Aleksi Heikkilä

Verkkosivun toimivuuden varmistaminen eri selaimissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

6.5.2022

Tiivistelmä

Tekijä:	Aleksi Heikkilä
Otsikko:	Verkkosivun toimivuuden varmistaminen eri verkkoselaimissa
Sivumäärä:	35 sivua
Aika:	6.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Yliopettaja Auvo Häkkinen

Verkkosivuja käytetään useilla selaimilla ja laitteilla. Sivustot eivät aina toimi samalla tavalla kaikilla alustoilla ohjelmien eroavaisuuksien takia. Sivustojen kehityksen yhteydessä tulisi siksi testata toimivuus mahdollisimman laajasti, jotta sivustot toimivat ainakin suurimmalla osalla käyttäjistä.

Tässä insinöörityössä tarkasteltiin keinoja yhteensopivuusongelmien havaitsemiseksi ja löydettyjen ongelmien korjaamiseksi. Ongelmien syiden ymmärtämiseksi työssä esiteltiin eri selainten ominaisuuksia ja eroavaisuuksia. Yhteensopivuuden edistämiseksi ohjelmoinnissa myös esiteltiin yleisimpiä yhteensopivuusongelmia verkkosivustojen kehityksessä. Näihin ongelmiin pyrittiin esittämään ratkaisuja. Osana työtä tutkittiin myös ongelmien ennaltaehkäisemistä ohjelmien ja koodikäytäntöjen avulla.

Yhteensopivuuden varmistamisen havainnollistamiseksi osana työtä tarkasteltiin esimerkkisivustoa, joka kehitettiin ottamatta huomioon yhteensopivuusongelmia. Kehityksen jälkeen sivustolta etsittiin suurin osa ongelmista ja korjattiin ne. Osana esimerkkisivuston tarkastelua pohdittiin myös, miten verkkosivujen kehittämisessä voidaan ennaltaehkäistä havaittuja ongelmia. Ennaltaehkäisemällä ongelmia voidaan vähentää niistä aiheutuvaa tulevaisuuden työkuormaa. Esimerkkisivustolta onnistuttiin löytämään useita yhteensopivuuteen liittyviä ongelmia, jotka korjattiin tehokkaasti.

Avainsanat: Verkkoselainten yhteensopivuus, Verkkosivun testaaminen, Käyttäjäkokemus

Abstract

Author:	Aleksi Heikkilä
Title:	Ensuring Cross Browser Compatibility
Number of Pages:	35 pages
Date:	6 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Supervisors:	Auvo Häkkinen, Principal Lecturer

Websites are used on many browsers and devices. Sites do not always work the same way on all platforms due to differences. When developing new websites, functionality should therefore be tested as widely as possible so that the websites work for at least most of the audience.

This thesis examines the means to detect compatibility problems and to correct the problems found. The features and differences of different browsers were examined to help understand the occurrence of the problems. The most common compatibility problems are also presented to further compatibility in web development. Solutions to the problems are also presented. Problem prevention through programs and code practices was also studied as part of the research.

A previously developed website was examined to better illustrate the ensuring of compatibility as part of the study. This website was created without taking compatibility issues into consideration. Efforts were made to find and fix most of the problems that appeared on the website. It was also pondered as part of the example website how the problems encountered in the process could be prevented to reduce the resulting workload caused by compatibility problems in the future. Several compatibility issues were successfully found on the sample site and were effectively fixed.

Keywords: Browser Compatibility, Website Testing, User Experience

Sisällys

Lyhenteet

CSS *Cascading Style Sheets*. Verkkosivujen tyylien määrittämiseen tarkoitettu säännöstö.

HTML *HyperText Markup Language*. Verkkosivujen merkintäkieli, joka määrittää verkkosivustojen rakenteen.

JS *JavaScript*. Ohjelmointikieli, jolla voidaan muun muassa lisätä verkkosivuille sisältöä muokkaavia toiminnallisuuksia.

NPM *Node Package Manager.* Node.js-suoritusympäristöön suunniteltu ohjelmistopakettien latausohjelma.

1 Johdanto

Insinöörityössä tutkitaan verkkosivujen kehittämistä eri selaimille ja alustoille. Selaimissa on usein eroavaisuuksia niiden ominaisuuksissa ja toiminnallisuuksissa. Siksi on tärkeää testata luodut verkkosivut läpikotaisesti yhteensopivuusongelmien varalta. Ongelmien korjaamiseen esitellään keinoja, jotka riippuvat ongelmien alkuperästä. Verkkosivujen mahdollisimman laajalla yhteensopivuudella saavutetaan laajempi yleisö luoduille verkkosivuille ja tuotteille. Jos sivustot eivät toimi asiakkaan laitteella oikein, asiakas voi helposti turhautua ja siirtyä kokonaan pois sivustolta. Tässä tilanteessa asiakkaan menetys on suora seuraus sivuston huonosta toimivuudesta.

Sivustojen hyvä toimivuus eri alustoilla on myös usein tärkeä osa suurten yritysten imagoa. Jos käyttäjä huomaa sivuston toimivan huonosti tai ei lainkaan, käyttäjä todennäköisesti poistuu sivustolta hyvinkin nopeasti. Tämä vuorostaan lisää poistumisprosenttia. Poistumisprosentilla tarkoitetaan käyttäjien määrää, jotka poistuvat sivustolta tarkastelematta muita sivuston sivuja. Poistumisprosentilla on suora vaikutus verkkosivuston näkyvyyteen hakukoneiden tuloksissa, mikä täten myös vähentää mahdollisten asiakkaiden määrää.

Insinöörityössä keskitytään pääosin verkkosivujen yhteensopivuuden manuaaliseen testaukseen ja löytyneiden ongelmien korjaamiseen. Työssä esitellään askelia, joita seuraamalla voidaan mahdollistaa mahdollisimman helppo ja nopea yhteensopivuus kehitetyissä verkkosivustoissa. Myös ongelmia ennaltaehkäiseviä menetelmiä esitellään osana verkkosivujen kehitystä. Näitä menetelmiä seuraamalla ilmaantuvia yhteensopivuusongelmia voidaan vähentää.

Verkkosivujen kehityksessä yhteensopivuus ja sen testaus usein jäävät jälkeen muista prioriteeteistä. Yhteensopivuuden testaaminen tehdään usein satunnaisesti ja nopeasti, jos sitä huomioidaan lainkaan. Tällöin usein jätetään huomioimatta mahdollisia ilmaantuvia ongelmia varsinkin vähemmän käytetyillä alus-

toilla tai selaimilla. Yhteensopivuuden testaustyössä kannattaa seurata johdonmukaista prosessia. Näin ongelmat löytyvät mahdollisimman laajalta alalta, ja harvinaisemmatkin ongelmat saadaan korjattua.

Työssä esitellään ratkaisuja yhteensopivuusongelmiin, kuten esimerkiksi Polyfill-toteutusta mahdollisiin selaimesta puuttuviin ominaisuuksiin. Polyfillillä tarkoitetaan usein koodinpätkää tai koodikirjastoa, jolla luodaan toiminnallisuus selaimelta puuttuville ominaisuuksille. Työssä keskitytään pääosin verkkosivuihin, jotka on kehitetty HTML-, CSS- ja JavaScript-ohjelmointikielillä. Työssä tarkastellaan myös esimerkkinettisivua, josta etsitään ja korjataan mahdollisimman monta ilmaantuvaa yhteensopivuusongelmaa. Kyseinen verkkosivu on luotu osana aiempaa työnhakuprosessia käyttäen edellä mainittujen teknologioiden lisäksi myös JavaScriptin React JS -ohjelmistokehystä (engl. framework). Ohjelmistokehyksellä tarkoitetaan eräänlaista valmiiden ohjelmistokomponenttien alustaa. Tätä alustaa käytetään pohjana, jonka päälle luodaan omaa ohjelmoitua sisältöä. [1.]

Työn tavoitteena on antaa lukijalle parempi ymmärrys eri selainten toiminnallisuuksista ja samalla vertailla niitä toisiinsa. Työssä halutaan myös auttaa lukijaa ymmärtämään, mistä kyseiset ongelmat johtuvat ja miten niitä voidaan korjata. Työn luettuaan lukijalla on perustason valmiudet alkaa etsimään ja korjaamaan yhteensopivuusongelmia.

Työssä ei ole varsinaista toimeksiantajaa, mutta työn avulla tutustutaan paremmin yhteensopivuuden varmistamiseen. Tutustumisen kautta kehitetään omaa työskentelyä Web Developerin tehtävissä.

2 Selainten yhteensopivuudesta

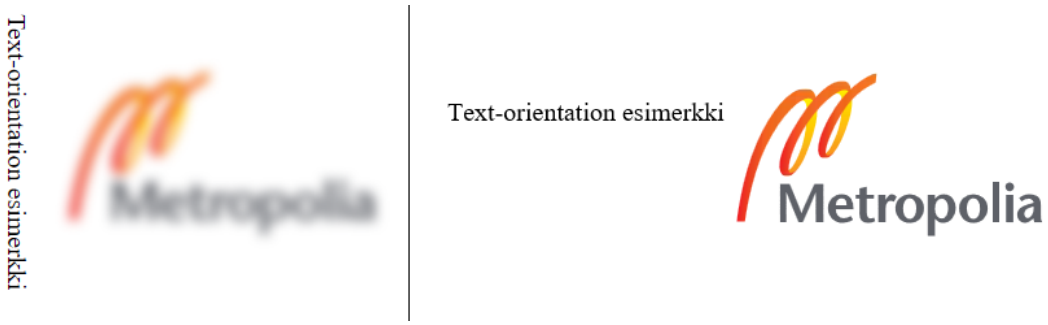
Tärkeä osa ohjelmistokehitystä on yhteensopivuuden varmistaminen eri alustoilla. Tämä pätee erityisesti verkkosivujen kehittämisessä, sillä sivuja käytetään usein erilaisilla selaimilla ja laitteilla. Tämän takia on erittäin tärkeää varmistaa sivun toiminnallisuus mahdollisimman monessa ympäristössä.

Osana varmistamista voidaan luoda vaihtoehtoiset toiminnallisuudet juuri yhteensopivuusongelmia sisältäviä alustoja varten. Toisena vaihtoehtona on varmistaa sivustojen vikasietoisuus. Vikasietoisuudella tarkoitetaan tässä tapauksessa varmistusta siitä, että sivusto jatkaa toimintaansa mahdollisista alustojen puutteista huolimatta. Sivusto ei saisi lakata toimimasta täysin esimerkiksi vain yhden viallisen tyylimäärityksen takia. Jos jostain syystä joitakin selaimia ei haluta tukea, tulisi tuen puuttuminen ilmaista jollain tavalla sivulla esimerkiksi popup-ikkunan muodossa. Tuen puute tulisi mainita, jotta käyttäjä voi halutessaan siirtyä käyttämään selainta, jolla sivusto toimii oikein tai paremmin.

Verkkosivujen yhteensopivuuden helpottamiseksi on perustettu vuonna 1994 World Wide Web Consortium (W3C), joka on kansainvälinen verkkosivustojen kehittäjien yhteisö. Yhteisön päätarkoituksena on pohtia ja sopia uusia verkkokehityksen standardeja, joita sen jäsenet pyrkivät tulevaisuudessa noudattamaan muun muassa verkkoselaimissaan. Yhteisö on sitoutunut pitämään internetin kaikenlaisten ihmisten ja laitteiden käytettävänä. Yhteisöön kuuluu suurin osa isoimmista verkkoselainten kehittäjäyrityksistä, kuten Google, Mozilla Foundation, Apple ja Microsoft. Yhteisön toiminnalla pyritään helpottamaan verkkosivujen kehittäjien työtä mahdollistamalla yleinen tuki selainten uusille ominaisuuksille. Tähän mennessä yhteisö on onnistunut standardisoimaan tuen muun muassa HTML:lle (HyperText Markup Language) ja CSS:lle (Cascading Style Sheets). HTML on verkkosivujen merkintäkieli, jolla voidaan määritellä verkkosivustojen rakenne käyttäen esimerkiksi tekstielementtejä, kuvia ja linkkejä. [2.]

Verkkosivujen kehityksessä usein päätetään aikaisessa vaiheessa, mitä selaimia aiotaan tukea ja mitä jättää pois tuen piiristä. Tuettujen alustojen päättämisessä voidaan käyttää apuna analytiikkaa, jos sellaista on saatavilla. Analytiikan avulla voidaan seurata, millä selaimilla ja laitteilla verkkosivustoja avataan. Tämän tiedon perusteella voidaan tehdä päätös siitä, mihin selaimen tuki halutaan keskittää. Yrityksissä tämä päätös on usein riippuvainen itse kehityksen hinnan ja hyödyn suhteesta. Jos analytiikan perusteella voidaan todeta tuottojen nousevan laajemman alustatuen seurauksena, saadaan yhteensopivuuden kehittämiselle helpommin lupa.

Selainten yhteensopivuudessa usein kiistanalainen verkkoselain on Internet Explorer, sillä kyseisen selaimen tuki nykyaikaisille ominaisuuksille on usein puutteellinen. Siitä huolimatta selaimen käyttäjäkanta on edelleen suurempien joukossa [3]. Esimerkkinä ominaisuuksien puutteesta kuvassa 1 vertaillaan text-orientation ja filter CSS-ominaisuuksia Chrome- (vas.) ja Internet Explorer -selaimissa (oik.).



Kuva 1. Vertailu Google Chrome- (vas.) ja Internet Explorer -selainten (oik.) tuesta eri toiminnallisuuksille.

Kuvassa on vasemmalla puolella Google Chrome -selaimessa oikein toimivat text-orientation ja filter CSS-ominaisuudet. Näillä ominaisuuksilla tekstin suuntaa voidaan vaihtaa ja kuvaan voidaan asettaa suodattimia, kuten esimerkiksi sumennuksen. Oikealla puolella Internet Explorer -selaimessa samat ominaisuudet eivät toimi, minkä takia kuva- ja tekstielementit näyttävät samalta kuin ilman määrityksiä.

Eri verkkoselaimissa on usein toisistaan eroava selainmoottori (engl. browser engine). Selainmoottorilla tarkoitetaan selaimessa olevaa komponenttia, jonka tehtävänä on tulkita sivuston tiedostot koodista visuaaliseksi sisällöksi ja toiminnallisuudeksi. Eroavaisuuksien takia usein kaikki toteutetut verkkosivun sisällöt ja toiminnallisuudet eivät välttämättä toimi lainkaan ilman juuri tiettyjä selaimia silmällä pitäen luotuja vaihtoehtoisia tyylejä ja toiminnallisuuksia. Esimerkkikoodissa 1 on määritelty useaan eri selainmoottoriin omat CSS-määritykset border-radius -komennolle, jotta tyyli toimisi mahdollisimman yhtenäisesti eri selainten välillä.

```
-moz-border-radius: 2em;
-ms-border-radius: 2em;
-o-border-radius: 2em;
-webkit-border-radius: 2em;
border-radius: 2em;
```

Esimerkkikoodi 1. Selainkohtaisia CSS-tyylimäärityksiä.

Selainkohtaisia CSS-määrityksiä käytetään usein, kun tietyt ominaisuudet ovat vielä kehitysvaiheessa eri selaimissa. Tyylimäärityksen alussa sijaitseva etuliite kertoo, mistä selaimesta on kyse. Esimerkiksi -moz tarkoittaa Mozilla Firefox -selainta ja -o tarkoittaa Opera-selainta.

Yhteensopivuuden varmistamiseen on olemassa useita ratkaisuja. Tässä insinöörityössä keskitytään pääosin verkkosivun manuaaliseen testaamiseen ja manuaaliseen toiminnallisuuden varmistamiseen. Verkkosivuja testataan manuaalisesti useimmiten avaamalla verkkosivu eri selaimissa tai niiden emulaattoreissa ja laitteissa, ja kokeilemalla mahdollisimman läpikotaisesti luodut toiminnallisuudet ja tyylit.

Tarvetta verkkosivujen testaamiselle voi myös vähentää ennaltaehkäisemällä yleisimpiä ongelmia. Ennaltaehkäisyyn voi muun muassa käyttää ohjelmointiympäristöjen liitännäisiä. Ne varoittavat jo ohjelmointivaiheessa monista mahdollisista virheistä tai puutteista.

Verkkosivuja kehitettäessä tulisi testata uusien komponenttien ja sisältöjen yhteensopivuutta mahdollisimman usein, jotta uudet ilmaantuvat ongelmat eivät unohdu ja jää valmiille verkkosivustolle.

Verkkosivujen manuaalitestauksessa noudatetaan usein tietynlaista prosessia. Tämä prosessi sisältää usein seuraavat askeleet:

1. Verkkosivun uusi sisältö suunnitellaan ennen varsinaisen toteutuksen tekemistä.
2. Sisältö kehitetään ja ohjelmoidaan.
3. Uuden sisällön toimivuutta testataan.
4. Jos ohjelmoidussa sisällössä löytyy ongelmia, ne korjataan mahdollisimman laajasti.

Kehitysprosessin askeleet 2–4 suoritetaan mahdollisimman monta kertaa, kunnes implementaatio on valmis ja toimivuus taattu.

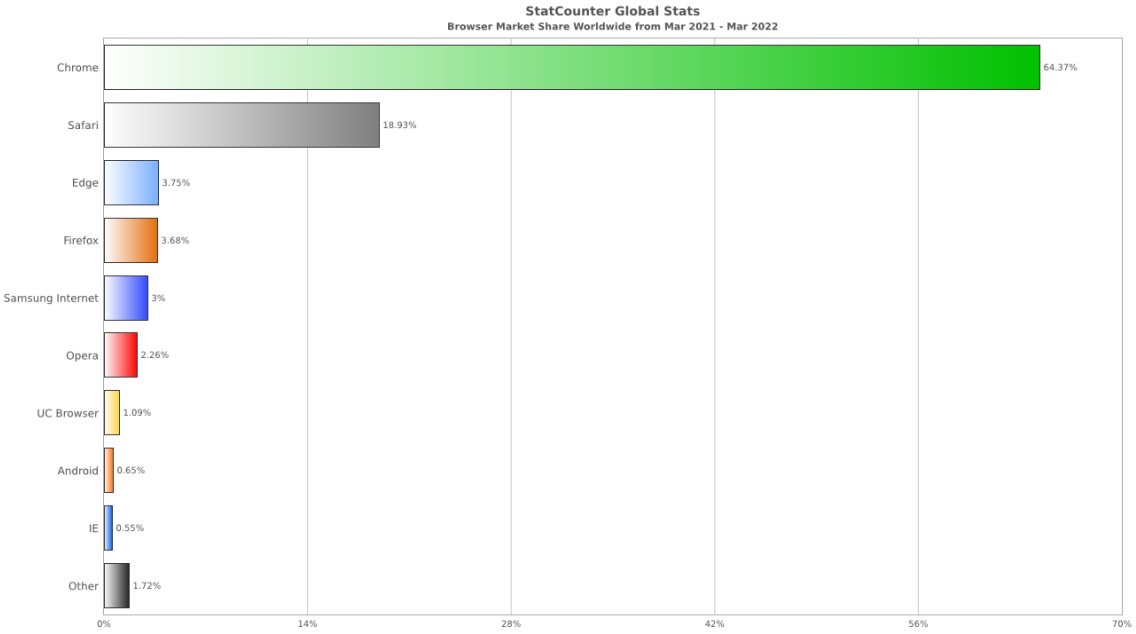
Testaamisen osana tulisi myös tarkistaa sivun toimivuus eri selaimissa esimerkiksi pelkällä näppäimistöllä, ja mahdollisesti myös ruudunlukijalla, jotta saavutettavuus toteutuu eri selaimissa. Manuaalitestausta tulisi myös tehdä käyttäen yhtä tai useampaa mobiililaitetta, jotta sivuston toimivuus mobiililaitteilla toteutuu tietokoneen käytön lisäksi.

Uusia luotuja verkkosisältöjä voidaan myös testata eri selainten kehitysversioissa. Tällä voi varmistaa uusien ominaisuuksien toimivuuden myös lähitulevaisuudessa, kun eri selaimiin julkaistaan uusia päivityksiä. [4.]

2.1 Selainten ominaisuuksien vertailua

Verkkosivujen käyttöä varten on useita verkkoselaimia. Selaimissa on usein hieman erilaisia ominaisuuksia, eivätkä kaikki selaimet välttämättä tue aina samoja CSS-tyylejä tai JavaScript-toiminnallisuuksia. Harvoin myös jotkin HTML-toiminnallisuudet voivat puuttua tietyistä selaimista.

Eri selaimet toimivat erilaisilla ohjelmistomoottoreilla, joista yleisimmin käytössä olevat ovat muun muassa Blink, WebKit, Gecko, Trident ja EdgeHTML. Näistä moottoreista WebKit-moottoriin pohjautuva Blink-moottori on laajimmassa käytössä. Se on käytössä muun muassa Google Chrome -selaimessa [3, 5]. Kuvan 2 pylväsdiagrammissa on myös esiteltynä muiden selainten markkinaosuus Chromen lisäksi.



Kuva 2. Pylväsdiagrammi selaimien markkinaosuuksista [3].

Kuvassa on pylväinä yhdeksän suosituinta selainta käyttäjämäärien mukaan. Viimeiseen pylvääseen on koottu muut pienemmät selaimet. Pylväsdiagrammi on saatu gs.statcounter.com-sivustolta.

Edellä mainituista moottoreista Trident (toiselta nimeltään MSHTML) on ollut ja on edelleenkin käytössä Internet Explorer-selaimessa. Selaimen tuki on päättymässä vuoden 2022 kesäkuussa [6]. Yritykset ja ihmiset käyttävät edelleen kyseistä selainta, ja juuri tämän selaimen ikä aiheuttaa useasti ongelmia erilaisten toiminnallisuuksien ja tyylien käytössä. Jos verkkosivun kehityksessä halutaan säilyttää mahdollisuus käyttää juuri Internet Explorer -selainta, sitä varten tulee luoda hyvinkin laajat vaihtoehtoiset toteutukset toiminnallisuuden varmistamiseksi.

Google Chrome -selaimen valta-asemaan on kohdistunut kritiikkiä, sillä selaimen suuren käyttäjäkannan pelätään johtavan internetin ja sen ohjelmointikäytäntöjen monopolisoitumiseen. Jos suurin osa verkkosivujen kehittäjistä suunnittelee sivustonsa ainoastaan Google Chromelle, muiden verkkoselainten toimimattomuus voi siirtää yhä enemmän käyttäjiä Googlen selaimelle. Tämän lähes

monopoliaseman vuoksi voidaan selaimeen täten myös helpommin lisätä ominaisuuksia, joita käyttäjät eivät haluaisi, ilman suurempia seurauksia. Edellä mainittujen seikkojen takia verkkosivustojen kehittäjien tulisi tukea tulevaisuudessakin mahdollisimman laajasti muitakin selaimia Google Chromen lisäksi. [7.]

Verkkoselaimia jatkokehitetään jatkuvasti, minkä seurauksena verkkokehitykseen tulee jatkuvasti uusia ominaisuuksia. Verkkoselainten vanhemmissa versioissa kyseisten ominaisuuksien käyttö on todennäköisesti mahdotonta, jolloin niiden käyttäminen vaatii usein selainten uusimpia versioita. Tällaisissa tapauksissa tulee jälleen luoda vaihtoehtoiset toteutukset ominaisuuksille, jos toimivuus halutaan varmistaa. Tämän takia työtaakan vähentämiseksi useimmiten kannattaa välttää kaikista uusimpien ominaisuuksien käyttöä, ellei olla varmoja uusimpien ominaisuuksien toimivuudesta kehitetyn sivuston tukemilla selaimilla.

Osa verkkoselaimista on avoimen lähdekoodin projekteja (engl. open-source). Avoimen lähdekoodin projektilla tarkoitetaan sellaista projektia, jonka lähdekoodi on julkisena yleisön tarkasteltavissa, muutettavissa ja uudelleenjaettavissa. Avoin lähdekoodi selaimessa mahdollistaa yhteisön tekemät ongelmien korjaukset suoraan selaimen lähdekoodiin. Selaimesta puuttuville ominaisuuksille voi myös lähes kuka tahansa luoda toteutuksia juuri tämän avoimuuden takia. Avoimen lähdekoodin projektin omistajat tarkistavat ja hyväksyvät tehdyt muutokset ennen niiden siirtämistä varsinaiseen ladattavaan projektiin. Lähdekoodin avoimuus auttaa myös selaimen puuttuvien ominaisuuksien toteuttamisessa muissa selaimissa, sillä ohjelmoija voi tarkastella suoraan koodista tiettyjen ominaisuuksien toteutustapoja. Myös tämän ansiosta voidaan edistää laajempaa tukea ominaisuuksille selainten välillä.

Jos ohjelmoija käyttää suoraan tai saa inspiraatiota muiden tekemistä koodinpätkistä, tulisi aina tarkistaa kyseisen koodin lisenssi ja noudattaa siinä määriteltyjä vaatimuksia. Koodilisenssi voi myös estää koodin käytön muissa projekteissa. Lisenssin noudattamatta jättäminen voi johtaa jopa oikeudellisiin toimiin, joten ehtojen seuraaminen kannattaa ottaa aina vakavasti.

2.2 Google Chrome

Google Chrome on Googlen kehittämä verkkoselain, joka on tällä hetkellä selaimista eniten käytetty [3].

Alun perin Chromen selainmoottorina toimi Applen WebKit-moottori, mutta selaimen moottori vaihdettiin myöhemmin WebKitistä forkattuun Blink-moottoriin avoimen lähdekoodin Chromium-selainprojektin kehityksen yhteydessä. Chromium on avoimen lähdekoodin verkkoselain, johon perustuen Google rakensi oman selaimensa. Tästä huolimatta Google Chrome ei ole avoimen lähdekoodin selain.

Yksi suurimmista Google Chrome -selaimen hyödyistä verkkoselainten kehityksessä on sen laaja käyttäjäkanta. Kun kehitetyn sivuston toimivuus varmistetaan Chromessa, voidaan jo todeta verkkosivuston toimivuus suurimmalla osalla käyttäjistä. Etuna Chromelle kehittämisessä on myös hyvin läheinen rakenne Mozilla Firefox -selaimeen. Useimmiten Chromessa kehitetyt sivustot toimivat suoraan Firefoxilla. Jos tämä ei toteudu, voidaan yhteensopivuus taata usein hyvinkin nopeilla korjauksilla.

Google Chromen tärkeimpiä ominaisuuksia verrattuna muihin selaimiin ovat muun muassa sen laajat kehitystyökalut. Muissakin selaimissa on lähes aina jonkinlaiset kehitystyökalut, mutta Chromen kehitystyökaluja pidetään usein kaikista helpoiten käytettävinä ja laajoina. Verkkosivujen kehityksessä käytetään hyvinkin paljon näitä kehitystyökaluja ja niiden hyvä toimivuus on erittäin tärkeää. Chromelle etuja verkkosivujen kehityksessä antaa sen uutuus, jonka ansiosta suurin osa kaikista uusimmista ominaisuuksista on tuettuina hyvinkin laajasti. Uutuus myös vähentää niin sanottuja legacy-ongelmia, jotka aiheuttavat muissa selaimissa ongelmia nykypäivänäkin.

2.3 Mozilla Firefox

Mozilla Firefox on Mozilla Corporationin kehittämä verkkoselain, joka on yksi suosituimpien joukossa. Selain julkaistiin jo vuonna 2002, ja se on kokenut hyvinkin suuria muutoksia sen kehityksen aikana erityisesti sen ulkonäköön liittyen.

Firefoxia pidetään usein muita selaimia parempana käyttäjän yksityisyyden huomioinnissa ja juuri yksityisyys mainitaan usein Firefoxin mainostuksessa. Selaimessa arvostetaan usein myös sen suoritustehokkuutta esimerkiksi Google Chromeen verrattuna.

Verkkokehityksessä Firefox ei eroa paljoakaan Google Chrome -selaimesta ominaisuuksiltaan. Samanlaisuudestaan huolimatta Chromessa on joitakin Firefoxista puuttuvia ominaisuuksia, kuten jotkut CSS-toiminnallisuudet. Esimerkiksi mobiililaitteisiin liittyvä text-size-adjust CSS-komento ei toimi lainkaan Firefoxia käyttäessä, ja ominaisuudelle joudutaan kehittämään vaihtoehtoinen toteutus. Caniuse.com -sivuston mukaan Mozilla Firefoxista puuttuu Google Chromeen verrattuna hyvin pieni määrä vain vähemmän käytettyjä CSS-, HTML- ja JavaScript-ominaisuuksia. [8.]

2.4 Internet Explorer

Internet Explorer on Microsoftin kehittämä verkkoselain, jonka ensimmäinen versio julkaistiin jo vuonna 1995. Selaimen iän takia se on jäänyt erittäin paljon jälkeen nykyaikaisista teknologioista ja ohjelmoinnin ominaisuuksista. Tämän takia verkkosivustojen kehittäjille aiheutuu usein ongelmia erityisesti yhteensopivuutta käsitellessä. Ohjelmointiprojekteissa usein Internet Explorer jätetään tarkoituksellisesti pois tuetuista käyttöympäristöistä, sillä modernien ominaisuuksien puutteet aiheuttavat usein päänsärkyä kehittäjille. Jos selainta halutaan tukea, hyvinkin monelle nykyaikaisten selainten toiminnallisuuksille joudutaan luomaan alusta lähtien vaihtoehtoiset toteutukset.

Toisaalta Internet Explorer -selainta käytetään senkin takia, että useat vanhemmat verkkosivustot eivät välttämättä aina toimi tai eivät ole täysin tuettuina kaikista uusimilla laitteilla ja selaimilla. Näiden vanhojen sivustojen käyttö uudemmilla selaimilla vaatisi vuorostaan uudempien koodivaihtoehtojen käyttämistä ominaisuuksien toimivuuden varmistamiseksi. Ajan ja rahan säästämiseksi usein tätä uudistamista joko lykätään mahdollisimman pitkään tai jätetään kokonaan tekemättä.

Statcounter.com-sivuston mukaan vuoden 2022 helmikuussa tietokoneen käyttäjistä edelleen 1,14 % käytti Internet Exploreria verrattuna esimerkiksi Firefox-selaimen 9,46 % [3]. Tulevaisuudessa Internet Explorer todennäköisesti menettää yhä merkittävyyttään, kun verkkosivustojen käyttö siirtyy jatkuvasti enemmän mobiililaitteille. Microsoftin uusimmassa käyttöjärjestelmässä Windows 11:ssa Internet Exploreria ei enää tarjota lainkaan valmiiksi asennettuna, mutta Microsoftin uusi selain Microsoft Edge mahdollistaa Internet Explorer -tilan käytön tarvittaessa. Microsoft lopettaa tulevaisuudessa kokonaan tuen Internet Explorerille, joka tulee todennäköisesti myös vähentämään selaimen käyttäjäkantaa ja samalla helpottamaan ohjelmoijan optimointityötä.

3 Yhteensopivuusongelmien havaitseminen

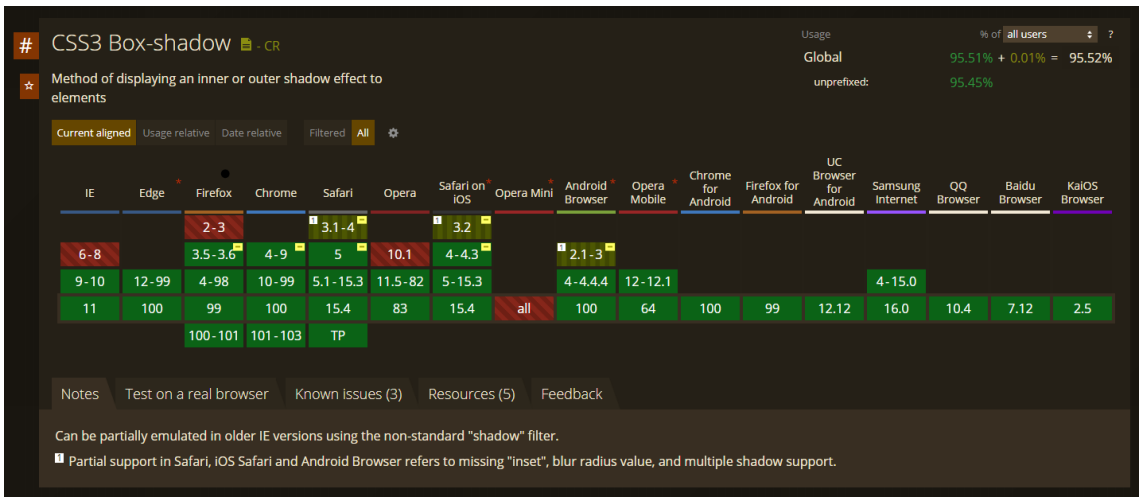
Yhteensopivuuden varmistamisessa on tärkeää löytää itse ongelmat niiden korjaamiseksi. Usein helpoin tapa löytää yhteensopivuusongelmia on testata luotua sivustoa eri selaimilla ja laitteilla. Jos kehittäjälle ei ole mahdollista testata usealla eri laitteella tai selaimella, voidaan myös käyttää apuna alustojen emulaatiota. Emulaatiolla simuloidaan laitteiden tai selainten toimintaympäristöä, jolloin näiden ongelmat voidaan löytää ja korjata. Emulaatio ei ole kuitenkaan täysin varma tapa testaamiselle, sillä emulaattorin ja oikean laitteen välillä voi esiintyä joitakin eroja erityisesti itse laitteiston kyvykkyyksissä.

Sivustojen yhteensopivuuden testausta kannattaa tehdä myös itse selainten lisäksi mahdollisimman monella erilaisella laitteella, jotta myös laitekohtaiset on-

gelmat saadaan havaittua mahdollisimman laajasti. Jos kehittäjällä ei kuitenkaan ole mahdollisuutta päästä käsiksi eri laitteisiin, voidaan sivustoa testata edellä mainittuun tapaan käyttöjärjestelmillä emulaatiota käyttämällä esimerkiksi virtuaalikoneen kautta. Virtuaalikoneella tarkoitetaan ohjelmistoa, jolla voidaan saada tämänhetkinen käytössä oleva tietokone esittämään ohjelmallisesti toista fyysistä tietokonetta, joka saattaa sisältää esimerkiksi eri käyttöjärjestelmän tai huonommat laitetehot [9].

Usein sivuston halutaan toimivan myös mobiililaitteilla, jolloin niillä testaaminen on myös hyödyllistä. Jos mobiililaitteilla testaamiseen ei ole mahdollisuuksia tai aikaa, voidaan myös käyttää selainten, kuten Google Chromen, kehitystyökaluja. Näitä kehitystyökaluja käyttämällä voidaan tarkastella sivuston ulkoasua ja toiminnallisuutta eri ruudunleveyksissä ja -korkeuksissa.

HTML-, CSS- ja JavaScript-toiminnallisuuksien yhteensopivuutta voidaan tarkastella jo ennen toteuttamista esimerkiksi caniuse.com-sivustolta [10]. Kyseisellä sivustolla voidaan hakea ominaisuuksia nimellä, jolloin sivusto esittelee taulukon eri selainten tuesta haetulle ominaisuudelle. Sivustolla on usein myös ilmoitettu vaihtoehtoinen toteutus eri selaimiin niissä tapauksissa, joissa ominaisuus on puutteellinen tai kokonaan kadoksissa. Esimerkiksi CSS:n box-shadow-tyylimäärittelyllä hakemalla saadaan sivustolta kuvassa 3 esiintyvät tulokset.



Kuva 3. CSS:n box-shadow-tyylimäärittelyn yhteensopivuustaulukko [10].

Kuvassa on esitelty vierekkäin selainten tuki ominaisuudelle. Pystysuunnassa olevissa laatikoissa vihreä väri ilmoittaa tuen löytymisestä, kun taas punainen väri sen puuttumisesta. Kellertävä väri puolestaan tarkoittaa osittaista tukea ominaisuudelle. Numerot laatikoiden keskellä tarkoittavat selaimen versioita.

Kyseinen sivusto sisältää myös mahdollisuuden selainten kokonaisvaltaiselle vertailulle Compare browsers -painikkeesta. Tätä ominaisuutta käyttämällä voidaan nopeasti tulkita, mikä selain sisältää parhaimman tuen millekin ominaisuuksille.

Yhteensopivuusongelmien havaitsemisessa voidaan myös käyttää apuna pilvessä toimivia sivustontestausalustoja. Tällaiset sivut usein mahdollistavat verkkosivun nopean avaamisen laajassa selainten ja laitteiden kirjossa hyvinkin nopeasti. Näin voidaan nopeasti ja tehokkaasti testata kaikki tarvittavat sivuston käyttökohteet. Tosin usein kyseisten palveluiden käyttäminen ei ole mahdollista pienemmille yrityksille tai kehitystiimeille, sillä näiden sivustojen käyttäminen vaatii useimmiten jonkinlaisen maksun niiden käytöstä.

4 Yhteensopivuuden yleisiä ongelmia

Yleisimpiä ongelmia selainyhteensopivuudessa ilmenee useimmiten CSS- ja JavaScript-toiminnallisuuksissa. Nämä voi havaita usein helposti nopealla testauksella eri alustoilla. Havaitsemisen jälkeen tulee keksiä jonkinlainen kiertotie toiminnallisuuden toteuttamiselle puutteellisissa alustoissa. Tähän toteutukseen voi käyttää usein vaihtoehtoisia tyylejä tai ohjelmointia, mutta joihinkin ominaisuuksiin on olemassa myös valmiita toteutuksia esimerkiksi Polyfill-toteutusten muodossa. Polyfill on koodinpätkä tai koodikirjasto, jolla luodaan toiminnallisuus selaimelta puuttuville ominaisuuksille.

Joissain tapauksissa vaihtoehtoista toteutusta esimerkiksi JavaScript ominaisuuksille ei ole mahdollista toteuttaa. Tällaisissa tapauksissa ominaisuudelle

kannattaa kehittää alkuperäistä muistuttava yksinkertaisempi versio. Esimerkiksi jos jokin animaatio ei toimisi jossain selaimessa, se voitaisiin mahdollisesti jättää kokonaan pois tai siitä voitaisiin tehdä yksinkertaisempi toimiva versio.

4.1 HTML

Yleisimpiä ongelmia HTML-yhteensopivuudessa ovat sellaisten elementtien tai elementtien ominaisuuksien käyttö, joita ei tueta muissa selaimissa.

Kuten kuvasta 3 nähdään, yksittäiset HTML-elementit voivat myös sisältää yksittäisiä ominaisuuksia tai attribuutteja, jotka eivät ole tuettuina kaikissa selaimissa. Myös näihin ominaisuuksiin on mahdollista luoda oma toteutus tai käyttää valmiita polyfillejä.

<textarea>	Yes	Yes	Yes	Yes	Yes
autocomplete	No	No	59.0	13.0	No
autofocus	Yes	10.0	4.0	Yes	Yes
cols	Yes	Yes	Yes	Yes	Yes
dirname	Yes	79.0	No	Yes	Yes
disabled	Yes	Yes	Yes	Yes	Yes
form	Yes	11.0	Yes	Yes	Yes
maxlength	Yes	10.0	4.0	Yes	Yes
minlength	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
placeholder	Yes	10.0	4.0	5.0	11.5
readonly	Yes	Yes	Yes	Yes	Yes
required	Yes	10.0	4.0	Yes	Yes
rows	Yes	Yes	Yes	Yes	Yes
spellcheck	Yes	11.0	Yes	Yes	Yes
wrap	Yes	Yes	Yes	Yes	Yes

Kuva 3. Yhteensopivuustaulukko textarea-elementistä [11].

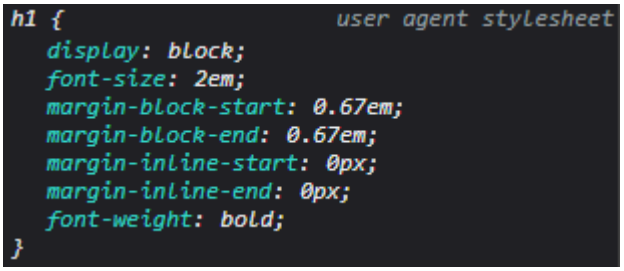
Kuvan yläosassa on käsiteltyjen verkkoselainten logot, joiden alla on pystysuunnassa selitys vasemmalla ilmoitetun ominaisuuden tuesta kyseisessä selaimessa. Tuki ominaisuudelle on Yes eli kyllä, No eli ei, tai versionumero, jolla kerrotaan, mistä verkkoselaimen versiosta lähtien ominaisuus on tuettuna.

Luodun verkkosivuston koodisyntaksi voi joissakin selaimissa aiheuttaa ongelmia. Tämä johtuu usein siitä, että nykyaikaisemmat selaimet korjaavat automaattisesti suoritusvaiheessa syntaksivirheet, kun taas vanhemmat selaimet eivät tätä osaa tehdä. Verkkosivuston koodisyntaksin pystyy tarkistamaan usealla sivustolla, kuten esimerkiksi https://validator.w3.org/-sivustolla. Kyseinen sivusto ilmoittaa ohjelmoijalla sivustolta ilmenevät mahdolliset ongelmat.

Usein ongelmia vanhemmissa selaimissa aiheuttaa HTML-tiedoston alusta puuttuva tai vääränlainen DOCTYPE-tunniste. DOCTYPE-tunnisteella ilmoitetaan verkkoselaimelle, minkä tyyppisenä tiedostona sen tulisi tulkita seuraava tiedosto. Esimerkiksi vanhemmat Internet Explorer -versiot tarvitsevat kyseistä tunnistetta sivuston oikeanlaiseen kuvantamiseen.

4.2 CSS

Eri selaimet sisältävät useimmin oletuksena oman suunnittelupohjansa, jonka takia luotu sivusto saattaa näyttää hieman erilaiselta eri selaimilla. Kuten kuvasta 4 nähdään, suunnittelupohjan oletusmääritykset voivat esimerkiksi sisältää margin-, display- tai font-määrityksiä eri elementeille.



Kuva 4. Google Chrome -selaimen oletusmääritykset h1-otsikkoelementille.

Kuvassa on listattu allekkain h1-elementin CSS-tyylimäärityksiä, joille selain antaa oletusmääritykset oman oletustyylipohjansa mukaan. Tyylimääritykset asettavat elementin näkyvyydeksi (engl. display) block-tyypin, joka asettaa elementin alkamaan uudelta riviltä ja viemään kokonaisen rivin verran tilaa leveyssuunnassa. Fonttikoko (engl. font-size) on 2em, eli kaksinkertainen tämänhetkisen

kirjasinkoko. Marginaalimäärityksillä (engl. margin) voidaan asettaa nimensä mukaisesti marginaalia elementtiin. Fontin painolla (engl. font-weight) bold vaihdetaan fontti lihavoiduksi.

Oletusmääritysten takia HTML-dokumentissa tulisi käyttää alussa nollaukseen tai standardoimiseen tarkoitettua tyyliohjetta (engl. stylesheet). Nämä tyyliohjeet varmistavat, että selaimen omat oletustyylimääritykset eivät sekoitu luodun verkkosivuston tyylimääritysten kanssa. Yksi esimerkki tällaisesta tyyliohjeesta on Normalize.css. Kyseinen tyyliohje on käytössä esimerkiksi Twitterin, GitHubin ja Soundcloudin verkkosivustojen oletustyylien nollauksessa. Kun selaimen oletustyylit on nollattu, voidaan eri elementeille luoda omat oletustyylit. Esimerkiksi h1-otsikkoelementille voidaan itse asettaa fonttikoko ja tekstinpaksuus.

Kaikki CSS-tyylimääritykset eivät aina toimi kaikissa selaimissa, jolloin niiden käyttöä tulisi välttää tai niille tulisi luoda vaihtoehtoiset tyylit erityisesti puutteellisia selaimia silmällä pitäen. [12, luku 2.]

4.3 JavaScript

Eniten ongelmia JavaScriptin yhteensopivuudessa aiheuttaa tuen puute kielen ominaisuuksille. Varsinkin vanhemmissa selaimissa kaikista uusimpiin toiminnallisuuksiin ei usein ole toteutusta. Tällaisissa tilanteissa on hyvä käyttää apuna polyfillejä.

Ohjelmointikirjastojen käyttö voi myös olla syypäänä yhteensopivuusongelmiin, minkä takia käytössä olevista ohjelmointikirjastoista tulisi tarkistaa aina niiden tuki eri selaimille.

JavaScript -koodin yhteensopivuusongelmia voidaan myös ehkäistä kääntämällä koodi yhteensopivaan muotoon. Tämä voidaan tehdä käyttämällä esimerkiksi Babel-nimistä verkkosivustoa, joka muuttaa modernejakin ominaisuuksia käyttävän JavaScript-koodin mahdollisimman yhteensopivaksi versioksi.

5 Ongelmien korjaamisen menetelmiä

Kun verkkosivustolta on löydetty yhteensopivuusongelmia, ne tulee korjata. Näiden ongelmien korjaamiseen on olemassa useita keinoja. Tässä osiossa esitellään joitakin näistä keinoista.

5.1 Polyfill

Polyfillillä tarkoitetaan koodia, jolla luodaan selainkohtaisia määrityksiä. Näillä määrityksillä toteutetaan selaimelta puuttuvia ominaisuuksia. Polyfill-termillä voidaan myös viitata itse JavaScript-kirjastoihin, jotka sisältävät useampia valmiiksi kehitettyjä vaihtoehtoisia ohjelmointikokonaisuuksia eri ominaisuuksille. Polyfill-kirjastoja voidaan asentaa osaksi projektia esimerkiksi käyttämällä Node Package Manageria eli NPM:ää.

5.2 Lint-ohjelmat

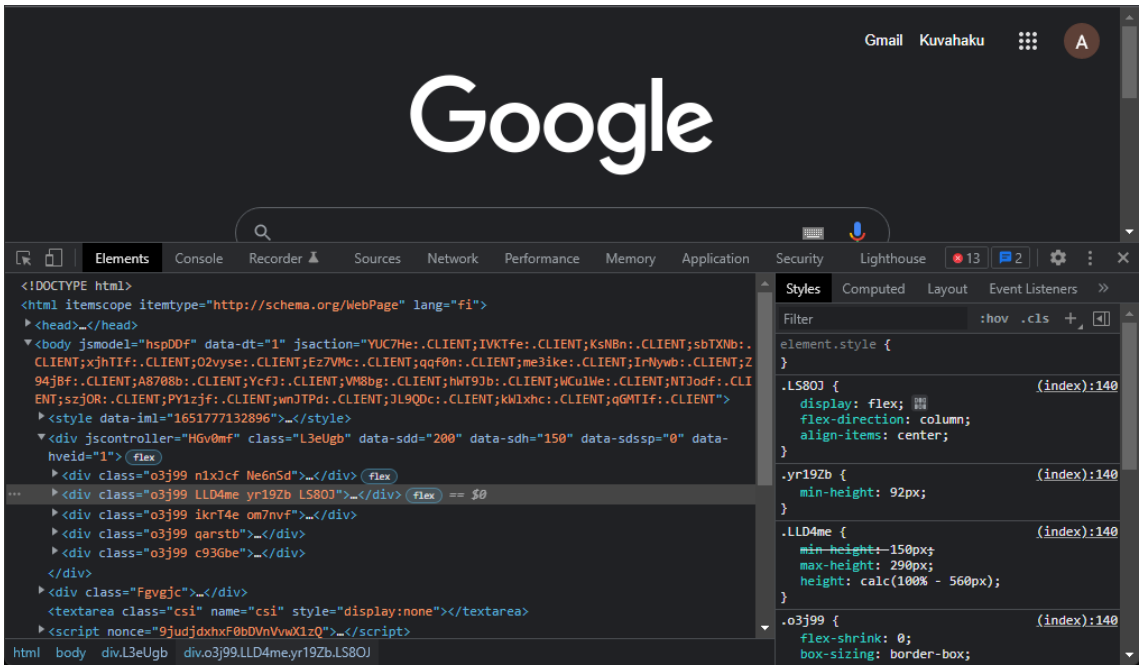
Ohjelmistokehityksessä käytetään usein niin kutsuttuja Lint-ohjelmia. Nämä ohjelmat muun muassa tarkistavat ja ilmoittavat koodin mahdollisista eroavaisuuksista ennalta-asetettuihin sääntöihin.

Myös selainten yhteensopivuuteen on olemassa Lint-ohjelmia tai Lint-ohjelmien liitännäisiä. Esimerkiksi eslint-ohjelman liitännäinen caniuse kertoo ohjelmoijalle koodin yhteensopivuusongelmista. Myös näitä lisäosia voidaan ladata esimerkiksi aiemmin mainitulla NPM:llä.

5.3 Selainten kehitystyökalut

Ehkä jopa kaikista tärkeimmät työkalut verkkosivujen kehityksessä ovat selaimen kehitystyökalut. Lähes kaikki nykyaikaiset verkkoselaimet sisältävät ne valmiiksi sisäänrakennettuina. Kehitystyökalujen tarkoituksena on helpottaa verkkosivustojen kehitystyötä ja mahdollistaa helppo debuggaus eli virheiden etsiminen ja korjaaminen.

Eri selainten kehitystyökalut ovat ominaisuuksiltaan hyvin samankaltaisia. Tässä osiossa keskitytään pääosin Google Chromen kehitystyökaluihin, jotka näkyvät kuvan 5 alalaidassa.



Kuva 5. Näkymä Google Chrome -verkkoselaimesta ja sen kehitystyökaluista.

Verkkosivustojen kehittäjät suosivat useimmiten virheiden etsimisen käytössä juuri Google Chrome -selainta. Selaimen laajan käyttäjäkannan takia Chromella toimivat sivustot tulevat todennäköisesti toimimaan suurimmalla osalla käyttäjistä. Chromella on myös caniuse.com-sivuston mukaan laajin tuki eri HTML-, CSS- ja JavaScript-ominaisuuksille [3].

Kehitystyökalujen Elements-välilehdeltä löydetään itse sivuston HTML-rakenne. Kyseisestä näkymästä nähdään jokainen HTML-tiedostoon asetettu HTML-komponentti sisennettyinä elementteinä sivuston rakenteessa. Näkymästä voidaan myös sivuston suorituksen aikana muokata suoraan komponentteja ja testata uusia toteutuksia. Nämä testitoteutukset voidaan siirtää lähes suoraan koodiin kopioimalla ne esimerkiksi Copy element -toiminnon avulla.

Select an element -painiketta käyttämällä voidaan valita sivustolta mikä tahansa näkyvä elementti, jolloin kehitystyökalut näyttävät kyseisen elementin elements-välilehdellä sivuston hierarkiassa. Elementistä voidaan nähdä myös samalla voimassa olevat tyylit, ja niitä voidaan myös muokata sivuston suorituksen aikana. Kun oikeanlaiset tyylit on löydetty, ne on helppo siirtää työkaluista suoraan sivuston koodiin.

Luotujen verkkosivujen JavaScript-koodista voidaan korjata ongelmia helpommin asettamalla välitulostuksia tiettyihin kohtiin console.log()-funktiota käyttäen. Kun ohjelmoija laittaa koodiinsa kyseisiä tulostuksia, ne ilmaantuvat suorituksen aikana kehitystyökalujen Console-välilehdelle. Kyseisellä komennolla voidaan myös tulostaa muuttujia suorituksen eri hetkillä, jolloin muuttujista ilmaantuvia ongelmia on helpompi korjata. Console-välilehden tekstikenttään voidaan myös kirjoittaa suoraan uusia JavaScript-funktioita ja tätä kautta testata niitä ennen varsinaiseen koodiin lisäämistä.

Recorder-välilehdeltä voidaan tallentaa käyttäjän tekemät toimet verkkosivustolla ja myöhemmin toistaa ne. Tallennetun prosessin suorituskykyä voidaan myös mitata tästä näkymästä. Tallennuksia voidaan luoda useampi ja niitä voidaan käyttää eräänlaisena automaatiotestauksena uusien luotujen muutosten jälkeen.

Sources-välilehdeltä nähdään verkkosivuston lataamat resurssit. Näitä ovat esimerkiksi sivustolle ladatut kuvat ja koodia sisältävät tiedostot.

Network-välilehdeltä voidaan tarkastella sivuston lataukseen liittyviä tietoja. Sources-välilehden resurssit ja niiden latausaika näkyvät myös tällä välilehdellä. Latauksia voidaan myös estää tarvittaessa. Näkymästä voidaan myös asettaa sivusto offline-tilaan. Esimerkiksi sivustolla olevaa kaavaketta testattaessa voidaan haluta estää kaavakkeen lähetys, jotta ohjelmoijan täyttämät kuvitteelliset kaavakkeen tiedot eivät lähde sivustolta eteenpäin.

Performance-välilehdeltä voidaan tarkastella halutuista ajankohdista sivuston suorituskykyä ja resurssien latausta. Memory-välilehteä voidaan käyttää apuna

tietokoneen muistiongelmien ratkaisemisessa. Välilehteä käytetään esimerkiksi muistivuotojen tai hitaan suorituskyvyn syiden etsimisessä.

Application-välilehti sisältää sivuston tallennukseen liittyviä asioita. Välilehdeltä voidaan tarkastella keksejä (engl. Cookies), paikallismuistia (engl. Local Storage) ja istuntomuistia (engl. Session Storage). Näihin voidaan tallettaa sivuston tietoja, jotta kävijä olisi esimerkiksi sisäänkirjautuneena seuraavallakin vierailukerralla. Security-välilehdeltä löydetään nimensä mukaisesti turvallisuuteen liittyviä asioita.

Lighthouse-välilehdeltä löytyy Googlen luoma avoimen lähdekoodin ohjelma, jolla voidaan analysoida verkkosivuston suorituskykyä. Ohjelma esittää esimerkiksi sivuston mahdolliseen hitaaseen lataukseen vaikuttavia ongelmia. Lighthouse antaa myös sivustolle arvosanan löytämiensä tietojen perusteella. Ohjelma voidaan ajaa joko työpöytä- tai mobiilimoodissa. Lighthouse löytyy esimerkiksi Google Chrome- ja Microsoft Edge -selaimista, mutta se ei ole sisäänrakennettuna kaikkiin selaimiin. Mozilla Firefox -selaimesta se puuttuu, mutta työkalun saa ladattua selaimelle ulkoisena laajennuksena.

Toggle device toolbar -painikkeesta aukeavasta näkymästä voidaan muuttaa sivuston ikkunan kokoa ja simuloida eri laitteiden huonompaa suorituskykyä tai jopa internetyhteyden puuttumista. Näkymä sisältää myös valmiita ikkunan kokomäärityksiä eri laitteille, mikä helpottaa verkkosivuston optimointia juuri tietyille laitteille. Näkymän sivuston koon muuttaminen on erityisen hyödyllinen varsinkin mobiililaitteiden testauksen yhteydessä.

Näkymää hyödyntäen voidaan etsiä tietty sivuston leveys tai korkeus, jossa jotkin ominaisuudet lakkaavat toimimasta. Tähän leveyteen tai korkeuteen voidaan asettaa esimerkiksi CSS-koodin niin kutsuttu esimerkkikoodissa 2 esiintyvä @media-määritys, jota käyttämällä voidaan asettaa muun muassa leveys- tai korkeuskohtaisia tyylimäärityksiä.

```
@media only screen and (max-width: 483px) {
  div {
    background-color: green;
  }
}
```

Esimerkkikoodi 2. Ruudun leveyden havaitseminen @media-komennolla.

Esimerkkikoodissa on luotu määritys div-elementtien taustavärin vaihdolle, kun sivuston leveys on 483 pikseliä leveä tai sitä pienempi.

5.4 Selainkohtaiset määrittelyt

Kun verkkosivun sisällöstä halutaan tehdä yhteensopivaa, apuna voidaan käyttää täysin erillisiä toimivuuksia. Vaihtoehtoisia toteutuksia voidaan luoda muun muassa HTML-, CSS- ja JS-koodilla. Esimerkkinä tyylin vaihtoehtoisesta määrittelystä kuvassa 6 on korjattu aiemmin esitetty kuvan 1 yhteensopivuusongelma Google Chrome- ja Internet Explorer -selainten välillä.

Kuva 6. Internet Explorer -selaimen (oik.) puuttuva toiminnallisuus korjattu. Vasemmalla Google Chrome -selain.

Kuvassa olevat elementit saatiin muistuttamaan toisiaan molemmilla selaimilla havaitsemalla Internet Explorer -selaimen käyttö ja asettamalla kyseiseen selaimeen omat vaihtoehtoiset tyylimääritykset.

Jotta yhteensopivuusongelman voi korjata, tulee ensin pystyä havaitsemaan käyttäjällä käytössä oleva verkkoselain. Esimerkkikoodissa 3 esitellään yksi tapa selvittää tämä tieto JavaScript-koodia hyväksikäyttäen.

```
function isIE(){
     return window.navigator.userAgent.match(/(MSIE|Trident)/);
}

if (isIE()) {
     //ieFunction();
} else {
     //otherBrowserFunction();
}
```

Esimerkkikoodi 3. Internet Explorer -selaimen havaitseminen JavaScript-funktiolla [13.]

Koodissa katsotaan selaimen tiedoista, sisältääkö se maininnan MSIE- tai Trident-termeistä. Nämä termit viittaavat Internet Explorerin selainmoottoriin.

Verkkosivun käyttäjän verkkoselain voidaan havaita myös CSS-määrittelyjä käyttämällä esimerkkikoodin 4 tavoin.

```
@media all and (-ms-high-contrast: none), (-ms-high-contrast: active){
     .otherBrowser {
          display: none;
     }

     .ie10 {
          display: block;
     }
}
```

Esimerkkikoodi 4. Internet Explorer -selaimen havaitseminen CSS-media komennolla [14].

Koodissa etsitään selaimesta vain Internet Explorer -selaimessa esiintyviä määrityksiä, jolloin niiden löytyessä suoritetaan sitä seuraavat CSS-tyylit. Esimerkissä sivulta piilotetaan tai näytetään elementtejä riippuen selaimesta, jolla sivusto avataan.

Myös HTML-koodilla on oma roolinsa yhteensopivuuden varmistamisessa muissa selaimissa. Edellisen kappaleen CSS-toteutusta apuna käyttäen esimerkkikoodissa 5 näytetään Internet Explorer -selaimella img-kuvatyypin sijasta

svg-kuvatyyppi. Tässä kuvatyypissä voi luoda muiden selainten CSS-sumennusominaisuutta muistuttavan vaihtoehdon, joka toimii myös Internet Explorer -selaimella.

```
<div class="otherBrowser">
     <img
     src="https://upload.wikimedia.org/wikipedia/fi/6/61/Metropolia_Amm
     attikorkeakoulu_logo.svg">
</div>

<div class="ie10">
     <svg xmlns=http://www.w3.org/2000/svg
     xmlns:xlink="http://www.w3.org/1999/xlink">
          <filter id="svgBlurFilter">
               <feGaussianBlur in="SourceGraphic" stdDeviation="5" />
          </filter>
          <image
          xlink:href="https://upload.wikimedia.org/wikipedia/fi/6/61/Me
          tropolia_Ammattikorkeakoulu_logo.svg" width=246 height=127
          filter="url(#svgBlurFilter)" />
     </svg>
</div>
```

Esimerkkikoodi 5. Vaihtoehtoinen tapa toteuttaa kuvan sumennus Internet Explorer -selaimella [15].

Koodissa on allekkain kaksi div-elementtiä, joiden näkyvyyttä hallitaan luokkaa (engl. class) käyttäen. OtherBrowser-luokan div-elementti näytetään, kun käyttäjän selain ei ole Internet Explorer. Alempi div-elementti näytetään ylemmän tilalla Internet Explorerilla. Svg-tyyppiselle kuvalle on asetettu oma erillinen suodatinelementti, joka luo sumennuksen kuvaan.

5.5 Tyylikirjastojen selainyhteensopivuus

Verkkosivujen kehityksessä käytetään usein apuna valmiita tyylikirjastoja, jotka saattavat sisältää valmiiksi ohjelmoituja selainyhteensopivuuden mahdollistavia toimintoja. Näiden käyttö voi säästää paljonkin aikaa ohjelmoijalta mahdollisten ongelmien etsinnältä, sillä mahdolliset ongelmat ovat usein jo muiden kehittäjien korjaamia. Tästä huolimatta verkkosivun kehittäjän kannattaa aina tarkastaa sivun yhteensopivuus myös itse.

Material UI on yksi näistä tyylikirjastoista, jota käytetään React JS -ohjelmistokehyksen yhteydessä. Kyseinen tyylikirjasto sisältää useita valmiiksi kehitettyjä elementtejä, joita voidaan käyttää verkkosivulla. Näihin elementteihin on usein sisäänrakennettu yhteensopivuus suosituimmille selaimille.

6 Esimerkki sivuston ongelmista ja niiden korjaamisesta

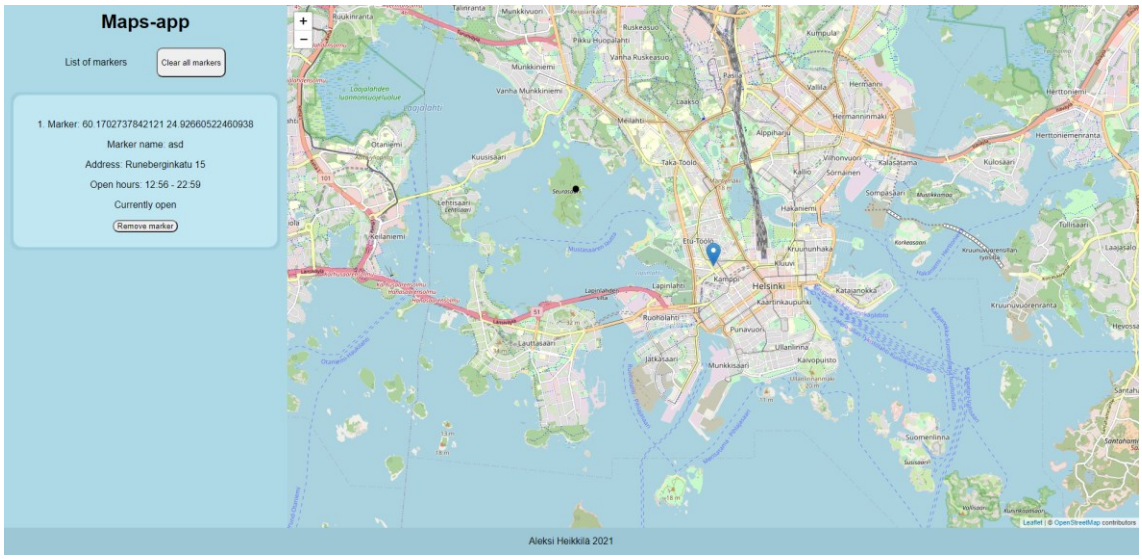
Selainyhteensopivuusongelmien havainnollistamiseksi työssä tutkitaan aiemmin luotua verkkosivua, jossa esiintyy erilaisia yhteensopivuusongelmia eri selainten ja alustojen välillä. Verkkosivu luotiin aiemmin osana työnhakuprosessia. Tässä osiossa pyritään etsimään manuaalitestauksella kyseiseltä verkkosivulta suurin osa yhteensopivuusongelmista ja korjaamaan ne mahdollisimman tehokkaasti. Prosessin jälkeen verkkosivun tulisi toimia mahdollisimman monessa eri verkkoselaimessa ja laitteessa.

Esimerkkisivusto on luotu käyttämällä HTML-, CSS- ja JavaScript-teknologioita. Myös JavaScriptin React JS -ohjelmistokehystä käytettiin projektissa.

Kuvan 2 pylväsdiagrammiin viitaten sivuston yhteensopivuus päätettiin varmistaa kolmessa top 5 -käytetyimmistä Windows-käyttöjärjestelmän työpöytäselaimista. Nämä selaimet olivat Google Chrome, Microsoft Edge ja Mozilla Firefox. Näiden lisäksi myös Internet Explorer -selain otettiin mukaan testaamiseen, jotta edes joitain ongelmia ilmaantuisi ja täten korjaamista olisi helpompi havainnollistaa. Viidenneksi suosituin selain Samsung Internet otettiin mukaan mobiiliselaimen ja -laitteen ongelmien korjaamisen esittelyksi.

6.1 Esimerkkisivun esittely

Esimerkkiverkkosivu on kuvan 7 Maps App, jossa voidaan asettaa interaktiiviselle kartalle useita merkkejä.



Kuva 7. Maps App -sivuston perusnäkymä.

Kuvassa esiintyy Maps App -verkkosivu, jonka päätarkoituksena on mahdollistaa merkkien lisääminen kartalle. Merkeille voidaan asettaa nimi ja aukioloaika. Luodut merkit ilmaantuvat ruudun vasempaan reunaan. Merkkejä napsauttamalla merkkilistauksesta voidaan siirtää ruudun näkymä napsautetun merkin kohdalle. Merkkejä voidaan myös poistaa tästä näkymästä.

6.2 Ongelmien havaitseminen

Osana testaamista oletettiin sivuston toimivan ilman yhteensopivuusongelmia Google Chrome -selaimessa, sillä sivusto kehitettiin juuri kyseistä selainta apuna käyttäen. On siis täten melko turhaa testata sivua yhteensopivuusongelmilta käyttämällä sen kehitysselainta, sillä kyseisessä selaimessa ongelmat on jo korjattu kehitysvaiheessa. Poikkeuksena tähän ongelmien havaitsemiseen ja ratkaisemiseen voidaan käyttää apuna Chromen kehitystyökaluja.

Ensimmäiseksi ongelmien havaitseminen aloitettiin avaamalla sivusto Google Chrome -selaimella, josta avattiin kehitystyökalut. Kehitystyökaluista avattiin device toolbar -näkymä, jota käyttämällä tarkistettiin sivuston toimivuus mobiililaitteissa. Tästä havaittiin heti tyyliongelmia sivuilla, jotka johtuivat mobiililaitteiden

pienemmästä ruudunleveydestä. Ongelmat johtuivat siitä, että sivuston elementit menivät kasaan puhelimen ruudunleveyden takia. Ongelmien ilmaantuvuus tarkastettiin myös oikealla mobiililaitteella, josta ne myös löytyivät. Kyseiset ongelmat lisättiin ongelmalistaan ja ongelmien etsinnässä siirryttiin eteenpäin.

Seuraavaksi aloitettiin varsinainen työpöytäselaimilla testaaminen. Ensiksi testaustietokoneelle ladattiin puuttuvia selaimia testauksen helpottamiseksi. Testattavalla tietokoneella oli jo valmiiksi asennettuina Google Chrome- ja Microsoft Edge -verkkoselaimet. Tietokoneelle asennettiin Mozilla Firefox -selain. Internet Explorer -selaimen käyttöä emuloitiin Microsoft Edgen Internet Explorer -tilaa apuna käyttämällä.

Mozilla Firefox -selaimella yritettiin lisätä uusi merkki kartalle, jolloin huomattiin pieni yhteensopivuusongelma. Merkinlisäysikkunassa on lisättävän merkin aukioloaikojen syöttämistä varten luodut kentät. Kyseisissä kentissä on Google Chromessa pieni kelloikoni, jota napsauttamalla voidaan valita kellonajat eräänlaisesta valikosta. Tämä ominaisuus puuttuu Mozilla Firefoxia käyttäessä. Tästä ongelmasta huolimatta sivusto vaikutti toimivan muuten samalla tavoin kuin Google Chrome -selaimessakin. Sivustolta testattiin vielä lopuksi istuntomuistilla (engl. Session Storage) toteutettua merkkien säilymistä sivuston päivityksien välillä. Sivustoa päivitettiin muutaman kerran ja asetettujen merkkien huomattiin säilyvän näiden sivunpäivitysten välillä.

Testaamisessa siirryttiin Microsoft Edge -selaimen käyttöön. Sivustolta testattiin kaikkia ominaisuuksia napsauttamalla painikkeita ja lisäämällä merkkejä kartalle. Kaikki ominaisuudet vaikuttivat toimivan samalla tavoin kuin Google Chrome -selaimessa jopa ajanvalitsinta myöten. Microsoft Edge perustuu Chromium-selaimeen Google Chromen tavoin. Koska sivusto on kehitetty alun perin Chromella, yhteensopivuus sivuston kanssa on hyvä. Myös Edgessä istuntomuisti toimi halutulla tavalla.

Seuraavaksi sivusto avattiin Microsoft Edgen Internet Explorer -tilassa. Tämän tilan saa avattua asettamalla selaimen asetuksista Allow sites to be reloaded in

Internet Explorer mode -asetuksen päälle. Tämän jälkeen itse sivustolla voidaan napsauttaa Edgen sivuvalikosta Reload in Internet Explorer mode -painiketta.

Sivusto aukesi selaimessa onnistuneesti, mutta sivustolta huomattiin heti puuttuvan joitakin tyylejä. Internet Explorerin näkymää verrattiin Google Chromen näkymään, jolloin puuttuvat tyylit löydettiin. Google Chromessa puuttuvien tyylien sisältämää elementtiä tarkasteltiin kehitystyökalujen avulla valitsemalla elementti inspect element -toiminnolla. Tämä tehtiin Google Chromessa, sillä Microsoft Edgen Internet Explorer -tilaa käytettäessä kehitystyökaluja ei pystytä käyttämään. Elementistä huomattiin puuttuvan background-color-määritys. Tyylimäärityksen yhteensopivuus tarkastettiin caniuse.com-sivustoa käyttämällä. Sivuston mukaan kyseisen tyylin tulisi toimia oikein, sillä se on kyseisessä selaimessa tuettuna. Ongelman olemassaolon varmistamiseksi sivusto avattiin vielä toisen tietokoneen oikeassa Internet Explorer -selaimessa, josta voidaan käyttää inspect element -toimintoa. Elementtiä tarkastelemalla havaittiin background-color-tyylimäärityksen puuttuvan kokonaan listatuista tyylimäärityksistä.

Ratkaisun etsinnässä selvisi, että React JS ei ole automaattisesti tuettuna Internet Explorer -selaimessa. Tämä ongelma päätettiin ratkaista ennen kuin edellisen ongelman ratkomista jatkettiin.

Seuraava ongelma ilmeni uutta karttamerkkiä lisättäessä. Näkymän tausta muuttui tummemmaksi niin kuin sen kuuluisi, mutta merkinlisäysikkuna ei ilmaantunut lainkaan ruudulle. Caniuse.com-sivuston mukaan tämä johtui inset-ominaisuuden puutteesta Internet Explorer -selaimesta. Inset-tyylimäärittelyllä voidaan asettaa top-, right-, bottom- ja left-tyylimääritykset yhdellä komennolla.

Seuraavaksi ongelmaksi ilmeni itse korjatussa laatikossa ilmaantuva scrollbar. Kyseinen elementti on laitettu pois päältä overflow-y-komennolla, ja se toimii oikein muissa selaimissa, mutta Internet Explorerissa kyseinen komento ei toimi, jos elementille ei ole asetettu korkeutta.

Internet Explorerissa myös ilmeni merkin luonnin yhteydessä, että input-elementin tyyppi time ei ole tuettuna kyseisessä selaimessa. Tämä aiheuttaa sen, että laatikoihin voi syöttää mitä tahansa tietoa ja verkkosivu hyväksyy sen ja lisää merkkilistaan. Tämä oli suuren prioriteetin ongelma, sillä se mahdollisti verkkosivun käytön tavalla, johon sitä ei ollut suunniteltu.

6.3 Ongelmien korjaamisen suunnittelu

Ensimmäiseksi havaittuun ongelmaan eli mobiililaitteiden yhteensopivuuteen tehtiin korjaus käyttämällä apuna CSS:n @media-komentoja, jotka mahdollistavat ruudunleveyden havaitsemisen. Kun ruudunleveys on havaittu, kyseiselle leveydelle voidaan asettaa työpöytäselaimen normaalista leveydestä poikkeavat tyylimääritykset. Mobiililaitteiden ruudunleveyttä varten luotiin oma painike, jota napsauttamalla voitaisiin näkymää vaihtaa listaus- ja karttanäkymien välillä.

Aikavalitsimen puuttuminen Firefox-selaimesta todettiin niin pieneksi ongelmaksi, että tässä vaiheessa kyseiselle ominaisuudelle ei luotu vaihtoehtoista toteutusta vain kyseisen selaimen käyttäjille. Ajan voi edelleen syöttää ikkunaan numeronäppäimiä käyttämällä. Verkkosivuprojektiin ladattiin React JS polyfill yhteensopivuuden varmistamiseksi Internet Explorerin kanssa.

Koska inset-komento puuttuu Internet Explorerista, vaihdettiin kyseisen komennon tilalle erillisenä kaikki neljä sijaintimääritystä. Myös Internet Explorerissa toimivat sijaintimääritykset olivat siis top, right, bottom ja left. Scrollbarin toimivuus korjattiin asettamalla korkeus ilmaantuvalle laatikolle.

Merkinlisäyslaatikon input-elementtien validoinnin ratkaisemiseen käytettiin säännöllisiä lausekkeita (engl. regular expression, lyh. regex). Ominaisuuden toimivuus varmistettiin jälleen käyttämällä caniuse.com-sivustoa, jonka jälkeen luotiin itse tarkistuslauseke.

6.4 Ongelmien varsinainen korjaaminen

Sivuston tavallinen sivupalkki karttamerkkejä varten piilotettiin käyttämällä CSS:n display-ominaisuutta, kun ruudunleveyden havaitaan olevan alle 1000 pikseliä leveä. Kun alkuperäinen sivupalkki saatiin pois tieltä pienessä ruudunleveydessä, luotiin koko ruudun kokoinen karttamerkkien listausnäkymä. Tämä toteutettiin luomalla uusi React JS -komponentti, joka on aluksi piilotettuna CSS:n z-index-ominaisuudella. Näkymänvaihtoa varten lisättiin myös painike, jota napsauttamalla listausnäkymän z-index vaihtuu itse karttaa korkeammaksi. Tällöin listaus ilmestyy käyttöliittymässä kartan päälle, ja itse kartta menee piiloon. Kun painiketta napsautetaan uudelleen, tapahtuu sivulla päinvastainen operaatio, ja listausnäkymä menee jälleen piiloon.

Jotta sivusto toimisi oikein Internet Explorer -selaimessa, ladattiin projektiin toimivuuden mahdollistava polyfill react-app-polyfill. Polyfillin lataus tehtiin Node Package Managerilla eli npm:llä. Jotta polyfill olisi käytössä, index.js-tiedoston alkuun asetettiin seuraava komento:

```
import 'react-app-polyfill/ie11';
```

Tämän jälkeen package.json -tiedostoon lisättiin browserlist-otsikon alle tuki Internet Explorerin versiolle 11:

```
"browserslist": [
    ">0.2%",
    "not dead",
    "ie >= 11",
    "not op_mini all"
  ]
```

Näiden muutosten jälkeen elementtiin ilmestyi siihen kuuluva background-color.

Suunnittelua noudattaen merkinlisäyslaatikon inset-tyylimääritys vaihdettiin neljään yksittäiseen määritykseen. Tämän muutoksen jälkeen laatikko ilmaantui näkyviin oikealla tavalla.

Merkinlisäyslaatikolle asetettiin korkeus esimerkkikoodissa 6 näkyvällä tavalla vain Internet Explorer -selaimessa, jonka jälkeen laatikko toimii jälleen halutulla tavalla ja scrollbar hävisi ikkunasta.

```
@media all and (-ms-high-contrast: none), (-ms-high-contrast: active)
{
    #popupBox {
        height: 60%;
    }
}
```

Esimerkkikoodi 6. Popupbox-elementtiin asetettu CSS-korkeusmääritys.

Koodissa havaitaan @media-komennolla Internet Explorer -selaimen käyttö, jolloin popupBox-elementtiin asetetaan 60 prosentin korkeusmääritys.

Merkinlisäyslaatikon ajansyöttölaatikoille luotiin RegEx. Esimerkkikoodissa 7 esiintyvän ratkaisun käyttöönoton jälkeen aikakenttiin ei voi enää asettaa muuta kuin kellonaikoja Internet Explorerissa tai muissa selaimissa, joista aikakenttien automaattinen validointi puuttuu.

```
let regex = /^(([0-1][0-9])|([2][0-3]))[:][0-5][0-9]$/;
let startIsValid = regex.test(markerStartTime);
let stopIsValid = regex.test(markerStopTime);
if (startIsValid && stopIsValid) {
     addToMarkers(newMarker)
     setIsPopupVisible(false)
} else {
     alert('Ajan täytyy olla validi! (Muodossa "00:00"!)')
}
```

Esimerkkikoodi 7. Aikakenttien RegEx-validointi merkin luonnin funktiossa.

Koodissa asetetaan regex-muuttujaan sääntö, johon on asetettu haluttu aikaformaatti. Kyseistä muuttujaa käyttämällä testataan asetetun merkin aukeamis- ja sulkemisaikojen kelpoisuutta. Jos kenttien tiedot täyttävät vaatimukset, merkki lisätään kartalle.

Näiden korjausten jälkeen ei Internet Explorer -selaimessa enää havaittu muita ongelmia sivuston käytössä. Kuten korjausprosessista voidaan huomata, Internet Explorerissa on ominaisuuksien rikkoutumiselle hyvinkin suuri mahdollisuus verkkosivustoa luotaessa. Muut verkkoselaimet eivät aiheuta yleensä lainkaan

yhtä paljon ongelmia kuin kyseinen selain. Tämän takia tulisi aina harkita pitkään ja hartaasti, halutaanko oikeasti kyseistä selainta tukea sivustolla.

6.5 Korjausten varmistaminen

Ensimmäinen ruudunleveydestä aiheutuva ongelma ja sen korjaus testattiin simuloimalla pienempää ruudunleveyttä Google Chromessa. Tässä vaiheessa korjauksen toimivuus testattiin vielä varmuuden vuoksi oikealla mobiililaitteella, jotta korjaus olisi varmasti toimiva. Korjattu versio ladattiin Heroku-pilvipalvelualustalle, jotta puhelimen verkkoselaimella päästäisiin käymään tehdyllä esimerkkisivustolla. Ominaisuuksien läpikotaisen testaamisen jälkeen todettiin sivuston toimivan samalla lailla myös oikealla mobiililaitteella. Sivusto testattiin mobiililaitteella myös vaakatasossa, sillä myös leveämmän vaakatasossa olevan näytön pitäisi toimia oikein sivustoa käyttäessä.

Sivustoa testattiin lopuksi myös toista tietokonetta käyttämällä laitekohtaisten ongelmien havaitsemiseksi. Kyseisellä tietokoneella testattiin sivustoa Google Chrome-, Mozilla Firefox-, Microsoft Edge- ja Internet Explorer -selaimilla, jotka olivat jo valmiiksi ladattuina. Sivusto toimi oikein myös toisella tietokoneella kaikkien korjausten jälkeen.

Projektiin ladattiin vielä lopuksi Eslint-liitännäinen caniuse, jonka tehtävänä on ilmoittaa mahdollisista yhteensopivuusongelmista. Liitännäinen huomautti yhdestä fetch-komennosta, sillä sitä ei tueta Internet Explorerissa. Tämä huomio asetettiin piiloon, sillä projektiin aiemmin ladattu React JS -polyfill korjaa jo kyseisen puuttuvan ominaisuuden.

6.6 Lopputulosten analyysi

Verkkosivulta onnistuttiin löytämään virheitä ja korjaamaan ne melko helposti ja nopeasti. Sivuston testaaminen muilla selaimilla ja laitteilla osoittautui hyödylliseksi, sillä sivulta löydettiin odottamattomia ongelmia. Verkkosivujen kehityk-

sessä kannattaisi testata juuri näiden ongelmien varalta, sillä tämä ei vaadi paljoa työtä, mutta siitä saadaan paljonkin hyötyä. Pahimmassa tapauksessa hyvinkin nopeasti korjattavat ongelmat saattaisivat tehdä sivustosta mahdottoman käyttää joissakin selaimissa. Ongelmien korjaamisen kannattavuutta kannattaa pohtia riippuen selainten käyttäjien määrästä.

Internet Explorer -selaimen tukeminen sivustolla aiheutti kaikista eniten työtä. Tämän takia vanhempien selainten tukemista kannattaa harkita vakavasti. Uudemmissa selaimissa ei ilmennyt lähes lainkaan ongelmia, mutta tästä huolimatta testaaminen kannattaa silti suorittaa niilläkin.

Vaikka esimerkkisivuston testaaminen osoittautui helpoksi, pitää ottaa huomioon sivuston laajuus. Verkkosivusto sisältää vain yhden näkymän, mikä myös helpottaa ja nopeuttaa testausta ja ongelmien korjausta. Isomman kokoisissa enemmän sisältöä sisältävissä sivustoissa testaus ja korjaus tulisi todennäköisesti vaatimaan enemmän aikaa ja työskentelyä. Jos isompien sivustojen yhteensopivuus halutaan taata, kannattaisi uudet sisällöt aina testata heti niiden luonnin jälkeen. Tällä tavoin yhteensopivuuden varmistaminen sujuu luontevammin ja nopeammin. Näin käyttäjät eivät myöskään joudu käyttämään sivustoa rikkinäisenä ennen korjausten tekemistä.

7 Yhteenveto

Insinöörityössä onnistuttiin osoittamaan yhteensopivuuden varmistamisen tärkeys. Työssä esiteltiin keinoja havaita ja toteuttaa tätä yhteensopivuutta. Insinöörityön tekemisen kautta perehdyttiin yhteensopivuuden varmistamiseen osana verkkosivujen kehitystä, ja tätä kautta onnistuin parantamaan ja laajentamaan omia ohjelmointitaitojani. Kaikkiin johdannossa asetettuihin tavoitteisiin päästiin.

Insinöörityö antaa lukijalle perustason valmiudet sivustojen yhteensopivuuden testaamiseen ja korjaamiseen. Työssä huomattiin, että parhain tapa sisäistää uusia ohjelmoinnin ja testaamisen keinoja on kokeilla niitä käytännössä. Ennen

insinöörityön tekoa verkkosivustojen yhteensopivuuteen liittyvää työtä ja sen tärkeyttä ei ymmärretty. Työn tekemisen jälkeen aiheen tärkeys tuli selväksi.

Esimerkkisivuston kautta esiteltiin eri yhteensopivuusongelmia ja niiden mahdollisia ratkaisuja käytännössä. Jatkokehityksenä työssä olisi voitu ehkä vielä luoda toinenkin esimerkkisivusto käyttäen jotain tyylikirjastoa. Täten olisi voitu pohtia paremmin käytännössä tyylikirjastojen vaikutusta yhteensopivuuden helpottamiseen. Toisena vaihtoehtona olisi voitu tarkastella yhtä tai useampaa valmista verkkosivustoa. Näitä sivustoja, ja niiden testaamisen ja korjaamisen helppoutta olisi voitu verrata ensimmäiseen sivustoon.

Lähteet

1 techterms.com. Verkkoaineisto. https://techterms.com/definition/framework Luettu 11.4.2022.

2 w3.org. Verkkoaineisto. https://www.w3.org/ Luettu 3.4.2022.

3 gs.statcounter.com. Verkkoaineisto. https://gs.statcounter.com/browser-market-share/desktop/worldwide/#monthly-202102-202202 Luettu 13.3.2022.

4 developer.mozilla.com. Verkkoaineisto. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing Luettu 10.3.2022.

5 lambdatest.com. Verkkoaineisto. https://www.lambdatest.com/blog/browser-engines-the-crux-of-cross-browser-compatibility/ Luettu 13.3.2022.

6 docs.microsoft.com. Verkkoaineisto. https://docs.microsoft.com/fi-fi/lifecycle/announcements/internet-explorer-11-end-of-support Luettu 13.3.2022.

7 creativebloq.com. Verkkoaineisto. https://www.creativebloq.com/features/future-web-browsers Luettu 23.4.2022

8 cloudwards.net. Verkkoaineisto. https://www.cloudwards.net/firefox-vs-google-chrome/ Luettu 3.5.2022.

9 vmware.com. Verkkoaineisto. https://www.vmware.com/topics/glossary/content/virtual-machine.html Luettu 15.4.2022.

10 caniuse.com. Verkkoaineisto. https://caniuse.com/ Luettu 21.4.2022.

11 w3schools.com. Verkkoaineisto. https://www.w3schools.com/tags/ref_html_browsersupport.asp Luettu 21.4.2022.

12 Practical CSS3: Develop and Design, Mills Chris, Metropolia Finna E-kirja.

13 stackoverflow.com. Verkkoaineisto. https://stackoverflow.com/questions/49986720/how-to-detect-internet-explorer-11-and-below-versions Luettu 24.4.2022.

14 stackoverflow.com. Verkkoaineisto. https://stackoverflow.com/questions/20541306/how-to-write-a-css-hack-for-ie-11 Luettu 24.4.2022.

15 thenewcode.com. Verkkoaineisto. http://thenewcode.com/534/Cross-browser-Image-Blur-with-CSS Luettu 27.4.2022.