Haaga-Helia
ammattikorkeakoulu Oy

Grid Layout ja Flexible Box -moduulit CSS-ohjelmistokehyksissä

Tuomas Velling

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2016

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

18.5.2016

Tekijä(t)
Tuomas Velling

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Grid Layout ja Flexible Box -moduulit CSS-ohjelmistokehyksissä

Sivu- ja liitesivumäärä
41 + 4

Opinnäytetyön otsikko englanniksi
Grid Layout and Flexible Box modules in CSS frameworks

Responsiiviset verkkosivut ja CSS-ohjelmistokehykset ovat olleet 2010-luvun suurin trendi verkkosivustoilla. Niiden hyöty on ollut merkittävä verkkosivustojen kehityksen tehostamisessa sekä uusien ulkoasuratkaisujen luomisessa. Olemassa olevat ohjelmistokehykset ovat kuitenkin saaneet osakseen kritiikkiä muun muassa koodin paisumisen takia. Ne myös laajentavat vanhojen tekniikoiden käyttöä tarkoituksiin, joihin niitä ei ole alun perin suunniteltu.

Uudet Grid Layout ja Flexible Box -moduulit ovat HTML- ja CSS-kieliä standardoivan W3C-organisaation ehdotukset uusiksi rakennustekniikoiksi, joiden avulla voidaan ratkoa perinteisten ohjelmistokehysten ongelmia. Grid Layout mahdollistaa verkkosivustoilla kaksiulotteisen ruudukkorakenteen, josta on hyötyä erityisesti sivuston rungon toteutuksessa. Flexible Box on perinteisiä ruudukkoja muistuttava yksiulotteinen ruudukkorakenne, jolla voidaan toteuttaa sivuston sisäisiä elementtejä vaivattomasti.

Näiden tekniikoiden käyttöönottoa on hidastanut puutteellinen selaintuki, joka on kuitenkin jatkuvasti parantumassa. Flexible Box on jo käytössä yleisimmissä selaimissa ja Grid Layout voidaan kytkeä päälle kokeilukäyttöön Chrome ja Firefox -selaimissa.

Tässä työssä pureudutaan moduulien soveltumiseen osana responsiivista suunnittelua sekä näiden käyttöä CSS-ohjelmistokehyksissä. Teoriaosuudessa käydään läpi moduulien toiminnallisuudet havainnollistavin esimerkein. Produktiosuudessa haetaan parhaita käytäntöjä moduulien käyttämiseksi CSS-ohjelmistokehyksissä ja käsitellään näiden integroimista asiakasyritys G-Worksin omaan ohjelmistokehykseen.

Työssä löydettiin molemmille moduuleille omat käyttötarkoitukset ja näille luotiin käytännöt sekä Sass-apuvälineet. Löydetyt käytännöt ja ratkaisut ovat hyvin potentiaalisia ja mahdollistavat erilaisten elementtien toteutuksen helposti. Ratkaisut kuitenkin vaativat vielä asiakasyritykseltä työntekijöiden kattavaa kouluttamista sekä käytäntöjen kehittämistä oikeiden asiakasprojektien yhteydessä.

Asiasanat
Flexible Box Layout, Grid Layout, CSS-ohjelmistokehykset, HTML, CSS

Haaga-Helia
University of Applied Sciences

Abstract

18.5.2016

Author(s)
Tuomas Velling

Degree programme
Business Information Technology

Report/thesis title
Grid Layout and Flexible Box modules in CSS frameworks

Number of pages and appendix pages
41 + 4

Responsive web pages and CSS frameworks have been the web industry's biggest trend in the 2010s. Their benefits have been significant when it comes to making website development more efficient and enabling new layout possibilities. Existing frameworks have gained some amount of criticism for bloating HTML code and stretching existing CSS techniques beyond their original purpose.

Grid Layout and Flexible Box modules are the HTML and CSS language standardization organization W3C's proposals for new building techniques and are designed to fix problems in traditional CSS frameworks. Grid Layout enables the use of a two-dimensional grid structure, which helps in building overall page layouts. Flexible Box functions as a traditional one-dimensional grid and it can be used to create different internal components inside a page.

The introduction of these technologies has been slowed down by the lack of browser support. Fortunately, browser support is constantly improving. Flexible Box is already usable in all the major browsers but Grid Layout is still in development. Grid Layout can be enabled separately in the newest versions of the browsers Chrome and Firefox.

This thesis focuses on these new modules as a part of responsive web design and using them in CSS frameworks. The theory part goes through all the features and possibilities of both Grid Layout and Flexible box with the help of examples. The product part consists of developing the best practices for using these new modules and extending these practices into the existing CSS framework of digital agency G-Works.

In this thesis different use cases were defined for both modules, and several different practices and Sass helpers were developed. Found practices and solutions show huge potential and offer an easy way to create different page components and layouts. However, these solutions need to be thoroughly introduced to developers and tested in actual client cases.

Keywords
Flexible Box Layout, Grid Layout, CSS frameworks, HTML, CSS

Sisällys

1 Johdanto

Responsiivinen suunnittelu ja sen mukana tulleet responsiiviset CSS-ohjelmistokehykset ovat antaneet organisaatioille ja yksityisille ihmisille mahdollisuuden toteuttaa omat verkkosivunsa entistä selkeämpinä ja käyttäjäystävällisempinä. Monesti responsiivinen suunnittelu tuo myös kustannussäästöjä, kun verkkosivuista ei tarvitse toteuttaa erillisiä versioita jokaiselle laitteelle. (Whittington 2013.)

Graafiselle suunnittelijalle ja ohjelmoijalle CSS-ohjelmistokehykset tarjoavat toimiviksi todetut konseptit sekä ratkaisut verkkosivujen suunnitteluun. Tämän ansiosta verkkosivujen suunnittelussa ei tarvitse joka kerta keksiä pyörää uudestaan. (Awwwards 2013.)

Maaliskuussa 2016 suosituimpia CSS-ohjelmistokehyksiä oli käytössä yli 8 miljoonassa verkkosivussa ja trendi on jatkuvasti kasvamaan päin (BuiltWith 2016a). CSS-ohjelmistokehysten käytön yleistyminen on tuonut ne myös tutkimuksen piiriin. Suomalaisesta ammattikorkeakoulujen opinnäytetöiden tietokanta Theseuksesta löytyi oman hakuni perusteella useita kymmeniä erilaisiin CSS-ohjelmistokehyksiä käsitteleviä opinnäytetöitä, mutta Flexible Box ja Grid Layout -moduulit olivat aiheena vain muutamassa opinnäytetyössä.

Uudet moduulit ovat verkkosivujen ohjelmointikieliä standardoivan W3C-organisaation määrittelyitä uusiksi työkaluiksi verkkosivujen ulkoasun toteuttamiseksi. Molemmat moduulit olivat maaliskuussa 2016 vielä W3C standardivedoksia, mutta moni selainvalmistaja on saanut teknisen toteutuksen pitkälle selaimissa. (Andrew 2016a.)

Uusien ulkoasumoduulien tarkoituksena on mahdollistaa verkkosivujen ruudukkorakenteen ja yksittäisten elementtien helpompi asettelu sekä järjestäminen (Andrew 2014, 21; 41). Aikaisemmin näiden moduulien tilalla on käytetty muun muassa ”table” ja ”float” -tekniikoita, joita ei kuitenkaan alun perin suunniteltu tähän käyttöön. Näissä tekniikoissa onkin havaittu myöhemmin ongelmia ja uudet moduulit on kehitetty auttamaan näissä tilanteissa (Walton 2016a).

Jo vuonna 2011 Peter Gasston ennusti, että silloin W3C-vedoksena olleet moduulit tulisivat laajempaan käyttöön vuonna 2016. Myös moduuleja kehittävä Tab Atkins Jr. arvioi vuonna 2014 CSS Day -konferenssissa, että vanhat ruudukkorakenteet ja niitä käyttävät ohjelmistokehykset poistuvat käytöstä, kun tuki Grid Layout -moduulille tulee tarpeeksi kattavaksi.

1.1 Toimeksianto ja tavoitteet

Tässä opinnäytetyössä käsitellään Flexible Box ja Grid Layout -moduuleja ja niiden soveltumista responsiivisten verkkosivujen suunnitteluun. Opinnäytetyön produktin toimeksianto tuli helsinkiläiseltä digimainostoimisto G-Worksilta. Toimeksiantajan yksi liiketoiminnan kohteista on responsiivisten verkkosivujen suunnittelu ja toteutus asiakasyritystensä käyttöön.

Keväällä 2016 G-Worksissa työskentelee yksitoista henkilöä, joista viisi on ohjelmistosuunnittelijoita, kolme graafista suunnittelijaa, kaksi projektipäällikköä ja yksi palvelinasiantuntija. Tyypillisiä asiakkaita ovat keskisuuret yritykset ja kolmannen sektorin toimijat pääkaupunkiseudulla. Yritys on pitänyt perustamisesta (2010) asti yllä luonnollista kasvua liiketoiminnassaan ja vuoden 2016 liikevaihtoennuste on noin 500 000 euroa. (Keto 1.4.2016.)

Opinnäytetyön produktin tavoitteeksi asetettiin luoda käytännöt uusien HTML-moduulien käytölle ja toteuttaa CSS-ohjelmistokehys näiden pohjalta. Uusien moduulien käyttäminen mahdollistaa erilaisia toteutustapoja asiakkaiden verkkosivustoilla sekä tehostaa verkkosivustojen kehitystä. Tällä tavoin saadaan mahdollisimman suuri tehokkuushyöty, kun käyttöönottoon menee vähemmän aikaa sekä toisen ohjelmistosuunnittelijan on vaivatonta ymmärtää ja kehittää edelleen toteutettua ohjelmistokoodia.

1.2 Tutkimusmenetelmät ja työn rakenne

Työn teoriaosuus toteutettiin kirjallisuustutkimuksena, jossa lähteinä käytetään pääasiassa alan ammattilaisten ja aiheena olevien moduulien kehittäjien artikkeleita. Haasteen muodosti aiheen tuoreus, jonka takia käytettyjä lähteiden luotettavuutta oli pakko arvioida henkilöiden tunnettuuden perusteella ja samalla hakea vahvistusta muilta ammattilaisilta.

Teoriaosuudessa lukujen kaksi ja kolme tarkoituksena on taustoittaa responsiivisen verkkosuunnittelun ja CSS-ohjelmistokehysten perusteita sekä historiaa. Luvuissa neljä ja viisi käydään läpi Grid Layout ja Flexible Box -moduulien toiminnallisuuksia ja eri käyttötapoja.

Moduulien osalta on käyty läpi kaikki oleelliset ominaisuudet teknisen standardin perusteella käyttäen samalla tukena ammattilaisten kirjoituksia aiheesta. Ominaisuudet on myös testattu ja testeistä on otettu kuvakaappaukset moduulien toiminnan havainnollistamiseksi.

Teoriaosuudessa kerätyn tiedon perusteella toteutusvaiheessa käytiin läpi mahdollisuudet käyttää uusia moduuleja yhdessä CSS-ohjelmistokehysten kanssa. Tämä toteutettiin tekemällä erilaisia kokeiluja luomalla alkeellisia sivustoja ja elementtejä (liitteet 1-2). Näitä kokeiluja arvioitiin käyttämällä perusteena ohjelmointityön tehokkuutta, koodin selkeyttä ja mahdollistettuja ulkoasuratkaisuja.

Näiden kokeilujen perusteella valittiin ratkaisut, joiden avulla lähdettiin toteuttamaan uudelleen erästä G-Worksin aikaisemmin toteuttamaa verkkosivustoa käyttäen uusia moduuleja (liitteet 3-4). Tämän vaiheen tarkoituksena oli testata valittuja ratkaisuja käytännön työssä ja kehittää niitä edelleen.

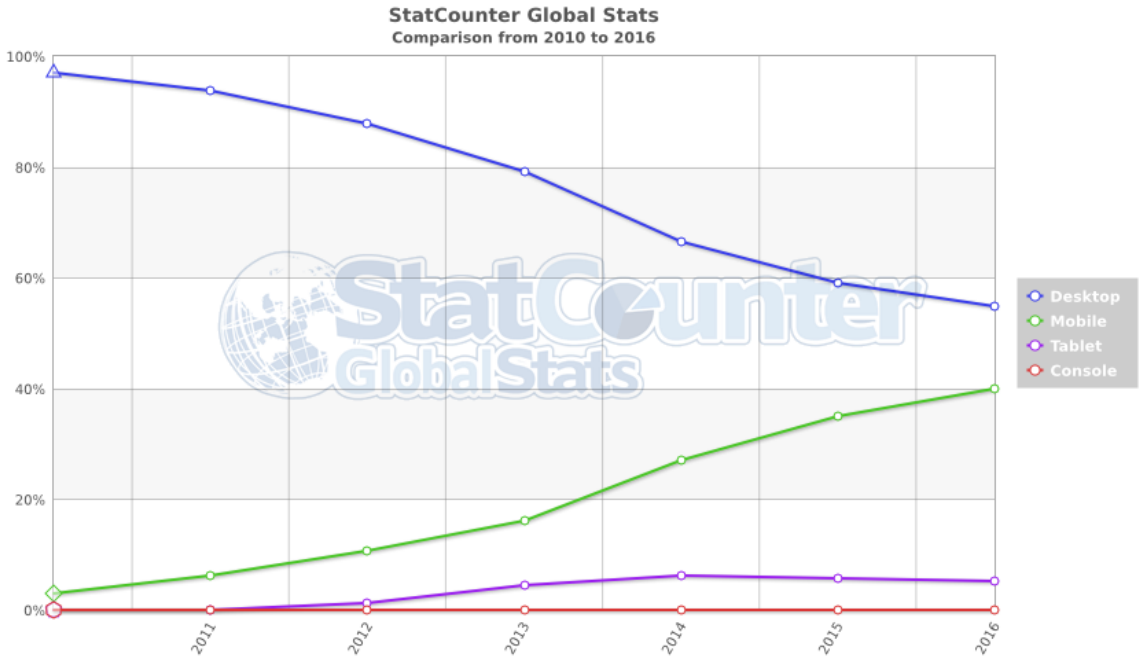
Lopuksi nämä parhaat ratkaisut kerättiin yhteen ja integroitiin osaksi yrityksen CSS-ohjelmistokehystä. Lopulliset ohjelmistoratkaisut on jätetty pois yrityssalaisuuksien suojaamisen vuoksi.

Työhön osallistui asiakasyrityksestä Art Director Roope Sandberg, jonka tehtävänä oli kommentoida toteutettuja ratkaisuja graafisen suunnittelun sekä frontend-ohjelmoinnin näkökulmasta.

2 Responsiivinen web-suunnittelu

Responsiivinen web-suunnittelu nousi web-kehittäjien keskuudessa isoksi trendiksi vuonna 2012. Responsiivinen suunnittelu tarjosi kehittäjille sekä suunnittelijoille työkalut saman verkkosivun näyttämiseksi erikokoisilla näytöillä, joka taas mahdollisti uudenlaisen ajattelutavan verkkosivujen teossa. (Johnson 2012.)

Tätä muutosta on vauhdittanut olennaisesti mobiili- ja tablettilaitteiden yleistyminen 2010-luvun aikana (kuva 1). Vuonna 2010 maailmalla lähes kaikki verkkosivujen käyttö tapahtui perinteisillä koneilla, mutta vuoteen 2016 mennessä näiden osuus on tippunut lähes 50 prosenttiin. Käyttö on siirtynyt erityisesti älypuhelimiin (39%), mutta myös tabletit (5%) tekevät nousuaan. (StatCounter 2016.)



Kuva 1. Verkkosivustojen käyttö maailmanlaajuisesti eri laitteilla vuosina 2010-2016 (StatCounter 2016)

Tapoja määritellä responsiivinen web-suunnittelu on lähes yhtä paljon kuin on web-suunnittelijoita. Termin isä, Ethan Marcotte, määritteli responsiiviselle web-suunnittelulle kolme teknistä vaatimusta artikkelissaan A List Apart -sivustolla vuonna 2010:

- Joustava ruudukko (fluid grid)
- Media query CSS-määritysten käyttö
- Venyvät kuvat (flexible images)

Joustavalla ruudukolla tarkoitetaan ruudun koon perusteella mukautuvaa ruudukkorakennetta sivuston pohjalla. Media query -määrittelyt mahdollistavat elementtien asettelujen ja CSS-tyylimäärittelyjen tekemisen ruudun koon perusteella ja venyvien kuvien koko skaalautuu kuvasuhteet säilyttäen ylätason elementin mukaan. Responsiivisen suunnittelun tärkein ominaisuus onkin mahdollistaa verkkosivujen kehitys laitteen näytöstä tai sen koosta riippumatta (kuva 2). (Marcotte 2010.)

Kuva 2. Responsiivisen verkkosivuston elementit mukautuvat ruudun koon mukaisesti

Marcotte lähti määritellessään responsiivista verkkosuunnittelua siitä ajatuksesta, että verkkosivuja ei tulisi suunnitella jokaiselle laitteelle erikseen vaan yhden ulkoasun tulisi mukautua näytön koon mukaan. (Kizler 2013; Marcotte 2010.)

Verkkosivujen historia alkoi 90-luvulta ja tästä yli kaksi vuosikymmentä sivustot suunniteltiin pääasiallisesti kiinteäkokoisen ulkoasun pohjalle. Vaikka kiinteään kokoon suunnitellut sivustot ovat olleet perinteisesti helpompia suunnitella sekä toteuttaa, on niiden suurimmaksi ongelmaksi koettu juuri laite- ja näyttömukautuvuus. (Kizler 2013; Soojian 2015.)

Marcotten vallankumouksellinen artikkeli koostui hänen omasta tutkimuksestaan näiden kolmen eri tekniikan parista. Vaikka nykyään Marcotten määrittelemä responsiivinen suunnittelu on verkkosivustojen suunnittelussa kuuma trendi, ovat monet kyseenalaistaneet näiden kolmen vaatimuksen riittävyyden. (Zeldman 2011.)

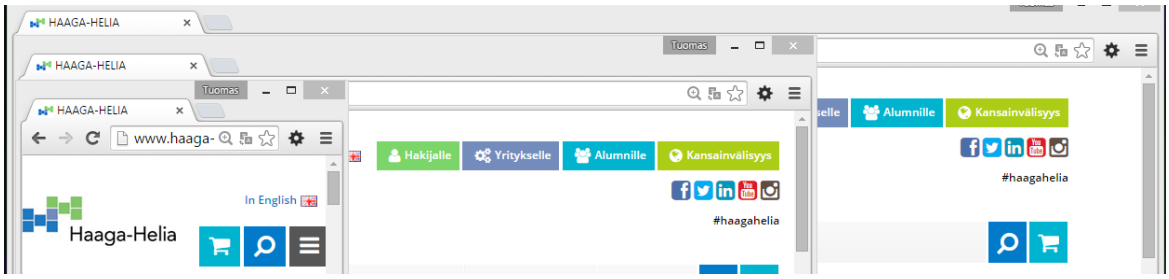
Nykyisin uudet tekniikat ja teknologiat ovat mahdollistaneet käyttäjäkokemuksen ja esteettömyyden viemisen uudelle tasolle. Marcotte myönsi saman asian keskustelussaan Jason Grigsbyn kanssa. Hän totesi responsiivisen verkkosuunnittelun alkavan olla vanhentunut

termi ja meidän pitäisi puhua siitä, minkälainen verkko haluaa olla: joustava, kohdistettava ja kaikkien tavoitettavissa. (Grigsby 2014.)

2.1 Laiteriippumaton suunnittelu

Samoihin aikoihin responsiivisen suunnittelun kanssa käynnistyi keskustelu laiteriippumattomasta suunnittelusta. PC Magazine -lehden verkkosanakirja (2016) määrittelee laiteriippumattomuuden seuraavalla tavalla: ”Ei ole sidottu tiettyyn laitteeseen. Viittaa yleensä ohjelmistoon, joka toimii useammalla alustalla, kuten esimerkiksi Java-ohjelmistot. Viittaa myös verkkosivuihin, jotka ovat luettavissa samalla tavalla mobiililaitteilla kuin myös pöytäkoneilla.”

Ethan Marcotte (2011, 2-5) toi esille kirjassaan responsiivisesta suunnittelusta, että verkkosivujen suunnittelu on taidemuotona vielä nuori ja suunnittelijat pyrkivät virheellisesti lainaamaan käytäntöjä ja malleja perinteisistä taiteenlajeista. Esimerkkinä hän mainitsi taiteilijan piirtoalustan, joka antaa teokselle rajat ja määrittää sen koon. Verkkosivuilla suunnittelijan piirtoalustana toimii selain, jonka tarkkaa kokoa ei voi rajoittaa tai määrittää (kuva 3).



Kuva 3. Selainikkuna toimii suunnittelijan piirtoalustana ja sen koko määrittää verkkosivujen rakennetta sekä ulkoasua

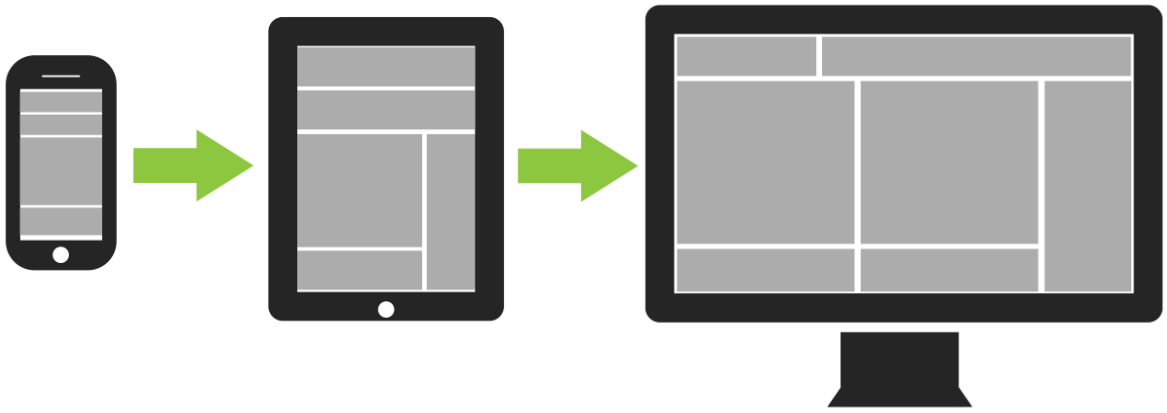
Laiteriippumaton suunnittelu perustuu pohjimmiltaan ajatukseen, että verkkosivujen suunnittelun keskipiste ei saa olla selaimessa tai laitteessa vaan sisällössä. Suunnittelun lähtökohtana tulee olla, että sivujen tulee toimia myös kaikkein vaativimmissa ympäristöissä. Trent Walton (2014) määritteli artikkelissaan ”Device-Agnostic” seuraavat määritteet vaativille ympäristöille: vihamieliset, eli epästandardit, selaimet, pienet ruudut, hitaat verkkoyhteydet sekä kosketusnäytöt.

Myös responsiivinen suunnittelu nojaa hyvin pitkälti Waltonin mainitsemiin periaatteisiin. Samalla se on tärkein nykypäivän suunnittelumenetelmistä, jotka mahdollistavat laiteriippumattoman suunnittelun. Laiteriippumattomuus ja responsiivisuus tarjoavat yhdessä toimivan pitkän aikavälin ratkaisun, kun sivustoa ei tarvitse suunnitella uudestaan jokaiselle laitteelle. (Ruska 2014.)

Laiteriippumattomuuden ideologia on saanut myös paljon kritiikkiä. Laiteriippumattomuuden ongelmaksi on koettu se, että jos käyttäjän laite ei vaikuta sivustoon, katoaa konteksti ja käyttöympäristö. Verkkosivuston pitäisi osata hyödyntää laitteen tarjoamat edut kuten esimerkiksi puhelintoiminto numeroa painamalla tai navigointi kohteeseen osoitetta painamalla. (Ruska 2014; Harbour 2013.)

2.2 Mobile First

Mobile First lasketaan useasti osaksi responsiivista suunnittelua. Mobile First -ajatusmallin mukaan verkkosivujen suunnittelu tulisi aloittaa pienimmästä mahdollisesta laitekoosta ja progressiivisesti edetä suurempaan (kuva 4). Tämän ansiosta verkkosivun ydintoiminnallisuuden voidaan taata toimivan varmasti myös mobiililaitteilla. Tämän jälkeen suunnittelussa tuodaan parannuksia ja uusia ominaisuuksia laitteiden koon ja suorituskyvyn kasvaessa. (Johnson 2013.)



Kuva 4. Mobile First -mallissa suunnittelu aloitetaan pienimmästä laitteesta ja edetään suurempaan

Mobile First nousi trendiksi, kun Googlen hallituksen puheenjohtaja Eric Schmidt totesi Mobile World Congressissa pitämässään esityksessä Googlen siirtävän huomionsa työpöytälaitteista mobiililaitteisiin. Samalla Schmidt kannusti kehittäjiä tekemään mobiilista uuden ympäristön verkon tulevaisuudelle. (Albanesius 2010.)

Mobile First tuodaan monesti esille kilpailevana ajatusmallina laiteriippumattoman suunnittelun kanssa. Käytäntö on kuitenkin osoittanut, että tilanne ei ole niin mustavalkoinen kuin teknologiset vastakkainasettelut antavat ymmärtää. Laiteriippumattomuus ja samalla laitteiden ominaisuuksien huomioiminen ovat asioita, jotka toimivat parhaimmillaan yhdessä. Tällöin verkkosivu voi tarjota juuri käyttäjän haluamia ominaisuuksia ja näkymiä. (Ruska 2014; Harbour 2013.)

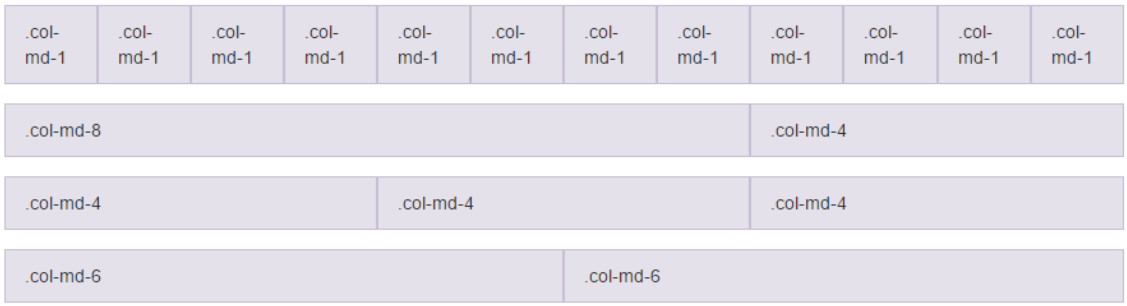
3 CSS-ohjelmistokehykset

CSS-ohjelmistokehysten tarkoituksena on tarjota valmis pohja ja työkalut verkkosivustojen toteutukseen. Useimmiten näiden tarjoamiin ominaisuuksiin kuuluvat ruudukkorakenne, valmiiksi tyyliteltyjä käyttöliittymäkomponentteja ja muita erilaisia toiminnallisuuksia. Etuina ohjelmistokehyksillä on erityisesti selainyhteensopivuus sekä saavutettavuus esimerkiksi lukulaitteilla. (Kramer 2014.)

Ohjelmistokehyksiä kohtaan on myös esitetty paljon kritiikkiä. Haittapuolina nähdään muun muassa niiden tuoma lisäkuormaa sivulatauksessa sekä HTML-koodin paisuminen turhilla elementeillä ja luokkamäärityksillä. Nämä ongelmat eivät kuitenkaan ole nykypäivänä suuria ja ohjelmistokehysten tuoma kehitysnopeus koetaan suuremmaksi eduksi. (Kramer 2014.)

CSS-ohjelmistokehykset alkoivat yleistyä viime vuosikymmenen vaihteessa, jolloin joustavaa ruudukkoa tai responsiivista suunnittelua ei oltu vielä kehitetty. Kuitenkin jo tällöin niiden tärkein ominaisuus oli tarjota kiinteän kokoinen ruudukkorakenne sivun pohjaksi. Tuolloin verkkosivuilla sisältöalueen maksimileveys oli useimmiten 960 pikseliä, joka johtui tuohon aikaan yleisestä 1024 pikselin resoluutiosta näytöissä. (Cannon 2011.)

CSS-ohjelmistokehysten tärkeimpänä ominaisuutena pidetään niiden tarjoamaa ruudukkorakennetta. Responsiivisissa ohjelmistokehyksissä tämä on toteutettu joustavan ruudukon päälle. Joustavan ruudukon tarkoituksena on määrittää sivulla olevien elementtien koko prosenteissa, jolloin elementtien koko vaihtelee joustavasti selaimen koon mukaan (kuva 5). (Awwwards 2013.)



Kuva 5. Bootstrap-ohjelmistokehyksen ruudukkorakenne (Bootstrap 2015a)

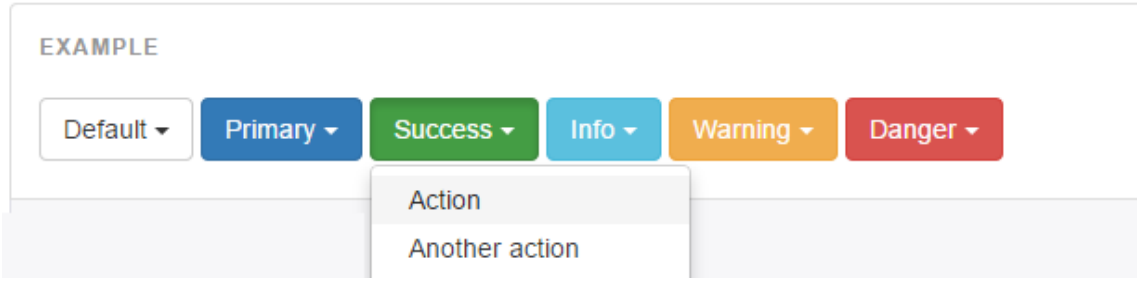
Responsiivisista CSS-ohjelmistokehyksistä selkeästi suosituimpia ovat tällä hetkellä Bootstrap ja Foundation (taulukko 1). Suurta osaa verkossa olevista sivustoista ei kuitenkaan ole vielä suunniteltu responsiivisen ohjelmistokehyksen päälle. Suosituimpien ei-responsiivisten CSS-ohjelmistokehysten, HTML5 Boilerplaten ja 960gs:n, markkinaosuus onkin vielä lähes 60 prosenttiyksikköä. (BuiltWith 2016a.)

Taulukko 1. Suosittujen responsiivisten CSS-ohjelmistokehysten ominaisuuksien vertailu (Vermilion 2015)

	Bootstrap 4.0.0-alpha	**Foundation 6**	**Skeleton 2.0.1**
Viimeisin julkaisupäivä	19.8.2015	19.11.2015	11.12.2014
Julkaisija	Mark Otto, Jacob Thornton	ZURB	Dave Gamache
Lisenssi	MIT License	MIT License	MIT License
CSS-esikäsittelijä	SASS	SASS	-
CSS:än alustuskirjasto	oma	normalize.css	normalize.css
Ruudukon palstojen oletusmäärä	12	12	12
Flexbox-tuki	valinnainen	valinnainen	-
Käyttöliittymäkomponentit	hyvin laaja valikoima erilaisia CSS- ja JS-komponentteja	laajin valikoima näistä kolmesta, mm. ikonit ja vaakasuuntaiset listat, joita ei ole Bootstrapissa	vain peruselementit (taulukot, painikkeet, listat ja lomakkeet)

3.1 Käyttöliittymäkomponentit

Laajemmat CSS-ohjelmistokehykset tarjoavat kehittäjälle myös valmiita komponentteja käyttöliittymien rakentamiseen (kuva 6). Näiden komponenttien avulla voidaan helposti toteuttaa visuaalisesti eheitä ja toimivia prototyyppejä tai jopa kokonaisia verkkosivustoja. Valmiiden käyttöliittymäkomponenttien käyttöä on kuitenkin kritisoitu. Monen mielestä niitä ei tulisi käyttää kokonaisten verkkosivujen toteutukseen vaan lähinnä suunnittelun pohjaksi, jotta niillä toteutetut verkkosivut eivät näyttäisi toistensa kopioilta. (Kramer 2014.)



Kuva 6. Bootstrap tarjoaa kehittäjille valmiiksi tyyliteltyjä komponentteja kuten pudotusvalikoita (Bootstrap 2015b)

Käyttöliittymäkomponenttien käytöstä on pyritty tekemään yksinkertaista ja niitä käytetään useimmiten asettamalla HTML-elementin luokiksi komponentin nimi sekä tarkentavat luokat (Awwards 2013). Tarkentavat luokat sallivat helposti komponenttien ulkoasun ja toiminnallisuuksien muokkaamisen myös sivu- ja elementtikohtaisesti.

Usein käyttöliittymäkomponenttien tekoon on käytetty avuksi myös komentosarjakieli JavaScriptiä. Yleisin JavaScript-kirjasto verkkosivuilla on jQuery, joka oli maaliskuussa 2016 BuiltWith-sivuston (2016b) tilastojen mukaan käytössä yli 60 prosentissa suurimmista verkkosivustoista. JavaScriptillä toteutetuista komponenteista ovat esimerkiksi erilaiset ilmoitus- ja häiriöviestit, animaatiot sekä kuvakarusellit (Bootstrap 2015c).

3.2 CSS-esikäsittelijät

Esikäsittelijöiden avulla kehittäjät voivat selkeyttää verkkosivustojen ulkoasua määrittelevien CSS-tyylitiedostojen (Cascading Style Sheets) rakennetta sekä helpottaa monimutkaisten tyylirakenteiden tekoa. Esikäsittelijät laajentavat perinteisten CSS-tyylitiedostojen mahdollisuuksia tuomalla mukaan muun muassa sisäkkäiset määritykset, muuttujat, ehto- ja silmukkalauseet, funktiot sekä mixinit. Erityisesti mixinit ovat hyödyllisiä apuvälineitä, joilla voidaan lisätä joukko CSS-määrityksiä useammalle elementille sekä antaa niille määrityksiä käyttäen muuttujia. (Cederholm 2013.)

Tällä hetkellä suosituimmat esikäsittelijät julkisessa koodin versionhallintapalvelu GitHubissa (2015) ovat Less, Sass/SCSS sekä Stylus. Näistä jokainen tuo omat vaatimuksensa koodin rakenteeseen, mutta perustoiminallisuudet ovat samat. Less ja Stylus pohjautuvat JavaScriptiin ja ovat helpompia ottaa käyttöön verrattuna Ruby-ohjelmointikieleen perustuvaan Sassiin. Lessin ominaisuudet taas ovat vajaammat verrattuna Stylukseen tai Sassiin. (Slant 2015.)

Myös monet tämän hetken suosituimmista ohjelmistokehyksistä käyttävät hyödyksi CSS-esikäsittelijöitä. Bootstrap on rakennettu tukemaan sekä Less- että Sass-esikäsittelijöitä ja Foundation taas on tehty Sass-esikäsittelijälle. Ensimmäisten ohjelmistokehysten, kuten 960gs:n, aikaan ei vielä ollut olemassa esikäsittelijöitä ja ne ovatkin rakennettu pelkästään puhtaalla CSS:llä. (Mehrabani, Emrani 2016.)

Esikäsittelijöiden tehtävänä on kääntää ohjelmistosuunnittelijan tekemät määritykset toimiviksi CSS-tiedostoiksi (kuva 7). Esikäsittelijöillä on oma syntaksi, joka kuitenkin muistuttaa perinteistä CSS-kieltä, mutta sisältää uusia ominaisuuksia. Esikäsittelijöiden käytön aloittamisen helpottamiseksi alkuperäinen CSS-kieli on lähes aina yhteensopivaa esikäsittelijöiden käyttämän syntaksin kanssa. (Cederholm 2013.)

```
// Definitions
$headingColor: #28c0da;
$headingFont: "Comic Sans MS" sans-serif;

// Heading
header.main{
    .big-title{
        font-family:$headingFont;
        a{
            color:$headingColor;
        }
    }
}
nav.main{
    ul{
        &:before{
            content: ' ';
            background: url('nav-bg.png');
        }
        li{
            font-family:$headingFont;
        }
    }
}
```

```
header.main .big-title {
  font-family: "Comic Sans MS" sans-serif;
}

header.main .big-title a {
  color: #28c0da;
}

nav.main ul:before {
  content: ' ';
  background: url("nav-bg.png");
}

nav.main ul li {
  font-family: "Comic Sans MS" sans-serif;
}
```

Kuva 7. Esimerkki SASS-esikäsittelijällä tehdystä CSS-tiedostosta. Vasemmalla alkuperäinen SCSS-tiedosto ja oikealla tästä luotu CSS-tiedosto

4 Grid Layout

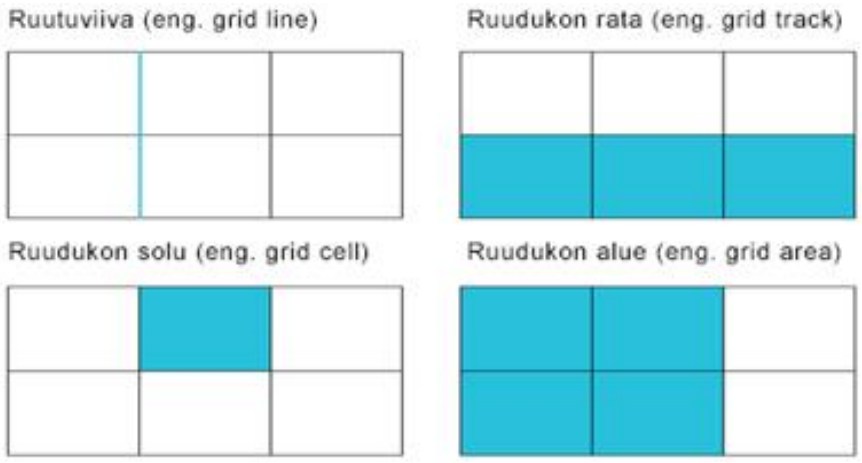
Grid Layout -moduulin ensimmäinen luonnos julkaistiin vuonna 2011, kun responsiivinen verkkosuunnittelu yleistyi ja kehittäjäyhteisöt alkoivat kaipaamaan HTML-kieleen standardoitua tapaa tehdä ruudukkorakenteita. (Andrew 2016b, 1-2.)

Moduulin tarkoituksena on mahdollistaa natiivi kaksiulotteinen ruudukkorakenne, jonka sisälle voidaan asetella elementtejä määrättyihin ruutuihin. Grid Layout tukee myös elementtien asetteluja eri sijainteihin responsiivisesti käyttäen hyödyksi media-query-määrittelyjä. (Andrew 2016b, 4.)

Moduulin avulla koko verkkosivun ylätason elementtien asettelut voidaan määritellä pelkästään CSS-kieltä käyttämällä. Tällöin kehittäjien on mahdollista pitää verkkosivuston tyyli- ja ulkoasumäärittelyt CSS-tiedostossa ja HTML-tiedostoissa säilyy nimensä mukaisesti (HyperText Markup Language) vain sivuston merkkauselementit sekä itse sisältö. (House 2016.)

Ruudukko määritellään asettamalla sivustolle HTML-elementti, esimerkiksi ”div”, jolle annetaan CSS-määritys ”display:grid”. Tämän jälkeen kyseinen HTML-elementti toimii ruudukkorakenteen ylätason elementtinä. (Andrew 2016b, 5.)

Yläelementille määritellään ruudukon radat (eng. grid tracks), eli rivit ja palstat, sekä niiden leveydet. Ratojen perusteella muodostuvat ruutuviivat (eng. grid lines) ja neljän ruutuviivan reunustamasta osiosta käytetään nimitystä ruudukon solu (eng. grid cell). Monen solun kokonaisuutta kutsutaan ruudukon alueeksi (eng. grid area). (Kuva 8.) (Andrew 2016b, 11-12.)

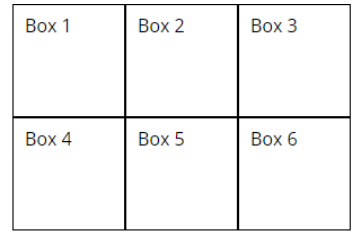


Kuva 8. Grid Layout -ruudukolle määritellään ruutuviivat, jotka muodostavat ruudukon palstat ja rivit, kuvassa turkoosilla merkitty osa vastaa ylläolevaa nimeä

4.1 Ruudukon ratojen määrittely

Ylätason elementille ruudukon radat määritellään CSS-määrityksillä “grid-template-columns” ja “grid-template-rows” (Atkins, Etemad & Atanassov 2015). Kuvassa 9 on määritelty kolme palstaa, jotka ovat 100 pikseliä leveitä sekä kolme riviä, jotka ovat kaikki 100 pikseliä korkeita.

```
.grid-container{
    display:grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
}
```



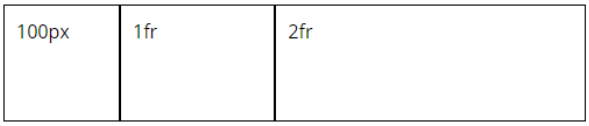
Kuva 9. Grid Layout -ruudukko kolmella 100px leveällä ja korkealla palstalla

Ratojen määrittelyyn voi käyttää kiinteitä pikselimäärien lisäksi prosentteja, määritystä ”auto” tai uutta fr-mittayksikköä. Prosentit mahdollistavat joustavan ruudukkorakenteen, jossa radan koko muuttuu ylätason elementin koon mukaan. ”Auto”-määrityksellä radan koko muuttuu alueen sisältämän sisällön mukaan, jolloin se on looginen vaihtoehto esimerkiksi sisältöalueelle. (Atkins, Etemad & Atanassov 2015.)

Fr-mittayksiköllä (fraction) tarkoitetaan jäljellä olevaa tilaa yläelementissä. Radan kaikille fr-mittayksikköä käyttäville alueille jaetaan tila sen mukaan, kuinka monta fr-yksikköä ne käyttävät (Atkins, Etemad & Atanassov 2015). Esimerkiksi arvolla ”2fr” osio vie jäljellä olevasta tilasta kaksi kertaa enemmän kuin osio jolle on annettu arvoksi ”1fr” (kuva 10).

```
.grid-container{
    display:grid;
    grid-template-rows: 100px 100px 100px;

    width:500px;
    grid-template-columns: 100px 1fr 2fr;
}
```

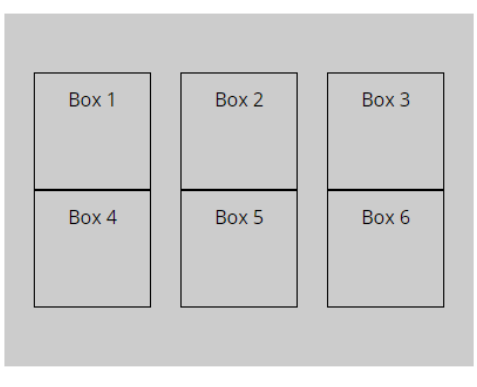


Kuva 10. Fr-mittayksikkö (rivi 6) määrittelee kuinka ison osan jäljellä olevasta tilasta elementti vie

Jos ruudukko on suurempi kuin sen sisältämät radat, alkavat palstat ja rivit oletuksena ylätason elementin vasemmasta ylälaidasta. Tähän voidaan vaikuttaa pystyakselilla CSS-määrityksellä ”align-content” ja vaaka-akselilla määrityksellä ”justify-content” (kuva 11). (Atkins, Etemad & Atanassov 2015.)

Arvolla ”start” radat alkavat akselin alkupäästä, arvolla ”end” loppupäästä ja arvolla ”center” radat keskittyvät akselin mukaan. Radat voi myös tasata koko ruudukon alueelle arvoilla ”space-between”, ”space-evenly” tai ”space-around”, jotka jakavat tilan eri tavoin ratojen välille. Arvolla ”stretch” voidaan pakottaa radat venymään koko ruudukon alueelle. (Atkins, Etemad & Atanassov 2015.)

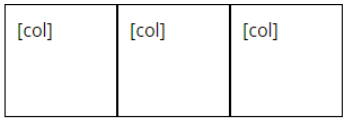
```
1  .grid-container{
2      display:grid;
3      grid-template-rows: 100px 100px;
4      grid-template-columns: 100px 100px 100px;
5      width:400px;
6      height:300px;
7
8      align-content:center;
9      justify-content:space-evenly;
10
11 }
12
13
14
15
16
17
18
```



Kuva 11. Ruudukon ratojen asettuminen ruudukossa tapahtuu ”align-content” ja ”justify-content CSS-määrityksillä (rivit 8-9)

Ruudukon käyttämisen helpottamiseksi ruutuviivat voidaan myös nimetä. Tällöin kehityksessä ei tarvitse muistaa ruutuviivojen järjestysnumeroa vaan ruudukon alueet voidaan määrittää käyttämällä kuvaavia nimiä. Nimet voivat olla yksilöllisiä, alueen sisältöä kuvaavia, kuten ”sivupalkki” tai niitä voi määritellä useampia samannimisiä. Vastaavan tapaisesti monissa perinteisissä ruudukkorakenteissa käytetään englannin palstaa tarkoittavasta ”column” sanastaa lyhennettä ”col” (kuva 12). (Atkins, Etemad & Atanassov 2015.)

```
1  .grid-container{
2      display:grid;
3      grid-template-rows: 100px 100px 100px;
4
5      grid-template-columns: [col] 100px [col] 100px [col] 100px;
6  }
7
```



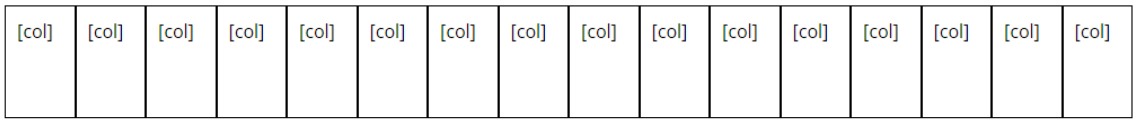
Kuva 12. Grid Layout -ruudukko, jossa on määritelty kolme ”col”-nimistä ruutuviivaa (rivi 5)

Ratojen määrittelemistä helpottaa myös toistuvien viivamääritysten teko. Toistuvilla viivamäärityksillä voidaan määritellä nimetty tai nimeämätön viiva toistumaan ruudukon rakenteessa tietty määrä kertoja (Atkins, Etemad & Atanassov 2015). Nimetyillä viivoilla tämä mahdollistaa perinteisten CSS-ohjelmistokehysten kaltaisten ruudukkorakenteiden luomisen (kuva 13).

```
.grid-container{
    display:grid;
    grid-template-rows: 100px 100px 100px;

    max-width:1000px;
    grid-template-columns: repeat(16, [col] 1fr);
}

```



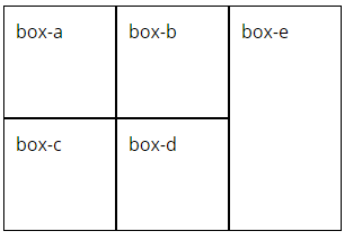
Kuva 13. Grid Layout -ruudukko, jossa on määritelty 16 ”col”-nimistä ruutuviivaa ”repeat”-arvolla (rivi 6)

4.2 Ruudukon alueiden määrittely

Ruudukkoalueet määritellään asettamalla ruudukon sisällä olevalle HTML-elementille palstat ja rivit, jotka se täyttää. Oletuksena elementit sijoittuvat seuraavaan vapaaseen soluun kuten aikaisemmissa esimerkeissä. (Atkins, Etemad & Atanassov 2015.)

Käytettävät palstat annetaan CSS-määrityksellä ”grid-column” ja käytettävät rivit määrityksellä ”grid-row” (kuva 14). Määrityksille annetaan alueen ensimmäisen radan järjestysnumero ruudukossa sekä viimeisen radan järjestysnumero. (Atkins, Etemad & Atanassov 2015.)

```
.grid-container{
    display:grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
}
.box-a{
    grid-row: 1;
    grid-column: 1;
}
.box-b{
    grid-row: 1;
    grid-column: 2;
}
.box-c{
    grid-row: 2;
    grid-column: 1;
}
.box-d{
    grid-row: 2;
    grid-column: 2;
}
.box-e{
    grid-row: 1 / 3;
    grid-column: 3;
}
```

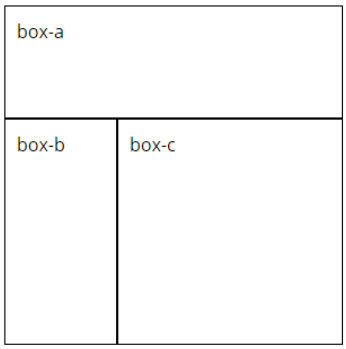


Kuva 14. Ruudukon elementtien alueet määritellään käyttäen ”grid-column”, ”grid-row” ja ”grid-area” määrityksiä (rivit 6-25)

Määrityksistä on myös käytettävissä pidemmät ”grid-column/row-start” ja ”grid-column/row-end” -muodot sekä lyhyempi ”grid-area” muoto, jolla voidaan määrittää sekä käytettävien palstojen että rivien määrä. (Atkins, Etemad & Atanassov 2015.)

Jos ei haluta määritellä mihin ruutuviivaan alue loppuu, vaan montako rataa leveä alue on, voidaan avainsanalla ”span” määritellä alueen koko palstoissa tai riveissä. Esimerkiksi määritys ”grid-column: 1 / span 3” määrittää alueen alkamaan ruutuviivasta yksi ja päättymään viivaan neljä (kuva 15). (Atkins, Etemad & Atanassov 2015.)

```
1  .grid-container{
2      display:grid;
3      grid-template-columns: 100px 100px 100px;
4      grid-template-rows: 100px 100px 100px;
5  }
6  .box-a{
7      grid-row: 1;
8      grid-column: 1 / span 3;
9  }
10 .box-b{
11     grid-row: 2 / span 2;
12     grid-column: 1;
13 }
14 .box-c{
15     grid-row: 2 / span 2;
16     grid-column: 2 / span 2;
17 }
```

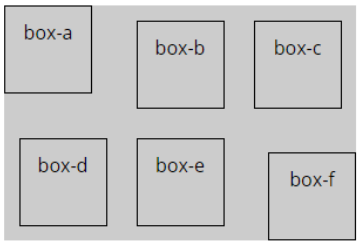


Kuva 15. ”span”-arvolla (rivit 8, 11, 15 ja 16) voidaan määrittää elementti jatkumaan tietyn määrän palstoja tai rivejä eteenpäin

Joissakin tilanteissa ruudukkoon määritelty elementti saattaa olla pienempi kuin sille annettu ruudukkoalue. Tällöin elementin sijaintia alueella voidaan määritellä pystyakselilla määrityksellä ”align-self” ja vaaka-akselilla ”justify-self” (kuva 16). Ruudukon kaikkien elementtien oletussijainti voidaan määrittää antamalla ruudukon ylätason elementille vastaavat arvot CSS-määrityksillä ”align-items” ja ”justify-items”. (Atkins, Etemad & Atanassov 2015.)

Oletuksena elementti sijaitsee alueen vasemmassa ylälaidassa, kun molemmilla määrityksillä on oletusarvo ”start”. Vaihtamalla määrityksen arvoksi ”end” saadaan elementit sijoittumaan akselin loppupäähän, arvolla ”center” elementti keskittyy ja arvolla ”stretch” elementti venyy täyttämään koko alueen. (Atkins, Etemad & Atanassov 2015.)

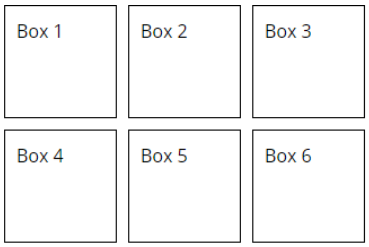
```
1  .grid-container{
2      display:grid;
3      grid-template-rows: 100px 100px;
4      grid-template-columns: 100px 100px 100px;
5      align-items: center;
6      justify-items: center;
7  }
8  .box{
9      width:75px;
10     height:75px;
11 }
12 .box-a{
13     align-self: start;
14     justify-self: start;
15 }
16 .box-f{
17     align-self: end;
18     justify-self: end;
19 }
20
```



Kuva 16. Elementtien asettuminen ruudukkoalueella tapahtuu ”align-items” ja ”justify-items” CSS-määrityksillä (rivit 5-6) tai elementtikohtaisesti ”align-self” ja ”justify-self” määrityksillä (rivit 13-14 ja 17-18)

Perinteisissä ruudukkorakenteissa on yleensä mahdollisuus lisätä ruudukon palstojen välille marginaali jaottelemaan sisältöä. Grid Layout -moduulissa tämä tehdään antamalla ylätason elementille CSS-määritys ”grid-column-gap”. Tämän lisäksi Grid Layout mahdollistaa marginaalin määrittelemisen rivien välille määrityksellä ”grid-row-gap”. Näistä voidaan myös käyttää lyhempää muotoa ”grid-gap”, jolla voidaan määritellä molemmat arvot (kuva 17). (Atkins, Etemad & Atanassov 2015.)

```
1  .grid-container{
2      display:grid;
3      grid-template-columns: 100px 100px 100px;
4      grid-template-rows: 100px 100px 100px;
5
6      grid-gap: 10px;
7  }
8
9
10
11
12
```

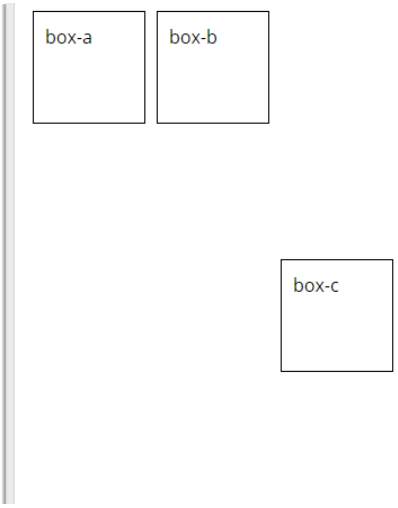


Kuva 17. ”grid-gap”-määrityksellä (rivi 6) voidaan asettaa ruudukon palstoille ja riveille marginaalit

Ylätason elementin sisälle voidaan myös luoda alueita, jotka sijaitsevat ruudukon ulkopuolella. Esimerkiksi jos ruudukkorakenteessa on vain neljä palstaa, voidaan määrityksellä ”grid-column: 5 / 6” sijoittaa elementti ruudukon ulkopuolelle. Tällöin elementti ilmestyy ruudukon oikealle puolelle. Jos halutaan määritellä elementti ruudukon vasemmalle puolelle, voidaan sille antaa negatiivisia arvoja. Sama logiikka toimii myös riveissä ruudukon ylä- ja alapuolelle. (Atkins, Etemad & Atanassov 2015.)

Kun ruudukon ulkopuolelle määritellään elementtejä, ylätason elementin palstojen ja rivien koko määräytyy ”grid-auto-columns” ja ”grid-auto-rows” määrityksillä. Myös ruudukkoon nimettyjä ruutuviivoja voidaan käyttää ulkopuolelle sijoittuvien elementtien koon määrittelyssä (kuva 18). (Atkins, Etemad & Atanassov 2015.)

```
1  .grid-container{
2      display:grid;
3      grid-template-columns:100px 100px;
4      grid-template-rows:100px 100px;
5
6      grid-auto-columns: 100px;
7      grid-auto-rows: 100px;
8      grid-gap: 10px;
9  }
10 .box-a{
11     grid-row: 1;
12     grid-column: 1;
13 }
14 .box-b{
15     grid-row: 1;
16     grid-column: 2;
17 }
18 .box-c{
19     grid-row: 3;
20     grid-column: 3;
21 }
22
```



Kuva 18. Elementtejä voidaan myös asettaa määritellyn ruudukon ulkopuolelle (rivit 18-21)

4.3 Sisäkkäiset ruudukot

Sisäkkäisillä ruudukkorakenteilla (eng. nested grid) tarkoitetaan ruudukoita, jotka sijaitsevat toisen ruudukon sisällä. Sisäkkäisiä ruudukoita käytetään yksinkertaistenkin ulkoasujen asettelujen toteuttamiseen, mutta ne ovat aiheuttaneet kehittäjille haasteita aikaisemmissa CSS-ohjelmistokehyksissä (Davis, 2012). Grid Layout -moduulin toteutukset eri selaimissa tukevat jo osittain sisäkkäisiä ruudukkoja (Andrew 2016b, 24-25).

Selainten toteutuksista isoin puuttuva osa standardista ovat tällä hetkellä sisäkkäiset alaruudukot (eng. subgrid). Alaruudukot mahdollistavat sen, että sisäkkäinen ruudukko ei muodosta omia palstojaan vaan kaikki noudattavat ylemmän tason ruudukon määrittelemää rakennetta. (Andrew 2016b, 24-25.)

Alaruudukkojen tarve konkretisoituu, kun ruudukkorakenne otetaan käyttöön hoitamaan toistuvien elementtien asettelut identtisesti toisiinsa nähden. Eric Meyer käyttää tästä esimerkkinä lomaketta, jonka kenttien otsikot voivat olla pitkiä ja rivittyä monelle. Tällöin ilman alaruudukkoja jokainen lomakkeen kenttä muodostaisi oman ruudukkonsa huolimatta siitä minkä kokoisia muut kentät ovat. (Meyer 2016.)

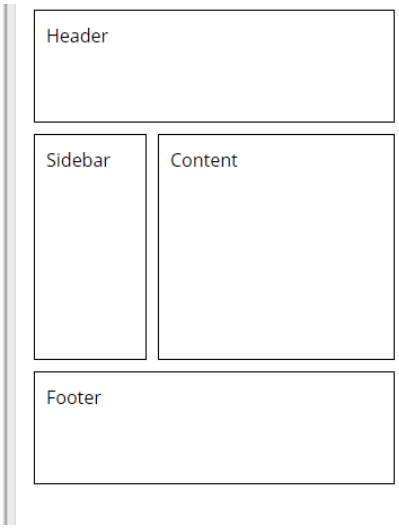
Meyerin artikkelissa (2016) myös kommentoidaan, että saman asian voi toteuttaa jo nyt esimerkiksi taulukko-elementillä (eng. table). Samoin selainten ja moduulin kehittäjät keskustelevat teknisen toteutuksen muista ongelmista ja siitä, pitäisikö niiden toteutus siirtää W3C-määrityksen myöhempään versioon (Rego 2016).

Kuitenkin monet tunnetuista frontend-alalla toimivista ihmisistä, kuten Rachel Andrew (2015b) ja Eric Meyer (2016), ovat vaatineet sisäkkäisten ruudukoiden kehittämistä selaimiin jo tässä vaiheessa. Heidän mielestään vanhat ratkaisut ulkoasujen tekemisessä eivät ole nykypäivää ja verkkosivut sekä niiden kehittäjät ansaitsevat vihdoin kunnollisen ulkoasumoottorin.

4.4 Nimetyt ruutualueet

Nimetyt ruutualueet (eng. named grid areas) ovat vaihtoehtoinen tapa määritellä ruudukon alueiden sijainti ja koko. Tällöin ruudukossa sijaitsevat elementit nimetään CSS-määrityksellä ”grid-area” ja ruudukkorakenteen ylätason elementissä asetellaan elementit eri ruudukkoviivojen sisälle (kuva 19) määrityksellä ”grid-template-areas”. (Atkins, Etemad & Atanassov 2015.)

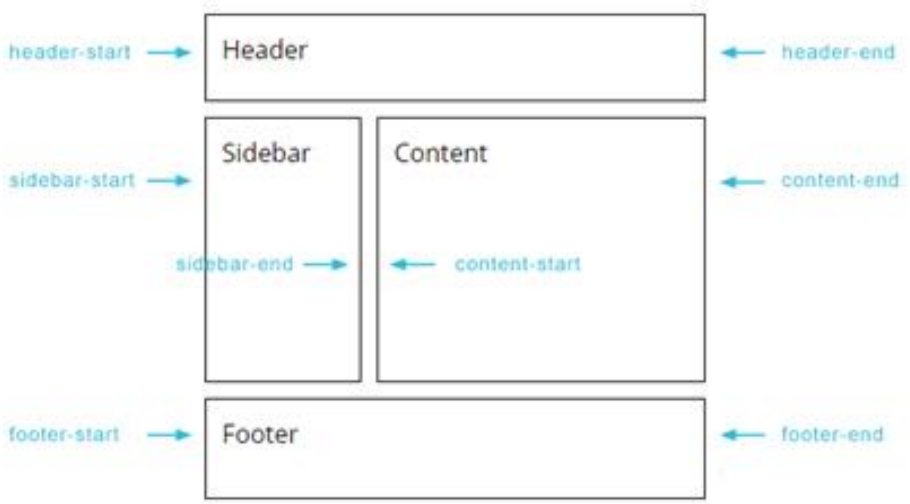
```
1  .grid-container{
2      display:grid;
3      grid-template-columns:100px 100px 100px;
4      grid-template-rows:100px 200px 100px;
5      grid-gap: 10px;
6
7      grid-template-areas:"header header header"
8                          "sidebar content content"
9                          "footer footer footer";
10 }
11 .sidebar {
12     grid-area: sidebar;
13 }
14 .content {
15     grid-area: content;
16 }
17 .header {
18     grid-area: header;
19 }
20 .footer{
21     grid-area: footer;
22 }
23
```



Kuva 19. Nimetyt ruutualueet (rivit 7-9) mahdollistavat asettelun määrittelemisen ylätason elementissä

Ruutualueet selkeyttävät ruudukon asettelujen määritystä, kun määrittely voidaan tehdä muuttamalla vain ylätason elementtiä. Tällöin kehittäjän ei tarvitse huolehtia alatason elementtien määritysten muuttamisesta. Tämä helpottaa myös responsiivisuuden suunnittelua, kun media-query -määrittelyjen avulla ruudukolle saadaan yhdestä paikasta asetettua vaihtoehtoiset asettelut erikokoisilla ruuduilla. (Atkins, Etemad & Atanassov 2015.)

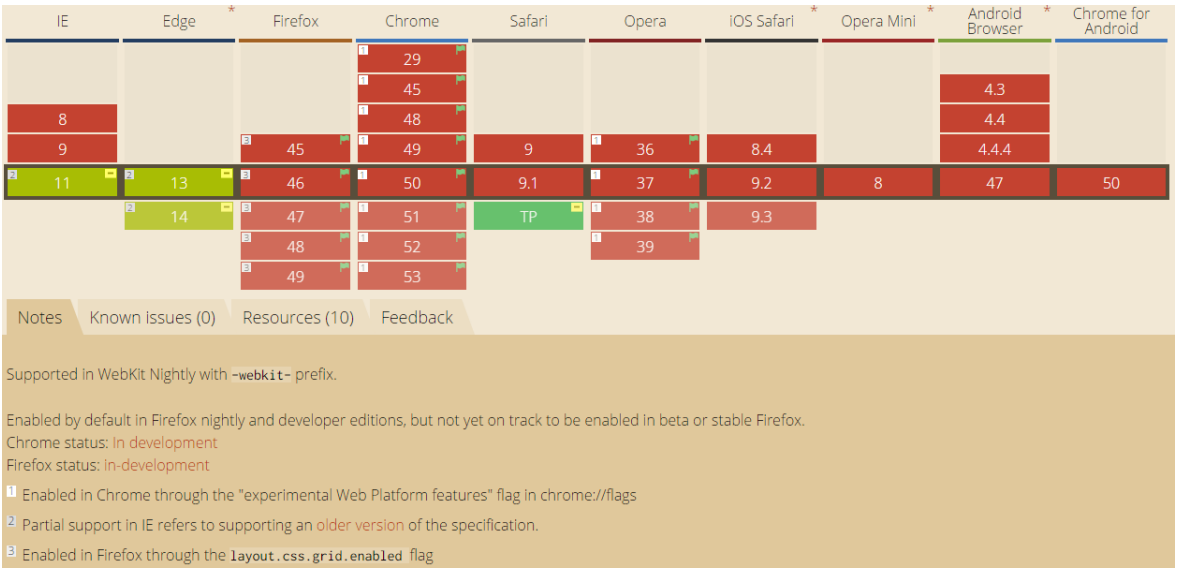
Käytettäessä nimettyjä ruutualueita alueita rajaaville viivoille muodostuu myös nimet, joita voi käyttää hyödyksi muissa määrittelyssä (Atkins, Etemad & Atanassov 2015). Aiemmassa esimerkissä toiselle palstaviivalle muodostuu nimi ”content-start” ja viimeiselle viivalle ”content-end”. Samat nimet muodostuvat myös rivien viivoille 3 ja 5 (kuva 20).



Kuva 20. Ruutualueita käytettäessä viivoille muodostuu uudet nimet alueiden mukaan

4.5 Selaintuki

Suurin ongelma Grid Layout -moduulin käyttämiselle on sen puutteellinen tuki selaimissa, jolloin sitä käyttävät sivustot vaikuttavat rikkinäisiltä käyttäjille. Vaikka moduuli sai alkunsa jo vuonna 2011 Internet Explorerin kehittäjien toimesta, niin yksikään moderni selain ei tue sitä täysin natiivisti (kuva 21). (Andrew 2016a.)



Kuva 21. Grid Layout -moduulin tuki yleisimmissä selaimissa (Caniuse.com 2016a)

Internet Explorer on kirjoitushetkellä maaliskuussa 2016 ainoa selain, joka ilmoittaa suoraan tukevansa Grid Layout -moduulia. Tieto on kuitenkin harhaanjohtava, sillä Internet Explorer ja Microsoft Edge tukevat vain moduulin aikaisempaa ja vanhentunutta standardiluonnosta (Andrew 2016a). Microsoft Edgen osalta on kuitenkin merkitty korkeaksi prioriteetiksi saada tuki uudemmalle standardille (Microsoft 2014).

Tällä hetkellä pisimmällä kehityksessä on Blink-selainmoottori, joka toimii Chrome ja Opera -selainten taustalla. Blink-moottorissa toiminnallisuus on standardin mukainen, mutta se on julkaistu Chromessa vasta vapaaehtoisesti käyttöönotettavana kokeellisena ominaisuutena. (Andrew 2016a.)

Myös WebKit-moottori, josta Blink on eriytetty, tukee Grid Layout -moduulia, mutta on hieman jäljessä Blink-moottorin toteutusta. Tämä tarkoittaa kuitenkin, että WebKit-moottorin avulla toimivat selaimet kuten Safari ja Android Browser luultavasta saavat Grid Layout -tuen lähiaikoina. Samoin Firefoxin taustalla pyörivä Gecko-moottori tukee uusimmissa kokeellisissa Firefox Nightly -versioissa Grid Layout-moduulia, mutta sekin vain erillisen asetuksen ollessa päällä. (Andrew 2016a.)

Julkaiseminen testikäyttöön antaa sivustojen kehittäjille mahdollisuuden tutustua standardiluonnokseen ennalta ja kommentoida sitä. Tämän ansiosta myös standardia kehittävät tahot saavat ensi käden tietoa kehittäjiltä, miten viedä standardia eteenpäin. Tästä johtuen onkin esitetty arvioita, että Grid Layout -moduuli on julkaistaessa hyvin lähellä lopullista muotoaan ja valmis oikeaan käyttöön. (Andrew 2015.)

5 Flexible Box

Flexible Box -moduuli sai alkunsa jo viime vuosikymmenellä ollen Grid Layout -moduulia huomattavasti vanhempi, mistä johtuen se on tällä hetkellä myös valmiimpi käyttöön verkkosivustoilla. Ensimmäiset toimivat versiot selaimissa siitä julkaistiin jo vuonna 2009. (Williamson 2012.)

Vuonna 2011 moduulin toiminnallisuudet kokivat kuitenkin suuren remontin kahteen eri otteeseen, kun myös Grid Layout -moduulia kehittävä Tab Atkins Jr. otti määritystyön haltuunsa. Suositus on käyttää verkkosivustoilla vain moduulin uusinta versiota, vaikka osa selaimista tukeekin myös vanhempia versioita. Nykyään verkosta löytyy vielä paljon molempiin vanhentuneisiin standardiluonnoksiin viittaavaa aineistoa, mikä vaikeuttaa verkkosivustojen kehittäjien työtä. (Coyier 2012; Williamson 2012).

Flexible Box ja Grid Layout -moduulit sekoitetaan usein keskenään, vaikka niiden käyttötarkoitukset ovat erilaiset. Grid Layout -moduuli toimii kaksiulotteisesti ja se soveltuu parhaiten sivuston kokonaisrakenteen toteutukseen, kun Flexible Box -moduulin tarkoituksena on asemoida sisältöelementtejä sivustolla yksiulotteisesti. (Atkins 2013.)

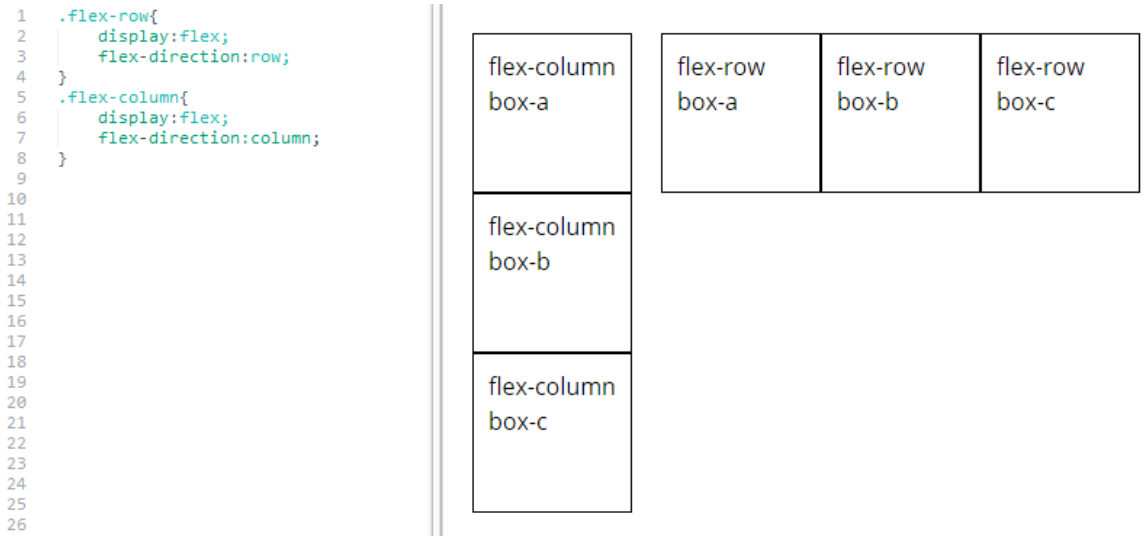
Flexible Box -moduulin avulla pystytään ratkaisemaan paljon perinteisten ruudukkojärjestelmien ongelmia. Näihin lukeutuvat muun muassa sisältöelementtien järjestäminen, elementtien keskittäminen, keskinäisten suhteiden ja sijainnin määrittäminen sekä sisällön rivittäminen responsiivisesti. (Andrew 2014, 21–37.)

5.1 Flexible Box ylätason elementin määrittäminen

Flexible Box -moduuli otetaan käyttöön määrittelemällä sivustolle HTML-elementti, jolle annetaan CSS-määritys ”display: flex;” tai ”display: inline-flex;”. Tällöin kyseisen elementin sisällä olevat elementit muuttuvat flex-elementeiksi (eng. flex items). Normaalilla ”flex”-arvolla ylätason elementti on ”block”-tyyppinen ja ”inline-flex”-arvolla elementti on ”inline” tyyppinen. (Atkins, Etemad & Atanassov 2016.)

Ylätason elementille annettavan ”flex-direction”-määrityksen tarkoituksena on määrittää flex-elementtien sijainnin lähtöpiste ja suunta, johon seuraavat elementit sijoitetaan. Lähtöpiste ja suunta määrittelevät kumpi, X- vai Y-akseli, on pääakseli (eng. main axis) ja kumpi on risteävä akseli (eng. cross axis). Oletuksena elementit järjestyvät vaakatasoon alkaen vasemmalta, eli X-akseli on pääakseli ja Y-akseli risteävä akseli. (Atkins, Etemad & Atanassov 2016.)

Pääakselia voi vaihtaa antamalla ”flex-direction” määritykselle arvoksi ”column”, jolloin flex-elementit järjestyvät pystyriviin Y-akselin mukaisesti (kuva 22). Elementtien sijoittamisen alkupisteen voi vaihtaa lisäämällä ”flex-direction” määrityksen arvon loppuun ”-reverse”. Esimerkiksi ”column-reverse” määrittää elementit järjestymään pystyriviin alkaen pohjalta. Vastaavasti ”row-reverse” -arvolla elementit järjestyvät vaakatasoon alkaen ylätason elementin oikeasta reunasta. (Atkins, Etemad & Atanassov 2016.)



Kuva 22. Flexible Box -moduulissa elementit sijoittuvat yksiulotteisesti joko pysty- tai vaakasuunnassa ”flex-direction”-määrityksen (rivit 3 ja 7) mukaisesti

5.2 Flex-elementtien koon määrittely

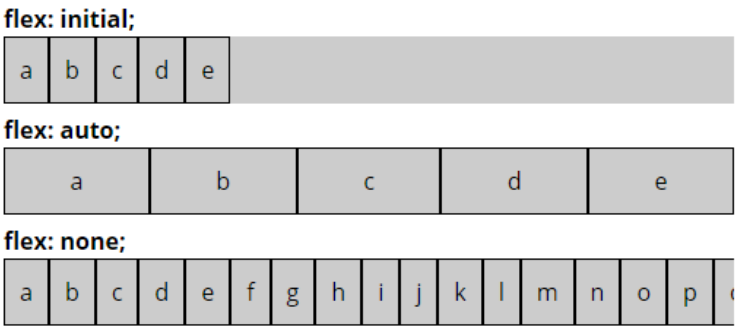
Flex-elementtien kokoja voi säädellä CSS-määrityksillä ”flex-grow”, ”flex-shrink” ja ”flex-basis”. Määritykset suositellaan yleisesti annettavan lyhemmällä ”flex” määrityksellä, jolloin ne annetaan muodossa ”flex: <flex-grow> <flex-shrink> <flex-basis>”. ”Flex”-määrityksellä on myös nimettyjä arvoja (kuva 23).

Oletuksena elementtien koko määräytyy niiden sisällön mukaan, mutta mikäli elementtien sisältö on leveämpi kuin sille annettu tila, elementti pyrkii kutistumaan mahtuakseen kyseiseen tilaan. Tätä vastaava flex-määrityksen arvo on ”initial” tai ”flex: 0 1 auto”. (Atkins, Etemad & Atanassov 2016.)

Flex-määritykselle voidaan myös antaa arvoksi ”auto”, jolloin elementit venyvät koko ylätason elementin akseleiden leveydelle. Tämä vastaa määritystä ”flex: 1 1 auto”. Arvolla ”none” voidaan kyseisestä elementistä tehdä täysin joustamaton. Tällöin elementti säilyt-

tää oman kokonsa, vaikka ylätason elementistä loppuisi tila. (Atkins, Etemad & Atanassov 2016.)

```
1  .flex{
2      display: flex;
3      width: 500px;
4  }
5
6  .initial .box{
7      flex: initial;
8  }
9
10 .auto .box{
11     flex: auto;
12 }
13
14 .none .box{
15     flex: none;
16 }
17
18
19
```

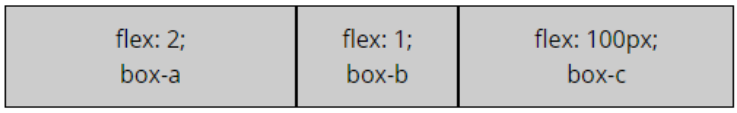


Kuva 23. Flex-määritykselle voi antaa arvoksi muun muassa ”initial”, ”auto” tai ”none” (rivit 7-16)

Flex-elementtien suurentumista ja kutistumista voi säädellä tarkemmin ”flex-shrink” ja ”flex-grow” -määrityksillä. Antamalla näille arvoksi 1 määritetään, että elementti saa kutistua tai suurentua tilan mukaan ja arvolla 0 estetään kyseinen toiminnallisuus. (Atkins, Etemad & Atanassov 2016.)

”Flex-basis”-määrityksellä voidaan säädellä elementtien kokoa ja keskinäisiä suhteita. Yksinkertaisin tapa määritellä elementin koko on määrittää elementin ”flex-basis”-arvo määrityksellä ”flex: <numero>”. Tämä määrittää kuinka ison osan vapaasta tilasta elementti täyttää. Kuvassa 24 ensimmäisellä elementillä on ”flex-basis”-arvo 2, jolloin elementti täyttää 2/3 jäljellä olevasta tilasta. Toisella elementillä on ”flex-basis”-arvo 1, jolloin elementti täyttää 1/3 tilasta. Mitä enemmän elementtejä on, sitä useampaan osaan ylätason tila jakautuu. Muita mahdollisuuksia määrittää elementin koko on antaa ”flex”-määritykselle arvoksi tarkka pituus esimerkiksi pikseleinä tai prosentteina. (Atkins, Etemad & Atanassov 2016.)

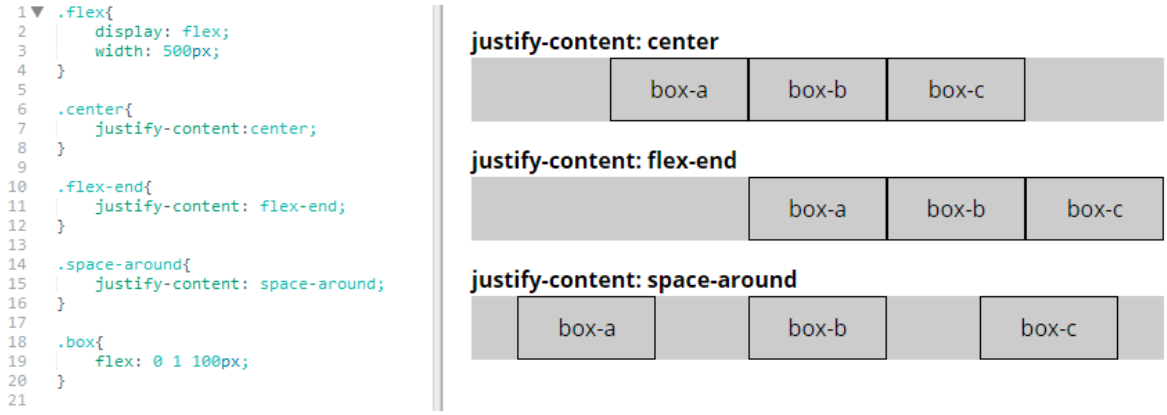
```
1  .flex{
2      display: flex;
3      width: 500px;
4  }
5
6  .box-a{
7      flex: 2;
8  }
9
10 .box-b{
11     flex: 1;
12 }
13
14 .box-c{
15     flex: 100px;
16 }
17
```



Kuva 24. ”Flex” määritys säätelee elementtien kokosuhteita (rivit 6-16)

5.3 Flex-elementtien sijainti akseleilla

Ylätason elementille annettavalla CSS-määrityksellä ”justify-content” voidaan vaikuttaa siihen, missä kohtaa flex-elementit ovat pääakselilla (kuva 25). Arvo ”flex-start” sijoittaa elementit akselin alkupisteeseen ja arvo ”flex-end” vastaavasti akselin loppupisteeseen. Arvolla ”center” elementit keskitetään akselin suuntaisesti. Tasaaminen tapahtuu joko arvolla ”space-between” tai ”space-around”. Ensimmäinen tasaa jäljellä olevan tilan elementtien väliin, kun taas jälkimmäinen tasaa tilan elementtien ympärille. (Atkins, Etemad & Atanassov 2016.)

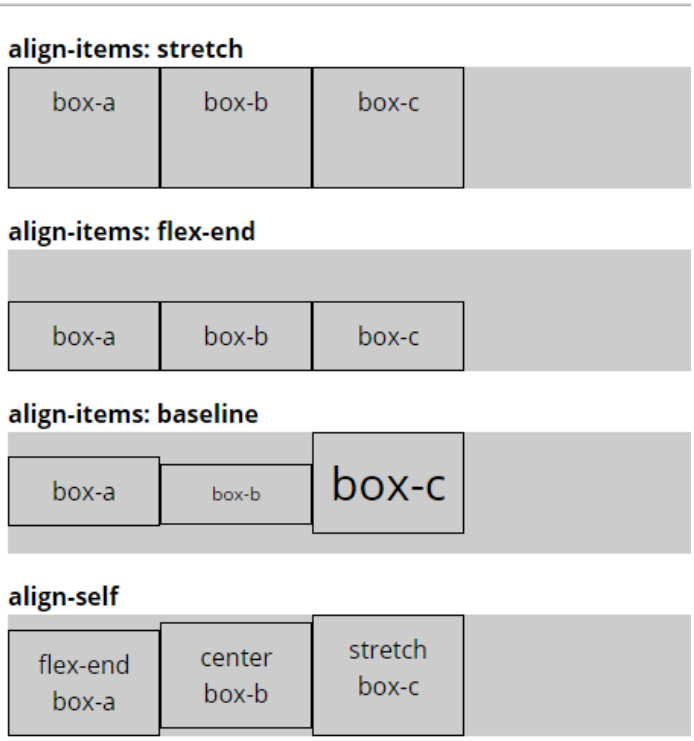


Kuva 25. Elementtien sijaintiin pääakselilla vaikuttaa ”justify-content” määritys (rivit 6-16)

Vastaavasti on olemassa ”align-items”-määritys, joka määrittää elementtien sijainnin risteävällä akselilla (kuva 26). Tällä määrityksellä on samanlaiset ”flex-start”, ”flex-end” ja ”center” arvot kuin ”justify-content”-määrityksellä. Näiden lisäksi arvo voi olla ”stretch” tai ”baseline”. Oletuksena elementeilllä on arvo ”stretch”, joka venyttää elementit täyttämään koko akselin korkeus. ”Baseline”-arvon avulla voidaan tasata elementtien aloituskohdat niiden sisältämän tekstin mukaan. (Atkins, Etemad & Atanassov 2016.)

Risteävän akselin elementtien sijaintia voidaan myös muuttaa elementtikohtaisesti määrityksellä ”align-self”, joka annetaan ylätason elementin sijaan flex-elementille. Arvot ovat samat kuin ”align-items”-määrityksellä. Elementti sijoittuu tällöin oman asetuksensa mukaisesti. (Atkins, Etemad & Atanassov 2016.)

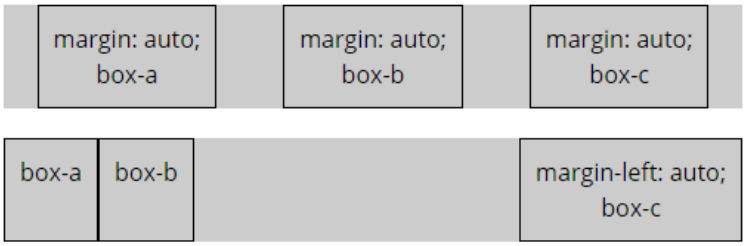
```
1  .flex{
2      display: flex;
3      width: 500px;
4      height:80px;
5  }
6
7  .box{
8      flex: 0 0 100px;
9  }
10
11 .flex-end{
12     align-items: flex-end;
13 }
14
15 .stretch{
16     align-items: stretch;
17 }
18
19 .baseline{
20     align-items: baseline;
21 }
22
23 .baseline .box-b{
24     font-size:12px;
25 }
26
27 .baseline .box-c{
28     font-size:30px;
29 }
30
31 .box.flex-end{
32     align-self: flex-end;
33 }
34
35 .box.center{
36     align-self: center;
37 }
38
39 .box.stretch{
40     align-self: stretch;
41 }
42
```



Kuva 26. ”Align-items” määrittää elementtien sijaintia risteävällä akselilla (rivit 11-29), minkä voi yliajaa elementtikohtaisesti ”align-self”-määrityksellä (rivit 31-41)

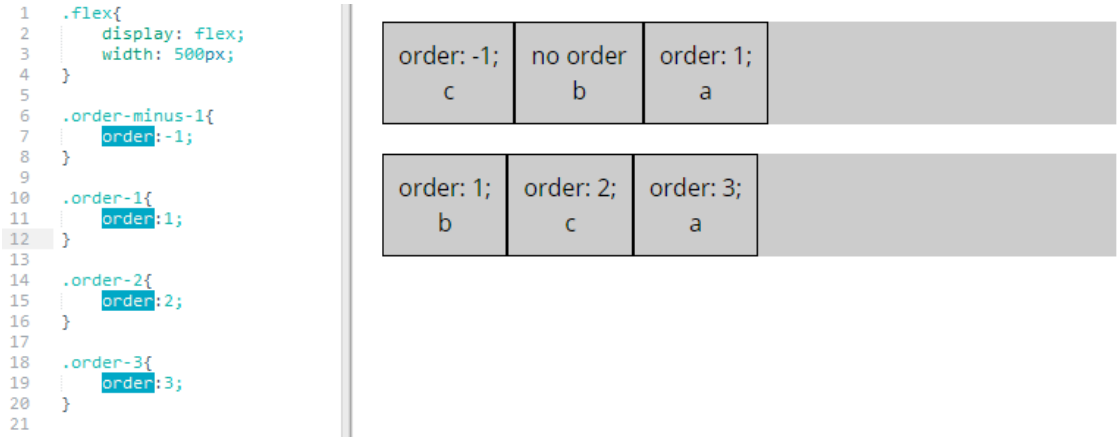
Toinen elementtikohtainen määritys, jolla voidaan vaikuttaa elementin sijaintiin, on ”margin” (kuva 27). Asettamalla elementin marginaalit määrittymään automaattisesti arvolla ”auto”, käyttää elementti kaiken ylimääräisen tilan suuntaan, johon marginaali on määritelty. Jos useammalla elementillä on automaattinen marginaali, ne jakavat käytössä olevan ylimääräisen tilan. (Atkins, Etemad & Atanassov 2016.)

```
1  .flex{
2      display: flex;
3      width: 500px;
4  }
5
6  .margin .box{
7      margin:auto;
8  }
9  .margin-left .box-c{
10     margin-left:auto;
11
12 }
13
14
15
```



Kuva 27. Elementtejä voidaan tasata tai niiden toiselle puolelle voidaan pakottaa ylimääräistä tilaa ”margin”-määrityksellä

Elementtien keskinäistä järjestystä voidaan myös muuttaa. Kaikkien flex-elementtien CSS-määritys ”order” on oletuksena 0 ja ne järjestyvät siihen järjestykseen, jossa ne on laitettu HTML-koodiin (kuva 28). Mikäli tietty elementti halutaan jostain syystä siirtää ensimmäiseksi, sille voidaan antaa arvo -1 tai antaa kaikille muille elementeille arvoksi nollaa suurempi numero. Vastaavasti oletustilanteessa arvo 1 siirtää elementin viimeiseksi. Näin kaikki elementit voidaan järjestää haluttuun järjestykseen antamalla ”order”-määritys useammalle elementille. (Atkins, Etemad & Atanassov 2016.)



Kuva 28. ”Order”-määrityksellä voidaan vaikuttaa elementtien oletusjärjestykseen

5.4 Flex-elementtien rivitys

Oletuksena Flexible Box -moduuli ei rivitä flex-elementtejä useammalle riville, vaan ne jakautuvat tasaisesti yhdelle riville. Määrityksellä ”flex-wrap: wrap;” voidaan pakottaa elementit jakautumaan useammalle riville, mikäli ne eivät mahdu yhdelle (kuva 29). (Atkins, Etemad & Atanassov 2016.)

”Flex-wrap” määrityksellä ylätason elementin sisälle muodostuu uusi flex-rivi (eng. Flex Line) sekä sille uusi pääakseli ja risteävä akseli. Arvolla ”wrap-reverse” voidaan määrittää uusi rivi muodostumaan ennen aikaisempaa riviä, jolloin myös elementit tulevat esille käänteisessä järjestyksessä. (Atkins, Etemad & Atanassov 2016.)

Rivien sijaintia ylätason elementissä voidaan säädellä ”align-content” määrityksellä, joka vaikuttaa rivien sijaintiin risteävällä akselilla. Määritykselle voidaan antaa seuraavia arvoja: stretch, flex-start, flex-end, center, space-between ja space-around. Oletuksena flex-rivit ovat arvossa “stretch”, joka venyttää rivit täyttämään ylätason elementin tila. Muut arvot toimivat samalla logiikalla kuin elementtien sijoittamiseen tarkoitettu ”justify-content”-määritys. (Atkins, Etemad & Atanassov 2016.)

```
.flex{
    display: flex;
    width: 500px;
}

.box{
    flex:0 0 120px;
}

.nowrap{
    flex-wrap: nowrap;
}

.wrap{
    flex-wrap: wrap;
}
```

flex-wrap: nowrap;

a	b	c	d	e

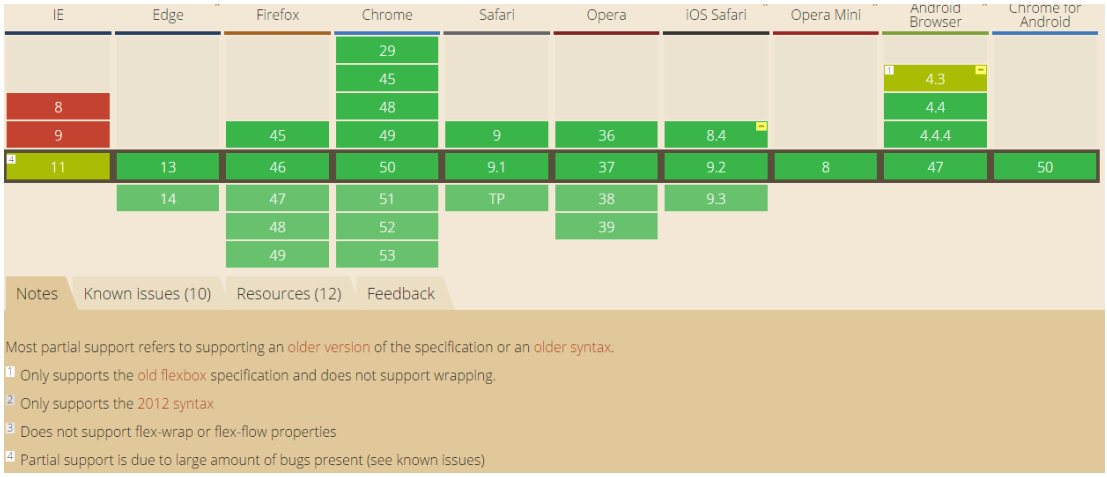
flex-wrap: wrap;

a	b	c	d
e	f	g	

Kuva 29. ”Flex-wrap”-määrityksellä saadaan elementit rivittymään, mikäli ne eivät mahdu yhdelle flex-riville

5.5 Selaintuki

Flexible Box -moduulin selaintuki on huomattavasti Grid Layout -moduulia pidemmällä. Kaikkien suosituimpien selainten versiot tukevat Flexible Box -moduulia jollakin tasolla (kuva 30). (Caniuse.com 2016b.)



Kuva 30. Flexible Box -moduulin tuki yleisimmissä selaimissa (Caniuse.com 2016b)

Internet Explorerin versiot 10 ja 11 ovat ainoat yleisimmin käytetyistä selaimista, joissa on vielä puutteellinen selaintuki Flexible Box -moduulin osalta. Microsoftin kehittäjät ovat tosin todenneet, että Internet Explorerista ei enää tule uutta versiota eikä näitä ongelmia korjata Internet Exploreriin. Microsoftin uudesta Edge-selaimesta nämä ongelmat on jo korjattu. (Microsoft 2015; Walton 2016b.)

Selaintuen puolesta Flexible Box -moduuli voidaan ottaa käyttöön verkkosivustojen kehityksessä. Jos esimerkiksi Internet Explorer versioiden 8 ja 9 tukeminen on tarpeellista, voidaan tällöin ottaa käyttöön Flexibility JavaScript-kirjaston. Tämä kirjasto simuloi Flexible Box -moduulin toimintaa vanhemmille selaimille. (Atkins 2014; Lovett 2015.)

6 Ohjelmistokehyksen toteutus

Teoriaosuutta tehdessä tuli selväksi, että siirtyminen pelkästään uusien moduulien käyttöön ei olisi järkevää, koska selaintuki on vielä vajavainen. Päätimme toteuttaa ohjelmistokehyksen, joka tukisi sekä vanhanmallista ruudukkorakennetta että uusia moduuleja. Tämä toteutettiin laajentamalla olemassa olevaa CSS-ohjelmistokehystä.

Vanhaa ohjelmistokehystä laajentaessa ei tullut isoja ongelmia yhteensopivuuksien kanssa, sillä uudet moduulit toimivat täysin itsenäisinä vanhoista osista riippumatta. Nimeämiskäytännöissä täytyi huomioida vain, että vanhan mallin mukaisille ruudukkorakenteille oli varattu tiettyjä CSS-luokkanimiä.

Vanhan ohjelmistokehyksen laajentaminen uuden toteuttamiseen sijaan myös tehosti kehitystä, koska ohjelmiston tiedostorakenne ja arkkitehtuuri oli mietitty valmiiksi ennalta. CSS-esikäsittelijänä vanhassa ohjelmistokehyksessä toimi Sass, jonka käyttöä jatkettiin edelleen. Sass osoittautui myös erittäin hyödylliseksi, koska sen avulla pystyttiin luomaan omia funktioita moduulien käyttöön.

Yksi usein esille nouseva keskustelunaihe teoriaosuutta tehdessä oli se, että Grid Layout ja Flexible Box -moduuleja pystyy käyttämään toistensa korvikkeena. HTML-standardin kehitysryhmässä vaikuttava Tab Atkins Jr. (2013) suosittelee kuitenkin käyttämään Grid Layoutia pääasiassa sivuston ylätason elementtien sijoitteluun ja Flexible Boxia sisällössä toistuvien elementtien määrittelyyn. Ohjelmistokehyksen elementtejä ja käytäntöjä suunnitellessa tämä suositus havaittiin hyväksi.

6.1 Grid Layout

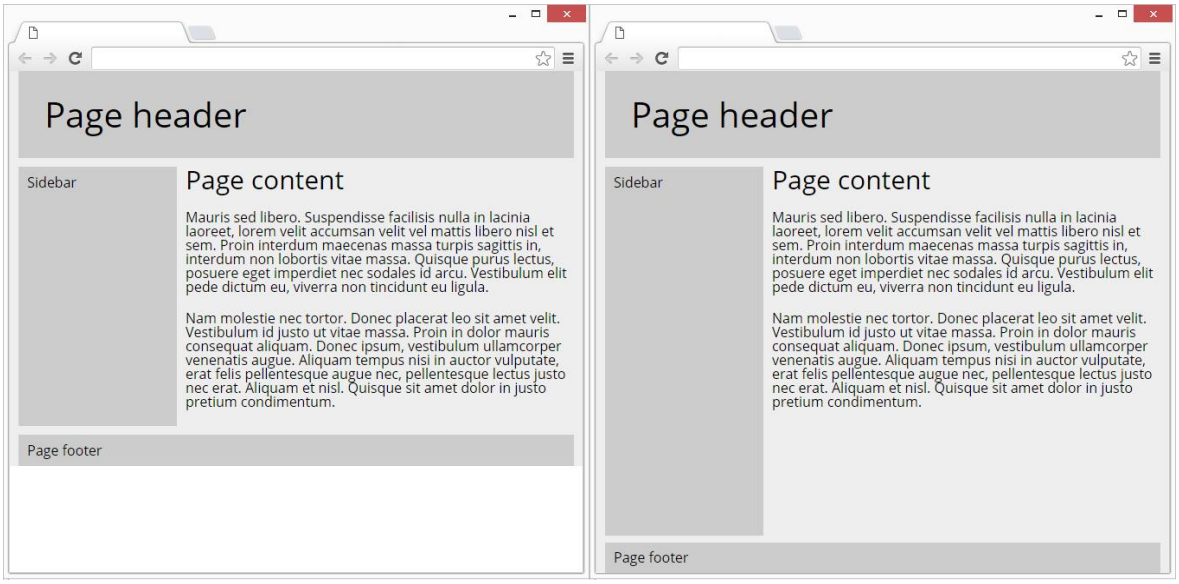
Grid Layout -moduulissa yksi suurimmista kysymyksistä oli syntaksi, jolla ulkoasu kannattaa toteuttaa. Numeraalinen määrittely huomattiin nopeasti selkeämmäksi suuria palsta- ja rivimääriä (yli 6) käytettäessä verrattuna nimettyihin ruutualueisiin. Toisaalta nimetyt ruutualueet ovat helppolukuisia, jos rivi- ja palstamäärä oli pieni.

Useimmiten palsta- ja rivimäärän ei tarvitsisi olla verkkosivuilla kovin suuri, mutta nykymalliset ruudukkorakenteet ovat vieneet suunnittelu- ja ohjelmointityötä joko 12, 16 tai 24 palstan tyyliin. Tämän takia uusilla käytännöillä haluttiin mahdollistaa helposti myös suurempien palsta- ja rivimäärien toteutus.

Molempien numeraalisten ja nimettyjen ruutualueiden hyvien puolien yhdistämiseksi ratkaisuksi muodostui kaksi Sass-mixiniä. Näiden avulla nimettyjen ruutualueiden syntaksissa alueen nimeä voidaan toistaa haluttu määrä ja määrittää yksi alue useamman radan kokoiseksi. Tällöin vältytään toistamasta määrityksessä alueen nimeä 24 kertaa alueen ollessa koko ruudukkorakenteen levyinen.

Ratkaisussa palstojen ja rivien määrä voitiin myös jättää määriteltäväksi tapauskohtaisesti. Palstojen oletusleveydeksi määriteltiin arvo ”1fr”, jolloin kaikkien palstojen leveys on suhteessa palstojen määrään.

Rivien oletuskooksi määriteltiin ”auto”, jolloin ne venyvät sisällön korkeuden mukaan. Tämä toimi lukuun ottamatta tilannetta, jossa sivun sisältötekstin ollessa lyhyt alatunniste jäi ruudun keskelle (kuva 31). Tämä tilanne voitiin ratkaista määrittämällä ruudukon minimikorkeudeksi ”100%” ja sisältöalueen rivin kooksi ”1fr”, jolloin kyseinen rivi täytti loput jäljellä olevasta ruudusta.



Kuva 31. Alatunnisteen kiinnittäminen ruudun alalaitaan (eng. sticky footer) on perinteinen haaste monilla verkkosivuilla

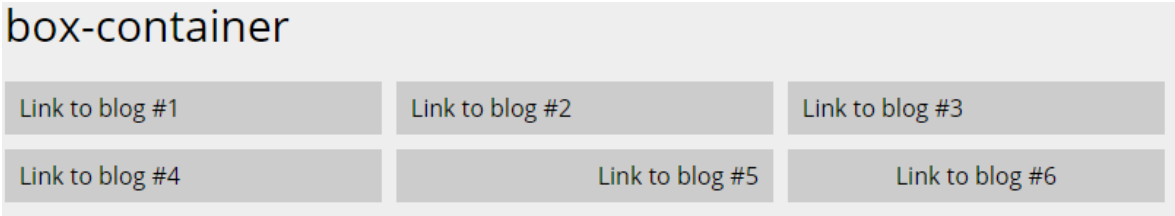
Grid Layoutin ja luotujen mixinien avulla saatiin korjattua myös muita perinteisiä ruudukkorakenteissa esiintyviä ongelmia. Ylimääräisiä elementtejä ja paisuneita luokkamäärityksiä ei enää tarvita, kun ulkoasusijoittelut saadaan toteutettua paremmin CSS-koodin puolella. Responsiivista toteuttamista helpottaa myös media queryjen käyttö, joiden avulla jokaisessa ruutukoossa ruutualueet saadaan sijoiteltua uudelleen.

6.2 Flexible Box

Flexible Box -moduuli osoittautui hyvin erilaiseksi käyttötarkoitukseltaan ja luonteeltaan Grid Layout -moduuliin verrattuna. Grid Layout -moduulin käyttöön oli mahdollista määrittää yksi tapa ja syntaksi, kun taas Flexible Box -moduulilla rakennetaan toisistaan hyvinkin paljon poikkeavia käyttöliittymäkomponentteja.

Tästä johtuen moduulin käyttöönotossa päädyttiin toteuttamaan ohjelmistokehykseen valmiita komponentteja, jotka ovat olleet aikaisemmin haasteellisia toteuttaa ilman Flexible Box -moduulia. Ideat komponentteihin kerättiin asiakasyritys G-Worksin toteuttamilta sivustoilta, Philip Waltonin Solved By Flexbox -projektista (2016a) sekä Roger Flanaganin blogikirjoituksesta (2015).

Flexible Box -moduulin yksi luontevimmista käyttötarkoituksista on toistuvien elementtien listaaminen. Tämänkaltaisista toistuvista elementeistä esimerkkejä ovat muun muassa henkilöstö- ja uutislistaukset. Näiden toteuttamiseksi ohjelmistokehykseen luotiin mixin nimeltä ”box-container” (kuva 32), joka luo elementistä ylätason säiliön Flexible Box -moduulille ja antaa sisältöelementeille tarpeelliset flex-kokomääritykset. Mixinille annetaan parametreinä sisältöelementtien välinen marginaali sekä eri ruutuko’oissa vierekkäin näytettävien elementtien määrä.



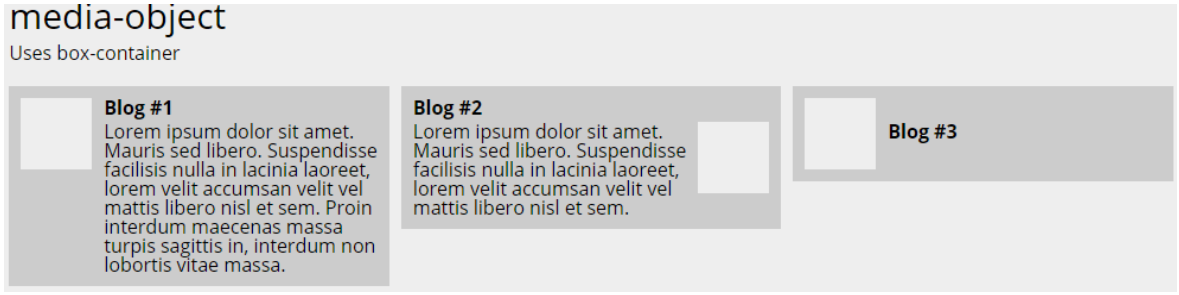
Kuva 32. ”Box-container”-mixinillä luodaan Flexible Box -moduulin mukainen elementti, jonka sisälle sisältöelementit asettuvat

Useasti sisältöelementeille halutaan määrittää joko minimi- tai kiinteä korkeus, mikä onnistuu määrittämällä se suoraan ”box-container”-elementin sisältöelementeille. Kolmas vaihtoehto on määrittää elementtien korkeus suhteellisena niiden leveyteen. Tätä tarkoitusta varten suunniteltiin ”box-container-ratio”-mixin (kuva 33), joka käyttää pohjana ”box-container”-mixiniä. Tälle annetaan samat parametrit kuin ”box-container” mixinille, mutta lisäksi määritetään elementin korkeuden suhde leveyteen.

Kuva 33. Kuvassa on ”box-container-ratio” mixinillä luotu alue, jossa sisältöelementtien korkeuden suhde leveyteen on 0.75

Flexible Box -moduuli on hyödyllinen myös yllä kuvatuissa tilanteissa sisältöelementtien oman sisällön asettelussa. Philip Walton (2016a) tuo esille Flexible Boxin hyödyn ”media-object”-elementeissä. Nimi viittaa siihen, että kyseiset elemenit sisältävät vierekkäin olevat tekstipalstan ja mediaelementin, joka voi olla esimerkiksi kuva, ikoni tai video. Yleisimpänä esimerkkinä näistä mainitaan usein Facebookin yksittäisten kommenttien asettelu.

”Media-object”-elementtejä varten ohjelmistokehykseen luotiin samanniminen mixin, joka määrittää elementistä Flexible Box ylätason elementin (kuva 34). Antamalla elementille luokan ”reverse” vaihtaa mediasisältö paikkaa elementin oikeaan laitaan. Sisältöjen keskittäminen onnistuu CSS-luokalla ”middle”, joka voidaan antaa joko ylätason elementille tai yksittäisille media- ja sisältöelementeille.



Kuva 34. Kuvan ”media-object”-elementeissä on kuva ja tekstialue, joiden keskinäinen sijainti vaihtelee

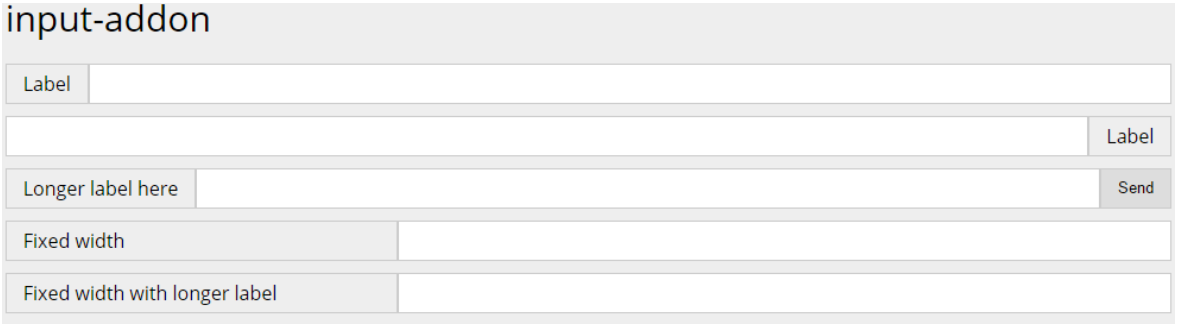
Usein verkkosivustojen käyttöliittymissä halutaan myös lisätä vastaavanlaisten elementtien alalaitaan jonkinlainen toimintakutsu tai graafinen elementti. Perinteisesti tämä on toteutettu ”display: absolute”-määritystä käyttäen, mikä tosin usein on johtanut ongelmiin tekstin korkeuksien suhteen. (Flanagan 2015.)

Tarkoituksena oli luoda tätäkin toiminnallisuutta varten oma mixin CSS-ohjelmistokehykseen, mutta Flexible Boxin avulla toiminnallisuus voidaan toteuttaa anta-

malla CSS-määritys ”margin-top: auto” alalaitaan haluttavalle elementille (kuva 35). Tällöin mixin olisi käsittänyt vain yhden rivin CSS-määrityksen ja jäänyt turhaksi.

Kuva 35. Elementti voidaan pakottaa ylätason elementin alalaitaan Flexible Box -moduulin ”margin-top: auto” määrityksen avulla

Näiden lisäksi Flexible Box elementit on koettu hyödyllisiksi lomake-elementtien asetteluissa. Lomake-elementtien suurin ongelma on ollut niiden nimiöiden (eng. label) ja itse kenttien sijoittaminen vierekkäin riippumatta nimiöiden leveydestä. CSS-määritys ”display: table-cell” tarjoaa yhden ratkaisun, mutta siinäkin on omat ongelmansa. Flexible Box -ratkaisulla saadaan toteutus, jossa nimiöt voivat olla joko kiinteän kokoisia tai mukautua tekstin pituuden mukaan (kuva 36). (Walton 2016a.)



Kuva 36. Flexible Box -moduulilla saadaan toteutettua lomakkeiden asettelut vaivattomasti

Ohjelmistokehykseen luotiin tätä varten oma mixin, joka määrittää lomakekentän ja nimiön ylätason elementin käyttämään Flexible Box -moduulia. Tämän lisäksi lomakekentälle asetetaan määritys ”flex: 1”, jonka ansiosta kenttä venyy täyttämään loput käytössä olevasta leveydestä.

6.3 Käyttöönotto

Toimeksiantajan kanssa käydyissä keskusteluissa opinnäytetyön tuloksia tullaan ottamaan käyttöön todennäköisesti vaiheittain. Flexible Box -moduulilla toteutettavat elementit pystytään ottamaan käyttöön jo tässä vaiheessa, kun taas Grid Layout -moduulin käyttöönotto pitää jättää odottamaan parempaa tukea selainvalmistajilta. Tämän lisäksi tuki vanhemmalle ruudukkorakenteelle säilytetään ohjelmistokehyksessä vielä määräämättömän ajan.

Käyttöönotossa tulee huolehtia erityisesti riittävästä koulutuksesta sekä graafikoille että ohjelmistosuunnittelijoille. Nyt toteutetut käytännöt ja Sass-työkalut kannattaa myös dokumentoida nykyistä laajemmin sekä luoda lisää käytännön esimerkkejä niiden käytöstä.

Moduulien käyttö ja niihin soveltuvat parhaat käytännöt ovat vielä lapsenkengissä, jolloin koulutusta kannattaa kokeilla myös työpajatyyppisenä. Tällöin olisi mahdollista hyödyntää yrityksestä löytyvää ammattitaitoa ja löytää uusia käyttötarkoituksia sekä käytäntöjä moduulien hyödyntämiseksi.

7 Pohdinta

Uudet HTML-moduulit ovat aiheina hyvin ajankohtaisia, vaikka Grid Layout -moduuli ei olekaan vielä täysin valmis käyttöönotettavaksi. Aiheen ympärillä olevan positiivisen keskustelun perusteella uudet moduulit ovat tulevaisuudessa merkittävä tekijä verkkosivustojen toteutuksissa. Tämän muutoksen johdosta kehittäjät tarvitsevat jo nyt valmiita käytäntöjä ja ohjelmistokehyksiä uusien moduulien käyttöön.

Uusien tekniikoiden käyttöönotto on kuitenkin haasteellista yrityksessä, jossa on useampia ohjelmistosuunnittelijoita sekä useampia jatkuvia projekteja. Ohjelmistokehystä käyttöönottaessa G-Worksissa tulee huomioida muun muassa vanhojen sivustojen toiminta ja niiden jatkokehitys sekä henkilöstön koulutus.

Ammatillisesti opinnäytetyö oli erittäin mielenkiintoinen, sillä nämä aiheet ja ongelmat ovat jatkuvasti esillä päivittäisessä työssä verkkosivustojen kehityksessä. Opinnäytetyön tekeminen antoi arvokasta kokemusta moduulien käytöstä konkreettisessa ohjelmointityössä. Moduulien kehittyessä jatkuvasti niiden aktiivista seuraamista täytyy jatkaa, jotta osaaminen ja tieto eivät vanhene.

Opinnäytetyöprojekti käynnistyi teoriaosuuden osalta keväällä 2015, mutta valmistuminen viivästyi kuitenkin alkuperäisestä aikataulusta työ- ja perhekiireistä johtuen. Todennäköisesti viivästyminen oli kuitenkin hyväksi opinnäytetyön tasolle, sillä vuoden aikana moduulit olivat kehittyneet eteenpäin ja uutta materiaalia oli tullut runsaasti.

Teoriaosuudessa haasteen muodosti termien kääntäminen, koska aiheena olleista moduuleista ei vielä löydy juurikaan suomenkielistä aineistoa. Koin kuitenkin, että termit ovat hyvä kääntää ja jättää samalla englanninkielinen versio esille. Asiaa verkosta haettaessa suomenkielistä termistöä löytyi etenkin yliopistoilta, mutta termistöt olivat joko aihetta sivuavia tai jo vanhentuneita.

Itse CSS-ohjelmistokehyksen toteuttamiseen löytyi monia erilaisia ratkaisumalleja ja suunta myös vaihtui projektin aikana muutamaan otteeseen. Toteutuksen lopputulos vaikuttaa tässä vaiheessa erittäin potentiaaliselta, mutta lopullinen arviointi jää odottamaan käyttöönottoa oikeissa asiakasprojekteissa. Jatkokehityksen kannalta valittuihin ratkaisuihin ei myöskään kannata sitoutua liiaksi, vaan tarkastella kokonaisuutta hyvinkin kriittisesti.

Lähteet

Albanesius, C. 2010. Google's New Rule: Mobile First. Luettavissa: http://www.pcmag.com/article2/0,2817,2359752,00.asp. Luettu: 17.11.2015.

Andrew, R. 2014.CSS3 Layout Modules, 2nd edition. edgeofmyseat.com. Maidenhead.

Andrew, R. 2015. Three years with CSS Grid Layout. Luettavissa: https://rachelandrew.co.uk/archives/2015/11/03/three-years-with-css-grid-layout/. Luettu: 13.3.2016.

Andrew, R. 2016a. Current Browser/Rendering Engine Information. Luettavissa: http://gridbyexample.com/browsers/. Luettu: 13.3.2016.

Andrew, R. 2016b. Get ready for CSS Grid Layout. A Book Apart. New York.

Atkins, T. 2013. Relationship with Flexbox. Luettavissa: http://lists.w3.org/Archives/Public/www-style/2013May/0114.html. Luettu: 13.3.2016.

Atkins, T. 2014. Present and Future of CSS Layout. Katsottavissa: https://vimeo.com/98746172. Katsottu: 13.3.2016.

Atkins, T., Etemad E. & Atanassov R. 2015. CSS Grid Layout Module Level 1. Luettavissa: http://www.w3.org/TR/2015/WD-css-grid-1-20150917/. Luettu: 19.3.2016.

Atkins, T., Etemad E. & Atanassov R. 2016. CSS Flexible Box Layout Module Level 1. Luettavissa: http://www.w3.org/TR/2016/CR-css-flexbox-1-20160301/. Luettu: 19.3.2016.

Awwwards 2013. What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. Luettavissa: http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html. Luettu: 16.11.2015.

Bootstrap 2015a. Bootstrap CSS. Luettavissa: http://getbootstrap.com/css/. Luettu: 19.11.2015.

Bootstrap 2015b. Bootstrap Components. Luettavissa: http://getbootstrap.com/components/. Luettu: 19.11.2015.

Bootstrap 2015c. Bootstrap JavaScript. Luettavissa: http://getbootstrap.com/javascript/. Luettu: 19.11.2015.

BuiltWith 2016a. Design Framework Usage. Luettavissa: http://trends.builtwith.com/docinfo/design-framework. Luettu: 3.3.2016.

BuiltWith 2016b. jQuery Usage Statistics. Luettavissa: http://trends.builtwith.com/javascript/jQuery. Luettu: 3.3.2016.

Caniuse.com 2016a. CSS Grid Layout. Luettavissa: http://caniuse.com/#search=Grid%20layout. Luettu: 25.11.2015.

Caniuse.com 2016b. Flexible Box Layout Module. Luettavissa: http://caniuse.com/#search=flexbox. Luettu: 25.11.2015.

Cannon, T.2011. Using the 960 Grid System as a Design Framework. Luettavissa: http://webdesign.tutsplus.com/articles/using-the-960-grid-system-as-a-design-framework--webdesign-2036. Luettu: 19.11.2015.

Cederholm, D. 2013. Why Sass? Luettavissa: http://alistapart.com/article/why-sass. Luettu: 19.11.2015.

Coyier, C. 2012. “Old” Flexbox and “New” Flexbox. Luettavissa: https://css-tricks.com/old-flexbox-and-new-flexbox/. Luettu: 13.3.2016.

Davis, T.2012. Building a Nested Responsive Grid with Sass & Compass. Luettavissa: https://www.viget.com/articles/building-a-nested-responsive-grid-with-sass-compass. Luettu: 19.3.2016.

Flanagan, R. 2016.CSS Issues Solved with Flexbox (No More Magic Numbers). Luettavissa: https://about.zoosk.com/en/engineering-blog/css-issues-solved-with-flexbox-no-more-magic-numbers/. Luettu: 7.5.2016.

Gasston, P. 2011. How We'll Layout Websites in 2016. Katsottavissa: https://vimeo.com/18999428. Katsottu: 17.11.2015.

GitHub2015. CSS preprocessors Luettavissa: https://github.com/showcases/css-preprocessors. Luettu: 19.11.2015

Grigsby, J. 2014. Defining Responsiveness. Luettavissa: http://blog.cloudfour.com/defining-responsiveness/. Luettu: 22.11.2015.

Harbour, S. 2013. The device-agnostic approach to responsive design. Luettavissa: http://www.webdesignerdepot.com/2013/01/the-device-agnostic-approach-to-responsive-design/. Luettu: 28.11.2015.

House, C. 2016. A Complete Guide to Grid. Luettavissa: https://css-tricks.com/snippets/css/complete-guide-grid/. Luettu: 7.5.2016.

Johnson, J. 2012. The Top 10 Web Design Buzzwords and Hot Topics for 2012. Luettavissa: http://designshack.net/articles/webstandards/the-top-10-web-design-buzzwords-and-hot-topics-for-2012/. Luettu: 22.11.2015.

Johnson, J. 2013. Mobile First Design: Why It's Great and Why It Sucks. Luettavissa: http://designshack.net/articles/css/mobilefirst/. Luettu: 17.11.2015.

Keto, J. 1.4.2016. Toimitusjohtaja. Haastattelu. Helsinki.

Kizler, D. 2013. What is Responsive, Adaptive and Fluid Design? Defining Terms in a Multi-Device World. Luettavissa: http://www.hyperarts.com/blog/what-is-responsive-adaptive-and-fluid-design-defining-terms-in-a-multi-device-world. Luettu: 19.11.2015.

Kramer, J. 2014. Responsive Design Frameworks: Just Because You Can, Should You? Luettavissa: http://www.smashingmagazine.com/2014/02/19/responsive-design-frameworks-just-because-you-can-should-you/. Luettu: 16.11.2015.

Lovett, T. 2015. Flexibility: Flexbox support for Internet Explorer. Luettavissa: http://10up.com/blog/2015/flexibility-flexbox-ie/. Luettu: 13.3.2016.

Marcotte, E. 2010. Responsive Web Design. Luettavissa: http://alistapart.com/article/responsive-web-design. Luettu: 19.11.2015.

Marcotte, E. 2011. Responsive Web Design. A Book Apart. New York.

Mehrabani A. & Emrani H.2016. A Collection Of Best Front End Frameworks For Faster&easier Web Development. Luettavissa: http://usablica.github.io/front-end-frameworks/compare.html. Luettu: 19.11.2015.

Meyer, E. 2016. Subgrids Considered Essential. Luettavissa: http://meyerweb.com/eric/thoughts/2016/01/15/subgrids-considered-essential/. Luettu: 19.3.2016.

Microsoft 2014. Update CSS Grid. Luettavissa: https://wpdev.uservoice.com/forums/257854-microsoft-edge-developer/suggestions/6514853-update-css-grid. Luettu: 13.3.2016.

Microsoft 2015. Min-height and flexbox (flex-direction:column) don't work together in IE 10 & 11-preview. Luettavissa: https://connect.microsoft.com/IE/feedback/details/802625/min-height-and-flexbox-flex-direction-column-dont-work-together-in-ie-10-11-preview. Luettu: 19.3.2016.

PC Magazine 2016. Definition of: device agnostic. Luettavissa: http://www.pcmag.com/encyclopedia/term/63798/device-agnostic. Luettu: 13.1.2016.

Rego, M. 2016. Deep Dive into Grid Layout Placement. Luettavissa: http://blogs.igalia.com/mrego/2016/02/01/deep-dive-into-grid-layout-placement/. Luettu: 13.3.2016.

Ruska, L. 2014. Responsive web design – does not depend on the device. Luettavissa: http://www.web-integration.info/en/blog/responsive-web-design-does-not-depend-on-the-device/. Luettu: 5.11.2015.

Slant 2015. What are the best CSS preprocessors? Luettavissa: http://www.slant.co/topics/217/~what-are-the-best-css-preprocessors. Luettu: 19.11.2015.

Soojian, C. 2015. A Brief History of Responsive Web Design. Luettavissa: http://engage.synecoretech.com/marketing-technology-for-growth/bid/204297/A-Brief-History-of-Responsive-Web-Design. Luettu: 22.11.2015.

StatCounter 2016. StatCounter Global Stats. Comparison from 2010 to 2016. Luettavissa: http://gs.statcounter.com/#all-comparison-ww-yearly-2010-2016. Luettu: 5.5.2016.

Vermilion 2015. Responsive CSS Framework Comparison. Luettavissa: http://responsive.vermilion.com/compare.php. Luettu: 5.5.2016.

Walton, P. 2016a. Solved by Flexbox. Luettavissa: http://philipwalton.github.io/solved-by-flexbox/. Luettu: 7.5.2016.

Walton, P. 2016b. Flexbugs. Luettavissa: https://github.com/philipwalton/flexbugs. Luettu: 7.5.2016.

Walton, T. 2014. Device-Agnostic. Luettavissa: http://trentwalton.com/2014/03/10/device-agnostic/. Luettu: 5.11.2015.

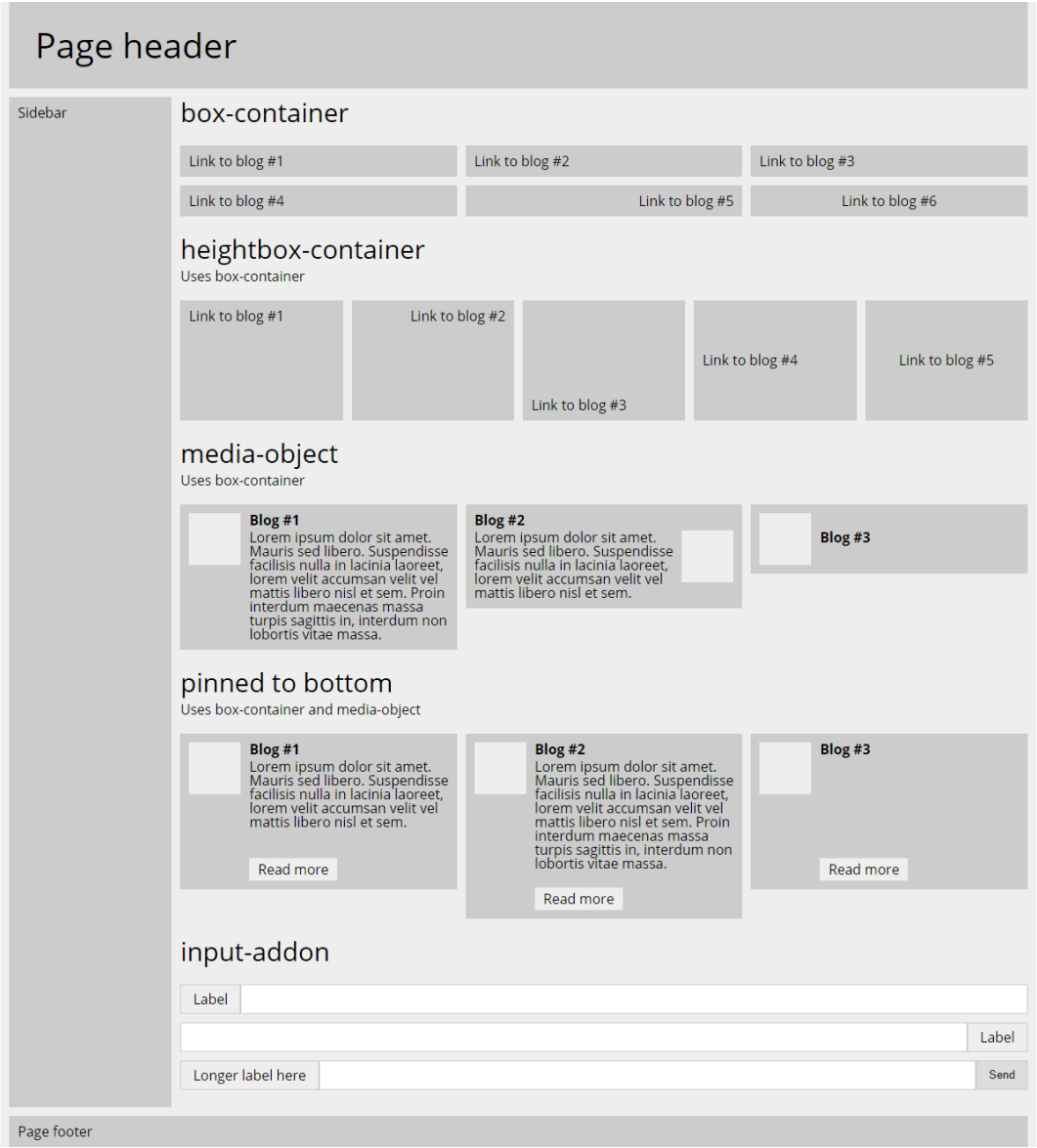
Whittington, R. 2013. Should You Consider a Responsive Website Design? Luettavissa: http://www.rickwhittington.com/blog/should-you-consider-a-responsive-website-design/. Luettu: 16.11.2015.

Williamson, J. 2012. Is Flexbox the Future of Layout? Luettavissa: http://www.slideshare.net/jameswillweb/is-flexbox-the-future-of-layout. Luettu: 13.3.2016.

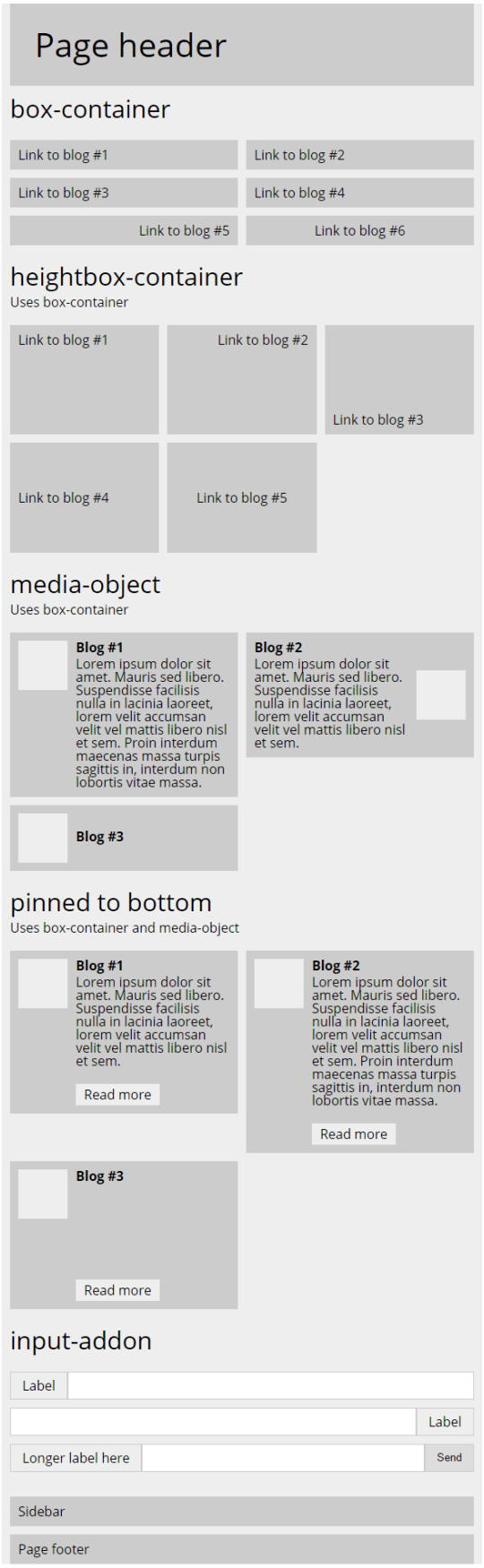
Zeldman, J. 2011. Responsive design. I don't think that word means what you think it means. Luettavissa: http://www.zeldman.com/2011/07/06/responsive-design-i-dont-think-that-word-means-what-you-think-it-means. Luettu: 19.11.2015.

Liitteet

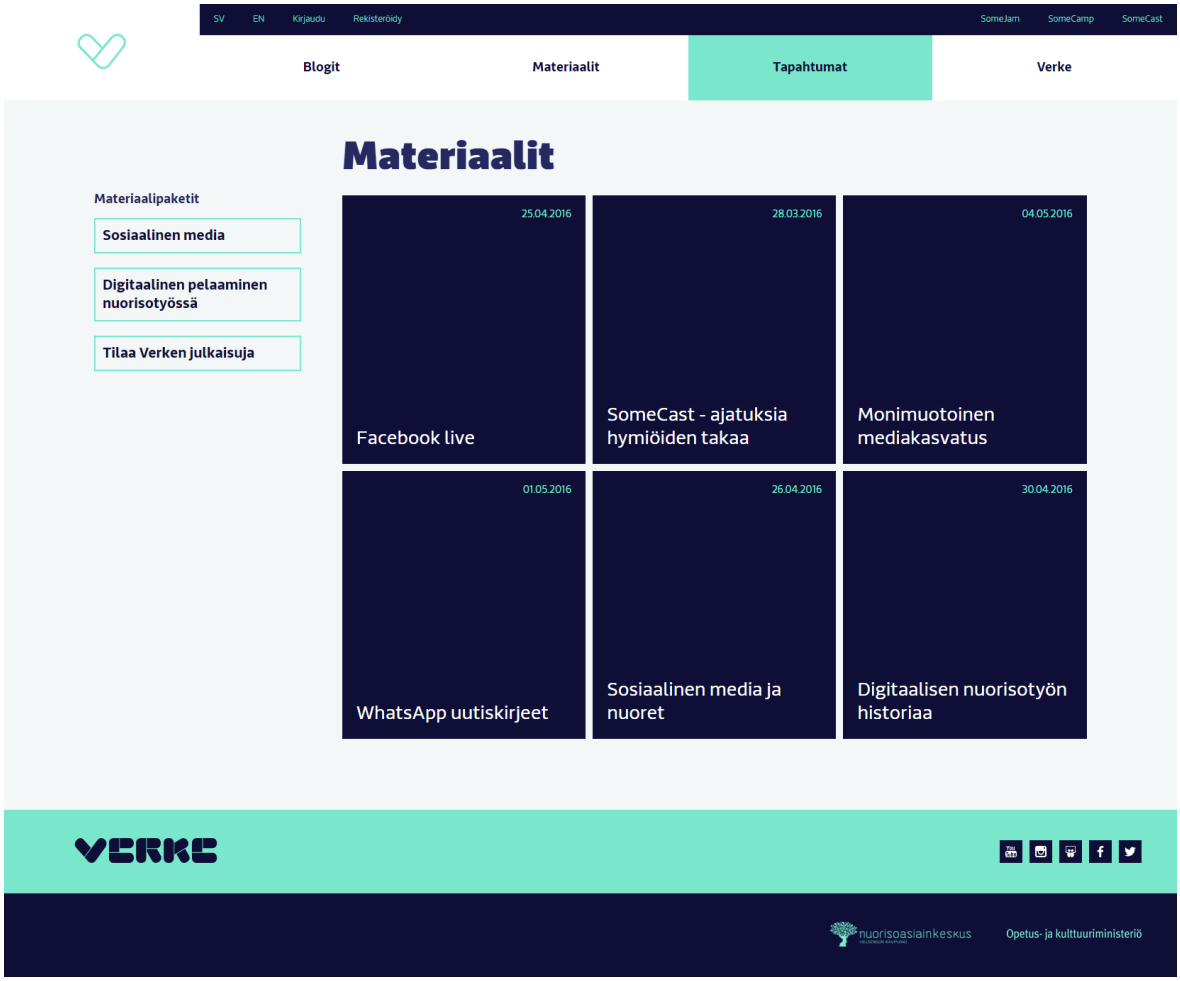
Liite 1. Työpöytäversio uusia moduuleja käyttävästä esimerkkisivustosta



Liite 2. Mobiiliversio uusia moduuleja käyttävästä esimerkkisivustosta



Liite 3. Työpöytäversio Verke.org verkkosivustosta käyttäen uusia HTML-moduuleja



Liite 4. Mobiiliversio Verke.org verkkosivustosta käyttäen uusia HTML-moduuleja