Miia Lousaari

Modulaarinen front-end WordPress-sisällönhallintajärjestelmässä

Aloittelijaystävällinen katsaus moderniin front-endiin

Metropolia Ammattikorkeakoulu

Medianomi

Viestinnän tutkinto-ohjelma

Opinnäytetyö

13.5.2022

Tiivistelmä

Tekijä(t):	Miia Lousaari
Otsikko:	Modulaarinen front-end WordPress-sisällönhallintajärjestelmässä
Sivumäärä:	56 sivua + 7 liitettä
Aika:	13.5.2022
Tutkinto:	Medianomi
Tutkinto-ohjelma:	Viestinnän tutkinto-ohjelma
Suuntautumisvaihtoehto:	Digitaalinen viestintä
Ohjaaja(t):	Lehtori Markus Norrena

Opinnäytetyön ensisijaisena tavoitteena on lisätä omaa ymmärrystä modulaarisista front-end-teknologioista, joita WordPressissä voidaan hyödyntää. Toissijaisena tavoitteena on jakaa front-endistä kiinnostuneille tietoa aiheesta. Työn aiheen taustalla on henkilökohtainen ja ammatillinen kiinnostus WordPressiin ja moderniin front-endiin.

Työssä noudatetaan laadullisen tutkimuksen kriteereitä. Työssä käytettävät menetelmät ovat havainnollistavia ja konstruktiivisia. Käytettävät menetelmät ovat työn kannalta ideaaleja, sillä tarkoituksena on uuden koodin ja konseptien oppiminen. Toiminnallisessa osuudessa tavoitteena on luoda yhden sivun verkkosivusto ja kolme moduulia WordPressin Gutenberg-editoriin. Työn teoreettinen osuus on kattava katsaus modulaarisuuteen, moderniin front-end-kehitykseen ja WordPressin käytäntöihin. Temaattista lähestymistapaa hyödynnetään modulaarisuuden avulla punaisen langan muodostamiseksi työssä. Teoreettinen osuus keskittyy selittämään modulaarisuuden ilmiötä, verkkosivujen rakententeita ja käytäntöjä, moderneja front-end-teknologioita ja kertoo siitä, kuinka WordPress toimii.

Modulaarisuuden havainnollistamiseksi työssä kehitettiin Reactin avulla kolme moduulia WordPressin Gutenberg-editorille käytettäväksi: sisältöön, korteille ja sosiaalisen median riville. Moduuleista tuli käytettäviä, muokattavia ja esteettisesti miellyttäviä. Niiden avulla pystyy havainnollistamaan modulaarisuutta WordPressissä aloittelijaystävällisesti. Toteutuksessa käytettiin Underscores-aloittelijateemaan pohjautuvaa minimalistista teemaa, jonka avulla teeman mukauttimessa pystyy hyödyntämään ennalta asetettuja SCSS:n muuttujia.

Työn tuloksena oli lisääntynyt ymmärrys modulaarisuuden konseptista ja Wordpressin toimintatavoista sekä modernin front-endin toimintaperiaatteista. Työn aikana kehitetyt moduulit olivat opettavainen tapa oppia lisää Reactin hyödyntämisestä WordPressissä.

Avainsanat: Modulaarisuus, front-end, WordPress, Gutenberg, React, SCSS

Abstract

Author(s):	Miia Lousaari
Title:	Modular Front-End in WordPress CMS
Number of Pages:	56 pages + 7 appendices
Date:	13 May 2022
Degree:	Bachelor of Culture and Arts
Degree Programme:	Media
Specialisation option:	Digital Media
Instructor(s):	Markus Norrena, Senior Lecturer

The primary purpose of this final project is to gain more insight about modular front-end technologies in WordPress CMS. The secondary purpose is to share information about the subject for those interested in modern front-end. The subject of the work is based on personal and professional interest in WordPress and modern front-end. The project follows the criteria of qualitative research and uses illustration and demonstration together with constructive and practice-based methods. Constructive and practice-based methods are ideal methods for learning new code and concepts, which is why they are used. The objective in the practice-based section is to create a one-page website and three modules, with which modularity can be demonstrated.

The theoretical section of the project is an extensive review of modularity, modern front-end development and WordPress practices. Thematic approach is used via modularity. The section focuses on explaining modularity as a phenomenon, discusses website structures and practices and examines modern front-end technologies and provides information on how WordPress works.

For illustrating modularity and front-end technologies in WordPress, three modules for Gutenberg are developed with React: a content module, a card module and a module for social media icons. The modules are usable, customisable and aesthetically pleasing. Modules achieve their purpose of demonstrating modularity. Starter theme based on Underscores is used for the one-pager. The theme's functions are set up to use SCSS variables, and the theme's colours and font sizes can be widely controlled in Customizer.

The project resulted in increased personal knowledge of using React and modern front-end technologies in WordPress and also helped to forge a better understanding of modern web concepts, especially modularity. During the project, planned modules were constructed and can be further developed by anyone. In conclusion, modularity is a widely used concept which can be seen in the structure of the web itself, in programming languages and their reusable components, as well as in website implementation and front-end development.

Keywords: Modularity, front-end, WordPress, Gutenberg, React, SCSS

Sisällysluettelo

1 Johdanto

Modulaarisuus on viimeisen vuosikymmenen aikana noussut puheenaiheeksi verkkosivujen suunnittelun ja toteutuksen parissa toimivien keskuudessa. Useat modernit suunnittelutoimistot mainostavatkin palvelujaan moduulien ja modulaarisuuden kautta, esimerkiksi WordPressiä hyödyntävät toimijat kuten Into Digital, Karhu Helsinki ja Paper Planes. Esillä ollut modulaarisuuden konsepti sai minut miettimään, kuinka voisin oppia hyödyntämään modulaarisuutta WordPress-sivustojen toteuttamisessa. Työn avulla tavoittelenkin parempaa ymmärrystä modulaarisesta front-end-toteuttamisesta. Työn toissijaisena tarkoituksena on koostaa aloittelijaystävällisesti aiheesta tietoa sekä käytännön esimerkkejä. Kohdistan siis työn front-endistä kiinnostuneille aloitteleville kehittäjille. Koska aihe on laaja ja sisältää useita eri osa-alueita, on myös työn teoriaosuus kattava. Laajuuteen vaikuttaa työn toissijainen tarkoitus, jonka vuoksi työssä esitellään verkkosivujen toteutukseen liittyviä termistöjä ja käytäntöjä.

Opinnäytetyö on muodoltaan toiminnallinen ja työssä noudatetaan laadullisia menetelmiä. Teoriaosuutta varten lähdemateriaalia on kerätty erilaisista virallisten toimijoiden tarjoamista materiaaleista, alan portaaleista ja alan ammattilaisten kirjoituksista. Sisällön rakenteen temaattisuudella mahdollistetaan aiheeseen vapaamuotoisempaa perehtymistä, ja modulaarisuuden teeman avulla aihetta voidaan tarkastella laajasti hyödyntämällä erilaisia näkökulmia ja lähestymistapoja (Pönni 2021). Työssä hyödynnetään myös konstruktiivista tutkimusmenetelmää, jonka avulla voidaan hankkia teoreettista ja käytännön informaatiota aiheesta osallistumalla itse havainnollistettavan tuotteen tuottamiseen. (Ojasalo, Moilanen & Ritalahti 2015, 66.) Koska toiminnallinen osuus pohjautuu teknisiin oppaisiin ja koodiin, on sen sisältämän informaation todentaminen helppoa kokeilemalla koodien toimintaa itse. Toiminnalliseksi menetelmäksi on valittu havainnollistaminen. Tätä varten luodaan SCSS- ja React-teknologioita hyödyntäen mukautettavia sisältölohkoja WordPressin Gutenberg-editoriin. Havainnollistaminen tapahtuu esittelemällä toiminnallisessa osiossa luodut lohkomoduulit ja avaamalla niiden taustalla olevia toimintaperiaatteita.

Opinnäytetyön tietoperusta rakentuu luvuissa 2–4. Toinen luku perehtyy modulaarisuuden käsitteeseen, arkisiin esimerkkeihin ja historiaan, ja lisäksi luvussa käydään läpi verkkosivujen toteutukseen liittyviä oleellisia asioita, kuten termistöä. Kolmannes luku avaa front-end-kehityksen ja verkkopalvelujen toteutuksen taustalla olevia teknologioita. Neljännes luku keskittyy WordPressiin ja sen alaluvuissa esitellään WordPressin keskeisiä toimintatapoja, käsitteitä ja ominaisuuksia. Viidennessä luvussa käydään läpi toiminnallisen osion taustoja ja tavoitteita, lohkojen kehitystä ja haasteita sekä SCSS:n hyödyntämistä WordPressin ympäristössä. Samassa luvussa myös esitellään alaluvuittain sekä lohkomoduulien että SCSS:ää hyödyntävän one pagerin toteutusta ja lopputuloksia. Kuudennessa luvussa pohditaan työn aikana syntyneitä havaintoja, toiminnallisen osion tuloksia ja opittua.

2 Modulaarisuus ja verkkosivustoista yleisesti

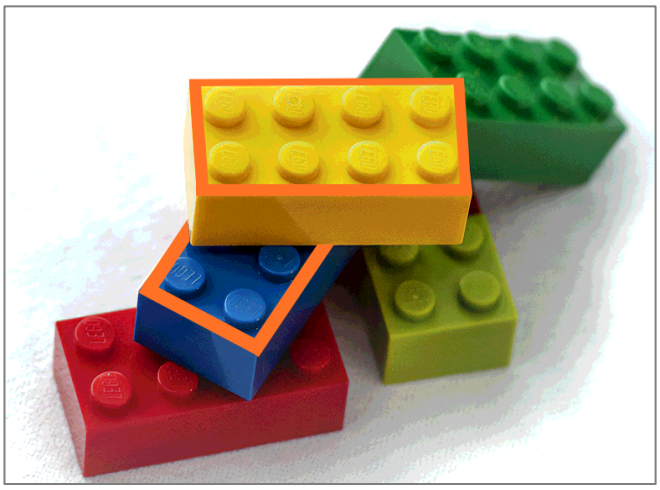
Modulaarisuus on suhteellisen yksinkertainen konsepti. Verkkosivustojen toteuttaminen on puolestaan laaja kokonaisuus, johon liittyy useita erilaisia käytäntöjä, sanastoa ja rakenteellisia seikkoja. Aiheen laajuuden johdosta seuraavissa alaluvuissa käydään kattavasti läpi modulaarisuutta sekä verkkosivujen termistöä, rakennetta, ulkoasua ja suunnitteluun liittyviä asioita, jotka ovat oleellisia myös front-end-kehityksessä.

2.1 Modulaarisuus

Sanakirjan mukaisesti moduuli tarkoittaa standardia tai mittayksikköä. Modulaarisuus puolestaan tarkoittaa jonkin asian koostuvan näistä standardoiduista yksiköistä tai mittasuhteista joustavuuden ja monipuolisen käytettävyyden vuoksi. Standardilla puolestaan tarkoitetaan normia tai vakiota. (Merriam-Webster 2022a; Merriab-Webster 2022b; Kielitoimiston Sanakirja 2021.) Tämän työn kontekstissa standardilla tarkoitetaan vakiota.

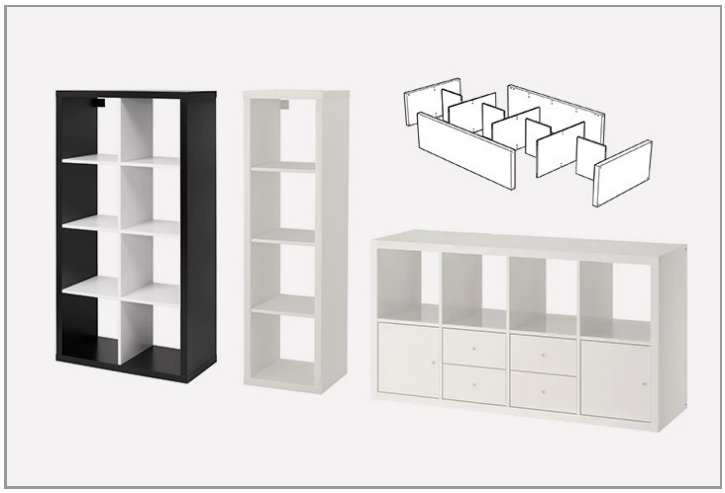
2.1.1 Arkisia esimerkkejä modulaarisuudesta

Arkisesti moduulit voidaankin kuvitella ikään kuin Lego-palikoina ja palikoiden vaihdettavissa sekä laajennettavissa olevina osasina, joilla palikoilla leikkivä voi luoda erilaisia rakennelmia tiettyjen määrättyjen standardien puitteissa. Standardeina voivat toimia esimerkiksi Legojen tapauksessa saatavilla olevat värit ja jokaisen palikan reunassa oleva tila, tämä tila on havainnollistettu kuvassa 1 oranssilla värillä. (Johnson 2021; Ramalhete 2017.)



Kuva 1. Erivärisiä Lego-palikoita. Oranssilla korostettu reunatila on Legoissa standardi, joka esiintyy samankokoisena jokaisessa palikassa. (Kuva: Theresa Muth 2021)

Toisena arkisena esimerkkinä modulaarisuudesta toimivat Ikean huonekalut, joiden osat rakennetaan itse ohjeiden mukaan itse, ensin yksikköinä, jotka myöhemmin yhdistetään koottavaan kokonaisuuteen. Joissain näissä huonekaluissa modulaarisuuden määritys täyttyy sekä suunnittelun, tuotannon ja yleisen muokattavuuden puolesta. Kuvan 2 Kallax-hyllyn kontekstissa moduuli on hyllyyn hankittava lisäosa tai laatikko, ja hylly tukee modulaarisuuden määritettä mahdollistamalla näille moduuleille vakioidun tilan sekä helpon uudelleenkäytön ja -asettelun. (Ramalhete 2017.)



Kuva 2. Ikean Kallax-moduulihylly. (Kuva: Ikea & UX Studio Team 2017)

Tuotantovaiheessa modulaarisuus tarjoaa kustannustehokkuutta, sillä osia voidaan parhaimmillaan vaihtaa, korjata ja suunnitella uudelleen vähäisellä vaivalla. Modulaarisesti suunnittelevien tulee tosin pitää mielessään moduuleihin liittyviä esteettisiä rajoituksia, sillä moduulit soveltuvat parhaiten lohkoihin, neliöihin ja kolmioihin. (Ramalhete 2017; Wiltz 2015.)

2.1.2 Modulaarinen rakentaminen historiallisesti

Modulaarisen arkkitehtuurin hyödyntäminen rakentamisessa ei ole ihmiskunnalle uutta. Jo muinaisen Rooman aikana roomalaissotilaiden kerrotaan valmistaneen linnoituksia pieniin osiin pilkottuina tehostaakseen logistiikkaa ja rakentamista. Rakentamisessa modulaarisuus yleistyi teollistumisen aikakaudella, jolloin kasvavien kaupunkien tarpeisiin tarvittiin keinoja rakentaa asuntoja kustannustehokkaasti ja nopeasti. (Gasc n.d.) Modulaarisuutta on hyödynnetty myös sanomalehtien suunnittelussa: 1960-luvun lopulla suunnittelija Frank Ariss uudisti Minneapolis Tribunen ulkoasun hyödyntämällä ruudukoita ja loi näin modulaarisen järjestelmän sanomalehden sivujen rakentamiselle. Uudistus oli suosittu, ja se modernisoi useiden lehtien tapaa rakentaa lehden sivuilla esiintyvää informaatiota näkyviin. Tekstit ja kuvat toimivat ikään kuin moduuleina, joiden avulla voidaan tehokkaasti rakentaa sisältöä esitettäväksi sanomalehden eri sivuille. (Ramalhete 2017; García 2014.) Sama tapa asetella sisältöä on

periytynyt myös digitaalisille alustoille sekä verkkosivuille, joilla usein esiintyy erilaisia ruudukkoja ja lohkoja, joita käsitellään lisää luvussa 2.1.5.

2.1.3 Modulaarisuus verkkosivustoilla

Koska verkkosivut ja -palvelut ovat ihmiskunnan historiaan nähden varsin uusia, on modulaarisuus verkkosivustoilla muodostunut ilmiöksi vahvemmin vasta tietotekniikan ja verkkoteknologioiden kehityksen myötä. Aiemmassa luvussa kerrotun päätteellä voidaan perustella, että historiaan nähden tarve modulaarisuudelle syntyy silloin, kun halutaan rakentaa nopeasti ja tehokkaasti laajoja kokonaisuuksia, jotka sisältävät muunneltavia, uudelleenkäytettäviä osia, jotka puolestaan ovat logistisesti helposti siirrettävissä. Modernit verkkosivustot pitävät usein sisällään kompleksisia komponentteja ja hyödyntävät myös useita eri teknologioita. Modulaarisuus tarjoaa sopivaa tehokkuutta erilaisten taustalla olevien teknologioiden hyödyntämiseen, sillä modulaarisuuden hyödyntämisen perimmäisenä tarkoituksena on vähentää toiston tarvetta ja helpottaa osien uudelleenkäyttöä, poistamista ja korvaamista. Modulaarisuutta voidaan hyödyntää verkkosivustojen suunnittelussa ja toteutuksessa erilaisten responsiivisten ruudukkojen ja sisältöön liittyvien laattojen sekä esimerkiksi kortien avulla. (Ramalhete 2017.)

World Wide Webin idean sanotaan myös olevan kokonaisuutena modulaarinen, sillä web koostuu lukuisista verkkosivuista, jotka puolestaan koostuvat erillisistä elementeistä, joita voidaan itsenäisesti muokata. Näin idea täyttääkin aiemmin läpikäydyn modulaarisuuden määrityksen: itsenäisistä yksiköistä koostuva, mukautettava, uudelleen hyödynnettävä ja standardeihin perustuva. (Manovich 2001, 31.)

2.1.4 Moduuli, komponentti ja elementti

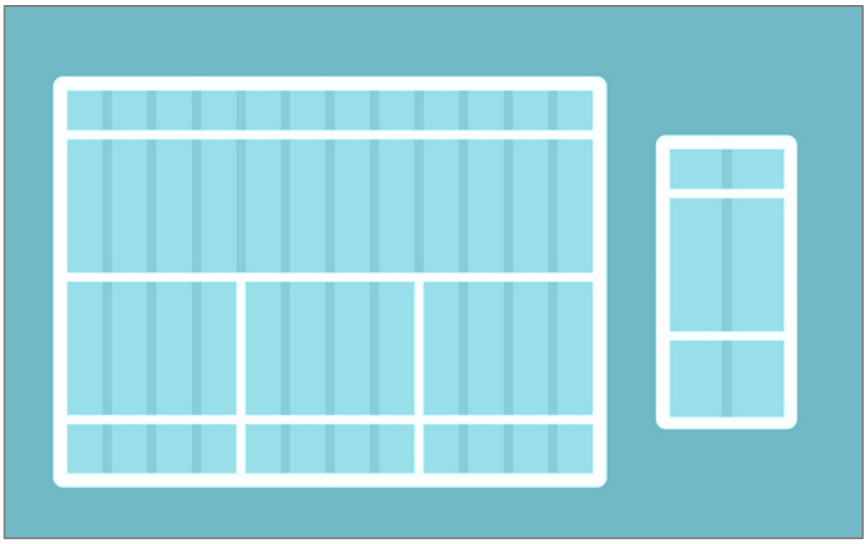
Alan ammattilaisten moduuleja käsittelevissä teksteissä esiintyy jonkin verran päällekkäisyyksiä moduulin, komponentin ja elementin määritysten välillä. UX-suunnittelija Dennis Kardysin (2014) mukaan komponentilla tarkoitetaan laajempaa kokonaisuutta, josta modulaarisen tekee standardisoitu ulkoasu, koodi,

järjestelmän sääntöihin perustuva kontrollointi sekä mukautettavuus. Graafinen suunnittelija Jenny Kowalski (2021) puolestaan määrittää komponentit sivustolle tulevina uudelleenkäytettävinä objekteina, jotka voivat olla esimerkiksi nappeja, kortteja tai mitä tahansa elementtejä, joita halutaan käyttää uudelleen. UX-suunnittelija Taurie Davis ja verkkokehittäjä Sarrah Vesselov (2019, luku 2) määrittävät komponentin yhden tai useamman elementin yhdistelmänä, joka toimii kokonaisuutena, kuten esimerkiksi lomakkeena. Elementillä tarkoitetaan käytettävissä olevia pieniä informaatiota sisältäviä paloja, kuten esimerkiksi linkkejä, tekstejä, listoja, alueita, kuvia ja painikkeita (Oracle 2009; Freedom Scientific n.d). Toisaalta taas tuotesuunnittelija Rui Ramalheten (2017) mukaan modulaarisessa verkkosuunnittelussa moduuli tarkoittaa yksinkertaisesti uudelleenkäytettävää elementtiä. Termistöjen limittäytyminen on ymmärrettävää, sillä moduulin, komponentin ja elementin määrityksissä on paljon samankaltaisuuksia.

Tässä opinnäytetyössä komponentilla tarkoitetaan Kardysin (2014) mukaisesti laajempaa kokonaisuutta, joka voi olla myös modulaarinen ja joka koostuu vähintään muutamasta elementistä. Modernien JavaScript-kirjastojen komponentit ovat usein modulaarisia. Toiminnallisessa osiossa toteutusta esiteltäessä komponentilla tarkoitetaan koodissa hyödynnettävää uudelleen käytettävää osaa. Elementillä tarkoitetaan yksittäistä tiettyä asiaa, kuten vaikkapa sivun otsikkoa, kuvaa tai tekstinostoa. Moduulilla tarkoitetaan muutoin samaa kuin lohkolla, mutta sen tulee olla vakioihin sitoutunut yksikkö, joka sisältää mukautettavuutta, joustavuutta ja sitä tulee voida käyttää uudelleen. Verkkosivujen kontekstissa moduulien voidaan katsoa olevan standardisoituneita eli vakiintuneita yksiköitä. Tämän voidaan päätellä tarkoittavan, että ollakseen modulaarinen verkkosivun tulee koostua erilaisista loogisesti toimivista uudelleenkäytettävistä komponenteista (toisin sanottuna moduuleista) ja että sen rakenteen tulee tukea modulaarisuutta, aivan kuten Ikean Kallax-hyllyn runko (kuva 2) tukee siihen liitettäviä laatikoita.

2.1.5 Ruudukot ja lohkot verkkosivustoilla

Elementtien sijoittelussa hyödynnetään jatkuvasti erilaisia lohkoja, ruudukkoja ja neliöitä sisällön rakentamisessa ja esittämisessä, mikä puolestaan tarjoaa optimaalista lähtökohtaa modulaarisuuteen. Verkkosivustoilla sisältöä voidaan esittää ruudukkojen riveissä ja kolumneissa. Suunnittelija Monica Galvan (2021) kertoo ruudukkojen auttavan ulkoasun ja hierarkian muodostamisessa digitaalisilla alustoilla. Ruudukot helpottavat elementtien sijoittelua ja ne ovat isossa osassa modernia responsiivista toteutusta. Kuvassa 3 on nähtävissä Galvanin yksinkertainen havainnollistus mahdollisesta verkkosivun ruudukkoasettelusta työpöytä- ja mobiilinäkymällä. Vasemmalla työpöytä- ja oikealla mobiilinäkymä.



Kuva 3. Havainnollistus ruudukoista työpöytä- ja mobiilinäkymässä. (Kuva: Monica Galvan 2021)

Näihin ruudukoihin voidaan sijoittaa sivuille tulevia elementtejä ja sisältöä, kuten esimerkiksi erilaisia lohkoja.

2.2 Käsitteitä liittyen verkkosivustojen toteutukseen

Modernien verkkopalvelujen suunnitteluun ja toteutukseen liittyy useita erilaisia käsitteitä, suuntauksia ja standardeja. Käsitteiden runsauden vuoksi alalukuun on valikoitu kolme myös tämän työn kannalta oleellista käsitettä: saavutettavuus, responsiivisuus ja käytettävyys.

Saavutettavuudella tarkoitetaan, että erilaisten käyttöliittymien ja teknologisten ratkaisujen tulisi palvella myös ihmisiä, joilla on erilaisia fyysisiä tai kognitiivisia vammoja tai rajoitteita. Tarkoituksena on yhdenvertaisuuden edistäminen. Saavutettavuuteen on olemassa erilaisia standardeja, kuten esimerkiksi Web Content Accessibility Guidelines (WCAG), joiden avulla pyritään tarjoamaan vähimmäisvaatimuksen mukainen käyttökokemus rajoitteita omaaville ihmisille. (Joyce 2022.) Euroopan unionin vuonna 2018 voimaan tullut saavutettavuusdirektiivi pohjautuu WCAG:iin ja vaatii, että julkisten verkkopalvelujen tulee olla saavutettavia. Julkisilla verkkopalveluilla tarkoitetaan palveluja, joiden tarjoajina toimivat esimerkiksi valtiot, kunnat ja yliopistot. Direktiivin myötä etenkin julkisten palvelujen suunnittelijoiden ja kehittäjien tulee huomioida, että verkkopalvelun sisällön tulee olla ymmärrettävää ja lähestyttävää, että otsikkojen ja nostojen on oltava loogisia ja kerrottava sisällöstään tarpeeksi ja että palvelun linkkien ja muiden sisällön rakenteiden on oltava asianmukaisesti kuvattuja. (Röksä 2018.)

Responsiivisuus on termi, jolla tarkoitetaan käytäntöä, jolla saadaan verkkosivuston ulkoasu vastaamaan (response) eri näyttöpäätteiden resoluutioihin eli ruutukokoihin. Onnistunut responsiivinen toteutus tarjoaa sivustolle käytettävyyttä ja luettavuutta laitteesta tai näyttöpäätteestä riippumatta. (Mozilla Developer Network 2022b.)

Käytettävyys on laadullinen tekijä, jota voidaan käytettävyystutkija Jakob Nielsenin (2012) mukaisesti määrittää viiden erilaisen kriteerin kautta eli opittavuuden, vaikuttavuuden, muistettavuuden, virheiden määrän ja tyytyväisyyden avulla. Taulukossa 1 on nähtävissä käytettävyyden viisi kriteeriä ja niihin liittyvät laadulliset kysymykset eriteltyinä.

Taulukko 1. Käytettävyyden viisi kriteeriä (Nielsen 2012)

Kriteeri	**Laadulliset kysymykset, joiden avulla tekijää mitataan**
Opittavuus	Kuinka helposti käyttäjät suoriutuvat perustehtävistä kohdatessaan palvelun ensimmäistä kertaa?
Vaikuttavuus	Opittuaan kuinka palvelu toimii, kuinka nopeasti käyttäjät voivat suoriutua tehtävistä?
Muistettavuus	Palatessaan palveluun, oltuaan hetken käyttämättä sitä, kuinka hyvin käyttäjät muistavat kuinka asiat toimivat?
Virheiden määrä	Kuinka monta virhettä käyttäjät tekevät, kuinka vakavia virheet ovat ja kuinka helposti niistä voi toipua?
Tyytyväisyys	Onko palvelua miellyttävä käyttää?

Vaikka käytettävyyttä pohditaan eniten suunnitteluvaiheessa, on myös toteutuksen parissa toimivien huomioitava ja tiedostettava käytettävyyteen liittyviä tekijöitä.

2.3 Verkkosivun rakenne ja osat

Sivujen rakenteen ja sisällön asettelulla ei ole virallista vakiintunutta standardia, mutta käytettävyys heikkenee sivuston toimintojen ollessa liian vaikeaselkoisia tai käyttäjälle vieraita. Useat sivustoilla esiintyvät osat ovat sen verran yleisesti käytettyjä, että niiden voidaan myös olettaa toimivan tutulla tavalla. Jos ne eivät toimi oletuksen mukaisesti, saattaa aiheutua hämmennystä tai turhautumista, joka puolestaan johtaa sivulta tai palvelusta poistumiseen. (Nielsen 2004.) Tämän vuoksi verkkosivustoa toteuttaessa on suotavaa noudattaa tuttuja käytäntöjä eri osien ja elementtien sijoittelussa. Taulukossa 2 on verkkosivustojen yleiset rakenteelliset osat eriteltyinä ja kuvattuina.

Taulukko 2. Verkkosivun yleiset rakenteelliset osat (Mozilla Developer Network 2022a)

Nimi	**Kuvaus**
Ylätunniste	Sivun yläosassa esiintyvä iso alue, jossa on yleensä iso otsikko, logo ja mahdollinen slogan.
Navigointipalkki (engl. navigation bar)	Sijaitsee yleensä sivun yläosassa, sisältää linkit sivuston pääsisältöihin. On usein rakennettu osaksi sivuston ylätunnistetta, mutta on saavutettavuuden kannalta parempi, jos navigointipalkilla on erillinen säiliö ja omat komponenttinsa.
Pääsisältö (engl. main content)	Alue sivun keskellä, jolla sivuston uniikki sisältö esitetään. Sisältö vaihtelee sivulta sivulle.
Sivupalkki (engl. sidebar)	Sivupalkeissa esitetään usein oheisinformaatiota sekä linkkejä, ja ne ovat myös usein navigaatioita.
Alatunniste (engl. footer)	Tila sivuston alaosassa, jossa yleensä esitetään tekijänoikeusilmoitukset tai kontaktitiedot. Sisältää usein ei-kriittistä ja toissijaista tietoa.

Sivustoilla esiintyvillä sisällön osilla on myös vakiintuneita nimiä. Taulukossa 3 on yleiset sisällön osat eriteltynä ja kuvattuna.

Taulukko 3. Verkkosivun yleiset sisällön osat (Kowalski 2021)

Nimi	**Kuvaus**
Hero-kuva (engl. hero image)	Ylätunnisteissa esiintyvä, joskus koko ruudun kokoinen, iso ja täysleveä kuva. Sen tarkoituksena on kiinnittää kävijän huomio visuaalisesti.
Banneri (engl. banner)	Bannerilla voidaan tarkoittaa joko hero-kuvan kaltaista huomiota herättävää visuaalista elementtiä tai mainosbanneria.
Karuselli (engl. carousel) tai liukusäädin (engl. slider)	Lista muuttuvia kortteja tai kuvia, jotka esitetään yksi kerrallaan.
Kortit (engl. cards)	Pieniä sisältölaatikoita, joita voidaan sommitella eri tavoin ja hyödyntää informaation esittämisessä.
Toimintakehotukset (engl. call to action)	Kehottavat käyttäjää tiettyyn toimintoon, kuten vaikkapa rekisteröitymään tai katsomaan uusimmat tarjoukset. Toimintakehoitukset voivat olla useanlaisia, esimerkiksi nappeja, jotka aktivoituessaan laukaisevat jonkin tapahtuman sivustolla.

Lisäksi sivustoilla esiintyy tietenkin myös tekstisisältöä, jota voidaan muotoilla visuaalisesti erilaisiin otsikoihin, leipäteksteihin, nostoihin ja muihin vastaaviin informaation esittämisen hierarkiaan liittyviin tekijöihin. (Kowalski 2021.)

Kuten verkkosivun rakenteen osat, myös sisällön osat toistuvat usein eri sivuilla. Tämän johdosta modulaarinen lähestymistapa on luonteva ratkaisu verkkosivujen toteuttamisessa.

2.4 Suunnittelujärjestelmät

Suunniteltavan verkkopalvelun tai sivuston takana on ammattimaisesti isossa työyhteisössä toimittaessa usein suunnittelujärjestelmä (engl. design system), jonka avulla muotoillaan suunniteltavan palvelun visuaalista sisältöä ja esityshierarkiaa eri näkymiin ennen varsinaista toteutusta. Suunnittelujärjestelmän tarkoituksena on toimia joukkona standardeja, jotka auttavat suunnittelun ja toteuttamisen hallintaa, vähentävät toistamisen tarvetta, luovat visuaalisen eheyden palvelun eri sivuille ja kanaville sekä tarjoavat yhtenäisen kielen projektissa työskentelevien välille. (Davis & Vesselov 2019, luvut 2–4.) Suunnittelujärjestelmä sisältää kokoelman asiakirjoja ja artikkeleita, esimerkkejä, kuvakaappauksia, suunnitteluohjeita, filosofioita ja muita digitaalisia resursseja, joiden avulla palvelun suunnittelua toteutetaan. Usein siihen sisältyy myös front-end-kehittäjille tarkoitettuja koodipätkiä ja ohjeita sekä modulaarinen kokoelma kaikista niistä elementeistä ja komponenteista, joita toteutettavien palvelujen käyttöliittymät voisivat sisältää. Tämä helpottaa suuresti kehittäjien työtä, sillä koodipohjat ovat koostetusti käytettävissä. Osana suunnittelujärjestelmää ovat myös kuviokirjastot, jotka sisältävät suunnittelumallijoukkoja sekä tyylioppaan, jonka avulla kuvataan suunnittelujärjestelmää ja sitä, miltä tuotteiden tulee näyttää ja vaikuttaa. Tyyliopas sisältää havainnollistuksia käyttöliittymän kuvioiden käyttöön sekä oikeat skaalat esimerkiksi kirjasimiin ja responsiivisuuteen. (Alli & DesignerUp 2020; Laubheimer 2016; UXPin n.d.)

Ulkoasun kannalta suunnittelujärjestelmän keskiössä ovat kriittiset komponentit eli ne komponentit, joita käytetään kaikissa tuotteissa tai sivuilla.

Verkkosivustoilla nämä kriittiset komponentit ovat rakenteissa toistuvasti esiintyviä osia ja elementtejä, esimerkiksi ylä- ja alaotsakkeita sekä navigaatioita. (Davis & Vesselov 2019, luku 1; Fessenden 2021.)

3 Front-end-kehitys

Seuraavissa alaluvuissa käydään läpi front-end-kehitystä sekä yleisimpiä, vakiintuneita front-end-teknologioita ja niihin liittyviä käytäntöjä. Teknologioita ja tapoja mahdollistaa verkkosivustoja on useita, joten luvussa käsitellään vain työn ja WordPressin kannalta oleellisia toimintaperiaatteita ja teknologioita. Luvun perimmäinen tarkoitus on pohjustaa lisäymmärrystä etenkin toiminnallisen osion toteuttamista varten.

3.1 Front-end, back-end ja full-stack

Front-endillä tarkoitetaan sitä verkkosivun näkyvää osaa, jonka kanssa käyttäjät suorittavat interaktioita. Kaikki sivustoilla selatessa näkyvät asiat selaimessa ovat front-endiä: muun muassa navigaatiot, fontit, värimaailmat ja sivun sisällön näkyvät osat. Näiden näkyvien elementtien toteutus on front-end-kehittäjien vastuulla. Oleellisimmat front-end-kehittäjältä vaadittavat osaamistaidot liittyvät HTML:ään, CSS:ään ja Javascript-ohjelmointiin. Lisäksi kehittäjän tulee olla tietoinen erilaisten ohjelmistokehysten ja ajoympäristöjen toiminnasta eli hänen tulee hallita erilaisia stackeja. Näiden työkalujen kanssa front-end-kehittäjät toteuttavat erilaisia suunnittelijoiden laatimia luonnoksia ja niiden sisältämiä komponentteja pyrkien tuottamaan eheitä palvelukokonaisuuksia. (Wales 2020.)

Verkkosivustot koostuvat front-endin lisäksi myös back-endistä, jolla tarkoitetaan verkkosivun näkymätöntä osaa eli palvelinpuolta. Back-end-kehittäjä rakentaa ja ylläpitää erilaisia teknisiä ratkaisuja, joilla palveluja ja komponentteja toteutetaan. On yleistä, että ennen pitkää kehittäjä päätyy täydentämään osaamistaan teknologioiden kehittyessä ja koulutusmahdollisuuksien avautuessa. Full-stack-kehittäjällä tarkoitetaan, että kehittäjä toimii sekä back-endin että front-endin parissa ja pystyy soveltamaan moninaisesti erilaisia työkaluja ja että

hän ymmärtää verkkopalvelujen eri kerroksia, joihin nimikkeessä esiintyvällä sanalla full-stack viitataan. (Wales 2020.)

3.2 Kehitysympäristöt

Kehitysympäristöt koostuvat erilaisista menettelyistä ja työkaluista, jotka liittyvät sovellus- ja ohjelmistokehitykseen sekä ohjelmistojen ja sovellusten testaamiseen ja virheiden korjaamiseen. Kehitysympäristöjä tarvitaan verkkosivujen kehityksessä, sivuston rakentamisessa ja prosessiin liittyvien teknologioiden hyödyntämisessä aina asentamisesta lähtien. Tarjolla olevia teknologioita ja olemassa olevia toimintatapoja on paljon, joten kehitysympäristöissä käytetyt työkalut, menetelmät ja teknologiat ovat sidoksissa työympäristön tapoihin. (Technopedia 2016; Arimetrics n.d.)

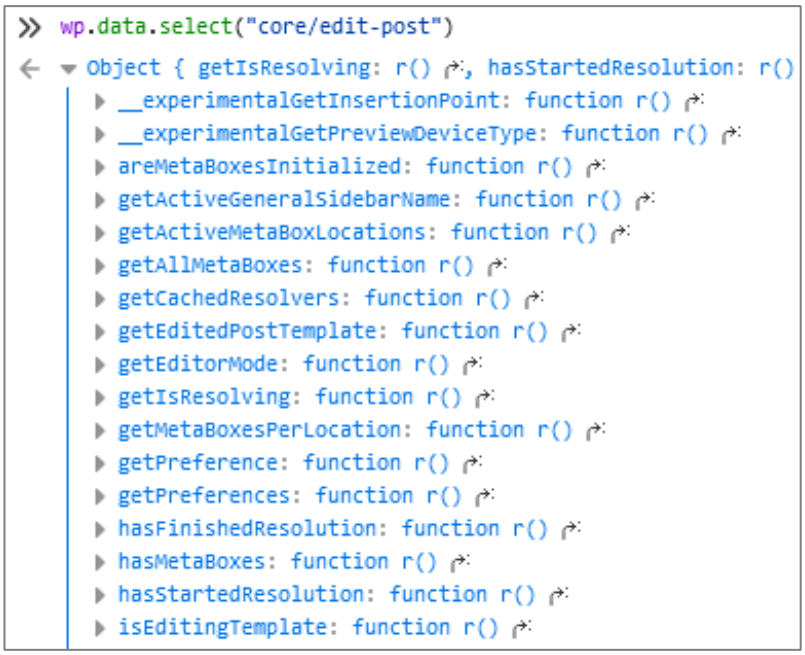
Kehitysympäristössä keskeisessä asemassa on koodieditori, jolla koodia, merkintäkieltä tai skriptiä toteutetaan ja muunnetaan luettavaksi tiedostoksi. Editoriohjelmat sisältävät usein ominaisuuksia tai lisäosia, joiden avulla koodin syöttämistä voidaan helpottaa. Koodieditoriohjelmia, kuten Visual Studio Code tai Atom, on ladattavissa ilmaiseksi internetistä. Tämän lisäksi useat modernit front-end-teknologiat ja sivustojen palvelinpuolen toimintoja hyödyntävät ohjelmistot ovat asennettavissa ja hallittavissa vain terminaali- tai komentotulkkityöskentelyn kautta. Eri käyttöjärjestelmille löytyy erilaisia terminaaleja sekä komentotulkkiohjelmia (engl. command-line interface), kuten Git Bash, Ubuntu tai PowerShell. Näiden ohjelmien kautta voidaan tekstipohjaisesti esimerkiksi suorittaa erilaisia komentoja, navigoida sisältöä ja muuttaa tiedostojen sekä kansioiden lukuoikeuksia. (McBain 2021.) Ohjelmistokehityksessä hyödynnetään laajalti erilaisista repositoryista eli tietovarastoista saatavia paketteja, joiden avulla ohjelmistoja kehitetään. Näissä paketeissa on esimerkiksi erilaisia uudelleenkäytettäviä komponentteja, sovellettavia funktioita ja kehitysvalmiita pohjia. (TechTerms 2011.)

Kehitysympäristöön kuuluu myös ohjelmistostack, joka sisältää Apache-verkkopalvelimen (A), MySQL:n (M) ja PHP:n (P) käyttömahdollisuudet. Apachen

avulla verkkosivu tuodaan näkyviin, MySQL tarjoaa tietokannan hallintajärjestelmän ja PHP:n avulla voidaan hyödyntää erilaisia PHP-toimintoja. (Young 2022; Rascia 2016.)

Modernien verkkoteknologioiden, kuten JavaScript-kirjastojen tai Sassin, käyttö on mahdollista vain kehitys- ja ajoympäristöjen kautta. Ajoympäristöllä tarkoitetaan sitä ympäristöä, jossa toteutettava ohjelma tai applikaatio kehitetään ja suoritetaan. Ympäristöihin tarvittavien ohjelmistojen asennukseen ja päivitykseen on olemassa paketinhallintatyökaluja (engl. package manager), joiden avulla voidaan käyttää verkkokehitykseen tarvittavia ohjelmistokehyksiä ja kirjastoja. (Technopedia 2020; Debian Manual n.d.)

Tämän lisäksi selaimet tarjoavat erilaisia työkaluja front-endin parissa työskenteleville. Inspector-työkalujen avulla voidaan tutkia näkyvän sivuston rakennetta ja kokeilla erilaisia CSS-tyylittelyjä live-esikatselun myötä. Näin tehdyt tyylittelyt eivät tietenkään ole pysyviä, mutta inspectorin avulla voidaan löytää säiliöt ja elementit, joita halutaan tyylitellä. Consolen avulla voidaan testata JavaScriptin toimintaa samalla tapaa kuin inspectorissa voidaan tyylitellä CSS:n avulla. Ennen kaikkea consolesta on hyötyä JavaScriptillä koodatessa, sillä sen avulla voidaan nähdä virheet koodissa ja pystytään etsiä globaaleja objekteja eli sellaisia objekteja, joiden ominaisuutena kaikki muuttujat ovat (Sallasmaa & Salmela n.d). WordPressin tapauksessa nämä objektit voivat olla esimerkiksi Reactilla toteutettujen lohkojen komponentteja ja niiden funktioita, joita pystytään hakemaan listattavaksi hyödyntämällä datamoduulia nimeltä select. Esimerkiksi Gutenbergin editorisivulla ollessa consolen komennolla wp.data.select("core/edit-post") voidaan listata kaikki valitun objektin mahdolliset funktiot. Kuvassa 4 on nähtävissä kuvakaappaus wp.data.selectista edit-postin ollessa valittuna.



Kuva 4. Kuvakaappaus Firefoxin consolen wp.data.select(“core/edit-post”) -komennosta.

Gutenberg hyödyntää JavaScript-kirjasto Reduxia datan tallentamisessa. Reduxin avulla ilmoitetaan, onko esimerkiksi sivupalkki valittu editorista näytettäväksi. (Alaa 2022.)

3.3 Front-end-teknologioita

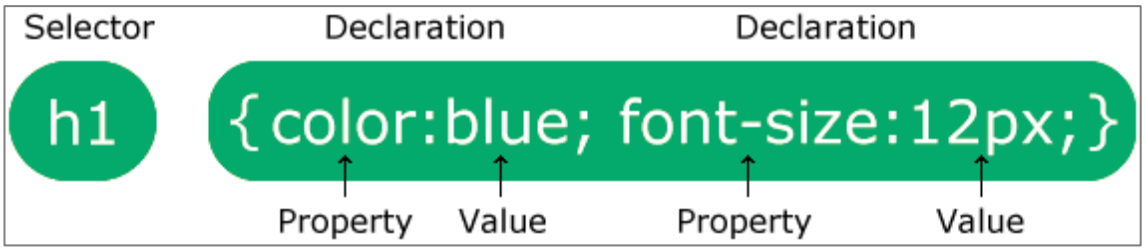
Internetin alkuaikoina pelkästään HyperText Markup Languagella (HTML) toteutetut verkkosivustot muistuttivat Wordin tyylittelemättömiä dokumentteja, ja alkupään verkkoteknologioiden tarjoamat mahdollisuudet sisällön asetteluun ja tyylittelyyn olivat hyvin rajoitettuja. Sanomalehdistä tuttuja modulaarisia ulkoasuun ja ruudukoihin liittyviä käytäntöjä on ollut mahdollista hyödyntää verkkosivustoilla vasta 1990-luvun loppupuolella. CSS:n ja HTML:n kehityksen myötä saapuneista mahdollisuuksista huolimatta sivustot koostuivat pitkään yksittäisistä sivuista, joiden rakentamisessa käytettyä koodia ei pystytty hyödyntämään tehokkaasti uudelleen, vaan osien ja elementtien koodit täytyi kerta kerran jälkeen kopioida ja liittää manuaalisesti sivuston jokaiselle eri sivulle. Sivustot eivät myöskään olleet responsiivisia ja saattoivat selaimesta tai näyttöpäätteestä riippuen näyttää aivan erilaiselta kuin alun perin oli tarkoitus. Vuonna 2008

julkaistu HTML5 piti sisällään modernisoivia uudistuksia, esimerkiksi mediatiedostojen esittämiseen tarkoitettuja elementtejä ja mahdollisuuden sivustojen rakentamiseen komponenttipohjaisesti, eli osa osalta. 2000-luvun jälkeen HTML:n ja CSS:n kehitys sekä suosioon noussut JavaScript alkoivat mahdollistamaan nykyisenkaltaisia interaktiivisia ja visuaalisia verkkosivustoja. Nykyään HTML, CSS ja JavaScript tarjoavatkin vakaan pohjan verkkopalveluille ja -sivustoille, useiden muiden ohella hyödynnettävien teknologioiden lisäksi. (Davis & Vesselov 2019, luku 1.)

3.3.1 HTML & CSS

HTML (engl. HyperText Markup Language) on merkintäkieli, jonka avulla voidaan rakentaa sivuja ja dokumentteja, hakea informaatiota, luoda kaavakkeita, suorittaa hakutoimintoja sekä käsitellä multimediaa. Kieleen sisältyy erilaisia elementtejä, joiden avulla sivun rakennetta määritellään, kuten esimerkiksi , <paragraph>, <table>, <list>, <aside>, <header> yms. HTML-elementit luokitellaan joko semanttisiksi tai ei-semanttisiksi. Ei-semanttiset elementit eivät juuri kerro sisällöstään rakenteellisesti. Semanttiset elementit taas puolestaan kertovat minkälaisesta sisällöstä on kyse, esimerkiksi lomakkeiden <form> ja artikkelin <article> viestivät selkeästi elementtien tarkoituksen. Semanttiset elementit ovat saavutettavuuden kannalta tärkeitä, ja niitä kannattaakin pyrkiä hyödyntämään mahdollisuuksien mukaan. (W3C 2016; W3Schools n.d. h.) Elementtejä voi myös nimetä, lähinnä tyylittelyn kohdistamista varten.

CSS (engl. Cascading Style Sheets) on tyylikieli, jonka avulla määritetään verkkopalvelun ulkoasuun liittyviä tekijöitä, kuten esimerkiksi värejä, esitysjärjestystä, fontteja ja animaatioita. CSS:n syntaksi koostuu valitsimesta (engl. selector), jonka avulla valitaan HTML-elementti tyyliteltäväksi, ja määrittelyistä (engl. declaration), joiden avulla esitetään haluttavat ominaisuudet (engl. properties) ja niiden arvot (engl. value). CSS:n syntaksi on havainnollistettu kuvassa 5. (W3Schools n.d. a.)

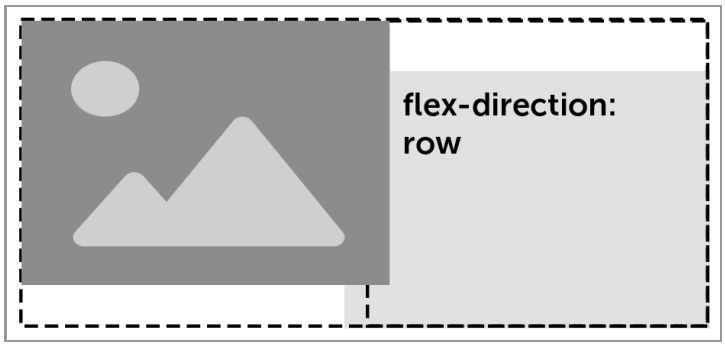


Kuva 5. CSS:n syntaksi. (Kuva: W3Schools n.d. a)

Valitsimia on kolmea tyyppiä eli tyypin valitsimet (ei merkkiä edessä), luokkavalitsimet (edessä piste) ja ID-valitsimet (edessä risuaita). Tyylittelyn kohdistamisessa CSS noudattaa hierarkiaa, jossa ID priorisoidaan yli saman elementin luokkien ja luokat puolestaan esitetään yli tyyppien. Kuitenkin, jos hierarkiaa hyödyntämällä tyylittelyt eivät näytä asettuvan, voidaan hyödyntää !important-komentoa, jolla pakotetaan haluttu tyyli tietylle elementille yli kaikkien muiden aiempien määrittelyiden. (Mozilla Developer 2022c; W3Schools n.d. i.)

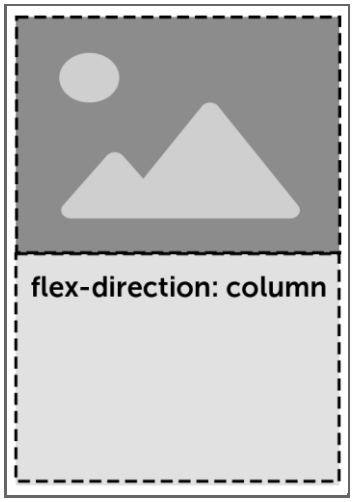
Mainitsemisen arvoisia CSS:n sisällön esittämiseen liittyviä moduuleja (eli lisäosia) ovat Flexbox Layout ja CSS Grid. Flexbox ja Grid ovat vakiintuneita verkkosivustojen toteuttamisessa sekä paljolti käytettyjä. Kummankin lisäosan kehityksen takana on ollut halu tarjota tehokkaita tapoja esittää ja järjestää sisältöä. Flexbox on kehitetty enemmän yksiulotteiseen ulkoasuun, eli se venyy joko riviin tai kolumniin, siinä missä Gridissä ruudukot puolestaan pystyvät hyödyntämään molempia. (Coyier 2013; Mozilla Developer Network 2022b.)

Esimerkiksi Flexboxin display: inline-flex-määrittelyn avulla saadaan sisällön esityssuunta asetettua yhteen riviin. Säiliön sisällön suuntaa on Flexboxin avulla helppoa muuttaa ominaisuuden flex-directionin arvolla row (kuva 6).



Kuva 6. Havainnollistus sisällön esityssuunnasta flex-direction: row -määrityksellä.

Vaihtoehtoisesti suunnan vaihto käy column-arvolla (kuva 7), jolloin sisältö saadaan helposti esitettyä kolumneissa (Coyier 2013).



Kuva 7. Havainnollistus sisällön esityssuunnasta flex-direction: column -määrityksellä.

Flexbox Layout ja CSS Grid tarjoavat tehokkuutta front-end-tyylittelyyn. Niistä on hyötyä etenkin sivustojen responsiivisuudessa, sillä niiden avulla voidaan helposti sovittaa sisältöä eri laitteille ja näyttöruuduille CSS:n media queryja hyödyntämällä (W3Schools n.d. b). Tyylimäärittelyjä on useampia ja tyylittelyjä voidaan luoda mm. erilaisten text-shadow-, box-shadow- ja transition -ominaisuuksien avulla. CSS:tä löytyy lisäksi useita muita interaktiivisuuteen ja sisällön asetteluun liittyviä keinoja, joita tässä työssä ei kuitenkaan käsitellä.

3.3.2 PHP

PHP (engl. Hypertext Preprocessor) on vuonna 1996 julkaistu ohjelmointikieli. W3Techin (2022) statistiikan mukaan PHP:tä käyttää 78,1 % sivustoista, joiden palvelinpuolen ohjelmointikieli tunnetaan. PHP:n suosio on kestänyt 26 vuotta, ja yksi iso syy suosioon on kielen jatkuva kehitys. Ohjelmointikieltä hyödyntää myös suosittu sisällönhallintajärjestelmä WordPress (MacManus 2021), jota käsitellään tarkemmin luvussa 4.

PHP on aloittelijaystävällinen ja monipuolinen ohjelmointikieli, joka painottuu palvelinpuolen skriptaamiseen. PHP:n avulla voidaan hyödyntää erilaisia muuttujia ja funktioita toimintojen toteuttamiseksi. Sitä käytetään yleensä sivustoilla HTML:ään upotettuna lisätoimintojen mahdollistamiseksi. PHP:ssä on sisällytettynä yli tuhat sisäänrakennettua funktiota, joita voidaan tarvittaessa luoda lisää. (PHP Manual n.d.; W3Schools n.d. e.) Se mahdollistaa myös ehtolausekkeita ja silmukoita, jotka toistavat tiettyjä toimintoja erilaisin annetuin ehdoin. Yksinkertaisesti sanottuna silmukat tulostavat esimerkiksi tietyn luvun, arvon tai syötteen. PHP:n avulla on mahdollista hyödyntää ja manipuloida erilaisia tietokantoja. Tietokantoihin voidaan tallentaa muuttujia, arvoja, taulukoita ja istuntoihin liittyviä tietoja. Nykyaikaisiin ohjelmointikieliin nähden PHP saatetaan kokea vanhanaikaisena ja työläänä, mutta sen kerrotaan myös kykenevän kaikkeen siihen, mihin muillakin ohjelmointikielillä kykenee. (W3Schools n.d. f; W3Schools n.d. d; Laaksonen 2011.)

3.3.3 Sass

Sass eli Syntactically Awesome Style Sheets on CSS:n esikääntäjä, joka sisältää CSS:stä toistaiseksi puuttuvia ominaisuuksia. Nämä ominaisuudet ovat mahdollisuuksia sisäkkäistää (engl. nesting) elementtejä, luoda erilaisia sekoituksia (engl. mixin) ja jatkumoita (engl. extend) sekä hyödyntää ohjelmoinnille tuttuja muuttujia (engl. variables). Koska Sass on esikääntäjä, täytyy sillä muodostettu tiedosto kääntää tavalliseksi CSS-tiedostoksi. Käännöstä varten tulee käyttää terminaalia ja tiettyjä ennalta määriteltyjä komentoja (esimerkiksi kuva 8), joilla muutetut tiedostot käännetään (engl. compile) selaimelle luettavaksi yksittäiseksi CSS-tiedostoksi joka tallennuksen jälkeen.

```
sass –watch sass/style.scss:style.css
```

Kuva 8. Sass watchin avulla tapahtuva tarkkailu kääntämiseksi.

Sassin ensisijainen toteutus kulkee nimellä Dart Sass (tiedostomuoto .sass). Se ei tue tavallista CSS-syntaksia, toisin kuin Sassy CSS (tiedostomuoto .scss). SCSS:n syntaksit ovat CSS:n tutustuneille lähestyttävämpiä kuin Dart Sassin,

jossa aaltosulkeita ja puolipisteitä ei käytetä. Syntaksien ero on nähtävissä kuvassa 9. (Sass n.d. a; Sass n.d. b; Sass n.d. c.)

Kuva 9. SCSS:n (ylhäällä) ja Sassin (alhaalla) syntaksin merkintätavan eroja. (Kuva: Sass n.d. b)

Sass tarjoaa tehokkaan tavan jakaa koodia pienempiin osiin eli partiaaleihin. Partiaalien edessä on yleensä alaviiva ja usein ne sijoitetaan omaan kansioonsa (_partials). Tämän toimintaperiaateen voidaan katsoa olevan modulaarinen, koska sisältö pilkotaan uudelleenkäytettäviin osiin. Sassiin saatavilla olevia sisäänrakennettuja lisäosia kutsutaan myös moduuleiksi. Jotta päätiedosto osaisi hyödyntää partiaaleja, tulee ne tuoda päätiedostoon. (Leverenz 2014.) Noden Sassia käytettäessä tuominen tapahtuu @import -tuontikomennolla (oikea puoli kuvasta 10).

Kuva 10. Vasemmalla puolella kuvakaappaus Sassin partiaalikansion tiedostoista. Oikealla puolella kuvakaappaus Node Sassin @import -tuontikomennosta main.scss-tiedostossa.

Nestingin ja muuttujien hyödyntäminen sekä erilaiset mixins- ja extend-toiminnot vähentävät tarvetta toistaa koodia ja helpottavat sen uudelleenkäyttöä (kuva 11).

```
.footer {
    grid-column: 1 / -1;
    grid-row: 4 / 5;
    @include flexPosition(space-around);
    padding: 5rem 0;
    border-top: .1rem solid color(primary);


    @include response(lg) {
        flex-wrap: wrap;
    }
}
```

Kuva 11. footer-elementissä mixinsin hyödyntäminen (@includet flexPosition ja response)

Mixinien avulla voidaan luoda tyylittelyn määrittelyille pohjia, joita tehokkaasti hyödynnetään yhdessä erilaisten vakioiden kanssa. Nämä vakiot voivat olla esimerkiksi sisällön kokosuhteita, kuten esimerkiksi kuvan 12 response(lg) on.

```
@mixin response($breakpoint) {
    @if($breakpoint == xl) {
        @media(max-width: 1200px) {
            @content;
        }
    }

    @if($breakpoint == lg) {
        @media(max-width: 1000px) {
            @content;
        }
    }
}
```

Kuva 12. Mixins-tiedostoon määritelty resoluutiopiste

Esitetyn mixinin avulla hyödynnetään muutujia responsiivisuuden säätelemisessä. Koodia voidaan käyttää uudelleen nopeasti useissa eri elementeissä ja komponenteissa.

3.3.4 JavaScript

JavaScript on kehitetty verkon elävöittämiseen. Se on skriptauskieli, joka integroituu tehokkaasti HTML:ään ja CSS:ään. Sen avulla toteutaan erilaisia toimintoja verkkosivustoilla. Perinteisellä JavaScriptillä pystyy esimerkiksi muokata olemassa olevia sivuston elementtejä ja sisältöä, tarjota interaktiivisuutta käyttäjän toimintoihin, lähettää palvelinpyyntöjä, asettaa ja hakea evästeitä sekä näyttää erilaisia viestejä. (Javascript.info 2021.) JavaScriptiin on saatavilla useita erilaisia kirjastoja ja ohjelmistokehyksiä, joiden avulla nopeutetaan skriptausta sekä toteutusta.

JavaScript-kirjastot sisältävät koodijoukkoja (liittyen esimerkiksi toiminnallisiin komponentteihin ja osiin), joita voidaan hyödyntää useissa eri projekteissa. Ohjelmistokehykset ovat laajempia kokonaisuuksia, joiden hyödyntämiseksi tulee osata käyttää niihin liittyviä strukturaalisia koodiin liittyviä sääntöjä. (Code With Random 2021.) Stack Overflowin vuonna 2021 teettämän kyselyn mukaan yleisimpiä käytettyjä JavaScript-kirjastoja ja -kehyksiä olivat jQuery, React, Node.js:ää hyödyntävä Express, Angular ja Vue. Kokonaisuudessaan kyselyyn vastasi yli 80 000 kehittäjää.

Node.js on JavaScriptin ajamiseen tarkoitettu ympäristö, joka tarjoaa erilaisille JavaScript-kirjastoille ja ohjelmistokehyksille sopivan perustan (Node.js n.d).

Kirjastot ja ohjelmistokehykset tarjoavat modulaarisuutta uudelleenkäytettävien komponenttien ja niiden mukautettavuuden myötä. Kirjastojen ja ohjelmistokehysten käyttö helpottaa modernien verkkopalvelujen toteuttamista, sillä ladattavien riippuvuuksien myötä käytettäviin saa useita erilaisia valmiiksi rakennettuja komponentteja, joita käyttämällä toteutuskin käy nopeammin.

3.3.5 API & JSON

API:tä (Application Programming Interface) eli ohjelmointirajapintaa käytetään laajentamaan selaimen toimintoja. API tarjoaa helppokäyttöisen syntaksin, jota voidaan hyödyntää myös kompleksisille toiminnoille. Rajapintoja löytyy

sisäänrakennettuina kaikista moderneista verkkoselaimista ja ne ovat oleellinen osa selainten toimintojen mahdollistamista. (W3Schools n.d. g.)

JSON (engl. JavaScript Object Notation) on kevyt datansiirtoon ja -tallennukseen tarkoitettu tiedostomuoto, jota voidaan käyttää verkkosivulta tietoja palvelinpuolelle lähettäessä. Oleellista teemoja tai lisäosia kehittäessä on tietää, kuinka muokata JSON-tiedostoja, joiden tehtävänä on tallentaa tärkeää metadataa projektista tai määritellä sen toiminnallisia ominaisuuksia. JSON-tiedostojen editointi on tärkeässä roolissa WordPressin teemojen ja lisäosien jatkokehityksessä ja toiminnallisuuksien mahdollistamisessa, sillä niihin tallennetaan metadataa, ominaisuuksia ja määritteitä esimerkiksi riippuvuuksille eri pakettien välillä. (W3Schools n.d. c; heynode n.d; nodejs 2011.)

4 WordPress

WordPress on maailman suosituin sisällönhallintajärjestelmä, jota hyödynnetään useilla eri tarkoitukseen tehdyillä verkkosivustoilla. WordPressin suosion takana on palvelun mukautettavuus ja mahdollisuus hyödyntää erilaisia JavaScript-kirjastoja ja ohjelmistokehyksiä sivustojen toteuttamisessa. Alla olevissa luvuissa käydään läpi yleisiä ja oleellisia asioita liittyen WordPressiin, sen taustoihin ja toimintaperiaatteisiin.

4.1 WordPressin ominaisuuksia

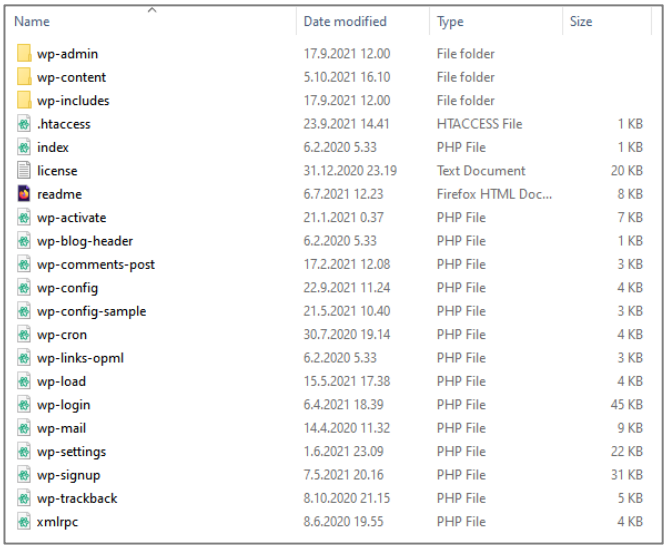
WordPress on PHP:tä ja MySQL-tietokantaa hyödyntävä avoimen lähdekoodin sisällönhallintajärjestelmä, jota W3Techsin statistiikan (2022) mukaan käyttää 43,3 % kaikista tunnetuista verkon verkkosivustoista. Tunnetuista sisällönhallintajärjestelmää hyödyntävistä sivustoista WordPressiä käyttää 65,3 %. Sisällönhallintajärjestelmät ovat nimensä mukaisesti sisällön hallintaan tarkoitettuja järjestelmiä, jotka tarjoavat esimerkiksi mahdollisuuden luoda, editoida, julkaista ja arkistoida sivustoja, artikkeleita ja blogimerkintöjä sekä lisätä tapahtumia, tuotteita ja tuotekuvauksia tai katsella sivuston statistiikkaa. Siitä on olemassa kaksi versiota: wordpress.com ja erilliselle palvelimelle ladattava wordpress.org.

Tässä työssä käsitellään palvelimelle asennettavaa WordPressiä. (W3Techs 2022; Kohan 2010; Wordpress.org n.d. a.)

WordPressin suosio perustuu sen ilmaisuuteen, mukautettavuuteen, hakukoneystävällisyyteen, pysyvyyteen ja tietoturvallisuuteen. WordPressin käyttöön ja koodien soveltamiseen on löydettävissä paljon opastavia materiaaleja, joiden avulla myös aloittelijoiden on mahdollista oppia ymmärtämään WordPress-sivustojen toimintatapoja ja -periaatteita. WordPressin avoin lähdekoodi, jatkuva kehitys ja taustalla oleva laaja joukko kehittäjiä, uusien teknologioiden hyödyntäminen sekä helppokäyttöisyys tekevät siitä helposti lähestyttävän myös aloitteleville kehittäjille. Se tarjoaa ratkaisuja ja mahdollisuuksia verkkosivustojen toteutukseen, eikä saatavilla toistaiseksi ole yhtä monipuolista sisällönhallintajärjestelmää. (WPBeginner 2022.)

4.1.1 WordPressin koostavat tiedostot

WordPress koostuu sadoista tiedostoista ja kymmenistä hakemistoista, joita palvelu hyödyntää sivustojen esittämiseksi. Kuvakaappaus asennuskansion juuresta on nähtävissä kuvassa 13.



Name	Date modified	Type	Size
wp-admin	17.9.2021 12.00	File folder	
wp-content	5.10.2021 16.10	File folder	
wp-includes	17.9.2021 12.00	File folder	
.htaccess	23.9.2021 14.41	HTACCESS File	1 KB
index	6.2.2020 5.33	PHP File	1 KB
license	31.12.2020 23.19	Text Document	20 KB
readme	6.7.2021 12.23	Firefox HTML Doc...	8 KB
wp-activate	21.1.2021 0.37	PHP File	7 KB
wp-blog-header	6.2.2020 5.33	PHP File	1 KB
wp-comments-post	17.2.2021 12.08	PHP File	3 KB
wp-config	22.9.2021 11.24	PHP File	4 KB
wp-config-sample	21.5.2021 10.40	PHP File	3 KB
wp-cron	30.7.2020 19.14	PHP File	4 KB
wp-links-opml	6.2.2020 5.33	PHP File	3 KB
wp-load	15.5.2021 17.38	PHP File	4 KB
wp-login	6.4.2021 18.39	PHP File	45 KB
wp-mail	14.4.2020 11.32	PHP File	9 KB
wp-settings	1.6.2021 23.09	PHP File	22 KB
wp-signup	7.5.2021 20.16	PHP File	31 KB
wp-trackback	8.10.2020 21.15	PHP File	5 KB
xmlrpc	8.6.2020 19.55	PHP File	4 KB

Kuva 13. Kuvakaappaus WordPressin asennuksen juurikansiosta.

WordPressin tiedostot voidaan luokitella ydin- ja sisältötiedostoihin. Ydintiedostojen (engl. core) muokkaaminen vaikuttaa palvelinpuolen käyttöliittymään ja toimintoihin. Ydintiedostoja käytetään määrittämään mm. hallinnointipaneelin (engl. admin panel) aluetta, jossa sisältöä luodaan sekä lisäosien ja ulkoasun asetuksia muokataan. Lisäksi niissä määritellään WordPressin tärkeitä toiminnallisuuksia. WordPressin ydintiedostoja ovat kaikki muut tiedostot paitsi teema- ja lisäosatiedostot, jotka puolestaan sijaitsevat kansiossa wp-content. (Hughes 2022; WPBeginner n.d. a.)

4.1.2 WordPressin kehittäjäystävällisyys

WordPressin REST (engl. Representational State Transfer) API mahdollistaa lähettämällä ja vastaanottamalla dataa JSON-muodossa esimerkiksi Reactin hyödyntämisen WordPressissä. RESTin avulla voidaan kehittää hyvinkin joustavia verkkosivustoja, joissa teeman sijaan sivustolla toimii sivuston tarpeisiin rakennettu verkkosovellus, joka hyödyntämällä WordPressin API:n tarjoamaa dataa mahdollistaa voimakkaita käyttöliittymiä sisällön hallinnointiin. Tämän kaltaista konseptia kutsutaan Headless-toteutukseksi (engl. Headless Content Management System). Headless-toteutuksessa WordPress on olemassa vain sisällön lisäämistä, editoimista ja tallentamista varten. Käytäntö on yleistynyt alan ammattilaisten parissa, sillä se tarjoaa saavutettavuutta, turvallisuutta, skaalautuvuutta, turvallisuutta tiedonsiirtoon sekä kevyitä sivustoja ja koska sen avulla sivustoja pystytään toteuttamaan ilman turhia toiminnallisuuksia. Headless-toteutuksessa WordPressin ytimen tarjoamat mahdollisuudet muokata teemaa mukauttimen avulla eivät ole käytettävissä. Se kuinka paljon Headless-toteutus mahdollistaa mukautettavuutta, riippuu täysin toteutuksen taustalla olevasta kehitystyöstä (WordPress.org n.d. h; Myers 2022.)

WordPress tarjoaa kehittäjille työkaluja ja paketteja lisäosien kehitykseen sekä kattavia ohjeistuksia. Theme Developer Handbookin lisäksi WordPressin sivuilta löytää myös Plugin Developer Handbookin, joka keskittyy lisäosien kehitykseen, sekä Block Developer Handbookin, joka tarjoaa informaatiota Gutenbergin

lohkojen kehityksestä. Handbookit ovat aloittelijaystävällisiä ja niiden sisältämistä aiheista löytää myös tarvittaessa lisää informaatiota muualta internetistä.

4.1.3 Mukautetut kentät ja taksonomiat

Työssä ei esitellä mukautettujen kenttien tai taksonomioiden toimintaa, mutta ne mainitaan, sillä ne ovat isossa osassa WordPress-sivustojen sisällön esittämisen mahdollistamista.

Mukautettujen kenttien (engl. custom fields) avulla voidaan tallentaa, lisätä ja esittää lisäinformaatiota sisällöstä. Tarkemmin sanottuna mukautettuihin kenttiin tallennetaan metadataa eli informaatiota informaatiosta, jota voidaan kutsua esitettäväksi esimerkiksi blogipostauksessa “tunnelma”-lisäkenttänä. Mukautettuihin kenttiin löytyy suosittuja lisäosia, joiden avulla kenttien hallinnointia ja toimintoja saadaan helpommin lähestyttäviksi loppukäyttäjälle. (Duò 2022; WordPress n.d. p.)

Taksonomioiden, kuten kategorioiden ja tagien, avulla voidaan jaotella blogipostauksia. Niitä hyödynnetään sisällön esittämisessä kutsumalla esimerkiksi johonkin korttiin tietyn kategorian artikkelin sisältöä näkyviin. (WordPress.org n.d. q.)

4.2 Teemat

WordPressin tärkeimpiä ominaisuuksia on teemat, jotka hyödyntävät erilaisia template-tiedostoja sisällön esittämisessä. Teemoja on olemassa klassisina, pohjautuen PHP:hen, sekä lohkoteemoina, jotka pohjautuvat HTML:ään ja Gutenbergiin. Näissä teemoissa lohkoja voidaan helposti sijoittaa mihin tahansa niille varattuun paikkaan, myös ylä- ja alaotsakkeisiin. Lohkoteemat tukevat moduuleja, eli ne mahdollistavat modulaarista lähtötapaa. Lohkoteemat eivät kuitenkaan välttämättä tarjoa esimerkiksi mukautinta, vaan sisältöä säädetään Gutenbergin tai teeman kautta mahdollistettujen, saatavilla olevien tyylittelyjen avulla sekä itse rakennettujen käyttöliittymien avulla. Teemojen rakentamiseen

ja mukauttamiseen löytyy kattavasti aloittelijaystävällisiä ohjeita WordPressin omilta sivuilta ja muualta verkosta. (WordPress.org n.d. s; WPBeginner 2022; Wordpress.org n.d. a.)

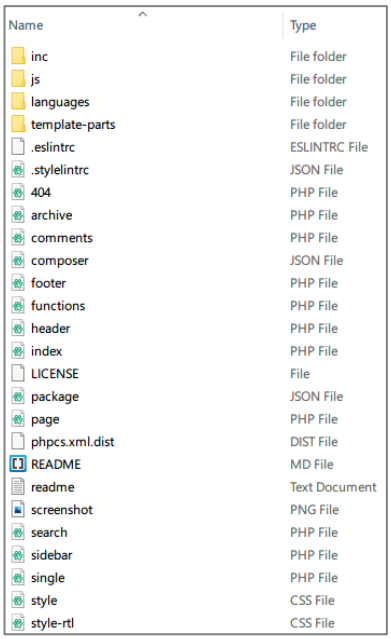
Teemat toimivat koostamalla näkyviin WordPressin tietokantaan tallennettua tietoa. Teeman avulla sisältöä voidaan esittää ennalta määritellyissä kohdissa, teeman ulkoasua säätää ja sisällön esitystapaa voidaan hallinnoida. Teeman vaihtaminen muuttaa sivuston ulkoasua. Sivun selaimeen koostamista varten teemojen täytyy yksinkertaisimmillaan sisältää vain index.php- ja style.css -tiedostot, joissa tulee olla WordPressin hyödyntämät PHP-koodit sisällön kutsumiseksi. Lohkoteemojen tulee sisältää myös index.html-tiedosto. Teemoja on saatavilla ilmaisina, maksullisina ja niitä voidaan myös tehdä itse. Lisäksi tarjolla on aloittelijateemoja (engl. starter themes), jotka usein sisältävät kaikki oleelliset perustoiminnallisuudet sivustolle ja jättävät tarkemman tyylittelyn ja teemankehityksen kehittäjälle. (Wordpress.org n.d. b; WordPress.org n.d. s; Wordpress.org n.d. c.) Omien teemojen toteuttaminen alusta loppuun vaatii syvällistä perehtymistä ja ymmärrystä WordPressin taustalla olevista teknologioista ja toimintatavoista.

4.2.1 Sivut ja artikkelit

WordPressilla toteutetut sisältösivut voivat pohjautua joko sivuihin (engl. page) tai artikkeleihin (engl. posts). Sivut ovat staattisia ja ovat yleensä listattuina navigaatioon. Artikkelit ovat sivustolla kronologisesti käänteisesti luokiteltua sisältöä, joiden avulla kävijöille voidaan tarjota lisäinformaatiota. (WordPress.com n.d. b.)

4.2.2 Teeman template-tiedostot

Teemoissa on useita tiedostoja, jotka mahdollistavat erilaisia interaktioita ja sisältöä sivustoilla (Wordpress.org n.d. c). Kuvassa 14 on Underscores-teeman kansio, jossa template-tiedostot sijaitsevat.



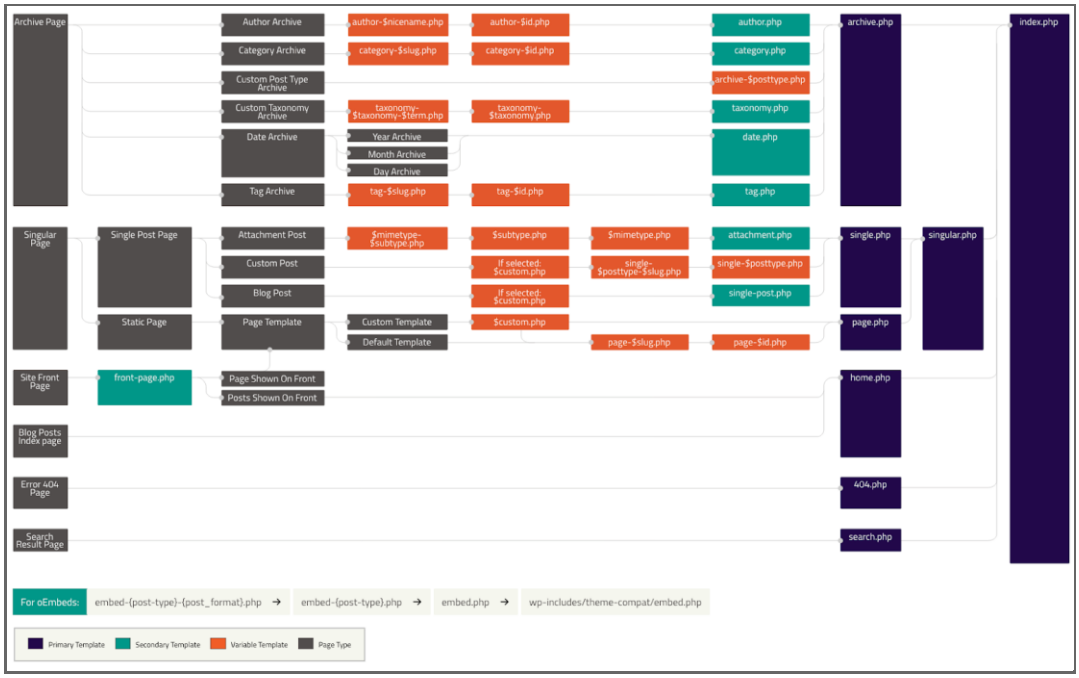
Kuva 14. Kuvakaappaus Underscores-teeman kansiosta.

Template-tiedostoja voidaan muokata ja rakentaa, kopioida ja soveltaa WordPressin toiminnallisuuksien kanssa. Ne ovat myös modulaarisia, eli niitä pystytään hyödyntämään uudelleen ja muokata tiettyjen WordPressiin asetettujen sääntöjen mukaisesti. (WordPress.org n.d. d; Wordpress.org n.d. f.) Template-tiedostoja on sekä sivujen koostamiseen että sivuilla esiintyvien komponenttien koostamiseen. Sivutemplateja (engl. page templates) ovat mm. index.php, page.php ja blog.php. Partiaaleja (engl. template partials) ovat puolestaan esimerkiksi ala- ja yläotsakkeiden footer.php ja header.php. Templaten partiaaleja voi templaten tagien avulla hyödyntää useissa eri sivupohjissa. (WordPress.org n.d. g.) Kuvassa 15 on nähtävissä hyvin yksinkertaisen WordPress-teeman index.php-tiedosto.

```php
<?php get_header(); // Kutsuu headerin ?>
<div id="ttr_main" class="row">
<div id="ttr_content" class="col-lg-8 col-sm-8 col-md-8 col-xs-12">

<div class="row">

<?php if (have_posts()) : while (have_posts()) : the_post(); //  Näyttää postaukset sivustolla ?>
<div class="col-lg-6 col-sm-6 col-md-6 col-xs-12">
<h4 class="postedon">Postattu <?php the_time('F jS, Y') // Hakee päivämäärän postauksesta ?></h4>
<p><?php the_content(__('(Lisää...)')); // Näyttää sisällön ?></p>
</div>
<?php endwhile; else: // Jos postauksia ei ole, kertoo sen. ?>
<p><?php _e('Ei postauksia.'); ?></p>
<?php endif; ?>
</div>
</div>
<?php get_sidebar(); // Hakee sidebarin, joka ei näy kaikilla sivustoilla ?>
</div>

<?php get_footer(); ?>
```

Kuva 15. Yksinkertainen index.php-tiedosto tekstihavainnollistuksien kera.

Mm. get_header();, get_sidebar(); ja the_content(...) -templatetageilla koostetaan eri partiaaleja ja niiden sisältämää informaatiota esitettäväksi sivustolle. Sivuiksi koostetut templatet noudattavat koostamisessa WordPressiin ennalta määrättyä hierarkiaa, joka havainnollistuu kuvassa 16. (WordPress.org n.d. e.)



Kuva 16. WordPressin templatehierarkia visualisoituna. (Kuva: Wordpress.org n.d. e.)

Template-tiedostojen hierarkiaa katsoessa voidaan huomata sivutemplatejen rakentuvan useista osista ja partiaalitemplateista. Jos jotain hierarkkisessa vuorossa esiintyvää tiedostoa ei löydy teeman kansiosta, siirtyy WordPress hierarkkisesti seuraavaan saatavilla olevaan tiedostoon. Hierarkian mukaisesti ainoa tarvittava tiedosto sisällön esittämiseksi on viimeisimpänä oikealla näkyvä index.php. (Wordpress.org n.d. f.)

4.2.3 Koukut, toiminnot ja filtterit

WordPressin koukut (engl. hooks) mahdollistavat koodipätkien avulla toimintoja tietyissä ennalta määrätyissä kohdissa. Ne luovat perustan lisäosien ja teemojen yhteistoimintaan WordPressin ydintoimintojen kanssa. Myös ydintoiminnot hyödyntävät niitä laajalti. Koukkuja voidaan luokitella sekä toimintoihin (engl. actions) että suodattimiin (engl. filters). Toiminnoilla lisätään dataa tai muutetaan WordPressin toimintatapoja, ja niitä voidaan määrätä suoritettavaksi tietyissä ytimen, lisäosan tai teeman kohdissa. Ne eivät palauta dataa takaisin callback-funktiolla, vaan suorittavat vain toimintoja. Suodattimien avulla voidaan muokata dataa ytimen, lisäosien ja teemojen suorittamisen aikana. Suodattimet odottavat saavansa dataa takaisin. Ne toimivat eristyksissä, eikä niillä saisi olla sivuvaikutuksia esimerkiksi globaaleihin muuttujiin. (McCollin 2016; WordPress.org n.d. i.)

Aiemmassa alaluvun alaluvussa esiintynessä kuvan 15 ensimmäisellä rivillä oleva PHP-koodipätkä <?php get_header(); ?> suorittaa toimintokoukun, joka on on ennalta määrätty hakemaan sivuston yläotsakkeen PHP-tiedoston teeman kansiosta. Jotta muut kuin ennalta määrätyt toiminnot ja suodattimet toimisivat, vaaditaan teeman kansiosta löytyvään functions.php-tiedostoon määrittely ja callback-funktioille koukut, joiden avulla kerrotaan, miten toiminnon tai suodattimen tulisi toimia. (WordPress.org n.d. j.)

4.2.4 Funktiot ja functions.php

WordPressissä on useita ennalta määrättyjä ja käytännöllisiä PHP-funktioita. Toiminnot ja suodattimet vaativat määritellyt callback-funktiot, joita luodaan

PHP:llä teeman kansiosta löytyvään functions.php-tiedostoon. Tiedoston avulla voidaan vaikuttaa useisiin teemassa käytettäviin toimintoihin, koukuttaa toimintoja ja suodattimia, rekisteröidä dataa sekä määritellä, miten dataa käytetään. Näin voidaan rekisteröidä hyödynnettäväksi lohkoeditoriin esimerkiksi väripaletteja, joiden värit määritetään taulukkoina (engl. array) functions.php-tiedostoon. (WordPress.org n.d. j; WordPress.org n.d. k; WordPress.org n.d. l.)

```
function wptoiminto_callback() {
	// määritteet toiminnalle
}
add_action( 'init', 'wptoiminto_callback' );
```

Yllä olevassa esimerkissä ensimmäisellä rivillä esiintyvä funktio wptoiminto_callback() määrittää toiminnan (joka voisi sisältää myös esimerkiksi suodattimia) ja viimeisellä rivillä oleva add_action(‘init’, ‘wptoiminto_callback’) koukuttaa funktion ‘init’-koukulla suoritettavaksi sivun latautumisen jälkeen, mutta ennen kuin yläotsake lähetetään selaimelle. (WordPress.org n.d. m; Wordpress.org n.d. n.)

Kuvasta 17 voidaan huomata, että functions.php-tiedosto voi sisältää erilaisia PHP:n avulla luotuja ehtolausekkeita (if), taulukoita (array (‘’)), teeman toiminnallisuuksien lisäämistä (add_theme_support {...}) ja navigaatioiden rekisteröintiin liittyviä määritteitä (register_nav_menus()). Esimerkiksi uusien navigaatioiden rekisteröintiin ei tarvitse määrittää kuin muutamia rivejä koodia, sillä tarkemmat määritteet löytyvät WordPressin ydintiedostoista (WordPress.org n.d. o). Opinnäytetyössä käytetyn functions.php-tiedoston löytää myös liitteestä 7.

functions.php

```php
<?php
/**
 * oppari functions and definitions
 *
 * @link https://developer.wordpress.org/themes/basics/theme-functions/
 *
 * @package oppari
 */

if ( ! defined( '_S_VERSION' ) ) {
	define( '_S_VERSION', '1.0.0' );
}

if ( ! function_exists( 'oppari_setup' ) ) :
	function oppari_setup() {
		load_theme_textdomain( 'oppari', get_template_directory() . '/languages' );

		add_theme_support( 'automatic-feed-links' );
		add_theme_support( 'title-tag' );
		add_theme_support( 'responsive-embeds' );
		add_theme_support( 'align-wide' );
		add_theme_support( 'editor-styles' );
		add_editor_style( 'style-editor.css' );
		add_theme_support( 'post-thumbnails' );

		register_nav_menus(
			array(
				'menu-1' => esc_html__( 'Primary', 'oppari' ),
			)
		);

		add_theme_support(
			'html5',
			array(
				'search-form',
				'comment-form',
				'comment-list',
				'gallery',
				'caption',
				'style',
				'script',
			)
		);

		function oppari_load_dashicons() {
			wp_enqueue_style( 'dashicons' );
		}
```

Kuva 17. Kuvakaappaus Underscoresilla muodostetun opinnäytetyön teeman functions.php:stä

Mukautettua koodia lisäämällä voidaan esimerkiksi luoda postaukseen tyyppejä, taksonomioita tai shortcodeja. (WPBeginner n.d. b.)

4.2.5 Mukautin (engl. customizer)

Mukautin on WordPressin teemojen säätöön tarkoitettu käyttöliittymä, jolla sivuston näkyvyyttä ja ulkoasua voidaan määrittää. Sen avulla vaikutetaan sivuston identiteettiin eli nimeen, logoon ja ikoniin, ulkoasuun eli väreihin, kirjasimiin ja otsakekuviin sekä esimerkiksi navigaation sisältöön. Mukauttimeen sisältyy mahdollisuus live-esikatselun ja CSS-editorin hyödyntämiseen. Mukauttimen avulla voidaan valita myös sivuston esitystapa, eli onko etusivu staattinen vai artikkeleita näyttävä sivu. Mukauttimen mahdollistamat toiminnot riippuvat pitkälti teemasta ja sen taustalla olevasta kehityksestä: lisätoiminnallisuuksien

tulee olla PHP:n (ja mahdollisesti myös JavaScriptin) avulla asianmukaisesti määriteltyinä teemaan. (Wordpress.com n.d. a.)

4.3 Lisäosat

WordPressiin on saatavilla tuhansittain erilaisia toteutusta helpottavia lisäosia ja vimpaimia, joiden avulla sivuston sisältöä voidaan luoda ja hallita sekä sivuston toiminnallisuuksia lisätä. Lisäosien avulla voidaan mahdollistaa lisäominaisuuksia teemaan, sivustolle tai toisiin lisäosiin. Lisäosien tarjoamat toiminnallisuudet eivät ole sidoksissa teemoihin, vaan ne säilyvät myös teemaa vaihtaessa. Esimerkiksi sivurakentajat, verkkokaupat, sosiaalisen median feedit ja hakukoneoptimointiin tarkoitetut työkalut ovat yleisesti käytettyjä lisäosia. Lisäosien mahdollistama vaikutus sivustolla voi vaihdella pienestä suureen, riippuen siitä mihin lisäosa on tarkoitettu ja mihin se joustaa. (Kinsta 2021.)

4.3.1 Sivueditorit ja rakentajat

Sivueditorilla tarkoitetaan sivun sisällön editoimisen mahdollistavaa työkalua, jossa ei ole vedä ja pudota -periaatteella toimivaa mahdollisuutta sivun rakentamiseksi. Sivurakentajat ovat puolestaan WordPressiin asennettavia lisäosia, joiden avulla voidaan rakentaa sekä muokata sivuston sisältöä ja ulkoasua. Ne tarjoavat sivueditoreja kattavampia mahdollisuuksia sivuston ulkoasun ja rakenteen luomiseen sekä sisällön tyylittelyyn, usein mahdollistamalla rakentamisen vedä ja pudota -tyylisesti eli ilman koodaamista. (WPBeginner n.d. c.)

4.3.2 Gutenberg ja lohkojen kehitys

WordPressin lohkoihin (engl. blocks) perustuva sivueditori Gutenberg omaa paljon potentiaalia kehittäjien kannalta. Gutenbergilla pystyy vaikuttamaan joihinkin ulkoasun tyyleihin, kuten esimerkiksi fontin kokoon, lohkojen taustan väriin ja niissä esiintyviin kuviin. Pääosin editorilla kuitenkin rakennetaan sisältöä, jonka peruselementtien täsmällisemmässä tyylittelyssä tulee osata hyödyntää perinteisiä front-end-teknologioita. Yksi syy Gutenbergin suosioon kehittäjien keskuudessa on sen käyttämä rajapinta, joka mahdollistaa omien lohkojen kehityksen

modernia JavaScriptiä hyödyntämällä, esimerkiksi Reactilla tai Vue.js:llä. (WPBeginner n.d. c.)

Gutenbergin kaikki näkyvät lohkot ovat toteutettu Reactilla, jonka syntaksin hyödyntämiseen WordPressin API tarjoaa helpon väylän, joskaan ei suoraan Reactin oman repositoryn, vaan WordPressin repositoryn kautta. Block Developer Handbookista löytää paljon kattavia ohjeistuksia esimerkiksi komponenttien, elementtien ja toiminnallisuuksien hyödyntämiseen sekä rakentamiseen. Reactiin pohjautuvien kirjastojen avulla on mahdollista rakentaa erilaisia lohkoja sisältöä varten, eli esimerkiksi kortteja, herobannereita tai vastaavia “moduuleja”, joiden avulla sisällön syöttäminen ja muokkaaminen helpottuu loppukäyttäjälle sekä sisällön ulkoasu pysyy vakioituna tilanteesta riippumatta. Koska lohkojen rakentamiseen on saatavilla runsaasti ohjeita ja opastuksia, on omien moduulien muodostaminen suhteellisen lähestyttävää myös aloittelijalle (tästä lisää luvussa 4.4). (WordPress.org n.d. r.)

4.3.3 Vimpaimet

Vimpaimet ovat sisältölohkoja, joita yleensä sijoitetaan sivun sivu- tai alapalkkiin sekä muualle teemassa vimpaimille varatuille alueille. Vimpainten avulla voidaan lisätä helposti kontrolloitavia toimintoja ja ominaisuuksia, esimerkiksi sosiaalisen median ikoneja sivuston alaotsakkeeseen tai sivupalkkiin listauksia artikkeleista. Vimpaimia on ladattavissa lisää hallintapaneelista löytyvästä Lisäosat-osiosta. (WPBeginner n.d. d.)

5 Moduuleja hyödyntävä one pager -verkkosivusto

Tässä luvussa kerrotaan opinnäytetyötä varten tehdyistä moduuleista ja SCSS:n avulla muodostetusta one pagerista, joka hyödyntää mukauttimen tyylittelytoiminnoissa SCSS:n muuttujia. Luvun alussa käydään läpi myös opinnäytetyön taustat ja tavoitteet yleisesti sekä moduulien kehityksen että SCSS:n teemassa hyödyntämisen suhteen. Luvun toisessa osiossa esitellään sekä kehitetyt moduulit että toteutettu one pager -sivusto.

5.1 Työn taustat ja tavoitteet

Opinnäytetyö on toiminnallinen työ, jossa modulaarista front-endiä lähestytään laajan teoriaosuuden avulla. Työssä käytetyt menetelmät ovat laadullisia, kuten johdannossa mainitaan. Koska tarkoituksena on oppia lisää ja kertoa opitusta kattavasti, on toiminnallinen menetelmä ja havainnollistavan teoksen konstruktointi luontevan tuntuinen ratkaisu. Havainnollistaminen tapahtuu esittelemällä projektia varten Gutenbergiin kehitettyjä lohkoja sekä mukauttimen ja Sassy CSS:n yhteistoimintaa teemassa. Havainnollistus toteutetaan yhden sivun verkkosivustona, jonka toteutuksessa käytetyt mukautetut koodit ovat löydettävissä GitHubista, sillä näin koen voivani tarjota kokonaisvaltaisempaa mahdollisuutta tarkastella aihetta. Linkin löydät lähdeluettelon viimeiseltä sivulta eli sivulta 65.

Koodin suhteen on huomioitava, että tapoja koodata on useita, ja samat komponentit sekä keinot mukauttimen ja SCSS:n yhteistoimintaan voisi varmasti toteuttaa myös muilla keinoilla. Lohkoja kehitetään yleisesti Reactin lisäksi myös Vue.js:llä, joka olisi ollut toinen varteenotettava vaihtoehto. Työn kehitysympäristössä käytetään WAMP:ia, jonka avulla WordPressin saa asennettua lokaalisti omalle koneelle. Koodieditorina käytössä on Visual Studio Code sekä Brackets ja terminaalityökaluna Git Bash. Lohkojen kehityksessä hyödynnetään WordPressin repositorysta löytyviä paketteja, sivustoilta saatavia aineistoja sekä aiemmin hankittua osaamista eli lähinnä kykyä soveltaa Reactia, PHP:tä ja SCSS:ää.

5.1.1 Gutenbergille toteutettavat moduulit

Ennen toteuttamisen aloittamista Gutenbergin toiminta oli tullut tutuksi lähinnä mielenkiintoisena sivueditorina, jossa on omat ongelmansa: esimerkiksi käyttöliittymän avulla lohkojen ja elementtien hallinta ei aina onnistu toivotulla tavalla. Vaikka Gutenbergissä voi tallentaa editorilla tehtyjä lohkoja uudelleenkäytettäväksi, mahdollistavat itse kehitetyt lohkot pysyvyyttä sivuston tyylittelyihin, asetteluun ja muokkaamiseen. Loppukäyttäjälle tehdyt lohkot, joissa on asianmukaiset syötteet ilman turhia säätimiä, pysyvät vakaammin hallittavissa ja ennalta

määrätyn tyylin mukaisena. Lisäksi saavutettavuutta voidaan varmentaa paremmin (esimerkiksi kuvaavilla luokilla) ja samalla ehkäistä riskejä tyylittelyn rikkoontumisesta, joita editorin kautta väärään paikkaan lisätyt elementit tai lohkot voivat aiheuttaa pelkästään Gutenbergin editorilla toteutetuille lohkokokonaisuuksille.

Kuten luvussa 4.1.2 käytiin läpi, WordPressin API mahdollistaa JavaScript-kirjasto Reactin hyödyntämisen lohkokehityksessä. WordPressin huhtikuussa 2022 käyttämä Reactin versio on 16.6.3, siinä missä Reactin versio on 18.1.0. WordPressin tarjoama React on siis useamman vuoden jäljessä, joten ihan uusimpia Reactin toimintoja sillä ei pääse toteuttamaan. Lohkoja kehittäessä Reactin pakettien sijaan hyödynnetään WordPressin omasta repositorysta saatavia paketteja. (WordPress.org n.d. t; React n.d.) Vanhemmasta versiosta huolimatta React on luonteva ratkaisu omien lohkojen kehitykseen työssä, mutta muina syinä sen käyttöön ovat myös aloittelijaystävällisyys ja ohjeiden saatavuus.

Moduulien havainnollistamista varten työssä kehitetään kolme yksinkertaista lohkoa:

- Sisältölohko, johon voi lisätä kuvan vasemmalle ja tekstiä oikealle.
- Kortti, johon voi lisätä otsikon, sisältötekstin ja linkin.
- Sosiaalisen median muokattava rivi, jonka voi lisätä sivuston alaotsakkeeseen.

Sisältölohko, kortti ja sosiaalisen median rivi valikoituivat toteutettaviksi, koska ne ovat rakenteeltaan aloittelijalle lähestyttäviä ja koska niissä kaikissa hyödynnetään samoja komponentteja sekä funktioita. Koen, että sisältölohkojen avulla on suhteellisen helppoa oppia ja havainnollistaa yksinkertaisesti kuinka moduuleja luodaan, ja kuinka niitä editoidaan. Kuten myös luvussa 2.3 kerrotaan, kortit, sisältölohkot ja erilaiset alaotsakkeen tulevat toissijaiset informatiiviset elementit ovat yleisiä sekä paljolti käytettyjä verkkosivustoilla ja ne ovat lukijalle myös varmasti entuudestaan tuttuja. Niiden voidaan katsoa olevan keskeisiä

osia tarvittavan sisällön esittämisessä, eli tämän työn kriittisiä komponentteja (Davis & Vesselov 2019, luku 5).

Pyrkimyksenä on havainnollistaa lohkojen kehitystä ja esitellä lohkojen tärkeitä yleispiirteitä sekä myös samalla kertoa niiden oleellisista toimintaperiaatteista ja komponenteista. Kaikkien moduulien tulisi olla saman lisäosamoduulin alaisina, jotta asentamiseen ja hakemiseen kuluisi loppukäyttäjältä mahdollisimman vähän aikaa. Tekstikenttien tulee olla täydennettävissä ja toimivia. Lohkojen kehitykseen tarvitaan kehitysympäristö, johon on asennettu tyhjä WordPress-sivusto. Käyttövalmiina tulee olla myös terminaalityökalu ja koodieditori. Komponenttien toteuttamista varten on sovellettu verkosta saatavilla olevia aineistoja, kuten Ali Alaan (2022) oppikurssia Reactista ja Gutenbergin lohkoista. Moduulien tyylittelyssä halutaan hyödyntää SCSS:ta, ja WordPressin tarjoama wordpress-scripts-paketti tarjoaa käännösmahdollisuuden. Työssä halutaan myös hyödyntää JavaScriptin ESNext-version syntaksia, siistimmän koodin vuoksi. ESNext tulee kääntää selaimille luettavaksi, joten lohkojen riippuvuuksia ja lyhenteitä määrittävässä package.jsonissa tulee olla määritteet npm-komentoihin, eli käännös tapahtuu lisäosan juurikansiosta npm runin avulla. Lisäksi työssä käytetään WordPressin omia ikoneja, eli dashiconeja, jotka löytyvät WordPressin sivuston Developer Resource -osiosta (WordPress.org n.d. u).

5.1.2 SCSS ja projektin teema

Sass antaa CSS:lle ohjelmointikielien ominaisuuksia, kuten luvussa 3.3.3 käsiteltiin. WordPressissä Sassia voidaan hyödyntää Sassy CSS:n avulla lisäosien ja teemojen kehitysympäristössä tapahtuvan tyylittelyn lisäksi myös klassisten teemojen mukauttimessa. Kuten luvussa 4.2.5 läpikäytiin, mukautin tarjoaa käyttöliittymän sivuston editointiin ja sen mahdollistamat muokkaustoiminnot riippuvat täysin teeman takana olevasta kehitystyöstä. Suoraa väylää SCSS:n käyttöön mukauttimessa ei ole, vaan tätä varten tarvitaan teeman kansioon ladattava erillinen kääntäjä, jotta SCSS ja sivuston käyttämä vanhanaikainen PHP saadaan toimimaan yhteistyössä. Tätä varten opinnäytetyössä käytetään

scssphp-kääntäjää, helppokäyttöisyyden ja kattavien ohjeistuksien vuoksi. (A White Pixel 2020; scssphp 2022.)

Sassy CSS:n hyödyntämisen suhteen tavoitteena ovat seuraavat asiat:

- Toteutettavien komponenttien tulee olla mukauttimen kautta tyyliteltävissä.
- Sivuston otsakkeiden, lohkojen ja tärkeiden elementtien tulee olla mukautettavissa.
- Sivuston fontin koon ja reunojen pyöreyden tulee myös olla hallitusti säädettävissä mukauttimessa.
- Tyylittelyn muutoksia tulee pystyä havainnollistamaan mahdollisimman selvästi loppukäyttäjälle.

scssphpcompilerin avulla voidaan hallinnoida muuttujia (engl. variable) reaaliajassa sivuston mukauttimen kautta. Taustatyö vaatii asettelua etenkin teeman functions.php-tiedoston kanssa: erittäin tärkeää on muistaa määrittää .scss-tiedostoon kaikki arvot ja muuttujat sekä hyödyntää niitä sivustolla olevissa elementeissä. (A White Pixel 2020.) Hyödyntäminen tapahtuu antamalla muuttuja arvoksi kehitettäville lohkoille tai Gutenbergissä esiintyville editorilla rakennetuille elementeille, luokan tai id:n kautta.

Teeman kannalta tärkeää on minimalistisuus, jotta sitä voidaan hyödyntää havainnollistamisessa ilman turhia ominaisuuksia, ja jotta kehitettävien moduulien tyylittelyt eivät aiheuttaisi ristiriitoja teeman mukana tulevien tyylittelyjen kanssa. Ruudukkoja voidaan hyödyntää näin helpommin, koska sivustolla ei tarvitse käyttää !important -sääntöä tyylittelyjen läpisaamiseksi. Työssä on päädytty käyttämään teemana klassista aloittelijateema Underscoresia. Underscores (2022) on suunnattu teemojen kehityksestä kiinnostuneille ja se tarjoaa tuen moniin erilaisiin teeman toiminnallisuuksiin, mutta vaatii jonkin verran asettelua. Se on kuitenkin ideaali pohja kehitysprojekteja varten minimalistisuudellaan. Underscoressissa ei esimerkiksi ole alaotsakkeen vimpaimille tarkoitettua tilaa, vaan se tulee luoda itse functions.php ja footer.php-tiedostoja täydentämällä.

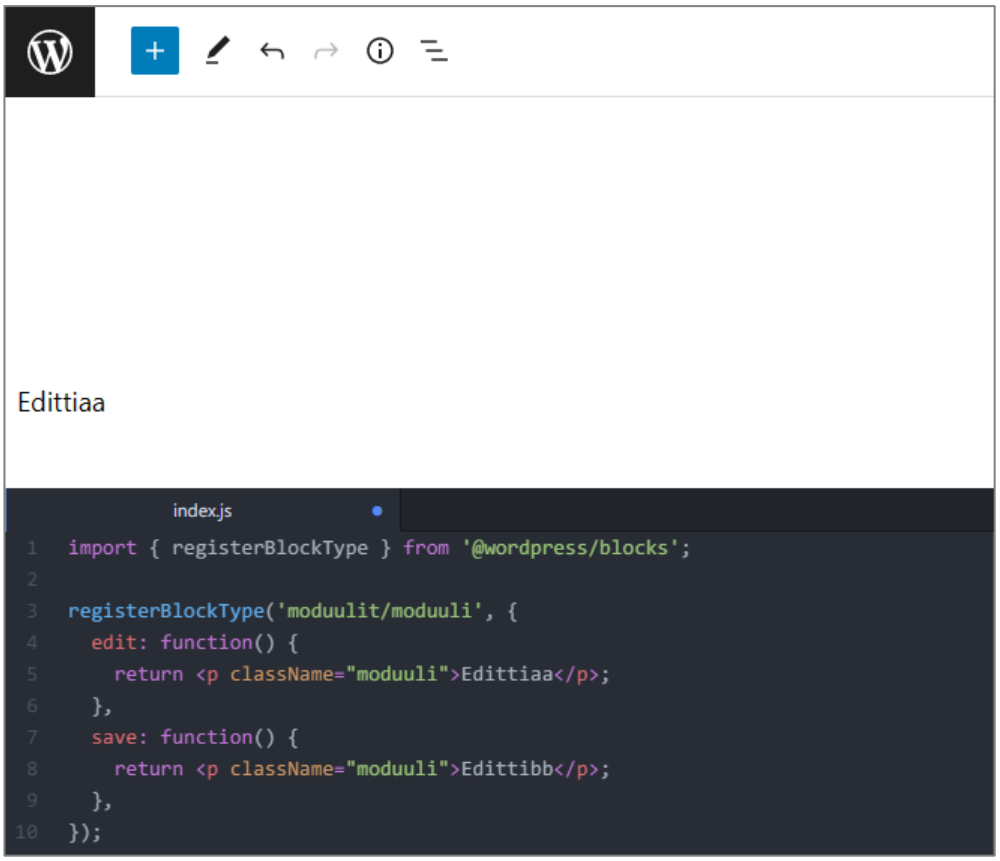
Underscoresin tarjoaman mukauttimen avulla voidaan vaikuttaa myös vain taustan ja tekstin väriin. Koska työn tarkoituksena on toteuttaa one pager havainnollistamaan moduuleja, työssä ei hyödynnetä artikkeleita. Tämän johdosta Underscoresin sivutemplate kopioidaan ja sivupalkki poistetaan sisällön templaten koodista, samalla alaotsakkeeseen luodaan kolme eri vimpainlohkoa. Teemassa on päädytty käyttämään kirjaisimina Roboto Monoa sekä Space Monoa, koska ne sopivat front-endin teemaan kirjaisimien muistuttaessa useiden eri koodieditorien kirjaisimia.

5.2 Lohkojen kehitys

Koska Gutenbergin lohkojen kehitys on sidoksissa WordPressin käyttämään Reactin versioon ja WordPressin repositorysta löytyviin paketteihin, ei työssä käytetä Reactia suoraan. Pelkän create-blockin avulla saatavat tiedostot eivät riitä muuhun kuin yksinkertaisen tekstin tulostamiseen ruudulle editorissa ja sivulla, joten työssä hyödynnetään myös seuraavia paketteja: @wordpress/i18n, @wordpress/scripts @wordpress/component, @wordpress/blob, @wordpress/block-editor, @wordpress/element ja @wordpress/compose.

Create-blockin myötä tuleviin block.json ja package.json-tiedostoihin tallennetaan kehitettävien lohkojen tiedot metadatana. Block.json-tiedoston avulla lohkon tyylitiedostot ladataan vain silloin, kun lohko esiintyy näkyvällä sivulla (WordPress.org n.d. t). Tiedostossa voidaan määrittää esimerkiksi attribuutteja lohkon kolumnien määrän vakiolle sekä mahdollistaa erilaisia tekstin muokkaustoimintoja.

Yksinkertaisimmillaan tulostettava lohko on jokin elementti, joka JavaScriptin avulla esitetään näkyviin. Kuvan 18 tiedostoa index.js:ää tarkastellessa voidaankin huomata, että registerBlockTypen alaisuudessa ovat edit ja save -funktiot, joiden avulla sisältö sekä näytetään editorissa että tallennetaan tietokantaan sivustolle näytettäväksi.



Kuva 18. Kehitettävän lohkon miltein tyhjä koodi. Ylhäällä editoriin printtaantuva teksti, joka sivulle esiintymistä varten hyödyntää alla olevaa koodia.

Savelle ja editille kannattaa luoda erilliset tiedostot, jotta tiedostojen editoiminen ja koodi olisi selkeämpää ja sitä ei ajetta turhaan.

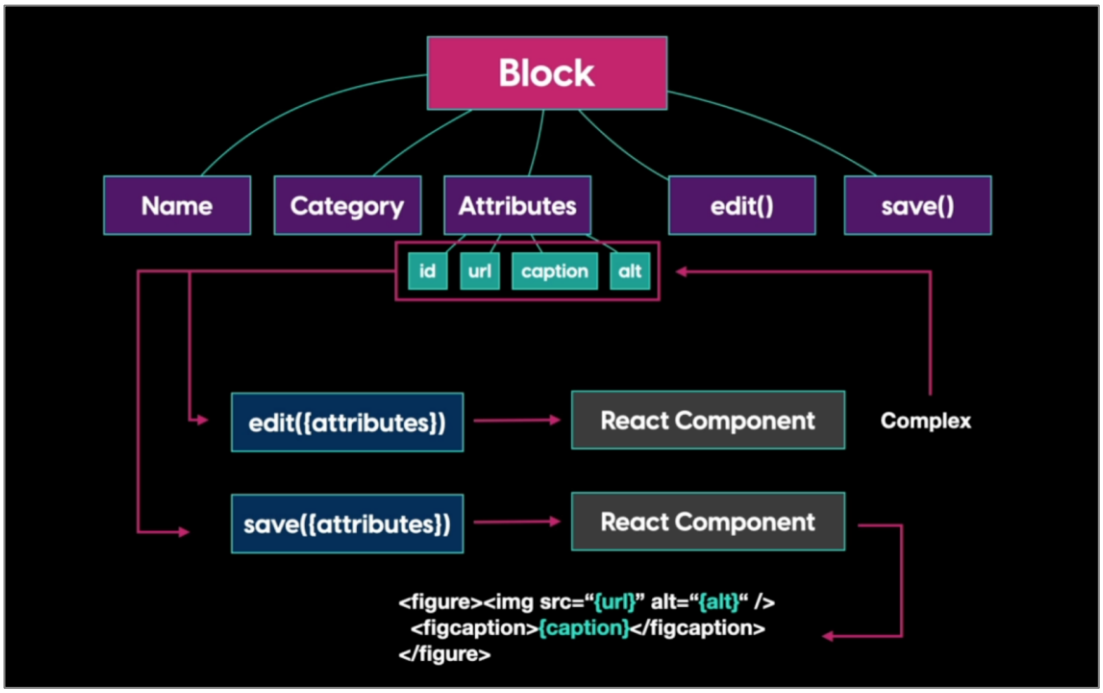
Kehitettyjä lohkoja voidaan myös muuntaa kokonaisuuksiksi, joissa on useampia lohkoja. Tällöin uusille kehitettäville alalohkoille kannattaa luoda omat kansionsa, joissa niillä on omat index.js-, edit.js- ja save.js-tiedostonsa. Tätä varten WordPress tarjoaa InnerBlocks-komponentin, jonka ominaisuuksia (esim. esityssuunta, sallitut lohkot ja kolumnien määrä) voidaan määrittää lohkokansion juuresta löytyvään edit-tiedostoon. Sekä editin että saven koodit vaativat usein erilaisia proppeja ja määritteitä (esim. const) toisiinsa nähden, mutta ne hyödyntävät aina samaa esiin kutsuttavaa ja muokattavaa metadataa index.js-tiedostosta. Jotta tekstistä saadaan editoitava, tarvitaan komponenttia nimeltä RichText. RichText sisältyy block-editorin pakettiin.

Liitteestä 1 löytyy pelkistetysti koodia RichTextin hyödyntämisestä.

RichTextia hyödynnetään täydentämällä kehitettävän lohkon index.js-tiedostoon attribuutit, joita kutsutaan esiin editin ja saven alaisuuteen. Koska kyseessä on tekstiä, haluamme määritellä tyypiksi stringin, sourceksi html-koodin ja valitsimeksi p-tagin tekstille ja h2-tagin otsikolle (ks. liite 1. a). Tämän jälkeen niissä tiedostoissa, joissa attribuutteja tarvitaan, tulee hakea attribuutit ja määrittää ne constin avulla hyödynnettäväksi, täsmälliset nimet aaltosulkeiden sisään (ks. liite 1. b). Aaltosulkeilla kerrotaan Reactille JSX:n avulla kyseessä olevan muuttuja. Jotta tekstejä voisi editoida, tulee edit.js:n muutokselle määrittää Reactin event handler onChangen avulla, että uusia attribuutteja tulee generoida komponentin syötöstä ja vanha attribuutti muuttaa uudeksi (ks. liite 1. c). Lisäksi komponentti tulee määrittää molempien tiedostojen (edit ja save) returniin. Liiten 1 kohdassa d) olevassa esimerkissä on nähtävissä edit-tiedoston RichText-komponentin käyttö ja saman liitteen kohdasta e) nähdään save-tiedoston käyttö, johon on määritelty {attribuutti && <RichText..\> }, jonka avulla kerrotaan että elementit esitetään sivulla vain, jos niihin on syötetty jotain. Reactin syntaksia käyttämällä RichTextiin määritetään tag (tässä tapauksessa p ja h2), luokkanimi, placeholder-teksti, value (eli attribuutti) ja onChangen nimi (eli contenttext ja contenttitle). Samankaltaisella periaatteella toimii muidenkin tekstinsyöttökenttien tekeminen.

Kuvansyöttöä varten tarvitaan useampaa komponenttia ja toimintoa: mm. MediaPlaceHolderia, joka hyödyntää osoitteen url-attribuuttia, WordPressin blob-pakettia sekä MediaReplaceFlowia, jonka avulla kuvan voi vaihtaa. Blob-paketin avulla kuva saadaan tuotua näkyviin ja se voidaan myös poistaa näkyvistä. Sitä hyödyntää const, joka katsoo BlobURL:n tilaa Reactin useState-hookin avulla. Lisäksi tarvitaan useEffect-hookkia hyödyntäviä määrittelyjä, joiden avulla mahdollistetaan url-attribuutin, alt-tekstin ja id:n muutoksia. Sisältölohkon edit.js-tiedoston vaatimat säädöt voi nähdä pelkistetysti liitteessä 2. Lohkojen kannalta kokonaisuus toimii usean eri attribuutin, koodien ja komponenttien kautta. Kuvassa 19 on nähtävissä visuaalinen havainnollistus siitä, kuinka lohkot koostuvat ja toimivat. Gutenbergille kehitettävät lohkot ovat useiden eri teknologioiden summa ja Reactin hookkien, komponenttien ja yleisesti metadatan avulla

pystytään vaikuttamaan moninaisesti siihen mitä verkkosivustolla tallennetaan ja esitetään.



Kuva 19. Lohkojen toimintaperiaate. (Kuva: Ali Alaa 2022)

Työssä kehitetyt lohkot toteutetaan pääosin esiteltyjä komponentteja hyödyntämällä. Lisäksi sosiaalisen median rivistössä hyödynnetään Reactille tarkoitettua ulkoista lisäosaa eli dnd kittia, joka on moderni vedä ja pudota -tyylinen kirjasto Reactille (dndkit n.d).

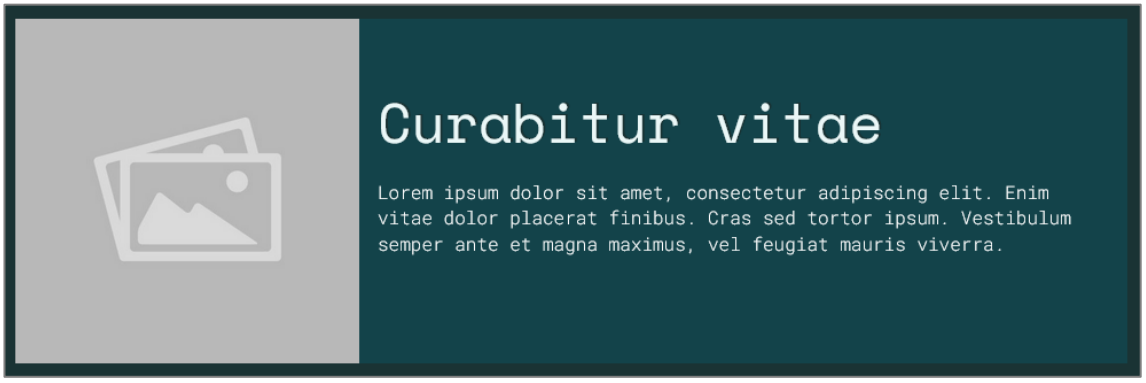
5.2.1 Moduuli, sisältömoduuli, kortti ja somerivi

Aiemmin perustellun johdosta päädyin luomaan kolme Gutenberg-lohkoa havainnollistamaan moduuleja. Koska kyseessä on havainnollistus, on toteutuksessa nähtävien säiliöiden tekstuaalinen sisältö täytetty lorem ipsumilla.

Kehitettävän lohkon kansioon on luotu projektin mukaan nimetyt PHP- ja package.json -tiedostot, joiden koodit on nähtävissä liitteessä 2. Päämoduulin koodi toimii NestedBlocks -komponentin avulla tilana, johon muita moduuleja voi sijoittaa (ks. liite 3). Se hyödyntää erilaisia Gutenbergin editorinäkymän

sivupalkin toimintoja mahdollistavia komponentteja, kuten esimerkiksi kolumnien määrän säädintä, jolle tiedostossa on asetettu kolumnien minimi- ja maksimimäärä komponentin RangeControl avulla. InnerBlocksille on määritelty ominaisuuden avulla sallittavat lohkot, joita sen sisällä voidaan näyttää. Sisällön horisontaaliseen esittämiseen käytetään proppia. Päämoduulin tallennuksen hakevassa tiedostossa save.js nähdään useBlockPropsin sekä kolumnisäädön myötä kolumnien määrään sidonnaisen luokan hyödyntäminen aaltosulkeissa funktion sisällä. Ruudukkoon asettelua tukee display: flexbox ja CSS:n laskufunktioa hyödyntävä maksimileveys. Lisäksi kolumnien leveys tasataan SCSS:n avulla, laskemalla säiliössä olevien kolumnien määrä WordPressin generoimasta koodista for $i from 1 through 3 { valitsin, jossa kerrotaan luokka, josta luku hakea { leveys jaetaan } } -koodilla.

Kuvassa 20 näkyvä sisältömoduuli hyödyntää sisällön esittämiseen MediaPlaceHolderia ja RichTextia sekä niihin liittyviä attribuutteja ja muita komponentteja. Moduuli käyttää Flexboxia, ja sen sisältö on tyylitelty näkymään rivissä. Moduulin käyttämä koodi on nähtävissä kokonaisuudessaan liittessä 4.



Kuva 20. Sisältömoduulin desktop -näkymä tallennetulla sivulla.

Jos kuvaa ei ole ladattuna, ei kuvalle varattua tilaa myöskään näy (kuva 21). Editorinäkymässä tosin kuvan placeholder näkyy silloinkin, minkä johdosta lohko vaatisi vielä jatkokehitystä.

Kuva 21. Sisältömoduuli tallennetulla sivulla ilman kuvaa.

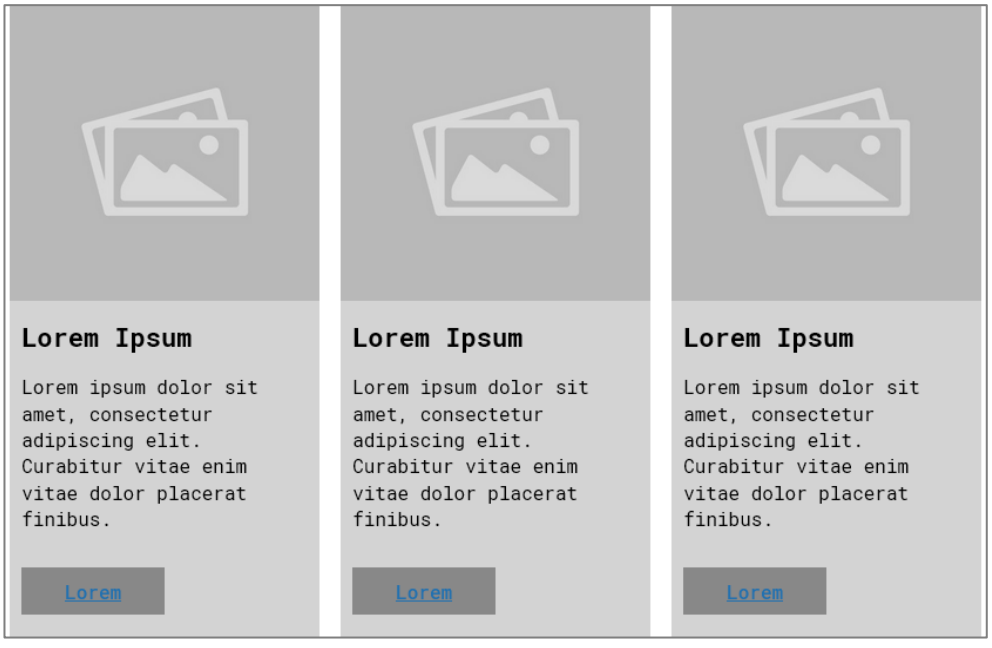
Sisältömoduuli näyttää hieman erilaiselta editorissa, sillä teeman värit tulevat mukauttimesta, jonka kautta asetellut värit eivät toimi sivueditorissa. On turhaa värittää lohkot joillain tietyillä väreillä, jos ne eivät vastaakaan käyttäjän asettamaa värimaailmaa. Tämän johdosta päädyin tyylittelemään taustat harmaalla, jotta lohko olisi kuitenkin tarpeeksi havainnollistava (kuva 22).

Kuva 22. Kuvan sisältävä sisältömoduuli editorin näkymässä.

Toisena toteutettuna moduulina on kortti, joka koostuu yhdestä MediaPlaceHolderista ja kolmesta RichText-komponentista: otsikko, teksti ja painike, eli se on rakenteeltaan hyvin geneerinen. Geneerisyyteen on päädytty selkeyden ja tuttuuden johdosta. Kuvassa 23 nähtävissä on kolmeen kolumniin asetellut kortit. Kaikki kortit hyödyntävät samaa komponenttia, joka toimii parhaiten kolmen kolumnin asettelussa.

Kuva 23. Kortteja rivissä tallennetulla sivulla.

Myös kortit hyödyntävät Flexboxia asetteluissaan ja toimivat pienellä resoluutiolla vaihtamalla sisällön esityssuunnan display: columnista display: rowiin pienemmällä resoluutiolla. Korttiin tulevalle kuvalle on säädetty mahdollisuus uuteen alt-tekstiin, joka voidaan määrittää editorisivun kautta. Kuvassa 24 on kuvakaappaus korteista editorin näkymässä. Korttien paikkaa pystyy vaihtamaan ja korttien tekstisäiliöiden tilan laatikot seuraavat pääsäiliön korkeutta ja painikkeet asettuvat aina samalle korkeudelle.



Kuva 24. Luodut kortit editorin näkymässä.

Kortit näyttävät pelkistetyltä editorinäkymässä, sillä ne vaatisivat vielä lisätyylittelyä, johon jäi rajoitetusti aikaa työn laajuuden puitteissa. Tästä huolimatta niiden avulla voidaan havainnollistaa tarpeeksi hyvin loppukäyttäjälle sitä, mitä mihinkin kenttään tulee täyttää. Näytettävän sisällön kannalta otsikko on h5-tagilla, teksti p-tagilla ja painike div-tagilla. Kuvassa 25 on kuvakaappaus kortin moduulin save.js-tiedoston koodista, jonka avulla sisältö esitetään sivulla.

save.js

```
import { useBlockProps, RichText } from '@wordpress/block-editor';

export default function Save({attributes}) {
    const { titlec, textb, textc, url, alt, id } = attributes;
    return (
        <div { ...useBlockProps.save() }>
            {url && (
                <img src={url} alt={alt} className={ id ? `wp-image-${id}` : null }
             />)}
              <div className="cardwrap">
              <div className="textwrap">
            {titlec && <RichText.Content tagName="h5" value={titlec} />}
            {textc && <RichText.Content tagName="p" value={textc} />}</div>
            {textb && <RichText.Content tagName="div" className="cardb" value={textb} />}
        </div></div>
    );
}

```

Kuva 25. Korttimoduulin save.js-tiedoston koodi, jonka avulla korttien sisältö esitetään sivulla.

Korttimoduulissa käytetty koodi on nähtävissä liitteessä 5. RichText-komponenteille on luotu omat div-säiliöt helpottamaan tyylittelyä. Pohjana hyödynnettävän Underscoresin minimalistinen tyylittely ei aiheuta konflikteja säiliön korkeuksien ja muiden tyylittelyjen asettelussa. Kuvassa näkyvä useBlockProps on WordPressin block-editor-paketin kautta hyödynnettävä hookki moduulien luokkien esittämiseen WordPressin vakioimalla wp-block-alkuliitteisellä luokalla. (WordPress.org n.d. v).

Sosiaalisen median riviä varten työssä on hyödynnetty aiemmin mainittua dnd kit -kirjastoa, jonka avulla ikoneita voi vetää ja pudottaa rivin alueella. Asentamisen ja käyttöönoton kuvaus kokonaisuudessaan on turhan laaja opinnäytetyön tarkoitukseen nähden, joten oleellista aiheesta on tietää, että kirjaston tarjoama

vedä ja pudota -toimintatapa perustuu monen eri tuotavan komponentin yhteistoimintaan sekä vaatii toimiakseen useamman eri funktion määrittelyn. Ikonien lisäämiseksi tarvitaan Icon-, Tooltip- ja Button -komponentit @wordpress/components-paketista. Lisäksi kehitettävän lohkon index.js-tiedostoon tulee määrittää sosiaalisen median linkeille taulukko (engl. array), lähde (engl.source) josta haetaan ja valitsin (engl. selector). Nämä attribuutit tuodaan jälleen kerran constin avulla hyödynnettävään tiedostoon. Linkkien lisäämiseksi käytetään Reactin useState ja useEffect-hookkeja ja ikonien poistamiseen sekä lisäämiseen slicea, jonka avulla taulukossa olevia attribuutteja voidaan muokata. Lisäksi pitää tietenkin tehdä asiaan kuuluvat tekstilaatikot TextControlia hyödyntämällä. Kuvassa 26 nähtävissä on ote sosiaalisen median rivistön edit.js-tiedoston returnia. Aaltosulkeiden sisällä oleva selectedLink !== undefined tarkoittaa että koodi esiintyy vain silloin kun editorissa on ikoni valittuna, eli valittu kohde ei ole määrittämätön. SelectedLink on aiemmin koodissa määritelty hyödyntämään index.js-tiedostoon tallennettuja attribuutteja.

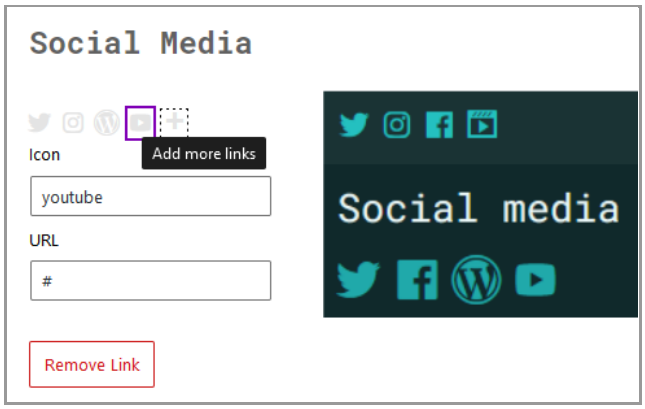
edit.js

```
                                    );
                                } ) }
                            </SortableContext>
                        </DndContext>

                { isSelected && (
                <li className="wp-block-moduuli-somerow-add-more">
                   <Tooltip text={__("Add more links", 'somerow')}>
                       <button aria-label={__("Add more links", 'somerow')} onClick={addNewSomeItem}>
                           <Icon icon="plus" />
                           </button>
                           </Tooltip>
                </li>
                )}
            </ul>
        </div>

            {selectedLink !== undefined && (
            <div className="wp-block-moduuli-somerow-link-form">
                <TextControl label={__('Icon', 'text-somerow')} value={socialLinks[selectedLink].icon } onChange=
                {(icon) => {
                    updateSocialItem( 'icon', icon);
                } }
                />
                <TextControl label={__('URL', 'text-somerow')} value={socialLinks[selectedLink].link} onChange={(link)
                => {
                    updateSocialItem( 'link', link );
                }} />
                <br />
                <Button isDestructive onClick={removeSocialItem}>
                    { __('Remove Link', 'text-somerow')}
                </Button>

                </div>
            )}
            </div>
    );
}

```

Kuva 26. Kuvakaappaus sosiaalisen median lohkon edit.js-tiedostosta.

Kokonaisuudessaan sosiaalisen median lohkojen toteutuksessa käytetty koodi löytyy liitteestä 6. Sosiaalisen median riviä voi hyödyntää moduulin sisällä ja

esimerkiksi teeman alaotsakkeessa. Kuvan 27 ylärivillä on nähtävissä tallennetun sivun sosiaalisen median rivistö sivun sisältöalueella ja alhaalla olevassa rivissä, alaotsakkeen vimpaimeen upotettuna. Kuvan rivistö on tyylitelty hyödyntämään display: inline-block -määrittelyä.



Kuva 27. Sosiaalisen median lohko editorissa (vasen) sekä moduulin sisällä sisältöalueella (oikealla ylhäällä) että alaotsakkeessa (oikealla alhaalla).

Alaotsakkeeseen upotettava sosiaalisen median lohko on tyylitelty huomattavasti isommaksi kuin sisältösivulle tuleva, sillä alaotsakkeen ikonien tarkoitus on olla selkeästi havaittavissa sivuston loppuun scrollatessa.

Kehitettyjen lohkojen järjestystä voi muokata ja niitä voi käyttää uudelleen eri kohdissa ja tarkoituksissa. Lohkot ovat täten ainakin alkeellisesti modulaarisia.

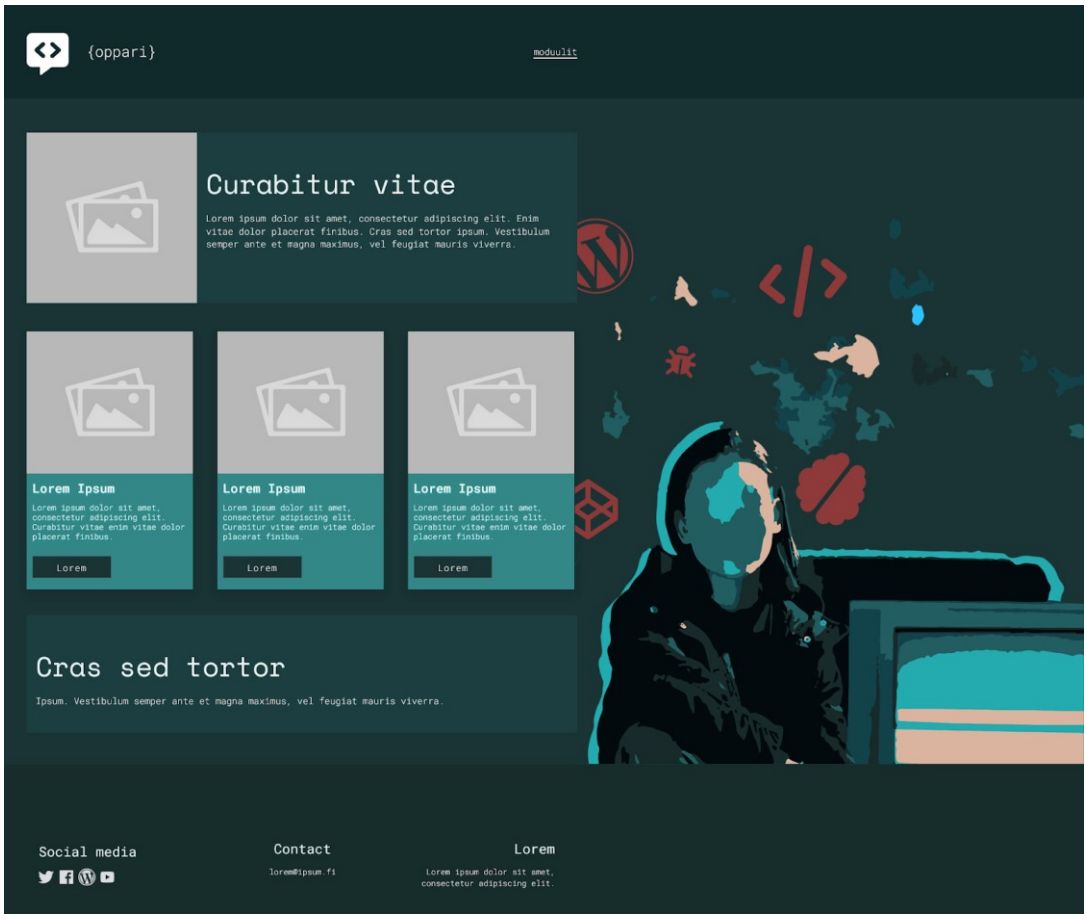
5.2.2 Lohkojen kehityksen haasteet

Lohkojen kehityksessä tulee ymmärtää vahvasti HTML:n, CSS:n ja JavaScriptin perusteita. Kuten kaikessa ohjelmoinnissa, syntaksin oikeaoppinen käyttö on uutta ohjelmointikieltä opettelevalle välillä hankalaa. Selaimen consolen avulla saa kuitenkin paljon informaatiota ja wordpress-scriptsin käännöstoiminto ilmoittaa virheistä. Koska kyseessä on ohjelmointikieli, syntaksin ollessa esimerkiksi väärällä päätteellä “suljettu”, lopettaa kehitettävä lohko toimintansa. Editorin sivua sai päivittää urakalla ja usein muutoksia tehdessä tallennetut sisällöt joutui asettelemaan uusiksi. RichTextin hyödyntäminen osoittautui yksinkertaiseksi, sillä se ei vaadi toimiakseen kuin onChangen, attribuutit ja muutaman rivin

koodia returniin ollakseen HTML-muodossa esitettävä. Etenkin kuvasyötön soveltaminen tuotti aluksi päänvaivaa, vaatiessaan useampia erilaisia määritteitä, mutta komponentin käyttö avautui hyvin nopeasti sovellettavaksi kopioimalla, tutkimalla ja itse yrittämällä. Lohkojen tyylittely SCSS:n avulla puhtaaseen teemaan oli kuitenkin hyvin yksinkertaista.

5.3 SCSS:n hyödyntäminen teemassa

Luotuihin Gutenberg-lohkoihin pohjautuva one pager on suhteellisen yksinkertainen kokonaisuus, jonka tyylittelyissä on hyödynnetty runsaasti Flexboxin sisällön asettelun määrittelyjä ja SCSS:n muuttujia. Lohkojen ja teeman tyylittelyt ovat toteutettu erillisinä tiedostoina toteuttuja, jotta kumpaakin eli sekä teemaa että lohkoja voisi käyttää erillisenä toisistaan. Koska tavoitteena on pitää lopputulos yksinkertaisena ja havainnollistavana, ei teeman kehityksessä ollut tarvetta laajemmille tyylittelyille kuin esitystavat, leveydet, korkeudet, värit, fontit ja tilat. Toteutettu kokonainen sivu on nähtävissä kuvassa 28.



Kuva 28. Lopullinen tuotos, jossa esiintyvien lohkojen järjestystä ja esitystapaa voi muuttaa.

Värimaailmaan on päädytty, koska se sointuu taustakuvaan (oikealla sivussa). Taustakuva on editorin kautta ladattava featured image, joka on luotu työtä varten tekijänoikeudellisesti vapaista elementeistä hyödyntämällä Adobe Photoshopia, Illustratoria ja FontAwesomen ikoneja. Sivun sisällön ruudukko on asetettu maksimileveyteen 970 px, sillä halusin desktop-näkymän taustakuvan sijaitsevan oikealla puolella sivua. Sivun yläotsake koostuu mukauttimen kautta lisätystä kuvasta ja sivun nimestä {oppari} sekä navigaatiosta, johon luodut sivut tulevat. Koska työtä varten tehtiin vain yksi sivu, ei navigaatiossa ole sen enempää linkkejä. Sisältöruudukossa ensimmäisenä oleva moduuli on aiemmin esitelty sisältömoduuli. Sen alla olevat kortit esiintyvät visuaalisesti miellyttävästi vierekkäin. Moduuleissa käytetty kuva on yleinen tilanvaraaja, jonka avulla voidaan havainnollistaa sijan olevan varattu kuvasisällölle. Viimeinen sisältöruudukon moduuli on myös sisältömoduuli, mutta ilman kuvaa ja eri tekstillä.

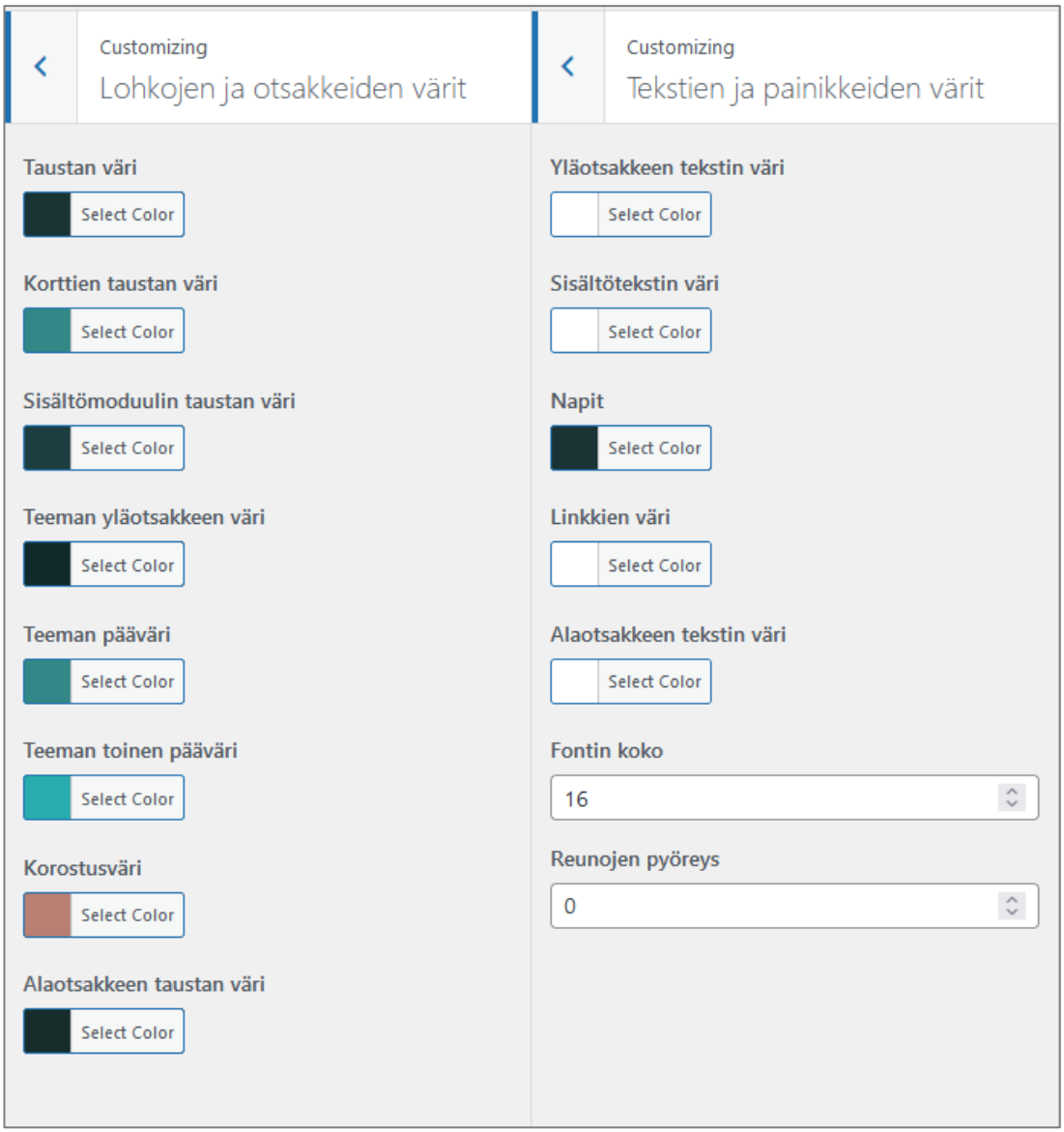
Jotta SCSS:n muuttujia voidaan hyödyntää WordPressin mukauttimessa, tulee luvun 5.1.2 mukaisesti käytössä olevan teeman kansioon ladata scssphp-kirjasto. Kirjaston käyttö on yksinkertaista, sillä se sisällytetään teeman functions.php-tiedostoon, johon myös luodaan mukauttimen asetukset ja kentät customize_register -toiminnolla sekä add_sectionilla, add_setting ja add_controllilla. Lisäksi tiedostoon scssphp:n käynnistämistä ja määrittelyä koskevia asetuksia sekä muuttujien määritykset. Functions.php-tiedostosta löytyvät muuttujat nähtävissä kuvassa 29. Liitteessä 7 on myös löydettävissä sama koodi kokonaisuudessaan.

```
$variables = [
    '$main' => get_theme_mod('main', '#328787'),
    '$header' => get_theme_mod('header', '#10292b'),
    '$bg' => get_theme_mod('bg', '#1a3334'),
    '$headers' => get_theme_mod('headers', '#fff'),
    '$cards' => get_theme_mod('cards', '#328787'),
    '$content' => get_theme_mod('content', '#13434a'),
    '$footer-c' => get_theme_mod('footer-c', '#383053'),
    '$footer-t' => get_theme_mod('footer-t', '#fff'),
    '$secondary' => get_theme_mod('secondary', '#28acad'),
    '$color-tertiary' => get_theme_mod('color-tertiary', '#b1e1df'),
    '$color-fourth' => get_theme_mod('color-fourth', '#1a3334'),
    '$color-ctext' => get_theme_mod('color-ctext', '#fff'),
    '$link-b' => get_theme_mod('link-b', '#24c0c0'),
    '$theme-text-size' => get_theme_mod('theme-text-size', '16') . 'px',
    '$theme-border-radius' => get_theme_mod('theme-border-radius', '10') . 'px',
];
$compiler->setVariables($variables);
```

Kuva 29. scssphp:n avulla luodut muuttujat functions.php tiedostossa.

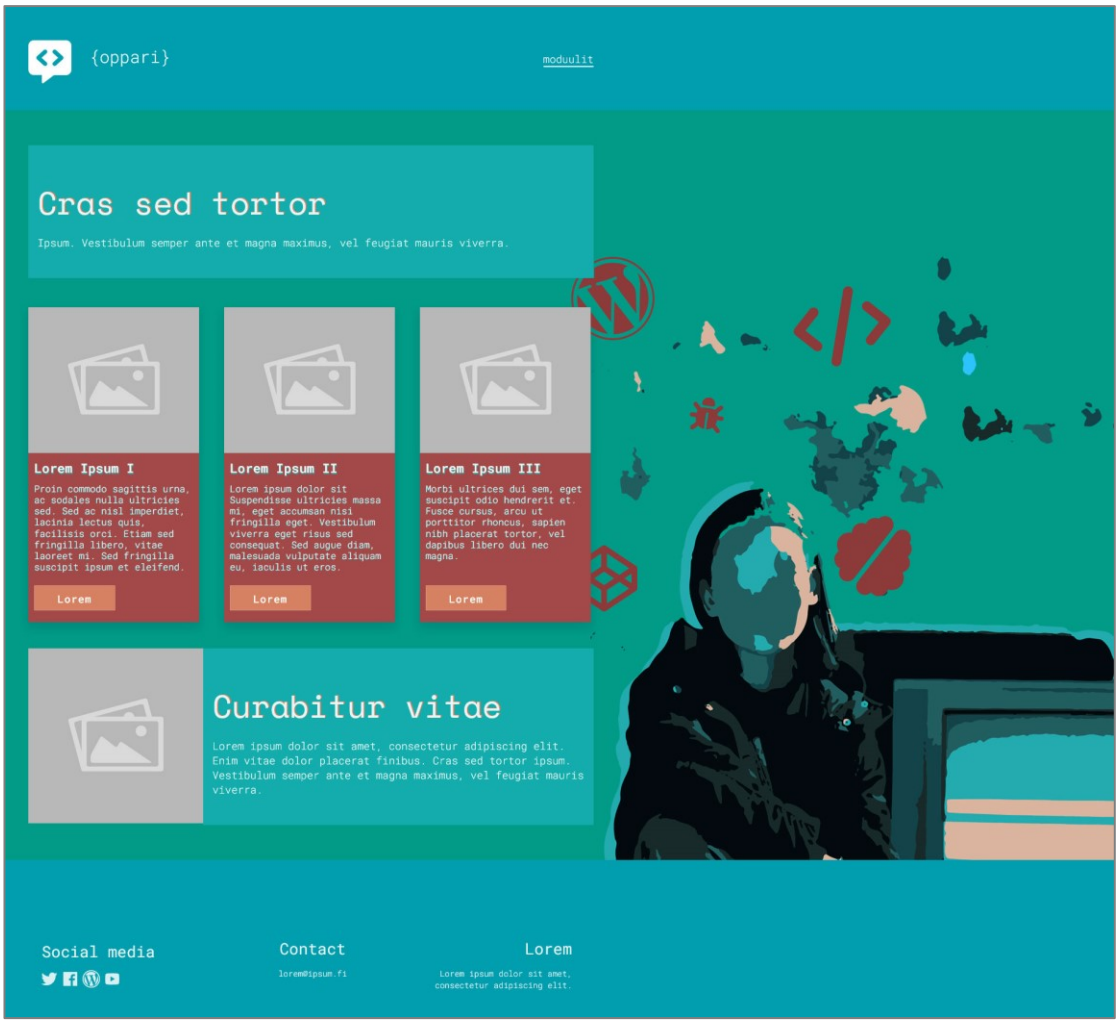
Näiden muuttujien tulee myös löytyä teeman kansiosta löytyvästä .scss-tiedostosta (!default -säännöllä), samoilla arvoilla kuin functions.php-tiedostossa esiintyvät muuttujat. Alueet myös vaativat vielä erillisen rekisteröinnin ja toimintoja pystyäkseen myös tallentamaan sivuston mukauttimeen tehtävät muutokset. (A White Pixel 2020.) Kokonaisuudessaan työssä käytetyn functions.php-tiedoston koodi on nähtävissä liittessä 7.

Työtä varten mukauttimeen on toteutettu kaksi erillistä osiota elementtien värien säätämiseen: lohkoille ja otsakkeille sekä teksteille ja painikkeille. Kuvassa 30 on nähtävissä mukauttimen kautta mahdollistetut värimuutokset.



Kuva 30. scssphp:n avulla luodut mukauttamismahdollisuudet

Mukauttimesta löytyvien luotujen säätimien kautta voidaan vaikuttaa useisiin teeman väreihin: taustoihin, teksteihin sekä fontin kokoon ja reunojen pyöreyteen. Functions.php-tiedostoon tallennetun datan ja SCSS-tiedostosta löytyvien muuttujien avulla mukauttimesta voidaan nopeasti palauttaa alkuperäiset värisäädöt yksittäin, jos värien säätäminen menee pieleen. Kuvassa 31 on nähtävissä mukauttimen kautta säädetyt uudet värit ja hieman eri tavalla asetellut moduulit. Lisäksi fonttikoko on vaihdettu 16 pikselistä 17 pikseliin.



Kuva 31. Mukauttimen kautta säädetyt värit, jotka voi helposti palauttaa takaisin vakioasetuksiin.

A White Pixelin (2020) tarjoaman ohjeen myötä scssphp -kirjastoa on ollut helppo käyttää, eikä sen asentamisessa tai soveltamisessa esiintynyt ongelmia.

6 Pohdinta ja lopputulokset

Työn aikana pääsin perehtymään modulaarisuuden käsitteeseen, verkkosivustojen toteuttamiseen liittyviin termistöihin, moderneihin front-end-teknologioihin ja WordPressin toimintatapoihin. Toiminnallisen osion myötä sain kerrytettyä lisää informaatiota lohkojen toimintatavasta, Reactista ja SCSS:stä. Toiminnallista osiota toteuttaessa opin erilaisten Reactin ja WordPressin tarjoamien komponenttien sekä erilaisten funktioiden käytöstä. Lisäksi opin lisää koodin

soveltamisesta ja parantamaan ymmärrystäni metadatan, API:n ja selaimen yhteistyöstä sekä modulaarisuuden konseptista. Omalta osaltani opinnäytetyön tavoitteet täyttyivät. Työn teoriaosuuden olisi voinut jättää lyhyemmäksi, mutta silloin sen avulla välitettävä informaatio olisi ollut turhan niukkaa laajaan aiheeseen nähden. Etenkin lohkojen kehityksessä vaaditaan syvällistä ymmärrystä Reactista ja Gutenbergistä, ja siinä hyötyy SCSS:n käyttämisestä, mutta itse koodin soveltaminen on suhteellisen aloittelijaystävällistä. Lohkojen näkyvät osat ovat usean eri komponentin, teknologian ja tiedoston yhteistyöstä syntyviä tuloksia.

Modulaaristen teknologioiden ja moduulien havainnollistaminen onnistui alkuperäisen suunnitelman mukaisesti, joskin osoittautui vaativammaksi kuin alkuun luulin. Moduulien muodostaminen ilman minkäänlaista kokemusta front-endistä olisi varmasti ollut hyvin hankalaa. Itse kehitystyö osoittautui miellyttäväksi ja lähestyttäväksi kattavien ohjeistuksien avulla. On joitakin lisätoimintoja, joita olisi voinut lisätä kehitettyihin lohkoihin, mutta ne jäivät toteuttamatta, koska aikaa oli rajatusti ja koska työn kannalta toiminnot eivät olleet oleellisia. Editorissa näkyvien MediaPlaceHolderien olisi ollut toivottava olla sellaisia, että niiden säiliöt suurenevat editorinäkymässä vasta kuvan ollessa lisättynä. Editorissa näkyviin lohkoihin olisi myös ollut suotavaa saada toimimaan mukauttimen kautta syötettävät värit sekä sosiaalisen median ikonit hyödyntämään FontAwesome -ikonipankkia. Lisäksi sisältömoduulin kuvan ja tekstin tilan olisi ollut hyvä olla liikuteltavissa, kun nyt se on staattinen tekstiä ja kuvaa vailla oleva lohko. Tämä toisi lisäarvoa käytettävyyden kannalta. Mahdollisesti myös liukusäädin laatikoiden leveydelle olisi voinut olla kätevä. Sosiaalisen median rivistö on jostain syystä sitä mukauttimen kautta muokatessa hankalakäyttöinen, etenkin alaotsakkeen vimpaimen lohkossa: linkkejä lisätessä selain pyrkii jatkuvasti päivittämään sivua ikonia klikattaessa. Tämän voi olettaa johtuvan siitä, että toiminto tapahtuu mukauttimen kautta olevassa editorin pienoisversiossa. Hallinnointipaneelin kautta löytää kuitenkin ulkoasun alaisuudesta vimpaimille paremman editointinäkymän, jossa sosiaalisen median riviä voi muokata ongelmitta.

Kaiken työn aikana opitun ja aiemman tiedon perusteella modulaarisuudelle on olemassa useita eri nimityksiä, joita voidaan käyttää erilaisissa sisältöön, sisällön pilkkomiseen ja uudelleenkäytettävyyteen liittyvissä konteksteissa. Vaikka esimerkiksi aiemmin mainittiin World Wide Webin olevan modulaarinen, se ei tarkoita, että jokainen verkkosivusto olisi automaattisesti modulaarinen ja sisältäisi moduuleja. Verkkosivujen modulaarisuus mahdollistuu juurikin mm. työssä esiteltyjen modulaaristen teknologioiden avulla.

Erilaisten Reactin hookkien käyttö ja WordPressin omien komponenttien uudelleen hyödyntäminen moduuleja luodessa on myös modulaarisuuden periaatteiden mukaista: komponentteja tarjoavat, osista koostuvat tiedostot, joiden järjestystä voi vaihtaa ja jotka toimivat itsenäisinä kokonaisuuksina.

SCSS:n hyödyntäminen teemassa ja mukauttimessa tarjoaa joustavuutta ja vapautta loppukäyttäjälle, sillä loppukäyttäjä voi asettaa toivomansa värit sivustolle itse. Moduuleissa olisi hyvinkin voinut hyödyntää värien vaihtoa Gutenbergin kautta, mutta scssphp:n hyödyntäminen oli oikeastaan helpompi tapa tähän. Lisäksi mukauttimen kautta tapahtuva tyylittely näyttää suoraan sivustolle tapahtuvat tyylimuutokset esikatselun avulla, minkä johdosta koen tavan paremmaksi keinoksi toteuttaa mukautettavuutta. Kuitenkin lisäkehitys esimerkiksi RGBA:n sallimiseksi olisi ollut kiinnostavaa, mutta aika ei riittänyt siihen. RGBA olisi mahdollistanut läpinäkyvyyttä moduulien taustoihin. Työn aikana SCSS:n partiaalit osottautuivat tehokkaaksi tavaksi parantaa koodin selattavuutta ja nestingin hyödyntäminen lyhensi tarvittavien koodien kirjoittamista. Kyky hyödyntää SCSS:n erilaisia muuttujia vakioiduissa koodeissa tekee Sassista modulaarisen. Kehitettyjen moduulien vakiintunut ulkoasu auttaa sivustolle tulevaisuudessa syötettävän sisällön ulkoasua pysymään määritellyn tyylin mukaisena. Moduulien vakioihin (eli tyylittelyihin) sitoutuminen, järjestettävyys, säädettävyys ja uudelleenkäytettävyys tekevät niistä ainakin alkeellisesti modulaarisuuden määritystä vastaavia.

Moduuleja tulisi vielä jatkokehittää, jotta niiden käytettävyys ja visuaalinen täsmäävyys paranisi. Työtä varten moduulit ajavat kuitenkin asiansa, sillä

tarkoituksena oli ensisijaisesti oppia aiheesta lisää. Kävijän näkemällä sivulla moduulit ovat tyylittelyjensä puolesta kohtuullisen silmää miellyttäviä. Niiden avulla ymmärtää visuaalisesti ja tekstuaalisesti sen, mihin sisältö tulee syöttää. Sisältö myös pysyy vakioituna tyylittelyjen johdosta. Koska kehittäjien käytettävissä on useita erilaisia Reactin avulla luotuja komponentteja, tarjoaa Gutenbergin lohkokehitys potentiaalia todella teknisten ja modernien moduulien tuottamiseksi. Tämän johdosta ei olekaan ihme, että modulaarisuus on puheenaihe myös WordPressin parissa toimivien keskuudessa.

Kuten johdannossa todettiin, useat WordPressiin keskittyneet toimijat toteuttavat sivustoja hyödyntämällä sisällön syötössä ja esittämisessä itse kehitettyjä moduuleja Gutenbergille. Useiden työyhteisöjen suunnittelujärjestelmien sisältämien front-end-oppaiden avulla voidaan jakaa toteutuksessa ja moduuleissa hyödynnettäviä koodeja työyhteisön sisällä. Kokeneemmat kehittäjät voivat olla vastuussa kompleksisista kokonaisuuksista, joita muiden tulee osata lähinnä soveltaa. Oppaiden avulla koodeja voivat soveltaa myös sellaiset kehittäjät, joilla ei välttämättä ole tarpeeksi kokemusta luoda samaa koodia itse. Sama pätee tietenkin myös työyhteisöjen ulkopuolella. Soveltamalla oppii tehokkaasti, etenkin tarpeeksi monta kertaa koodia toistettaessa ja samalla pääsee myös itse osallistumaan kokonaisuuksien tuottamiseen. Lisäarvoa tuodakseni päädyin laittamaan moduuleja ja SCSS:n asettelua WordPressiin varten tuottamani koodin GitHubiin, jotta sitä voisi katsoa rauhassa itse kontekstissaan. Ainakin itselleni modulaarisiin teknologioihin ja lohkojen kehitykseen perehtyminen avasi WordPressin toimintaa enemmän kuin yksikään sivu, joita olen opintojeni aikana tehnyt. Tämän vuoksi toivon työstäni löytyvän hyödyllistä informaatiota aiheesta kiinnostuneille.

Lähteet

Alaa, Ali 2022. Gutenberg Blocks for React and WordPress Developers. Verkkoaineisto. Udemy.com. Esikatseltavissa osoitteessa: <https://www.udemy.com/course/gutenberg/>

Alli, Elizabeth & DesignerUp 2020. 10 Best Design Systems and How To Learn (and Steal) From Them. Designerup.com <https://designerup.co/blog/10-best-design-systems-and-how-to-learn-and-steal-from-them/> (Luettu 25.1.2022)

Arimetrics N.d. What is Development Environment - Definition and Examples. Arimetrics.com <https://www.arimetrics.com/en/digital-glossary/development-environment> (Luettu 14.2.2022)

A White Pixel 2020. How to Compile SCSS With PHP: Add Variables to WordPress Customizers for Compiling Theme CSS. Verkkoaineisto. awhitepixel.com <https://awhitepixel.com/blog/compile-scss-with-php-add-variables-to-wordpress-customizer/> (Luettu 30.2.2022)

Code With Random 2021. Library v/s Framework. Codewithrandom.com <https://www.codewithrandom.com/2021/10/JavaScript-Libraries-and-framework.html> (Luettu 19.4.2022)

Coyier, Chris 2013. A Complete Guide to Flexbox. Css-tricks.com <https://css-tricks.com/snippets/css/a-guide-to-flexbox/> (Luettu 7.2.2022)

Davis, Taurie & Vesselov, Sarrah 2019. Building Design Systems. Luvut 0–5. Yhdysvallat, Apress. Oreilly.com <https://learning.oreilly.com/library/view/building-design-systems/9781484245149/> (Luettu 5.2.2022)

Debian Manual N.d. What is a package manager? Debian.org <https://www.debian.org/doc/manuals/aptitude/pr01s02.en.html> (Luettu 15.1.2022)

dnd kit. dnd kit - a modern drag and drop toolkit for React. dndkit.com <https://dndkit.com/> (Luettu 22.4.2022)

Fessenden, Therese 2021. Design Systems 101. Nngroup.com <https://www.nngroup.com/articles/design-systems-101/> (Luettu 11.2.2022)

Freedom Scientific N.d. Web Page Elements. Freedomscientific.com <https://www.freedomscientific.com/SurfsUp/Elements.htm> (Luettu 13.2.2022)

Gasc, Marcos N.d. A Brief History on Modular Architecture. Qkvarchitects.com <https://www.gkvarchitects.com/news/a-brief-history-on-modular-architecture> (Luettu 1.2.2022)

Galvan, Monica 2021. Getting started with grids in digital design. Uxplanet.org <https://uxplanet.org/getting-started-with-grids-in-digital-design-7aa3bcc8c881> (Luettu 10.2.2022)

García, Maria 2014. Remembering designer Frank Ariss. Garciamedia.com <https://garciamedia.com/blog/remembering_designer_frank_ariss/> (Luettu 15.2.2022)

heynode N.d. What Is package.json? heynode.com <https://heynode.com/tutorial/what-packagejson/> (Luettu 10.2.2022)

Hughes, John 2022. An Introduction to WordPress Core Files. Themeisle.com <https://themeisle.com/blog/wordpress-core-files/> (Luettu 11.2.2022)

Javascript.info 2021. An Introduction to JavaScript. Javascript.info <https://javascript.info/intro> (Luettu 10.2.2022)

Johnson, Stephen 2021. Modular Construction: Using Lego-like blocks to build structures of the future. Bigthink.com <https://bigthink.com/the-future/modular-construction/> (Luettu 3.1.2022)

Joyce, Alita 2022. Inclusive Design. Nngroup.com <https://www.nngroup.com/articles/inclusive-design/> (Luettu 14.3.2022)

Kardys, Dennis 2014. Modular Web Design: Designing with components. wearediagram.com <https://www.wearediagram.com/blog/modular-web-design-designing-with-components> (Luettu 1.2.2022)

Kielitoimiston Sanakirja 2021. Standardi. Kielitoimistonsanakirja.fi <https://www.kielitoimistonsanakirja.fi/#/standardi> (Luettu 9.5.2022)

Kinsta 2021. What Is a WordPress Plugin? Kinsta.com <https://kinsta.com/knowledgebase/wordpress-plugin/> (Luettu 16.2.2022)

Kohan, Bernard 2010. What is a Content Management System (CMS)? Comentum.com <https://www.comentum.com/what-is-cms-content-management-system.html> (Luettu 10.2.2022)

Kowalski, Jenny 2021. Website Design 101: Essential Terms. Medium.com <https://medium.com/tylergaid/website-design-101-essential-terms-64bc4a52847f> (Luettu 4.2.2022)

Laaksonen, Antti 2011. PHP-ohjelmoinnin opas. Ohjelmointiputka.net <https://www.ohjelmointiputka.net/oppaat/sarja.php?tunnus=php> (Luettu 7.2.2022)

Laubheimer, Page 2016. Front-End Style-Guides: Definition, Requirements, Component Checklist. Nngroup.com <https://www.nngroup.com/articles/front-end-style-guides/> (Luettu 6.2.2022)

Leverenz, Andy 2014. How to Use Sass With WordPress - A Step by Step Guide. Elegantthemes.com <https://www.elegantthemes.com/blog/tips-tricks/how-to-use-sass-with-wordpress-a-step-by-step-guide> (Luettu 28.10.2021)

MacManus, Richard 2021. PHP Has Survived for 26 years Because it Keeps Evolving. Thenewstack.io <https://thenewstack.io/php-has-survived-for-26-years-because-it-keeps-evolving/> (Luettu 7.2.2022)

Manovich, Lev 2001. The Language of New Media. Sivu 31. Massachusetts Yhdysvallat, The MIT Press Cambridge. 1 painos. Luettavissa osoitteessa: <https://dss-edit.com/plu/Manovich-Lev_The_Language_of_the_New_Media.pdf> (Luettu 19.4.2022)

McBain, Graham 2021. Programming 101: How to use the terminal & command line. Galvanize.com <https://blog.galvanize.com/how-to-use-the-terminal-command-line/> (Luettu 8.2.2022)

McCollin, Rachel 2016. WordPress Actions and Filters: What's the Difference? Tutsplus.com <https://code.tutsplus.com/articles/wordpress-actions-and-filters-whats-the-difference--cms-25700> (Luettu 13.2.2022)

Merriam-Webster 2022 a. Modular Definition & Meaning. Merriam-webster.com <https://www.merriam-webster.com/dictionary/modular> (Luettu 29.11.2021)

Merriam-Webster 2022 b. Module Definition & Meaning. Merriam-webster.com <https://www.merriam-webster.com/dictionary/module> (Luettu 29.11.2021)

Moilanen, Teemu & Ojasalo, Katri & Ritalahti, Jarmo 2015. Kehittämistyön menetelmät. s. 65-68. Helsinki: Sanoma Pro Oy. 6. painos, 2020. E-kirja. (Luettu 13.2.2022)

Mozilla Developer Network 2022 a. Document and Website Structure. Mozilla.org <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure> (Luettu 5.2.2022)

Mozilla Developer Network 2022 b. Responsive Design. Mozilla.org <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design> (Luettu 5.2.2022)

Mozilla Developer Network 2022 c. Specificity. Mozilla.org <https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity> (Luettu 11.2.2022)

Myers, Erin 2022. Headless WordPress and Content Management Systems. Wpengine.com <https://wpengine.com/resources/headless-cms-and-wordpress/> (Luettu 23.3.2022)

Nielsen, Jakob 2004. The Need for Web Design Standards. Nngroup.com <https://www.nngroup.com/articles/the-need-for-web-design-standards/> (Luettu 5.2.2022)

Nielsen, Jakob 2012. Usability 101: Introduction to Usability. Nngroup.com <https://www.nngroup.com/articles/usability-101-introduction-to-usability/> (Luettu 21.3.2022)

NodeJS 2011. What is the file `package.json`. Nodejs.org <https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/> (Luettu 13.2.2022)

Oracle. Elements and Element Definitions. Oracle® Universal Content Management 2009 (10.1.4). Oracle.com <https://docs.oracle.com/cd/E10316_01/SiteStudio/10gr4/WebHelp-Designer/c02_websites012.htm> (Luettu 9.2.2022)

PHP Manual N.d. What is PHP? Php.net <https://www.php.net/manual/en/intro-whatis.php> (Luettu 7.2.2022)

Pönni, Antti 2021. Opinnäytetyö ja menetelmät. Google Slides -diat ovat nähtävissä osoitteessa: <https://docs.google.com/presentation/d/1pepftppMnCgnu7q33YSv_suHi5FjjMNXhcjhed2XyWY/edit#slide=id.p1> (Luettu 2.3.2022)

Ramalhete, Rui 2017. Does a Modular Design Approach Future-Proof your concept? Uxstudioteam.com <https://uxstudioteam.com/ux-blog/modular-design/> (Luettu 15.2.2022)

Rascia, Tania 2016. My Complete Front End Web Development Setup. taniarascia.com <https://www.taniarascia.com/my-front-end-web-development-setup/> (Luettu 15.2.2022)

React N.d. React Versions. Reactjs.org <https://reactjs.org/versions> (Luettu 28.4.2022)

Röksä, Jarmo 2018. EU:n saavutettavuusdirektiivi - mitä siitä pitäisi tietää? Humak.fi <https://www.humak.fi/blogit/eun-saavutettavuusdirektiivi-mita-siita-pitaisi-tietaa/> (Luettu 13.3.2022)

Sallasmaa, Petri & Salmela, Petri N.d. Petrit.net <https://petrit.net/www-ohjelmointi/javascript/window_ja_document/> (Luettu 2.5.2022)

Sass N.d. a. Sass: Syntax. Sass-lang.com <https://sass-lang.com/documentation/syntax> (Luettu 8.2.2022)

Sass N.d. b. Sass: Sass Basics. Sass-lang.com <https://sass-lang.com/guide> (Luettu 8.2.2022)

Sass N.d. c. Sass: Dart Sass. Sass-lang.com <https://sass-lang.com/dart-sass> (Luettu 8.2.2022)

scssphp 2022. SCSS Compiler in PHP - scssphp. Verkkoaineisto. Github.io <https://scssphp.github.io/scssphp/>

Stack Overflow N.d. Stack Overflow Developer Survey 2021. stackoverflow.com <https://insights.stackoverflow.com/survey/2021> (Luettu 8.2.2022)

Technopedia 2020. Runtime Environment (RTE). Technopedia.com <https://www.techopedia.com/definition/5466/runtime-environment-rte> (Luettu 10.2.2022)

Technopedia 2016. What is Development Environment. Technopedia.com <https://www.techopedia.com/definition/16376/development-environment> (Luettu 7.2.2022)

TechTerms 2011. Repository Definition. Techterms.com <https://techterms.com/definition/repository> (Luettu 9.5.2022)

Underscores N.d. Underscores | A Starter Theme for WordPress. Verkkoaineisto. underscores.me. <https://underscores.me/> (Luettu 3.5.2022)

UXPin N.d. Design Systems vs. Pattern Libraries vs. Style Guides - What's the Difference? Uxpin.com <https://www.uxpin.com/studio/blog/design-systems-vs-pattern-libraries-vs-style-guides-whats-difference/> (Luettu 6.2.2022)

W3C 2016. Web Design Standards: HTML & CSS. W3.org <https://www.w3.org/standards/webdesign/htmlcss> (Luettu 7.2.2022)

W3Schools N.d. a. CSS Syntax. W3schools.com <https://www.w3schools.com/css/css_syntax.ASP> (Luettu 5.2.2022)

W3Schools N.d. b. Responsive Web Design: Media Queries. W3schools.com <https://www.w3schools.com/css/css_rwd_mediaqueries.asp> (Luettu 8.2.2022)

W3Schools N.d. c. What is JSON? W3schools.com <https://www.w3schools.com/whatis/whatis_json.asp> (Luettu 10.2.2022)

W3Schools N.d. d. PHP MySQL Database. W3schools.com <https://www.w3schools.com/php/php_mysql_intro.asp> (Luettu 7.2.2022)

W3Schools N.d. e. PHP Functions. W3schools.com <https://www.w3schools.com/php/php_functions.asp> (Luettu 16.2.2022)

W3Schools N.d. f. PHP if…else…elseif Statements. W3schools.com <https://www.w3schools.com/php/php_if_else.asp> (Luettu 16.2.2022)

W3Schools N.d. g. Web APIs - Introduction. W3schools.com <https://www.w3schools.com/js/js_api_intro.asp> (Luettu 28.2.2022)

W3Schools N.d. h. HTML Semantic Elements. W3schools.com <https://www.w3schools.com/html/html5_semantic_elements.asp> (Luettu 8.3.2022)

W3Schools N.d. i. CSS !important property. W3schools.com <https://www.w3schools.com/css/css_important.asp> (Luettu 3.5.2022)

W3Techs 2022. Usage statistics and market share of WordPress. W3techs.com <https://w3techs.com/technologies/details/cm-wordpress> (Luettu 10.2.2022)

Wales, Michael 2020. 3 Web Dev Careers Decoded: Front-end vs Back-end vs Full stack. Udacity.com <https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-web-developers.html> (Luettu 7.3.2022)

Wiltz, Chris 2015. What Can Design Engineers Learn from Ikea? Designnews.com <https://www.designnews.com/design-hardware-software/what-can-design-engineers-learn-ikea> (Luettu 1.2.2022)

WordPress.com N.d. a. Customizer | WordPress.com Support. Wordpress.com <https://wordpress.com/support/customizer/> (Luettu 16.2.2022)

WordPress.com N.d. b. Post vs. Page | WordPress.com Support. Wordpress.com <https://wordpress.com/support/post-vs-page/> (Luettu 7.3.2022)

WordPress.org N.d. a. About Us: Our Mission. Wordpress.org <https://wordpress.org/about/> (Luettu 10.2.2022)

Wordpress.org N.d. b. Theme Handbook. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/> (Luettu 10.2.2022)

Wordpress.org N.d. c. What is a theme? WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/getting-started/what-is-a-theme/> (Luettu 10.2.2022)

Wordpress.org N.d. d. Template tags. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/basics/template-tags/> (Luettu 11.2.2022)

Wordpress.org N.d. e. Template Hierarchy. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/basics/template-hierarchy/> (Luettu 11.2.2022)

WordPress.org N.d. f. Template tags. WordPress Codex. Wordpress.org <https://codex.wordpress.org/Template_Tags> (Luettu 13.2.2022)

WordPress.org N.d. g. Page Templates. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/template-files-section/page-template-files/> (Luettu 13.2.2022)

WordPress.org N.d. h. Rest API Handbook. WordPress Developer Resources. Wordpress.org <https://developer.wordpress.org/rest-api/> (Luettu 20.2.2022)

WordPress.org N.d. i. Hooks. WordPress Plugin Developer Handbook. Wordpress.org <https://developer.wordpress.org/plugins/hooks/> (Luettu 16.2.2022)

WordPress.org N.d. j. get_header. WordPress Developer Resources. Wordpress.org <https://developer.wordpress.org/reference/hooks/get_header/> (Luettu 16.2.2022)

WordPress.org N.d. k. Function Reference. WordPress Codex. Wordpress.org. <https://codex.wordpress.org/Function_Reference> (Luettu 16.2.2022)

WordPress.org N.d. l. Theme Functions. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/basics/theme-functions/> (Luettu 16.2.2022)

WordPress.org N.d. m. Actions. Plugin Handbook. Wordpress.org <https://developer.wordpress.org/plugins/hooks/actions/> (Luettu 16.2.2022)

WordPress.org N.d. n. init. WordPress Developer Resource. Wordpress.org <https://developer.wordpress.org/reference/hooks/init/> (Luettu 16.2.2022)

WordPress.org N.d. o. register_nav_menus(). WordPress Developer Resource. Wordpress.org <https://developer.wordpress.org/reference/functions/register_nav_menus/> (Luettu 16.2.2022)

WordPress.org N.d. p. Metadata. WordPress Plugin Handbook. Wordpress.org <https://developer.wordpress.org/plugins/metadata/> (Luettu 18.2.2022)

WordPress.org N.d. q. Taxonomies. WordPress.org Support. Wordpress.org <https://wordpress.org/support/article/taxonomies/> (Luettu 2.3.2022)

WordPress.org N.d. r. Modularity. WordPress Block Editor Handbook. Wordpress.org <https://developer.wordpress.org/block-editor/explanations/architecture/modularity/> (Luettu 21.4.2022)

WordPress.org N.d. s. Block Theme Setup. WordPress Theme Handbook. Wordpress.org <https://developer.wordpress.org/themes/block-themes/block-theme-setup/> (Luettu 23.4.2022)

WordPress.org N.d. t. Metadata in block.json. WordPress Block Editor Handbook. Wordpress.org <https://developer.wordpress.org/block-editor/reference-guides/block-api/block-metadata/> (Luettu 3.5.2022)

WordPress.org N.d. u. Dashicons. WordPress Developer Resources. Verkkoaineisto. Wordpress.org <https://developer.wordpress.org/resource/dashicons/> (Luettu 3.5.2022)

WordPress.org N.d. v. Edit and Save. WordPress Block Editor Handbook. Wordpress.org <https://developer.wordpress.org/block-editor/reference-guides/block-api/block-edit-save/> (Luettu 4.5.2022)

WPBeginner Editorial Staff 2022. 6 Most Important Reasons to Use WordPress in 2022. Wpbeginnger.com <https://www.wpbeginner.com/why-you-should-use-wordpress/> (Luettu 10.2.2022.)

WPBeginner Editorial Staff N.d. a. What is Backend in Development? (Explained in Plain English). Wpbeginner.com <https://www.wpbeginner.com/glossary/backend/> (Luettu 11.2.2022)

WPBeginner Editorial Staff N.d. b. What is functions.php file in WordPress? Wpbeginner.com <https://www.wpbeginner.com/glossary/functions-php/> (Luettu 16.2.2022)

WPBeginner Editorial Staff N.d. c. Gutenberg vs. WordPress Page Builders - What's the Real Difference? Wpbeginner.com <https://www.wpbeginner.com/beginners-guide/gutenberg-vs-wordpress-page-builders/> (Luettu 19.2.2022)

WPBeginner Editorial Staff N.d. d. What is a Widget in WordPress? Wpbeginner.com <https://www.wpbeginner.com/glossary/widgets/> (Luettu 20.2.2022)

Young, Emma 2022. What is WAMP (Beginners Friendly Guide). Hostinger.com <https://www.hostinger.com/tutorials/what-is-wamp> (Luettu 15.2.2022)

Kuvien lähteet

Alaa, Ali 2022. Gutenbergin lohkojen toimintaperiaate. Udemy.com. Esikatseltavissa osoitteessa: <https://www.udemy.com/course/gutenberg/> (Kuvakaappaus otettu 13.4.2022)

Galvan, Monica 2021. Ruudukot. Medium.com <https://miro.medium.com/max/1400/1*XByO3GyouewolhL2gwDj6w.jpeg> (Ladattu 10.2.2022)

Ikea & UX Studio Team 2017. Kallax-hylly. Uxstudioteam.com <https://uxstudioteam.com/website/wp-content/uploads/2017/01/Ikea-kallax-modular-shelf-unit.jpg> (Ladattu 28.1.2022)

Muth, Theresa 2021. Kuva Lego-palikoista. Pixabay.com <https://pixabay.com/photos/toys-lego-bricks-lego-bricks-5993702/> (Ladattu 9.2.2022)

Sass N.d. b. Kuvakaappaus SCSS:n ja Sassin syntaksin eroista. Sass-lang.com <https://sass-lang.com/guide> (Kuvakaappaus otettu 13.2.2022)

W3Schools N.d. a. Kuvakaappaus CSS:n syntaksista. W3Schools.com <https://www.w3schools.com/css/img_selector.gif> (Kuvakaappaus otettu 16.2.2022)

WordPress.org N.d. e. Template Hierarchy. Wordpress.org <https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png> (Ladattu 14.2.2022)

GitHub

Opinnäytetyössä käytettävät koodit löytyvät osoitteesta:
<https://github.com/smiia/oppari>

Liitteet

RichTextin hyödyntäminen (pelkistetty)

a) **Attribuutit**
((registerBlockType(‘moduulit/kontenttimoduuli’, { … attributes: { contenttext: { type: 'string', source: 'html', selector: 'p', }, contenttitle: { type: 'string', source: 'html', selector: 'h2', }, … }

b) **Attribuuttien hakeminen tiedostoon**
function Edit({ attributes, setAttributes }) { const { contenttext, contenttitle, url, alt, id } = attributes; ...a)Attribuuttien muutosten rekisteröinti ja tallentaminen onChangen avullaconst onChangeContenttext = (newContenttext) => { setAttributes({contenttext: newContenttext}); }; const onChangeContenttitle = (newContenttitle) => { setAttributes({contenttitle: newContenttitle}); };

c) **Edit.js:n returnin sisältä löytyvä RichText -komponentti editoriin.**
<RichText ref={titleRef} tagName="h2" className="content-title" placeholder={ __('Kirjoita otsikko tähän', 'kontenttimoduuli') } value={ contenttitle } onChange={ onChangeContenttitle } /><RichText tagName="p" className="content-text" placeholder={ __('Kirjoita otsikko tähän', 'kontenttimoduuli') } value={ contenttext } onChange={ onChangeContenttext } />

d) **Save.js:n returnin sisältä löytyvä RichText -komponentti editoriin.**
{contenttitle && <RichText.Content tagName="h2" value={contenttitle} />}{contenttext && <RichText.Content tagName="p" value={contenttext} />}

Asennuskansion lohkojen rekisteröinnin koodit

Tiedostot löytyvät luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja sijaitsevat kansiossa nimeltä moduulit.

a) moduulit/moduuli.php

<?php/** * Plugin Name: Moduuli * Description: Moduuliwrapperi * Requires at least: 5.8 * Requires PHP: 7.0 * Version: 0.1.0 * Author: The WordPress Contributors * License: GPL-2.0-or-later * License URI: https://www.gnu.org/licenses/gpl-2.0.html * Text Domain: moduuli * * @package create-block *//** * Registers the block using the metadata loaded from the `block.json` file. * Behind the scenes, it registers also all assets so they can be enqueued * through the block editor in the corresponding context. * * @see https://developer.wordpress.org/reference/functions/register_block_type/ */function create_block_moduuli_block_init() {register_block_type(__DIR__ . '/build');}add_action('init', 'create_block_moduuli_block_init');

b) moduulit/package.json

{"name": "moduuli","version": "0.1.0","description": "Wrapper for your content","author": "Miia Lousaari","license": "GPL-2.0-or-later","main": "build/index.js","scripts": {"build": "wp-scripts build","format": "wp-scripts format && stylelint \"**/*.scss\" --fix","lint:css": "wp-scripts lint-style","lint:js": "wp-scripts lint-js","packages-update": "wp-scripts packages-update","plugin-zip": "wp-scripts plugin-zip","start": "wp-scripts start","prepare": "husky install"},"prettier": "@wordpress/prettier-config","stylelint": {"extends": "@wordpress/stylelint-config/scss"},"lint-staged": {"*.js": ["wp-scripts lint-js","wp-scripts format"],"*.scss": "npx stylelint --fix"},"dependencies": {"@dnd-kit/core": "^5.0.3","@dnd-kit/modifiers": "^5.0.0","@dnd-kit/sortable": "^6.0.1","@dnd-kit/utilities": "^3.1.0","@wordpress/blob": "^3.6.0","@wordpress/block-editor": "^8.5.1","@wordpress/blocks": "^11.5.1","@wordpress/components": "^19.8.0","@wordpress/compose": "^5.5.0","@wordpress/data": "^6.7.0","@wordpress/element": "^4.5.0","@wordpress/i18n": "^4.7.0"},"devDependencies": {"@wordpress/eslint-plugin": "^12.0.0","@wordpress/scripts": "^22.4.0","eslint-config-prettier": "^8.5.0","husky": "^7.0.0","stylelint": "^14.7.1"}}

Päämoduulin koodit

Tiedostot löytyvät luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja ne sijaitsevat kansiossa moduulit/src.

a) moduulit/src/edit.js

```
import { __ } from '@wordpress/i18n';import { PanelBody, RangeControl } from '@wordpress/components';import { useBlockProps, InnerBlocks, InspectorControls } from '@wordpress/block-editor';import './editor.scss';export default function Edit({attributes, setAttributes}) {const {columns} = attributes;const onChangecolumns = (newColumns) => {setAttributes({columns: newColumns});};return (<div {...useBlockProps({className: `has-${columns}-columns`,})}><InspectorControls><PanelBody><RangeControl label={__('Columns', 'moduuli') }min={ 1 }max={ 4 }onChange={ onChangecolumns }value={columns} /></PanelBody></InspectorControls><InnerBlocks allowedBlocks={[ 'moduulit/korttimoduuli', 'moduulit/kontenttimoduuli', 'moduulit/somerow' ]}orientation="horizontal" /></div>);}
```

b) moduulit/src/save.js

```
import { __ } from '@wordpress/i18n';import { useBlockProps, InnerBlocks } from '@wordpress/block-editor';export default function save({attributes}) {const {columns} = attributes;return (<div {...useBlockProps.save({className: `has-${columns}-columns`,}) }><InnerBlocks.Content /></div>);}
```

c) moduulit/src/index.js

```
import { registerBlockType } from '@wordpress/blocks';import './korttimoduuli';import './somerow';import './kontenttimoduuli';import './style.scss';import Edit from './edit';import save from './save';registerBlockType('moduulit/moduuli', {edit: Edit,save,});
```

d) moduulit/src/block.json

```
{"$schema": "https://schemas.wp.org/trunk/block.json","apiVersion": 2,"name": "moduulit/moduuli","version": "0.1.0","title": "Moduuli","category": "widgets","icon": "block-default","description": "Supermoduuli :o","supports": {"html": false,"align": ["wide", "full"]},"textdomain": "moduuli","editorScript": "file:./index.js","editorStyle": "file:./index.css","style": "file:./style-index.css","attributes": {"columns": {"type": "number","default": 1},"backgroundColor": {"type": "string"},"textColor": {"type": "string"}}}
```

Sisältömoduulin koodit

Tiedostot löytyvät luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja ne sijaitsevat kansiossa moduulit/src/kontenttimoduuli.

a) moduulit/src/kontenttimoduuli/edit.js

import { useBlockProps, RichText, MediaPlaceholder, BlockControls, MediaReplaceFlow, InspectorControls, store as blockEditorStore,} from '@wordpress/block-editor';import { __ } from '@wordpress/i18n';import { isBlobURL, revokeBlobURL } from '@wordpress/blob';import { useEffect, useState, useRef } from '@wordpress/element';import { usePrevious } from '@wordpress/compose';import { useSelect } from '@wordpress/data';import { Spinner, withNotices, ToolbarButton, PanelBody, TextareaControl, SelectControl } from '@wordpress/components';function Edit({ attributes, setAttributes, noticeOperations, noticeUI }) { const { contenttext, contenttitle, url, alt, id } = attributes; const titleRef = useRef(); const [blobURL, setBlobURL] = useState(); const previousURL = usePrevious(url); const imageObject = useSelect((select) => {const { getMedia } = select("core");return id ? getMedia(id) : null;},[id]); const onChangeContenttext = (newContenttext) => { setAttributes({contenttext: newContenttext}); }; const onChangeContenttitle = (newContenttitle) => { setAttributes({contenttitle: newContenttitle}); }; const onChangeImgSize = (newURL) => {setAttributes({ url: newURL });}; const imageSizes = useSelect((select) => {return select(blockEditorStore).getSettings().imageSizes;}, []); const getImgSizeOptions = () => {if (! imageObject) return [];const options = [];const sizes = imageObject.media_details.sizes;for (const key in sizes) {const size = sizes[key];const imageSize = imageSizes.find((s) => s.slug === key);if (imageSize) {options.push({label: imageSize.name,value: size.source_url,});}}return options;}; const onSelectImg = (image) => {if (! image || ! image.url) {setAttributes({ url: undefined, id: undefined, alt: '' });return;}setAttributes({ url: image.url, id: image.id, alt: image.alt });}; const onSelURL = (newURL) => { setAttributes({ url: newURL, id: undefined, alt: '', });}; const onUploadErr = (message) => { noticeOperations.removeAllNotices(); noticeOperations.createErrorNotice(message); }; const onChangeAltII = (newAltII) => { setAttributes({alt: newAltII}); }; const removeImg = () => { setAttributes({ url: undefined, alt: '', id: undefined, }) }; useEffect(() => {if (! id && isBlobURL(url)) {setAttributes({url: undefined,alt: '',});}}, []);useEffect(() => {if (isBlobURL(url)) {setBlobURL(url);} else {revokeBlobURL(blobURL);setBlobURL();}}, [url]); useEffect(() => { if(url && !previousURL) { titleRef.current.focus();} }, [url, previousURL]); return (<> <InspectorControls> <PanelBody title={__('Image Settings' ,'kontenttimoduuli')}> { id && (<SelectControllabel={ __('Image

```
Size', 'kontenttimoduuli' ) }options={ getImgSizeOptions() }value={ url }onChange={ onChangeImgSize }/>) } {url && !isBlobURL(url) && ( <TextareaControl label={__('Alt Text' ,'kontenttimoduuli')} value={alt} onChange={onChangeAltII} help={__("Alternative text describes your image for people who cannot see it. Add a short description with its key details.", 'kontenttimoduuli')} /> ) } </PanelBody> </InspectorControls> {url && ( <BlockControls group="inline"><MediaReplaceFlowname={ __( 'Change Picture', 'kontenttimoduuli' ) }onSelect={ onSelectImg }onSelURL={ onSelURL }onError={ onUploadErr }accept="image/*"allowedTypes={ [ 'image' ] }mediaId={ id }mediaURL={ url }/> <ToolbarButton onClick={ removeImg}>{__('Remove Image', 'kontenttimoduuli')}</ToolbarButton></BlockControls> )} <div { ...useBlockProps() }> { url && (<div className={ `wp-block-moduulit-kontenttimoduuli-kuva${ isBlobURL( url ) ? ' is-loading' : ''}` }><img src={ url } alt={ alt } />{ isBlobURL( url ) && <Spinner /> }</div>) } <MediaPlaceholder icon="format-image" onSelect={ onSelectImg } onSelURL={ onSelURL } onError={ onUploadErr } accept="image/*" allowedTypes={['image']} disableMediaButtons={url} notices={noticeUI} /><div className="kontenttiwrap"> <RichText ref={titleRef} tagName="h2" className="content-title"placeholder={ __( 'Kirjoita otsikko tähän', 'kontenttimoduuli') }value={ contenttitle }onChange={ onChangeContenttitle }/> <RichText className="content-text" tagName="p" placeholder={ __( 'Kirjoita teksti tähän', 'kontenttimoduuli') }value={ contenttext }onChange={ onChangeContenttext }/> </div></div></> );}export default withNotices( Edit );
```

b) **moduulit/src/kontenttimoduuli/save.js**

```
import { useBlockProps, RichText } from '@wordpress/block-editor';export default function Save({attributes}) { const { contenttext, contenttitle, url, alt, id } = attributes; return ( <div { ...useBlockProps.save() }> {url && (<div className="kuva-kontent"> <img src={url} alt={alt} className={ id ? `wp-image-${id}` : null } /></div>)} <div className="kontenttiwrap"> {contenttitle && <RichText.Content tagName="h2" value={contenttitle} />} {contenttext && <RichText.Content tagName="p" value={contenttext} />} </div> </div> );}
```

c) moduulit/src/kontenttimoduuli/index.js

```
import { registerBlockType } from '@wordpress/blocks';import { __ } from '@wordpress/i18n';import Edit from './edit';import Save from './save';registerBlockType('moduulit/kontenttimoduuli', {title: __('Kontenttimoduuli', 'moduulit'), description: __('Kontenttimoduulin osa', 'moduulit'), icon: 'id-alt', parent: ['moduulit/moduuli'], supports: { html: false, align: true, }, attributes: { contenttext: { type: 'string', source: 'html', selector: 'p', }, contenttitle: { type: 'string', source: 'html', selector: 'h2', }, id: { type: 'number',
```

```
}, alt: { type: 'string', source: 'attribute', selector: 'img', attribute: 'alt', de-
fault: '', }, url: { type: 'string', source: 'attribute', selector: 'img', attribute:
'src', }, }, edit: Edit,save: Save,});
```

Korttimoduulin koodit

Tiedostot löytyvät luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja ne sijaitsevat kansiossa moduulit/src/korttimoduuli.

a) moduulit/src/korttimoduuli/edit.js

```
import { useBlockProps, RichText, MediaPlaceholder, BlockControls, MediaReplaceFlow, InspectorControls, store as blockEditorStore,} from '@wordpress/block-editor';import { __ } from '@wordpress/i18n';import { isBlobURL, revokeBlobURL } from '@wordpress/blob';import { useEffect, useState, useRef } from '@wordpress/element';import { usePrevious } from '@wordpress/compose';import { Spinner, withNotices, ToolbarButton, PanelBody, TextareaControl } from '@wordpress/components'; function Edit({attributes, setAttributes, noticeOperations, noticeUI }) { const {titlec, textc, url, alt, id, textb } = attributes; const [blobURL, setBlobURL] = useState(); const titleRef = useRef(); const prevURL = usePrevious(url); const onChangeTitlec = (newTitlec) => { setAttributes({titlec: newTitlec}); }; const onChangeTextc = (newTextc) => { setAttributes({textc: newTextc}); }; const onChangeTextb = (newTextb) => { setAttributes({textb: newTextb}); }; const onSelectImage = ( image ) => {if ( ! image || ! image.url ) {setAttributes( { url: undefined, id: undefined, alt: '' } );return;}setAttributes( { url: image.url, id: image.id, alt: image.alt } );}; const onSelectURL = (newURL) => { setAttributes({ url: newURL, id: undefined, alt: '', });}; const onUploadError = (message) => { noticeOperations.removeAllNotices(); noticeOperations.createErrorNotice(message); }; const onChangeAlt = (newAlt) => { setAttributes({alt: newAlt}); }; const removeImage = () => { setAttributes({ url: undefined, alt: '', id: undefined, }) }; useEffect( () => {if ( ! id && isBlobURL( url ) ) {setAttributes( {url: undefined,alt: '',} );}}, [] );useEffect( () => {if ( isBlobURL( url ) ) {setBlobURL( url );} else {revokeBlobURL( blobURL );setBlobURL();}}, [ url ] ); useEffect(() => { if(url && !prevURL) { titleRef.current.focus();} }, [url, prevURL] ); return ( <> <InspectorControls> <PanelBody title={__('Image Settings' ,'korttimoduuli')}> {url && !isBlobURL(url) && ( <TextareaControl label={__('Alt Text' ,'korttimoduuli')} value={alt} onChange={onChangeAlt} help={__("Alternative text describes your image for people who cannot see it. Add a short description with its key details.", 'korttimoduuli')} /> ) } </PanelBody> </InspectorControls> {url && ( <BlockControls group="inline"><MediaReplaceFlowname={ __( 'Change Picture', 'korttimoduuli' ) }onSelect={ onSelectImage }onSelectURL={ onSelectURL }onError={ onUploadError }accept="image/*"allowedTypes={ [ 'image' ] }mediaId={ id }mediaURL={ url }/> <ToolbarButton onClick={ removeImage}>{__('Remove Image', 'korttimoduuli')}</ToolbarButton></BlockControls> )} <div { ...useBlockProps() }> { url && (<div className={ `wp-block-moduulit-korttimoduuli-kuva${ isBlobURL( url ) ? ' is-loading' : ''}` }><img src={ url } alt={ alt } />{
```

isBlobURL(url) && <Spinner /> }) } <MediaPlaceholder icon="format-image" onSelect={ onSelectImage } onSelectURL={ onSelectURL } onError={ onUploadError } accept="image/*" allowedTypes={['image']} disableMediaButtons={url} notices={noticeUI} />   <RichText ref={titleRef} placeholder={ __('Kortin otsikko', 'korttimoduuli') } tagName="h5" onChange={onChangeTitlec} value={ titlec } /> <RichText placeholder={ __('Kortin sisältöteksti', 'korttimoduuli') } tagName="p" onChange={onChangeTextc} value={ textc } /> <RichText className="cardb" tagName="div"aria-label={ __('Napin teksti') }placeholder={ __('Napin teksti', 'korttimoduuli') }value={ textb }onChange={ onChangeTextb }/> </>);}export default withNotices(Edit);

b) moduulit/src/korttimoduuli/save.js

import { useBlockProps, RichText } from '@wordpress/block-editor';export default function Save({attributes}) { const { titlec, textb, textc, url, alt, id } = attributes; return (<div { ...useBlockProps.save() }> {url && ( )}   {titlec && <RichText.Content tagName="h5" value={titlec} />} {textc && <RichText.Content tagName="p" value={textc} />} {textb && <RichText.Content tagName="div" className="cardb" value={textb} />});}

c) moduulit/src/korttimoduuli/index.js

import { registerBlockType } from '@wordpress/blocks';import { __ } from '@wordpress/i18n';import Edit from './edit';import Save from './save';registerBlockType('moduulit/korttimoduuli', {title: __('Korttimoduuli', 'moduulit'), description: __('Korttimoduulin osa', 'moduulit'), icon: 'id-alt', parent: ['moduulit/moduuli'], supports: { reusable: false, html: false, }, attributes: { titlec: { type: 'string', source: 'html', selector: 'h5', }, textc: { type: 'string', source: 'html', selector: 'p', }, textb: { type: 'string', source: 'html', selector: 'p', }, textbu: { type: 'string', source: 'html', selector: 'p', }, id: { type: 'number', }, alt: { type: 'string', source: 'attribute', selector: 'img', attribute: 'alt', default: '', }, url: { type: 'string', source: 'attribute', selector: 'img', attribute: 'src', }, }, edit: Edit,save: Save,});

Sosiaalisen median rivin koodit

Tiedostot löytyvät luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja sijaitsevat kansiossa moduulit/src/somerow. Riviä varten tarvitsee dnd kitin, joka tulee erikseen ladata ja asentaa, jotta sen komponentteja voidaan käyttää vedä ja pudota -toiminnon mahdollistamiseksi.

a) moduulit/src/somerow/edit.js

import { __ } from '@wordpress/i18n';import { Icon, Tooltip , TextControl, Button} from '@wordpress/components';import { useBlockProps, RichText } from '@wordpress/block-editor';import { useEffect, useState, useRef } from '@wordpress/element';import { usePrevious } from '@wordpress/compose';import { DndContext, useSensor, useSensors, PointerSensor } from '@dnd-kit/core';import { SortableContext, horizontalListSortingStrategy, arrayMove } from '@dnd-kit/sortable';import { restrictToHorizontalAxis } from '@dnd-kit/modifiers';import SortableItem from './sortable-item';export default function Edit({ attributes, setAttributes, isSelected, }) { const { sometext, url, socialLinks } = attributes; const [selectedLink, setSelectedLink] = useState(); const titleRef = useRef(); const onChangeSomeText = (newSomeText) => { setAttributes({sometext: newSomeText}); }; const sensors = useSensors(useSensor(PointerSensor, {activationConstraint: { distance: 5 },})); const prevURL = usePrevious(url);const prevIsSelected = usePrevious(isSelected); const addNewSomeItem = () => {setAttributes({socialLinks: [...socialLinks, { icon: 'wordpress', link: '' }],});setSelectedLink(socialLinks.length);}; const updateSocialItem = (type, value) => {const socialLinksCopy = [...socialLinks];socialLinksCopy[selectedLink][type] = value;setAttributes({ socialLinks: socialLinksCopy });};const removeSocialItem = () => {setAttributes({socialLinks: [...socialLinks.slice(0, selectedLink),...socialLinks.slice(selectedLink + 1),],});setSelectedLink();}; const handleDragEnd = (event) => {const { active, over } = event;if (active && over && active.id !== over.id) {const oldIndex = socialLinks.findIndex((i) => active.id === `${ i.icon }-${ i.link }`);const newIndex = socialLinks.findIndex((i) => over.id === `${ i.icon }-${ i.link }`);setAttributes({socialLinks: arrayMove(socialLinks, oldIndex, newIndex),});setSelectedLink(newIndex);}}; useEffect(() => {if (url && ! prevURL) {titleRef.current.focus();}}, [url, prevURL]);useEffect(() => {if (prevIsSelected && ! isSelected) {setSelectedLink();}}, [isSelected, prevIsSelected]); return (<div { ...useBlockProps() }> <RichText placeholder={ __('Social Media', 'somerow') } tagName="h4" onChange={onChangeSomeText} value={ sometext } allowedFormats={[]}/> <DndContextsensors={ sensors }onDragEnd={ handleDragEnd } modifiers={ [restrictToHorizontalAxis]

}><SortableContextitems={ socialLinks.map((item) => `${ item.icon }-${ item.link }`) }strategy={ horizontalListSortingStrategy }>{ socialLinks.map((item, index) => {return (<SortableItemkey={ `${ item.icon }-${ item.link }` }id={ `${ item.icon }-${ item.link }` }index={ index }selectedLink={ selectedLink }setSelectedLink={ setSelectedLink }icon={ item.icon }/>);}) }</SortableContext></DndContext> { isSelected && (<li className="wp-block-moduuli-somerow-add-more"> <Tooltip text={__("Add more links", 'somerow')}> <button aria-label={__("Add more links", 'somerow')} onClick={addNewSomeItem}> <Icon icon="plus" /> </button> </Tooltip>)} {selectedLink !== undefined && (<TextControl label={__('Icon', 'text-somerow')} value={socialLinks[selectedLink].icon } onChange={(icon) => { updateSocialItem('icon', icon); } } /> <TextControl label={__('URL', 'text-somerow')} value={socialLinks[selectedLink].link} onChange={(link) => { updateSocialItem('link', link); }} />
 <Button isDestructive onClick={removeSocialItem}> { __('Remove Link', 'text-somerow')} </Button>)});}

b) moduulit/src/somerow/save.js

import { useBlockProps, RichText } from '@wordpress/block-editor';import { __ } from '@wordpress/i18n';import { Icon } from '@wordpress/components';export default function Save({attributes}) { const { sometext, socialLinks } = attributes; return (<div { ...useBlockProps.save() }> { socialLinks.length > 0 && ({sometext && <RichText.Content tagName="h4" value={sometext} />} { socialLinks.map((item, index) => {return (<li key={ index } data-icon={ item.icon }><Icon icon={ item.icon } />);}) }) });}index.js

c) moduulit/src/somerow/index.js

import { registerBlockType } from '@wordpress/blocks';import { __ } from '@wordpress/i18n';import Edit from './edit';import Save from './save';registerBlockType('moduulit/somerow', {title: __('Somerow', 'moduulit'), description: __('Somerow', 'moduulit'), icon: 'admin-site-alt2', parent: ['moduulit/moduuli'], supports: { reusable: true, html: false, align: false, }, attributes: { sometext: { type: 'string', source: 'html', selector: 'h4', }, id: { type: 'number', }, alt: { type: 'string', source: 'attribute', selector: 'img', attribute: 'alt', default: '', }, url: { type: 'string', source: 'attribute', selector: 'img', attribute: 'src', }, socialLinks: {type: 'array',default: [{ link: 'https://twitter.com', icon: 'twitter' },{ link: 'https://instagram.com', icon: 'instagram' },],source: 'query',selector: '.wp-block-moduuli-somerow ul li',query: {icon: {source: 'attribute',attribute: 'data-icon',},link: {source: 'attribute',selector: 'a',attribute: 'href',}, }, }, }, edit: Edit,save: Save,});

d) moduulit/src/somerow/sortable-item.js

```
import { useSortable } from '@dnd-kit/sortable';import { CSS } from
'@dnd-kit/utilities';import { Icon } from '@wordpress/components';import {
__ } from '@wordpress/i18n';export default function SortableItem( props )
{const {attributes,listeners,setNodeRef,transform,transition,} = useSorta-
ble( { id: props.id } );const style = {transform: CSS.Transform.toString(
transform ),transition,};return (<liref={ setNodeRef }style={ style }{ ...at-
tributes }{ ...listeners }className={props.selectedLink === props.index ?
'is-selected' : null}><buttonaria-label={ __( 'Linkki & teksti tähän', 'somer-
ow' ) }onClick={ () => props.setSelectedLink( props.index ) }><Icon
icon={ props.icon } /></button></li>);}
```

Teeman functions.php-tiedosto

Tiedosto löytyy luettavammassa muodossa GitHubista <https://github.com/smiia/oppari> ja se sijaitsee kansiossa nimeltä teema.

<?php/** * oppari functions and definitions * * @link https://developer.wordpress.org/themes/basics/theme-functions/ * * @package oppari */if (! defined('_S_VERSION')) {define('_S_VERSION', '1.0.0');}if (! function_exists('oppari_setup')) :function oppari_setup() {load_theme_textdomain('oppari', get_template_directory() . '/languages');add_theme_support('automatic-feed-links');add_theme_support('title-tag');add_theme_support('responsive-embeds');add_theme_support('align-wide');add_theme_support('editor-styles');add_editor_style('style-editor.css');add_theme_support('post-thumbnails');register_nav_menus(array('menu-1' => esc_html__('Primary', 'oppari'),));add_theme_support('html5',array('search-form','comment-form','comment-list','gallery','caption','style','script',));function oppari_load_dashicons() {wp_enqueue_style('dashicons');}add_action('wp_enqueue_scripts', 'oppari_load_dashicons');add_theme_support ('custom-spacing');add_theme_support ('custom-units', 'px', 'rem', 'em');add_theme_support('editor-color-palette', array(array('name' => esc_attr__('Martinique','themelangDomain'),'slug' => 'martinique',"color" => '#383053',),array('name' => esc_attr__('Martinique','themelangDomain'),'slug' => 'gablegreen',"color" => '#10292B',),array('name' => esc_attr__('Eastern Blue','themelangDomain'),'slug' => 'easternblue',"color" => '#28ACAD',),array('name' => esc_attr__('Old Rose','themelangDomain'),'slug' => 'oldrose',"color" => '#B87F72',),array('name' => esc_attr__('Young Rose','themelangDomain'),'slug' => 'youngrose',"color" => '#13434a',),array('name' => esc_attr__('Paradiso','themelangDomain'),'slug' => 'paradiso',"color" => '#328787',),array('name' => esc_attr__('Eggplant','themelangDomain'),'slug' => 'eggplant',"color" => '#6c446c',),array('name' => esc_attr__('Hopbush','themelangDomain'),'slug' => 'hopbush',"color" => '#c66299',),array('name' => esc_attr__('Melanie','themelangDomain'),'slug' => 'melanie',"color" => '#e1b1c7',),));add_theme_support('editor-gradient-presets',array(array('name' => esc_attr__('Martinique to Eggplant','themeLangDomain'),'gradient' => 'linear-gradient(135deg,#383053 0%, #6c445c 100%)','slug' => 'martinique-to-eggplant')));add_theme_support('customize-selective-refresh-widgets');add_theme_support('custom-logo',array('height' => 250,'width' => 250,'flex-width' => true,'flex-height' => true,));}endif;add_action('after_setup_theme', 'oppari_setup');function oppari_content_width() {$GLOBALS['content_width'] = apply_filters('oppari_content_width', 670);}add_action('after_setup_theme', 'oppari_content_width', 0);function oppari_widgets_init() {register_sidebar(array('name' => esc_html__('Sidebar', 'oppari'),'id' => 'sidebar-1','description' => esc_html__('Add widgets here.', 'oppari'),'before_widget' => '<section id="%1$s" class="widget %2$s">','after_widget' => '</section>','before_title' => '<h2 class="widget-title">','after_title' => '</h2>',));register_sidebar(array('name' => esc_html__('Footer column one', 'oppari'),'id' => 'footer_col_1','description' => esc_html__('Add widgets here.', 'oppari'),'before_widget' => '<section id="footercol" class="footer-col-

```
1">','after_widget' => '</section>','before_title' => '<h4 class="footer-title">','af-
ter_title' => '</h4>',));register_sidebar(array('name' => esc_html__( 'Footer col-
umn two', 'oppari' ),'id' => 'footer_col_2','description' => esc_html__( 'Add widg-
ets here.', 'oppari' ),'before_widget' => '<section id="footercol" class="footer-col-
2">','after_widget' => '</section>','before_title' => '<h4 class="footer-title">','af-
ter_title' => '</h4>',));register_sidebar(array('name' => esc_html__( 'Footer col-
umn three', 'oppari' ),'id' => 'footer_col_3','description' => esc_html__( 'Add
widgets here.', 'oppari' ),'before_widget' => '<section id="footercol"
class="footer-col-3">','after_widget' => '</section>','before_title' => '<h4
class="footer-title">','after_title' => '</h4>',));}add_action( 'widgets_init', 'op-
pari_widgets_init' );function oppari_scripts() {wp_enqueue_style( 'oppari-style',
get_stylesheet_uri(), array(), _S_VERSION );wp_style_add_data( 'oppari-style',
'rtl', 'replace' ); wp_enqueue_script( 'custom', get_template_directory_uri() .
'/js/custom.js', array(), _S_VERSION, true );wp_enqueue_script( 'oppari-naviga-
tion', get_template_directory_uri() . '/js/navigation.js', array(), _S_VERSION,
true );if ( is_singular() && comments_open() && get_option( 'thread_comments'
) ) {wp_enqueue_script( 'comment-reply' );}}add_action( 'wp_enqueue_scripts',
'oppari_scripts' );require get_template_directory() . '/inc/custom-header.php';re-
quire get_template_directory() . '/inc/template-tags.php';require get_tem-
plate_directory() . '/inc/template-functions.php';require get_template_directory()
. '/inc/customizer.php';if ( defined( 'JETPACK__VERSION' ) ) {require get_tem-
plate_directory() . '/inc/jetpack.php';}register_default_headers( array( 'default-
image' => array( 'url' => get_stylesheet_directory_uri() . '/assets/img/default-
header.jpg', 'thumbnail_url' => get_stylesheet_directory_uri() . '/assets/img/de-
fault-header.jpg', 'description' => __( 'Default Header Image', 'textdomain' ) ),)
);require_once(get_stylesheet_directory() . '/inc/scssphp/scss.inc.php');if
(is_customize_preview()) {add_action('wp_head', function() {$compiler = new
ScssPhp\ScssPhp\Compiler();$source_scss = get_stylesheet_directory() . '/as-
sets/scss/style.scss';$scssContents = file_get_contents($source_scss);$im-
port_path = get_stylesheet_directory() . '/assets/scss';$compiler->addIm-
portPath($import_path);$variables = ['$main' => get_theme_mod('main',
'#328787'),'$header' => get_theme_mod('header', '#10292b'),'$bg' =>
get_theme_mod('bg', '#1a3334'),'$headers' => get_theme_mod('headers',
'#fff'),'$cards' => get_theme_mod('cards', '#328787'),'$content' =>
get_theme_mod('content', '#13434a'),'$footer-c' => get_theme_mod('footer-c',
'#383053'),'$footer-t' => get_theme_mod('footer-t', '#fff'),'$secondary' =>
get_theme_mod('secondary', '#28acad'),'$color-tertiary' =>
get_theme_mod('color-tertiary', '#b1e1df'),'$color-fourth' =>
get_theme_mod('color-fourth', '#1a3334'),'$color-ctext' =>
get_theme_mod('color-ctext', '#fff'),'$link-b' => get_theme_mod('link-b',
'#24c0c0'),'$theme-text-size' => get_theme_mod('theme-text-size', '16') .
'px','$theme-border-radius' => get_theme_mod('theme-border-radius', '10') .
'px',];$compiler->setVariables($variables);$css = $compiler->com-
pile($scssContents);if (!empty($css) && is_string($css)) {echo '<style
type="text/css">' . $css . '</style>';}});}add_action('customize_register', func-
tion($wp_customize) {$wp_customize->add_section('theme-fonts', ['title' =>
__('Tekstien ja painikkeiden värit', 'txtdomain'),'priority' => 30]);$wp_customize-
```

```
>add_setting('headers', ['default' => '#fff']);$wp_customize->add_control(new
WP_Customize_Color_Control($wp_customize, 'headers', ['section' => 'theme-
fonts','label' => __('Yläotsakkeen tekstin väri', 'txtdomain'),'priority' =>
2]));$wp_customize->add_setting('footer-t', ['default' => '#fff']);$wp_customize-
>add_control(new WP_Customize_Color_Control($wp_customize, 'footer-t',
['section' => 'theme-fonts','label' => __('Alaotsakkeen tekstin väri', 'txtdo-
main'),'priority' => 54]));$wp_customize->add_setting('color-fourth', ['default' =>
'#1a3334']);$wp_customize->add_control(new WP_Customize_Color_Con-
trol($wp_customize, 'color-fourth', ['section' => 'theme-fonts','label' => __('Napit',
'txtdomain'),'priority' => 40]));$wp_customize->add_setting('color-ctext', ['default'
=> '#fff']);$wp_customize->add_control(new WP_Customize_Color_Con-
trol($wp_customize, 'color-ctext', ['section' => 'theme-fonts','label' => __('Sisältö-
tekstin väri', 'txtdomain'),'priority' => 38]));$wp_customize->add_setting('link-b',
['default' => '#24c0c0']);$wp_customize->add_control(new WP_Custom-
ize_Color_Control($wp_customize, 'link-b', ['section' => 'theme-fonts','label' =>
__('Linkkien väri', 'txtdomain'),'priority' => 50]));$wp_customize->add_set-
ting('theme-text-size', ['default' => '16']);$wp_customize->add_control('theme-
text-size', ['section' => 'theme-fonts','label' => __('Fontin koko', 'txtdomain'),'type'
=> 'number','priority' => 60,'input_attrs' => ['min' => 8, 'max' => 20, 'step' =>
1]]);$wp_customize->add_setting('theme-border-radius', ['default' =>
'0']);$wp_customize->add_control('theme-border-radius', ['section' => 'theme-
fonts','label' => __('Reunojen pyöreys', 'txtdomain'),'type' => 'number','priority'
=> 60,'input_attrs' => ['min' => 0, 'max' => 30, 'step' => 1]]);});add_action('cus-
tomize_register', function($wp_customize) {$wp_customize->add_sec-
tion('theme-variables', ['title' => __('Lohkojen ja otsakkeiden värit', 'txtdo-
main'),'priority' => 20]);$wp_customize->add_setting('cards', ['default' =>
'#328787']);$wp_customize->add_control(new WP_Customize_Color_Con-
trol($wp_customize, 'cards', ['section' => 'theme-variables','label' => __('Korttien
taustan väri', 'txtdomain'),'priority' => 3]));$wp_customize->add_setting('content',
['default' => '#13434a']);$wp_customize->add_control(new WP_Custom-
ize_Color_Control($wp_customize, 'content', ['section' => 'theme-variables','la-
bel' => __('Sisältömoduulin taustan väri', 'txtdomain'),'priority' => 4]));$wp_cus-
tomize->add_setting('header', ['default' => '#10292B']);$wp_customize-
>add_control(new WP_Customize_Color_Control($wp_customize, 'header',
['section' => 'theme-variables','label' => __('Teeman yläotsakkeen väri', 'txtdo-
main'),'priority' => 8]));$wp_customize->add_setting('main', ['default' =>
'#328787']);$wp_customize->add_control(new WP_Customize_Color_Con-
trol($wp_customize, 'main', ['section' => 'theme-variables','label' => __('Teeman
pääväri', 'txtdomain'),'priority' => 10]));$wp_customize->add_setting('bg', ['de-
fault' => '#1a3334']);$wp_customize->add_control(new WP_Custom-
ize_Color_Control($wp_customize, 'bg', ['section' => 'theme-variables','label' =>
__('Taustan väri', 'txtdomain'),'priority' => 1]));$wp_customize->add_set-
ting('secondary', ['default' => '#28ACAD']);$wp_customize->add_control(new
WP_Customize_Color_Control($wp_customize, 'secondary', ['section' =>
'theme-variables','label' => __('Teeman toinen pääväri', 'txtdomain'),'priority' =>
20]));$wp_customize->add_setting('color-tertiary', ['default' =>
'#b1e1df']);$wp_customize->add_control(new
```

```
WP_Customize_Color_Control($wp_customize, 'color-tertiary', ['section' =>
'theme-variables','label' => __('Korostusväri', 'txtdomain'),'priority' =>
30]));$wp_customize->add_setting('footer-c', ['default' => '#10292B']);$wp_cus-
tomize->add_control(new WP_Customize_Color_Control($wp_customize,
'footer-c', ['section' => 'theme-variables','label' => __('Alaotsakkeen taustan väri',
'txtdomain'),'priority' => 55]));});add_action('customize_save_after', function()
{$compiler = new ScssPhp\ScssPhp\Compiler();$source_scss =
get_stylesheet_directory() . '/assets/scss/style.scss';$scssContents =
file_get_contents($source_scss);$import_path = get_stylesheet_directory() .
'/assets/scss';$compiler->addImportPath($import_path);$target_css =
get_stylesheet_directory() . '/style.css';$variables = ['$main' =>
get_theme_mod('main', '#328787'),'$header' => get_theme_mod('header',
'#10292b'),'$bg' => get_theme_mod('bg', '#1a3334'),'$headers' =>
get_theme_mod('headers', '#fff'),'$cards' => get_theme_mod('cards',
'#328787'),'$content' => get_theme_mod('content', '#13434a'),'$footer-c' =>
get_theme_mod('footer-c', '#383053'),'$footer-t' => get_theme_mod('footer-t',
'#fff'),'$secondary' => get_theme_mod('secondary', '#28acad'),'$color-tertiary'
=> get_theme_mod('color-tertiary', '#b1e1df'),'$color-fourth' =>
get_theme_mod('color-fourth', '#1a3334'),'$color-ctext' =>
get_theme_mod('color-ctext', '#fff'),'$link-b' => get_theme_mod('link-b',
'#24c0c0'),'$theme-text-size' => get_theme_mod('theme-text-size', '16') .
'px','$theme-border-radius' => get_theme_mod('theme-border-radius', '10') .
'px',];$compiler->setVariables($variables);$css = $compiler->com-
pile($scssContents);if (!empty($css) && is_string($css)) {file_put_contents($tar-
get_css, $css);}});
```