Haaga-Helia
ammattikorkeakoulu Oy

Kotisivujen toteutus Lumica Creativelle React-kirjastoa käyttäen

Jenni Aho

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2020

Tiivistelmä

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)
Jenni Aho

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi

Kotisivujen toteutus Lumica Creativelle React-kirjastoa käyttäen

Sivu- ja liitesivumäärä
20 + 4

Tämän toiminnallisen opinnäytetyön tarkoituksena on uusien kotisivujen toteutus Lumica Creative -yritykselle React-kirjastoa käyttäen. Kotisivujen luomisessa käytetään lisäksi styled-components:ia ja Material-UI:ta. Styled-components on kirjasto, jonka avulla kotisivujen visuaalista ilmettä on mahdollista muokata selkeästi ja yksinkertaisesti. Material-UI taas tarjoaa valmiita komponentteja, kuten ruudukoita ja nappuloita, tehden kotisivujen toteuttamisesta sujuvampaa vähentämällä kirjoitettavan koodin määrää. Lumica Creative on media-alan yritys, jonka palveluihin kuuluvat kotisivut, valokuvaus sekä videokuvaus ja ohessa artikkeleiden kirjoitus sekä kääntäminen englanti-suomi -kieliparilla. Opinnäytetyön aikana oma osaaminen React-kirjaston käytössä tulee niin ikään syventymään. Opinnäytetyön toteutukseen on varattu aikaa marraskuusta 2019 helmikuun 2020 loppuun.

Opinnäytetyö alkaa johdannolla, jossa kuvaillaan mitä opinnäytetyössä tullaan tekemään ja rajataan opinnäytetyön ulkopuolelle jäävät tehtävät. Tietoperustassa esitellään React, Material-UI ja styled-components sekä potentiaaliset vaihtoehdot styled-componentsille. Lisäksi tarkastellaan MobX:ä sekä Reduxia, jotka ovat tilanhallintatyökaluja. Opinnäytetyön empiirisessä osassa käydään läpi kotisivujen tämänhetkinen tilanne, uusien kotisivujen suunnittelu, syyt tiettyjen tekniikoiden valitsemiselle kotisivujen toteuttamiseksi sekä toteutuksen eteneminen. Lopuksi esitellään valmis tuotos ja pohditaan opinnäytetyön onnistumista.

Asiasanat
www-sivut, web-kehitys, React-kirjasto

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena on toteuttaa Lumica Creative -yritykselle uudet kotisivut, sillä nykyisissä kotisivuissa on puutteita, jotka tulevat näkyviin yrityksen keskittyessä yhden toimialan sijaan tulevaisuudessa useampaan toimialaan. Uusien kotisivujen toteutus on tarpeellista, sillä nykyiset kotisivut tarjoavat potentiaalisille asiakkaille nähtäväksi ainoastaan valokuvauspalveluiden portfolion. Videotuotanto ja web-kehitys mainitaan palveluina etusivulla sekä hinnastossa, mutta kyseisten osa-alueiden osaamista ei tuoda esiin portfolioiden muodossa. Tämän opinnäytetyön aikana toteutettavat uudet kotisivut tulevat kertomaan sivujen kävijöille enemmän Lumica Creativen osaamisesta näyttämällä konkreettisesti portfolioiden avulla toteutettuja projekteja. Niin ikään Lumica Creativen kotisivut ovat osoitus yrityksen osaamisesta, sillä yrittäjä itse on niiden ja tämän opinnäytetyön tekijä. Tällä pyritään osaltaan kasvattamaan maksavien asiakkaiden määrää.

Uusille kotisivuille tullaan tekemään laajennettu portfolio, johon kuuluu portfolion etusivu ja erilliset portfoliot valokuvaukselle, videokuvaukselle ja web-kehitykselle. Lisäksi ilmaisista kuvapankeista käyttöönotetut kuvituskuvat tullaan vaihtamaan Lumica Creativen omiin kuviin. Portfolioiden sisältö sekä uudet kuvituskuvat ovat jo olemassa eli niiden tuotanto ei sisälly tähän opinnäytetyöhön. Kotisivujen olemassa olevaa tekstisisältöä ei tulla opinnäytetyön aikana muuttamaan. Uusille kotisivuille tehdään hakukoneoptimointi, mutta se ei kuulu tämän opinnäytetyön piiriin eikä sitä siksi tässä käsitellä. Viimeisenä osana tätä opinnäytetyötä Lumica Creativelle ostetaan oma www-osoite sekä webhotelli ja uudet kotisivut siirretään nykyisestä internet-palveluntarjoajan kotisivutilasta webhotelliin.

Lumica Creative on vuonna 2018 perustettu toiminimi, jonka tarjontaan kuuluvat monipuoliset media-alan palvelut. Toimintaa on ollut jo ennen toiminimen perustamista vuodesta 2015 alkaen Ukko-laskutuspalvelun kautta. Lumica Creativen pääasialliset palvelut koostuvat kotisivujen suunnittelusta, toteutuksesta ja ylläpidosta, valokuvauksesta sekä videokuvauksesta ja niihin liittyvästä editoinnista. Pääasiallisten palveluiden ohessa Lumica Creative tarjoaa lisäksi copywriting- ja kääntäjäpalveluita sekä tekstien oikolukemista kieliparilla englanti-suomi. Asiakkaita lisäksi neuvotaan tarvittaessa sosiaalisen median tilien käytössä Facebookissa, Instagramissa sekä YouTubessa. Lumica Creative toimii pääasiassa pääkaupunkiseudulla, mutta matkustaminen asiakkaan luokse kauemmaksikin Suomen sisällä on tarvittaessa mahdollista. Tekstinkäsittelypalveluita sekä sosiaalisen median neuvontaa pystytään tarjoamaan myös etänä ympäri maailman niiden paikkariippumattoman luonteen vuoksi.

1.1 Sanasto

DOM: Document Object Model, standardi sille miten internetsivustot esitetään loogisella tavalla. DOM voidaan nähdä puuna, jonka oksia ovat kaikki sivuston elementit ja niiden ominaisuudet. Reactia käytettäessä DOMia voidaan kutsua React DOMiksi. (PronabM 2020.)

Virtuaalinen DOM: DOMia edustava malli tietokoneen muistissa, jolle voidaan tehdä muutoksia nopeammin ja kevyemmin kuin varsinaiselle DOMille. Virtuaalisesta DOMista muutokset siirretään varsinaiseen DOMiin. (React 2020a.)

Renderöiminen: Sovelluksen päivittyminen.

Uudelleenrenderöiminen: Sovelluksen päivittyminen tehtyjen muutosten jälkeen.

Komponentit: Komponentit voivat olla funktioita tai muuttujia. Ne ottavat vastaan ominaisuuksia propseina ja palauttavat elementtejä. (React 2020b.)

Elementit: Elementit ovat komponenttien osia, jotka määrittelevät mitä näytöllä näkyy (React 2020b).

Propsit: Komponentille annetut ominaisuudet.

Vanilla JavaScript: Alkuperäinen JavaScript, jonka kirjoittamiseen ei käytetä muita kirjastoja, kuten React, Angular tai Vue.

Kaksisuuntainen datan sitominen (two-way data binding): Dataa eli tietoa liikkuu molempiin suuntiin kahden lähettäjän välillä (Jain 2018).

Yksisuuntainen datan sitominen (one-way data binding): Dataa liikkuu vain yhteen suuntaan, lähettäjältä vastaanottajalle (Jain 2018).

Inline-tyylit: Tyylittely, joka on kirjoitettu suoraan elementin style-attribuuttiin sen sijaan, että se olisi eri kohdassa sivua tai jopa erillisessä tiedostossa. Esimerkiksi HTML-elementin kohdalla inline-tyyli kirjoitettaisiin näin: `<p style="color: black; font-size: 12px;">teksti</p>`. (Codecademy 2020.)

2 Käytetyt tekniikat

Tässä luvussa perehdytään tarkemmin React-kirjastoon ja verrataan sitä Vueen sekä Angulariin, jotka ovat niin ikään JavaScript-kirjastoja. Seuraavaksi käydään läpi Material-UI:n ominaisuudet ja tutustutaan styled-componentsiin. Styled-componentsin jälkeen tarkastellaan aphroditea, emotionia sekä radiumia, jotka ovat styled-componentsin tavoin CSS-in-JavaScript -kirjastoja. Lopuksi käydään läpi tilanhallintatyökalut MobX ja Redux.

2.1 React

React on Facebookin suunnittelema komponenteista rakentuva JavaScript-kirjasto, jonka tarkoituksena on yksinkertaistaa sekä nopeuttaa koodin kirjoittamista ja helpottaa monimuotoisten interaktiivisten käyttöliittymien luomista. React-kirjastoa käytettäessä uusien ominaisuuksien lisääminen olemassa olevaan projektiin on suoraviivaista eikä vaadi kaiken olemassa olevan koodin uudelleenkirjoittamista ja muiden kirjastojen sekä viitekehysten (framework) käyttäminen yhdessä Reactin kanssa on sujuvaa. React julkaistiin vuonna 2013 ja se aiheutti A M:n & Sonpatkin (2017) sanoin sekä innostusta että inhoa web-kehittäjissä, sillä React haastoi yleisiä käytäntöjä ja tarjoaa uusia sekä vaihtoehtoisia tapoja sovellusten toteuttamiseksi. (A M & Sonpatki 2017; Facebook 2019; React 2020b; Thinkwik 2017.)

Sovellusten toteuttaminen Reactilla tekee niistä nopeita reagoimaan käyttäjän sovelluksessa suorittamiin toimintoihin, mikä lisää käyttäjäystävällisyyttä. A M:n & Sonpatkin (2017) mukaan yhtenä Reactin periaatteista on, että DOMin muokkaaminen on raskasta ja pitäisi sen vuoksi minimoida. React ratkaisee tämän käyttämällä virtuaalista DOMia renderöimisessä oikean DOMin sijaan. Virtuaalisen DOMin ja oikean DOMin erojen perusteella oikea DOM muokataan uuden tilan mukaiseksi. React DOM päivittää ainoastaan elementit, jotka ovat muuttuneet viimeisestä tallennetusta tilasta. Reactin nopeudesta on osaltaan vastuussa juuri sen käyttämä virtuaalinen DOM. (A M & Sonpatki 2017; Facebook 2019; React 2020b; Thinkwik 2017.)

Reactin kirjoittamiseen käytetään JSX-syntaksia eli kirjoitustapaa, joka on JavaScript-laajennus. JSX-lyhenne tulee sanoista JavaScript XML. Näin ollen sillä on JavaScriptin ominaisuudet, mutta kirjoitustapa on erilainen. JSX tekee sekä koodin kirjoittamisesta että lukemisesta helpompaa ja nopeampaa, minkä lisäksi React-komponentit on mahdollista jäsennellä kuin HTML-elementit ja sen käyttö edesauttaa selkeämpien virheilmoituksien saamisessa. Sovelluksen kokoonpanovaiheessa JSX-syntaksilla kirjoitetut tiedostot muunnetaan automaattisesti JavaScript-tiedostoksi. Koodia kirjoitettaessa komponenttien nimeämisessä käytetään camelCase-nimeämistapaa, mikä tarkoittaa useasta eri sanasta

koostuvan nimen sanojen kirjoittamista isolla alkukirjaimella lukuun ottamatta sen ensimmäistä sanaa. (A M & Sonpatki 2017; React, 2020.)

Reactin perustana toimivat komponentit, jotka rakentuvat elementeistä. Elementit puolestaan ovat objekteja, jotka kertovat mitä näytöllä näkyy. Yksilöllisesti toimivat komponentit tarjoavat ohjelmoijalle mahdollisuuden jakaa isokin projekti pieniin osiin, tässä tapauksessa komponentteihin, ja viedä projektia eteenpäin komponentti kerrallaan. Juuri tämä tekee koodin kirjoittamisesta helpompaa sekä nopeampaa. Toinen Reactin keskeinen ominaisuus ovat propsit, joiden avulla komponentteihin voidaan tuoda itse määriteltyä tietoa ominaisuuksien muodossa. Reactin oman sisäänrakennetun tilan (state) avulla kaikki tietyllä sivustolla tapahtuvat muutokset tallennetaan yhteen paikkaan eli tilaan. Tällainen toimintatapa tekee yksittäisten web-sovellusten tekemisestä todella yksinkertaista, sillä kaikki tiedot haetaan samasta paikasta eli tilasta, johon ne on tallennettu. (React 2020c; Thinkwik 2017.)

Sovelluksien toteuttaminen Reactilla tekee niistä turvallisempia verrattuna esimerkiksi pelkällä HTML:llä tai Vanilla JavaScriptilla toteutettuihin sivustoihin verrattuna, kun kyseessä on esimerkiksi täytettäviä lomakkeita sisältävä sivusto. Reactin DOM varmistaa, että sivustolla vierailevan kävijän täyttäessä lomakkeita sivustolle ei voida tehdä lomakkeen kautta XSS eli cross-site scripting -iskua muuntamalla lomakkeiden sisällön merkkijonoiksi eli stringeiksi. Tällainen isku on mahdollista toteuttaa syöttämällä lomakkeeseen koodia, joka toimii sivulla vierailevan kävijän internetselaimessa muuttaen sivuston toimintaa esimerkiksi rikkomalla kävijän yksityisyyssuojaa. XSS:n liittyvä heikkous on olemassa HTML:n ja Vanilla JavaScriptin kanssa toimittaessa. React puolestaan on varautunut sitä vastaan edellä mainituin tavoin. (React 2020d; Veracode 2020.)

Vanilla JavaScriptiin verrattuna React tekee virheiden löytämisestä ja korjaamisesta helpompaa. Kun projektissa on virhe, selain näyttää virheilmoituksen heti ja kertoo mistä tiedostosta ja miltä riviltä kyseinen virhe löytyy. Vanilla JavaScriptia käytettäessä käyttäjä sen sijaan pääsee katsomaan työtään selaimella ilman virheilmoitusta, jolloin tiedot mahdollisista virheistä täytyy itse muistaa katsoa konsolista. React varmistaa siis käyttäjän kirjoittavan toimivaa koodia. (React 2020e.)

2.2 Vaihtoehdot Reactille

React ei ole ainoa vaihtoehto, kun tarvitaan JavaScript-kirjastoa. Kyseisiä kirjastoja on useita muitakin ja niiden piirteet eroavat osittain toisistaan, jolloin voidaan valita projektin

tarpeita parhaiten vastaava kirjasto käytettäväksi. Kaksi vaihtoehtoa Reactille ovat Vue ja Angular.

Evan You loi Vuen alun perin luodakseen nopeasti prototyyppejä suurta käyttöliittymäprojektia. Kyseiseen tehtävään ei ollut valmiiksi tarjolla työkalua Youn työskennellessä Google Creative Labsissa, joten You loi Vuen täyttämään tämän tyhjiön. Vuen käyttötarkoitus on muovautunut ja Filipovan (2016) mukaan sitä käytetään nykyään monipuolisten skaalautuvien ja reaktiivisten web-sovellusten toteuttamiseen. Vuen käyttöönotto ei vaadi kirjastojen tai riippuvuuksien (dependencies) asentamista kuten React. Vue otetaan käyttöön asettamalla vue.js-tiedosto HTML-sivun koodiin ja se on heti käytettävissä. Yksi Vuen eduista on, että se on suhteellisen yksinkertainen ja sen koodia on helppo oppia tuottamaan. Projekteja on mahdollista toteuttaa käyttäen pelkkää Vueta tai yhdistämällä Vue johonkin toiseen kirjastoon tai viitekehykseen. (Filipova 2016.)

Vue tukee yleisimpiä integroitujen kehitysympäristöjen (integrated development environment, IDE) lisäosia ja sen avulla voidaan luoda laajennuksia, joita on mahdollista liittää projekteihin sovellusten muokatun toiminnan lisäämiseksi. Globaalisti määriteltyjä muuttujia on mahdollista muokata Vueta käytettäessä, jolloin voidaan luoda tai editoida tyylejä, jotka ovat vain tietyn komponentin käytössä. Kun koko laajuudessa tietty ominaisuus muuttuu, vain kyseisen ominaisuuden tarkkailija arvioidaan uudelleen. (Filipova 2016.)

Vuessa ja Reactissa on useita samankaltaisuuksia. Molemmat käyttävät virtuaalista DOMia, tosin Vue alkoi käyttää sitä vasta toisesta versiosta eteenpäin. Tämän lisäksi sekä Vue että React rakentuvat uudelleenkäytettävistä komponenteista ja käsittelevät reaktiivista dataa. (Filipova 2016.)

Reactista poiketen Vue ei vaadi tai edes suosittele kirjoittamaan koodia tietyllä syntaksilla vaan koodin voi jakaa HTML-osioihin, CSS-osioihin ja Vanilla JavaScript-osioihin aivan kuten HTML-sivuja tehdessä. Vanilla JavaScript voidaan kirjoittaa erilliseen tiedostoon tai samaan tiedostoon muun koodin kanssa. Vue käyttää kaksisuuntaista datan sitomista (two-way data binding), kun React taas käyttää yksisuuntaista (one-way data binding). Näistä kaksisuuntainen on hankalampi ja aiheuttaa todennäköisemmin ongelmia isossa projektissa. (Filipova 2016; Jain 2018.)

Angular on vuonna 2010 julkaistu Googlen ylläpitämä JavaScript-kirjasto, josta voidaan käyttää myös nimitystä AngularJS. Angularin lisäksi on olemassa myös Angular 2 sekä Angular 4, Angular 5, ja niin edelleen. Angular 3 -versionumero on jätetty tarkoituksella käyttämättä. Angular 2 ja sitä seuraavat versiot eroavat alkuperäisestä Angularista täysin,

sillä Angular 2:n kohdalla kyseinen kirjasto kirjoitettiin kokonaan uudelleen. JavaScriptin sijaan se sekä sitä seuraavat versiot pohjautuvat TypeScriptiin. Tämä tarkoittaa, että haluttaessa siirtyä Angularista käyttämään esimerkiksi Angular 4, tulee koko sovellus tehdä kokonaan uudelleen. (Janbask Training 2018.)

Angular rakentuu suuresta moduuleista koostuvasta ekosysteemistä, joka on paljon suurempi kuin esimerkiksi Reactin ekosysteemi. Angularista voi löytää moduulin tekemään useimpia tehtävä. On harvinaisempaa, että jollekin tehtävälle ei olisi olemassa valmiina moduulia. Angular soveltuu nopeaan prototyyppien tuottamiseen ja se tarjoaa työkalut reitittämiselle. (Stern 2015.)

Angularin huonoihin ominaisuuksiin kuuluvat sen monimutkaisuus ja vaikeustaso, sillä suhteellisen helpon aloituksen jälkeen koodin kirjoittaminen vaikeutuu nopeasti. Lisäksi HTML on sidottuna koodiin ja sen lukeminen, ylläpito ja muutosten teko on hankalaa. Jokainen uusi komponentti yhdistyy toisiin komponentteihin yhdellä tai useammalla sidoksella, mikä hidastaa Angularilla tehtyjä sovelluksia. Hitautta lisää se, että sovellus ei tiedä mitkä osat ovat yhteydessä toisiinsa, joten se tarkistaa kaikki sidokset. Angularilla toteutetut sovellukset ovatkin paljon Reactilla tehtyjä sovelluksia hitaampia. (Stern 2015.)

Sternin (2015) mukaan Angularilla toteutettu projekti on mahdollista tehdä käyttäjäystävällisemmäksi ja nopeammaksi käyttämällä osaan koodista Reactia. Esimerkiksi Angularin kaksisuuntainen datan sitominen voi saada aikaan niin sanotusti kehän kiertämistä, jossa koodi toistaa itseään loputtomasti (infinite loop). Korvaamalla tämän Reactin yksisuuntaisella datan sidonnalla tällaisilta tilanteilta voidaan välttyä. Samassa projektissa voidaan käyttää useampaa kirjastoa, joten Angularin ja Reactin samanaikainen käyttö on mahdollista. (Stern 2015.)

2.3 Material-UI

Material-UI on Reactia varten luotu avoimeen lähdekoodiin pohjautuva käyttöliittymäkirjasto, joka toteuttaa Googlen Material Designin periaatteita. Material-UI tarjoaa laajan valikoiman valmiita komponentteja, kuten ruudukoita, laatikoita, nappuloita, menuja ja listoja sekä paljon muuta käytettäväksi React-projekteissa. Material-UI:ta käyttävät tällä hetkellä muun muassa Shutterstock, Bethesda sekä NASA. (Code Realm 2018; Material-UI 2019.)

Tarvittavat komponentit otetaan käyttöön tuomalla ne haluttuun tiedostoon, minkä jälkeen kyseessä olevien komponenttien koodi on mahdollista kopioida Material-UI:n kotisivuilta.

Nämä erilliset komponentit ovat heti käyttövalmiita eivätkä vaadi ylimääräistä koodin kirjoittamista, mikä nopeuttaa projektien toteuttamista. Komponenteissa on lisäksi tyylittelyt valmiina, mutta ne on mahdollista ohittaa esimerkiksi käyttämällä styled-componentsia, jos komponentit halutaan tyylitellä itse omaan sovellukseen paremmin sopiviksi. (Material-UI 2019; Material-UI 2020a.)

Material-UI ei vaadi erillistä CSS-tiedostoa, joka sisältäisi komponenttien tyylittelyt. Sen sijaan ne toimivat itsenäisesti tunnistaen mitkä komponentit ovat käytössä ja ladaten vain niiden tyylit. Tämä tekee Material-UI:ta käyttävän sivuston latausajasta verrattuna paljon nopeamman ja lisää näin ollen sivuston käyttäjäystävällisyyttä. (Material-UI 2020b.)

2.4 Styled-components

Styled-components on React-komponenttikirjasto, jonka avulla on mahdollista kirjoittaa CSS eli cascading style sheet -tyylittelyt suoraan Vanilla JavaScriptin sisään. Styled-components-komponentit luodaan kutsumalla styled-metodia, jonka jälkeen valitaan HTML-elementti, jolle halutaan asettaa tyylejä. (Kade 2019; Lotanna 2019; styled-components 2019.) Esimerkiksi

```
const Container = styled.div`
  color: rgb(24, 43, 62);
```

kutsuu styled-metodin ja valitsee HTML-elementin div, jolle asetetaan tyylinä väriksi rgb(24, 43, 62) eli tummansininen.

Styled-componentsia suunniteltaessa otettiin erityisesti huomioon ongelmat, joita muut samaan tarkoitukseen luodut kirjastot ovat kohdanneet ja näihin ongelmiin pyrittiin löytämään ratkaisuja. Tällaisia ongelmia ovat muun muassa luokkien nimien väärinkirjoittaminen, saman nimen antaminen usealle eri luokalle ja luokkia huonosti kuvaavat nimet. Näiden tilanteiden vuoksi luokat nähdään styled-componentissa todella epäkäytännöllistä ja tästä syystä niistä on kokonaan luovuttu siirtyen luokattomuuteen. Luokkien sijaan käytössä ovat propsit eli valmiina annetut ominaisuudet. (styled-components, 2019; Lotanna 2019.) Esimerkiksi alla olevassa koodissa Containerin taustalle määritellään väriksi sininen (blue) ja ehdoksi, että saadessaan propsin primary, Containerin tausta onkin väriltään valkoinen (white). Propsi ohittaa siis alkuperäisen määrittelyn.

```
const Container = styled.div`
  background: ${props => props.primary ? "blue" : "white"};
```

Styled-componentsin CSS-määrittely tekee tyylien lisäämisestä, muokkaamisesta ja poistamisesta vaivatonta, sillä tyylittelyjen vaikutukset on rajattu koskemaan ainoastaan sitä komponenttia, jonka yhteyteen kyseiset tyylit on liitetty. Tällöin muutokset tietyn komponentin tyylittelyihin eivät vaikuta muihin sivuston komponentteihin. Luokallista CSS:ä käytettäessä taas tyylittelyjen muutoksilla saattoi vahingossa niin sanotusti rikkoa sivuston ulkoasun, koska luokat liittivät komponenttien tyylit toisiinsa ja muutos yhteen luokkaan saattoi aiheuttaa epätoivotun muutoksen useaan komponenttiin. (Kade 2019.)

Styled-componentsin avulla komponenttien uudelleen käyttäminen on helppoa, sillä komponentteihin on määritelty tyylit niitä luotaessa. Tyylit ovat sen jälkeen ikään kuin kiinteä osa komponenttia eikä niitä tarvitse määritellä uudestaan joka kerta, kun kyseistä komponenttia käytetään. (Kade 2019.) Esimerkiksi haluttaessa kaikille sivun Container-nimisille osioille tietyn suuruinen padding eli osion rajojen sisäpuolella oleva tyhjä tila, kirjoitetaan

```
const Container = styled.div`
  padding-right: 10%;
  padding-left: 10%;
  padding-top: 2%
  padding-bottom: 5%;
```

minkä jälkeen tämä tyylittely koskee kaikkia sivun Container-osioita. Mikäli jollekin osiolle halutaan tästä poikkeava tyylittely, on sen toteuttamiseksi kaksi vaihtoehtoa. Tulee joko luoda uusi osio eri nimellä ja antaa sille halutut tyylimäärittelyt tai käyttää propseja, joiden avulla vanhan tyylittelyn voi ohittaa. Tämä käytiin läpi aikaisemmassa esimerkissä.

Styled-components lisää koodin kirjoittamisen dynaamisuutta tarjoamalla mahdollisuuden luoda ja käyttää teemoja. Luomalla teemakomponentin ja kirjoittamalla siihen tyylittelyjä, on kyseiset tyylittelyt mahdollista jakaa kaikille sivuston sivuille ja halutuille komponenteille. Jos teemalle esimerkiksi määritellään väriksi valkoinen kirjoittamalla color: white voidaan kyseistä väriä käyttää muissakin komponenteissa ilman tarvetta kirjoittaa jokaiselle valkoiseksi tyyliteltävälle komponentille erikseen color: white. Halutuille komponenteille voidaan sen sijaan antaa propsina eli ominaisuutena vain color. Tämä helpottaa tyylittelyjen päivittämistä, sillä haluttaessa vaihtaa valkoinen väri vaikkapa punaiseksi, on muutos tehtävä ainoastaan teemakomponenttiin. (Kade 2019; Bertoli 2017.)

2.5 Muut CSS-in-JavaScript-kirjastot

Styled-componentsin lisäksi on olemassa useita muita kirjastoja, joiden avulla voidaan toteuttaa komponenttien tyylittelyjä CSS-in-JavaScriptiä käyttäen. Tällaisia ovat muun muassa aphrodite, emotion sekä radium. Eri CSS-in-JavaScript-kirjastoissa voi olla samankaltaisuuksia keskenään, esimerkiksi aphrodite, emotion ja radium tukevat kaikki mediakyselyitä ja niiden avulla on mahdollista luoda animaatioita. Yhtäläisyyksien lisäksi näistä kirjastoista löytyy monia eroavaisuuksiakin. (Ho 2016; Lotanna 2019; Saring 2018).

Aphrodite on Khan Academyn kehittämä CSS-in-JavaScript-kirjasto, joka on toimiva valinta myös muille kuin Reactilla toteutetuille projekteille. Aphroditen hyvinä ominaisuuksina pidetään sen helppolukuisuutta erilaisia tarkastelutyökaluja käytettäessä, mahdollisuutta toteuttaa CSS-animaatioita sekä ohittaa sisäkkäisiä luokkia. Kolmansien osapuolien komponenttien käyttäminen on niin ikään tehty sujuvaksi. Aphrodite muuntaa tyylit luokiksi ja käyttää luokka-attribuuttia tyylittelemiseen. (Jordão 2018; Saring 2018.)

Emotion on Sunil Pain glam-kirjastoon pohjautuva CSS-in-JavaScript-kirjasto, jolla tyylittely on mahdollista toteuttaa käyttämällä stringejä eli tekstiä tai objekteja. Teemojen käyttö on lisäksi mahdollista aivan kuten styled-componentsissakin. Emotion tarjoaa käyttäjälle useita ominaisuuksia kuten lähdekartat, testauksen apuohjelmat ja leimat. Emotionin (2019) mukaan sen kirjaston käyttöönotto ei vaadi erityisiä asetuksia, laajennusten asentamista tai muutosten tekemistä olemassa oleviin määrittelyihin. Emotionin käyttäminen tyylittelyssä vähentää ylimääräisen ja turhan tekstin muodostumista komponentteja luotaessa ja tyyliteltäessä. (emotion 2019.)

Emotionia on mahdollista käyttää tilanteissa, joissa sovelluksen konfigurointia on rajoitettu tai se on kokonaan estetty, kuten esimerkiksi Create React App:n kanssa. Emotionia ei kuitenkaan ole sidottu Reactiin vaan sitä voidaan käyttää muissakin projekteissa, jolloin se toimii hieman eri tavalla. Se muun muassa tarjoaa Reactille serveripuolen renderöimisen ilman erillistä konfigurointia, mikä ei muille projekteille ole saatavilla. (emotion 2019.)

FormidableLabsin luoma radium on tarkoitettu inline-tyylien hallinnointiin Reactilla käyttämättä CSS:ä. Radiumin vahvuuksia CSS-in-JavaScript-kirjastona ovat sen tuki käyttöliittymille ja abstrakteille käsitteille, joiden käyttöön inline-tyylit eivät tavallisesti mukaudu. Tyylittelyyn radium käyttää HTML:n style-attribuuttia ja se sallii Reactin tavoin propsien käytön. Suurimman osan toiminnallisuuksistaan radium saa laajennuksina. (Saring 2018.)

Yksi radiumin suurimmista rajoitteista on, että se tukee vain muutamia selaimen tilatyylejä, joihin kuuluvat :hover, :focus ja :active. Mikäli tilatyylejä tarvitaan enemmän, tulee niitä varten luoda erillinen key prop ja tilat täytyy ottaa käyttöön manuaalisesti. Hon (2016) mukaan tilatyylien lisäksi animaatioiden toteutus toimii radiumilla huonommin kuin aikaisemmin mainitulla Aphroditella. Radiumia pidetään myös monimutkaisena kirjastona etenkin aloittelevan web-kehittäjän käytettäväksi. (Ho 2016; Hohenberger 2017; radium 2019.)

2.6 Tilanhallinta MobX:llä ja Reduxilla

Tilanhallinnan avulla on mahdollista tietää, missä tilassa sovellus, tässä tapauksessa kotisivut, ovat. Tilaan tallentuvat kaikki kotisivuilla vierailevan käyttäjän tekemät toiminnot, esimerkiksi linkkien klikkaamiset. Tilanhallintaa käyttämällä on helpompi seurata koodista mitä sovelluksessa on tehty, mutta ilman tilanhallintatyökalua sovelluksen tila on mahdollista päätellä niin ikään DOMista. (Moss 2020.)

Vaikka Reactissa on sisäänrakennettuna oma tilanhallinta this.state ja this.setState() -toiminnoilla, on tilanhallintaa varten luotu muitakin vaihtoehtoja. Syynä tähän on muun muassa Reactin oman tilanhallinnan kankeus. React ei esimerkiksi aina renderöi komponentteja heti muutosten tapahtuessa vaan se pyrkii suorittamaan useamman muutoksen kerralla. Tämä tarkoittaa, että käyttäjän sovelluksessa tekemät valinnat eivät välttämättä astu voimaan heti valinnan jälkeen vaan ne kaikki saattavat tulla näkyviin yhtä aikaa. Tilanhallinta voi muuttua niin ikään sekavaksi suurissa projekteissa, joissa komponentin täytyy jakaa tilansa muiden komponenttien kanssa tai muuttaa muiden komponenttien tilaa. Tällöin tilanhallinnan suorittaminen siihen erikoistuneella kirjastolla, kuten MobX:llä tai Reduxilla, auttaa selkeyttämään koodia. MobX:ä ja Reduxia ei ole sidottu käytettäviksi pelkästään Reactin kanssa, vaan ne palvelevat muitakin JavaScript-kirjastoja, kuten Angularia tai Vueta. (React 2020f; Wieruch 2017.)

MobX käyttää toiminnallista reaktiivista ohjelmointia (functional reactive programming, FRP), mikä tekee tilanhallinnasta yksinkertaista ja skaalautuvaa. Reactin kanssa käytettynä MobX tarjoaa säilytysmekanismin ja päivittää sovelluksen tilaa, jota React sitten käyttää. MobX koostuu muutamasta pääkonseptista, joita ovat havaittava tila, käyttäjän määrittelemät automaattisesti johdettavat arvot sekä reaktiot, joihin kuuluvat esimerkiksi ilmoituksen kirjoittaminen konsoliin tai verkkopyyntöjen tekeminen, ja toiminnot. Toimintojen osalta tapahtumat voidaan suorittaa flux-arkkitehtuurin tavoin, reaktiivista ohjelmointia käyttäen tai yksinkertaisimmillaan onClick-tapahtumana eli klikkaaminen saa aikaan toiminnon suorittamisen. (MobX 2019.)

Tilattomat (stateless) funktiot on mahdollista muuttaa reaktiivisiksi komponenteiksi lisäämällä niihin MobX:n observer-funktio ja haluttaessa decorator. Observer eli tarkkailija tekee komponentin tilasta näkyvän MobX:lle, jolloin se pystyy reagoimaan komponentin tarkkailtaviin ominaisuuksiin seurattavan funktion toiminnan aikana. Decorator taas varmistaa, että komponentit uudelleenrenderöidään ainoastaan silloin, kun niitä on päivitetty. Renderöinti kuitenkin suoritetaan heti, kun komponentteja on muutettu toisin kuin Reactin omalla tilanhallinnalla, kuten edellä mainittu. (MobX 2019.)

Redux on JavaScript-sovelluksille tehty tilanhallintatyökalu, jota voidaan käyttää yhdessä Reactin kanssa, mutta se toimii niin ikään muita kirjastoja käytettäessä. Redux pohjautuu flux-arkkitehtuuriin ja toiminnalliseen ohjelmointiin (functional programming). Se tallentaa tilan yhteen varastoon ja käyttää tilan päivittämiseen puhtaita funktioita. Kuten Wieruch (2017) selittää, puhdas funktio on funktio, joka palauttaa aina saman tuloksen saadessaan samat arvot muuttujina. Esimerkki tästä on 1+1, josta tulos on aina 2. Tila itsessään on muuttumaton, joten tilan vaihtuessa Redux palauttaa aina uuden tilan. (Wieruch 2017.)

Reduxia käytettäessä tilaa ei muuteta suoraan, vaan muutokset määritellään toimintojen avulla. Reducer-nimiseen funktioon määritellään muutokset, jotka määräävät miten kukin toiminto muuttaa koko sovelluksen tilaa. Yksi reducer saattaa riittää pienen sovelluksen tilanhallintaan, mutta isoissa projekteissa niitä voi olla useita. Redux soveltuu paremmin suuriin projekteihin, minkä vuoksi sen toiminnallisuus voidaan nähdä liian monimutkaisena ja vaativana pienten sovellusten kohdalla. (Redux 2020.)

Reduxin kanssa on mahdollista käyttää kehitystyökaluja, jotka pystyvät selvittämään jokaisen muutoksen aina sen aiheuttaneeseen toimintoon asti, mikä tekee ongelmienratkaisusta helpompaa. Reduxiin on saatavilla Redux Toolkit-lisäosa, jolla itse Reduxia voidaan käyttää tehokkaammin ja sitä suositellaan käytettäväksi logiikkaa kirjoitettaessa. (Redux 2020.)

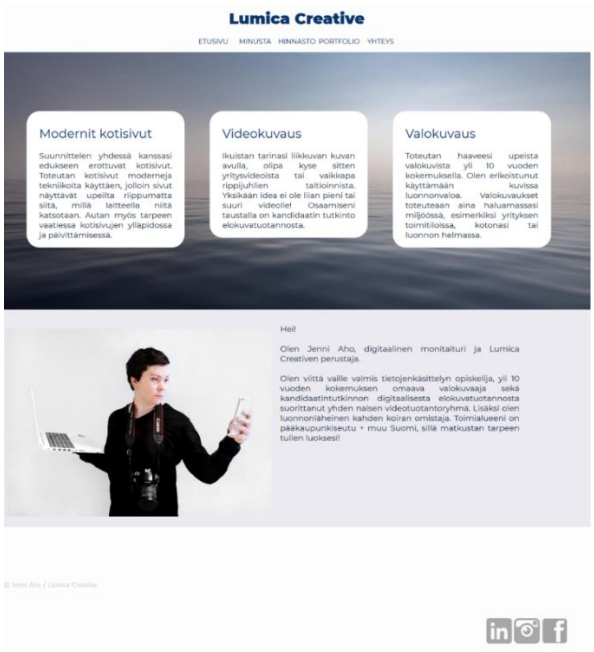
Tilanhallintatyökalujen parissa ensi kertaa työskentelevälle MobX nähdään sopivampana vaihtoehtona, sillä Redux on laadultaan vaativampi ja sen toiminnan oppiminen on pidempi prosessi. (Wieruch 2017.)

3 Lumica Creativen uusien kotisivujen toteuttamisprosessi

Lumica Creativen uusien kotisivujen toteuttamisprosessi aloitettiin nykyisten kotisivujen kriittisellä tarkastelulla tarkoituksena selvittää mitä ominaisuuksia sivustolle tulee lisätä ja mitä tulisi muuttaa erilaisiksi. Uudet kotisivut suunniteltiin piirtämällä jokaisen sivun tuleva ulkoasu paperille sekä miettimällä, mitkä tekniikat toimisivat parhaiten niiden toteuttamiseksi. Kyseisten tekniikoiden valinta perusteltiin. Lopuksi kotisivujen rakentaminen käytiin läpi askel askeleelta.

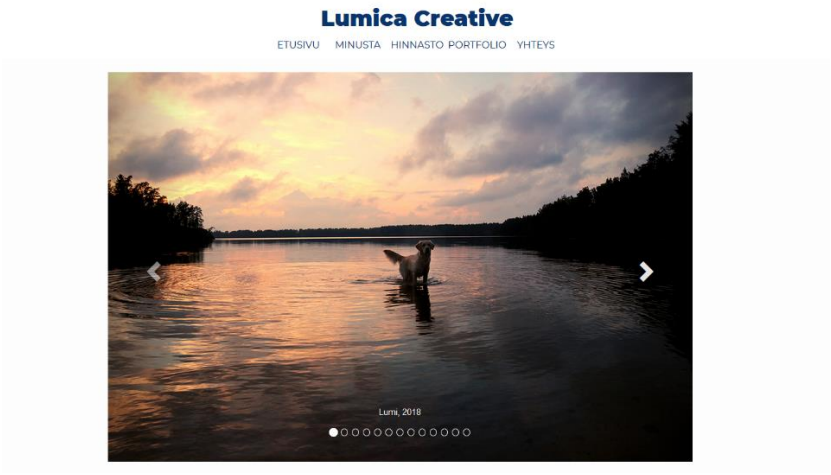
3.1 Kotisivujen tämänhetkinen tila – vanhat kotisivut

Lumica Creativen tämänhetkiset kotisivut nähdään kuvassa 1. Kotisivut on toteutettu Visual Studio Code -ohjelmalla käyttäen HTML:ä, CSS:ä sekä Vanilla JavaSriptiä. Tämänhetkiset eli vanhat kotisivut on toteuttanut opinnäytetyön tekijä. Kotisivujen toteutus Visual Studio Codea käyttäen on johtunut mieltymyksestä kyseisen ohjelman käytettävyyteen, kun taas HTML, CSS ja Vanilla JavaScript kuvastavat opinnäytetyön tekijän aiempaa osaamista ja vahvuuksia, minkä vuoksi kotisivut alun perin toteutettiin näillä tekniikoilla.



Kuva 1. Lumica Creativen vanhat kotisivut (Lumica Creative 2019)

Tällä hetkellä Lumica Creativen kotisivut ovat sisällöllisesti suurelta osin onnistuneet. Isoin ongelma sisällön osalta on, että kotisivut eivät tarjoa laajaa portfoliota kaikista pääpalveluista, sillä nykyinen portfolio koostuu ainoastaan valokuvista, kuten kuvasta 2 huomataan. Syynä tähän on ollut yrityksen pääpaino valokuvauspalveluissa, mitä halutaan lähteä tasapainottamaan muilla pääasiallisilla palveluilla.



Kuva 2. Lumica Creativen vanhojen kotisivujen portfolio (Lumica Creative 2019)

Kotisivut hyödyntävät tällä hetkellä Elisa-palveluntarjoajan ilmaista kotisivutilaa, mihin on ollut syynä yrityksen kulujen minimoiminen. Kyseinen toimintatapa kuitenkin vie Lumica Creativelta uskottavuutta ja toimii lisäksi mainoksena Elisalle, sillä esimerkiksi Google-hakukoneessa Lumica Creativen hakutulos on muotoa ”Lumica Creative – Elisa”. Niin ikään yrityksen kotisivujen www-osoite http://www.elisanet.fi/jenniaho/LumicaCreative/index.html on todella pitkä Elisa-palveluntarjoajan kotisivutilalle tyypillisellä tavalla eikä se jää potentiaalisten asiakkaiden mieleen.

3.2 Syyt uusien kotisivujen toteuttamiselle

Lumica Creativen nykyisistä kotisivuista eivät tule ilmi kaikki yrityksen tarjoamat palvelut; portfoliossa esitellään ainoastaan valokuvaus eikä web-kehitykselle ja videotuotannolle ole omia portfolioita. Yrityksen sivuilla vierailevat potentiaaliset asiakkaat eivät näin ollen saa kattavaa kuvaa yrityksen palvelutarjonnasta ja osaamisesta. Uusilla kotisivuilla halutaan tarjota oleellinen informaatio yrityksestä ja sen palveluista selkeästi ja käyttäjäystävällisesti modernilla käyttöliittymällä.

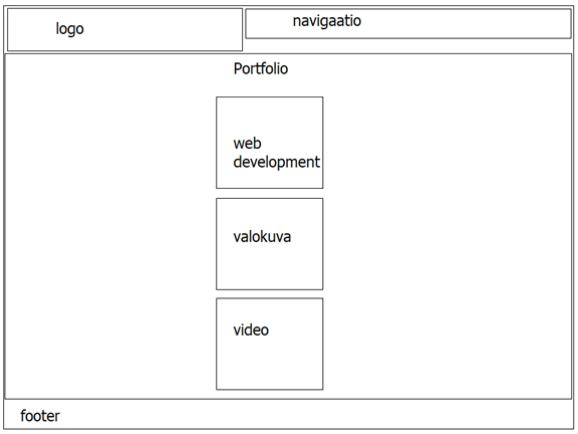
Lumica Creativen nykyiset kotisivut eivät tällä hetkellä erotu edukseen muista videokuvaukseen, valokuvaukseen tai web-kehitykseen erikoistuneiden yritysten kotisivuista. Muista erottuminen on suuressa roolissa etenkin internetissä, jossa ihmisten käytös kertoo usein kärsimättömyydestä ja kyllästymisestä visuaalisesti samankaltaisiin tai puutteellisesti toteutettuihin kotisivuihin, mikä johtaa nopeaan poistumiseen tällaisilta sivuilta. Lisäksi kilpailu on kovaa aloilla, joiden palveluista on ylitarjontaa; yksi esimerkki tästä on Lumica Creativenkin palveluihin kuuluva valokuvaus. Tällöin on tärkeää erottua muista.

Kotisivujen kohdalla muista erottumiseen vaikuttavat eniten sisältö, sekä teksti että kuvat ja video. Näiden kaikkien tulisi olla yrityksen itse tuottamia, jotta yrityksen taidot ja persoonallisuus ovat nähtävissä. Persoonallisuuden esiintuominen erottaa yrityksen muista, sillä yritykset nähdään monesti kasvottomina toimijoina ja persoonallisuuden korostaminen voi tehdä yrityksestä helpommin lähestyttävän, kun yrittäjäkin nähdään ihmisenä. Yritysesittelyn ja portfolioiden avulla tuodaan esiin yrityksen arvoja ja painopisteitä, esimerkiksi onko pienyritysten auttaminen Lumica Creativelle tärkeää vai onko kohderyhmänä suuremmat yritykset, jotka voisivat esimerkiksi ulkoistaa Lumica Creativelle osan web-kehityksestään. Portfoliot antavat kotisivuilla kävijöille lisäksi konkreettisia esimerkkejä aikaisemmista yrityksen toteuttamista projekteista. (Forbes Agency Council 2017.) Lisäksi tarkoituksena on tarjota jatkossa potentiaalisille asiakkaille Reactilla luotuja visuaalisia kotisivuja, jolloin yrityksen omien kotisivujen toteuttaminen Reactilla kertoo yrityksen osaamistasosta.

3.3 Kotisivujen suunnittelu

Saapuessaan Lumica Creativen uusille kotisivuille kävijä näkee etusivulla koko sivun kokoisena jatkuvasti toistuvan videon sekä nuolen, joka indikoi etusivun informaation sijaitsevan alempana, kuten liitteessä 1. Kotisivujen yläreunassa on jokaisella sivulla näkyvä navigaatio helpottamassa sivulta toiselle siirtymistä.

Lumica Creativen uudet kotisivut tulevat sisältämään enemmän esimerkkejä yrityksen osaamisesta visuaalisten portfolioiden muodossa. Visuaalisuuteen panostetaan uusissa kotisivuissa enemmän kuin aikaisemmissa kotisivujen versioissa käyttämällä useampia kuvia ja videoita. Uusi, kaikki yrityksen pääpalvelut kattava portfolio tullaan jakamaan kolmeen osaan, kuten kuvassa 5. Portfolion osat ovat valokuvaportfolio, videoportfolio sekä kotisivuportfolio, joiden tuleva ulkoasu on nähtävissä liitteissä 2–4. Sisältö näihin portfolioihin on jo olemassa ja se tullaan liittämään osaksi uusia kotisivuja opinnäyteyön toteutuksen aikana. Uusi laajempi portfolio antaa kattavamman kuvan Lumica Creativen monipuolisesta osaamisesta ja toimii näin ollen apuvälineenä markkinoinnissa sekä asiakashankinnassa.



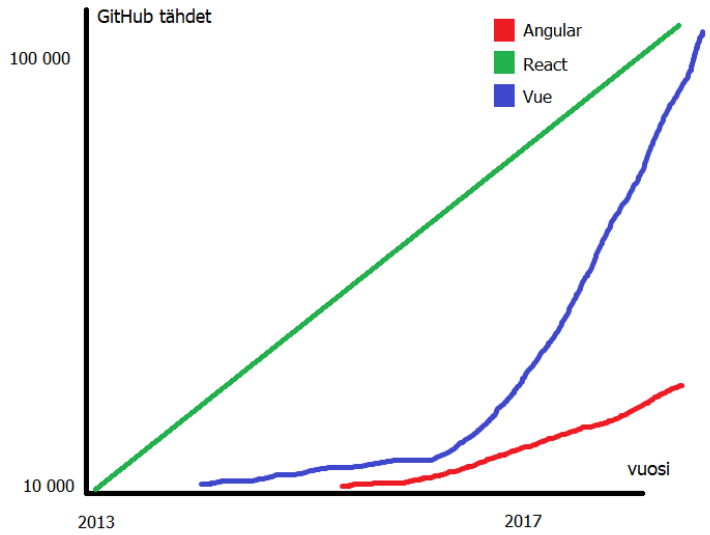
Kuva 5. Portfolion etusivu

On olennaista, että uudet kotisivut ovat responsiiviset eli ne toimivat sekä tietokoneella, tabletilla että älypuhelimella ja skaalautuvat näytön koon mukaan säilyttäen ulkoasunsa sekä toimien käyttäjäystävällisesti laitteella kuin laitteella. Tämä otetaan huomioon kotisivuja toteutettaessa kirjoittamalla koodia, joka mukauttaa kotisivujen ulkoasun käytössä olevalle laitteelle joustavien ruudukoiden ja kuvien sekä mediakyselyiden avulla. React sisältää ehdollisen renderöintiominaisuuden, jossa käytetään else-if-lausekkeita määrittelemään tilan muuttuminen eli responsiivisuuden ollessa kyseessä näytön resoluution muuttuminen. Else-if-lausekkeet määräävät milloin näytetään esimerkiksi älypuhelinversio ulkoasusta ja milloin tietokoneversio ulkoasusta. Styled-componentsin avulla responsiivisuuteen voidaan lisätä tyylittelyjä, kuten esimerkiksi rivin tyhjennyksen. Tällainen toiminto tarvitaan muunnettaessa ruudukko yksittäisiksi allekkaisiksi ruuduiksi Lumica Creativen valokuvaportfoliossa. (Chowdhury 2019; Kirkwood 2018; Reiterer 2017.)

Uudet kotisivut tullaan niiden valmistuttua lisäksi siirtämään yrityksen omaan www-osoitteeseen, mikä tulee antamaan yrityksestä ammattimaisemman kuvan potentiaalisten asiakkaiden silmissä kuin nykyinen www-osoite Elisa-palveluntarjoajan kotisivutilassa. Uusi www-osoite sekä webhotelli ostetaan Bluehost-palveluntarjoajalta. Valinta kohdistui Bluehostiin, koska Lumica Creativella on kyseisestä palveluntarjoajasta aikaisempaa kokemusta sitä käyttävien asiakkaiden kautta. Bluehostin asiakaspalvelu on nopea ja osaava vastaamaan ongelmatilanteisiin, palvelun hinnat ovat edulliset ja se tukee Reactilla toteutettuja kotisivuja. Kotisivujen ylläpito ja päivittäminen ovat opinnäytetyön valmistumisen jälkeen yrittäjän vastuulla.

3.4 Syyt React-kirjaston valitsemiseksi

React valikoitui käytettäväksi tässä opinnäytetyössä, koska se on moderni ja kevyt tapa luoda kotisivut. Reactin Create React App helpottaa erilaisten projektien, tässä tapauksessa kotisivujen, luomista konfiguroimalla applikaatioympäristön tekijälle valmiiksi, mikä niin ikään vaikutti vahvasti Reactin valintaan. Lisäksi Reactin suosio oli vaikuttavana tekijänä valinnassa. Reactia käyttäviin suuryrityksiin kuuluvat Facebookin lisäksi muun muassa Airbnb, Instagram ja Paypal sekä Uber. Reactin käyttäjämäärien kehitystä verrattuna sen kilpailijoihin Angulariin ja Vuehen on mahdollista tarkastella kuvasta 3. (React 2019; TechMagic 2018; Thinkwik 2017.)



Kuva 6. Reactin suosion kasvu (kuva luotu lähteen Kumar 2018 perusteella)

3.5 Syyt styled-componentsin valinnalle

Päätös käyttää juuri styled-componentsia muiden samaan tehtävään suunniteltujen kirjastojen, kuten esimerkiksi aphroditen, emotionin tai radiumin sijaan johtui styled-componentsin suosiosta. Muun muassa Google, lyft, Patreon, reddit, Under Armour sekä Vogue ovat tyylitelleet sivustonsa styled-componentsia käyttäen. (styled-components 2019.) Toinen tekijä styled-componentsin valinnassa oli, että muut CSS-in-JavaScript -kirjastot eivät olleet itselle edes nimellisesti tuttuja entuudestaan.

Styled-componentsin käyttöönottoa puoltavia teknisiä syitä olivat sen mukanaan tuomat useat erilaiset hyödyt. Oleellisimpia tässä projektissa ovat erillisen CSS-tiedoston puuttuminen sekä käyttämättömien tyylittelyjen eliminointi. CSS:ä käytettäessä muutokset erilliseen CSS-tiedostoon saattavat vaikuttaa muihinkin elementteihin kuin siihen, mitä oli tarkoitus muuttaa. Styled-componentsissa ei ole näitä ongelmia, sillä sen vaikutukset ovat

helpommin rajattavissa. Käyttämättömien tyylittelyjen poistaminen erillisestä CSS-tiedostosta taas voi olla hankalaa, koska täytyy olla selvillä siitä, mitkä tyylit ovat käytössä sivustolla ja mitkä voi poistaa turhina. Mikäli CSS-tiedosto on kovin suuri, voi sen päivittämiseen mennä paljonkin aikaa. Styled-componentsia käytettäessä tyylittelyt löytyvät aina sen elementin yhteydestä, jota ne koskevat. Näin ollen tiettyä elementtiä koskevat tyylit on helppo löytää eikä käyttämättömiä tyylittelyjä pääse syntymään. (Lotanna 2019.)

3.6 Uusien kotisivujen toteutus

Uusien kotisivujen toteuttaminen suoritettiin käyttäen Visual Studio Code -ohjelmaa ja Ubuntu bash -komentotulkkia. Bash on sh-yhteensopiva komentotulkki, joka pystyy suorittamaan sekä siihen suoraan kirjoitetut komennot että lukemaan komentoja myös tiedostoista. Basheja on useita ja niiden ulkoasu sekä toiminnot hieman eroavat toisistaan. Ubuntu bash:n valintaan syynä oli oma mieltymys. (Canonical 2019.)

Kotisivujen suunnitteluvaiheessa toteutettujen piirustusten pohjalta kotisivuja alettiin rakentaa osa kerrallaan. Etusivulle videon päälle suunniteltu alaspäin osoittava nuoli päätettiin jättää pois, sillä se olisi vienyt huomiota videolta. Lisäksi suunnittelupiirroksissa allekkain olleet ruudukot portfolion etusivulla muutettiin vaakatasossa vierekkäin oleviksi ruuduiksi, jotka näytön resoluution pienentyessä siirtyvät allekkain. Kyseinen muutos ulkoasussa mahdollistaa sivun sisällön näkemisen kerralla ja poistaa tarpeen sivun alaspäin rullaamiselle tarkasteltaessa sivustoa tietokoneen näytöltä.

Valokuvaportfolion kuvat oli tarkoitus tehdä aukeamaan pop-up-ikkunan tavoin niitä klikattaessa, mutta toteutusvaiheessa tämä todettiin turhaksi. Valokuvat näkyvät portfoliossa riittävän suurina ollakseen selkeitä, joten niitä ei tarvitse saada avattua isommiksi. Lisäksi mobiililaitteilla, joilla sivun rullaaminen tapahtuu koskettamalla näyttöä ja raahaamalla sitä, valokuvan kohdalta painaminen saisi pop-up-ikkunan aukeamaan, mikä ei ole toivottua.

Web-kehitysportfolion ulkoasua muutettiin kuvan 7 mukaisesti lisäämällä sivustojen esittelytekstit suoraan ruudukoihin toisin kuin alkuperäisessä suunnitelmassa, jossa tekstien oli tarkoitus tulla näkyviin vietäessä hiiri kuvan päälle. Tähän päädyttiin, jotta sivun kävijä ei olettaisi tekstien puuttuvan kokonaan. Lisäksi tarkasteltaessa kotisivuja mobiililaitteella ei hiiren vieminen kuvien päälle ole mahdollista, mikä saatiin niin ikään ratkaistua tällä muutoksella. Videoportfolion toteutus oltiin suunniteltu samankaltaiseksi kuin web-kehitysportfolio, joten senkin ulkoasuun tehtiin samat muutokset eli videoiden esittelytekstit lisättiin suoraan videoiden alle.

Kuva 7. Web-kehitysportfolion toteutettu ulkoasu

Videoiden lisäämiseksi kotisivuille käyttöönotettiin ReactPlayer, joka sopii kotisivujen kanssa samaan kotisivutilaan tallennettujen videoiden näyttämiseen. ReactPlayerin yhteyteen määriteltiin videoiden koko sekä muut ominaisuudet, esimerkiksi play/pause -napin näkyvyys ja kelausmahdollisuus. Aluksi videot oli tarkoitus tuoda YouTubesta kotisivuille, sillä kyseiset videot olivat jo valmiiksi YouTubessa. Tämä kokeiltiin toteuttaa käyttämällä react-youtube-kirjastoa, jonka on tarkoitus tehdä YouTubeen tallennettujen videoiden koon ja responsiivisuuden muokkaamisesta sujuvaa. React-youtube-kirjaston käyttäminen koettiin kuitenkin todella hankalaksi ja lisäksi jokaisen videon lopussa näkyi ehdotuksia muista videoista, joita katsoja voisi haluta nähdä, mikä ei ollut toivottua. Tarkoituksena ei ole viedä kävijöitä pois Lumica Creativen kotisivuilta vaan pitää vierailija mahdollisimman kauan sivustolla, joten videot päätettiin siirtää kotisivutilaan. ReactPlayer sopii myös YouTube-videoiden näyttämiseen, mutta tässä opinnäytetyössä sitä käytettiin kotisivutilaan tallennettujen videoiden kanssa. (npm 2020a, npm 2020b.)

Tyyliteltäessä kotisivuja styled-componentsin avulla huomattiin, että samaa koodia tulisi kirjoitettua jokaiselle sivulle, koska kaikilla sivuilla on samat tyylittelyt otsikoilla, tekstillä ja taustaväreillä. Tämän vuoksi päädyttiin käyttämään teemoja eli luomaan theme.js-niminen tiedosto, johon tulisivat kaikkia kotisivujen sivuja koskevat tyylittelyt. Teemat helpottavat kotisivujen ulkoasun muuttamista jatkossakin, sillä muutoksia esimerkiksi fontteihin ei tarvitse tehdä jokaiselle sivulle erikseen vaan ne voi tehdä vain theme.js-tiedostoon. Mikäli halutaan muuttaa vain yhden sivun fontin tyylittelyjä, muutos voidaan tehdä kyseisen sivun fontin ominaisuuksia muuttamalla, jolloin ne ohittavat theme.js-tiedoston sisältämät tyylittelyt. (Kade 2019; Bertoli 2017.)

Suunnitteluvaiheessa navigaatio kuvailtiin aina näkyvissä olevaksi palkiksi sivun yläreunaan, mutta muiden toimijoiden kotisivuja tutkittaessa huomattiin, että niin sanottu burger

menu -tyyppinen navigaatio olisi modernimpi ratkaisu. Kyseinen navigaatio näkyy menu-kuvakkeena sivun yläkulmassa, ja navigaatiolinkit tulevat esiin sitä klikatessa. Tällöin navigaatiokuvake muuttuu ruksin kuvaksi, jota klikkaamalla navigaatio voidaan sulkea. Burger menu otettiin käyttöön, sillä se on modernimpi vaihtoehto perinteiselle navigaatiopalkille ja tarjoaa sivustolla vieraileville mahdollisuuden suurempaan interaktiivisuuteen sivuston kanssa.

Tilanhallinta on tärkeää suurissa sovelluksissa, mutta Lumica Creativen uusien kotisivujen kohdalla sitä ei koettu tarpeelliseksi niiden suhteellisen pienen koon vuoksi. Burger menun tilanhallinnassa käytettiin Reactin omaa setState-tilanhallintaa, sillä navigaation toiminnallisuus on niin pientä. Lisäksi se on ainoa tilanhallintaa vaativa elementti uusilla kotisivuilla, joten vaativamman tilanhallintatyökalun käyttöönottoa ei nähty tarpeellisena.

Burger menun navigaatiossa kohdattiin ongelma, kun linkki etusivulle avasi etusivun sijaan tyhjän sivun. Koodissa etusivu oli kohdennettu main.js-tiedostoon alla olevan koodin mukaisesti ja tämä sai aikaan sen, että etusivu näkyi heti kun kotisivut avattiin:

```
<Route path="/" exact component={Main} />
```

Ongelma ratkaistiin kokeilun kautta. Muiden linkkien huomattiin olevan kirjoitettu seuraavassa muodossa: `<Route path="/sivunNimi/" exact component={sivunNimi} />`

Tästä syystä etusivun kohdennus muutettiin samaa kaavaa noudattavaksi, eli

```
<Route path="/main/" exact component={Main} />
```

Tämä ei kuitenkaan ratkaissut ongelmaa, sillä nyt etusivu ei tullut näkyviin heti kotisivut avattaessa. Lopulta etusivu saatiin toimimaan halutulla tavalla eli aukeamaan heti kotisivuille mentäessä ja myös navigaation linkki saatiin viemään kävijän takaisin etusivulle, kun koodiin lisättiin siinä alun perin ollut

```
<Route path="/" exact component={Main} />
```

Kotisivujen ollessa valmiit, ne koottiin Create React App:n build-komennolla, joka rakentaa kotisivujen tiedostoista optimaalisen toteutuksen julkaisua varten ja tallentaa sen projektikansiossa sijaitsevaan build-kansioon (Create React App 2020). Valmiit kotisivut siirrettiin WinSCP-sovelluksen avulla Bluehost-palveluntarjoajalta ostettuun kotisivutilaan. WinSCP

on tiedostonsiirtoon käytetty sovellus (WinSCP 2020), jonka valintaan johti aikaisempi kokemus sen käytöstä. Siirtoa varten WinSCP-sovellukselle tuli antaa palvelimen osoite, portin numero, käyttäjätunnus ja salasana. Tämän Bluehostin palvelimelle kirjautumisen onnistuttua projektikansiosta löytyvä build-kansio oli mahdollista raahata palvelimelle, minkä jälkeen sivusto oli valmis vastaanottamaan vierailijoita.

4 Yhteenveto

Kuva 8. Lumica Creativen uudet kotisivut

Opinnäytetyön lopputuloksena syntyivät uudet kotisivut Lumica Creative -yritykselle, joiden etusivu nähdään kuvassa 8. Kotisivut ovat responsiiviset ja skaalautuvat eli ne toimivat moitteettomasti sekä tietokoneella, tabletilla että älypuhelimella mukautuen käytössä olevan laitteen näytön resoluutioon eli mittoihin. Uudet kotisivut sisältävät halutun määrän portfolioita, joista on mahdollista nähdä yrittäjän osaamistaso sekä monipuoliset kyvyt toivotulla tavalla.

Uusia kotisivuja ei toteutettu täysin suunnitelman mukaisesti vaan suunniteltuun ulkoasuun tehtiin muutoksia vielä sivuston koodauksen aikana. Tehdyt muutokset nähtiin aikaisempia suunnitelmia parempina vaihtoehtoina visuaalisuutensa ja toiminnallisuutensa puolesta, kuten muutosten yhteydessä on edellisessä alaluvussa kuvailtu. Tämän opinnäytetyön lopputulos eli itse kotisivut ovat nähtävissä osoitteessa https://lumicacreative.com.

Alun perin opinnäytetyö oli tarkoitus tehdä toiselle yritykselle toimeksiantona, mutta kyseinen yritys ei ollutkaan valmis työn aloittamiseen ajankohdan koittaessa. Tämän vuoksi opinnäytetyö päätettiin tehdä opinnäytetyön tekijän yritykselle Lumica Creativelle, joka tarvitsi uudet kotisivut tässä opinnäytetyössä aiemmin käsitellyistä syistä. Kyseinen päätös nähtiin positiivisena, sillä se tarjosi mahdollisuuden hallita projektin aikataulua paremmin ilman riippuvaisuutta erillisen toimeksiantajan omasta aikataulusta.

React valittiin tuotoksen toteuttamista varten yrityksen näkökulmasta nähtyjen syiden lisäksi henkilökohtaisen osaamisen kehittämiseksi. Opinnäytetyötä aloitettaessa osaaminen koostui pitkälti HTML:stä, CSS:stä sekä Vanilla JavaScriptistä ja osaamisaluetta haluttiin laajentaa Reactiin. React nähdään kiinnostavana osa-alueena, joka tarjoaa mahdollisuuden toteuttaa tulevaisuudessa Lumica Creativen asiakkaille modernilla tekniikalla luotuja interaktiivisia kotisivuja. Niin ikään työnantajat vaativat usein React-osaamista mihin halutaan pystyä vastaamaan mahdollisena työnhakijana. Opinnäytetyön myötä oma React-osaaminen on kehittynyt paljon, sillä ennen tätä projektia osaaminen oli vasta alkeistasoa. Reactiin tutustuminen ja sen täyteen hallitsemiseen pyrkiminen jatkuu tämän opinnäytetyön jälkeenkin itsenäisenä opiskeluna.

Create React App oli erittäin hyödyllinen väline uusien kotisivujen toteuttamisessa, sillä kotisivujen luominen pystyttiin aloittamaan heti sen asentamisen jälkeen. Mikäli applikaatioympäristö olisi konfiguroitu itse, olisi mennyt paljon aikaa ennen kuin itse kotisivujen toteuttaminen olisi voitu aloittaa. Tämä osaltaan olisi hidastanut opinnäytetyön valmistumista.

Käytetyt työkalut eli React, styled-components ja Material-UI nähdään hyvinä vaihtoehtoina tämän tapaisen projektin toteuttamiselle. Kyseiset työkalut toimivat hyvin yhteen helpottaen ja nopeuttaen projektin eteenpäinviemistä ominaisuuksiensa avulla. Niiden avulla toteutetut kotisivut vastaavat suunnitelmia suurimmalta osin ja poikkeamat suunnitelmista tehtiin tarkkaan harkiten siinä uskossa, että muutokset toimivat käytännössä paremmin kuin alkuperäiset suunnitelmat. Lopputulokseen eli Lumica Creativen uusiin kotisivuihin ollaan tyytyväisiä.

Tämän opinnäytetyön aikaa vievin osa-alue oli teoriapohjan kirjoittaminen, vaikka koodin toteutus vaati niin ikään opiskelua ja vastausten etsimistä. Käytännön työ eli koodin kirjoittaminen oli omasta mielestä helpompaa ja eteni nopeammin verrattuna aiheista lukemiseen sekä tietoperustan kirjoittamiseen. Tietoperustan kirjoittamista vaikeutti epätietoisuus siitä, kuinka yksityiskohtaisesti tai syvällisesti asioista tulisi kirjoittaa. Lopputuloksena syntyi monipuolinen teksti, joka tutustuttaa lukijan useisiin erilaisiin työvälineisiin menemättä liiaksi teknisiin yksityiskohtiin. Teksti olisi mahdollisesti voinut käsitellä työvälineitä syvällisemmin, jotta niistä olisi saatu kokonaisvaltaisempi käsitys.

Lähteet

A M, V. & Sonpatki, P. 2017. ReactJS by Example - Building Modern Web Applications with React. Packt Publishing Ltd. Birmingham, UK.

Bertoli, M. 2017. React Design Patterns and Best Practices. Packt Publishing Ltd. Birmingham, UK.

Canonical 2019. Ubuntu Manpage: bash - GNU Bourne-Again SHell. Luettavissa: http://manpages.ubuntu.com/manpages/xenial/man1/bash.1.html Luettu: 5.12.2019

Chowdhury, S. 2019. React-responsive: Conditional rendering of components. Luettavissa: https://medium.com/applike/https-medium-com-applike-react-responsive-conditional-rendering-of-component-c97ab247097d Luettu: 14.1.2020

Codecademy 2020. Inline Styles in HTML. Luettavissa: https://www.codecademy.com/articles/html-inline-styles Luettu: 7.2.2020

Code Realm 2018. Meet Material-UI — your new favorite user interface library. Luettavissa: https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c Luettu: 11.11.2019

Create React App 2020. Available Scripts. Luettavissa: https://create-react-app.dev/docs/available-scripts/#npm-run-build Luettu: 18.2.2020

Emotion 2019. Emotion - Introduction. Luettavissa: https://emotion.sh/docs/introduction Luettu: 17.11.2019

Facebook 2019. React - A ES6 library for building user interfaces. Luettavissa: https://reactjs.org. Luettu: 11.11.2019

Filipova, O. 2016. Learning Vue.js 2. Packt Publishing Ltd. Birmingham, UK.

Forbes Agency Council 2017. 10 Ways To Make Your Website Stand Out From Your Competitors'. Luettavissa: https://www.forbes.com/sites/forbesagencycouncil/2017/04/21/10-ways-to-make-your-website-stand-out-from-your-competitors Luettu: 11.1.2020

Ho, J. 2016. Why I Love CSS in JS, Aphrodite vs Radium. Luettavissa: https://medium.com/@himrc/why-i-love-css-in-js-aphrodite-vs-radium-b2c9bea9a182 Luettu: 17.11.2019

Hohenberger, K. 2017. emotion. Luettavissa: https://medium.com/@tkh44/emotion-ad1c45c6d28b Luettu: 17.11.2019

Jain, S. 2018. React vs Vue: Programming Experience. Luettavissa: https://medium.com/@shashankjain16/react-vs-vue-programming-experience-21e8d394e479 Luettu: 31.1.2020

Janbask Training 2018. Difference Between AngularJs vs. Angular 2 vs. Angular 4 vs. Angular 5 vs. Angular 6. Luettavissa: https://www.janbasktraining.com/blog/angularjs-vs-angular/ Luettu: 31.1.2020

Jordão, G. 2018. Why Aphrodite became our choice for styling React apps. Luettavissa: https://medium.com/evodeck/why-aphrodite-became-our-choice-for-styling-react-apps-1f56950017f7 Luettu: 17.11.2019

Kade, C. 2019. Styled components: what, why and how? Luettavissa: https://dev.to/christopherkade/styled-component-what-why-and-how-5gh3 Luettu: 14.11.2019

Kirkwood, J. 2018. 11 powerful examples of responsive web design. Luettavissa: https://www.invisionapp.com/inside-design/examples-responsive-web-design/ Luettu: 5.12.2019

Kumar, A. 2018. React vs. Angular vs. Vue.js: A Complete Comparison Guide. Luettavissa: https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu. Luettu: 11.11.2019

Lotanna, N. 2019. 8 reasons to use styled-components. Luettavissa: https://blog.logrocket.com/8-reasons-to-use-styled-components-cf3788f0bb4d. Luettu: 11.11.2019

Lumica Creative 2019. Lumica Creative. Luettavissa: http://www.elisanet.fi/jenniaho/LumicaCreative/index.html Luettu: 5.12.2019

Material-UI 2019. Material-UI – A popular React framework. Luettavissa: https://material-ui.com/ Luettu: 13.11.2019

Material-UI 2020a. Style Library Interoperability. Luettavissa: https://material-ui.com/guides/interoperability/#styled-components Luettu: 14.1.2020

Material-UI 2020b. Usage. Luettavissa: https://material-ui.com/getting-started/usage/ Luettu: 14.1.2020

MobX 2019. Introduction MobX. Luettavissa: https://mobx.js.org/README.html Luettu: 26.11.2019

Moss, R. 2020. What is state? Why do I need to manage it? Luettavissa: https://egghead.io/articles/what-is-state-why-do-i-need-to-manage-it Luettu: 13.1.2020

npm 2020a. react-player. Luettavissa: https://www.npmjs.com/package/react-player Luettu: 17.2.2020

npm 2020b. react-youtube. Luettavissa: https://www.npmjs.com/package/react-youtube Luettu: 17.2.2020

PronabM 2020. ReactJS | ReactDOM. Luettavissa: https://www.geeksforgeeks.org/reactjs-reactdom/ Luettu: 26.1.2020

React 2019. Create a New React App. Luettavissa: https://reactjs.org/docs/create-a-new-react-app.html Luettu: 23.11.2019

React 2020a. Virtual DOM and Internals. Luettavissa: https://reactjs.org/docs/faq-internals.html Luettu: 26.1.2020

React 2020b. Components and Props. Luettavissa: https://reactjs.org/docs/components-and-props.html Luettu: 7.2.2020

React 2020c. Rendering Elements. Luettavissa: https://reactjs.org/docs/rendering-elements.html Luettu: 11.1.2020

React 2020d. Introducing JSX. Luettavissa: https://reactjs.org/docs/introducing-jsx.html Luettu: 11.1.2020

React 2020e. React A JavaScript library for building user interfaces. Luettavissa: https://reactjs.org/ Luettu: 11.1.2020

React 2020f. State and lifecycle. Luettavissa: https://reactjs.org/docs/state-and-lifecycle.html Luettu: 13.1.2020

Redux 2020. Getting Started with Redux. Luettavissa: https://redux.js.org/introduction/getting-started/ Luettu: 11.2.2020

Reiterer, A. 2017. How to create responsive UI with styled-components. Luettavissa: https://medium.com/styled-components/how-to-create-responsive-ui-with-styled-components-c6b71a3ce172 Luettu: 14.1.2020

Saring, J. 9 CSS in JS Libraries you should Know in 2019. Luettavissa: https://blog.bitsrc.io/9-css-in-js-libraries-you-should-know-in-2018-25afb4025b9b Luettu: 17.11.2019

Stern, D. 2015. Getting started with React and Angular: Understanding capabilities, interoperation, and basic implementation techniques. Infinite Skills.

styled-components 2019. styled-components: Basics. Luettavissa: https://www.styled-components.com/docs/basics#motivation Luettu: 13.11.2019

TechMagic 2018. React vs Angular vs Vue.js — What to choose in 2019? (updated). Luettavissa: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d Luettu: 23.11.2019

Thinkwik 2017. Why ReactJS is gaining so much popularity these days. Luettavissa: https://medium.com/@thinkwik/why-reactjs-is-gaining-so-much-popularity-these-days-c3aa686ec0b3 Luettu: 17.11.2019

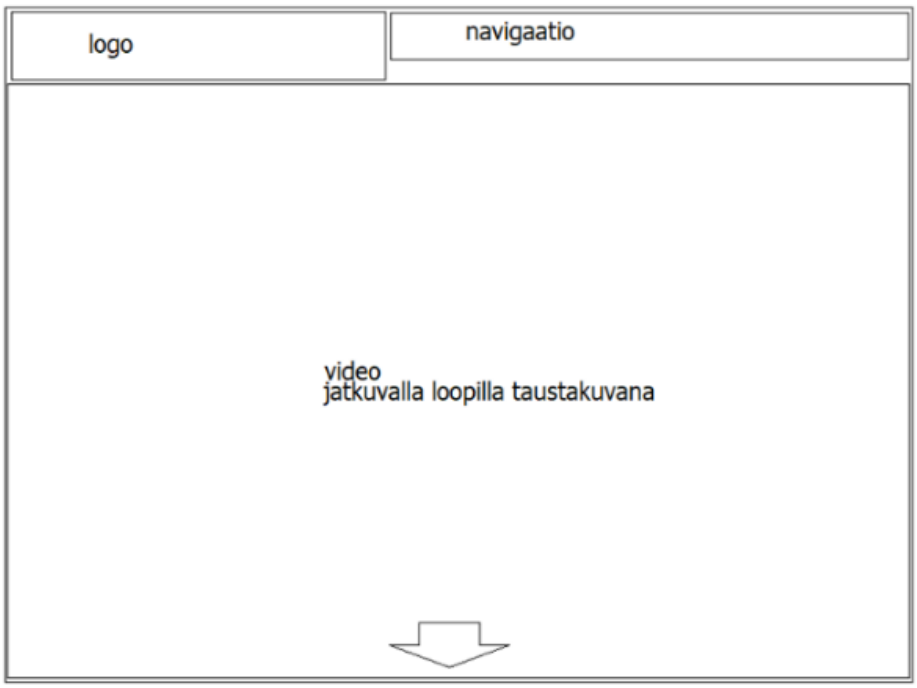
Veracode 2020. XSS - What Is Cross-Site Scripting? Luettavissa: https://www.veracode.com/security/xss Luettu: 11.1.2020

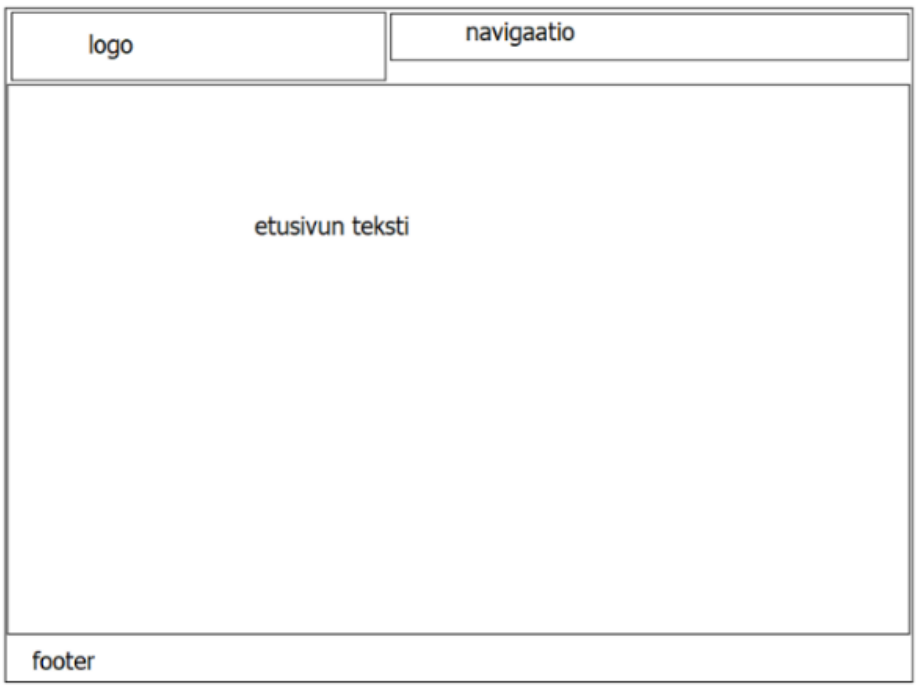
Wieruch, R. Redux vs MobX without Confusion. Luettavissa: https://www.robin-wieruch.de/redux-mobx Luettu: 13.1.2020

WinSCP 2020. Introduction. Luettavissa: https://winscp.net/eng/docs/introduction Luettu: 18.2.2020

Liitteet

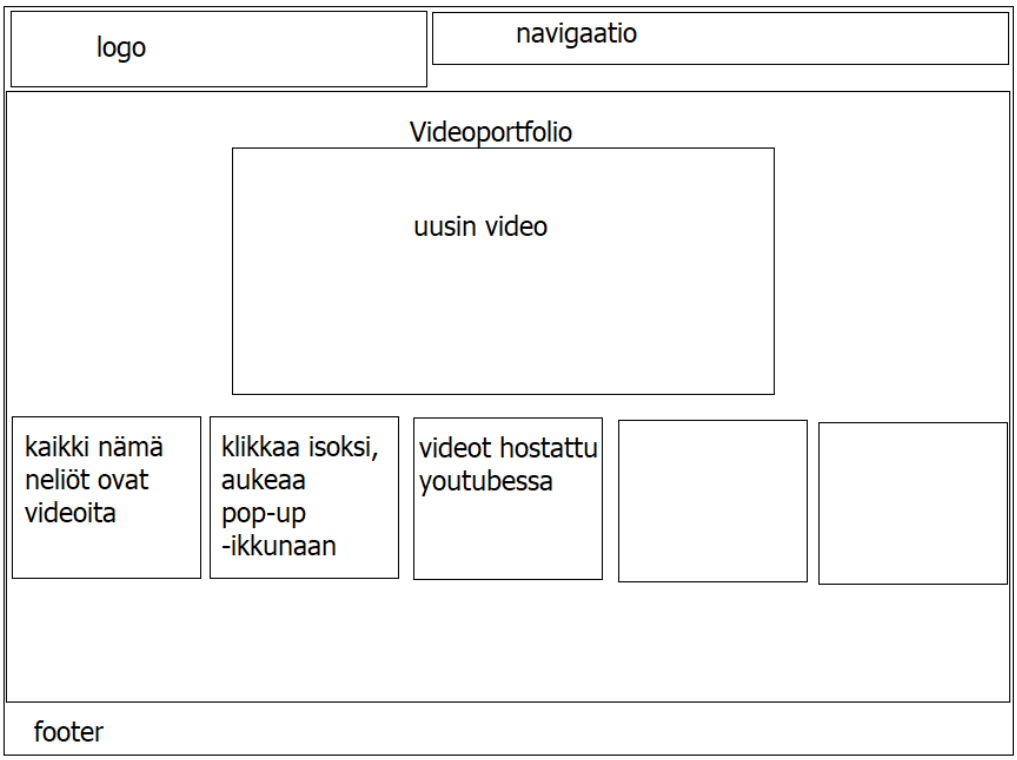
Liite 1. Etusivu





Liite 2. Valokuvausportfolio

Liite 3. Videoportfolio



Liite 4. Web-kehitysportfolio