jamk.fi

Tyyliopaslähtöiseen käyttöliittymäkehitykseen soveltuvien tyylioppaan generointityökalujen kartoittaminen

Johanna Ström

Opinnäytetyö
Toukokuu 2018
Liiketalouden ala
Tradenomi (AMK), Tietojenkäsittelyn koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ström, Johanna	Opinnäytetyö, AMK	Toukokuu 2018
	Sivumäärä 60	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Tyyliopaslähtöiseen käyttöliittymäkehitykseen soveltuvien tyylioppaan generointityökalujen kartoittaminen

Tutkinto-ohjelma

Tietojenkäsittelyn koulutusohjelma

Työn ohjaaja(t)

Niko Kiviaho

Toimeksiantaja(t)

Solteq Oyj

Tiivistelmä

Verkkosivustolle voidaan luoda tyyliopas, jonka tarkoituksena on dokumentoida sivustolla käytetyt tyylit ja toiminnallisuudet. Tyyliopaslähtöisessä käyttöliittymäkehityksessä tämä tyyliopas toimii käyttöliittymän suunnittelun ja kehityksen lähtökohtana ja ohjaavana tekijänä. Tyyliopas voidaan luoda manuaalisesti tai sen luonnin ja ylläpidon automatisoinnissa voidaan käyttää generointityökalua.

Tutkimuksen tavoitteena oli tunnistaa, millaisia vaatimuksia generointityökalun ja sillä luotavan tyylioppaan tulisi täyttää, jotta työkalu soveltuisi parhaiten tyyliopaslähtöiseen kehitysmalliin. Tutkimuksessa oli tunnistettavien vaatimusten perusteella tavoitteena arvioida neljän generointityökalun soveltuvuutta tyyliopaslähtöiseen käyttöliittymäkehitykseen. Työn toimeksiantajana oli Solteq Oyj, joka on digitaalisen liiketoiminnan ratkaisuihin erikoistunut osakeyhtiö.

Tutkimus toteutettiin laadullisena tutkimuksena. Tutkimuksen teoriapohja koottiin kirjallisista ja audiovisuaalisista lähteistä. Teoriapohjan perusteella laadittiin lista vaatimuksista, joiden perusteella arvioitiin seuraavien generointityökalujen soveltuvuutta tyyliopaslähtöiseen käyttöliittymäkehitykseen: KSS, Devbridge Styleguide, SC5 Style Guide Generator ja Fractal. Arvioinnin tulokset perustuvat tutkimuksen tekijän havaintoihin ja niistä koottuun havaintoraporttiin.

Tutkimuksen tuloksena todettiin, että tyyliopaslähtöisyys asettaa generointityökalulle ja tyylioppaalle sekä käytettävyyteen että rakenteeseen liittyviä vaatimuksia, joita mikään tutkituista työkaluista ei täyttänyt. Jatkokehityksenä suositellaan joko tutkimuksen ulkopuolelle jääneiden generointityökalujen arviointia tai uuden generointityökalun rakentamista tutkimuksessa koottujen vaatimusten pohjalta.

Avainsanat (asiasanat)

Tyyliopaslähtöinen käyttöliittymäkehitys, elävä tyyliopas, generointityökalu, automatisointi

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Ström, Johanna	Bachelor's thesis	May 2018
		Language of publication: Finnish
	Number of pages: 60	Permission for web publication: x

Title of publication

Charting style guide generators suitable for style guide driven development

Degree programme

Business Information Systems

Supervisor(s)

Kiviaho, Niko

Assigned by

Solteq Oyj

Abstract

The styles and functionalities of a website can be documented in a style guide. In the style guide driven development method the design and development of the user interface of the website are based on and guided by this style guide. The style guide can be created manually or its creation and maintenance can be automated with a generator tool.

The purpose of the study was to recognize the requirements the style guide driven development method sets for a style guide generator and the style guide it creates. The objective was to assess the suitability of four generators for style guide driven development using the recognized requirements. The study was assigned by Solteq Oyj which is a limited liability company specialized in digital business solutions.

The study was implemented with qualitative methods. The theory basis was gathered from a collection of literary and audiovisual sources. This basis was used to form a requirement list with which the following four style guide generators were evaluated: KSS, Devbridge Styleguide, SC5 Style Guide Generator, and Fractal. The results of the evaluation were based on the observations of the author and the report compiled from them.

The result of the study was that the requirements set by style guide driven development for the usability and structure of both the generator and the style guide created with it were not met by any of the evaluated generator tools. Further development is advised to be implemented by either evaluating the generator tools excluded from the study or by developing a new generator tool utilizing the requirements recognized in the study.

Keywords/tags (subjects)

Style guide driven development, living style guide, style guide generator, automatization

Miscellaneous (Confidential information)

Sisältö

Kuviot

Käsitteet

CSS	Tyyliohjeiden laji, jolla määritellään ulkoasu www-dokumenteille. Tyylitiedostojen pääte on .css. Lyhenne sanoista Cascading Style Sheets.
GitHub	Kehitysalusta Git-versionhallintaa käyttäville ohjelmakehitysprojekteille.
Grunt	JavaScript-pohjaisten tehtävien automatisointiin tarkoitettu työkalu. Gruntin avulla voidaan automatisoida erilaisia tehtäviä, kuten erilaisten tiedostojen yhdistely, minimointi ja versiointi. Gruntin käyttämät asetukset määritetään Gruntfile.js -tiedostoon.
Gulp	JavaScript-pohjaisten tehtävien automatisointiin tarkoitettu työkalu. Vaihtoehto Gruntille.
HTML	Avoimesti standardoitu kuvauskieli, jota voidaan käyttää internetsivujen rakentamiseen. Lyhenne sanoista Hypertext Markup Language.
JavaScript	Www-ympäristössä käytettävä dynaaminen komentosarjakieli.
Verkkosivuston käyttöliittymä	Liittymä, jonka kautta käyttäjä käyttää verkkosivustoa.
Node.js	Node.js on avoimen lähdekoodin alustariippumaton ympäristö, jonka avulla JavaScriptia voidaan suorittaa selaimen sijaan palvelimessa.

npm	JavaScript-ohjelmointikieltä varten luotu pakettien hallintajärjestelmä, jota käytetään Node.js-ympäristössä. Lyhenne sanoista node package manager.
Sass	Tyylisivujen ohjelmointikieli. Tiedostojen pääte on .sass tai .scss. Lyhenne sanoista Syntactically Awesome Style Sheets.

1 Johdanto

Ympärillämme olevat esineet muodostuvat erilaisista moduuleista eli itsenäisistä osasista. Kun autostamme hajoaa rengas, voimme vaihtaa sen tilalle ehjän ilman uuden auton hankkimista. Yhden renkaan vaihtaminen uuden auton ostamisen sijaan on meille edullisempaa. Kun taas vihdoin vaihdamme vanhan auto uuteen, saattavat vanhan auton renkaat tarvittaessa sopia uuteenkin autoon. Tällöin uutta autoa varten ei tarvitse ostaa uusia renkaita, koska vanhaa moduulia voidaan hyödyntää uudessa moduulikokoelmassa. Myös verkkosivuston käyttöliittymä voidaan jakaa moduuleihin. Käyttöliittymäkehityksessä moduuleja kutsutaan komponenteiksi, mutta niiden periaate on sama: iso kokonaisuus pilkotaan pienempiin monikäyttöisiin osiin. Verkkosivustolla komponentteja voivat olla esimerkiksi painikkeet ja syöttökentät.

Käyttöliittymän muodostavat komponentit voidaan listata samaan dokumenttiin, jolloin käytettävissä olevat komponentit voidaan tarkistaa yhdestä paikasta. Tätä dokumentaatiota kutsutaan verkkosivuston käyttöliittymän tyylioppaaksi. Jos tyyliopas on erillinen osa käyttöliittymän kehitystä, opas saattaa jäädä esimerkiksi kiireen tai unohtamisen takia päivittämättä ja samoja komponentteja saatetaan luoda uudelleen ja uudelleen. Tyylioppaan ajantasaisuuden varmistamiseksi käyttöliittymää voidaan kehittää tyyliopaslähtöisesti eli tyylioppaan sisältämät jo olemassa olevat tyylit ohjaavat suunnittelua ja komponenttien kehitystyö tehdään tyylioppaassa.

Käyttöliittymän tyyliopas voidaan luoda ja sitä voidaan päivittää automatisoidusti. Tällöin tyylioppaan rakenteen ja ominaisuudet määrittävät oppaan luonnissa käytettävä generointityökalu. Generointityökaluissa on useita vaihtoehtoja ja oikean työkalun valinnassa tulee huomioida muun muassa projektissa käytetty kehitysmalli ja koodin luonnissa käytettävät tekniikat. Tässä opinnäytetyössä tutkitaan tyyliopaslähtöistä käyttöliittymäkehitystä ja pyritään tunnistamaan sen asettamat vaatimukset generointityökalulle ja sillä luodulle tyylioppaalle. Vaatimusten perusteella työssä arvioidaan neljän työkalun soveltuvuutta tyyliopaslähtöiseen kehitysmalliin.

2 Tutkimusasetelma

2.1 Toimeksiantajan esittely

Opinnäytetyön toimeksiantaja on Solteq Oyj. Solteq on vuonna 1982 perustettu toimialariippumaton digitaalisen liiketoiminnan ratkaisuihin erikoistunut osakeyhtiö. Solteqin palveluksessa työskentelee yli 550 asiantuntijaa Suomessa, Ruotsissa, Norjassa, Tanskassa ja Puolassa. Solteqin asiakaskunta koostuu pääosin pohjoismaalaisista yrityksistä, mutta asiakasyrityksiä on myös muualta Euroopasta, Pohjois-Amerikasta, Aasiasta ja Australiasta. Solteqin liikevaihto oli vuonna 2017 61,5 miljoonaa euroa. (Solteq - Rakenna tulevaisuutta ja kehitä itseäsi huippuseurassa n.d.; Solteq Oyj - Vuosikertomus 2017, 3.)

Solteqin tarjoamat palvelut on jaettu kolmeen osa-alueeseen: liiketoimintaratkaisut, asiakaskokemus ja digitaalinen markkinointi. Liiketoimintaratkaisut-osa-alue koostuu toiminnanohjaukseen ja myymäläjärjestelmiin liittyvien ratkaisujen konsultoinnista ja toteutuksesta. Asiakaskokemus-osa-alueeseen kuuluvat asiakaskokemukseen liittyvät palvelut ja järjestelmät kuten muun muassa verkkokaupan konsultointi, asiakaskokemus ja design, alustat ja ratkaisut sekä mobiili- ja verkkosovellukset. Digitaalinen markkinointi -osa-alue pitää sisällään muun muassa hakukoneoptimointiin ja markkinointiautomaatioon liittyvät palvelut. Tämä opinnäytetyö on tehty asiakaskokemus-osa-alueen verkkokauppayksikölle. (Asiakaskokemus – Solteq n.d.; Digitaalinen markkinointi – Solteq n.d.; Liiketoimintaratkaisut – Solteq n.d.)

2.2 Tutkimuksen taustat, tavoitteet ja rajaukset

Solteq haluaisi kokeilla verkkokauppojen käyttöliittymäkehittämisessä tyyliopaslähtöistä kehittämismetodia sekä sidosryhmien välisen kommunikaation tehostamiseksi että käyttöliittymäkoodin laadun ja ylläpidettävyyden parantamiseksi. Tyyliopaslähtöisen käyttöliittymäkehityksen kulmakivenä on nimensä mukaisesti tyyliopas. Jotta kehitysmalli toimisi ja siitä saataisiin paras mahdollinen hyöty, tyylioppaan tulee täyttää rakenteellisesti tietyt vaatimukset. Jos tyylioppaan luonti ja ylläpito on automatisoitu, automatisoinnissa käytettävä generointityökalu määrittää tyylioppaan rakenteen. Generointityökaluja on olemassa useita ja ne ovat keskenään hyvin erilaisia.

Tämän opinnäytetyön tavoitteena on tutkia tyyliopaslähtöistä käyttöliittymäkehitystä ja tunnistaa kehitysmallin asettamat vaatimukset käyttöliittymän tyylioppaalle ja sen generointityökalulle. Opinnäytetyön teorian viitekehyksen pohjalta muodostetaan vaatimuslista, jonka perusteella arvioidaan neljää toimeksiantajan vaatimukset täyttävää tyylioppaan generointityökalua. Työn tuloksena on tieto siitä, soveltuuko jokin tarkastelussa olevista generointityökaluista sellaisenaan käytettäväksi tyyliopaslähtöisessä käyttöliittymäkehityksessä Solteqin projekteissa.

Opinnäytetyö rajataan koskemaan elävän tyylioppaan tuottamiseen liittyvien työkalujen arviointia. Rajallisten aikaresurssien vuoksi kaikkia toimeksiantajan vaatimukset täyttäviä työkaluja ei voida arvioida tässä tutkimuksessa vaan työ on rajattu neljän työkalun soveltuvuuden arviointiin. Työssä ei käsitellä tyylioppaan varsinaista sisältöä eikä rakenneta oikeaa tyyliopasta vaan generointityökaluja arvioidaan vain tutkimusta varten luodulla sisällöllä.

2.3 Tutkimuskysymykset

Tutkimuskysymysten tarkoituksena on esittää tutkimusongelma helpommin käsiteltävässä muodossa. Tutkimuskysymyksiin pyritään vastaamaan työn aikana kerätyn aineiston perusteella. (Kananen 2015, 12.)

Tässä opinnäytetyössä tutkimusongelma on jaoteltu seuraaviin tutkimuskysymyksiin:

1. Mitä on tyyliopaslähtöinen käyttöliittymäkehitys?
2. Millaisia vaatimuksia tyyliopaslähtöinen käyttöliittymäkehitys asettaa tyylioppaalle ja sen generointityökalulle?
3. Soveltuuko jokin arvioitavista tyylioppaan generointityökaluista käytettäväksi toimeksiantajan tyyliopaslähtöistä kehitysmallia käyttävissä projekteissa?

2.4 Tutkimusmenetelmät

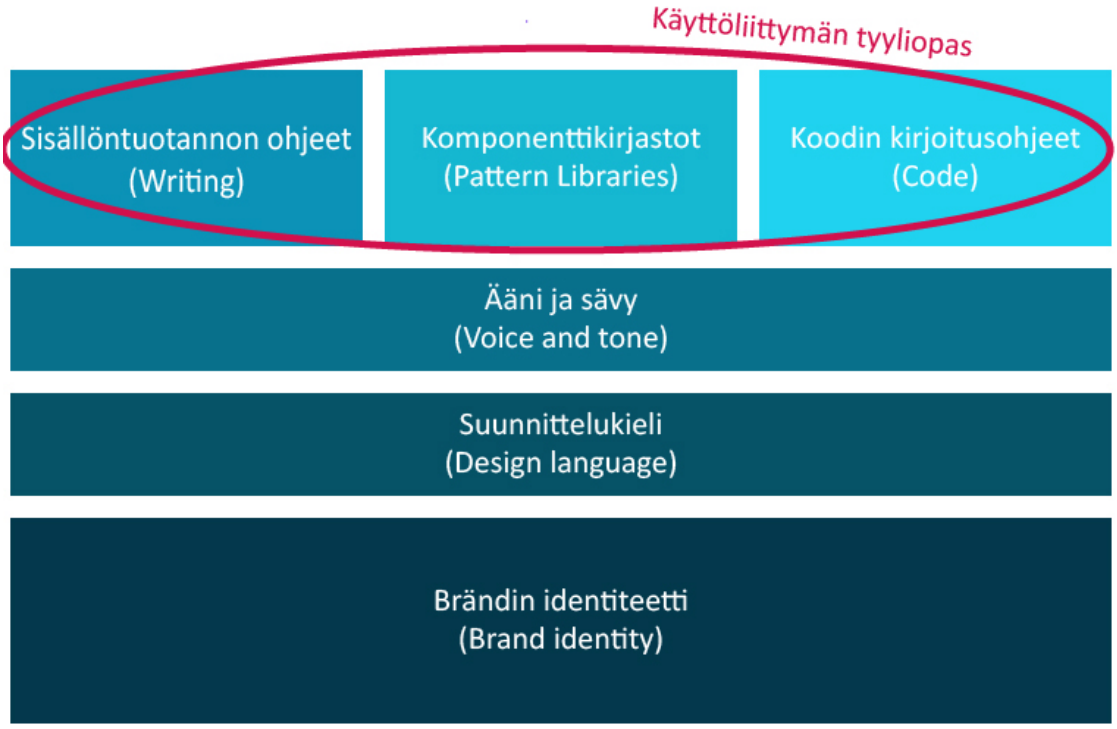
Tutkimusasetelma ja -menetelmät tulisi valita siten, että tutkimustulokset ovat mahdollisimman luotettavia (Kananen 2015, 13). Tyylioppaan generointityökaluista löytyy joitakin vertailevia artikkeleita. Artikkeleissa generointityökaluja ei kuitenkaan arvioida tyyliopaslähtöisen käyttöliittymäkehityksen näkökulmasta. Tutkimusta ei voi siis perustaa täysin kirjalliseen aineistoon vaan tutkimusongelman ratkaisu vaatii tutkittavien työkalujen testaamista. Tämän vuoksi opinnäytetyön tutkimusongelma pyritään ratkaisemaan kehittämistutkimuksen keinoin. Kehittämistutkimuksessa tutkimusongelma pyritään pienentämään tai poistamaan kokonaan käytännössä. (Kananen 2015, 11).

Tutkimuksen tietopohja koostetaan alan ammattilaisten kirjoittamista kirjoista ja artikkeleista sekä heidän pitämistään luennoista. Tällä tavoin työssä pyritään saamaan mahdollisimman laaja yleiskuva tyyliopaslähtöisyydestä ja vaatimuksista, jotka se asettaa tyylioppaalle ja generointityökalulle. Suurin osa työssä käytetyistä lähteistä on sähköisiä, sillä tieto vanhenee IT-alalla nopeasti. Esimerkiksi ohjelmistotuotannon professori Philippe Kruchten (2008, 10) on esittänyt hypoteesin, jonka mukaan sovelluskehityksen ideat vanhenevat viidessä vuodessa.

Tutkimusaineiston keräämisessä käytetään laadullista tutkimusotetta. Laadullisia aineistonkeruumenetelmiä ovat havainnointi, haastattelut, kyselyt ja dokumentit (Kananen 2015, 12). Tässä opinnäytetyössä tutkimusaineistoa kerätään asentamalla työkalut ja testaamalla niiden toimintaa ja ominaisuuksia käytännössä. Työkaluista kerättyä tutkimusaineistoa analysoidaan tutkimuksen osana koostetun vaatimuslistan perusteella.

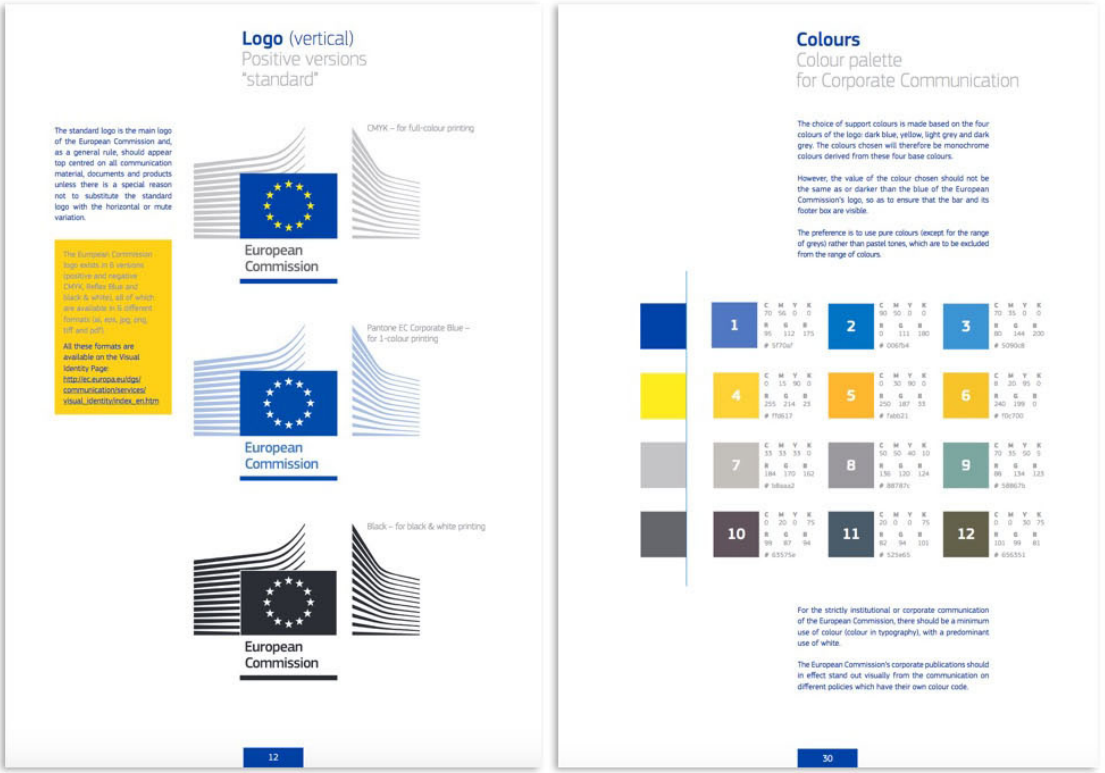
3 Tyyliopas

Yrityksen menestyksen kannalta on tärkeää, että sen brändi on yhtenäinen. Brändin luominen ja ylläpitäminen ei ole helppoa, sillä niin medioita, joissa yritys on läsnä, kuin yritystä edustavia henkilöitä on monesti useita. Yhtenäisen ilmeen säilyttämistä varten yritykset luovat käyttöönsä erilaisia tyylioppaita, joissa kuvataan brändiin liittyvät yleiset linjat ja tyylioppaan tyyppikohtaiset asiat. (Frost 2014.) Mitä suurempi yritys on, sitä yksityiskohtaisempia tyylioppaiden tulisi olla. Yrityksen koon suurentuessa myös brändin hallinta hankaloituu. (Debenham 2017, 8.)



Kuvio 1. Tyylioppaiden erilaiset tyypit (Frost 2014)

Brad Frostin (2014) mukaan yritysten tyylioppaat voidaan jakaa kuuteen eri tyyppiin (ks. kuvio 1). Yleisin tyylioppaan muoto on brändin identiteettityyliopas, jossa kuvataan yleisellä tasolla brändiin liittyvät linjaukset. Oppaassa esitetään usein muun muassa yrityksen logo ja sen käyttöön liittyvät rajaukset ja brändivärit. Näin on tehty esimerkiksi Euroopan Unionin visuaalista ilmettä kuvaavassa tyylioppaassa (ks. kuvio 2).



Kuvio 2. Otteita Euroopan Unionin visuaalisen ilmeen tyylioppaasta (European Commission visual identity 2016)

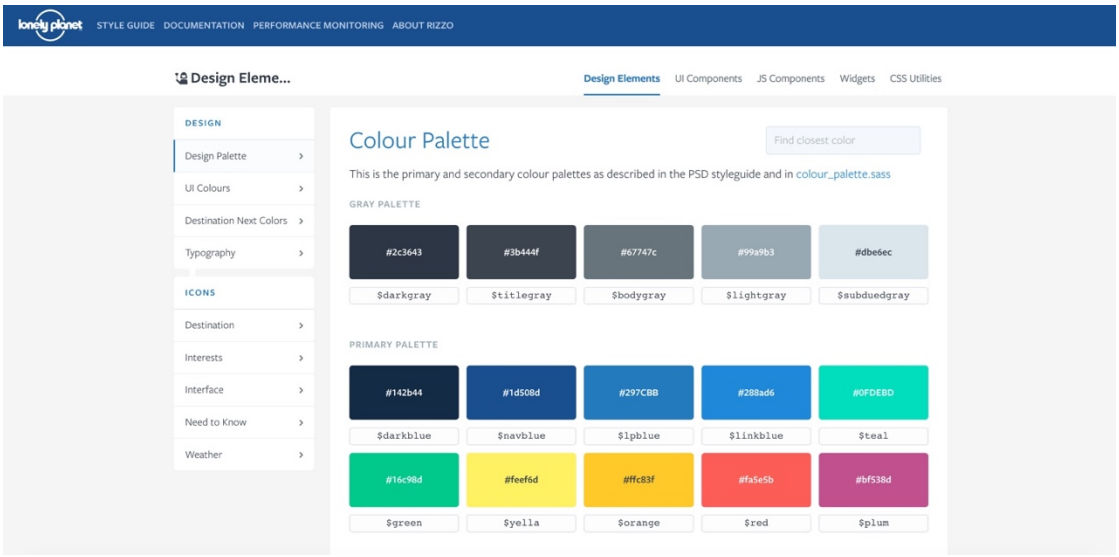
Yrityksillä voi olla omat tyylioppaansa myös suunnittelukielelle, äänelle ja sävylle sekä verkkosivustojen ja -palveluiden käyttöliittymille. Brändityylioppaat toimivat näiden yksityiskohtaisempien tyylioppaiden pohjana ja ohjaavana tekijänä. (Frost 2014.) Brändin identiteettityyliopas ei riitä sellaisenaan verkkosivuston käyttöliittymän tyylioppaaksi, koska se harvoin kuvaa käyttöliittymää koskevia asioita, kuten ulkoasua ja toiminnallisuuksia, riittävän tarkalla tasolla. Lisäksi identiteettityyliopas on usein tehty painomateriaalia varten Pantone-värikoodeineen ja millimetrimittoineen. Verkkosivuston käyttöliittymää varten kannattaakin luoda oma tyylioppaansa. (Debenham 2017, 5.)

3.1 Verkkosivuston elävä tyyliopas

Siinä missä brändin identiteettityylioppaan tarkoituksena on ylläpitää brändin yhteneväisyyttä, käyttöliittymän tyyliopasta käytetään verkkosivuston tai -sovelluksen yhteneväisyyden säilyttämisessä ja ylläpidon ohjaamisessa (De La Cuadra 2018). Käyttöliittymän tyylioppaan pääasiallisena tarkoituksena on dokumentoida verkkosivuston ulkoasu ja toiminnallisuudet ja toimia käyttöliittymää koskevan kommunikoinnin apuvälineenä. Käyttöliittymän tyylioppaan sisältö voidaan jakaa kolmeen osa-alueeseen:

- sisällön tuottamiseen liittyvät ohjeistukset
- komponenttikirjasto eli sivuston toiminnallisuudet, rakenteet ja muotoilut
- koodin tuottamiseen liittyvät ohjeistukset.

Tyylioppaassa voidaan ohjeistaa sisällöntuottajia esimerkiksi tekstin kirjoitustyyliin tai verkkosivustolla käytettävien kuvien optimointiin liittyvissä asioissa. Komponenttikirjasto-osiossa voidaan kuvata esimerkiksi verkkosivustolla käytössä olevat värit, fontit ja erilaisten komponenttien kuten syöttökenttien ja painikkeiden tyylit ja toiminnallisuudet. Komponenttikirjastosta käytetään joskus myös nimitystä mallikirjasto (englanniksi pattern library). Tyyliopas voi lisäksi sisältää koodin tuottamista varten omat ohjeistuksensa, joissa voidaan ottaa kantaa esimerkiksi koodissa käytettävien sisennysten kokoon, tiedostojen nimeämiseen ja koodin kommentointiin (Debenham 2017, 8–14). Anna Debenham (2017, 52–53) ja Brad Frost (2016b, luku 3) mainitsevat Lonely Planetin tyylioppaan hyväksi esimerkiksi selkeästä ja monipuolisesta käyttöliittymän tyylioppaasta (ks. kuvio 3). Tyyliopas löytyy osoitteesta http://rizzo.lonelyplanet.com/styleguide. Tyyliopas on heidän mielestään monipuolinen ja selkeä.



Kuvio 3. Lonely Planetin verkkosivuston tyyliopas

Käyttöliittymän tyyliopas eroaa esimerkiksi PDF- tai PowerPoint-pohjaisesta identiteettityylioppaasta siten, että se rakennetaan verkkosivuston kaltaiseksi. Tyyliopas koostuu siis HTML-, CSS- ja JavaScript-tiedostoista ja sitä voidaan tarkastella selaimella. Käyttöliittymän tyyliopas on interaktiivinen, jolloin siitä käy paremmin ilmi esimerkiksi eri osioiden toiminnallisuudet, kuten painikkeiden ja linkkien erilaiset tilavaihtelut ja animaatiot. Käyttöliittymän tyyliopas voidaan ja on suositeltavaakin luoda eläväksi. Elävällä tyylioppaalla tarkoitetaan opasta, joka käyttää samaa CSS-tyylitiedostoa verkkosivuston kanssa. Tyylioppaaseen dokumentoidut tyylit elävät eli päivittyvät yhdessä verkkosivuston kanssa ja ovat aina ajan tasalla. Jos esimerkiksi sivustolla käytetyt värit muuttuvat, valuu muutos verkkosivuston lisäksi myös tyylioppaaseen. Tällä vältetään tyylioppaan vanheneminen. (Debenham 2017, 10–11.) Ideaalitilanteessa myös tyylioppaan käyttämät HTML- ja JavaScript-tiedostot ovat samat kuin sivustolla. HTML-tiedostojen yhteiskäyttö ei kuitenkaan aina ole mahdollista verkkosivuston rakenteesta johtuen. (Friedman 2016.)

3.2 Elävän tyylioppaan ominaisuudet

Yksin käyttöliittymän tyylioppaan tärkeimmistä tehtävistä on toimia projektitiimin sisäisen ja projektin sidosryhmien välisen kommunikoinnin apuvälineenä. Brad Frost (2016b, luku 5) kehottaa luomaan tyylioppaan, joka on ulkoasultaan kutsuva ja jossa

on helppo navigoida, jotta tyylioppaan hyödyntäminen olisi houkuttelevaa. Tyylioppaan komponenttikirjastossa yksittäisestä komponentista voidaan Frostin (2016a) ja Friedmanin (2016) mukaan projektin tarpeista riippuen näyttää

- komponentin nimi
- komponentin kuvaus
- komponentin käyttöohjeet ja lähteet
- elävä esimerkki komponentista
- kopioitava koodiesimerkki komponentista
- komponenttiin liittyvät koodit kuten HTML, CSS ja JavaScript
- komponentin käyttökonteksti
- tieto komponentin tekijöistä
- komponentin metatiedot.

Yksittäisen komponentin nimeäminen ja nimen näyttäminen tyylioppaassa on viestinnän näkökulmasta usein tarpeellista. Komponenttien nimistä muodostuvan yhteisen termistön avulla on mahdollista vähentää väärinymmärryksiä ja yksinkertaistaa sidosryhmien välistä kommunikaatiota. Komponentin yhteydessä voidaan näyttää nimen lisäksi lyhyt kuvaus komponentista ja sen tarkoituksesta. Joissakin tilanteissa komponentille voi olla tarpeen lisätä myös erillinen pidempi käyttöohje, jossa kuvataan tarkemmin komponentin käyttöön liittyvät mahdollisuudet ja rajaukset. Tämän lisäksi ohjeisiin voidaan kirjoittaa opastusta komponentin valintaan liittyen. Komponentin yhteydessä voidaan näyttää myös linkkejä ulkopuolisiin lähteisiin. Esimerkiksi navigaatio-komponentin kohdalla saatetaan haluta viitata saavutettavuusohjeistuksiin, jotka on huomioitu komponenttia suunnitellessa ja kehittäessä ja tulee jatkossakin muistaa. (Friedman 2016; Frost 2016a.)

Komponentin ulkoasua ja käyttäytymistä voidaan havainnollistaa tyylioppaassa elävällä esimerkillä. Esimerkki voi olla responsiivinen, jolloin komponentti voidaan tarkistaa eri ruutuko'oissa. Myös komponentin tulosteissa käytettävät tyylit voi olla tarpeen esittää komponentin yhteydessä. Tyylioppaassa voidaan näyttää elävän esimerkin HTML-merkkaus, jolloin se on kopioitavissa tarpeen tullen. (Friedman 2016; Frost 2016a.) Komponenttiin liittyvistä koodeista kannattaa Brad Frostin (2016a) mukaan näyttää HTML-merkkaus ja tarvittaessa muun muassa CSS- ja JavaScript-koodit.

Tyylioppaassa komponentit ovat irrallaan toisistaan ja niiden todellista käyttöpaikkaa verkkosivustolla voi olla vaikea hahmottaa. Friedman (2016) ja Frost (2016a) mainitsevat, että kokonaisuuden hahmottuminen helpottuu, jos tyylioppaaseen määritetään komponentin konteksti. Kontekstilla kuvataan sitä, mitkä komponentit muodostavat kokonaisuuden ja millä sivuilla yksittäisiä komponentteja ja komponenttikokonaisuuksia käytetään. Konteksti voidaan osoittaa manuaalisesti otetuilla kuvakaappauksilla tai dynaamisilla linkeillä, joiden avulla tyylioppaassa voidaan navigoida komponenttien välillä.

Komponenttiin voidaan lisätä sen ovat suunnitelleiden ja toteuttaneiden henkilöiden nimet. Friedmanin (2016) mukaan nimitiedot helpottavat komponentin ylläpitoa ja jatkokehitystä. Komponenttiin voi liittää myös hyödyllisiä metatietoja kuten statustiedon eli maininnan onko komponentti kesken, valmis tai vanhentunut. Myös komponenttiin liittyvät versiotiedot, kuten versionumero ja viimeisimmät muutokset, voidaan esittää tyylioppaassa muutosten seuraamisen helpottamiseksi. (Frost 2016a.)

Aiemman listauksen lisäksi Friedmanin (2016) mukaan on hyödyllistä, jos komponentti voidaan avata omana sivunaan selaimessa. Tällöin sivulla näytetään vain komponentti ja tyylioppaan oma rakenne ja muut komponentit eivät ole näkyvissä. Omalla sivullaan näytettävän eli eristetyn komponentin kehittäminen, korjaaminen ja testaaminen helpottuvat, kun komponentin ympärillä oleva rakenne poistuu. Friedman mainitsee myös, että komponentille voi muodostua tyylioppaaseen oma linkkinsä, jonka avulla komponenttia koskeva kohta tai sivu voidaan jakaa linkkinä sidosryhmille. Komponenttikohtainen linkki on hyödyllinen erityisesti, jos tyylioppaassa näytetään monta komponenttia samalla sivulla, koska yksittäisen komponentin löytäminen komponenttilistauksesta on suoran linkin avulla nopeampaa.

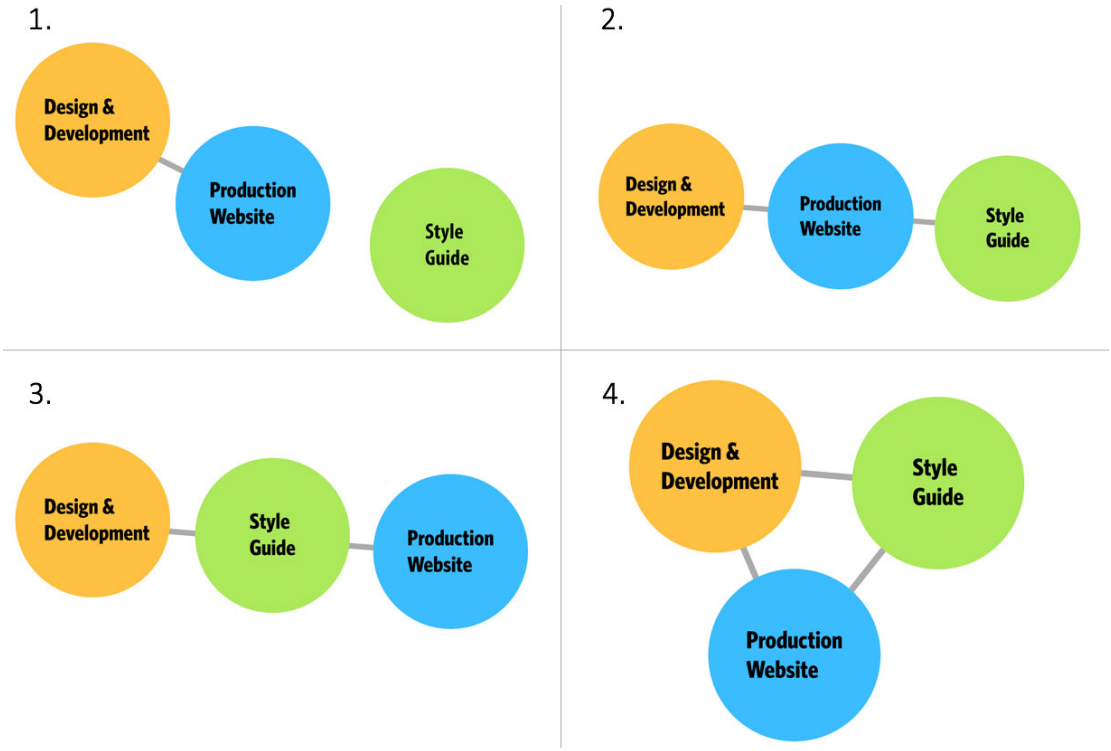
Komponenttikirjaston lisäksi tyylioppaaseen kuuluvat koodin ja sisällön kirjoittamista varten luotavat ohjeistukset. Tällaisilla tekstimuotoisilla ohjeistuksilla voi olla tyylioppaassa omat sivunsa. Ohjeistusten lisäksi tyyliopas voi sisältää myös muita sisältösivuja kuten Ajankohtaista-osion, josta voisi käy ilmi tyylioppaan viimeisimmät muutokset. (Debenham 2017, 14.)

3.3 Elävän tyylioppaan generointi

Käyttöliittymän elävä tyyliopas voidaan luoda manuaalisesti tai sen generoinnissa voidaan käyttää automatisointityökalua, joka luo tyylioppaan rakenteen ja ominaisuudet automaattisesti. Generaattorit nopeuttavat tyylioppaan luontia ja ylläpitoa automatisoimalla joitakin toimintoja. Generointityökalu voi esimerkiksi seurata tyylitiedostoja ja tiedostojen muuttuessa generoida tyylioppaan automaattisesti, jolloin tyyliopas pysyy aina ajan tasalla. Generointityökalu ei kuitenkaan automatisoi aivan kaikkea, joten tyylioppaan ylläpito vaatii yhä jonkin verran manuaalista työtä, joskin vähemmän kuin ilman työkalua. (Robertson 2015; De La Cuadra 2018.)

Generaattoreita on olemassa useita erilaisia ja niiden ominaisuudet ja toimintatavat saattavat erota paljonkin toisistaan. Osa generointityökaluista luo tyylioppaan esimerkiksi suoraan CSS-tyylitiedostoihin kirjoitettujen merkintöjen pohjalta, jolloin tyyliopasta varten ei tarvitse luoda HTML-tiedostoja manuaalisesti. Jotkin työkalut taas vaativat erillisten HTML-sivujen luomista. Osa generaattoreista on luotu erityisesti tiettyä koodikieltä, kehitysmetodia tai kehitysalustaa varten, kun toiset taas tukevat useaa eri yhdistelmää. Projektille sopivan generointityökalun valinta tulisikin aloittaa kartoittamalla projektin tarpeet. (Robertson, 2015.)

4 Tyyliopaslähtöinen käyttöliittymäkehitys



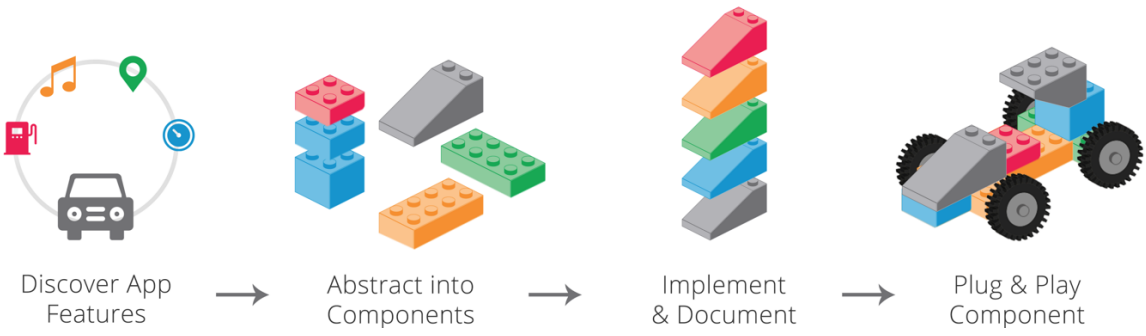
Kuvio 4. Tyylioppaan ylläpidon sijainti käyttöliittymäkehityksestä (Coyier 2015, muokattu)

Verkkosivuston tyyliopasta voidaan ylläpitää käyttöliittymäkehityksen eri vaiheissa (ks. kuvio 4). Coyier (2015) on tunnistanut artikkelissaan neljä vaihetta, jossa tyyliopasta yleensä ylläpidetään. Tyyliopas voi olla kokonaan erillään verkkosivustosta ja sen kehitysprosessista. Tällöin tyylioppaalla on oma koodipohja ja sitä kehitetään erillisenä kokonaisuutena, mikä aiheuttaa haasteen tyylioppaan ajantasaisuudelle. Koska tyyliopas on irrallaan verkkosivustosta, se saattaa jäädä päivittämättä. Toisaalta tyyliopas saattaa jakaa osan koodipohjastaan verkkosivuston kanssa, jolloin se päivittyy, kun verkkosivustoa muokataan. Tämä ei kuitenkaan kannusta komponenttimuotoiseen kehittämiseen, koska verkkosivustoa kehitetään kokonaisuutena. Tyylioppaan ollessa toissijainen kehityskohde, oppaan elävien esimerkkien HTML-merkintöjen päivittäminen saattaa unohtua. (Coyier 2015.)

Tyyliopasta voidaan pitää myös kehitystä ohjaavana tekijänä, jolloin mitään käyttöliittymää koskevia muutoksia ei viedä verkkosivustolle ennen kuin ne on luotu tyylioppaaseen. Tyyliopasta ja verkkosivustoa voidaan kehittää myös samanaikaisesti. Tällöin tyyliopasta muokatessa verkkosivusto päivittyy ja toisinpäin. Tyyliopas ja verkkosivusto käyttävät siis samaa koodipohjaa tai päivitys on automatisoitu. (Coyier 2015.)

Tyyliopaslähtöisessä käyttöliittymäkehityksessä pyritään soveltamaan tapoja 3 ja 4 projektin luonteesta riippuen, koska saman HTML-pohjan käyttäminen ei ole aina mahdollista. Tyyliopaslähtöisyydessä käyttöliittymäkehityksen kulmakivenä on elävä tyyliopas, jota käytetään suunnittelu- ja kehitystyön pohjana. (Fordham 2014.) Käyttöliittymän kehitys tehdään ensin varsinaisen sovelluksen sijaan tyylioppaassa, jolloin tyyliopas pysyy varmasti aina ajantasaisena. Tyyliopaslähtöisyyden avulla käyttöliittymäkehitys voidaan erottaa taustalogiikan kehityksestä. Näin kehitystyö voidaan aloittaa aikaisemmassa vaiheessa projektia. Kun taustalogiikka on valmis, käyttöliittymän toteutus voidaan kopioida tyylioppaasta oikeaan ympäristöön. (Lewis 2015.)

4.1 Tyyliopaslähtöinen kehitysprosessi



Kuvio 5. Tyyliopaslähtöisen suunnitteluprosessin vaiheet (De La Cuadra 2015)

Adriana De La Cuadra (2015) jakaa artikkelissaan tyyliopaslähtöisen kehitysprosessin neljään eri vaiheeseen. Vaiheet on havainnollistettu kuviossa 5. De La Cuadran mukaan käyttöliittymän kehitysprosessi alkaa ratkaisemista vaativan ongelman tunnistamisella ja vaatimusten keräämisellä. Tässä vaiheessa käyttöliittymäsuunnittelija pyrkii

suunnittelemaan uuden toiminnallisuuden tyylioppaan osoittaman visuaalisen suunnan ja ohjeistuksen pohjalta. Joskus toiminnallisuus voidaan muodostaa jo olemassa olevan komponentin avulla tai komponentteja yhdistelemällä, mutta toisinaan se vaatii uusien komponenttien toteuttamista. Tarkoituksena on kuitenkin käyttää tyyliopasta suunnittelun lähtökohtana eikä aloittaa suunnittelutyötä alusta. (De La Cuadra 2015.)

Kun käyttöliittymän graafinen suunnitelma on valmis, se pilkotaan pienempiin osiin. Käyttöliittymäsuunnittelija ja -kehittäjä työskentelevät yhdessä tunnistaakseen suunnitelmasta sekä jo käytettävissä olevat että uudet kehitystyötä vaativat komponentit. Suunnittelijan ja kehittäjän on tärkeää käydä suunnitelma läpi yhdessä, koska staattisesta suunnitelmasta ei aina käy ilmi kaikki ulkoasuun ja sen toimintaan liittyvät asiat kuten animaatiot. Keskustelussa käytetään referenssinä elävää tyyliopasta ja siihen dokumentoituja toiminnallisuuksia. Interaktiivisen tyylioppaan avulla voidaan varmistaa, että molemmilla on sama näkemys keskustelluista asioista. Kun suunnitelma on pilkottu pienempiin osiin ja uudet komponentit tunnistettu, kehitystyöhön tarvittavan työmäärän hahmottaminen ja työn jakaminen helpottuvat. (De La Cuadra 2015.)

Koska kehitystyötä tehdään komponenttikohtaisesti tyylioppaassa, käyttöliittymäkehittäjä voi aloittaa kehitystyön, vaikka suunnitelma ei olisikaan vielä kokonaan valmis. Edellisessä vaiheessa tunnistetut komponentit luodaan ensin tyylioppaaseen, jolloin ne ovat irti kontekstistaan ja muista komponenteista. Tyyliopaslähtöinen kehittäminen eroaa tässä vaiheessa muista kehitystavoista siten, että kehittämisvaiheessa pidetään yllä samaan aikaan myös elävää tyyliopasta. Komponenttien eristäminen kannustaa kehittäjiä luomaan komponentteja, jotka ovat hyödynnettävissä useassa sivuston eri osiossa ja tarkoituksessa. (De La Cuadra 2015.)

Kun komponentti on luotu tyylioppaaseen, se voidaan integroida varsinaiseen verkkosivustoon. Jos tyyliopas ja verkkosivusto jakavat esimerkiksi samat tyylitiedostot, integrointi voi tarkoittaa HTML-merkkauksen kopiointia sivuston koodipohjaan. Tässä vaiheessa komponenttia voidaan testata oikealla sisällöllä. Joskus sisältö aiheuttaa omat haasteensa esimerkiksi sanojen pituuksien vuoksi ja komponentti saattaa vaatia korjausta. Tällöin palataan edelliseen vaiheeseen eli tehdään korjaukset ensin tyylioppaaseen ja tuodaan ne sitten tarvittaessa varsinaiselle verkkosivustolle. (De La Cuadra 2015.)

4.2 Kehitysmetodin hyödyt

Hyvin tehdystä tyylioppaasta ja toimivasta tyyliopaslähtöisestä kehitysmallista hyötyvät kaikki verkkosivuston ylläpitoon ja kehittämiseen liittyvät sidosryhmien edustajat kuten asiakas, verkkosivuston ulkoasusta vastaava käyttöliittymäsuunnittelija ja koodillisen toteutusta tekevä käyttöliittymäkehittäjä (Robertson 2014). Kun tyyliopas toimii sekä suunnittelun että toteutuksen perustana, verkkosivusto säilyy todennäköisemmin yhteneväisenä niin ulkoasullisesti kuin koodillisesti. Käyttöliittymäsuunnittelijan voi tarkistaa oppaasta verkkosivuston peruselementit kuten värit ja fontit ja käytettävissä olevat komponentit. Tyylioppaan avulla suunnittelija voi varmistaa, että uusi suunnitelma sopii hyvin verkkosivuston yleisilmeeseen. Myös erilaisten komponenttien määrää saadaan hillittyä, kun kaikki käytettävissä olevat vaihtoehdot ovat esillä tyylioppaassa. Käyttöliittymän eheä ilme lisää yrityksen uskottavuutta. (Frost 2016b; Sullivan 2014.)

Tyyliopasta voidaan käyttää kommunikointivälineenä niin kehitystiimin sisällä kuin sidosryhmien kanssa. Tyylioppaassa käytetty sanasto muodostaa eräänlaisen termistön sivuston komponenteista. Termistön avulla viestintä asiakkaan ja muiden sidosryhmien kanssa helpottuu. Koska tyyliopasta päivitetään tyyliopaslähtöisyydessä kehitysmallissa ensin, komponentit nimetään jo projektin alkuvaiheessa. Näin nimet vakiintuvat nopeammin ja väärinkäsitysten määrä vähenee. Tiimin sisällä tyylioppaalla voidaan jakaa tietoa jo olemassa olevasta koodista, jolloin vältytään koodien uudelleenkirjoittamiselta. Tällä säästetään aikaa ja saadaan pidettyä tyylitiedosto kooltaan pienempänä. Pienempi tyylitiedosto on tärkeää erityisesti verkkosivuston suorituskyvyn vuoksi. (Frost 2016b; Sullivan 2014.)

Hyvin toimiva tyyliopaslähtöinen kehitysmalli aikaistaa käyttöliittymän kehityksen aloittamista. Koska käyttöliittymäsuunnitelma muodostuu komponenteista, yksittäisten komponenttien kehitys voidaan aloittaa, vaikka kokonaissuunnitelma tai varsinainen sisältö, kuten tekstit ja kuvat, eivät olisikaan vielä täysin valmiita. Tyylioppaan avulla myös käyttöliittymäkehitys itsessään nopeutuu, koska käytössä olevia komponentteja ei tarvitse etsiä verkkosivustolta. Käyttöliittymäkehityksen aikaistaminen ja nopeutuminen mahdollistavat sen, että käyttöliittymästä voidaan saada palautetta eri sidosryhmiltä kuten asiakkaalta ja käyttöliittymäsuunnittelijalta nopeammin ja

mahdollisiin muutostarpeisiin ehditään reagoimaan aikaisemmassa vaiheessa projektia. (Fordham 2014; LoPresto 2016; Sullivan 2014.)

Anna Debenham (2017, 25–26) mainitsee tyylioppaan nopeuttavan komponenttien testaamista eri selaimilla. Sen sijaan, että kaikki sivuston sivut testataan erikseen, käyttöliittymätason testaus voidaan tehdä osittain tyylioppaassa. Myös käytettävyys- ja saavutettavuusongelmien tunnistaminen on Debenhamin mukaan nopeampaa eristetyssä ympäristössä. Tyyliopaslähtöisessä kehitysmetodissa kehityksen nopeutuessa myös komponenttien testaus voidaan aloittaa aikaisemmin.

5 Tyylioppaan generointityökalujen arviointi

5.1 Vaatimusten tunnistaminen ja koostaminen

5.1.1 Työkalun asentamiseen ja käyttämiseen liittyvät vaatimukset

Elävä tyyliopas toimii tyyliopaslähtöisen käyttöliittymäkehityksen kulmakivenä. Jotta tyyliopaslähtöisyydestä on mahdollisimman paljon hyötyä, tyylioppaan generointityökalun ja sillä luodun tyylioppaan tulee täyttää kehitysmallin asettamat vaatimukset. Koska tyyliopasta päivitetään ennen verkkosivustoa, tyylioppaan päivitys ei saa De La Cuadran (2016) mukaan muodostua kehitystyön pullonkaulaksi. Jos tyylioppaan muokkaus on hidasta, menetetään tyyliopaslähtöisyyden mahdollistama kehitysprosessin nopeutuminen. De La Cuadra mainitseekin, että generaattoriksi kannattaa valita sellainen työkalu, joka on helppo asentaa ja jota on helppo käyttää.

Helppouden arviointi on haasteellista, sillä työkalun asentamisen ja käyttämisen helppous riippuu käyttäjästä itsestään. Asennusprosessissa tärkeässä roolissa on työkalun oma asennusdokumentaatio, jota seuraten asennusta yleensä tehdään. Asentamisen helppouden sijaan vaatimukseksi voidaankin johtaa asennusdokumentaation kattavuus ja virheettömyys. Työkalun tarkoituksena on automatisoida tyylioppaan luonti ja ylläpito, joten työkalun käytön helppoutta voidaan arvioida sen päätehtävän eli automatisoinnin kautta. Työkalun tulisi seurata muutoksia ja päivittää tyyliopasta automaattisesti heti muutoksia havaitessaan ilman erillisten komentojen ajamista. Päivityksen tulisi tapahtua myös nopeasti, jotta se ei hidastaisi kehitystyötä.

Työkalun asentamiseen ja käyttämiseen liittyvät vaatimukset tiivistettynä:

- työkalun asennusdokumentaatio on kattava ja virheetön
- työkalu päivittää tyyliopasta automaattisesti
- tyyliopas päivittyy nopeasti.

5.1.2 Komponenttikirjastoon liittyvät vaatimukset

Komponenttikirjaston osalta voidaan tarkastella kappaleessa 3.2 listattuja komponenttikirjaston ominaisuuksia ja pohtia tyyliopaslähtöisen kehitysmallin vaatimuksia niiden kautta. Tyylioppaan merkitys kommunikointivälineenä korostuu tyyliopaslähtöisessä käyttöliittymäkehityksessä. Jotta tyyliopaslähtöisyydellä saavutetaan termistöön liittyvät edut, yksittäiselle komponentille tulee voida määrittää nimi ja kuvaus. Myös lisätietojen, kuten käyttöohjeiden ja lähdelinkkien, määrittäminen komponentille on tarpeellista, jotta komponenttia koskevista suunnittelu- ja kehityspäätöksistä ja käyttökohteista voidaan viestiä tehokkaasti. Komponentille tulisi voida myös lisätä tieto komponentin suunnittelijasta ja kehittäjästä. Tämä tieto helpottaa kommunikointia ja tiedonjakoa jatkossa, koska tylioppaasta voidaan tarkistaa komponentista eniten tietävät henkilöt. Jotta yksittäisestä komponentista viestiminen olisi helppoa, komponentille tulisi muodostua oma linkki, johon voidaan viitata ja joka voidaan tarvittaessa jakaa muille projektitiimiin kuuluville.

Tyyliopaslähtöisessä kehitysmallissa komponentit luodaan ja niitä muokataan tylioppaassa, joten niiden ulkoasun ja käyttäytymisen tulee olla tarkistettavissa tylioppaasta. Yksittäiselle komponentille tulisi siis olla lisättävissä elävä esimerkki tyyliopasta varten. Jotta koko käyttöliittymäkehitys voidaan tehdä alusta loppuun tylioppaan kautta, esimerkin tulee olla responsiivinen ja sen tulostustyylien tarkistettavissa oppaan kautta. Kehitystyön päätteeksi komponentti siirretään varsinaiselle verkkosivustolle. Tällöin on tärkeää, että elävässä esimerkissä käytetty HTML-merkkaus on kopioitavissa, jotta komponentit siirtäminen verkkosivustolle on nopeaa. Jotta komponentin kokonaisvaltainen kehittäminen tylioppaan kautta olisi mahdollista, HTML-merkkauksen lisäksi muiden komponenttiin liittyvien koodien, kuten CSS- ja JavaScript-koodien tulisi näkyä tylioppaassa.

Jos tyylioppaassa näytetään useampi komponentti samalla sivulla, yksittäinen komponentti tulee voida avata omaan ikkunaansa komponenttikohtaista kehitystä ja testausta varten. Toisaalta, koska komponentit ovat käyttöliittymän irrallisia osia, kokonaisuuden hahmottaminen tyylioppaan kautta voi olla hankalaa. Erityisesti tyyliopaslähtöisessä kehitysmallissa on tärkeää tietää mihin kaikkiin komponenttikokonaisuuksiin yksittäisen komponentin muokkaaminen vaikuttaa. Jotta komponenttien keskinäinen riippuvuus olisi erotettavissa, tyylioppaaseen tulisi voida määrittää automaattisia viittauksia komponenttien kesken.

Komponenttien kehitystyö aloitetaan tyylioppaassa, joten yksittäisestä komponentista tulee käydä ilmi sen status. Status viestii siitä, onko komponentin kehitys vielä kesken vai onko komponentti valmis siirrettäväksi verkkosivustolle ja hyödynnettäväksi esimerkiksi komponenttikokonaisuuksissa. Komponentin statuksen lisäksi komponentille olisi tarpeellista muodostua automaattinen versionumero, jotta tyylioppaasta voidaan tarkistaa, mikä komponenttiversio julkaistussa verkkosivustossa on käytössä. Testauksen näkökulmasta kaikkien komponenttien tulisi listautua yhdelle sivulle, jotta komponenttien testaus nopeutuisi ja kaikille komponenteille voidaan ajaa esimerkiksi käytettävyyteen ja saavutettavuuteen liittyviä testejä samalla kertaa.

Komponenttikirjastoon liittyvät vaatimukset tiivistettynä:

- komponentille voidaan määrittää nimi
- komponentille voidaan määrittää kuvaus
- komponentille voidaan lisätä käyttöohjeita ja linkkejä lähteisiin
- komponentille voidaan lisätä tieto sen tekijöistä
- komponentille muodostuu linkki
- komponentille voidaan lisätä elävä esimerkki
- elävä esimerkki on responsiivinen
- elävässä esimerkissä saa näkyviin tulostustyylit
- elävän esimerkin koodi on näkyvillä ja kopioitavissa
- kaikki komponenttiin liittyvät koodit ovat näkyvillä
- komponenttien väliset riippuvuudet ovat määritettävissä
- komponentti voidaan avata omaan ikkunaansa
- komponentilla on oma statustieto

- komponentilla on oma versiotieto
- komponentit saadaan listattua yhdelle sivulle.

5.1.3 Tyylioppaan rakenteeseen liittyvät vaatimukset

Tyyliopas toimii suunnittelun ja kehityksen lähtökohtana, joten tyyliopasta tulisi olla mielekästä käyttää. Frostin (2016b, luku 5) mukaan ulkoasultaan houkuttelevan ja projektin näköinen tyyliopas saa projektin jäsenet todennäköisemmin hyödyntämään opasta. Generointityökalulla luotava tyyliopas on ilman muokkausta geneerisen eli persoonattoman näköinen. Jotta tyyliopasta saadaan projektin näköinen, generaattorilla luodun tyylioppaan ulkoasun tulee olla muokattavissa. Mielenkiintoisen ulkoasun lisäksi tyyliopasta käytetään todennäköisemmin, jos sieltä on helppo löytää tarvitsemansa tiedot haun tai navigaation kautta (Frost 2016b, luku 5). Verkkosivuston kehittyessä ja tylioppaan monipuolistuessa haun merkitys kasvaa. Erityisesti tyyliopaslähtöisyyden kannalta tylioppaassa tulisi olla hakutoiminnallisuus, jonka avulla suunnittelun ja kehityksen kannalta relevanttien tietojen löytäminen voidaan varmistaa.

Tylioppaan rakenteeseen liittyvät vaatimukset tiivistettynä:

- tylioppaan ulkoasu on muokattavissa
- tylioppaan navigaatio on selkeä
- tylioppaassa on haku.

5.1.4 Ohjesivuihin liittyvät vaatimukset

Komponenttikirjaston lisäksi tyyliopas voi sisältää erilaisia verkkosivuston ylläpitoon liittyviä ohjeistuksia. Näitä varten oppaaseen tulee voida lisätä tekstimuotoisia sivuja ilman, että sivuilla näytetään komponentteihin liittyviä osiota kuten esimerkkikoodia.

Ohjesivuihin liittyvät vaatimukset tiivistettynä:

- tylioppaaseen voidaan lisätä sivuja ohjeistuksia varten.

5.2 Arvioitavien työkalujen valinta

Opinnäytetyön tekemiseen varattu aika on rajallinen ja tyylioppaan generointityökaluja on olemassa useita. Kaikkia työkaluja ei tässä tutkimuksessa voida kattavasti arvioida, joten tutkimus rajattiin neljän suosituimman generointityökalun arviointiin. Suosituimmuus valittiin työkalujen valintakriteeriksi toimeksiantajan pyynnöstä.

Anna Debenham (2017, 56) mainitsee, että styleguides.io-palvelusta löytyy kattavin lista olemassa olevista tyyliopasgeneraattoreista. Styleguides.io on käyttöliittymäammattilaisten yhteistyössä ylläpitämä sivusto, jossa jaetaan erilaisia käyttöliittymän tyylioppaan luontiin liittyviä vinkkejä aina artikkeleista työkaluihin. Työkalut löytyvät osoitteesta http://styleguides.io/tools. Palvelun työkalulistauksen pohjalta koostettiin tutkimusta varten oma lista generointityökaluista, jotka täyttävät seuraavat toimeksiantajan työkalua varten asettamat vaatimukset:

- työkalu on ilmainen
- työkalu toimii Gruntilla
- työkalu ei ole ohjelmointikieli- tai tekniikkasidonnainen.

Työkalun käyttöönottamisen ja vaihtamisen tulee olla mahdollisimman joustavaa ja nopeaa, joten tutkimuksesta rajattiin pois maksulliset työkalut. Toimeksiantajan projekteissa käytetään JavaScript-pohjaisten toimintojen automatisoinnissa pääsääntöisesti Grunt-työkalua, minkä vuoksi myös generointityökalun tulisi olla ajettavissa Gruntin avulla. Työkaluista rajattiin pois myös ohjelmointikieli- ja tekniikkasidonnaiset generaattorit, jotta tutkimuksen arvioinnin tulosta voidaan soveltaa useaan eri projektiin tekniikasta riippumatta. Tällaisia ohjelmointikieli- ja tekniikkasidonnaisia työkaluja ovat esimerkiksi vain PHP- tai Java-kielillä toimivat tai Angular-, React- tai Vue-sovelluskehystä varten luodut generaattorit.

Työkalujen kartoittamisen jälkeen tutkittiin niiden suosituimmuutta. Tässä tutkimuksessa sovellusten suosituimmuus perustuu GitHubin tähditysjärjestelmään, sillä kaikki tarkastelun alla olevat työkalut käyttävät GitHubia kehitysalustanaan ja projekteihin liittyvä kehitys ja kommunikointi kuten virheraportoinnit keskittyvät sinne. GitHub on kehitysalusta Git-versionhallintaa käyttäville ohjelmakehitysprojekteille ja sitä käyttää 1,8 miljoonaa eri yritystä ja organisaatiota. GitHubissa käyttäjät voivat

tähdittää eli eräällä tavalla tykätä ohjelmavarastoja (englanniksi repository). Tähtien avulla käyttäjien on helpompi löytää kiinnostavat ohjelmavarastot uudelleen ja myös osoittaa arvostuksensa varaston ylläpitäjän tekemää työtä kohtaan. (The world's leading software development platform · GitHub n.d; About stars n.d.). Joistakin työkalusta on olemassa useita eri versioita, joista yksi on Gruntilla toimiva. Tällaisissa tilanteissa vertailussa käytettiin alkuperäisen työkalun suosituimmuutta, sillä Grunt-versio on vain sovitin.

Tähditysten vertailussa arvioitaviksi tyyliopasgeneraattoreiksi valikoituivat seuraavat työkalut:

- KSS (https://github.com/kneath/kss), 4048 tähteä
- Devbridge Styleguide (https://github.com/devbridge/Styleguide), 1471 tähteä
- SC5 Style Guide Generator (https://github.com/SC5/sc5-styleguide), 1221 tähteä
- Fractal (https://github.com/frctl/fractal), 1119 tähteä

Liitteessä 1 näkyy listaus kaikista tähtien perusteella vertailluista tyylioppaiden generointityökaluista.

5.3 Työkalujen arviointi

Tyylioppaiden generointityökalujen arviointi suoritettiin virtuaalisessa kehitysympäristössä, jonka käyttöjärjestelmänä on Windows 7. Kehitysympäristöön oli esiasennettu Node.js, npm, Grunt ja grunt-sass, joita tarvitaan generointityökalujen asentamiseen ja käyttämiseen. Tutkimuksessa on käytetty seuraavia versionumeroita:

- Node.js 8.11.0
- npm 5.8.0
- Grunt 1.0.2.
- grunt-sass 2.1.0

Seuraavissa luvuissa arvioidaan tutkimukseen valittuja generointityökaluja luvussa 5.1 tunnistettujen vaatimusten pohjalta.

5.3.1 KSS (grunt-kss)

KSS, aiemmin Knyle Style Sheets, on Kyle Neathin ja Tom Preston-Wernerin luoma CSS:n dokumentointityökalu. KSS toimi alun perin Ruby-ohjelmointikielellä, mutta siitä on tehty versioita eri ohjelmointikieliin ja -tekniikoihin sopivaksi. Esimerkiksi kss-node -versiota voidaan käyttää sellaisenaan Node.js:n avulla. Grunt- ja Gulp-työkaluille on olemassa myös omat versionsa. Tässä tutkimuksessa käytettiin KSS:n Grunt-versiota eli grunt-kss -työkalua, joka löytyy GitHubista osoitteesta https://github.com/kss-node/grunt-kss. Version ovat luoneet Koji Ishimoto ja John Wilkins. Työkalun asentamisessa ja asetusten tekemisessä seurattiin KSS:n dokumentaatiota osoitteissa https://github.com/kss-node/grunt-kss ja https://github.com/kneath/kss/blob/master/SPEC.md. Tutkimuksessa käytetty grunt-kss:n versionumero on 5.0.1.

Työkalun asentaminen ja käyttäminen

KSS:n Grunt-versio asennetaan navigoimalla komentorivillä kansiosijaintiin, jonne Grunt on asennettu ja ajamalla seuraava komento:

```
npm install grunt-kss --save-dev
```

Asentamisen jälkeen Gruntin asetustiedostoon eli Gruntfile.js-tiedostoon määritetään halutut asetukset, joita käytetään tyylioppaan generoinnissa. Tässä tutkimuksessa käytettiin seuraavia asetuksia:

```
module.exports = function(grunt) {
    require('load-grunt-tasks')(grunt);
    grunt.initConfig({
        kss: {
            options: {
                css: ['../generated/styles.css'],
                verbose: true
            },
            dist: {
                src: ['styles'],
                dest: 'styleguide-kss'
            }
        },
        sass: {
            options: {
                sourceMap: true
```

```
            },
            dist: {
                files: {
                    'generated/styles.css': 'styles/styles.scss'
                }
            }
        },
        watch: {
            sass: {
                files: 'styles/**/*.*',
                tasks: ['sass', 'kss']
            }
        }
    });
};
```

Työkalun asentaminen ja asetusten teko sujuivat ilman ongelmia ja nopeasti dokumentaation avulla. Tutkimuksen aikana ei huomattu virheitä tai vajavaisuuksia työkalun asentamiseen liittyvissä ohjeistuksissa.

Työkalua käytetään Gruntin avulla. Gruntfile.js-tiedostoon voidaan määrittää watch-tehtävä, joka valvoo tyylitiedostoja. Jos tyylitiedostoihin tulee muutoksia, generaattori päivittää tyyliopasta reaaliaikaisesti. Tyylioppaan generointiin ei tarvita erillistä palvelinta ja työkalu generoi oppaan suhteellisen nopeasti. Tutkimuksen pienellä tyylimäärällä generointiin kului 1-3 sekuntia.

Komponenttikirjasto

KSS muodostaa tyylioppaan tyylitiedostoihin tehtyjen kommenttiosioiden perusteella. Tutkimusta varten luotiin painikkeille buttons.scss-tiedosto, jonka sisältö oli seuraavanlainen:

```
// Buttons
//
// All button styles used in the web site.
//
// Author: Johanna Ström
//
// Markup:
// <button class="btn {{modifier_class}}">Button</button>
//
// .primary   - Primary action button.
// .secondary - Secondary action button.
// :disabled  - Button disabled effects.
//
// Styleguide Buttons
```

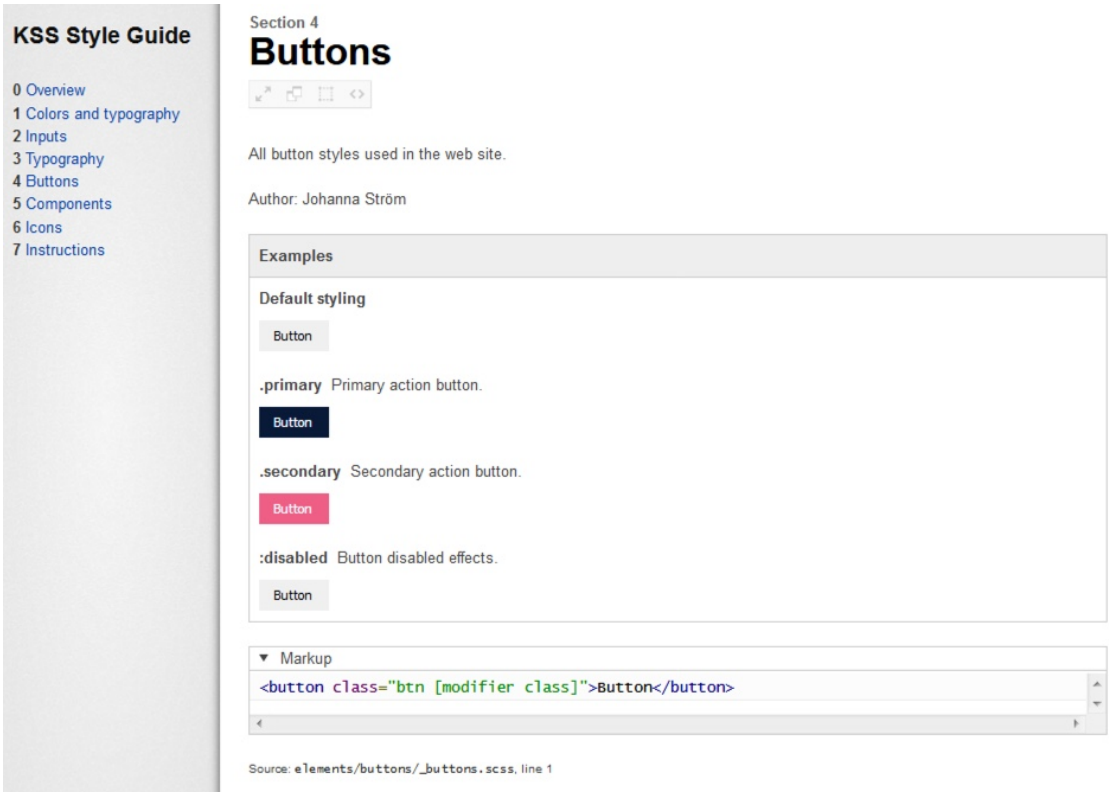
```
//
// Deprecated: This button is deprecated since 11.4.2018
//

.btn {
    border: solid 1px transparent;
    padding: 0.4rem 1rem;

    &.primary {
        background - color: map-get($colors, primary);
        border - color: map-get($colors, primary);
        color: #fff;

        &:hover {
            background-color: grey;
            border-color: grey;
        }
    }
    // Loput tyylit tähän…
}
```

Edellä kuvatun merkkauksen pohjalta KSS generoi kuviossa 6 näkyvän sivun tyylioppaaseen.



Kuvio 6. KSS-työkalulla generoitu komponentti

Komponentille voidaan lisätä nimi ja kuvaus, jotka näkyvät tyylioppaassa heti komponentin alussa. Tekijöiden nimien lisäämiselle ei varsinaisesti ole omaa käsittelyä,

mutta ne voidaan lisätä kuvauksen jälkeen omalle rivilleen. Komponentin käyttöohjeet ja lähteitä voidaan lisätä myös samaan kenttään.

Komponenteille voidaan lisätä esimerkki, joka näkyy sekä elävänä versiona että koodina tyylioppaassa. HTML-merkkaus on näkyvillä ja kopioitavissa tyylioppaasta verkkosivustolle. Tyylioppaassa näytetään vain komponentin HTML-merkkaus eivätkä komponenttiin liittyvät CSS- tai JavaScript-koodit näy tyylioppaassa ollenkaan.

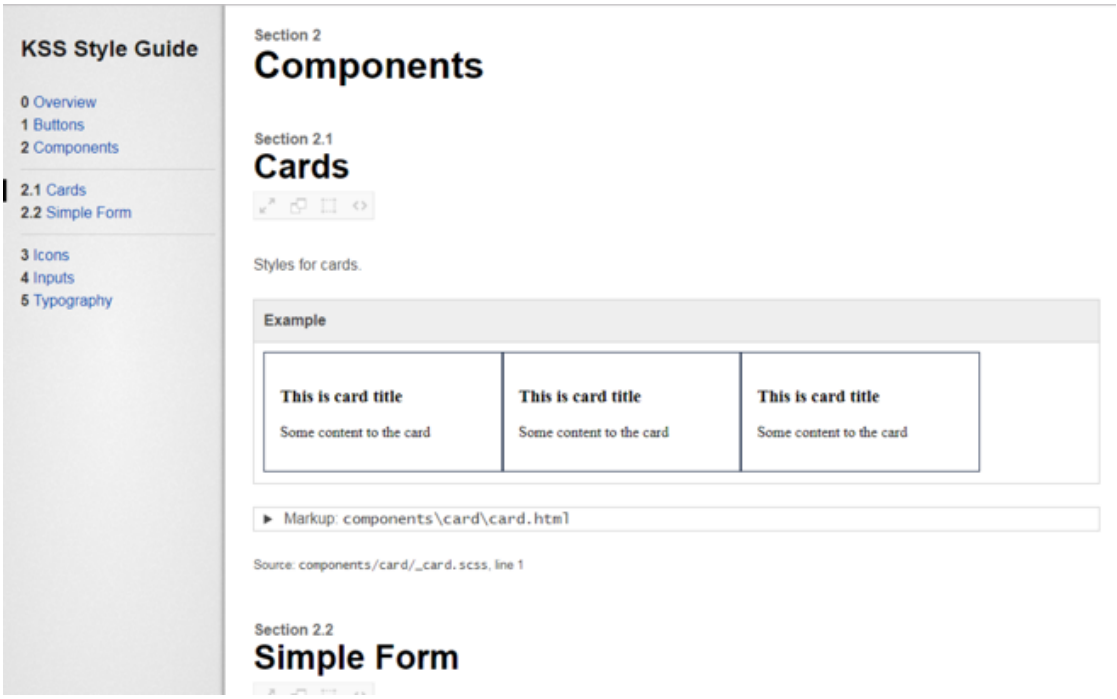
Komponenttien välisiä riippuvuuksia ja käyttökontekstia ei saa KSS-työkalululla luotua automaattisesti. Jos komponentissa halutaan viitata toiseen komponenttiin, viittaus tulisi tehdä käsin, mikä ei ole ylläpidettävyyden kannalta suositeltavaa.

Komponentti voidaan avata tyylioppaassa omaan ikkunaan kehitystä ja testausta varten. Tällöin sen responsiivista käyttäytyminen ja tulostustyylit voidaan tarkistaa muista komponenteista riippumatta. Komponentille muodostuu oma linkki, jonka avulla se on nopeampi löytää jatkossa.

Komponentille voidaan määrittää statukseksi experimental tai deprecated. Jos komponenttiin on merkitty kumpikaan edellä mainituista arvoista, se lakkaa näkymästä tyylioppaan navigaatiossa. Komponentin sivulle pääsee kuitenkin yhä suoralla linkillä. Komponentin häviäminen navigaatiosta vaikuttaa viestinnällisestä näkökulmasta ongelmalliselta. Erityisesti kehityksen alla olevat komponentit tulisi voida merkitä siten, että ne eivät katoa tyylioppaasta. Statuksen määrittäminen on siis mahdollista, mutta tyyliopaslähtöisessä kehitysmetodissa ominaisuus on käyttökelvoton. Automaattista versionumerointia ei komponenteille muodostu.

Kaikki samaan osa-alueeseen liittyvät komponentit listautuvat samalle sivulle, mutta kaikkien osa-alueiden komponentteja ei saa listattua yhdelle sivulle, ellei niitä kopioi jälleen uuteen scss-tiedostoon. Tämä aiheuttaisi koodin toistamista ja vaikeuttaisi ylläpitoa, eikä siksi ole suositeltavaa.

Tyylioppaan rakenne



Kuvio 7. KSS-työkalulla generoidun tyylioppaan ulkoasu

KSS:n generoiman tyylioppaan ulkoasu on ilman muokkauksia persoonaton ja vaikeaselkoinen. Navigaatiorakenne on epäselvä erityisesti alaosioiden puuttuvien sisennysten takia ja dokumenttiosiossa olevia otsikoita on välillä vaikea erottaa toisistaan (ks. kuvio 7). Tyylioppaaseen voidaan luoda uudet tyylit, minkä avulla siitä saa käytettävämmän ja houkuttelevamman. Tyylioppaan omien tyylien muokkaus onkin tarpeellista, jotta oppaan rakenne olisi paremmin hahmotettavissa ja sitä olisi miellyttävämpi käyttää. KSS-työkalua varten on luotu myös joitakin valmiita teemoja, joilla ulkoasun muuttaminen selkeämmäksi on nopeaa.

KSS:ssä ei ole oletuksena hakutoiminnallisuutta. Haku voidaan kuitenkin lisätä erillisellä kss-search -lisäosalla. Lisäosa on huonosti dokumentoitu eikä se asentunut tutkimustilanteessa oikein. Lisäosan toimivuutta ei saatu testattua tässä tutkimuksessa.

Ohjesivut

KSS-työkalussa ei ole tapaa luoda pelkkää tekstimuotoista ohjeistusta sisältäviä sivuja. Uuden sivun lisääminen onnistuu vain luomalla uusi tyylitiedosto. Tyylitiedosto

ei ole semanttisesti oikea paikka pitkälle ohjetekstille. Lisäksi jokaisella tyylioppaan sivulla näkyy esimerkiksi elävälle esimerkille ja sen koodille varatut osiot, jotka ovat ohjesivulla turhia.

5.3.2 Devbridge Styleguide

Devbridge Styleguide on kansainvälisen ohjelmistoyrityksen Devbridge Groupin vuonna 2015 luoma tyylioppaiden generointityökalu. Työkalu löytyy GitHubista osoitteesta https://github.com/devbridge/Styleguide. Työkalua voidaan käyttää sekä Gruntilla että Gulpilla. Tutkimuksessa käytetty versionumero on 0.4.17.

Työkalun asentaminen ja käyttäminen

Työkalun asentaminen projektiin suoritetaan seuraavalla komennolla:

```
npm install devbridge-styleguide --save-dev
```

Asentamisen yhteydessä pitää ajaa myös seuraava komento, jotta DevBridge Styleguide asentuu globaalisti koko kehitysympäristöön:

```
npm install devbridge-styleguide -g
```

Tyyliopas voidaan luoda styleguide initialize -komennolla, jolloin opas generoidaan styleguide-kansioon. Tutkimusta varten tyyliopas haluttiin asentaa eri nimiseen kansioon, joten tyylioppaan generointi käynnistettiin seuraavalla komennolla:

```
styleguide initialize styleguide-devbridge
```

Devbridge Styleguide loi joukon kansioita ja tiedostoja, mutta ei vielä tyyliopasta. Ennen tyylioppaan varsinaista luontia asetukset pitää määrittää Gruntfile.js-tiedostoon. Tässä tutkimuksessa Gruntfile.js-tiedostossa käytettiin seuraavia asetuksia:

```
module.exports = function(grunt) {
    require('load-grunt-tasks')(grunt);
    var styleguide = require('devbridge-styleguide');
```

```
    grunt.registerTask('generatestyleguide', function() {
        var done = this.async();
        styleguide.startServer().then(function(instance) {
            instance.on('close', done);
        });
    });
};
```

Edellä mainituilla asetuksilla tyylioppaan pitäisi generoitua ajamalla seuraava komento komentokehotteessa:

```
grunt generatestyleguide
```

Tyyliopas ei kuitenkaan generoitunut, sillä työkalu yritti etsiä config.txt-tiedostoa styleguide-kansiosta. Kyseistä kansiota ei ollut olemassa, koska asennus tehtiin styleguide-devbridge -kansioon. Ratkaisu ongelmaan löytyi dokumentaation Gulp-osiosta. Gruntfile.js-tiedoston asetuksissa pitääkin määrittää tyylioppaan polku, jos se on eri kuin oletuspolku:

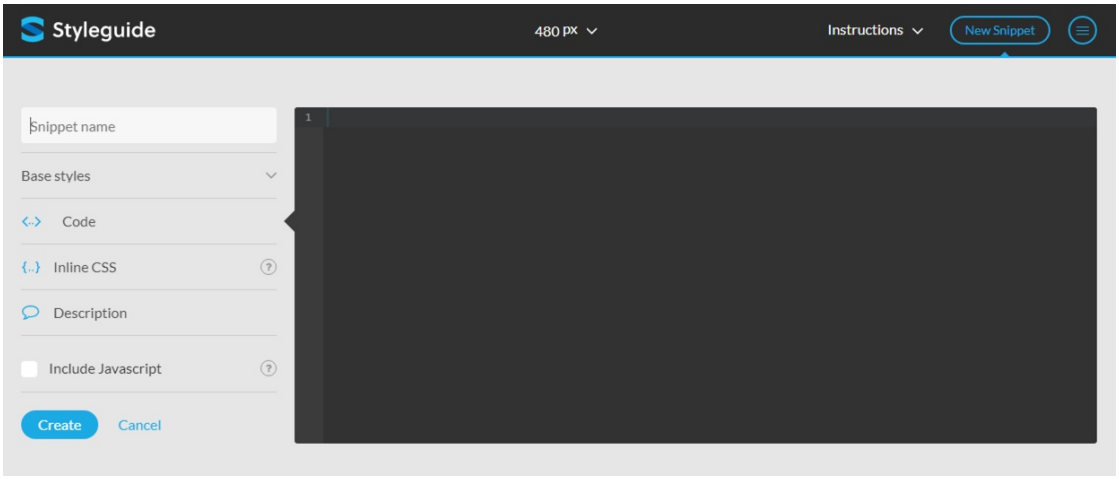
```
grunt.registerTask('generatestyleguide', function() {
    var done = this.async();
    styleguide.startServer({
        styleguidePath: 'styleguide-devbridge'
    }).then(function(instance) {
        instance.on('close', done);
    });
});
```

Korjauksen jälkeen generatestyleguide-tehtävä ajautui läpi onnistuneesti, mutta tyyliopas ei silti generoitunut. Asentamisessa ilmenneiden ongelmien vuoksi tyylioppaan luonti jouduttiin aloittamaan alusta oletuskansiomäärityksiä käyttäen. Määritysten muuttamisen jälkeen styleguide initialize -komento ajettiin uudelleen ilman kansiomääritettä ja tyyliopas asentui styleguide-oletuskansioon. Oletuskansioon asentaminen sujui ilman ongelmia. Puutteellisen ja virheellisen dokumentaation vuoksi tyylioppaan asentaminen oli hidasta ja työlästä.

Tyyliopasta ei voi avata selaimessa ilman palvelinta. Työkalu seuraa palvelimen avulla tyylitiedostoihin tehtyjä muutoksia ja päivittää tyyliopasta automaattisesti. Työkalun käyttö palvelimen kautta hidastutti tyylioppaan selaamista kehitysympäristössä huomattavasti ja sivun päivittäminen oli hetkittäin erittäin hidasta. Työkalun käyttö oli siis tutkimuksen aikana hankalaa ja hidasta.

Komponenttikirjasto

Devbridge Styleguide -työkalulla komponentteja luodaan pääosin suoraan tyylioppaan kautta. Tyylioppaan osioiden luonti on hyvin ohjattu (ks. kuvio 8). Komponenttien esimerkit voidaan luoda suoraan tyylioppaassa selaimella. Komponentteja varten tarvittavia tyylejä ei kuitenkaan pysty tyylioppaassa luomaan tai muokkaamaan, vaan muokkaukset on tehtävä tavalliseen tapaan suoraan tyylitiedostoihin.



Kuvio 8. Komponenttien luominen on ohjattua Devbridge Styleguidella

Komponentille voidaan lisätä nimi ja kuvaus. Komponentin tekijälle ei ole omaa kenttää, mutta tieto voidaan lisätä kuvauskenttään samoin kuin linkit käyttöohjeisiin ja lähteisiin. Komponenttiin saa liitettyä tyylejä, joita käytetään vain tyylioppaassa. Tyylioppaassa kirjoitetut tyylit eivät vaikuta varsinaiseen tyylitiedostoon, jota myös verkkosivusto käyttää. Tämä voi olla hyödyllistä tilanteessa, jossa komponentit osaset, kuten eriväriset painikkeet, halutaan selkeyden vuoksi rivittää tyylioppaassa, vaikka verkkosivustolla ne näkyisivät rinnakkain. Toisaalta kenttä saattaa hämätä luulemaan, että siihen lisätyt tyylit päätyvät myös varsinaiseen tyylitiedostoon ja sitä kautta käyttöön verkkosivustolle.

Komponentille voidaan lisätä elävässä esimerkissä käytettävät HTML-merkkaukset, jotka näkyvät tyylioppaassa sekä esikatselu- että koodimuodossa. Koodiesimerkki on oletuksena piilossa, jolloin tyylioppaan selaaminen on nopeampaa. Koodin saa näkyviin Show code -painiketta klikkaamalla, jolloin se voidaan kopioida verkkosivustolle.

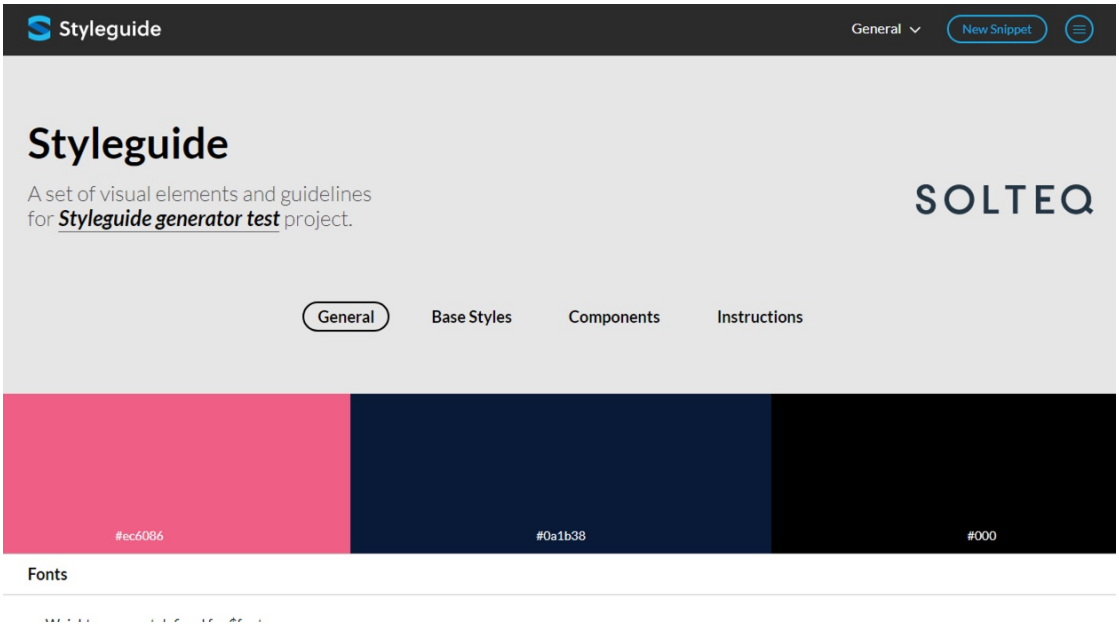
Muita komponenttiin liittyviä koodeja kuten CSS- ja JavaScript-koodeja ei tyylioppaassa näytetä. Komponenttiin ei saa määritettyä riippuvuuksia toisiin komponentteihin.

Kaikki samaan kategoriaan kuuluvat komponentit listautuvat samalle sivulle. Koska yksittäiselle komponentille ei muodostu omaa linkkiä, etsityn komponentin löytäminen voi olla isossa kategoriassa haastavaa. Komponenttia ei voi avata omaan ikkunaansa, minkä vuoksi sen tulostustyylejä ei voi kehittää tai testata tyylioppaassa. Responsiivisen käyttäytymisen voi sen sijaan tarkistaa listausnäkymässä, sillä jokaisella komponentilla on omat asetuksensa leveyden säätöön. Komponenttien esitysleveyttä voi siis säätää listausnäkymässä toisistaan riippumatta.

Komponenteille ei saa lisättyä statustietoa, jolloin tyylioppaasta ei erota mitkä komponentit ovat esimerkiksi vielä työn alla ja mitkä jo valmiita käytettäviksi. Automaattista versionumerointia ei myöskään komponenteille saa.

Kaikki samaan kategoriaan lisätyt komponentit saa listattua samalle sivulle, mutta aivan kaikkia komponentteja ei samalla sivulla listata.

Tyylioppaan rakenne



Kuvio 9. Devbridge Styleguide -työkalulla luodun tyylioppaan ulkoasu

Ulkoasultaan generoituva tyyliopas on selkeän ja houkuttelevan näköinen (ks. kuvio 9). Tyylioppaan omien tyylien muokkauksesta ei ole mainintaa työkalun dokumentaatiossa eikä tutkimuksen aikana kokeilemallakaan löytynyt tapaa muokata tyylioppaan ulkoasua. Oppaan navigointilinkit on piilotettu pudotusvalikon alle, mikä hidastaa navigointia. Komponenttien välillä navigointi vaatii siis aina kaksi hiiren klikkausta yhden sijaan. Tyylioppaassa ei ole hakutoiminnallisuutta.

Ohjesivut

Tyylioppaaseen ei saa luotua uusia sivuja pelkästään tekstimuotoista ohjeistusta varten. Tekstiohjeita voisi lisätä samalla tavalla kuin komponentteja, mutta jättämällä HTML-esimerkki tyhjäksi. Tällöin ohjesivulle kuitenkin tulostuu yhä esimerkille tarkoitettu esikatselunäkymä, joka vie sivuilta tilaa ja hankaloittaa tyylioppaan käyttöä.

5.3.3 SC5 Style Guide Generator

SC5 Style Guide Generator on suomalaisen SC5 Online Oy -yrityksen vuonna 2014 kehittämä tyylioppaan generointityökalu. SC5 Online oli suomalainen pilvipalveluiden ja digitaalisten palveluiden toimittaja. SC5 ja kansainvälinen pilvipalveluiden suunnittelu-, automaatio- ja hallintapalveluita toimittava Nordcloud yhdistyivät tammikuussa 2018, jolloin SC5 Online muuttui Nordcloud Solutioniksi. (Nordcloudista ja SC5:stä Euroopan johtava pilvinatiivien IT-palvelujen tarjoaja 2017.) Työkalun dokumentaatio löytyy osoitteesta https://github.com/SC5/sc5-styleguide. Tutkimuksessa käytetty versionumero on 2.0.4.

Työkalun asentaminen ja käyttäminen

Tyyliopas generoidaan Gulpilla, mutta työkaluun liittyvät asetukset voidaan tehdä Gruntiin käyttämällä grunt-gulp -käännöstyökalua. Tyyliopas generoidaan kuitenkin yhä Gulpilla, joten projektiin tulee olla asennettuna sekä Grunt että Gulp. SC5 Style Guide Generator, Gulp ja Grunt-Gulp -kääntäjä voidaan asentaa samalla lauseella:

```
npm install sc5-styleguide gulp grunt-gulp --save-dev
```

SC5 Style Guide Generatorilla on kattava dokumentaatio, joka on hyödyllinen sekä tyylioppaan luomisessa että sen ylläpidossa ja jatkokehittämisessä. Työkalun asetusten osalta dokumentaatiossa on kuitenkin tyylitiedostojen määrittelyssä virhe, jonka vuoksi dokumentaation esimerkkikoodi ei toimi. Tutkimuksessa päädyttiin käyttämään seuraavia asetuksia Gruntfile.js-tiedostossa:

```
module.exports = function(grunt) {

    require('load-grunt-tasks')(grunt);

    var gulp = require('gulp'),
        styleguide = require('sc5-styleguide');

    grunt.initConfig({
        pkg: grunt.file.readJSON('package.json'),
        gulp: {
            'styleguide-generate': function() {
                var outputPath = 'styleguide-sc5';
                // Return should not be empty as in documentation
                return gulp.src(['styles/**/*.scss'])
                .pipe(styleguide.generate({
                    title: 'Styleguide Generator Test',
                    customColors: '_styleguide-custom-styles.scss',
                    favIcon: 'img/favicon.ico',
                    server: true,
                    showMarkupSection: true,
                    rootPath: outputPath,
                    overviewPath: 'README.md'
                })).pipe(gulp.dest(outputPath));
            },
            'styleguide-applystyles': function() {
                return gulp.src('generated/styles.css')
                .pipe(styleguide.applyStyles())
                .pipe(gulp.dest('styleguide-sc5'));
            }
        },
        sass: {
            options: {
                sourceMap: true
            },
            dist: {
                files: {
                    'generated/styles.css': 'styles/styles.scss'
                }
            }
        },
        watch: {
            sass: {
                files: 'styles/**/*.*',
                tasks: [
                    'sass',
                    'gulp:styleguide-generate',
```

```
                    'gulp:styleguide-applystyles'
                ]
            }
        }
    });
    grunt.registerTask('generatestyleguide', [
        'gulp:styleguide-generate',
        'gulp:styleguide-applystyles',
        'watch'
    ]);
};
```

Asennusdokumentaatiossa oleva virhe hidasti omalta osaltaan asennusprosessia.

Tyylioppaan päivittämistä varten kannattaa luoda watch-tehtävä, joka seuraa tyylitiedostoihin tehtyjä muutoksia ja päivittää tyyliopasta automaattisesti muutoksia havaitessaan. Tyylitiedostojen seuranta voidaan käynnistää tutkimuksessa käytettävien asetusten perusteella komentokehotteessa seuraavalla komennolla:

```
grunt generatestyleguide
```

Komento luo tyylioppaan ja käynnistää palvelimen, jonka avulla tyyliopasta voidaan tarkastella palvelimella osoitteessa localhost:3000. Palvelin tunnistaa automaattisesti tyylioppaaseen tehtävät muutokset, generoi tyylioppaan uudelleen ja päivittää tyylioppaan selaimessa automaattisesti. Tämä toiminnallisuus nopeuttaa käyttöliittymäkehitystä, koska kehittäjän ei tarvitse hyppiä selaimen ja kehitystyökalun välillä tai odottaa selaimessa sivun latautumista. Tyyliopasta voi käyttää myös ilman palvelinta. Työkalu rakentaa tyylioppaan Angularilla, joka on www-sovellusten luomiseen käytettävä JavaScript-pohjainen alusta (What is Angular? n.d.).

Komponenttikirjasto

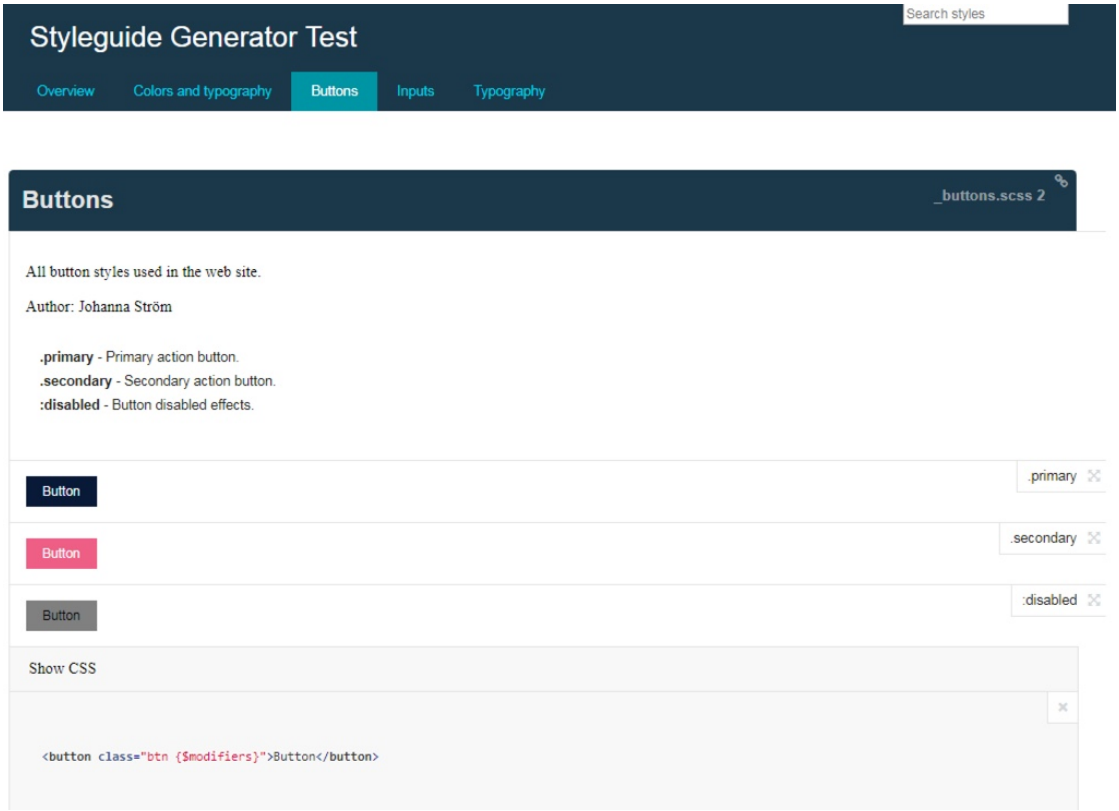
SC5 Style Guide Generator -työkalu käyttää samaa KSS-syntaksia kuin aiemmin arvioitu KSS-työkalu ja tyyliopas generoituu samalla tavalla eli tyylitiedostoihin tehtyjen kommenttien perusteella. Esimerkiksi painikkeille voidaan luoda oma sivu tyylioppaaseen luomalla scss-tiedosto, joka on sisällöltään seuraava

```
// Buttons
//
// All button styles used in the web site.
//
// Author: Johanna Ström
//
// Markup: <button class="btn {$modifiers}">Button</button>
//
// .primary   - Primary action button.
// .secondary - Secondary action button.
// :disabled  - Button disabled effects.
//
// Styleguide 2
//

.btn {
    border: solid 1px transparent;
    padding: 0.4rem 1rem;

    &:disabled {
        background-color: grey;
    }
    // Loput tyylit tähän…
}
```

Koodin pohjalta tyylioppaaseen generoituu sivu, joka käy ilmi kuviosta 10.



Kuvio 10. SC5 Style Guide Generator -työkalulla generoitu komponentti

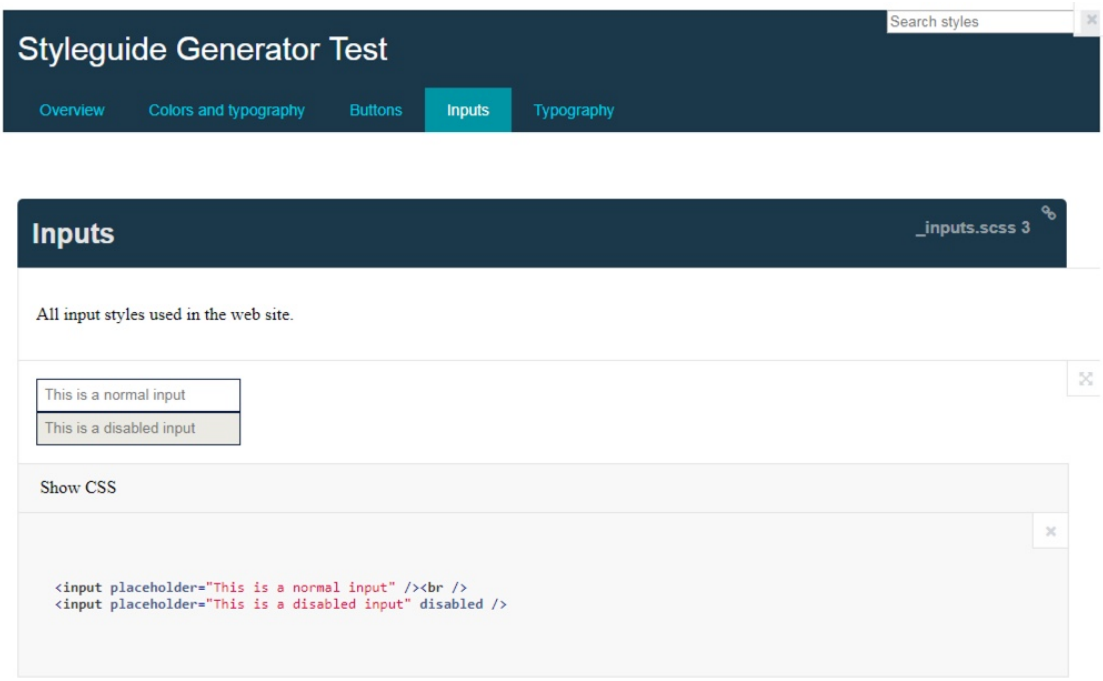
SC5 Style Guide Generatorilla komponenteille voidaan määrittää nimi- ja kuvaustiedot. Tekijälle tai ulkoisille lähteille ei työkalussa ole erillistä käsittelyä, mutta tiedot voidaan lisätä kuvaustietokenttään. Komponentista voidaan luoda esimerkki, joka näkyy sekä esikatseluna että koodina tyylioppaassa. HTML:n lisäksi esimerkin yhteydessä näkyy myös komponenttiin liittyvät tyylit eli CSS-koodit. Koodikentät ovat tarvittaessa piilotettavissa tyylioppaan selailun nopeuttamiseksi. Komponenttiin liittyvää JavaScript-koodia ei kuitenkaan näytetä. Tyylioppaaseen ei myöskään voi määrittää komponenttien välisiä riippuvuuksia, ellei niitä kirjoiteta kuvauskenttään ja ylläpidetä manuaalisesti.

Komponentti voidaan avata omaan ikkunaansa, jolloin sen responsiivisuus ja tulostustyylit ovat tyylioppaan kautta kehitettävissä ja testattavissa. Komponentille muodostuu myös oma linkki, jonka avulla se voidaan nopeasti löytää tyylioppaasta.

Komponenteille saa lisättyä KSS:n sallimat Experimental- ja Deprecated -statukset samalla tavalla kuin KSS-työkalulla. Komponentit, joilla on edes toinen statuksista, eivät näy ollenkaan tyylioppaassa. Tällöin komponenttia ei näe edes suoran linkin kautta. Tämä tekee statustiedon määrittämisestä hyödyttömän tyyliopaslähtöisyyden näkökulmasta. Komponenttia ei pysty kehittämään tai testaamaan tyylioppaassa lainkaan, jos sille on määritetty status. Komponentille ei saa automaattista versiotietoa.

Kaikkia komponentteja ei saa automaattisesti listautumaan samalle sivulle. Komponentit olisi käytännössä mahdollista listata samalle sivulle haun avulla. Toimiakseen tämä tarkoittaa sitä, että jokaiseen komponenttimäärittееseen tulee lisätä esimerkiksi kuvaustekstiin sama sana ja tehdä haku kyseisen sanan avulla. Tämä on kuitenkin vain kiertotie, jonka ylläpito on haasteellista.

Tyylioppaan rakenne



Kuvio 11. SC5 Style Guide Generator -työkalulla generoidun tyylioppaan ulkoasu

Tyylioppaan ulkoasu on selkeä ilman muokkauksiakin, kuten kuviosta 11 voidaan nähdä. Työkaluun on luotu tyylioppaan ulkoasua varten muuttujia, joiden arvoa muuttamalla ulkoasua on nopea muokata. Esimerkiksi koko tyylioppaan fonttiperheen saa muutettua yliajamalla yhden muuttujan arvon tyylioppaan omassa tyylitiedostossa. Tyyliopas on ilman värimuutoksiakin jo käyttökelpoinen, mutta värimuutosten avulla siitä saadaan vieläkin houkuttelevamman näköinen.

SC5 Style Guide Generator -työkalun generoimassa tyylioppaassa on helppo navigoida ja komponenttien etsimistä nopeuttaa hakutoiminnallisuus, joka toimi tutkimuksen aikana hyvin. Minimihakusanan pituus on kolme merkkiä ja haku osaa näyttää hakutuloksia, vaikka hakutermi olisi otettu sanan keskeltä.

Ohjesivut

SC5-styleguide perustuu KSS:ään, joten siinä ilmenee sama ongelma kuin KSS-työkalun kanssa: pelkästään tekstimuotoista ohjeistusta sisältäviä sivuja ei pysty luomaan ilman, että tyylioppaassa ohjesivulla näkyisi turhaan myös koodin esikatseluun ja esittämiseen liittyvät kentät.

5.3.4 Fractal

Fractal on Mark Perkinsin vuonna 2016 luoma komponenttikirjastojen rakentamista helpottava työkalu. Se on luotu Clearleft-yrityksen projektina. Clearleft on englantilainen konsultointia muun muassa strategiseen suunnitteluun ja innovaatioihin tarjoava yritys (Strategic Design & Innovation Consultancy n.d.). Fractalin dokumentaatio löytyy osoitteesta https://fractal.build/guide. Tutkimuksessa käytetty versionumero on 1.1.7.

Työkalun asentaminen ja käyttäminen

Fractal asennetaan Node.js:n pakettihallinnan eli npm:n avulla ajamalla komentokehotteessa seuraava komento:

```
npm install --save @frctl/fractal
```

Työkalua voidaan käyttää SC5 Style Guide Generator -työkalun tavoin Grunt-Gulp -kääntäjällä, jolloin Fractalin asetukset voidaan määrittään Gruntin asetustiedostoon eli Gruntfile.js:ään. Tällön Gruntfile.js-tiedoston sisältö näyttää seuraavalta:

```
module.exports = function(grunt) {
    require('load-grunt-tasks')(grunt);

    const gulp = require('gulp'),
    fractal = require('@frctl/fractal').create();

    fractal.set('project.title', 'Styleguide generator test');
    fractal.web.set('builder.dest', 'styleguide /build');
    fractal.docs.set('path', 'styleguide/docs');
    fractal.components.set('path', 'styleguide/components');

    const logger = fractal.cli.console;

    grunt.initConfig({
        gulp: {
            'fractal-build': function() {
                const builder = fractal.web.builder();
                builder.on('progress', (item, total) =>
                   logger.update('Exported '+item+'/'+total,'info'));
                builder.on('error',err => logger.error(err.message));

                return builder.build().then(() => {
                    logger.success('Fractal build completed!');
                });
            },
            'fractal-start': function() {
                const server = fractal.web.server({
                    sync: true
                });
                server.on('error', err => logger.error(err.message));
                return server.start().then(() => {
                    logger.success('Server running at '+server.url);
                });
            }
        },
        watch: {
            // Server is handling the watch-task…
        }
    });

    grunt.registerTask('generatestyleguide', [
        'gulp:fractal-build',
        'gulp:fractal-start',
        'watch'
    ]);
};
```

Fractal asentui onnistuneesti asennusdokumentaation avulla.

Tyylioppaan generoinnin tulisi käynnistyä komentokehotteessa komennolla

```
fractal new
```

Windows 7 -käyttöjärjestelmän ympäristössä komento ei kuitenkaan toimi, koska Fractalia yritetään ajaa Node.js:n sijaan käyttöjärjestelmän Windows Script Hostilla.

Ongelmasta ja sen korjaamisesta ei ole Fractalin omassa dokumentaatiosta mitään mainintaa, vaan ratkaisu löytyy käyttäjien tekemästä virheraportista osoitteesta https://github.com/frctl/fractal/issues/56. Ongelman voi kiertää luomalla tyylioppaan komennolla

```
fractal.cmd new
```

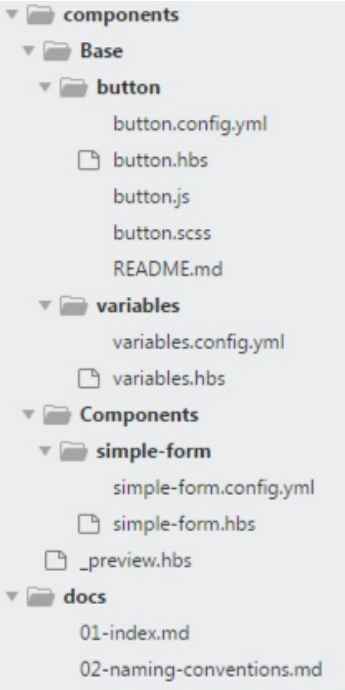
Fractal ohjaa tyylioppaan luontia kysymällä tyylioppaan generoinnin alkuvaiheessa kysymyksiä, joiden perusteella se luo tyylioppaan. Tämä nopeuttaa ja selkeyttää tyylioppaan luontiprosessia. Fractal tarvitsee toimiakseen palvelimen, joka voidaan dokumentaation mukaan käynnistää Fractalilla luodun tyylioppaan kansiosijainnissa komennolla

```
fractal.cmd start
```

Kun palvelin on päällä, työkalu tunnistaa tyylioppaan käyttämiin tiedostoihin tehdyt muutokset ja selaimella avattu tyyliopas päivittyy automaattisesti eikä sivua tarvitsee päivittää manuaalisesti. Palvelimen avulla työkalun käyttäminen on yksinkertaista ja tyyliopas generoituu nopeasti.

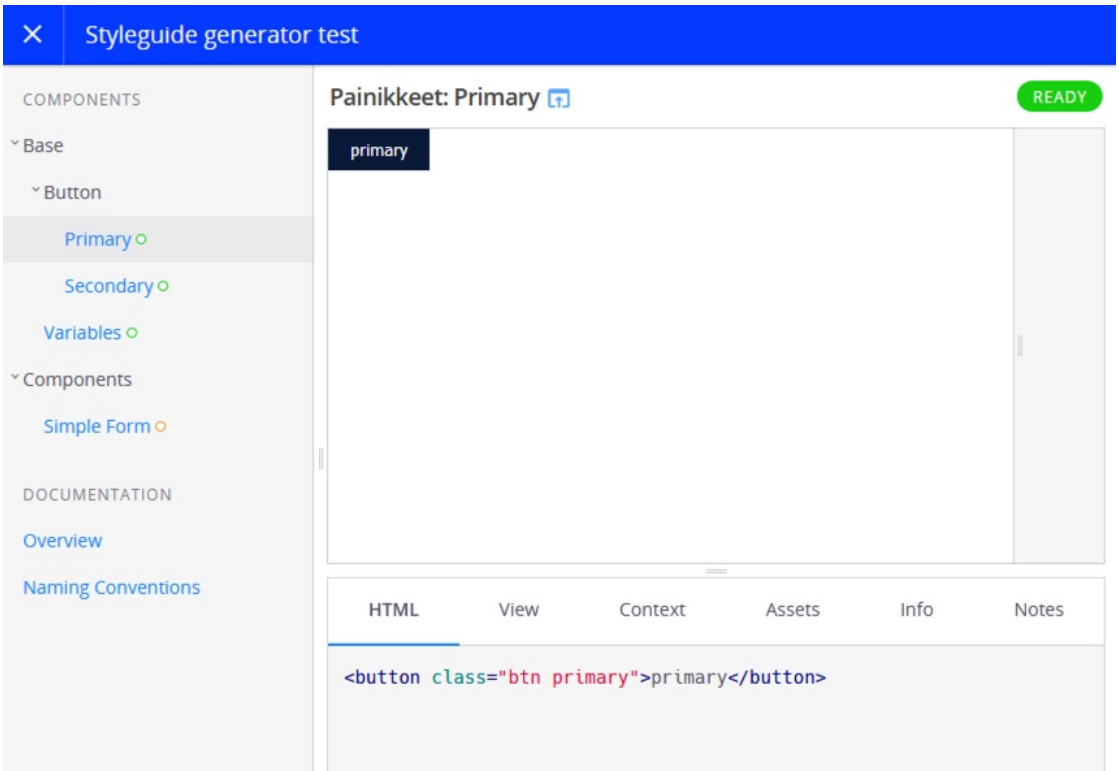
Komponenttikirjasto

Komponentit luodaan suoraan tyylioppaan omaan kansiorakenteeseen tiedostoina. Komponenteille on varattu oma components-kansio, jonka sisälle luodut komponentit poimitaan tyylioppaaseen automaattisesti. Komponenteille luodaan omat kansionsa, joihin voidaan laittaa kaikki komponentteihin liittyvät tiedostot (ks. kuvio 12).



Kuvio 12. Kansio- ja tiedostorakenne Fractalilla luodussa tyylioppaassa

Kuviossa 12 näkyvän tiedostorakenteen perusteella Fractal generoi kuviossa 13 näkyvän tyylioppaan rakenteen.



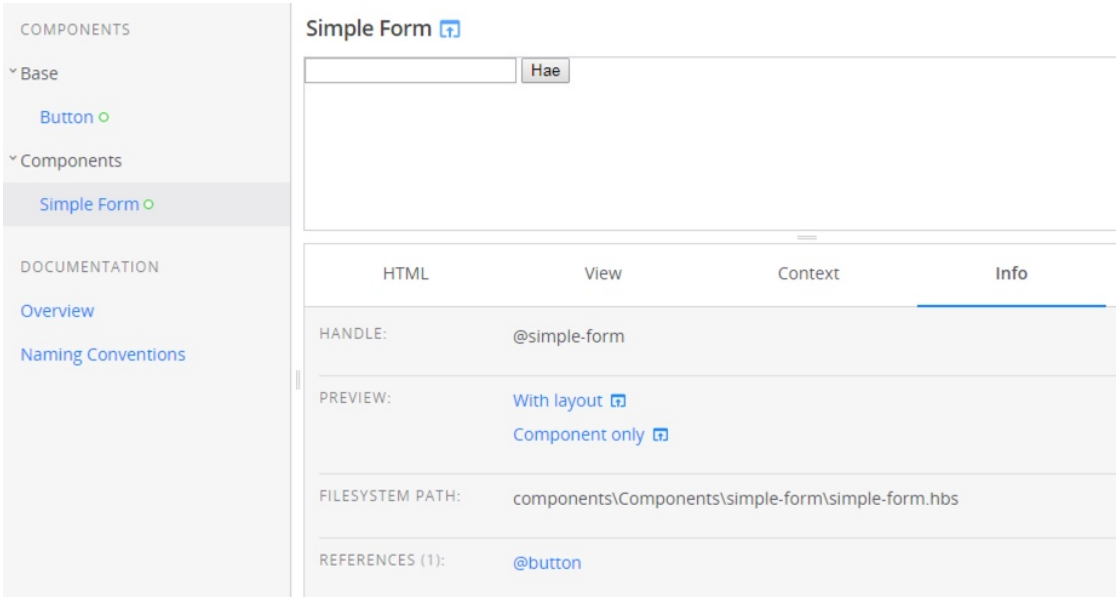
Kuvio 13. Fractal -työkalulla generoitu komponentti

Komponenteille voidaan määrittää nimi, joka näkyy tyylioppaassa heti komponentin alussa. Jokaiselle komponentille voidaan määrittää oma README.md -tiedosto, johon voidaan kirjoittaa lisätietoja komponentista kuten komponentin kuvaus, tieto komponentin tekijästä ja linkkejä ulkoisiin lähteisiin. Tiedoston sisältö näkyy tyylioppaassa komponentin näkymässä Notes-välilehdellä.

HTML-esimerkki kirjoitetaan hbs-tiedostoon, joka on nimetty samalla tavalla kuin kansio, jossa se sijaitsee. HTML:n generointia ei pysty automatisoimaan samalla tavalla kuin KSS -työkalulla, vaan esimerkiksi painikkeen kaikki ilmentymät tulee kirjoittaa erikseen manuaalisesti. Esimerkki näytetään sekä esikatseluna että koodina, joka on kopioitavissa.

Fractal osaa näyttää kaikki komponenttiin liittyvät tiedostot tyylioppaassa, jos tiedostot on nimetty samalla tavalla kuin komponentin esimerkin HTML-merkkauksen sisältävä hbs-tiedosto. Tiedostojen sisällöt ovat nähtävillä ja kopioitavissa. Esikatselussa näkyvä esimerkki voidaan avata uuteen ikkunaan, jolloin sen responsiivisuuden ja tulostustyylien kehittäminen ja testaaminen on mahdollista muista komponenteista riippumatta.

Fractalin avulla tyylioppaaseen voidaan luoda viittauksia komponenttien kesken. Esimerkiksi lomake-komponentissa voidaan käyttää syöttökentän ja painikkeen HTML-merkkausta kirjoittamatta sitä uudelleen. Viittaus myös näytetään tyylioppaassa References-kohdassa, jolloin alkuperäisen komponentti voidaan jäljittää nopeasti (ks. kuvio 14). Näin komponentteja saa myös tarvittaessa yhdisteltyä kokonaisiksi sivuiksi.

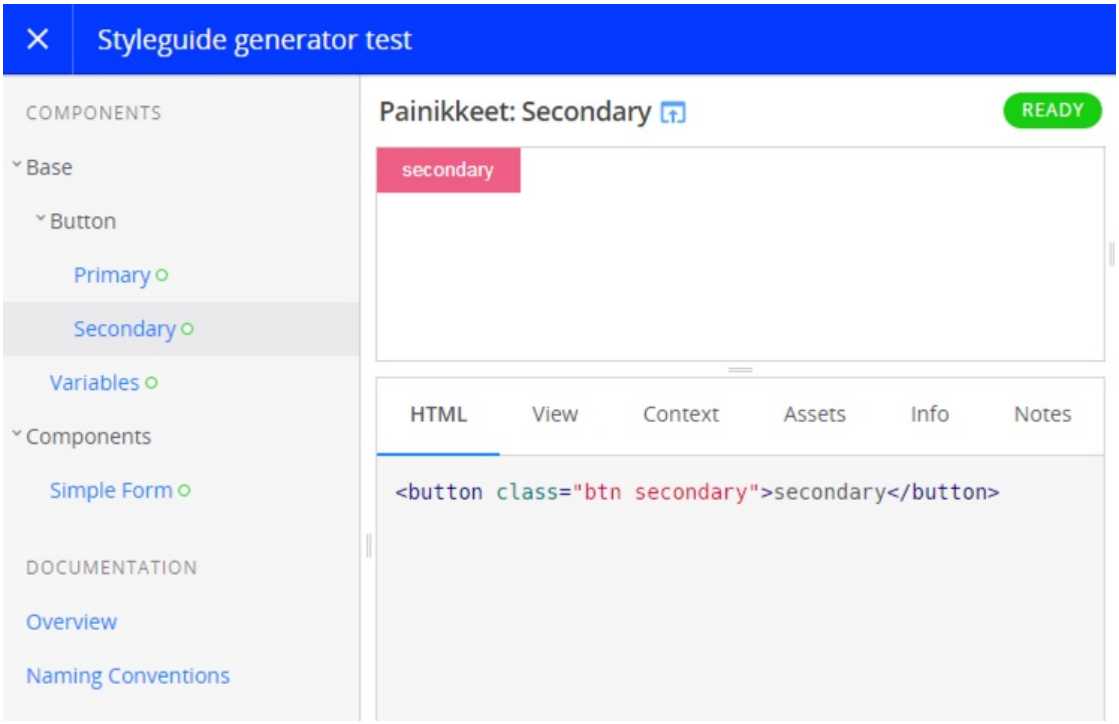


Kuvio 14. Komponenttiviittaus Fractalilla luodussa tyylioppaassa

Jokaiselle komponentille muodostuu oma linkki, jonka avulla komponentti voidaan löytää tyylioppaasta nopeammin. Jokainen komponentti avautuu tyylioppaaseen omana sivunaan eikä kaikkia komponentteja saa näkymään samalla sivulla. Komponentit eivät listaudu edes kategoriakohtaisesti samalle sivulle, vaan yhdellä tyylioppaan sivulla näkyy maksimissaan vain yksi komponentti.

Komponenteille voidaan oletuksena asettaa statukseksi ready eli valmis, wip eli keskeneräinen ja prototype eli prototyyppi. Status näkyy värikoodattuna tyylioppaassa komponenttinäkymän oikeassa reunassa. Tyylioppaaseen voidaan myös lisätä uusia statuksia ja määrittää niille halutut tyylit. Automaattista versionumerointia ei komponenteille saa lisättyä.

Tyylioppaan rakenne



Kuvio 15. Fractal-työkalulla luodun tyylioppaan ulkoasu

Fractalin luoman tyylioppaan ulkoasu on selkeä. Oppaan ulkoasu käy ilmi kuviosta 15. Tyylioppaan omien tyylien muokkaus vaatii uuden tyyliteeman tekemistä. Fractalin dokumentaatio teeman tekemiseen on vajavainen ja teeman rakenne tulee päätellä oletusteemaa tutkimalla. Tämä tekee tyylioppaan personoinnista hidasta. Fractalia varten on luotu joitakin valmiita teemoja, joiden avulla tyylioppaan ulkoasua saa muokattua nopeastikin, mutta näin tyylioppaasta ei saa tehtyä personoitua ja asiakkaan oman brändin mukaista. Tyylioppaan personointi vaatiikin oman teeman tekemistä.

Tyylioppaan navigaatio on selkeä ja eri tasoisten linkkien selkeä sisennys auttaa sisällön rakenteen hahmottamisessa. Komponentit ja dokumentaatio on erotettu omiin osa-alueisiinsa ja niiden välillä navigointi on oppaassa nopeaa. Tyylioppaassa ei ole hakua, minkä vuoksi tyylioppaan kasvaessa ja kehittyessä tiedonhaku saattaa hidastua.

Ohjesivut

Fractalilla luotuun tyylioppaaseen voidaan lisätä uusia sivuja pelkästään tekstimuotoisille ohjeistussivuille, joilla ei tarvitse näyttää koodiosiota. Dokumentaatiota varten tyylioppaassa on oletuksena navigaatiossa Documentation-osio (ks. kuvio 15). Ohjeistussivut luodaan .md-tiedostomuotoisina tyylioppaan docs-kansioon.

6 Johtopäätökset

Tutkimuksessa arvioitiin tyylioppaan generointityökalujen soveltuvuutta tyyliopaslähtöiseen käyttöliittymän kehitysmetodiin. Arvioinnin tulokset löytyvät tiivistettynä liitteestä 2. Seuraavissa kappaleissa käydään läpi tutkimuksen aikana tehdyt huomiot työkalukohtaisesti. Läpikäynnin jälkeen vastataan vielä lyhyesti tutkimuskysymyksiin.

KSS oli arvioitavista työkaluista yksinkertaisin. Sen asentaminen onnistui hyvin asennusdokumentaation pohjalta ja työkalun vaatimien asetusten muokkaus projektikohtaiseksi oli yksinkertaista. Komponenteille oli mahdollista määrittää perustiedot kuten nimi, kuvaus ja esikatseltava esimerkki. Monipuolisempien tietojen, kuten komponenttien statuksen ja komponenttien välisten riippuvuuksien, lisääminen ei kuitenkaan ollut mahdollista. KSS:llä generoidusta tyylioppaasta jäikin puuttumaan useita tyyliopaslähtöisyyden kannalta olennaisia toiminnallisuuksia, kuten haku ja ohjesivujen lisäysmahdollisuus. Tyylioppaan ulkoasu oli persoonaton ja paikoin vaikeaselkoinen eikä sellaisenaan houkuttele käyttämään opasta. Tyylioppaan omien tyylien muokkaus on erittäin suositeltavaa, jos tyyliopas generoidaan KSS-työkalulla. Näiden puutteiden vuoksi työkalua ei voi sellaisenaan suositella tyyliopaslähtöiseen kehitykseen.

Devbridge Styleguide täytti arvioiduista työkaluista huonoiten tyyliopaslähtöisyyden vaatimukset. Työkalu oli puutteellisen ja virheellisen dokumentaation vuoksi hankala asentaa. Komponenttien luonti ja ylläpito olivat sen sijaan hyvin ohjattua ja poikkeuksena muihin arvioituihin generaattoreihin uusien komponenttien luonti onnistui selaimella tyylioppaassa itsessään. Suurimpina ongelmina työkalussa oli tyylioppaan päivittymisen hitaus, komponenttien omien linkkien puute ja hakutoiminnallisuuden

puuttuminen sekä se, ettei tyylioppaaseen voinut lisätä ohjesivuja. Kaikki edellä mainitut asiat vähentävät tyylioppaan arvoa kommunikoinnin apuvälineenä, eikä työkalua voi suositella tyyliopaslähtöiseen käyttöliittymäkehitykseen.

SC5 Style Guide Generator -työkalun dokumentaatio oli kattava ja siitä oli paljon hyötyä asennuksessa ja asetusten teossa. Asentamista käsittelevässä osassa oli kuitenkin virhe, mikä hidasti työkalun asentamista tutkimuksessa huomattavasti. Työkalu oli ulkoasultaan siisti ja ulkoasun muokkaus yksinkertaista. SC5 Style Guide Generator oli tutkimuksessa ainoa, jolla luodussa tyylioppaassa oli haku. Haku myös toimi testattaessa hyvin. Suurimpina ongelmina tyyliopaslähtöisyyden näkökulmasta olivat ne, ettei komponenttien yhteydessä näytetty kaikkia niihin liittyviä koodeja ja ettei tekstimuotoisten ohjeistussivujen luominen ollut mahdollista. Näistä syistä SC5 Style Guide Generator -työkalua ei suositella käytettäväksi tyyliopaslähtöisessä kehitysmetodissa.

Parhaiten tyyliopaslähtöisyyden vaatimuksiin vastasi Fractal. Työkalun asennusdokumentaation avulla työkalu oli nopea asentaa ja tyyliopas päivittyi muutoksia tehdessä nopeasti. Fractalilla luodun tyylioppaan ulkoasu oli selkeä ja se oli muokattavissa yrityksen brändin näköiseksi, joskin muokkaamiseen liittyvä dokumentaatio oli tutkimusta tehdessä keskeneräistä. Fractalin komponenttikirjaston ominaisuudet olivat arvioitavista työkaluista monipuolisimmat. Komponenteille pystyi lisäämään perustietojen kuten nimen ja kuvauksen lisäksi viittauksia toisiin komponentteihin, mikä nopeuttaa tyylioppaan ylläpitoa. Fractal oli myös arvioitavista työkaluista ainoa, johon oli mahdollista lisätä tekstimuotoisia ohjesivuja. Suurin ongelma Fractalissa tyyliopaslähtöisyyden näkökulmasta on hakutoiminnallisuuden puute. Koska tyyliopaslähtöisessä kehitysmetodissa tyyliopas on tärkeässä roolissa, siinä olevien tietojen tulisi olla nopeasti löydettävissä. Ilman hakua tyylioppaasta on hankalampi löytää etsimäänsä tietoa. Tällöin on vaarana, että käyttöliittymään suunnitellaan turhaan uusia komponentteja tai kehitetään samaa koodia kahteen kertaan. Molemmat ovat asioita, joita pyritään tyyliopaslähtöisyydellä vähentämään. Hakutoiminnallisuuden puutteen vuoksi Fractal-työkalua ei tässä tutkimuksessa suositella käytettäväksi tyylioppaan luomiseen ja ylläpitoon tyyliopaslähtöisessä kehitysmetodissa.

Seuraavissa luvuissa vastataan tutkimuskysymyksiin vielä tiivistetyssä muodossa.

Mitä on tyyliopaslähtöinen käyttöliittymäkehitys?

Tyyliopaslähtöinen käyttöliittymäkehitys on kehitysmetodi, jossa tyyliopas toimii kaiken verkkosivuston käyttöliittymään liittyvän suunnittelun ja toteutuksen pohjana. Käyttöliittymäkehitys tehdään ensin tyylioppaassa, minkä jälkeen toteutus viedään varsinaiseen verkkosivustoon tai -sovellukseen. Tyyliopaslähtöisyyden tarkoituksena on tehostaa projektin sisäistä ja eri sidosryhmien välistä kommunikaatiota jakamalla tietoa käyttöliittymään liittyvistä asioista ja nopeuttaa käyttöliittymän kehitysprosessia, koska kehitys voidaan aloittaa tylioppaassa ennen verkkosivuston taustalogiikan valmistumista.

Millaisia vaatimuksia tyyliopaslähtöinen käyttöliittymäkehitys asettaa tylioppaalle ja sen generointityökalulle?

Tyyliopaslähtöinen kehitysmalli asettaa vaatimuksia sekä generointityökalulle että sillä luodulle tylioppaalle. Tässä tutkimuksessa tunnistetut vaatimukset voidaan jakaa karkeasti neljään eri kokonaisuuteen:

- Työkalun asentamiseen ja käyttämiseen liittyvät vaatimukset
- Komponenttikirjastoon liittyvät vaatimukset
- Tylioppaan rakenteeseen liittyvät vaatimukset
- Ohjesivuihin liittyvät vaatimukset

Soveltuuko jokin arvioitavista tylioppaan generointityökaluista käytettäväksi toimeksiantajan tyyliopaslähtöistä kehitysmallia käyttävissä projekteissa?

Tutkimuksessa arvioitiin seuraavia työkaluja: KSS (grunt-kss), Devbridge Styleguide, SC5 Style Guide Generator ja Fractal. Mikään työkaluista ei täyttänyt kaikkia tutkimuksessa tunnistettuja tyyliopaslähtöisyyden asettamia vaatimuksia. Parhaiten vaatimukset täytti Fractal, jota ei voi siltikään suositella käytettäväksi toimeksiantajan projekteissa hakutoiminnallisuuden puutteen vuoksi.

7 Pohdinta

Opinnäytetyön tavoitteena oli tutkia tyyliopaslähtöistä käyttöliittymäkehitystä ja tunnistaa kehitysmallin asettamat vaatimukset käyttöliittymän tyylioppaalle ja sen generointityökalulle. Opinnäytetyön teoreettinen viitekehys muodostettiin käyttöliittymien suunnittelu- ja kehitystyön parissa toimivien asiantuntijoiden kirjoista, artikkeleista ja luennoista. Empiirisessä osuudessa koottiin teorian pohjalta vaatimuslista, jonka avulla arvioitiin neljän työnkalun soveltuvuutta tyyliopaslähtöiseen käyttöliittymäkehitykseen. Seuraavissa kappaleissa pohditaan tutkimuksen tekoon ja tulosten luotettavuuteen liittyviä kysymyksiä.

Aiheen rajaus ja menetelmän valinta

Tutkimuksessa käsiteltävän aiheen rajaus oli haastavaa, sillä tyylioppaan tekemiseen liittyy useita eri asioita ja näkökulmia. Teoriaosuudessa olisi voitu avata käyttöliittymän komponenttiajattelua tarkemmin tai käsitellä tyylioppaan varsinaista sisältöä. Molemmat aiheet rajattiin tutkimuksesta pois, jotta opinnäytetyö ei paisuisi liikaa. Esimerkiksi myös tyylioppaan teknisen toteutuksen käsittely olisi kasvattanut tutkimuksen kokoa huomattavasti. Näistä aiheista voisi aineiston laajuuden perusteella tehdä omat opinnäytetyönsä. Nykyisellä rajauksella työstä kasautui tasapainoinen kokonaisuus, joka tuli valmiiksi toimeksiantajan kanssa sovitussa aikataulussa.

Tutkimusongelmasta johdetut tutkimuskysymykset pyritään ratkaisemaan tutkimusmenetelmiä ja tutkimusaineistoa apuna käyttäen (Kananen 2015, 11). Tutkimus toteutettiin kehittämistutkimuksena, sillä sen aikana koettiin tarpeelliseksi asentaa työkalut ja testata niiden toimintaa tyyliopaslähtöisyyden näkökulmasta. Tutkimus olisi voitu toteuttaa myös työkalujen dokumentaatiota lukien ja näin työkalujen ominaisuuksiin tutustuen. Vain kirjalliseen aineistoon perustuva menetelmä olisi kuitenkin rajannut työstä pois dokumentaation oikeellisuuteen ja riittävyyteen liittyvät kysymykset. Tutkimuksen aikana huomattiinkin, että joidenkin arvioitujen työkalujen dokumentaatio oli osin niin keskeneräistä ja virheellistä, ettei pelkästään dokumentaation avulla olisi voinut muodostaa todenmukaista käsitystä työkalusta.

Tulosten luotettavuus

Kehittämistutkimus ei ole oma tutkimusmenetelmänsä, vaan laadullisista ja määrällisistä tutkimusotteista koostuva kokonaisuus. Tästä ominaispiirteestä johtuen Jorma Kananen (2015, 111) mainitsee kehittämistutkimuksen luotettavuustarkastelun olevan haasteellista ja tarkastelu tulisi tehdä käytettyjen menetelmien mukaan. Tässä tutkimuksessa hyödynnettiin laadullisen tutkimusotteen metodeja, jolloin myös tulosten luotettavuutta arvioidaan laadullisen tutkimuksen luotettavuusmittarein.

Opinnäytetyössä tutkittiin tyyliopaslähtöisyyttä ja sen asettamia vaatimuksia tyylioppaalle ja sen generointityökalulle. Vaikka tyyliopaslähtöisestä kehitysmetodista löytyy artikkeleita jo vuodelta 2013, aiheesta löytyvä teoreettinen tieto keskittyy muutaman eri asiantuntijan kirjoittamiin artikkeleihin ja pitämiin luentoihin. Jorma Kananen (2015, 114) painottaa, että laadullista tutkimusta tekevän tutkijan tulisi pyrkiä varmistamaan tutkimuksessa käyttämänsä tiedot useasta lähteestä. Lähteiden vähyys hankaloitti tutkimuksessa käytetyn tiedon vahvistamista. Monipuolisten lähteiden etsiminen ja laadullinen arviointi olivatkin tämän opinnäytetyön suurimmat haasteet.

Pelkästään käyttöliittymän tyyliopasta käsitteleviä lähteitä on olemassa runsaasti. Anna Debenham (2017, 11 & 14) mainitsee kirjassaan Front-End Style Guides, että tyyliopas-termillä ei lähteestä riippuen tarkoiteta aina samaa asiaa. Sama kävi ilmi tutkimuksen tiedonhakuvaiheessa. Joissakin materiaaleissa tyylioppaalla tarkoitettiin pelkästään komponenttikirjastoa. Toisten lähteiden mukaan tyyliopas sisältää puolestaan komponenttikirjaston lisäksi erilaisia ohjeistuksia sisällön ylläpitoon ja koodin rakenteeseen liittyen. Termin kirjava käyttö aiheutti omat haasteensa tiedonhakuun ja lähteiden validointiin.

Arvioitavien tyylioppaan generointityökalujen löytäminen tutkimukseen oli haasteellista. Styleguides.io-palvelun Tools-osio tuntui sisältävän työkaluja kartoittaessa kaikki yleisessä tiedossa olevat generointityökalut. Se sisälsi niin muissa lähteissä mainitut kuin myös suoraan Google-hakukoneella löytyvät työkalut. Vasta tyylioppaita arvioitaessa ja niihin liittyvää tietoa hakiessa kävi ilmi, että lista ei sisälläkään aivan kaikkia generointityökaluja. Erityisesti epäjohdonmukaisesti nimetyt ja kuvatut työkalut olivat jääneet listauksen ulkopuolelle. Mikään jälkikäteen löydetyistä työkaluista ei olisi suosituimmuuden perusteella vaikuttanut tämän tutkimuksen otokseen.

On kuitenkin mahdollista, että tutkimuksen ulkopuolelle jäi generointityökaluja, jotka olisivat voineet kuulua tutkimuksen otokseen.

Laadullisen tutkimuksen luotettavuuden arvioinnin edellytyksenä on aineistojen ja tulkinnan dokumentaatio (Kananen 2015, 115). Työkalujen arviointiraporttiin pyrittiin dokumentoimaan mahdollisimman tarkasti tutkimuksen tulosten luotettavuuden arviointiin ja tutkimuksen toistamiseen tarvittavat tiedot. Tällaisia ovat tämän tutkimuksen tapauksessa muun muassa työkalujen arvioinnissa käytetty asennusympäristö ja sinne esiasennetut sovellukset, arvioitujen työkalujen versionumerot ja työkaluissa käytetyt asetukset. Dokumentaatio pyrittiin luomaan sellaisella tarkkuudella, että sen avulla tutkimus voitaisiin tarvittaessa toistaa ja tulokset varmentaa.

Tulosten yleistettävyys ja merkittävyys

Jorma Kanasen (2015, 58) mukaan kehittämistutkimuksen tulosten yleistettävyys on heikko, sillä tuloksia voidaan soveltaa vain tutkimustilannetta ja -asetelmaa vastaaviin tilanteisiin. Tässä kehittämistutkimuksessa toimeksiantajan asettamat vaatimukset vaikuttivat tutkimuksessa arvioitujen työkalujen valintaan. Tutkimuksen tulokset ovat näin ollen yleistettävissä projekteihin, joissa on samat lähtökohdat ja vaatimukset kuin tässä tutkimuksessa. Esimerkiksi projektit, jotka eivät käytä Grunt-automatisointityökalua eivät hyödy työkalujen arvioinnista. Toisaalta osana tutkimusta laadittua vaatimuslistaa voidaan hyödyntää monipuolisemmin, sillä se ei ole tekniikka- tai työkaluriippuvainen.

Samoin kuin tulosten yleistettävyyttä myös niiden merkittävyyttä voidaan tässä kehittämistutkimuksessa arvioida lähinnä toimeksiantajan näkökulmasta. Vaikka tutkimuksen tuloksena on, ettei mikään arvioitu työkalu sovellu sellaisenaan toimeksiantajan tyyliopaslähtöistä kehitysmallia käyttäviin projekteihin, tuloksella on silti toimeksiantajan kannalta merkitystä. Projektille sopimaton generointityökalu voidaan myöhemmin vaihtaa toiseen, mutta joissakin tapauksissa yhtä työkalua varten luotua koodia ei voida käyttää lainkaan hyväksi toisessa työkalussa. Työnkalun vaihtaminen tarkoittaa siis kokonaan uuden tyylioppaan luontia. Tämän vuoksi oikean työkalun valinta on tärkeää.

Jatkotutkimukset

Mikään tutkimuksessa arvioitu tyyliopasgeneraattori ei täyttänyt tutkimuksessa tunnistettuja tyyliopaslähtöisyyden vaatimuksia. Jatkotutkimuksia täytyy siis tehdä, jotta tyyliopaslähtöiseen kehitysmalliin sopiva generaattori löytyy. Generaattoreita valittiin tutkimukseen neljä ja tutkimuksen ulkopuolelle jäi vielä useita työkaluja. Jatkotutkimuksena voidaan valita tutkimuksen ulkopuolelle jääneitä työkaluja ja arvioida niitä tässä tutkimuksessa koostettujen vaatimusten perusteella. Jos sopivaa työkalua ei jatkotutkimuksessakaan löydetä, voidaan tässä tutkimuksessa koottua vaatimuslistaa hyödyntää uuden generointityökalun suunnittelussa ja vaatimusmäärittelyssä.

Lähteet

About stars. N.d. Viitattu 26.3.2018. https://help.github.com/articles/about-stars/.

Asiakaskokemus – Solteq. N.d. Viitattu 23.3.2018. https://www.solteq.com/fi/asiakaskokemus/.

Coyier, C. 2015. Where Style Guides Fit Into Process. Viitattu 11.4.2018. https://css-tricks.com/where-style-guides-fit-into-process/.

Debenham, A. 2017. Front-end Style Guides. E-kirja. Maban.co.uk: Anna Debenham.

De La Cuadra, A. 2015. Style Guide Driven Development: A How To Guide To Improve your Development Workflow. Viitattu 15.3.2018. https://www.bitovi.com/blog/style-guide-driven-development.

De La Cuadra, A. 2016. Designing Modular UI Systems Via Style Guide-Driven Development. Viitattu 15.3.2018. https://www.smashingmagazine.com/2016/06/designing-modular-ui-systems-via-style-guide-driven-development/.

De La Cuadra, A. 2018. How To Create a Living Style Guide. Viitattu 11.4.2018. https://www.webdesignerdepot.com/2018/01/how-to-create-a-living-style-guide/.

Digitaalinen markkinointi – Solteq. N.d. Viitattu 23.3.2018. https://www.solteq.com/fi/digitaalinen-markkinointi/.

European Commission visual identity. 2016. Viitattu 13.4.2018. https://ec.europa.eu/info/sites/info/files/charter_en.pdf.

Fordham, M. 2014. Styleguide Driven Development. Viitattu 15.3.2018. https://speakerdeck.com/mattfordham/styleguide-driven-development.

Friedman, V. 2016. Taking The Pattern Library To The Next Level. Viitattu 15.3.2018. https://www.smashingmagazine.com/taking-pattern-libraries-next-level/.

Frost, B. 2014. Style Guides. Viitattu 3.3.2018. http://bradfrost.com/blog/post/style-guides/.

Frost, B. 2016. Anatomy of a Pattern in a Pattern Library. Viitattu 13.4.2018. http://bradfrost.com/blog/post/anatomy-of-a-pattern-in-a-pattern-library/.

Frost, B. 2016. Atomic Design. E-kirja. Viitattu 15.3.2018. http://atomicdesign.bradfrost.com.

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas: miten kirjoitan kehittämistutkimuksen vaihe vaiheelta. Jyväskylä: Jyväskylän ammattikorkeakoulun julkaisuja -sarja.

Kruchten, P. 2008. The Biological Half-Life of Software Engineering Ideas. IEEE Software, 25, 5, 10-11.

Lewis, J. 2015. Simplify your workflow with Styleguide Driven Development. Viitattu 13.4.2018. https://www.slideshare.net/jordanlewiz/jordan-lewis-simplfy-your-

workflow-with-sdd-v1-agile-australia?qid=531b921e-93ae-4e4a-9062-aecab5b38e42&v=&b=&from_search=12.

Liiketoimintaratkaisut – Solteq. N.d. Viitattu 23.3.2018. https://www.solteq.com/fi/liiketoimintaratkaisut/.

LoPresto, C. 2016. Living Style Guide Driven Development. Luento tapahtumassa EmberConf 2016. Viitattu 12.4.2018. https://www.youtube.com/watch?v=Z1lL_Zo62h0.

Nordcloudista ja SC5:stä Euroopan johtava pilvinatiivien IT-palvelujen tarjoaja. 2017. Viitattu 5.4.2018. https://sc5.io/posts/nordcloudista-ja-sc5sta-euroopan-johtava-pilvinatiivien-it-palvelujen-tarjoaja.

Robertson, S. 2014. Creating Style Guides. Viitattu 11.4.2018. http://alistapart.com/article/creating-style-guides.

Robertson, S. 2015. Style Guide Generator Roundup. Viitattu 28.3.2018. http://alistapart.com/blog/post/style-guide-generator-roundup.

Solteq - Rakenna tulevaisuutta ja kehitä itseäsi huippuseurassa. Viitattu 23.3.2018. https://ura.solteq.com/.

Solteq Oyj - Vuosikertomus 2017. 2018. Viitattu 23.3.2018. https://www.solteq.com/wp-content/uploads/2018/02/793244.pdf.

Strategic Design & Innovation Consultancy. N.d. Clearleft-yrityksen kotisivut. Viitattu 14.4.2018. https://clearleft.com/.

Sullivan, N. 2014. Style Guide Driven Development. Luento tapahtumassa CSSConf 2014. Viitattu 25.2.2018. https://www.youtube.com/watch?v=ldW7zVmqu5g.

The world's leading software development platform · GitHub. N.d. Viitattu 26.3.2018. https://github.com/.

What is Angular? N.d. Viitattu 14.4.2018. https://angular.io/docs.

Liitteet

Liite 1. Generointityökalujen suosio GitHub-palvelussa

Generointityökalu	GitHub-tili	Tähtien määrä
KSS	https://github.com/kneath/kss	4048
Devbridge Styleguide	https://github.com/devbridge/Styleguide	1471
SC5 Style Guide Generator	https://github.com/SC5/sc5-styleguide	1221
Fractal	https://github.com/frctl/fractal	1119
Living Styleguide	https://github.com/livingstyleguide/livingstyleguide	849
grunt-styleguide	https://github.com/indieisaconcept/grunt-styleguide/	556
SourceJS	https://github.com/sourcejs/Source	541
PatternPack	https://github.com/patternpack/patternpack	140
DocumentCSS	https://github.com/bitovi/documentcss	63

Liite 2. Tyylioppaan generointityökalujen arviointitulokset

	KSS (grunt-kss)	Devbridge Styleguide	SC5 Style Guide Generator	Fractal
Työkalun asentamiseen ja käyttämiseen liittyvät vaatimukset				
Työkalun asennus-dokumentaatio on kattava ja virheetön	X			X
Työkalu päivittää tyyliopasta auto-maattisesti	X	X	X	X
Tyyliopas päivittyy nopeasti	X		X	X
Komponenttikirjastoon liittyvät vaatimukset				
Komponentille voi-daan määrittää nimi	X	X	X	X
Komponentille voi-daan määrittää ku-vaus	X	X	X	X
Komponentille voi-daan lisätä käyttö-ohjeita ja linkkejä lähteisiin	X	X	X	X
Komponentille voi-daan lisätä tieto sen tekijöistä	X	X	X	X
Komponentille muodostuu oma linkki	X		X	X
Komponentille voi-daan lisätä elävä esimerkki	X	X	X	X
Elävä esimerkki on responsiivinen	X	X	X	X
Elävässä esimer-kissä saa näkyviin tulostustyylit	X		X	X
Elävän esimerkin koodi on näkyvillä ja kopioitavissa	X	X	X	X

Kaikki komponenttiin liittyvät koodit ovat näkyvillä				X
Komponenttien väliset riippuvuudet ovat määritettävissä			X	X
Komponentti voidaan avata omaan ikkunaansa	X		X	X
Komponentilla on oma statustieto				X
Komponentilla on oma versiotieto				
Komponentit saadaan listattua yhdelle sivulle				
Tyylioppaan rakenteeseen liittyvät vaatimukset				
Tyylioppaan ulkoasu on muokattavissa	X		X	X
Tyylioppaan navigaatio on selkeä			X	X
Tyylioppaassa on haku			X	
Ohjesivuihin liittyvät vaatimukset				
Tyylioppaaseen voidaan lisätä sivuja ohjeistuksia varten				X