Käyttäjien moninaisuuden huomiointi ohjelmistokehittäjänä – saavutettavan Vue-päivämääräkomponentin kehitys

Laura Laukkanen

Haaga-Helia ammattikorkeakoulu
Amk-opinnäytetyö
2022
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tekijä
Laura Laukkanen

Tutkinto
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Käyttäjien moninaisuuden huomiointi ohjelmistokehittäjänä – saavutettavan Vue-päivämääräkomponentin kehitys

Sivu- ja liitesivumäärä
56 + 17

Opinnäytetyössä pyritään tuottamaan mahdollisimman saavutettava, Vue-kirjastolla toteutettu päivämääräkomponentti. Sen tarkoituksena on pieneltä osin vastata Vuen kaltaisten JavaScript-kirjastojen haasteisiin kustomoitujen komponenttien saavutettavuudessa. Työ alkaa toiminnalliselle opinnäytetyölle laajahkolla teoriaosuudella, joka pohjustaa komponentin kehitystyötä ohjelmistokehittäjälle oleellisista näkökulmista. Se tuo digitaalisten palveluiden käyttäjien moninaisuuden esille kehitystyön lähtökohtana.

Saavutettavuuden sisällyttämisessä komponenttiin hyödynnetään teoriaosuuden koostamisessa saavutettua osaamista sekä opinnäytetyön tekijän omaa, pääasiassa työn kautta kertynyttä kokemusta saavutettavuusmuutosten toteutuksesta. Kehitystä ohjaa World Wide Web Consortiumin ylläpitämät Verkkosisällön saavutettavuusohjeet sekä niihin liittyvä muu dokumentaatio. Kehityksessä ja testauksessa työtä peilataan erityisesti WCAG 2.1 -onnistumiskriteereihin.

Kehitys toteutetaan ketterän ohjelmistokehityksen tapoja mukaillen ja sitä jaksotetaan sprinteillä. Sprintit ajoittuvat pääasiassa keväälle 2022 ja ovat kestoltaan noin kuukauden mittaisia. Kehityksessä toteutetut tekniset ratkaisut ilmenevät sprinttikohtaisista raporteista. Komponentissa käytetään semanttisen HTML:n lisäksi JavaScript-pohjaisia funktioita, ARIA-attribuutteja ja CSS-sääntöjä saavutettavuuden toteuttamiseksi.

Kehitysvaiheen tuloksena syntyy monilta osin saavutettava Vue-päivämääräkomponentti, joka julkaistaan yleiseen käyttöön Node Package Manager -rekisterin kautta. Työn jatkokehitystarpeet liittyvät osin saavutettavuuteen, mutta pääosin ne painottuvat yleiseen käytettävyyteen sekä komponentin toisiin projekteihin sisällyttämisen helpottamiseen. Komponentti on tarkoitettu käytettäväksi erilaisissa ohjelmistoprojekteissa, kuten verkkolomakkeissa.

Asiasanat
Saavutettavuus, WCAG, päivämääräkenttä, Vue, ohjelmistokomponentti, NPM

Sisällys

1 Johdanto

Suunnittelemalla ja toteuttamalla saavutettavia digitaalisia palveluita voidaan edistää yhdenvertaisuutta. Viime vuosien aikana saavutettavuuteen onkin alettu kiinnittää entistä enemmän huomiota ja sen toteutumista varmistamaan on luotu lakeja ja muuta sääntelyä ympäri maailman. Kuitenkin yhä monet verkkopalvelut tuottavat esimerkiksi avustavaa teknologiaa käyttäville henkilöille haasteita tai ovat jopa täysin tavoittamattomissa.

Saavutettavuus tarkoittaa sitä, että vammaiset ja toimintarajoitteiset ihmiset pystyvät käyttämään digitaalisia palveluita siinä missä muutkin. Jotta saavutettavia digitaalisia palveluita voidaan tarjota, tulee ihmisten moninaiset käyttötavat ja tarpeet palveluiden käytön onnistumiselle huomioida palveluiden elinkaaren eri vaiheissa. Digitaalisen maailman tulee olla rakennettu siten, että tiedonkulku, vuorovaikutus ja toimintojen suorittaminen ovat mahdollisia usealla eri tavalla, esimerkiksi erilaisia avustavia teknologioita käyttämällä. Yksi ammattiryhmä, jolla on merkittävät mahdollisuudet edistää saavutettavuutta, ovat ohjelmistokehittäjät. Meille ohjelmistokehittäjille lisähaasteen luo nykypäivänä suositut JavaScript-pohjaiset sovelluskehykset, joissa saavutettavuuden sisällyttäminen on osoittautunut mutkikkaaksi.

Tässä opinnäytetyössä tartutaan konkreettiseen saavutettavuuden kannalta haastavaan verkkosivujen osaan, päivämääräkenttään ja sen yhteyteen avautuvaan kalenterinäkymään. Myöhemmin päivämääräkenttä viittaa tähän kokonaisuuteen. Päivämääräkenttä toistuu paljon verkkosivustoilla kuten erilaisissa sähköisissä lomakkeissa. Silti kokonaisvaltaisesti saavutettavia päivämääräkenttiä on vaikeaa löytää. Laajemmin opinnäytetyöllä pyritään vastaamaan tarpeeseen kehittää JavaScript-sovelluskehyksissä käytettävien komponenttien saavutettavuutta.

Tavoitteena on rakentaa saavutettava ja uudelleenkäytettävä päivämääräkomponentti Vue-sovelluskehystä hyödyntäen. Yksi Vuen ominaispiirteistä onkin uudelleenkäytettävien komponenttien hyödyntäminen kehityksessä. Teoriaosuudessa kartoitan saavutettavuutta koskevia vaatimuksia ja toiminnallisessa osuudessa pyrin kehittämään parhaalla mahdollisella tavalla näihin vaatimuksiin vastaavan päivämääräkentän sisältävän komponentin. Pyrin opinnäytetyön aikana syventämään ymmärrystäni saavutettavuuden ohjelmallisesta toteutuksesta ja muista saavutettavuudessa huomioitavista seikoista.

Toimeksiantaja opinnäytetyössä on oma työpaikkani. Opinnäytetyön luonne ei vaadi toimeksiantajaa koskevan tiedon käsittelyä. Tarkoituksena kuitenkin on, että lopputuloksena syntyvää komponenttia voitaisiin hyödyntää myös työpaikallani. Julkaistavasta komponentista voi lisäksi hyötyä laajempikin joukko sovelluskehittäjiä ja käyttäjiä.

Vaikka opinnäytetyön tarkoituksena ei varsinaisesti ole toimia saavutettavuuden tietopakettina, toivon sen silti luovan kuvaa aiheesta niin teknisille kuin muiden alojen edustajille tai muuten aiheesta kiinnostuneille henkilöille. Tarkastelen saavutettavuutta ensin yleisemmällä tasolla ja sitä mukaa kun työ lähenee toiminnallista osuutta, sovelluskehittäjän näkökulma alkaa painottua tarkastelussa.

2 Käsitteiden määrittely

Englanninkielistä termiä ”disabled” käytetään paljon puhuttaessa saavutettavuudesta. Kehitysvammaliitto ylläpitää monipuolista selkeään viestintään keskittyvää Papunet-verkkosivustoa (Papunet 2021a.), jolta löytyy laajasti tietoa myös saavutettavuudesta. Koska verkkosivusto on luotettavan yhteiskunnallisen toimijan ja vammaisuuteen liittyvän asiantuntijaorganisaation ylläpitämä, päädyin käyttämään sivustolla toistettavia vastineita ”henkilö, jolla on jokin vamma”, ”vammaisuus” ja ”toimintarajoitteisuus”. Näitä termejä käytetään eri muodoissa opinnäytetyössä. Olen listannut tässä kappaleessa myös muiden keskeisten termien määrityksiä:

CSS (Cascading Style Sheets)
HTML-verkkosivujen tyylien muokkaukseen käytettävä kieli, jonka avulla määritellään, millä tavoin elementit näytetään käyttöliittymässä. Samoja CSS-sääntöjä voidaan käyttää useilla eri verkkosivuilla samanaikaisesti. (W3Schools 2021a.)

Digipalvelut
Verkkosivut ja mobiilisovellukset

Git
Versionhallintaan käytettävä työkalu, jolla voi hallita erikokoisten projektien työnkulkua. Gitissä järjestelmä on hajautettu eli se on suunniteltu toimimaan useammalla palvelimella. Tämä ominaisuus edesauttaa sitä, että tarvittaessa useampi ohjelmistokehittäjä voi ylläpitää ja kehittää projektia samanaikaisesti. Gitin lähdekoodi on avoin ja työkalun käyttö on ilmaista. (Git 2022, Chacon & Straub 2022, 126)

HTML (HyperText Markup Language)
Verkkosisällön rakentamisen peruselementti. HTML:n avulla määritellään sisällön tarkoitus ja rakenne. HTML:n ohella on tavallista käyttää muita teknologioita esimerkiksi ulkoasun ja toiminnallisuuksien määrittelyyn. (MDN Web Docs 2021a.)

JavaScript (lyhennetään ”JS”)
Verkon käyttöliittymäpuolella toimiva komentosarjakieli, jota käytetään hyvin yleisesti verkkosivujen toiminnallisuuksien hallintaan. JavaScript soveltuu myös muualle kuin selainympäristöön. Yksi sen käyttötarkoituksista on toimia palvelinpuolen komentosarjakielenä, mistä esimerkkinä voi mainita Node.js:n. (MDN Web Docs 2021b.)

Mobiilisovellus
Mobiililaitteella, kuten matkapuhelimella, tabletilla tai kellolla käytettävä ohjelmisto.

Mobiilisovellusten käyttö mobiililaitteilla on yleisempää kuin verkkoselainten käyttö vastaavissa laitteissa. Mobiilisovellusten suunnittelussa tulee huomioida laitteiden erityispiirteet, kuten vaihteleva ruudun koko. (IteWiki s.a.)

Päivämääräkenttä
Esimerkiksi verkkolomakkeella käytettävä lomakkeen osa, johon käyttäjä voi syöttää päivämäärän esimerkiksi valitsemalla sen kalenterinäkymästä tai kirjoittamalla. Päivämäärän syötön tarkoitus on usein kuvattu kentän yhteydessä. Kentän toimintaan voi olla liitettynä validointia esimerkiksi ohjaamaan käyttäjää syöttämään päivämäärän oikeassa muodossa tai sijoittamaan päivämäärän tietylle ajanjaksolle. Myös kentän pakollisuus voidaan määritellä.

Saavutettavuus
Verkkopalvelujen esteettömyyttä kuvaava termi. Saavutettavuus kertoo siitä, onko ihmisten moninaisuus huomioitu verkkopalveluissa. (Aluehallintovirasto s.a.a)

Saavutettavuusdirektiivi
Euroopan parlamentin ja neuvoston direktiivi (EU) 2016/2102, jonka tarkoituksena yhdessä kansallisen lainsäädännön kanssa on velvoittaa viranomaiset tekemään digitaalisista palveluistaan saavutettavia. Direktiivi astui voimaan 22.12.2016. (Valtionvarainministeriö s.a.)

Scrum
Löyhästi määritelty viitekehys tiimityöskentelyyn. Scrumissa keskeistä on joustavuus, jonka nähdään tuottavan lisäarvoa monimutkaisten ongelmien ratkaisussa. Scrum sisältää erilaisia rooleja ja työ jakautuu sprinteiksi kutsuttuihin ajanjaksoihin. (Schwaber & Sutherland 2020, 3.)

Verkko (World Wide Web)
Verkko tarkoittaa tietynlaista tiedon siirtämisen tapaa internetissä. Esimerkiksi sähköposti on toinen tapa siirtää tietoa internetissä. Verkon muodostavat nykypäivänä miljardit verkkosivut, joita on mahdollista käyttää verkkoselaimen välityksellä. Termit verkko ja internet saattavat usein sekoittua keskenään arkikielessä. Ne ovat kuitenkin kaksi eri teknologiaa, ja verkko on internetin osa. Internet mahdollistaa verkon toimimisen. (Gil 2020)

Verkkoselain, selain
Ohjelma, jota käytetään pääasiassa verkkosivujen näyttämiseen sekä niiden tarjoamien toiminnallisuuksien käyttämiseen. Vaikka selaimia kutsutaan myös verkkoselaimiksi, niillä

voi käyttää myös muuta internetin sisältöä kuin verkkoa. Esimerkkejä verkkoselaimista ovat Chrome, Safari, Opera, Microsoft Edge ja Mozilla Firefox.

Verkkosivusto
Joukko digitaalisia tietosisältöjä, jotka ovat verkkotunnuksella yksilöitävissä ja niitä voidaan käyttää erilaisilla päätelaitteilla ja niiden ohjelmistoilla (Laki digitaalisten palvelujen tarjoamisesta 2019/306, 2 §).

Vue.js (myös Vue)
Käyttöliittymien luomiseen tarkoitettu avoimen lähdekoodin MIT-lisensoitu, progressiivinen JavaScript-sovelluskehys (engl. framework) (Vue.js 2022).

Vue.js-komponentti
Uudelleenkäytettävän koodin kapseloiva elementti, jonka toteutuksessa on hyödynnetty Vue.js-kirjastoa.

WAI-ARIA (Accessible Rich Internet Applications Suite)
Joukko erityisesti monimutkaisen ja dynaamisen verkkosisällön saavutettavuuden edistämiseen tarkoitettuja menetelmiä. WAI-ARIAn avulla voidaan esimerkiksi lisätä elementeille attribuutteja, jotka muun muassa ilmaisevat eri osien välisiä yhteyksiä ja tiloja. (Cooper 2020).

W3C (World Wide Web Consortium)
Verkkostandardeja kehittävä ja ylläpitävä kansainvälinen yhteenliittymä (W3C 2021a).

WCAG (Web Content Accessibility Guidelines, Verkkosisällön saavutettavuusohjeet)
W3C.n kehittämä laaja joukko ohjeistuksia, jotka koskevat verkkosisällön saavutettavuutta. Ohjeiden noudattaminen parantaa sisällön saavutettavuutta ihmisille, joilla on vammoja ja rajoitteita. Ohjeet huomioivat ”sokeuden ja heikkonäköisyyden, kuurouden ja huonokuuloisuuden, liikuntarajoitteet, puhevammat, valoherkkyyden ja näiden yhdistelmät. Se sisältää lisäksi joitain parannuksia henkilöille, joilla on oppimisen ja ymmärtämisen ongelmia tai muita kognitiivisia rajoitteita, mutta ei huomioi em. henkilöiden kaikkia tarpeita. Nämä ohjeet kattavat verkkosisällön saavutettavuuden työpöytä- ja kannettavilla tietokoneilla, tablet-laitteilla sekä mobiililaitteilla.” Nykyinen versio 2.1 täydentää vuonna 2008 julkaistua versiota 2.0. (W3C 2018a.)

3 Saavutettavuus

Jo pitkään on ollut yleisesti tiedossa, että eri käyttäjäryhmillä on verkkosivujen käyttöön liittyviä erityistarpeita. Näin kuuluivat jo vuonna 1997 World Wide Webin kehittäjänä toimineen Tim Barnes-Leen paljon siteeratut sanat: ”Verkon voima on siinä, että se kuuluu kaikille. Kaikkien pääsy sinne vammaisuudesta riippumatta on olennaista.” (W3C 1997.) Verkon (engl. web) oletetaan siis lähtökohtaisesti olevan saavutettava maailmanlaajuistesti, riippumatta käytettävästä teknologiasta, kielestä tai käyttäjän kyvyistä.

Verkko on auttanut poistamaan kommunikointiin ja vuorovaikutukseen liittyviä fyysisen maailman esteitä, joita vammaiset henkilöt mahdollisesti kohtaavat. Monien ihmisten mahdollisuuksia osallisuuteen voidaan kuitenkin estää myös verkossa, jos esimerkiksi verkkosivut tai mobiilisovellukset ovat huonosti suunniteltuja. Siksi on ensiarvoisen tärkeää, että eri teknologioiden kehittäjät ovat tietoisia saavutettavuuden merkityksestä. (W3C 2018b.)

Tässä luvussa selvitän, mikä saavutettavuuden tila on Suomessa ja millä keinoin yhteiskunnassa pyritään kohti saavutettavampia verkkopalveluita. Lähden liikkeelle saavutettavuuden määritelmästä ja merkityksestä verkon käyttäjille. Tällä pyrin hahmottamaan lähtökohtia sille, miksi ja kenelle on tärkeää, että verkkopalveluiden saavutettavuutta kehitetään.

3.1 Saavutettavuuden määritelmä

Eri tahot määrittelevät saavutettavuuden hieman eri tavoin, osa laajemmin ja osa suppeammin. World Wide Web -konsortion (myöh. W3C) verkkosivuilla saavutettavuus määritellään sen kautta, pystyvätkö vammaiset henkilöt käyttämään verkkopalveluja tai verkon toimintaan liittyviä työkaluja ja muita teknologioita monipuolisesti ja interaktiivisesti. Tässä huomioidaan vammaisten ihmisten tasavertainen osallisuus verkon käyttöön ilman ylimääräisiä esteitä. (W3C 2021b; Henry, White & Abou-Zahra 2016.)

Invalidiliiton verkkosivujen määritelmä sen sijaan on hieman laajempi. Sen mukaan saavutettavuus viittaa muuhun kuin fyysiseen ympäristöön. Fyysisestä ympäristöstä on totuttu tässä asiayhteydessä käyttämään termiä esteettömyys. Esteettömyyttä ja saavutettavuutta yhdistää se, että sekä fyysisen että ”aineettoman” maailman tulisi olla kaikkien ihmisten käytettävissä toimintakyvystä riippumatta. Tämä niin kutsuttu aineeton ympäristö kattaa Invalidiliiton mukaan myös ilmapiirin ja asenteet. (Invalidiliitto s.a.) Myös UX-asiantuntija ja kouluttaja Regine M. Gilbert toteaa kirjassaan ”Inclusive Design for a Digital World - Designing with Accessibility in Mind”, että niin kutsuttu disabilismi eli

vammaisuuden perusteella tapahtuva syrjintä ja asenteellisuus aiheuttavat vaikeuksia osallistua yhteiskunnan toimintaan (Gilbert 2019, luku 1).

Näkövammaisten liiton mukaan saavutettavuus viittaa erityisesti digitaalisiin palveluihin. Eri käyttäjäryhmien näkökulmat saavutettavuuteen saattavat hieman poiketa toisistaan, mutta käsitteen merkitys kiteytyy kuitenkin yhdenvertaisuudessa. Verkkopalveluiden tulee olla jokaisen ihmisen käytettävissä eikä niiden käyttö vaikeudu, vaikka henkilöllä olisi jokin toimintarajoite. (Näkövammaisten liitto 2021.)

Suomen lainsäädännöstä, tarkemmin digipalvelulaista, löytyvä määritelmä täydentää aiemmin mainittuja huomioimalla saavutettavuuden myös palvelujen käytännön tuottamisen näkökulmasta. Sen mukaan saavutettavuus on ”periaatteita ja tekniikoita, joita on noudatettava digitaalisten palvelujen suunnittelussa, kehittämisessä, ylläpidossa ja päivittämisessä, jotta ne olisivat paremmin käyttäjien, erityisesti vammaisten henkilöiden, saavutettavissa” (Digipalvelulaki 2019/306, 2 §). Tässä määritelmässä tulevat esille sovelluskehittäjänkin näkökulmasta merkittävät digitaalisten palvelujen kehitysvaiheet.

Tässä opinnäytetyössä saavutettavuutta käsitellään sovelluskehittäjän työn kannalta merkityksellisiin teknisiin kriteereihin ja toteutustapoihin pureutuvina yksityiskohtina, joiden taustalla kuitenkin tunnistetaan inhimilliset tarpeet. Toiminnallinen osio keskittyy päivämääräkentän rakentamiseen nimenomaan teknisestä saavutettavuudesta käsin. Komponentin kehityksessä huomioidaan teknisen saavutettavuuden lisäksi myös yleinen käytettävyys. Invalidiliiton sivuilla mainitaan myös ”kaikille sopiva suunnittelu” (Design for All/Universal Design). Nimensä mukaisesti siinä on lähtökohtaisesti tarkoitus huomioida suunnittelussa ja ratkaisuissa ihmisten moninaisuus. (Invalidiliitto s.a.)

3.2 Osa-alueet

Esimerkkejä yksittäisistä erilaisten digipalvelujen saavutettavuuteen vaikuttavista ominaisuuksista on lukuisia. Karkeasti ajateltuna on tärkeää ottaa huomioon kolme osa-aluetta; Saavutettavien digipalvelujen tulee olla teknisesti virheettömiä, niillä tulee olla selkeä ja hahmotettava käyttöliittymä ja sisällön tulee olla ymmärrettävää.

Saavutettavuuden osa-alueista teknisesti virheetön toteutus tarkoittaa käytännössä sitä, että lähdekoodi on loogisesti rakennettu eikä sisällä virheitä. Tähän liittyvä HTML-standardin sekä WCAG-ohjeistuksen noudattaminen käydään läpi opinnäytetyön myöhemmissä luvuissa. Lisäksi tekniseen saavutettavuuteen liittyy digipalvelun käytettävyys erilaisilla päätelaitteilla ja avustavilla teknologioilla.

Digipalvelun helppokäyttöisyys sen sijaan tarkoittaa sitä, että palvelu rakenne on riittävän yksinkertaisesti rakennettu ja helposti hahmotettavissa. Lisäksi käytön ja haluttujen toimintojen suorittamisen tulisi olla mutkatonta. Helppokäyttöisen digipalvelun, kuten verkkosivun, rakentamisessa tulisi huomioida muun muassa seuraavat asiat:

- yksinkertainen ja selkeä navigaatio
- sisältö ja toiminnot vaivattomasti löydettävissä
- kuvaavat ja selkeät sivujen nimet
- pääsisällön selkeä esittäminen erillään muista elementeistä

Ymmärrettävyys sen sijaan on esimerkiksi sitä, että digipalvelun kieli on selkeää ja hyvän jäsentelyn vuoksi helposti hahmotettavaa. Myös selkokieltä voidaan käyttää edistämään digipalvelun ymmärrettävyyttä.

Ymmärrettävyyttä digipalveluissa lisää se, että sisältöä tarjotaan monikanavaisesti eli muussakin muodossa kuin vain tekstinä. Sisältöä voidaan tällöin käyttää esimerkiksi videoiden tai äänen välityksellä. Myös kuvia voidaan hyödyntää tekstin lisäksi. (Aluehallintovirasto s.a.b.) Monikanavaisuus voi siis antaa mahdollisuuden vastaanottaa tietoa toimivan aistin välityksellä käyttäjille, joilla jonkin aistin käyttö on esimerkiksi kokonaan estynyt. Joillekin taas monikanavaisuus voi vahvistaa viestiä, jos esimerkiksi jonkin aistin käyttö on heikentynyt tai vaikeutunut esimerkiksi meluisan ympäristön vaikutuksesta.

Suunnittele kaikille -periaate (engl. Design for all) liittyy tiiviisti saavutettavuuteen. Siinä pyritään miettimään käyttäjäkuntaa mahdollisimman laajasti ja tekemään lähtökohtaisesti kaikille eri käyttäjäryhmille soveltuva toteutus. Eri käyttäjäryhmien tarpeet huomioidaan läpi palvelun suunnittelun ja toteutuksen. (Aluehallintovirasto s.a.c.) Komponentin suunnitteluprosessissa pyrin noudattamaan suunnittele kaikille -periaatetta.

W3C:n verkkosivuilla listataan verkon käyttöön ja näin saavutettavuuteen liittyviä vammaisuuden tai toimintarajoitteisuuden osa-alueita. Huomioitavia alueita sen mukaan ovat kuuloon, puheeseen tai näkökykyyn liittyvät, kognitiiviset, neurologiset sekä fyysiset toimintarajoitteet. (W3C 2021b.) Myös Gilbert listaa jo aiemmin mainitussa kirjassaan saavutettavuuden osa-alueet. Kun vertaa listaa W3C:n mainitsemiin osa-alueisiin, Gilbert mainitsee neurologisten, fyysisten ja puheeseen liittyvien osa-alueiden sijasta motoriset rajoitteet. Hän myös jakaa osa-alueet pieniin osakokonaisuuksiin ja tasoihin. Esimerkiksi näkökyvyn kohdalla on huomioitava niin sokeus, heikko näkökyky kuin värisokeuskin. (Gilbert 2019, luku 1.)

Saavutettavuuden ymmärtämiseksi onkin tärkeää huomioida se, että vammaisuus ei aina tarkoita esimerkiksi täysin jonkin aistin puuttumista. Myös ikääntyvät ihmiset kuuluvat usein saavutettavuuden kohderyhmään. Ikääntyminen voi tuoda mukanaan eri osa-

alueisiin vaikuttavia muutoksia. Lisäksi minkä tahansa ikäiset ihmiset voivat kokea ohimeneviä vammoja tai tilanteita, joissa verkkosisällön käyttö on tilapäisesti vaikeutunut. (W3C 2021b.) Jokainen ihminen voi siis joissakin tilanteissa hyötyä saavutettavuudesta. Monille vammaisille ihmisille se kuitenkin on välttämättömyys.

3.3 Miksi saavutettavuuden huomiointi on tärkeää

Maailman terveysjärjestö WHO:n (World Health Organization) vammaisuudesta kertovalla verkkosivulla todetaan, että vammaisuus kuuluu ihmisyyteen. Sivun mukaan lähes jokainen ihminen elää jossakin vaiheessa elämäänsä jonkin, joko pysyvän tai väliaikaisen, vamman kanssa. Vammaisten osuus on arvioiden mukaan noin 15 % maailman väestöstä. Tämä tarkoittaa, että yli miljardi ihmistä elää jonkinasteisen vamman tai toimintarajoitteen kanssa. Vammaisten ja toimintarajoitteisten ihmisten määrä myös kasvaa nopeasti. Tämä johtuu osittain väestön ikääntymisestä ja osittain siitä, että krooniset sairaudet lisääntyvät. (World Health Organization 2021a.) Suomessakin on arvioiden mukaan yli miljoona ihmistä, joille tuottaa vaikeuksia käyttää verkkopalveluja. Käyttöä vaikeuttaa erityisesti se, jos palvelua ei ole tehty saavutettavaksi. (Aluehallintovirasto s.a.a.)

Kun etsii tietoa saavutettavuudesta, hyvin usein samassa yhteydessä mainitaan W3C. W3C:n tehtävänä onkin kehittää ja ylläpitää maailmanlaajuisia verkkostandardeja (W3C 2021c). Patenttikäytäntöjensä (W3C Patent Policy) mukaisesti W3C pyrkii siihen, että sen antamat verkon käyttöön liittyvät suositukset toteutetaan tekijänoikeusvapaasti (Royalty-Free) (W3C 2004). W3C on siis ovat merkittävässä osassa siinä, että World Wide Web toimii nykyisellä tavalla ja että verkkostandardit ovat laajasti käytettävissä ilmaiseksi.

Jotta voi suunnitella ja rakentaa saavutettavaa digitaalista sisältöä, on tärkeää ymmärtää, kuinka eri tavoin ihmiset voivat käyttää verkkoa (Shadi 2017). Web Accessibility Initiativen sivuilla on listattuna lyhyitä tarinoita saavutettavuudesta hyötyvistä henkilöistä. Tarinoissa muun muassa puna-viherväärisokea henkilö hyötyy tekstien ja kuvien riittävästä kontrastista sekä mahdollisuudesta vaikuttaa itse selaimen kontrastiasetuksiin. Lisäksi mikäli väri välittää käyttäjälle informaatiota, tulee tämän informaation olla saatavilla myös esimerkiksi tekstivastineena. Muita tarinoissa esille tulevia tilanteita:

- rasitusvamma estää tietokoneen hiiren käyttämisen ja henkilö käyttää digitaalista sisältöä näppäinkomennoilla, äänentunnistustoiminnolla ja assistive-touch-toiminnolla puhelimessaan
- opiskelija, jolla on heikentynyt kuulo, hyötyy tekstitetystä äänisisällöstä luennolla
- pistekirjoitusta osaamaton näkövammainen henkilö käyttää ruudunlukuohjelmaa
- tarkkaavaisuus- ja lukihäiriön omaava opiskelija hyötyy puhesynteesi (Text-To-Speech) -sovelluksesta

Usein ikääntymiseen liittyvistä toimintarajoitteista tarinoissa tulee esiin heikkonäköisyys, vapina ja muistiin liittyvät haasteet. Näihin voivat auttaa esimerkiksi mahdollisuus suurentaa sivujen tekstejä ilman, että sivun rakenne tai esimerkiksi sivulta löytyvä CAPTCHA-toiminto hajoaa, erikoisvalmisteinen hiiri sekä mahdollisuus tallentaa selaimeen kuvina muistiin sivustot, joilla usein vierailee. (Shadi & Sinclair 2017.)

Verkkosivujen tulee siis olla käytettävissä henkilöille, jotka esimerkiksi eivät voi käyttää hiirtä tai näppäimistöä, tarvitsevat ruudunlukuohjelman tai antavat tietokoneelle komentoja puheen avulla. Lisäksi osalla käyttäjistä voi olla useampia samanaikaisia rajoitteita. On tärkeää, että digitaalisesta sisällöstä tehdään mahdollisimman saavutettavaa mahdollisimman suurelle joukolle ihmisiä.

Vaikka vallalla tuntuu olevan saavutettavuuden edistämiseen tähtäävä ilmapiiri ja keskustelukulttuuri, joskus uudistukset ja muutokset esimerkiksi verkkosivuilla voivat silti heikentää palvelun saavutettavuutta. Kyseessä voi olla elintärkeän tiedon saaminen terveydenhuollon verkkosivuilta ja joskus taas kyse on esimerkiksi osallisuudesta tai yhteydestä toisiin ihmisiin ja näin elämänlaadusta (UiAccess 2010).

Erityisesti julkisten toimijoiden tietojärjestelmien saavutettavuuden voisi viimeaikaisten muutosten myötä ajatella olevan menossa yleisesti parempaan suuntaan. Kehityskulku ei kuitenkaan ole vielä täysin vakiintunut. Helsingin Sanomissa 21.5.2021 ilmestyneessä Anna Takalan kirjoittamassa artikkelissa ”’En pysty tekemään itse mitään’ – Helsingin yliopiston suursatsauksena tehty tietojärjestelmä on osalle käyttäjistä painajainen” kerrotaan, että Helsingin yliopiston uudesta Sisu-opintotietojärjestelmästä on tehty kantelu Etelä-Suomen aluehallintovirastoon, koska järjestelmää esimerkiksi ei voi käyttää ruudunlukuohjelmalla tai pistenäppäimistöllä. Järjestelmän kehitys on aloitettu vuonna 2013. (Takala 2021.)

On olemassa saavutettavuuteen liittyviä toimintatapoja, joita olisi hyvä tarkastella ohjelmistoprojekteissa. Näin voitaisiin yltää parempaan saavutettavuuden tasoon. Yksi esimerkki haitallisista toimintatavoista on se, että saavutettavuusmuutoksia aletaan tehdä vasta, kun projekti on muilta osin valmis. Tällöin muutosten tekeminen vie todennäköisesti huomattavasti enemmän aikaa ja rahaa kuin jos se olisi huomioitu heti projektin alusta lähtien.

Toinen yleinen toimintatapa on se, että saavutettavuutta ajatellaan ohjeistuksista koottuna muistilistana, josta voidaan yliviivata kohtia standardien täyttyessä. Standardien ei kuitenkaan ole tarkoituksenmukaista olla saavutettavuusmuutosten ainoana lähtökohtana. Ensisijaisen tärkeää onkin ymmärtää, että saavutettavuudessa on kyse siitä, että

erityistarpeita omaavat ja vammaiset henkilöt pystyvät käyttämään verkkoa. Siksi on tärkeää myös tietää erilaisia tapoja käyttää digitaalisia palveluita. (Henry 2006, luku 1.) Pyrin noudattamaan tätä opinnäytetyössä ja tutustumaan erilaisiin verkkosisällön käyttötapoihin ennen komponentin kehitystä.

Saavutettavuus on yksi esimerkki siitä, että kun monille vammaisille välttämättömät saavutettavuusmuutokset toteutetaan, ne samalla kehittävät verkkopalveluita paremmin käytettäviksi kaikille ihmisille. Inhimillisten hyödyn lisäksi saavutettavuudesta on hyötyä myös yritysten toiminnalle. Saavutettavuuden avulla organisaatioilla on mahdollisuus tavoittaa maailman 1,3 miljardia vammaista ihmistä (Haben 2017).

3.4 Ohjeistuksia ja standardeja

Vaikka verkon käyttöön tarkoitetut laitteet, teknologit tai ohjeistukset muuttuisivat, saavutettavuuden periaatteet pysyvät suhteellisen muuttumattomina (Jokinen 2020, 31). Periaatteiden lisäksi on vielä paljon yksityiskohtiin meneviä ohjeistuksia sekä erialisia vaatimustasoja, joiden saavuttamiseksi digitaalisen palvelun sisällön tulee täyttää tietyt, yksityiskohtaiset kriteerit. Avaan tässä kappaleessa saavutettavuuden periaatteita ja ohjeistuksia, joita myöhemmin hyödynnän päivämääräkomponentin kehityksessä.

Sarah Lewthwaiten ja Abi Jamesin Disability & Society -lehden artikkelin ”Accessible at last?: what do new European digital accessibility laws mean for disabled people in the UK?” (2020, 1363) mukaan rakenteet eivät ole vielä riittävät saavutettavuutta vaalivan kulttuurin edistämiseksi. Artikkelin kirjoittajien mukaan on olemassa vaara sille, että saavutettavuuteen aletaan suhtautua ”yliteknisesti”, jos vallalla on liiallisesti vaatimusten toteuttamiseen keskittyvä kulttuuri. Samalla tämä voi jättää varjoonsa vammaisten henkilöiden verkon käyttöön liittyvät moniulotteiset sosiotekniset sekä vammaisten henkilöiden ja vammaisyhteisöjen tukemiseen liittyvät kysymykset.

Kuitenkin standardit ovat tärkeä työkalu, kun luodaan pohjavaatimuksia saavutettavuudelle. Lewthwaiten ja Jamesin mukaan asia on kiireinen, jotta vammaiset ihmiset eivät enenevissä määrin jää yhteiskunnan mahdollisuuksien ulkopuolelle eri osa-alueilla. (Lewthwaite & James 2020, 1363.) Päivämääräkomponentin kehityksen suhteen yliteknisyyden välttäminen voisi tarkoittaa sitä, että kriteerien täyttämisen yhteydessä tapoja toteutukselle pohditaan myös ihmislähtöisesti. Pidetään siis mielessä, ketä varten saavutettavuusmuutoksia tehdään ja mitkä ovat todelliset, inhimillisyydestä kumpuavat päämäärät kriteereissä.

WAI eli Web Accessibility Initiative on W3C:n aloite, joka on luotu edistämään käytettävyyden korkeaa laatua vammaisille henkilöille. (Brewer & Henry 2020.) WAI on luonut maailmanlaajuisesti yhtenäisen saavutettavuusohjeistuksen. Ohjeistuksen nimi on suomeksi Verkkosisällön saavutettavuusohjeet. Se lienee kuitenkin tunnetumpi lyhenteestään WCAG, joka tulee englanninkielisistä sanoista Web Content Accessibility Guidelines. (Henry 2021a.)

Verkkosisällön lisäksi esimerkiksi sisällön saatavuuteen ja tuottamiseen käytettävien työkalujen saavutettavuus vaikuttavat verkon kokonaissaavutettavuuteen. (Zahra 2019.) W3C onkin tuottanut myös muita saavutettavuuteen liittyviä WAI-ohjeistuksia. Näitä ovat ATAG (Authoring Tool Accessibility Guidelines) ja UAAG (User Agent Accessibility Guidelines). Työstettävä päivämääräkenttä on kuitenkin osa verkkosisältöä, joten kehityksen kannalta huomioitava ohjeistus on nimenomaan WCAG. Seuraavaksi käyn läpi WCAG:n sisältöä ja tulkintatapoja.

3.4.1 WCAG

WCAG on saavutettavuutta koskevien ohjeiden de facto -standardi (Jokinen 2020, 91) ja se toimii myös päivämääräkomponentin saavutettavuusvaatimusten määrittelyn ensisijaisena lähteenä. Komponentin kehityksen tarkoituksenmukaisen etenemisen ja tyydyttävän lopputuloksen kannalta on olennaista tiedostaa, mitä ohjeistuksia ja standardeja tulee ottaa huomioon ja oppia tapoja tulkita niiden sisältöä.

Opinnäytetyön kirjoittamisen aikana käytössä oleva WCAG-versio on vuonna 2018 julkaistu WCAG 2.1. WCAG:n eri versiot eivät ole toisiaan poissulkevia, mutta W3C suosittelee käyttämään tuoreinta versiota verkon saavutettavuusohjeistuksissa (Kirkpatrick, O'Connor, Campbell & Cooper 2018). Noudatan tätä ohjetta, jotta kriteerit olisivat mahdollisimman ajantasaisia.

Nimensä mukaisesti WCAG koskee nimenomaan verkkosisältöä. Sisällöllä (engl. content) viitataan verkon sisältämään erilaiseen informaatioon. Tämä informaatio voi olla esimerkiksi visuaalisesti havaittavia tekstiä ja kuvia tai auditiivista, kuten erilaiset äänet. Lisäksi sisältö kattaa esimerkiksi rakenteeseen ja esitystapaan liittyvän merkintäkielen tai koodin. (Henry 2021a.) Päivämääräkomponentti on sisällöltään monipuolinen yksittäiseksi lomakkeen kentäksi, koska siinä on tyypillisesti totuttu käyttämään päivämäärän valitsemiseen avautuvaa kalenterinäkymää. Tämä näkymä sisältää erilaisia painikkeita, päivämäärien numerot sisältävän ”ruudukon” sekä ainakin kuukauden ja vuoden tekstinä.

WCAG 2.1 on suuri joukko suosituksia, joiden avulla pöytä- ja kannettavien tietokoneiden sekä erilaisten mobiililaitteiden verkkosisällön saavutettavuutta voidaan parantaa. Ohjeistus huomioi laajasti erilaisia vammoja ja rajoitteita, joita ihmisillä voi olla. Lisäksi se huomioi ikääntyneiden ihmisten mahdollisia tarpeita sekä tekee käytettävyydestä yleisesti parempaa. Ohjeistus ei kuitenkaan vastaa täydellisesti kaikkiin käyttäjien mahdollisiin tarpeisiin ja sen kehityksessä on myös törmätty haasteisiin esimerkiksi kognitiivisten rajoitteiden huomioinnissa.

WCAG 2.1:n kriteerien toteutus ei ole teknologiasidonnaista. Tiettyihin teknologisiin toteutustapoihin pureutuvia ohjeistuksia on kuitenkin tarjolla erillisissä dokumenteissa. Lisäksi onnistumiskriteereiden tulkintaan löytyy apua. (Kirkpatrick ym. 2018.). W3C:n sivuilta selviää, että mobiililaitteita koskeva saavutettavuus on sisällytetty nykyisiin ohjeistuksiin ja standardeihin. W3C sisällyttää mobiilisaavutettavuuden piiriin monia erilaisia laitteita, joissa yhdistävinä ominaisuuksina saattaa olla esimerkiksi pieni näyttökoko, kosketusnäyttö tai poikkeavat tavat antaa komentoja esimerkiksi puheen avulla. Näihin lukeutuu esimerkiksi erilaiset älykellot. (Henry 2021a.) Tässä opinnäytetyössä mobiililaitteiden saavutettavuus kattaa tabletit ja matkapuhelimet.

WCAG-ohjeistus on tarkoitettu hyvin monenlaisten kohderyhmien hyödynnettäväksi. Näin ollen sama ohjeistus pyrkii vastaamaan monenlaisiin tarpeisiin. Esimerkiksi päättäjien ja yksityiskohtaisesti verkkosovelluksia rakentavien ohjelmoijien tietotarpeet voivat poiketa toisistaan paljonkin. Tämän vuoksi ohjeistus on jaettu eri tarpeisiin vastaaviin tasoihin. Tasoja ovat periaatteet, ohjeistukset, onnistumiskriteerit sekä tekniikat (riittävät ja neuvoa-antavat tekniikat).

Saavutettavuuden periaatteita on neljä. Verkon sisällön tulee olla havaittavaa, hallittavissa olevaa, ymmärrettävää ja toimintavarmaa. Puhutaan myös POUR-periaatteista. (Kirkpatrick ym. 2018.) Aiemmin opinnäytetyössä mainitut saavutettavuuden osa-alueet täydentyvät siis aktiiviseen käyttöön viittaavalla hallittavuudella. Ohjelmistokehittäjän näkökulmasta mainitut periaatteet ovat hyvä lähtökohta, mutta ovat konkreettisen ohjelmoinnin kannalta vielä kovin abstraktilla tasolla.

Seuraava taso WCAG 2.1 -saavutettavuusohjeistuksessa on itse ohjeet, joita on yhteensä 13. Ne kertovat, kuinka huomioida erilaiset vammat ja toimintarajoitteet saavutettavuudessa. Ohjeet on tarkoitettu ikään kuin ohjaamaan seuraavalla tasolla olevien onnistumiskriteerien käyttöä. Esimerkiksi havaittavuuden periaatteeseen liittyy ohje tekstivastineiden tarjoamisesta verkkosisällössä. (Kirkpatrick ym. 2018.) Ohjeet eivät ole itsessään testattavia toisin kuin onnistumiskriteerit (Kirkpatrick ym. 2018), joihin pureudutaan seuraavaksi.

Jokaista ohjetta koskee useampi testattavissa oleva onnistumiskriteeri. Kriteerien luonne on erotettavissa tavallisista käytettävyysvaatimuksista. Tämä tarkoittaa, että niiden toteutumisesta on suhteessa enemmän hyötyä vammaisille tai toimintarajoitteisille henkilöille kuin henkilöille, joilla vammaa tai toimintarajoitetta ei ole. (WAI 2021.) Kriteereitä on yhteensä 78 ja ne on jaettu kolmeen eri vaatimusluokkaan, jotka matalimmasta korkeimpaan ovat A, AA ja AAA. Kuitenkaan edes korkeimman AAA-tason noudattaminen ei tee verkkosisällöstä absoluuttisen saavutettavaa. (Kirkpatrick ym. 2018).

Viimeisenä WCAG-ohjeistuksen neljästä tasosta ovat tekniikat, joita voi hyödyntää ohjeiden ja onnistumiskriteerien tulkitsemisessa ja toteuttamisessa (Kirkpatrick ym. 2018). Huomionarvoista tässä on tekniikoiden hyödyntämisen vapaaehtoisuus. W3C painottaa onnistumiskriteereihin keskittyvällä verkkosivullaan, että tekniikoiden toteutumisen ei koskaan pitäisi olla vaatimuksena toisin kuin kriteerien toteutumisen. (WAI 2021.) Kriteerien toteuttajalle jätetään siis laajasti harkinnanvaraa siihen, millä tekniikoilla ne saavutetaan.

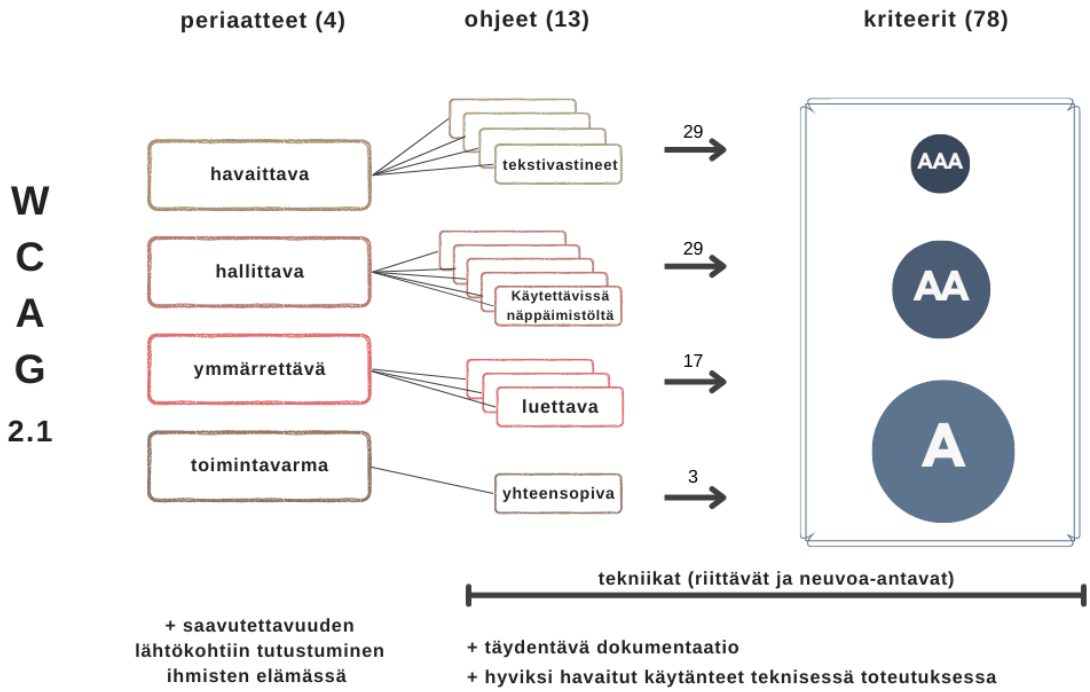
Neuvoa-antavien tekniikoiden toteuttaminen kuitenkin lisää saavutettavuutta verrattuna tilanteeseen, jossa hyödynnetään pelkästään testattavia kriteerejä. Tämä johtuu siitä, että neuvoa-antavat tekniikat ovat tarkempia sekä kattavat suuremman joukon saavutettavuusongelmia kuin itse onnistumiskriteerit. (Kirkpatrick ym. 2018.) Riittävät tekniikat sen sijaan kuvaavat tapoja, joilla onnistumiskriteerit voidaan täyttää (WAI 2021.) Ne siis tukevat suoraan testattavien kriteerien täyttymistä.

Jotta saavutettaisiin mahdollisimman monien käyttäjien tarpeisiin vastaava saavutettavuuden taso, kannattaa verkkosisällön tuottajan pyrkiä huomioimaan sekä riittävät että neuvoa-antavat tekniikat. Näiden lisäksi W3C suosittelee etsimään teknisiä toteutustapoja myös WCAG:n ulkopuolisista, hyväksi havaituista käytänteistä. (Kirkpatrick ym. 2018.)

Varsinainen WCAG 2.1 -ohjeistuksen lisäksi W3C on tuottanut ohjeistusta täydentäviä dokumentteja. Ne kuvaavat muuttuvien teknologioiden mahdollisia tapoja soveltaa WCAG:ta. (Kirkpatrick ym. 2018.) Täydentävä dokumentaatio sisältää kolme kokonaisuutta, jotka linkittyvät toisiinsa. Näitä ovat “How to Meet WCAG 2”, “Understanding WCAG 2” ja “Techniques for WCAG 2”. Lisäksi on olemassa näiden dokumenttien välisiä suhteita kuvaava ”WCAG 2 Documents”. W3C suosittelee yksityiskohtaisen ymmärryksen saavuttamiseksi tutustumaan saavutettavuuden lähtökohtiin ja merkitykseen konkreettisesti siitä erityisesti hyötyvien ihmisten elämässä. (Henry 2021d.)

Kaikki kriteereistä eivät koske sisältöä, jota löytyisi päivämääräkentästä. Sen vuoksi lista kriteereistä tulee komponenttia kehitettäessä olemaan hieman alkuperäistä kriteeristöä suppeampi. Olennaisiksi arvioidut kriteerit on kerätty liitteistä löytyvään taulukkoon, johon

merkitään kehityksen ja lopputestauksen aikana, toteutuuko kriteeri komponentin sisällön kohdalla. Epäolennaisten kriteerien lisäksi pois jäävät AAA-tason kriteerit, joita on suhteellisesti myös vähiten.



Kuva 1. Kaavio WCAG 2.1-ohjeistuksen sisällöstä ja osien välisistä suhteista sekä ohjeistukseen liittyvät muut dokumentit ja tavat saada tietoa saavutettavuuden vaatimuksista.

Yksityiskohtaisempia ohjeita varten sovelluskehittäjille suositellaan erityisesti ”Techniques for WCAG 2” -dokumenttia. Se sisältää muun muassa tietoa siitä, kuinka eri tavoin vammaiset tai toimintarajoitteiset ihmiset voivat hyötyä ohjeesta tai onnistumiskriteeristä, esimerkkejä vaatimusten toteutuksen HTML-rakenteesta ja tietoa selaimen sekä avustavien teknologioiden tuettavuudesta. Varsinaisesta WCAG-ohjeistuksesta pääsee linkkien kautta täydentävään dokumentaatioon. (Henry 2021d.)

Euroopan Unionin tasolla on määritelty oma standardi EN 301549 (Accessibility requirements for ICT products and services), jonka tarkoituksena on yhtenäistää sekä luoda yksityiskohtaiset ja mitattavissa olevat vaatimukset julkisen sektorin hankkiman tieto- ja viestintätekniikan saavutettavuudelle Euroopassa. Standardissa huomioidaan maailmanlaajuiset aloitteet saavutettavuuden edistämiseksi. Tämä tarkoittaa sitä, että myös WCAG on huomioitu hyvin laajasti standardissa. Verkkosivujen osalta on määritelty vähimmäisvaatimustasoksi WCAG:n taso AA. (European Telecommunications Standards Institute 2015, 15, 39; European Telecommunications Standards Institute 2021, 51.)

EN 301549 -standardin vaatimukset ovat perustana saavutettavuusdirektiivissä (Rahkola 2017). Tätä ohjeistusten, standardien ja direktiivien jatkumoa tarkastellessa voidaan nähdä, että WCAG-ohjeistuksen vaatimukset kulkevat mukana aina Suomen kansalliseen lainsäädäntöön asti.

3.4.2 WAI-ARIA

WCAG:n lisäksi toinen erittäin oleellinen standardi päivämääräkomponentin rakentamisessa on WAI-ARIA eli Web Accessibility Initiative - Accessible Rich Internet Applications. Se määrittelee tapoja tehdä verkkosisällöstä ja -sovelluksista saavutettavampaa (Cooper 2020). WAI-ARIA tarjoaa pääasiassa ohjelmistokehittäjille keinoja tehdä saavutettavuudeltaan monipuolista sisältöä (W3C 2021d). Sen toimita perustuu HTML-elementeille lisättävien attribuuttien ja roolien käyttöön. Niiden avulla välitetään esimerkiksi avustavalle teknologialle tietoa elementtien tarkoituksesta ja toiminnoista. (Cooper 2020, MDN Web Docs 2022h).

WAI-ARIAn käyttö tulee tarpeelliseksi erityisesti silloin, kun verkkosivun rakentamisessa on käytetty elementtejä, jotka eivät suoraan vastaa käyttötarkoitustaan. HTML:n uusin versio HTML5 toi mukanaan joitakin parannuksia, jotka vähentävät tarvetta käyttää WAI-ARIAa. Se toi mukanaan elementtejä, joissa on valmiiksi sisäänrakennettuna osasia, joita WAI-ARIAn avulla tavallisesti luodaan. Näitä ovat esimerkiksi näppäimistötukeen, roolien ilmaisemiseen sekä tilasta kertovan tiedon välittämiseen liittyvät määritykset.

Ohjelmistokehittäjien tulisi suosia natiiveja HTML-elementtejä aina, kun kulloinkin kyseessä olevaan käyttötarkoitukseen sellainen vain löytyy. (MDN Web Docs 2022h.) Päivämääräkomponentissa osan elementeistä voi tehdä ilman WAI-ARIAa, mutta erityisesti visuaalisen kalenterinäkymän kohdalla WAI-ARIAn merkitys on suuri, koska siihen löytyy vain vähän valmiita HTML-elementtejä. Esimerkiksi painikkeille sellainen kuitenkin löytyy.

Saavutettavuusstandardeista WCAG on toiminut myös lainsäädännön perustana monissa maissa. Seuraavassa kappaleessa tutustutaan muun muassa siihen, mitä osia onnistumiskriteereistä laki velvoittaa huomioimaan verkkosisältöä kehitettäessä.

3.5 Saavutettavuutta koskeva sääntely

Monissa maissa saavutettavuus on lailla turvattu ainakin julkisissa palveluissa. Tämä tarkoittaa sitä, että ”saavuttamattomuus” saattaa johtaa syytteisiin ja näin aiheuttaa kuluja organisaatiolle. (Roggio 2015.) Kuitenkin se, mitkä yhteiskunnan toimijat ovat velvoitettuja noudattamaan saavutettavuutta, vaihtelee maittain. (Aalen 2018). Käsittelen tässä luvussa aihetta kuitenkin pääasiassa Suomea suoraan koskevan sääntelyn näkökulmasta,

koska eri maiden sääntelyn yhtäläisyyksien ja eroavaisuuksien käsittely laajentaisi aihetta liikaa.

Suomessa toteutettavaan saavutettavuuteen vaikuttavat lisäksi esimerkiksi Euroopan unionin tasolla luotavat direktiivit sekä kansainväliset sopimukset. Pyrin tuomaan esille ne muutokset, jotka ovat vaikuttaneet siihen, että saavutettavuutta edistetään nykyisessä laajuudessaan. Sääntely ohjaa komponentin kehitysprosessia erityisesti siltä kannalta, että noudatettavien vaatimusten minimitaso on säädetty laissa.

Euroopan unionin tasolla on vuoden 2016 lopulla julkaistu Euroopan parlamentin ja neuvoston direktiivi ”julkisen sektorin elinten verkkosivustojen ja mobiilisovellusten saavutettavuudesta” (myöh. saavutettavuusdirektiivi) (2016/2102/EU). Direktiivi tukee Euroopan digitaalistrategiaa sekä YK:n vammaissopimuksen täytäntöönpanoa jäsenmaissa. (Lewthwaite & James 2020, 1361.) Saavutettavuusdirektiivi koskee julkisen hallinnon sekä julkista hallintotehtävää hoitavien organisaatioiden verkkosivustoja ja mobiilisovelluksia. (Terveyden ja hyvinvoinnin laitos 2021.)

Euroopan tasolla saavutettavuusdirektiivi velvoittaa siis julkisen sektorin toimijoita. Suomessa laki digitaalisten palvelujen tarjoamisesta (myöh. digipalvelulaki) kuitenkin määrittelee laajemman joukon toimijoita saavutettavuusvaatimusten piiriin. Digipalvelulain vaatimukset koskevat viranomaisia, julkisoikeudellisia laitoksia, joitakin järjestöjä sekä osaa yksityisestä sektorista. Yksityisen sektorin toimijoista esimerkkejä ovat pankit, vakuutusyhtiöt ja Posti. Järjestöjen kohdalla saavutettavuusvaatimusten piiriin kuulumiseen vaikuttavat esimerkiksi viranomaiselta saatavat avustukset. (Aluehallintovirasto s.a.f.) Vaikka laki velvoittaa nykypäivänä vain rajatun osan toimijoista tarjoamaan käyttäjille saavutettavia digipalveluita, inhimillinen tarve saavutettavuudelle koskettaa koko palveluiden laajaa kirjoa. Esimerkiksi Norjassa on jo nykypäivänä yksityiset yritykset velvoitettuja noudattamaan saavutettavuusvaatimuksia (Aalen 2018).

Suomessa digipalvelulain yhtenä tarkoituksena on edistää digitaalisten palvelujen sisällön saavutettavuutta. Tällä pyritään parantamaan digipalvelujen käytön yhdenvertaisuutta. Lisäksi lailla saatetaan saavutettavuusdirektiivi (2016/2102/EU) osaksi kansallista lainsäädäntöä. (digipalvelulaki 2019/306.) Digipalvelulaki määrää saavutettavuuden minimitasoksi senhetkisen WCAG-kriteeristön AA-tason, joka kattaa myös A-tason onnistumiskriteerit. Näin ollen lain mukaan täytettäviä kriteereitä on yhteensä 49, kun AAA-tason tai suorien lähetysten tekstityksiä koskevan kriteerin 1.2.4 toteutumista ei velvoiteta. (Aluehallintovirasto s.a.e.) Noudatan tätä yleisesti käytössä olevaa saavutettavuuden minimitasoa komponentin kehityksessä. Seuraavassa luvussa siirryn askelen lähemmäksi

komponentin konkreettista toteutusta, kun aiheessa siirrytään verkkosivujen kehitykseen saavutettavuuden näkökulmasta.

3.6 Saavutettavuuden huomiointi verkkosivujen kehityksessä

Saavutettavuuden huomiointi verkkosivujen kehityksessä on laaja kokonaisuus. Siksi rajaan tämän luvun koskemaan nimenomaan verkkosivujen käytännön toteutusta, enkä esimerkiksi syvenny toimintaympäristössä vallitsevien toimintatapojen, tietoisuuden, koulutuksen, työnjaon, budjetin tai asenteiden merkitykseen tai niiden kehittämiseen. Tiedostan kuitenkin näiden merkityksen saavutettavuuden toteuttamisessa.

3.6.1 Vaatimusten määrittely ja suunnittelu

Aiemmin opinnäytetyössä käytiin läpi sitä, että on tärkeää valmistautua saavutettavuusmuutoksiin perehtymällä sen tosiasiallisiin hyötyihin vammaisten ja toimintarajoitteisten ihmisten elämässä. Tämän lisäksi alussa on hyvä tutustua toimintaympäristöön, jossa saavutettavuusmuutoksia on tarkoitus toteuttaa. Jotta edessä olevan työn määrästä on helpompaa tehdä arvio, on hyödyllistä ottaa selvää toimintaympäristön, kuten organisaation, senhetkisistä saavutettavuuteen liittyvistä tavoista ja tottumuksista.

Myös toimintaympäristöön kohdistuva, saavutettavuutta koskeva eritasoinen sääntely on tärkeää tuntea. Näiden lisäksi valmistautumisessa on kannattavaa panostaa jo olemassa oleviin tai tuleviin verkkosivuihin ja niiden merkitykseen käyttäjien näkökulmasta.

Saavutettavuutta koskevien selkeiden tavoitteiden asettaminen auttaa suunnittelemaan ja priorisoimaan työtä sekä määrittelemään työhön käytettävää aikaa. Tavoitteet voivat kohdistua niin verkkosisällön saavutettavuuteen ja laadunvarmistukseen kuin yleisesti työnkulkuun vaikuttaviin tekijöihinkin. (White, Abou-Zahra, & Henry 2016a.)

Saavutettavuusmuutosten suunnittelussa on tärkeää käydä läpi toimintaympäristön tarjoamat resurssit saavutettavuuden toteutukseen. Esimerkiksi työkalut saavutettavan sisällön luomiseen, ylläpitämiseen ja testaamiseen sekä näiden yhteensopivuus vaikuttavat merkittävästi saavutettavuuden edistämisprosessiin. Mikäli saavutettavuusmuutokset kohdistuvat olemassa olevaan verkkosisältöön, on sivustojen muutostarpeet hyvä arvioida tavoitteiden pohjalta.

Suunnitteluvaiheessa merkityksellistä on myös kyseiseen projektiin sopivista arviointi- ja raportointikäytänteistä päättäminen. Näiden tarkoituksena on havainnollistaa saavutettavuuden edistymistä sekä saavutettavuuden tason pysymistä myös ylläpitovaiheessa. Yksi

esimerkki, jolla voidaan mitata saavutettavuutta, ovat juuri aiemminkin tässä opinnäytetyössä esiin tulleiden WCAG-kriteerien toteutuminen. (White, Abou-Zahra, & Henry 2019.)

3.6.2 Toteutus

Oletetaan, että kuvitellun organisaation työntekijät ovat aktiivisesti kehittäneet ymmärrystään ja käytännön osaamistaan ja organisaation sisäiset prosessit tukevat saavutettavuuden toteutusta. Seuraavassa vaiheessa on aika alkaa toteuttaa saavutettavuutta konkreettisesti. On tärkeää, että tässä vaiheessa tavoitteet ohjaavat toimintaa ja vastuualueet ja niistä vastaavat henkilöt ovat tekijöiden tiedossa. Vastuualueet voivat jakautua eri tavoin, mutta on tärkeää, että ne ovat kaikille selkeitä. Tuen saaminen tarvittaessa sekä toiminta-ohjeiden selkeys ongelmatilanteissa auttavat tilanteiden selvittämisessä ja hallitussa ratkaisussa.

Myös toteutusvaiheeseen kuuluu saavutettavuusarviointien tekeminen. Arviointia voi toteuttaa jatkuvana prosessina, jonka merkitys korostuu esimerkiksi silloin kun saavutetaan tiettyjä kehityksen vaiheita. Opinnäytetyön toiminnallisen osion teeman mukaisesti on hyvä mainita, että myös käytettävien erillisten komponenttien saavutettavuus on yhtä lailla tärkeää arvioida kuin muunkin verkkosisällön. Havaittuihin ongelmiin tarttuminen mahdollisimman varhaisessa vaiheessa tyypillisesti edesauttaa riskien hallintaa ja hillitsee resurssien kulumista. (White, Abou-Zahra, & Henry 2018.)

Arviointia tulee tehdä jatkuvasti tuotannon ja ylläpidon eri vaiheissa. Tässä on hyvä huomioida erilaisten päivitysten mahdolliset vaikutukset verkkosisällön saavutettavuuteen sekä valmiiksi suunnitellut tavat korjata mahdollisesti eteen tulevat saavutettavuusongelmat. Vamman tai toimintarajoitteen kanssa elävien ihmisten osallistuminen arviointiprosessiin ja testaukseen voi tuoda konkreettista tietoa saavutettavuusongelmien aiheuttamista tilanteista koko tiimille.

Standardimallit raportoinnissa mahdollistavat eri verkkosisältöjen saavutettavuuden vertailun sekä osoittavat tietyn palvelun kehittymisen elinkaaren aikana, jos sen eri versioista on saatavilla raportteja. Haasteiden esiin tuominen edistää sitä, että niiden yli pääsemiseksi varataan riittävästi resursseja.

Priorisoimalla käytettävissä olevia resursseja saavutettavuusongelmien ratkaisemiseen niiden vaikuttavuuden ja haastavuuden mukaan, voidaan saavutettavuuden edistymistä tehostaa. W3C suosittelee nostamaan työjonon kärkeen merkittävästi saavutettavuuteen vaikuttavat, mutta helposti korjattavat ongelmat. Samalla se kuitenkin muistuttaa, että kaikki asetetut tavoitteet tulee saavuttaa.

Myös hyvän kommunikaation avulla voi edistää saavutettavuuden toteutumista. Saavutettavuustavoitteiden täyttämisessä käytettyjen hyvien käytänteiden sekä myös epäonnistumisten jakamisella voidaan edistää saavutettavuuden ymmärrettävyyttä ja edesauttaa osaamisen ja asiantuntijuuden lisääntymistä tiimissä ja myös laajemmin. (White, Abou-Zahra, & Henry 2019.)

3.6.3 Ylläpito

Verkkosisällön säännöllinen tarkistaminen auttaa saavutettavuuden ylläpidossa. Samoin saavutettavuuden toteuttamiseen liittyvien prosessien ja käytettävissä olevien resurssien vaaliminen edesauttaa mahdollisesti tarvittavien muutosten tekemistä. Teknologioiden muuttuessa ja kehittyessä niitä tulee aika-ajoin päivittää. Esimerkiksi selaimen päivitetty versio saattaa tuoda mukanaan saavutettavuutta entistä paremmin tukevia ominaisuuksia. Muutostarvetta saavutettavuuteen saattavat aiheuttaa myös uudistukset standardeissa tai lain asettamat uudet vaateet, käyttäjien antamaa palautetta unohtamatta. (White, Abou-Zahra, & Henry 2016b.)

3.6.4 Arviointi ja testaus

Verkkosivuston saavutettavuuden arvioimiseen on olemassa useita eri tapoja. Arvioinnissa kannattaa hyödyntää useampaa eri menetelmää, jotta se tulee tehtyä mahdollisimman kattavasti. Eri menetelmät tukevat ja täydentävät toisiaan.

Saavutettavuuden arviointia on suositeltavaa tehdä kaikissa eri kehityksen vaiheissa kehityksen suunnittelusta alkaen. Kun saavutettavuuden arviointiin on panostettu varhaisessa vaiheessa, saattaa se säästää vaivaa ja kustannuksia. Arvioinnin ei tulisi loppua siihen, kun verkkosivusto on julkaisuvalmis. Jatkokehityksessä on yhtä lailla tärkeää huomioida, että uusi sisältö on saavutettavaa. (Papunet 2021d.)

Kuten aiempien, verkkosivujen kehityksen eri vaiheista kertovien kappaleiden sisällöstä voi päätellä, saavutettavuuden arviointi on tavalla tai toisella osa niistä jokaista. Se kietoutuu moniin muihin muutoksiin ja uudistuksiin esimerkiksi verkkosivujen rakenteessa, ohjelmointikielessä tai verkkoselaimessa. Saavutettavuuden implementointi ja ylläpito voivat uusien teknologioiden ansiosta helpottua, mutta joskus nämä voivat myös hankaloitua. Näin on käynyt uusien komponenttipohjaisten JavaScript-kirjastojen kanssa.

WAI tarjoaa useanlaista apua testauksen ja arvioinnin tueksi. Verkkosivuilta löytyy myös ei-teknisille henkilöille suunnattu, kevyeen saavutettavuusarviointiin soveltuva lista tarkistettavista yksityiskohdista sekä ohjeet näiden toteuttamiseen. Tarkkaan WCAG-

ohjeistuksen kriteerien noudattamista tarkastelevaan arviointiin sen sijaan on tarjolla Website Accessibility Conformance Evaluation Methodology eli WCAG-EM sekä siihen liittyvät raportointikäytänteet. (Henry 2021d.)

Lisäksi WAI ylläpitää eri toimijoiden tarjoaman saavutettavuustyökalun listaa. Vaihtoehtoja löytyy, sillä sivustolla on listattuna yli 100 työkalua, joista voi lisäksi eri kriteerien perusteella suodattaa omiin tarpeisiinsa sopivimmat. Työkalujen toimintaperiaate vaihtelee online-työkaluista erillisiin ohjelmistoihin. Niiden tarkoituksena on avustaa standardienmukaisen verkkosisällön määrittelyssä. (Henry 2021d.) Työkaluista voi olla suurtakin apua saavutettavuusongelmien määrittelyssä. Arviointia ei kuitenkaan voi jättää pelkästään automaatiotyökalujen varaan, vaan ihmisen harkintaa tarvitaan lopullisten ratkaisujen tekemiseen. (Abou-Zahra, Steenhout & Keen 2017.)

Työkalujen, ohjeistuslistojen ja raportoinnin lisäksi WAI suosittelee testaukseen avuksi ihmisiä, joilla on omakohtaista kokemusta erilaisista tavoista käyttää verkkosisältöä. Tämä monipuolistaa arvioinnissa käytettävää konkreettista tietoa esimerkiksi avustavien teknologioiden strategioista, joilla ne mukautuvat antamaan ja saamaan tietoa verkkosisällöstä. (Education and Outreach Working Group 2021.) Eli jos esimerkiksi kehittäjä ei itse käytä verkon selaamiseen ja käyttöön avustavia teknologioita, on hänen kannattavaa pyytää testaukseen avuksi henkilö, jolla on omakohtaista kokemusta avustavan teknologian käytöstä.

3.7 Avustavat teknologiat

Uuden-Seelannin opetusministeriö on listannut omilla verkkosivuillaan esimerkkitapauksia 6–13-vuotiaista oppilaista, jotka käyttävät avustavaa teknologiaa oppimisen tukena. Jokaisen oppilaan kohdalla on kerrottu, kuinka avustavan teknologian käyttö on kuuden kuukauden aikana vaikuttanut tämän mahdollisuuksiin toimia oppimisympäristössään. Kertomuksista käy ilmi, että avustava teknologia on tuonut apua jokaiselle esimerkeissä esiintyvälle oppilaalle. Esimerkiksi lukihäiriötä sairastavan 12-vuotiaan oppilaan kirjoitustuotokset ovat monipuolistuneet näppäimistökirjoituksen ja sanoja automaattisesti täydentävän ohjelmiston avustuksella. (Ministry of Education 2021.) Tässä luvussa selvitän avustavien teknologioiden erityispiirteitä ja erilaisia käyttötarkoituksia. Tavoitteena on saada käsitystä siitä, millä tavoin ohjelmistokehittäjä voi koodissaan huomioida erilaisten avustavien teknologioiden toimivuuden verkkosivuilla.

Staattisilla verkkosivuilla avustavat teknologiat toimivat tavallisesti hyvin, kunhan niissä on käytetty HTML-standardeja. Monimutkaisia, dynaamisia sivustoja luotaessa on myös erityisen hyvä tietää avustavista teknologioista. (Papunet 2021e.) Verkkosivuilla lähdekoodin

rakenne ja sisältö määrittelevät, tukeeko se saavutettavuutta ja avustavia teknologioita. Termi avustavat teknologiat kattavat sekä käyttölaitteisiin asennettavia ohjelmistoja että kokonaan erillisiä laitteita. Tämäntyyppiset ohjelmistot alkavat olla yhä yleisemmin myös oletuksena selaimissa ja esimerkiksi mobiililaitteissa. (Aluehallintovirasto s.a.c.)

Kehitysvammaliiton ylläpitämällä Papunet-verkkosivustolla on listattuna erilaisia avustavia teknologioita. Näitä ovat esimerkiksi kytkinohjaimet, ruudunlukuohjelmat ja sivun sisällön esitystavan muuttaminen. (Papunet 2021e.) Esittelen kolme erilaista avustavaa teknologiaa hieman tarkemmin. Ensimmäinen, näppäimistöselaaminen lienee suurelle yleisölle avustavista teknologioista tutuimpia. Sen sijaan kohdistin, jota ohjataan ruudulla silmän liikkeillä, saattaa olla monille tuntemattomampi.

Vaikka näppäimistö on hyvin paljon käytetty vuorovaikutuksen osa tietokoneen ja ihmisen väillä, myös näppäimistöselaaminen lasketaan avustaviin teknologioihin. Tällä tarkoitetaan tilannetta, jossa käyttäjä tekee myös oletusarvoisesti hiirellä tehtävät toiminnot pelkkää näppäimistöä käyttäen. Tämä voi johtua esimerkiksi näkökyvyn rajoitteista, jolloin ei näe hiiren kohdistinta näytöllä tai käsien rajoittuneesta toiminnasta, jolloin hiiren käyttö on hankalaa tai mahdotonta. (Papunet 2021f.)

Ruudunlukijat nimensä mukaisesti lukevat käyttäjälle ääneen käyttöliittymän sisältöä. Ruudunlukuohjelmia käytettäessä tietokonetta ohjataan näppäimistöltä. Jos kyseessä on kosketusnäytöllinen laite, sitä ohjataan sormia käyttäen. (Papunet 2021g.)

Jotta näppäimistöselaaminen olisi mahdollista verkkosivuilla, tulee ohjelmakoodin tekijän tai muokkaajan lisätä koodiin mahdollisuus suorittaa kaikki tarpeelliset toiminnot näppäimistöä käyttäen. Esimerkiksi päivämäärän lisäyksessä tämä voi tarkoittaa mahdollisuutta kirjoittaa haluamansa päivämäärä tekstikenttään. Käyttäjää ei siis tässä tapauksessa pakotettaisi avaamaan visuaalista kalenteria esimerkiksi kalenteri-ikonista. Kuitenkin jos avaaminen on välttämätöntä, se tulisi pystyä tekemään fokusoimalla avauspainike ja painamalla enter-näppäintä. Siirtyminen eri kenttien välillä tapahtuu tyypillisesti sarkainpainikkeella.

Jos henkilö ei pysty lainkaan liikuttamaan käsiään, hänen voi olla kuitenkin mahdollista käyttää tietokonetta silmän liikkeitä hyödyntämällä. Tämä tarkoittaa sitä, että ruudulla näkyvää kohdistinta ohjataan katsetta liikuttamalla silmänliikeantureiden avulla. Kohteen klikkaamiseen voidaan käyttää ulkoista laitetta, kuten käyttäjälle soveltuvaa kytkintä. Myös Dwell-toiminnon käyttäminen, jossa ohjelmallisesti toteutetaan toiminto katseen viipyessä riittävän kauan painikkeessa, on mahdollinen. (Papunet 2021h.)

Avustavia teknologioita on monenlaisia ja niiden vaikutus mahdollisuuksiin suorittaa erilaisten toimintoja verkossa voi olla käyttäjilleen hyvin merkittävä. Ohjelmistokehittäjänä on tärkeää tietää, millä tavoin lähdekoodista voi tehdä mahdollisimman hyvin yhteensopivaa avustavien teknologioiden kanssa.

Avustava teknologia käyttää toimiakseen pääasiassa tiedon merkitykseen perustuvaa (semanttista) informaatiota, joka jättää ulkopuolelleen tyylimääritelmiin ja JavaScriptiin liittyvän informaation. Mikäli kuitenkin semanttinen informaatio puuttuu eli natiivit HTML-elementit eivät tätä tarjoa, voidaan informaatiota lisätä WAI-ARIAn avulla.

Käyttöjärjestelmät tarjoavat verkkoselainten käyttöön saavutettavuusrajapintoja (engl. accessibility APIs). Rajapinnan kautta avustava teknologia saa käyttöönsä edellisessä kappaleessa kuvattua informaatiota puumaisen mallin muodossa. (engl. accessibility tree). Esimerkiksi ruudunlukija siis toimii verkkoselaimessa, mikäli sillä on käytettävänään saavutettavuusrajapinta, josta se saa tarvitsemansa informaation. Tähän käyttöön sopivia verkkoselaimia löytyy yleisimmin käytössä olevista käyttöjärjestelmistä yhdestä kahteen kappaletta. Sen lisäksi ruudunlukijoiden valmiudet hyödyntää WAI-ARIAa monipuolisesti vaihtelevat lukijoiden välillä. (MDN Web Docs 2022i; MDN Web Docs 2022j.)

Olisi kiinnostavaa perehtyä tarkemmin siihen, kuinka verkkoselain välittää semanttista informaatiota saavutettavuusrajapinnoille ja minkälaisia eroja näissä rajapinnoissa ilmenee. Syvempi perehtyminen aiheeseen rajataan kuitenkin tämän opinnäytetyön ulkopuolelle. Seuraavaksi siirrytään lähemmäksi konkreettista saavutettavuuden ohjelmallista toteuttamista käymällä läpi komponentin rakentamisen pohjana käytettäviä teknologioita. Niiden ja WAI-ARIAn avulla syntyy lopullinen kokonaisuus kehitystyön päätteeksi.

4 Verkkosivujen kehitys Vue-kirjastolla – arkkitehtuuri ja teknologiat

Tässä luvussa käyn läpi saavutettavan Vue-komponentin rakentamiseen liittyviä teknologioita. Lisäksi kerron hieman Vue.js-sovelluskehyksen arkkitehtuurista. Vue-kirjaston valikoitumiseen saavutettavan komponentin kehityksessä käytettäväksi vaikuttivat erityisesti kolme seikkaa. Ensinnäkin Vuen kaltaiset JavaScript-sovelluskehykset ovat hyvin suosittuja sovelluskehittäjien keskuudessa opinnäytetyön kirjoittamisen hetkellä. Vue kuuluu viiden käytetyimmän web-sovelluskehyksen joukkoon vuoden 2021 mittausten mukaan. Niiden mukaan se on lisäksi toisiksi halutuimmaksi osoittautunut web-sovelluskehys heti Reactin jälkeen. (Daws, 2021.)

Toisekseen minulle oli muodostunut käsitys siitä, että suosiostaan huolimatta Vuen komponenttien saavutettavuus ei ole itsestäänselvyys ja valmiisiin, kolmansien osapuolten tekemiin komponentteihin voi olla haastavaa tehdä muutoksia. Lisäksi, koska Vuen pohjalta rakennetut sovellukset tyypillisesti nojaavat paljon ulkoisten, valmiiden komponenttien varaan, näin näiden komponenttien kehityksessä merkittävän kehityshaasteen sovelluskehittäjille. Kolmas syy Vuen valikoitumiselle oli oma kokemukseni siitä esimerkiksi töiden kautta sekä halu kehittyä sen käytössä.

Ennen Vuen erityispiirteisiin syventymistä käydään läpi mitä tarkoittavat jo aiemmin opinnäytetyössä mainitut HTML, CSS ja JavaScript. Ne ovat myös Vue-komponenttitiedostojen perusosasia ja niitä eri tavoin käyttämällä voidaan muodostaa haluttu kokonaisuus ja sen toiminnallisuudet.

4.1 HTML

Verkkosivuston HTML-dokumentin rakenne määrittelee, mitä sisältöjä verkkoselain muodostaa käyttöliittymään. HTML:n perusosia kutsutaan elementeiksi. (MDN Web Docs 2022a.) Elementtien tunnisteet vaihtelevat sen mukaan, mitä sisältöä selaimessa halutaan näyttää. Ne voivat myös olla sisäkkäisiä, riippuen elementin luonteesta.

HTML-elementti alkaa kulmasulkeisiin kirjoitetulla tunnisteen määritelmällä. Elementtien tunnisteita ovat esimerkiksi <head>, <body>, , , <nav> tai <img>. (MDN Web Docs 2021a.) Esimerkiksi jos käyttöliittymässä halutaan näyttää tekstikappale, HTML-dokumenttiin kirjoitetaan tekstikappaleen alkuun . Tekstikappale loppuu seuraavaan -merkintään. Elementeillä voi olla lisäksi attribuutteja. Niiden tarkoituksena on tehdä tarkempia määrityksiä sille, kuinka elementti halutaan näyttää verkkoselaimessa. (MDN Web Docs 2021a.)

4.2 CSS

CSS eli Cascading Style Sheets on verkon sisällön tyylien muokkaukseen käytettävä kieli, joka ei lukeudu ohjelmointikieleksi. CSS :n avulla voidaan määritellä, mihin HTML-elementteihin muutokset halutaan kohdistaa ja millä tavalla erilaiset mediasisällöt halutaan hahmontaa (engl. render). Toimiakseen CSS-sääntöjen tulee olla yhdistettynä haluttuun HTML-mallipohjaan. (MDN Web Docs 2022b, MDN Web Docs 2022c.) CSS:n avulla voidaan myös määritellä erilaisia näkymiä eri kokoisille näytöille. HTML:n lisäksi CSS toimii kaikkien XML-pohjaisten merkintäkielien kanssa. (W3C 2016.)

4.3 JavaScript

JavaScript on maailman eniten käytetty ohjelmointikieli, jota on tyypillisesti käytetty selainpuolen komentosarjoissa. Sitä voidaan kuitenkin käyttää myös palvelinpuolen ajoympäristöissä, joista esimerkkinä Node.js. JavaScriptillä voidaan käsitellä verkkosisältöä monipuolisesti ja sen avulla on mahdollista vaikuttaa sekä sivun HTML- että CSS-sisältöön. (MDN Web Docs 2022d; W3Schools 2022b; W3Schools 2022c.)

4.4 Vue

Vue on MIT-lisensoitu, JavaScript-ohjelmointikieleen pohjautuva progressiivinen sovelluskehys (Vue.js 2022). Sen peruspilareita ovat JavaScriptin lisäksi HTML- ja CSS-standardit (Vue.js s.a.c.). Vuea käytetään ohjelmoinnissa niin sanottuun front-end- eli selainpään ohjelmointiin. Se toimii samalla teknologiana, joka erottaa back-end- eli palvelinpuolen sisällön selainpäästä. (Nian Li & Bo Zhang 2021, 1.) Vue voidaan sovelluskehyksen ohella mieltää kirjastoksi. Sitä voidaan myös hyödyntää osana toisia kirjastoja ja projekteja. (Nian Li & Bo Zhang 2021, 4.) Sen merkitys projektissa voi siis vaihdella suurestikin.

Vuella rakennetaan erilaisia käyttöliittymiä ja komponentit ovat oleellinen osa sen ohjelmointimallia (Vue.js s.a.c.). Lisäksi Vuen rakennetta kuvaava malli on SPA eli Single Page Application. Se tarkoittaa, että kehitettävässä projektissa on vain yksi varsinainen HTML-sivu, jonka sisälle ladataan muut osat. (Nian Li & Bo Zhang 2021, 2.) Virtuaalisen DOMin ansiosta käyttöliittymästä on mahdollista päivittää elementtikohtaisesti siten, että muutosten päivittäminen ei vaadi koko sivun uudelleenlatautumista. Tämä edistää kirjaston suorituskykyä. (Vainio 2018, 24.)

4.4.1 Arkkitehtuuri

Data kulkee Vue-sovelluksessa niin kutsutun Model-View-ViewModel-arkkitehtuurimallin mukaisesti. MVVM-malli on pyrkimys vastata tarpeeseen parantaa kehityksen tehokkuutta sekä

vähentää ylläpitokustannuksia, kun selainpuolella suoritettavat operaatiot muuttuvat yhä monimutkaisemmiksi.

Tietojen käsittelyyn liittyvä logiikka alkaakin painottua nykyään yhä enemmän selainpäähän, jolloin sen toimintakyvyltä myös odotetaan entistä enemmän. MVVM-arkkitehtuurimalli on yksi keino pyrkiä vastaamaan tähän haasteeseen. MVVM kuvaa tapaa, jolla data liikkuu käyttöliittymässä ja se voidaan jakaa kolmeen osaan. Ne ovat näkymä, malli sekä näiden väliin sijoittuva näkymämalli eli näkymän ja mallin yhdistävä osa. Näkymämalli erottaa käyttäjälle näkyvän osan taustalla olevasta mallista, jolloin ne eivät suoraan kommunikoi toistensa kanssa. (Nian Li & Bo Zhang 2021, 1–2.)

Ville Vainio tarkentaa pro gradu -tutkielmassaan ”Järjestelmä web-palveluiden nopeaan kehitykseen”, että Vuen toimintalogiikka kohdistuu nimenomaan näkymämalliin (viewmodel). Taustalla Vue-instanssi päivittää mallia sekä vastaanottaa siltä ilmoituksia. Toisessa suunnassa Vue-instanssi sen sijaan kytkeytyy näkymän dokumenttioliomalliin eli DOMiin (engl. Document Object Model). (Vainio 2018, 23.) DOM kuvastaa dokumentin, esimerkiksi HTML-verkkosivun, puumaista rakennetta muistissa (MDN Web Docs 2022e). DOM on siis ohjelmointirajapinta, joka yhdistää verkkosivun sisältämät elementit ohjelmointikieliin. (MDN Web Docs 2022f.)

Yksi Vuen ominaispiirteistä on, että se ylläpitää kahdenlaista, niin sanotusti oikeaa sekä virtuaalista DOMia. Tätä mallia käyttävät myös muut sovelluskehykset ja React on ollut sen toteuttamisessa edelläkävijä. Mallissa käyttöliittymä renderöinti perustuu virtuaaliseen DOMiin, joka kuvastaa oikean DOMin ihanteellista kuvaa muistissa. Virtuaalinen DOM synkronoidaan oikean DOMin kanssa. Tämä mahdollistaa sen, että ohjelmoija voi keskittyä haluttujen näkymien luontiin ilman DOMin suoraa käsittelyä, joka tapahtuu ajonaikaisen renderöinnin aikana. Tätä kutsutaan deklaratiiviseksi ohjelmoinniksi. (MDN Web Docs 2022g.)

4.4.2 Komponentit

Komponentit ovat käyttöliittymän uudelleenkäytettäviä osasia, jotka muodostavat oman kokonaisuutensa sisältöineen ja toimintalogiikkoineen. Komponentteja voidaan sisällyttää niin kutsuttuihin vanhempikomponentteihin (engl. parent component). Ohjelmakoodissa Vue-komponentin voi tyypillisesti tunnistaa .vue-päätteisestä tiedostosta, jota kutsutaan myös lyhenteellä SFC (Single-File Component). Komponenttien käyttäminen mahdollistaa käyttöliittymän osasten ajattelemisen omina pieninä kokonaisuuksinaan.

Vanhempikomponentit voivat välittää tietoa lapsikomponentteihin mukautettujen attribuuttien avulla. Näitä attribuutteja kutsutaan nimellä props. Lapsikomponentti sisältää listan prop-attribuuteista, joita se on valmis vastaanottamaan. Myös kommunikointi toiseen suuntaan on mahdollista; vanhempikomponentti voi kuunnella tapahtumia (events), joita lapsikomponentti emitoi tarvittaessa. (Vue.js s.a.d.)

On olemassa erilaisia tapoja käyttää Vue-komponentteja osana projekteja. Niitä voidaan muodostaa rakentamalla komponentin näkymä sekä sen logiikka ja tyylit projektin sisällä (Vainio 2018, 23). Toinen tässä opinnäytetyössä oleellinen komponenttien käytölle on niiden lataus julkisesta sovellusrekisteristä. NPM (Node Package Manager) on reksitereistä suurin ja tarkoitettu JavaScript-pohjaisten pakettien julkaisuun. NPM tarjoaa komentorivityökalun, jolla pystyy kommunikoimaan rekisterin kanssa esimerkiksi, kun tarkoituksena on sisällyttää tietty paketti omaan projektiin. Tällä hetkellä rekisterin tietokannassa on noin kaksi miljoonaa pakettia. Vue sisältyy siellä kymmenen suosituimman kirjaston joukkoon. (NPM Docs s.a.; NPM 2022.) Tarkoituksena on julkaista myös opinnäytetyössä syntyvä päivämääräkomponentti NPM:ssä, jotta sitä voidaan käyttää projekteissa edellä mainitulla tavalla.

5 Päivämääräkomponentin kehitys

Tässä osiossa kerron saavutettavan Vue-päivämääräkomponentin kehityksen vaiheista. Hyödynnän muissa osioissa keräämääni tietoa saavutettavuudesta kehitykseen liittyviä valintoja tehdessäni. Saavutettavuuden kriteerit on määritelty opinnäytetyön liitteistä löytyvässä listassa. Poimin niitä mukaan kehitykseen, kun komponentti on kyseisen kriteerin huomioimiseen otollisessa vaiheessa.

Hyödynnän komponentin kehityksessä alun perin juuri ohjelmistokehitykseen suunniteltua Scrum-viitekehystä, jossa kehitys toteutetaan sprinteiksi kutsutuissa sykleissä. Käytän tätä viitekehystä soveltaen projektissani. Tavallisesti Scrum-tiimiin kuuluvat kehittäjät, tuoteomistaja sekä Scrum Master. Tässä tapauksessa edustan kuitenkin itse kaikkia näitä rooleja. Perustan valintani uusimman, vuonna 2020 julkaistun oppaan kohtaan, jossa todetaan seuraavaa: ”Scrumia käyttämällä voimme löytää ja kehittää omia toimintamalleja, prosesseja ja oivalluksia, jotka sopivat Scrum-viitekehykseen. Niitä ei kuvata Scrum-oppaassa, koska ne ovat tilannesidonnaisia ja hyvin erilaisia Scrumin eri käyttäjillä.” (Schwaber & Sutherland 2020.)

Käyttämällä Scrum-viitekehystä säännölliseen raportointiin pyrin luomaan kehitysprosessille läpinäkyvyyttä. Kehityksen vaiheet näkyvät tässä luvussa sprintteinä, joiden pituus on noin yksi kuukausi. Sprinttien lopullinen lukumäärä määräytyy sen mukaan, missä vaiheessa tulkitsen komponentin täyttävän sille määritellyt saavutettavuuskriteerit ja olevan myös muilta osin valmis. Sprintit sisältävät tilanteen mukaan muun muassa työjonon päivitystä, tavoitteiden määrittelyä, päivämääräkomponentin kehitystä sekä tulosten tarkastelua ja raportointia.

5.1 Päivämääräkentän saavutettavuuteen liittyviä haasteita

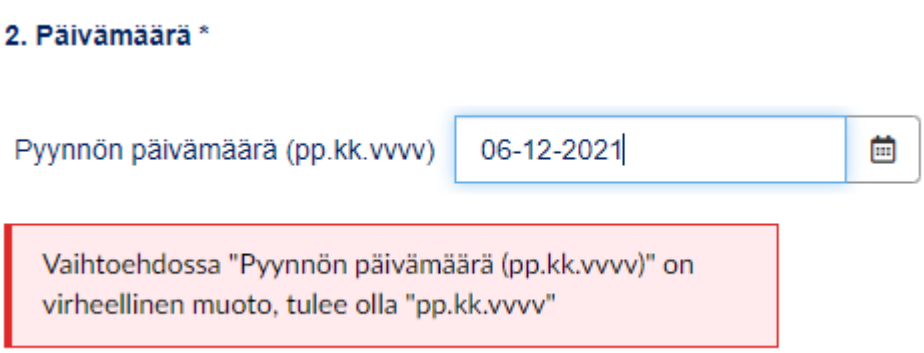
Eficode on suomalainen yritys, joka muun toiminnan ohella on keskittynyt saavutettavuuteen ja esimerkiksi saavutettavuusarviointien tekoon. Yrityksen verkkosivuilla listataan viisi yleistä saavutettavuusongelmaa. Heti toisena saavutettavuushaasteita aiheuttavana ongelmana on mainittu kustomoitujen komponenttien käyttö.

Sivulla mainitaan, että ilman paneutumista asiaan kustomoiduista komponenteista ei todennäköisesti tule saavutettavia esimerkiksi muille kuin hiiren käyttäjille ja ruudunlukijatuki saattaa puuttua. Todennäköisesti näitä ongelmia sisältävinä komponentteina mainitaan erikseen juuri päivämääräkentät. Eficoden sivuilla ratkaisuna suositellaan korvaamaan komponentteja yksinkertaisella HTML-standardikomponentilla, jos se vain on mahdollista (Eficode 2021a; Eficode 2021b.)

Oletukseni mukaan tässä tapauksessa standardikomponenteilla tarkoitetaan komponentteja, jotka pääasiassa koostuvat käyttötarkoituksensa ilmaisevista ja määrittelevistä HTML-elementeistä. Päivämääräkentälle tällaista vaihtoehtoa ei kuitenkaan ole saatavilla. Yksi keino luoda helposti kalenterinäkymän sisältävä päivämääräkenttä HTML:ssä on <input type=”date”></input> -elementti. Elementin attribuutti ”type” ja sen arvo ”date” sekä mahdolliset muut attribuuttimääritykset määrittelevät kentälle päivämääräkentälle ominaisia piirteitä. Tämä vaihtoehto sisältää kuitenkin useita saavutettavuuteen ja muuhun käytettävyyteen liittyviä puutteita. Muun muassa Graham Armfield on selvittänyt kyseisen elementtiratkaisun toimintaa blogitekstissään ”Is input type=”date” ready for use in accessible websites?”. Tekstissä todettuja saavutettavuusongelmia ovat muun muassa validointivirheisiin liittyvät haasteet sekä Safari-selaimen tuen puute. (Armfield 2019.)

Esittelen tässä kappaleessa vielä muutaman konkreettisen esimerkin päivämääräkenttien saavutettavuusongelmista. Esimerkit löytyvät Suomen julkisen hallinnon tai vakuutusalan toimijoiden verkkosivuilta. Ongelmat ilmenevät päivämääräkenttiä koskevista saavutettavuusselosteiden kommenteista. Tyypillisesti päivämääräkenttiä sisältävien verkkolomakkeiden avaaminen vaatii usein tunnistautumisen järjestelmään, mutta valitsin esimerkeiksi lomakkeet, joille avaamiseksi kirjautumista ei vaadita.

Ensimmäisenä esimerkkinä on Tullin passituksen varamenettelyä koskeva verkkolomake (Tulli s.a.a.) Tullin lomaketta koskevan saavutettavuusselosteen huomiot ongelmista päivämääräkentän osalta kohdistuvat siihen, että kentän täyttäminen ei onnistu pelkällä näppäimistöllä sekä siihen, että päivämääräkentän voi syöttää vain yhdessä muodossa. Kentän voi avata näppäimistöllä, mutta päivämäärän valinta pelkkää näppäimistöä käyttäen ei onnistu. Päivämäärän syöttäminen rikkoo kahta WCAG-kriteeriä, kun se hyväksyy ainoastaan ”pp.kk.vvvv”-muodossa syötetyn päivämäärän eikä ilmoita käyttäjälle riittävän selkeästi virheen tapahtumisesta. (Tulli s.a.b.)



Kuva 2. Kuvakaappaus päivämääräkentästä Tullin verkkolomakkeella. (Tulli s.a.a.) Kenttään on syötetty päivämäärä muodossa, joka ei ole kentän vieressä näkyvän validointivirheestä kertovan tekstin mukaan hyväksytty.

Toinen esimerkki koskee Finavian hallinnoimaa Helsinki-Vantaan lentoaseman pysäköinninvarausjärjestelmää (Finavia 2021a). Saavutettavuusselosteen mukaan myös tämän päivämäärävalitsimen ongelmana on vaikeus käyttää sitä näppäimistöllä tai ruudunlukijalla. Lisäksi kenttiin ei ole mahdollista syöttää itse päivämäärätekstiä. (Finavia 2021a.) Näin ollen päivämäärän voi syöttää vain ”pp.kk.vvvv”-muodossa, joka kentälle on ohjelmallisesti määritelty.

Kuva 3. Kuvakaappaus pysäköinnin varausjärjestelmän päivämäärä- ja aikavalitsimista Finavian verkkosivuilta. (Finavia 2021a.)

Kotitalousasiakkaiden Oma Fennian sivuilla oleva päivämääräkenttä on kirjautumisen takana, joten siitä ei ole kuvaa. Sivuja koskevan saavutettavuusselosteen mukaan ongelmia on aiemman esimerkin kaltaisesti päivämäärätiedon syöttämisessä näppäimistöllä tai apuvälineillä. Tämä johtuu siitä, että kenttä on tehty vain luku -tyyppiseksi. (Fennia 2020.) Päivämääräkentän yhteydessä esiintyy usein kalenterin kuvaa esittävä painike, josta painamalla aukeaa varsinainen kalenterinäkymä. Tässä tapauksessa kalenteripainikkeelta puuttuu tekstivastine eikä sen yhteydessä ilmoiteta, että se toimii painikkeen kaltaisesti (Fennia 2020).

5.2 Menetelmä ja tavoitteet

Pääasiallisena menetelmänä komponentin saavutettavuuden kehityksessä käytetään komponentin osien peilaamista W3C:n WCAG 2.1 -onnistumiskriteereihin. Kriteerien merkitystä kartoitetaan tarvittaessa täydentävästä dokumentaatiosta. Saavutettua saavutettavuuden tasoa mitataan subjektiivisen onnistumiskriteerien saavuttamisen arvioinnin lisäksi käyttäjätestauksella.

Tavoitteena on kehittää komponentti, joka vastaa mahdollisimman hyvin saavutettavuuden WCAG 2.1 AA -tason onnistumiskriteereihin. Lisäksi pyrin huomioimaan käytettävyyden yleisluotoisesti. En siis syvenny käytettävyyden yksityiskohtiin, vaan pyrin varmistamaan, että päivämääräkenttä on looginen ja miellyttävä käyttää myös saavutettavuusvaatimusten ulkopuolella.

Käytettävyys kuitenkin perustuu vain omaan arviooni ja on siten subjektiivisen kokemuksen tulos. Käytettävyyteen voi mahdollisesti vaikuttaa jatkokehityksessä enemmän. Pyrin

lisäksi siihen, että saavutettavuuden toteuttaminen ei heikentäisi päivämääräkentän muita ominaisuuksia.

Valitsen päivämääräkomponentin muodoksi yleisesti käytössä olevan tekstikentän ja visuaalisen, avattavan kalenterinäkymän yhdistelmän. Helpointa voisi olla jättää kalenterinäkymä kokonaan pois ja tehdä saavutettava päivämääräkenttä pelkästään hyödyntäen tekstikentän ominaisuuksia. Visuaalinen kalenteri kuitenkin on niin yleisesti käytössä, että tulkitsen sen olevan vakiintunut käytäntö ja siksi liitän sen mukaan komponenttiin.

Tästä huolimatta tavoitteenani on tehdä komponentti, jota käyttäjä voi halutessaan käyttää myös pelkästään syöttämällä päivämäärän tekstikenttään. Tämä voi helpottaa osaa käyttäjistä. Halutessaan käyttäjä voi kuitenkin myös avata tekstikentän yhteyteen sijoitetun visuaalisen kalenterin ja valita päivämäärän sen kautta.

Vaikka toivoisin, että komponentti olisi poikkeuksetta sen jokaiselle potentiaaliselle käyttäjälle saavutettava, absoluuttisen saavutettavuuden saavuttaminen tai sen arviointi ei tämän opinnäytetyön puitteissa ole mahdollista. Siksi yritän tehdä komponentista mahdollisimman saavutettavan mahdollisimman monelle. Tällä kertaa se tarkoittaa WCAG 2.1:n sisältämien kriteerien mahdollisimman hyvää toteutumista.

5.3 Käytettävät teknologiat

Käytän toiminnallisessa osuudessa GitHubia versionhallintaan ja hyödynnän sitä myös osana laajempaa projektinhallintaa. Merkittävät projektin kehitystarpeet sekä mahdolliset korjaukset tallentuvat GitHubiin, kun teen niistä virtuaalisia kortteja projektini sisältämään Kanban-tauluun. Projektissa käytettävät ohjelmointiin liittyvät tekstieditorit, hallintasovellukset, kirjastot ynnä muut tarkentuvat työn edetessä.

GitHub mahdollistaa uusien ohjelmakoodin kehityshaarojen (engl. branches) tekemisen siten, että muutokset sisältävä haara on suoraan linkitettyinä muutoksia kuvaavaan korttiin. Hyödynnän tätä ominaisuutta ja teen uuden kehityshaaran aina lisäämästäni kehitystehtävästä, joka voi kuvata muutostarvetta tai bugin korjaustarvetta. Lisäksi ylläpidän työjonoa tulevista kehitystehtävistä fyysisillä kartonkikorteilla, joita kirjaan GitHub-projektiin sitä mukaa kun ne tulevat ajankohtaisiksi. Päivitän kortteja myös, kun tehtävät tarkentuvat työn edetessä.

Minulla on projektia aloittaessani käytössäni kehitykseen pääasiassa käytettävällä tietokoneellani Windows-käyttöjärjestelmä ja pääasiallisena selaimena päivämääräkentän

kehityksessä ja testauksessa käytän Chrome-selainta. Lisätietoa siitä, kuinka toteutan päivämääräkentän arviointia ja testausta, löytyy opinnäytetyön liitteestä 2.

Pyrin siihen, että komponentti olisi sitä mahdollisesti lähdekoodissaan hyödyntävän ohjelmistokehittäjän näkökulmasta mahdollisimman helppo ottaa käyttöön. Vue.js sopii tähän tarkoitukseen progressiivisena JavaScript-kirjastona hyvin.

5.4 Sprintti 1

Ensimmäisen sprintin tavoitteena oli pystyttää Vue.js-komponentin kehitysprojekti. Käytännössä tämä tarkoitti oleellisimpien teknologioiden valintaa ja installointia kehitykseen käytettävälle tietokoneelle sekä version- ja projektinhallintaan käytettävien työkalujen käyttöönottoa. Työkalujen ja ohjelmistojen valinnan ja päivitysten jälkeen ensimmäiseen sprintin lopussa oli tavoitteena tehdä ensimmäinen, yksinkertainen versio päivämääräkentästä.

Teknologia kehittyy ja uudet versiot työkaluista ja teknologioista usein sisältävät parannuksia ja päivityksiä edellisiin versioihin verrattuna. Silti kannattaa miettiä kokonaisvaltaisesti, onko järkevintä käyttää omassa projektissaan juuri uusimpia versioita vai voisiko jossakin tapauksessa vanhempi versio kuitenkin olla kannattavampi vaihtoehto. Itse päädyin Vue 2 -versioon, vaikka Vue 3 on tuorein Vue-kirjaston versio. Pohdin versioiden 2 ja 3 välillä, kumman käyttöön olisi paremmat perustelut saavutettavan komponentin rakentamisessa. Harkinnan jälkeen päädyin käyttämään Vuen versiota 2.

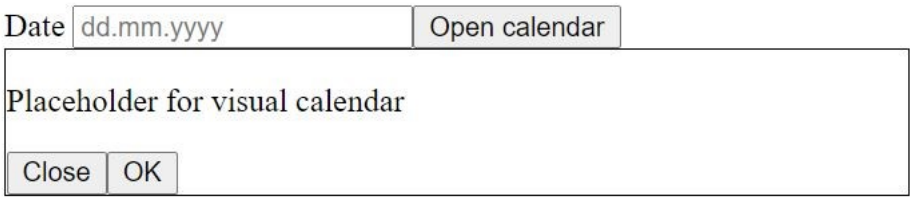
Vaikka versio 3 uudistustensa vuoksi veti puoleensa, koin, että mahdollisimman saavutettavan komponentin tekemiseen oli varmempi perusta vanhemmassa versiossa. Päätökseen vaikutti esimerkiksi se, että käyttäjät eivät välttämättä pysy selainten päivityksissä mukana, vaikka teknologian kehitys monelta osin sitä saattaisikin jo puoltaa. Siksi halusin huomioida myös käyttäjien mahdollisesti käyttämien teknologioiden ja eri versioiden yhteensopivuuden. Tässä tapauksessa ratkaisevaa oli, että Vue 3 ei tue Internet Explorer 11 (IE11) -selainta. Vaikka kyseessä on selain, jonka tuki on jo loppunut, jotkut käyttäjistä saattavat edelleen selata verkkosisältöä sen kautta. Koneellani oli valmiiksi Vuen versio oli 2.6.14 edellisen käytön jäljiltä. Lisäksi päivitin Node.js:n uusimpaan versioonsa.

Asensin koneelleni vue-sfc-rollup-kirjaston. Kirjasto sisältää valmiiksi asetuksia, joiden avulla komponentti on helppoa siirtää NPM-sovellusrekisteriin, mikäli se julkaistaan. Lisäksi asensin SourceTree-nimisen graafisen käyttöliittymän versionhallintaan liittyvien komentojen ajamiseen. Tällä pyrin minimoimaan riskejä sille, että tekisin komentorivityökalun kautta projektin sekoittavia git-komentoja, joiden selvittely voisi syödä aikaa muulta kehitystyöltä.

Ennen projektin luomista tarkistin, olisiko jokin suunnittelemistani komponentin nimistä vapaana NPM-rekisterissä. Nimen voi tarkistaa esimerkiksi ajamalla komentorivityökalulla ensin komennon *npm install --global npm-name-cli*, jolla asennetaan nimentarkastuspaketti omalle koneelle globaalisti. Tämän jälkeen ajetaan komento *npm-name <name>*, jossa ”<name>” on korvattu tarkistettavalla nimellä. Ensimmäinen ajatukseni oli "accessible-datepicker", mutta se ei ollut saatavilla. Pyrin nimen osalta siihen, että se kuvaa mahdollisimman hyvin komponentin tarkoitusta. Loppujen lopuksi päädyin vue-accessible-date-field-nimeen.

Seuraavaksi loin projektin omalle koneelleni vue-sfc-rollup-kirjaston avulla ja tein yksityisen projektin GitHubiin. Tarkoituksena on muuttaa projekti julkiseksi siinä vaiheessa, kun komponentti on riittävän valmis muiden käytettäväksi. Yhdistin projektiin sen luomisvaiheessa Kanban-taulun, lisäsin avoimen lähdekoodin projekteissa suositun MIT-lisenssin ja README.md-tiedoston.

Ensimmäisen sprintin lopussa testasin vielä, että kaikki on valmista kehitystä varten ja muutokset tulevat näkyviin selaimeen, kun koodin ajaa lokaalissa ympäristössä. Lähdekoodin muokkaukseen päätin käyttää Visual Studio Codea (VS Code), joka on avoimen lähdekoodin tekstieditori. Tein hyvin yksinkertaisten version päivämääräkomponentille. Siinä näkyivät <input>-elementti, joka toimi tekstikenttänä sekä painike, josta voi avata modaalin, jonka sisällä kalenterinäkymä voidaan rakentaa.



Kuva 4. Ensimmäinen versio päivämääräkentästä ja aukinaisesta kalenterimodaalista kuvattuna Chrome-selaimessa.

5.5 Sprintti 2

Toisen sprintin tavoitteenani oli luoda visuaalisen kalenterinäkymän sisältö. Tiesin sen vievän aikaa, koska kalenterinnäkymässä on hyvin monta toisiinsa vaikuttavaa osaa. Lisäksi tavoitteenani oli tehdä alustava versio kalenteri-ikonista. Kokonaistavoitteena oli siis lisätä osia, joiden avulla komponentista saisi valmiiksi seuraavan ehjän kokonaisuuden ja jatkokehitystarpeita olisi helpompaa hahmottaa. Tässä sprintissä saavutettavuus näkyi HTML-elementtien valinnoissa. Pohdin myös yksikkötestien lisäämistä komponenttiin, mutta

toistaiseksi en niitä lisännyt. Mietin, että niiden ylläpito suhteessa hyötyihin saattaisi viedä tässä vaiheessa liian paljon aikaa. Päätin harkita testien lisäystä myöhemmin uudelleen.

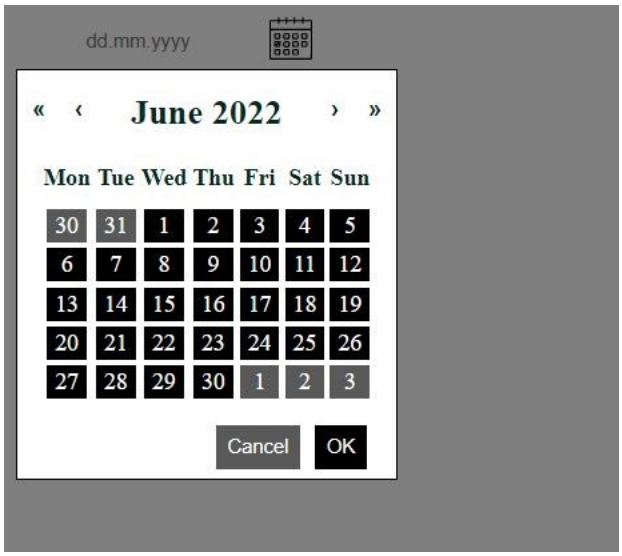
Visuaalisen kalenterin muutokset pitivät sisällään ylälaidassa h2-tason otsikon, joka esittää kalenterissa kulloinkin näkyvän kuukauden nimen sekä kyseisen vuoden numerona. Lisäksi tämän otsikon molemmille puolille tein painikkeet, joista vasemmanpuoleisista voi vaihtaa kuukautta tai vuotta yhden aiempaan ja oikeanpuoleisista vastaavasti eteenpäin. Näiden alapuolelle tein kalenterinäkymään table- eli taulukkoelementin, joka pitää sisällään kuukauden päivät järjestettyinä seitsemän päivän riveihin eli viikkoihin.

W3C:n sivuilla on esimerkki puhtaalla JavaScriptillä toteutetusta saavutettavasta päivämääräkentästä, jonka yhteydessä on avautuva kalenterinäkymä. (W3C s.a.). Pidän sitä silmällä kehityksen edistyessä ja hankalissa valintatilanteissa käytän sitä päätöksenteon tukena. Koska kyseessä on muun muassa saavutettavuuteen liittyvien asioiden asiantuntijaorganisaation tuottama visuaalisella kalenterilla varustettu päivämääräkenttä, koin sen olevan luotettava lähde saavutettavuuteen liittyvissä valinnoissa. Koska oma komponenttini rakennetaan Vuella, tulee lähdekoodin rakenteesta erilainen.

Toisen sprintin pituus venähti noin puoleentoista kuukauteen, kun päädyin kuitenkin vaihtamaan komponentin käyttämään Vue 3 -versiota. Lisäksi tämän muutoksen myötä päädyin myös valitsemaan komponentin kieleksi TypeScriptin JavaScriptin sijasta. TypeScript pohjautuu JavaScriptiin, mutta tarjoaa sen lisäksi erityisen tyypitysjärjestelmän (TypeScript 2022). Minulla on kokemusta TypeScriptin käytöstä ja koin sen tuovan tehokkuutta koodin rakentamiseen. Tässä vaiheessa näiden muutosten tekeminen oli vielä suhteellisen vaivatonta.

Vuen version kohdalla uusimman version tuomat hyödyt menivät vanhojen selainten tuen tuoman hyödyn edelle. Internet Explorer 11:en huomioiminen olisi saattanut tuoda merkittäviä lisähaasteita kehitykselle. Lisäksi kyseinen selain on virallisesti siirtymässä ”eläkkeelle” kesäkuussa 2022 (Van Aelstyn 2021) eli hyvin pian tämän opinnäytetyön valmistumisen jälkeen.

Sprintin lopputuloksena syntyi modaalin kalenterisisältö, joka mahdollistaa kuukausien ja vuosien selaamisen. Lisäksi tein kalenteri-ikonin Paint.net-ohjelmalla. Komponentin modaalin voi sulkea painamalla mitä tahansa kohtaa sen ulkopuolella. Muokkasin lisäksi jonkin verran CSS-tyylejä. Pyrin kuitenkin pitämään tyylimuutokset minimissä tässä vaiheessa.



Kuva 5. Selaimesta otetussa kuvakaappauksessa näkyy taustalla päivämääräkenttä sekä aukinainen kalenterinäkymä. Muu sivu kalenterinäkymän taustalla on sumennettu, kun näkymä on auki.

5.6 Sprintti 3

Kolmannen sprintin tavoitteet koskivat sekä uusien ominaisuuksien luontia että vanhojen ratkaisujen optimointia. Esimerkiksi kalenteritaulukon viikkoja esittävien rivien muodostumiseen vaikuttavien funktioiden toimintaa muokattiin sekä mahdollistettiin päivämäärän valinta kalenterista hiirellä. Tässä vaiheessa näppäimistötukea ei siis vielä lisätty, vaan tarkoituksena oli mahdollistaa tekstikentän ja kalenterinäkymän välinen yhteys.

Lisäksi sprintin tavoitteena oli lisätä päivämääräkomponentin valmiuksia kommunikoida mahdollisten vanhempikomponenttien kanssa. Tämä sisälsi mahdollisuuden ottaa vastaan vanhempikomponentilta syötettävä oletuspäivämäärä props-listan attribuuttina ja käyttää sitä valittuna päivämääränä ennen käyttäjän tekemiä valintoja. Komponentilta vanhempikomponentin suuntaan tapahtuvan kommunikaation kehityksen tavoitteena oli emitoida valittu päivämäärä aina kun sen arvo muuttuu. Viimeisenä tavoitteena oli viedä komponentin ensimmäinen, 0.0.1-versio NPM-rekisteriin ja testata sen installointi yksinkertaisen testiprojektin kautta.

Tavoitteiden asettelusta ei vielä kolmannessa sprintissä varsinaisesti näkynyt saavutettavuuskriteerien huomiointi. Olen kuitenkin pyrkinyt tekemään kaikki kehitystehtävät saavutettavuuden näkökulma huomioiden. Olen siis tarkistanut saavutettavuuskriteerejä kehityksen aikana ja pyrkinyt tekemään valintoja tavalla, joka tukee saavutettavuuden toteutumista joko sillä hetkellä tai tulevia kehitysvaiheita silmällä pitäen. Tässä on auttanut liitteistä löytyvän kriteerilistan ja W3C-dokumenttien lisäksi myös sekä töissä kertynyt

saavutettavuusosaaminen että teoriaosuuden kirjoittamisessa karttunut ymmärrys esimerkiksi HTML-rakenteen vaikutuksista saavutettavuuteen.

Pääsin sprintissä kutakuinkin tavoitteisiin. Päivämäärän valinnassa päivää kuvastavaa kalenteritaulukon päivää klikkaamalla päivä kyllä tulee näkyville tekstikenttään, mutta sen muoto oli vielä väärä. Sprintin varsinaisten kehitystehtävien ohessa tein myös joitakin kehityksen yhteydessä tarpeellisiksi havaitsemiani muutoksia kuten valmiuden tarkistaa päivämäärän merkkijonon muoto säännöllisen lausekkeen (engl. regular expression) eli regexin avulla. Lisätty regex sisältää karkausvuoden huomioimisen päivämäärissä.

Aloitin myös lokalisoinnin määrittelyä eli pyrin lisäämään mahdollisuuden käyttää komponenttia eri kielillä. Päätin tehdä komponentille oletusmäärittelyt suomen-, ruotsin- ja englannin kielillä. Lokalisoinnissa käytettävien käännösilmaisujen inspiraationa käytin Duet Date Picker -nimisen avoimen lähdekoodin komponentin lokalisaatiota (Duet Date Picker 2021) sekä jo aiemmin mainittua W3C:n mallia.

Lisäksi testasin komponentin toimintoja Firefox- ja Edge-selaimilla sekä saavutettavuutta Chromen Wave-työkalulla. Työkalu havaitsi nimilapun (label) puuttumisen tekstikentän yhteydestä. Labelin lisääminen jää kuitenkin vanhempikomponentin vastuulle, koska sen sisältö ja tyyli ovat niin riippuvaisia käyttötarkoituksestaan. Lisään kuitenkin komponentin käyttöohjeisiin ohjeistuksen, kuinka labelin-elementin for-attribuutin saa vastaamaan tekstikentän id:tä. Tämä auttaa saavutettavaa teknologiaa yhdistämään labelin oikeaan kenttään.

Työskentely alkoi tässä vaiheessa muistuttaa enenevissä määrin vesiputousmallia. Päätin tehdä seuraavan kehitysjakson tehtävät enemmän spontaanisti tarpeen mukaan, koska tulevien tehtävien määrittely ei voi enää olla yhtä suurpiirteistä ja minun oli vaikeaa ennakoida niitä tarkasti tavoitteenasettelussa. Päätin kuitenkin nimetä sen Scrum-tavan mukaan sprintiksi yhtenevän raportointitavan säilyttämiseksi.

Päätin käyttää komponentissa lähdekoodin puolella päivämäärien käsittelyyn pääasiassa kansainvälisen ISO 8601 -standardin merkintätapaa eli vvvv-kk-pp. Se on muoto, jolla esimerkiksi vanhempikomponentin tulee syöttää oletuspäivämäärän arvo päivämääräkomponentin käyttöön. Käyttäjälle päivämäärä kuitenkin näytetään käyttäjäystävällisemmässä muodossa vakiintuneiden tapojen mukaan. Esimerkiksi Suomessa vakiintunut muoto on pp.kk.vvvv.

NPM-rekisteriin vienti onnistui ja installoinnin jälkeen päivämääräkenttä tuli näkyville tekemäni testiprojektin testilomakkeelle muiden lomakkeen kenttien oheen. Päivitin tyypillisesti

komponentin sisällöstä ja toiminnasta kertovaan README.md-tiedostoon tiedon siitä, että komponentin ja sen saavutettavuuden kehitys on vielä kesken ja komponenttia suositellaan käyttämään vasta, kun versio 1.0.0 on julkaistu. Teksti näkyy myös pakettia kuvaavalla sivulla NPM:ssä.

5.7 Sprintti 4

Neljäs sprintti kattoi komponentin ominaisuuksien monipuolista kehitystä sekä saavutettavuuden testausta. Kävin läpi saavutettavuuskriteerejä aiempaa tarkemmin. Tein lisäyksiä ja muokkauksia komponenttiin. Erityisesti näppäimistötuen monipuolisessa toteutuksessa verrattomana apuna toimi W3C:n päivämääräkentän ja kalenterinäkymän yhdistävä esimerkki. Päivitin komponenttia aika ajoin NPM:ään, kun kehitys oli edistynyt pisteeseen, jossa julkaisu oli mielekästä.

```
<tbody>
  <tr
    v-for="(week, index) in daysVisibleCurrentMonth"
    :key="index"
    class="datepicker-table-row">
    <td
      v-for="(dayItem, index) in week"
      :key="index"
      @click="handleDatePress($event, dayItem, true)"
      tabindex="-1"
      :class="['datepicker-day',
      { 'selected-date': createDate(dayItem) === selectedISODate },
      { 'disabled-day': checkDisabledDay(dayItem) }]"
      :data-date="createDate(dayItem)"
      role="gridcell"
      :aria-selected="checkSelected(dayItem)"
      @keydown.esc="closeDatePickerModal()"
      @keydown.space="handleDatePress($event, dayItem, false)"
      @keydown.enter="handleDatePress($event, dayItem, true)"
      @keydown.up="goToPreviousWeek(dayItem, $event)"
      @keydown.down="goToNextWeek(dayItem, $event)"
      @keydown.right="goToNextDay(dayItem, $event)"
      @keydown.left="goToPreviousDay(dayItem, $event)"
      @keydown.home="goToFirstDayOfWeek(dayItem, $event)"
      @keydown.end="goToLastDayOfWeek(dayItem, $event)"
      @keydown.page-down="handlePageDown($event, dayItem)"
      @keydown.page-up="handlePageUp($event, dayItem)">
      {{ dayItem.day }}
    </td>
  </tr>
</tbody>
```

Kuva 6. Näppäimistötuen lisäämisessä huomioitiin kalenterinäkymän päivää kuvaavien taulukon soluelementtien kohdalla useita eri näppäinvaihtoehtoja, joista osa esimerkiksi vaihtoi kalenterinäkymän kuukautta. Kuvassa näkyy myös, että soluille on lisätty WAI-ARIA-rooliksi ”gridcell”.

Komponentissa ei ole käytetty ulkoisia kirjastoja muodostamaan osia komponentista tai esimerkiksi helpottamaan tyylien määrittelyä vaan lähes kaikki on tehty komponenttitiedoston sisällä alusta lähtien. Tiedoston sisältö oli kasvanut laajaksi huomioiden, että kyse on yhden sivun komponentista. Yksinkertaistaakseni sisältöä siirsin osan määrittelemistäni TypeScript-rajapinnoista (engl. interface) komponentin ulkopuolisiin tiedostoihin. Kyseisiä rajapintoja käytetään komponentissa määrittelemään eri tarkoituksiin käytettävien muuttujien tyypityksiä.

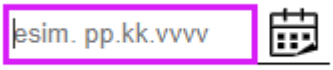
Jatkoin sprintin aikana lokalisaation kehitystä. Saavutettavuuskriteerin 3.1.2 mukaan sisällön osien luonnollinen kieli tulee pääsääntöisesti voida selvittää ohjelmallisesti jokaisen sisällössä esiintyvän tekstikatkelman tai ilmaisun osalta (Kirkpatrick, A., O'Connor, J., Campbell, A. & Cooper, M. 2018; Kehitysvammaliitto ry 2019). Kun komponentin sisältämä sivu ladataan ensimmäisen kerran, tarkistetaan sivun käyttämä kieli HTML:n lang-attribuutista. Tässä tapauksessa se tarkoittaisi komponenttia käyttävään sovellukseen määriteltyä kieltä, jota myös komponentti käyttäisi. Tätä ei kuitenkaan välttämättä ole määritelty eikä luonnollisen kielen varmistamisen mahdollisuutta täten ole taattu ohjelmallisesti. Saattaa olla, että kriteerin toteutumisen varmistamiseksi tulisi lisätä komponentin tekstikatkelmia tai ilmaisua sisältäville osille kieliattribuutit, jotka muuttuisivat samojen ehtojen mukaisesti kuin käytetyt kielitermit. Lisäsin asian selvittämisen jatkokehitysideoiden listalle.

Jos vanhempikomponentista on kerrottu lapsikomponentille käytettävä kieli ja se on joko suomi, ruotsi tai englanti, käytetään komponentissa tätä kieltä. Jos kumpaakaan tietoa edellä mainituista ei saada selvitettyä, käytetään oletuskieltä. Komponentissa voidaan edellä mainituilla kielillä käyttää komponenttiin asetettuja oletustermejä eri osissa kuten kuukausien nimissä ja käyttäjälle kerrotuissa komponentin tilaan liittyvissä viesteissä. Komponentille voidaan kuitenkin myös antaa täysin kustomoidut kielitermit attribuutteina.

Neljännessä sprintissä lisäsin myös käyttäjälle välitettäviä, komponentin tilasta kertovia viestejä. Näihin sisältyi virheviesti, joka ilmaistaan käyttäjälle, jos havaitaan että käyttäjän syöte tekstikenttään ei ole mennyt päivämäärän muodon tarkistuksista läpi. Määrittelin tekstikentän merkitystä ruudunlukijalle kuvailevan elementin viestiksi päivämäärän mahdollisia kirjoitusasuja esittelevän tekstin, mikäli syötteessä ei ole havaittu virheitä tekstikentän ollessa kohdistettuna. Sen sijaan, jos virhe on havaittu, muuttuu kuvailuteksti virheestä kertovaksi, myös visuaalisesti havaittavaksi virheviestiksi.

Kävin läpi komponentissa käyttämäni värit ja niiden kontrastit vieressä olevien värien kanssa (saavutettavuuskriteerit 1.4.3 ja 1.4.11). Hyödynsin tähän WebAIM-organisaation ylläpitämää kontrastin tarkistukseen tarkoitettua verkkotyökalua. (WebAIM s.a.) Muutin

tarvittaessa värejä tummemmiksi tai vaaleammiksi, jotta kontrastit ovat riittäviä. Komponenttiin valitut värit ovat muokattavissa myös komponenttia hyödyntävien sovellusten kautta esimerkiksi yliajamalla määritellyt CSS-säännöt tarkemmilla määrityksillä. Tämä on hyvin oletettava skenaario, koska monille verkkosivuille on määritelty tietyt väriteemat, joita noudatetaan.



Kuva 7. Komponentin päivämäärän syöttökenttä värien kontrastitarkistuksen jälkeen. Kuvassa näkyy tekstikenttä kohdistettuna (focus), jolloin kentän reunat korostuvat. Lisäksi kuvassa matkan varrella hieman yksinkertaistunut SVG-muotoinen kalenteri-ikoni.

Kuva 8. Päivämääräkomponentin kalenterinäkymä värien kontrastitarkistuksen jälkeen. Kuvassa näkyvät päivän tyylit valittuna (turkoosi katkoviiva reunoina), kohdistettuna, mutta ei valittuna (keltainen tausta ja turkoosi reuna) sekä tilassa, jossa hiiren kohdistin on viety tietyn päivän päälle (hover-tila) (harmaa tausta ja pinkki reuna).

Päätös tehdä sprintissä eteen tulevia kehitystehtäviä vesiputoustaktiikalla osoittautui tehokkuuden kannalta toimivaksi ratkaisuksi. Sain edistettyä komponenttia aina siltä osin kuin se milläkin hetkellä tuntui perustelluimmalta, jotta lopputuloksena syntyisi mahdollisimman saavutettava ja käyttökelpoinen komponentti. Onnistumiskriteerien läpikäynti oli olennainen osa työn osien priorisointia.

Opinnäytetyölle asettamani aikaraja alkoi tulla vastaan enkä valitettavasti ehtinyt etsiä käyttäjätestaajaksi henkilöä, joka todennäköisesti hyötyisi komponentin saavutettavuudesta keskimääräistä enemmän. Tämän sijasta käyttäjä/asiantuntijatestauksen lupautui tekemään saavutettavuuteen perehtynyt ohjelmistokehittäjä omalta työpaikaltani. Kerron tarkemmin lopputestauksesta sille varatussa luvussa. Sprintin lopussa julkaisin vielä päivämääräkomponentin sisältävän testilomakkeen Heroku-pilvipalveluun, jonka kautta komponentti oli helppoa jakaa testattavaksi.

5.8 Lopputestaus

Lopputestaus koostuu ulkopuolisen käyttäjän arviosta sekä arviointisuunnitelman mukaisista toimenpiteistä, joita ei vielä kehityksen aikana ollut toteutettu. Näihin lukeutuvat muun muassa jäljelle jääneet onnistumiskriteerit sekä Safari-selaimella testaaminen.

Tunnin kestäneen käyttäjätestauksen aikana sain monipuolisesti palautetta. Seuraavassa on listattuna seikat, joita testaaja suositteli korjaamaan tai tarkistamaan:

- sisällytetään ikonipainikkeeseen kattava ohjeistus ruudunlukijalle siitä, kuinka näppäimistöllä liikutaan kalenterinäkymän sisällä. Tähän ei ole olemassa standardeja. Ohjeistus alkaa sen jälkeen, kun on ensin kerrottu painikkeen tarkoitus ja mahdollisesti valittu päiväys
- modaalin aria-label-attribuutin toimivuuden tarkistus
- kuukausien ja vuosien vaihtamiseen tarkoitetuille nuolinäppäimille aria-labelit
- emitoinnin muuttaminen siten, että jos oletuspäivämäärää ei ole syötetty komponentille, kompoentti ei silti emitoi kuluvaakaan päivää, joka kuitenkin on kohdistettuna kalenterinäkymässä
- tarkistus, olisiko Page Up ja Page Down -näppäinten toiminnallisuudet syytä vaihtaa päittäin
- CSS-säännöille oma tunniste

Avustavista teknologioista testauksessa käytettiin ruudunlukijaa sekä pelkkää näppäimistöselausta. Saavutettavuuden arviointiin käytettävistä työkaluista taas käytössä oli Wave. Kokonaisarvio komponentista oli hyvin positiivinen. Testaaja sanoi, että he ottaisivat tiimissään komponentin mahdollisesti käyttöön, kun siihen on tehty vielä muutamia parannuksia. Päätin toteuttaa edellä mainitut muutokset vielä opinnäytetyön aikana lukuun ottamatta viimeistä CSS-sääntöjen muutoksia koskevaa palautetta. Sen päätin siirtää jatkokehitysehdotukseksi. Esimerkiksi ikonipainikkeen ohjeistus näppäimistökomennoille toteutui lisäämällä WAI-ARIAan kuuluva ”aria-description”-attribuutti elementille. Lisäsin myös ohjeistuksen käännökset.

```
export const localizationDefaultDataSv: Localization = {
  locale: 'sv-SE',
  placeholderText: 'T.ex. dd.mm.åååå',
  dateFormatString: 'datumformat: ',
  wordOrTranslated: 'eller',
  dateFormatOptions: ['dd.mm.åååå', 'dd-mm-åååå', 'dd/mm/åååå'],
  generalDateFieldError: 'Datumformatet du angav är ogiltigt',
  keyboardNavInstructions: 'Du kan bläddra igenom kalenderinnehållet med följande tangenter:' +
  'TAB-tangent: Flytta fokus mellan piltangenter, måldatum samt OK och Avbryt på skärmen, ' +
  'vänsterpil: gå till föregående dag, ' +
  'högerpil: Gå till nästa dag, ' +
  'uppåtpil: gå till samma veckodag förra veckan, ' +
  'nedåtpil: gå till samma veckodag nästa vecka, ' +
  'HOME: gå till den första dagen i samma vecka, ' +
  'END: gå till den sista dagen i samma vecka, ' +
  'PAGE DOWN: gå till föregående månad ,' +
  'SHIFT + PAGE DOWN: gå till föregående år, ' +
  'PAGE UP: gå till nästa månad, ' +
  'SHIFT + PAGE UP: gå till nästa år, ' +
  'ENTER: välj den inriktade dagen och stäng kalendern, ' +
  'SPACE: välj en inriktad dag, kalendern förblir öppen, ' +
  'ESC: stäng kalendern',
  buttonLabelChoose: 'Välj ett datum',
  buttonLabelChange: 'Ändra det valda datumet, ',
  buttonLabelPreviousMonth: 'Gå till föregående månad',
  buttonLabelPreviousYear: 'Gå till föregående år',
  buttonLabelNextMonth: 'Gå till nästa månad',
  buttonLabelNextYear: 'Gå till nästa år',
  dayNames: ['måndag', 'tisdag', 'onsdag', 'torsdag', 'fredag', 'lördag', 'söndag'],
  dayNamesShort: ['må', 'ti', 'on', 'to', 'fr', 'lö', 'sö'],
  monthNames: ['januari', 'februari', 'mars', 'april', 'maj', 'juni', 'juli', 'augusti', 'septembe
  monthNamesForMessage: ['januari', 'februari', 'mars', 'april', 'maj', 'juni', 'juli', 'augusti',
  selectFocusedButtonLabel: 'OK',
  cancelButtonLabel: 'Avbryt'
}
```

Kuva 9. Lopputestauksessa suurin muutos kohdistui loppukäyttäjälle välitettävään näppäimistötuen ohjeistukseen, joka lisättiin käännöksiin (kuvassa keyboardNavInstructions). Kuvassa ruotsinkieliset käännökset, joita hyödynnetään komponentin eri osissa, kun käyttäjälle halutaan välittää informaatiota.

Käyttäjätestauksen lisäksi tarkastelin komponenttia itsenäisesti vielä loppujen onnistumiskriteerien näkökulmasta. Osa kriteereistä oli täyttynyt jo kehityksen eri vaiheissa joko automaattisesti tehtyjen valintojen kautta tai niitä varten oli tehty varta vasten muutoksia. Läpi käymättömät kriteerit joko vaativat pieniä muutoksia, ne toteutuivat osittain tai niiden arviointiin liittyi epäselvyyttä täydentävän dokumentaation tutkimisesta huolimatta.

Testasin komponentin perustoiminnallisuuksia lopuksi vielä Safari-selaimella muutamalla iPhonen eri mallilla. Käytin tässä apuna BrowserStack-ohjelmaa. Muut arviointisuunnitelmassa mainitut selaimet oli testattu kehityksen aikana. Safarilla testatessa ei ilmennyt poikkeavuuksia komponentin toiminnassa.

6 Lopputulos

Opinnäytetyössä kehitettiin Vue-päivämääräkomponentti, jonka saavutettavuus monilta osin täyttää WCAG 2.1 -onnistumiskriteerit. Komponentti on julkaistu NPM-rekisteriin nimellä vue-accessible-date-field. Kuka tahansa voi ladata sen rekisteristä komentorivikomennolla osaksi omaa ohjelmistoprojektiaan.

Lopputuloksessa 37:stä onnistumiskriteeristä 29 toteutui, neljä toteutui osittain, yksi ei toteutunut ja kolmen toteutumisesta ei ole tietoa (liite 2). Ne, joiden toteutuminen on epäselvää ovat pääosin kriteerejä, joiden loppuunsaattaminen jää komponenttia hyödyntävän projektin ylläpitäjän vastuulle. Niiden toteutuminen on kuitenkin olennainen osa komponentin saavutettavuutta.

Selkeästi toteutumatta jäänyt kriteeri 3.3.3 koskee virheilmoitusten korjausehdotuksia. Käyttäjän antaessa päivämäärän virheellisessä muodossa komponentti kyllä ilmoittaa tästä käyttäjälle eri kanavia pitkin, mutta ilmoitus on hyvin geneerinen eikä korjausehdotuksia anneta.

Tällä hetkellä komponentin GitHub- ja NPM -sivuilla on vielä teksti, että käyttöä ei suositella ennen kuin komponentin versio on 1.0.0. Tämä johtuu siitä, että haluan vielä tehdä jatkokehitystä saavutettavuuden sekä muun muassa lokalisaation parissa sekä helpottaa komponentin kustomointia CSS-parannuksilla ennen kuin suosittelen sitä käyttöön.

6.1 Ylläpitotarve ja jatkokehitys

Lähdekoodi komponenttitiedostossa on rakennettu ilman ulkoisia riippuvuuksia esimerkiksi komponentin osia tai valmiita tyylejä sisältäviin kirjastoihin. Nämä olisivat voineet vähentää työmäärää, mutta samalla riski riippuvuuksien vanhenemiseen tai käyttöä haittaaviin muutoksiin olisi kasvanut. Tämä yksinkertaistaa osaltaan komponentin ylläpitoa.

Jatkokehitykseen kuuluu muun muassa viimeisten onnistumiskriteerien toteutumisen toteutus ja varmistaminen. Virheiden käsittelyä ja virheviestien välittämistä käyttäjälle tulee kehittää. Lisäksi jatkosuunnitelmaan kuuluu CSS-sääntöjen implementoinnin helpottamiseen liittyviä muutoksia sekä tumman teeman lisäyksen komponenttiin. Teeman lisäystä on alustettu esimerkiksi tekemällä ikonista sekä vaaleasta taustasta erottuva tumma versio että tummaan taustaan käyvä vaalea versio.

Komponenttitiedoston lähdekoodin pituus kasvoi kehityksen aikana laajaksi ollakseen yhden tiedoston komponentti. Harkitsen koodin jakamista useampiin alikomponentteihin

esimerkiksi ylläpidettävyyden ja hahmotettavuuden lisäämiseksi. Tällä hetkellä joitakin TypeScript-tiedostoja on erotettu komponentista. Lisäksi minulla oli haasteita saada svg-muotoinen kalenteri-ikoni erillisenä tiedostona NPM-pakettiin. Lokaalisti tämä toimi, mutta jotta sain sen toimimaan NPM:ssä, jouduin lopulta lisäämään sen suoraan osaksi komponenttitiedostoa. Haluan jatkokehityksessä selvittää, kuinka saan kuvat toimimaan halutulla tavalla.

Jotta kehittäjät tietävät, kuinka käyttää komponenttia projekteissaan, minun tulee tarkentaa projektin README.md-tiedostoon kattavat ohjeistukset. README tulee näkyviin sekä projektin GitHub- että NPM-sivuille. Tällä hetkellä teksti on vielä hyvin ylimalkainen.

Aikomuksena on avata komponentin jatkokehitys yleisesti avoimen lähdekoodin projektiksi, jolloin muutkin ohjelmistokehittäjät voivat osallistua siihen. Haluan kuitenkin ensin lisätä ohjeistukset README-sivulle siitä, millä tavoin kehitys tässä tapauksessa tapahtuisi. Avoimen lähdekoodin projektina kehitysehdotuksia tulee todennäköisesti paljon enemmän, kun useampi silmäpari tutkii komponentin toimivuutta. Tämä voisi myös tehostaa kehitystä mukavasti.

7 Pohdinta

Tavoitteenani opinnäytetyössä oli rakentaa saavutettava, uudelleenkäytettävä päivämääräkomponentti Vuella ja oppia tätä kautta lisää saavutettavuuden ohjelmallisesta toteutuksesta. Koen suurelta osin päässeeni tavoitteisiin, joskin komponentin saavutettavuuteen jäi vielä jonkin verran parannettavaa jatkokehityksen aikana toteutettavaksi.

Teoriaosuudesta kehkeytyi laajahko kokonaisuus, joka jälkikäteen katsottuna olisi voinut suuntautua vielä enemmän saavutettavuuden tekniseen toteutukseen. Laajasti saavutettavuuteen perehtyminen kuitenkin auttoi hahmottamaan käyttäjien tarpeita onnistumiskriteerien taustalla. Komponentin kehitystä aloittaessani minulle oli selvää, mistä löytäisin kriteerien tulkintaan apua. Toivon voivani hyödyntää kertynyttä ymmärrystä jatkossa esimerkiksi työpaikkani sisällä osaamisen jakamisen muodossa.

Vaikka olin jo opinnäytetyötä aloittaessani tietoinen progressiivisilla JavaScript-kirjastoilla toteutettujen, saavutettavien päivämääräkomponenttien vähyydestä, asia valkeni minulle entisestään kehitysprosessin aikana. Etsinnästä huolimatta minulla ei ole opinnäytetyön loppuvaiheilla mielessäni yhtään julkista Vue-päivämääräkomponenttia, jonka saavutettavuudesta olisin varma. Tämä ei silti tarkoita, etteikö niitä olisi olemassa. Kuitenkin tämä lisäsi tunnetta työn tarpeellisuudesta. Opinnäytetyöprojekti on myös kirvoittanut ideoita uusien saavutettavien komponenttien kehityksestä tulevaisuudessa.

Päivämääräkomponentin muodolle olisi ollut monia vaihtoehtoja. Yksinkertaisimmillaan siitä olisi voinut tehdä pelkän tekstikentän tai esimerkiksi päivän, kuukauden ja vuoden syöttämiselle omat erilliset tekstikenttänsä tai alasvetovalikkonsa. Saavutettavuusmuutokset olisivat saattaneet olla huomattavasti kevyempiä toteuttaa kuin visuaalisen kalenterin sisältävässä komponentissa. Yksi tavoitteenani oli kuitenkin juuri oppia lisää saavutettavuudesta myös ohjelmallisesti. Tätä tuki valinta tehdä toiminnoiltaan ja sisällöltään hieman monipuolisempi komponentti, jonka muotoa pidin myös vakiintuneena käytäntönä.

Jos olisin ottanut komponenttiin osia tai esimerkiksi tyylejä valmiista ulkoisista kirjastoista, olisi ongelma todennäköisesti ollut sama kuin mikä JavaScript-valmiskomponenttien käyttöön yleisestikin liittyy; palikoiden sisäisiin toimintoihin tai muihin ominaisuuksiin olisi saattanut olla hyvin vaikeaa vaikuttaa. Siksi, vaikka alussa komponentin rakentaminen alusta lähtien vaati paljon tarkkaa työtä, kehityksen loppua kohden enemmän ja enemmän sen hyötyjä alkoi tulla esille. Pystyin helposti muokkaamaan esimerkiksi yksittäisten näppäimistö-eventien aiheuttamia muutoksia ja niiden poikkeuksia. Opin myös arvioimaan WAI-ARIAn tarpeellisuutta entistä paremmin erilaisten elementtien kohdalla.

Opinnäytetyö antoi hyvän pohjan asiantuntemuksen kasvattamiselle saavutettavuudessa sekä avoimen lähdekoodin komponenttien rakentamiselle. Syvällinen perehtyminen WCAG:n sisältöön ja siihen liittyvien dokumenttien suhteisiin auttoi huomattavasti onnistumiskriteerien sisällön tulkitsemisessa. Kriteerien lisäksi arviointisuunnitelman tekeminen auttoi suuntaamaan tekemistä kehityksen aikana ja lopputestauksessa. Käyttäjätestaus saavutettavuudesta erityisesti hyötyvän henkilön toimesta jäi valitettavasti opinnäytetyön puitteissa toteutumatta. Olen kuitenkin iloinen siitä, että sain saavutettavuuteen perehtyneen ohjelmistokehittäjän arvion komponentista.

Opinnäytetyön tekeminen on ollut minulle hyvin mieluinen ja antoisa kokemus. Jos vertaa lähtötilannetta keväällä 2021 opinnäytetyön palautuksen hetkiin keväällä 2022, koen ymmärrykseni niin yleisen kuin teknisenkin saavutettavuuden osalta lisääntyneen huimasti. Antoisuutta lisää se, että tämän aihepiirin kautta sain yhdistää nykyiseen alaani ohjelmistokehittäjänä elementtejä aiemmasta tutkinnostani sosiaalialalta. Kummallakin ammattialalla ihmisten tarpeiden tulee olla tekemisen lähtökohtana.

Lähteet

Aalen, I. 2018. It’s illegal to have an inaccessible website in Norway — and that’s good news for all of us. Confrere. Luettavissa: https://medium.com/confrere/its-illegal-to-have-an-inaccessible-website-in-norway-and-that-s-good-news-for-all-of-us-b59a9e929d54. Luettu: 29.9.2021.

Abou-Zahra, S., Steenhout, N., & Keen, L. 2017. Selecting Web Accessibility Evaluation Tools. Luettavissa: https://www.w3.org/WAI/test-evaluate/tools/selecting/. Luettu: 3.5.2022.

Aluehallintovirasto s.a.a. Kenelle saavutettavuus on tärkeää? Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/. Luettu: 9.6.2021.

Aluehallintovirasto s.a.b. Yleistä saavutettavuudesta. Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/. Luettu 12.8.2021.

Aluehallintovirasto s.a.c. Sanastoa ja termejä. Luettavissa: https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/sanastoa-ja-termeja/#design-for-all-kaikille-sopiva-suunnittelu-suunnittele-kaikille-periaate. Luettu: 29.9.2021.

Aluehallintovirasto s.a.d. Tietoa WCAG-ohjeistuksesta. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/. Luettu: 21.11.2021.

Aluehallintovirasto s.a.e. WCAG 2.1: lain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/. Luettu: 1.9.2021.

Aluehallintovirasto s.a.f. Soveltamisala: kuulummeko lain piiriin? Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/soveltamisala-kuulummeko-lain-piiriin/#yksityiset. Luettu: 1.9.2021.

Armfield, G. 2019. Is input type=”date” ready for use in accessible websites? Luettavissa: https://www.hassellinclusion.com/blog/input-type-date-ready-for-use/. Luettu: 30.4.2019.

Brewer, J. & Henry, S. 2020. About W3C WAI. Luettavissa: https://www.w3.org/WAI/about/#world-wide-web-consortium-w3c-web-accessibility-initiative-wai. Luettu: 31.10.2021.

Chacon, S. & Straub, B. 2022. Pro Git. 2. painos. The Expert's Voice. E-kirja. Luettu 14.5.2022.

Chisholm, W., Vanderheiden, M., Jacobs, I. 1999. Web Content Accessibility Guidelines 1.0. W3C (MIT, INRIA, Keio). Luettavissa: https://www.w3.org/TR/WAI-WEBCONTENT/. Luettu: 31.10.2021.

Cooper, M. 2020. WAI-ARIA Overview. Luettavissa: https://www.w3.org/WAI/standards-guidelines/aria/. Luettu: 11.5.2022.

Dawns, B. 2021. 2021 Stack Overflow Survey: React.js takes the web framework crown, Python is in-demand, and devs still love Rust. Developer Tech. Luettavissa: https://www.developer-tech.com/news/2021/aug/03/2021-stack-overflow-survey-react-js-takes-the-web-framework-crown-python-is-in-demand-and-devs-still-love-rust/. Luettu: 29.4.2022.

Duet Date Picker 2021. Komponentin Node Package Manager -sivu. Luettavissa: https://www.npmjs.com/package/@duetds/date-picker. Luettu 26.5.2022.

Education and Outreach Working Group 2021. Involving Users in Web Accessibility Overview. Video. Lataaja: Web Accessibility Initiative. Katsottavissa: https://www.w3.org/WAI/test-evaluate/. Katsottu: 31.12.2021.

Eficode 2021a. Saavutettavuus. Huomioi jokainen palvelun käyttäjä. Luettavissa: https://www.eficode.com/fi/palvelut/saavutettavuus. Luettu: 6.12.2021.

Eficode 2021b. Verkkopalveluiden top 5 -saavutettavuusongelmat. Luettavissa: https://www.eficode.com/fi/blog/verkkopalveluiden-saavutettavuusongelmat. Luettu: 6.12.2021.

Eur-lex 2016. Euroopan parlamentin ja neuvoston direktiivi julkisen sektorin elinten verkkosivustojen ja mobiilisovellusten saavutettavuudesta (EU) 2016/2102. Annettu 26.10.2016. Luettavissa: https://eur-lex.europa.eu/legal-content/FI/TXT/?uri=CELEX%3A32016L2102. Luettu: 26.3.2021.

European Telecommunications Standards Institute 2015. Accessibility requirements suitable for public procurement of ICT products and services in Europe. EN 301 549 V1.1.2 (2015-04). Luettavissa:

https://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.02_60/en_301549v010102p.pdf. Luettu: 31.10.2021.

European Telecommunications Standards Institute 2021. Accessibility requirements for ICT products and services. EN 301 549 V3.2.1 (2021-03). Luettavissa: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf. Luettu: 31.10.2021.

Fennia 2020. Kotitalousasiakkaiden Oma Fennian saavutettavuusseloste. Luettavissa: https://www.fennia.fi/saavutettavuus/kotitalousasiakkaiden-oma-fennian-saavutettavuus-seloste-julkaistu-22-9-2020. Luettu: 6.12.2021.

Finavia 2021a. Helsinki-Vantaan virallinen pysäköinti. Luettavissa: https://www.finavia.fi/fi/lentoasemat/helsinki-vantaa/pysakointi. Luettu 6.12.2021.

Finavia 2021b. Saavutettavuusseloste. Luettavissa: https://www.finavia.fi/fi/saavutettavuus. Luettu 6.12.2021.

Finlex 2019. Laki digitaalisten palvelujen tarjoamisesta 15.3.2019/306. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2019/20190306. Luettu: 26.3.2021.

Gil, P. 2020. Internet vs. Web: What's the Difference. Lifewire. Luettavissa: https://www.lifewire.com/difference-between-the-internet-and-the-web-2483335. Luettu: 24.9.2021.

Gilbert, R 2019. Inclusive Design for a Digital World - Designing with Accessibility in Mind. Apress. New York, NY, USA. Luettu 26.9.2021.

Git 2022. Git--distributed-is-the-new-centralized. Luettavissa: https://git-scm.com/. Luettu: 14.5.2022.

Haben, G. 2017. Break down disability barriers to spur growth and innovation. Financial Times. Workplace diversity & equality. Luettavissa: https://www.ft.com/content/d8997604-97ab-11e7-8c5c-c8d8fa6961bb. Luettu: 29.9.2021.

Henry, S. 2006. Understanding Web Accessibility. Teoksessa Regan, B., Rutter, r., Urban, M., Heilmann, C., Burks, M., Lawson, B., Waddell, C., Lawton Henry, S., Thatcher,

J., Lauke, P., & Kirkpatrick, A. (toim.). Web Accessibility: Web Standards and Regulatory Compliance, luku 1. Friends of ED.

Henry, S. 2021a. Web Content Accessibility Guidelines (WCAG) Overview. Luettavissa: https://www.w3.org/WAI/standards-guidelines/wcag/. Luettu: 31.10.2021.

Henry, S. 2021b. W3C Accessibility Standards Overview. Luettavissa: https://www.w3.org/WAI/standards-guidelines/. Luettu: 3.11.2021.

Henry, S. 2021c. Mobile Accessibility at W3C. Luettavissa: https://www.w3.org/WAI/standards-guidelines/mobile/. Luettu: 11.5.2021.

Henry, S. 2021d. WCAG 2 Documents. Luettavissa: https://www.w3.org/WAI/standards-guidelines/wcag/docs/. Luettu: 20.11.2021.

Henry, S. 2021d. WCAG 2 Evaluating Web Accessibility Overview. Luettavissa: https://www.w3.org/WAI/test-evaluate/. Luettu: 28.12.2021.

Henry, S., White, K. & Abou-Zahra, S. 2016. Accessibility, Usability, and Inclusion. Luettavissa: https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/. Luettu: 3.12.2021.

Invalidiliitto s.a. Saavutettavuus. Luettavissa: https://www.invalidiliitto.fi/esteettomyys/saavutettavuus. Luettu: 27.9.2021.

IteWiki s.a. Mobiilisovellus. Luettavissa: https://www.itewiki.fi/opas/mobiilisovellus/. Luettu 10.6.2021.

Jokinen, J. 2020. Implementing web accessibility to an existing web application. Master's Thesis in Technology. Software Engineering. Department of Future Technologies. University of Turku.

Kehitysvammaliitto ry 2019. Verkkosisällön saavutettavuusohjeet (WCAG) 2.1. Alkuperäinen suositus on julkaistu 2018. Luettavissa: https://www.w3.org/Translations/WCAG21-fi-20191122/. Luettu: 3.11.2021.

Kirkpatrick, A., O'Connor, J., Campbell, A. & Cooper, M. 2018.
Web Content Accessibility Guidelines (WCAG) 2.1. Luettavissa: https://www.w3.org/TR/2018/REC-WCAG21-20180605/. Luettu: 3.11.2021.

Lewthwaite, S. & James, A. (2020) Accessible at last?: what do new European digital accessibility laws mean for disabled people in the UK?. Disability & Society, 35:8, 1360-1365, DOI: 10.1080/09687599.2020.1717446. Luettavissa: https://doi.org/10.1080/09687599.2020.1717446. Luettu: 1.6.2021.

Nian Li & Bo Zhang. 2021. The Research on Single Page Application Frontend development Based on Vue. Journal of Physics: Conference Series 1883 012030. Luettavissa: https://iopscience.iop.org/article/10.1088/1742-6596/1883/1/012030/pdf. Luettu 14.5.2022.

NPM 2022. NPM. Luettavissa: https://www.npmjs.com/. Luettu 15.5.2022.

NPM Docs s.a. About npm. Luettavissa: https://docs.npmjs.com/about-npm. Luettu 15.5.2022.

MDN Web Docs 2021a. HTML: HyperText Markup Language. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/HTML. Luettu: 5.9.2021.

MDN Web Docs 2021b. About JavaScript. What is JavaScript? Luettavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript#what_is_javascript. Luettu: 10.6.2021.

MDN Web Docs 2022a. HTML. Luettavissa: https://developer.mozilla.org/en-US/docs/Glossary/HTML. Luettu: 29.4.2022.

MDN Web Docs 2022b. CSS basics. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics. Luettu: 29.4.2022.

MDN Web Docs 2022c. CSS: Cascading Style Sheets. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/CSS. Luettu: 29.4.2022.

MDN Web Docs 2022d. JavaScript. Luettavissa: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript. Luettu: 29.4.2022.

MDN Web Docs 2022e. Document Object Model (DOM). Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Luettu: 15.5.2022.

MDN Web Docs 2022f. Introduction to the DOM. Luettavissa: https://vuejs.org/guide/extras/rendering-mechanism.html. Luettu: 15.5.2022.

MDN Web Docs 2022g. Rendering Mechanism. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Luettu: 15.5.2022.

MDN Web Docs 2022h. ARIA. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA. Luettu: 26.5.2022.

MDN Web Docs 2022i. What is accessibility? Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility. Luettu: 26.5.2022.

MDN Web Docs 2022j. WAI-ARIA basics. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics. Luettu: 26.5.2022.

Van Aelstyn, E. 2021. Internet Explorer 11 desktop app retirement FAQ. Microsoft. Luettavissa: https://techcommunity.microsoft.com/t5/windows-it-pro-blog/internet-explorer-11-desktop-app-retirement-faq/ba-p/2366549. Luettu: 2.5.2022.

New Zealand Ministry of Education 2021. Examples of students using assistive technology. Luettavissa: https://www.education.govt.nz/school/student-support/special-education/assistive-technology/examples-of-students-using-assistive-technology/ Luettu: 5.12.2021.

Näkövammaisten liitto 2021. Saavutettavuus. Luettavissa: https://www.nkl.fi/fi/saavutettavuus-1. Luettu: 26.9.2021.

OHCHR 2020. United Nations Human Rights Office of the High Commissioner. COVID-19 Response. Covid-19 and the Rights of Persons With Disabilities: Guidance. Newsletter. Luettavissa: https://www.ohchr.org/Documents/Issues/Disability/COVID-19_and_The_Rights_of_Persons_with_Disabilities.pdf. Luettu: 9.6.2021.

Papunet 2021a. Papunet. Luettavissa: https://papunet.net/. Luettu: 7.6.2021.

Papunet 2021a. Saavutettavuus. Luettavissa: https://papunet.net/saavutettavuus/. Luettu: 7.6.2021.

Papunet 2021b. Saavutettavuus. Miksi saavutettava? Luettavissa: https://papunet.net/saavutettavuus/miksi-saavutettava. Luettu: 12.8.2021.

Papunet 2021c. Saavutettavuuden arvioinnin suunnitelma. Luettavissa: https://papunet.net/saavutettavuus/saavutettavuuden-arvioinnin-suunnitelma. Luettu 1.5.2022.

Papunet 2021d. Saavutettavuuden arviointi. Luettavissa: https://papunet.net/saavutettavuus/saavutettavuuden-arviointi. Luettu 7.9.2021.

Papunet 2021e. Avustavat teknologiat. Luettavissa: https://papunet.net/saavutettavuus/avustavat-teknologiat. Luettu 1.5.2022.

Papunet 2021f. Näppäimistöselaaminen. Luettavissa: https://papunet.net/saavutettavuus/nappaimistoselaaminen. Luettu 3.5.2022.

Papunet 2021g. Ruudunlukuohjelmat. Luettavissa: https://papunet.net/saavutettavuus/ruudunlukuohjelmat. Luettu 15.5.2022.

Papunet 2021h. Silmän liikkeillä ohjattava kohdistin. Luettavissa: https://papunet.net/saavutettavuus/silman-liikkeilla-ohjattava-kohdistin. Luettu 15.5.2022.

Rahkola, M. 2017. Saavutettavuusdirektiivi ja sen kansallinen toimeenpano. PowerPoint-esitys. Valtionvarainministeriö. Luettavissa: https://www.kuntaliitto.fi/sites/default/files/media/file/Saavutettavuusdirektiivin%20kansallinen%20toimeenpanoRahkolaMarkus.pdf. Luettu: 31.10.2021.

Roggio, A 2015. Absence of U.S. Regulation Leads to Web Accessibility Lawsuits. Practical Ecommerce. Luettavissa: https://www.practicalecommerce.com/Absence-of-U-S-Regulation-Leads-to-Web-Accessibility-Lawsuits. Luettu: 29.9.2021.

Schwaber, K & Sutherland, J. 2020 Scrum-opas. Scrumin määritelmä ja pelisäännöt. Luettavissa: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Finnish.pdf. Luettu: 14.5.2022.

Shadi, Z. 2017. How People with Disabilities Use the Web. Luettavissa: https://www.w3.org/WAI/people-use-web/. Luettu: 5.12.2021.

Shadi, Z. & Sinclair, N. 2017. Stories of Web Users. Luettavissa: https://www.w3.org/WAI/people-use-web/user-stories/. Luettu: 5.12.2021.

Shaping Europe’s digital future 2021. Web Accessibility. Luettavissa: https://thl.fi/fi/web/vammaispalvelujen-kasikirja/vammaisuus-yhteiskunnassa/esteetto-myys-ja-saavutettavuus#normi. Luettu: 7.6.2021.

Suomen YK-liitto 2015. YK:n yleissopimus vammaisten henkilöiden oikeuksista ja sopimuksen valinnainen pöytäkirja. Luettavissa: https://www.ykliitto.fi/sites/www.ykliitto.fi/files/vammaisten_oikeudet_2016_net.pdf. Luettu 9.6.2021.

Takala, A. 2021. ”En pysty tekemään itse mitään” – Helsingin yliopiston suursatsauksena tehty tietojärjestelmä on osalle käyttäjistä painajainen. Helsingin sanomat. Luettavissa: https://www.hs.fi/kaupunki/helsinki/art-2000007984231.html. Luettu: 8.6.2021.

TechTerms 2021. Web Browser. Luettavissa: https://techterms.com/definition/web_browser. Luettu 10.6.2021.

Terveyden ja hyvinvoinnin laitos 2021. Esteettömyys ja saavutettavuus. Luettavissa: https://digital-strategy.ec.europa.eu/en/policies/web-accessibility. Luettu: 2.6.2021.

Tulli s.a.a. Passituksen varamenettely (Toiminnan jatkuvuutta koskeva menettely). Luettavissa: https://link.webropolsurveys.com/Participation/Public/5555c842-3089-408f-9b20-4ea8219a36e1?displayId=Fin1882826&surveyLocale=fi. Luettu: 6.12.2021.

Tulli s.a.b. Saavutettavuusseloste - Ilmoitus passituksen varamenettelyyn siirtymisestä. Luettavissa: https://tulli.fi/tietoa-tullista/saavutettavuus/ilmoitus-passituksen-varamenettelyyn-siirtymisesta. Luettu: 6.12.2021.

TypeScript 2022. TypeScript for JavaScript Programmers. Luettavissa: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Luettu 2.5.2022.

UiAccess 2010. Luettavissa: http://uiaccess.com/teachDI.html#projbits. Luettu: 6.12.2021.

United Nations Convention on the Rights of Persons with Disabilities 2006. Luettavissa: https://www.un.org/disabilities/documents/convention/convention_accessible_pdf.pdf. Luettu: 9.6.2021.

Vainio, V. 2018. Järjestelmä web-palveluiden nopeaan kehitykseen. Pro gradu -tutkielma. Helsingin yliopisto. Tietojenkäsittelytieteen osasto. Luettavissa: https://helda.helsinki.fi/bitstream/handle/10138/299753/J%c3%a4rjestelm%c3%a4%20web-palveluiden%20nopeaan%20kehitykseen.pdf?sequence=2&isAllowed=y. Luettu 14.5.2022.

Valtionvarainministeriö s.a. Saavutettavuus. Luettavissa: https://vm.fi/saavutettavuusdirektiivi. Luettu 10.6.2021.

Vue.js 2021. Introduction. Luettavissa: https://vuejs.org/v2/guide/. Luettu 10.6.2021.

Vue.js 2022. The Progressive JavaScript Framework. Luettavissa: https://vuejs.org/. Luettu 14.5.2022.

Vue.js s.a. Components Basics. Luettavissa: https://vuejs.org/v2/guide/components.html. Luettu: 10.6.2021.

Vue.js s.a.b. Components. Luettavissa: https://v1.vuejs.org/guide/components.html. Luettu 10.6.2021.

Vue.js s.a.c. Introduction. Luettavissa: https://vuejs.org/guide/introduction.html. Luettu 14.5.2022.

Vue.js s.a.d. Components Basics. Luettavissa: https://vuejs.org/guide/essentials/component-basics.html. Luettu 15.5.2022.

W3C 1997. World Wide Web Consortium Launches International Program Office for Web Accessibility Initiative. Luettavissa: https://www.w3.org/Press/IPO-announce [Google Scholar]. Luettu 6.6.2021.

W3C 2004. W3C Patent Policy. Luettavissa: https://www.w3.org/Consortium/Patent-Policy-20040205/. Luettu: 24.10.2021.

W3C 2016. HTML & CSS. Luettavissa: https://www.w3.org/standards/webdesign/htmlcss. Luettu: 29.4.2022.

W3C 2018a. Web Content Accessibility Guidelines (WCAG) 2.1. Luettavissa: https://www.w3.org/Translations/WCAG21-fi/. Luettu: 10.6.2021.

W3C 2018b. Accessibility. Luettavissa: https://www.w3.org/standards/webdesign/accessibility. Luettu: 11.8.2021.

W3C 2021a. About W3C. Luettavissa: https://www.w3.org/Consortium/. Luettu: 10.6.2021.

W3C 2021b. Introduction to Web Accessibility. Luettavissa: https://www.w3.org/WAI/fundamentals/accessibility-intro/ Luettu: 27.9.2021.

W3C 2021c. W3C Mission. Luettavissa: https://www.w3.org/Consortium/mission. Luettu: 20.10.2021.

W3C 2021d. WAI-ARIA Authoring Practices 1.2. Luettavissa: https://www.w3.org/TR/wai-aria-practices/#grid. Luettu: 11.05.2021.

W3C s.a. Date Picker Dialog Example. Luettavissa: https://www.w3.org/TR/wai-aria-practices/examples/dialog-modal/datepicker-dialog.html. Luettu: 8.3.2022.

W3Schools 2021a. CSS Introduction. Luettavissa: https://www.w3schools.com/css/css_intro.asp. Luettu: 10.6.2021.

W3Schools 2022a. HTML Accessibility. Luettavissa: https://www.w3schools.com/html/html_accessibility.asp. Luettu: 29.4.2022.

W3Schools 2022b. JavaScript Tutorial. Luettavissa: https://www.w3schools.com/js/default.asp. Luettu: 1.5.2022.

W3Schools 2022c. What is JavaScript. Luettavissa: https://www.w3schools.com/whatis/whatis_js.asp. Luettu: 1.5.2022.

WAI 2021. Understanding WCAG 2.1. Luettavissa: https://www.w3.org/WAI/WCAG21/Understanding/. Luettu: 17.11.2021.

WebAIM s.a. Contrast Checker. Luettavissa: https://webaim.org/resources/contrastchecker/. Luettu: 15.5.2022.

White, K., Abou-Zahra, S.& Henry, S. 2016a. Planning & Policies. Initiate. Luettavissa: https://www.w3.org/WAI/planning-and-managing/initiate/. Luettu 27.12.2021.

White, K., Abou-Zahra, S.& Henry, S. 2016b. Planning & Policies. Sustain. Luettavissa: https://www.w3.org/WAI/planning-and-managing/sustain/. Luettu 27.12.2021.

White, K., Abou-Zahra, S.& Henry, S. 2019. Planning & Policies. Plan. Luettavissa: https://www.w3.org/WAI/planning-and-managing/plan/. Luettu 27.12.2021.

World Health Organization 2021a. Disability and Health. Luettavissa: https://www.who.int/news-room/fact-sheets/detail/disability-and-health. Luettu: 8.6.2021.

World Health Organization 2021b. Disability. Luettavissa: https://www.who.int/health-topics/disability#tab=tab_1. Luettu: 8.6.2021.

Zahra, S. 2019. Accessibility Principles. Luettavissa: https://www.w3.org/WAI/fundamentals/accessibility-principles/#standards. Luettu: 3.11.2021.

Liitteet

Liite 1. Päivämääräkomponentin saavutettavuuden arviointisuunnitelma

Tämä arviointisuunnitelma on toteutettu Papunet-sivustolta löytyvän saavutettavuuden arvioinnin suunnitelman (Papunet 2021c) pohjalta, käyttötarkoitukseen soveltaen.

Arvioinnin tavoite ja tarkoitus

Arvioinnilla pyritään varmistamaan, että päivämääräkomponentin kehityksessä huomioidaan saavutettavuus tarkkuudella, joka mahdollistaa komponentin käytön henkilöille, joilla on erilaisia vammoja ja toimintarajoitteita. Lisäksi halutaan saavuttaa komponentti, jonka käyttö on mahdollista myös avustavilla teknologioilla.

Päivämääräkomponentin saavutettavuuden tavoiteltava taso on täytetty sitten, kun sen voi katsoa läpäisevän kaikki WCAG 2.1 -onnistumiskriteerien tasojen A ja AA kriteerit. Arvioinnin tavoitteena on tuoda esille, mitä muutoksia eri kriteerit asettavat komponentille.

Arviointimenetelmät

Kehityksen aikana teen arviointia itse peilaamalla komponentin osien toimintaa WCAG 2.1 -kriteereihin. Lisäksi hyödynnän sprinttien aikana erilaisia, kuhunkin tilanteeseen sopivia ulkoisia työkaluja saavutettavuuden testaamiseen. Kun komponentti on omasta mielestäni valmis, pyrin löytämään ulkopuolisen, potentiaalisen käyttäjän arvioimaan sen saavutettavuutta.

Käyn WCAG 2.1 -kriteereitä läpi ja arvioin niiden toteutumista komponentin kehityksen yhteydessä, kun se on ajankohtaista. Ryhmittelen kriteereitä tarpeen tullen ja jätän täysin epäolennaiset kohdat huomiotta. Lopussa summaan mitkä olennaisista kriteereistä täyttyvät ja mitkä eivät.

Arviointi toteutetaan pääasiassa Windows-käyttöjärjestelmän sisältävällä kannettavalla tietokoneella. Komponentin testaamiseen erikokoisilla laitteilla käytetään sekä selaimeen rakennettuja ominaisuuksia että mahdollisesti testaukseen soveltuvia ohjelmistoja ja erillisiä fyysisiä laitteita. Selaimessa näyttökoon pienentäminen vastaamaan tablet- ja puhelinkokoa toteutetaan selaimeen avattavan developer tools -ikkunan toggle device toolbar -valinnan kautta sekä mahdollisesti myös puhelimen näytöllä.

Loppuarvioinnin aikana varmistan mobiililaitetta käyttäen, että päivämääräkenttä toimii myös kosketusnäytöllä halutulla tavalla. Chromen lisäksi testaan komponentin toimintoja ja ulkoasua ainakin Firefox-, Microsoft Edge- ja Safari -selaimilla. Huomioin loppuarvioinnissani testauksessa käytettyjen selainvaihtoehtojen kattavuuden.

Komponenttia tulisi pystyä käyttämään myös ilman CSS:ää. Testaan komponentin toimintaa myös poistamalla tämän käytöstä. Tyylimääritysten poiston voi toteuttaa ainakin osalla selaimeen asennettavista testaustyökaluista. Käytän apuna todennäköisesti Wave-nimistä selainlaajennusta.

Avustavista teknologioista omissa testeissä käytetään niitä, jotka ovat jo valmiiksi asennettuna omalla kannettavalla koneellani ja joiden käyttö on helposti omaksuttavissa. Näitä ovat esimerkiksi näppäimistöselaaminen sekä ruudunlukuohjelma NVDA:n käyttö. Jos ulkopuolinen testaaja on mukana arvioinnissa kehityksen loppupuolella, hänellä saattaa olla käytettävänään jokin avustava teknologia.

Arviointi kohdistuu nimenomaan päivämääräkomponentin tekniseen saavutettavuuteen. Tämän ulkopuolista komponentin käytettävyyden arviointia vamman tai toimintarajoitteen kanssa eläville ei suunnitella toteutettavan opinnäytetyön puitteissa.

Arvioinnin kohde

Arvioinnin kohteena on päivämääräkomponentin toiminta ja ulkoasu. Tämä on siis hyvin rajattu alue, jossa ei huomioida esimerkiksi verkkosivulle tyypillisiä saavutettavuuteen liittyviä yksityiskohtia, kuten sivun title-arvoa. Tämä johtuu siitä, että komponenttia ei ole tarkoitus hyödyntää käytössä yksinään, vaan sen on tarkoitus olla osa toista verkkosivua. Komponenttia käyttävän verkkosivun kehittäjän vastuulla on huolehtia sivun saavutettavuudesta yleisesti.

Myös kalenterikomponentin label eli kentän tarkoituksen ilmaiseva nimi tai kysymys jätetään lopullisesta komponentista pois. Tämä saattaisi muutoin vaikuttaa liikaa mahdollisen käyttäjän komponentin ulkoasuun. Tämän pitäisi siis muokata esimerkiksi fontit ja labelin sijainti ylikirjoittamalla komponentissa määriteltyjä ulkoasusääntöjä.

Erityistä huomiota arvioinnin kohteessa vaatii päivämääräkentän yhteyteen avautuva kalenterinäkymä, josta voi visuaalisen kalenteri-ikkunan kautta valita haluamansa päivän. Kalenterinäkymän toteutukseen tarvitaan WAI-ARIA-määrityksillä höystettyjä elementtejä, koska sen eri osille ei ole olemassa niiden käyttötarkoituksia kuvaavia, semanttisia HTML-elementtejä.

Arvioinnin aikataulu

Arviointia toteutetaan kehityssprinttien aikana sen mukaan, kun se on ajankohtaista. Komponenttia rakentaessa pyritään alusta lähtien tekemään valintoja, jotka edesauttavat komponentin saavutettavuutta. Tässä hyödynnetään kriteerien lisäksi ennalta kertynyttä tietämystä saavutettavuudesta.

Kun kaikki potentiaalisesti komponenttia koskevat onnistumiskriteerit on käyty läpi sprinttien aikana, tehdään lopuksi vielä yhteenveto siitä, miltä osin komponentti täyttää saavutettavuuskriteerit. Lisäksi kerrotaan mahdolliset saavutettavuusongelmat. Saavutettavuusongelmat ilmenevät todennäköisesti siinä, että osan onnistumiskriteereistä ei komponentissa voida katsoa täyttyvän.

Komponentin saavutettavuusarviointi linkittyy vahvasti yhteen muiden kehityksessä tarvittavien teknisten ratkaisujen toteutuksen kanssa. Tästä syystä arvioinnin toteutuksen aikataulu määräytyy kehityksen kokonaiskeston mukaan. Ulkopuolisen henkilön tekemään arviointiin tulee tämän lisäksi varata aikaa. Tähän sisältyvät mahdolliset testauksen jälkeiset korjaustyöt.

Liite 2. WCAG 2.1 -kriteerien toteutuminen päivämääräkomponentissa

Tämän liitteen sisällön periaatteissa, ohjeissa ja kriteereissä sekä osassa niitä koskevia huomioita on käytetty lähteenä sekä englanninkielistä W3C:n Web Content Accessibility Guidelines (WCAG) 2.1 -suositusta että sen virallista suomenkielistä käännöstä Verkkosisällön saavutettavuusohjeet (WCAG) 2.1. Käännöksen on toteuttanut Kehitysvammaliitto ry. (Kirkpatrick, A., O'Connor, J., Campbell, A. & Cooper, M. 2018; Kehitysvammaliitto ry 2019.)

Listasta on poistettu ilmeisen epäoleelliset A- ja AA-tason onnistumiskriteerit. Lisäksi siinä ei ole mainittu AAA-tason kriteerejä, jotka on jätetty arvioinnin ulkopuolelle. Poistojen jälkeen jäi jäljelle 37 kriteeriä. Esimerkkinä poistetusta kriteeristä pelkkää audio- tai videosisältöä koskeva kriteeri 1.2.1 sekä 2.4.4, joka koskee linkin tarkoituksen ilmenemistä kontekstissa. Komponentti ei ole pelkästään audio- tai videosisältö eikä se sisällä linkkejä. Kriteerien toteutuminen perustuu opinnäytetyön tekijän omaan arvioon. Osa kriteereistä toteutuu komponentissa, jos se on osana toista verkkosivua, kuten sähköistä lomaketta ja se on sisällytetty odotetulla tavalla. Esimerkiksi tekstikentän nimilapun eli labelin toteutus jää komponentin käyttäjän vastuulle.

Periaate: 1. Havaittava

Ohje 1.1 Tekstivastineet: Ei-tekstuaalinen sisältö tulee voida muuttaa erilaisiin saavutettavuudessa huomioitaviin muotoihin kuten suurikokoiseksi tekstiksi tai puheeksi. Tämän vuoksi kaikelle ei-tekstuaaliselle sisällölle tulee olla tarjolla tekstivastine.

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
1.1.1 Ei-tekstuaalinen sisältö (A): Kaikelle sisällölle, joka käyttäjälle esitetään, on olemassa samaan tarkoitusta vastaava tekstivastine	Sisältää useita poikkeuksia sääntöön. Niistä seuraavat voivat koskea kehitettävää päivämääräkomponenttia: - Käyttöliittymäkomponenteilla ja syötettä vastaanottavalla sisällöllä on käyttötarkoituksen osoittava nimi - ei-tekstuaalista sisältöä käytetään yksinomaan koristeena, osana visuaalista	**OK**

	muotoilua tai sitä ei lain-kaan esitetä käyttäjälle	

Ohje 1.2 Aikasidonnainen media: Aikasidoinnaiselle medialle tulee tarjota vastine

- mitkään ohjeen 1.2 kriteerit eivät kosketa päivämääräkomponenttia.

Ohje 1.3 Mukautettava: Sisällön esitystavan vaihtumisen ei tule aiheuttaa sitä, että sisältö tai rakenne menetetään.

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/to-teutuu osittain/ei toteudu/ei tietoa)
1.3.1 Informaatio ja suhteet (A): Hahmontamisen (engl. render) kautta välitettävä informaatio, rakenteet ja suhteet on mahdollista määrittää ohjelmallisesti tai ne ovat tarjolla tekstinä		**OK**
1.3.2 Merkityksen vaikutus järjestykseen (A): Merkityk-senmukainen sisällön jär-jestys on mahdollista selvit-tää ohjelmallisesti		**OK**
1.3.3 Aistinvaraiset omi-naispiirteet (A): Sisällön ymmärtämiseen ja hallitse-miseen vaikuttava tieto ei ole yksinomaan aistinva-raisten ominaispiirteiden varassa	Näppäimistöselaukseen liit-tyvien painikkeiden tarkka ohjeistus on tarjolla käyttä-jälle vain kuuloaistin välityk-sellä, samoin lista kaikista hyväksyttävistä päivämää-rämuodoista. Muilta osin kriteeri vaikuttaa toteutu-van.	**toteutuu osittain**
1.3.4 Asento (AA): Sisällön ulkoasu ja toiminta eivät ra-joitu yksinomaan päätelait-teen asentoon pysty- tai vaakasuunnassa. Jos näyt-tölaitteen asento kuitenkin	Komponentin ulkoasu ja toiminnot eivät ole riippu-vaisia siitä, onko laite vaaka- vai pystysuunnassa.	**OK**

on olennainen, voidaan säännöstä poiketa.		
1.3.5 Syötteen tarkoituksen määrittely (AA): Syötekenttien tarkoitus tulee olla selvitettävissä ohjelmallisesti, jos niissä kerätään käyttäjän tietoja ja jos - käyttötarkoitus löytyy kriteerien ohessa Verkkosisällön saavutettavuusohjeet -sivulla esitettävästä listasta ja - käytetty teknologia mahdollistaa syötteen tarkoituksen kuvaamisen	Mikäli kyseessä on päivämääräkomponentin käyttötarkoitus, jossa pyydetään esimerkiksi käyttäjän syntymäaikaa, tämä kriteeri pätee.	**toteutuu osittain**

Ohje 1.4 Erottuva: Sisällön näkemisen ja kuulemisen helpottaminen sekä etualan erottaminen taustasta

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
1.4.1 Värien käyttö (A): Väri ei ole ainut tapa, jolla - välitetään informaatiota - esitetään toimintoja tai - esitetään visuaalisia elementtejä		**OK**
1.4.3 Minimikontrasti (AA): Kontrastisuhteen tulee tekstin sekä tekstiä esittävien kuvien osalta olla vähintään 4,5:1. Poikkeukset: - teksti on isokokoista - teksti tai tekstiä esittävä kuva voidaan laskea oheissisällöksi - kyseessä on logo	Tekstiä on komponentissa input-kentässä placeholder-tekstinä sekä kun käyttäjä on valinnut jonkin päivämäärän. Kalenterinäkymässä on myös kuukautta ja vuotta kuvaavat otsikkotekstit ja kuukauden päiviä kuvaavat numerot. Niiden taustaväri muuttuu sen	**OK**

	mukaan, onko päivä valittuna tai kohdistettuna. Siksi kontrasti tarkistetaan kaikilla taustavärivaihtoehdoilla. Lisäksi peruuta- ja OK -painikkeiden tekstien kontrasti painikkeiden taustaväriin nähden tulee tarkistaa.	
1.4.4 Tekstikoon muuttaminen (AA): Sisällön ja sen toiminnallisuuksien tulee kestää tekstin suurentaminen ilman avustavaa teknologiaa aina 200 prosenttiin. Poikkeukset: - tekstitykset - tekstiä esittävät kuvat		**OK**
1.4.5 Tekstiä esittävien kuvien käyttö (AA): Tekstiä tulisi suosia informaation välittämisessä suhteessa tekstiä esittäviin kuviin, kun kyseessä on teknologia, jolla voi tuottaa visuaalisen esityksen. Poikkeukset: - tekstiä esittävä kuva on mukautettavissa - esitystapa on olennainen osa informaation välitystä	Päivämääräkomponenttiin ei opinnäytetyön tekemisen aikana ilmennyt tarvetta sisällyttää tekstiä esittäviä kuvia, joten tulkitaan että kriteeri läpäistään.	**OK**
1.4.10 Responsiivisuus (AA): Vieritystä (engl. scroll) ei tule ilmetä kahteen suuntaan eikä sisältöä tai sen toimintoja tule menettää, kun sisältö esitetään		**OK**

- 320 CSS-pikselin kokoisena ja vieritys tapahtuu pystysuunnassa - 256 CSS-pikselin kokoisena ja vieritys tapahtuu vaakasuunnassa Poikkeus: - sisällön osat, jotka toimintojen tai merkityksen vuoksi vaativat kahdensuuntaisen esitystavan		
1.4.11 Ei-tekstuaalisen sisällön kontasti (AA): Elementin kontrastisuhde viereisiin väreihin tulee olla vähintään 3:1, kun kyseessä on - käyttöliittymäkomponentin ja sen tilojen tunnistamiseksi vaadittu visuaalinen informaatio, paitsi jos komponentti on inaktiivinen tai jos uuden sisällön visuaalinen esitystapa on käyttäjäagentin määrittelemä ja sisältö on tuottajan puolesta muokkaamatonta - kyseessä on sisällön ymmärtämisen mahdollistava graafinen objekti, paitsi jos ulkoasu liittyy olennaisesti tietosisällön välitykseen		**OK**
1.4.12 Tekstin välistys (AA): Kun toteutetaan seuraavat tyylimääritykset merkkauskieliä sisältävään tai tiettyjä tekstin muotoilun ominaisuuksia tukevaan	Testattu seuraavilla CSS-määrityksillä sekä tekstikenttä että visuaalisen kalenterin sisältö: line-height:1.5; letter-spacing:0.12em;	**OK**

sisältöön ilman muita tyylimäärityksiä, sisältöä tai sen toiminnallisuutta ei menetetä: - riviväli (rivin korkeus) vähintään 1,5 kertaa käytetyn fontin koko - kappaleen jälkeinen tyhjä tila vähintään kaksi kertaa käytetyn fontin koko - kirjainväli vähintään 0,12 kertaa käytetyn fontin koko - sanojen väli vähintään 0,16 kertaa käytetyn fontin koko Kielet ja kirjoitustavat, joissa ei noudateta jotakin yllä luetelluista ominaisuuksista, ovat poikkeuksia.	word-spacing:0.16em; margin-bottom:2em;	
1.4.13 Sisällön näkyminen osoitettaessa tai kohdistettaessa (AA): Mikäli kursorin tai näppäimistökohdistuksen siirtäminen elementin päälle tuo näkyviin sisältöä ja pois siirtyminen taas piilottaa sitä, tulee seuraavien ehtojen täyttyä: - Esiin tulleen sisällön saa piilotettua ilman kursorin tai näppäimistökohdistuksen siirtoa, paitsi jos kyseessä on syötevirheestä kertova teksti tai sisältö ei asetu muun sisällön päälle tai tilalle		**OK**

- kursori on mahdollista viedä esiin tulleen sisällön päälle ilman että se katoaa - uusi sisältö ei katoa näkyvistä ennen kuin kursori tai näppäimistökohdistus on viety pois, käyttäjä poistaa sisällön näkyvistä tai sisällön välittämä informaatio ei enää päde		

Periaate: 2. Hallittava

Ohje 2.1 Käytettävissä näppäimistöltä: Kaikissa toiminnallisuuksissa tulee olla vaihtoehtona käyttää niitä näppäimistöllä

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
2.1.1 Näppäimistö (A): Sisältöä ja sen kaikkia toiminnallisuuksia voidaan käyttää näppäimistöllä ilman, että käyttäjän tulee ajoittaa yksittäisiä näppäinpainalluksiaan. Poikkeus: - taustatoiminto vaatii käyttäjältä syötteen, joka on riippuvainen käyttäjän etenemisreitistä eikä riipu pelkästään päätepisteestä		**OK**
2.1.2 Ei mahdollisuutta ajautua ansaan näppäimistöä käytettäessä (A): Mikäli pelkästään näppäimistörajapintaa käyttämällä on mahdollista siirtää kohdistus elementtiin, tulee kohdistus pystyä siirtämään		**OK**

myös elementin ulkopuolelle pelkästään näppäimistörajapinnan avulla. Lisäksi mikäli poistuminen vaatii muita menetelmiä kuin oletusarvoisesti käytettäviä menetelmiä, kuten muokkaamattomien nuoli- ja tab -näppäien käyttöä, tulee käyttäjälle kertoa vaihtoehtoinen poistumismenetelmä		
2.1.4 Pikanäppäimet, jotka koostuvat vain yhdestä merkistä (A): Mikäli toteutus sisältää yhden merkin näppäinoikotien toteutuksen, jonkin seuraavista säännöistä tulee toteutua: - näppäinoikotie on mahdollista kytkeä pois päältä - näppäinoikotie voidaan uudelleenmääritellä käyttämään vaihtoehtoista näppäinkomentoa, joka sisältää yhden tai useamman komentonäppäimen - näppäinoikotie on käytössä vain, kun kohdistus on sen sisältävässä komponentissa		**OK**

Ohje 2.2 Tarpeeksi aikaa: Käyttäjälle tulee antaa riittävästi aikaa sisällön lukemiseen ja käyttämiseen

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
2.2.1 Ajoituksen säätäminen (A): Mikäli sisältö		**OK**

asettaa aikarajan, tulee ainakin yhden seuraavista vaatimuksista toteutua: - Aikaraja on mahdollista kytkeä pois päältä ennen kuin aika on täynnä - ennen kun aika on täynnä, käyttäjän on mahdollista säätää aikarajaa laajasti siten, että aika on mahdollista säätää ainakin kymmenkertaiseksi oletusasetukseen nähden - Käyttäjälle näytetään varoitus vähintään 20 sekuntia ennen kuin aika loppuu ja käyttäjän on mahdollista yksinkertaisella toiminnolla siirtää aikarajaa ainakin kymmenen kertaa - aikaraja liittyy reaaliaikaiseen tapahtumaan eikä sitä sen vuoksi voida muuttaa - aikarajan pidentäminen liittyy olennaisesti toimintoon - aikarajan pituus on yli 20 tuntia		

Ohje 2.4 Navigoitava: Käyttäjälle tulee tarjota tapoja navigoida, etsiä sisältöä ja määritellä sijaintinsa

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
2.4.1 Usein toistuvien lohkojen ohitus (A): Mikäli sivuilla on toistuvia	Käyttäjän on mahdollista siirtyä päivämääräkomponentissa päivien ja	**OK**

sisältölohkoja, tulee käyttäjälle tarjota mahdollisuus niiden ohittamiseen	viikkojen tai kuukausien yli painikkeilla tai näppäimistön näppäimiä käyttämällä.	
2.4.2 Verkkosivujen otsikot (A): Verkkosivu sisältää otsikoita ja otsikot kuvaavat sivun aihetta tai merkitystä	Toteutuu päivämääräkomponentin sisällön osalta. Kalenterinäkymän ylälaidassa on tason 3 otsikko, joka kuvaa kulloinkin näkyvillä olevaa kuukautta ja vuotta. Koko verkkosivun osalta kriteerin toteutuminen jää komponenttia käyttävän verkkosivun sisällön toteuttajan vastuulle.	**OK**
2.4.3 Kohdistusjärjestys (A): Kohdistus siirtyy kohdistettavissa komponenteissa merkityksen ja toimivuuden säilyttävässä järjestyksessä, mikäli verkkosivu on luotu siten että sen läpi voi navigoida järjestyksessä ja navigointijärjestyksellä on vaikutusta merkityksen tai toimintojen kannalta		**OK**
2.4.6 Otsikot ja nimilaput (AA): Otsikot ja nimilaput (engl. labels) sisältävät aiheen tai tarkoituksen kuvaamisen	Riippuu parent-komponentista, onko päivämääräkentän nimilappu asetettu oikein. Edellytykset kriteerin toteutumiselle on huomioitu komponentissa.	**toteutuu osittain**
2.4.7 Kohdistuksen näkyminen (AA): Kaikki käyttöliittymät, joita voidaan käyttää näppäimistöltä, sisältävät käyttötilan, jossa kohdistus on näkyvissä		**OK**

Ohje 2.5 Syötetavat:

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
2.5.2 Mahdollisuus perua osoitinlaitteella tehty valinta (A): Yhden osoittimen toiminnoissa vähintään yksi seuraavista toteutuu: - alas-tapahtuman yhteydessä ei suoriteta mitään toiminnallisuuden osaa - ylös-tapahtuman yhteydessä päätetään toiminto ja toiminto voidaan perua ylös-tapahtumaa ennen tai sen jälkeen - alas-tapahtuman aikana suoritettu toiminnallisuus kumotaan ylös-tapahtumalla - on olemassa olennainen syy sille, miksi toiminto tulee päättää alas-tapahtuman yhteydessä		**OK**
2.5.3 Nimilapun (label) sisältämä teksti sama kuin komponentin nimessä (A): Mikäli komponentin nimilapussa (engl. label) on tekstiä tai sitä esittävä kuva, komponentin nimi (eri kuin nimilappu) sisältää visuaalisesti nähtävissä olevan tekstin	Kriteerin toteutuminen riippuu parent-komponenttiin asetettavasta labelista.	**ei tietoa**

Periaate: 3. Ymmärrettävä

Ohje 3.1 Luettava: Tekstisisällön tulee olla luettavissa olevaa ja ymmärrettävää

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
3.1.1 Verkkosivun luonnollinen kieli (A): On ohjelmallisesti mahdollista selvittää verkkosivun oletuksena käytetty luonnollinen kieli	Verkkosivun luonnollisen kielen ilmentäminen ohjelmallisesti on komponentin sisältävän sivuston ylläpitäjän määritettävissä. Komponenttiin on kuitenkin sisällytetty luonnollisen kielen tarkastus kieltä valittaessa, mikäli sitä ei ole syötetty sille attribuuttina parent-komponentista.	**ei tietoa**
3.1.2 Sisällön osien luonnollinen kieli (A): Luonnollinen kieli voidaan selvittää ohjelmallisesti jokaisen sisällössä esiintyvän tekstikatkelman tai ilmaisun osalta. Poikkeuksena - erisnimet - tekniset termit - sanat, jotka kuuluvat määrittelemättömään kieleen - sanat ja ilmaukset, jotka ovat siirtyneet toisessa kielessä käytettäväksi murteeksi		**OK**

Ohje 3.2 Ennakoitava: Verkkosivujen ilmaisun ja toiminnan tulee olla ennakoitavaa

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)

3.2.1 Kohdistus (A): Kohdistaminen ei aiheuta kontekstin muutosta		**OK**
3.2.2 Syöte (A): Komponentin asetuksen muuttaminen ei muuta kontekstia, jos käyttäjälle ei ole annettu asiasta etukäteen ohjeistusta ennen komponentin käyttöä		**OK**
3.2.3 Navigoitavien mekanismien järjestyksen johdonmukaisuus (AA): Verkkosivujoukossa tai useilla eri verkkosivuilla toistuvien navigointimekanismien järjestys suhteessa toisiinsa pysyy aina samana. Poikkeuksena tilanne, jossa käyttäjä valitsee toisin		**OK**
3.2.4 Komponenttien merkitsemisen johdonmukaisuus (AA): Samat toiminnallisuudet verkkosivujoukossa sisältävien komponenttien osalta noudatetaan johdonmukaista merkintätapaa		**OK**

Ohje 3.3 Syötteen avustaminen: Käyttäjien auttaminen virheiden välttämisessä ja korjaamisessa

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
3.3.1 Virheen tunnistaminen (A): Mikäli virhe tunnistetaan automaattisesti, käyttäjälle osoitetaan kohta		**OK**

jossa virhe on tapahtunut ja käyttäjä saa tiedon virheestä tekstimuodossa		
3.3.2 Nimilaput tai ohjeet saatavilla (A): Nimilaput tai ohjeet ovat saatavilla, mikäli käyttäjältä edellytetään syötettä	Kriteerin toteutuminen riippuu parent-komponenttiin asetettavasta labelista.	**ei tietoa**
3.3.3 Virheiden korjausehdotukset (AA): Mikäli virhe tunnistetaan automaattisesti ja korjausehdotukset ovat tiedossa, käyttäjä saa niistä tiedon. Poikkeuksena tietoturvan tai sisällön merkityksen vaarantuminen	Komponentti antaa käyttäjälle tiedon virheestä, mikäli syöttökentässä on annettu päivämäärä ei-hyväksyttävässä muodossa. Korjausehdotukset kuitenkin puuttuvat.	**ei toteudu**

Periaate: 4. Toimintavarma

Ohje 4.1 Yhteensopiva: Yhteensopivuus erilaisten käyttäjäagenttien, kuten avustavien teknologioiden kanssa tulee maksimoida

Kriteeri (taso)	Huomioita	Toteutuuko kriteeri (OK/toteutuu osittain/ei toteudu/ei tietoa)
4.1.1 Jäsentäminen (A): Merkintäkielillä toteutetussa sisällössä - elementtien alku- ja lopputagit ovat täydellisiä - elementtien sisäkkäisyys noudattaa spesifikaatiota - attribuutti esiintyy elementeillä korkeintaan kerran - id-tunnisteet ovat yksilöitävissä Poikkeuksena tilanteet, joissa määritykset hyväksyvät kyseiset ominaisuudet		**OK**

4.1.2 Nimi, rooli, arvo (A): Kaikille käyttöliittymäkomponenteille, muun muassa lomake-elementeille, linkeille ja komponenteille, jotka komentosarja muodostaa, - voidaan määrittää nimi ja rooli ohjelmallisesti - mikäli käyttäjä voi niitä asettaa, voidaan tilat, ominaisuudet ja arvot asettaa ohjelmallisesti - tieto edellä mainittujen muutoksista on mahdollista saada käyttäjäagenttien (esim. avustavat teknologiat) tietoon	Toteutuminen riippuu osittain parent-komponentista. Muutoin toteutuu.	**toteutuu osittain**
4.1.3 Tilasta kertovien viestien esittäminen (AA): Kun käyttäjälle välitetään tilasta kertovia viestejä, ne voidaan selvittää ohjelmallisesti avustavan teknologian avulla käyttäjälle ilman, että kohdistus siirtyy. Tämä koskee merkintäkielillä toteutettua sisältöä ja toteutukseen voidaan käyttää rooleja tai muita ominaisuuksia		**OK**