**960 Grid System**

960 Grid System (lyh. 960.gs) on Nathan Smithin kehittämä framework, jota käytetään web-sivujen asettelussa.

960.gs:n avulla verkkosivu jaetaan joko 12, 16 tai 24 sarakkeeseen, ja nimensä mukaisesti kaikissa kolmessa tapauksessa sarakkeiden yhteisleveydeksi tulee 960 pikseliä (kuvio 8).

Frameworkin tarkoituksena on, että verkkosivun leveys pysyy koko näytön levyisenä näyttökoosta huolimatta. Sarakkeiden tarkoituksena on helpottaa tekstien ja kuvien asettelussa. (Smith 2015a.)

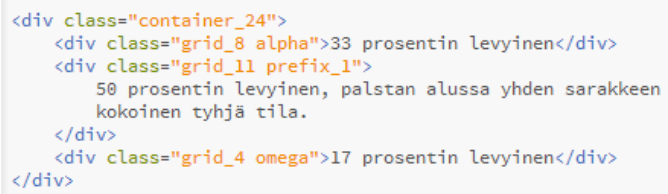


24-sarakkeinen mallipohja

Frameworkissa sivuston rakenne suunnitellaan container-luokan sisälle. Container-luokka määrittelee myös sen, mihin sarakemäärään sivun rakenne tehdään (kuvio 9).

Sivun ollessa esimerkiksi kolmepalstainen lisätään ensimmäiseen palstaan alpha-luokka ja kolmanteen palstaan omega-luokka, sillä muuten sarakkeiden asettelu toimii väärin, mikä rikkoo sivun rakenteen.

Sisällön kuten tekstin alkamis-ja loppumispisteeseen voidaan vaikuttaa prefix-ja suffix-luokilla, jotka lisäävät tyhjän tilan tekstin alkuun tai loppuun. (Roydee 2010.)



KUVIO 9. Esimerkki 24-sarakkeisesta rakennejaosta

Frameworkin käyttö nopeuttaa verkkosivujen suunnittelemista sekä koodaamista, sillä sarakkeiden avulla esimerkiksi web-ohjelmoijan on yksinkertaisempaa päätellä, mistä graafikko haluaa tekstin tarkalleen alkavan ja mihin päättyvän.

960.gs mahdollistaa myös monimutkaisten sivujen tekemisenerilevyisillä sarakkeillaan. (Way 2009.)

960 Grid System -sivu mobiililaitteella

Mobillilaitteella katsottaessa verkkosivun ulkoasu näyttää samalta kuin tietokoneen näytöltä katsottaessa. 960.gs:n avulla verkkosivu skaalautuu koko näytön levyiseksi, mutta ongelmaksi muodostuu sisällön pieneneminen (kuvio 10).

Esimerkiksi viiden tuuman näytöllä (360 pikseliä leveä) frameworkia käyttävän sivun sisältö pienenee lähes kolme kertaa pienemmäksi, minkä vuoksi sivustoa pitää suurentaa (zoomata).



960 Grid System -demosivu älypuhelimen näytöltä

960.gs ratkaisee joitain skaalautuvuuteen liittyviä ongelmia, mutta se ei silti korjaa mobiililaitteella verkkosivujen käyttökokemusta riittävästi.

Skaalautuva ja ruudulle optimoituva verkkopalvelu on mahdollista tehdä responsiivisen verkkosuunnittelun avulla. (Hornor 2013.)

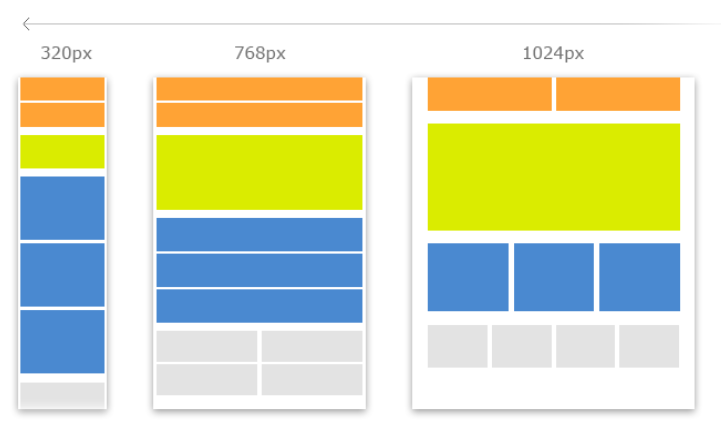
RESPONSIIVINEN VERKKOSUUNNITTELU

Määritelmä

Responsiivisen verkkosuunnittelun avulla verkkosivu mukautuu oikean näköiseksi eri resoluutioiden ja eri näytön kokojen mukaan (kuvio 11).

Tämä tapahtuu CSS3:n media queryjen sekä CSS-tyylitiedostojen avulla, joita käyttämällä on mahdollista vaihtaa sivun kokoa aina tietokoneen suuresta näytöstä pienimmänkin älypuhelimen näytön kokoiseksi (Quinta Group 2015).

Esimerkiksi älypuhelimen näytöllä näkyvä yksipalstainen sisältöelementti voi näkyä tabletin näytöllä kaksipalstaisena sisältöelementtinä (LePage 2015). Tarkoituksena on luoda käyttäjälle verkkosivusta mahdollisimman optimaalinen käyttökokemus, näytön koosta tai päätelaitteesta riippumatta. (Quinta Group 2015.)



Responsiivinen nettisivu kolmella erikokoisella näytöllä (Quinta Group 2015)

Responsiivisen verkkosivusuunnittelun kehittäjänä pidetään Ethan Marcottea. Hän kirjoitti vuonna 2010 artikkelin Responsive Web Design, jossa painotettiin, että mobiililaitteet tulevat yleistymään merkityksellisen paljon muutaman seuraavan vuoden ajan. Marcotte on tiivistänyt responsiivisuuden kolmeen käsitteeseen: ”fluid grids, flexible images and media queries”, joista vain jälkimmäinen käsite oli todella jotain uutta. (Korpela 2012.)

Responsiivisessa verkkosuunnittelussa käytetään usein mobile first -ajatusta, jonka mukaan ei ole aina järkevintä suunnitella ensin työpöytänäkymää ja karsia siitä mobiilinäkymä, vaan tuottaa pääasiat mobiilinäkymälle sopivaksi ja tämän jälkeen laajentaa siitä toimiva kokonaisuus työpöytänäkymään. Mobile first -ajatuksen kautta sivuston suunnittelu ja rakenne tapahtuvat eri tavalla kuin normaalisti ja tällöin sivusto saattaa saada paljon selkeämmän ja toimivamman rakenteen sekä mobiililaitteella ja tietokoneella käytettäessä. (Leiniö 2012.)

Kilpailijat

Responsiivisen verkkosuunnittelun lisäksi mobiilisivujen tekemiseen on kolme vaihtoehtoa:

1.Luoda erillinen mobiilisivu verkkosivuista.

2.Luoda erillinen natiivi-sovellus.

3.Luoda erillinen hybridisovellus.

Erillisen mobiilisivun tekeminen aiheuttaa runsaasti lisätyötä, sillä ylläpidettäväksi tulee 2 erillistä sivustoa, mobiili sekä normaali. Sivustoilla saattaa myös olla täysin eri lähdekoodi, minkä vuoksi niitä ei pysty hyödyntämään keskenään (Leiniö 2012). Mikäli verkkosivun tekemiseen käytetään sisällönhallintajärjestelmää, on responsiivinen verkkosivu parempi vaihtoehto, sillä sen päivittämiseen ei tarvitse kuin yhden järjestelmän ylläpidon. (Dainow 2014.)

Natiivi-sovelluksenhaasteena on, että joka käyttöjärjestelmälle täytyy tehdä oma sovelluksensa. Esimerkiksi Androidille ja Windows Phonelle tehtävät sovellukset eivät ole yhteensopivia. Sovelluksiin tehtävät muutokset on myös tehtävä jokaisen alustan sovelluksiin erikseen, mikä vaatii runsaasti osaamista. (Leiniö 2012.)

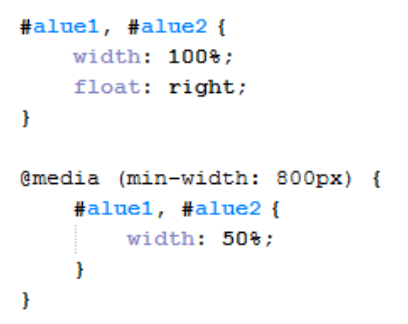
20Hybridisovellus on yhdistelmä HTML5-koodia ja natiivi-koodia. Tämä tarkoittaa sitä, että käyttöliittymä tai osa siitä tehdään käyttämällä HTML5-koodia. Koodi pakataan sovellukseksi käyttämällä esimerkiksi PhoneGap frameworkia. Natiivi-sovellukseen verrattuna hybridisovelluksen etuna on, että se voidaan kääntää yksinkertaisesti eri mobiilialustoille sopivaksi.

Hybridisovelluksen heikkoutena on, että mobiilisovelluksen tavoin joudutaan ylläpitämään 2:ta sivustoa. Lisäksi hybridisovelluksen koodiin joudutaan usein tekemään korjauksia, jotta sovellus kääntyy toimivaksi kaikilla tarvittavilla mobiilialustoilla. (Riippi 2013.)

Responsiivisen verkkosuunnittelun käyttö

Media queryMedia query on CSS3:n osa, jonka avulla voi tehdä kohdistettua CSS-tyylittelyä esimerkiksi näytön koon tai orientaation mukaan. Media queryn tarkoituksena on optimoida sivu hyvännäköiseksi millä näyttölaitteella tahansa (van Hove 2015). Media queryn avulla lajitellaan laitteet ryhmiin, joista jokaiselle tehdään omat asettelunsa. Jokaisen asettelun sisällä noudatetaan joustavaan sommitteluun liittyviä periaatteita. (Korpela 2012.)

Media queryn käytön kannalta breakpoint on oleellinen käsite. Breakpoint on niin sanottu taitekohta, jonka jälkeen sivuston rakenne tai tyyli muuttuu. Breakpoint voidaan määrittää joko CSS-tiedostossa (kuvio 12) tai HTML-tiedoston Head-elementissä (kuvio 13). Esimerkiksi tabletin ruudulla voidaan näyttää enemmän informaatiota asioista kuin mobiilinäytöllä, jolloin mobiililaitteen ja tabletin määritetyssä breakpointissa epäoleellinen informaatio piilotetaan näkyvistä (LePage 2014a).

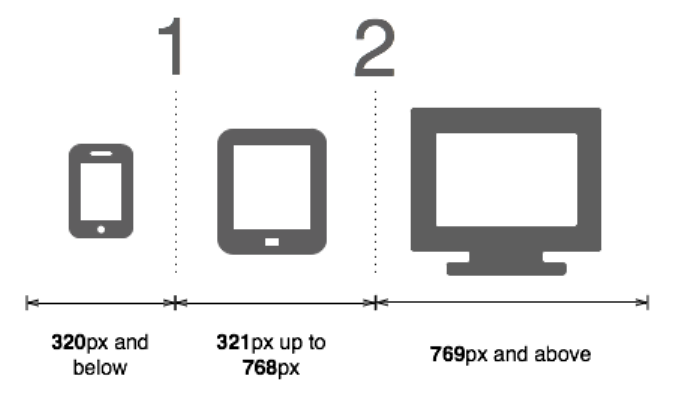


Esim. mediaqueryn käytöstä CSS-tiedoston sisällä



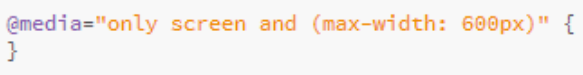
Näyttökokokohtaiset CSS-tiedostomäärittelyt Head-elementissä (LePage 2014a)

Breakpointteja käytetään pääasiassa vain pakollisiin tarkoituksiin, sillä mitä enemmän on taitekohtia, sitä enemmän on tyyliteltävää. Breakpointteja on kuitenkin yleensä vähintään kaksi, joista toinen erottaa älypuhelimen ja tabletin näytön ja toinen tabletin ja tietokoneen näytön (kuvio 14) (LePage 2014b.)



Älypuhelimen, tabletin ja tietokoneen breakpointit (Ford 2013)

Kaikki internet-selaimet eivät tue media querya kokonaisuutena, mutta selaimet yrittävät silti suorittaa sen. Tämä johtuu siitä, että media queryt on pääsääntöisesti suunniteltu niin, että ne toimivat kaikissa moderneissa mobiiliselaimissa (Korpela 2012). Media queryyn voidaan kuitenkin lisätä erillinen esto ”only” (kuvio 15). Eston avulla vanhat selaimet ohittavat media queryn määrittelyn kokonaan mutta uudemmat tukevat selaimet pystyvät silti suorittamaan määrittelyn (Knight 2011). Vanhemmat selaimet voidaan kuitenkin ottaa huomioon sopivien JavaScript-koodienavulla. Skriptin tarkoituksena on saada selain lukemaan ja jopa toteuttamaan CSS-koodissa olevat määritykset. Esimerkkinä JavaScript-kirjastosta on Respond.js, joka pystyy suorittamaan esimerkiksi max-width-ja min-width-rajoitukset. (Korpela 2012.)



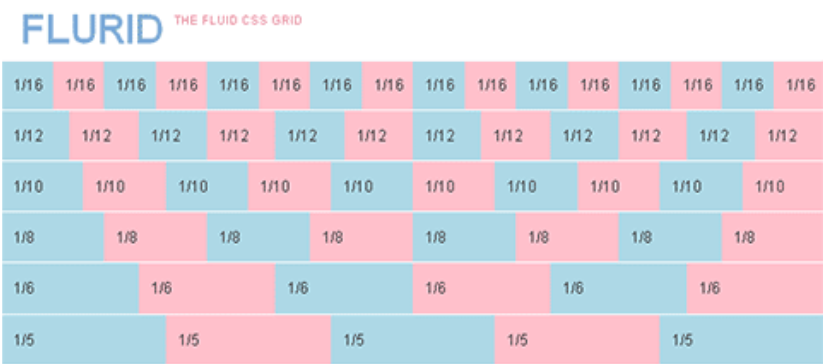
Eston käyttö media queryssa

Viewport

Viewport on HTML-tiedoston meta-tietoihin laitettava tagi, jolla voidaan määrittää, miltä verkkosivu näyttää mobiililaitteen selaimessa. Viewportin avulla voidaan säätää sivun leveyttä, korkeutta ja skaalautuvuutta. Viewport toimii lähes kaikissa moderneissa mobiiliselaimissa. (Ala-Äijälä 2012.)Viewportin leveys-parametrille voidaan asettaa numeerinen pikseliarvo, mutta sen arvoksi voidaan asettaa myös ”width = device-width”, jolloin sivu näkyy koko näytön levyisenä eli skaalautuvana. Skaalautuvuudelle voidaan myös asettaa rajat tai se voidaan estää kokonaan. (Ala-Äijälä 2012.)

Fluid Grid

Fluid Gridin eli joustavan taittopohjan idea on samankaltainen kuin 960.gs:ssä eli sivu jaetaan sarakkakkeisiin. 960.gs:n vakiomittaisten sarakkeiden sijaan Fluid Gridissä käytetään suhteellisen mittaisia sarakkeita (kuvio 16). (Knight 2009.)



Esimerkkitaulu suhteellisesta jakamisesta (Flurid 2014)

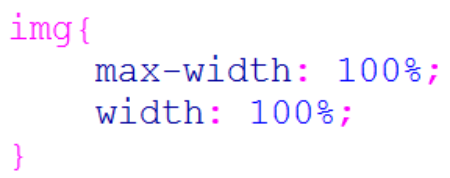
Fluid Gridin lisäksi voidaan käyttää Elastic Gridiä eli joustavaa taittopohjaa sekä Fixed Gridiä eli määrätyn kokoista taittopohjaa. Elastic Gridin idea on samanlainen kuin Fluid Gridin: se skaalautuu ja mukautuu samalla tavalla, mutta vain tiettyyn rajaan saakka. Elastic Gridillä on maksimi-ja miniarvo, jonka jälkeen taittopohja ei enää veny. Tämän ansiosta verkkosivun sisältö ei voi venyä liian suureksi, vaikka näyttökoko olisikin suuri.

Fixed Grid toimii hyvin samalla tavalla kuin 960.gs-framework. Taittopohja pysyy tietyn kokoisena, eikä se skaalaudu sisältönsä tai näytön koon mukaan. Fixed Gridin avulla voidaan kuitenkin käyttää breakpointteja, jolloin sivuston ulkoasu muuttuu tiettyjen taitekohtien mukaan. (Knight 2009.)

Fluid Images

Fluid Images eli joustavat kuvat ovat kuvia, jotka skaalautuvat näytön koon mukaan. Responsiivisessa verkkosuunnittelussa on tärkeää, että kuvat ovat skaalautuvia, sillä muuten kuvat voivat aiheuttaa ongelmia käyttöliittymän rakenteessa ja hajottaa sen. Skaalautuvuus asetetaan CSS-tiedostossa, jossa kuva-attribuutin maksimileveydeksi asetetaan 100 prosenttia.

Kyseinen määritys ei toimi Internet Explorer-selaimella, mutta se on mahdollista korjata laittamalla kuva-attribuutin kokonaisleveydeksi 100 prosenttia (kuvio 17).(Leiniö 2012.)



Skaalautuvien kuvien määritykset CSS-tiedostossa

Sisällön piilottaminen

Mobiilinäytölle ei mahdu niin paljon sisältöä kuin esimerkiksi tietokoneen näytölle, vaikka elementtejä pienentäisi kuinka paljon. Ongelma voidaan ratkaista sijoittamalla sisältö uudelleen mobiilisivulle mutta tämä saattaa pidentää sivua huomattavasti. (Leiniö 2012.)Toinen vaihtoehto on piilottaa vähemmän tärkeät elementit kokonaan mobiilisivusta (kuvio 18).

Elementtien piilottaminen tapahtuu CSS:n avulla, ja se tapahtuu joko käyttämällä ”visibility: hidden-” tai ”display = none” -määritettä. Ensimmäinen vaihtoehto ainoastaan piilottaa sisällön, mutta jälkimmäinen myös poistaa sisällön. Mikäli sisältöä ei poisteta, sisällön kohdalle jää sille varattu tyhjä tila. (Leiniö 2012.)



Vähemmän tärkeän elementin piilottaminen mobiilisivulta (Weil 2014)

4.3.6

Yleisiä responsiivisuuteen vaikuttavia määrityksiä

Responsiivinen suunnittelu on kokonaisuudessaan laajempi käsite kuin aikaisemmin on käsitelty. On tärkeää, että sivun kokonaisasettelu on suhteessa laitteen kokoon responsiivinen. Lisäksi tulee huomoida, että painikkeet sekä linkit ovat riittävän suuret, jotta niitä voidaan käyttää kosketusnäytöllä. Kuvien tulee olla mahdollisimman pienikokoisia, sillä on huomioitava myös hidasyhteyksisten laitteiden käyttäjät. Lisäksi ylimääräisten ja tarpeettomien kuvien käyttöä tulee välttää, sillä ne vievät runsaasti ylimääräistä tilaa sisällöltä näytöllä. (Korpela 2012.)

Sisällön tulee myös olla tiivis ja tarpeellinen. Mikäli vähemmän tärkeän sisällön haluaa pitää mobiilisivulla, se kannattaa piilottaa niin, että sen voi tarvittaessa saada näkyviin esimerkiksi ”lue lisää” -painikkeen avulla. Sivustossa tulee ottaa myös huomioon fonttivalikoima. Mobiililaitteen fonttivalikoima on melko suppea, ja tämän vuoksi fonttien määrää voi lisätä käyttämällä ladattavia fontteja, kuten esimerkiksi Googlen Web-fontteja. Lisäksi tekstin koon tulee olla riittävän isoa, jottei sivua tarvitse suurentaa jatkuvasti. (Korpela 2012.)

RESPONSIIVISET FRAMEWORKIT

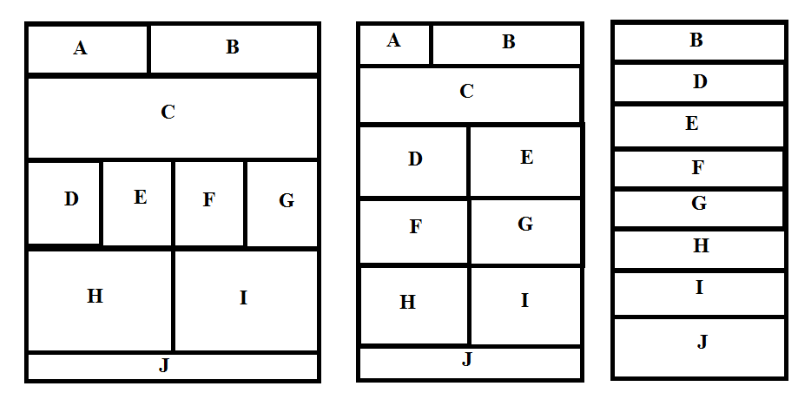
Tässä luvussa käsittellään kolmea eri frameworkia, jotka on valittu vertailuun keskenään.

Unsemantic on 960.gs:n kehittäjän tekemä responsiivinen framework, ja tämän vuoksi se on valittu yhdeksi kolmesta (Smith 2015d).

Toisena frameworkina käsittellään Pure.css-frameworkia, joka on pienikokoinen ja erittäin kevyt framework (Pure 2014).

Kolmantena frameworkina käsitelläänBootstrapia, joka on maailman käytetyin responsiivinen framework. (Bootstrap 2015.)

Jokaista frameworkia varten on tehty oma demosivu, ja niiden rakenne pyritään pitämään samana (kuvio 19). Koska alkuperäinen demosivu on suunniteltu ja tehty 24-sarakkeiseen 960.gs:n pohjaan, joudutaan joissain tapauksissa rakenteessa poikkeamaan hieman, sillä kaikissa frameworkeissa ei ole käytössä 24-sarakkeista jakoa. Mobiilisivun rakenteesta on poistettu kohdat A ja C, sillä tarkoituksena on tuoda näytön koon pienuuden vuoksi sisältö paremmin esille. A-kohdassa on paikka logolle ja C-kohdassa paikka vierityspalkille (slider) tai kuvalle.



Demosivuston rakennemallit

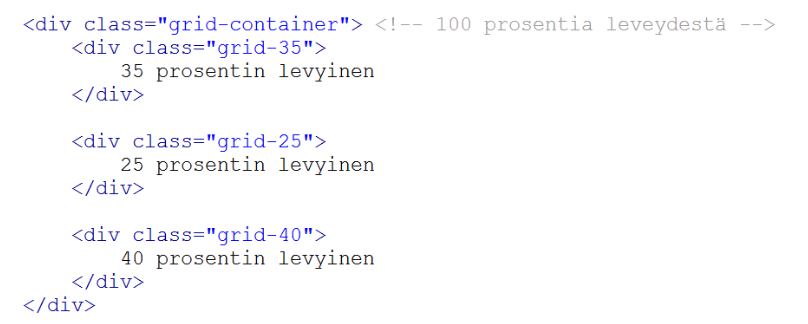
Demosivujen on tarkotus mukautua tietokoneen (kuvio 20), tabletin (kuvio 21) ja älypuhelimen näytölle sopivaksi (kuvio 22). Jokaisessa sivussa käytetään siis ainoastaan kahta breakpointia: yhtä mobiilinäytön ja tabletin näytön ja toista tabletin näytön ja tietokoneen näytön välille. Tarkoituksena

on myös käsitellä, viekö jonkin frameworkin käyttö enemmän aikaa kuin toisen.

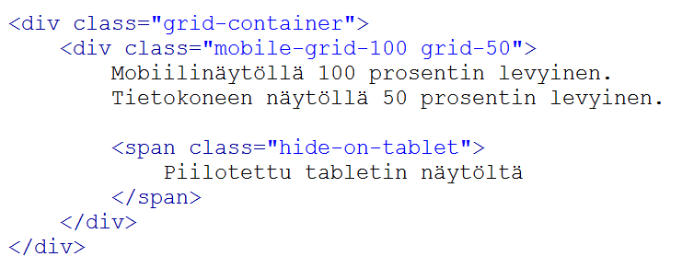
Unsemantic

Kuvaus

Unsemantic on Nathan Smithin kehittämä responsiivinen framework, joka on 960.gs:n seuraaja. Se toimii samankaltaisesti kuin 960.gs, mutta sarakkeiden lukumäärän sijaan se noudattaa prosentuaalista jaottelua (kuvio 23). Jaottelussa sarakkeidenarvot menevät viiden kertotaululla viidestä sataan asti. Unsemantic ymmärtää myös arvot 33 ja 66 eli yksi kolmasosaa ja kaksi kolmasosaa. Toinen eroavaisuus 960.gs:n ja Unsemanticin välillä on gridin leveys. 960.gs:n960 pikselin sijaan Unsemanticin gridin leveys on 1180 pikseliä. (Smith 2015c.)



Esimerkki Unsemanticin käytöstäUnsemanticin avulla voidaan näyttää tietty elementti eri kokoisena eri näyttölaitteella. Sen avulla voidaan myös piilottaa elementtejä tai näyttää tietty elementti vain tietyllä näyttökoolla (kuvio 24). (Smith 2015b.)



Esimerkki eri kokoisista elementeistä sekä elementin piilottamisesta

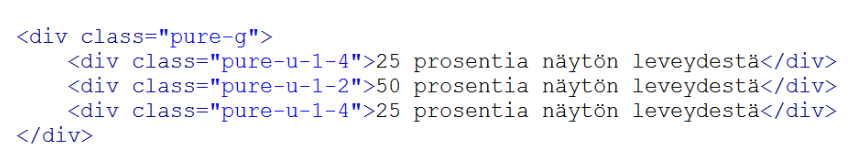
**Unsemanticinkäyttö demosivulla**

Unsemantic vie tilaa tietokoneelta alle yhden megantavun, mikä on kohtalainen koko frameworkille. Unse mantic sisältää valtavan määrän tiedostoja, joista pelkästään CSS-tyylitiedostoja on 34 kappaletta. Dokumentoinnin avulla löytyy lopulta oikeat tyylitiedostot, jotka ovat reset.css ja unsemantic-grid-responsive-tablet.css. Jälkimmäinen tyylitiedosto sisältää breakpointit mobiililaitteen ja tabletin välille sekä tabletin ja tietokoneen näytön välille.

Verkkosivujen tekeminen aloitettiin tietokonenäkymästä. Koska alkuperäiset sivut on tehty 960 pikseliä leveällä gridillä, tuli sivusta hieman leveämpi, sillä Unsemanticin gridin leveys on 1180 pikseliä. Suurimmaksi ongelmaksi koitui ylemmässä sisältöelementissä olevien neljän palstan välit. Alunperin välit tulivat suoraan 960.gs:n tyylimäärityksistä, mutta Unsemanticin määrityksissä kyseiset välit olivat paljon leveämmät, ja tämän vuoksi niiden käyttäminen ei sopinut ulkoasuun. Välien sijaan palstojen oikeaan reunaan asetettiin oikean levyinen valkoinen kehys (border), jonka ansiosta sivun ulkoasu vastasi alkuperäistä.

Pure

KuvausPure on erittäin pienikokoinen res elliset CSS-määritykset, koska Puren tekijöiden mukaan on helpompaa tehdä ponsiivinen framework. Puren tarkoituksena on sisältää vain kaikki ole uusi CSS-määritys kuin päällekirjoittaa olemassa oleva. Myös Pureen liittyvät tyylit on tehty niin,että ne on tarvittaessa helppo päällekirjoittaa (Pure 2014b). Purea käytetään yhdessä Normalize.css:n kanssa. Normalize.css on tiedosto, joka renderöi kaikki elementit johdonmukaisiksi ja modernien standardien mukaisiksi. (Gallagher & Neal 2015.)Puren grid on mahdollista jakaa joko 5 tai 24 osaan. Gridin jaottelu tapahtuu prosentien sijaan murtoluvuilla. Esimerkiksi Puren merkintä 1-4 tarkoittaa 25 % näytön leveydestä (kuvio 25). (Pure 2014b.)



**Esimerkki Puren käytöstä**

Puren CSS-tiedosto sisältää valmiita tyylimäärityksiä esimerkiski painikkeille, taulokoille ja lomakkeille (kuvio 26) (Pure 2014a). Valmiit

31määritykset ovat hyödyksi esimerkiksi graafikolle, joka suunnittelee sivuston ulkoasua, sillä hän voi valmiiksi valita oikean näköisen elementin ja näin vähentää omaa sekä koodaajan työmäärää



Puren määritykset kahdelle napille (Pure 2014)

**Puren käyttö demosivulla**

Purea ei tarvitse ladata tietokoneelle lainkaan, sillä css-tiedoston voi sisällyttää HTML-tiedostoon suoraan verkkosivulta. Puren tiedostot voi myös ladata koneelleen. Tiedostot vievät hieman yli 200 kilotavua tilaa, ja ne ovat suurimmaksi osaksi CSS-tyylitiedostoja lomake-ja taulukkomäärityksiä varten. Vaikka CSS-tiedostoja on lähes yhtä monta kuin Unsemanticissa, on tiedostot nimetty paljon selkeämmin ja ymmärrettävämmin. Puresta käytetään siis vain pure-min.css-sekä normalize.css-tiedostoa.

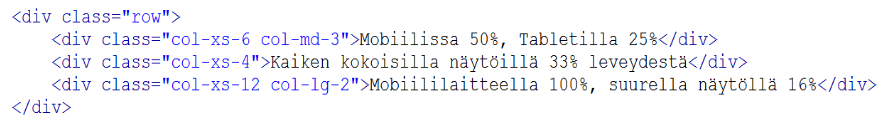
Pure perustuu mobile first -ideaan, minkä vuoksi sivujen tekeminen aloitettiin mobiilinäkymästä. Puren gridin koko on myös koko ruudun levyinen, minkä vuoksi fontteja ja kuvia joutui suurentamaan, jotta ne näkyisivät hyvin. Puressa on mahdollisuus käyttää kolmea breakpointia, mutta suurin näyttökoko laitettiin vastaamaan tietokoneen näytön kokoa.

Puren gridin jaottelu noudattaa 24 sarakkeen jakoa, minkä vuoksi palstojen jako toimi tietokoneen näytöllä samalla tavalla kuin 960 Grid Systemissa. Mobiilisivun tekemisen jälkeen oli yksinkertaista laajentaa sivu vastaamaan tabletille ja tietokoneen näytölle suunniteltua rakennetta. Unsemanticin lisäksi myös Puren ylemmän sisältöelementin välien tekemiseen käytettiin kehyksiä.

Purella sivujen tekeminen oli nopeampaa kuin Unsemanticilla. Mobiilisivun laajentaminen suuremmaksi vei huomattavasti vähemmän aikaa, sillä yhteensopivuusongelmilta vältyttiin.

**Bootstrap**

Bootstrap on maailman suosituin responsiivinen framework. Bootstrap perustuu **mobile first -ideaan**, eli ensin suunnitellaan verkkosivu mobiililaitteelle ja tämän jälkeen laajennetaan se tietokoneen näytölle sopivaksi. Bootstrap on **avoimeen lähdekoodiin perustuva julkaisu** ja sitä **ylläpidetään Githubin kautta**. Puren lisäksi myös Bootstrap käyttää **Normalize.css**:ää. (Bootstrap 2015).Bootstrapin grid on jaoteltu **12 sarakkeen mukaisesti**, ja sen leveys on **maksimissaan 1170 pikseliä**. Bootstrapissa on neljä kokoluokkaa aina erittäin pienestä suureen näyttökokoon. **Jokaiselle kokoluokalle on annettu oma lyhenteensä, ja niitä käytetään sarakkeiden lisäksi nettisivujen teossa** (kuvio 27). (Bootstrap 2015.)



<https://github.com/mdbootstrap/Bootstarters>

<https://mdbootstrap.com/>

<https://startbootstrap.com/template/full-width-pics>

<https://github.com/learning-zone/website-templates> lataa templatesit esim. githubiin ja sieltä Visual Studio Codeen…

**Esimerkki Bootstrapin käytöstä**

Kuten Puressa myös Bootstrapissa on valmiita tyylimäärityksiä taulukoille, lomakkeille ja painikkeille. Frameworkiin on myös tehty runsaasti muita tyylimäärityksiä, minkä vuoksi netissä oleva dokumentointi onkin erittäin laaja (Bootstrap 2015). Runsaat tyylimääritykset ovat osittain myös haitta, sillä mahdollisissa päällekirjoitustilanteissa tulee todella pitää huoli siitä, että oikeat tyylit ovat käytössä

Bootstrapin käyttö demosivulla

Bootstrap on valituista frameworkeista kaikista suurin, sillä sen koko on tietokoneella vähän yli yhden megantavun, mikä ei kuitenkaan ole liian paljon web-ohjelmoinnissa. Tiedostomäärällisesti Bootstrap on taas kaikista pienin, sillä se sisältää ainoastaan neljä CSS-tiedostoa, joista kaksi on pienennettyjä ja kompressoituja versioita alkuperäisistä tiedostoista. Puren tapaan myös Bootstrapin voi sisällyttää nettisivulle internetistä. Bootstrapin tiedostoista käytetään bootstrap.css-sekä normalize.css-tyylitiedostoja.

Bootstrap on toiminnaltaan hyvin samankaltainen kunPure. Puren lisäksi myös Bootstrap perustuu mobile first -ajatukseen, ja täten verkkosivujen tekeminen aloitettiin mobiilisivusta. Nettisivujen laajentaminen onnistui helposti ja myös tässä tapauksessa suurin näytön koko laitettiin vastaamaan tietokoneen näytön kokoa. Suurin ero sivun tekemisessä Pureen oli, että Bootstrap noudattaa ainoastaan 12-sarakkeista jakoa, kun taas Pure noudattaa 24-sarakkeista jakoa. Bootstrap tarvitsee myös toimiakseen JavaScriptin mutta Puren käyttö ei sitä vaadi.

Puren ja Bootstrapin välillä ei ollut juurikaan eroa nettisivujen tekemisessä. Kyseisen nettisivun tekemiseen kului vähän vähemmän aikaa Purella, mutta todennäköisenä syynä on yhtenevä sarakejako 960.gs:n kanssa.

**960 Grid System-frameworkin korvaamiseen Bootstrap on erittäin hyvä vaihtoehto. Se on yksinkertainen käyttää, ja se noudattaa mobile first -ideaa, jonka ansioista verkkosivut toimivat hyvin mobiilissa. Lisäksi Bootstrap tarjoaa graafikolle valmiita malleja taulukoiden ja painikkeiden suunnitteluun, minkä vuoksi graafikon ja web-ohjelmoijan välinen yhteistyö helpottuu ja selkeytyy. Lisäksi modernin frameworkin käyttö saattaa kasvattaa yrityksen positiivista imagoa asiakkaan silmissä.**