Opinnäytetyö (AMK)

Tietojenkäsittely

2019

Mikko Saarimaa

VERKKOSIVUN KÄYTTÖLIITTYMÄN SUUNNITTELU

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2019 | 27 sivua

Mikko Saarimaa

VERKKOSIVUN KÄYTTÖLIITTYMÄN SUUNNITTELU

Opinnäytetyössä tutkitaan millainen on hyvä web-sivuston tai applikaation käyttöliittymä, teoriaa johon tämä perustuu ja miten sellainen toteutetaan käytännössä. Opinnäytetyössä tarkastellaan käyttäjäkeskeistä suunnittelua, käyttöliittymien kehitysmenetelmiä ja niiden käyttötarkoituksia aina käytettävyyden näkökulmasta, sekä responsiivista kehitystä. Käsitellään myös miksi web-kehittäjän tulisi ymmärtää graafisen suunnittelun perusteita ja hyödyntää niitä alusta pitäen kehitysprosessissa, sekä millaisia hyviä menetelmiä kehittäjän kannattaa noudattaa. Opinnäytetyö myös selittää esimerkkejä lähdekoodista.

Työmetodini muodostuivat erinäisten internet-julkaisujen kriittisestä lukemisesta, sekä tekemistäni toteutustapojen kokeiluista. Keskeisinä aiheina ovat käyttäjäkeskeinen suunnittelu, web-sivujen tyypit, rakenne ja asettelu sekä responsiivinen kehitys.

Opinnäytetyön tuloksena selvitettiin useita menetelmiä, joilla web-kehittäjä voi tehdä työnsä jäljestä siistiä ja modernia. Käyttäjäkeskeisestä suunnittelusta opittiin, että sivun toimivuus tulee ensin, ja tämän aikaansaamiseen on olemassa monta teoriaa. Käyttäjää kannattaa kuunnella ja sivu pitää yksinkertaisena. Selvitettiin myös menetelmiä, joilla kehitetään responsiiviset sivustot alusta alkaen kaikkiin laitteisiin, jotka toimivat tulevaisuudessakin.

ASIASANAT:

Web-kehitys, käyttöliittymä, käyttäjäkeskeinen suunnittelu, responsiivinen kehitys

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology

2019 | 27 pages

Mikko Saarimaa

WEB USER INTERFACE DESIGN

The thesis will examine what a good website or web app user interface is like, the theory behind it, and how to implement one in practice. The thesis goes into user-centered design, user interface design and implementation, through the lens of usability, and always adhering to responsive design. Also explaining why a web developer should understand fundamental graphic design and coding principles, that will help them throughout the development process.

My working methods consisted of critical reading of various web sources and books, from learning through work, and from experimentation and observations I made in my free time. Central themes include web applications, user-centered design, interface structure and design, responsive design and practical explanation of the code involved. The goal is to give examples of how to implement useful features, by looking at some of the code involved.

The thesis examined various methods, that a web developer can use to make his work clean and modern. User-centered design was discussed, for example that functionality comes first, and that there are many good design principles. User feedback should be heard and the site kept as simple as possible. Methods were discussed to develop responsive websites that work on all devices, that will also work in the future.

KEYWORDS:

Web development, user interface, user-centered design, responsive design

SISÄLTÖ

1 JOHDANTO

Käyttöliittymä aiheena on ajankohtainen, sillä käyttöliittymiä on kaikkialla. Kaikelle on nykypäivänä web-sivut tai -sovellus. Käyttöliittymä vaikuttaa suoraan käyttäjän ja palvelun väliseen tiedonkulkuun ja asiointiin. Huono käyttöliittymä voi kokonaan pilata muutoin hyvän palvelun. Yrityksen sivut antavat asiakkaalle ensivaikutelman. Sivujen toimivuus ja käytön luontevuus ratkaisee, palaako asiakas sivulle vai etsiikö hän paremman palvelun muualta. Siirtyminen digitaalisiin palveluihin edellyttää niiden olevan parempia ja intuitiivisempia kuin aikaisempien palvelujen.

Opinnäytetyössä tutkitaan millainen on hyvä web-sivuston käyttöliittymä, mihin tämä perustuu ja miten sellaista voi lähteä toteuttamaan. Opinnäytetyö tutkii käyttäjäkeskeisen suunnittelun ja käyttöliittymän suunnittelun teoriaa, sekä näiden yhteensovitusta aiheeseen liittyvillä esimerkeillä, joissa tarkastellaan sivujen toimintaperiaatteita ja koodia.

Nykypäivän käyttäjä olettaa, että palvelu toimii hänen laitteellaan, oli se sitten tietokone tai älypuhelin. Laitteita on monen merkkisiä ja kokoisia, joissa on monesti eri verkkoselaimet. Kehittäjän työstä yhä enemmän aikaa menee siihen, että varmistetaan, että palvelu toimii oikein useimmissa laitteissa ja ohjelmistoissa. Tätä varten on olemassa responsiivisia kehitysmenetelmiä. Opinnäytetyö käsitelee käyttöliittymiä sekä sivun vierailijan että kehittäjän näkökulmasta, ja miten niitä suunnitellaan vastuullisesti.

Opinnäytetyö on tehty sähköisen kirjallisuuden sekä verkkolähteiden kriittisen tarkastelun pohjalta.

2 WEB-SIVUT JA SOVELLUKSET

2.1 Web-sivujen ja -sovellusten tyypit

Web-sivut tai sovellukset ovat joko dokumentteja tai ohjelmia. Dokumenttityyppinen web-sivu on usein muuttumaton ja ladataan selaimeen kerralla, sisältäen esimerkiksi tekstiä ja taulukoita.

Verkkokaupat yleisesti ovat tietokannan päälle rakennettuja web-sovelluksia. Web-sovellus on ohjelma, joka sijaitsee palvelimella ja esittää palvelunsa web-selaimen avulla. Web-sovelluksessa käyttöliittymä koostuu web-sivuista, joiden käyttö tapahtuu hyperlinkkien ja web-lomakkeiden kautta. Web-sovellusten määrä on lisääntynyt tasaisesti viime vuosina. Samalla on lisääntynyt niiden käyttäjien määrä. (Pop 2002.)

Web-sovelluksia on monen tyyppisiä, kuten natiivisovellus, web-sovellus tai hybridisovellus. Sivustoista tehdään yhä useammin yksi versio, sen sijaan että olisi erillinen mobiili- ja työpöytäversio. Siinä missä Android- ja iOS-käyttöjärjestelmiä varten pitäisi kehittää kaksi eri sovellusta, yksi web-sovellus toimii molemmissa yleisten selainten kautta. Monia työpöytäsovelluksia on viime vuosina käännetty web-sovelluksiksi, vähentäen näiden kehityksen tarvetta erillisille laitteille, sekä kehityksen, jakelun ja ylläpidon hintoja.

Progressive Web App (PWA) on nykypäivänä yhä suositumpi web-sovelluksen muoto, jossa yhdistyvät mobiilisovelluksen helppous ja web-sivun teknologiat. Tämä on suunniteltu toimimaan lähes millä tahansa modernilla alustalla. PWA ei tarvitse asennusta, sillä se ladataan suoraan selaimeen. Nämä sovellukset toimivat samalla tavalla sekä tietokonella että mobiililaitteilla. Sovellusten on tarkoitus myös toimia osin ilman internettiä. (Wikipedia 2019.)

2.2 Yleiset merkintä- ja ohjelmointikielet

Markup-kielet ovat tietokonekieliä, jotka käyttävät merkkejä ilmaisemaan elementtejä dokumentissa. HTML ja XML-kielet ovat yleisimpiä. Nämä ovat ihmiselle luettavia kieliä, joissa sanat usein ilmaisevat sisällön tyypin. HTML on rakenteinen teksti, joka muodostuu sisäkkäisistä ja peräkkäisistä elementeistä. (TechTerms 2019.)

Web-sivun sisältö rakennetaan HTML-kielellä, jonka jälkeen CSS-kielellä lisätään ohjeita, miltä sivun tulisi näyttää. CSS-kielellä sivulle annetaan joukko tyyliohjeita jotka web-selain lukee, muuttaen sivun halutun näköiseksi. CSS-kieli on W3C:in (World Wide Web Consortium) standardi sivujen esittämiseen. Sekä CSS-kielestä että sivujen tyyleistä on tullut hienostuneita, ja kielellä tehdäänkin pääasiallinen sivun ulkoasun muotoilu.

HTML ja CSS pidetään usein erillään, jotta saman sivun voi muotoilla uudelleen miten haluaa. Tämä perustuu ‘source order independence’ -käsitteeseen, joka tarkoittaa sivun ulkoasun riippumattomuutta siitä, mitä sisältöä sivulla on. Sivun tyylin voi vaihtaa kerralla, vaihtamalla luettavaa CSS-tiedostoa. Yhdellä tiedostolla voi asettaa yhtenäisen formatoinnin kaikille sivuille, tai voidaan antaa eri tyyli jokaiselle sivulle. Tyyliä voi joillain sivuilla vaihtaa valikosta.

2.3 World Wide Web Consortium

World Wide Web Consortium, lyhyesti W3C on kansainvälinen yritysten ja yhteisöjen yhteenliittymä, joka ylläpitää ja kehittää WWW:n standardeja tai suosituksia kuten W3C niitä kutsuu. Organisaatio pyrkii edistämään yhteensopivuutta ja yhteisymmärrystä alan jäsenten välillä W3C:n määrittelemien uusien standardien hyväksymisessä. Eri tahot tarjoavat HTML-sovellusten yhteensopimattomia versioita, mikä aiheuttaa epäjohdonmukaisuutta verkkosivujen esittämisessä. Konsortio yrittää saada kaikki nämä tahot toteuttamaan joukon keskeisiä periaatteita ja komponentteja, jotka konsortio valitsee. (w3.org)

2.4 JavaScript -sovelluskehykset

JavaScript on dynaaminen komentosarjakieli, jolla sivuun voidaan lisätä erilaisia funktioita. JavaScriptin funktiot voivat dynaamisesti manipuloida sivun sisältöä. Skriptit voidaan sisällyttää HTML-tiedostossa tai omissa tiedostoissaan. JavaScript mahdollistaa asiakaspuolisten skriptien suorittamisen selaimessa, mutta siitä on tullut myös yleinen palvelinpuolen kieli. (Goel 2019.)

JavaScriptille löytyy monia suosittuja sovelluskehyksiä kuten Angular, React ja Vue.js. Sovelluskehys, eli englanniksi ’framework’, on JavaScriptillä kirjoitettu kehys tai pohja, jonka päälle voidaan rakentaa sovellus, manipuloiden sen tarjoamia funktioita eri tavoin.

Sovelluskehykset tekevät usein työskentelyn JavaScriptin kanssa helpommaksi. Ne myös tarjoavat responsiivisuutta, eli niillä luotujen sovelluksien toimivuutta eri laitteiden välillä. (Goel 2019.)

Angular on Googlen operoima sovelluskehys. React taas on Facebookin luoma kehys joka on tullut lyhyessä ajassa hyvin suosituksi. Sillä luodaan dynaamisia käyttöliittymiä web-applikaatioille, jotka kestävät suurtakin liikennettä. (Goel 2019.)

Reactilla tai Angularilla voidaan luoda SPA- eli yhden sivun applikaatioita. Ne ovat web-applikaatioita jotka lataavat yhden HTML-sivun, jota voidaan päivittää eli muuttaa käyttäjän toimesta käytön aikana. SPA pystyy myös kommunikoimaan palvelinten kanssa lataamatta sivua uudelleen, joka tarjoaa paremman käyttäjäkokemuksen. (Goel 2019.)

Edellä mainitut sovelluskehykset tukevat web-komponentteja, ja ne usein rakennetaan niistä. Nämä ovat yleensä tiedostoja, joihin on sisällytetty HTML-elementtejä ja JavaScriptiä, ja jotka vastaavat usein sivun tietyn osan näyttämisestä. Halutut komponentit voidaan ladata esimerkiksi painikkeesta, tai jo sivun avautuessa. Komponentteja voidaan sovelluksen ollessa käytössä näyttää ja piilottaa vapaasti. (Goel 2019.)

3 KÄYTTÄJÄKESKEINEN SUUNNITTELU

Käyttöliittymän eli käytännössä web-sivun tai –sovelluksen kehitys koostuu yleensä seuraavista vaiheesta: tutkimus ja suunnittelu sekä toteutus.

Käyttöliittymä on enemmän kuin tyylilaji. Käyttöliittymien kehitys ei ole aina niinkään subjektiivista. On olemassa hyvin sekä huonosti toimivia menetelmiä, perustuen tutkimuksiin ihmisen havainnoinnista ja ihmisten käyttäytymisestä internetissä.

Kun suunnittelee websivuja, kannattaa miettiä sisältöä ennen ulkoasua. Vierailija ei todennäköisesti kiinnitä huomiota sivun koodiin, responsiivisuuteen tai muuhun tekniseen asiaan. Tarkoituksen täytyy tulla ennen ulkomuotoa. (Peterson 2014.)

Käyttäjän tullessa sivulle hän usein aloittaa skannaamalla sen läpi, etsien jotain tiettyä asiaa. Kaikki muu on tässä vaiheessa tiellä, ja käyttäjän aikaa ei tule tuhlata. Sivun kannattaa käyttää lyhyitä lauseita ja helppoa kieltä. Hyvä muistisääntö kehittäjälle on, että sivu tarvitsee vähemmän sisältöä sivulla kuin tämä ajattelee. (Peterson 2014.)

Käyttäjäkeskeisessä suunnittelussa on tärkeää asettua käyttäjän asemaan. Kehittäjä miettii, mitä käyttäjä tekee sivulla, löytääkö tämä helposti ja nopeasti etsimänsä, sekä toimiiko kaikki odotetusti. Käyttäjäkeskeinen suunnittelu perustuu ideaan, että monet erilaiset ihmiset tulevat käyttämään sivua ja monesta eri lähtökohdasta.

On yrityksen oman menestyksen ja asiakkaiden tyytyväisyyden mukaista luoda helppokäyttöinen, tarkoituksenmukainen ja persoonallinen web-sivusto. Web-sivusto antaa kävijälle ensivaikutelman. Sivuston ulkoasun tulisi olla vakuuttava ja sopia käyttötarkoitukseen, eikä sisältää ylimääräisiä asioita, jotka vievät huomion pois käyttäjälle tärkeistä toiminnoista. Pitkään päivittämätön sivusto ei välttämättä anna hyvää vaikutelmaa yrityksen ajantasaisuudesta.

Käyttäjän näkökulmasta sivusto joko toimii oikein tai ei, joka tekee kehittäjän työstä haastavan. Kaiken oletetaan toimivan, sen sijasta että lueteltaisiin yhteensopivia laitteita, mikä olisi nykyään mahdotonta. Kehittäjän pitää varmistaa, että sivusto on yhteensopiva kaikenlaisissa mobililaitteissa sekä web-selaimissa.

Jotkut sanovat, että käyttäjää ei pidä sotkea kehitysprosessiin, kun taas toiset painottavat sen hyötyjä. “Usein ihmiset eivät tiedä mitä haluavat ennen kuin näytät sen heille” on Steve Jobsin kommentti aiheeseen. Todellisuus vaikuttaisi olevan näiden kahden välillä.

Tulisi kuitenkin tietää mitä asiakas haluaa, mutta olisi suotavaa ehdottaa ominaisuuksia joita asiakas ei välttämättä olisi itse ajatellut, esimerkiksi päivitettäessä vanhoja sivuja. Kannattaa kuitenkin varoa, ettei sivulle kerry päivityksien mukana liikaa ominaisuuksia, jotka tekevät sen käytön sekavammaksi. Ei kuitenkaan kannata tehdä turhaa työtä ja tulisi olla priorisoitu lista sovittuja asioita jotka toteutetaan. Tekemiseen menevän ajan voi ylimitoittaa asiakkaalle, jolloin jää aikaa testaamiseen sekä mahdollisten ongelmien taklaamiseen. Lisäksi ei haittaa jos projekti valmistuu ennen ajankohtaa. (Lowdermilk 2013.)

Kaikilla kehittäjillä ei välttämättä ole yhtä hyvää visuaalista silmää tai kokemusta asiakkaiden palvelemisesta. Joissain yrityksissä on erillinen UX eli käyttäjäkokemukseen keskittyvä tiimi, joka voi suunnitella sivuston graafisen ulkoasun yhdessä mahdollisten asiakkaiden kanssa, jonka jälkeen ohjelmoijat voivat toteuttaa sen käytännössä, dokumentoiden työn vaiheista. Sivusta luodaan usein aluksi rautalankamalli, josta erottuu millaisiin kokonaisuuksiin sisältö jaotellaan ja kuinka monelle sivulle se menee. Ulkoasun selkeytyessä voidaan luoda graafisia malleja tietokoneella, joiden ei tarvitse vielä sisältää toiminnallisuuksia.

Neuvotellessa asiakkaan tai käyttäjien kanssa jolle työstää sivustoa, kannattaa myöntää jos ei ymmärrä kaikkea, ja antaa heidän selittää asiat kuin tavalliselle käyttäjälle. Näin voi saada hyödyllisempää tietoa ja laajemman näkökulman eri käyttäjien tarpeista. Kun lähtee abstrakteista tarpeista, voi siten helpottaa käytännön ratkaisujen löytymistä. Hyvä suunnitelma, selkeät tavoitteet ja roolit tiimissä sekä vaiheiden dokumentointi on tärkeää. Sivun suunnittelussa on olennaista selkeä visio ja yhtenäinen tyylisuunta. Miten palveluun voidaan tuottaa lisäarvoa on kysymys kaiken keskellä. Aivoriihi tiimin kanssa ja ideoiden kerääminen on kannattavaa. Esimerkiksi päivitettäessä vanhoja sivustoja kannattaa miettiä, mitkä ominaisuudet menevät jatkoon ja mitkä jäävät pois, sekä mitä mahdollisia uusia tarpeita tulee esiin. Web-kehitys on dynaaminen prosessi: ei ole olemassa valmiita ratkaisuja, vaan se vaatii mukautumista, luovuutta ja jatkuvaa oppimista. (Lowdermilk 2013.)

Käytämme ymmärrystämme aiemmista käyttöliittymistä uusia koittaessa. Meillä on tietty ajatusmalli, eli aiempi käsitys siitä miten asioiden tulisi toimia, mistä halutut toiminnot löytyvät ja miten niitä käytetään. Aiempi tieto vaikuttaa uuden tulkitsemiseen. Esimerkiksi disketin ikoni tarkoittaa tallentamista. Nuoremmat henkilöt eivät tätä ikonia välttämättä tunnista, sillä heidän aikanaan ei tällaisia ole käytetty. Kannattaa sisällyttää monta eri ilmaisutapaa, kuten osoittimen viereen ilmestyvä kuvaava teksti tämän ollessa elementin

päällä. Kaikkea ei kannata yrittää keksiä itse, jos on olemassa valmiita ja hyvin testattuja ratkaisuja. Internetissä on olemassa paljon ilmaista lähdekoodia jota on vertaisarvioitu paljon. Kannattaa keskittyä siihen mikä toimii ja tehdä siitä toistuvaa dokumentoinnin kautta. (Lowdermilk 2013.)

'Principle of proximity' on ihmisen kognitioon eli havainnointiin nojaava periaate. Tämän mukaan ihminen havaitsee yhteyksiä asioiden välillä enemmän, jos ne ovat lähekkäin, kun taas kaukana toisistaan ne ovat toisiinsa liittymättömiä. Tämän perusteella asiat sivulla kannattaa ryhmitellä siten, että toisiinsa liittyvät toiminnallisuudet ovat lähekkäin. Käyttäjä löytää helposti toiminnot koska asiat on harkitusti organisoitu. (Lowdermilk 2013. 63-64)

'Fitt's Law' on sääntö, jonka mukaan mitä kauempana elementti on toisesta elementistä, eli mitä pidemmäs kursoria joutuu raahaamaan, sitä enemmän tarkkuus kärsii ja aikaa kuluu. Sääntö on korrelatiivinen: mitä kauemmas toiseen elementtiin liikutaan, sitä suurempi sen myös tulisi olla. Tämä on hyvä periaate hiirtä käyttäville käyttöliittymille. (Lowdermilk 2013.)

Journalismista tulee ylösalaisen pyramidin periaate. Tämä perustuu ideaan, että ensimmäisenä esitettävä tieto on tärkeintä, ja siitä eteenpäin vähenevissä määrin tärkeää. Näin lukija saa aiheesta sen mitä etsii ja voi omasta kiinnostuksesta jatkaa lukemista. (Peterson 2014.)

'Hick's Law' taas on sääntö, joka auttaa laskemaan ajan joka käyttäjällä menee tehdä päätös suhteessa annettujen vaihtoehtojen määrään. Se tunnetaan myös reaktioaikana. Tämä viittaisi siihen, että prosessoimme tietoa tietyllä vakionopeudella. Siksi ei tule kuormittaa käyttäjää liiallisella sisällöllä. (Lowdermilk 2013. 74-75)

Sivun toimintojen tulee olla nopeita, eikä latausajat sivujen välillä voi olla liian pitkiä. Kuinka paljon sivulla menee aikaa vaikuttaa käyttökokemukseen.

Kehittäjän on hyvä ottaa selville, miksi muiden sivut ovat suosittuja ja toimivia. On helppo ottaa itsestäänselvyytenä, että sivustot toimivat tietyllä tavalla, ymmärtämättä miten kyseiseen ratkaisuun päädyttiin. Hyvien menetelmien lisäksi on kannattavaa paneutua yleisempiin virheisiin, jotta niitä voisi välttää käymästä läpi. (Lowdermilk 2013.)

Sivusto tulee suunnitella niin, että käyttäjä ei helposti tee virhettä, ja tehdessäänkin suurempaa ongelmaa ei pääse muodostumaan. Oletuksena on, että käyttäjät tulevat teke-

mään virheitä, ja sitä ei voi pitää vain käyttäjän vastuuna. Käyttäjää ei voi syyttää tyhmyydestä. Lee Ross, Stanfordin yliopiston psykologi sanoi, että on virhe ajatella ihmisten käytöstä sellaisena kuin se on, eikä sellaisena kuin tilanne saa aikaan. Esimerkkinä uloskirjautumisen painikkeen ollessa näkyvissä helposti, käyttäjä muistaa kirjautua ulos. Palvelu voisi myös kirjata käyttäjän ulos 30 minuuttia käytön loppumisesta, mutta tämä on viimeinen turvakeino. Kyse voi olla käyttäjän turvallisuudesta. (Lowdermilk 2013.)

Käyttäjällä tulisi olla helppo tapa ottaa kehittäjään yhteyttä ongelmien tai virheiden sattuessa. Kehittäjien pitäisi saada tietää mitä käyttäjä yritti tehdä, joka johti ongelmaan. Tarvitaan vain yksi ihminen löytämään uusi ongelma tai haavoittuvuus. Joillain sivuilla on chat-ikoni, josta pääsee kirjoittamaan palautetta tai kysymään apua. Tähän voi vastata tekoäly, joka osaa vastata yleisiin kysymyksiin, tai oikea ihminen jos ongelma ei ratkea. Hyvä tapa kerätä palautetta olisi myös lyhyt kysely, johon vastaaminen ei ole pakollista, ja kysymykset ovat toimintojen arvostelua asteikolla 1-5. Tähän vastaaminen ei vie aikaa ja vastaajia tulee todennäköisesti enemmän. (Lowdermilk 2013.)

Ihmisten tulisi päästä kokeilemaan sivuja jo kehitysvaiheessa. Näin kehittäjä saa käsitystä siitä mikä toimii, onko ulkoasussa tai toiminnoissa puutteita, sekä miten eri ihmiset yrittävät käyttää sivuja. Ei tulisi pitää työtä itsellään kunnes se on tehty, sillä on melko varmaa että jotain puutteita on. Siitä huolimatta, että on olemassa visionäärejä jotka tietävät mitä ihmiset tarvitsevat, tulisi olettaa että oma idea on alustava. Käyttäjillä on usein eri näkemyksiä siitä mitä tarvitaan. Sivun asteittainen kehittäminen ajan myötä palautteen kautta on olennaista. Myös negatiivinen palaute on hyödyllistä jos sitä soveltaa konstruktiivisesti. Tämän kehittäjä voi käyttää kehittymiseen sekä suunnittelijana että ohjelmoijana. (Lowdermilk 2013.)

4 KEHITYSMENETELMÄT

4.1 Typografia

Typografia eli painoasu on yksi keskeisimpiä taitoja jotka web-kehittäjän on hallittava. Työnkuvaan kuuluu suuren tietomäärän tulkinta ja jakaminen luettavaan muotoon, josta lukija löytää sen mikä häntä kiinnostaa. (Reichenstein 2006.)

Hyvistä otsikoista käy ilmi, mitä sisältöä sivulla tulee olemaan. Näiden perusteella käyttäjä voi navigoida kohtaan joka häntä kiinnostaa. Hyvät otsikot nostavat myös sivua hakukonesijoituksissa, sillä näin hakukoneet tietävät mitkä tärkeimmät sivun aiheet ovat. Kuvaavat otsikot ovat myös ääneen lukemista -ominaisuutta käyttäville ihmisille hyödyllisiä. (Peterson 2014.)

Web-sivujen olennainen sisältö muodostuu yleensä kahdesta asiasta: kuvista ja leipätekstistä. Selkeä fonttivalinta on ensimmäinen asia. Tekstin eri värien sekä kokojen käyttö, erottaen otsikot, alaotsikot ja leipätekstit toisistaan jo ensikatsauksella. Tekstien koot ja muotoilu vaikuttavat sisällön luettavuuteen huomattavasti. Tekstiä voi muokata myös rivien, kirjainten ja sanojen välien koon osalta. Hyvä fontti luo koko sivusta oikean vaikutelman, huono taas voi pilata sen. Jotkut fontit ovat tosi hienoja, mutta joidenkin ihmisten voi olla vaivalloista lukea niitä. Tekstin muotoilu on tärkeää myös jotta ihmiset lukemisen tai näkemisen rajoituksista huolimatta pystyvät lukemaan vaivatta. Arial ja Calibri ovat suosittuja, koska ne ovat helppolukuisia ja sopivat useimpiin asiayhteyksiin.

Sivun kielenkäytön tulisi olla helposti luettavaa. Sivuston lukijat eivät välttämättä ole kaikki nopeita lukijoita. Luettavuutta voi helpottaa esimerkiksi jakamalla tekstiä palloja käyttäviin listoihin. (Peterson 2014.)

Google Fonts ja Adobe Fonts ovat esimerkkejä palveluista, jotka tarjoavat laajan valikoiman ilmaisia avoimen lähdekoodin fontteja. Google Fonts hoitaa fonttien lisensoinnin sekä jakelun. Ne antavat valittuun fonttiin linkin, joka tulee HTML-tiedoston head-osioon. Web-selain käy siten hakemassa fontin palvelimelta. Google Fonts kertoo sivullaan järjestävän fontit niiden suosion ja sijaintisi perusteella. Voit myös hakea sieltä fontteja fonttiperheen tai paksuuden perusteella sekä testata fontteja. Näiden palveluiden fontit ovat ns. web-safe fontteja. Tämä tarkoittaa niiden toimivan lähes joka selaimessa ja laitteessa.

Fontin määrittelyssä tulisi antaa myös fallback- eli varafontteja, joita käytetään jos edellä mainittua fonttia ei löydy. Ainakin varafontin tulisi olla sellainen, joka on kaikille saatavilla. Lopussa määritellään joko serif tai sans-serif fontti. Serifiin kuuluu ’väkäset’, sans-serifistä nämä ovat poissa. Useissa kirjoissa ja lehdissä käytetään serifiä osittain perinteen, osittain väitetysti paremman luettavuuden vuoksi. Riippuu kuitenkin käytettävästä fontista sekä sivun teemasta, kumpi näyttää paremmalta.

Tekstin sisällön tulisi olla informatiivista mutta helppolukuista. Ei tulisi toistaa samoja asioita uudelleen. Tekstin pituus voi vaikuttaa, kuinka pitkälle käyttäjä lukee sitä, tai lukeeko ollenkaan. Yleistä on, ettei pommiteta käyttäjää liialla tiedolla, vaan annetaan tiivistelmiä joiden perässä seuraa ‘lue lisää’ painike. Tämä painike voi avata laatikon suuremmaksi, viedä toiselle sivulle tai JavaScriptillä näyttää dynaamisesti haluttua sisältöä.

Suurella fontilla tai värillä korostaen voi tuoda esiin jotain, minkä haluaa käyttäjän lukevan ensin, kuten lainauksen tekstistä. Tämän voisi sijoitella ennen tekstiä tai sen viereen.

Fonttien skaalauksessa voi aluksi asettaa kaikille fonteille tietyn aloituskoon. Selain antaa muuten oletuksena koon 16px. Fonttikoon voi asettaa esimerkiksi ’px’ eli pikseliarvolla, mutta Internet Explorerilla tätä kokoa ei voida muuttaa. px tai pt-arvoja käyttäviä tekstejä ei voida muuttaa prosenteilla, jonka vuoksi ne eivät ole web-sivuilla hyviä vaihtoehtoja, sillä fonttikokoja joudutaan monissa tilanteissa muuttamaan. ’rem’ eli root em-arvo tarkoittaa HTML-dokumentin ylimmän elementin fonttikokoa, ja muiden elementtien fonttikoot suhteutetaan tähän arvoon. Esimerkiksi 0.9 rem on kaikkialla sama, 90% alkuperäisen kokoinen fontti. Käyttäessä rem-arvoa tulee huomata, ettei se toimi Internet Explorer 8:ssa, jota varten elementeille voidaan antaa varalle myös pikseliarvo. ’em’ viittaa senhetkiseen fonttikokoon, vaikka arvolla 2 em voidaan tehdä fontista kaksi kertaa alkuperäisen kokoinen. Prosenttiarvoja voidaan myös käyttää, esimerkiksi 50% on puolet ulomman elementin käyttämästä koosta. (Schaeffer 2008.)

Laitteissa, kuten iPhone, näytön pikselitiheys on korkea. Elementtien ja etenkin tekstin tulisi skaalautua suhteessa ruudun kokoon eikä pelkkään pikselitarkkuuteen. Tällaiseen skaalaukseen on myös olemassa yksikkö ’vw’ eli viewport width. Tämä sitoo elementin koon suoraan selainikkunan leveyteen, jolloin se pienenee ja suurenee sen mukana. Tämä kuitenkin johtaa usein liian suureen tai pieneen fonttiin, sen sijaan että teksti vain jatkuisi seuraavalle riville.

‘text-rendering’ on CSS-ominaisuus jolle voi antaa tietoa siitä mitä optimoida selaimen renderöidessä tekstiä. ‘auto’ -arvolla se valitsee itsestään käytettävän arvon riippuen tilanteesta, ja voi vaikuttaa nopeuteen, luettavuuteen tai asettelun tarkkuuteen. Jos käyttäjän laite on hidas, tähän valittaisiin arvo ‘optimizeSpeed’, joka muiden hienouksien kustannuksella lataa nopeiten.

‘optimizeLegibility' käyttää tekstin parannuksia kirjainten ja sanojen välien suhteen, maksimoiden luettavuuden, sekä valinnaisia fonttien tietoja jotka voivat sijaita fonttien tiedostoissa.

Jos fonttikoko olisi 9, jolle olisi annettu jälkeenpäin 140 % koko, lasketun fonttikoon pitäisi olla 12.6. Tällaista kokoa ei ole järjestelmän fonteissa, joten selain pyöristäisi sen kokoon 12. ‘geometricPrecision’ painottaa geometristä tarkkuutta. Tämä tarkoittaa, että fonteissa jotka eivät skaalaudu lineaarisesti, tämä laskee tarkat pikselimäärät fonteille. (MDN Web Docs)

4.2 Kuvat

W3C suosittelee CSS lisäksi mm. käyttämään kuvina websivuilla mahdollisuuden mukaan PNG tai SVG-tiedostomuotoja. PNG on on häviötön bittikarttagrafiikan tallennusformaatti. PNG on patentiton eli kaikkien vapaasti käytettävissä ilman lisenssimaksuja ja W3C:n standardi, ja suositus web-sivuilla käytettäväksi. (Wikipedia 2019.)

Kuvat ovat usein sivuston suurimpia tiedostoja ja niiden määrä ja tyyppi vaikuttaa sivun latausaikaan. PNG:n kuvanlaatu on vähän pakattuna parempi kuin JPG:n, mutta on helposti suuremman kokoinen tiedosto ja latautuu hitaammin. PNG tukee läpinäkyvyyttä, kun taas JPG ei. Esimerkiksi sivun logo voisi olla PNG, ja suuri taustakuva JPG.

Ikonit on helppo nähdä kaukaa ja niiden merkitys on universaali. Esimerkiksi kaupan kärry on ostoskorin ja ihmishahmo käyttäjätilin ikoni. Kuvakkeen päälle viedessä hiirtä se voi muuttua eri väriseksi, ilmaistakseen että käyttäjä voi painaa sitä. Peruskäyttäjä internetissä on nähnyt samankaltaisen ikonin monta kertaa ja tietää mitä se tekee. Käyttämällä ikoneita myös sivuston tekstimäärä pienenee, ja se voi olla silmää miellyttävämpi vaihtoehto. Myös lukuvaikeuksista kärsivät ihmiset voivat huomata ikonin helpommin kuin pienen tekstin. Ikoni voi tilanteesta riippuen näyttää vieressään numeron, kuten ostoskorissa olevien tuotteiden määrän. Tämä vähentää sivujen välillä vaihtamisen tarvetta, sillä tieto on helposti saatavilla.

Kun käytetään kuvia, pitää harkita ovatko ne tarpeellisia ja vahvistavatko ne viestiä, joka halutaan antaa. Kuville voi olla monta tarkoitusta, kuten luoda tunnelmaa tai näyttää esimerkiksi tuotekuva. Kuvia voidaan käyttää rikkomaan sivun monotonisuutta. Kuva voi olla yksittäinen, tai vaihtuva kuva, joskus antaen käyttäjän vaihtaa niiden välillä. Tätä varten voidaan luoda kuvakaruselli. Taustakuvia voidaan käyttää koko sivun taustalla tai yksittäisten elementtien taustana. Tekstilaatikon taustakuva voi näyttää hienolta mutta luettavuus tulee ensin. Kuvan läpinäkyvyys voidaan asettaa haluttuun pisteeseen, tai taustan päälle voidaan asettaa filtteri. Esimerkiksi tekstilaatikko, jonka taustakuvaa on sumennettu ominaisuudella ’filter: blur(20px)’. Tekstin näkyvyys on tällöin parempi.

Kuvien responsiivisuus tarkoittaa, että varmistetaan etteivät kuvat mene liian suureksi tai pieneksi, sekä ettei kuvan sisältö näy huonosti. Esimerkiksi kuvan leveyden ollessa sataprosenttinen sivun koosta, tulee sen korkeutta rajoitettaa, ettei se vie liikaa korkeutta. Sitten täytyy varmistaa, että kuvan sisältö pysyy näkyvissä eikä leikkaannu. ’background-size: cover’ -ominaisuudella kuva täyttää automaattisesti laatikon. Yhdessä ’background-position: center’ -arvon kanssa kuvan keskipiste pysyy ympäröivän laatikon keskipisteessä.

4.3 Asettelu

Sivun ulkoasua suunnitellessa jo pitää miettiä, millainen asettelu sivulle tulee, mitä käyttäjän on tärkeää nähdä jo silmäilemällä, tai mikä jää valikon taakse. Käyttökokemukseen vaikuttaa, kuinka monella klikkauksella käyttäjä pääsee halutulle sivulle. Sivun asettelun on oltava selkeä ja mahdollisimman yksinkertainen. Tämä tarkoittaa usein minimalistista tyyliä, jonka käyttöä ei tarvitse erikseen pysähtyä miettimään sillä navigointi tuntuu intuitiiviselta. Lisäksi pitää miettiä miten asetella paljon liikkuvia osia, ilman että ne täyttävät sivun, pitäen kuitenkin olennaisen tiedon ja toiminnot käden ulottuvilla. Sivuston käyttöliittymän kehitys ei ole itsestäänselvyys, vaan haaste joka vaatii suunnittelua.

Sivustosta kannattaa suunnitella etukäteen rautalankamalleja esimerkiksi kynällä ja paperilla, jonka jälkeen graafisen suunnittelun ohjelmistot, kuten Adobe Photoshop tai InDesign auttavat visualisoimaan mallin.

Etusivun mallin selkeytyessä on helpompaa rakentaa muut sivut yhtenäiseen asetelmaan. Mahdollisen asiakkaan mielipidettä tulee kysyä monessa välivaiheessa. Lisäksi tulee selvittää onko yleinen tyylisuunta oikea, mitkä osat sivusta vaikuttavat toimivilta tai

tarpeettomilta, sekä mitä voisi lisätä. On hyvä sopia konkreettisia tavoitteita, joita kohti työskennellä. Kehittäjän työ on helpompaa hyvän mallin pohjalta, ja turhien ominaisuuksien tekemiseen ei kulu aikaa.

Sivujen suunnitteluun on myös olemassa valmiita pohjia. Näiden kopioiminen onnistuu, jos ymmärtää miten ne toimivat, ja niitä voi muokata eteenpäin. Esimerkiksi W3Schools sivustolla on ilmaisia malleja. Mallit sisältävät usein valmiita perusominaisuuksia kuten navigointipalkin, skaalautuvat sisältölaatikot tai footerin. Sivun esimerkit vaikuttavat toimivan myös monissa näytön tarkkuuksissa. Pohjien luominen on myös hyvä keino nopeuttaa työtä myöhemmissä projekteissa, ja ne voi varastoida esimerkiksi pilvipalveluun nopeaa käyttöä varten.

‘display: block' ja ‘margin: 0 auto’ -attribuutit yhdessä käytettynä mahdollistavat elementin, esimerkiksi koko sivun asettamisen keskelle ruutua vaakasuunnassa. ‘display: block' tarkoittaa, että elementti varaa kokonaisen rivin. ‘margin: 0 auto’ tarkoittaa, että tietokone laskee sivujen marginaalin yhtä suureksi. Numero 0 viittaa ylä- ja alamarginaalin olevan tyhjiä. (W3Schools 2019.)

CSS Grid on CSS-kielen oma, useimpiin selaimiin sisäänrakennettu menetelmä, jota voi hyödyntää sivuston asettelussa. Yleisiin CSS-kielen arvoihin verrattuna CSS Grid mahdollistaa monipuolisemman ja tehokkaamman tavan asetella sisältöä:

‘display: grid’ - asettaa elementin sisäkkäisille elementeille ruudukkoasettelun

‘grid-template-columns' - määrittää kolumnien määrän

Esimerkiksi ‘grid-template-columns: auto auto' saa aikaan kaksi saraketta vierekkäin, jotka vievät yhtä paljon saatavilla olevasta tilasta.

‘grid-template-rows' - määrittää rivien määrän pystysuunnassa.

CSS Gridillä koodia on helpompi pitää siistinä. CSS Grid korvaa pääosin ‘float’ ja ‘position’-attribuutteja. (W3Schools 2019.)

Sisältöä voi siirtää haluttuun riviin tai sarakkeeseen mielivaltaisesti, yhdessä media queryn käytön kanssa, esimerkiksi näytön koosta riippuen. (Borgen 2017.)

Bootstrap on yksi tapa teoriassa helpottaa sivujen asettelua ja responsiivista kehitystä. Bootstrap on yksi suosituimmista HTML, CSS ja JavaScript kirjastoista, ja mahdollistaa sivun sisällön asettelun melko mielivaltaisesti, käyttäen sen omia ennalta määriteltyjä

tägejä. Bootstrap käyttää Flexboxia. Tägit on nimetty esimerkiksi ”container”, jolla määritel-lään tämän olevan ulompi laatikko, pitäen sisällään muita elementtejä, tai “row”, joka tekisi uuden elementin yhdeksi riviksi laatikon sisään. Tällä voidaan hoitaa sama asia kuin monella rivillä CSS-koodia. (Borgen 2017.)

Tämä voi säästää aikaa kehittäjältä alussa, mutta ei välttämättä ole paras ratkaisu. Bootstrapin luokkien nimet ovat pitkiä ja kaikelle pitää antaa oma luokka. Bootstrapin heikkouksiin kuuluu, että sivun asettelun muuttaminen ei onnistu ilman näkyvää CSS koodia muuten kuin muuttamalla HTML rakennetta. Bootstrap rajoittaa sivun sarakkeiden määrän 12 osaan. (Borgen 2017.)

CSS Gridissä on etuna Bootstrappiin se, että vaikka CSS Grid tyylit eivät toimisi vanhassa selaimessa, sisällön ollessa normaalia HTML koodia latautuu tämä normaalisti, sekä muut CSS tyylit jotka ovat tuettuja. (Borgen 2017.)

4.4 Värit

Kannattaa käyttää värisävyjä, jotka eroavat tarpeeksi paljon toisistaan, jotta sisältö pysyy luettavana käyttäjillä, joiden näytön kalibrointi ei välttämättä ole ideaali. Etenkin tulee välttää väriarvoja hyvin lähellä toisiaan, jotka muuttuvat saman näköisiksi huonolla kontrastilla. On olemassa syy, miksi valkoinen teema web-sivuilla on niin yleinen: musta teksti valkoisella taustalla on vähemmän rasittavaa silmille, kuin valkoinen teksti tummalla taustalla, joka ikään kuin vuotaa ympäröivään väriin ja voi aiheuttaa silmien rasitusta. Kuitenkin esimerkiksi yöaikaan ympäröivän valon ollessa himmeää tumma teema voi parantaa luettavuutta. Käyttäjälle voi antaa valikon, josta teemaa voi vaihtaa.

4.5 Efektit

Efektejä, joita useat sivustot hyödyntävät, on CSS-kielessä paljon. Elementeille ja tekstille voi antaa esimerkiksi varjoja. Varjot luovat syvyysvaikutelman ja näyttävät luonnollisilta. Efektejä kannattaa käyttää kuitenkin harkitusti. Hyvin käytettyinä ne ovat osa sivua ja huonosti käytettyinä ne pistävät silmään.

’box-shadow: 0 0 2px rgba(0,0,0,0.5)’. - box-shadow määrittää elementille varjon. Ensimmäiset kaksi nollaa ovat X ja Y -akselilla varjoa siirrettävä määrä, tässä tapauksessa suoraan laatikon alla. Kolmas numero 2px on varjon sumennus, joka luo pehmeät reunat.

Rgba tarkoittaa red green blue alpha, eli kolmea väriarvoa viimeisen luvun ollessa läpinäkyvyys. Kohdassa ’0.5’ varjo on puoliksi läpinäkyvä. Varjolle voi asettaa arvon ‘inset’, jolloin se näkyy laatikon sisäreunassa. Näitä ei voi kuitenkaan käyttää samanaikaisesti. Yksi yleinen käyttötapa on valkoinen laatikko valkoisella taustalla, jolle annettu varjo erottaa nämä toisistaan.

CSS-kielellä on mahdollista tehdä enemmänkin kuin asettelua. CSS mahdollistaa animaatiot. Animaatiot perustuvat elementtien CSS-attribuuttien muuttamiseen sivun ollessa auki. Animaation kestolle voidaan antaa ajanmääre kuten 0,5 s, jonka aikana esimerkiksi elementin koko muuttuu suuremmaksi ja sen sijainti siirtyy.

Lisäksi on mahdollista luoda animaatioita joissa on keyframe-pisteitä, eli pisteitä animaation aikana, joissa tapahtuu haluttu muutos. Esimerkiksi avautuvan valikon taustan väri vaihtuu mustaksi kohdassa 50 %. Animaatioiden nopeuden saa myös mm. alkamaan hitaasti, nopeutuvan puolivälissä ja hidastuvan lopussa, tai päinvastoin. Samaa animaatiota voi hyödyntää useammassa elementissä sen nimeen viittaamalla, olettaen että se suorittaa halutun toiminnon.

Useat selaimet käyttävät ’highlight’-efektiä elementtien ympärillä, eli reunoja jotka näkyvät kun elementtiä klikkaa, tai kun osoitin sijaitsee tämän päällä. Moni kehittäjä haluaa poistaa tämän käytöstä sivullaan ulkonäöllisistä syistä, mutta tässä on haittapuoli: pelkillä painikkeilla sivua navigoivat käyttäjät eivät pysty helposti liikkumaan valintojen välillä, koska eivät näe mikä elementti on valittuna.

5 RESPONSIIVINEN KEHITYS

5.1 Responsiivisuus

Älymobiililaitteita on ihmisillä jokapäiväisessä käytössä jatkuvasti enemmän. Yhä suurempi osa palveluista on siirtynyt yhteen taskussa kulkevaan laitteeseen, joten mobiilikäyttäjälle tulee tarjota kokonainen sivu. Responsiivinen kehitys tarkoittaa sovelluksen tai sivun kehitystä yhteensopivaksi mobiililaitteiden, kuten älypuhelinten ja tablettien kanssa. Yhä suurempi osa web-sivuista noudattaa tätä oletuksena. Myös yhteensopivuus lukuisten web-selainten ja niiden vanhojen versioiden kanssa täytyy huomioida. Käytännössä hyvin koodattu sivusto tai sovellus voi näyttää hieman erilaiselta asettelun kannalta, mutta aina luettavalta riippumatta käytettävästä laitteesta. Kehittäjän työstä suuri osa on kirjoittaa koodia, joka mahdollistaa että kaikki sovelluksen liikkuvat osat näkyvät oikein mahdollisimman monessa tilanteessa. Kehittäjä etsii kohtia, joissa sivu lakkaa näkymästä oikein tietyissä olosuhteissa. Kehitykseen kuuluu paljon jatkuvaa testaamista. Sovelluksen kuuluu säilyttää visuaalinen ilmeensä, mutta mukautua elementtien kuten valikoiden, laatikoiden, tekstin tai kuvien kokojen ja asettelun puolesta. Lisäksi täytyy varmistaa, etteivät elementit mene päällekkäin tai muutu lukemattomaan muotoon. Näiden pitää sopia käytettävän laitteen ruutuun, oli se sitten pysty tai vaakasuuntainen, sen tarkkuuteen, sekä fyysiseen kokoluokkaan.

Tietokoneiden näyttöjen tarkkuudet ovat nousseet viimeisen 10 vuoden aikana jopa 2-4 kertaa suuremmiksi. Näytöt ovat muuttuneet laajakuvaksi, useimmiten 16:9 kuvasuhteeksi. Mobiililaitteita on hyvin pienestä hyvin suureen tarkkuuteen, ja näissä voidaan näyttää sivut sekä pysty- että vaakasuunnassa. Yhä useampi omaa UHD-näytön tai television, jossa on 3840x2160 pikseliä. On myös älykelloja, joiden tarkkuuksissa puhutaan enää sadoista pikseleistä, samoin kuin vanhoissa puhelimissa.

Tietyt elementit muuttuvat useilla sivuilla erilaisiksi mobiilikoossa, esimerkkinä navigaatiopalkki. Työpöytäversiossa tämä on usein luettelo sivulinkkejä sivun yläreunassa. Kun näyttö on pienempi, tämän tilalle tulee yleinen burgeri-ikoni, jota painamalla valikko näytetään ja piilotetaan.

Sivuja on vielä pitkään suunniteltu periaatteella, että sivu näyttää samalta kaikilla näytöillä. Sivu on luotu staattisen kokoiseksi, jolloin laajemmalla ruudulla reunoihin muodos-

tuu tyhjää tilaa. Sittemmin on tullut ajatus ns. liukuvasta suunnittelusta, jossa sivun sisältö ikään kuin liukuu käyttäen tyhjän tilan hyödyksi. Suuri hyöty responsiivisessa kehityksessä on se, että sivusta tarvitsee kehittää vain yksi versio. Sivun tekemiseen voi mennä aikaa, mutta tämä maksaa itsensä takaisin vähentäessään ylläpitoa. (Peterson 2014.)

Mobiililaitteissa pystyttiin pitkään vain näyttämään tekstisisältöä. Älypuhelinten tulon myötä tämä muuttui. Tällöin syntyivät myös uudet tavat selata sivua kosketustoimintojen avulla. Eri laitteille alkoi kuitenkin syntyä liikaa eri sivuja, joita olisi pitkällä tähtäimellä vaikeaa ylläpitää. Responsiivinen sivu tähtää olemaan relevantti myös tulevaisuuden laitteissa. (Peterson 2014.)

Kolme vierekkäistä saraketta käyttävät hyvin suurta ruutua, mutta pienellä ne muuttuvat liian kapeiksi. Allekkaiset rivit taas näyttävät hyvältä puhelimella mutta muuttuvat liian leveiksi isommilla ruuduilla. Media queryjen, prosenttiarvojen ja muiden responsiivisten menetelmien yleistyttyä sivuista alettiin tehdä vain yksi versio, joka mukautuu automaattisesti käytettävään laitteeseen. (Peterson 2014.)

Kun suunnitellaan responsiivisesti, voi olla hyvä lähteä kehittämään ensimmäisenä sivun pientä versiota. Tämä voi suoraviivaistaa loppuosaa kehityksestä, laittaessa kehittäjän ajattelemaan, mitä oikeasti tarvitaan ja miten yksinkertaisesti sen voi muotoilla.

Ei ole käyttökokemukselle hyvä asia, jos käyttäjä tarvitsee erillisen linkin mobiiliversioon, tai jos hän joutuu vaihtamaan versiota jo sivulla ollessaan. On myös huono asia jos laitteesta riippuen sivulta katoaa näkyvistä sisältöä, tai menee liian kauas siitä mistä sen olettaisi löytyvän. (Peterson 2014.)

5.2 Menetelmät

Responsiivisen sivun HTML:n head-osuudesta löytyy tagit ‘width=device-width' sekä ‘initial-scale=1’. Ensimmäinen varmistaa, että sivun leveys vastaa käytettävän laitteen ruudun leveyttä. Jälkimmäinen varmistaa sivun avaamisen yhteydessä 1:1 suhteen CSS pikseleiden ja laitteen pikseleiden välille, käytännössä sivun koon olevan haluttu normaalikoko.

‘Minimum-scale’, ‘maximum-scale’ ja ‘user-scalable’-ominaisuudet huolehtivat sivun minimi ja maksimikoon. Ne eivät ole välttämättä aina kannattavia, sillä ne estävät käyttäjää

zoomaamasta sivua tietyn pisteen jälkeen. Käyttäjälle tulisi antaa vaihtoehtoja. Esimerkkinä liian suuren minimikoon kanssa sivu menisi horisontaalisesti yli ruudun, vaatien käyttäjää skrollaamaan oikealle.

Responsiivisten elementtien kannattaa hyödyntää prosenttiarvoja, sekä mitä tahansa tietokoneen itsestään laskemia arvoja, jotka mukautuvat eri näytön kokoihin. Tällä voidaan vähentää media queryjen liiallista käyttöä ja pitää koodi helpommin ylläpidettävänä. Jos koodi muuttuu pitkäksi ja vaikealukuiseksi, on muokkauksia tai uusia ominaisuuksia vaikeaa lisätä, varsinkin toiselle kehittäjälle.

Media Query tarkoittaa CSS-menetelmää jolla muutetaan CSS-arvoja halutunlaisiksi, kun ruutu on yli tai ali annetun tarkkuuden. Esimerkiksi sivun otsikko, joka menisi reunan yli pienellä ruudulla. Media queryllä yhden tai useamman elementin skaalaus asetetaan haluttuun arvoon halutussa pisteessä, kuten 80 % alkuperäisestä. Esimerkiksi:

```
@media only screen and (max-width: 800px) {
    .textbox {
        Border: none;
    }
}
```

Esimerkissä selainikkunan leveyden ollessa alle 800 pikseliä, luokan ’textbox’ omaavat elementit eivät käytä reunuksia, vaikka sellaiset olisi aiemmin määritelty. Media queryt tulee sijoittaa elementin varsinaisten tyylien jälkeen, sillä ne korvaavat aiempia attribuutteja.

Viewport tarkoittaa aluetta sivusta, joka näkyy selaimen sisällä. Mobiililaitteissa tämä alue pysyy usein saman kokoisena, kun selaimen kokoa ei voi muuttaa. Media queryt viittaavat yleisesti tähän alueeseen, eivätkä koko laitteen ruudun kokoon. (Peterson 2014.)

Elementtien asettelun muuttaminen on media queryn parhaita ominaisuuksia. Tämä voi muuttaa mitä tahansa CSS-arvoa ja ylikirjoittaa aiempia. Puhelinten ruudut ovat pystysuuntaisesti korkeampia, joten elementit voidaan vaihtaa vierekkäisestä asettelusta al-

lekkaiseen. Tämä tapahtuu esimerkiksi vaihtamalla laatikon sisäiset vierekkäiset elementit arvosta ‘display: inline-block' arvoon ‘display: block’. Tällöin elementti vie kokonaisen rivin, jolloin nämä menevät allekkain.

Joissain tapauksissa elementit eivät vain toimi mobiililaitteella. Tässä tapauksessa ne voidaan poistaa näkyvistä. ‘display: none’ -arvolla elementti voidaan poistaa näkyvistä niin kuin sitä ei olisi olemassa. ’display: hidden’ piilottaa elementin, mutta jättää sen varaaman tyhjän tilan, jota voi joissain tilanteissa hyödyntää.

Tietyt luettelot kannattaa pienentää mobiililaitteilla. Esimerkiksi painike, josta liukuu auki alavalikko kun hiiren vie tämän päälle, sekä liukuu kiinni kun hiiren vie pois päältä. Vietäessä hiiren painikkeen päälle voidaan valikon näkyvyys asettaa ‘display: none’ arvosta ‘display: inline-block' arvoon.

Reunoille tulee antaa tyhjää tilaa, jotta teksti on aseteltu siististi. Tähän on olemassa ‘padding’ ja ‘margin’ –ominaisuudet. Näissä voi käyttää ‘px’ eli pikseliarvoja, suosittelen myös prosenttiarvoja. Prosenteilla reunukset kasvavat suhteessa laatikon kokoon. Laatikon koolle voidaan antaa minimi- ja maksimiarvot, jolloin myöskään sisältö ei mene kiinni reunoihin tai liian kauas niistä, ollessaan esimerkiksi 5% irti reunasta.

Pidän hyvänä saavutuksena sitä, jos sivu näkyy oikein Nokia Lumia 520 -laitteessa selaimen Inspect-tilassa sivua testatessa. Tällöin sivu toimii varmasti lähes joka tilanteessa, kyseisen laitteen tarkkuuden ollessa vain 320x533.

5.3 Yhteensopivuus

Kehittäjän täytyy huomioida käyttää oikeita merkintöjä koodissa, jotka mahdollistavat ominaisuuksien yhteensopivuuden selainten välillä. Merkintöjen ‘-moz-’ sekä ‘-webkit-’ lisääminen tiettyjen ominaisuuksien alkuun on tarpeellista Mozillan sekä Safarin vanhojen versioiden yhteensopivuuden vuoksi, ja voi osoittautua kriittiseksi esimerkiksi ‘box-sizing’ -ominaisuuden käytössä. ‘box-sizing' CSS-ominaisuutena mahdollistaa arvon ‘box-sizing: border-box’, joka varmistaa laatikolle annettujen mittojen kuten 200x200 olevan tosia, huolimatta onko laatikon sisällä tai ulkopuolella sisennystä, marginaalia tai reunaa. Ilman selainyhteensopivia tägejä voisivat vanhojen selainversioiden käyttäjät kohdata sivuston, jossa elementtien koot eivät täsmää, jossa elementit menevät päällekkäin tai ovat muuten väärin aseteltuja.

5.4 Esimerkki

Yle.fi sivusto on mielestäni hyvä esimerkki responsiivisesta ja muutenkin tehokkaasta sivuston käyttöliittymästä. Se käyttää yksinkertaista laatikko-asetelmaa, jossa kiinnostava sisältö osuu silmään heti, tai vaatii vain muutaman rullauksen.

Se käyttää hyvin kuvia, siinä miten ne liittyvät aiheeseen ja kiinnittävät huomion, mutta myös responsiivisuuden kannalta. Kuvan sisältävät elementit pysyvät kaikissa kokoluokissa samassa kuvasuhteessa, ja näkyvät aina kokonaan, pienentyen sivun mukana. Taustakuva on täyttämässä tyhjää tilaa.

Kuvien alla on aihetta kuvaavat tekstit, jotka erottuvat väreillä. Otsikot ovat hyviä tiivistelmiä aiheesta, ja kuvan tai otsikon klikkaus vie suoraan artikkelin sivulle.

Sivusto on alusta asti rakennettu annetun maksimikoon levyiseksi keskelle ruutua, sivun näyttäen samalta useimmissa laitteissa. Jos ruudun kokoa aletaan pienentää, laatikot pienenevät tiettyyn pisteeseen asti kunnes menevät allekkain. Lopulta laatikoiden välit katoavat, antaen sisällölle tilaa. Fontit pysyvät mahdollisimman suurina ja luettavina. Haku- ja kirjautumisnapit sivun yläreunassa omaavat näkyvät ikonit, ja sivun logon klikkaaminen tai koskettaminen vie takaisin etusivulle, joka pätee kaikilla sivuilla. Yläpalkki pysyy kiinnitettynä ruudun yläreunaan, vaikka sivua skrollaa alaspäin. Nämä asiat yhdessä luovat kokonaisuuden jota on vaivatonta käyttää tilanteesta riippumatta.

6 LOPUKSI

Opinnäytetyö lähti kysymyksestä, millainen on hyvä käyttöliittymä ja miten sellaista lähdetään toteuttamaan. Tutkimus aloitettiin katsastamalla web-sivujen ja sovellusten yleisiä tyyppejä ja kehitystapoja.Tutkittuja pääaiheita olivat käyttäjäkeskeisen suunnittelun teoria ja toimintaperiaatteet, websivujen elementit ja rakenne, sekä responsiiviset kehitysmenetelmät. Opinnäytetyön tuloksena selvitettiin menetelmiä, joilla web-kehittäjä voi tehdä sivuistaan siistit ja modernit. Käyttäjäkeskeisestä suunnittelusta opittiin, kuten että sivun toimivuus tulee ensin, käyttäjää kannattaa kuunnella ja sivu pitää yksinkertaisena. On olemassa monta hyväksi todettua menettelytapaa ja kannattaakin perehtyä testattuihin menetelmiin. Selvitettiin myös kehitysmenetelmiä, joilla tehdään responsiiviset sivustot alusta alkaen kaikkiin laitteisiin ja selaimiin, ja jotka toimivat tulevaisuudessakin. Opinnäytetyö toi yhteen käytännönläheistä teoriaa ja avasi ratkaisuja, kuten miten ja missä tilanteissa menetelmiä sovelletaan ja mitä vaihtoehtoja on olemassa.

LÄHTEET

Borgen, P. 2017. Why CSS Grid is better than Bootstrap for creating layouts. https://hackernoon.com/how-css-grid-beats-bootstrap-85d5881cf163

Google Fonts. Lainattu 7.12.2019. https://fonts.google.com/about

Goel, A. 2019. 10 Best JavaScript Frameworks to Use in 2019 https://hackr.io/blog/10-best-javascript-frameworks-2019

Lowdermilk, T. 2013. User-Centered Design. https://www.dawsonera.com

MDN Web Docs. Text-rendering. Lainattu 7.12.2019. https://developer.mozilla.org/en-US/docs/Web/CSS/text-rendering

Peterson, C. 2014. Learning Responsive Web Design. https://www.dawsonera.com/readonline/9781449363697

Pop, P. 2002. Comparing Web Applications with Desktop Applications: An Empirical Study. https://backend.orbit.dtu.dk/ws/portalfiles/portal/3715402/pop_hci.pdf

Reichenstein, O. 2006. Web Design is 95% Typography. https://ia.net/topics/the-web-is-all-about-typography-period

Schaeffer, K. 2008. https://kyleschaeffer.com/css-font-size-em-vs-px-vs-pt-vs-percent

TechTerms. Markup Language Definition. Lainattu 7.12.2019. https://techterms.com/definition/markup_language

W3Schools. Lainattu 13.12.2019. https://www.w3schools.com/css/css_grid.asp

Wikipedia. HTML. Lainattu 7.12.2019. https://fi.wikipedia.org/wiki/HTML

Wikipedia. Typografia. Lainattu 14.10.2019. https://fi.wikipedia.org/wiki/Typografia

Wikipedia. Progressive Web Application. Lainattu 7.12.2019. https://en.wikipedia.org/wiki/Progressive_web_application