Sam Roos

# Tyylikehys verkkokehittäjille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

11.4.2016

Metropolia

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Sam Roos<br>Tyylikehys verkkokehittäjille<br><br>26 sivua<br>11.4.2016 |
| Tutkinto | Insinööri (AMK) |
| Koulutusohjelma | Mediatekniikka |
| Suuntautumisvaihtoehto | Digitaalinen media |
| Ohjaaja | Yliopettaja Harri Airaksinen |


Insinöörityön tavoitteena oli tehdä verkkokehittäjille tyylikehys, joka on helppo ja nopea opetella. Työn toisena tavoitteena oli, ettei kehys sisältäisi lainkaan verkkosivujen ohjelmointikieltä.

Insinöörityö sisälsi vaiheet tyylikehyksen suunnittelusta aina toteutukseen saakka. Tyylikehyksen sisältö määrittyi tutkimalla muita kehyksiä, sekä aikaisempien projektien myötä. Tyylikehyksen ominaisuuksia ovat muun muassa responsiivisuus, Parallax-rullaus ja valikot, jotka rakennettiin kaikki erillisinä osioina. Myöhemmin nämä osiot yhdistettiin yhdeksi kokonaisuudeksi. Työssä käytettiin CSS- ja HTML-merkintäkieliä, joiden avulla muodostettiin esimerkit tyylikehyksen osioista. Tyylikehyksen esimerkkisivu on rakennettu Wordpress-nimisen sisällönhallintajärjestelmän päälle.

Insinöörityön tuloksena syntyi toimiva tyylikehys, joka helpottaa kaikkia web-kehittäjiä tekemään työläitä osioita ilman omia tyylisääntöjä. Projektin aikana opittiin paljon kehysten tekemiseen vaadittavasta työstä ja huolellisesta suunnittelusta. Insinöörityön tuloksista ilmeni, että uuden tyylikehyksen luominen vaatii sen tekijältä runsaasti aikaa ja paneutumista. Projektin aikana ilmenivät myös oman tyylikehyksen rakentamisen edut ja mahdolliset haitat.

Tyylikehyksen oltua pidempään käytössä voidaan ymmärtää sen lopullinen merkitys ja hyöty. Tyylikehys ei todennäköisesti koskaan ole täysin valmis, vaan sitä voidaan kehittää jatkuvasti. Kehykseen voidaan lisätä uusia ominaisuuksia, jotta siitä on hyötyä kehittäjille myös tulevaisuudessa.



| | |
|---|---|
| Avainsanat | CSS, framework, HTML, front-end, tyylikehys |

| | |
|---|---|
| Author<br>Title<br><br>Number of Pages<br>Date | Sam Roos<br>Front-end framework for web developers<br><br>26 pages<br>11.4.2016 |
| Degree | Bachelor of Engineering |
| Degree Programme | Media |
| Specialisation option | Digital |
| Instructor | Harri Airaksinen, Principal Lecturer |

The goal of the study was to make a front-end framework for web developers that is easy to use and fast to learn. The second goal of the study was that the framework would not include any programming language like JavaScript.

The thesis describes all stages from planning to execution. The content of the front-end framework was made by investigating other frameworks and by combining recent projects combined. A few of these features are responsiveness, Parallax-scrolling and menus. All of these features were made separately and combined later on. The front-end framework was made by using the CSS and HTML markup languages. Also the examples were made by using these markup languages. The example page of the front-end framework was made on top of a content management system called Wordpress.

As a result, a working front-end framework was created, which makes the work of web developers easier without creating any own rules to the style sheet. During the project a lot was learned about making a framework and about careful planning. The results indicate that making a new front-end framework requires a lot of time and commitment. The project also showed what the benefits and the possible disadvantages of making one's own framework are.

When the front-end framework has been in use for a longer time, its final purpose and benefits can be understood. The framework will probably never be fully completed, but it can be developed continually. More features can be added to the framework, so that it will be useful to developers in the future.

| | |
|---|---|
| Keywords | CSS, framework, HTML, front-end, front-end framework |

## Sisällys

# 1 Johdanto

Insinöörityössä on tarkoitus tehdä karsittu avoimen koodin tyylikehys verkkokehittäjille. Tyylikehykseen on tarkoitus tehdä mahdollisimman yksinkertainen rakenne, jotta sen opiskeluun ei menisi liikaa aikaa. Tyylikehyksestä pyritään karsimaan mahdollisimman paljon pois, jotta tiedostokoko pysyy pienenä, mutta silti auttaa kehittäjää tekemään projektin perusasiat tehokkaasti. Insinöörityötä ei tehdä millekään yritykselle, vaan tarkoituksena on tehdä tyylikehys, jota voi käyttää kuka vain ilman opiskelua.

Insinöörityö tehdään lokaalisti omalle tietokoneelle. Valmis työ siirretään myöhemmin palvelimille ja yleiseen jakeluun. Työ tehdään Wordpress-alustalle, joka sisältää myös dokumentaation.

Insinöörityöraportissa käydään läpi ensimmäiseksi verkkosivustojen merkintäkielien ja tyylikehysten historiaa. Tämän jälkeen työssä paneudutaan itse kehyksiin. Kehyksien ominaisuudet koetetaan selvittää: minkälaisia kehyksiä on olemassa ja minkä takia on hyvä tehdä oma kehys. Työn loppupuolella käydään läpi työn toteutuksen vaiheet ja mitä toteutuksesta vielä puuttuu.

Työn toteutuksia pyritään selventämään kuvilla ja koodiesimerkeillä. Työn suunnittelu ja toteutus pohjautuu aiemmin tehtyihin projekteihin ja lähteiden tukemiin väittämiin kehyksen tarpeista. Työssä pyritään tekemään tarvittavat testaukset, jotta lopullisessa versiossa ei olisi ohjelmointivirheitä. Testauksen tärkeyden painotus on myös olennainen osa työn laadun varmistamiseksi.

Lopuksi työssä selvitetään tyylikehyksen tekemisen tuloksia ja vaiheiden ongelmakohtia. Tuloksissa pyritään selkeyttämään, mitkä osiot ovat hyödyllisiä ja olisiko työssä kannattanut tehdä jotain toisin. Ongelmakohdat käydään tarkasti läpi ja selvennetään, onko ongelma ratkennut vai onko se vielä avoin.

# 2 CSS- ja HTML-merkintäkielet ja tyylikehykset

## 2.1 CSS-merkintäkieli

CSS (Cascading Style Sheets) on tyylien määrittämisen merkintäkieli, jonka avulla voidaan käsitellä verkkosivustojen ulkoasua. Ensimmäinen toimiva versio CSS-kielestä julkaistiin vuonna 1995, alkeellisia versioita oli jo vuoden 1995 alkupuolella. Kielen kehittäjiä ovat Bert Bos ja Håkan Lie. Tässä vaiheessa kieltä kutsuttiin nimellä CSS1. Tässä vaiheessa kielen päätarkoituksena oli saada mahdollisuus käsitellä HTML-tiedostojen tekstin tyylejä, kuten värejä, fontteja ja rivien välitystä. Ensimmäisessä vaiheessa oli myös mahdollista luoda rajoja tekstin ympärille. CSS-kielestä tuli vuonna 1996 W3C (World Wide Web Consortium) -yhteisön suosittelema. [1.]

CSS2-kieli tehtiin edellisen kielen päälle. Toinen versio julkaistiin vuoden 1997 lopussa, minkä jälkeen meni puoli vuotta, kunnes siitä tuli W3C-yhteisön suosittelema. CSS2-kielen kehittäjiin liittyivät Ian Jacobs ja Chris Lilley. Versio toi tullessaan mediaelementit, joiden avulla esimerkiksi pystyttiin luomaan sivuille ominaisuuksia, joiden avulla sokeat voivat käydä sivuilla. Lisäksi versiossa laajennettiin elementtien asemointiominaisuuksia. [2.]

CSS-kielen seuraavan vaiheen nimeksi tuli CSS2.1. Siinä lisättiin tyyliominaisuuksia ja poistettiin tiettyjä osia. Poistettujen osioiden on kuitenkin suunniteltu tulevan seuraavaan vaiheeseen. Tavoitteena oli luoda mahdollisimman moneen selaimeen tuki, jotta kaikkia ominaisuuksia pystytään käyttämään selaimesta riippumatta. Paranneltu versio toisesta CSS kielestä tuli vuonna 2008. [3.]

Tämänhetkinen tyylikielen nimi on CSS3, ja sitä korjataan ja parannellaan parhaillaan. Edellinen versio on siirtynyt kokonaisuudessaan uuden version pohjaksi. Uuteen versioon on tullut esimerkiksi valitsijat sekä muutamia pseudoelementtejä. Nämä lisäykset antavat uusia mahdollisuuksia kielen kirjoittamiseen. Esimerkiksi yhdellä luokalla voidaan määrittää joka toisen elementin arvoiksi erilaisia arvoja. [4.]

## 2.2 HTML-merkintäkieli

HTML (Hypertext Markup Language) on verkkosivustojen merkintäkieli, jolla tehdään sivuston rakenne ja sisältö. Kielessä käytetään tunnisteita, jotka alkavat pienempi kuin -merkillä ja päättyvät suurempi kuin -merkkiin. Hyvinä esimerkkeinä voivat toimia tunnisteet, jotka löytyvät jokaisesta HTML-tiedostosta, <html>, <head>, <body>. Tunnisteilla pitää olla myös sulkeva tunniste, joka loppuu kenoviivaan, esimerkiksi <html/>. Aloittavan ja lopettavan tunnisteen väliin tulevat siihen osioon kuuluvat tunnisteet tai data. Tunnisteilla voi olla attribuutteja, kuten esimerkiksi tyyli, nimi, luokka tai identifikaatio. Attribuuttien avulla pystytään vaikka tunnistamaan tietyn luokan avulla sille kuuluvat tyylit. [5.]

HTML-kielen kehitti henkilö nimeltä Tim Berners-Lee vuonna 1991. Hänet on myös mainittu internetin keksimisen yhteydessä. Kieli kehittyi melko nopeasti, sillä vuodesta 1991 vuoteen 1998 mennessä kieli oli kehittynyt versiosta 1 aina versioon 4 asti. [6; 7.]

Yhdistys nimeltä WHATWG (Web Hypertext Application Technology Working Group) halusi tehdä kielestä ainoan, jota käytettäisiin verkkoon kirjoittamiseen. Lisäksi WHATWG halusi kehittää kieltä taaksepäin tuetuksi, eli vaikka kieltä kehitetään, se toimii vanhemmillakin versioilla. Vuosina 2004–2006 yhdistys sai uudelle ideallensa tukea, sillä suurimmat selaimet antoivat sille täyden tuen. Myös W3C antoi ilmoituksen tuestaan WHATWG:lle vuonna 2006, ja sen ansiosta vuosia myöhemmin WHATWG määritti HTML-kielestä tulevan niin sanottu elävä standardi. Elävä standardi tarkoittaa sitä, että kieli ei ole koskaan valmis, vaan sitä parannellaan ja kehitetään jatkuvasti. Elävästä standardista ei oteta mitään pois, mutta siihen voidaan lisätä sisältöä jatkuvasti. [6; 7.]

Nykyinen kielen versio on HTML 5.0, josta tuli standardi vuonna 2014, ja siitä on tekeillä seuraava versio 5.1. Ensimmäinen julkinen vedos versiosta 5.0 on tullut vuonna 2008, ja neljä vuotta myöhemmin siitä alettiin tehdä standardia. [6; 7.]

## 2.3 Tyylikehykset

Front end -kehysten (eli selainpuolen kehysten) historiasta ei ole kirjoitettu juurikaan, sillä julkiset avoimen lähdekoodin kehykset ovat tulleet esille vasta tällä vuosikymmenellä. Kehysten teko ei kuitenkaan ole mitään uutta teknologiaa. Yritykset ovat tehneet

omia kehyksiä jo pitkän aikaa, mutta niitten hyödyt ovat olleet vain yritysten sisäisiä. Nykyään monissa yrityksissä käytetään olemassa olevia avoimen lähdekoodin kehyksiä, mikä auttaa uusia työntekijöitä pääsemään nopeammin yrityksen tapoihin mukaan. Mikäli yritys käyttää paljon esimerkiksi Bootstrap-nimistä kehystä, voi uudella työntekijällä olla jo taustaa sillä työskentelystä. Usein vaaditaan työnhaun yhteydessä jo jonkin kehyksen tuntemista. Kehysidea on ollut esillä jo pitkään ennen front end -kehyksiä. Esimerkiksi .Net-ohjelmointikielelle tehtyjä kehyksiä on ollut jo vuodesta 2002. [8; 14.]

# 3 Verkkokehittäjän tyylikehykset

## 3.1 Tyylikehys

Kehys tarkoittaa jonkinnäköistä valmista pohjaa, joka auttaa verkkosivuston rakentajaa rakentamaan jonkin asian tehokkaasti. Kehyksessä tehdään rakenne, joka määrittää ennalta tiettyjä asioita, joita tarvitaan yleisesti rakennuksen eri vaiheessa. Kehyksen tehtävä ei ole rakentaa mitään valmiiksi, vaan tarjota tekijälle lisäominaisuus. Lisäominaisuus liitetään tekijän tekemään osioon, mikä nopeuttaa projektin tekemistä. [10.]

CSS-tyylikehys on CSS-tyylitiedosto, joka sisältää samanlaista tietoa, jota kirjoitetaan tavallisesti tyylitiedostoon. Tyylikehyksessä on valmiiksi määriteltyjä tyylejä, joiden tarkoitus on ratkoa yleisiä ongelmatilanteita verkkosivuston tyyleissä. Tyylikehyksen käyttäminen nopeuttaa verkkosivuston tekemistä, koska kaikkia tyylimääritelmiä ei tarvitse kirjoittaa aina uudestaan. Hyvänä esimerkkinä toimii verkkosivuston responsiivisuuden määrittäminen, joka vaatii erittäin monta riviä koodia tyylitiedostoon. Tämän voi kuitenkin tehdä tyylikehykseen valmiiksi, jolloin käyttäjän ei tarvitse muistaa kuin kyseiset valmiiksi luodut luokat. Valmiita tyylikehyksiä on monia erilaisia, vaikka moni kehys toimii melkein samalla tavalla. [11.]

Tyylikehyksiä voidaan tehdä suoraan tyylitiedostoon CSS-kieltä käyttäen, mutta nykyään yhä useammat tekevät oman tyylikehyksen käyttäen apuvälineitä. Suosituimmat tavat tehdä tyylikehyksiä ovat varmasti LESS ja SASS, jotka ovat merkintäkieliä. Näiden kielien avulla pystytään tekemään tyylejä vähäisemmällä kirjoitusmäärällä. Kummallakin näistä kielistä on omat kääntäjät, jotka tekevät lopullisen CSS-tiedoston. Esimerkiksi SASS-kieli käyttää Ruby-ohejlmointikieltä käännökseen ja tiedoston luontiin. [12.]

## 3.2 Erilaiset kehykset

Kehyksiä on olemassa erittäin paljon, ja kaikki ovat erilaisia. Kehyksien tavoitteetkin poikkeavat toisistaan. Jonkin kehyksen tavoite on tuoda kaikki tarpeellinen kehittäjän ulottuville, kun taas toinen kehys pitää tärkeänä luoda vain pohjan ja pieniä osia, jotta kehittäjän olisi helppo rakentaa sen päälle. Kehykset voivat sisältää monta eri kieltä, kuten esimerkiksi CSS ja Javascript. Seuraavassa esitellään muutama suosittu kehys ja niiden ominaisuuksia. [13.]

### Bootstrap

Bootstrap on suunniteltu mobiililaitteille kehittämistä varten. Se (http://getbootstrap.com/) sisältää monta eri kokonaisuutta, ja tätä kehystä käyttäessä tulee ladata monia CSS-, Javascript- ja fonttitiedostoja. Bootstrap on erittäin monipuolinen, ja sen avulla voi rakentaa nopeasti eri asioita, mutta sen opettelu vie aikansa. Bootstrapin opiskelun takia muutamat ensimmäiset projektit saattavat viedä tavallista enemmän aikaa, mutta kun tottuu sitä käyttämään, aikaa säästyy paljon. Kehittäjinä ovat olleet Mark Otto ja Jacob Thornton. Bootstrap-kehys julkaistiin vuonna 2011, ja tämänhetkinen versio on 3. Neljättä versiota tehdään parhaillaan: se on tällä hetkellä Alpha-tasolla. [13; 14.]

### Pure

Pure (http://purecss.io/) on toisesta ääripäästä kuin Bootstrap: se on tehty mahdollisimman minimaaliseksi. Pure sisältää vain yhden tyylitiedoston, joka sisältää pohjan, taulukkojärjestelmän, napit, lomakkeet, valikot ja taulukot. Kehys on pieni, joten sen käyttäminen on alusta alkaen nopeaa. [13.]

### Skeleton

Skeleton (http://getskeleton.com/) kuuluu yhdessä Puren kanssa pienikokoisiin kehyksiin. Tämän kehyksen tiedostoja on vain kolme, ja niihin kuuluu samat ominaisuudet kuin Puressa. Kolmen tiedoston jako tekee luettavuudesta hieman helpompaa, vaikka koko on melkein sama. Skeleton mainostaa itseänsä pienenä ja helposti käyttöönotettavana kehyksenä. Kehyksen tekijä on Dave Gamache. [13; 14.]

Foundation

Foundation (http://foundation.zurb.com/) kuuluu kokonaisvaltaisiin kehyksiin, kuten Bootstrap. Ominaisuuksia on kaikkiin tarkoituksiin, ja opetteleminen vaatii oman aikansa. Kehyksen on luonut yhtiö nimeltä ZURB. Foundation ja Bootstrap eroavat syntaksiltaan, vaikka samankaltaisuuksiakin on paljon. Lisäksi Foundationissa on hieman enemmän ominaisuuksia kuin Bootstrapissa. [14.]

Näiden kehyksien lisäksi on monta muuta suositeltua kehystä. Kaikissa on eri ominaisuudet ja tuet. Kehykset ilmoittavat oman tukensa eri selaimissa ja mitä tekniikoita ne tukevat. Esimerkiksi Bootstrap tukee Internet Explorerin versiota 8 ja sen jälkeisiä. Muista selaimista se tukee viimeisintä versiota, ja se sisältää LESS- ja SASS-kääntäjien tiedostot. Tämän lisäksi listassa ilmoitetaan, tukeeko kehys puhelinnäyttöjä ja kenelle lisenssi kuuluu. [Kuva 1.]

| Name | Ie | Chrome | Firefox | Opera | LESS | SASS | Mobile | License |
|---|---|---|---|---|---|---|---|---|
| Bootstrap | 8+ | latest | latest | latest | ✔ | ✔ | ✔ | MIT |
| Semantic UI | 9+ | latest | latest | 12+ | ✔ | ✖ | ✔ | MIT |
| Foundation | 9+ | latest | latest | latest | ✖ | ✔ | ✔ | MIT |
| UIkit | 9+ | latest | latest | latest | ✔ | ✖ | ✔ | MIT |
| 960 Grid System | 7+ | latest | latest | latest | ✖ | ✖ | ✔ | GPL & MIT |
| Skeleton | 7+ | latest | 3+ | latest | ✖ | ✖ | ✔ | MIT |
| 99lime HTML KickStart | 8+ | latest | latest | latest | ✖ | ✖ | ✔ | MIT |
| Kube | 8+ | latest | latest | latest | ✔ | ✖ | ✔ | Open Source |
| Less Framework | 7+ | latest | latest | latest | ✔ | ✖ | ✔ | MIT |
| Flaminwork | 7+ | latest | latest | latest | ✖ | ✖ | ✖ | Open Source |
| G5 Framework | 7+ | latest | latest | latest | ✖ | ✖ | ✖ | Open Source |
| Easy Framework | 7+ | latest | latest | latest | ✖ | ✖ | ✖ | CC |
| Blueprint | 7+ | latest | latest | latest | ✖ | ✖ | ✖ | MIT |
| YAML | 6+ | latest | 3.6+ | 10+ | ✖ | ✔ | ✔ | CC-BY |
| BlueTrip | 7+ | latest | latest | latest | ✖ | ✖ | ✖ | Open Source |
| YUI CSS | 8+ | latest | latest | latest | ✖ | ✖ | ✔ | BSD License |
| 52framework | 6+ | 4+ | 3+ | latest | ✖ | ✖ | ✔ | CC |
| elastiCSS | 7+ | latest | latest | latest | ✖ | ✖ | ✔ | MIT |
| Boilerplate | 7+ | latest | latest | latest | ✖ | ✖ | ✔ | New BSD |
| Emastic | 5+ | 3+ | 3+ | 9+ | ✖ | ✖ | ✖ | Open Source |
| Malo | 5+ | 3+ | 1.5+ | 8+ | ✖ | ✖ | ✖ | MIT |
| Dismantle.gs | Latest | Latest | Latest | Latest | ✖ | ✖ | ✔ | MIT |

Kuva 1. Suositut front end -kehykset ja niiden tietoja [15].

Kuvan 1 kaltaiset listaukset auttavat kehittäjiä valitsemaan juuri heitä parhaiten palvelevat kehykset ja karsimaan ei toivotut. Listauksen lisäksi pitää vielä kartoittaa kehyksen sisällä olevat ominaisuudet, joita kehittäjä tarvitsee ja haluaa käyttää.

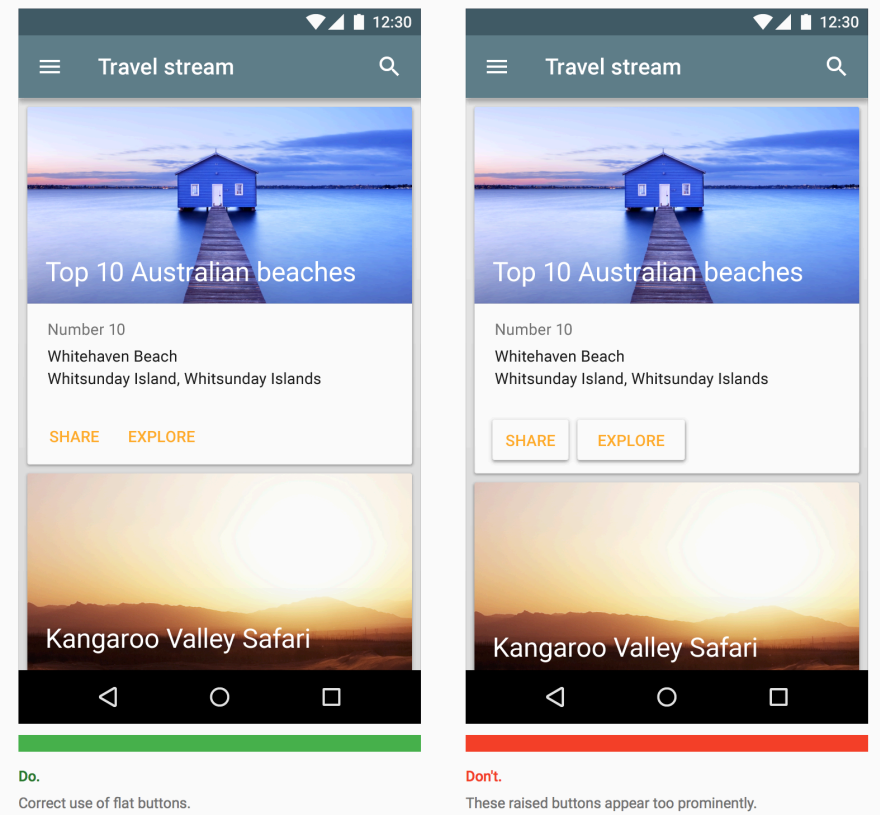
## 3.3 Tyylikehyksen sisältö

Tyylikehyksen sisällöksi valitaan sellaiset ominaisuudet, jotka toistuvat usein eri projekteissa. Tällaiset ominaisuudet voidaan kehyksen avulla ottaa käyttöön ilman, että niitä kirjoitettaisiin joka kerta uudestaan. Yleisin ominaisuus tyylikehyksissä on varmasti responsiivisuuden määrittäminen. Tyylikehyksen sisältö tulisi olla helposti opeteltavissa, mikä luo haasteita ominaisuuksien valitsemiseen. Valikkoja on joka sivustolla, ja varmasti myös on sama määrä erilaisia variaatioita kuin sivustoja. Valikkojen sisällyttäminen kehykseen on helppoa, mutta samalla on hankalaa valita oikeantyyppiset valikot.

Esimerkiksi responsiivisuusominaisuuden käyttöönotto tapahtuu HTML-koodin tunnisteen luokkamäärittelyssä. Luokan määrittelyyn kirjoitetaan kehyksessä ennalta määritelty luokka, joka määrittelee tässä tapauksessa tunnisteen leveyttä eri näytön leveyksissä. Esimerkiksi mikäli kehittäjä haluaa tunnisteen olevan puolen sivun levyinen pienillä näytöillä, tulee kehittäjän kirjoittaa luokan määrittelyn kohtaan ”koko-xs-12”. Tässä luokassa on määriteltynä solun leveys tietyllä näytön leveydellä. Kehittäjän tulee siis ainoastaan tietää luokkien nimet, mikä tarkoittaa, että kehittäjä säästää paljon aikaa, kun ei tarvitse kirjoittaa tyylejä erikseen. Mikäli kehittäjä tekisi tämän itse, tulisi hänen määrittää omat näytönleveyden katkopisteet ja niiden sisällä toimivat tyylit.

Verkkosivuston tulee nykypäivänä noudattaa monia laatustandardeja. Laatustandardit eivät ole kiveen kirjoitettuja, vaan ne tarkoittaa tekniikoita, joilla nykyään tehdään sivuja ja niiden osia. Responsiivisuus on nykypäivänä jo standardi, jota tulee noudattaa, mikäli haluaa luoda nykyaikaiset verkkosivut. Verkkosivujen selaaminen on siirtymässä koko ajan enemmän pienille laitteille. Tämän takia ei voida enää suunnitella sivustoja vain tietokoneen näyttöjä varten, vaan nykyään suositellaan puhelimen näytöille suunnattua suunnittelua. Tällaista suunnittelutapaa kutsutaan nimellä ”mobile first” eli mobiili ensin. Valikoiden tyylit ja responsiivinen toimivuus kuuluvat myös nykypäivän vaatimuksiin. Kaikki osiot verkkosivustolla tehdään tietyllä tavalla ja vertaillaan, miten kaikki on tehty.

Esimerkiksi napit olivat ennen selkeitä erillisiä painikkeita, joita korostettiin taustavarjoilla, reunuksilla ja reunojen viereen tehdyillä tummennuksilla. Nykyään sivuston yleisten nappien pitää olla hillittyjä, eivätkä ne saa näyttää liikaa napeilta, mutta silti sen verran, että niistä ymmärretään painaa. Poikkeuksia tietysti on, mikäli halutaan tahallaan korostaa jotakin osiota sivusta, mutta yleislinjaus on tämä. [17.] [Kuva 2.]




Kuva 2. Esimerkki siitä, miten nappi kannattaa tehdä ja miten sitä ei kannata tehdä [17].

Verkkosivuston nopeus on myös nykyään syytä ottaa tarkasti huomioon, sillä hakukoneet antavat paremmat tulokset kevyille sivuille. Sivuston latausajat koostuvat monista eri seikoista, ja tässä työssä pyritään kehyksen avulla poistamaan Javascript-kielen tuomia kuormituksia, esimerkkinä Parallax-vieritysominaisuuden tekeminen ilman Javascript-kieltä. Myös käyttäjät kiinnittävät huomiota latausaikoihin, koska verkkoyhteydet ovat nykyään erittäin nopeat. Tämän takia on syytä ottaa huomioon verkkosivuston nopeuden optimointi tarkasti. [9.]

### 3.4 Oman tyylikehyksen luominen vai valmiin käyttäminen

Kysymys, joka tulee varmasti kaikille mieleen, on, miksi kehittää oma, kun valmiita kehyksiä on jo paljon valikoimassa.

Oman tyylikehyksen luominen on hyvä keino opetella CSS-kieltä, ja omalla tyylikehyksellä voi luoda omantyyppisiä ominaisuuksia. Muiden tekemissä kehyksissä voi olla esimerkiksi yhdelle painikkeelle kymmenen eri tyyliä, joista useita ei tule koskaan käytettyä. Omaan kehykseen voi luoda muutamat painiketyylit, joita yleisesti tykkää käyttää. Valmiiden kehyksien ongelmana on yleisesti, se kuinka paljon ominaisuuksia niihin on laitettu. Kehittäjän tulee ensin löytää kehyksestä haluttu ominaisuus ja tämän jälkeen todennäköisesti vielä muokata ominaisuus oman ajatuksen mukaiseksi. Omassa kehyksessä ominaisuuksia voi lisätä tarpeen tullen kerralla halutun tyyliseksi. Tämän jälkeen se on aina käytettävissä ja kehittäjä tietää suoraan, minkätyylinen ominaisuus on.

Oman tyylikehyksen luominen auttaa enemmän pitkällä tähtäimellä kuin valmiin kehyksen kanssa tekeminen. Oman kehyksen tuntemisen ja sen persoonallisuuden vuoksi ei tarvitse aina muokata valmiita ominaisuuksia halutun tyyliseksi. Oman kehyksen koko ei ole yhtä iso, koska se sisältää vain niitä ominaisuuksia, mitä kehittäjä itse käyttää. Valmiin kehyksen vahvuutena on silti se, että kehykset ovat valmiiksi testattuja ja ominaisuuksien käyttämiseen on useita ohjeita verkossa muilta käyttäjiltä. [16.]

### 3.5 Tyylikehyksien hyödyt ja heikkoudet

Tyylikehyksen käyttämisessä on hyötyjen lisäksi aina omat ongelmansa. Tyylikehyksien tarkoitus on nopeuttaa ja poistaa pienet virheet projektissa. Kehyksien nopeus tulee usein silti esille vasta siinä vaiheessa, kun kehys on joko opeteltu tai itse tehty. Opettelu tai itse kehyksen tekeminen vie aluksi paljon aikaa, joten hyöty ensimmäisiin projekteihin on usein jopa negatiivinen. Kehyksen tultua tutuksi on sen käyttö erittäin hyödyllistä ja tehokasta. Valmiiden kehyksien heikkona puolena on myös oman kehityksen nopeus, sillä muiden kehyksiä käyttämällä ei opi itse tyylikoodista läheskään niin paljon kuin itse tehdystä. Mikäli aina on käyttänyt vain valmiita kehyksiä, ominaisuuksien muokkaaminen ja lisäyksien teko on hankalaa. Itse tehdyn kehyksen ohessa oppii ominaisuuksien tekemisen ja niiden muokkaamisen sekä itse tyylikoodin nopeamman lukemisen.

Valmiiden kehyksien käytössä on myös ongelmia, mikäli jotain ominaisuutta haluaisi muokata oman maun mukaan. Mikäli ominaisuuksia haluaa muokata, tulee kehyksen tuntemus olla perinpohjaista, sillä kehyksen sisällä saattaa olla monia elementtejä, jotka yliajavat toisiaan. Tämän takia muokkaus voi koitua tavallista haastavammaksi ja projektin tekemisen tehokkuus laskee huomattavasti. Samoin myös oman kehyksen tekemisessä: ominaisuuden muokkaus voi olla hankalaa, vaikka sen on itse suunnitellutkin.

Tyylikehyksien yhtenä ongelmana on myös toistuvuus ulkoasussa. Itse tehdyssä kehyksessä tämä ongelma ei välttämättä tule niin helposti esille kuin valmiissa kehyksissä. Valmiiden kehyksien ongelmana on se, että useat sivustot näyttävät samalta, koska niin moni käyttää samoja valikkoja ja rakenteita kehyksistä. Omissa kehyksissä vaarana on ominaisuuksien pieni määrä, mikä tarkoittaa sivujen samankaltaisuutta. Omissa sivuissa se on silti huomaamattomampaa, sillä todennäköisesti sivustoja ei ole niin paljon kuin joukolla, joka tekee samoilla säädöillä. [18.]

# 4 Tyylikehyksen luontivaiheet

## 4.1 Tyylikehyksen suunnittelu

Tyylikehyksen sisällön suunnittelu pohjautuu siihen, mitä ominaisuuksia käytetään eniten verkkosivustojen tekemisessä. Erilaisten kehyksien ominaisuuksia tutkimalla selvisi, että responsiivisuus, valikot ja alkuperäisten tyylien poisto ovat ne sellaisia ominaisuuksia, joita käytetään lähes kaikissa kehyksissä. Kehykseen valitaan ominaisuudet, jotka ovat työläitä ja toistuvia. Mikäli todetaan, että ominaisuus on usein käytetty sivustossa, mutta sen tekeminen on vaivatonta tai sen ulkoasu vaihtelee todella paljon, on syytä miettiä, kannattaako se lisätä kehykseen. Vaihtoehtoina on tehdä ominaisuuden pohja, jota on helppo kehittää oman näköiseksi tai jättää ominaisuus kokonaan pois, koska se on vaivatonta tehdä aina uudestaan.

Nykyään jokaisen hyvän verkkosivun vaatimuksiin kuuluu sivuston responsiivisuus eli se, miltä sivusto näyttää erikokoisilla näytöillä. Responsiivisuuden tekeminen on melko työläs taakka, mikäli sen kirjoittaa jokaisessa projektissa aina uudelleen. Valikkojen tekeminen on myös työläs tehdä, mikäli sen haluaa responsiiviseksi ja puhtaasti tyylejä käyttäen.

Valikkojen tekemisessä pitää ottaa huomioon, että sivustoissa käytetään paljon eri valikkoja, joten valikon pitää olla helposti muokattavissa. Valikoita tehdessä tulee ottaa huomioon, minkälaisia valikoita haluaa mukaan. Tähän työhön valikoiden suhteen tulee hieman haasteita, sillä valikot on tarkoitus tehdä puhtaasti tyylikielellä.

Tässä työssä tehtävän kehyksen toinen tarkoitus on luoda Parallax-vieritysominaisuus sivustoon, jolloin on hyvä tehdä tämän ominaisuuden pohja suoraan kehykseen. Parallax-ominaisuuden tekeminen vaatii sivustolta myös oikeantyyppistä rakennetta HTML-koodin puolella. Dokumentaatioon pitää tehdä tarkat määrittelyt rakenteen osalta, jotta ominaisuus on toimiva. Kehyksen Parallax-ominaisuutta ei tehdä vielä kokonaisuudessaan kehykseen, sillä useiden osien teko on liian monimutkaista tai mahdotonta vielä tyylikoodilla.

Kehyksen testaaminen on yksi tärkeimmistä kohdista projektissa. Testausta tulee tehdä jatkuvasti eri päätelaitteilla ja eri tavoilla. Testaamiseen olisi hyvä ottaa mukaan aina ulkopuolinen taho, joka ei ole tutustunut kehykseen aiemmin. Kehittäjän tekemiä testejä pitäisi tehdä joka kohdassa ominaisuuksien valmistuessa.

## 4.2 Tyylikehyksen responsiivisuuden määrittäminen

Tyylikehyksen responsiivisuuden tekeminen on laajennettu versio Bootstrap-tyylikehyksen oletusversiosta. Responsiivisuuden määrittely tehdään taulukkotyyppisesti, jossa sisältö jaotellaan eri soluihin. Solujen kokoa voidaan määritellä näytön koon mukaan. Näytön koot on jaettu neljään tyyppiin: suuret näytöt, tietokoneen näytöt, tablettikokoiset näytöt ja puhelinnäytöt. Puhelinnäytöt on määritelty niin, että näytön maksimileveys on alle 767 pikseliä. Tablettinäyttöjen määrittely on luotu jatkumaan puhelinnäytöistä, eli näytön minimileveys on 768 pikseliä ja maksimileveys on 992 pikseliä. Tietokonenäytöt ovat vuorostaan jatkoa tablettinäytöille. Tietokoneen näytöt siirtyvät tablettinäytön maksimista aina 1 200 pikseliin asti, minkä jälkeen näytön koko noudattaa suurien näyttöjen mallia. Suurille näytöille ei ole maksimia, sillä se on viimeinen ja suurin kokoluokittelu. Koodiesimerkki 1 näyttää, miten tablettinäyttökoko on tehty tyylikehykseen. Näytön koko määritellään ensimmäisellä rivillä niin sanotulla media querylä eli mediakyselyllä. Näytön leveyden pitää olla vähintään 768 pikseliä, jotta nämä kirjoitetut säännöt pätevät.

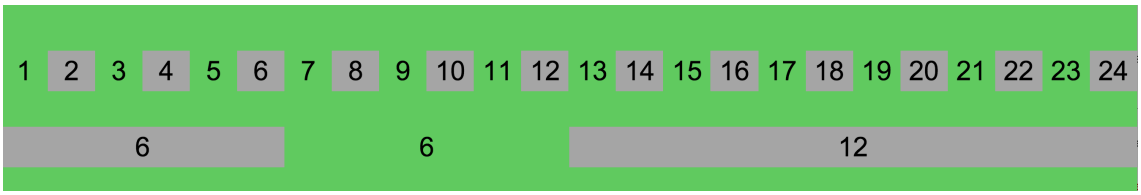
```
@media (min-width:768px){
```

```
.koko-sm-1,.koko-sm-2,.koko-sm-3,.koko-sm-4,
.koko-sm-5,.koko-sm-6,.koko-sm-7,.koko-sm-8,
.koko-sm-9,.koko-sm-10,.koko-sm-11,.koko-sm-12,
.koko-sm-13,.koko-sm-14,.koko-sm-15,.koko-sm-16,
.koko-sm-17,.koko-sm-18,.koko-sm-19,.koko-sm-20,
.koko-sm-21,.koko-sm-22,.koko-sm-23,.koko-sm-24{
float:left
}
.koko-sm-24{width:100%}
.koko-sm-23{width:95.8333333%}
.koko-sm-22{width:91.66666667%}
.koko-sm-21{width:87.5%}
.koko-sm-20{width:83.33333333%}
.koko-sm-19{width:79.1666667%}
.koko-sm-18{width:75%}
.koko-sm-17{width:70.8333333%}
.koko-sm-16{width:66.66666667%}
.koko-sm-15{width:62.5%}
.koko-sm-14{width:58.33333333%}
.koko-sm-13{width:54.1666667%}
.koko-sm-12{width:50%}
.koko-sm-11{width:45.8333333%}
.koko-sm-10{width:41.66666667%}
.koko-sm-9{width:37.5%}
.koko-sm-8{width:33.33333333%}
.koko-sm-7{width:29.1666667%}
.koko-sm-6{width:25%}
.koko-sm-5{width:20.8333333%}
.koko-sm-4{width:16.66666667%}
.koko-sm-3{width:12.5%}
.koko-sm-2{width:8.33333333%}
.koko-sm-1{width:4.16666667%}
```

Koodiesimerkki 1. Tyylikehyksessä määritelty tablettinäytön rakenne.

Solut voidaan määrittää enintään 24 osaan näytön koko leveydestä. Erikokoisissa näytöissä voidaan näyttää eri määrä soluja. Mikäli solukokoa halutaan suurentaa näyttökoon pienentyessä, ylimääräiset solut siirtyvät seuraavalle riville. Ylimääräiset solut voidaan myös ottaa pois eli piilottaa kokonaan tai tuoda esille erikokoisissa näytöissä. Soluja voidaan poistaa tai tuoda esille myös riveittäin samalla tavalla kuin koodiesimerkissä 1. Soluja on myös mahdollista vaihtaa keskenään, mikäli halutaan näyttää esimerkiksi eri kuva eri näyttöko'oissa. Solujen korvaaminen on hyvä tapa tehdä esimerkiksi valikkoja. Valikot voi olla tehty yhdelle täysimittaiselle riville, mutta pienemmissä näyttöko'oissa voidaan piilottaa alkuperäinen valikko kokonaan ja tehdä pienemmille näytöille sopiva valikko. Kuvassa 3 näytetään, miten soluja voidaan jakaa riville. Ensimmäisellä rivillä on tehty kaksikymmentäneljä samankokoista laatikkoa vierekkäin. Toisella rivillä on tehty kolme laatikkoa, ja näiden laatikoiden koko on määritelty niin,

että ensimmäiset kaksi laatikkoa ovat kuuden yksittäisen solun kokoisia. Viimeinen laatikko on puolet sivun maksimista eli kahdentoista solun kokoinen laatikko.




Kuva 3. Solujärjestelmä.

Solut määritellään HTML-koodin luokkamäärittelyssä. Luokkakohdissa näkyy, miten solu näytetään erikokoisissa näytöissä. Ensimmäinen määrittely ”koko-lg-1” tarkoittaa sitä, että solu näytetään suurissa näytöissä 4.1666667-prosenttisena koko näytön leveydestä. Tässä esimerkissä solujen koko ei muutu eri näyttöko’oissa. Viimeisissä riveissä on esimerkki suuremmista soluista. Näissäkin riveissä on sama koko kaikilla eri näytöillä, mutta ensimmäisen ja toisen solun leveys on 25 prosenttia, kun viimeisen solun leveys on 50 prosenttia. [Koodiesimerkki 2.]

```
<div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >1</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >2</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >3</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >4</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >5</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >6</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >7</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >8</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >9</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >10</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >11</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >12</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >13</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >14</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >15</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >16</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >17</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >18</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >19</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >20</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >21</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >22</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >23</div>
        <div class="koko-lg-1 koko-md-1 koko-sm-1 koko-xs-1" >24</div>
    </div>

    <div>
      <div class="koko-lg-6 koko-md-6 koko-sm-6 koko-xs-6" >6</div>
      <div class="koko-lg-6 koko-md-6 koko-sm-6 koko-xs-6" >6</div>
      <div class="koko-lg-12 koko-md-12 koko-sm-12 koko-xs-12">12</div>
    </div>
```

Koodiesimerkki 2. Kuvan 3 solujärjestelmän taustalla oleva koodi.

Taulukkojärjestelmä tuo myös mahdollisuuden sijoitteluun. Taulukoissa voidaan määrittää solujen tarkka paikka. Mikäli haluaa vain yhden tietynkokoisen keskitetyn osion sivulle, se onnistuu solujen ohittamisella. Solujen ohittaminen tehdään samalla tavalla kuin itse solujen koon määrittäminen. Sen sijaan, että tehtäisiin tyhjä solu ensin ja sen jälkeen vasta itse haluttu sisältö, voidaan halutun sisällön luokkiin lisätä määritys ”offset”. Koodiesimerkissä 3 on tehty keskitetty solu, joka on puolen sivun kokoinen ja se on keskitetty käyttäen ”offset”-ominaisuutta.

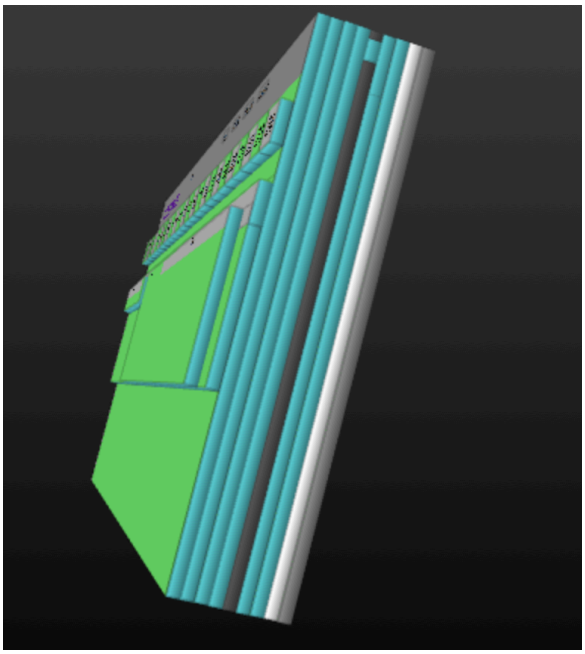
```
<div class="koko-sm-offset-6 koko-sm-12" >Puolen sivun kokoinen pala keskitettynä</div>
```

Koodiesimerkki 3. Malli tablettinäytöille tehdyn keskitetyn solun koodista.

### 4.3 Puhtaan CSS Parallax -vieritysominaisuuden liittäminen

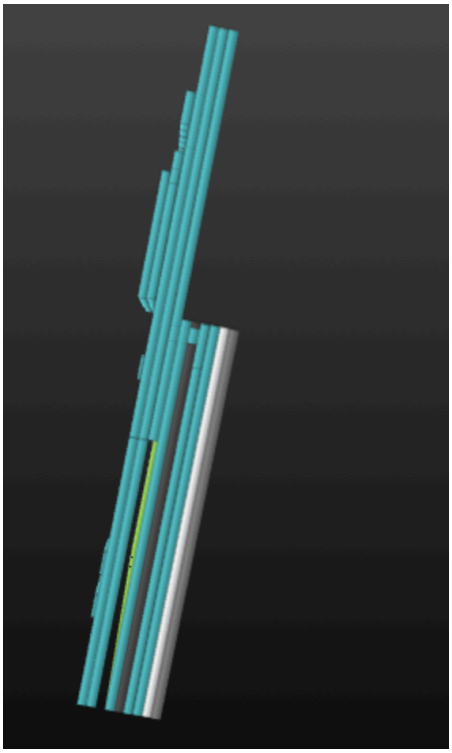
Parallax-vieritysominaisuus on sivuston rakennetyyli, joka ohjaa sivuston vieritykseen liittyviä tapahtumia. Tämän ominaisuuden tueksi tulee dokumentoida myös HTML-rakenne, sillä sitä ei voi liittää sivustoon ilman oikeanlaista rakennetta HTML:n puolella. Parallax –vieritysominaisuus on usein tehty käyttämällä Javascript-kieltä, joka kuormittaa sivuston lataamista huomattavasti. Tämän takia tavoitteena oli käyttää puhdasta CSS-kieltä ominaisuuden tekoon. Ominaisuudessa on käytännössä datakerroksia, jotka menevät tietyssä järjestyksessä päällekkäin, ja vierittäessä sivua taustalla oleva data rullautuu eri vauhtia kuin päällimmäiset kerrokset. Kerroksia ovat pohja-, väli- ja yläkerros. Yleensä pohjimmaisessa kerroksessa on kuvia, jotka vaihtuvat samaan aikaan, kun väli- tai yläkerros menee pohjakerroksen yli. Välikerroksessa on yleisesti itse tekstidata ja yläkerroksessa on joko kuvia tai teksti dataa. Kokonaisuudessaan kaikkien kerrosten tulee olla yhden koodiosion sisällä, jotta kerrostus menee oikein ja ominaisuus toimii. Haasteena ominaisuuden liittämisessä on tehdä dokumentaatioon selkeä selostus tapahtumista.

Kuvassa 4 nähdään ominaisuuden datakerrokset lähtötilanteessa. Kerrokset ovat päällekkäin, joten mikäli kaikki kerrokset olisivat osittain läpinäkyviä, koko sivuston data näkyisi samanaikaisesti.



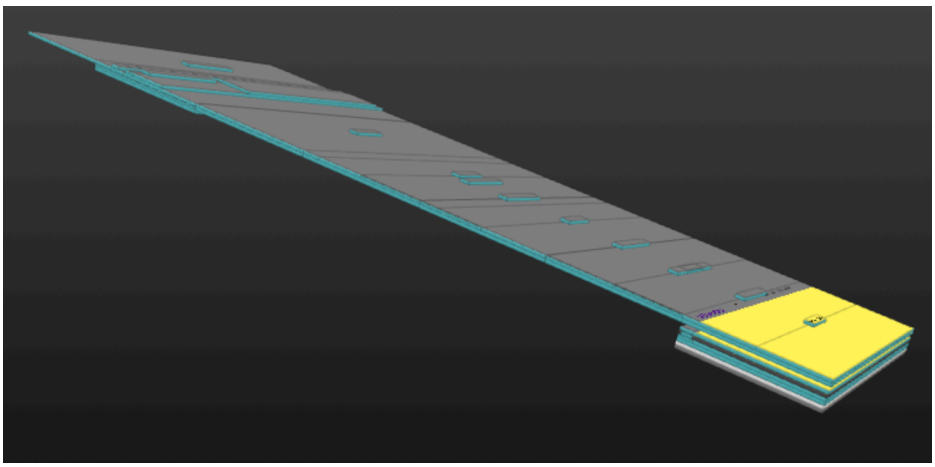
Kuva 4. Parallax-vieritysominaisuuden lähtötilanne datakerroksissa.

Kuvassa 5 näytetään ominaisuuden ensimmäinen siirtymä. Siirtymässä päällimmäisen kerroksen data jää paikoilleen ja loppukerrokset siirtyvät alaspäin. Käytännössä datakerrokset jättävät jälkeensä ohuen polun osioiden dataa.



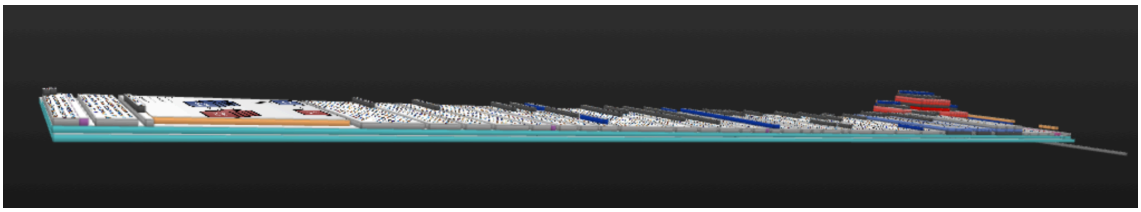
Kuva 5. Parallax-vieritysominaisuuden ensimmäinen vierityssiirtymä datakerroksissa.

Rullauksen lopussa (kuva 6) sivuston rakennemalli näyttää lähes samalta kuin tavanomaisesti tehty sivusto, joka on rakennettu peräkkäin.



Kuva 6. Parallax-vieritysominaisuuden lopputilanne datarakenteessa.

Kuvassa 7 näkyy tavalliseen tyyliin rakennetun verkkosivuston datarakenne, joka vastaa kuvaa 6.



Kuva 7. Tavanomaisen sivustorakenteen malli [19].

Datakerroksien kuvat 4–7 on otettu Mozilla Firefox -selaimen 3D view -ominaisuudella.

## 4.4 CSS-valikkojen tekeminen

Valikot ovat tärkeä osa jokaista verkkosivustoa, ja niiden tekeminen on tyylillisesti työlästä. Valikkoihin kuuluvat linkit ja listaukset, jotka ovat ennalta määriteltyjä tyylejä täynnä. Valikot voidaan muodostaa eri tyyleillä. Mikäli kyseessä on ”pöytäkoneelle” tarkoitettu valikko, se voidaan tehdä vaivatta eikä siihen tarvita kehykseen erikseen määriteltyä osiota. Nykyään kuitenkin valikot ovat myös responsiivisia, joten niiden tulee mukautua myös puhelinkokoon. Puhelimen näytöllä sivuston päävalikko voi olla aivan liian pieni, jotta siitä saisi mitään selvää. Puhelimelle pitää määrittää erikseen valikko, joka on kuitenkin samojen sääntöjen mukainen kuin päävalikko. Useasti puhelimen näytöille tarkoitetut valikot tehdään yhdeksi napiksi, josta avautuu pudotusvalikko päävalikon materiaaleista.

Valikot voidaan tehdä Javascript-kieltä käyttäen, mutta tämä kuormittaa sivustoa turhaan. Mikäli sivuston ulkoasusta päättävä henkilö ei vaadi mitään kummallisia valikoita, pystyy valikot tekemään melko varmasti käyttäen ainoastaan CSS-koodia.

Valikkoihin pitää määrittää muun muassa tekstin koko ja fontti, linkkeihin viittaavat tyylit, valikon kokonaiskuvan tyylit sekä responsiivisuus.

- Tekstin tyyleihin pitää määrittää fonttikoko jokaiselle eri näyttökoolle. Itse fontti tulee määrittää sellaiseksi, että teksti on helposti luettavissa, ja myös tekstin väri pitää miettiä tarkkaan. Väri on tärkeä osio tekstissä, sillä sen pitää erottua selkeästi valikon päätyylistä, jotta tekstin pystyy havaitsemaan helposti.

- Linkkityyleissä on monia eri määrityksiä, jotka on tärkeää ottaa huomioon. Linkkiominaisuuksiin kuuluu linkkien väri ennen kuin linkkiä on painettu, väri kun linkkiä on painettu, linkin tapahtumat kun hiiren osoitin on linkin päällä ja pois päältä.

- Valikon ulkoasun tyylit ovat valikkojen sielu. Valikoissa voidaan määrittää esimerkiksi värejä, taustoja, reunuksien muotoja ja varjostuksia.

- Responsiivisuus on osana jokaista kohtaa sivustossa, mutta valikoissa mukautuminen tarkoittaa useasti koko valikon koostumuksen vaihtumista. Tekstit saattavat mennä vaikka napin taakse. Tämän takia on hyvä tehdä tyylikehykseen omat responsiivisuussäädöt omille valikoille valmiiksi.

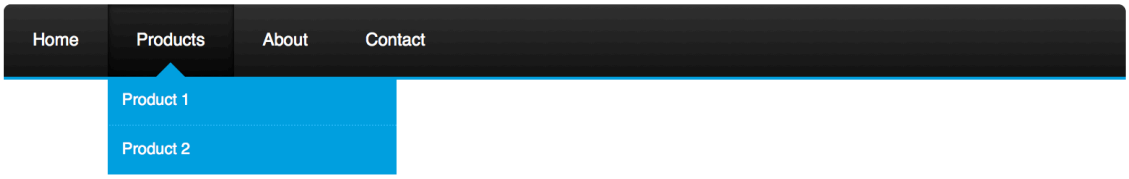
Kaikkien valikkojen ominaisuuksien pitäisi keskustella toisiensa kanssa, jotta valikon tulos on mahdollisimman hyvä. Suurin haaste valikkoja tehdessä pelkästään tyylikoodilla on responsiivisuuden määrittäminen ja se, miten pudotusvalikot tehdään. Pudotusvalikot voidaan tehdä monella eri tavalla, kuten käyttämällä ”select”-tunnistetta. Tätä tunnistetta käyttäessä valikosta tulee kumminkin hieman omituisen oloinen. Toinen vaihtoehto oli tehdä valikkoon pudotusvalikko linkin ”hover”-ominaisuutta käyttäen. Ominaisuus toimii silloin, kun käytetään hiiren osoitinta, eli puhelinnäytöillä se ei kelpaa, sillä kosketusnäytöillä painetaan sormella eikä hiiren osoitinta ole. Ominaisuus on silti hyvin käyttökelpoinen ja miellyttävä käyttää suuremmilla näytöillä, joten isojen näyttöjen puolella se on oiva valinta.

Tässä työssä tehty valikko tehtiin ”hover”-ominaisuutta käyttäen. Valikossa on erillisiä listoja, joista pudotusvalikkoon kuuluvat listat ovat piilotettuina. Piilotetut listat tulevat esille vasta, kun hiiren osoitin tulee listan päälle. Piilotus ja esille tuonti on tehty koodilla ”display: none;” ja ”display: block;”. [Koodiesimerkki 4 ja kuva 8.]

```
#cssmenu .has-sub {
  z-index: 1;
}
#cssmenu .has-sub:hover > ul {
  display: block;
}
#cssmenu .has-sub ul {
  display: none;
  position: absolute;
  width: 200px;
  top: 100%;
  left: 0;
```

```
}
```

Koodiesimerkki 4. Valikon pudotusominaisuuden piilotus ja esille tuonti.




Kuva 8. Valikko tehtiin CSS-kielellä, johon tehtiin pudotusvalikot.

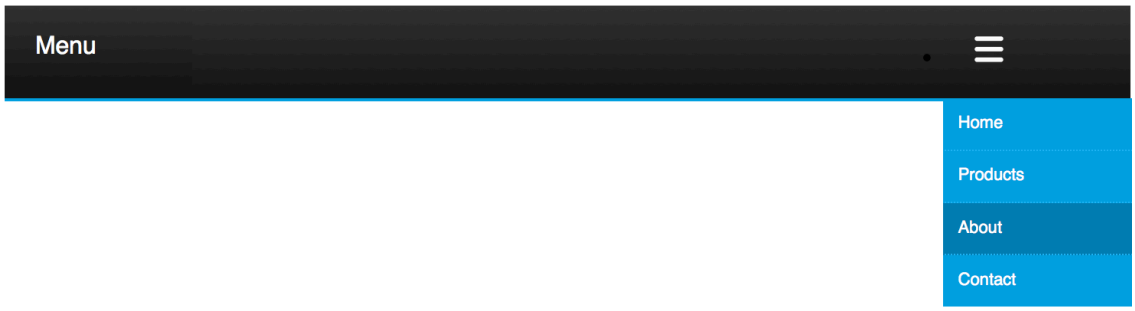
Pienemmille näytöille tehtiin responsiivinen vaihtoehto, jossa pudotusvalikko toimii samalla tavalla kuin isolla näytöllä. Tässä esimerkissä tehtiin kaksi kokonaan erillistä valikkoa, jotka tulevat esille tietyillä näytön ko'oilla. Kuvan 8 näköinen näyttö tulee esille, kun näytön leveys on yli 768 pikseliä. Näytön leveyden ollessa alle 768 pikseliä tulee esille pienempi valikko. Valikkojen näkyvyyden vaihto tehdään käyttäen luokkaa "hidden", jossa määrätään osion piilotus. Piilotus tapahtuu näytön kokoluokkien mukaan. Ensimmäisellä rivillä esimerkkikoodissa 5 on viimeisenä "hidden-xs", jossa piilotetaan koko solu pienillä näytöillä. Tämä sama tehdään mutta toisinpäin myöhemmin toisen valikon ensimmäisellä rivillä. [Kuva 9.]

```
<div id='cssmenu' class="koko-md-24 koko-sm-24 koko-lg-24 hidden-xs">
  <ul>
    <li><a href='#'><span>Home</span></a></li>
    <li class='active has-sub'><a href='#'><span>Products</span></a>
      <ul>
        <li class='has-sub'><a href='#'><span>Product 1</span></a>
        </li>
        <li class='has-sub'><a href='#'><span>Product 2</span></a></li>
      </ul>
    </li>
    <li><a href='#'><span>About</span></a></li>
    <li class='last'><a href='#'><span>Contact</span></a></li>
  </ul>
</div>

<div id='cssmenu' class="koko-xs-24 hidden-md hidden-lg hidden-sm">
  <div class="koko-xs-4">
    <li><a href='#'><span>Menu</span></a></li>
  </div>
  <div class="koko-xs-4 koko-xs-offset-16">
    <li class='active has-sub'><a href='#'><img style="width: 20px;" src="menu.png"/></a>
      <ul>
        <li class='active has-sub'><a href='#'><span>Home</span></a>
        <li class='has-sub'><a href='#'><span>Products</span></a>
```

```
            <ul>
              <li><a href='#'><span>Sub Product</span></a></li>
              <li class='last'><a href='#'><span>Sub Product</span></a></li>
            </ul>
          </li>
          <li class='active has-sub'><a href='#'><span>About</span></a></li>
          <li class='active has-sub'><a href='#'><span>Contact</span></a></li>
        </ul>
      </li>
    </div>
</div>
```

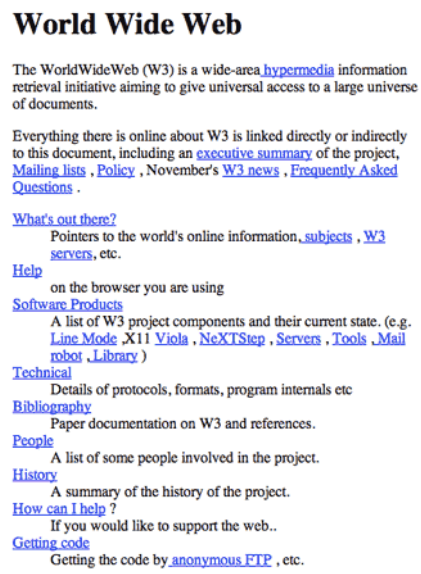
Koodiesimerkki 5. Isojen ja pienien näyttöjen erilliset koodit ja niiden responsiivisuus.




Kuva 9. Ote puhelimenkokoiselta näytöltä. Valikko ei mahdu kunnolla puhelimen näytölle, jolloin se siirretään napin taakse pudotusvalikoksi.

## 4.5 CSS reset -tiedosto

CSS reset on yleisesti kompressoitu CSS-tiedosto, jolla poistetaan tyylitiedoston vakiotyylit. Eri selaimilla on erilaiset tyylitunnisteiden vakiot, ja sen takia on erittäin hyödyllistä ottaa vakiot pois, jotta kaikilla selaimilla sivusto näyttää samalta. Vakiotyylejä on monissa tunnisteissa. Esimerkiksi linkit ovat usein vakiona sinisenvärisiä ja käytetty linkki on liilanvärinen. DIV-tunnisteella on myös vakiona tietty määrä tilaa ympärillä, riippuen siitä, mitä selainta käyttää. CSS reset poistaa kaikki vakiot, jotta tyylit voidaan itse kirjoittaa eikä mitään piilossa olevia vakiotyylejä tule voimaan. [Kuvat 10 ja 11.]



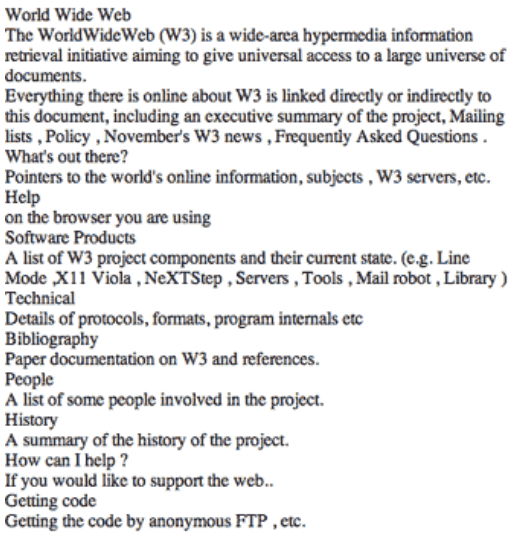
**World Wide Web**

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library )
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Kuva 10. HTML-koodia, jossa ei ole poistettu vakiotyylejä [20].



World Wide Web
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.
Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .
What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library )
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Kuva 11. Sama HTML-koodi kuin kuvassa 10, mutta on tehty CSS Reset, joka poistaa HTML-koodin tyylivakiot [20].

Vakiotyylien poistotiedostoja on monia, ja niistä yleisimpiä ovat olleet vuonna 2016 Eric Meyer's "Reset CSS" 2.0, HTML5 Doctor CSS Reset ja Yahoo! (YUI 3) Reset CSS. [20; 21.]

## 4.6 Vielä tekemättömät osiot

### Dokumentaation tekeminen

Insinöörityönä tehdyn kehyksen dokumentaatio tehdään, kun kehys on loppuvaiheessa ja testattu toimivaksi. Dokumentaatio sisältää kaikki kehyksessä olevat ominaisuudet, niiden käyttötarkoitukset sekä ominaisuuksien varsinaisen käytön. Ominaisuuksien käytön selventämiseen käytetään koodiesimerkkejä ja kuvia. Esimerkit ja kuvat tulevat esimerkkisivusta.

Ominaisuuksien dokumentoinnissa näytetään tyylitiedoston relevantti osio ja HTML-tiedoston koodi. Kummassakin osiossa selitetään, mitä koodi tekee ja mikä sen tarkoitus on. Tämän jälkeen lisätään tarvittaessa kuvat valmiista esimerkistä, miten kyseessä oleva esimerkki toimii. Kuvan sijasta voi dokumentaatiossa olla koodilla tehty osio. Esimerkiksi responsiivisuuden dokumentaatio-osio voidaan tehdä suoraan dokumentaation sisään. Esimerkissä voi olla kuvan 2 tyyppinen osio, joka muokkaantuu, kun vaihtaa ikkunan leveyttä.

### Kokonaisvaltainen testaus

Kehyksen testaaminen vaatii useita eri laitteita, jotta pystytään varmentamaan kaikkien eri osien toimivuus eri alustoilla ja eri selaimilla. Perustestaaminen tehtiin osio kerrallaan, mutta ei kaikilla päätelaitteilla. Lopullisen testauksen teko, ennen kuin kehys on valmis julkaistavaksi, on kuitenkin tekemättä, sillä testaus on työn viimeinen osio yhdessä dokumentaation kanssa. Kehyksen testaaminen vaatii kunnollisen testaussuunnitelman, oikeat työkalut ja muutaman ulkopuolisen testaajan, jotta saadaan kehykselle kunnollinen toimivuustesti. Testauksen jälkeen tulee lähes poikkeuksetta muutamia korjauskierroksia kehyksen osiin. Niitä voivat olla osien yhteensopivuus, kuten responsiivisuusosion yhteensopivuus valikkojen tai Parallax-vieritysominaisuuksien kanssa. Testaus jatkuu aina, kun uusi ominaisuus lisätään kehykseen, ja testausta olisi hyvä tehdä jatkuvasti, vaikka uusia osia ei tulisikaan.

### Työn julkaiseminen

Kehys julkaistaan, kun työn tähän asti suunnitellut osiot ovat valmiit. Kehys julkaistaan GitHub-palveluun ja verkkosivustolle. Verkkosivusto tulee sisältämään dokumentaation

ja itse työn esimerkkeineen. Verkkosivustoon on suunniteltu myös kommenttiosion tekemistä yleistä palautetta varten, jotta kehystä voitaisiin kehittää. Verkkosivuston siirto lokaalista tietokoneesta vaatii palvelintilan ja verkkosivusto-osoitteen eli ”domainin” hankkimista.

# 5 Työn tulokset

Insinöörityössä tehtiin toimiva tyylikehys verkkokehittäjille. Kehys sisältää muutamia yleisiä kehyksen ominaisuuksia, kuten responsiivisuus ja valikko. Kehyksen tekemiseen kuluu suuri määrä aikaa ja vaatii keskittymiskykyä, vaikka kehyksen arvo ja todellinen hyöty tulee ilmi vasta, kun kehittäjät alkavat käyttää sitä. Tällä hetkellä kehyksessä on vielä muutamia ongelmia, ja kaikkea ei ole testattu lopullisesti. Tämä ei silti tarkoita sitä, etteikö kehys olisi täysin toimiva. Tämänhetkinen kehys on helppokäyttöinen ja selkeä. Kehystä käyttävät kehittäjät säästävät sen käyttämisellä varmasti aikaa ja vaivaa.

## Tyylikehyksen tulevaisuus

Tyylikehys muokkautuu koko ajan, sitä mukaa kuin tulee uusia tarpeita. Kehys ei todennäköisesti tule valmiiksi koskaan, vaan sitä kehitetään jokaisessa projektissa. Kehystä saatetaan myös jatkaa tulevaisuudessa Javascript-kieltä hyödyntäen. Javascript-kielellä on tehty paljon lisäosia kehyksiin, mutta toistaiseksi tässä työssä pyritään pärjäämään ainoastaan CSS-kielellä. Erityisesti valikkojen ja Parallax-vieritysominaisuuden parantamista tullaan tekemään, jotta niistä saataisiin entistä toimivammat ja miellyttävät käyttää.

## Projektin onnistuminen

Projektiin jäi vielä paljon tekemistä, ja täyttä varmuutta ei ole siitä, kuinka kannattavaa on vielä tällaista työtä kehittää. Varmuus tulee varmasti vasta tulevaisuuden puolella. Työssä oppi todella paljon tyylikehyksen tekemiseen liittyvistä vaiheista ja tavasta ajatella. Työtä oli vaikea tehdä, sillä opastusta työn tekoon oli vähän ja työ oli ensimmäinen laatuaan. Aikaa työhön käytettiin paljon, ja hyödyt nähdään vasta tulevaisuudessa. Työn aikana tuli monta eri vaihetta, jotka olisi voinut jo huomata suunnitteluvaiheessa.

Kokemattomuus näkyi siis suunnittelussa, mutta jatkoa varten on erittäin tärkeää, että tällaiset asiat havaittiin.

Ongelmakohdat

Työn suurin ongelma on Parallax-vieritysominaisuuden toimivuus. Ominaisuus toimii hyvin kaikilla testatuilla alustoilla paitsi Android-käyttöjärjestelmää käyttävillä puhelimilla. Ominaisuus testattiin Firefox-, Chrome-, Safari- ja Explorer-selaimilla. Lisäksi testit suoritettiin IOS- ja Android-käyttöjärjestelmällä. Rullauksessa kerrokset eivät toimi oikein Android-käyttöjärjestelmällä, vaan ne menevät allekkain staattisiksi. Tätä ongelmaa ei ole vielä ratkaistu, vaikka useita tunteja on ongelman äärellä kulunut.

Responsiivisuutta tehdessäni huomasin sivuston, joka tarjoaa juuri saman työn, minkä olin tehnyt manuaalisesti. Bootstrap-kehyksen lisäpalveluna pystyy muokkaamaan solujen kokonaismäärää. Työni alussa oli tarkoitus kaksinkertaistaa solumäärä kahdestatoista kahteenkymmeneenneljään, ja tämä työ oli tarkoitus tehdä alusta alkaen käsin. Juuri kun kaksinkertaistaminen oli saatu suoritettua, tuli vastaani tämä sivusto, joka olisi säästänyt aikaani huomattavasti. Toisaalta itse koodin muokkaaminen on myös opettanut tekemään asian itse, eikä vain antamaan valmiin osion tehdä sitä puolestani.

# 6 Yhteenveto

Insinöörityössä käsiteltiin HTML- ja CSS-merkintäkielien historiaa ja verkkosivukehysten historiaa. Historiaa tutkiessa huomattiin, että verkkosivutekniikoiden kehitys on sulautumassa hieman yhteen, mutta tuskin HTML-kielen syrjäyttäjää tulee lähitulevaisuudessa. Myös HTML-kielen kehitys jatkuu, sillä nykyään on jo niin paljon kehittäjiä, että itse kehittäminen ja tarvittavat korjaukset sujuvat nopeasti.

Verkkosivukehysten historiaa on vaikeata tulkita, sillä historia on melko lyhyt, mutta kehysten käyttämisen trendi on ylöspäin suuntautuva. Työssä käytiin myös läpi itse kehyksiä ja niiden ominaisuuksia. Varmasti tärkein ominaisuus kehyksille on responsiivisuus. Se on tärkeä, koska sitä on raskasta kirjoittaa aina uudelleen ja sitä on hankala testata kaikilla laitteilla. Tämän vaivan kun säästää kehittäjältä, se jo itsessään riittää syyksi käyttää kehystä. Projektissa otettiin kantaa siihen, kannattaako käyttää valmiita kehyksiä vai tehdä itse oma kehys. Kysymykseen pyrittiin vastaamaan puolueettomas-

ti, ja vastaus on monimutkainen. Valmiita kehyksiä kannattaa käyttää, mikäli tekee kehyksellä vain muutaman sivun eikä ole aikaa tehdä kokonaan omaa, tai jos ei jaksa nähdä vaivaa oman tekemiseen ja testaamiseen.

Tällaisessa vaiheessa projektista ei varmasti vielä ole hyötyä muille verkkokehittäjille. Vasta kun projektin yhteensopivuus kaikkien ominaisuuksien kanssa on tehty ja testattu, voidaan harkita muiden kehittäjien mukaan tuomista.

## Lähteet


1 Wilson Brian. Cascading Style Sheets, level1. Verkkodokumentti. <http://www.blooberry.com/indexdot/history/css1.htm>. Luettu 30.11.2015.

2 Wilson Brian. Cascading Style Sheets, level2. Verkkodokumentti. <http://www.blooberry.com/indexdot/history/css2.htm>. Luettu 30.11.2015.

3 Wilson Brian. CSS2.1. Verkkodokumentti. <http://www.blooberry.com/indexdot/history/css21.htm>. Luettu 30.11.2015.

4 Kyrnin Jennifer. 2015. What is the difference between CSS2 and CSS3. Verkkodokumentti. <http://webdesign.about.com/od/css3/a/differences-css2-css3.htm>. Luettu 1.12.2015.

5 HTML tags. Verkkodokumentti. W3. <https://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html>. Luettu 1.4.2016.

6 A brief history of HTML. Verkkodokumentti. University of Washington. <https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html>. Luettu 1.4.2016.

7 HTML Historia. Verkkodokumentti. Jyväskylän yliopiston tutkimuslaitos. <https://peda.net/konnevesi/ylakoulu/oppiaineet/tietoteknikka/html5/0-html-historia>. Luettu 1.4.2016.

8 Skyivben. Verkkodokumentti. Internet Archive. <http://web.archive.org/web/20080504160116/http://ben.skyiv.com/clrversion.html>. Luettu 4.4.2016.

9 Verkkosovelluksen nopeuden optimointi. 2016. Verkkodokumentti. 4D Software. <http://4dsoftware.fi/verkkosovelluksen-nopeuden-optimointi/>. Luettu 2.4.2016.

10 Rouse Margaret. 2015. Framework. Verkkodokumentti. < http://whatis.techtarget.com/definition/framework>. Luettu 11.1.2016.

11 What are Frameworks?. Verkkodokumentti. Awwwards-team. <http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html>. Luettu 10.1.2016.

12 Kay Seb. 2014. Front-End Framework or Rolling Your Own. Verkkodokumentti. <http://inspirationalpixels.com/articles/front-end-framework-or-rolling-your-own>. Luettu 26.2.2016.

13 Russell Martin. 8.2013. 5 CSS Frameworks you should be using. Verkkodokumentti. <http://cssmenumaker.com/blog/5-css-frameworks-you-should-be-using>. Luettu 20.2.2016.

14 Responsive CSS Framework Comparison. 2015. Verkkodokumentti. Vermilion. <http://responsive.vermilion.com/compare.php>. Luettu 20.2.2016.

15 Hakim James. A collection of the best front end frameworks. Verkkodokumentti. <http://cssframeworks.org/>. Luettu 4.4.2016.

16 Januska Antonin. 2013. Why it's a good idea to roll your own CSS Framework. Verkkodokumentti.< http://antjanus.com/blog/web-development-tutorials/front-end-development/why-its-a-good-idea-to-roll-your-own-css-framework/>. Luettu 22.2.2016.

17 Buttons. Verkkodokumentti. Google. <https://www.google.com/design/spec/components/buttons.html#buttons-flat-buttons>. Luettu 23.2.2016.

18 Harrop Jamie. 2016. An introduction to frontend frameworks. Verkkodokumentti. <http://makeawebsitehub.com/frontend-frameworks-intro/>. Luettu 2.4.2016.

19 Datarakenne. Verkkodokumentti. Software Engineering Institute. <http://www.sei.cmu.edu/productlines/frame_report/on_testing.htm>. Luettu 3.4.2016.

20 What is a CSS Reset? Verkkodokumentti. CSS Reset. <http://cssreset.com/what-is-a-css-reset/>. Luettu 2.3.2016.

21 2016's most popular CSS Reset scripts.Verkkodokumentti. CSS Reset. <http://cssreset.com/>. Luettu 2.3.2016