Karelia-ammattikorkeakoulu
Tradenomi (AMK), tietojenkäsittely

Responsiivinen suunnittelu ja responsiiviset tekniikat web-sovelluskehityksessä

Satu Kontinen

Opinnäytetyö, tammikuu 2024

www.karelia.fi

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
tammikuu 2024
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600

Tekijä(t)
Satu Kontinen

Nimeke
Responsiivinen suunnittelu ja responsiiviset tekniikat web-sovelluskehityksessä

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli tarkastella responsiivisuuden huomioimista osana verkkosivujen suunnittelua ja toteutusta. Opinnäytetyön tavoitteena oli toteuttaa responsiivinen verkkosivusto hyödyntämällä sivuston toteutuksessa siihen parhaiten soveltuvia suunnitteluperiaatteita ja sovelluskehitystekniikoita.

Opinnäytetyöraportti käsittelee ensin responsiivisuuden huomioimista osana verkkosivujen suunnittelua ja esittelee sitten keskeisimmät sovelluskehitystekniikat, joiden avulla responsiivisuus voidaan sisällyttää osaksi verkkosivujen toteutusta HTML-merkintäkielen ja CSS-tyylimäärittelykielen avulla. Opinnäytetyöprosessin aikana kehitetty verkkosivusto toteutettiin single-page web-applikaationa käyttäen ohjelmointikielinä JavaScript-kirjasto Reactia, HTML:ää ja CSS:ää. Toteutetun sivuston responsiivisuutta testattiin selaimen kehittäjätyökalujen ja testiautomaation avulla sekä tarkastelemalla sivustoa erilaisilla päätelaitteilla.

Sivusto suunniteltiin ja toteutettiin opinnäytetyöraportin käsittelemiä responsiivisuuteen liittyviä suunnitteluperiaatteita ja tekniikoita noudattaen. Suoritetun testauksen perusteella sivuston responsiivisuuden toteutus on onnistunut.

Kieli
suomi

Sivuja 37

Asiasanat
web-sovelluskehitys, käyttöliittymäsuunnittelu, responsiivisuus, Mobile First

THESIS
January 2024
Degree Programme in Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600

Author (s)
Satu Kontinen

Title
Responsive Design and Responsive Techniques in Web Application Development

Abstract

The purpose of this thesis was to examine how responsiveness should be taken in account during web application design and development stages. The objective was to create a responsive website by utilizing the most suitable design principles and development techniques in the implementation.

The thesis report first discusses how responsiveness should be considered during web design and then introduces the key application development techniques through which responsiveness can be included into the implementation of websites using HTML markup language and CSS style sheets. The developed website was built as a single-page web application by using JavaScript library React, HTML, and CSS as programming languages. The responsiveness of the developed website was tested by using browser developer tools and test automation, as well as by examining the site on various devices.

The website was designed and developed by following the responsive design principles and techniques discussed in the thesis report. Based on the test results, the implementation of the website's responsiveness has been successful.

Language
Finnish

Pages 37

Keywords
web development, user interface design, responsive design, Mobile First

Sisältö

1 Johdanto

Responsiivinen suunnittelu ja responsiiviset tekniikat osana web-sovelluskehitystä yleistyivät 2010-luvulla, kun verkon selaamiseen käytettävien päätelaitteiden kirjo laajentui tietokoneista muille laitteille, esimerkiksi älypuhelimille ja tableteille. Responsiivisuus on tärkeää huomioida osana verkkosivujen suunnittelua ja toteutusta, koska 2020-luvulle tultaessa on jo muodostunut yleiseksi käsitykseksi, että verkkosivujen tulee pystyä mukautumaan erilaisten selaamiseen käytettävien laitteiden näyttökokoon ja kuvasuhteeseen sopiviksi. Responsiivisuus liittyy olennaisesti myös verkkosivujen käytettävyyteen, saavutettavuuteen ja hakukonenäkyvyyteen. Näistä syistä se on tärkeä osa lähes minkä tahansa verkkoon tarkoitetun sisällön suunnittelua ja toteutusta.

Tämän opinnäytetyön aiheeksi responsiivisuus valikoitui web-käyttöliittymäsuunnitteluun ja web-sovelluskehitykseen liittyvän kiinnostukseni kautta. Opinnäytetyön kautta syvensin omaa webkehitykseen liittyvää ammatillista osaamistani perehtymällä tarkemmin responsiivisuuden huomioimiseen osana web-käyttöliittymien suunnittelua ja toteutusta. Opinnäytetyön tarkoituksena oli kartoittaa tähän parhaiten soveltuvia menetelmiä, toimintatapoja ja tekniikoita perehtymällä aihealuetta käsittelevään kirjallisuuteen sekä toteuttaa responsiivinen verkkosivusto. Opinnäytetyön tuotoksena syntynyt verkkosivusto on henkilökohtainen web-portfolio, jonka toteutus painottui responsiivisen websuunnittelun periaatteiden ja responsiivisten tekniikoiden hyödyntämiseen sovelluskehitysprosessin aikana.

2 Responsiivinen suunnittelu

2.1 Responsiivisen suunnittelun tarkoitus

Verkkoliikenteen analytiikka- ja tilastopalveluja tarjoavan Statcounterin mukaan vuonna 2023 maailmanlaajuisesta verkkoliikenteestä hieman yli puolet tapahtui mobiililaitteilla, tietokoneiden osuuden ollessa hieman alle puolet ja tablettien noin 1 %. Suomessa vastaavat osuudet vuonna 2023 olivat noin 40 % mobiililaitteiden ja noin 60 % tietokoneiden osalta, tablettien osuuden ollessa myös Suomessa noin 1 %. Selaimista suosituin oli maailmanlaajuisesti Google Chrome noin 60 % osuudella, Safarin ollessa toiseksi suosituin hieman alle 20 % osuudella, muiden selainten jakaessa jäljelle jäävän osuuden melko tasaisesti. Myös Suomessa Chrome oli selaimista suosituin hieman yli 60 % osuudellaan, jonka jälkeen tulivat Safari noin 15 % ja Firefox noin 10 % osuuksillaan. (Statcounter 2023.)

Tämän statistiikan valossa on selvää, että ei ole riittävää suunnitella verkkosisältöä käytettäväksi ainoastaan jollakin tietyllä laitteella, näyttökoolla tai selaimella. Nykyään pidetään oletuksena, että verkossa oleva sisältö on *responsiivista*, millä tarkoitetaan sen mukautuvan erilaisille päätelaitteille. Responsiivisuuden mahdollistaminen on kuitenkin vaatinut aikaa ja niiden tekniikoiden kehittymistä, joilla responsiivisuus toteutetaan. Vaikka selaimet ovat sallineet HTML-sisällön uudelleenjärjestyvän selainikkunan koon mukaisesti jo verkon alkuajoista lähtien, ennen 2010-lukua yleinen tapa oli kehittää vain tietylle näyttökokoalueelle suunniteltuja, pikseleillä kiinteästi määriteltyjä ulkoasuja. (Eckles 2023.) Ennen 2000-lukua verkkosivuja suurin osa monitoreista oli 800 tai 1024 pikseliä leveitä, joten silloin verkkosivut suunniteltiin tarkasteltavaksi tällä kokoalueella (Social Media Today 2015). 2000-luvun aikana monitorien koot alkoivat laajentua ja älypuhelimet yleistyä, jolloin verkkosivuista alettiin kehittää erillisiä, eri laitteille tarkoitettuja versioita. Tämä onnistui, mutta oli aikaa vievää ja ylläpidon kannalta myös epäkäytännöllistä. (Little 2021.)

Vuonna 2007 Apple julkaisi iPhonen, jota pidetään ensimmäisenä verkon selaamiseen käyttökelpoisena älypuhelimena. Tämän jälkeen ihmisten tapa käyttää ja vuorovaikuttaa verkon kanssa muuttui pysyvästi, kun mobiililaitteiden käyttäminen verkon selaamiseen alkoi yleistymään. (Frain 2022, luku 1.) Viimeistään nyt absoluuttisilla mittayksiköillä eli pikseleillä määritellyistä ulkoasuista muodostui ongelma, koska ne eivät mukautuneet mobiililaitteille. Oli keksittävä, kuinka verkossa oleva sisältö pystyisi paremmin mukautumaan sille laajentuvalle joukolle erilaisia laitteita, joilla sitä oli alettu tarkastelemaan. Vuonna 2010 Ethan Marcotte esitti ajatuksen responsiivisesta suunnittelutavasta helpottamaan näitä haasteita. Marcotten esittämät suunnitteluperiaatteet ovat edelleen käytössä 2020-luvulla. (Eckles 2023; Marcotte 2010 mukaan.) 2020-luvulla mobiililaitteiden käyttäminen internetin selaamiseen onkin jatkanut edelleen kasvuaan, ja toisaalta myös 40-tuumaisten ultralaajakuvanäyttöjen, tablettien ja pelikonsolien käyttäminen internetin selaamiseen on arkipäiväistynyt. Ero pienimpien ja suurimpien verkon selaamiseen käytettävien näyttökokojen välillä vaikuttaa siis edelleen vain kasvavan. (Frain 2022, luku 1.)

Responsiivinen suunnittelu ei ole yksittäinen teknologia, vaan lähestymistapa. Se on termi, jota käytetään kuvaamaan niitä parhaita käytäntöjä ja tekniikoita, joilla voidaan luoda erilaisille päätelaitteille mukautuvia ulkoasuja ja sisältöjä. (Mozilla Developer Network 2023.) Responsiivisen suunnittelun tarkoitus on mahdollistaa verkkosivun mukautuminen erilaisille päätelaitteille, jolloin käyttöliittymästä ei tarvitse kehittää erillisiä, esimerkiksi mobiililaitteille tai tableteille tarkoitettuja versioita. Tämän lisäksi erilaisille päätelaitteille mukautuminen tuo mukanaan myös muita etuja. Monet hakukoneet, kuten esimerkiksi Google, suosivat hakutuloksissaan mobiiliystävällistä ja responsiivista sisältöä (Google 2023). Sen lisäksi, että responsiivisen verkkosivun käyttökokemus on parempi erilaisilla laitteilla, paremman hakukonesijoituksen ansiosta responsiivinen verkkosivu tavoittaa myös laajemman käyttäjäjoukon. Responsiivista verkkosivua on lisäksi myös helpompi lukea ja käyttää, mikä parantaa myös sen saavutettavuutta (Hart-Davis 2023).

Käytännössä responsiivinen websuunnittelu tarkoittaa HTML- ja CSS-tekniikoiden käyttämistä sivuston sisällön muokkaamiseen, piilottamiseen,

kutistamiseen tai suurentamiseen niin, että se mukautuu ja näyttää hyvältä erilaisilla laitteilla (Mozilla Developer Network 2023). Näitä tarkastellaan luvussa 3, joka käsittelee erilaisia responsiivisia tekniikoita. Sitä ennen tarkastellaan vielä keskeisiä responsiiviseen websuunnitteluun liittyviä periaatteita.

2.2 Responsiivisuuden huomioiminen osana websuunnittelua

Verkkosisällön kansainvälisten saavutettavuusohjeiden (WCAG) mukaan responsiivisuus on yksi saavutettavan verkkosisällön onnistumiskriteereistä. WCAG-saavutettavuusohjeiden osio 1.4.10 käsittelee responsiivisuutta. Siinä kuvattuja responsiivisuuteen liittyviä onnistumiskriteerejä voidaan ajatella myös hyvinä suunnitteluperiaatteina, jotka kannattaa huomioida lähes minkä tahansa verkkojulkaisun toteutuksessa. WCAG-saavutettavuusohjeiden osion 1.4.10 mukaan verkkosisältö tulisi voida esittää erilaisissa esitysmuodoissa ilman informaation tai toiminnallisuuden menettämistä esitystavan vaihtuessa. (World Wide Web Consortium 2023.) Tämän lisäksi myöskään sisällön esitystavan ja rakenteen ei tulisi kärsiä eri muodoissa esitettynä. (Minnick 2022, luku 19). Myös sitä, että käyttäjä joutuu suurentamaan tekstiä zoomaamalla tai selaamaan sivua horisontaalitasossa paljastaakseen näytön ulkopuolelle jäävää tekstiä tulisi ehdottomasti pyrkiä välttämään. Tämä hankaloittaa sekä sivun käytettävyyttä että saavutettavuutta, lisäten merkittävästi ihmisen lukemiseen käyttämää vaivannäköä. Sekä pysty- että vaakatasossa tapahtuvaa esitys- ja selaustapaa onkin suositeltavaa käyttää lähinnä vain kartoille, diagrammeille, videopeleille ja esityksille sekä taulukkomuodossa esitettävälle datalle. (World Wide Web Consortium 2023.)

Yleisesti ottaen hyvä suunnitteluperiaate edellä lueteltujen tilanteiden välttämiseksi on järjestää kaikki sisältö käytettävissä olevan selainikkunan leveyden perusteella, jolloin kahdessa suunnassa tapahtuva vierittäminen voidaan välttää. Jotta sivu mukautuisi ja sen esitystapa näyttäisi hyvältä erilaisilla laitteilla, myös sen esitystapaa täytyy useimmiten muuttaa käytettävissä olevan näyttötilan perusteella. Esimerkiksi navigaatiovalikot ovat tyypillinen rakenne, jonka esitystapa vaihtelee riippuen käytettävissä olevasta tilasta.

Navigaatiovalikko kannattaa suunnitella sellaiseksi, että näyttökoon pienentyessä se järjestyy pienemmäksi kokonaisuudeksi ja lopulta avattavaksi yhden painikkeen kautta. Tavoitteena on, että navigaatiovalikko veisi ruudulla aina vähemmän tilaa suhteessa sisältöön. Sisällön osalta työpöytälaitteilla näyttötilaa on käytössä vaakasuunnassa enemmän mobiililaitteisiin verrattuna, jolloin sisältö on mahdollista esittää useammassa sarakkeessa. Tällöin sisällön esitystapa on useimmissa tapauksissa suositeltavaa vaihtaa yhdessä sarakkeessa esitettäväksi mobiililaitteilla esitettäessä. (World Wide Web Consortium 2023.)

Responsiivisen suunnittelun tehtävänä onkin määrittää säännöt sille, miten verkkosivun ja sen sisällön tulee mukautua kulloinkin käytettävissä olevan näyttötilan kokoon ja resoluutioon. Responsiivisessa suunnittelussa tämä tulee huomioida ulkoasun lisäksi myös sisällössä, tarkoittaen että esimerkiksi myös tekstin, kuvien ja median tulee olla responsiivisia. (Interaction Design Foundation 2022.) Responsiivista verkkosivustoa kannattaakin ajatella joustavista ja itsenäisesti mukautuvista komponenteista muodostuvana kokonaisuutena (Shadeed 2023).

2.3 Mobile First -suunnitteluperiaate

Mobile First -suunnitteluperiaate yleistyi 2010-luvulla ja on edelleen vuonna 2023 monella tapaa hyödyllinen lähestymistapa verkkosivujen suunnitteluun. Suunnitteluperiaatteen esitteli ensi kerran Luke Worblewski vuonna 2009. Kirjassaan ”Mobile First” hän esitti, että käyttöliittymän suunnittelemisesta ensisijaisesti tietokoneen monitoreille tulisi luopua ja siirtyä aloittamaan suunnittelu mobiililähtöisesti (Interaction Design Foundation 2016; Worblewski 2009 mukaan.) Tapa suunnitella käyttöliittymä ensin mobiililaitteille sopivaksi ja sen jälkeen jatkokehittää ulkoasua mukauttamalla se suuremmille näytöille sopivaksi on nykyään yleisesti käytetty toimintamalli, koska on vaikeampaa tehdä päinvastoin, eli toteuttaa ensiksi tietokoneen monitorille tarkoitettu käyttöliittymä ja sitten yrittää sovittaa sen elementtejä mahtumaan pienempään tilaan mobiiliversiota varten (Staiano 2022).

Mobiilikäyttöliittymän suunnittelussa keskeisintä on pitää mielessä, että käyttöliittymän tulisi olla helposti navigoitavissa vain peukaloa käyttämällä. Siksi kosketettavien alueiden tulisi olla tarpeeksi suuria, jotta käyttäjän ei tarvitsisi yrittää suurentaa selainikkunaa zoomaamalla esimerkiksi painaakseen linkkiä tai painiketta. (Stimac 2023.) Kosketusnäytöille tarkoitetun käyttöliittymän suunniteltaessa tulisi myös huomioida, että kosketusnäytöllä ei ole mahdollisuutta näyttää CSS-hiiriohjaimia, koska kursoria ei ole. CSS-hiiriohjaimet esimerkiksi linkkien tyylien määrittelyyn kannattaa lisätä vasta kun kosketusnäytölle suunniteltua käyttöliittymää siirrytään mukauttamaan kursoria käyttäville laitteille. (Friedman 2018.)

Hakukoneet, kuten esimerkiksi Google, suosivat responsiivisia ja mobiiliystävällisiä verkkosivuja hakutuloksissaan. Vuodesta 2018 alkaen Google on käyttänyt pääasiallisesti sivun sisällön mobiiliversioita hakutulosten indeksointiin ja sijoittamiseen käyttämällä tähän älypuhelimen käyttöjärjestelmäagenttia. Tätä kutsutaan mobiiliensisijaiseksi indeksoinniksi. (Google 2023.) Google alentaa sivun hakukonesijoitusta merkittävästi, mikäli se ei toimi hyvin mobiililaitteilla (Grant 2022). Kun verkkosivu on suunniteltu toimivaksi ensisijaisesti mobiililaitteilla, se saa paremman hakukonesijoituksen, mikä lisää näkyvyyttä hakutuloksissa ja verkkoliikennettä sivulle. Mobile First -suunnittelun periaatteita noudattamalla voidaan siis helpottaa kehitystyötä, taata parempi käyttökokemus mobiililaitteille ja parantaa verkkosivun näkyvyyttä paremmilla hakukonesijoituksilla. Mobiililähtöinen suunnittelu on hyödyllinen suunnittelutapa niin kehittäjän, käyttäjän kuin paremman hakukonenäkyvyydenkin näkökulmista ajateltuna.

3 Responsiiviset tekniikat

3.1 Erilaiset toteutustavat ja tekniikat

Web-sovelluskehityksessä käytetään tyypillisesti HTML-merkintäkielen, CSS-tyylimäärittelykielen ja JavaScript-ohjelmointikielen yhdistelmää mahdollistamaan interaktiivisen ja visuaalisesti miellyttävän kokonaisuuden toteuttaminen. Näistä JavaScript on dynaaminen ohjelmointikieli, jonka avulla verkkosivun toteutus voidaan laajentaa staattisesta dynaamiseksi toteuttamalla sovellukseen esimerkiksi käyttäjän toimiin reagoivaa tilanhallintaa, tapahtumankäsittelyä ja lisäämällä toteutukseen monimutkaisempaa sovelluslogiikkaa (Mozilla Developer Network 2023). Responsiiviset tekniikat kuitenkin liittyvät pääasiassa HTML:ään ja erityisesti CSS:ään, joten JavaScriptiä ei tässä yhteydessä käsitellä laajemmin.

Responsiivinen verkkosivusto on mahdollista toteuttaa myös esimerkiksi käyttämällä jotakin sisällönhallintaohjelmaa. Sisällönhallintaohjelmat tarjoavat valmiita ulkoasupohja ja toiminnallisuuksia, jolloin sivuston toteuttaminen ja ylläpitämien tapahtuu editoimalla, eikä web-ohjelmointitaitoja välttämättä tarvita (Rampton 2021). Kehitystyötä voidaan helpottaa myös käyttämällä jotakin CSS-*frameworkia*, eli CSS-tyylikehystä. Valmista tyylikehystä käyttämällä verkkosivun ulkoasun toteuttamiseen riittää yksinkertaisimmillaan HTML-rakenteen toteuttaminen oikeita luokkia ja tunnisteita käyttäen (Bose 2023). Maailmanlaajuisesti eniten käytetty tällainen tyylikehys on *Bootstrap*, joka mahdollistaa responsiivisiksi suunniteltujen valmiiden komponenttien, rakenteiden ja CSS-tyylien käyttämisen projektissa (Adarsh 2023). Tämän opinnäytetyön tarkoitus oli kuitenkin perehtyä niihin tekniikoihin, joilla responsiivisuus on näihin valmiisiin ratkaisuihin toteutettu, joten kirjastojen käyttöä ei tässä työssä käsitellä laajemmin.

Responsiivisuuden toteuttamisessa kolmena tärkeimpänä tekniikkana pidetään yleisesti ottaen edelleen mukautuvaa ulkoasua, mediakyselyjä (*media queries*) sekä sisällön mukautuvaa asettelua, kuten Ethan Marcotte jo vuonna 2010 esitti

(Frain 2022; Marcotte 2010 mukaan). Mukautuvalla ulkoasulla tarkoitetaan sitä, että sivuston ulkoasu ja sen elementit mukautuvat ja skaalautuvat eri näyttökokoihin. Mediakyselyt puolestaan ovat yksi responsiivisuutta toteuttava CSS-tekniikka, joka mahdollistaa tyylisääntöjen asettamisen eri näyttökokoalueita varten. Sisällön mukautuvalla asettelulla tarkoitetaan sitä, että responsiivisuus tulee huomioida ulkoasun lisäksi myös osana kaiken sisällön toteutusta. Vaikka responsiivisuuden toteuttaminen perustuu edelleen pitkälti tähän Marcotten näkemykseen, responsiivisuuteen liittyvä CSS on kehittynyt sitten 2010-luvun. Yksi esimerkki tästä ovat vasta vuonna 2020 täyden selaintuen saaneet CSS-funktiot, jotka mahdollistavat aiempaa optimoidumman tavan skaalata erilaisten käyttöliittymäelementtien kokoa (Eckles 2023). Koska tarve responsiiviselle suunnittelulle ei näytä olevan katoamassa, on todennäköistä, että responsiivisuuden mahdollistavat tekniikat jatkavat kehittymistään myös tulevaisuudessa (Shadeed 2023).

3.2 HTML-merkintäkieli ja CSS-tyylimäärittelykieli

HTML (*HyperText Markup Language*) on merkintäkieli, joka koostuu erilaisista elementeistä, jotka kuvaavat sisällön rakennetta. Esimerkiksi otsikoille, kappaleille, linkeille ja kuville on omat HTML-elementtinsä. HTML-elementtien (puhutaan myös HTML-tageista) tehtävänä on määrittää verkkosivun rakenne, elementit ja sisältö, jonka perusteella selain renderöi sivun. (W3Schools 2023.) Koska responsiivisen käyttöliittymän tulee mukautua erikokoisille näytöille sekä vastata myös esimerkiksi mobiililaitteen suuntaamiseen pysty- tai vaaka-asentoon, verkkosivulle tulee määritellä skaalautuvuusasetukset. (Minnick 2022 luku 19.) Tämä tapahtuu HTML:n *"meta viewport"* -tagin avulla. Kun *viewportin*, eli käytössä olevan näyttötilan (= selainikkuna) leveydeksi määritetään laitteen leveys, selain käyttää tätä tietoa sivun renderöimiseen. Näin voidaan varmistaa sivun skaalautuvan aina kulloinkin käytössä olevan päätelaitteen resoluution mukaisesti. Tämä on tärkeä elementti, joka tulee sisällyttää osaksi minkä tahansa responsiivisen verkkosivun toteutusta. (LePage & Andrew 2019.)

CSS-tyylimäärittelykieltä (*Cascading Style Sheets*) puolestaan käytetään HTML-elementtien esitystavan, asettelun ja tyylien muokkaamiseen (Mozilla Developer Network 2023). CSS-tekniikat mahdollistavat erilaisten käyttöliittymäelementtien visuaalisen esitystavan hallinnan mahdollistamalla elementtien asettelun, näkyvyyden ja koon, muotojen ja värien määrittämisen (Hart-Davis 2023). CSS tukee suhteellisia mittayksikköjä, joita voidaan käyttää absoluuttisten mittayksiköiden sijaan mahdollistamaan elementtien koon mukautuminen suhteutettuna kulloinkin käytettävissä olevaan tilaan. Suhteellisia mittayksikköjä voidaan hyödyntää myös elementtien joustavassa asettelussa, jolloin selain uudelleen järjestää elementit niille määriteltyjen suhteellisten osuuksien perusteella. (Hart-Davis 2023, luku 10.) Nykyään yleistyvä tapa on kirjoittaa responsiivista CSS:ää komponenteille ja antaa selaimen renderöidä komponentti sille määriteltyjen tyyliasetusten perusteella (Shadeed 2023).

3.3 CSS-mittayksiköt ja funktiot

Jotta verkkosivun elementit voidaan esittää toivotulla tavalla, niille täytyy useimmiten määrittää koko- ja asettelusäännöt. CSS:ssä elementtien koon määrittäminen tapahtuu useimmiten määrittelemällä niille leveys ja/tai korkeus. Koon määrittämiseen voidaan käyttää joko absoluuttisia tai suhteellisia mittayksiköitä. Responsiivisessa suunnittelussa tulisi käyttää absoluuttisten mittayksiköiden sijaan pääasiallisesti suhteellisia mittayksiköitä, sillä ainoastaan absoluuttisia arvoja käyttämällä toteutuksesta ei tule responsiivinen. (Hart-Davis 2023, luku 10.)

Pikselit (px) ovat webkehityksessä yleisimmin käytetty absoluuttinen mittayksikkö. Absoluuttisina mittayksiköinä on mahdollista käyttää myös tuumia (in) ja senttimetrejä (cm), mutta yleisesti ottaen ne soveltuvat paremmin fyysisen maailman tarpeisiin. Suhteellisista mittayksiköistä prosenttiosuudella (%) tarkoitetaan responsiivisen suunnittelun yhteydessä prosentuaalista osuutta siitä tilasta, jonka sisälle elementti on sijoitettuna. Selainikkunan (*viewport*) kokoon perustuvat suhteelliset mittayksiköt *viewport width* (vw) ja *viewport height* (vh) puolestaan kuvaavat yhtä sadasosaa selainikkunan leveydestä tai korkeudesta.

Juurielementtiin (*root element*) perustuva suhteellinen mittayksikkö rem taas perustuu juurielementin kokoon, joka on yleensä HTML-elementti. Esimerkiksi jos juurielementin koko on 16 px, 1 rem = 16 px ja 2 rem = 32 px. (Hart-Davis 2023, luku 9.)

Kuvilla on omat absoluuttiset mittasuhteensa, jotka tuovat mukanaan omat haasteensa. Niiden lisääminen verkkosivulle aiheuttaa ongelmia erityisesti kuvan ollessa leveämpi kuin se elementti, jonka sisälle se sijoitetaan. Yleinen tapa sovittaa kuva elementin sisälle on hyödyntää suhteellisia mittayksiköitä määrittämällä sen maksimileveydeksi 100 % käyttämällä *max-width* CSS-määrittelyä. Näin kuva kutistuu sopivaksi siihen tilaan, johon se on sijoitettu, mikäli kuva on tilaa suurempi ilman kuvan alkuperäisten mittasuhteiden vääristymistä. Silti kuville olisi suotavaa määrittää maksimileveyden lisäksi myös leveys ja korkeus, koska selain käyttää näitä tietoja varatessaan kuvalle tilan ennen sivun lataamista. Näin voidaan estää tilanne, jossa ulkoasun tai sisällön järjestys rikkoutuu sivun lataamisen aikana. (LePage & Andrew 2019.)

Viime vuosina responsiivisen suunnittelun haasteisiin vastaamaan on kehitetty CSS-funktioita, kuten min(), max(), calc() ja clamp(). Näitä funktioita voidaan hyödyntää erityisesti visuaalisten elementtien mittasuhteiden määrittämiseen. Esimerkiksi kolmea argumenttia käyttävä clamp(min, preferred, max) sopii responsiivisen suunnittelun tarpeisiin, sillä sen avulla voidaan määritellä elementeille minimi- ja maksimikoko ja ohjeistaa selainta skaalaamaan elementin koko näiden välille esimerkiksi selainikkunan levyteen perustuen (Frain 2022). CSS-funktioiden etuna on se, että niiden avulla selainta voidaan ohjeistaa skaalaaman elementtien kokoa portaattomasti vain yhden lyhyen määrittelyn avulla. Samaan tarkoitukseen voidaan käyttää myös seuraavassa luvussa käsiteltäviä mediakyselyitä, mutta käyttämällä ainoastaan mediakyselyitä skaalautuminen ei ole portaatonta ja koodin määrä sekä ylläpitotarpeet kasvavat nopeasti. (Eckles, S. 2023.)

3.4 Mediakyselyt

Mediakyselyt ovat CSS-tekniikka, jonka avulla voidaan selvittää, millaisella laitteella ja resoluutiolla verkkosivua esitetään (Hart-Davis 2023, luku 10). Samalla mediakysely on mekanismi, joka sallii ehdollisen logiikan käyttämisen CSS-tyyleille, eli tyylien muuttamisen sen mukaisesti, milloin jokin ehto täyttyy (Frain 2022, luku 3). Mediakyselyjä käytetään sisällön esitystavan mukauttamiseen tyypillisesti joko näytön leveyden tai resoluution perusteella, tehden niistä oleellisen osan responsiivista CSS:ää. Mediakysely implementoi uuden CSS-tyylimäärittelyn tai tyylimäärittelyiden joukon *breakpointien* mukaisesti. Termiä breakpoint käytetään niistä "pisteistä", joissa ulkoasun esitystavan halutaan vaihtuvan. (Mozilla Developer Network 2023.)

Tietokoneiden monitorien suosituimmat resoluutiot maailmanlaajuisesti lokakuussa 2023 olivat 1920 x 1080 px noin 22 % osuudella, 1366 x 768 px noin 14 % osuudella ja 1536 x 864 px noin 10 % osuudella. Mobiililaitteiden suosituimmat resoluutiot maailmanlaajuisesti lokakuussa 2023 olivat 360 x 800 px noin 11 % osuudella, 390 x 844 px noin 8 % osuudella ja 414 x 896 px noin 6 % osuudella. (Statcounter 2023.) Tästäkin statistiikasta käy ilmi, että sironta erilaisten päätelaitteiden välillä on suurta, ja siksi breakpointien määrittämiseen ei ole olemassa yleispäteviä sääntöjä. Nykykäsitys vaikuttaa olevan, että niitä tulisi käyttää mahdollisimman vähän ja vain tarvittaessa. Syy tähän on se, että breakpointien määrän lisääntyessä syntyy ikään kuin uusia, eri näyttökokoalueille tarkoitettuja ulkoasuversioita, joiden määrän lisääntyessä niille yksilöllisten tyylien ylläpitäminen muodostuu haastavaksi (Staiano 2022.)

Breakpointeja täytyy kuitenkin käyttää, ja niitä tulisi olla ainakin yksi, sillä responsiivisella sivustolla tulisi olla erilainen ulkoasu ainakin mobiili- ja työpöytälaitteille (Minnick 2022). Breakpointien lisäämiseen websovellukseen voidaan soveltaa Mobile First -suunnittelun periaatteita, eli aloittaa lisäämällä ensimmäinen breakpoint siihen pisteeseen, jossa käyttöliittymän halutaan mukautuvan mobiililaitteita suuremmille näytöille (LePage & Andrew 2019). Mobile First -suunnittelutapaa käytettäessä on suositeltavaa kehittää ensin yleensä yhdessä sarakkeessa esitetty ulkoasu mobiililaitteita varten. Tämän

jälkeen voidaan lisätä breakpoint ja mediakysely siihen pisteeseen, jossa ulkoasun esitystavan halutaan vaihtuvan. Mediakyselyn sisälle lisätään sitten tyylimäärittelyt työpöytälaitteille tarkoitetulle ulkoasulle. Näin mediakyselyn sisälle määritetyt tyylit implementoidaan vain, mikäli näyttökoon havaitaan olevan kyseiselle esitystavalle riittävä (Mozilla Developer Network 2023).

3.5 Joustava asettelu

Verkkosivuston ulkoasun toteutuksessa voidaan käyttää monenlaisia tekniikoita, mutta parhaiten tähän sopivat *CSS Grid* ja *CSS Flexbox*, jotka oikein käytettyinä yksinkertaistavat verkkosivuston ulkoasun kehittämistä monin tavoin. Näistä Gridiä on suositeltavaa käyttää ulkoasun määrittämiseen, kun taas Flexbox sopii paremmin sisällön järjestämiseen. (Frain 2022.) Keskeisin ero Flexboxin ja Gridin välillä on se, että Gridin sisältö voidaan järjestää samanaikaisesti kahdessa suunnassa (Hartl & Donahoe 2022). Flexboxissa tätä mahdollisuutta ei ole, vaan kaikki flexboxiin kuuluvat elementit sijaitsevat ikään kuin samalla rivillä, vaikka flexboxin sisältämät elementit voidaankin saada jakautumaan eri riveille komennon *flex-wrap* avulla. Flexbox soveltuu parhaiten käytettäväksi tilanteisiin, joissa elementtien, esimerkiksi kuvien, halutaan järjestyvän itsensä automaattisesti kulloinkin käytettävissä olevan tilan sisällä. (Frain 2022.)

Grid puolestaan soveltuu paremmin ulkoasun jakamiseen osiin, esimerkiksi ylä- ja alatunnisteisiin, navigaatiovalikkoon ja sisältöön. Gridiä käytettäessä selaimelle on kerrottava kuinka moneen riviin ja sarakkeeseen Grid jakautuu, ja minkä kokoisia näistä koostuvat osa-alueet ovat. (Frain 2022). Jotta Grid olisi skaalautuva, sen osa-alueiden kokojen määrittämiseen on suositeltavaa käyttää fraktioita (fr). Prosenttiyksiköiden tapaan fraktioiden avulla käytettävissä oleva tila voidaan jakaa eri osien välille suhteellisina osuuksina. (Mozilla Developer Network). Määrittelyjä voidaan mediakyselyiden avulla antaa myös niille tilanteille, joissa Gridin koko muuttuu tai siihen halutaan lisätä uusia elementtejä. Gridin sisältämien rivien ja sarakkeiden sisällä voidaan käyttää myös muita esitystapoja, esimerkiksi Flexbox voidaan sijoittaa Gridin sisään. (Frain 2022.)

4 Opinnäytetyön toteutustapa

4.1 Toiminnallinen opinnäytetyö

Tämä opinnäytetyö on toteutettu toiminnallisena opinnäytetyönä. Toiminnallinen opinnäytetyö koostuu opinnäytetyön kirjallisesta osuudesta eli opinnäytetyöraportista sekä opinnäytetyön tuotoksesta, joka voi olla esimerkiksi kirja, käyttöopas tai verkkojulkaisu. Toiminnallisessa opinnäytetyössä opiskelijan tulisi pystyä osoittamaan alansa asiantuntemusta yhdistämällä alan teoreettinen tietoperusta ammatilliseen osaamiseensa. (Vilkka & Airaksinen 2003.) Tämän opinnäytetyön aiheeksi valitsin responsiivisen suunnittelun ja responsiiviset tekniikat web-sovelluskehityksessä, koska halusin syventää web-käyttöliittymäsuunnitteluun liittyvää osaamistani oppimalla suunnittelemaan ja toteuttamaan responsiivista verkkosisältöä itse, turvautumatta valmiisiin ratkaisuihin. Koska tämä vaatii responsiivisuuden mahdollistavien sovelluskehitystekniikoiden tuntemista ja käytännön web-ohjelmointitaitoja, halusin syventää aiheeseen liittyvää osaamistani teoreettiseen tietoperustaan perehtymisen lisäksi myös käytännössä. Näistä syistä tuntui luontevalta toteuttaa opinnäytetyö toiminnallisen opinnäytetyön muodossa.

Aloitin opinnäytetyöprosessini perehtymällä edellisissä luvuissa käsiteltyyn aihealueeseen liittyvään teoreettiseen tietoperustaan. Tämän jälkeen toteutin opinnäytetyön tuotoksen hyödyntämällä toteutuksessa hankkimaani tietoperustaa. Raportoidessani omaa työskentelyäni olen pyrkinyt osoittamaan, millä tavoin olen työskentelyssäni hyödyntänyt hankkimaani tietoperustaa, kuvaamaan työskentelyn etenemisen sekä perustelemaan sen aikana tekemiäni ratkaisuja. Opinnäytetyöraportin pohdintaosuudessa olen lisäksi pyrkinyt arvioimaan opinnäytetyön onnistumista suhteessa sille asetettuihin tavoitteisiin sekä opinnäytetyöprosessin aikana kertyneen ammatillisen osaamisen kehittymistä.

4.2 Tuotos

Valitsin toteuttaa opinnäytetyöprosessin aikana toteutettavan verkkosivuston omaan käyttööni toteutettavan web-portfolion muotoon. Web-portfolio valikoitui tuotokseksi osittain siksi, koska en löytänyt opinnäytetyölleni sopivaa toimeksiantajaa ja osittain siksi, koska näin opinnäytetyöprosessin mahdollisuutena toteuttaa itselleni verkkoon esittelykanava omalle osaamiselleni. Hyvin toteutettu web-portfolio on yksi mahdollinen tapa edistää omaa työllistymistä IT-alan tehtäviin, joten sellaisen toteuttaminen tuntui oppimiskokemuksen lisäksi hyödylliseltä ajatukselta.

Web-portfoliota varten minulla oli myös jo olemassa olevaa sisältöä: tekemiäni käyttöliittymäsuunnitteluun liittyviä töitä ja digitaalista taidetta, jonka tekeminen on pitkäaikainen harrastukseni. Näiden lisäksi suunnittelin toteuttavani portfolioon osion, jossa esittäydyn ja kerron omasta osaamisestani. Responsiivisuuden toteutuminen ja toteutuksen mukautuminen saumattomasti eri laitetyypeille oli tärkein asia, jota toteutuksessa tavoittelin, ja sen ohella käytettävyyden huomioiminen mahdollistamaan miellyttävä käyttökokemus. Web-portfolio tuntui siis myös riittävän laajalta kokonaisuudelta, jotta sen kehittämisen aikana olisi mahdollista harjoitella riittävän monipuolisesti responsiivisuuteen liittyvien suunnitteluperiaatteiden ja tekniikoiden soveltamista käytäntöön. Ajatus siitä, että opinnäytetyön tuotos tulisi itselleni käyttöön tuntui myös motivoivalta lähtökohdalta aloittaa työskentely, toteuttaa työ huolellisesti ja saada aikaan toimiva ja viimeistelty kokonaisuus.

4.3 Valitut menetelmät ja työkalut

Toteutin web-portfolion ulkoasusuunnitelman suunnittelutyökalu Figmalla, minkä jälkeen toteutin sivuston single-page web-applikaationa Mobile First -suunnitteluperiaatteita noutaen. Ohjelmointikielinä käytin seuraavia: JavaScript/React.js, HTML5 ja CSS. Kehitystyö tapahtui käyttämällä Visual Studio Codea ohjelmointiympäristönä, Google Chromea selaimena ja sen

kehittäjätyökaluja sovelluksen testaamiseen. Kehitystyön aikana käytin kehittäjätyökaluja pääasiassa testatakseni sivuston käyttäytymistä ja mukautumista erikokoisille näytöille ja resoluutioille.

Reaalimaailman tarpeisiin nähden valmiita komponentteja tarjoavien CSS-tyylikehysten tai jonkin sisällönhallintaohjelman käyttäminen voisi olla valitsemaani toteutustapaan verrattuna monessa tapauksessa kannattavampi vaihtoehto. On selvää, että näiden toteutusratkaisujen tarjoamat selainyhteensopivat ja uudelleenkäytettävät valmiit ulkoasupohjat ja komponentit nopeuttavat ja yksinkertaistavat verkkosivustojen ja kehittämistä ja helpottavat niiden ylläpitoa. Niiden käyttäminen onkin monessa tapauksessa perusteltua. Tämän opinnäytetyöprosessin tavoite oli kuitenkin toimia oppimiskokemuksena, jonka aikana opin suunnittelemaan ja toteuttamaan responsiivista verkkosisältöä alusta alkaen, joten nämä vaihtoehdot eivät olisi mielestäni tukeneet tämän opinnäytetyön tarkoitusta ja tavoitteita. Toinen syy tekemääni valintaan oli myös mahdollisuus harjoitella webohjelmoinnin ydintekniikoiden eli HTML:n, CSS:n sekä JavaScript-kirjasto Reactin käyttämistä jatkaakseni kehittymistäni web-sovelluskehittäjänä.

Päätin käyttää web-portfolion toteutukseen JavaScript-kirjasto Reactia, vaikka senkään käyttämien projektissa ei olisi ollut välttämätöntä. Tämäntyyppinen sivusto olisi mahdollista rakentaa myös ilman Reactia käyttämällä pelkkää JavaScriptiä käyttöliittymäkomponenttien tilanhallinnan ja tapahtumakuuntelijoiden toteuttamiseen, joihin sitä tässä projektissa pääasiassa käytin. Reactin komponenttipohjaisen ohjelmointitavan käyttäminen web-sovelluskehityksessä on kuitenkin moderni web-sovelluskehityksen muoto, ja olin aloittanut Reactiin tutustumisen aiemmissa web-ohjelmointiin liittyvistä opinnoistani. Koska projektissa oli mahdollisuus päästä soveltamaan myös tätä osaamista, päätin ottaa Reactin osaksi projektin toteutustapaa.

5 Opinnäytetyön toteuttaminen

5.1 Aikataulu

Olin suunnitellut toteuttavani opinnäytetyöni muiden opintojen ohella syksyn 2023 ja tarvittaessa tammikuun 2024 aikana. Syyskuun 2023 loppupuolella aihe-ehdotuksen hyväksymisen jälkeen aloitin työskentelyn perehtymällä opinnäytetyön aihealueeseen ja aloittamalla tiedonhaun lähdeaineiston keräämistä varten. Samalla aloitin tekemään lähdemateriaaleissa käsiteltyihin responsiivisiin tekniikoihin liittyviä alustavia kokeiluja käytännössä. Lokakuussa opinnäytetyöprosessi ei edistynyt, mutta palasin jatkamaan työskentelyä taas marraskuussa, jolloin lähdeaineistoon perehtymisen lisäksi aloitin myös suunnittelemaan ja toteuttamaan opinnäytetyön tuotosta. Joulukuun alussa olin mielestäni perehtynyt riittävästi lähdeaineistoon, ja laadin kirjallisuuskatsauksen keräämäni aineiston pohjalta. Opinnäytetyön suunnitelmavaihe valmistui joulukuussa 2023, jolloin myös tuotoksen eli web-portfolion toteuttaminen oli jo hyvässä vaiheessa. Vuodenvaihteen ja tammikuun 2024 aikana viimeistelin tuotoksen ja työstin opinnäytetyön raporttiosuuden valmiiksi. Seuraaviin lukuihin 5.1–5.3 olen raportoinut opinnäytetyön tuotoksen työstämisen suunnitteluasteelta valmiiksi websovellukseksi.

5.2 Responsiivisen sivuston suunnitteleminen

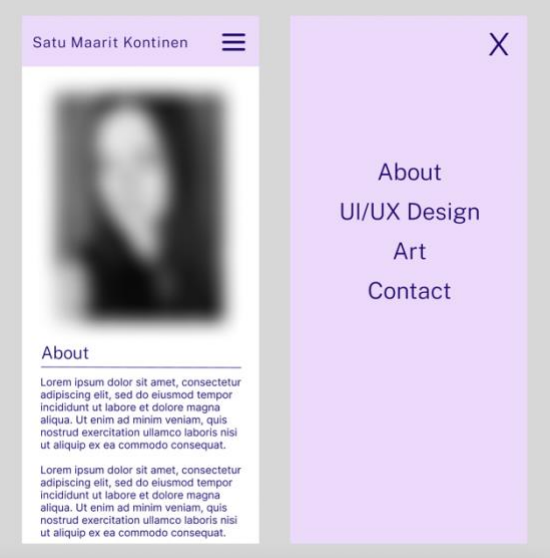
Sivuston toteutuksessa tärkeintä oli responsiivisuuden toteutuminen. Sivusto tuli suunnitella sellaiseksi, että se mukautuisi kaikille laitteille, joilla on mahdollista käyttää selainta ja internetiä. Siksi asetin toteutukselle responsiivisuuteen liittyvät vaatimuskriteerit, joiden halusin valmiissa toteutuksessa toteutuvan. Vaatimuskriteerit on esitetty oheisessa taulukossa (taulukko 1).

Vaatimus 1	Yhdessä sarakkeessa esitetty ulkoasu mobiililaitteille
Vaatimus 2	Kahdessa sarakkeessa esitetty ulkoasu mobiililaitteita suuremmille päätelaitteille
Vaatimus 3	Sisällön tulee keskittyä laajakuvanäytöillä tarkasteltuna
Vaatimus 4	Sisällön tulee järjestyä aina niin, että 2-suuntaista vierittämistä ei tarvita
Vaatimus 5	Fonttikokojen tulee mukautua niin, että teksti on helppolukuista kaikilla laitetyypeillä
Vaatimus 6	Mobiiliversion tulee olla navigoitavissa peukalolla

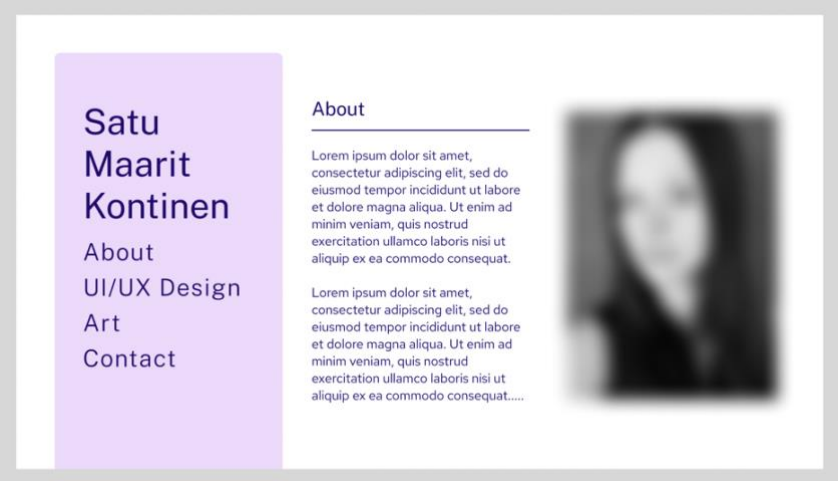
Taulukko 1. Web-portfolion vaatimusmäärittelyt.

Helpoin tapa toteuttaa tämäntyyppinen sivusto olisi toteuttaa se yhdessä sarakkeessa esitettynä, jolloin tarvetta ulkoasun esitystavan vaihtamiselle ei syntyisi välttämättä lainkaan. Halusin kuitenkin toteuttaa kaksi erilaista esitystapaa eli 1-sarakkeisen ulkoasun mobiililaitteille (vaatimus 1) ja 2-sarakkeiseen ulkoasun suuremmille näytöille (vaatimus 2). Päädyin tähän päätökseen oppimiskokemuksen vuoksi, koska halusin oppia, miten toisistaan huomattavasti poikkeavia esitysmuotoja voidaan sisällyttää osaksi yhden websovelluksen toteutusta. Myöskään koska koko näyttötilan käyttäminen laajakuvanäytöillä ei omasta kokemuksestani useimmissa tapauksissa ole tarkoituksenmukaista, halusin toteuttaa sisällön keskittämisen laajakuvanäytöillä (vaatimus 3). Toteutukselle asettamissani vaatimuksissa halusin lisäksi huomioida responsiivisuuteen liittyviä yleisiä hyvän käyttöliittymäsuunnittelun periaatteita, sekä WCAG-saavutettavuusohjeiden responsiivisuuteen liittyvät suositukset (vaatimukset 4–6).

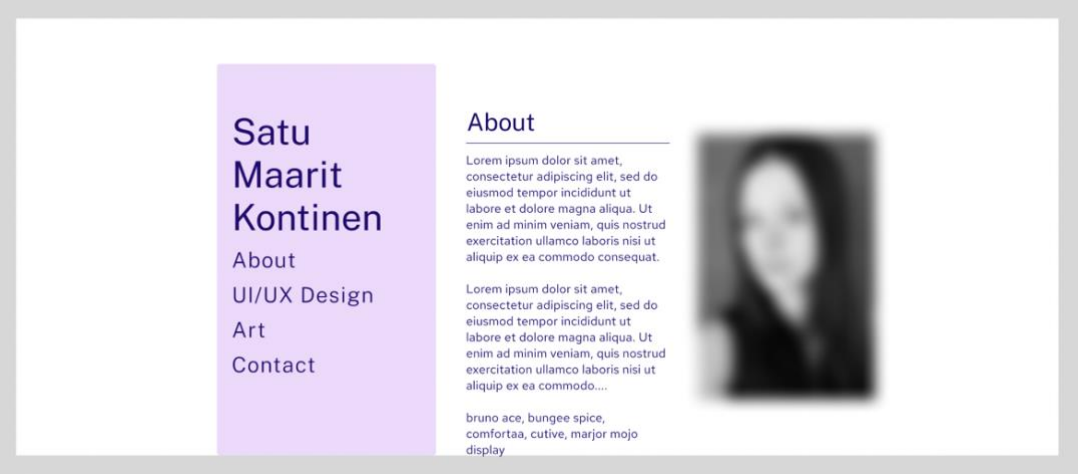
Koska kehitettävän sivuston oli tarkoitus toimia oman osaamiseni esittelykanavana, ulkoasun graafinen ilme oli myös yksi tekijä, jonka halusin huomioida ulkoasusuunnitelmassani, jossa tavoittelin informatiivisuutta, selkeyttä ja miellyttävää visuaalista ilmettä osana hyvän käyttökokemuksen mahdollistamista. Aloitin seuraamaan Mobile First -suunnittelutapaa jo suunnitteluvaiheessa aloittamalla suunnittelun mobiililaitteille tarkoitetusta ulkoasusta, minkä jälkeen laajensin suunnitelmaa suuremmille näytöille sopivaan muotoon. Tekemäni ulkoasusuunnitelmat vastaavat oikeiden päätelaitteiden kuvasuhteita, mutta ovat muutoin suuntaa-antavia. Ulkoasusuunnitelmien tarkoitus oli kuvata mobiili- ja työpöytäversion esitys- ja navigointitavat pääpiirteissään - en siis tavoitellut sitä, että valmis toteutus vastaisi suunnitelmia tarkalleen. Ulkoasusuunnitelmat on esitetty oheisissa kuvissa (kuvat 1–3).



Kuva 1. Ulkoasun esitystapa mobiililaitteilla. Suunnitelma.



Kuva 2. Ulkoasun esitystapa työpöytälaitteella. Suunnitelma.



Kuva 3. Ulkoasun esitystapa laajakuvanäytöllä. Suunnitelma.

5.3 Responsiivisen sivuston toteuttaminen

Koska olin aloittamassa kehittämään React-pohjaista web-applikaatiota, oli projektin luomisen jälkeen mietittävä hieman sovelluksen rakennetta, jotta sivuston ylläpitäminen olisi helpompaa. Jaoin sovelluksen rakenteen siihen kuuluviin sivuihin ja sivuilla esitettäviin komponentteihin. Kehitystyön aikana laajensinkin käsitystäni siitä, mitä uudelleenkäytettäviin komponentteihin perustuva tapa rakentaa websovelluksia käytännössä tarkoittaa. Seuraava työvaiheeni oli luoda tyhjät ”raakileet” sivustoon kuuluvista sivuista, jotta pystyin toteuttamaan sovellukseen reitityksen React Routerin avulla ja varmistumaan

reitityksen toiminnasta. Kun nämä alkutoimet oli tehty, pohdin vielä CSS-tyylien sijoittelua sovelluksessa ennen toteutuksen jatkamista.

Työskentelyä aloittaessani tiesin, että CSS-tyylit on mahdollista sijoittaa joko ulkoisiin CSS-tyylitiedostoihin tai niitä voi kirjoittaa suoraan React-komponenttien koodiin, josta käytetään englanninkielistä termiä "*inline styling".* Päätin ottaa selvää siitä, kumpi tapa on suositeltava - ja löysin mielipiteitä sekä puolesta että vastaan. Esimerkiksi Chris Minnick suosittelee vuonna 2022 julkaistussa React-sovelluskehitystä käsittelevässä kirjassaan käyttämään ulkoisia tyylitiedostoja myös uudelleenkäytettäville komponenteille. Minnickin mielestä ulkoisten tyylitiedostojen käyttäminen on sovelluksen ylläpidon kannalta selkeämpää, tukee hyvän käyttöliittymäsuunnittelun periaatteita sekä vähentää kirjoitettavan koodin määrää. Carlos Roldán puolestaan suosittelee CSS:n kirjoittamista suoraan React-komponenttien koodiin vuonna 2023 ilmestyneessä React-sovelluskehitystä käsittelevässä kirjassaan. Roldánin mielestä ulkoisten tyylitiedostojen ylläpidettävyys on haastavaa erityisesti suurissa projekteissa, joissa niiden määrä voi kasvaa todella suureksi. Mahdollisesti useiden eri komponenttien tyylejä sisältävistä tyylitiedostosta voi myös olla haastavaa etsiä, mikä tyyli kuuluu millekin elementille. CSS-tyylit ovat myös ylikirjoitettavissa, mikä voi erityisesti suuremmissa projekteissa aiheuttaa ongelmia ylläpidettävyyteen. Omassa toteutuksessani päädyin kuitenkin käyttämään tyylien määrittelyyn ulkoisia CSS-tyylitiedostoja, koska toteuttamani sovellus on yhden henkilön toteuttama pieni projekti, johon arvelin ulkoisten CSS-tyylitiedostojen käyttämisen sopivan paremmin.

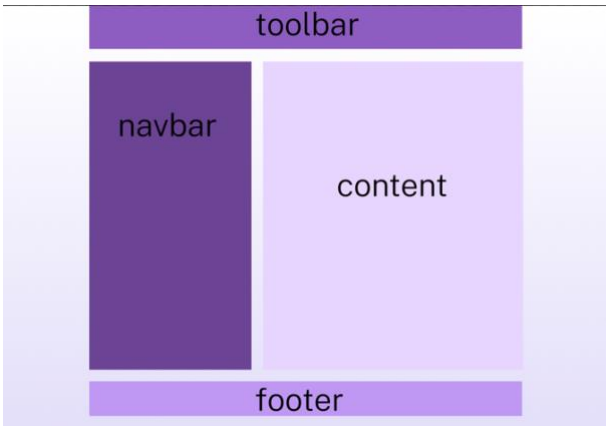
Mobile First -suunnittelun periaatteiden mukaisesti minun oli määrä aloittaa ulkoasun toteuttaminen mobiiliversiosta. Ulkoasun esitystavan vaihtamisen toteuttaminen sovellukseen kuitenkin mietitytti, koska mobiililaitteilla navigaatiopalkki tulisi renderöidä sisällön yläpuolelle ja muutoin sisällön vasemmalle puolelle. Ennen kuin jatkoin työskentelyä päätin pohtia ensin, millä tavalla esitysmuodon vaihtamisen voisi toteuttaa. Aluksi ajattelin, että tarvitsen toteutukseen Reactin tilanhallintaa, eli että käytössä olevan näyttötilan leveyttä seuraavan muuttujan ja tapahtumakuuntelijan avulla renderöitäisiin joko mobiiliversion tai selainversion navigaatiovalikko. Testasin tätä ja havaitsin

menetelmän toimivaksi, mutta Mozilla Developer Networkin CSS Gridiä käsittelevä opas neuvoi yksinkertaisemman tavan. Kun Gridin sisältämään ruudukkoon kuuluvat osa-alueet nimetään, niiden näkyvyyttä voidaan kontrolloida helposti mediakyselyn ja CSS:n *display*-ominaisuuden avulla. Seuraava esimerkki tekniikasta (kuva 4) on sovellukseni CSS-tyylitiedostosta.

```
/* yläpalkki piilotetaan, jos käytössä ei ole mobiililaite */
@media (min-width: 601px) {
  .toolbar {
    display: none;
  }
}
```

Kuva 4. Gridiin kuuluva osio "toolbar" piilotetaan jos mediakysely havaitsee näyttökoon olevan suurempi kuin 601px.

Päätin toteuttaa ulkoasun CSS Gridillä, joka jälkeenpäin katsottuna oli oikea päätös. Seuraava havainnekuva (kuva 5) esittää sovellukseen Gridillä määrittelemääni ruudukkoa, jota tarvitsin ulkoasun jakamiseen eri osioihin.



Kuva 5. CSS Gridin rakenne sovelluksessa ja osiot, joihin se on jaettu. Havainnekuva.

Tapoja määritellä Grid ja siihen kuuluvat osiot on useita erilaisia, mutta kaikille yhteistä on tarve määrittää ruudukkoon kuuluvien sarakkeiden ja rivien määrä ja nimetä Gridiin kuuluvat osiot. Tapoja määritellä kunkin osion ruudukossa käyttämä tila on sen sijaan useita, ja myös osioiden koon määrittelyssä voidaan käyttää erilaisia mittayksiköitä. Gridin toimintatapa oli aluksi vaikeaa hahmottaa ja erilaisten määrittelytapojen kirjo hämmensi, mutta internetistä löytyi useita

asiaan liittyviä oppaita. Toteutin kuvassa 5 visualisoidun ruudukon kehittämääni sovellukseen alla olevassa kuvassa (kuva 6) esitetyillä CSS-määrittelyillä.

```css
/* Grid container */
.container {
  display: grid;
  grid-template-areas:
    "toolbar toolbar"
    "sidebar content"
    "footer footer";
  grid-template-columns: 1.1fr 2fr;
  grid-template-rows: 0 1fr 0fr;
  column-gap: 1.5%;
  margin-left: clamp(1%, 5%, 6%);
  margin-right: clamp(1%, 5%, 6%);
}
```

Kuva 6. Kuvassa 5 visualisoidun ruudukon toteuttaminen sovellukseen, tekniikkana CSS Grid.

Oltuani tyytyväinen ulkoasun mobiili- ja työpöytäversioihin ja ulkoasun käyttäytymiseen selainikkunaa skaalattaessa, aloin lisäämään sisältöä sivustolle. Kehitystyö eteni tämän jälkeen siten, että kehitin jokaisen sivustoon kuuluvan sivun yksi kerrallaan aloittamalla Mobile First -suunnitteluperiaatteiden mukaisesti sivun mobiiliversion kehittämisestä. Kun olin tyytyväinen mobiiliversion toteutukseen, seuraavassa vaiheessa mukautin sivun mediakyselyä käyttämällä työpöytälaitteelle sopivaksi, ja viimeisessä vaiheessa tarkistin kuinka sivu mukautuu laajakuvanäytöille ja tarvittaessa laajensin sivun tyylimäärittelyjä toisen mediakyselyn avulla. Toistin tämän prosessin jokaiselle sivustoon kuuluvalle sivulle. Jos harhauduin poikkeamaan tästä työskentelytavasta, huomasin että työskentely muuttui nopeasti sekavaksi, kun korjattavaa löytyi milloin mistäkin versiosta.

Osana sivuston responsiivisuuden toteutusta perehdyin myös sisällön, eli pääasiassa tekstin ja kuvien responsiivisuuden toteuttamiseen. Käyttöjärjestelmiin valmiiksi asennettujen fonttien määrä on suppea, ja niiden esitystapa voi olla myös keskenään hieman erilainen eri laitteilla. Esimerkiksi omalla älypuhelimeni tällainen fontti näytti aivan erilaiselta, kuin miltä se oli näyttänyt kehitystyön aikana kehittäjätyökaluilla simuloituna. Päätinkin siirtyä käyttämään Google Fonts -palvelun kautta ladattavia ilmaisfontteja. Fonttien

lataaminen palvelusta hidastaa hieman sivun latausnopeutta, mutta fonttien esitystapa on sama kaikilla laitteilla. Mikäli fonttia ei voida jostakin syystä ladata, selainta voidaan fontille kuuluvassa CSS-määrittelyssä ohjeistaa käyttämään jotakin laitteen käyttöjärjestelmään valmiiksi asennettua fonttia.

Fontit on mahdollista toteuttaa selainikkunan koon mukaan automaattisesti skaalautuvina käyttämällä suhteellisia mittayksiköitä, tai yhdistämällä ne CSS-funktioon. Sovelluksessani käytin tähän funktioita clamp() ja calc(). Alla olevissa koodiesimerkissä näkyy esimerkki skaalautuvan fontin toteutuksesta clamp() -funktion avulla (kuva 7). Selain saa funktion kautta ohjeen skaalata fonttikoko pikseleillä määriteltyjen minimi- ja maksimikokojen välille näytön leveyteen (vw) perustuen. Samassa määrittelyssä näkyy myös, kuinka fonttina käytetään ensisijaisetsi Google Fonts -palvelusta ladattua fonttia ”Red Hat Text” ja toissijaisesti kaikille käyttöjärjestelmille yhteisen sans-serif-fonttiperheen fonttia.

```css
/* Leipätekstin fontti */
.content {
  font-size: clamp(18px, 1.2vw, 34px);
  font-family: 'Red Hat Text', sans-serif;
  color: #1E0A6B;
}
```

Kuva 7. Fontin skaalautuvuuden toteuttava CSS-funktio.

Kuvien osalta jouduin perehtymään myös niiden koon ja asettelutavan toteuttamiseen responsiivisesti. Kuvat sain skaalautumaan haluamallani tavalla pitkälti pelkän *max-width* CSS-määrittelyn avulla, jolloin kuvien leveys ei voi kasvaa niiden esittämiselle varatun elementin leveyttä suuremmaksi. Kuvien asettelu selainikkunan leveyden perusteella sen sijaan oli haastavampaa, ja sitä varten minun täytyi perehtyä CSS Flexboxin toimintaan. Flexboxin avulla sain kuvat järjestymään haluamallani tavalla eri esitysmuodoissa. tulee antaa kuvat sisältävälle elementille. Kuvien lisäksi toteutin sivuille myös selattavan kuvagallerian, jonka toteutus on ainoa sivustoon toteuttamani toiminnallisuus, jonka toteutuksessa hyödynsin valmista Rect-komponenttia (react-image-gallery).

5.4 Työn viimeistely ja testaaminen

Edellisessä luvussa kuvaamillani tavoilla sain rakennettua sekä web-portfolion ulkoasun että sisällön sellaisiksi, että toteutukselle asettamani responsiivisuuteen liittyvät vaatimusmäärittelyt täyttyivät. Koko kehitysvaiheen läpi simuloin sivuston käyttäytymistä erilaisilla resoluutioilla käyttämällä Google Chrome -selaimen kehittäjätyökaluja (*developer tools*). Kehitystyö tapahtui omalla kannettavalla tietokoneellani, joten käytin kehittäjätyökaluja pääasiallisesti simuloimaan sivuston renderöitymistä sekä mobiililaitteilla että oman tietokoneeni näyttöä suuremmilla laajakuvanäytöillä. Kun sivusto oli mielestäni valmis julkaistavaksi verkkoon, ostin sille nimipalvelimen ja julkaisin sen verkkoon ilmaista palvelintilaa tarjoavan hostauspalvelu Netlifyn kautta. Sekä julkaiseminen että https-protokollan käyttöönotto onnistuivat helposti Netlifyn kautta. Julkaisemisen jälkeen jatkoin testaamista omalla älypuhelimellani sekä laajakuvanäyttöön kytketyllä tietokoneella varmistuakseni siitä, oliko toteutus onnistunut.

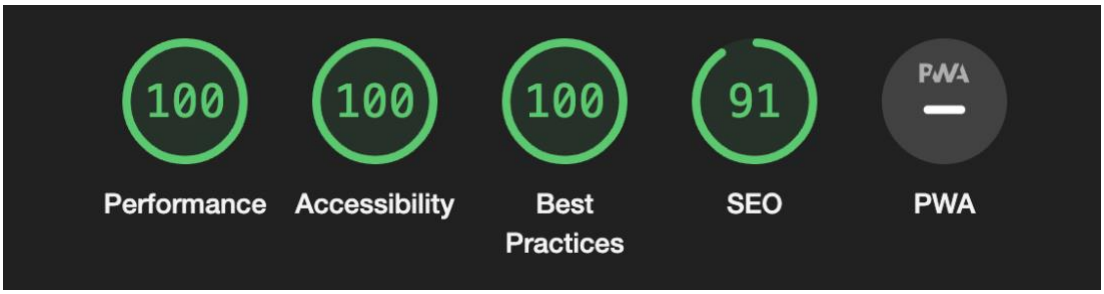
Tässä vaiheessa havaitsin vielä korjattavaa ja hienosäädettävää sivuston eri elementeissä ja niiden skaalautuvuudessa, joita en selaimen kehittäjätyökalujen avulla ollut havainnut. Yksi näistä ongelmista liittyi liukuvärjätyn taustan ei-toivottuun käyttäytymiseen mobiililaitteella tarkasteltaessa. Liukuvärjätyn taustan asettelu ja käyttäytyminen oli yllättävän vaikeaa saada toimimaan toivotulla tavalla sekä mobiili- että selainversioissa. Tätä ja muutamaa muuta estetiikkaan liittyvää näennäisen pientä, mutta häiritsevää ongelmaa täytyi selvitellä ja säätää työn viimeistelyvaiheen aikana vielä kohtuullisen paljon.

Näiden lisäksi julkaisemisen jälkeen havaitsin, että sivujen päivittäminen selaimen *refresh*-painikkeen kautta ei onnistunut vaan johti virhetilanteeseen, jossa selain ei löydä sivustoon kuuluvaa sivua. Ongelmaa selvittäessäni havaitsin tämän liittyvän single-page web-applikaatioiden ominaisuuteen. Single-page applikaatioissa palvelin ei säilytä sovelluksen tilatietoja, vaan niiden käsittely tapahtuu käyttäjän (*client*) päässä. Koska reitityksen tarvitsemat tilatiedot eivät palautuneet palvelimelta sivua päivitettäessä, sivun lataaminen ei onnistunut. Ratkaisuksi löytyi käyttää konfiguraatiotiedostoa, jonka avulla

palvelinpuolelta tapahtuva reitityksen uudelleenohjaus pystyttiin liittämään toteutukseen.

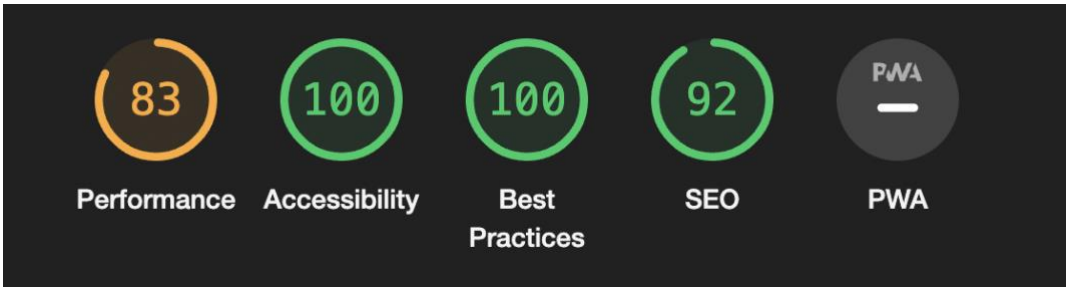
Kun toteutus oli viimeistelty ja havaitsemani virheet korjattu, suoritin sivustolle Google Lighthouse -testauksen selaimen kehittäjätyökalujen kautta. Google Lighthouse on Googlen kehittämä avoimeen lähdekoodiin perustuva automatisoitu testaustyökalu, jota voidaan käyttää verkkosivujen suorituskyvyn, saavutettavuuden, parhaiden käytäntöjen ja hakukoneoptimoinnin arvioimiseen. Testauksen osa-alueiden arvioiminen tapahtuu asteikolle 0–100 sijoittuvalla pisteytysjärjestelmällä, jossa arvon 90 ylittävä tulos tarkoittaa läpäistyä testitulosta. (Hudson 2023.) Syy suorittaa Lighthouse-testaus oli saada ulkopuolinen arvio sivuston responsiivisuudesta, suorituskyvystä ja saavutettavuudesta.

Tavoitteeni oli päästä kaikkien Lighthouse-testauksen osa-alueiden osalta yli 90 pisteeseen. Ensimmäisessä tekemässäni testauksessa tulokset jäivät suorituskyvyn osalta alle 90:n, koska kuvien latausnopeus oli liian hidas. Päätin parantaa tätä kompressoimalla vektorimuotoiset kuvat ja tallentamalla bittikarttamuotoiset kuvat alkuperäisen tiedostomuodon lisäksi myös webp-tiedostomuotoon. Päädyin tähän ratkaisuun, koska webp-tiedostomuoto pienentää jpg ja png -muotoisten kuvatiedostojen kokoa yleisesti ottaen noin 25 %. Nykyään kaikki uudet käyttöjärjestelmät ja selaimet myös tukevat sitä. (Frain 2022, luku 9.) Tallennettuani kuvat webp-muotoon toteutin sovellukseen kuvien lataaminen ensisijaisesti tässä tiedostomuodossa. Tämä muutos paransi kuvien latausnopeutta, ja työpöytäversiolle suorittamissani uusituissa testeissä testitulokset nousivat jokaisen sivustoon kuuluvan sivun osalta yli 90 pisteeseen. Testitulokset on esitetty oheisessa kuvakaappauksessa (kuva 8).

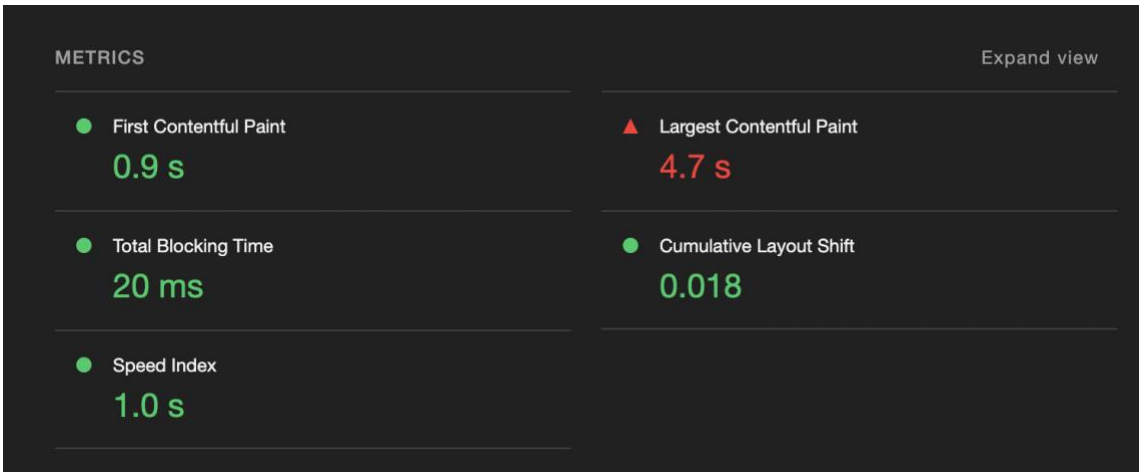


Kuva 8. Työpöytäversiolle suoritetun Lighthouse-testauksen tulokset olivat samat kuin kuvassa esitetty tulos kaikkien sivustoon kuuluvien sivujen osalta.

Mobiiliversion tulosten osalta on huomionarvoista mainita, että mobiililaitteiden osalta tulokset voivat olla matalammat, koska niiden prosessorit ovat tyypillisesti tietokoneiden prosessoreita hitaampia ja myös internetyhteys voi olla heikompi (Hudson 2023). Mobiiliversion osalta kaikki muut sivustoon kuuluvat sivut läpäisivät kaikki testauksen osa-alueet, mutta etusivun osalta suorituskykyä (kuvassa *performance*) mittaavan testin tulos jäi 83:een (kuva 9). Syy voidaan havaita suorituskykyyn liittyvien testien tulosten erittelystä (kuva 10).



Kuva 9. Sivuston etusivun testitulos jäi mobiiliversiossa suorituskyvyn osalta alle 90:n.



Kuva 10. Mobiiliversion etusivun testitulokset suorituskyvyn osalta eriteltyinä.

Kuvassa 10 punaisella värillä esitetty hylätty testitulos ”*Largest Contentful Paint 4.7 s”* tarkoittaa aikaa, joka kuluu sivun suurimman elementin, esimerkiksi kuvan tai muun mediaelementin lataamiseen. Sivuston etusivun vektorimuotoisen (svg) kuvatiedoston koko on suuri, minkä vuoksi mobiililaitteilla sen lataamiseen kuluva aika jää alle testirajan. Kyseinen vektorikuva on jo kompressoitu pienimpään mahdolliseen tiedostomuotoonsa, joten sitä ei voi enää pienentää ilman että kuvan laatu kärsii. Etusivun latausnopeus ei kuitenkaan mielestäni ole ihmisen silmään häiritsevän hidas, joten 90 pisteen alittava testitulos jää sen osalta voimaan siitä syystä, että etusivulla esitettävän vektorimuotoisen kuvan toteutus ei ole optimaalinen verkossa esitettäväksi.

6 Opinnäytetyön tuotos

Web-sovellusten käyttöliittymäsuunnittelussa tarvitaan graafista osaamista mahdollistamaan miellyttävä käyttökokemus. Mielestäni opinnäytetyön tuotoksessa välittyy hyvin, miten tekninen osaamiseni yhdistyy graafiseen osaamiseeni. Tämä ei ollut toteutukselle asetettu vaatimus, koska vaatimukset liittyivät suoraan responsiivisuuteen, mutta halusin tämän välittyvän toteutuksesta. Mielestäni onnistuin toteuttamaan sivustolle tyylikkään ulkoasun ja hyvän käyttökokemuksen. Tietysti tärkeintä oli, että toteutukselle asettamani responsiivisuutta koskevat vaatimuskriteerit toteutuisivat, koska pääasiallinen tavoitteeni oli responsiivisuuden toteuttaminen. Mielestäni kaikki kuusi toteutukselle asettamaani vaatimusta toteutuivat, ja tätä tukevat myös suorittamani Google Lighthouse automaatiotestauksen tulokset.

Opinnäytetyöni tarkoitus oli perehtyä responsiivisuuteen osana web-sovellusten kehitystä ja toiminnallisena osuutena toteuttaa responsiivinen websivusto. Toteuttamani web-portfolion kehittämisen aikana oli mielestäni mahdollista perehtyä responsiivisuuden toteuttamiseen riittävän laajasti käytännössä. Sen kehittäminen oli siis mielestäni tarkoituksenmukaista ja perusteltua suhteutettuna opinnäytetyöprosessille asettamiini tavoitteisiin. Se kannatti mielestäni toteuttaa

verrattuna siihen, että työ olisi tutkimuksen omaisesti käsitellyt asioita vain teoriassa tai että toteutettava esimerkkisovellus olisi jäänyt vaille oikeaa käyttötarkoitusta. Työn lopputulos jää verkkoon katseltavaksi ja tulee ainakin toistaiseksi toimimaan oman osaamiseni markkinointikanavana. Toteutus tulee säilymään omassa käytössäni, ja aion ylläpitää ja jatkokehittää sitä. Opinnäytetyöprosessin aikana käytettävä aika on rajallinen, mutta sain sen toteutettua sivulle kaiken sen sisällön, jonka olin suunnitellut toteuttavani.

7 Pohdinta

7.1 Ajankäyttö

Työskentelyni opinnäytetyöprosessin ajan oli hyvin itsenäistä ja eteni ilman että kohtasin suuria haasteita työskentelyn aikana. Sekä opinnäytetyön toiminnallinen että kirjallinen osuus vaativat kuitenkin yhtä lailla sekä ajallista panostusta että suunnitelmallisuutta ja huolellista keskittymistä tekemiseen. Työskentelyn raportoiminen oli paikoin haasteellista johtuen siitä, että web-portfolion toteuttamisen aikana tekemäni muistiinpanot työskentelystäni olivat melko suppeat. Kattavampien muistiinpanojen tekeminen olisi helpottanut ja nopeuttanut työskentelyn raportoimista, joten tässä suunnitelmallisuus hieman petti. Silti koen, että juuri suunnitelmallisuus yhdistettynä motivaation ja tehokkaaseen ajankäyttöön olivat avaintekijät työn onnistumisen taustalla.

Ajankäyttöä on tosin jälkeenpäin vaikeaa arvioida, koska en käyttänyt opinnäytetyöprosessin aikana työajanseurantaa. Työajanseurannan puuttumisen onkin toinen asia, joka tuli mieleeni pohtiessani sitä, olisinko voinut tehdä opinnäytetyöprosessin aikana jotakin paremmin tai toisin. Työajanseurannan avulla olisin voinut todentaa, paljonko aikaa eri tehtävien suorittamiseen kului, ja paljonko aikaa kului esimerkiksi opinnäytetyön tuotoksen tekemiseen verrattuna kirjallisen osuuden työstämiseen. Valitettavasti aloittaessani työskentelyn en tullut huomioineeksi asiaa. Työajanseuranta olisi ollut hyödyllinen menetelmä

todentaa ajankäyttö opinnäytetyöprosessin aikana, joten sen puuttuminen tuntuu jälkikäteen harmilliselta.

7.2 Luotettavuus ja eettisyys

Opinnäytetyössäni käytin lähteinä pääasiassa O'Reillyn korkeakouluopiskelijoille suunnatusta verkko-oppimisalustasta löytämiäni opinnäytetyön aihealueeseen liittyviä e-kirjoja sekä internetistä löytämiäni asiantuntijoiden laatimia artikkeleita, dokumentaatioita ja oppaita. Käytin myös hieman statistiikkaa esittämieni väitteiden tukena. Lähdeaineistoa valitessani ja lähteiden luotettavuutta pohtiessani pyrin selvittämään, onko tekstin laatineella taholla riittävästi aihealueeseen liittyvää asiantuntijuutta ja onko tekstin sävy puolueeton. Varmistaakseni tiedon luotettavuuden esittäessäni tietoa faktana pyrin lisäksi tarkistamaan, että tieto on löydettävissä useammasta eri lähteestä. Lähteiden käytössä pyrin myös huomioimaan tiedon ajantasaisuuden ja pitämään mielessäni opinnäytetyön aiherajauksen varmistaakseni, että tieto ei ole vanhentunutta ja liittyy olennaisesti opinnäytetyön aihealueeseen.

Työn eettisyyttä arvioidessani en havainnut eettisyyteen liittyviä ongelmia tai ristiriitoja. Opinnäytetyöni tuotos eli web-portfolio ei käytä evästeitä eikä kerää dataa sivustolla vierailijoista, jotka ovat sellaisia asioita, joissa tulisi huomioida käyttäjän suostumukseen ja tietosuojalakiin liittyviä asioita. Tietoturvaan liittyen käytän sivuston toteutuksessa https-protokollaa taatakseni suojatun yhteyden sivustolla vierailijoille. Vaikka sivuston kautta ei tapahdukaan arkaluontoista tiedonsiirtoa, https-protokollan käyttäminen on suositeltavaa ja myös yksi keino viestiä sivustolla vierailijoille, että sivustolla vieraileminen on turvallista.

7.3 Ammatillisen osaamisen kehittyminen

Web-sovelluskehittäjän on hyvä tuntea responsiivisen suunnittelun periaatteita ja tekniikoita, koska verkossa olevien sivustojen tulee toimia saumattomasti laajalla laitekirjolla. Tämän opinnäytetyöprosessin aikana syvensin web-

sovelluskehitykseen liittyvää ammatillistani osaamistani perehtymällä responsiivisuuteen yhtenä webkehityksen osa-alueena. Opinnäytetyöprosessin aikana opin uusia taitoja ja sovelluskehitystekniikoita, joiden soveltamista käytäntöön aion jatkaa myös tulevaisuudessa. Esimerkiksi Mobile First -suunnittelufilosofian periaatteet osoittautuivat hyödyllisiksi, ja aion jatkaa suunnittelutavan soveltamista osana verkkosivustojen suunnittelua ja toteutusta. Ennen opinnäytetyöprosessia olisi tuntunut luontevammalle aloittaa käyttöliittymän kehittäminen tietokoneen monitoreille tarkoitetusta versiosta, mutta projektin aikana huomasin, että kehittäminen todella kannattaa aloittaa mobiiliversiosta.

Toisena hyödyllisimmistä opinnäytetyöprosessin aikana oppimistani taidoista pidän CSS Gridiin perehtymistä. Opinnäytetyöprosessin alussa en tiennyt tekniikan olemassaolossa, ja suunnittelinkin aluksi toteuttavani ulkoasun ilman Gridiä. Muutin kuitenkin mieleni, koska käyttämäni lähteet suosittelivat vahvasti Gridin käyttämistä, joista osa piti sitä ainoana oikeana tapana määritellä käyttöliittymän ulkoasu, ja jakaa se eri osioihin. Gridiin perehtyminen osoittautui kannattavaksi, ja tulen jatkamaan myös tämän tekniikan käyttämistä tulevissa web-sovelluskehitysprojekteissa. Tekniikka vaati alkuun opettelua, mutta kun opin hahmottamaan sen käyttötavan ja tarkoituksen, sain tehokkaan työkalun mahdollistamaan keskenään hyvin erilaisten esitystapojen rakentaminen. Gridin käyttämisen lisäksi opin toteuttamaan erilaisia responsiivisia käyttöliittymäelementtejä perehtymällä suhteellisiin mittayksiköihin ja CSS-funktioihin. CSS-funktioista erityisesti calc() ja clamp() osoittautuivat mielestäni hyödyllisimmiksi, koska niiden avulla voi skaalata esimerkiksi tekstin kokoa portaattomasti. Gridin tavoin havaitsin myös CSS-funktioiden helpottavan käyttöliittymän ylläpidettävyyttä ja vähentävän tarvittavien CSS-tyylien ja mediakyselyiden määrää.

Opinnäytetyöprosessin aikana laajensin siis huomattavasti erityisesti CSS-kieleen liittyvää osaamistani. Samalla harjoittelin verkkosivuston toteuttamista komponenteista rakentuvana single-page applikaationa ja toteutin erilaisia käyttäjän toimiin reagoivia sovelluskomponentteja Reactilla. Opinnäytetyöprosessin jälkeen osaan rakentaa kustomoituja, responsiivisia web-

käyttöliittymiä ja käyttöliittymäelementtejä aiempaa tehokkaammin ja ammattimaisemmin. Työtehokkuuteni parani, koska tunnen nyt hyvin responsiivisten web-käyttöliittymien toteuttamisessa tarvittavia tekniikoita, ja tunnen eri käyttötarkoituksiin sopivia toteutustapoja. Myös valmiiden komponenttien ja käyttöliittymäelementtien kustomoiminen on jatkossa todennäköisesti helpompaa, koska myös niiden responsiivisuus pohjautuu opinnäytetyöprosessin aikana opiskelemiini ominaisuuksiin ja tekniikoihin.

Web-portfoliota kehittäessäni huomasin aika-ajoin, että jos jonkin asian tekeminen tuntui tarpeettoman monimutkaiselta, se yleensä myös oli sitä. Kehitystyön aikana opin, että asiat voi tehdä monella eri tavalla, mutta asiantuntemuksen avulla asioita voi tehdä yksinkertaisemmin ja siten myös tehokkaammin. Aihealueeseen liittyvän ymmärrykseni lisääntyessä pystyinkin yksinkertaistaman web-portfolion toteutusta monin tavoin, esimerkiksi yksinkertaistamalla komponenttirakennetta, tarvetta sisäkkäisille rakenteille sekä tarvittavien CSS-tyylien määrää. Uskon, että opinnäytetyöprosessin aikana oppimani taidot tulevat tekemään myös tulevaisuuden web-kehitysprojekteistani ammattimaisempia. Opinnäytetyöprosessin aikana opin web-käyttöliittymäsuunnittelun lisäksi uutta myös verkkosivujen julkaisemisesta ja ylläpitämisestä.

7.4 Tavoitteiden toteutuminen

Tässä opinnäytetyössä tarkoituksena oli perehtyä responsiivisuuteen osana web-sovelluskehitystä ja toteuttaa esimerkkisovellus eli responsiivinen web-portfolio soveltamalla hankittua tietoperustaa käytäntöön. Tavoitteena oli oppia suunnittelemaan ja toteuttamaan responsiivisia käyttöliittymiä ja responsiivista verkkosisältöä. Tuotoksen osalta tavoitteena oli, että se mukautuisi saumattomasti mobiililaitteilta suurille laajakuvanäytöille. Koska opinnäytetyön aihe oli kiinnostava, jaksoin käyttää aihealueeseen perehtymiseen, responsiivisuuteen liittyvien tekniikoiden harjoittelemiseen ja tuotoksen toteuttamiseen riittävästi aikaa mahdollistaakseni näiden tavoitteiden toteutumisen. Opinnäytetyön tuotoksena syntynyt web-portfolio on mielestäni

viimeistelty, toteuttaa sille asetetut vaatimusmäärittelyt ja on käyttötarkoitukseensa sopiva. Lopputulosta on testattu selaimen kehittäjätyökalujen ja testiautomaation avulla, sekä tarkastelemalla toteutettua sivustoa erilaisilla laitteilla. Saatujen testitulosten perusteella lopputulos on toimiva ja responsiivinen.

Tuotoksen lisäksi olen tyytyväinen myös opinnäytetyön kirjalliseen osuuteen. Kirjoittamisessani olen panostanut ilmaisun selkeyteen ja hyvään suomen kieleen. Kirjallinen osuus etenee mielestäni johdonmukaisesti muodostaen ymmärrettävän ja aiherajauksen sisällä pysyvän tiiviin kokonaisuuden. Toivon, että opinnäytetyön aihealueeseen liittyvä mielenkiintoni ja siihen perehtymiseen käyttämäni vaivannäkö välittyy myös tekstistäni. Näistä syistä koen, että opinnäytetyöni onnistui kokonaisuudessaan hyvin ja että sille asetetut tavoitteet täyttyivät.

Lähteet

Adarsh, M. 2023. 24 Best CSS Frameworks To Look Forward In 2023. https://www.lambdatest.com/blog/best-css-frameworks/. 6.11.2023.

Bose, S. 2023. Top 5 CSS Frameworks for Developers and Designers. https://www.browserstack.com/guide/top-css-frameworks/. 17.1.2024.

Eckles, S. 2023. Unleashing the Power of CSS: Advanced Techniques for Responsive User Interfaces. Australia: SitePoint. O'Reilly's Learning Platform for Higher Education. 6.11.2023.

Friedman, V. 2018. Responsive Web Design: What It Is And How To Use It. https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/. 21.11.2023.

Google. 2023. Mobile site and mobile-first indexing best practices. https://developers.google.com/search/docs/crawling-indexing/mobile/mobile-sites-mobile-first-indexing?hl=en/. 22.10.2023.

Frain, B. 2022. Responsive Web Design with HTML5 and CSS -Fourth Edition. Iso-Britannia: Packt Publishing. O'Reilly's Learning Platform for Higher Education. 8.11.2023.

Grant, W. 2022. 101 UX Principles - Second Edition. Iso-Britannia: Packt Publishing. O'Reilly's Learning Platform for Higher Education. 6.11.2023.

Hart-Davis, G. 2023. Teach Yourself VISUALLY HTML and CSS, 2nd Edition. USA: Wiley. O'Reilly's Learning Platform for Higher Education. 6.11.2023.

Hartl, M. & Donahoe, L. 2022. Learn Enough HTML, CSS and Layout to Be Dangerous: An Introduction to Modern Website Creation and Templating Systems. USA: Addison-Wesley Professional. O'Reilly's Learning Platform for Higher Education. 8.11.2023.

Hudson, H. 2023. Google Lighthouse Metrics to Track for Client SEO Success. https://agencyanalytics.com/blog/google-lighthouse-metrics/. 27.12.2023.

Interaction Design Foundation. 2016. What is Mobile First? https://www.interaction-design.org/literature/topics/mobile-first/. 21.11.2023.

Interaction Design Foundation. 2022. Responsive Design: Best Practices. https://www.interaction-design.org/literature/article/responsive-design-let-the-device-do-the-work/ 21.11.2023.

LePage, P. & Andrew, R. 2019. Responsive web design basics. https://web.dev/articles/responsive-web-design-basics/. 10.11.2023.

Little, C. 2021. The history of web design. https://tillerdigital.com/blog/the-history-of-web-design/ 18.1.2024.

Minnick, C. 2022. Beginning ReactJS Foundations Building User Interfaces with ReactJS. USA: Wiley. O'Reilly's Learning Platform for Higher Education. 9.11.2023.

Mozilla Developer Network. 2023. Grids. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids/. 5.11.2023.

Mozilla Developer Network. 2023. Responsive design. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design/. 2.11.2023.

Mozilla Developer Network. 2023. What is JavaScript? https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript/ 3.11.2023.

Rampton, J. 2021. 5 Reasons You Should Have a Content Management System. https://blogs.oracle.com/marketingcloud/post/5-reasons-you-should-have-a-content-management-system/. 17.1.2024.

Roldán, C. 2023. React 18 Design Patterns and Best Practices - Fourth Edition. Iso-Britannia: Packt Publishing. O'Reilly's Learning Platform for Higher Education. 14.11.2023.

Shadeed, A. 2023. The Guide To Responsive Design In 2023 and Beyond. https://ishadeed.com/article/responsive-design/. 21.10.2023.

Social Media Today. 2015. Responsive Web Design: A Brief History. https://www.socialmediatoday.com/content/responsive-web-design-brief-history/ 18.1.2024.

Staiano, F. 2022. Designing and Prototyping Interfaces with Figma. Iso-Britannia: Packt Publishing. O'Reilly's Learning Platform for Higher Education. 6.11.2023.

Statcounter. 2023. Desktop vs Mobile vs Tablet Market Share worldwide - October 2023. https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/. 20.10.2023.

Statcounter. 2023. Browser Market Share Worldwide – October 2023. https://gs.statcounter.com/browser-market-share/. 20.10.2023.

Stimac, S. 2023. Design for Developers. USA: Manning Publications. O'Reilly's Learning Platform for Higher Education. 6.11.2023.

Vilkka, H. & Airaksinen, T. 2003. Toiminnallinen opinnäytetyö. Kustannusosakeyhtiö Tammi. 27.12.2023.

World Wide Web Consortium. 2023. Understanding SC 1.4.10: Reflow (Level AA). https://www.w3.org/WAI/WCAG21/Understanding/reflow.html/. 9.11.2023.

W3Schools. 2023. HTML Responsive Web Design. https://www.w3schools.com/html/html_responsive.asp/. 20.10.202