Jere Huttunen

Gatsbyn hyödyntäminen staattisessa verkkotuotannossa

Metropolia Ammattikorkeakoulu

Medianomi

Viestinnän tutkinto-ohjelma

Opinnäytetyö

9.5.2021

Tiivistelmä

Tekijä:	Jere Huttunen
Otsikko:	Gatsbyn hyödyntäminen staattisessa verkkotuotannossa
Sivumäärä:	45 sivua
Aika:	9.5.2021

Tutkinto:	Medianomi
Tutkinto-ohjelma:	Viestinnän tutkinto-ohjelma
Suuntautumisvaihtoehto:	Digitaalinen viestintä
Ohjaaja:	Markus Norrena, Lehtori

Työn tarkoitus on tarkastella ohjelmistokehys Gatsbyn roolia staattisessa verkkotuotannossa. Työn keskeisiä kysymyksiä ovat: mikä Gatsby on, miksi sitä kannattaa käyttää ja milloin sitä ei kannata käyttää?

Tässä työssä on kaksi osiota: teoreettinen osio, joka esittelee ja tuo esille hyödyt ja haitat Gatsbysta ja staattisesta verkkotuotannosta, sekä toiminnallinen osio, joka vie teorian käytäntöön tuottamalla Gatsbylla luodun verkkosivun, ja sitä kautta tuoden Gatsbyn oikeat hyödyt ja haasteet esille.

Työn tietopohja perustuu kirjallisuuteen, verkon lähteisiin (kuten blogit ja podcastit), tekijän havaintoihin olemalla osana Gatsbyn yhteisöä ja hänen kokemuksiin käytettyään Gatsbya viimeiset 11 kuukautta.

Työn tulokset tuovat esille sen, että Gatsby on parhaimmillaan hyödynnettynä verkkosivustojen ja -sovellusten luomiseen silloin, kun sivustoa ei tarvitse päivittää usein ja sivuston nopeus ja hakukoneoptimointi ovat prioriteetteja. Sopivia sivustoja Gatsbylle ovat usein blogit, uutissivut, yrityssivut ja landing-sivut. Gatsbyn suuria hyötyjä ovat sen pluginien ekosysteemi, valmiit teemat, hyvä kehittäjäkokemus, valmiiksi asennettu GraphQL, React ja staattinen HTML-koodin generoiminen. Gatsby ei sovi sellaiselle sivustolle, joka päivittyy usein ja toimittaa yksittäisille kävijöilleen uniikkia sisältöä.

Toiminnallinen osio loi nopean, hakukoneystävällisen verkkosivun, jota on helppo päivittää uusilla artikkeleilla yhdistetyn sisällönhallintajärjestelmän kautta.

Avainsanat: Gatsby, staattinen verkkotuotanto, staattinen verkkosivu

Abstract

Author:	Jere Huttunen
Title:	Utilizing Gatsby in Static Web Development
Number of Pages:	45 pages
Date:	9 May 2021
Degree:	Bachelor of Culture and Arts
Degree Programme:	Media
Specialisation option:	Digital Media
Instructor:	Markus Norrena, Senior Lecturer

The purpose of this study is to dive into the role of Gatsby framework in static web development. This study investigates what is Gatsby, why should it be used and when should it not be used.

This study has two main sections: the first section is theoretical and focuses on explaining the concepts of static websites and Gatsby, whereas the second section is a practice-based section where the theories are turned into an actual website and thus shows Gatsby's true benefits and challenges.

This study is based on literature, online sources such as blogs and podcasts, the author's own observations as part of the Gatsby community and his experiences after using Gatsby for 11 months.

The results of this work show that Gatsby is best used to create websites and web apps that do not update very often and that need high performance and search engine optimization. Websites of this type are commonly article sites, blogs, company pages and landing pages. The greatest benefits of Gatsby are its plugin ecosystem, premade themes, good developer experience, pre-installed GraphQL, React and static HTML generation. Gatsby does not fit well for pages that update often and provide its individual users unique content.

The practice-based section of this work created a fast performing, search engine friendly website that is easy to update with new articles from a connected content management system.

Keywords: Gatsby, static web development, static website

Sisällys

Sanasto

API	Ohjelmointirajapinta, joka siirtää dataa ohjelmasta tai palvelusta toiseen (*Application Programming Interface*)
CDN	Sisällönjakeluverkko, jossa käyttäjä saa sisällön lähimmästä sijainnista. Esimerkiksi suomalainen saa sisällön Suomen sisältä, ja yhdysvaltalainen saa sisällön Yhdysvaltojen sisältä. (*Content Delivery Network*)
PWA	Progressiivinen selainsovellus eli selaimessa toimiva ladattava sovellus (*Progressive Web Application*)
JSON	Tiedonsiirron tiedostomuoto, jonka tiedostopääte on .json (*JavaScript Object Notation*)
JSX	Reactin käyttämä syntaksilisäosa JavaScriptiin. Mahdollistaa HTML:n kirjoittamisen JavaScriptissä. (*JavaScript XML*)
NPM	Node-pakettihallinta, eli terminaalin kautta ladattavien koodipakettien ekosysteemi. (*Node Package Manager*)
CMS	Sisällönhallintajärjestelmä, joka tekee sisällön tuottamisesta sivuille tai sovellukseen helpompaa (*Content Management System*)
AWS	Amazonin pilvipalvelu *(Amazon Web Services)*

1 Johdanto

Staattinen verkkotuotanto on nousemassa taas verkkokehittäjien suosioon. Samoin on Gatsby – staattisessa verkkotuotannossa käytettävä ohjelmistokehys, joka on lyhyessä ajassa kehittynyt verkkokehittäjien keskuudessa suosituksi staattisen tuotannon välineeksi. Uusin versio siitä, Gatsby v3, julkaistiin 2021, ja se keskittyy nopeampien sivustojen tuottamiseen ja paremman kehittäjäkokemuksen saavuttamiseen (Warren 2021).

Mikä Gatsby on? Mihin sitä käytetään? Milloin sitä kannattaa käyttää ja milloin ei? Tämä opinnäytetyö esittelee ja tutkii Gatsbyn toiminnallisuutta generoida verkkosivuja ja verkkosovelluksia staattisiksi versioiksi, eli toimimaan ilman palvelinpuolen prosessointia.

Tässä työssä on kaksi osiota: teoreettinen osio, joka on lähteisiin perustuvaa, sekä toiminnallinen osio, joka vie teorian käytäntöön ja sitä kautta tuo oikeat Gatsbyn hyödyt ja haasteet esille. Teoreettinen osa tulee vastaamaan keskeisiin kysymyksiin Gatsbysta: mikä se on, mitä varten se on luotu ja milloin sitä kannattaa käyttää? Toiminnallinen osa kertoo siitä, kuinka rakensin staattisen artikkelisivuston Suljin.fi käyttäen Gatsbya. Siinä kerron, mitä vahvuuksia ja heikkouksia löysin Gatsbyssa, mitä ongelmia matkalla tuli vastaan sekä mitä kompromisseja minun on täytynyt tehdä.

Työn tietopohja perustuu kirjallisuuteen, verkon lähteisiin (blogit, podcastit), havaintoihini olemalla osana Gatsbyn yhteisöä, sekä omiin kokemuksiini käytettyäni Gatsbya nyt viimeiset 11 kuukautta.

Tavoitteeni on tuoda verkkokehittäjälle ymmärrys siitä, mitä ongelmia Gatsby ratkaisee sekä miten se ne ratkaisee. Oletan lukijalla olevan hieman ymmärrystä verkkoon kohdistuvasta koodauksesta ja sen käsitteistä, mutta perusteita edistyneempiä käsitteitä työ tulee avaamaan auki.

2 Mitä on staattinen verkkotuotanto

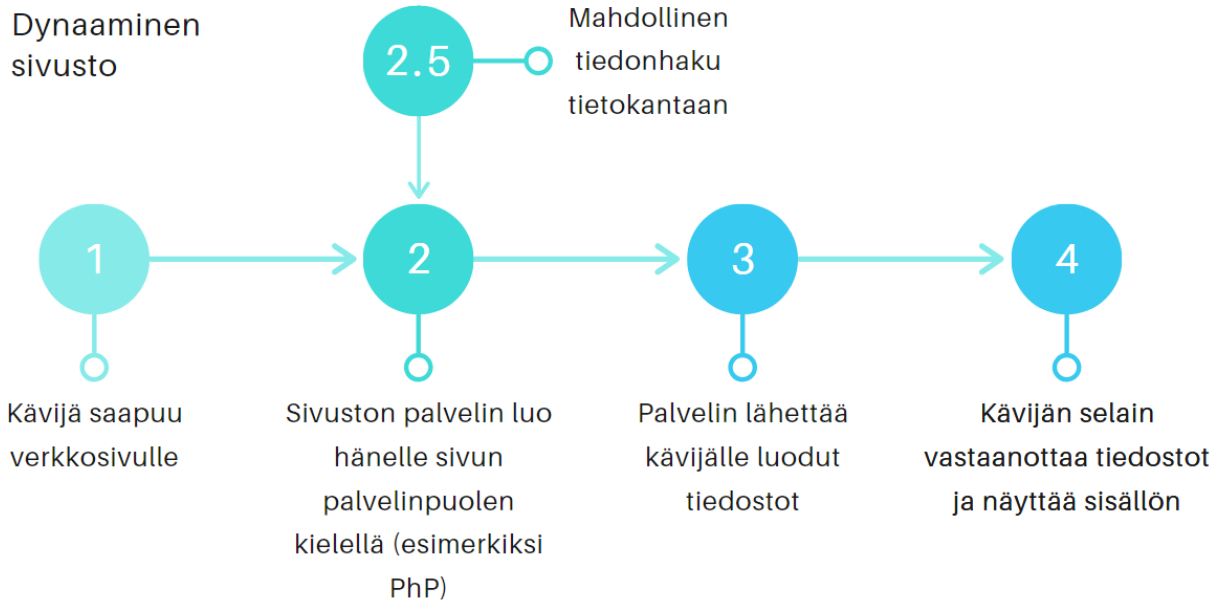
Staattinen verkkotuotanto on staattisten verkkosivujen ja -sovellusten tuottamista, eli sellaisten sivustojen, joilla ei ole palvelinpuolen prosessointia tai tietokantaa. Staattinen verkkosivu (engl. *static website*) on yksi tai useampi HTML-tiedosto, jotka on ladattu palvelimelle, josta verkkosivulla kävijän selain vastaanottaa ne sellaisenaan. (Kalkman 2020.) Voisin jopa kuvailla, että staattisen sivuston voi periaatteessa lähettää esimerkiksi sähköpostillakin toiselle käyttäjälle – se on vain HTML-tiedosto mahdollisilla CSS- ja JS-tiedostoilla, joiden avaamiseen ei tarvitse muuta kuin selaimen. Alla oleva kuvio 1 esittelee, mitä tapahtuu, kun kävijä vierailee staattisella sivustolla.

Kuvio 1. Infograafi siitä, kun kävijä vierailee staattisella sivustolla.

Internetin alkuvaiheessa periaatteessa jokainen sivusto oli staattinen sivusto. Toki kukaan ei kutsunut sitä sellaiseksi, koska se oli ainoa tapa julkaista verkkosivu. (Camden & Rinaldi 2017, Chapter 1.) Sitten tulivat dynaamiset verkkosivut, ja internet muuttui huomattavasti.

Staattisen verkkosivun vastakohta on dynaaminen sivu, joka käyttää palvelinpuolen teknologiaa (esimerkiksi PHP) ja mahdollisesti tietokantaa (esimerkiksi SQL) sivuston rakentamiseen silloin, kun kävijä käy sivustolla. Eli kun sivuston kävijä saapuu sivun osoitteeseen, palvelin generoi hänelle sivun, jonka kävijän selain vastaanottaa. (Kalkman 2020.) Palvelimen luoma sivu voi

olla kävijälle uniikki näyttäen juuri hänelle kohdistettua sisältöä. Näin toimii esimerkiksi verkkopankki: kirjauduttuaan verkkopankkiin jokainen kävijä saa itselleen uniikkia sisältöä eli palvelimen generoimia tiedostoja. Nämä tiedostot ovat sellaisia, joita vain juuri kyseinen kävijä vastaanottaa – muut kävijät saavat samalla tavalla vain itselleen uniikkeja tiedostoja. Staattisella sivulla tämä ei olisi mahdollista, sillä tämä toiminnallisuus vaatii palvelinpuolen prosessointia ja tietokannan, eli dynaamisen sivun. Alla oleva kuvio 2 esittelee, mitä tapahtuu, kun kävijä vierailee dynaamisella sivustolla.



Kuvio 2. Infograafi siitä, kun kävijä vierailee dynaamisella sivustolla.

2.1 Staattisen verkkosivun vahvuudet

Staattisten sivujen suosion nousussa on takana kaksi erityistä syytä: ne ovat nopeita, sekä ne ovat turvallisia (Camden & Rinaldi 2017, Chapter 1). Tämän lisäksi ne ovat luotettavia, skaalattavia ja niiden ylläpito palvelimella on halpaa (Kumari 2018).

Staattinen sivu on nopea, koska sillä ei ole samoja hidasteita kuin palvelinpuolella toimivalla sivulla. Staattisen sivuston ei tarvitse suorittaa

palvelinpuolen koodia, tarkistaa tietokantaa tai renderöidä dynaamisesti sisältöä. Kaikki staattisen sivuston kävijät saavat samat tiedostot, jonka vuoksi sivusto on myös helppo toimittaa kävijälle CDN:n kautta. (Camden & Rinaldi 2017, Chapter 1.)

Staattisen sivuston turvallisuus tulee siitä, että se suojaa sivustoa ja kävijää paremmin kuin dynaaminen sivusto. Kaksi yleisintä tapaa murtaa verkkosivuston turvallisuus on tehdä SQL-injektio tietokannan kautta, jossa hyökkääjä voi tehdä haitallisia komentoja suoraan tietokantaan ja vaarantaa sivun turvallisuuden sitä kautta (Acunetix n.d. a), sekä sivustojen välinen komentosarjahyökkäys (engl. *cross-site scripting attack*), jossa hyökkääjä voi suorittaa pahantahtoista skriptiä käyttäjän selaimessa hänen vieraillessaan sivulla (Acunetix n.d. b). Yleensä hyökkääjät pääsevät käsiksi sivuun koodissa olevien haavoittuvuuksien takia, esimerkiksi palvelinpuolen palveluiden kautta. Staattisella sivustolla ei ole tietokantaa, johon hyökkääjä voisi päästä käsiksi, eikä palvelinpuolen palveluita, joiden haavoittuvuuksia hyökkääjä voisi hyödyntää – hyökkääjällä on vain samat tiedostot, jotka on kaikilla muillakin sivun kävijöillä. (Camden & Rinaldi 2017, Chapter 1.)

Staattisen sivuston luotettavuus ja skaalautuvuus tulevat sivuston staattisesta luonteesta. Dynaamisen sivuston kasvaessa kasvaa myös sen vaatima palvelinpuolen prosessoinnin määrä. Staattisilla sivuilla ei ole tätä ongelmaa, sillä niillä sivuston skaalaus tapahtuu vain nostamalla palvelimen kaistanleveyttä, eli datan määrää, jonka palvelin pystyy lähettämään kerralla. (Kumari 2018.)

Muita staattisen verkkosivun hyötyjä ovat sen palvelintilan hankinnan helppous ja koodin versiointi (Camden & Rinaldi 2017, Chapter 1; Kumari 2018). Staattisen sivuston palvelintila toimii usein pienellä kustannuksella, mahdollisesti jopa ilmaiseksi. Esimerkiksi Netlify ja GitHub Pages tarjoavat ilmaisen palvelintilan staattisille sivuille. Versioinnin avulla voi seurata staattisen sivuston muutoksia helposti esimerkiksi Gitin ja Githubin avulla. (Camden & Rinaldi 2017, Chapter 1.)

2.2 Staattisen verkkosivun heikkoudet

Ironisesti staattisen sivuston suurimmat heikkoudet ovat juuri ne asiat, jotka ovat sen suurimmat vahvuudetkin: palvelinpuolen prosessoinnin ja tietokannan puuttuminen. Niiden puute vähentää sitä, mitkä sivustot voivat olla staattisia luonnostaan. Jos sivusto vaatii esimerkiksi käyttäjän rekisteröintiä, kirjautumisia tai yksilöllistä tietoa, vaatii se palvelinpuolen prosessointia, joten se ei voi olla itsestään staattinen sivusto (Camden & Rinaldi 2017, Chapter 1).

Staattisesta sivusta myös puuttuu hyvin yleinen verkkosivun ominaisuus: sisäänrakennettu CMS eli sisällönhallintajärjestelmä. CMS:n avulla sivuston sisällöntuottajat voivat muuttaa nykyisiä sisältöjä ja lisätä uusia sisältöjä täysin ilman koodin ymmärtämistä tai sen kirjoittamista. Tällaisia sisältöjä voivat olla esimerkiksi blogi-julkaisut tai artikkelit. Esimerkiksi tämän uskon olevan yksi syistä, miksi WordPress on niin suosittu. Ilman CMS:ää staattiselle sivustolle joutuu julkaisemaan tekstit ja artikkelit perinteisellä tavalla: kirjoittamalla HTML-koodina haluamansa tekstit ja formatoinnit. Tämä on hyvin aikaa vievää, sekä vain verkkokehittäjä voi tehdä sitä – esimerkiksi yrityksen markkinointitiimi ei välttämättä osaisi julkaista uusia julkaisuja tällä tavalla.

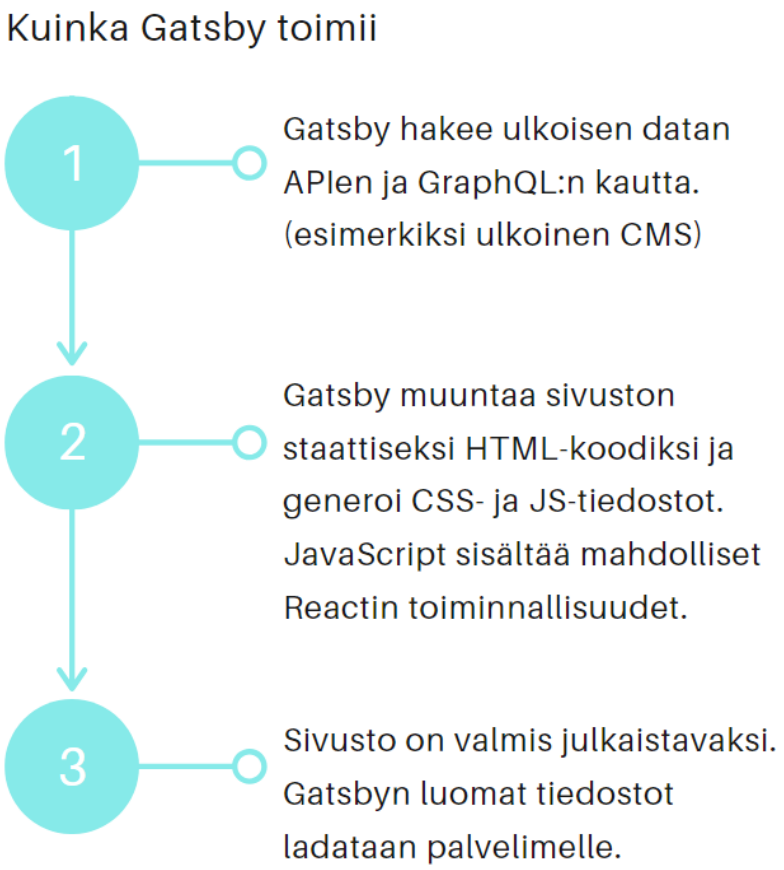
Yksi heikkouksista on myös saman koodin toistuminen kaikissa HTML-tiedostoissa. Jos verkkosivulla on viisi sivua, se tarkoittaa sitä, että siinä on viisi eri HTML-tiedostoa. Jos jokaisella sivulla on samanlainen alaosa, jokaisessa viidessä HTML-tiedostossa on myös sama alaosan koodi. Jos kehittäjä haluaa tehdä alaosaan muutoksia, joutuu hän tekemään saman muutoksen jokaiseen tiedostoon. (Kalkman 2020). No, viiden eri tiedoston alaosan muuttaminen kopioinnilla ja liittämisellä ei ole vaikeaa, mutta mitä jos sivusto laajenee tulevaisuudessa? Viidestä sivusta tuleekin 15 sivua? 50 sivua? 100 sivua? Sivuston päivittäminen vaikeutuu laajentumisen yhteydessä jatkuvasti, sekä mahdollisuus tiedoston unohtamiselle tai virheille nousee. Koodin toistaminen on huonoa käytäntöä, ja sitä kannattaa välttää (Verreckt 2017).

3 Mikä on Gatsby

Luvut 2.1 ja 2.2 ovat käsitelleet staattisen verkkotuotannon hyödyt sekä haitat. Sen tuomat hyödyt ovat houkuttelevia, mutta usein sen heikkoudet saattavat estää sen käytön erittäin pienten projektien ulkopuolella. Tässä tilanteessa Gatsby tulee esille.

Gatsby on avoimen lähdekoodin *framework* eli ohjelmistokehys, joka toimii staattisten verkkosivujen generaattorina hyödyntäen Reactia, GraphQL:ia ja API:a (So 2021, Chapter 1). Gatsbyn ensimmäinen virallinen versio, v1, julkaistiin kesäkuussa 2017 (Mathews 2017), uudempi, paranneltu Gatsby v2 julkaistiin syyskuussa 2018 (Mathews 2018), sekä tämän opinnäytetyön tuorein Gatsby v3 julkaistiin 2021 (Warren 2021).

Gatsby toimii staattisten sivujen generaattorina. Sen avulla voi uudelleenkäyttää samaa koodia ja komponentteja Reactin avulla, tuoda sivustolle ulkoista dataa APIen kautta, hyödyntää CMS:n helppoutta julkaista uutta sisältöä, sekä renderöidä staattista HTML-koodia Gatsbyn rakennusprosessissa (*build process*). Silti sivuston lopputulos on staattinen sivusto – eli ryhmä HTML-, JS- ja CSS-tiedostoja, joilla on samat aiemmin mainitut staattisen verkkosivun hyödyt, mutta tällä kertaa ilman osaa staattisen verkkotuotannon heikkouksista.



Kuvio 3. Kuinka Gatsby toimii.

Gatsby on vielä suhteellisen nuori ohjelmistokehys, mutta olen huomannut sen levinneen kehittäjien keskuudessa yhdeksi vahvaksi vaihtoehdoksi verkkosivujen ja verkkosovellusten luonnissa. Sen keskeisimpiä ominaisuuksia ovat hyvä kehittäjän kokemus, valmiit teemat, plugin-ekosysteemi ja nopeus (So 2021, Chapter 1). Näistä ominaisuuksista Gatsbyn etusivun nykyinen 2021 iskulausekin tulee: "*Create blazing fast websites and apps AND harness the power of 2000+ plugins*" (gatsbyjs.com), eli luo nopeita nettisivuja ja sovelluksia sekä ota käyttöösi yli 2000 pluginin voima.

3.1 Mitä ongelmia Gatsby ratkaisee

Gatsbylla on neljä erityistä hyötyä: nopeus, kehittäjäkokemus, turvallisuus ja saavutettavuus (So 2021, Chapter 1). Itse lisäisin listaan PWA:n luonnin

helppouden. Olen myös huomannut, että suuri osa käyttäjistä käyttää Gatsbya myös SEO- eli hakukoneoptimointihyötyjen takia, itseni mukaan lukien. Käyn nämä väitteet nyt läpi yksi kerrallaan.

3.1.1 Nopeus

Gatsbylla rakennettu sivu on nopea. Tämä johtuu siitä, että sivut ovat renderöity HTML-tiedostoiksi, jolloin kävijän ei tarvitse ladata vielä ollenkaan JavaScriptiä nähdäkseen sivun sisältöä. Vasta lisätoiminnallisuudet tulevat Reactin kautta JavaScriptinä HTML:n jälkeen. (So 2021, Chapter 1.) Tämä on selaimelle nopeampi tapa näyttää sisältöä kuin esimerkiksi Reactilla tuotettu staattinen sivu, sillä pelkällä Reactilla itse HTML-tiedosto on lähes tyhjä, koska sivuston sisältö renderöityy vasta selaimessa JavaScriptin kautta. Tämä tarkoittaa sitä, että Reactilla tehdyllä sivustolla sisältö voi näkyä vasta, kun selain on ladannut ja suorittanut JavaScript-tiedostot, toisin kuin Gatsbylla. (So 2021, Chapter 1.)

Gatsbylla on myös mahdollisuus tuoda CDN:n eli sisällönjakeluverkon voima koko sivustolle. Gatsbylla tehty sivusto on vain staattisia tiedostoja, joten koko sivuston voi tarjoilla kävijöille CDN:n kautta. Näin sivulla kävijä saa jokaisen tiedoston lähimmältä palvelimeltaan, esimerkiksi oman maansa sisältä. Tämä sisältää jopa HTML-tiedostot. (So 2021, Chapter 1.)

Kuvien kannalta Gatsbylla tehdyn sivuston valttikortti nopeuteen on Gatsbyn mahdollisuus prosessoida sivuston kuvat automaattisesti moneen eri resoluutioon ja tiedostomuotoon sekä lisätä kuville paikanpitäjät (engl. *placeholder*). Prosessoinnin jälkeen kuvien tiedostokoko on pienempi, sekä monen eri resoluution kuvan tekeminen mahdollistaa sen, että kävijä saa juuri omalle näytölleen sopivan tiedoston. Miksi esimerkiksi antaa mobiilikäyttäjille sama kuva kuin tietokoneen käyttäjille, jos mobiililla näyttö on paljon pienempi? Tämän lisäksi sivuston kävijä voi nähdä ennen kuvan latautumista paikanpitäjän, joka on esimerkiksi väliaikainen sumennettu pienen resoluution versio kuvasta. Tämä mahdollistaa sivuston nopealta tuntumisen lisäksi myös sen, että kuvan latautuminen ei siirrä sivun sisältöä latautuessaan. (Gatsby n.d.

a.) Gatsbylla toimivalle sivustolle saa asennettua kyseiset toiminnot käyttäen pluginia `gatsby-plugin-image`, joka on uusi Gatsby v3 -päivityksessä tullut plugini. Edellisissäkin Gatsbyn versioissa oli vastaava plugini, mutta uusi v3:n plugini nostaa kuvien nopeutta jopa edellistäkin enemmän. (Warren 2021.)

Linkitys sivujen välillä Gatsbyssa tapahtuu käyttäjän kokemuksessa lähes välittömästi. Tämän mahdollistaa Gatsbyn sisäänrakennettu `<Link>` -linkkielementti, joka korvaa Gatsbyn sisällä tapahtuvassa sivuille linkittämisestä kehittäjälle tutut `<a>` -linkkielementin. `<Link>` -elementin avulla kävijän selain lataa ennakkoon seuraavan sivun silloin, kun käyttäjä vaikuttaa menevän kyseiselle sivulle. Tämä mahdollistaa lähes välittömän seuraavan sivun lataamisen, sillä uusi sivu on jo osittain latautunut selaimen taustalla käyttäjän tietämättä. Tämä toiminto toimii niin, että Gatsby alkaa lataamaan uutta sivua pienellä prioriteetilla jo silloin, kun siihen osoittava linkki ilmestyy käyttäjän ruudulle, ja suurella prioriteetilla silloin, kun käyttäjä asettaa hiiren linkin päälle. (Gatsby n.d. b.)

3.1.2 Kehittäjäkokemus

Gatsby tekee verkkosivun kehittämisestä suoraviivaista ja aikaa säästävää. Tässä se onnistuu helposti asennettavilla valmiilla teemoilla ja plugineilla, automaattisesti päivittyvällä lokaalilla ympäristöllä (engl. *hot reloading*), valmiiksi asennetulla GraphQL:lla, sisäänrakennetulla automaattisella testauksella ja linting-prosessilla, joka varoittaa saavutettavuusongelmista, parhaista koodauksen käytännöistä ja muista ongelmista. (So 2021, Chapter 1.)

Oma kehittäjäkokemukseni Gatsbyn kanssa on ollut positiivinen. Sen käyttöönotosta on löytynyt loistavia ohjeita niin Gatsbyn omilta sivuilta kuin ulkoisista lähteistäkin, kuten YouTubesta. Pidän myös siitä, miten automaattinen testaus ja linting-prosessi kertovat heti, jos projektin tiedostoissa on jotain väärin. Välillä ne toimivat hyvin kertoen tiedostosta tarkalleen virheen sisältävän rivinumeron, mutta välillä virhekoodit jättävät sivuilla olevan virheen täysin mysteeriksi. Kehittäminen on silti erittäin nopeaa, ja lokaalin ympäristön

automaattinen päivitys aina tiedostojen tallentamisen yhteydessä on erittäin mukavaa.

Sanoisin vielä, että laaja määrä plugineita mahdollistaa sen, että sivuston voi luoda juuri omalla haluamallaan tavalla. Jos sivu tehdään esimerkiksi SASS:n tai SCSS:n avulla normaalin CSS:n sijaan, ladataan sivulle plugini `gatsby-plugin-sass`. Jos sivustolle tarvitaan automaattisesti päivittyvä sitemap.xml -tiedosto, ladataan plugini `gatsby-plugin-sitemap`. Jos sivusto tarvitsee automaattisesti päivittyvät uusimmat Instagram-tilin päivitykset, ladataan plugini `gatsby-source-instagram`. Pluginit tekevät yleisistä tarpeista täysin helppoja täytettäviä – miksi nähdä vaivaa ongelmaan, jos siihen on jo tehty täysin toimiva ratkaisu käyttäen avointa lähdekoodia?

3.1.3 Turvallisuus

Gatsby tuottaa rakennusprosessissaan (*build process*) staattista koodia dynaamisen palvelinpuolelta tulevan koodin sijaan, joten verkkosivun turvallisuus huomattavasti parempi kuin palvelinpohjaisella sivulla. Rakennusprosessi mahdollistaa sen, että sivun kävijällä ei ole mahdollisuutta tietää, mitä kautta sivun data tulee, esimerkiksi mitä APIa sivusto käyttää. Tämän mahdollistaa se, että yhdistäminen APIiin on tapahtunut jo aiemmin rakennusprosessissa, eikä se ole jättänyt ollenkaan jälkiä julkisena olevaan sivustoon. Tämä voi olla hyvinkin hyödyllistä esimerkiksi sisällöntuotannon turvallisuuden kannalta – jos sivulta ei voi saada edes selväksi, mistä sivun sisältö tulee, se on heti paremmin turvassa vääriltä käsiltä. (So 2021, Chapter 1.)

3.1.4 Saavutettavuus

Gatsbylla on saavutettavuusseloste, joka on luettavissa Gatsbyn verkkosivuilla osoitteessa gatsbyjs.com/accessibility-statement/. Seloste ei anna paljoa lupailuja Gatsbyn saavutettavuudesta, mutta tuo esille saavutettavuuden

tärkeyden. Selosteessa Gatsbyn tiimi mainitsee haluavansa tehdä Gatsbyn ja sillä luodut verkkosivut saavutettaviksi ja tavoittelevansa WCAG 2.1 AA -saavutettavuuskriteeriä, jossa he omien sanojensa mukaan ovat osittain onnistuneet. (Gatsby n.d. c.)

Se, että Gatsby generoi sivuston koodin staattiseksi HTML:ksi, tarkoittaa sitä, että sivustolla käymiseen ei vaadita JavaSciptiä (Gatsby n.d. c). Esimerkiksi Mozilla ei suosittele sivuston HTML-koodin generoimista selaimessa JavaScriptin avulla, vaan suosittelee staattista HTML:ää, eli juuri sitä, mitä Gatsby generoi sivun rakennusprosessissa (MDN Web Docs n.d. a).

Lisää informaatiota Gatsbyn saavutettavuudesta antaa Gatsbyn blogissa Marcy Stutton (2019) blogijulkaisussa *Our Commitment to Accessibility in Gatsby*. Siinä hän mainitsee, että Gatsbyn tiimi on intohimoinen saavutettavuuden saavuttamiseen. Hän mainitsee myös sen, että sivustot, jotka on rakennettu suurella määrällä JavaScriptia ovat yleisesti vähemmän saavutettavia, ja muistuttaa, että Gatsbylla tehty sivusto generoituu käyttäen Nodea ja Reactia (joista kummatkin käyttävät JavaScriptia). Stutton mainitsee heidän ymmärtävän saavutettavien oletusasetusten ja esimerkkien käytön tärkeyden Gatsbyssa. (Stutton 2019.)

Oma kokemukseni Gatsbyn saavutettavuudesta on ollut hyvä. Itse nostaisin esille enemmän sitä, että omasta mielestäni suurimman osan sivuston koodin luomasta saavutettavuudesta hoitaa lopulta itse sivuston kehittäjä. Esimerkkinä tästä toimii sivustolla navigoiminen näppäimistöllä: jos kehittäjä on luonut sivuston HTML:n ja JavaScriptin väärin ja sen vuoksi näppäimistöllä navigoiminen on vaikeaa tai mahdotonta, ei Gatsby tai muu ohjelmistokehys voi korjata ongelmaa itsestään, vaan kehittäjällä itsellään täytyy olla ymmärtäminen saavutettavan koodin perusteista. Tässä kuitenkin Gatsby auttaa omalla linting-prosessillaan, joka kertoo, jos koodissa ilmenee automaattisesti tunnistettavia saavutettavuusongelmia (So 2021, Chapter 1).

3.1.5 Progressiivinen selainsovellus (PWA)

Yleisesti PWA:n luominen saattaa olla kehittäjälle haastavaa, mutta Gatsbylla sen tekeminen on helppoa. PWA (*Progressive Web Application*) eli progressiivinen selainsovellus on selaimessa tapahtuva sovelluksen ja verkkosivun välimaasto. Se yhdistää verkkosivuun sovellusten vahvuudet: PWA latautuu puhelimelle ja tuntuu kuin se olisi osa puhelinta. Käyttäjä pystyy lisäämään PWA:n kotiruudulleen ja käyttämään sitä huonollakin verkkoyhteydellä tai jopa ilman verkkoyhteyttä. Myös ilmoitukset (engl. *push notifications*) ovat mahdollisia PWA:n kautta. Tämän lisäksi PWA pystyy olemaan yhteydessä puhelimen tiedostojen, toimintojen ja bluetoothin kanssa mahdollistaen esimerkiksi puhelimen oman kalenterin käytön. (Richard & LePage 2020.)

PWA:n käyttöön on kolme kriteeriä: verkkosivun täytyy käyttää HTTPS:ää, sen täytyy sisältää *Web App Manifest*, joka kertoo selaimelle tiedot PWA:sta ja siitä, kuinka PWA:n pitäisi käyttäytyä, sekä sen täytyy ottaa käyttöön *service worker*, joka on skripti, joka kontrolloi sitä, kuinka selain käsittelee verkkopyyntönsä ja välimuistin. *Service worker* mahdollistaa sen, että PWA toimii myös ilman verkkoyhteyttä. (MDN Web Docs n.d. b.)

Gatsbylla tehty sivusto on jo valmiiksi hieman PWA:n tapainen, sillä sivusto tuntuu nopeudeltaan erittäin natiivin sovelluksen tapaiselta. Tämä mahdollistuu Gatsbyn ennakkoon tapahtuvalla sivujen lataamisella ja automaattisella kuvien optimoinnilla, joista kumpaakin käsitteli työn luku 3.1.1: *nopeus*. (Gatsby n.d. d.) PWA:n kolme kriteeriä täyttyvät Gatsbylla tehdyllä sivustolla helposti:

1. HTTPS. Tähän Gatsby ei vaikuta itsessään mitenkään, mutta palvelut, joihin yleisimmin Gatsbylla tehty sivu ladataan (Netlify, GitHub Pages, Gatsby Cloud), käyttävät oletuksena HTTPS-yhteyttä.
2. Web App Manifest. Manifestin saa asennettua Gatsby-sivulla käyttäen pluginia `gatsby-plugin-manifest`. (Gatsby n.d. d.) Pluginissa sivun tietojen lisääminen PWA:han tapahtuu `gatsby-config.js` -tiedoston kautta. Tietoja

ovat esimerkiksi sivuston nimi, ikoni, taustaväri ja ruudun tyyli (eli esimerkiksi näkyykö selaimen osoitepalkkia ollenkaan).

3. Service Worker. Tämän saa asennettua Gatsbyyn käyttäen pluginia gatsby-plugin-offline. (Gatsby n.d. d.) Nimensä mukaisesti se mahdollistaa sivuston käytön myös ilman internet-yhteyttä.

3.1.6 Hakukoneoptimointi (SEO)

SEO eli hakukoneoptimointi (*Search Engine Optimization*) on yksi niistä syistä, miksi olen huomannut monen käyttävän Gatsbya. Gatsbyn käyttäjä hyötyy SEO:ssa muun muassa valmiiksi renderöidystä HTML:stä, metatiedoista, jäsennellystä datasta (engl. *structured data*) ja Gatsbyn nopeudesta (Gatsby n.d. e).

Valmiiksi renderöity HTML ei ole vain käyttäjälle nopeampi tapa ladata sivusto, vaan se tuo myös sivuston sisällön paremmin esille hakukoneiden roboteille (Gatsby n.d. e). Esimerkiksi käyttämällä Reactia ilman Gatsbya sivulla kävijän selain vastaanottaa lähes tyhjän HTML-tiedoston, koska selain luo vasta JavaScriptin perusteella sivuston sisällön. Kaikki robotit eivät välttämättä pysty lukemaan JavaScriptillä luotua sisältöä, sillä ne lukevat vain HTML-tiedoston läpi eivätkä suorita JavaScript-koodia. Näin varoittaa myös Google, joka sanoo pystyvän hakukoneensa roboteillaan lukemaan myös JavaScriptillä luotua sisältöä, mutta samalla varoittaa, että muut palvelut ja sivustot eivät välttämättä pysty samaan (Google 2021a). Kuulen usein SEO-yhteisöissä monen sanovan, että valmiiksi renderöity HTML-tiedosto on myös Googlen hakutuloksille parempi kuin JavaScriptillä renderöity sisältö, mutta se on lähivuosina saattanut muuttua. Kuitenkin omana kokemuksenani vielä vuonna 2021 esimerkiksi Facebook ja Twitter vaativat “korteissaan”, eli linkin esikatseluissa, olevien sivuston metatietojen olevan HTML-tiedostossa, eikä JavaScriptillä generoituvia. Eli jos sivustolle linkittävän julkaisun yhteydessä näkyvät metatiedot – kuten kuva ja otsikko – eivät näy suoraan HTML-tiedostossa, julkaisun linkkiin ei tule esikatselua linkistä.

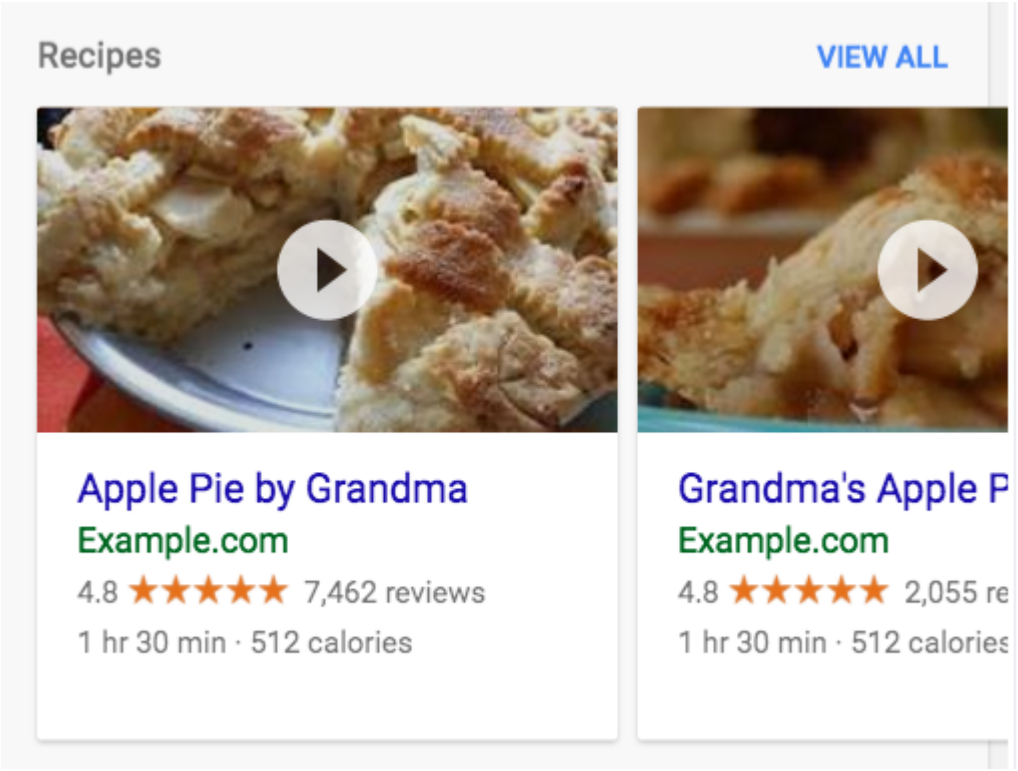
Yleisesti, metatietojen generoiminen Gatsbyssa tapahtuu pluginien react-helmet ja gatsby-plugin-react-helmet avulla. Aiempi react-helmet lisää koodia sivuston <head> -osioon, ja jälkeinen gatsby-plugin-react-helmet renderöi lisätyn koodin staattiseksi versioksi HTML-koodiin. (Bergé 2019; Gatsby n.d. e.) Käyttäen pelkkää react-helmet ilman gatsby-plugin-react-helmet tulisivat sivuston metatiedot vain React-koodin kautta JavaScriptinä, eli kuten aiemmassa kappaleessa kerroin, eivät kaikki robotit välttämättä näkisi lisättyä metatietoa. Eli tiivistetysti, Gatsbylla voi helposti generoida staattisia metatietoja sivustolle käyttäen plugineita, ja ne ovat sivustoa selaaville roboteille helposti nähtävissä, joka auttaa hakukoneoptimointia.

Jäsennelty data on sivustolle lisättävää koodia, joka auttaa hakukoneita – kuten Googlea – ymmärtämään sivuston sisältöä paremmin. Jäsennelty data on vakioitu koodi, johon kirjoitetaan sivustoa kuvaavaa informaatiota. Tällaisia tietoja voivat olla esimerkiksi sivuston nimi, artikkelin nimi, artikkelin kirjoittaja, yrityksen osoite, sivun julkaisupäivä tai jopa ruuanlaiton reseptin valmistamiseen menevä aika. Google suosittelee jäsennellyn datan käyttöä aina, kun mahdollista. (Google 2021b.) Katso esimerkki jäsennellyn datan koodista kuviosta 4 alla.

```
<html>
  <head>
    <title>Party Coffee Cake</title>
    <script type="application/ld+json">
    {
      "@context": "https://schema.org/",
      "@type": "Recipe",
      "name": "Party Coffee Cake",
      "author": {
        "@type": "Person",
        "name": "Mary Stone"
      },
      "datePublished": "2018-03-10",
      "description": "This coffee cake is awesome and perfect for parties.",
      "prepTime": "PT20M"
    }
    </script>
  </head>
```

Kuvio 4. Esimerkki jäsennellyn datan käytöstä sivuston koodissa reseptisivulla. (Google 2021b)

Jäsennelty data myös mahdollistaa tiettyihin kategorioihin kuuluvien sivustojen näkymisen graafisina tuloksina Googlessa, kuten reseptisivujen (Google 2021b). Katso esimerkki kuviosta 5 alla. Olen myös huomannut uutisten olevan samoin toimiva graafinen tulos Googlen hauissa.



Kuvio 5. Kuinka Google näyttää resepteiksi merkittyjä sivustoja hakutuloksissaan. (Google 2021b)

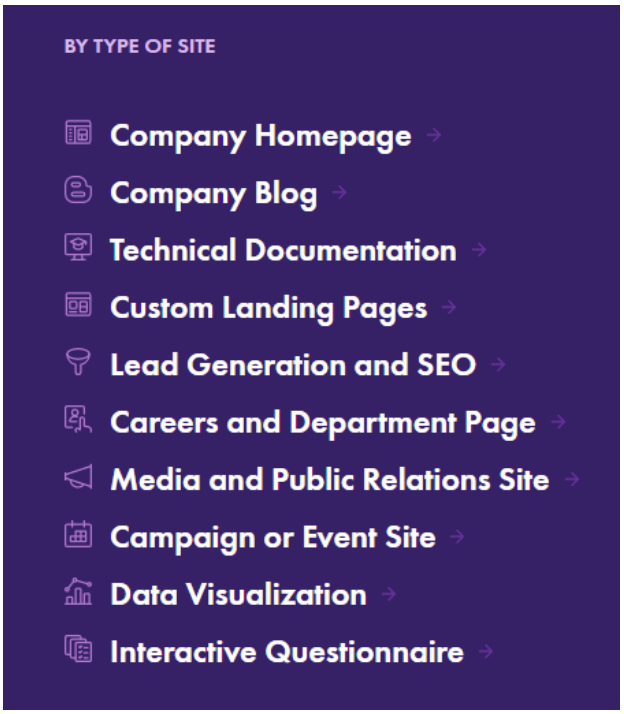
3.2 Mihin Gatsby soveltuu

Oman kokemukseni mukaan Gatsby sopii parhaiten sellaisille sivustoille, joita ei tarvitse päivittää jatkuvasti, ja joiden halutaan olla erittäin nopeita. Arvioisin, että noin pari kertaa päivässä päivittyvä sivu hyötyy yhä Gatsbysta, sillä itse rakennusprosessissa menee keskiverrosti omien havaintojeni mukaan suurimmalla osalla sivustoista noin 3–5 minuuttia. Sivusto on yhä livenä rakennusprosessin tapahtuessa, eli keskiverto Gatsby-sivusto voi saada päivityksiä useastikin saman päivän sisällä ilman ongelmia. Jos sivustoa

päivitetään erittäin useasti, esimerkiksi 10 minuutin välein, alkaa huomaamaan Gatsbyn huonoja puolia: rakennusprosessi on silloin jatkuvaa, jonka vuoksi kannattaa harkita dynaamista sivustoa sen sijaan.

Parhaita sivustoja Gatsbylle ovat esimerkiksi blogi-, portfolio-, uutis- ja yrityssivustot. Niiden sisältöä ei tarvitse päivittää pyyntökohtaisesti, koska itse sisältö (esimerkiksi blogikirjoitus) muuttuu niin harvoin, että renderointi kannattaa tehdä Gatsbyn tavalla rakennusprosessilla, kuin dynaamisen sivuston tavalla pyyntökohtaisesti käyttäjän käydessä esimerkiksi blogikirjoituksen osoitteessa. (Väänänen 2019, 13–14.)

Gatsbylle sopivina käyttötarkoituksina Gatsby itse mainitsee “Use Cases” -sivullaan (gatsbyjs.com/use-cases/) useita eri sivustotyylejä (katso kuvio 6 alla), joista itse nostaisin esiin seuraavat: yrityksen nettisivut, blogi, tekninen dokumentaatio, landing-sivusto ja tapahtumasivusto.



Kuvio 6. Kuvakaappaus Gatsbyn “Use Cases” -sivulta (gatsbyjs.com/use-cases/).

Omat nostoni ovat mielestäni juuri ne, joihin Gatsbya käytetään kaikista useimmin. Väitteeni perustuu Gatsby-yhteisöjen keskusteluihin ja jakamisiin omista projekteistaan. Arvioin, että nämä käyttötarkoitukset juuri ovatkin niitä, missä Gatsby loistaa: ne ovat informaatiopainotteisia sivuja, joihin jo valmiiksi julkaistu sisältö muuttuu harvoin, mutta uutta sisältöä on helppo lisätä.

Gatsby ei ole *vain* uusi trendikäs harrastajien framework, vaan suuretkin verkkosivut käyttävät sitä sivustoillaan. Tässä niistä muutama:

- ReactJS käyttää Gatsbya koko sivustoonsa, mukaan lukien etusivu ja dokumentaatio (reactjs.org) (W3Techs n.d).
- Figma käyttää sivustollaan Gatsbya. Itse Figman kirjautumisen takana oleva sovellus ei ole Gatsbya. (figma.com) (W3Techs n.d).
- KFC Global käyttää sivustollaan Gatsbya (global.kfc.com) (W3Techs n.d).
- Patreon käyttää Gatsbya blogiinsa (blog.patreon.com) (Gatsby n.d. f).
- Cloudflare käyttää Gatsbya dokumentaatiossaan (developers.cloudflare.com/docs) (Gatsby n.d. f).
- TypeScript käyttää sivustollaan Gatsbya etusivulla ja dokumentaatiossa (typescriptlang.org) (W3Techs n.d).

Varmistin vielä itse Gatsbyn käytön kaikilla mainituilla sivustoilla käyttäen selaimen lisäosaa Wappalyzer.

Toki kannattaa pitää mielessä, onko Gatsby liioittelua pienelle projektille. Joskus Gatsby ei ole oikea väline edes harvoin päivitettäville staattisille sivustoille – eli juuri niille sivustoille, joille aiemmin väitin sen olevan sopiva väline. Näen usein Gatsbyn yhteisöissä, kuinka käyttäjät luovat esimerkiksi 1–2 sivuisen portfolionsa Gatsbylla. Näissä sivustoissa sivulla kävijä lataa “turhaan” suurehkon kasan Gatsbyn omaa JavaScriptiä (mukaan lukien Reactin), vaikka sivusto voisi luultavasti toimia täysin samalla tavalla ilman mitään frameworkia, hyödyntäen vain HTML-, CSS- ja JS-koodia. Tätä mieltä on myös Eric Murphy (2020), joka tuo esille videossaan “Why I’m no longer using GatsbyJS” mielipiteitänsä siitä, miksi hän luovutti Gatsbyn käytön. Yksi hänen syistään on

“turhan” JavaScriptin lisäksi se, että Gatsby on ylisuunniteltu framework tietyille käyttötarkoituksille. Gatsby lataa kehitysvaiheessa suuren määrän riippuvuuksia (npm-paketteja), jolloin npm_modules -kansiosi (johon npm-paketit asentuvat) on luultavasti yli satoja megabittejä, ja riippuvuudet luovat uusia epäonnistumiskohtia. (Murphy 2020.) Olen samaa mieltä Murphyn kanssa, ja tämän vuoksi itse jättäisin Gatsbyn käyttämisen pienillä sivustoilla, ja pysyisin vain tavallisessa käsin koodatussa HTML, CSS ja JS -yhdistelmässä.

Voin myös asettua juuri mainitsemaani minun ja Eric Murphyn mielipidettä vastaan, ja sanoa, että Gatsby voi yhä olla sopiva väline kyseisiin sivustoihin, kuten 1–2 sivun portfolioon. Gatsbyn käyttö voi nopeuttaa pienenkin sivun rakentamista huomattavasti, jos sivusto tarvitsee toimintoja, joita löytyy Gatsbylle jo valmiiksi tehtyinä, esimerkiksi teemoina tai plugineina. Tällainen toiminto voi olla esimerkiksi GDPR-evästeilmoitus, jonka voi ladata sivustolle nopeasti pluginin gatsby-plugin-gdpr-cookies kautta. Myös suuri hyöty Gatsbyssa on mahdollisuus automaattisille kuvien optimoinnille, jolloin kuvista tulee responsiiviset versiot eri ruutukoille, sekä kuvat muunnetaan JPG:tä ja PNG:tä nopeampaan WebP-muotoon. Jos Gatsbyn lisäämä JavaScript on sivulla ongelma, voi Gatsbyyn ladata pluginin gatsby-plugin-no-javascript, joka poistaa Gatsbyn lisäämät JavaScriptit, pitäen sivustolla vain kehittäjän itse lisäämät skriptit.

4 Miten Gatsby toimii

Tämä osio käsittelee sitä, kuinka Gatsby todellisuudessa toimii. Osio käy läpi yksi kerrallaan Gatsbyn keskeisiä toimintoja, sekä antaa näkemystä siitä, miten sivuston tekeminen Gatsbylla tapahtuu. Osion keskeisiä toimintoja ovat Gatsbyn käyttöönotto, Reactin rooli Gatsbyssa, GraphQL, API:t, ulkoiset CMS-palvelut, Gatsbyn rakennusprosessi ja valmiin sivuston julkaisu.

4.1 Käyttöönotto

Gatsby vaatii tietokoneelle asennetun Node.js -ohjelman ja ohjelmointiympäristön, kuten Visual Studio Coden. Gatsbyn asentaminen tapahtuu npm:n kautta, tekemällä `npm install gatsby-cli`. Tämä asentaa Gatsbyn CLI:n, eli Gatsbyn oman komentoliittymän, jolla voi antaa käskyjä Gatsbyn ytimelle. (Gatsby n.d. g.) Käskyjä ovat esimerkiksi kehitysympäristön avaaminen, sivuston rakentaminen ja välimuistin tyhjentäminen.

4.2 Teemojen ja pluginien ekosysteemi

Käyttöönoton jälkeen seuraava askel on asentaa oma *starter*, eli teema, joka on valmiiksi tehty pohja verkkosivuille. Teemoja löytyy Gatsbyn sivuilta satoja, joista useat keskittyvät esimerkiksi blogien, portfolioiden ja yrityssivujen käyttötarkoituksiin. Jos et halua käyttää valmista teemaa, voit silti hyötyä teemasta nimeltä “gatsby-starter-hello-world”, joka pitää sisällään vain Gatsbyn toimintaan vaadittavat tiedostot, jonka avulla voit aloittaa oman sivuston kehittämisen heti. (Gatsby n.d. g).

Teeman asentaminen tapahtuu seuraavan komennon kautta:
`gatsby new hello-world https://github.com/gatsbyjs/gatsby-starter-hello-world`

Yllä oleva CLI:n lause asentaa uuden projektin ja nimeää sen “hello-world”. Sana *gatsby* on Gatsbyn omaa komentoliittymää. Sana *new* kertoo, että tarkoituksena on luoda uusi Gatsby-projekti. Sana *hello-world* on projektin nimi, eli se voi olla käyttäjän itse määrittelemä. Lopuksi oleva osoite on käytettävän teeman sijainti GitHubissa, tässä esimerkissä lähes tyhjä teematiedosto valmiilla Gatsbyn vaatimuksilla.

Pluginien lisääminen sivustolle tapahtuu käyttäen npm-komentoja pakettien lataamiseen. Esimerkiksi suositun kuvien optimointiin käytettävän pluginin

`gatsby-plugin-image` asentaminen tapahtuu seuraavan komennon kautta:
`npm install gatsby-plugin-image`

Yllä olevassa esimerkissä sanat “`npm install`” ovat käsky asentaa npm-paketti, jota seuraa pluginin nimi, `gatsby-plugin-image`.

4.3 React osana Gatsbya

Gatsbylla sivujen luominen tapahtuu käyttäen Reactin JSX-koodia, eli syntaksilisäosaa, joka mahdollistaa HTML:n kirjoittamisen JavaScriptissä (So 2021, Chapter 1). Valmiilla sivustolla käyvän kävijän selain ei luo itse HTML-sisältöä JavaScriptin perusteella, vaan Reactin JSX-koodin muuttaminen staattiseksi HTML-koodiksi on tapahtunut jo aiemmin Gatsbyn rakennusprosessissa (Gatsby n.d. c). Eli Gatsbyn HTML-koodi kirjoitetaan käyttäen JavaScriptiä, mutta Gatsby itse muuttaa sen staattiseksi HTML:ksi.

Reactin käyttäminen Gatsbyssa mahdollistaa komponenttien käytön. Gatsbyssa on neljä olennaisinta komponenttityyliä: sivukomponentti, sivupohja, HTML-komponentti ja osiokomponentti. (So 2021, kappale 1.1.1). Tässä on lyhyt selitys neljästä mainitusta komponenttityylistä:

1. *Sivukomponentti*: kokonainen sivu, joka päätyy sivuksi lopulliseen sivustoon. Jokainen komponentti joka asetetaan Gatsbyssa kansioon `src/pages/` päätyy tiedostonimensä perusteella sivuksi. Jos luot esimerkiksi tiedoston `tietoa.js` kansioon `src/pages/tietoa.js`, luo se sivustollesi sivun osoitteeseen esimerkki.com/tietoa. (So 2021, kappale 1.1.1.)

2. *Sivupohja*: pohja usein toistuvalle sivutyylille. Esimerkiksi blogijulkaisut tai artikkelit hyötyvät pohjasta, jonka avulla ne generoituvat aina samalla tavalla yhden tyylin perusteella. Jos pohjaa muutetaan, muuttuvat kaikki

sitä käyttävät sivut samalla. (So 2021, kappale 1.1.1.)

3. *HTML-komponentti*: Gatsbyssa on vain yksi HTML-komponentti, ja se sijaitsee osoitteessa src/html.js. Tiedostoa html.js käytetään esimerkiksi <head> -osion metatietojen lisäämiseen. (So 2021, kappale 1.1.1.) Tämä työ käy läpi paremman tavan lisätä metatietoja aiemmassa osiossa 3.1.6: *SEO*.

4. *Osiokomponentti*: sivulla käytettävä komponentti, joka on sivuston osio, kuten esimerkiksi sivun yläpalkki (engl. *header*) tai alapalkki (engl. *footer*). (So 2021, kappale 1.1.1.)

Kuviossa 7 on oma esimerkkini Gatsbyyn luodusta alasivusta src/ota-yhteytta.js, joka rakentuu sivustolle automaattisesti osoitteeseen esimerkki.com/ota-yhteytta:

```
import React from 'react'

import Footer from '../components/footer'
import Header from '../components/header'

const OtaYhteytta = () => {
    return(
        <>
            <Header/>

            <main>
                <h1>Ota yhteyttä</h1>
                <p>Hei! Voit ottaa meihin yhteyttä
puhelimitse...</p>
            </main>

            <Footer/>

        </>
    )
}
```

```
export default OtaYhteytta
```

Kuvio 7. Esimerkki sivukomponentista ota yhteyttä -sivulla.

Yllä olevassa kuviossa 7 on *sivukomponentti*, joka hyödyntää kahta *osiokomponenttia*: <Header /> ja <Footer />, jotka ovat sivustolla toistuva ylä- ja alaosa.

Tiedosto alkaa tuomalla Reactin, sillä Gatsbyn sivut vaativat Reactin toimiakseen.

```
import React from 'react'
```

Seuraavaksi tapahtuu osiokomponenttien “Footer” ja “Header” tuominen. Ne sijaitsevat kansiossa src/components/.

```
import Footer from '../components/footer'
import Header from '../components/header'
```

Sen jälkeen alkaa sivukomponentin ja sen sisältämän sisällön mainitseminen:

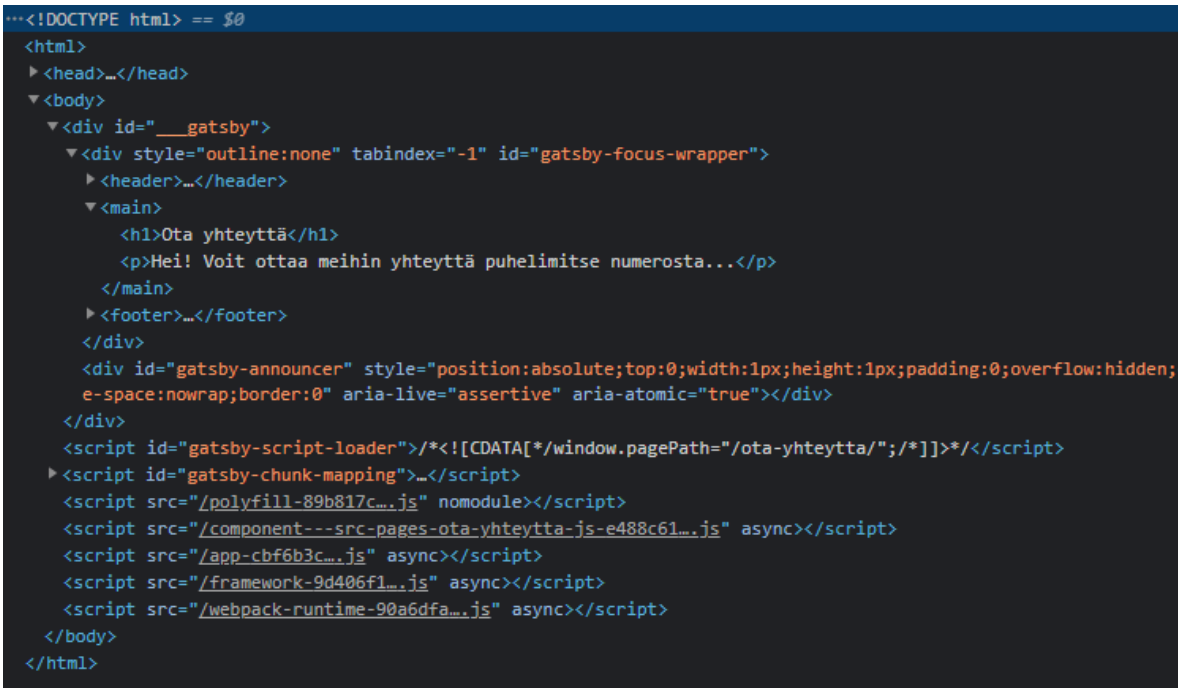
```
const OtaYhteytta = () => {
    return(
        <>
            <Header/>

            <main>
                <h1>Ota yhteyttä</h1>
                <p>Hei! Voit ottaa meihin yhteyttä
puhelimitse...</p>
            </main>

            <Footer/>

        </>
    )
}
```

Sivukomponentin osio sisältää sivun sisällöt ja mahdolliset muut komponentit, tässä tapauksessa komponentit <Header /> ja <Footer />.

Lopulta tiedosto merkitään vietäväksi sivustolle käyttäen:

```
export default OtaYhteytta
```

Seuraavaksi alla tuleva kuvio 8 näyttää, miltä yllä olevan koodin tuottama Gatsbyn HTML-rakenne näyttää rakennusprosessin jälkeen.



Kuvio 8. Osiossa 4.3 mainitun React-koodin HTML-rakenne. Kuvassa näkyvät muut elementit ovat Gatsbyn itse lisäämiä. Tältä näyttää yleinen Gatsbyn HTML-rakenne, käytetty Gatsbyn versio on v2.

4.4 GraphQL, API:t ja CMS-palvelut

Yksi Gatsbyn vahvuuksista on se, että GraphQL tulee valmiiksi asennettuna Gatsbyn sisällä. Käyttämällä GraphQL:ää voi pyytää tarvittavia tietoja ulkoisesta palvelimesta, jolloin kyseinen palvelin silloin lähettää pyydetyt tiedot. Tämä mahdollistaa datan tuonnin ulkoisista lähteistä, esimerkiksi ulkoisesta CMS-palvelusta (So 2021, kappale 1.1.2).

Ulkoisen datan keräämisessä Gatsbyyn tulevat apuun *source-pluginit*, eli lähteistä dataa noutavat pluginit. Source-pluginit toimivat välikätenä ulkoisen lähteen ja Gatsbyn välillä, siirtäen saadun tiedon käytettäväksi koodiksi. Gatsbylla on suuri määrä esimerkiksi API:hin, CMS:n, kauppajärjestelmiin ja jopa lokaaleihin tiedostoihin keskittyviä source-plugineita. (So 2020, Chapter 1.)

Oman esimerkkini GraphQL:n toiminnasta näet kuviosta 9 alla.

```
const data = useStaticQuery(graphql`

query{
  allContentfulSuljin(
    sort: {
    fields: published,
      order: DESC
    }
    limit: 15,
  ){
    edges{
      node{
        title
        slug
        published(formatString:"D.M.YYYY")
        thumbnail {
          file {
            url
          }
          description
        }
      }
    }
  }
}
`)
```

Kuvio 9. Esimerkki GraphQL:n toiminnasta Gatsbyssa.

Yllä oleva GraphQL-koodi on Suljin.fi -sivustosta, josta tulen kertomaan lisää toiminnallisessa osiossa. Tämä koodi tuo sisältöä Contentful-palvelusta, joka on *headless CMS*, eli sivuston ulkopuolella toimiva sisällönhallintajärjestelmä. Tiedonhaun Contentful-palvelusta mahdollistaa plugini `gatsby-source-contentful`. Kyseinen koodi hakee 15 uusinta artikkelia Contentfulista aikajärjestyksessä ja kerää niistä seuraavat tiedot: otsikko (*title*), lyhytkoodi (*slug*), julkaisuaika (*published*), artikkelikuvan URL (*thumbnail → file*

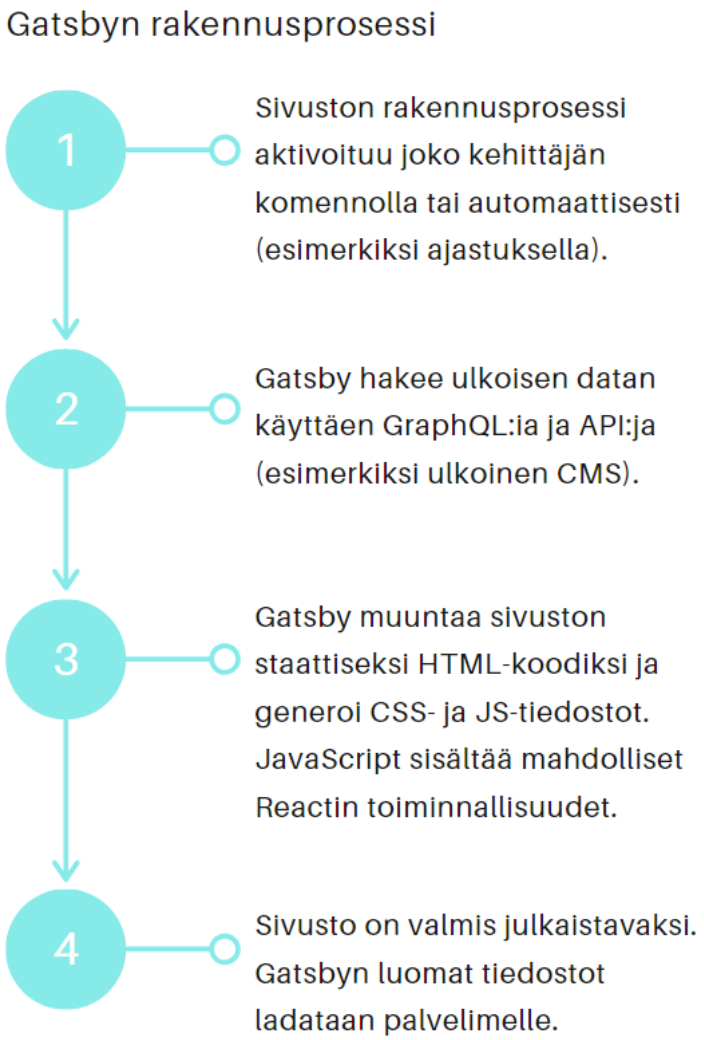
→ *url*) ja artikkelikuvan alt-teksti (*thumbnail* → *description*). Ulkoisen tiedonhaun avulla on mahdollista käyttää saatua dataa omalla sivustollaan, tässä esimerkissä sivusto käyttää sitä uusimpien artikkelien listaamiseen etusivulla.

Onko GraphQL pakollinen Gatsbyn käyttöön? Tähän kysymykseen vastasi Gatsby Conf 2021 -tapahtumassa Gatsby-tiimiin kuuluva Dustin Schau. Schaun mukaan se on yleinen harhaluulo, että Gatsby vaatisi GraphQL:n, mutta hänen mielestään Gatsby toimii paremmin sillä (Gatsby Conf 2021 live-tapahtuma, päivä 1).

4.5 Rakennusprosessi ja sivuston julkaisu

Gatsbyssa sivuston rakennusprosessi (*engl. build process*) luo optimoidun, julkaisuun tarkoitetun version sivustosta. Tämä eroaa Gatsbyn kehitysympäristöstä niin, että verkkokehittäjälle tarkoitettu kehitysympäristö ei ole optimoitu nopeuteen, vaan se on tarkoitettu nopeaan kehittämiseen. Rakennusprosessin läpi mennyt sivusto on sen sijaan tarkoitettu sivuston julkaisuun, eli se on optimoitu nopeuteen ja hyvään käyttäjäkokemukseen. (Gatsby n.d. h.)

Rakennusprosessi generoi sivuston staattisiksi tiedostoiksi kansioon `public` projektikansiossa. Rakennusprosessi käyttäen komentoa `gatsby build`. Rakennetun sivuston esikatselu tapahtuu komennolla `gatsby serve`, jonka jälkeen sivustoa voi käyttää lokaalissa ympäristössä osoitteessa `http://localhost:9000`. (Gatsby n.d. h.)



Kuvio 10. Aikajana Gatsbyn rakennusprosessista.

Rakennusprosessin jälkeen on aika julkaista sivusto. Olen huomannut, että Netlify on suosituin julkaisualusta Gatsby-sivustolle. Kaksi muuta yleistä alustaa ovat GitHub Pages ja Gatsby Cloud. Kaikista näistä kolmesta palvelusta löytyy myös ilmainen versio (tarkastettu sivuilta: Gatsby Cloud: gatsbyjs.com/docs/deploying-and-hosting, GitHub Pages: pages.github.com, Netlify: netlify.com/pricing). On suhteellisen harvinaista nähdä, että joku lataa sivuston omalle palvelimelleen, mutta tämä on kokemukseni yksittäisten kehittäjien kanssa – suuret yritykset saattavat tehdä toisin.

Sivuston julkaisussa pystyy Gatsbylla käyttämään jatkuvaa julkaisua (engl. *continuous deployment*), jonka avulla sivusto julkaisee itseään automaattisesti sivuston päivittämisen ohella. Kehittäjän on mahdollista asettaa valitsemansa

palvelun (kuten Netlifyn) rakentamaan sivuston automaattisesti esimerkiksi jokaisen uuden GitHub-koodin työntämisen (engl. *code push*) jälkeen. Näin kehittäjän ei tarvitse mennä itse rakentamaan sivustoa komentojen kautta tai nappia painamalla, vaan palvelu tekee sen itsestään. (Gatsby n.d. i).

5 Kuinka loin sivuston Gatsbylla

Nyt alkava opinnäytetyön toiminnallinen osuus kertoo, kuinka loin verkkosivuston Suljin.fi käyttäen Gatsbya. Kutsun sivustoa Suljin.fi tästä edespäin yksittäisellä sanalla Suljin helpomman lukukokemuksen saavuttamiseksi.

Suljin on verkkosivusto, joka pitää sisällään artikkelisisältöä valokuvauksesta ja virtuaalisen valokuvanäyttelyn. Sivuston etusivu näyttää uusimmat artikkelit, joita klikkaamalla päätyy itse artikkeliin. Valokuvanäyttely (suljin.fi/valokuvanayttely) on kuukausittain vaihtuva näyttely, johon sivulla kävijät voivat lähettää kuviaan. Kaikki sivuston artikkelit käyttävät samaa ulkoasua ja generoituvat automaattisesti.

Sulkimen artikkelisisältö tulee Contentfulin kautta, joka on kolmannen osapuolen palvelu CMS-toiminnallisuuden luontiin sivustolle, jolla sellaista ei ole. Sulkimen palvelintilana toimii AWS (Amazon Web Services), sekä testiympäristö toimii Netlifyllä. Sivusto on tehty ilman valmista teemaa, eli kaikki JSX (Reactin koodi) ja CSS ovat itse luomiani. Plugineita hyödynsin aina silloin, kun koin sen parhaaksi.

Sivuston koodin pääpaino on käyttäjäkokemuksella, nopeudella, hakukoneoptimoinnilla ja skaalautuvuudella. Eli hyvinkin niillä alueilla, joita tämän työn luku 3.1: *Mitä ongelmia Gatsby ratkaisee* piti sisällään.

5.1 Miten hyödyn Gatsbysta

Olin alun perin luomassa Sulkimen verkkosivut käyttämällä WordPressiä, mutta minulla oli jatkuva tunne siitä, että palvelinpuolella toimiva sivusto olisi liikaa tarkoituksellleni. Halusin vain sivuston, jolle voi julkaista helposti uusia artikkeleita, ja jonka ylläpito olisi mieluiten edullista. Kaikki dynaamiset palvelinpuolen sivujen generoinnit olisivat vain turhaa palvelimen prosessoinnin käyttöä, sillä en toimittaisi kävijöille mitään yksilöllistä sisältöä tai suurempaa palvelinpuoleen kohdistuvaa toiminnallisuutta. Kaikki sivun sisältö näkyisi jokaiselle kävijälle samalla tavalla. Näin päätin haluavani staattisen sivuston, ja sitä kautta päädyin Gatsbyyn.

Gatsbylla on mielestäni juuri kaikki tarvittava hyvän artikkeli- tai blogisivuston tekemiseen: uudelleenkäytettävät React-komponentit, CMS-palvelut, lukuisat pluginit, lukemattomia ohjeita (joko Reactiin tai Gatsbyyn liittyen) ja mahdollisuus ylläpitää sivustoa lähes millä tahansa palvelimella. Pystyn myös versioimaan ja ylläpitämään sivuston koodia suoraan GitHubissa, sekä jos haluan ladata sivuston kehitysympäristön toiselle koneelle, varmistan vain, että toisella koneella on valmiina ohjelmointiympäristöni Visual Studio Code ja Gatsbyn vaatima Node.js, jonka jälkeen voin ladata sivustoni koodin suoraan GitHubista. Koodin ladattuani komento `npm install` lataa Gatsbyn, sen riippuvuudet ja kaikki käytettävät pluginit.

Sillä Suljin on valokuvaukseen perustuva sivusto, pitää se sisällään artikkeleissaan huomattavia määriä valokuvia. Tämän vuoksi kuvien optimointi ja monen eri resoluution kuvan luominen on sivuston korkean nopeuden ylläpidon vuoksi kriittistä. Tässä Gatsby loistaa versiossa v3 tulleella pluginillaan `Gatsby-Plugin-Image`, joka optimoi sivuston kuvat automaattisesti ja luo jokaisesta kuvasta monta eri versiota eri resoluutioilla ja tiedostomuodoilla (Warren 2021). Jokainen sivuston valokuva käy läpi optimoinnin, joka mahdollistaa varsinkin mobiilikäyttäjien hyvän käyttäjäkokemuksen sivuston kuvien kanssa. Lisää tietoa kuvien optimoinnista oli työn luvussa 3.1.1: *nopeus*.

Seuraavaksi tulevat kuviot 11–13 näyttävät kuvien optimoinnin visuaalisesti tuoden oikeaa dataa Sulkimen artikkelin kuvista.

Kuvio 11. Sivuston artikkeliin "Kuun kuvaaminen" ladatun kuvan tiedostokoko on 663 kB. Tiedostomuoto on JPG. Prosessointi ei ole vielä tapahtunut.

Kuvio 12. Tietokoneella artikkelia lukiessa kuvan tiedostokoko on 74 kB. Kuva on optimoitu ja tiedostomuoto on WebP. Kuvan voi klikata suuremmaksi, jolloin tiedostokoko on 386 kB.

Kuvio 13. Puhelimella kuvan tiedostokoko on 23 kB ja tiedostomuoto on WebP. Kuvan voi painaa suuremmaksi, jolloin kuvan koko on 386 kB. Käyttäjä ei todennäköisesti huomaa suurta eroa kuvanlaadussa, sillä puhelimen näyttö on pienempi kuin tietokoneen näyttö.

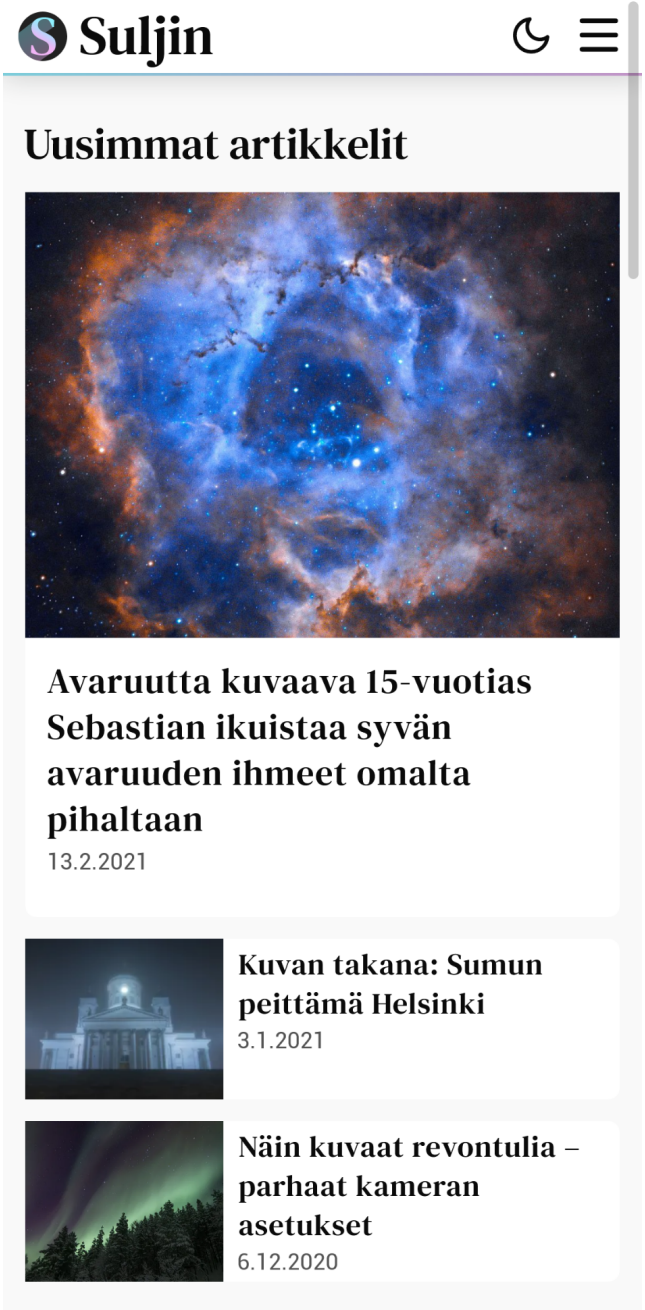
Yllä olevissa kuvioissa 11–13 tulee ilmi Gatsbyn automaattisesti mahdollistama kuvien optimoinnin suuruus. Kuva pieneni tiedostokoosta 663 kB kokoon 74 kB tietokoneella ja 23 kB mobiililla. Optimointi tapahtui ilman huomattavia kuvanlaadun menetyksiä, kunhan huomioi minkä kokoisilla näytöillä kuvaa katsellaan tai minkä kokoisena kuva siinä ilmenee.

5.2 Miten sivusto on rakennettu

Suljin on valmistettu hyödyntäen React-komponentteja, sivupohjia ja GraphQL-tiedosteluja. GraphQL-tiedostelut tulevat pluginin `gatsby-source-contentful` avulla Contentful-palvelusta, jossa sivun artikkelien luominen tapahtuu tutun käyttöliittymän pitävällä tekstinkäsittelyohjelmalla. Kaikki sivuston toistuvat osiot ovat React-komponentteja, kuten sivuston ylä- ja alaosa. Eri sivupohjia on vain muutama: etusivu, artikkeli, valokuvanäyttely ja näyttelyyn ilmoittautuminen.

Etusivu

Sulkimen etusivu on kirjoitettu normaalina HTML-koodina, mutta se pitää sisällään yhden Gatsbyssä generoituvan osion: lista uusimmista artikkeleista. Gatsby itse generoi listaan artikkelin otsikkokuvan, otsikon ja julkaisupäivämäärän hyödyntämällä GraphQL:ia ja Reactin ehdollista renderoimista. Tämä mahdollistaa sen, että jokaisen uuden julkaistun artikkelin kohdalla HTML-koodiin ei tarvitse mennä käsin lisäämään uutta artikkelia, vaan Gatsby tekee sen itsestään. Tämä tapahtuu Sulkimessa hakemalla uusimmat artikkelit Contentful-palvelusta GraphQL:n avulla. Tästä toiminnosta näet seuraavaksi esimerkin, sillä koen tämän toiminnon olevan yksi Gatsbyn suurimmista hyödyistä.



Kuvio 14. Kuvankaappaus mobiiliselaimella Sulkimen etusivulta (suljin.fi). Kuvassa näkyy kolme uusinta artikkelia.

Kuten yllä olevasta kuviosta 14 näet, etusivulla olevat artikkelinostot ovat eri kokoa. Ensimmäisestä artikkelista lähtien joka kolmas artikkeli on muita suurempi – tämä tuo mukavaa vaihtelua etusivun ulkoasuun. Tämä onnistuu käyttäen Reactin tekniikkaa *conditional rendering* eli ehdollinen renderointi. Seuraavaksi tuleva kuvio 15 näyttää, millainen etusivulle artikkelit tuova koodi on. Kuviota seuraa selitys eri koodin kohdista.

```
<div className="uusimmat-artikkelit">

{data.allContentfulSuljin.edges.map((edge, index) => {

    const artikkeli = edge.node;

    return(

        <article className={`${index % 3 === 0 ?
'suuri-artikkeli':'pieni-artikkeli'}`}>

          <Link to={`/${artikkeli.slug}/`}>

            <GatsbyImage className="artikkeli-thumbnail"
loading={`${index == 0 ? 'eager':'lazy'}`}

              image={index % 3 === 0 ?
artikkeli.thumbnail.localFile.childImageSharp.suuriKuva :
artikkeli.thumbnail.localFile.childImageSharp.pieniKuva
                    }

              alt={artikkeli.thumbnail.description} />

            <div className="artikkeli-otsikko">

              <h3>{artikkeli.title}</h3>

              <time
dateTime={artikkeli.publishedCode}>{artikkeli.published}</time>

            </div>

          </Link>

        </article>

    )

})}

</div>
```

Kuvio 15. Tämä koodi lisää Sulkimen etusivulle uusimmat artikkelit.

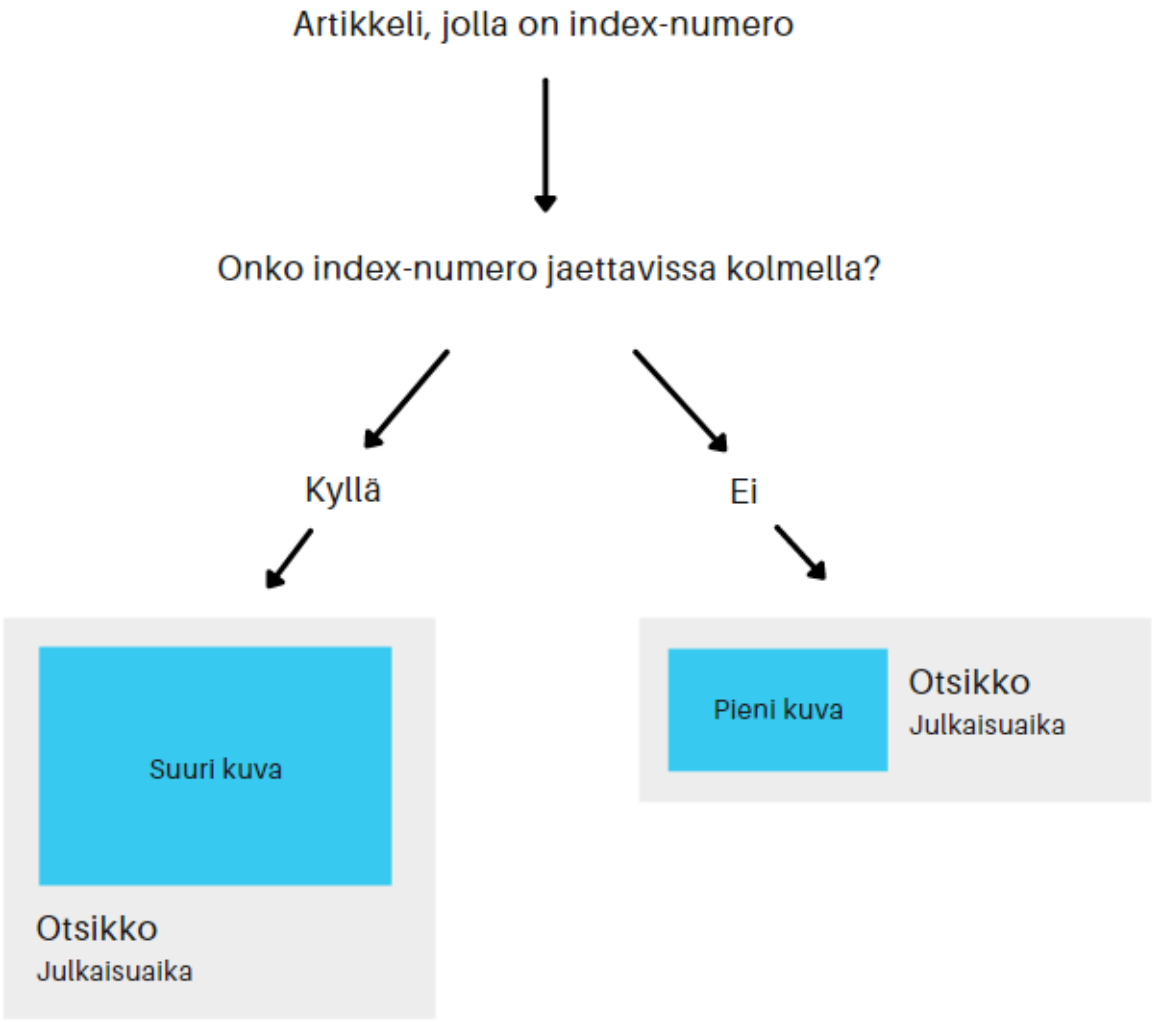
Yllä olevassa kuviossa 15 uusimmat artikkelit listautuvat `<div>` -elementin sisälle käyttäen seuraavaa lausetta: `{data.allContentfulSuljin.edges.map((edge,`

index) => { … }. Tämä komento toistaa sisällään olevan koodin jokaisen artikkelin kohdalla. Mukana on myös sana *index*, joka alkaa nollasta, ja nousee yhdellä aina uuden artikkelin kohdalla, eli 0 -> 1 -> 2 ja niin edelleen. Kaikki koodin { } -muodoissa olevat lähteet (kuten {artikkeli.title}) osoittavat GraphQL:n kautta Contentful-palveluun.

Koodi alkaa <article> -artikkelielementillä, joka pitää sisällään Reactin ehdollista renderoimistä. Ehdollisessa lauseessa oleva index % 3 === 0 tarkistaa onko artikkelin saama index-numero jaollinen kolmella, ja jos on, antaa sille luokan “suuri-artikkeli”, ja jos ei ole, antaa sille luokan “pieni-artikkeli”. Tämä tuo mobiilinäkymään mahdollisuuden näyttää joka kolmas artikkeli suurempana hyödyntäen Gatsbyn sisäänrakennettua Reactia, mutta silti saaden itse HTML-koodin staattisena. Tässä osiossa myöhemmin tuleva kuvio 16 antaa visuaalisen esityksen ehdollisesta renderoinnista.

Seuraavana oleva <Link> -elementti linkittää artikkeliin käyttäen GraphQL:n kautta tulevaa *slugia*, eli lyhytkoodia, joka on artikkelin URL-osoitteessa. Gatsbyn <Link> -elementti eroaa normaalista <a> -elementistä niin, että Gatsby lataa omilla linkkielementeillään sivuja ennakkoon jo valmiiksi jos käyttäjä näyttää tulevan klikkaamaan linkkiä (Gatsby n.d. b). Lisää <Link> -elementistä näet tämän työn osiosta 3.1.1: *nopeus*.

Artikkelikuva tulee pluginin Gatsby-Plugin-Image kautta elementtiin <GatsbyImage>, jonka avulla jokainen kävijä vastaanottaa näytön leveydelleen sopivan resoluution kuvan (Gatsby n.d. a). Kuvassa on mukana Reactin ehdollinen renderoiminen, jotta joka kolmas suurempi artikkelinosto saisi kuvaan suuremman resoluution kuin pienemmät nostot. Tässäkin tapahtuu ehdollinen koodi index % 3 === 0, mutta tällä kertaa jos artikkelin index-numero on jaettavilla kolmella, hakee koodi GraphQL:sta suuremman kuvatiedoston, ja jos numero ei ole jaettavissa kolmella, hakee se pienemmän kuvatiedoston. Alla oleva kuvio 16 antaa visuaalisen esityksen ehdollisesta renderoinnista.



Kuvio 16. Visuaalinen esitys tapahtuvasta ehdollisesta renderöinnistä.

Artikkelin otsikko tulee suoraan GraphQL:n kautta `<h3>` -otsikkoelementtiin ja julkaisuaika tulee `<time>` -aikaelementtiin. Julkaisuaika tulee kahdessa eri muodossa: artikkelissa visuaalisesti näkyvä aika on suomalaisille tutussa *DD.MM.YYYY* -muodossa (esimerkiksi *24.12.2021*) ja roboteille tarkoitettu aika on elementin koodin sisällä *YYYY-MM-DD* -muodossa (esimerkiksi *2021-12-24*).

Lopulta kaikki tämä koodi generoituu Gatsbyn rakennusprosessissa staattiseksi HTML-koodiksi, eli mitään yllä olevaa JavaScriptia ei päädy sivustolla kävijän selaimelle ratkottavaksi. Tämä on mielestäni yksi suurimmista Gatsbyn vahvuuksista.

Artikkeli

Sulkimen artikkelit toimivat kaikki käyttäen samaa sivupohjaa, joka pitää sisällään osion jokaiselle tarvittavalle artikkelin elementille. Kuten etusivulla, käyttää myös artikkelisivu GraphQL:ia Contentfulin sisällön lisäämiseen artikkeliin.

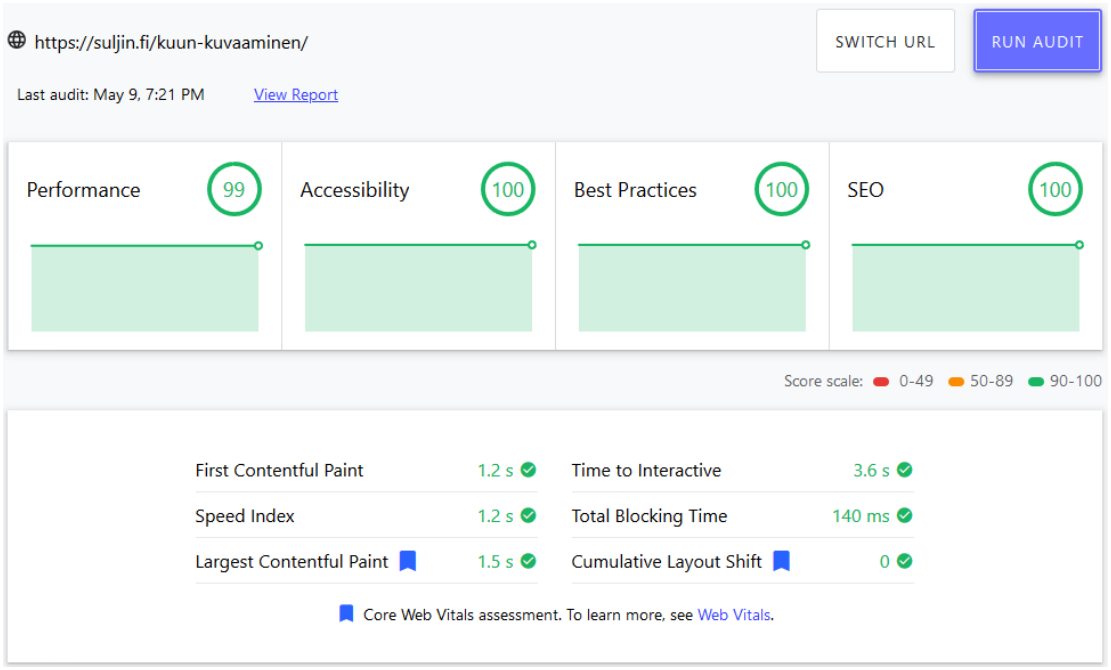
Artikkelin teksti ja kuvat tulevat GraphQL:n kautta *Rich Text* -muodossa, eli rikkaassa tekstimuodossa, joka pitää sisällään kaikki tekstin muotoilut, otsikoinnit ja kuvat. Tämä mahdollistaa tekstin muotoilun – kuten tekstin kursivoinnin ja lihavoimisen – tekemisen itse tekstieditorissa Contentfulissa, josta Gatsby vastaanottaa sen JSON-muodossa ja lisää sen suoraan artikkelin sivupohjaan HTML-muodoksi (Contentful n.d., Rich Text). Tämän lisäksi myös artikkelin otsikko ja artikkelikuva tulevat suoraan artikkeliin Contentfulista GraphQL:n kautta, eli uuden artikkelin julkaisu ei vaadi ollenkaan sivun koodiin koskemista tai muuta manuaalista toimintaa.

Sulkimen artikkelit pitävät sisällään metatiedot ja jäsennellyn datan, joista tämän työn osio 3.1: *Mitä ongelmia Gatsby ratkaisee* alaotsikko *SEO* kertoo lisää. Metatiedot mahdollistavat linkin esikatselun näkymisen sosiaalisen median julkaisuissa, joka on näkyvyyden kannalta hyödyllistä. Artikkelit pitävät sisällään myös jäsennellyn datan, sillä se auttaa hakukoneiden robotteja ymmärtämään sivuja, sekä Google suosittelee jäsennellyn datan käyttöä aina, kun mahdollista (Google 2021b).

Jokainen Sulkimen artikkeli hyödyntää pluginia `gatsby-plugin-image` kuvien optimointiin, jotta sivusto pysyisi nopeana. Tämä on käyttäjän kannalta erittäin miellyttävää, mutta teknisen hakukoneoptimoinnin kannalta myös kriittistä. Google on tehnyt selväksi, että sivuston nopeus on yksi kriteeri sivuston näkyvyydelle Google-hauissa (Osmani & Grigorik 2018). Googlen mukaan nopeampi sivun lataus johtaa käyttäjien sitoutumisen nousuun ja korkeampiin tuottoprosentteihin. Sivuston nopeuden arvioimiseen Google suosittelee omaa

Lighthouse-palveluaan, joka antaa ehdotuksia sivuston optimoimiseen. (Osmani & Grigorik 2018.) Lighthouse -palvelussa on kaksi nopeuden pisteytystä, joihin Gatsbyn plugini gatsby-plugin-image antaa huomattavan edun: suurin ensilatauksen elementti (engl. *largest contentful paint*) ja sivun elementtien liike (engl. *cumulative layout shift*). Tässä osiossa myöhemmin tulevasta kuviosta 17 näet esimerkin Lighthouse-palvelun antamasta arviosta Sulkimen artikkeliin.

Sulkimen artikkeleissa suurin ensilatauksen elementti on artikkelin alussa näkyvä artikkelikuva, sillä se näkyy heti käyttäjän ladatessa sivun ja se on ensinäkymän pisimmän latausajan vaativa elementti. Sulkimessa Gatsbyn plugini gatsby-plugin-image optimoi artikkelikuvan, eli pienentää kuvan tiedostokokoa ja siten mahdollistaa sille nopeamman latausajan kävijän selaimessa – eli nopeuttaa sivun suurimman ensilatauksen elementtiä, parantaen Lighthousen arviointia. Artikkelikuva on myös suurin vaikuttaja arvioinnissa olevaan sivun elementtien liikkeeseen. Jos sivuston kehittäjä ei anna tietoa kuvan koosta selaimelle, liikuttaa kuvan lataus sivuston sisältöä. Sulkimen artikkelikuvien kohdalla Gatsbyn plugini gatsby-plugin-image jättää automaattisesti tyhjää tilaa artikkelikuvan kohdalle, jonka avulla artikkelin tekstisisältö ei liikahda ollenkaan kuvan ladattua. Tämä nostaa Lighthousessa olevan sivun elementtien liikkeiden arvosanaa huomattavasti.



Kuvio 17. Lighthouse-palvelun arvio Sulkimen artikkelista. Arviointi saatu käyttämällä Googlen palvelua Web.dev osoitteessa https://web.dev/measure/. Kuvassa alla olevat arvot toimivat sivunopeuden kriteereinä.

Valokuvanäyttely

Suljin pitää sisällään avoimen kuratoidun valokuvanäyttelyn, johon sivuston kävijät voivat lähettää itse omia valokuviaan. Itse valokuvanäyttely toimii samalla tavalla kuin etusivu ja artikkelipohja, sillä sen sisältö tulee GraphQL:n kautta Contentful-palvelusta, jonka jälkeen plugini `gatsby-plugin-image` optimoi GraphQL:n kautta vastaanotetut kuvat. Näyttelyssä on silti yksi ongelma, jota Gatsby ei voi itsestään korjata: näyttelyyn ilmoittautumista varten sivulla täytyy olla ilmoittautumislomake, johon kävijä lisää kuvansa ja yhteystietonsa. Tätä varten sivusto tarvitsee palvelinpuolen prosessointia ja tietokannan, eli dynaamista toimintaa, jota staattinen sivusto ei pidä sisällään (Camden & Rinaldi 2017, Chapter 1). Dynaamisen ja staattisen sivun eroista näet lisää tämän työn osiossa 2.2: *Staattisen verkkosivun heikkoudet*. Tässä onneksi kolmannen osapuolen palvelut tulevat pelastukseen.

On olemassa monia kolmannen osapuolen palveluita, joilla voi lisätä dynaamisia toimintoja staattisille sivuille. Yksi suosituimmista toiminnoista on lomakkeiden lähettäminen, joka on hyvinkin yleinen esimerkiksi yritysten nettisivuilla. Gatsby itse mainitsee kaksi lomakkeita tarjoavaa palvelua sivuillaan: Netlify Forms ja Formspree (Gatsby n.d. j). Kummatkin mainitut palvelut antavat helpon tavan vastaanottaa uudet sivulla tapahtuvat yhteydenotot esimerkiksi sähköpostilla.

Jos olisin valinnut Sulkimen palvelimeksi Netlifyn, olisin käyttänyt heidän tarjoamaa Netlify Formsia. Suljin toimii AWS:llä, eli Amazonin pilvipalvelussa, joten päätin käyttää AWS:n omaa palvelinpuolen prosessoinnin palvelua hyödykseni: Lambdaa. Lambda on Amazonin palveliton koodiympäristö, eli sen avulla pystyy suorittamaan koodia ilman omaa palvelinta (AWS n.d. a). Asetin Lambdan tallentamaan käyttäjien täyttämät ilmoittautumislomakkeet, jotta voin käydä niitä läpi myöhemmin. Käytän AWS, koska heidän CDN-palvelussa CloudFrontissa on palvelin Helsingissä (AWS n.d. b). Tämä mahdollistaa sen, että kaikki Suomessa asuvat kävijät vastaanottavat koko sivun täysin Suomen sisältä, saaden suurimman hyödyn irti CDN:stä.

5.3 Mielipide Gatsbylla työskentelystä

Mielestäni Gatsby oli juuri täydellinen väline Sulkimen nettisivujen luontiin. Gatsbyn sivujen ennakkoon lataaminen, kuvien optimointi ja staattinen koodi mahdollistavat erittäin nopeasti toimivan verkkosivun luonnin.

Gatsbyssa on mielestäni huomattavasti enemmän hyviä puolia kuin huonoja puolia. Hyviä puolia ovat sen nopeus, pluginien ekosysteemi, valmiit teemat, lukuisat ohjeet, monet kolmannen osapuolen palvelut ja hyvä dokumentaatio. Huonoja puolia ovat olleet epäselvät ongelmat, jotka estävät sivuston rakentumisen, sekä hieman GraphQL:n epäselvyys Gatsbyssa. Lopulta jokaisen ongelman on kuitenkin voinut ratkaista etsimällä virhekoodeilla ja Google-hauilla internetistä tietoa.

Tekniseltä kannalta Gatsbyn sisäänrakennettu React, GraphQL ja pluginit ovat suureksi hyödyksi sivun rakentamisessa ja kehittäjän positiivisessa kokemuksessa. Jos sivuston täytyy hakea ulkoista dataa esimerkiksi CMS:tä, onnistuu se erittäin helposti Gatsbylla sen jälkeen, kun kehittäjä alkaa ymmärtämään GraphQL:n perusteita.

Ulkoisen CMS:n käyttö mahdollistaa myös Gatsbyn soveltuvuuden sellaisissa asiakasprojekteissa tai yrityksen sisäisissä projekteissa, joissa markkinointitiimin tai sisällöntuottajien täytyy pystyä itse muuttamaan sivun sisältöä ilman ymmärrystä koodista. Esimerkiksi ulkoinen CMS-palvelu *Prismic* mahdollistaa “raahaa ja pudota” -tyylisen (engl. *drag and drop*) sivujen luonnin käyttäen komponentteja (Prismic n.d.). Tämä tapahtuu niin, että ensiksi komponentit luodaan Prismicissä, jonka jälkeen kehittäjä antaa komponenteille HTML-, CSS- ja JS-rakenteen Gatsbyssa. Gatsby sitten hakee Prismicin sisällön ja käytetyt komponentit käyttäen pluginia `gatsby-source-prismic`. Mikä Prismicin ja Gatsbyn välisessä komponenttien käytössä on erikoista on se, miten Prismicin sisällä voi luoda sivun normaalia CMS-yhteyttä vahvemmilla muokkausmahdollisuuksilla. Yleensä kehittäjä tekee valmiin sivupohjan ja määrittää jokaiseen pohjan alueeseen CMS:n yhteyden, jonka jälkeen sisällöntuottaja vain muokkaa eri alueiden tekstejä tai kuvia pohjan pysyessä samana. Prismicin avulla kehittäjä luo pohjan sijaan käytettäviä komponentteja, jonka jälkeen sisällöntuottaja voi itse CMS:n avulla luoda sivun sisältöjä järjestäen komponentit mihin tahansa järjestykseen ja käyttäen mitä tahansa saatavilla olevia komponentteja. Tämä mahdollistaa sivupohjasta siirtymisen komponenttikirjastoon, jonka avulla sisällöntuottaja voi itse määrittää sivun ulkoasua. Tämä tieto perustuu siihen, että loin juuri asiakkaalle Gatsbya ja Prismiciä käyttävän sivun, jossa asiakas voi ilman koodituntemusta luoda itse sivuja Gatsbyyn käyttäen komponentteja.

Gatsby on lopulta vain väline sivun luontiin. Mielestäni kehittäjälle tarkoitetun välineen on tarkoitus joko nopeuttaa sivuston luontia tai tehdä sivustosta parempi. Niistä Gatsby onnistuu kummassakin. Gatsby tekee staattisen sivuston luomisesta kehittäjälle helpompaa ja skaalattavampaa käyttäen

Reactin komponentteja, plugineita ja valmiita teemoja. Tämän lisäksi Gatsby tekee sivustosta myös käyttäjälle paremman lataamalla sivulla olevia linkkejä ennakkoon, generoimalla staattista HTML:ää ja optimoimalla kuvia automaattisesti. Näin ollen mielestäni Gatsby tekee juuri sen, mitä sen kuuluukin tehdä: parantaa niin sivuston kehittäjän kuin kävijöidenkin kokemusta.

6 Johtopäätökset

Gatsby on moderni ohjelmistokehys, jolla vaikuttaisi olevan vahva tulevaisuus edessään. Tähän tämän työn perusteella vaikuttavat jatkuvasti laajeneva Gatsbyn pluginien ja valmiiden teemojen ekosysteemi, uudet Gatsbyn päivitykset, jatkuvasti laajeneva käyttäjämäärä ja Gatsbyn mahdollistama nopeampi sivujen kehittäminen.

Gatsbya käytetään staattisen sivun luomiseen hyödyntäen Reactia, GraphQL:ia ja Gatsbyn rakennusprosessia. Staattiset sivustot ovat yleisesti ottaen nopeita ja turvallisia, sillä ne eivät käytä palvelinpuolen prosessointia tai tietokantaa, vaan sivustolla kävijä vastaanottaa sivun suoraan palvelimelta staattisina HTML-, CSS- ja JS-tiedostoina omalle selaimelleen. Tämä mahdollistaa myös sen, että sivulla kävijä voi vastaanottaa koko sivuston CDN:n eli sisällönjakeluverkon sisältä, mukaan lukien sivuston HTML-tiedostot.

Gatsbylla on neljä erityistä hyötyä: nopeus, kehittäjäkokemus, turvallisuus ja saavutettavuus (So 2021, Chapter 1). Tämän lisäksi suuria lisähyötyjä ovat vahva hakukoneoptimointi ja mahdollisuus tuoda sisältöä ulkoisesta CMS:tä eli sisällönhallintajärjestelmästä. Nopeudessa Gatsby onnistuu hyödyntämällä linkkien ennakkoon lataamista, kuvien automaattista optimointia ja staattista HTML:ää. Hyvä kehittäjäkokemus onnistuu Gatsbyn pluginien ja teemojen ekosysteemillä, suurilla määrillä ohjeita ja dokumentaatioita sekä automaattisesti päivittyvällä lokaalilla ympäristöllä (engl. *hot reloading*). Gatsbylla tehty sivusto on turvallinen, sillä useimmiten sivuston turvallisuus murretaan palvelinpuolen toimintojen ja tietokannan kautta, mutta Gatsbylla ei

ole kumpaakaan niistä, sillä se generoi staattisia sivustoja. Gatsbyn kehittäjillä on saavutettavuus mielessään tehdessään Gatsbya, sekä he tavoittelevat WCAG 2.1 AA -saavutettavuuskriteeriä, jossa omien sanojensa mukaan he ovat jo osittain onnistuneet (Gatsby n.d. c.). Gatsbyssa vahvan hakukoneoptimoinnin mahdollistaa staattinen HTML, Gatsbyn nopeus sekä helposti luotavat sivuston metatiedot ja jäsennelty data (engl. *structured data*). Ulkoinen sisällönhallintajärjestelmä mahdollistaa sen, että sisällön kirjoittamiseen voi käyttää ulkoista tekstieditoria kuten Contentfulia tai Prismicia, joista Gatsby hakee sisällön rakennusprosessissaan käyttäen GraphQL:ia.

Gatsby on parhaillaan käytettynä sellaisella sivustolla, jota ei päivitetä usein, ja jonka halutaan olevan nopea ja hakukoneystävällinen. Tällaisia sivustoja ovat esimerkiksi blogi-, portfolio-, uutis- ja yrityssivustot. Näiden sivujen sisältö muuttuu yleensä niin harvoin, että sisältö kannattaa generoida Gatsbyn tavoin rakennusvaiheessa dynaamisen sivuston pyyntökohtaisen tavan sijasta (Väänänen 2019, 13–14).

Gatsby ei sovellu sivustoille, joiden sisältö päivittyy usein (esimerkiksi kymmenen sekunnin välein), tai sellaisille sivustoille, joiden tarvitsee tarjoilla yksittäisille kävijöille uniikkia sisältöä (kuten rekisteröitymisen ja kirjautumisen takana oleva sivusto).

Lähteet

AWS. N.d. a. Lambda. Amazon Web Services. <https://aws.amazon.com/lambda/> (luettu 28.4.2021).

AWS. N.d. b. Amazon CloudFront Key Features. Amazon Web Services. <https://aws.amazon.com/cloudfront/features/> (luettu 28.4.2021).

Acunetix. N.d. a. What is SQL Injection (SQLi) and How to Prevent It. <https://www.acunetix.com/websitesecurity/sql-injection/> (luettu: 15.2.2021).

Aconetix. N.d. b. Cross-site Scripting (XSS). <https://www.acunetix.com/websitesecurity/cross-site-scripting/> (luettu: 15.2.2021).

Bergé, Greg 2019. The missing guide of SEO with Gatsby. <https://gregberge.com/blog/gatsby-seo> (luettu 28.3.2021).

Camden, Raymond & Rinaldi, Brian 2017: Working with Static Sites. O'Reilly Media, Inc. Luettavissa osoitteessa <https://learning.oreilly.com/library/view/working-with-static/9781491960936/> (luettu: 2.2.2021).

Contentful. N.d. Rich Text. <https://www.contentful.com/developers/docs/concepts/rich-text/> (luettu 28.4.2021).

Gatsby. N.d. a. Why Gatsby's Automatic Image Optimizations Matter. <https://www.gatsbyjs.com/docs/conceptual/using-gatsby-image/> (luettu 24.4.2021).

Gatsby. N.d. b. Gatsby Link API. <https://www.gatsbyjs.com/docs/reference/built-in-components/gatsby-link/> (luettu 25.4.2021).

Gatsby. N.d. c. Accessibility Statement. <https://www.gatsbyjs.com/accessibility-statement/> (luettu 22.4.2021).

Gatsby. N.d. d. Progressive Web Apps (PWAs). <https://www.gatsbyjs.com/docs/progressive-web-app/> (Luettu 5.4.2021).

Gatsby. N.d. e. SEO with Gatsby. <https://www.gatsbyjs.com/docs/how-to/adding-common-features/seo/> (luettu 28.3.2021).

Gatsby. N.d. f. Showcase. <https://www.gatsbyjs.com/showcase/> (luettu 15.3.2021).

Gatsby. N.d. g. Set Up Your Development Environment. <https://www.gatsbyjs.com/docs/tutorial/part-zero/> (luettu 1.4.2021).

Gatsby. N.d. h. Preparing a site to go live. <https://www.gatsbyjs.com/docs/tutorial/part-eight/> (luettu 12.4.2021).

Gatsby. N.d. i. Continuous Deployment. <https://www.gatsbyjs.com/docs/glossary/continuous-deployment/> (luettu 19.4.2021).

Gatsby. N.d. j. Building a Contact Form. <https://www.gatsbyjs.com/docs/building-a-contact-form/> (luettu 28.4.2021).

Google 2021a. Understand the JavaScript SEO basics. <https://developers.google.com/search/docs/guides/javascript-seo-basics> (luettu 31.3.2021).

Google 2021b. Understand how structured data works. <https://developers.google.com/search/docs/guides/intro-structured-data> (luettu 12.4.2021).

Kalkman, Ben 2020. Static vs. Dynamic Websites: What are They and Which is Better? Rocketmedia. <https://rocketmedia.com/blog/static-vs-dynamic-websites> (luettu 15.2.2021).

Kumari, Tanya 2018. 6 Reasons Why You Should Go for a Static Website. Dzone. <https://dzone.com/articles/6-reasons-why-you-should-go-for-a-static-website> (luettu 4.5.2021).

MDN Web Docs. N.d. a. CSS and JavaScript accessibility best practices. Mozilla <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/CSS_and_JavaScript> (luettu 22.4.2021).

MDN Web Docs. N.d. b. Progressive web apps (PWAs). Mozilla. <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps> (luettu 5.4.2021).

Murphy, Eric 2020. Why I'm no longer using GatsbyJS. YouTube, kanava: Eric Murphy. <https://youtu.be/kOmqb8RzuhA> (katsottu 29.3.2021).

Osmani, Addy & Grigorik, Ilya 2018. Speed is now a landing page factor for Google Search and Ads. Google Developers. <https://developers.google.com/web/updates/2018/07/search-ads-speed> (luettu 9.5.2021).

Prismic. N.d. Gatsby JS. <https://prismic.io/gatsbyjs> (luettu 5.5.2021).

Richard, Sam & LePage, Pete 2020. What are Progressive Web Apps? Web.dev. <https://web.dev/what-are-pwas/> (luettu 5.4.2021).

So, Preston 2021. Gatsby: The Definitive Guide. O'Reilly Media, Inc. Luettavissa osoitteessa

<https://learning.oreilly.com/library/view/gatsby-the-definitive/9781492087502/> (luettu 10.2.2021).

Sutton, Marcy 2019. Our Commitment to Accessibility in Gatsby. Gatsby. <https://www.gatsbyjs.com/blog/2019-04-18-gatsby-commitment-to-accessibility/> (luettu 22.4.2021).

Verreckt, Jeffrey 2017. Don’t Repeat Yourself (DRY). Think To Code. <https://www.thinktocode.com/2017/11/20/dont-repeat-dry/> (luettu 10.3.2021).

Väänänen, Joonas 2019. JAMstack – Dynaamisesti toimiva staattinen verkkosivusto. Opinnäytetyö (AMK). Oulu: Oulun ammattikorkeakoulu, tietotekniikan tutkinto-ohjelma. Luettavissa osoitteessa <https://www.theseus.fi/handle/10024/171590> (luettu 1.3.2021).

W3Techs. N.d. Usage statistics and market share of Gatsby. <https://w3techs.com/technologies/details/cm-gatsby> (luettu 15.3.2021).

Warren, Hashim 2021. Introducing Gatsby 3.0 – Faster in Every Way that Matters. Gatsby. <https://www.gatsbyjs.com/blog/gatsby-v3/> (luettu 2.3.2021).