palveluissa jokin asia on mahdollista suorittaa, vaan palvelut täytyy kehittää siten, että niiden käyttäminen on miellyttävää, tuloksellista ja tehokasta. Övermarkin mukaan ”palvelun täytyy tukea käyttäjää siinä, että tehtävä tulee suoritettua mahdollisimman hyvin, nopeasti ja helposti ilman ylimääräistä älyllistä ponnistelua tai harmaita hiuksia.” (Etelä-Suomen aluehallintovirasto, n.d. -a; Leskelä, 2019, s. 67)

WHO:n mukaan maailmassa on yli miljardi ihmistä, joilla on jokin vamma. Saavutettavuudesta hyötyy siis erittäin suuri määrä ihmisiä. On kuitenkin tärkeä ymmärtää, että saavutettavuudesta hyötyvät lopulta kaikki, eivätkä vain henkilöt, joilla on jokin vamma. (Etelä-Suomen aluehallintovirasto, n.d. -a; WHO, 2021)

2.2 Saavutettavuudesta hyötyvät henkilöt

Vaikka saavutettavuudesta hyötyvät kaikki, tietyt henkilöt kuitenkin hyötyvät siitä paljon enemmän kuin toiset. Ilman saavutettavuutta, erilaisten verkkopalvelujen käyttö voi olla heille erittäin vaikeaa. Verkkosivustojen käyttämiseen vaikuttavat esimerkiksi erilaiset näköön, kuuloon ja motoriikkaan liittyvät tekijät. Tämän lisäksi käyttökokemukseen voi vaikuttaa erilaiset käyttötilanteet, kuten auringonpaiste tai meluisa ympäristö. (Etelä-Suomen aluehallintovirasto, n.d. -a)

2.2.1 Näköön liittyvät rajoitteet

Näkövammaisten liiton mukaan Suomessa on noin 55 000 näkövammaista, joista suurin osa on 60 vuotta täyttäneitä. Lukumäärästä noin 74 % on heikkonäköisiä ja 22 % sokeita. Neljällä prosentilla näkövammaisista näkövammaisuuden astetta ei ole määritelty. Yleisimpiä silmäsairauksia Suomessa ovat silmänpohjan rappeuma, verkkokalvon perinnölliset rappeumat ja näköratojen viat. (Näkövammaisten liitto ry, 2021b) Myös värisokeus on yksi näköön liittyvä rajoite, joka tulisi ottaa huomioon verkkosivustoja kehitettäessä (Kehitysvammaliitto ry, n.d. -a).

Näkövammaiset henkilöt tarvitsevat usein erilaisia avustavia teknologioita verkkosivustojen selaamiseen. Näitä voivat olla esimerkiksi ruudunlukuohjelma, ruudunsuurennusohjelma ja pistenäyttö, joita voi käyttää apunaan sekä tietokoneella että mobiililaitteilla. Kuvan 1 pistenäytön avulla käyttäjä voi lukea ruudulla näkyvää tekstiä pistekirjoituksena. Joissakin pistenäytöissä on myös pistekirjoitusnäppäimistö kirjoittamista varten. Muita avustavia keinoja ovat esimerkiksi verkkosivuston kontrastien muuttaminen sekä kirjainkoon vaihtaminen. (Näkövammaisten liitto ry, 2021a; ks. myös Kehitysvammaliitto ry, n.d. -a)

Kuva 1 Focus 40 Blue 5th Generation (Pavlíček, 2017)

2.2.2 Kuuloon liittyvät rajoitteet

Suomessa on yli 800 000 henkilöä, joilla on jonkinasteinen kuulon alenema. Kuuloviat voivat johtua muun muassa seuraavista syistä: korvakäytävä on epämuodostunut, tärykalvon ja kuuloluiden toiminta on rajoittunut tai vika keskushermostossa. Kuulovammaiset henkilöt voivat käyttää apunaan esimerkiksi kuulokojetta. Kuulokojeen käyttäminen ei kuitenkaan tarkoita sitä, että käyttäjä olisi normaalikuuloinen. Muita mahdollisia apuvälineitä ovat esimerkiksi sisäkorvaistute ja kommunikaattori. (Kuuloliitto ry, n.d. -a, -b, -c)

Kuulovamman vuoksi verkkosivustoilla olevat videot ja äänitiedostot voivatkin olla vaikeita tai mahdottomia käyttää. Tämän vuoksi esimerkiksi videoihin pitäisi laittaa tekstitykset tai tarjota jokin muu esitystapa tiedon välittämiseksi. Tiedonvälitykseen voi myös käyttää viittomakieltä, joka saattaa olla kuulovammaisen henkilön äidinkieli. (Kehitysvammaliitto ry, n.d. -b)

2.2.3 Fyysiset ja motoriset rajoitteet

Fyysisiä ja motorisia rajoitteita voivat olla esimerkiksi lihaksiston rajoitteet ja heikkoudet, kuten vapina tai halvaantuminen. Rajoitteita voivat olla myös tuntoaistin rajoitteet, vajaat liikeradat sekä puuttuvat raajat. Fyysiset ja motoriset rajoitteet eivät kokonaan estä tietokoneen käyttöä, mutta voivat hankaloittaa sitä ja tehdä siitä hitaampaa. (Kehitysvammaliitto ry, n.d. -c; Selovuo, 2019, s. 114)

Käyttäjät, joilla on fyysisiä tai motorisia rajoitteita, voivat selata internetiä erilaisten apuvälineiden avulla. Esimerkiksi henkilöt, joilla on käsien pakkoliikkeitä tai vapinaa, voivat käyttää kuvassa 2 näkyvää pallohiirtä apunaan. Muita apuvälineitä ovat erilaiset kytkinohjaimet (kuvassa 3) sekä katseohjaus (kuvassa 4). (Kehitysvammaliitto ry, n.d. -c; Kehitysvammaliitto, n.d. -d)

Kuva 2 Kensington SlimBlade Trackball (Wang, 2016)

Jos verkkosivusto on kehitetty siten, että sitä voi selata näppäimistön avulla, silloin sitä voi selata myös kytkimiä käyttämällä. Kytkinohjaimia on erilaisia: jotkin kytkimet kestävät kovempia painalluksia ja toisiin riittää pelkkä hipaisukosketus. Kytkimiä voi myös käyttää monin eri tavoin, kuten kädellä, jaloilla, päällä tai silmiä räpäyttämällä, jos kytkin on asetettu silmälaseihin. (Kehitysvammaliitto ry, 2021a)

Kuva 3 Erilaiset kytkinohjaimet (Kehitysvammaliitto ry, n.d. -n)

Katseohjauksessa tietokonetta voi käyttää katseen avulla, eikä käyttäjä siis tarvitse hiirtä tai näppäimistöä lainkaan. Katseohjauksen avulla verkkosivustoilla olevia kohteita voi valita

esimerkiksi pitämällä katse tietyssä kohdassa pidempään. Apuna on myös mahdollista käyttää kytkimiä, jolloin hiiren voi esimerkiksi ohjata katseella painikkeen päälle ja käyttää kytkintä painikkeen klikkaamiseen. Kuvassa 4 olevan katseohjausyksikön lisäksi on olemassa katseohjaustietokoneita. (Kehitysvammaliitto ry, 2021b; Kehitysvammaliitto n.d. -e)

Kuva 4 Katseohjausyksikkö (Kehitysvammaliitto ry, n.d. -o)

2.2.4 Kognitiiviset ja kielelliset rajoitteet

Kognitiiviset ja kielelliset rajoitteet liittyvät muun muassa ihmisen muistiin, ymmärtämiseen ja tarkkaavaisuuteen. Verkkosivustoja kehitettäessä kognitiiviset rajoitteet täytyy ottaa huomioon sisällössä, jota tulisi tarjota eri muodoissa ja mahdollisimman selkeästi. Tekstisisällössä tulisi esimerkiksi välttää sanontoja ja vaikeita ilmaisuja ja tekstiä voi myös tukea esimerkiksi kuvien avulla. Sisällön lisäksi kognitiiviset rajoitteet täytyy huomioida sivuston ulkoasussa. Käyttäjää ei saisi häiritä tarpeettomasti esimerkiksi liikkuvilla kuvilla. Verkkosivuston ulkoasun täytyy myös olla selkeä, esimerkiksi tekstit pitää tasata oikein sekä välttää alleviivauksia ja kursiivia. Verkkosivuston merkittävin sisältö olisi myös hyvä korostaa visuaalisesti. (Selovuo, 2019, s. 116; ks. myös Muistiliitto ry, n.d.)

Kielelliset vaikeudet voivat johtua esimerkiksi sairaudesta tai vieraskielisyydestä, mutta myös esimerkiksi koulutustaso vaikuttaa ymmärtämiseen. Arvioiden mukaan Suomessa selkokieltä tarvitsee noin 500 000 henkilöä ja selkeän kielen käytöstä hyötyy vielä useampi. Saavutettavuutta voidaan parantaa, kun verkkosivuston sisältö tuotetaan selkeällä kielellä.

Sisällössä tulisi käyttää siis vain yleiskielisiä sanoja ja jos erikoistermejä tarvitaan, niiden tarkoitus täytyy selittää. (Selovuo, 2019, s. 118)

2.2.5 Muut rajoitteet

Vaikka verkkosivuston käyttäjällä ei olisi mitään vammaa tai sairautta, joka vaikeuttaisi verkkosivustojen käyttöä, on saavutettavuudesta silti hyötyä. Käyttäjillä saattaa olla erilaisia tilapäisiä tai tilanteeseen liittyviä toimintarajoitteita. Auringonpaiste on yksi tilanteeseen liittyvä toimintarajoite, jonka vuoksi tietokoneen tai mobiililaitteen näytöltä voi olla vaikea nähdä sisältöä. Myös esimerkiksi meluisa ympäristö, sylissä pidettävä vauva tai heiluva ja tärisevä kulkuneuvo ovat tilanteeseen liittyviä toimintarajoitteita. (Haanperä, n.d.; Kehitysvammaliitto ry, n.d. -f)

Tilapäisiä toimintarajoitteita ovat esimerkiksi loukkaantuminen, jonka vuoksi käsi saattaa olla kipsattu tai kotiin unohtuneet silmälasit, jonka vuoksi sisällön havainnointi voi olla vaikeaa. Joskus myös hidas tietoliikenneyhteys tai vanha päätelaite saattaa aiheuttaa ongelmia verkkosivustojen selaamisessa. On myös henkilöitä, jotka eivät käytä internetiä usein, jolloin vähäinen käyttökokemus voi olla rajoite. (Haanperä, n.d.; Kehitysvammaliitto ry, n.d. -f)

3 Lainsäädäntö

Verkkosivustojen saavutettavuudesta on säädetty digipalvelulailla, joka tuli voimaan Suomessa 1.4.2019. Tämän lain taustalla on vuonna 2016 voimaan tullut Euroopan unionin saavutettavuusdirektiivi sekä YK:n yleissopimus vammaisten henkilöiden oikeuksista. Digipalvelulaki velvoittaa viranomaisia, julkisoikeudellisia laitoksia, osaa järjestöistä, viranomaisten rahoituksella tuotettuja palveluita sekä osaa yksityisen sektorin toimijoista kehittämään verkkosivustonsa WCAG-ohjeistuksen A- ja AA-tasojen kriteerien mukaisesti. Tämä ohjeistus sisältää myös AAA-tason, mutta laki ei velvoita noudattamaan sitä. WCAG-ohjeistuksesta kerrotaan lisää seuraavassa luvussa. Lain piiriin kuuluvien toimijoiden verkkosivustot voivat olla kaikille avoimia tai vaatia sisään kirjautumista. Verkkosivuston kohderyhmä voi myös olla mikä tahansa: laki koskee sekä yleisiä kaikkien käyttämiä verkkosivustoja että viranomaisten työpaikalla käytettyjä intra- ja extranettejä. Digipalvelulaki velvoittaa myös tekemään mobiilisovelluksista saavutettavia. (Etelä-Suomen aluehallintovirasto, n.d. -b, -c, -d, -e; ks. myös Valtiovarainministeriö, n.d.)

Lain piiriin kuuluvien toimijoiden täytyy lain mukaan tehdä verkkosivustoistaan ja mobiilisovelluksistaan saavutettavuusseloste, jonka pitää olla käyttäjien nähtävissä. Saavutettavuusselosteen pakollinen sisältö on määritelty vuoden 2018 EU-komission täytäntöönpanopäätöksessä (EU, 2018/1523). Selosteessa on mainittava, miten hyvin verkkosivusto tai mobiilisovellus vastaa saavutettavuusvaatimuksia sekä miltä osin ja miksi joitain vaatimuksia ei ole pystytty täyttämään. Saavutettavuusselosteen täytyy myös sisältää tieto, milloin se on laadittu ja milloin mahdolliset päivitykset on tehty sekä perustuvatko sen tiedot itsearvioon vai ulkopuoliseen asiantuntijan arvioon. Näiden lisäksi selosteessa täytyy mainita, mitä kautta käyttäjien on mahdollista antaa saavutettavuuspalautetta ja kuka on vastuussa palautteiden käsittelystä. Selosteen täytyy myös sisältää tieto siitä, miten käyttäjä voi ottaa yhteyttä Aluehallintovirastoon selvityspyynnön tai kantelun tekemistä varten. (Etelä-Suomen aluehallintovirasto, n.d. -f; Komission täytäntöönpanopäätös (EU) 1523/2018)

Saavutettavuusvaatimuksista voidaan poiketa, jos vaatimusten toteuttaminen aiheuttaisi kohtuutonta rasitetta. Poikkeusta ei voida kuitenkaan tehdä, jos ollaan kehittämässä täysin uutta verkkosivustoa. Jo olemassa olevien verkkosivustojen kohdalla poikkeus on mahdollista tehdä vain tilapäisesti ja tietyiltä osin verkkosivustoa eli kohtuuton rasite ei voi

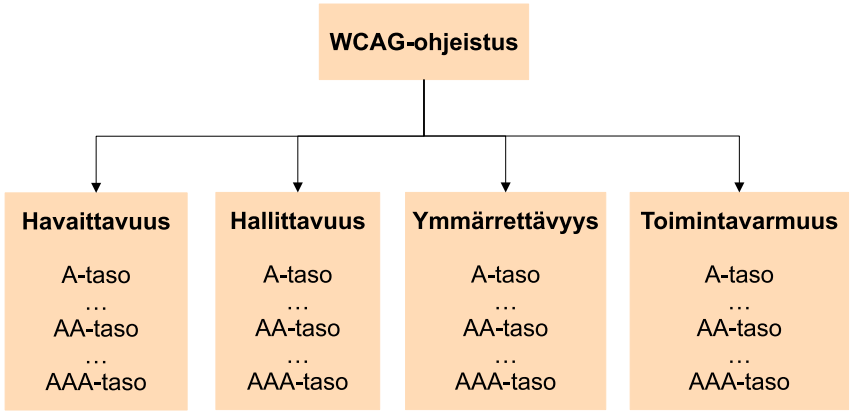
siis koskea koko verkkosivustoa. Kohtuutonta rasitetta arvioitaessa otetaan huomioon esimerkiksi vammaisten henkilöiden tarve käyttää palvelua, palveluntarjoajan koko ja taloudellinen asema. Jos palvelu on todella tärkeä ja laajasti käytetty, voi palveluntarjoajan olla vaikeampi vedota kohtuuttomaan rasitteeseen. (Etelä-Suomen aluehallintovirasto, n.d. -f)

4 WCAG-ohjeistus

WCAG-ohjeistus (Web Content Accessibility Guidelines) on Word Wide Web -konsortion eli W3C:n kehittämä ja ylläpitämä saavutettavuusohjeistus. Ensimmäisen kerran ohjeistus on julkaistu vuonna 1999 ja sitä päivitetään jatkuvasti tekniikan kehittyessä. Tällä hetkellä WC3 kehittää uutta versiota ohjeistuksesta, mutta suosittelee vielä käyttämään vuosina 2008 ja 2018 julkaistuja versioita 2.0 ja 2.1. W3C ohjeistaa suosituksestaan huolimatta myös käyttämään viimeisintä versiota 2.2, joka julkaistaan varsinaisesti kesällä 2022, jotta saavutettavuustavoitteiden sovellettavuus voidaan maksimoida tulevaisuudessa. Tässä opinnäytetyössä tutkitaan vuonna 2018 julkaistua versiota 2.1. (Celia, n.d. -a; ks. myös W3C, 2018)

WCAG-ohjeistuksen rakenne on esitetty kuvassa 5. Ohjeistus koostuu neljästä periaatteesta, jotka ovat havaittavuus, hallittavuus, ymmärrettävyys ja toimintavarmuus. Näiden periaatteiden alla olevat saavutettavuuskriteerit on jaettu A-, AA- ja AAA-tasoihin. Kuten edellisessä luvussa mainittiin, joidenkin verkkosivustojen täytyy vastata tasojen A ja AA kriteereitä. Taso AAA sisältää tiukimmat vaatimukset, joiden noudattamista laki ei vaadi. (Celia, n.d. -a)

Kuva 5 WCAG-ohjeistuksen rakenne



Vaikka WCAG-ohjeistusta noudattaisi lain vaatimalla tasolla, se ei kuitenkaan takaa sitä, että verkkosivusto olisi täysin saavutettava. Kriteerit on mahdollista täyttää monin eri tavoin ja aina lopputulos ei välttämättä ole saavutettava. Osa ohjeistuksen kriteereistä on tulkinnanvaraisia, joten kaikki eivät ole niiden toteutumisesta välttämättä samaa mieltä. On

myös tärkeää huomata, että WCAG-ohjeistus ei ota juuri kantaa verkkosivuston sisällön ymmärrettävyyteen tai käytettävyyteen, vaikka nämä ovat olennainen osa saavutettavuutta. (Etelä-Suomen aluehallintovirasto, n.d. -g)

Seuraavaksi tutkitaan, minkälaisia kriteerejä WCAG-ohjeistuksen neljässä periaatteessa on. Tarkoituksena on havainnollistaa, minkälaisia eri vaatimuksia verkkosivustojen kehitystyössä saattaa joutua huomioimaan. Esitellyt kriteerit ovat poimintoja WCAG-ohjeistuksesta, eivätkä siis kuulu mihinkään tiettyyn tasoon.

4.1 Havaittavuus

Periaatteen mukaan verkkosivuston sisällön täytyy olla käyttäjien havaittavissa. Tämän periaatteen alla on yhteensä 29 kriteeriä, jotka on jaettu neljään eri osioon: tekstivastineet, aikasidonnainen media, mukautettava ja erottuva. (W3C, n.d., 2018)

Tekstivastineet-osio sisältää kriteerin ei-tekstuaaliselle sisällölle. Kriteerin mukaan kaikki ei-tekstuaalinen sisältö pitää olla muutettavissa esimerkiksi pistekirjoitukseksi, puheeksi tai isokokoiseksi tekstiksi. Kriteeriin on kuitenkin olemassa muutama poikkeus, esimerkiksi jos sisältö on verkkosivustolla vain koristetarkoituksessa, se täytyy toteuttaa siten, että avustava teknologia voi jättää sen huomioimatta. (W3C, 2018)

Aikasidonnaiselle medialle – esimerkiksi videot ja äänitiedostot – on määritetty yhdeksän kriteeriä, joista kaksi koskee suoria lähetyksiä ja muut tallenteita. Kriteerien mukaan esimerkiksi verkkosivustolla esitettävissä videoissa täytyy olla tekstitys ja joissain tapauksissa myös kuvailutulkkaus tai viittomakielinen tulkkaus. Videoille ja äänitiedostoille täytyy myös olla vaihtoehtoinen esittämistapa, joka sisältää saman informaation. (W3C, 2018)

Mukautettava-osio sisältää kriteereitä sisällön esittämiseen eri tavoin ilman sisällön tai rakenteen muuttumista. Sisällön esittäminen eri tavoin viittaa käyttäjien käyttämiin avustaviin teknologioihin, esimerkiksi ruudunlukuohjelmaan tai käyttäjän tekemiin tyylimuutoksiin (Selovuo, 2019, s. 64). Osiossa on yhteensä kuusi kriteeriä, jotka liittyvät esimerkiksi käytettävän laitteen asentoon, syötekenttiin ja sisällön esitysjärjestykseen. (W3C, 2018)

Erottuva-osio sisältää kriteereitä sisällön näkemiseen ja kuulemiseen. Tässä osiossa on yhteensä 13 erilaista kriteeriä, jotka liittyvät esimerkiksi värien käyttöön, kontrastiin, tekstin kokoon ja äänien käyttöön. Värien käytön osalta on esimerkiksi määritetty, että niitä ei saa käyttää ainoana visuaalisena keinona informaation välittämiseen tai toiminnon esittämiseen. Selovuon Saavutettavuusopas (2019, s. 65) havainnollistaa kriteerin merkitystä: ”Ei esimerkiksi saa käskeä täyttämään vähintään punaisella merkityt lomakekentät, sillä kaikki käyttäjät eivät erota punaista väriä.” (W3C, 2018)

4.2 Hallittavuus

Periaatteen mukaan käyttöliittymän komponenttien ja navigaation täytyy olla hallittavia eli käyttöliittymä ei voi sisältää ominaisuuksia, jotka häiritsevät tai estävät käyttöä. Tämä periaate sisältää yhteensä 29 eri kriteeriä, jotka on jaettu viiteen eri osioon: käytettävissä näppäimistöltä, tarpeeksi aikaa, sairauskohtaukset ja fyysiset reaktiot, navigoitava ja syötetavat. (W3C, n.d., 2018; Selovuo, 2019, s. 25)

Käytettävissä näppäimistöltä -osio sisältää neljä kriteeriä näppäimistöön liittyen. Tämän osion kriteerien mukaan verkkosivuston täytyy olla käytettävissä pelkällä näppäimistöllä. Kriteereissä on myös mainittu näppäimistöansoista eli tilanteista, joissa käyttäjä pääsee näppäimistön avulla liikkumaan johonkin elementtiin, mutta ei pysty poistumaan siitä pelkkiä näppäinkomentoja käyttäen. (W3C, 2018; Selovuo, 2019, s. 70)

Tarpeeksi aikaa -osio sisältää kuusi kriteeriä. Tämän osion mukaan käyttäjälle on annettava tarpeeksi aikaa sisällön lukemiseen ja sen toimintojen käyttämiseen. Erilaisia aikarajoitettuja toimintoja esiintyy esimerkiksi verkkopankeissa ja verkkokaupoissa. Tämän osion kriteerit liittyvät muun muassa ajoituksiin, liikkuvan sisällön kontrollointiin ja taukoihin. Ajoituksien osalta on esimerkiksi määritetty, että käyttäjän täytyy voida kytkeä aikarajoitus pois päältä tai säätää aikarajoituksen määrää. (W3C, 2018; Selovuo, 2019, s. 74)

Sairauskohtaukset ja fyysiset reaktiot -osio sisältää kolme kriteeriä. Osion pääohjeena on, että sisältöä ei pitäisi suunnitella tavalla, jonka tiedetään aiheuttavan sairauskohtauksia. Esimerkiksi kirkkaat valot ja tiheään toistuvat välähdykset voivat aiheuttaa epileptisiä kohtauksia. Tämän osion kriteerien mukaan verkkosivustojen ei esimerkiksi pitäisi sisältää

mitään, joka välähtäisi useammin kuin kolme kertaa sekunnissa. Animaatioiden osalta on olemassa kriteeri, jonka mukaan käyttäjän vuorovaikutuksesta liikkuvat animaatiot voisi ottaa pois päältä, elleivät ne ole olennainen osa käytön tai informaation kannalta. (W3C, 2018; Selovuo, 2019, s. 76)

Navigoitava-osio sisältää kymmenen erilaista kriteeriä verkkosivuston yleiseen navigointiin eli se ei ota kantaa ainoastaan navigaatiopalkkiin vaan yleisesti sivustolla liikkumiseen. Kriteerit ottavat kantaa esimerkiksi linkkeihin, otsikoihin, elementtien kohdistusjärjestykseen ja elementtien kohdistukseen. Linkkien osalta on määritetty, että käyttäjän pitäisi ymmärtää linkkitekstin perusteella, mihin linkki vie. Linkkitekstin sisältö voisi esimerkiksi olla ”Lue lisää toimitusehdoistamme” eikä ainoastaan ”Lue lisää”. Elementtien kohdistuksesta on määritetty, että käyttäjän liikkuessa verkkosivustolla, sillä hetkellä aktiivinen elementti pitäisi erottua visuaalisesti, jotta käyttäjä tietää, missä kohdassa hän verkkosivustolla on. (W3C, 2018; Selovuo, 2019, s. 76–79)

Syötetavat-osio sisältää kuusi kriteeriä. Tämän osion pääohjeena on, että toimintojen käyttämisestä pitäisi tehdä helpompaa erilaisten syötetapojen kanssa. Kriteerit ottavat kantaa esimerkiksi osoitineleisiin, osoittimen toiminnon perumiseen ja liikkeen avulla tapahtuviin toimintoihin. Osoitineleiden osalta on määritetty, että kaikki toiminnot, joissa hyödynnetään monipistekosketuksia – esimerkiksi näytön zoomaamista kahdella sormella – tai toimintopolkuun perustuvia toimintoja, täytyy toteuttaa siten, että niitä voi käyttää esimerkiksi yhteen pisteeseen kohdistuvalla klikkauksella. (W3C, 2018; Selovuo, 2019, s. 80)

4.3 Ymmärrettävyys

On tärkeää ottaa huomioon eri tavat, joilla käyttäjät käyttävät verkkosivustoja. Esimerkiksi näkevä käyttäjä voi lukea informaation ruudulta, mutta sokeat henkilöt joutuvat käyttämään ruudunlukijaa. Tämän periaatteen mukaan verkkosivuston sisällön ja käyttöliittymän toiminnan täytyy olla ymmärrettävää. Tämä periaate sisältää yhteensä 17 kriteeriä, jotka on jaettu kolmeen eri osioon: luettava, ennakoitava ja syötteen avustaminen. (W3C, n.d., 2018; Selovuo, 2019, s. 83)

Luettava-osio koostuu kuudesta kriteeristä, jotka liittyvät muun muassa verkkosivuston kieleen, epätavallisiin sanoihin ja lyhenteisiin. Kriteerien mukaan sivustolla käytetty kieli tulisi olla ohjelmallisesti tunnistettavissa eli tieto kielestä tulisi asettaa HTML-koodiin. Muiden osiossa olevien kriteerien mukaan käyttäjä voisi halutessaan esimerkiksi selvittää epätavallisten sanojen tai lyhenteiden merkityksen. Kriteerit ottavat myös kantaa tekstin vaikeustasoon. (W3C, 2018; Selovuo, 2019, s. 84)

Ennakoitava-osiossa on viisi kriteeriä, jotka liittyvät sivuston toimintaan eli esimerkiksi navigointiin sekä elementtien kohdistukseen ja merkitsemiseen. Elementtien kohdistukseen liittyvän kriteerin mukaan elementtiin kohdistaminen ei saa aiheuttaa kontekstin muutosta. Tämä tarkoittaa esimerkiksi sitä, että kohdistettaessa painikkeeseen, painike ei saa aktivoitua ja johdattaa käyttäjää jollekin toiselle sivustolle ilman, että käyttäjä oikeasti klikkaa painiketta. Navigointiin liittyvän kriteerin mukaan navigaatiopalkin täytyy näyttää samalta jokaisella sivulla. (W3C, 2018; Selovuo, 2019, s. 85–86)

Syötteen avustaminen -osio sisältää kuusi kriteeriä. Osion pääohjeen mukaan käyttäjiä täytyy auttaa välttämään sekä korjaamaan virheitä. Jos esimerkiksi havaitaan virhe käyttäjän syötteessä, virheellinen kohta täytyy osoittaa ja virhe kuvata käyttäjälle tekstimuodossa. Kriteerien mukaan myös sisällön vaatiessa käyttäjän syötettä, täytyy käyttäjälle olla näkyvissä otsikko tai ohje, joka voi esimerkiksi sijaita syötekentän yläpuolella. On tärkeää huomata, että lomakkeissa esiintyvät placeholder-arvot eivät voi korvata lomakkeen otsikointia. (W3C, 2018; Selovuo, 2019, s. 86–88)

4.4 Toimintavarmuus

Periaatteen mukaan verkkosivuston täytyy toimia luotettavasti kaikilla laitteilla ja sen täytyy olla käytettävissä erilaisilla avustavilla teknologioilla. Tämä periaate sisältää vain yhden osion – yhteensopiva –, joka sisältää kolme kriteeriä. Tämän osion pääohjeena on maksimoida yhteensopivuus nykyisten ja tulevien käyttäjäagenttien eli verkkoselainten sekä avustavien teknologioiden kanssa. Kriteerit liittyvät verkkosivuston merkkauskielen jäsentämiseen, elementtien nimeämiseen, arvoihin ja rooleihin sekä tilailmoituksiin. (W3C, n.d., 2018)

Osion ensimmäisen kriteerin mukaan elementeillä täytyy esimerkiksi olla täydelliset alku- ja lopputagit ja niiden täytyy olla järjestetty oikein merkkauskielen syntaksin mukaisesti. Osion toisen kriteerin mukaan elementtien nimi, arvo ja rooli täytyy olla luettavissa ohjelmallisesti. Tämä kriteeri toteutuu, kun käytetään standardin mukaisia HTML-elementtejä oikein. Viimeinen kriteeri liittyy tilailmoituksiin. Kriteerin mukaan tilasta kertovat viestit tulisi voida selvittää ohjelmallisesti ilman kohdistuksen siirtämistä. ”Kyse on siis siitä, että tällaiset ilmoitukset, jotka ovat usein ilmiselviä näkeville käyttäjille ja näytetään sivuilla muualla kuin käsittelyn alla olevassa sivun kohdassa, on saatava myös avustavien tekniikoiden luettavaksi oikealla hetkellä.” (Selovuo, 2019, s.94). Tilailmoitus voi esimerkiksi ilmoittaa saatujen hakutulosten määrän tai ilmoituksen ostoskoriin lisätystä tuotteesta. (W3C, 2018; Selovuo, 2019, s.92–93)

5 Web-kehitys-tekniikoita

Seuraavaksi käsitellään tekniikoita, joilla verkkosivustoja voi kehittää. Web-kehityksen voi jakaa kahteen eri osa-alueeseen: frontend- ja backend-kehitykseen. Frontend-kehityksessä rakennetaan käyttöliittymää ja siihen kuuluvia toimintoja ja backend-kehityksessä keskitytään palvelinpuolen tapahtumiin. Molemmissa osa-alueissa on useita eri tekniikkavaihtoehtoja, joilla toimintoja voi toteuttaa. (GeeksforGeeks, 2021) Tässä opinnäytetyössä keskitytään frontend-kehitykseen. Tekniikoiden osalta käydään ensin läpi web-kehityksen perustekniikoita, jonka jälkeen syvennytään ReactJS-nimiseen JavaScript-kirjastoon, jolla rakennetaan tämän opinnäytetyön aikana yksinkertainen saavutettava verkkosivusto.

HTML, CSS ja JavaScript ovat verkkosivustojen rakentamisessa käytettäviä perustekniikoita. Kullakin tekniikalla on oma käyttötarkoituksensa: verkkosivustojen rakenne luodaan HTML-merkintäkielellä, tyylittely tehdään CSS-tekniikalla ja toiminnallisuudet ohjelmoidaan JavaScriptillä. Näistä tekniikoista CSS:llä ja JavaScriptillä on myös useita eri ohjelmistokehyksiä tai kirjastoja, joiden avulla kehitystyötä voi helpottaa ja nopeuttaa. (Ubah, 2021; BrowserStack, 2021; Acharya, 2021)

5.1 HTML

HTML eli Hypertext Markup Language on Tim Berners-Leen kehittämä merkintäkieli, jonka ensimmäinen versio julkaistiin vuonna 1993. HTML-merkintäkieltä käytetään verkkosivustojen rakenteen luomiseen. Ohjelmakoodissa 1 on esimerkki kahdesta HTML-elementistä: ohjelmakoodi sisältää otsikkoelementin (h1) ja tekstielementin (p). Elementit koostuvat alku- ja lopputagista sekä niiden välissä olevasta sisällöstä. HTML-merkintäkielessä on useita eri tageja, joiden avulla luodaan elementtejä. Esimerkiksi linkkien rakentamiseen käytetään a-tagia ja kuvissa käytetään img-tagia. (Mozilla, 2022a; University of Washington, n.d.)

Ohjelmakoodi 1 Esimerkki HTML-elementistä

```
<h1>Tämä on otsikko</h1>
<p>Tämä on tekstiä.</p>
```

5.2 CSS

CSS eli Cascading Style Sheets on Håkon Wium Lien vuonna 1994 kehittämä tekniikka, jolla luodaan tyylejä verkkosivustoille. CSS-tyyleille voi luoda kokonaan oman tiedoston tai kirjoittaa tyylejä HTML-tiedostoon. Ohjelmakoodissa 2 on esimerkki siitä, miten CSS:ää voi käyttää. Esimerkissä on määritelty tyylejä otsikkotekstille ja kuvalle. Kun luodaan tyylejä, on tärkeää ottaa huomioon eri verkkoselainten tuki, sillä jotkin CSS-määritykset eivät toimi tietyillä selaimilla tai tietyissä selainversioissa. (Mozilla, 2021a, 2021b; W3Schools, n.d.; Lie, n.d.)

Ohjelmakoodi 2 Esimerkki CSS:n käytöstä

```
h1 {
 color: green;
 font-size: 20px;
}
img {
   width: 80px;
   height: 80px;
}
```

CSS:lle on kehitetty useita sovelluskehyksiä, joiden avulla verkkosivustojen tyylittely on helpompaa ja nopeampaa. CSS:n sovelluskehyksiä ovat esimerkiksi Tailwind CSS ja Bootstrap. Näistä ohjelmistokehyksistä esimerkiksi Bootstrap sisältää valmiita elementtejä käyttöliittymiin. Sovelluskehysten lisäksi CSS:lle on kehitetty esiprosessoreita – esimerkiksi Sass ja Less –, jotka tuovat uusia mahdollisuuksia tyylittelyyn, joita CSS ei tavallisesti tarjoa: Sassin avulla voi käyttää muuttujia, joihin voi esimerkiksi tallentaa värejä. Muuttujia käyttämällä samoja arvoja ei siis tarvitse määrittää monesti, vaan riittää, että ne määritellään vain kerran, jolloin samaa muuttujaa voi käyttää eri elementtien tyyleissä. Ohjelmakoodissa 3 on esimerkki siitä, miten muuttujia voidaan käyttää tyylittelyssä. Sassin avulla on myös mahdollista käyttää esimerkiksi silmukkarakenteita ja ehtolauseita. (freeCodeCamp, 2020)

Ohjelmakoodi 3 Esimerkki Sass-esiprosessorin käytöstä (freeCodeCamp, 2020)

```
$yourcolor: #000056
.yourDiv {
  color: $yourcolor;
 }
```

5.3 JavaScript

JavaScript on Brendan Eichin vuonna 1995 kehittämä ohjelmointikieli. Se on yksi maailman käytetyimmistä ohjelmointikielistä ja sitä voi käyttää sekä selaimessa käyttäjäpuolen toiminnoissa että palvelinpuolella. JavaScriptiä ei tule sekoittaa Javaan, sillä molemmilla ohjelmointikielillä on oma syntaksinsa ja käyttökohteensa. (Mozilla, 2021c)

Kehitystyön helpottamiseksi JavaScriptille on kehitetty useita kirjastoja. Kirjastot sisältävät valmiita funktioita moniin erilaisiin käyttötarkoituksiin, kuten datan visualisointiin, animaatioihin, lomakkeiden rakentamiseen sekä numeroiden ja tekstien kanssa työskentelemiseen. Suosittuja kirjastoja ovat esimerkiksi Lodash, Anime.js ja Chart.js. (Acharya, 2021)

Kirjastojen lisäksi on kehitetty ohjelmistokehyksiä, jotka kirjastojen tapaan auttavat kehitystyössä. Kirjastoja käytettäessä kutsutaan kirjaston tarjoamia funktioita, mutta ohjelmistokehyksien tapauksessa ohjelmistokehys itse kutsuu koodeja ja käyttää niitä tietyllä tavalla eli ohjelmistokehyksiä ei voi kutsua. Suosittuja ohjelmistokehyksiä ovat esimerkiksi Googlen kehittämä Angular.js sekä Evan Youn kehittämä Vue.js. (Acharya, 2021)

5.4 ReactJS

Vuonna 2011 Jordan Walke kehitti Facebookilla työskennellessään ReactJS-nimisen JavaScript-kirjaston, joka julkaistiin avoimena lähdekoodina vuonna 2013 (Kozłowski, 2020). ReactJS on tullut erittäin suosituksi JavaScript-kirjastoksi julkaisunsa jälkeen ja monet suuret yritykset käyttävät sitä nykyään, esimerkiksi Facebook, Netflix, Uber ja Airbnb käyttävät sitä verkkopalveluissaan (Patel, 2022a).

Kun verkkosivustoja kehitetään ReactJS:llä käyttöliittymän elementit jaetaan komponentteihin. Esimerkiksi verkkosivustolla oleva hakukenttä voisi olla yksi komponentti, jota käytetään eri sivuilla. Komponenttien avulla samoja koodeja ei siis tarvitse kirjoittaa monta kertaa eri puolille verkkosivustoa, mikä säästää kehittäjien aikaa. (Patel, 2022b)

Yksi ReactJS:n hyödyistä on sen kyky päivittää vain tietty osa sivustolla lataamatta koko verkkosivustoa uudelleen. Tämä on mahdollista, sillä ReactJS käyttää virtuaalista DOM-

puuta. Käytännössä tämä tapahtuu siten, että verkkosivuston nykyisestä tilasta otetaan kopio ja kun jokin verkkosivuston osa muuttuu, verrataan otettua kopiota verkkosivuston uuteen tilaan, jolloin ainoastaan muuttunut elementti päivitetään. Virtuaalisen DOM-puun ansiosta ReactJS:llä rakennetut verkkosivustot ovat nopeita. (Patel, 2022b)

ReactJS on suosittu myös siinä käytetyn JSX:n vuoksi. JSX eli JavaScript XML mahdollistaa HTML- ja JavaScript-koodin kirjoittamisen samaan tiedostoon. HTML-koodia ei voi kuitenkaan kirjoittaa sen normaalin syntaksin mukaisesti. Joihinkin HTML-tageihin täytyy JSX:ssä esimerkiksi lisätä kauttaviiva ilmentämään, että se on itsestään sulkeutuva, vaikka tavallisesti HTML-koodissa näin ei tarvitse tehdä. Myös esimerkiksi määritettäessä HTML-elementille luokkaa tyylejä varten käytettäisiin tavallisesti class-sanaa, mutta JSX:ssä täytyy käyttää className-sanaa. JSX eroaa tavallisesta HTML:stä myös siten, että sitä ei voi suoraan näyttää selaimessa, vaan se pitää ensin kääntää Babelin eli JavaScript-kääntäjän avulla sellaiseen muotoon, jonka selaimet ymmärtävät. (Babel, n.d.; Kolade, 2021; Patel, 2022b) JSX-koodilla voi myös tuoda käyttäjän nähtäville erilaisia arvoja, joita JavaScript-koodeista saadaan (Arancio, 2021). Seuraavaksi katsotaan yksinkertainen esimerkki HTML:n ja JavaScriptin käytöstä JSX-koodissa. Ohjelmakoodissa 4 on luotu nimiLista-niminen muuttuja, jonka sisällä on lista. Tämä muuttuja tuodaan seuraavaksi HTML-elementtiin.

Ohjelmakoodi 4 Muuttujaan tallennettu HTML-elementti

```
let nimiLista = (
  <ul>
    <li>Ville</li>
    <li>Henna</li>
    <li>Laura</li>
  </ul>
);
```

Äsken luotu nimiLista-niminen muuttuja kirjoitetaan ohjelmakoodissa 5 olevan HTML-elementin keskelle. Ohjelmakoodista nähdään, että JavaScript-koodia voi tuoda HTML-elementtiin laittamalla koodin kaarisulkujen sisälle. Kuvasta 6 nähdään, miltä tämä koodi lopulta näyttää selaimessa.

Ohjelmakoodi 5 Muuttujan arvon näyttäminen JSX-koodissa

```
<section>
  <p>Tässä on nimilista:</p>
  {nimiLista}
</section>
```

Kuva 6 Muuttuja web-selaimessa

Tässä on nimilista:

- Ville
- Henna
- Laura

6 Verkkosivuston suunnittelu saavutettavuus huomioon ottaen

Kun suunnitellaan verkkosivustoja, on otettava huomioon sisältö ja tekninen toteutus, jotta verkkosivusto olisi saavutettava. Teknisen toteutuksen osalta on tärkeää, että sivuston lähdekoodi on virheetöntä ja loogista. Palvelu täytyy myös kehittää siten, että se toimii eri päätelaitteilla ja avustavilla teknologioilla. Sisällön osalta on tärkeää, että käyttöliittymä on selkeä ja ymmärrettävä, jotta käyttäjä voi suorittaa haluamansa toiminnot helposti. Koska ihmiset ovat erilaisia, kehitystyön aikana täytyy ottaa huomioon ihmisten erilaiset tilanteet ja tarpeet. (Etelä-Suomen aluehallintovirasto, n.d. -h)

6.1 Sisällön saavutettavuus

Sisällön saavutettavuus tarkoittaa kognitiivista saavutettavuutta. Verkkosivuston sisällön täytyy siis olla helposti omaksuttavissa ja käytettävissä. Verkkosivuston sisällön suunnittelussa täytyy ottaa huomioon esimerkiksi sivustolla käytetty kieli, ulkoasu ja sivupohjan selkeys. (Celia, n.d. -c; Kehitysvammaliitto ry, n.d. -k)

Sivuston sisältö täytyy aina kirjoittaa sivuston kohderyhmälle. Sivustolla käytetyn kielen tulisi olla yleiskieltä, eikä se saisi sisältää vaikeita ilmaisuja, sanontoja tai kielikuvia. Tekstissä ei myöskään tulisi käyttää jonkin alan erikoistermistöä. Jos termejä kuitenkin käytetään, niiden merkitys pitäisi selittää tekstin alussa. Tekstin sisältö täytyy myös kirjoittaa loogisessa järjestyksessä ja selkeyden vuoksi käyttää väliotsikoita. Kielelliseen saavutettavuuteen liittyy edellä mainittujen asioiden lisäksi eri kieliversiot. Suomalaisilla verkkosivustoilla on usein tarjolla englannin- ja ruotsinkielinen versio ja esimerkiksi Kelan ja Verohallinnon verkkosivustoilla on saatavilla sisältöä monilla muillakin kielillä. Jos eri kieliversioita ei ole saatavilla, on selkeä kieli tärkeää esimerkiksi maahanmuuttajille, jotka eivät puhu suomea äidinkielenään. (Selovuo, 2019, s. 118, Kehitysvammaliitto ry, n.d. -g; Leskelä, 2019, s. 68)

Ulkoasun suunnittelun osalta on tärkeää kiinnittää huomiota esimerkiksi sivulla käytettyihin väreihin, typografiaan ja käyttöliittymän elementtien suunnitteluun. Värivalinnoissa tulisi huomioida käyttäjien kyky nähdä värejä, esimerkiksi punasokean silmin värit näyttävät aivan erilaisilta todellisiin väreihin verrattuna. Väreihin liittyy myös olennaisesti kontrasti, joka WCAG-ohjeistuksen mukaan normaalin tekstin ja taustan välillä tulisi olla 4,5:1 ja suuren

tekstin osalta vähintään 3:1. Kuva 7 havainnollistaa eron hyvän ja huonon kontrastin välillä. Kuvan vasemmalla puolella taustan ja tekstin välinen kontrasti on 1,22:1 eli kontrastiarvo on kaukana WCAG-ohjeistuksen vähimmäiskontrastiarvosta. Kuvan oikealla puolella kontrasti on 4,53:1 eli vähimmäiskontrastiarvo täyttyy juuri. Väreihin liittyen on myös tärkeää huomata, että WCAG-ohjeistuksen mukaan värit eivät saa olla ainoa keino informaation esittämiseen. (Selovuo, 2019, s. 43–44, s. 110, s. 114; W3C, 2018)

Kuva 7 Taustan ja tekstin välinen kontrasti

Sisällön typografiassa täytyy huomioida muun muassa fontin koko, fontin ulkoasu ja tekstin välistys. Jos fonttikoko on liian pieni, tekstiä voi olla hankala lukea. Verkkosivustoa voi yrittää suurentaa ongelman korjaamiseksi, mutta silloin sivuston rakenne saattaa mennä rikki. Toisaalta taas liian suuri fonttikoko voi haitata lukemista, jolloin käyttäjä voi haluta pienentää fonttia. Ei ole kuitenkaan olemassa tiettyä fonttikokoa, joka sopisi kaikille käyttäjille ja tämän vuoksi verkkosivustoilla olisi hyvä olla toiminto, jonka avulla käyttäjät voivat muuttaa fonttikokoa siten, että käyttöliittymä ei muutu. (Laak, 2006a, 2006b; Kehitysvammaliitto ry, n.d. -h)

Fonttikoon lisäksi fontin ulkoasu on tärkeä tekijä tekstin luettavuudessa. Verkkosivustoilla kannattaa suosia päätteettömiä fontteja (sans-serif), jotka ovat suoraviivaisia ja tasaisia, eivätkä sisällä koukeroita. Nämä fontit ovat helppolukuisempia alhaisen resoluution päätelaitteilla. Verkkosivustolla on myös tärkeää huomioida tekstin välistys, sillä se parantaa sivuston luettavuutta. Esimerkiksi henkilöt, joilla on huono näkö tai lukihäiriö hyötyvät tästä. WCAG-ohjeistus sisältää seuraavat ohjeistukset tekstin välistykselle: rivin korkeuden tulisi olla vähintään 1,5 kertainen fonttikokoon nähden, kappalevälin vähintään kaksinkertainen fonttikokoon nähden, kirjainvälin vähintään 0,12 kertainen fonttikokoon nähden ja sanojen välin tulisi olla vähintään 0,16 kertainen fonttikokoon nähden. (Laak, 2006a; Kehitysvammaliitto ry, n.d. -h; W3C, 2018)

Kun suunnitellaan verkkosivustoja, siellä käytettyjen elementtien ja tyylien tulisi olla yhteneviä, sillä se auttaa ihmisiä tunnistamaan toiminnallisuuksia eri sivuilla. Esimerkiksi henkilöt, joilla on vaikeuksia lukea, hyötyvät yhteneväisyydestä. Elementtien tyylittelyssä tulisi myös ottaa huomioon, että ei poista käyttäjille tarpeellisia ominaisuuksia käyttöliittymästä: Selovuon Saavutettavuusoppaan mukaan web-suunnittelussa on viime aikoina ollut tyylitellessä tapana poistaa esimerkiksi aktiivisten elementtien visuaalinen korostus, jonka selaimet oletuksena elementeille tekevät. Tämä on kuitenkin erittäin tärkeä toiminto henkilöille, jotka selaavat verkkosivustoa näppäimistön avulla. Tässä tapauksessa korostuksen tyyliä tulisi muuttaa siten, että se sopii ulkonäöltään verkkosivustolle. Elementtien ulkoasussa ja toiminnassa täytyy myös huomioida ominaisuudet, jotka saattavat aiheuttaa sairauskohtauksia. Esimerkiksi välkkyvä animaatio saattaa aiheuttaa epileptisen kohtauksen. (Kaur, 2018; Selovuo, 2019, s. 79)

Saavutettavuuden kannalta on tärkeää, että verkkosivuston sivupohja on selkeä, mikä tarkoittaa sitä, että kehitysvaiheessa täytyy ottaa huomioon esimerkiksi lukemisjärjestys, navigaatio, otsikointi ja linkit. Loogisen lukemisjärjestyksen kannalta sivuston pääsisältö kannattaa tuoda heti sivun alussa esille ja sen tulisi olla erotettavissa sekä visuaalisesti että ohjelmallisesti. Ohjelmallisuudella tarkoitetaan sitä, että esimerkiksi sivuston HTML-koodissa pääsisältö on laitettu main-tagien sisälle, jolloin avustavat teknologiat tunnistavat osion tarkoituksen. Tämän lisäksi on tärkeää huomioida, että sivuston sisällön lukemisjärjestys täytyy olla sama kaikille käyttäjille: CSS-tyylittelyjen avulla verkkosivuston sisältö on mahdollista esittää näkeville käyttäjille eri järjestyksessä kuin käyttäjille, jotka käyttävät esimerkiksi ruudunlukuohjelmaa. Ruudunlukuohjelma etenee HTML-koodin järjestyksen mukaisesti, joten on tärkeää, että sivuston sisältö kirjoitetaan HTML-koodiin siinä järjestyksessä, kuin käyttäjien halutaan siinä etenevän. (Kehitysvammaliitto ry, n.d. -i, -j; W3C, 2019)

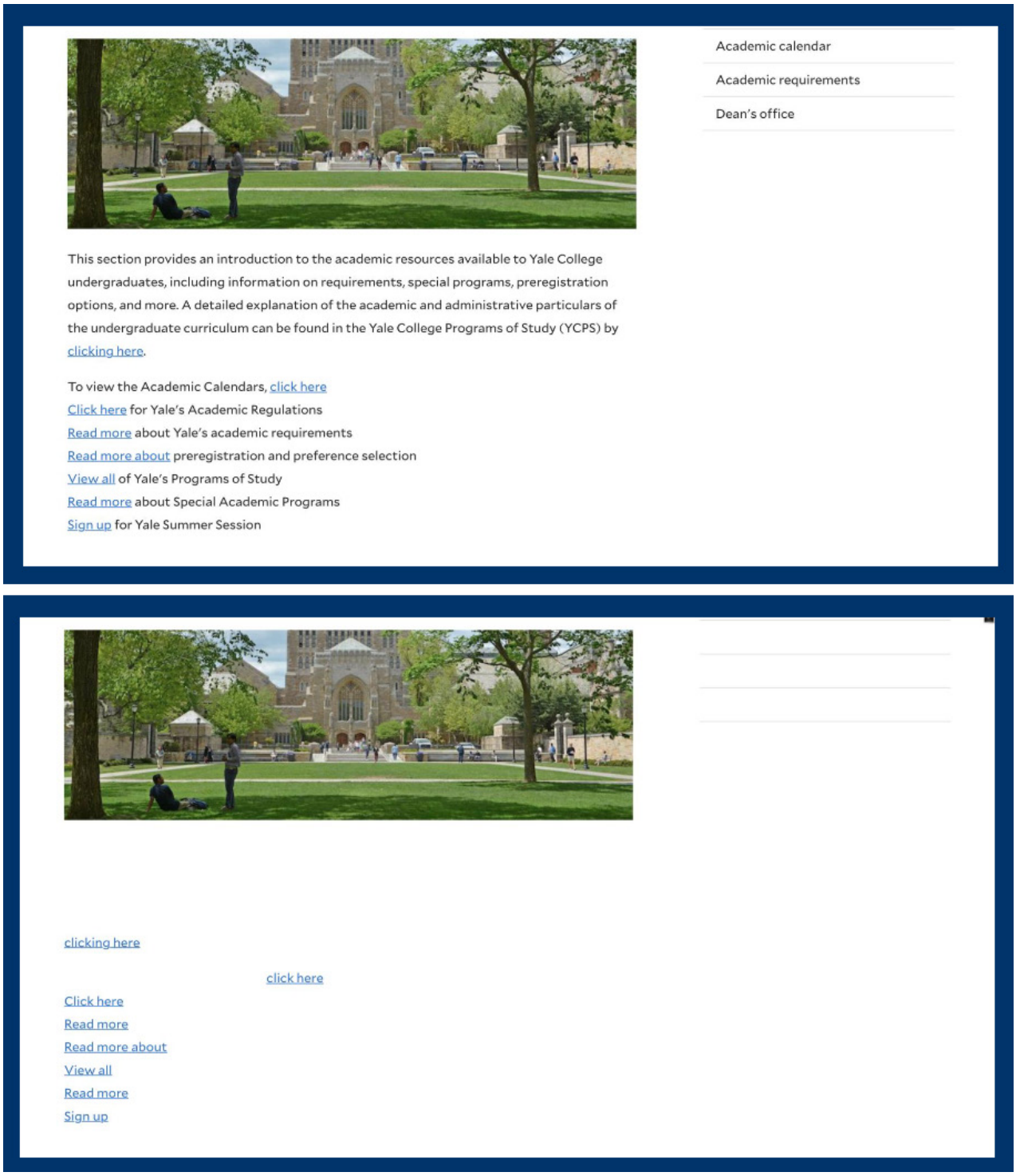
Verkkosivuston navigaatiopalkin tulisi sijaita tutussa paikassa, kuten sivuston ylälaidassa tai vasemmassa reunassa. Navigaatiopalkin täytyy olla selkeä ja sen tulisi auttaa käyttäjiä löytämään tarvitsemansa tieto verkkosivustolta. Selkeyden vuoksi eri sivujen täytyy olla selkeästi nimettyjä ja ne voidaan esimerkiksi jakaa loogisiin osioihin. Navigaatiopalkissa tulisi myös välttää liian syviä navigointirakenteita eli kehittäjien tulisi esimerkiksi välttää alavalikkojen tekemistä, sillä niiden käyttäminen saattaa olla teknisesti haastavaa joillekin

käyttäjille. Tavallisen navigaatiopalkin lisäksi käyttäjille tulisi tarjota muita navigointivaihtoehtoja verkkosivustolla, esimerkiksi heikkonäköiset henkilöt saattavat käyttää mielellään hakukenttää, sillä se voi olla helpompaa kuin selata suurta menuvalikkoa ja henkilöt, joilla on kognitiivisia rajoitteita, saattavat suosia sisällysluetteloa. (Selovuo, 2019, s. 46; Yale University, n.d. -a)

Hyvä otsikointi parantaa verkkosivuston luettavuutta. Otsikoiden avulla käyttäjät pystyvät silmäilemään tekstiä sekä hahmottamaan sen sisällön paremmin. Otsikoinnista on myös määrätty WCAG-ohjeistuksessa, että niiden tulisi kuvailla sisällön aihetta tai merkitystä. Ohjeistuksessa vaaditaan myös, että otsikoiden täytyy olla ohjelmallisesti tunnistettavissa. Tämän voi käytännössä toteuttaa käyttämällä HTML-merkintäkielen otsikkotageja. Otsikoinnissa on tärkeää kiinnittää huomiota niiden hierarkiaan: otsikkotageilla on eri tasoja, esimerkiksi h1-tagia tulisi käyttää ylimmän tason otsikossa ja siitä eteenpäin otsikkotasojen pitäisi mennä portaittain alaspäin eli h1-otsikon jälkeen pitäisi tulla h2-otsikko. (Celia n.d. -b)

Otsikoiden tapaan myös linkkitekstin täytyy kuvata linkin tarkoitusta eli se ei saisi olla esimerkiksi muodossa ”Lue lisää” tai ”Klikkaa tästä”. Kun ruudunlukuohjelma lukee tällaisen linkin ääneen ilman kontekstia, se saattaa olla käyttäjälle hämmentävää. Onkin erittäin tärkeää, että käyttäjä ymmärtää linkin tarkoituksen ilman ympäröiviä lauseita ja sisältöä. (Yale University, n.d. -b) Kuvassa 8 on havainnollistettu linkkitekstien merkitystä. Kuvan yläosassa on verkkosivusto, jossa linkkien teksti on ymmärrettävissä ympäröivän tekstin avulla. Kuvan alaosassa on sama verkkosivusto, mutta ilman tekstejä. Tässä tapauksessa linkkien tarkoitus on mahdotonta ymmärtää ilman kontekstia.

Kuva 8 Esimerkki linkkitekstien merkityksestä (Yale University, 2018)



Myös linkkien tyylittelyyn täytyy kiinnittää huomiota. Verkkosivustojen linkit on usein erotettu eri värillä teksteissä, mutta tämä ei tee niistä tarpeeksi erottuvia. Esimerkiksi värisokeat henkilöt eivät välttämättä erota linkkiä tekstistä ja tämän vuoksi niiden pitäisi olla esimerkiksi myös alleviivattuja. Linkkien täytyy myös olla tarpeeksi isoja, sillä pieniin linkkeihin voi olla vaikea osua hiirellä. Toisaalta taas linkit eivät saa olla liian isoja, sillä

mobiililaitetta käytettäessä käyttäjä saattaa vahingossa klikata linkkiä selatessaan nettisivua. Joskus myös kuvat saattavat toimia linkkeinä. Tällaista pitäisi välttää, mutta jos kuitenkin on tarpeen käyttää kuvaa tässä tarkoituksessa, kuvan alt-tekstissä täytyy kertoa linkin tarkoitus ja kohde. (Yale University, n.d. -b; ks. myös Silver, 2016)

6.2 Tekninen toteutus

Verkkosivustojen tekninen toteutus vaikuttaa paljon saavutettavuuteen. Teknisen toteutuksen osalta täytyy esimerkiksi tietää, miten HTML-elementtejä käytetään oikein ja mitä WAI-ARIA tarkoittaa sekä milloin sitä tarvitaan. Tämän lisäksi on tärkeää tietää, miten JavaScript ja CSS voivat vaikuttaa saavutettavuuteen. (Mozilla, 2022b)

Mozillan verkkosivuston mukaan hyvin suuri osa internetin sisällöstä voitaisiin tehdä saavutettavaksi käyttämällä oikeita HTML-elementtejä oikeaan tarkoitukseen. HTML-elementit voidaan jakaa ei-semanttisiin ja semanttisiin elementteihin. Ei-semanttiset elementit, kuten div ja span eivät kerro mitään niiden sisällöstä. Semanttiset HTML-elementit välittävät tietoa niiden sisällöstä ja tämän vuoksi niiden käyttäminen onkin tärkeää. Semanttisia elementtejä ovat esimerkiksi button, article ja nav. Jos verkkosivusto sisältäisi esimerkiksi otsikon ”Hedelmät”, joka on rakennettu käyttäen semanttista h2-otsikkoelementtiä, ruudunlukuohjelma voi ilmoittaa käyttäjälle ”Otsikkotaso 2, hedelmät”. Semanttisia HTML-elementtejä voi kuitenkin käyttää myös väärin, esimerkiksi otsikko on mahdollista rakentaa käyttäen semanttista p-elementtiä, joka tyyliteltäisiin CSS:n avulla otsikon näköiseksi. Tällöin ruudunlukuohjelma ei kuitenkaan tunnistaisi elementtiä otsikoksi, eikä käyttäjä voisi selata verkkosivustoa otsikoittain. (Mozilla, 2022c; Wunder, n.d. -a)

Saavutettavuuden lisäksi semanttisilla HTML-elementeillä on muitakin hyötyjä: Ne helpottavat kehitystyötä sisältämällä tarvittavia ominaisuuksia jo valmiiksi. Semanttinen koodi on myös parempi mobiilissa, sillä sen avulla tiedostokoko on pienempi ja sen lisäksi käyttöliittymä on helpompi tehdä responsiiviseksi. Näiden lisäksi semanttiset HTML-elementit auttavat hakukoneoptimoinnissa, sillä hakukoneet kiinnittävät enemmän huomiota avainsanoihin, jotka ovat semanttisten elementtien sisällä. (Mozilla, 2022c)

WAI-ARIA (Web Accessibility Initiative – Accessible Rich Internet Applications Suite) on W3C:n kirjoittama spesifikaatio, jonka avulla verkkosivuston elementeistä voi tehdä saavutettavia. Se auttaa erityisesti dynaamisen sisällön ja edistyneiden käyttöliittymien rakentamisessa. Käyttöliittymiä rakennettaessa kehittäjät saattavat joutua käyttämään ei-semanttisia HTML-elementtejä, jolloin WAI-ARIAn avulla niistä on mahdollista tehdä saavutettavia. Tällä hetkellä W3C suosittelee WAI-ARIA 1.1 -versiota, joka julkaistiin vuonna 2017. Uutta versiota 1.2. kehitetään, mutta sitä ei ole vielä julkaistu W3C:n uudeksi suositukseksi. (W3C, 2020; Mozilla, 2022d)

ARIA-määritteitä on kolmea eri tyyppiä: roolit, ominaisuudet ja tilat. Roolit määrittelevät elementin tarkoituksen käyttöliittymässä. Niillä voidaan esimerkiksi määrittää, että elementti on painike tai hakukenttä. Ominaisuudet määrittelevät elementin ominaisuuksia eli ARIA-koodilla voi esimerkiksi määrittää, että lomakkeelta on pakko valita jokin valintaruutu, ennen kuin lomakkeen voi lähettää. Tila määrittelee elementin nykyisen tilan eli esimerkiksi voidaan kertoa, että radiopainike on painettuna. (Wunder, n.d. -b)

WAI-ARIAn käyttämisessä täytyy olla huolellinen, sillä väärin käytettynä verkkosivuston saavutettavuus saattaa huonontua. Tämän vuoksi on siis parempi olla käyttämättä ARIA-koodeja kuin käyttää niitä väärin. Kehittäjien täytyy myös huomioida, että ARIA-koodit eivät välttämättä toimi samalla tavalla kaikissa ruudunlukuohjelmissa ja tämän vuoksi ARIA-koodien toimivuus tulisi testata hyvin. Käyttöliittymien rakentamisessa tulisikin suosia semanttisia HTML-elementtejä aina, kun se on mahdollista. Kun esimerkiksi vertaa semanttista nav-elementtiä ja ARIAn navigation-roolia, ruudunlukuohjelmat ilmoittavat yleensä nav-elementin oikein suomeksi sanomalla navigaatio, mutta ARIAn navigation-roolin kohdalla ohjelma saattaa sanoa englanninkielisen navigation-sanan suomalaisella aksentilla. Semanttiset HTML-elementit sisältävät myös interaktiivisissa elementeissä tarvittavan toiminnallisuuden jo valmiiksi, esimerkiksi button-elementti käyttäytyy oletuksena painikkeen tavoin. Jos painikkeen toteuttaisi jollain muulla tavalla ja lisäisi siihen ARIAn button-roolin, ei elementti käyttäytyisi automaattisesti painikkeen tavoin. (Wunder, n.d. -b; Selovuo, 2019, s. 96–98)

Verkkosivustoilla käytetty CSS ja JavaScript saattavat huonontaa saavutettavuutta, jos niitä käytetään väärin. CSS:n käytössä täytyy huomioida, että verkkosivuston elementtejä saa

tyylitellä, mutta niitä ei saa muuttaa liikaa, jolloin ne eivät enää ole ulkoasultaan tuttuja tai käyttäydy odotetusti. CSS:n käytössä täytyy esimerkiksi huomioida tiettyjen tyylimäärittelyjen vaikutus sisältöjen saatavuuteen. Joskus verkkosivuston ulkoasun vuoksi kaiken sisällön ei tarvitse olla kerralla näkyvissä, jolloin kehittäjät saattavat käyttää CSS-määrittelyjä sisällön piilottamiseksi. Visuaalisuudella ei kuitenkaan ole merkitystä kaikille käyttäjille, esimerkiksi ruudunlukuohjelmia käyttävien henkilöiden kannalta oleellista on, että sivuston lähdekoodissa kaikki sisältö on esitetty loogisessa järjestyksessä. Ruudunlukuohjelmien kannalta piilottamisessa täytyykin huomioida, että tiettyjä CSS-määrittelyjä käyttäen sisältö piilotetaan myös ruudunlukuohjelmilta. (Mozilla, 2022e)

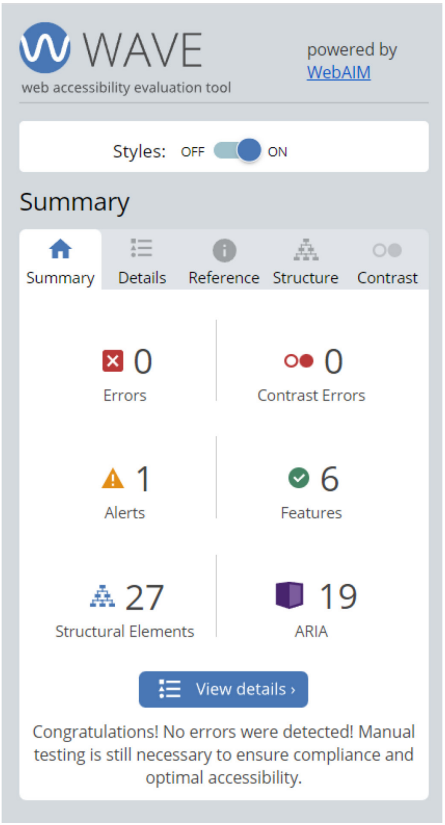
JavaScriptissä ongelma saavutettavuuden kannalta on se, että sitä käytetään joskus liikaa verkkosivustoilla. JavaScriptillä luotu sisältö on yleisesti avustavien teknologioiden saatavissa, mutta se saattaa silti aiheuttaa ongelmia. Joskus jatkuvasti muuttuva dynaaminen sisältö saattaa esimerkiksi vaikuttaa navigoimiseen tai selaimen toimintaan. Kun näppäimistön fokuksessa oleva elementti esimerkiksi muuttuu, menee piiloon tai poistuu, fokus saattaa siirtyä takaisin verkkosivuston alkuun. JavaScriptiä saatetaan myös käyttää turhaan esimerkiksi 3D-informaatiolaatikon rakentamiseen, vaikka tavallinen teksti saattaisi riittää informaation välittämiseksi. JavaScriptin tarkoituksena pitäisi siis olla toimintojen parantaminen, mutta toimintoja ei tulisi toteuttaa kokonaan sillä. JavaScriptiin liittyy myös huolenaihe verkkosivuston tapahtumien osalta. Jotkin elementit voivat esimerkiksi sisältää hiiren eri toimintoihin liittyviä tapahtumakäsittelijöitä, jolloin aktivoituva toiminto ei kuitenkaan ole näppäimistöä käyttävien henkilöiden saatavissa. Kehittäjien täytyy siis tapahtumakäsittelijöiden osalta lisätä tarvittavat ohjelmakoodit, jotta näppäimistökäyttäjätkin huomioidaan. (WebAIM, 2020; Mozilla, 2022e)

7 Saavutettavuuden testaaminen

Verkkosivuston saavutettavuutta voi testata itse käyttäen erilaisia tarkastustyökaluja ja selainlaajennuksia, jotka simuloivat erilaisia rajoitteita. Avustavien teknologioiden osalta tietokoneelle voi ladata esimerkiksi ruudunlukuohjelman, jolla verkkosivustoa voi testata. Testaamisessa on tärkeää ottaa huomioon erilaiset käyttäjät, apuvälineet ja selaimet. Testaamisen kannalta on tärkeää huomata, että itse testaaminen ei kuitenkaan korvaa oikeilla käyttäjillä toteutettua testausta. (Kehitysvammaliitto ry, n.d. -l; ks. myös Celia, 2022)

Testausta voi toteuttaa esimerkiksi WAVE-nimisellä tarkastustyökalulla, joka mittaa WCAG-kriteerien noudattamista. WAVE:n verkkosivustolla voi syöttää testattavan verkkosivuston URL-osoitteen ja saada palautetta sen saavutettavuudesta. (Celia, 2022) Kuvassa 9 näkyy työkalun antama palaute erään verkkosivuston saavutettavuudesta. Työkalun eri välilehdillä voi tutkia esimerkiksi löydettyjä virheitä, verkkosivuston rakennetta sekä värien kontrasteja.

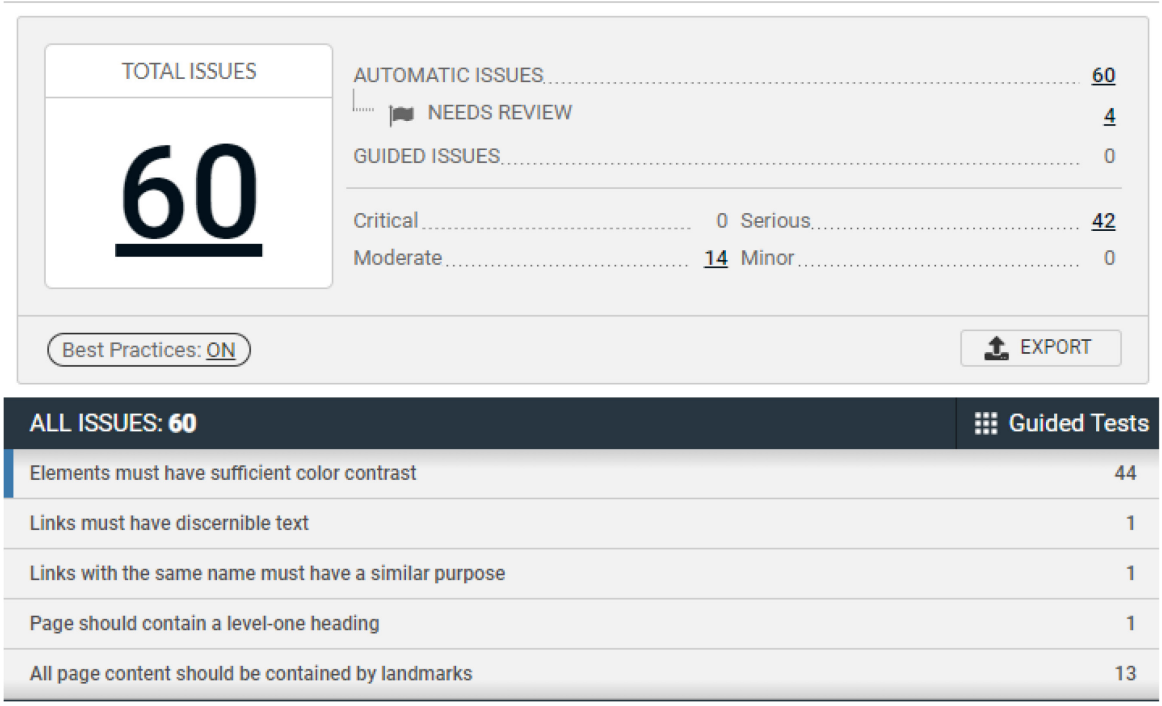
Kuva 9 WAVE-työkalu



Verkkosivustojen saavutettavuuden tarkistamiseen on myös saatavilla axe DevTools -niminen tarkastustyökalu, jonka voi ladata Chrome Web Storesta. Tarkastustyökalua voi käyttää menemällä haluamalleen verkkosivustolle, jonka jälkeen avataan selaimen kehittäjätyökalut. Kehittäjätyökaluissa käynnistetään verkkosivuston analysointi, jonka valmistuttua aukeaa saadut tulokset. Automaattisten tarkistusten kohdalla täytyy kuitenkin huomioida, että välillä tarkistukset eivät osaa tulkita verkkosivustojen sisältöä oikein, jolloin tuloksissa saattaa olla vääriä virhemerkintöjä. Automaattisia tarkistuksia voi kuitenkin käyttää manuaalisen testauksen tukena ja ne auttavat huomaamaan saavutettavuusongelmia. (Selovuo, n.d.; ks. myös Kuntaliitto, n.d.)

Kuvassa 10 näkyy axe DevToolsin tekemä analyysi eräällä verkkosivustolla. Tarkastustyökalu on tässä tapauksessa havainnut 60 virhettä, jotka täytyy tarkistaa. Saadut virheilmoitukset liittyvät esimerkiksi otsikointiin, linkkeihin sekä värikontrasteihin. Virheitä klikkaamalla aukeaa sivu, jossa on tietoa esimerkiksi virheen sijainnista ja mitä tulisi tehdä virheen korjaamiseksi.

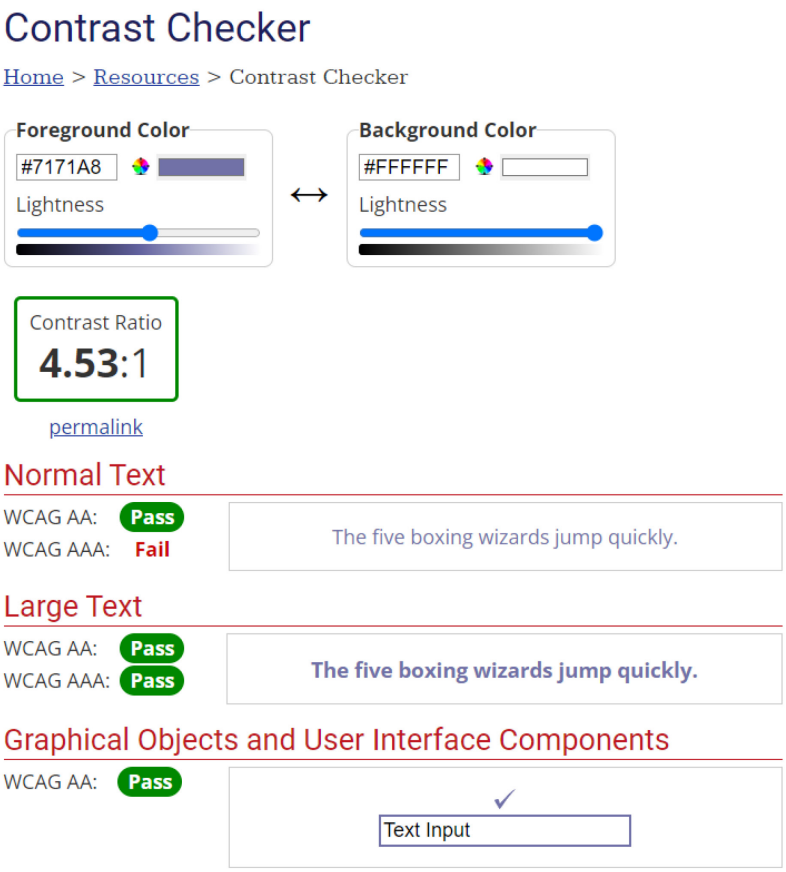
Kuva 10 axe DevTools



Vaikka automaattiset tarkastustyökalut tarkistavat värikontrasteja, värejä voi testata myös manuaalisesti esimerkiksi WebAIMin Contrast Checkerillä. Kuvassa 11 on tarkistettu kahden

värin välinen kontrasti ja tulokseksi on saatu 4,53:1. Näiden värien tapauksessa WCAG-ohjeistuksen AA-tason kontrastivaatimus täyttyy, jonka lisäksi AAA-tason kontrastivaatimus suurelle tekstille täyttyy.

Kuva 11 WebAIMin työkalu kontrastin tarkistamiseen



Tarkastustyökalujen lisäksi on saatavilla selainlaajennuksia ja ohjelmia, joilla voi simuloida erilaisia rajoitteita, joita käyttäjillä saattaa olla. Erilaisten selainlaajennusten avulla voi simuloida esimerkiksi näköön ja motoriikkaan liittyviä rajoitteita. (Celia, 2022) Kuvassa 12 on käytetty Silktide-selainlaajennusta, joka on saatavilla Chrome-selaimeen. Kuvassa Googlen hakusivu on simuloitu näkymään punasokean silmin, jonka huomaa siitä, että esimerkiksi Googlen logossa olevat värit ovat muuttuneet. Tämän lisäksi kuvassa on käytetty likinäköisyyden ja harmaakaihin simulointia, mikä tekee näkymästä sumean.

Kuva 12 Silktide-selainlaajennus rajoitteiden simuloimiseen

Koneellisen tarkistuksen lisäksi erityisesti vuorovaikutteiset toiminnot tulisi testata manuaalisesti. Kun verkkosivustoa testataan käyttämällä ainoastaan näppäimistöä, nähdään ovatko kaikki toiminnallisuudet käytettävissä ilman hiirtä. Tämän lisäksi nähdään, onko käyttöliittymän elementeissä WCAG-kriteerien mukainen fokus ja pystyykö käyttäjä havainnoimaan koko ajan, missä kohdassa verkkosivustolla hän on. (Näkövammaisten liitto ry, 2021c)

Avustavien teknologioiden osalta verkkosivustoa voi testata esimerkiksi ruudunlukuohjelmalla. Vaikka ruudunlukuohjelman tehokas käyttäminen vaatii opettelua, on muutamalla perustoiminnolla silti mahdollista tehdä suuntaa antava arvio sivuston käytettävyydestä. Kun verkkosivu kuunnellaan ruudunlukuohjelmalla, voi esimerkiksi arvioida, onko tekstisisältö vaikeaselkoista. Internetistä on mahdollista ladata esimerkiksi NVDA-niminen ruudunlukuohjelma. (Näkövammaisten liitto ry, 2021c; Kehitysvammaliitto, n.d. -m)

Verkkosivustojen testaamisessa täytyy myös huomioida eri päätelaitteet ja selaimet, sillä ne saattavat vaikuttaa sivuston toimintaan ja ulkoasuun. Päätelaitteiden osalta testausta tulisi suorittaa älypuhelimella, tabletilla sekä tietokoneella. Testaamisessa täytyy huomioida eri käyttöjärjestelmien ja niiden versioiden vaikutus. Myös eri selaimet ja niiden versiot voivat vaikuttaa saavutettavuuteen. Esimerkiksi jotkin ratkaisut verkkosivustolla eivät välttämättä toimi toisessa selaimessa tai selaimen vanhemmissa versioissa. (Kehitysvammaliitto ry, n.d. -l)

Verkkosivustoilla on tärkeää suorittaa käyttäjätestausta, sillä usein monet saavutettavuusongelmat huomataan käyttäjätestauksen aikana. Testaustuloksen kannalta testihenkilöiden tulisi edustaa erilaisia henkilöitä eli testauksessa olisi hyvä olla esimerkiksi vieraskielisiä, ikääntyneitä sekä henkilöitä, joilla on jokin rajoite. Tämän lisäksi olisi hyvä, jos henkilöt ovat eritasoisia tietokoneen käyttäjiä. Käyttäjätestauksessa testataan sekä teknistä saavutettavuutta että verkkosivuston käytettävyyttä eli onko sivuston käyttäminen tehokasta ja miellyttävää. Testaustapoja on erilaisia ja testauksessa voi esimerkiksi tehdä A/B-testausta tai katseenseurausta. A/B-testaamisessa käyttäjillä testataan kahta eri versiota palvelusta, jotta saadaan tietoa siitä, kumpi versio on parempi kohderyhmälle. Katseenseurannassa voidaan tutkia, mihin osioihin käyttäjät kiinnittävät huomiota ja huomioivatko käyttäjät joitain osia lainkaan. (Kehitysvammaliitto ry, n.d. -m; Agenda Helsinki, 2019)

8 Ohjelmointiprojekti

Seuraavaksi tutustutaan opinnäytetyön aikana tehtyyn ohjelmointiprojektiin, jonka tarkoituksena on tuoda käytäntöön aiemmissa luvuissa esiteltyjä asioita. Projektin avulla havainnollistetaan, miten saavutettavuus otetaan huomioon ohjelmoinnissa, sekä miten saavutettavuutta käytännössä testataan. Tämä projekti on toteutettu käyttämällä ReactJS-nimistä JavaScript-kirjastoa, koska se on tällä hetkellä erittäin suosittu tekniikka web-kehityksessä.

Projektin aikana on toteutettu yksinkertainen matkailuaiheinen verkkosivusto, joka sisältää kolme sivua: home (etusivu), about (tietoa meistä) ja destinations (kohteet). Verkkosivusto sisältää erilaisia elementtejä, joiden toteutusta tutkitaan saavutettavuuden näkökulmasta. Sivuston testausvaiheessa katsotaan, minkälaisia tuloksia saadaan ja mitä mahdollisesti voisi vielä parantaa. Ohjelmointiprojektissa käytetyt kuvat ja ikonit on etsitty Pexels- ja Flaticon-nimisistä palveluista.

8.1 Saavutettavuuden toteuttaminen projektissa

Yleisellä tasolla projektissa on otettu huomioon värit, tekstit, semantiikka ja responsiivisuus. Värien osalta saavutettavuus on otettu huomioon siten, että käytetyt värit eivät ole räikeitä ja että värien kontrasti on riittävä, esimerkiksi navigaatiopalkin taustan ja linkkien välinen värikontrasti on 5,47:1, joka ylittää WCAG-ohjeistuksen määrittämän vähimmäiskontrastivaatimuksen (4,5:1).

Verkkosivustolla käytetty fontti on haettu Google Fonts -palvelusta, jossa hakutuloksia rajattiin siten, että hakutuloksissa näytettiin ainoastaan sans-serif-fontteja, jotka ovat helppolukuisia. Verkkosivuston tekstien osalta on myös huomioitu, että rivikorkeus on WCAG-ohjeistuksen mukaisesti 1,5 kertaa rivi. Verkkosivuston rivikorkeutta on säädetty CSS:n line-height-ominaisuudella.

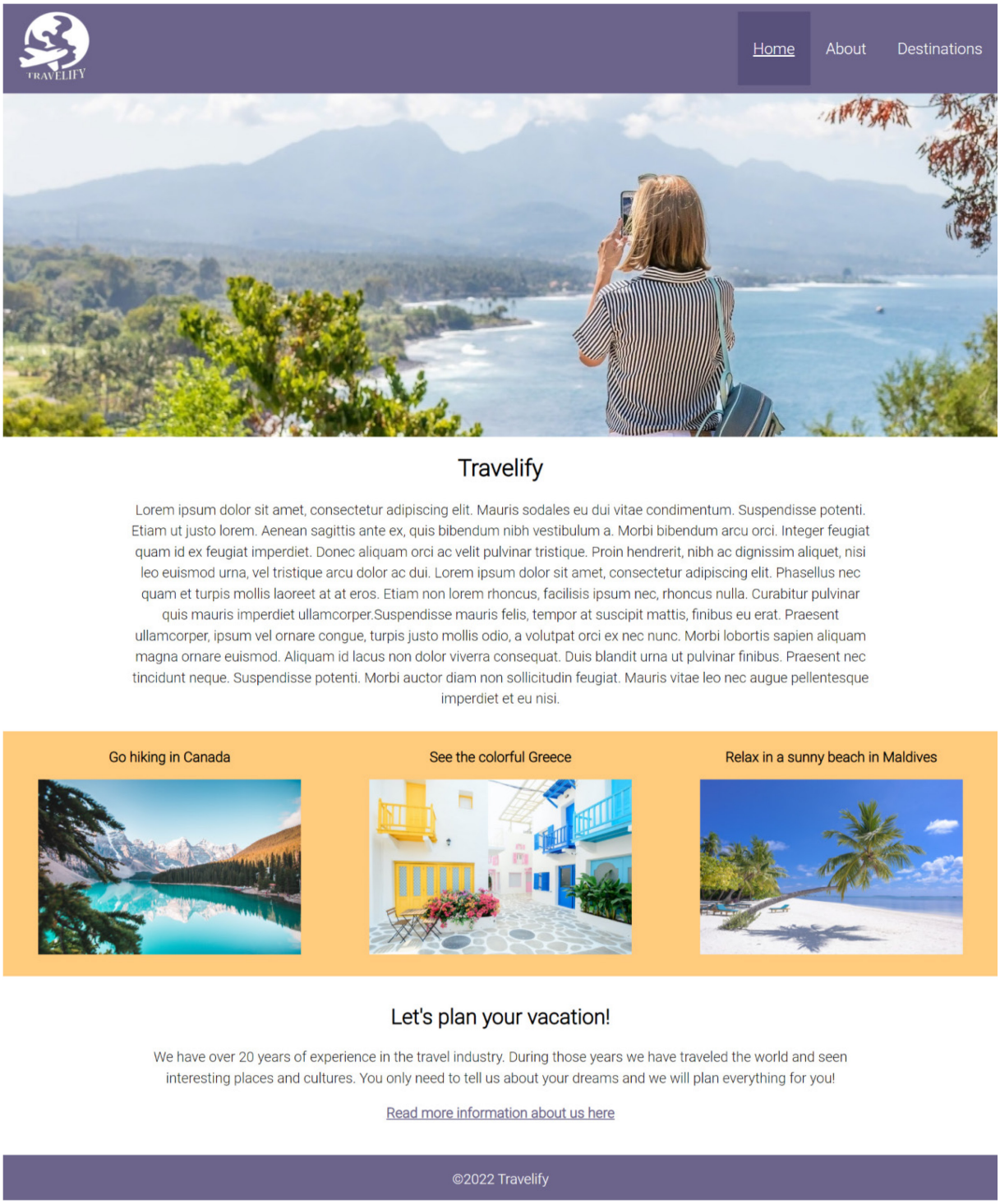
Verkkosivuston lähdekoodissa on käytetty ainoastaan semanttisia HTML-elementtejä, esimerkiksi sivujen pääsisältö on ympäröity main-tageilla, eri osiot pääsisällössä on eroteltu section-elementeillä div:ien sijaan ja painikkeessa on käytetty button-tagia. Näin kaikki

elementit ovat heti ohjelmallisesti tunnistettavissa ja ne sisältävät tarvittavat ominaisuudet. Verkkosivuston lähdekoodissa on myös huomioitu se, että semanttista HTML:ää käytetään oikein eli verkkosivustolla on esimerkiksi käytetty oikeita HTML-tageja oikeaan tarkoitukseen, HTML-elementit on sisennetty oikein – esimerkiksi footeria eli sivuston alatunnistetta ei ole laitettu main-osioon – ja että yksittäisten elementtien hierarkia on oikein eli esimerkiksi otsikkotasot on tehty loogisesti ja ne etenevät portaittain.

Verkkosivustolla on myös kiinnitetty huomiota siihen, että sisällöt etenevät HTML-koodin mukaisesti, eikä CSS-tyylittelyjen avulla ole muutettu lukujärjestystä. Responsiivisuuden osalta on tarkistettu, että fontti on tarpeeksi suurta ja että kaikki elementit ovat sopivan kokoisia ja asettuvat oikein kaikilla päätelaitteilla. Käyttöliittymän muokkaamiseen eri päätelaitteille sopivaksi on käytetty CSS:n media queryjä, joiden avulla tyylejä voi muuttaa riippuen näytön koosta.

Seuraavaksi tutustaan verkkosivuston jokaiseen sivuun erikseen ja katsotaan, minkälaisia ratkaisuja kullakin sivulla on tehty. Kuvassa 13 on verkkosivuston etusivu. Tällä sivulla saavutettavuus on huomioitu edellä mainittujen asioiden lisäksi navigaatiopalkissa, kuvissa ja sivun alaosan linkissä.

Kuva 13 Ohjelmointiprojekti - Home



Verkkosivuston navigoinnin toteuttamiseen on käytetty React Router -nimistä JavaScript-kirjastoa. Ennen kirjaston käyttämistä, se on asennettu projektiin Yarnin avulla, jonka jälkeen se on tuotu kooditiedostoon, jossa sitä käytetään. Ohjelmakoodista 6 nähdään, miten navigointi on toteutettu. Ensimmäinen saavutettavuuden kannalta huomioitava asia on hyppylinkki, joka on ohjelmakoodin rivillä 30. Tässä kohdassa esimerkiksi

ruudunlukuohjelmaa käyttäville henkilöille tarjotaan mahdollisuus siirtyä suoraan sivun pääsisältöön ja ohittaa navigaatiopalkki ja sivun alussa oleva suuri kuva. Sivun navigaatiopalkkiin kuuluva ohjelmakoodi on riveillä 31–38 ja kuten ohjelmakoodista nähdään, navigaatiopalkki täytyy ympäröidä semanttisilla nav-tageilla. Rivillä 32 on navigaation ensimmäinen elementti, joka on kuvitteellisen matkanjärjestäjän logo, joka toimii samalla linkkinä verkkosivuston etusivulle. Tässä kohdassa on huomattava ARIA-määreen käyttäminen. Aria-hidden-määreen tarkoituksena on piilottaa linkkinä toimiva logo ruudunlukuohjelmilta. Piilotuksen syynä on se, että navigaatiossa on peräkkäin kaksi linkkiä etusivulle, joista toinen on sivuston vasemmassa laidassa oleva yrityksen logo ja toinen sivuston oikealla oleva ensimmäinen navigaatiolinkki. Jos ruudunlukuohjelma lukisi käyttäjälleen kaksi samaan paikkaan johtavaa linkkiä peräkkäin, se saattaisi olla hämmentävää. Näkeville käyttäjille navigaatiopalkki on kuitenkin selkeä ja verkkosivustoilla on yleistä, että navigaatiopalkista olevasta logosta on mahdollista navigoida etusivulle. Rivit 33–37 sisältävät navigaatiopalkin varsinaiset navigaatiolinkit, jotka on toteutettu listassa olevina linkkeinä.

Ohjelmakoodi 6 Ohjelmointiprojekti - navigaatiopalkki

```
29    <Router>
30      <a id="skip-to-content-link" href="#mainContent">Skip to main content</a>
31      <nav className = "navbar">
32        <li aria-hidden="true" id="logo"><a href="/"><img alt="logo" id="logo-image" src={logo}/></a></li>
33        <ul>
34          <li><a className = "nav-item" href="/">Home</a></li>
35          <li><a className = "nav-item" href="/about">About</a></li>
36          <li><a className = "nav-item" href="/destinations">Destinations</a></li>
37        </ul>
38      </nav>
39      <Route exact path="/"><HomePage/></Route>
40      <Route exact path="/about" ><About /></Route>
41      <Route exact path="/destinations" ><Destinations /></Route>
42    </Router>
```

Koska käyttäjän sijainnin ilmaiseminen on tärkeää, verkkosivustolla tämä on huomioitu navigaatiopalkin ulkoasussa, välilehdellä näkyvässä tekstissä ja näppäimistön fokuksessa. Navigaatiopalkin osalta käyttäjän nykyinen sijainti ilmaistaan eri tyylisenä navigaatiolinkkinä. Ohjelmakoodista 7 nähdään, että tämä on toteutettu jQuery-nimisellä JavaScript-kirjastolla. Koodissa nykyistä sivua vastaavalle navigaatiolinkille asetetaan väliaikaiset CSS-tyylit, joiden avulla nykyinen sivu erottuu navigaatiopalkin muista navigaatiolinkeistä. Koodin alaosassa näkyy myös CSS-tiedostossa oleva väliaikainen tyylimäärittely navigaatiolinkille.

Ohjelmakoodi 7 Ohjelmointiprojekti - nykyisen sivun ilmaiseminen

```
// Indicate the current page
$(function(){
  $('a').each(function() {
    if ($(this).prop('href') === window.location.href) {
      $(this).addClass('current');
    }
  });
});
```

```
.current{
    color: white;
    text-decoration: underline;
    background-color: #59517c;
}
```

Näppäimistön fokuksen osalta selainten oletustyyliä elementtien korostamiselle ei ole otettu pois tai muutettu. Välilehdellä näkyvän tekstin muokkaamiseen on tässä projektissa valittu React Helmet -niminen ReactJS-komponentti. React Helmet on otettu käyttöön asentamalla se Yarnia käyttäen. Ohjelmakoodista 8 nähdään, miten React Helmetiä on käytetty etusivun koodeissa. Ensin React Helmet on tuotu tiedostoon, jonka jälkeen välilehden teksti on helppo määritellä Helmet-tagien sisällä title-elementtinä. React Helmetiä on hyödynnetty jokaisen sivun kooditiedostossa samalla tavalla.

Ohjelmakoodi 8 Ohjelmointiprojekti - välilehtien otsikot React Helmetillä

```
import {Helmet} from "react-helmet";
```

```
<Helmet>
  <title>Travelify | Home</title>
</Helmet>
```

Ohjelmakoodissa 9 on esimerkki semanttisesta HTML:stä etusivulla. Pääsisältö on main-tagien sisällä ja sivun ensimmäinen osio on ympäröity section-tageilla. Nämä olisi mahdollista toteuttaa myös div-tageja käyttämällä, mutta eivät kerro ohjelmallisesti mitään osioiden tarkoituksesta, eivätkä jäsennä osioita semanttisten elementtien tapaan.

Ohjelmakoodista nähdään myös, että sivulla oleva section-elementti sisältää sivun ensimmäisen otsikon sekä perustekstiä, jotka on tehty käyttäen h1- ja p-tageja.

Ohjelmakoodi 9 Ohjelmointiprojekti - semanttinen HTML

```
<main id="mainContent">
  <section>
    <h1>Travelify</h1>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Suspendisse potenti. Etiam ut justo lorem. Aenean sagitt
      Donec aliquam orci ac velit pulvinar tristique. Proin he
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Curabitur pulvinar quis mauris imperdiet ullamcorper.Sus
      a volutpat orci ex nec nunc. Morbi lobortis sapien aliqu
      Suspendisse potenti. Morbi auctor diam non sollicitudin
    </p>
  </section>
```

Ohjelmakoodista 10 nähdään, miten etusivun kuvaosio on toteutettu. Ohjelmakoodissa kuvat on ympäröity figure-tageilla ja kuville on kirjoitettu kuvateksti figcaptionia käyttäen. Saavutettavuuden kannalta erityisen tärkeää on kirjoittaa kuville vaihtoehtokuvaus, sillä sen avulla käyttäjä, joka ei jostain syystä voi katsoa kuvaa, saa tietää, mitä kuva esittää. Koodissa tämä on tehty alt-attribuutilla riveillä 49, 53 ja 57. Ensimmäisen kuvan kuvatekstissä on esimerkiksi kuvattu maisema, jossa on järvi, metsää ja vuoria Kanadassa.

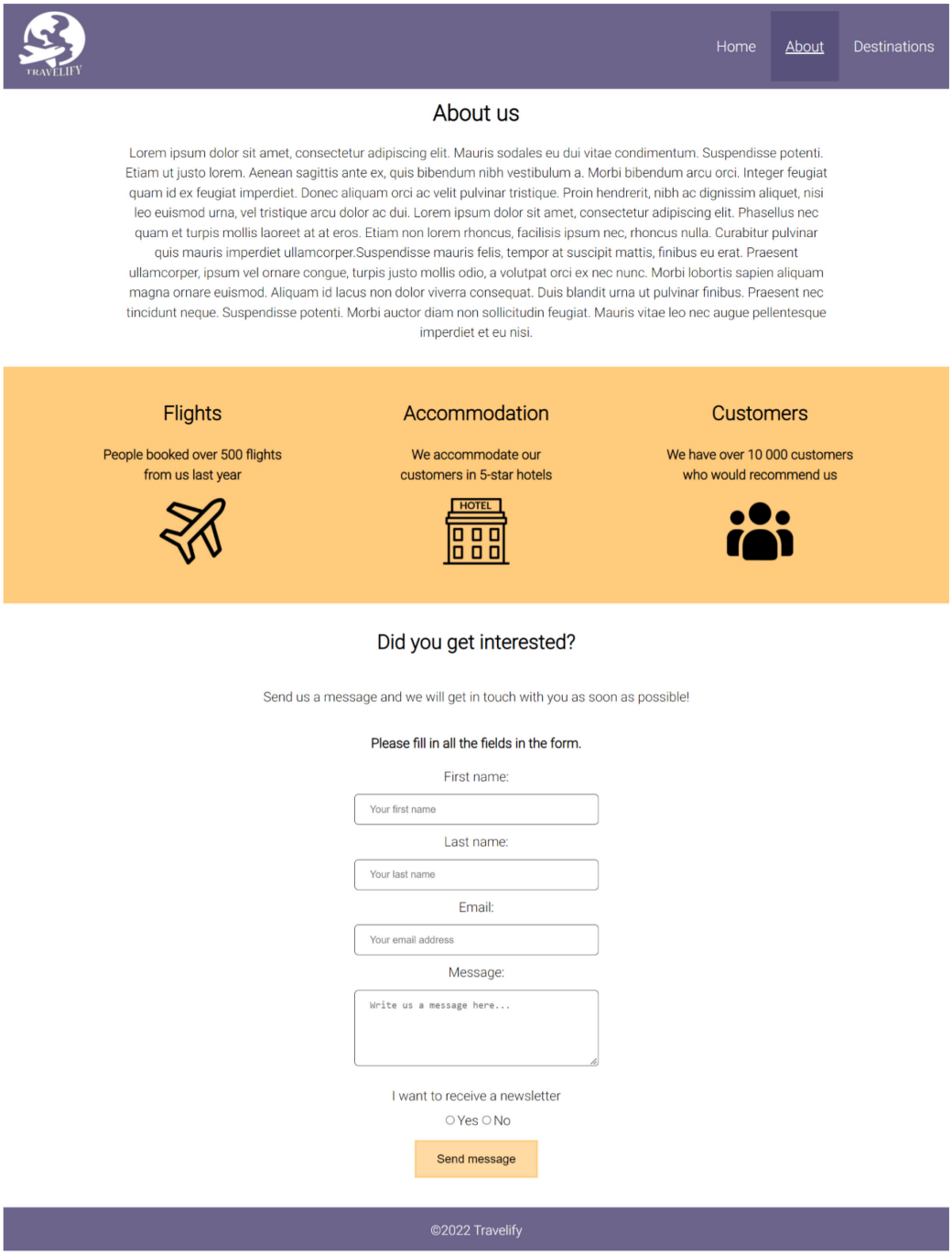
Ohjelmakoodi 10 Ohjelmointiprojekti - alt-teksti

```
46  <section id="image-section">
47    <figure className="figure-home">
48      <figcaption>Go hiking in Canada</figcaption>
49      <img alt="A view with a lake, forest and mountains in Canada" src={canada} />
50    </figure>
51    <figure className="figure-home">
52      <figcaption>See the colorful Greece</figcaption>
53      <img alt="A beautiful courtyard that has two chairs, a table and plants" src={greece} />
54    </figure>
55    <figure className="figure-home">
56      <figcaption>Relax in a sunny beach in Maldives</figcaption>
57      <img alt="Palm trees in a sunny beach in Maldives" src={maldives} />
58    </figure>
59  </section>
```

Etusivun alalaidassa olevan linkin osalta saavutettavuus on huomioitu siten, että linkin teksti on ymmärrettävissä ilman ympäröivää kontekstia ja että linkki erottuu muusta tekstistä sekä väriltään että alleviivauksen avulla. Linkki on tehty HTML:n semanttista a-tagia käyttämällä.

Seuraavaksi tarkastellaan verkkosivustolla olevaa About-sivua, joka näkyy kuvassa 14. Tällä sivulla saavutettavuuden kannalta huomioitavia elementtejä ovat ikonit sekä sivun alaosassa oleva yhteydenottolomake.

Kuva 14 Ohjelmointiprojekti – About



Sivun ikonit on toteutettu samalla tavalla kuin verkkosivuston etusivulla olevat kuvat. Ohjelmakoodissa 11 on ikoniosioon liittyvä koodi. Tämä ei eroa etusivun kuvaosion koodista paljoa, mutta se esitellään tässä figcaptioniin liittyvän eron vuoksi. Tässä tapauksessa figcaption-elementti sisältää h2-otsikon ja tavallista tekstiä p-elementissä eli sen sisälle on siis mahdollista laittaa erilaisia tekstielementtejä. Tässä tekstielementtien tarkoituksena on antaa ikonille otsikko ja siihen liittyvää tietoa, esimerkiksi hotelli-ikonin kohdalla käyttäjälle kerrotaan, että matkailuyritys majoittaa asiakkaitaan viiden tähden hotelleissa.

Ohjelmakoodi 11 Ohjelmointiprojekti - alt-teksti 2

```
<section id="image-section-about-us">
  <figure className="figure-about-us">
    <figcaption>
      <h2>Flights</h2>
      <p>People booked over 500 flights from us last year</p>
    </figcaption>
    <img alt="icon of a plane" src={plane} />
  </figure>
  <figure className="figure-about-us">
    <figcaption>
      <h2>Accommodation</h2>
      <p>We accommodate our customers in 5-star hotels</p>
    </figcaption>
    <img alt="icon of a hotel" src={hotel} />
  </figure>
  <figure className="figure-about-us">
    <figcaption>
      <h2>Customers</h2>
      <p>We have over 10 000 customers who would recommend us</p>
    </figcaption>
    <img alt="icon of customers" src={customer} />
  </figure>
</section>
```

Seuraavaksi tarkastellaan, miten sivulla oleva yhteydenottolomake on tehty ja mitä siinä täytyy ottaa huomioon saavutettavuuden kannalta. Yhteydenottolomaketta tarkastellaan tässä osioittain, mutta lomakkeen koodi on kokonaisuudessaan katsottavissa opinnäytetyön liitteessä 2.

Ohjelmakoodissa 12 on ote syöttökenttiin liittyvästä koodista. Ensimmäiseksi koodista on tärkeää huomata se, että lomakkeet tulee ympäröidä semanttisilla form-tageilla. Kun lomakkeelle aletaan luoda syöttökenttiä, niille täytyy antaa jokin tyyppi, jotta elementti toimii oikein ja sen merkitys on ohjelmallisesti tunnistettavissa. Ohjelmakoodin riveillä 81–83 ja 90–92 on lomakkeen kaksi syöttökenttää, joiden tyyppinä on text. Muissa lomakkeen syöttökentissä on tyyppeinä myös email ja radio. Jokaiselle syöttökentälle täytyy myös tehdä otsikko label-tagilla, joka on sidottu tiettyyn syöttökentään htmlFor-attribuutin avulla. Otsikon avulla ruudunlukijat voivat ilmoittaa syöttökentästä ja samalla käyttäjä saa informaation siitä, mitä syöttökenttään tulisi kirjoittaa. Label-tagin lisäksi syöttökenttään voi laittaa tekstivihjeen (placeholder), mutta se ei voi korvata label-tagia. Ohjelmakoodissa syöttökenttien otsikot ovat riveillä 77 ja 86.

Lomakkeisiin liittyy myös olennaisesti käyttäjän syötteiden validoiminen ja syötteisiin liittyvät virheilmoitukset. Koska tämän opinnäytetyön tarkoituksena on tarkastella koodia saavutettavuuden näkökulmasta, lomakkeella tarkistetaan ainoastaan, onko käyttäjä syöttänyt kenttään mitään ja katsotaan, miten käyttäjälle on mahdollista ilmoittaa virheestä. Oikealla verkkosivustolla käyttäjän syötteelle täytyy tehdä enemmän tarkistuksia. Tässä projektissa validoimiseen on käytetty React Hook Form -kirjastoa. Ennen kyseisen kirjaston käyttämistä se on asennettu projektiin Yarnin avulla, jonka jälkeen se on tuotu kooditiedostoon käyttöä varten. Jotta validoimista voi tehdä, lomakkeen elementit täytyy koodissa rekisteröidä kirjaston mukana tulleeseen useForm-hookiin, jonka avulla hallitaan lomakkeita. Ohjelmakoodissa olevat syöttökentät rekisteröidään riveillä 83 ja 92 ja samalla niille annetaan validointisääntö eli tässä tapauksessa ne asetetaan pakollisiksi kentiksi. Validointisäännön jälkeen on vielä aria-required-määre, joka on kirjoitettu input-elementtiin. Tämä määre viestii esimerkiksi ruudunlukijoille, että kenttä on pakollinen.

Kun käyttäjä lähettää lomakkeen, syöttökentät tarkistetaan ja mahdollisista virheistä ilmoitetaan käyttäjälle. Ohjelmakoodin riveillä 82 ja 91 on käytetty aria-invalid-määrettä, jolla esimerkiksi ruudunlukijoille voidaan viestiä virheellisestä syötteestä. Tämän lisäksi lomake sisältää logiikkaa visuaaliseen virheilmoitukseen riveillä 78–80 ja 87–89. Näillä riveillä tarkistetaan, onko syöttökentistä löytynyt virhe ja tarvittaessa näytetään virheilmoitus.

Ohjelmakoodi 12 Ohjelmointiprojekti - lomake

```
76   <form onSubmit={handleSubmit(onSubmit)}>
77     <label htmlFor="firstName">First name:</label><br />
78     {errors.firstName && errors.firstName.type === "required" && (
79       <p role="alert" className="alert">*This is a required field</p>
80     )}
81     <input placeholder="Your first name" type="text" id="firstName"
82       aria-invalid={errors.firstName ? "true" : "false"}
83       {...register('firstName', { required: true })} aria-required="true"/>
84     <br />
85
86     <label htmlFor="lastName">Last name:</label><br />
87     {errors.lastName && errors.lastName.type === "required" && (
88      <p role="alert" className="alert">*This is a required field</p>
89     )}
90     <input placeholder="Your last name" type="text" id="lastName"
91      aria-invalid={errors.lastName ? "true" : "false"}
92      {...register('lastName', { required: true })} aria-required="true"/>
```

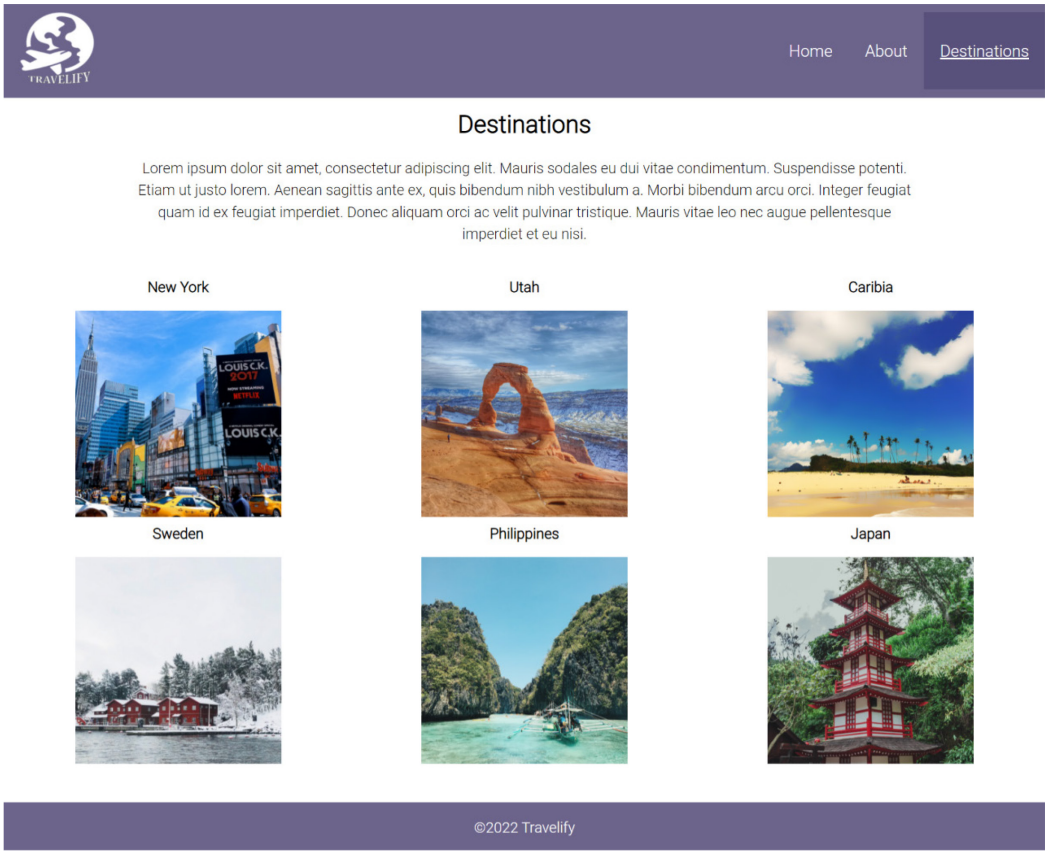
Ohjelmakoodissa 13 tarkastellaan lomakkeella olevaa radiopainikeryhmää. Saavutettavuuden kannalta toisiinsa liittyvistä radiopainikkeista on hyvä muodostaa ryhmä ympäröimällä elementit fieldset-tageilla. Tällaiselle ryhmälle voidaan antaa otsikko käyttämällä legend-tagia. Koodin riveillä 117 ja 122 luodaan radiopainikkeet ja niille annetaan tyypiksi radio. Radiopainikkeet rekisteröidään ohjelmakoodin 12 tapaan useForm-hookiin riveillä 119 ja 124, jonka jälkeen niiden arvo on mahdollista tarkistaa. Myös syötteen tarkistaminen ja virheistä ilmoittaminen tapahtuu samalla tavalla kuin ohjelmakoodissa 12 . Tässä tapauksessa virheiden tarkistaminen ja virheilmoituksen näyttäminen tapahtuu riveillä 114–116.

Ohjelmakoodi 13 Ohjelmointiprojekti - lomake 2

```
112     <fieldset>
113       <legend>I want to receive a newsletter</legend>
114       {errors.newsLetter && errors.newsLetter.type === "required" && (
115         <p role="alert" className="alert">*This is a required field</p>
116       )}
117       <input type="radio" name="newsletter" id="yes" value="Yes"
118       aria-invalid={errors.newsletter ? "true" : "false"}
119       {...register('newsLetter', { required: true })} aria-required="true"/>
120       <label htmlFor="yes">Yes</label>
121
122       <input type="radio" name="newsletter" id="no" value="No"
123       aria-invalid={errors.newsletter ? "true" : "false"}
124       {...register('newsLetter', { required: true })}/>
125       <label htmlFor="no">No</label>
126     </fieldset>
```

Kuvassa 15 on verkkosivuston Destinations-sivu. Tämä sivu on tehty verkkosivustolle täydentämään kokonaisuutta. Saavutettavuus on huomioitu tällä sivustolla edellisten sivujen tapaan otsikoinnissa, tekstirivien korkeudessa sekä kuvissa. Sen lisäksi on huomioitu, että sivu on responsiivinen eli elementit näkyvät oikein eri päätelaitteilla.

Kuva 15 Ohjelmointiprojekti - Destinations



8.2 Saavutettavuuden testaaminen projektissa

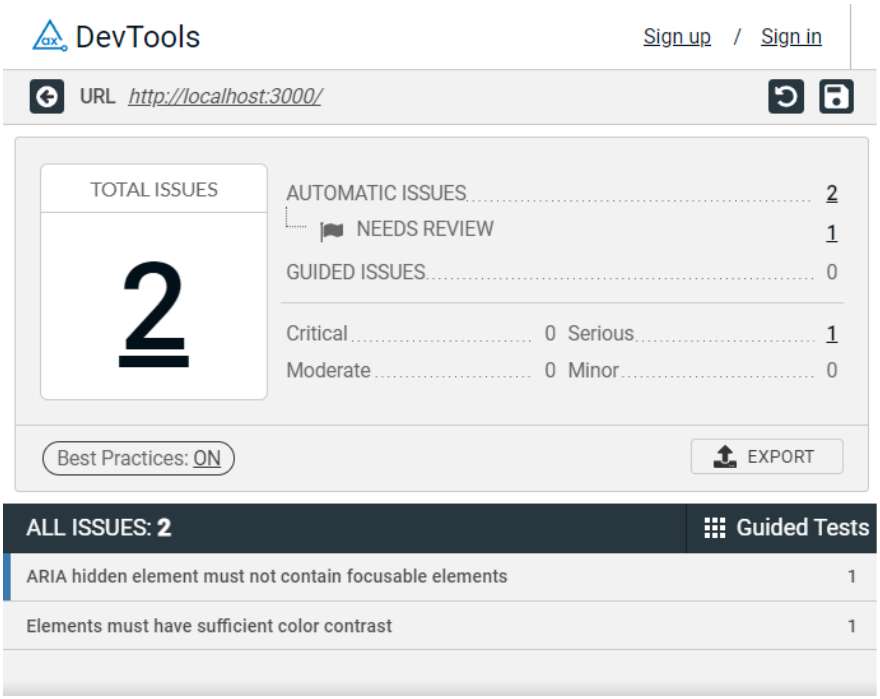
Seuraavaksi katsotaan, miten opinnäytetyön aikana toteutettua verkkosivustoa on testattu. Automaattisten tarkastustyökalujen osalta testaukseen on valittu axe DevTools ja WAVE. Manuaalisessa testauksessa sivustoa on testattu näppäimistöllä ja NVDA-ruudunlukuohjelmalla. Tämän lisäksi sivuston toimivuus on tarkistettu eri selaimissa ja eri päätelaitteilla. Manuaalisessa testauksessa hyödynnettiin myös Accessibility Insights -nimistä työkalua. Opinnäytetyön aikana ei ollut mahdollista toteuttaa käyttäjätestausta.

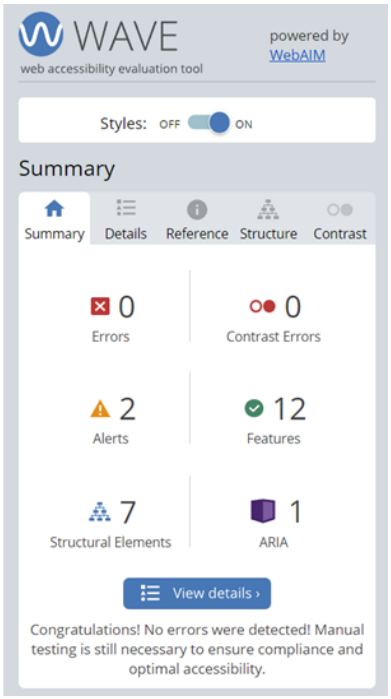
Testausta alettiin ensin suorittaa automaattisilla tarkastustyökaluilla. Kuvassa 16 on verkkosivuston etusivulta saadut testitulokset. Kuvan vasemmalla puolella olevat testitulokset ovat axe DevToolsista. Tämä työkalu löysi sivulta kaksi virhettä, jotka liittyivät värikontrastiin ja aria-hidden-määreen käyttämiseen. Värikontrastiin liittyvä virhe johtuu sivun alussa olevasta hyppylinkistä. Kun fokus ei ole hyppylinkissä, sen tekstin väri ja taustaväri ovat läpinäkyviä, mikä aiheuttaa virheilmoituksen värikontrastista. Kun käyttäjän fokus siirtyy hyppylinkkiin, se tulee näkyväksi ja siinä olevan taustavärin ja tekstin välinen kontrasti on hyvä. Aria-määreeseen liittyvä virhe johtuu navigaatiopalkissa olevasta logosta, joka on piilotettu ruudunlukijoilta. Virhe johtui siitä, että elementtiin oli mahdollista saada fokus näppäimistöllä. Testauksen jälkeen virhe korjattiin lisäämällä elementtiin tabIndex-määrittely, jolloin kyseinen elementti ei ollut enää fokusoitavissa näppäimistöllä. Tämä ei ole saavutettavuuden kannalta ongelma, sillä käyttäjät voivat siirtyä verkkosivuston etusivulle navigaatiopalkissa olevasta navigaatiolinkistä.

WAVE-tarkastustyökalu ei havainnut etusivulla virheitä, mutta antoi kaksi huomautusta, jotka täytyy tarkistaa. Ensimmäisen huomautuksen mukaan navigaatiopalkissa olevan logon alt-teksti on epäilyttävä, mikä tarkoittaa siis sitä, että se ei ole laadullisesti hyvä. Tarkistushetkellä logon alt-teksti oli ”logo”, joka testaustuloksen myötä muutettiin muotoon ”logo of the company that contains a globe and a plane that is flying around it”. Muutoksen jälkeen logo ei aiheuttanut enää huomautusta. Toinen saatu huomautus liittyy projektin luomisessa tulleeseen noscript-elementtiin. Tarkastustyökalun mukaan skriptatun sisällön eli JavaScriptillä luodun sisällön sekä noscript-elementissä olevan sisällön täytyy olla saavutettavaa. Noscript-elementin sisältö näytetään siis käyttäjille, joilla ei ole JavaScriptiä

käytössä selaimessaan. Tämän projektin aikana noscript-elementtiin ei lisätty sisältöä, mutta saavutettavuuden kannalta oikeissa projekteissa sinne kannattaa lisätä.

Kuva 16 Projektin testaus - axe DevTools ja WAVE

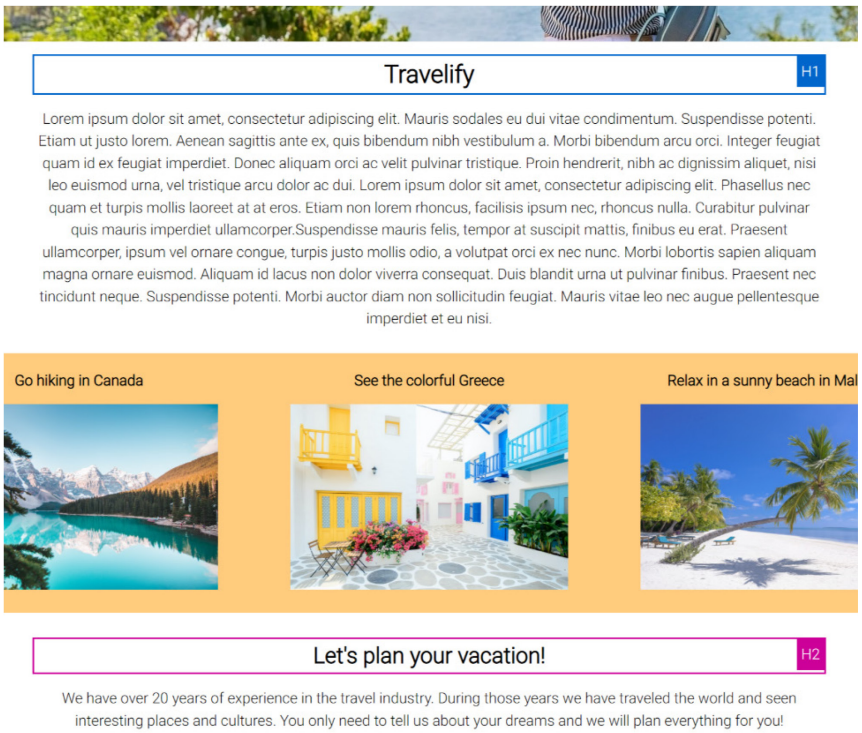




Verkkosivuston saavutettavuustestaus aloitettiin siis sivuston etusivulta axe DevToolsilla, jonka jälkeen testausta tehtiin WAVE:lla. Axe DevToolsilla tehdyn testauksen jälkeen sivulle ei tehty muutoksia eli WAVE:lla tehty testaus tapahtui täsmälleen samalle ohjelmakoodille. Molemmat työkalut antoivat tästä huolimatta täysin eri tulokset. Muilla sivuilla WAVE ei havainnut virheitä tai antanut huomautuksia, mutta axe DevToolsilla saatiin virheilmoitus sivujen alussa olevan hyppylinkin värikontrastista. Testausta kannattaa siis suorittaa useammalla kuin yhdellä työkalulla, jotta saavutettavuuteen liittyvät ongelmat havaitaan paremmin.

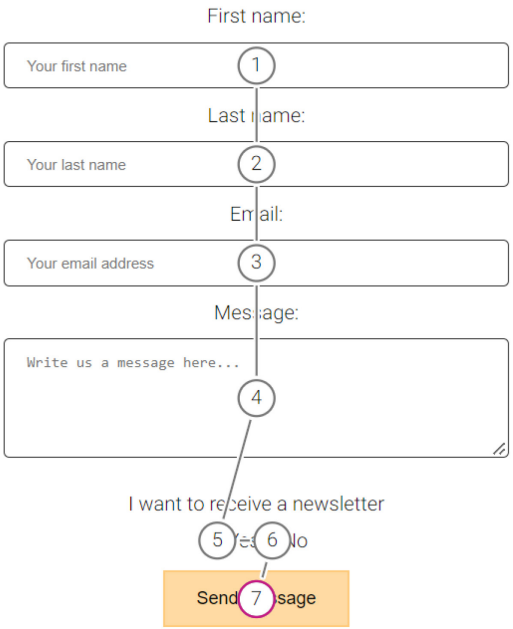
Testauksessa hyödynnettiin myös Accessibility Insights -nimistä työkalua, jonka avulla tarkastettiin sivujen otsikkotasot ja fokusjärjestys. Verkkosivuston otsikoissa ei havaittu ongelmia testauksen aikana. Kuvasta 17 nähdään, miten otsikkotasoja voi tarkistaa Accessibility Insightsin avulla. Työkalu tuo esille eri otsikkotasot todella selkeästi, mikä helpottaa varsinkin monitasoisten otsikkorakenteiden tarkistamista.

Kuva 17 Projektin testaus - otsikkotasot



Verkkosivuston fokusjärjestyksen testaamisessa ei sivustolla ilmennyt ongelmia eli kaikki sivuston toiminnot oli mahdollista saavuttaa näppäimistöllä ja fokusjärjestys vaikutti loogiselta. Kuvassa 18 on tarkistettu yhteydenottolomakkeen fokusjärjestys. Kuvasta nähdään, että Application Insights havainnollistaa fokusjärjestyksen todella selkeästi, mikä tekee tarkistustyöstä miellyttävää.

Kuva 18 Projektin testaus - fokusjärjestys



Avustavien teknologioiden osalta testaamisessa käytettiin NVDA-nimistä ruudunlukuohjelmaa, jonka avulla oli mahdollista tarkistaa, miltä verkkosivusto kuulostaa luettuna ja ovatko eri rakenteet, kuten navigaatiopalkki, yhteydenottolomake ja kuvaosiot loogisia. Testaamisen aikana todettiin, että vaikka ruudunlukuohjelman käyttämisestä ei olisi aikaisemmin kokemusta, on sillä kuitenkin mahdollista tehdä jonkin tasoinen arvio sivuston käytettävyydestä. Testaamisen aikana verkkosivuston yleinen käytettävyys vaikutti hyvältä. Ruudunlukuohjelmasta on myös hyötyä kehitystyössä, esimerkiksi yhteydenottolomakkeen ohjelmoinnissa käytettiin apuna ruudunlukuohjelmaa, jotta voitiin samalla havainnoida, mitä muutoksia koodiin täytyy tehdä, jotta lomake on käyttökelpoinen.

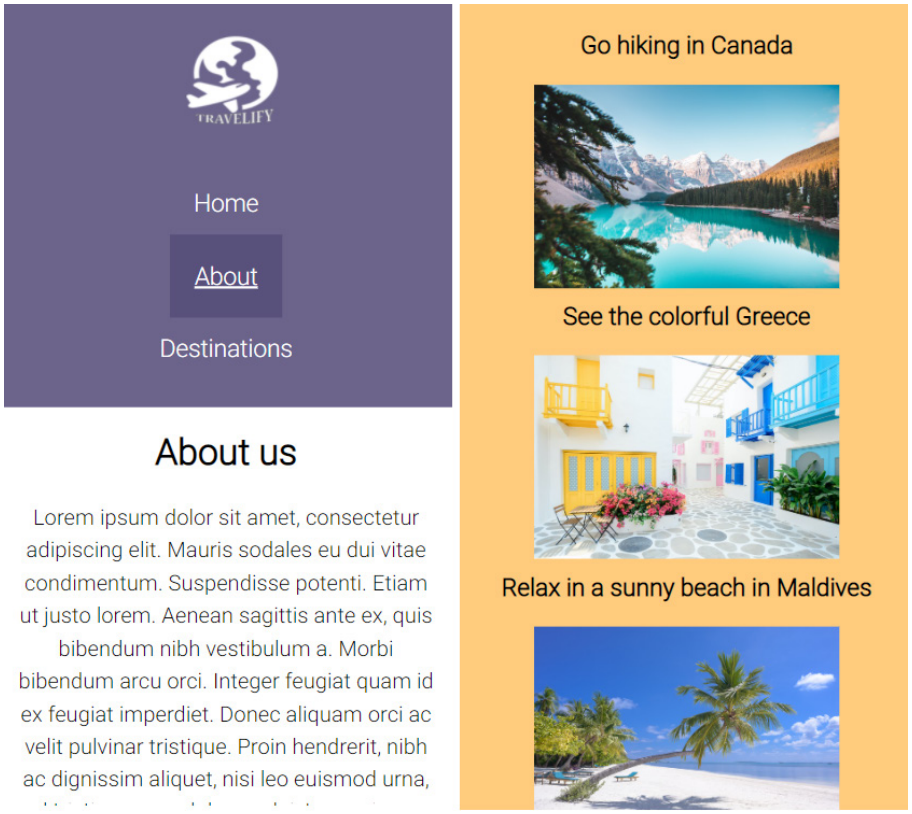
Koska toimivuuden testaaminen eri selaimilla on tärkeää, verkkosivustoa testattiin Google Chromella, Mozilla Firefoxilla ja Microsoft Edgellä. Testauksessa tuli ilmi, että elementtien saama fokus ei ollut riittävä Mozilla Firefoxissa. Testaustuloksen perusteella verkkosivuston CSS-tiedostoon lisättiin Mozilla Firefox -selainta koskeva tyylimäärittely, jonka avulla fokuksen tyyli muutettiin. Kuvan 19 yläosassa näkyy Mozilla Firefoxin automaattinen fokustyyli ja kuvan alaosassa muutosten jälkeinen lopputulos, joka on huomattavasti käyttäjäystävällisempi. Tämän lisäksi testauksen aikana huomattiin, että Mozilla Firefoxissa sivuston fontti oli erilainen kuin muissa selaimissa. Saavutettavuuden kannalta se ei kuitenkaan ole ongelma, sillä verkkosivustolle on valittu useita fonttivaihtoehtoja siitä syystä, että ensisijainen fontti ei välttämättä toimi kaikissa selaimissa.

Kuva 19 Projektin testaus - fokus

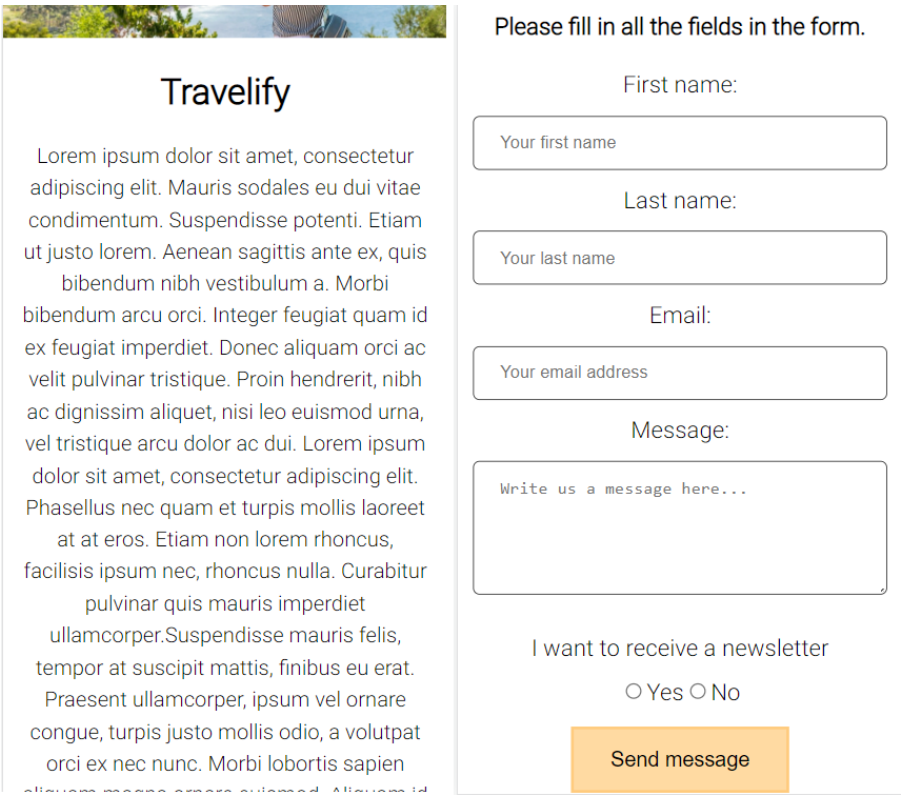
Selainten lisäksi testattiin, miltä verkkosivusto näyttää eri päätelaitteilla. Testaamisen aikana elementit asettuivat näytöllä hyvin ja olivat tarpeeksi suuria. Kuvasta 20 nähdään, että esimerkiksi navigaatiopalkin linkit ja etusivun kuvaosion kuvat ovat asettuneet päällekkäin puhelimen ruudulla, mikä on mobiilinäkymille tyypillistä. Tietokoneella nämä ovat

vierekkäin, koska näytöllä on tilaa enemmän. Testaamisen aikana tehtiin kuitenkin saavutettavuuteen liittyviä huomioita. Kuvan 21 vasemmalla puolella nähdään, etusivulla oleva teksti vaikuttaa erittäin pitkältä. Tekstin luettavuutta voisi parantaa esimerkiksi lisäämällä tekstiin kappalejaon. Toinen huomio tehtiin kuvan oikealla puolella olevassa yhteydenottolomakkeessa. Lomakkeen radiopainikkeet saattavat olla liian lähellä toisiaan sekä liian lähellä lomakkeen lähetyspainiketta, jolloin käyttäjä voi vahingossa valita väärän radiopainikkeen tai lähettää lomakkeen, vaikka radiopainiketta ei ole valittu.

Kuva 20 Projektin testaus - mobiilinäkymä



Kuva 21 Projektin testaus - mobiilinäkymä 2



9 Pohdinta

Saavutettavuus käsitteenä on yksinkertainen ja helppo ymmärtää, mutta kun siihen liittyviä asioita aletaan tutkia tarkemmin, huomataan pian, kuinka laajasta asiasta on oikeasti kyse. Saavutettavuuden kannalta täytyy ottaa huomioon sisällöllisiä ja teknisiä asioita, mikä tarkoittaa sitä, että kehitystiimistä täytyy löytyä laajasti tietoa erilaisilta aloilta. Joissakin verkkosivustoprojekteissa on pakko noudattaa saavutettavuuteen liittyvää lainsäädäntöä, jolloin tuloksena pitäisi syntyä saavutettavia sivustoja, mutta mitä tapahtuu niille projekteille, joissa ei lain mukaan tarvitse seurata WCAG-ohjeistusta? Onko projektiin esimerkiksi varattu varmasti tarpeeksi aikaa ja rahaa saavutettavuuden varmistamiseksi? Verkkosivustoja on myös mahdollista tehdä erilaisilla kotisivukoneilla, joiden käyttäminen ei vaadi ohjelmointikokemusta. Henkilöillä, jotka tekevät itse omalle yritykselleen kotisivut tällaista palvelua käyttäen, ei välttämättä ole lainkaan ohjelmistokehitykseen tai verkkosivustoihin liittyvää tietoa. Saavutettavuuden kannalta tällaiset verkkosivustot saattavatkin olla ongelmallisia. Tämän opinnäytetyön aikana ei ole tutkittu kotisivukoneita, mutta olisi mielenkiintoista tietää, miten niiden käyttäjiä on informoitu saavutettavuudesta.

Vaikka kaikissa verkkosivustoprojekteissa noudatettaisiin WCAG-ohjeistusta lain vaatimalla tasolla, se ei välttämättä tarkoita sitä, että tuloksena syntyy saavutettavia verkkosivustoja. Jotkin ohjeistuksen kriteereistä ovat vaikeita tulkita ja niitä on mahdollista täyttää eri tavoin, jolloin tulokset saattavat olla vaihtelevia. Tämän lisäksi WCAG-ohjeistus ei ota juurikaan kantaa verkkosivustojen kieleen ja sen ymmärrettävyyteen tai verkkosivuston käytettävyyteen, vaikka ne liittyvät olennaisesti saavutettavuuteen. Mikäli on mahdollista, tulevaisuuden kannalta olisi tärkeää tehdä lain vaatima kriteeristö esimerkiksi kieleen ja käytettävyyteen liittyen, jotta saavutettavuuden kaikki osa-alueet tulisivat varmasti huomioiduiksi, sillä vaikka verkkosivusto olisi teknisesti saavutettava, verkkosivustolla vieraileminen ei tuota käyttäjälle arvoa, jos sen sisältö ei ole ymmärrettävissä.

Jotta verkkosivusto olisi saavutettava, kehitystyön aikana tarvitaan esimerkiksi graafisia suunnittelijoita, palvelumuotoilijoita sekä ohjelmistokehittäjiä. Tämän opinnäytetyön aikana tehty ohjelmointiprojekti on auttanut ymmärtämään erityisesti laaja-alaisen osaamisen tarvetta verkkosivustojen kehityksessä. Vaikka projektin aikana kehitetty verkkosivusto on todella yksinkertainen, kehitystyön aikana täytyi esimerkiksi miettiä, minkälainen fontti

näyttäisi hyvältä ja olisi helppo lukea. Tämän lisäksi täytyi valita sopivat värit, suunnitella sivupohjan rakenne ja ohjelmoida kaikki siten, että sivusto olisi käytettävissä hiiren lisäksi esimerkiksi pelkällä näppäimistöllä ja ruudunlukuohjelmalla. Ohjelmoinnin näkökulmasta verkkosivuston tekeminen siten, että miettii kaikessa tekemisessään saavutettavuutta, vie aikaa hieman enemmän verrattuna siihen, että ohjelmoisi miettimättä saavutettavuutta. Jos saavutettavuus on kuitenkin jatkuvasti osa ohjelmoijan työpäivää, ei sitä tarvitse välttämättä miettiä enää niin paljon esimerkiksi semanttisten HTML-tagien ja ARIA-määreiden osalta, sillä niiden käyttäminen oikein tapahtuu automaattisesti, kun niihin on tutustunut ja niitä on käyttänyt tarpeeksi useasti. WCAG-ohjeistuksen lain vaatimien A- ja AA-tasojen kriteerien osalta huomattiin, että niiden noudattaminen ei ole erityisen vaikeaa, mutta on joitakin tapauksia, joissa niiden noudattaminen saattaa viedä hieman aikaa. Ilman ohjelmointiprojektia ja sen tuomaa kokemusta, ohjelmistokehityksen näkökulma aiheeseen olisi jäänyt vaillinaiseksi, sillä projekti toi uusia näkökulmia ohjelmoijan työhön ja tämän lisäksi projektista saatiin tulevaisuuden kannalta arvokasta kokemusta.

Opinnäytetyön kirjoittamisen aikana on myös pohdittu saavutettavuuden näkyvyyttä yhteiskunnassamme. Mielestäni yhteiskunnassamme jokainen tietää, mitä esteettömyys tarkoittaa, sillä siihen liittyvät asiat ovat kaikkien nähtävissä. Esimerkiksi rakennuksissa on rappusten lisäksi luiskia kulkemisen helpottamiseksi ja parkkipaikoilla leveämpiä parkkiruutuja liikkumisesteisten henkilöiden käyttöön. Saavutettavuus on kuitenkin asia, jota ei ole välttämättä samalla tavalla mahdollista nähdä. Tämän vuoksi saavutettavuuteen liittyvää keskustelua tulisi käydä julkisesti enemmän kuin tällä hetkellä, sillä mielestäni saavutettavuutta ei ole tuotu yhteiskunnassamme nyt samalla tavalla esille kuin esteettömyyttä. Kuinka moni esimerkiksi tietää ja osaa kertoa, mitä verkkosivustoilla esiintyvä saavutettavuusseloste sisältää tai mitkä tekijät vaikuttavat verkkosivuston saavutettavuuteen? Saavutettavuusongelmiin saatettaisiin kiinnittää vielä enemmän huomiota, jos ihmiset olisivat tietoisempia niistä.

10 Yhteenveto

Tämän opinnäytetyön alussa esitettiin neljä tutkimuskysymystä, jotka muodostettiin siten, että saavutettavuutta tutkittaisiin eri näkökulmista. Tässä opinnäytetyössä on selvitetty, mitä saavutettavuus tarkoittaa ja miksi se on tärkeää. Tämän lisäksi on tutkittu verkkosivustoille asetettuja saavutettavuusvaatimuksia ja miten niitä on mahdollista täyttää. Teknisen puolen osalta on tutkittu, miten saavutettavan verkkosivuston voi käytännössä ohjelmoida ja miten verkkosivuston saavutettavuutta voi testata. Mielestäni tutkimuskysymyksiin on onnistuttu vastaamaan hyvin, ja opinnäytetyön lukemalla lukija saa hyvän yleiskäsityksen saavutettavuudesta ja siihen liittyvistä asioista.

Ennen opinnäytetyön tekemistä tiesin saavutettavuudesta joitakin asioita. Siitä huolimatta olen oppinut todella paljon ja alkanut ymmärtää isoja asiakokonaisuuksia yksittäisten huomioiden sijaan. Ohjelmointiprojektin aikana teoriaosassa esitetyt asiat ovat vahvistuneet ja olen ymmärtänyt, mitä saavutettavuuden huomioon ottaminen vaatii verkkosivustojen kehityksessä.

Tämän opinnäytetyön aikana olen saanut myös paremman käsityksen siitä, kuinka suuri merkitys saavutettavuudella on. Koska tänä päivänä internetin käyttäminen kuuluu ihmisten arkeen, on verkkosivustojen suunnittelussa ja kehityksessä kiinnitettävä erityistä huomioita saavutettavuuteen. Mielestäni jokaisen verkkosivustojen kehityksen parissa työskentelevän henkilön täytyy ymmärtää saavutettavuuden merkitys ja tehdä parhaansa sen eteen, jotta kaikki pystyisivät käyttämään internetiä tasa-arvoisesti. Tämän opinnäytetyön jälkeen haluan kehittää saavutettavuusosaamistani ja hyödyntää sitä työelämässä.

Lähteet

Komission täytäntöönpanopäätös (EU) 1523/2018. https://eur-lex.europa.eu/legal-content/FI/TXT/HTML/?uri=CELEX:32018D1523&from=EN#d1e127-103-1

Acharya, D. (2021). *The 40 Best JavaScript Libraries and Frameworks for 2022*. Kinsta. https://kinsta.com/blog/javascript-libraries/

Agenda Helsinki. (2019). *Digitaalisen palvelun käytettävyystestaus – mitä, miksi ja miten?* https://agendahelsinki.fi/2019/08/08/kaytettavyystestaus-mita-miksi-miten/

Arancio, S. (2021). *What is JSX?* Medium. https://medium.com/@sjarancio/what-is-jsx-e3dda0af3490

Babel. (n.d.). *What is Babel?* https://babeljs.io/docs/en/

BrowserStack. (2021). *Top 5 CSS Frameworks for Developers and Designers*. https://www.browserstack.com/guide/top-css-frameworks

Celia. (2022). *Tietoa ja ohjeita saavutettavuudesta*. Haettu 11.2.2022 osoitteesta Saavutettavasti https://www.saavutettavasti.fi/verkkosisaltojen-saavutettavuus/linkkeja-ja-tyokaluja/

Celia. (n.d. -a). *WCAG*. Haettu 22.1.2022 osoitteesta Saavutettavasti https://www.saavutettavasti.fi/verkkosisaltojen-saavutettavuus/wcag/

Celia. (n.d. -b). *Selkeät rakenteet*. Saavutettavasti. https://www.saavutettavasti.fi/verkkosisaltojen-saavutettavuus/selkeat-rakenteet/

Celia. (n.d. -c). *Yleistä tietoa saavutettavuudesta*. https://www.celia.fi/saavutettavuus/

Code Institute. (2022). *What is a JavaScript Library?* https://codeinstitute.net/global/blog/what-is-a-javascript-library/

Etelä-Suomen aluehallintovirasto. (n.d. -a). *Kenelle saavutettavuus on tärkeää?* Saavutettavuusvaatimukset. https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/

Etelä-Suomen aluehallintovirasto. (n.d. -b). *Digipalvelulain vaatimukset*. Haettu 12.2.2022 osoitteesta Saavutettavuusvaatimukset https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/

Etelä-Suomen aluehallintovirasto. (n.d. -c). *Mitä palveluja ja sisältöjä laki koskee?* Haettu 22.1.2022 osoitteesta Saavutettavuusvaatimukset https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/mita-palveluja-ja-sisaltoja-laki-koskee/

Etelä-Suomen aluehallintovirasto. (n.d. -d). *WCAG 2.1: lain vaatimukset*. Haettu 22.1.2022 osoitteesta Saavutettavuusvaatimukset: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/

Etelä-Suomen aluehallintovirasto. (n.d. -e). *Soveltamisala: kuulummeko lain piiriin?* Haettu 22.1.2022 osoitteesta Saavutettavuusvaatimukset: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/soveltamisala-kuulummeko-lain-piiriin/

Etelä-Suomen aluehallintovirasto. (n.d. -f). *Tietoa saavutettavuusselosteesta*. Haettu 22.1.2022 osoitteesta Saavutettavuusvaatimukset: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-saavutettavuusselosteesta/

Etelä-Suomen aluehallintovirasto. (n.d. -g). *Tietoa WCAG-ohjeistuksesta*. Saavutettavuusvaatimukset. https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/

Etelä-Suomen aluehallintovirasto. (n.d. -h). *Yleistä saavutettavuudesta*. Saavutettavuusvaatimukset. https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/

freeCodeCamp. (2020). *CSS Preprocessors Explained*. https://www.freecodecamp.org/news/css-preprocessors/

GeeksforGeeks. (2021). *Web Development*. https://www.geeksforgeeks.org/web-development/

Haanperä, T. (n.d.). *Saavutettavuus verkkopalveluissa*. https://blogs.helsinki.fi/klaara-network/files/2019/06/Saavutettavuus-Tapio-Haanpera%CC%88.pdf

Kaur, A. (2018). *Accessibility guidelines for UX Designers*. UX Collective. https://uxdesign.cc/accessibility-guidelines-for-a-ux-designer-c3ba775539be

Kehitysvammaliitto ry. (n.d. -a). *Näköön liittyvät rajoitteet*. Papunet. https://papunet.net/saavutettavuus/nakoon-liittyvat-rajoitteet

Kehitysvammaliitto ry. (2021a). *Kytkimet ohjaimina*. Papunet. https://papunet.net/tietoa/kytkimet-ohjaimina

Kehitysvammaliitto ry. (2021b). *Katseohjaus*. Papunet. https://papunet.net/tietoa/katseohjaus

Kehitysvammaliitto ry. (n.d. -b). *Kuuloon liittyvät rajoitteet*. Papunet. https://papunet.net/saavutettavuus/kuuloon-liittyvat-rajoitteet

Kehitysvammaliitto ry. (n.d. -c). *Fyysiset ja motoriset rajoitteet*. Papunet. https://papunet.net/saavutettavuus/fyysiset-ja-motoriset-rajoitteet

Kehitysvammaliitto ry. (n.d. -d). *Pallohiiri (suuri)*. Papunet. https://papunet.net/saavutettavuus/pallohiiri-suuri

Kehitysvammaliitto ry. (n.d. -e). *Silmän liikkeillä ohjattava kohdistin*. Papunet. https://papunet.net/saavutettavuus/silman-liikkeilla-ohjattava-kohdistin

Kehitysvammaliitto ry. (n.d. -f). *Kuka hyötyy saavutettavuudesta?* Papunet. https://papunet.net/saavutettavuus/kuka-hyotyy-saavutettavuudesta

Kehitysvammaliitto ry. (n.d. -g). *Käytä selkeää ja ymmärrettävää kieltä*. Papunet. https://papunet.net/saavutettavuus/kayta-selkeaa-ja-ymmarrettavaa-kielta

Kehitysvammaliitto ry. (n.d. -h). *Tekstin koko ja kirjasintyyppi*. Papunet. https://papunet.net/saavutettavuus/tekstin-koko-ja-kirjasintyyppi

Kehitysvammaliitto ry. (n.d. -i). *Suunnittele selkeät sivupohjat*. Papunet. https://papunet.net/saavutettavuus/suunnittele-selkeat-sivupohjat

Kehitysvammaliitto ry. (n.d. -j). *Looginen lukemisjärjestys*. Papunet. https://papunet.net/saavutettavuus/looginen-lukemisjarjestys

Kehitysvammaliitto ry. (n.d. -k). *Saavutettavien verkkosivujen suunnitteluopas*. Papunet. https://papunet.net/saavutettavuus/saavutettavien-verkkosivujen-suunnitteluopas

Kehitysvammaliitto ry. (n.d. -l). *Saavutettavuuden testaaminen itse*. Papunet. https://papunet.net/saavutettavuus/saavutettavuuden-testaaminen-itse

Kehitysvammaliitto ry. (n.d. -m). *Käyttäjätestaaminen* [kuva]. https://papunet.net/saavutettavuus/kayttajatestaaminen

Kehitysvammaliitto ry. (n.d. -n). *Erilaiset kytkinohjaimet* [kuva]. https://papunet.net/saavutettavuus/kytkinohjaimet

Kehitysvammaliitto ry. (n.d. -o). *Katseohjausyksikkö* [kuva]. https://papunet.net/tietoa/katseohjaus

Kolade, C. (2021). *HTML vs JSX – What's the Difference?* Haettu 2.2.2022 osoitteesta freeCodeCamp https://www.freecodecamp.org/news/html-vs-jsx-whats-the-difference/

Kozłowski, M. (2020). *What is React? Is it a framework, and why should you care?* Startup Development House. https://start-up.house/en/blog/articles/what-is-react

Kuntaliitto. (n.d.). *7. Verkkopalveluiden saavutettavuuden testaaminen*. https://www.kuntaliitto.fi/tietotuotteet-ja-palvelut/verkko-oppaat/saavutettavuusopas/7-verkkopalveluiden-saavutettavuuden-testaaminen

Kuuloliitto ry. (n.d. -a). *Kuulo*. https://www.kuuloliitto.fi/kuulo/

Kuuloliitto ry. (n.d. -b). *Kuulovammat*. https://www.kuuloliitto.fi/kuulovammat/

Kuuloliitto ry. (n.d. -c). *Kuulokojeet ja apuvälineet*. https://www.kuuloliitto.fi/kuulo/kuulokojeet-ja-apuvalineet/

Laak, T. (2006a). *Saavutettavaa typografiaa – Osa 1*. Saavutettava.fi. https://saavutettava.fi/2006/03/24/saavutettavaa-typografiaa-osa-1/

Laak, T. (2006b). *Saavutettavaa typografiaa – Osa 2*. Saavutettava.fi. https://saavutettava.fi/2006/04/16/saavutettavaa-typografiaa-osa-2/

Leskelä, L. (2019). *Selkokieli: Saavutettavan kielen opas* (2 p.). Hansaprint Oy.

Lie, H. W. (n.d.). https://www.wiumlie.no/

Mozilla. (2021a). *CSS first steps*. Haettu 30.1.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps

Mozilla. (2021b). *How CSS is structured*. Haettu 30.1.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured

Mozilla. (2021c). *JavaScript*. Haettu 30.1.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Glossary/JavaScript

Mozilla. (2022a). *HTML basics*. Haettu 30.1.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Mozilla. (2022b). *Accessibility*. Haettu 7.2.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Web/Accessibility

Mozilla. (2022c). *HTML: A good basis for accessibility*. Haettu 7.2.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/Accessibility/HTML

Mozilla. (2022d). *WAI-ARIA basics*. Haettu 11.2.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics

Mozilla. (2022e). *CSS and JavaScript accessibility best practices*. Haettu 11.2.2022 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/Accessibility/CSS_and_JavaScript

Muistiliitto ry. (n.d.). *Käsitteistö*. https://www.muistiliitto.fi/fi/aivot-ja-muisti/kasitteisto

Näkövammaisten liitto ry. (2021a). *Miten näkövammainen käyttää tietokonetta tai mobiililaitetta*. Haettu 21.1.2022 osoitteesta https://www.nkl.fi/fi/miten-nakovammainen-kayttaa-tietokonetta-tai-mobiililaitetta

Näkövammaisten liitto ry. (2021b). *Näkövammaisuus Suomessa*. Haettu 16.1.2022 osoitteesta https://www.nkl.fi/fi/nakovammaisuus-suomessa

Näkövammaisten liitto ry. (2021c). *Saavutettavuuden testaaminen*. Haettu 12.1.2022 osoitteesta https://www.nkl.fi/fi/verkkosivujen-saavutettavuus

Patel, J. (2022a). *List of Top 12 Popular Websites Built With React in 2022*. Haettu 2.2.2022 osoitteesta Monocubed: https://www.monocubed.com/websites-built-with-react/

Patel, J. (2022b). *Why Use React? – Top 8 Reasons Experts Use React in 2022*. Haettu 2.2.2022 osoitteesta Monocubed: https://www.monocubed.com/why-use-react/

Pavlíček, R. (2017). *Focus 40 Blue 5th Generation* [kuva]. https://www.flickr.com/photos/radlicek/25998273398/in/photostream/

Selovuo, K. (n.d.). *Saavutettavuustestausta selaimessa*. Haettu 17.2.2022 osoitteesta Saavutettavuusopas https://saavutettavuusopas.fi/saavutettavuustestausta-selaimessa/

Selovuo, K. (2019). *Saavutettavuusopas* (1 p.). Euraprint.

Silver, A. (2016). *Improving The Color Accessibility For Color-Blind Users*. Smashing magazine. https://www.smashingmagazine.com/2016/06/improving-color-accessibility-for-color-blind-users/

Ubah, K. (2021). *Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners*. freeCodeCamp. https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/

University of Washington. (n.d.). *A Brief History of HTML*. https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html

Valtiovarainministeriö. (n.d.). *Saavutettavuus*. https://vm.fi/saavutettavuusdirektiivi

W3C. (2018). *Web Content Accessibility Guidelines (WCAG) 2.2*. https://www.w3.org/TR/WCAG/

W3C. (2019). *Clear Layout and Design*. Haettu 5. 2. 2022 osoitteesta https://www.w3.org/WAI/perspective-videos/layout/

W3C. (2020). *WAI-ARIA Overview.* Haettu 11.2.2022 osoitteesta https://www.w3.org/WAI/standards-guidelines/aria/

W3C. (n.d.). *Introduction to Understanding WCAG*. https://www.w3.org/WAI/WCAG21/Understanding/intro

W3Schools. (n.d.). *CSS Browser Support Reference*. Haettu 30.1.2022 osoitteesta W3Schools: https://www.w3schools.com/cssref/css3_browsersupport.asp

Wang, K. (2016). *Kensington SlimBlade Trackball* [kuva]. https://www.flickr.com/photos/kenming_wang/23639529214/in/photostream/

WebAIM. (2020). *Accessible JavaScript - Other issues*. https://webaim.org/techniques/javascript/other

World Health Organization. (2021). *Disability and health*. https://www.who.int/news-room/fact-sheets/detail/disability-and-health

Wunder. (n.d. -a). *Semanttinen HTML - Mikä se on?* https://wunder.io/fi/wunderpedia/saavutettavuus/saavutettava-kayttoliittyma-ui/semanttinen-html/

Wunder. (n.d. -b). *ARIA (Accessible Rich Internet Applications) - Mikä se on?* https://wunder.io/fi/wunderpedia/saavutettavuus/saavutettava-kayttoliittyma-ui/aria/

Yale University. (2018). *Esimerkki linkkitekstien merkityksestä* [kuva]. https://yale.app.box.com/s/ochc2c7zzhsj4wuk6fv41hbk5w8vhxt5

Yale University. (n.d. -a). *Navigation*. https://usability.yale.edu/web-accessibility/articles/navigation

Yale University. (n.d. -b). *Links*. Haettu 6.2.2022 osoitteesta Usability & Web Accessibility: https://usability.yale.edu/web-accessibility/articles/links

Liite 1: Aineistonhallintasuunnitelma

Tässä opinnäytetyössä käytettävä aineisto on kerätty internetistä ja kirjoista. Kaikki lähteet on merkitty opinnäytetyön lähdeluetteloon. Opinnäytetyön aikana ei ole toteutettu haastatteluja tai kyselyjä eli kaikki aineisto on peräisin julkisista lähteistä. Opinnäytetyössä käytettyjen kuvien käyttöoikeudet on tarkistettu ja tarvittaessa pyydetty lupa kuvien käyttämiseen. Opinnäytetyö sisältää myös itse tehtyjä kuvaesimerkkejä. Opinnäytetyön tekijän ei tarvitse säilyttää opinnäytetyön aikana käytettyä aineistoa itsellä opinnäytetyön valmistumisen jälkeen, sillä käytetyt lähteet ovat julkisesti kaikkien saatavilla. Opinnäytetyötä on säilytetty opinnäytetyön aikana tekijän omalla tietokoneella, joka on suojattu salasanalla. Opinnäytetyön tekijä omistaa opinnäytetyön tulokset.

Liite 2: Ohjelmointiprojekti - lomake

```
<form onSubmit={handleSubmit(onSubmit)}>
    <label htmlFor="firstName">First name:</label><br />
    {errors.firstName && errors.firstName.type === "required" && (
      <p role="alert" className="alert">*This is a required field</p>
    )}
    <input placeholder="Your first name" type="text" id="firstName"
      aria-invalid={errors.firstName ? "true" : "false"}
      {...register('firstName', { required: true })}
      aria-required=”true” />
    <br />

    <label htmlFor="lastName">Last name:</label><br />
    {errors.lastName && errors.lastName.type === "required" && (
     <p role="alert" className="alert">*This is a required field</p>
    )}
    <input placeholder="Your last name" type="text" id="lastName"
     aria-invalid={errors.lastName ? "true" : "false"}
     {...register('lastName', { required: true })}
     aria-required=”true” />
    <br />

    <label htmlFor="email">Email:</label><br />
    {errors.email && errors.email.type === "required" && (
      <p role="alert" className="alert">*This is a required field</p>
    )}
    <input placeholder="Your email address" type="email" id="email"
    aria-invalid={errors.email ? "true" : "false"}
    {...register('email', { required: true })}
    aria-required=”true” />
    <br />

    <label htmlFor="message">Message:</label><br />
    {errors.message && errors.message.type === "required" && (
     <p role="alert" className="alert">*This is a required field</p>
    )}
    <textarea placeholder="Write us a message here..." id="message"
    rows="5" cols="33"
    aria-invalid={errors.message ? "true" : "false"}
    {...register('message', { required: true })} aria-required=”true”>
    </textarea>

    <fieldset>
      <legend>I want to receive a newsletter</legend>
      {errors.newsLetter && errors.newsLetter.type === "required" && (
       <p role="alert" className="alert">*This is a required field</p>
       )}
      <input type="radio" name="newsletter" id="yes" value="Yes"
      aria-invalid={errors.newsletter ? "true" : "false"}
      {...register('newsLetter', { required: true })} aria-
      required=”true” />
      <label htmlFor="yes">Yes</label>

      <input type="radio" name="newsletter" id="no" value="No"
      aria-invalid={errors.newsletter ? "true" : "false"}
      {...register('newsLetter', { required: true })}/>
      <label htmlFor="no">No</label>
    </fieldset>

    <button type="submit" id="infoButton">Send message</button>
</form>
```