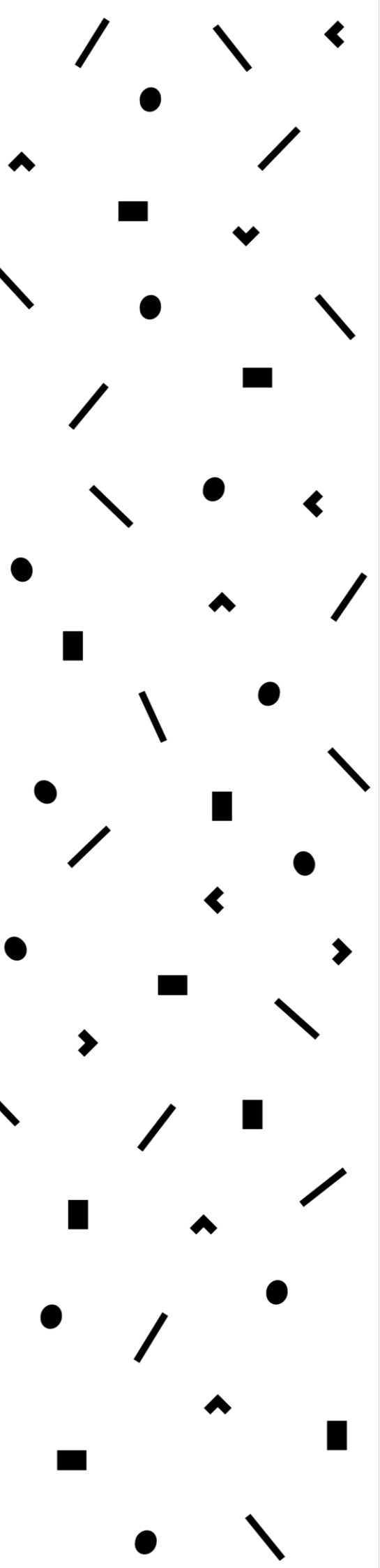


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

ESIKÄSITTELIJÖIDEN HYÖDYNTÄMINEN RESPONSIIVISEN INFORMAATIOPALVELUN KEHITTÄMISESSÄ

LAHDEN AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma
Kevät 2018
Aleksi Kinnunen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Kinnunen, Aleksi	Opinnäytetyö, AMK	Kevät 2018
	Sivumäärä	
	27	

Työn nimi

Esikäsittelijöiden hyödyntäminen responsiivisen informaatiopalvelun kehittämisessä

Tutkinto

Tieto- ja viestintätekniikan koulutusohjelma

Tiivistelmä

Opinnäytetyössä tutkittiin CSS- ja HTML-esikäsittelijöiden sekä niihin yhdistettyjen apukirjastojen tuomia etuja tavalliseen kehittämisprosessiin verrattuna. Esikäsittelijät ja apukirjastot tekevät kehittämisprosessista selkeämmän ja yksinkertaistivat muutoin monimutkaisia vaiheita. Myös eri selainten vaatimat omat erityismäärittelyt saadaan automaattisesti lisättyä tyyleihin apukirjaston avulla.

Kaikki yksittäisen elementin ominaisuudet, sen sisältämät elementit ja jokaisen näiden responsiivisuus on kaikki mahdollista määritellä keskitetysti samassa paikassa CSS-esikäsittelijän avulla. Tällä tavoin koko tyylitiedoston rakenne on helposti ymmärrettävissä ja hallittavissa. Lisäksi sivupohjamoottorin avulla HTML-koodista saadaan modulaarista ja helposti käsiteltävää.

Lopputuloksena esikäsittelijöitä apuna käyttäen kehitettiin responsiivinen informaatiopalvelu: Rakentamisen lupapolku. Rakentamisen lupapolku on visuaalinen esitys, jonka tarkoituksena on helpottaa asiakasta rakennusluvan hakemisessa askel askeleelta ja selkeyttää heille koko rakentamisprojektia.

Asiasanat

CSS, HTML, responsiivinen, esikäsittelijä, tyylitiedosto, sivupohjamoottori, Sass, Pug, Webpack

Abstract

Author(s)	Type of publication	Published
Kinnunen, Aleksi	Bachelor's thesis	Spring 2018
	Number of pages	
	27	

Title of publication

Utilization of preprocessors in development of a responsive information service

Name of Degree

Degree Programme in Information and Communication Technology

Abstract

This thesis work describes the advantages of CSS and HTML preprocessors and connected libraries over the usual standard development process. Preprocessors and additional processing libraries make the development process much clearer, and simplify steps that would otherwise be complicated. Also, browser specific vendor prefixes can be configured to be automatically added with the use of a library.

All styles of a specific element, its child elements, and media queries, can all be centralized using a CSS preprocessor. This way the structure of the stylesheet is easily understandable and manageable. Additionally, the use of a template engine enables HTML code to be modular and easily manageable.

As a result, a responsive information service was developed with the use of preprocessors. The service is a visual representation designed to guide the user about applying for a building permit step by step, and to clarify the whole construction project.

Keywords

CSS, HTML, Responsive, Preprocessor, Stylesheet, Template Engine, Sass, Pug, Webpack

SISÄLLYS

1 JOHDANTO

Responsiivinen web design on toimiva keino toteuttaa verkkopalvelu, joka oikein käytettynä toimii kaikilla laitteilla yhtä vaivattomasti tinkimättä sen sisällöstä, käytettävyydestä tai ulkoasusta. Useat monipuoliset kehykset (frameworkit) eli perusrakenteita tarjoavat koodipohjat sekä esikäsittelijät tekevät responsiivisten palveluiden toteuttamisesta helppoa. Myös kehittämisvaihe yksinkertaistuu ja helpottuu, kun ei tarvita kokonaan erillistä versiota palvelusta mobiililaitteita varten.

Tämän opinnäytetyön tavoitteena on osoittaa käytännön soveltamisen kautta esikäsittelijöiden tuomia etuja käyttäen Sass-esikäsittelijää tyylien määrittelyssä sekä Pug-sivupohjamoottoria sivuston rakenteen luomisessa. Käytännön sovelluksena Sipoon kunnan käyttöön kehitettiin esikäsittelijöiden avulla Rakentamisen lupapolku -informaatiopalvelu.

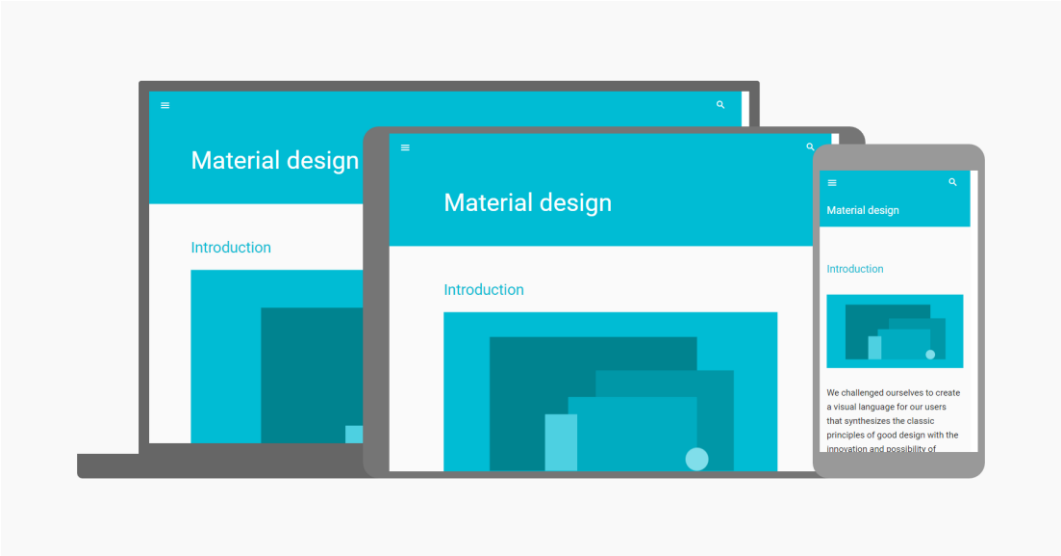
Rakentamisen lupapolku on visuaalinen esitys verkkosivuston muodossa. Palvelun tarkoituksena on helpottaa rakennusluvan hakemista ja selkeyttää koko rakentamisprojektia. Luvanhaussa on haasteellista lupaprosessin hahmottaminen ja siihen liittyvän termistön vaikeus. Rakentamisen lupapolku auttaa luvanhakijaa hahmottamaan vaikeaselkoisen lupaprosessin askel askeleelta. Visualisoinnissa lupaprosessi on avattu nimenomaan asiakkaan näkökulmasta.

2 RESPONSIIVISUUS

2.1 Responsiivinen kehittäminen

Yhdysvaltalainen tai eurooppalainen keskivertokuluttaja omistaa nykypäivänä kaksi ensisijaista laitetta tietojenkäsittelyyn: tietokoneen (yleensä kannettava) ja älypuhelimen (Voutilainen, Salonen & Mikkonen 2015, 1). Ihmiset käyttävät nykyään paljon erilaisia älylaitteita ja verkkopalveluiden täytyy olla yhtä sujuvasti käytettävissä niillä kaikilla. Käyttäjäkokemus ei häiriinny, kun palvelun ulkoasu ja käytettävyys säilyvät laitteiden välillä mahdollisimman yhdenmukaisesti.

Responsiivinen web design (RWD) on menetelmä, jossa palvelin lähettää aina saman HTML-koodin kaikille laitteille ja käyttämällä CSS:ää (Cascading Style Sheets) muutetaan sivun renderöintiä laitteessa (Google Inc 2015). Tämä menetelmätapa helpottaa kehittämistä huomattavasti, koska on olemassa vain yksi versio työstettävästä sivustosta. Mobiililaitteilla ja tableteilla tulisi olla kaikki samat tiedot ja toiminnot kuin työpöytäversiossakin, ja niihin tulisi päästä käsiksi myös yhtä vaivattomasti ja intuitiivisesti.



KUVIO 1. Responsiivinen ruudukko (Google Inc. 2017)

Ohjelmistokehykset yleisesti antavat mahdollisuuden jakaa sisällön ruudukkoon – useimmiten standardiin 12 osaan. Ruudukon osia voidaan varata halutuille elementeille tietty osuus käyttötarkoituksen mukaan. Ruudukon osat ”valuvat” automaattisesti uusille riveille, kun ne eivät enää mahdu luontevasti vierekkäin. Näin sisällöstä saadaan helppolukuista ja selkeää. Kaikki tämä riippuu määritellystä ruudukon jaosta, käytössä olevasta laitteesta ja

sen näytön koosta. Lukuisat kehykset (framework) tarjoavat valmiit pohjat ja tarkat ohjeet tällaisten ruudukoiden käyttämiseen. (KUVIO 1.)

Nykyään uusilla apukirjastoilla ruudukoita voi itse helposti määritellä haluamallaan tavalla pelkästään tyylitiedostoista käsin tietyille elementeille. Näin vältytään lukuisilta HTML-koodin ylimääräisiltä luokkamäärittelyiltä. Tällä tavoin jälkikäteen muokattavuus helpottuu ja tyylimäärittelyt ovat selkeämmin hallittavissa.

Responsiivisten sivustojen suunnittelu ja kehittäminen pitää ottaa yhdessä huomioon, jotta voidaan taata toimiva käyttökokemus kaikilla päätelaitteilla. RWD on usein kuin palapelin ratkaisemista – kuinka suuret sivuston elementit saisi mahtumaan kapeampaan tilaan tai päinvastoin. Ei kuitenkaan riitä pelkästään, että taataan elementtien mahtuminen sivulle. Responsiivisen designin tulee myös olla käyttökelpoista kaikilla erikokoisilla resoluutioilla ja näytöillä. (Schade 2014.)

Pienempikokoisilla kosketuslaitteilla käyttökelpoisuutta voidaan parantaa esimerkiksi tekemällä painikkeista ja valintaruuduista tarpeeksi isoja, jotta käyttäjä kykenee painamaan niitä tarkasti tekemättä virhepainalluksia. Myös intuitiiviset rakenteet ja toiminnot parantavat käyttökokemusta, kun ei ole epäselvyyttä siitä, kuinka elementit käyttäytyvät.

2.2 Nykyaikaiset käytännöt

Sisällön priorisointi on yksi avainasia hyvässä responsiivisessa designissa. Suurella pöytäkoneen tai kannettavan tietokoneen näytöllä on paljon enemmän sisältöä näkyvissä kerralla kuin pienellä älypuhelimella. Suuren näytön käyttäjä pystyy helposti yhdellä katsauksella löytämään etsimänsä, kun taas älypuhelimella saattaa joutua vierittämään ruutua paljon löytääkseen mielenkiinnon kohteen. Älykäs sisällön priorisointi auttaa käyttäjiä löytämään tehokkaammin tarvitsemansa. (Schade 2014.) Pienemmillä älypuhelimien ja tablettien näytöillä voidaan esimerkiksi valikko toteuttaa responsiivisesti viemättä kallisarvoista tilaa käyttämällä hyväksi kosketusnäyttöä. Käyttäjäystävällinen sivuvalikko voidaan mobiililaitteella tuoda esiin käyttäjän vetäessä sen esiin näytön reunasta. Indikaattorina tällaisesta valikosta on usein sivuvalikkokuvake, joka yleisesti toimii samalla myös painikkeena, joka avaa sekä sulkee valikon. Valikon toimintoihin pääse helposti käsiksi viemättä tilaa muulta tärkeältä sisällöltä sivun yläosasta.

Responsiivisten mobiilikäyttöliittymien designilla tulisi olla korkea prioriteetti ja tämä tulisi tiedostaa aikaisessa vaiheessa verkko-ohjelmien kehityksessä. Lähtökohtaisesti mobiililaitteille tehdyt mobile first -käyttöliittymät tukevat monien alustojen rajapintoja, sillä suurempikokoiset käyttöliittymät ovat helpompia toteuttaa jälkeenpäin verrattuna pienempiin. (Voutilainen ym. 2015, 4.) Responsiivisia sivustoja suunnitellessa tulisi pitää koko ajan

mielessä mobiilinäkymä ja käyttökokemus. Niiden pohjalta tulisi myös tehdä valinnat sivun elementeistä, joita päättää käyttää. Onko hyödyllistä toteuttaa jokin valikko siten, että se avautuu siirtämällä kursori sen päällä, kun mobiililaitteilla tällainen toiminto ei ole varsinaisesti mahdollista? Kun alusta asti suunnittelee kaikille päätelaitteille sopivia elementtejä, on myös vaivattomampaa tehdä muutoksia tulevaisuudessa.

Mitä useammin käyttäjien odotukset osoittautuvat oikeiksi, sitä enemmän he tuntevat olevansa hallinnassa järjestelmästä, ja sitä enemmän he pitävät siitä (Nielsen 2011). Tämä pätee oikein toteutetussa responsiivisessa web designissa, jolloin käyttäjä voi olettaa samaa kokemusta kaikilla laitteilla. Huonosti suunnitellut erilliset mobiilisivustot eivät välttämättä tarjoa edes samaa sisältöä tai palveluita kuin pääsivusto. Tämä on huono käytäntö, jolta vältytään käyttämällä RWD:tä.

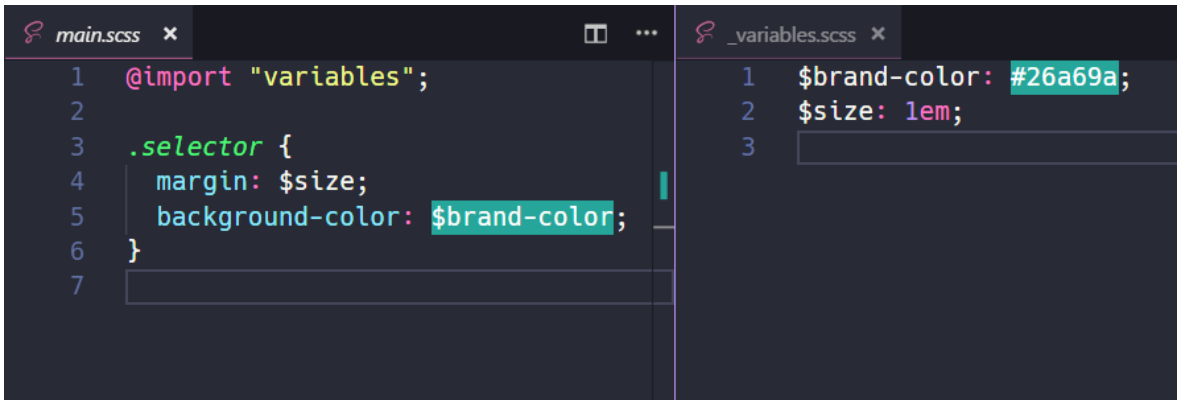
3 CSS-ESIKÄSITTELY

3.1 CSS-esikäsittelijöiden yhteisiä piirteitä

CSS-esikäsittelijät tuovat lukuisia uusia ominaisuuksia ja tekemistä nopeuttavia käytäntöjä CSS-syntaksin lisäksi kehittämisvaiheeseen. CSS-esikäsittelijän oma tyylitiedosto muunnetaan selaimia varten CSS-tyylitiedostoksi. Yksi suurimmista CSS-esikäsittelijän eduista on selkeämpi tiedosto- ja tyylirakenne, joiden avulla kehittäminen ja ylläpitäminen helpottuvat. Mobile first -suunnitteluperiaate on helpompi toteuttaa CSS-esikäsittelijän avulla; yksittäisen elementin ominaisuudet, sen sisältämät elementit ja jokaisen näiden responsiivisuus on kaikki mahdollista määritellä samassa paikassa. Tällä tavoin myös koko tyylitiedoston rakenne on helposti ymmärrettävissä ja hallittavissa.

Kaikille CSS-esikäsittelijöille yhteisiä pääominaisuuksia ovat muuttujat, uudelleen käytettävät mixin-yhdistelmätyylit sekä esimerkiksi tyylimäärittelyyn sisennettävä valitsin (&), jolla voi viitata itseensä (ks. kuvio 5).

Kaikissa CSS-esikäsittelijöissä tyylit ja määrittelyt voidaan myös jakaa useisiin eri tiedostoihin, jolloin ne ovat paremmin hallittavissa. Tällä tavoin projekti voidaan organisoida modulaarisesti, jolloin esimerkiksi kaikki muuttujat ovat yhdessä tyylitiedostossa. Useimmiten on tapana, että yksi pää-tyylitiedosto kokoaa kaikki eri tiedostot yhteen kääntämistä varten. Lopputuloksena on yksi ainut CSS-tyylitiedosto. Tämän hetken tunnetuimpia CSS-esikäsittelijöitä ovat Sass (http://sass-lang.com/), Less (http://lesscss.org/) sekä Stylus (http://stylus-lang.com/). (KUVIO 2.)



KUVIO 2. Tyylitiedostojen tuonti Sass-esikäsittelijällä

3.2 Sass-esikäsittelijä

Sass-esikäsittelijä on ollut kehityksessä jo vuodesta 2006 lähtien. Projektin takana on aktiivinen kehittäjätiimi, joka pitää yllä eri versioita sekä kehittää myös koko ajan uutta. Sass-esikäsittelijän ensimmäinen versio on kirjoitettu Ruby-ohjelmointikielellä ja nykyinen nopeampi versio LibSass versio on tehty C- ja C++-ohjelmointikielillä (Sass 2017). Uusimpana pyrkimyksenä on Sass:n uudelleenkirjoittaminen uudelle ohjelmointikielelle käsittelynopeuden parantamiseksi entisestään.

Sass-esikäsittelijälle on saatavilla kaksi eri syntaksia: SCSS (Sassy CSS), joka on laajennus CSS:n syntaksille, sekä SASS (Syntactically Awesome StyleSheets), joka on vähemmän käytetty vanhempi versio Sass:n syntaksista. SCSS-syntaksin tiedostopääte on .scss ja SASS-syntaksilla .sass. SASS-syntaksissa ei käytetä aaltosulkuja, vaan kaikki sisäkkäin asettelu määrittyy sisennyksien mukaan. Myös kaikki ominaisuudet erotellaan toisistaan ainoastaan rivinvaihdoilla ilman puolipisteitä. (Sass 2016.)

Näistä kahdesta syntaksista ainoastaan SCSS on yhteensopiva standardin CSS-syntaksin kanssa (Sass 2016). Molemmat versiot ovat kuitenkin edellä mainittuja eroja syntaksissa lukuun ottamatta täysin samat. SCSS-syntaksista on tullut yleisesti käytetympi kuin SASS-syntaksista.

Sass käyttää muuttujien merkitsemiseen $-merkkiä. Yksi Sass-esikäsittelijän hyödyllisimmistä ominaisuuksista on `@extend`. Sen avulla tiettyä koodia voidaan käyttää uudelleen useissa eri määrittelyissä vähentäen turhaa toistoa tyylitiedostossa. Yksinkertaisin tapa hyödyntää extend-ominaisuutta on luoda %-merkkiä käyttäen väliaikainen valitsin, jonka sisään kirjoitetaan halutut määrittelyt. Tällä tavoin tyylimäärittelyiden ylläpito helpottuu, kun useille elementeille yhteiset määrittelyt löytyvät samasta paikasta. Väliaikaiset valitsimet eivät itsessään tee mitään vaan niitä täytyy käyttää extendin kanssa. Kuviossa 3 näkyvän esimerkin `%step-pseudo` nimisen väliaikaisen valitsimen sisältämät määrittelyt siirtyvät kääntäessä joka paikkaan jossa siihen on viitattu `@extend` määrittelyn avulla.

```
136  // ------------------------------------------------
137  // Flowchart steps
138  // ------------------------------------------------
139
140  // Base for step pseudo lines
141  %step-pseudo {
142    content: '';
143    position: absolute;
144    background: $color-step-line;
145    pointer-events: none;
146  }
147
192
193    // Default step connecting pseudo line
194    &::after {
195      @extend %step-pseudo;
196      width: $step-line-weight;
197      height: $step-spacing;
198      left: calc(50% - #{$step-line-weight} / 2);
199      top: $step-height;
200    }
```

KUVIO 3. Esimerkki extend-ominaisuuden käytöstä Sass-esikäsittelijällä

Erillisten tyylitiedostojen tuonti Sass-esikäsittelijällä tapahtuu käyttäen `@import` -määrittelyä kuvion 2 mukaisesti. Samassa kuviossa on myös esimerkki muuttujien käytöstä.

```
// Variables
$background-color: #c5cae9;
$text-color: #fff;
$primary-color: #3f51b5;
$secondary-color: #ec407a;

.container {
  background: $background-color;
  color: $text-color;
  font-family: 'Roboto', sans-serif;
  font-size: 1em;
  lost-center: 1140px 30px flex;
}

.block {
  lost-column: 1;
  lost-flex-container: row;

  &__element {
    background: $primary-color;
    lost-column: 1;
    padding: 60px 20px;

    &--modifier {
      background: $secondary-color;
    }

    @include at-medium {
      lost-column: 1/2;
    }
  }

  @include at-small {
    lost-column: 1/2;
  }
}
```

KUVIO 4. Esimerkki tyylimäärittelyistä Sass-esikäsittelijällä

3.3 Less-esikäsittelijä

Less-esikäsittelijä on toiseksi pisimpään kehityksessä ollut CSS-esikäsittelijä, jo vuodesta 2009 lähtien. LESS (Leaner Style Sheets) -syntaksi on saanut vaikutteita Sass-esikäsittelijältä ja on hyvin saman tyyppinen. Joitakin eroja merkkaustavoissa lukuun ottamatta Less:n syntaksi on Sass:n SCSS-syntaksin kanssa lähes samanlainen ja vaati muuan muassa kaikki tavanomaiset aaltosulut, kaksoispisteet sekä puolipisteet. LESS-tyylitiedostojen pääte on .less.

Less-esikäsittelijä käyttää muuttujia varten @-merkkiä, josta esimerkkinä on kuvio 5. Lisäksi elementin sisäisiä määrittelyitä pystyy käyttämään muuttujina käyttäen $-merkkiä, mistä esimerkkinä on kuvio 6. Tämä on uusi ominaisuus Less:n versiossa 3.0.0.

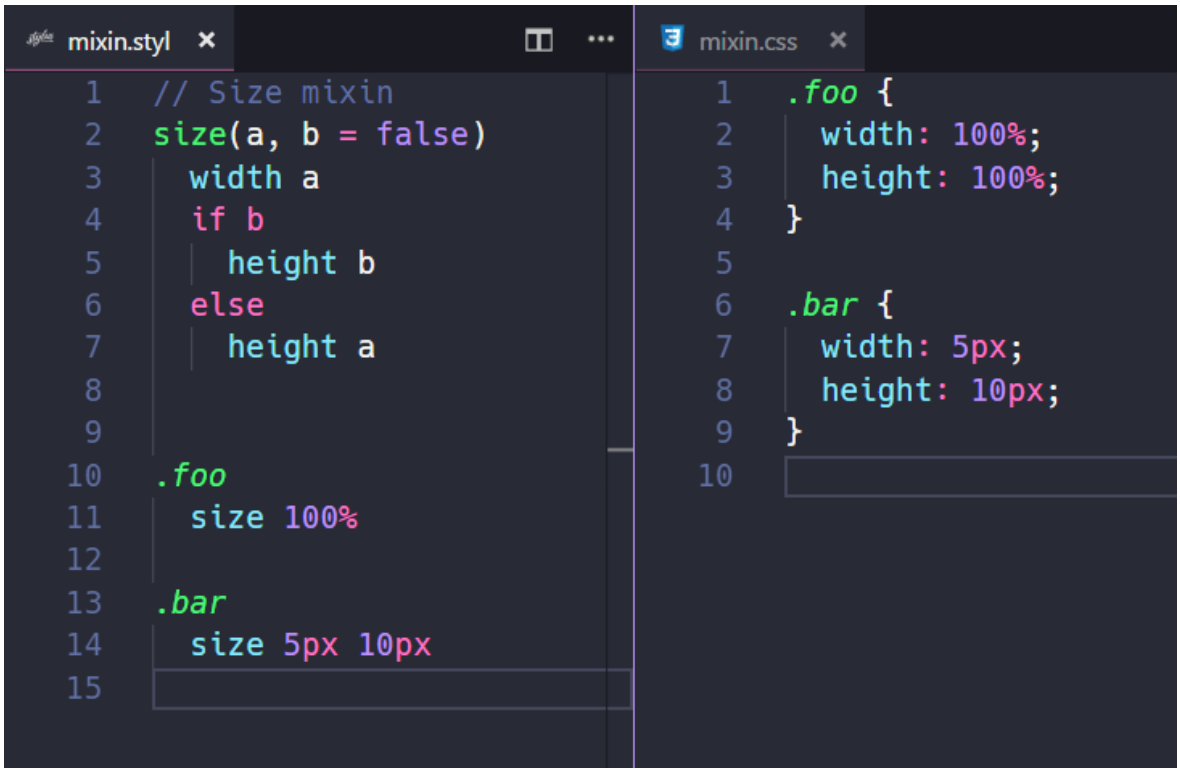
KUVIO 5. Esimerkki muuttujien käytöstä Less-esikäsittelijällä

KUVIO 6. Esimerkki määrittelyiden käytöstä muuttujina Less-esikäsittelijällä

3.4 Stylus-esikäsittelijä

Stylus on yksi uusimmista CSS-esikäsittelijöistä. Sen kehitys on aloitettu vuonna 2010. Stylus on saanut vaikutteita Sass- ja Less-esikäsittelijöiltä. Kirjoitushetkellä Styluksen kehityksen jatkuminen on epävarmaa, sillä aktiivisista tekijöistä on pulaa.

Stylus-esikäsittelijän syntaksi on sisennykseen ja väleihin perustuva ja siitä syystä erittäin vapaamuotoinen (KUVIO 7). Koodia voi esimerkiksi kirjoittaa kuin se olisi CSS:ää. Vaihtoehtoisesti kaikki aaltosulkeet, puolipisteet ja kaksoispisteet voi jättää pois edellyttäen kuitenkin, että sisennykset sekä tyhjät välit ovat kunnossa. Jopa kaikkien edellä mainittujen merkkien käyttäminen lähes sekalaisesti ympäri tyylitiedostoa lähes miten tahansa luetaan oikeaksi. Stylus-tyylitiedoston pääte on .styl ja kaikki mahdolliset variaatiot syntaksissa käyttävät samaa tiedostopäätettä. (Stylus 2017.)



KUVIO 7. Mixin-yhdistelmätyylin käyttö Stylus-esikäsittelijällä

3.5 Apukirjastot

CSS-esikäsittelijöille löytyy useita hyödyllisiä apukirjastoja, jotka tuovat uusia toiminnallisuuksia ja uudelleenkäytettäviä osia. Responsiivisen ruudukon luomiseen on olemassa monia apukirjastoja, joilta löytyy tuki useillekin eri esikäsittelijöille. Myös muun responsiivisuuden toteuttamiseen löytyy omat liitännäiset.

On olemassa myös apukirjastoja, jotka käsittelevät CSS-syntaksia lisäämällä siihen esimerkiksi kaikkien eri selainten tukemiseen tarvittavat lisämäärittelyt jälkikäteen, mikä nopeuttaa kehittämistä. (KUVIO 8.) Lisäksi on mahdollista muuntaa kokeellinen syntaksi tai apukirjastojen määrittelyt normaaliksi CSS:ksi. Tällaiset apukirjastot toimivat myös minkä tahansa esikäsittelijän kanssa, koska ne muokkaavat vain valmista CSS:ää.

```
.block {
  -webkit-box-flex: 0;
  -ms-flex-positive: 0;
  flex-grow: 0;
  -ms-flex-negative: 0;
  flex-shrink: 0;
  -ms-flex-preferred-size: calc(100% * 1 - (30px - 30px * 1));
  flex-basis: calc(100% * 1 - (30px - 30px * 1));
  width: calc(100% * 1 - (30px - 30px * 1));
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: horizontal;
  -webkit-box-direction: normal;
  -ms-flex-flow: row wrap;
  flex-flow: row wrap;
}
```

KUVIO 8. CSS-apukirjaston avulla luodun määrittelyn ulossyöte

4 HTML-ESIKÄSITTELY

4.1 Sivupohjamoottorit

Sivupohjamoottori (template engine) on HTML-sivun muodostamiseen erilaisia lisäominaisuuksia tuova esikäsittelijä. Sivupohjilla HTML-sivun eri osioita voidaan jakaa useisiin eri tiedostoihin sekä käyttää niitä täysin modulaarisesti. Tämä on yleensä sivupohjamoottorien pääasiallinen käyttötarkoitus. Sivupohjilla on tyypillisesti oma tiedostopäätteensä. Muuttujaan talletetun sisällön upottaminen määriteltyyn sivupohjan kohtaan on myös yksi sivupohjamoottorien pääominaisuuksista. Muita yleisiä ominaisuuksia ovat muun muassa muuttujat, funktiot ja ehdollisuudet.

Sivupohjamoottorien käyttämät sivupohjat saattavat näyttää tavalliselta HTML-tiedostolta eivätkä käytä aina täysin omaa syntaksia; niiden oma syntaksi voi olla upotettuna standardiin HTML-syntaksiin. Esimerkiksi suosittu Handlebars.js on yksi tällainen.

Sivupohjamoottorit nopeuttavat kehittämistyötä ja tekevät tiedostojen hallinnasta helpompaa CSS-esikäsittelijöiden tapaan. Sivupohjia käyttäen varsinkin sivuston kehittämisvaihe nopeutuu huomattavasti, kun erilaisia rakenteita pystyy muodostamaan paljon kompaktimmalla koodilla. Esimerkiksi prototyyppien luontia varten on olemassa funktioita, joilla voi esimerkiksi toistaa elementtejä sivulle vaivattomasti. Niin kuin CSS-esikäsittelijöilläkin, määrittelyitä voidaan sivupohjien avulla jakaa modulaarisesti useihin eri tiedostoihin. Tällä tavoin käytössä voi olla useita sivupohjia, joihin on linkitettynä esimerkiksi jokin kaikilla sivuilla yhteisesti esiintyvä elementti. Tämä elementti voidaan pitää omassa sivupohjassaan, jolloin kaikki muutokset saadaan tehtyä muokkaamalla ainoastaan yhtä sivupohjaa.

Sivupohjamoottoreilla on esikäsittelijöiden tapaan mahdollista olla oma syntaksinsa, joka kääntyy lopulta HTML-syntaksiksi. Yleensä sivupohjamoottorien syntaksi perustuu sisennyksiin, jolloin ei myöskään tarvita lopetustageja tai muita HTML-standardin merkkejä. Tämä ominaisuus tuo suuren edun varsinkin suurikokoisten HTML-tiedostojen käsittelyssä, jolloin kyetään esittämään suurempi määrä sisältöä samanaikaisesti.

4.2 Pug-sivupohjamoottori

Pug, entiseltä nimeltään Jade, on suosittu aktiivisessa kehityksessä oleva sivupohjamoottori. Pug toimii JavaScriptillä Node-suoritusympäristössä ja käyttää täysin omaa syntaksiaan. Pugin syntaksi perustuu välilyöntien ja sisennyksien käyttöön. Pug käyttää sivupohjissaan omaa tiedostopäätettä .pug.

```
15  body
16    div.container
17      aside.flowchart-container
18        div.header-container
19          header.header
20            div.header__select
21              a(title='suomeksi').select.select--active fi
22              a(href='/sv' title='på svenska').select sv
23              //- a(href='#' title='in english').select en
24            div(title='ohje').header__title
25              h2 Rakentamisen lupapolku
26              p Sipoon kunnan ohjeistus rakennusluvan hakemiseen
27        div.illustration: img(src='../images/hero.svg' width='100%')
28        div.flowchart
```

KUVIO 9. Esimerkki Pug-syntaksista

Pug-tiedoston rakenne muodostuu pelkästään elementtien nimistä ilman HTML:lle tyypillisiä kulmasulkeita. Näiden perään kirjoitetaan suoraan muut tarvittavat määrittelyt peräkkäin ilman välejä. Elementin sisältö asetetaan välimerkin jälkeen samalle riville kaiken muun määrittelyn jälkeen. Kääntämisvaiheessa Pug poistaa Pug-tiedostosta kaikki elementtien väliset tyhjämerkit. Erityisesti tästä syystä Pugin käyttö vaatii aluksi totuttelua. Tekstisisältöä pystyy kuitenkin tarvittaessa jakamaan useammalle riville pystyviivamerkkien avulla kuvion 10 mukaisesti.

piped_text.pug

```
1  p
2    | The pipe always goes at the beginning of its own line,
3    | not counting indentation.
4
```

piped_text.html

```
1  <p>The pipe always goes at the beginning of its own line, not
     counting indentation.</p>
2
```

KUVIO 10. Esimerkki tekstin jakamisesta useammalle riville Pug-sivupohjamoottorilla

Pugin avulla HTML-koodia voi jakaa modulaarisesti useisiin eri tiedostoihin kuten sivupohjamoottoreilla tyypillisesti on mahdollista. Lisäksi Pugin mixin-funktioilla pystyy luomaan

uudelleen käytettäviä koodin palasia, joihin voi syöttää lisäksi myös muuttujia. Eräs kätevä Pugin ominaisuus on sisäkkäisten elementtien asettelu samalle riville tilan säästämiseksi erottamalla ne kaksoispisteellä. Edellä mainittujen käytäntöjen lisäksi Pugilla on lukuisia muita hyödyllisiä ominaisuuksia kuten esimerkiksi

- JavaScript-koodin upottaminen suoraan HTML:ään
- ehtolauseet
- iteraatio
- ja interpolointi eli datan siirtäminen valittuihin kohtiin koodissa.

Kaikki ominaisuudet ja käyttöohjeet löytyvät Pugin omasta dokumentaatiosta.

4.3 Muita sivupohjamoottoreita

Slim on Ruby-suoritusympäristössä toimiva sivupohjamoottori. Slim on saanut kehityksensä aikana vaikutteita sekä HAML- että Pug-sivupohjamoottoreilta. Slimin syntaksi muistuttaa pitkälti Pugin syntaksia minimalistisuudellaan.

HAML-sivupohjamoottori käyttää myös Rubya. Pug on saanut paljon vaikutteita HAML:ista ja molemmista sivupohjamoottoreista löytyy useita yhtäläisyyksiä. HAML käyttää %-merkkiä HTML-vakioelementtien edessä. Lopetustageja ei muiden sivupohjamoottorien tapaan tarvita.

5 NIMEÄMISKÄYTÄNNÖT

5.1 Edut ja hyödyntäminen

Nimeämiskäytännöt ovat tapoja nimetä asioita, joilla koodista saadaan täsmällisempää, avoimempaa sekä informatiivisempaa (Roberts 2017.) Nimeämiskäytäntöjä käyttämällä vältytään mahdollisilta sekaannuksilta, kun kaikki elementtien määrittelyt on nimetty hierarkkiseen tapaan. Samalla myös elementtien väliset suhteet voidaan tunnistaa jo ensisilmäyksellä.

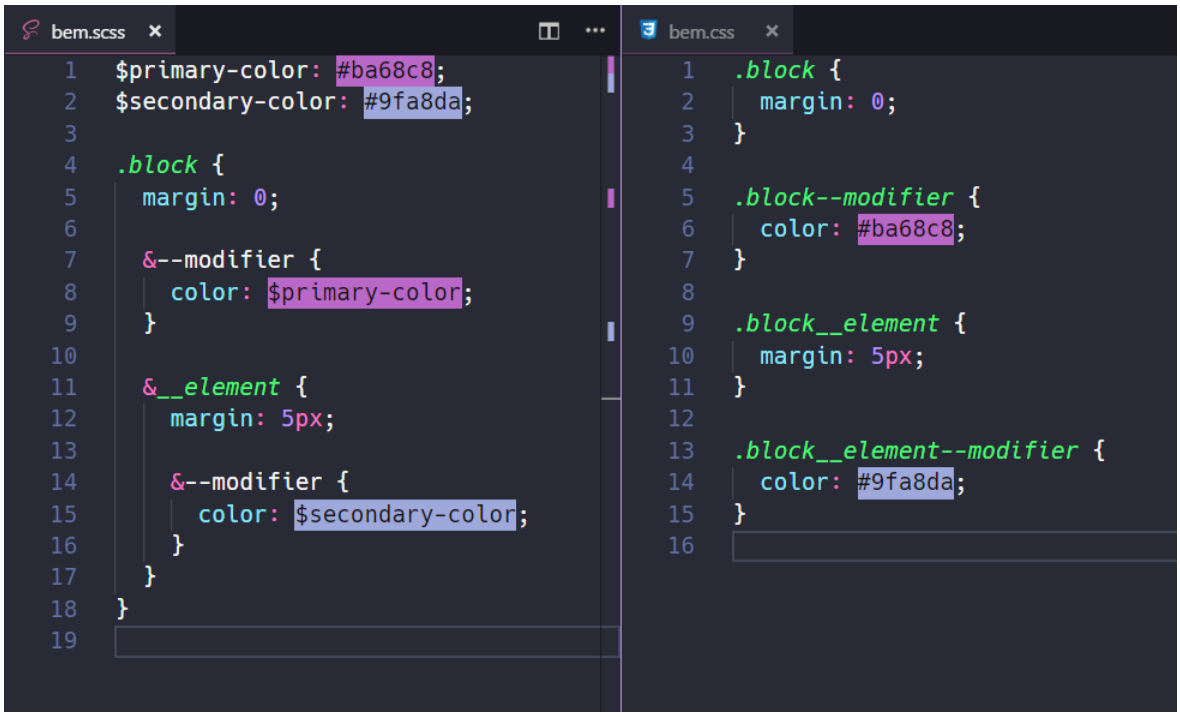
Nimeämiskäytännöistä on hyötyä erityisesti esikäsittelijöiden kanssa, koska yhtenäisten elementtien tyylit voidaan määritellä keskitetysti. Tällä tavoin puolestaan vältetään turhaa toistoa koodissa ja myös responsiivisuusmäärittelyt saadaan aseteltua samojen elementtien yhteyteen.

5.2 BEM-nimeämiskäytäntö

Yksi suosituimmista nimeämiskäytännöistä on BEM (KUVIO 11). Yleisesti ottaen BEM soveltuu omavaraisiin, irrallisiin käyttöliittymän osiin (Roberts 2017.) BEM:ssä kaikki valitut elementit erotellaan seuraaviin kategorioihin:

- pääelementteihin (block)
- niiden lapsielementteihin (element)
- ja määritteisiin (modifier).

BEM-nimeämiskäytännössä nimitykset ketjutetaan peräkkäin järjestyksessä kuten kuvion 11 esimerkin tapaan. Yhteen nimitykseen saa kuitenkin maksimissaan sisältyä kaikki kolme kategoriaa vain kertaalleen. Tästä syystä rakenteiden nimeäminen tulee suunnitella järjestelmällisesti varsinkin HTML-koodissa.



KUVIO 11. Malli BEM-tyyppisestä nimeämiskäytännöstä Sass-esikäsittelijän avulla

Muita eri nimeämiskäytäntöjä ovat muun muassa:

- OOCSS (Object-Oriented CSS)
- AMCSS (Attribute Modules for CSS)
- SMACSS (Scalable and Modular Architecture for CSS)
- sekä SUIT CSS.

Näistä ainoastaan BEM sopii käytettäväksi sekä CSS:n että JavaScriptin kanssa. (BEM 2017.)

6 CASE: RAKENTAMISEN LUPAPOLKU

6.1 Tavoite

> *Rakennusvalvonnan päätehtävä on valvoa kunnan/kaupungin alueella rakentamista sekä rakennettua ympäristöä. Rakennusvalvonnan työn keskeinen sisältö on erilaisten lupien myöntäminen sekä luvitettujen hankkeiden rakentamisaikainen valvonta. Haasteeksi asiakkaille luvanhaussa on osoittautunut lupaprosessin ymmärtäminen ja siihen liittyvän termistön vaikeus. (Upola 2017.)*

Tarvetta vastaamaan lähdettiin kehittämään Sipoon kunnan käyttöön julkaistavaa Rakentamisen lupapolku -informaatiopalvelua. Tavoitteena oli sekä helpottaa rakennusvalvonnan asiakkaille rakennusluvan hakemista että selkeyttää heille koko rakentamisprojektia. Sipoon kunnassa kehitetään voimakkaasti palvelujen digitalisointia kaikkien palvelualueiden osalta: tällaiselle älykkäälle ohjeistukselle on Sipoon kunnassa ollut merkittävä tarve ja tilaus.

Rakentamisen lupapolku auttaa luvanhakijaa hahmottamaan lupaprosessin askel askeleelta.

6.2 Sisällön suunnittelu

Visualisointi toteutettiin yhteistyössä rakennusvalvonnan henkilöstön ja teknisen osaston yhdyskuntamuotoilijan kanssa. Päävetäjänä hankkeelle toimi yhdyskuntamuotoilija Riikka Virta sekä rakennusvalvontapäällikkö Ulla-Maija Upola; ohjausryhmänä toimi koko rakennusvalvonnan henkilöstö.

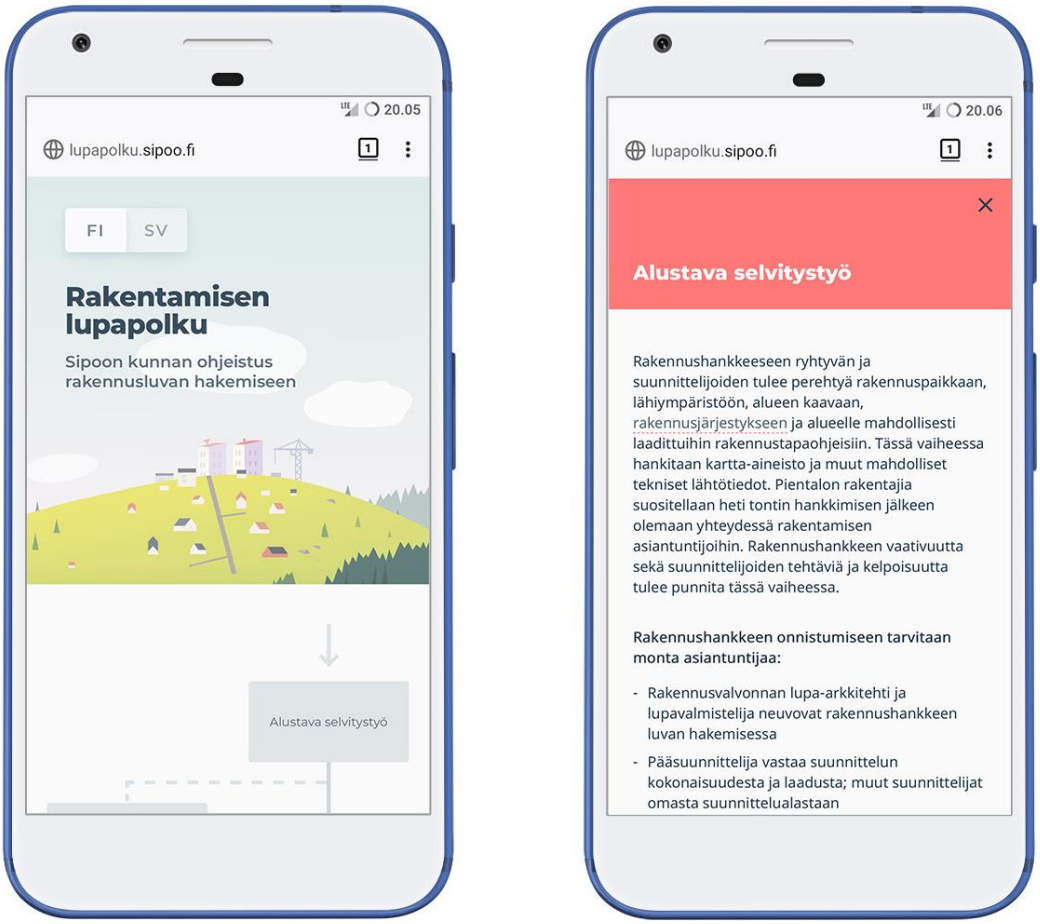
Aluksi ohjausryhmä keskusteli ja avasi lupaprosessia ja sen eri vaiheita. Keskustelun lopputuloksena syntyi monivaiheinen prosessikaavio rakennusluvan hakemisesta (LIITE 1). Vaiheittain läpikäytävä ja visuaalisesti esitetty lupaprosessi sekä selkeyttäisi luvan hakemista että mahdollistaisi kuitenkin vaiheiden käsittelyn yksityiskohtaisesti. Tästä kehittyi ajatus käyttäjäystävälliseen informaatiopalveluun verkossa. Sisällöllisen kehittämistyön yhteydessä haastateltiin myös rakennusvalvonnan tärkeimpiä sidosryhmiä ja heidän näkökulmiaan on myös otettu huomioon lopputuloksessa. Koko lupaprosessin osiin purkaminen koettiin tärkeänä myös siltä osin, että se tulee selkeyttämään viranomaisten ja luvanhaussa mukana olevien eri toimijoiden keskinäisiä toimintatapoja (Virta 2017).

Lupapolun tekstisisältö on yhdyskuntamuotoilijan toimesta kirjoitettu asiakkaalle ymmärrettävään muotoon, eli avattu viranomaisille tyypillistä ja vaikeaa lakiin tai muuhun vastaavaan perustuvaa termistöä. Oman lisänsä kehittämiselle muodostaa Sipoon kunnan kaksikielisyys eli palvelu tulee tarjota sekä suomen että ruotsin kielellä.

Tarkoituksena on selkeästi tuoda esiin luvanhaun eteneminen: luvanhakuun liittyvät keskeiset vaiheet tulisi kuvata polkumaisena etenemisenä sisältäen kaikki rakennusluvan hakemiseen tarvittavat tiedot. Tarvittaessa tarkempaa tietoa kustakin luvanhaun tai rakentamisen vaiheesta lukijan tulisi löytää kaavioon upotetuista otsakkeista ja linkeistä. Näin tiedon jakaminen tapahtuisi vähitellen sekä tiedon sisäistäminen helpottuisi. Rakentamisen lupapolun tulisi tarjota alusta ja työkaluja luvanhakuun ja rakentamisvaiheeseen ja siten tukea luvanhakijoiden itseohjautuvuutta.

6.3 Toteutus

Tavoitteena oli kehittää responsiivinen ja mahdollisimman kevyt staattinen sivusto rakennuslupaprosessia selventävää informaatiopalvelua varten. Palvelu toteutettiin ilman sisällönhallintajärjestelmää, sillä käyttökohde ei sen hetkisessä muodossaan vaatinut sitä vähäisen sisällön vuoksi. Responsiivisuus toteutettiin mobile first -suunnitteluperiaatteen mukaisesti käyttäen hyödyksi Sass CSS-esikäsittelijää sekä Pug-sivupohjamoottoria. Käyttöliittymä mukautuu mobiililaitteille säilyttäen kaiken saman sisällön kuin työpöytänäkymässäkin, kuvion 12 mukaisesti. Myös suurempiresoluutioiset ja tavallista leveämmän kuvasuhteen näyttökoot on otettu huomioon.



KUVIO 12. Rakentamisen lupapolku mobiililaitteella

Käytettävyyden kannalta sivustolla esiintyvä lupaprosessiaikajana näkyy aina samanaikaisesti valitun vaiheen ja sen sisältämän informaation kanssa kuten kuviossa 13. Mobiililaitteilla lupaprosessin aikajana varaa koko ruudun leveyden ja halutun vaiheen sisältö avautuu päällimmäiseksi, kuten kuviossa 12. Kaikki sivuston reititys tapahtuu yhdellä HTML-sivulla kielenvaihtoa lukuun ottamatta. Reititys tapahtuu yksinkertaisella JavaScript-koodilla, jolla piilotetaan tai näytetään valitun vaiheen sisältämä elementti. Tilankäytön optimoimiseksi lisäinformaatiota varten käytettiin hyödyksi pystysuunnassa laajenevia välilehtiä, jotta palvelun käyttäjä voisi helposti valita haluamansa tiedot näkyviin.

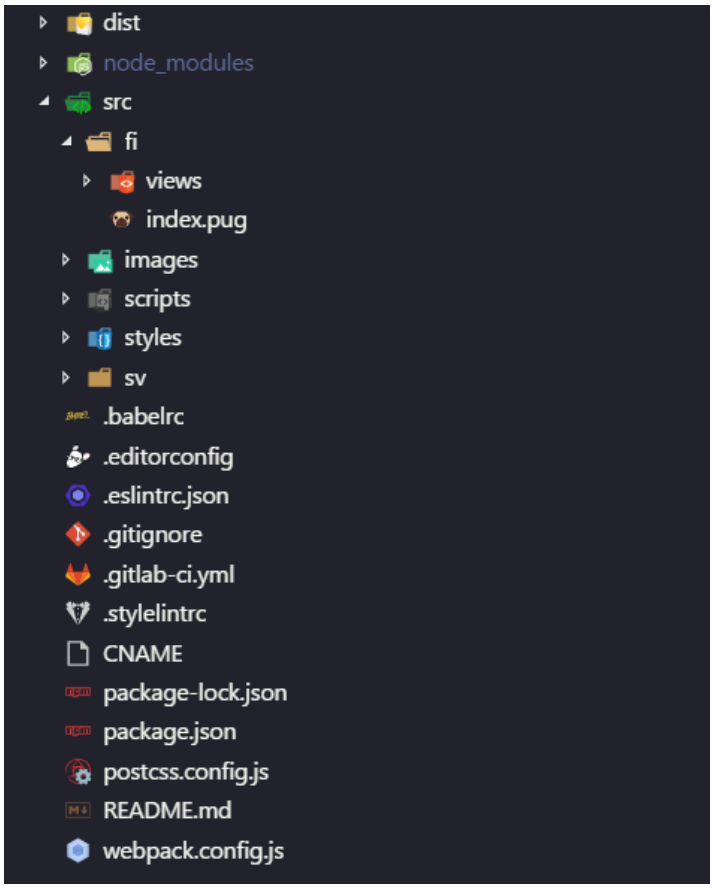
Sivustolla esiintyvän lupaprosessiaikajanan aktiivinen vaihe värjäytyy tunnusvärillä korostamaan sitä valituksi. Lisäksi jokainen aktiivista vaihetta edeltävä vaihe on haalennettu merkitsemään niiden suoritusta, kuten kuviosta 13 ilmenee. Kaikki vaiheet yhdistävät viivat ja katkoviivat ovat kokonaisuudessaan rakennettu CSS:n pseudoelementeillä; näin

HTML-tiedostot pysyvät selkeinä eivätkä tyylimäärittelyt ole sidonnaisia tiettyihin elementteihin. Esimerkkinä tästä on kuviossa 3 näkyvä pseudoelementin oletusmäärittely aikajanan yksittäiselle vaiheelle.

KUVIO 13. Yksi lupaprosessin vaiheista valittuna suurella ruutukoolla

6.3.1 Kansiorakenne

Kansio- ja tiedostorakenne toteutettiin yleiseen tapaan, jossa asetustiedostot sekä alakansiot kehittämistä ja tuotantoa varten sijaitsivat juurikansiossa. Kehittämisvaiheessa kaikki tapahtui lähdehakemistossa (src). Lähdehakemistossa oli omat kansiot JavaScriptille, Kuvatiedostoille ja tyylimäärittelyille. Lisäksi tyylimäärittelyt jaettiin kokonaisuuksien mukaisiin osioihin omiksi tiedostoikseen. (KUVIO 14.)



KUVIO 14. Rakentamisen lupapolku -palvelun kehittämispuolen kansiorakenne

Sivupohjat eroteltiin ruotsin- ja suomenkielisiin hakemistoihin, joissa pääsivupohjan lisäksi kaikki lupaprosessin vaiheet olivat eroteltuina omiin sivupohjiinsa. Tällä tavoin yksittäisten sivujen ylläpito onnistuu jatkossakin vaivatta eikä oikeaa vaihetta tarvitse etsiä muiden joukosta. Jokaisen vaiheen sivupohja on aina nimetty kyseessä olevan otsikon mukaan.

6.3.2 Valitut päätyökalut

Koodieditorina kehityksessä käytettiin Visual Studio Code -ohjelmaa. Sille saatavat erilaiset laajennukset olivat myös tarpeen kehittämisvaiheessa. Erityisesti eri esikäsitelijöiden syntaksin väritys sekä virheenkorjaus saatiin toimimaan.

Tyylimärittelyissä käytettiin hyödyksi Sass-esikäsittelijää sekä muita sitä tukevia aputyökaluja ja -kirjastoja. Tyylien määrittelyissä käytettiin lisäksi mobile first -suunnitteluperiaatetta sekä elementtien nimeämisessä BEM-nimeämiskäytäntöä. CSS-esikäsittelijäksi valittiin Sass sen monipuolisuuden ja aktiivisen kehityksen takia.

HTML-esikäsittelyä varten valittiin Pug-sivupohjamoottori. Informaatiopalvelussa esitetyn lupaprosessin jokaisen eri vaiheen sisältö voitiin Pugin avulla eritellä kaksikielisenä omiin

sivupohjiinsa. Tällä tavoin pääsivun sivupohja pysyy ylläpidon kannalta selkeänä ja kaikki muut erilliset sivupohjat ovat yksittäin helposti muokattavissa. Pugin käyttö lisäksi nopeutti huomattavasti kaiken HTML-koodin kirjoittamista minimalistisen syntaksinsa ansiosta.

Sivuston toiminnallisuutta varten kehitettiin mahdollisimman yksinkertainen koodilogiikka TypeScriptillä, joka hallitsee lupaprosessin vaiheiden aktiiviseksi asettamisen ja vastaavan sisällön näyttämisen. Koodi muokkaa HTML:ää, ja luokkamäärittelyiden avulla elementtejä voidaan tarvittaessa piilottaa tai asettaa näkyväksi. Tällä tavoin vältytään turhilta latausajoilta, eikä vaiheita varten tarvita erillisiä HTML-sivuja. Yhden sivun käyttämisen etuna lisäksi käyttäjän eteneminen lupaprosessissa säilyy vaiheita vaihtaessa automaattisesti, kun sivu ei päivity muuten kuin kieltä vaihtaessa.

Edellä mainittujen työkalujen lisäksi esimerkin tekemisessä hyödynnettiin Webpack-automaatiotyökalua. Webpackin avulla automatisoitiin muun muassa lähdetiedostojen käsittely ja sivuston pakkaaminen tuotantoa varten. Lisäksi Webpackin ja siihen lisättyjen lisäosien avulla sivustoon tehdyt muutokset saadaan päivittymään aina automaattisesti nähtäville kehittämisvaiheessa. Webpack tarkastelee muutoksia lähdetiedostoissa ja suorittaa määritellyt komennot muutosten tallentuessa tiedostoihin; kuten kuviossa 15, jossa näkyvät Webpack-asetustiedoston Sass-esikäsittelijää varten kirjoitetut määrittelyt. Muutosten reaaliaikainen päivitys nopeuttaa kehittämistyötä huomattavasti.

```
43        },
44        {
45          test: /\.scss$/,
46          use: ExtractTextPlugin.extract({
47            fallback: 'style-loader',
48            use: [
49              'css-loader',
50              'postcss-loader',
51              'sass-loader'
52            ]
53          })
54        }
```

KUVIO 15. Kuvankaappaus Webpack-asetustiedoston Sass-osiosta

6.3.3 Esikäsittelijöiden ja apukirjastojen edut

Sass-esikäsittelijälle yhteensopivia ja sitä varten tehtyjä hyödyllisiä apukirjastoja löytyy monia. Asettelun tekemiseen valittiin Lostgrid-apukirjasto (http://lostgrid.org), joka tuottaa halutun ruudukkoasettelun tyylitiedostosta käsin käyttäen PostCSS:ää. PostCSS on JavaScriptillä toimiva työkalu, joka käsittelee CSS:ään lisättyä uutta syntaksia ja palauttaa normaalin CSS-tyylitiedoston (Ćwirko 2016). PostCSS ei ole millään tavalla sidonnainen esikäsittelijöihin ja sitä voi hyödyntää esimerkiksi tavallisen CSS:n kanssa.

PostCSS:n mukaan lisättiin myös Autoprefixer-laajennus (https://github.com/postcss/autoprefixer), joka lisää automaattisesti eri selainten tarvitsemat lisämäärittelyt eri tyyleille kuviossa 6 esitetyn esimerkin tapaan. Ilman Autoprefixer-laajennusta määrittelyt täytyisi lisätä manuaalisesti jokaiselle elementille täyttäen samalla Sass-tyylitiedostoa turhaan. Lisäksi myös At-break-niminen kevyt Sass-apukirjasto (https://github.com/manumorante/sass.at-break) otettiin käyttöön, jotta erityisesti ruudukkoasettelusta saatiin responsiivinen. At-break:n mixin-yhdistelmätyylit yksinkertaistivat media query -määrittelyiden, eli ruutukoon mukaan muuttuvien määrittelyiden käyttämistä (ks. kuvio 4).

Pug-sivupohjamoottori mahdollisti monia kehityksen ja ylläpidon kannalta hyödyllisiä rakenteita. Palvelulle olennainen modulaarinen sisällönhallinta toteutui Pugin avulla linkitettyjen tiedostojen kautta kaksikielisenä suomeksi ja ruotsiksi.

6.3.4 Lopputulos

Lopputuloksena on edellä kuvatulla tavalla toimiva responsiivinen informaatiopalvelu: Rakentamisen lupapolku. Liitteessä 1 näkyvä alustava prosessikaavio onnistuttiin tiivistämään liitteen 2 mukaiseen, lopullisella sivustollakin esiintyvään versioon säilyttäen kuitenkin kaiken alkuperäisen informaation. Kaikki tarvittavat ominaisuudet saatiin sisällytettyä palveluun. Palvelun responsiivisuus on tärkeä ominaisuus, sillä varsinkin rakennustyömailla mobiililaitteiden käyttö on yleistä.

Kunta kerää luvanhakijoilta ja muilta Rakentamisen lupapolun käyttäjiltä palautetta, jotta kokonaisuutta voidaan edelleen kehittää. Lisäksi rakentamisen lainsäädännön muuttumisen myötä informaatiopalvelun jatkuva kehittäminen on ensiarvoisen tärkeää ajantasaisen tiedon jakamiseksi. Palautekanava löytyy Rakentamisen lupapolun yhteydestä Sipoon kunnan kotisivuilta.

KUVIO 16. Kuvankaappaus Rakentamisen lupapolku -informaatiopalvelun etusivusta

Lisäksi tunnistettavuutta ja positiivista imagoa luomaan sivun taustalle kehitettiin skaalautuva vektori-illustraatio svg-muodossa Adobe Illustrator -ohjelmalla. Illustraatio kuvastaa rakennettua sipoolaista ympäristöä. (KUVIO 17.) Kuvan värimaailma ja sivuston tunnusvärit sopivat keskenään yhteen ja piristävät palvelun ulkoasua. Palvelussa käytetyt elementit suunniteltiin moderniin minimalistiseen tapaan, jota myös vektori-illustraatio tukee yksinkertaistetuilla muodoillaan.

KUVIO 17. Vektori-illustraatio rakennetusta sipoolaisesta ympäristöstä palvelun taustalla

Rakentamisen lupapolku julkaistiin marraskuussa 2017 Sipoon kunnan kotisivuilla. Paikallislehdet, Sipoon Sanomat ja Uusimaa, kirjoittivat aiheesta artikkeleita ja myös Sipoon kunnanjohtaja julkaisi aiheessa Twitterissä. Jatkuva Rakentamisen lupapolusta tiedottaminen ja asiakkaiden muistuttaminen tapahtuu rakennusvalvonnan henkilöstön sähköpostin allekirjoituksiin sisältyvällä linkillä.

> *Luvanhakijaa ohjaa ja neuvoo nyt Rakentamisen lupapolku. Sipoon kunnan kotisivuilla on nyt julkaistu visuaalinen esitys nimeltä Rakentamisen lupapolku, jonka tarkoituksena on helpottaa rakennusluvan hakemista ja selkeyttää koko rakentamisprojektia. (Ropponen 2017, 10.)*

7 YHTEENVETO

Rakennusvalvonnassa oli luvanhaussa haasteeksi osoittautunut lupaprosessin ymmärtäminen ja siihen liittyvän termistön vaikeus. Tavoitteena oli helpottaa rakennusvalvonnan asiakkaille rakennusluvan hakemista ja selkeyttää koko rakentamisprojektia. Sipoon kunnassa kehitetään voimakkaasti palvelujen digitalisointia kaikkien palvelujen osalta, joten tällaiselle ohjeistukselle oli Sipoon kunnassa merkittävä tarve ja tilaus.

Tarvetta vastaamaan kehitettiin rakennusvalvonnan kotisivuilla julkaistu visuaalinen Rakentamisen lupapolku. Rakentamisen lupapolku auttaa hahmottamaan vaikeaselkoisen lupaprosessin käyttäjäystävällisen informaatiopalvelun avulla. Esikäsittelijät auttoivat ominaisuuksillaan sivuston kehittämisvaihetta ja käytetyt rakenteet helpottavat lisäksi myös jatkossa palvelun ylläpitoa.

Sass sekä siihen yhdistetyt apukirjastot ja työkalut ratkaisivat monia ongelmia elementtien tyylien hallinnan kanssa. Sass ja PostCSS vähensivät huomattavasti tarvittavien tyylimäärittelyiden lukumäärää ja selvensivät ulkoasun kokonaisuutta ja sen hallintaa. Apukirjastot tekivät responsiivisen ruudukon luomisesta sekä responsiivisuuden toteuttamisesta vaivatonta. Kaikki tietyn elementin määrittelyt, lapsielementit sekä lisämäärittelyt olivat hallittavissa tyylitiedostossa saman kokonaisuuden alta.

Pug-sivupohjamoottorin kehittämisvaiheeseen tuoma modulaarinen tiedostorakenne oli yksi suurimmista hyödyistä palvelun rakentamisessa. Palvelussa esitetyn lupaprosessin jokaisen eri vaiheen sisältö voitiin Pugin avulla eritellä kaksikielisenä omiin tiedostoihinsa, mikä selkeyttää ja nopeuttaa varsinkin sisällön ylläpitoa.

Rakentamisen lupapolku -informaatiopalvelu on toteutunut ja kaikkien käytettävissä Sipoon kunnan kotisivujen kautta osoitteessa lupapolku.sipoo.fi. Palvelusta kerätään jatkuvasti palautetta ja myös sen jatkokehittäminen on mahdollista. Rakentamisen lupaprosessi saatiin palvelua varten tiivistettyä helposti ymmärrettävään muotoon. Prosessin vaiheet esitetään käyttäjälle visuaalisesti selkeällä tavalla aikajanan muodossa.

Esikäsittelijät soveltuivat erinomaisesti staattisen ja responsiivisen informaatiopalvelun kehitykseen. Palvelu toimii jokaisella ruutukoolla ja mukautuu laitteelle sopivaksi. Sivuston tunnusvärit ovat helposti muokattavissa muuttujien avulla jälkikäteen.

Sivupohjamoottori mahdollisti monia kehityksen ja ylläpidon kannalta hyödyllisiä rakenteita. Palvelulle olennainen modulaarinen sisällönhallinta toteutui linkitettyjen sivupohja-

tiedostojen avulla. Tällä tavoin jokaisen vaiheen sisältö saatiin erilleen helposti muokattavaksi. Pugin käyttö lisäksi nopeutti huomattavasti kaiken HTML-koodin kirjoittamista minimalistisen syntaksinsa ansiosta.

Informaatiopalvelussa käytetty rakenne sopisi mielestäni myös muiden prosessien kuvaamiseen ja vaikeaselkoisten asioiden avaamiseen. Laajempaa käyttöä varten tarvittaisiin sisällönhallintajärjestelmä, jota Rakentamisen lupapolku ei vaatinut. Jos jatkossa käyttöön haluttaisiin yksinkertainen sisällönhallintajärjestelmä, niin se olisi mielestäni parasta toteuttaa itse, jotta siitä saataisiin yksilölliseen tarpeeseen sopiva ilman ylimääräisiä ominaisuuksia. Sipoon rakennusvalvontapäällikön mukaan kyseisen palvelun konsepti on herättänyt mielenkiintoa muissakin kunnissa ja kaupungeissa.

LÄHTEET

BEM 2017. How does BEM differ from OOCSS, AMCSS, SMACSS, SUITCSS? [viitattu 6.3.2017]. Saatavissa: https://en.bem.info/methodology/faq/#how-does-bem-differ-from-oocss-amcss-smacss-suitcss

Ćwirko, J. 2016. Some things you may think about PostCSS... and you might be wrong [viitattu 12.3.2017]. Saatavissa: http://julian.io/some-things-you-may-think-about-postcss-and-you-might-be-wrong/

Google Inc 2015. Responsive Web Design [viitattu 12.2.2017]. Saatavissa: https://developers.google.com/webmasters/mobile-sites/mobile-seo/responsive-design

Google Inc 2017. Resizer [viitattu 6.3.2017]. Saatavissa: http://material.io/resizer/

Less.js 2017. Language Features [viitattu 9.3.2017]. Saatavissa: http://lesscss.org/

Nielsen, J. 2011. Top 10 Mistakes in Web Design [viitattu 10.2.2017]. Saatavissa: http://www.nngroup.com/articles/top-10-mistakes-web-design/

Roberts, H. 2017. High-level advice and guidelines for writing sane, manageable, scalable CSS. Naming conventions [viitattu 6.3.2017]. Saatavissa: http://cssguidelin.es/#naming-conventions

Ropponen, P. 2017. Rakentamisen lupapolku ohjaa luvanhakijaa. Uusimaa 24.11.2017.

Sass 2016. Sass (Syntactically Awesome StyleSheets) [viitattu 11.2.2017]. Saatavissa: http://sass-lang.com/documentation/file.SASS_REFERENCE.html

Sass 2017. LibSass [viitattu 19.4.2017]. Saatavissa: http://sass-lang.com/libsass

Schade, A. 2014. Responsive Web Design (RWD) and User Experience [viitattu 11.2.2017]. Saatavissa: http://www.nngroup.com/articles/responsive-web-design-definition/

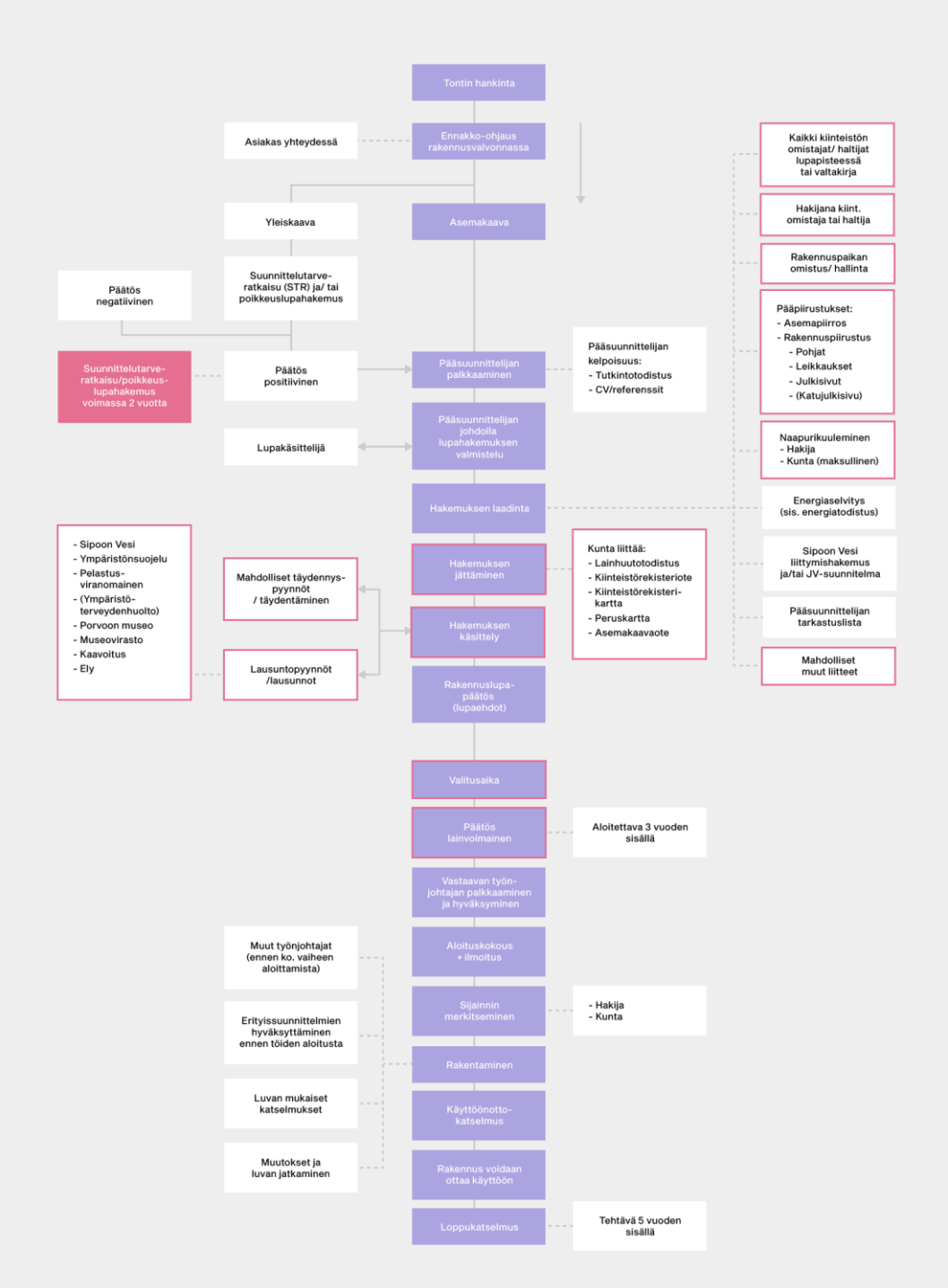
Stylus 2017. Stylus - Expressive, Dynamic, Robust CSS [viitattu 11.3.2017]. Saatavissa: http://stylus-lang.com/

Upola, U.-M. 2017. Rakennusvalvontapäällikkö. Sipoon kunta. Haastattelu 10.8.2017.

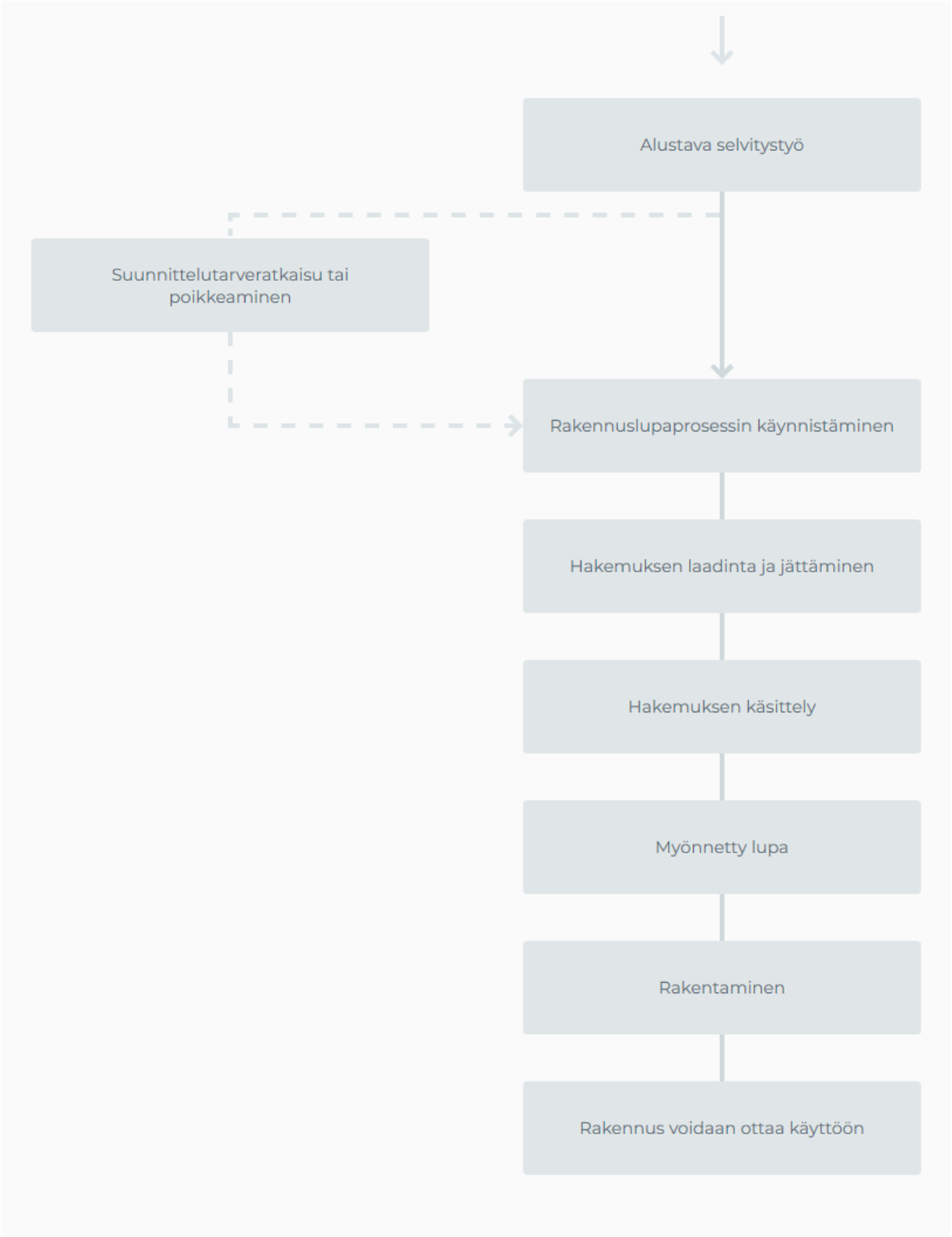
Virta, A. 2017. Yhdyskuntamuotoilija. Sipoon kunta. Tiedote.

Voutilainen, J.-P., Salonen, J. & Mikkonen, T. 2015. On the Design of a Responsive User Interface for a Multi-Device Web Service [viitattu 12.2.2017]. Saatavissa: http://ieeexplore.ieee.org.aineistot.lamk.fi/stamp/stamp.jsp?tp=&arnumber=7283029

LIITTEET



LIITE 1. Kaavio lupaprosessista kokonaisuudessaan



LIITE 2. Tiivistetty versio rakennuslupaprosessin vaiheista Rakentamisen lupapolku -sivustolla