TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Informaatika eriala

Mairit Vikat

CSS raamistiku kasutamine veebilehe kujunduse loomisel

Bakalaureusetöö (6EAP)

Juhendaja: MSc Gunnar Nellis

Autor: ……..... „…..” juuni 2010

Juhendaja: ……..... „…..” juuni 2010

Lubada kaitsmisele

Professor: ..... „…..” juuni 2010

Tartu 2010

Sisukord

Sissejuhatus

Levinuim veebilehtede valmistamiseks mõeldud markeerimiskeel on HTML (*HyperText Markup Language*). HTML lehe kujundamiseks kasutatakse CSS (*Cascading Style Sheets*) keelt, mis on mõeldud veebilehe kujunduselementide (ülesehitus, värvid, kirjastiilid) kirjeldamiseks.

Nii nagu programmeerimis- ja skriptimiskeeltes on levinud teegid ehk raamistikud (*frameworks*), mis kujutavad endast valmiskirjutatud koodi kogumikke, on ka veebilehtede kujundamise lihtsustamiseks loodud vastavaid raamistikke. CSS raamistik (*CSS framework*) ehk veebikujunduse raamistik on CSS failidest koosnev teek, mis sisaldab teatud hulga valikuid veebilehe kujundamiseks ja ülesehituseks. CSS raamistikud on mõeldud veebilehe kiiremaks, lihtsamaks ning standarditele vastavamaks kujundamiseks.

Käesolevas töös käsitletakse veebilehtede loomist HTML ja CSS keeli ning vabavaralist kujundusraamistikku Blueprint kasutades. Eesmärgiks on praktiliste näidete varal välja selgitada, millised probleemid võivad tekkida veebilehe loomisel ilma kujundusraamistikuta ja kujundusraamistikuga ning kas veebilehe kujundamisel on otstarbekas kasutada CSS raamistikku. Selleks luuakse antud töö käigus kaks veebilehte – üks lihtsama ning teine keerulisema struktuuriga veebileht, mõlemad nii kujundusraamistikuga kui raamistikuta.

Bakalaureusetöö on jaotatud kuueks osaks, mis tervikuna peaksid andma lugejale lühiülevaate CSS raamistike olemusest ja omadustest ning samuti võiks aidata otsustamisel, kas ning millal veebilehe kujundamisel CSS raamistikku kasutada.

Esimeses ja teises peatükis tehakse ülevaade CSS keelest ning CSS raamistikust - selle olemusest, headest ja halbadest külgedest, levinumatest raamistikest.

Kolmandas osas on välja toodud nõuded veebilehtedele, mida neljandas ja viiendas peatükis luuakse - lihtsama ning keerulisema ülesehitusega veebileht vastavalt raamistikuga ja ilma.

Neljandas ja viiendas peatükis tuuakse välja probleemid, mis ilmnesid veebilehtede kujundamisel vastavalt raamistikuta ning raamistikuga.

Viimases peatükis võrreldakse saavutatud tulemusi vastavalt kolmandas peatükis välja toodud nõuetele.

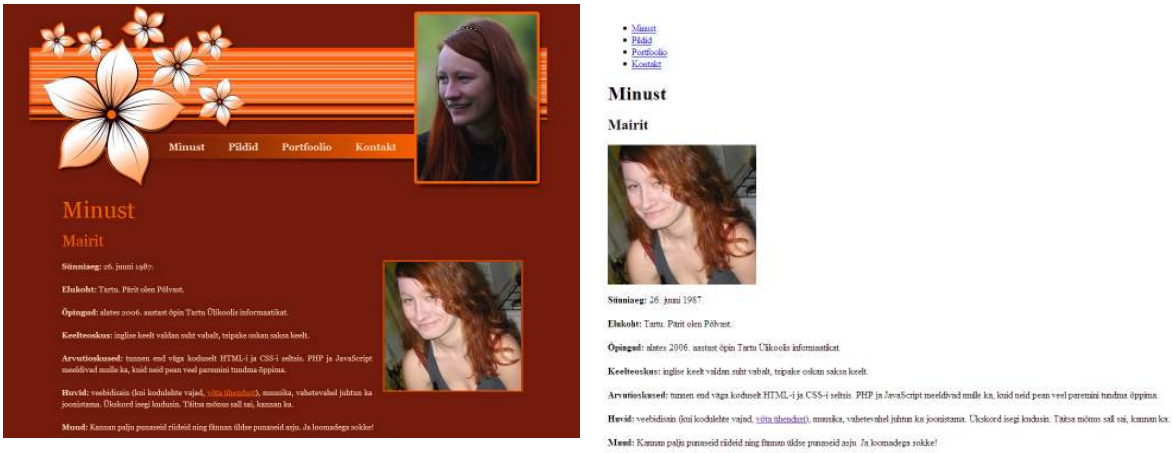
1. Ülevaade CSS keelest

HTML (*HyperText Markup Language*) on märgendkeel veebilehtede loomiseks. HTML annab võimaluse luua struktureeritud dokumente, kasutades teksti kirjeldamiseks struktuurset semantikat nagu pealkirjad, lõigud, nimekirjad, viited, tsitaadid ning teised elemendid. Samuti on võimalik lisada pilte ja objekte ning luua interaktiivseid vorme. HTML dokumendile saab lisada CSS stiilifaile (*Cascading StyleSheet*), millega saab defineerida teksti ning teiste materjalide väljanägemist (värvid, kirjastiilid jms) ning asetust. [16]

CSS stiililehtedega on võimalik kujundada kõikide HTML elementide (tekstid, tabelid, viited, nimekirjad, vormid jms) väljanägemist ning asetust. [4]

Samuti on CSS keele abiga võimalik luua HTML lehekülje ülesehitus (*layout*) - tavaliselt jaotatakse veebilehe sisu teatud alamjaotisteks, näiteks päis, menüü, sisu, jalus. Lehe ülesehitusega saab määrata, kuidas eelmainitud lehekülje osad teineteise suhtes paiknevad. [23]

Näide, milliseks on võimalik veebileht CSS stiilidega kujundada, on toodud joonisel 1.



Joonis 1. Vasakul CSSiga kujundatud veebileht, paremal sama leht ilma CSS määranguteta.

1.1. CSS keele süntaks

CSS koodil on lihtne süntaks, erinevate stiiliomaduste määramiseks kasutatakse inglisekeelseid märksõnu. Stiililеht koosneb reeglite nimekirjast, iga reegel või reeglitekogu omakorda valijast lisast ning loogeliste sulgude { } vahel olevast deklaratsiooniplokist ehk omadus-väärtus paaridest (*property-value pair*). Näide CSS koodi süntaksist:

```
h1 {
    color: white;
    background-color: orange;
}
```

Selles näites on `h1` valija, mis määrab, millisele elemendile järgnev stiil määratakse, ning `color: white;` ja `background-color: orange;` omadus-väärtus paarid, mis määravad elemendi esituse stiili. [4]

Tihtipeale tehakse CSS koodi kirjutades süntaksivigu, mis võivad lehe väljanägemist või funktsioneerimist mõjutada. Kontrollimaks, et CSS koodi süntaks on korrektne, kasutatakse valideerimist. See tähendab võrdlemist teatud reeglitega, millele vastavusel on kood valideeruv ehk ilma vigadeta. CSS koodi saab valideerida näiteks W3C CSS keele valideerimisteenuse veebilehel http://jigsaw.w3.org/css-validator/. Probleemide vältimiseks tuleks eelnevalt aga valideerida ka HTML lehe kood, näiteks W3C märgendkeele valideerimisteenuse veebilehel http://validator.w3.org/.

HTML ja CSS koodi valideerimiseks on mitmeid põhjuseid: korrektne kood aitab kaasa brauserite- ja platvormidevahelisele ühilduvusele, suurendab nähtavust otsingumootorites ning väljendab professionaalsust - veebiarendaja kood ei tohiks põhjustada lehe külastajatele nähtavaid vigu. [11]

1.2. CSS koodi lisamine veebilehele

CSS stiili on võimalik veebilehele lisada kolmel erineval moel:

1. kirjutades otse HTML elemendi sisse, kasutades selleks `style` atribuuti, näiteks
   `<h2 style="color: blue;">Pealkiri</h2>`

2. kirjutades HTML faili päisesse `style` märgiste vahele, näiteks

   ```
   <style type="text/css">h1 { color: red; }</style>
   ```

3. kaasates HTML faili päises link elemendiga välise CSS faili, näiteks
   `<link rel="stylesheet" type="text/css" href="style.css" media="screen" />`

Otse elemendi sisse kirjutatud stiili puuduseks on, et seda ei ole võimalik taaskasutada. See tähendab, et iga kord, kui on vaja sama stiili mõnele teisele elemendile lisada, tuleb stiil selle elemendi `style` atribuudi sisse kirjutada. Samuti peab igal stiilimuudatusel redigeerima kõiki neid HTML märgiseid, kuhu see stiil kopeeritud on. HTML elemendi sisse `style` atribuudina CSS stiili kirjutamist ei soovitata ka seetõttu, et koodi on raske hallata, kuna dokumendi sisu ja väljanägemine pole eristatud.

HTML faili päises `style` märgiste vahel CSS stiili lisamise puuduseks on, et mitmete HTML lehtede samasuguse stiili saavutamiseks tuleb stiilidefinitsioone korrata kõigi HTML lehtedel.

Koodi dubleerimise vältimiseks ning mitmetele HTML lehtedele samasuguse vormingu võimaldamiseks kasutatakse väliseid stiililehti - CSS kood kirjutatakse HTML koodist eraldi faili ning kaasatakse HTML lehele link elemendina dokumendi päises. CSS faile võib ühele lehele kaasata mitmeid ning mitmed HTML lehed võivad kasutada samu CSS faile. [27]

1.3. CSS meediatüübid

CSS stiililehtede üks tähtsaid omadusi on võimalus määrata, kuidas dokumenti erineval meedial esitatakse - ekraanil, trükis, kõnesüntesaatoril, pimedate kirjas, pihuseadmetes. Mõningad CSS omadused on mõeldud vaid teatud meedia jaoks (näiteks `page-break-before` kehtib ainult lehekülgedele jagatud meedia puhul). Samas on mitmeid omadusi, mis on küll kõigil meediatüüpidel olemas, kuid nõuavad erinevaid väärtusi (näiteks `font-size` - ekraanil kasutatakse tavaliselt suuremat kirjasuurust kui paberil). Seetõttu on oluline kirjeldada, millise meediatüübi jaoks konkreetne CSS fail mõeldud on.

Olemas on kümme erinevat CSS meediatüüpi, neist levinumad on *screen* - mõeldud arvutiekraanidel esitamiseks; *print* - lehekülgedeks jaotatud materjali esitamiseks ning printimiseks; *handheld* - lehtede kuvamiseks pihuseadmetes.

Meediatüüp määratakse HTML failis järgmiselt:

```
<link rel=”stylesheet” type=”text/css” href=”style.css” media=”handheld” />
```

Antud näites on tegemist CSS failiga, mis on mõeldud veebilehe esitamiseks pihuseadmetes. [21]

2. Ülevaade CSS raamistikest

CSS raamistikud (*CSS frameworks*) ehk veebikujunduse raamistikud on CSS failidest koosnevad kogumikud. Need sisaldavad CSS reegleid põhilise tüpograafia, vormide stiili, lehekülje paigutuse, veebilehitsejate vaikesätete nullimiste kohta. CSS raamistikud on suunatud veebilehe kujundajatele või kodeerijatele, kuna muudavad veebilehe kujundamise protsessi kiiremaks ja lihtsamaks. Samuti aitab raamistik vältida sagedasi pisivigu kodeerimisel ning hõlpsamini luua standarditele vastavamaid veebilehti. [7]

2.1 CSS raamistiku osad

CSS raamistik koosneb tavaliselt mitmetest CSS failidest, mis võivad sisaldada järgmisi osi:

1. võrestik
2. veebilehitsejate vaikeväärtuste nullimised
3. tüpograafia
4. vormide kujundus
5. stiil veebilehitseja Internet Exploreri jaoks
6. printimisstiil [3]

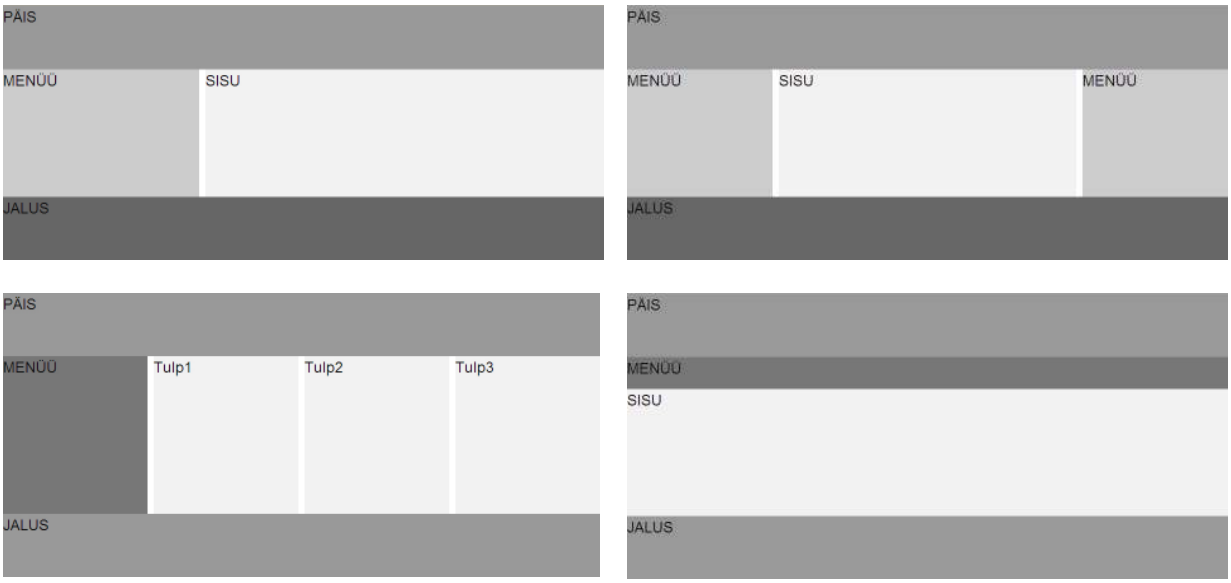
Igas raamistikus ei pruugi kõik need osad kaetud olla. Uurimise käigus selgus, et mõni raamistik katab ülalolevatest kõik osad (näiteks Blueprint - http://www.blueprintcss.org/), mõni aga on rohkem keskendunud konkreetsele osale (näiteks 960 Grid System - http://960.gs/, kus põhirõhk on asetatud võrestikule ning katmata on tüpograafia osa).

Kõiki raamistikus määratud stiile on võimalik enda kirjutatud CSS failis üle kirjutada ja kohandada konkreetse projektiga sobivaks. Samuti ei pea iga projekti juures kasutama raamistiku kõiki faile, vaid kasutaja saab vastavalt projekti nõuetele sobilikud osad ise valida.

Alljärgnevalt on võimalikud CSS raamistiku osad pikemalt lahti kirjutatud.

2.1.1. Võrestik

Võrestik on kahedimensiooniline struktuur, mis koosneb vertikaalsetest ja horisontaalsetest telgedest. [14] Veebilehe võrestik on justkui lehe „selgroog" - selle eesmärk on jagada veebilehe struktuur visuaalselt tasakaalustatud plokkideks, mida on inimsilmal hea vaadata. [2] Näiteks saab võrestikku kasutades luua veebilehel selliseid plokke nagu päis, menüü, sisuosa ja jalus. Joonisel 2 on kujutatud neli võimalikku veebilehe ülesehitust.



Joonis 2. Näited võimalikest võrestikul põhinevatest veebilehe ülesehitustest.

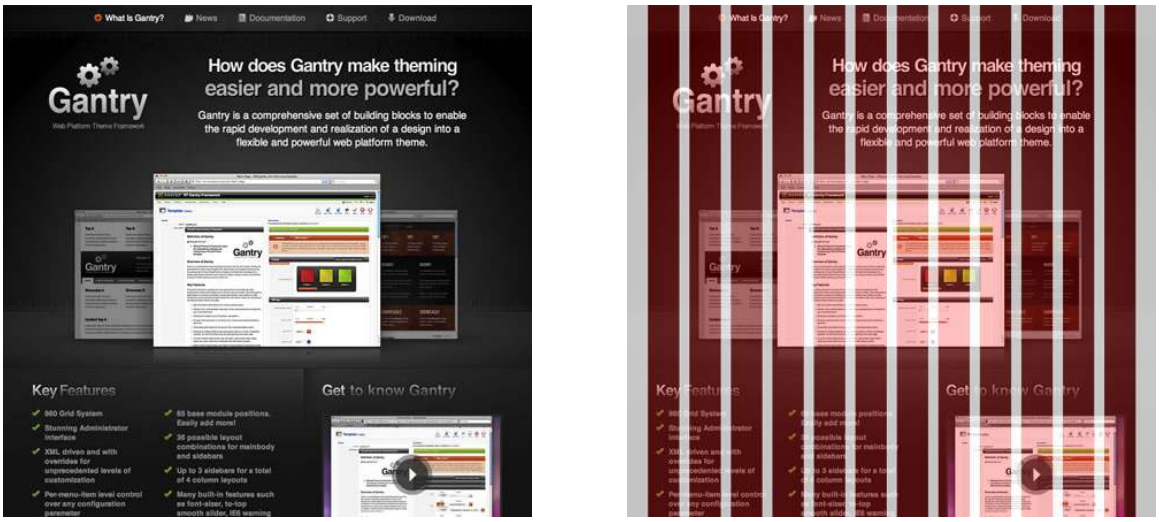
Tavaliselt määrab veebilehe sisu laiuse võrestiku konteiner, mille sisse paigutatakse tulbad, mis moodustavad veebilehe ülesehituse. Konteineri laius oleneb kasutatavast raamistikust. Näiteks Blueprint CSS raamistikus on võrestiku konteineri laiuseks vaikimisi 950px ning see on jaotatud kahekümne neljaks 30px laiuseks tulbaks, mille vahel on veerised 10px. Tulpasid saab kombineerida vastavalt sellele, kui laia plokki parasjagu sisu esitamiseks soovitakse. [26] Samas on ka võrestikke, kus veebilehe kujunduse laius ei pruugi olla fikseeritud, vaid paindlk - lehe laius muutub vastavalt veebilehitseja akna suurusele. Näiteks Emastic CSS raamistikus on võimalik luua nii fikseeritud kui ka muutliku laiusega veebilehti. [12]

Näide võimalikust võrestiku HTML koodist Blueprint raamistikus:

```
<body>
    <div class="container">
        <div id="header" class="span-24 last">PÄIS</div>
        <div id="menu" class="span-6">MENÜÜ</div>
        <div id="content" class="span-18 last">
            <div class="span-6 cols">Tulp1</div>
            <div class="span-6 cols">Tulp2</div>
            <div class="span-6 last cols">Tulp3</div>
        </div>
        <div id="footer" class="span-24 last">JALUS</div>
    </div>
</body>
```

Antud koodinäitega saavutatakse joonisel 2 kujutatud neljas ülesehitus. Klassiga `span-x` määratakse vastava jaotuse (*div*) laius.

Joonisel 3 on toodud näide 960 Grid Systemi 12-tulbalisel võrestikul põhinevast veebikujundusest.



Joonis 3. 960 Grid Systemi 12-tulbalisel võrestikul põhinev veebikujundus. [1]

2.1.2. Veebilehitsejate vaikeväärtuste nullimised

Kõigil veebilehitsejatel on määratud vaikimisi stiilid teatud elementide kuvamiseks, kuid need pole kõigil brauseritel ühesugused. Selleks, et veebilehte erinevates brauserites võimalikult identsena kuvataks, nullitakse raamistikus teatud elementide veerised (*margin*) ja vooderdused (*padding*), kirjatüübi suurused ja rea kõrgused. [13]

2.1.3. Tüpograafia

Tüpograafia osas määratakse tekstiosa põhiline kujundus - kirjatüübid, kirjasuurused, reakõrgused, lõikude veerised ja vooderdused. [20]

2.1.4. Vormide kujundus

Vormide kujunduse CSS failis määratakse lihtne kujundusstiil siltidele, tekstialadele ja teistele vormi osadele. [20]

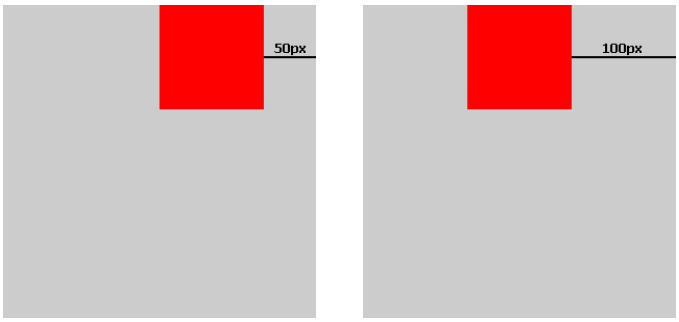
Joonisel 4 on võrdlus, missugune näeb vormi kujundus välja ilma raamistikuta kodeerides ning missugune Blueprint raamistiku vormide ning tüpograafia faile kasutades.

Joonis 4. Vasakul vormi kujundus raamistikuta, paremal Blueprint raamistikuga.

2.1.5. Stiil Internet Exploreri jaoks

Veebilehitseja Internet Exploreri jaoks on vajalik eraldi stiilifail, kuna see sisaldab lahendusi teatud probleemidele, mida teistes brauserites ei esine. Näiteks topeltveerise viga: elementidele määratud veerised saavad veebilehitsejas Internet Explorer 6 kaks korda suurema veerise, kui on määratud. [6]

Näide Internet Explorer 6 topeltveerise veast: loodud on 300x300px hall kast, mille sisse on paigutatud punane 100x100px kast, mis on joondatud paremale ning parempoolseks veeriseks on määratud 50px. Joonisel 5 on näha, kuidas paigutub punane kast halli kasti sisse brauserites Mozilla Firefox ning Internet Explorer 6.



Joonis 5. Vasakul kasti paigutus brauseris Mozilla Firefox, paremal brauseris Internet Explorer 6.

Eelneva probleemi lahenduseks on määrata joondatavale konteinerile CSS koodis reegel `display:inline;`. [29]

Veebilehitsejas Internet Explorer on ka teisi erinevusi, mida on püütud võimalikult palju siluda just Internet Exploreri jaoks mõeldud stiilifailis. See CSS fail tuleks veebilehele kaasata tingimuslausega vaid siis, kui veebilehitsejana kasutatakse Internet Explorerit:

```
<!--[if IE]>
    <link rel=”stylesheet” type=”text/css” href=”ie.css” />
<![endif]-->
```

2.1.6. Printimisstiil

Printimise CSS fail on mõeldud veebilehe välja printimiseks. Kuna paljudel veebilehe elementidel pole paberkujul otstarvet (navigatsioon, vormid, teatud kujunduselemendid), siis

on oluline määrata, et välja prinditaks vaid tähtsam põhisisu. Samuti määratakse prinditavatele elementidele sobiv stiil - paberile sobilikud kirjastiilid ja värvid. [19]

Printimise stiililehe lisamisel veebilehele on oluline määrata meediatüübiks *print*:

```
<link rel=”stylesheet” href=”print.css” type=”text/css” media=”print” />
```

2.2. CSS raamistiku eelised ja puudused

CSS raamistikel on ilma raamistikuta kodeerimise ees mitmeid eeliseid:

1. HTML elementide paigutuse ühtsus ja järjepidevus - vähendab ka CSS koodi vigu
2. HTML tabeliteta veebikujundus - semantiliselt korrektsem kood
3. visuaalne sidusus lehe elementide vahel
4. erinevate brauserite toetus
5. võrestiku põhjale on lihtne ka keerulise struktuuriga kujundusi luua
6. kasutatav staatiliste lehtede juures, sisuhaldusega lehtedel, blogi platvormidel
7. lihtsustab ja kiirendab kasutajaliidese kodeerijate tööd
    a. vähem vaeva veebilehtede ülesehituse loomisel, võrreldes sellega, kui peaks iga projekti juures vajaliku CSS koodi nullist kirjutama
    b. vähem vaeva tulevikus, kui on vaja mõnda elementi ümber asetada või muuta seotud elementide tunnuseid (näiteks tüpograafiat) [28]

CSS raamistike puudused võrreldes raamistikuta kodeerimisega:

1. oht, et sisaldab üleliigset koodi, mida paljudel kasutajatel kunagi vaja ei pruugi minna
2. raamistikuga harjumine ja selle täielikult mõistmine võtab aega
3. raskused raamistiku koodi ning enda kirjutatud koodi ühildamisel - igal kodeerijal on omad harjumused klasside ja identifikaatorite nimetamiseks ning raamistikus olevad klasside nimed ei pruugi olla mugavad ja meeldejäävad [8]
4. eraldi HTTP-päringud rohkematele failidele [10]
5. raamistikud pole üldiselt mõeldud CSSiga alustajatele

a. võivad pidurdada õppimisvõimet - kodeerija ei harju ise lihtsamaid CSS reegleid kirjutama ning siis võtab ka vigade leidmine rohkem aega [9]

b. vigast raamistikku kasutades võib kodeerija endalegi halva harjumuse külge saada [15]

2.3. Levinuimad CSS raamistikud

Aja jooksul on arendatud palju erinevaid CSS raamistikke, mille eesmärgid omavahel mõnevõrra varieeruvad. Kuna aga ka veebiprojektide mahud ning nõuded on erinevad, siis on iga projekti jaoks võimalik laiast valikust leida just sobiv raamistik.

Siinkohal on välja toodud mõningad näited levinumatest CSS raamistikest:

1. Blueprint http://www.blueprintcss.org/
2. 960 Grid System http://960.gs/
3. YAML http://www.yaml.de/en/
4. Elements http://elements.projectdesigns.org/
5. Emastic http://code.google.com/p/emastic/
6. Tripoli http://devkick.com/lab/tripoli/
7. YUI Grids CSS http://developer.yahoo.com/yui/grids/ [5]

Järgnevalt räägitakse lähemalt kolmest populaarsemast CSS raamistikust - 960 Grid System, YAML ning Blueprint CSS.

2.3.1. 960 Grid System

960 Grid System on 2008. aastal väljastatud minimaalne CSS raamistik, mis oli Nathan Smith'i poolt esialgu vaid enda tarbeks loodud. Ootamatult saavutas see aga laialdase populaarsuse ning on nüüdseks kasutust leidnud paljude projektide põhjana. [22]

960 Grid System on põhiliselt võrestikule keskendunud raamistik, kus pole kaetud tüpograafia osa ning samuti pole eraldi printimise stiililehte. 960 raamistikus on olemas kaks võrestiku varianti: 12 tulbaga ning 16 tulbaga, seejuures põhikonteiner on alati 960px laiune.

Kaheteistkümnest tulbast koosnevas võrestikus on iga tulba laiuseks 60px, kuueteistkümnest tulbast koosnevas võrestikus 40px. Kummalgi juhul on iga tulba veeriseks nii paremal kui vasakul pool 10px.

Samas on lisaks võrestikule kaetud raamistikus ka nullimiste osa, mis tähendab, et enamjaolt on lahendatud ka veebilehitsejate vaikeväärtuste erinevuste probleem.

Raamistiku allalaetava failidekogu suurus on ligikaudu 8MB, kuid see sisaldab lisaks raamistiku CSS koodile ka kujunduspõhjasid erinevate kujundusprogrammide jaoks, prinditavaid visandilehti ning HTML faile. *960.css* fail ise on kokkupakitult kõigest 3.6KB.

960 CSS raamistik on vabavaraline ning soovi korral võib igaüks seda enda vajadustele kohandada. [1]

2.3.2. YAML

YAML (*Yet Another Multicolumn Layout*) on (X)HTML/CSS raamistik moodsate ning paindlike veebilehe ülesehituste loomiseks. YAML raamistik väljastati esmakordselt 2005. aasta oktoobris. Sellest ajast alates on see arenenud stabiilseks ja mitmekülgseks raamistikuks, mida regulaarselt uuendatakse.

YAML raamistiku eesmärk on luua paindlikke ülesehitusi. See sisaldab võrestikku, tüpograafia ja vormide stiile, olemas on isegi näidis navigatsiooni kujundusest. Lisaks sellele on olemas eraldi stiilid printimise jaoks ning Internet Exploreri vigade parandused.

Raamistiku märgendstruktuur ning CSS komponendid jätavad kodeerijale kujunduse suhtes vabad käed. Võimalik on luua erinevaid lahendusi - paindliku, elastse või fikseeritud ülesehitusega, igasuguse arvu alamjaotustega.

YAML põhineb veebistandarditel ning toetab kõiki kaasaegseid veebilehitsejaid. Enamus veebilehitsejate kuvavigu on YAML raamistiku loojate poolt parandatud nii, et veebidisainer ei pea selleks ise vaeva nägema. Sealhulgas on parandatud ka kõik olulisemad Internet Exploreri versioonide 5.x kuni 8.0 kuvavead.

YAML on end tõestanud professionaalses kasutuses ning sellele aitab kindlasti kaasa ka fakt, et kõik raamistiku CSS komponendid ning mitmed erinevate ülesehituste meetodid on

põhjalikult dokumenteeritud nii inglise kui saksa keeles, illustreeritult arvukate näidetega. Samuti on loodud üha kasvav hulk kasutamiseks valmis YAML raamistikul põhinevaid kujunduspõhjasid (*template*) erinevate sisuhaldussüsteemide jaoks. Lisaks sellele on välja kujunenud pidevalt laienev aktiivne YAML kogukond. Olemas on kogukonna foorum (nii inglise kui saksa keeles), kus saab vahetada mõtteid ning arutleda tekkinud probleemide üle, samuti vajadusel abi küsida.

YAML CSS raamistikku võib tasuta kasutada sellisel juhul, kui veebilehele on lisatud link YAML projekti kodulehele. Kommertseesmärgil loodud veebilehe tellijale ei pruugi see variant aga sobida. Siis on alternatiivina võimalik osta tasuline litsents, mille korral ei ole YAML projektile tagasiviitamine vajalik.

YAML allalaetavate failide kogumaht on 2.36MB, kuid see sisaldab ka hulgaliselt näiteid ning JavaScripti koodi. [31]

2.3.3. Blueprint CSS

Blueprint on 2007. aastal loodud vabavaraline CSS raamistik, mille eesmärk on kokku hoida kodeerijate aega. Hõlpsasti kasutatav võrestik, mõistlik tüpograafia ning printimisstiil annavad tugeva põhja, mille peale oma projekte luua.

Blueprint raamistik on kompaktne ning loodud nii, et oleks kaetud kõik vajalikud aspektid, jäädes seejuures mahult minimaalseks. Raamistik koosneb kuuest failist: *reset.css*, *grid.css*, *typography.css*, *forms.css*, *print.css* ning *ie.css*.

Failis *reset.css* nullitakse brauserite poolt vaikimisi määratud väärtused, mis tekitavad veebilehe visuaalseid erinevusi veebilehitsejates.

Failis *grid.css* on käsitletud võrestikku, mille põhjale on võimalik ka keerulise struktuuriga veebilehti luua. Paika on seatud veebilehe sisukonteineri vaikeväärtus ning tulpade laius ja arv. Sisukonteineri laiuseks on 950px ning see on jaotatud kahekümne neljaks 30px laiuseks tulbaks, mille parem veeris on 10px. Kellele aga selline jaotus liiga piiratud tundub, võib kasutada võrestiku generaatorit (näiteks http://bgg.kematzy.com/), kus saab luua enda määratud laiuste ja veeristega võrestiku Blueprint CSS raamistiku jaoks.

Tekstide ja pealkirjade suurused, kirjastiilid ning minimaalne vormide kujundus on vastavalt failides *typography.css* ja *forms.css*.

Selleks, et veebileht paberile prinditult kena ja arusaadav välja näeks, on eraldi ka printimisstiil failis *print.css*.

Lisaks on Blueprint CSS raamistiku jaoks olemas igasugu erinevaid pluginaid nuppude, sakkide (*tabs*) ja *sprite*-pildifailide (*CSS sprites* - http://css-tricks.com/css-sprites/) kujundamiseks; tööriistu, redaktoreid ja kujunduspõhjasid tööprotsessi iga sammu jaoks.

Allalaetavate failide kogusuurus on 7.92MB. See sisaldab muuhulgas ka mitmeid näidisfaile ning Photoshopi võrestikupõhja. Kokkupakkimata CSS failid, mida projektide juures raamistikuna kasutada, moodustavad sellest mahust vaid 19.9KB. [3]

3. Nõuded veebilehtedele

Selles peatükis on välja toodud käesoleva töö autori poolt paika pandud nõuded, millele veebileht peab vastama.

3.1. Suurus

Näidetena loodavate veebilehtede laius on fikseeritud - 950px. Veebileht peab asetsema brauseriakna keskel. Veebilehe kõrgus muutub automaatselt vastavalt sisule.

Veebilehe laiuse valikul on silmas peetud enimlevinud ekraani resolutsioone ning ka kasutatava raamistiku võimalusi - Blueprint CSS raamistikus on vaikimisi sisukonteineri laiuseks 950px.

10 enimlevinud ekraani resolutsiooni ning nende osakaal protsentides 2009. aasta detsembris:

1. 1024x768 - 28.35%
2. 1280x800 - 20.33%
3. 1280x1024 - 11.11%
4. 1440x900 - 8.58%
5. 1680x1050 - 5.54%
6. 800x600 - 4.12%
7. 1366x768 - 3.64%
8. 1152x864 - 2.48%
9. 1920x1200 - 1.95%
10. 1280x768 - 1.57%

Statistika on tehtud iga W3Counteri poolt jälgitava veebilehe viimase 15 000 vaatamise põhjal. W3Counteri näited sisaldavad hetkel 33 586 veebilehte. [30]

Selle statistika kohaselt peaks veebileht laiusega 950px ekraanile laiuselt ära mahtuma vähemalt 83.55% vaatajatel.

3.2. Toetatud veebilehitsejad

Näidetena koostatud veebilehed peavad toetama järgnevaid veebilehitsejaid:

1. Internet Explorer (versioonid 6-8)
2. Mozilla Firefox
3. Google Chrome
4. Safari
5. Opera

See tähendab, et veebileht peab kõigis ülalmainitud brauserites võimalikult identne välja nägema. Põhjuseks, miks välja on valitud just need veebilehitsejad, on populaarseimate veebilehitsejate jaotus. 2009. aasta detsembris oli see järgmine:

1. Internet Explorer - 50.3%
2. Mozilla Firefox - 32.3%
3. Google Chrome - 5.5%
4. Safari - 5.1%
5. Opera - 2.1%

Statistika on tehtud iga W3Counteri poolt jälgitava veebilehe viimase 15 000 vaatamise põhjal. W3Counteri näited sisaldavad hetkel 33 586 veebilehte. [30]

Selle statistika kohaselt peaks 95.3% veebilehe vaatajate jaoks nägema leht välja ühesugune ning vigadeta.

3.3. Testimisvahendid

Veebilehtede visuaalseks kontrollimiseks kasutatakse töö käigus erinevaid veebilehitsejaid:

1. Internet Exploreri versioonid 6 ja 7 - IETester

   (http://www.my-debugbar.com/wiki/IETester/HomePage)

2. Internet Explorer 8

   (http://www.microsoft.com/windows/Internet-explorer/default.aspx)

3. Mozilla Firefox 3.6.3 (http://www.mozilla.com/en-US/firefox/personal.html)
4. Google Chrome 4.0.249.89 (http://www.google.com/chrome)
5. Safari 4.0.3 (http://www.apple.com/safari/download/)
6. Opera 10.53 (http://www.opera.com/download/)

Koodi valideeruvuse kontrollimine toimub järgmiste vahenditega:

- CSS koodi valideeruvus - W3C CSS keele valideerimisteenus

  (http://jigsaw.w3.org/css-validator/)

- HTML koodi valideeruvus - W3C märgistuskeele valideerimisteenus

  (http://validator.w3.org/)

- HTML koodi valideeruvus jooksvalt töö käigus – Mozilla Firefoxi laiendus HTML Tidy (http://users.skynet.be/mgueury/mozilla/)

Näidetena loodud veebilehti testitakse arvutis, mille andmed on järgmised: protsessor Intel(R) Celeron(R) M CPU 410 @ 1.46GHz, mälu 1.49GB RAM, operatsioonisüsteem Microsoft Windows XP Professional (Version 2002, Service Pack 3).

HTML ja CSS failide suurust ning HTTP-päringute mahtu ja suurust lehe laadimisel ning lehe laadimise kiirust mõõdetakse veebilehitsejas Mozilla Firefox. Selleks kasutatakse Mozilla Firefoxi laiendusi Firebug (http://getfirebug.com/) ning YSlow (https://addons.mozilla.org/en-US/firefox/addon/5369).

4. Lihtsa struktuuriga koduleht

Esimese näitena on eesmärgiks luua joonisel 6 toodud kujundusega isiklik veebileht.

Joonis 6. Näitelehena loodava isikliku veebilehe kujundus.

Koduleht on üheleheline, menüüs toodud viidad viivad ankruna ühel HTML lehel asuvasse vastavasse plokki. Sisukonteineri laiuseks on 950px, see paikneb veebilehitseja akna keskel. Veebilehe kõrgus muutub automaatselt vastavalt sisule.

Kirjeldatud veebileht luuakse esmalt ilma raamistikuta ning seejärel vabavaralist CSS raamistikku Blueprint kasutades.

Töö protsess kummalgi juhul:

1. Põhilise ülesehituse loomine - veebilehe päis, menüü, sisuala.
2. Põhikujunduse paika panemine - taustad, menüü viidad.
3. Sisu lisamine ning selle kujundamine – „Minust” teksti, pildigalerii, portfolio (pilt ja tekst) ning kontaktivormi loomine.

4. Brauseritevaheliste erinevuste likvideerimine - esmalt valmistatakse leht Mozilla Firefoxis sammhaaval protsessi jälgides, seejärel võrreldakse veebilehe väljanägemist teistes brauserites ning kirjeldatakse ja parandatakse tekkinud vead.
5. Kokkuvõtvate andmete kirjeldamine (HTML ja CSS failide suurused, HTTP päringute maht ning suurus, keskmine lehe laadimise kiirus).

4.1. Kodulehe loomine raamistikuta

Veebilehe kujunduse kodeerimine ilma raamistikuta tähendab, et kogu vajalik stiil kirjutatakse ise failis *style.css*. See paigutatakse näiteks kausta „css" ning kaasatakse seejärel veebilehele HTML lehe päises:

```
<link rel="stylesheet" type="text/css" href="css/style.css" media="screen" />
```

4.1.1. Põhilise ülesehituse loomine

Esimese sammuna on loodud veebilehe põhiline ülesehitus ilma kujunduseta. Tulemus on näidatud joonisel 7.

Joonis 7. Veebilehe põhiline ülesehitus.

Selline ülesehitus saavutatakse järgneva HTML koodiga:

```
<div id=”outer_container”>
    <div id=”container”>
        <div id=”menu”>MENÜÜ</div>
        <div id=”content”>SISU</div>
    </div>
</div>
```

Ilma raamistikuta isikliku kodulehe põhilise ülesehituse loomise etapi HTML kood on toodud Lisas 1 kaustas „isiklik” failis *index1.html* ja CSS kood failis *style1.css*.

4.1.2. Põhikujunduse paika sättimine

Joonisel 8 on näha veebileht, kui sellele on määratud taustapildid.

Joonis 8. Veebileht, millel on määratud taustapildid.

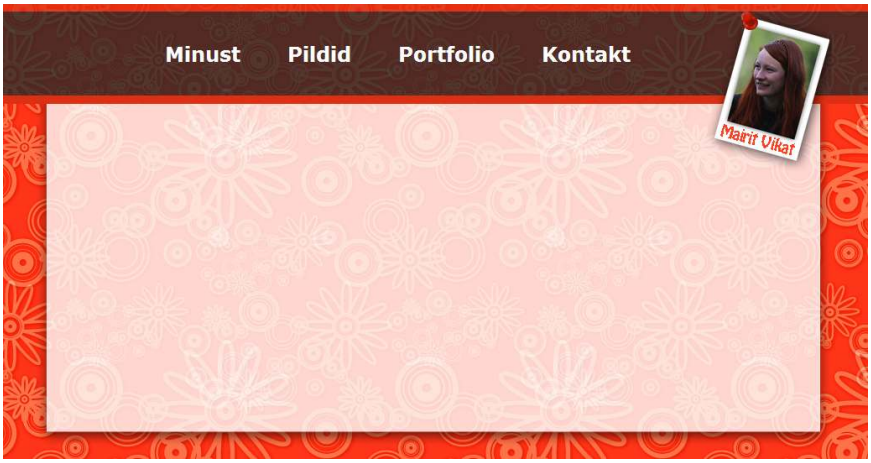
Siinkohal ilmneb probleem: jooniselt 8 on näha, et menüü taust ei ulatu brauseri akna ühest servast teise (nii ülemises ääres kui paremal ja vasakul on vahed sees). Selle parandamiseks tuleb HTML `body`'le lisada CSS reegel `margin: 0;`, siis näeb kujunduse põhi korrektne välja.

Sellele sammule vastav HTML kood on toodud Lisas 1 kaustas „isiklik” failis *index2-1.html* ja CSS kood failis *style2-1.css*.

Teise etapi teise sammuna on lisatud menüü viidad. Esialgu jääb menüü kujundus brauseri CSS vaikesätete tõttu selline, nagu näidatud joonisel 9.

Joonis 9. Brauseri CSS vaikesätetest tulenev menüü kujundus.

Vaikimisi kuvatakse nimekirja elemendid eraldi real, kuid loodavas kujunduses on neid vaja kuvada teineteise kõrval. Seetõttu tuleb nimekirja elemendile `li` lisada CSS reegel `display: inline;`. Menüü välimus pärast ülejäänud stiili lisamist on näidatud joonisel 10.



Joonis 10. Menüü õige kujundus.

Põhikujunduse paika seadmisele vastav HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index2-2.html* ja CSS kood failis *style2-2.css*.

4.1.3. Sisu lisamine ning selle kujundamine

„Minust” tekst

Kodulehele lisatakse esmalt „Minust” tekst (joonis 11).

Joonis 11. Veebilehe teksti esialgne kujundus.

Tekst näeb välja selline, nagu joonisel 11 on näidatud, seetõttu, et teksti esitamiseks kasutatakse veebilehitseja vaikimisi kirjastiile, -suuruseid ning veeriseid-vooderdusi. Soovitud teksti kujunduse saavutamiseks tuleb pealkirjadele, tekstidele, viitadele lisada oma äranägemise järgi stiilid - värvid, tekstisuurused, kirjastiilid, veerised-vooderdused. Seejärel näeb leht välja juba selline, nagu on näidatud joonisel 12.

Joonis 12. Soovitud kujundusega tekst.

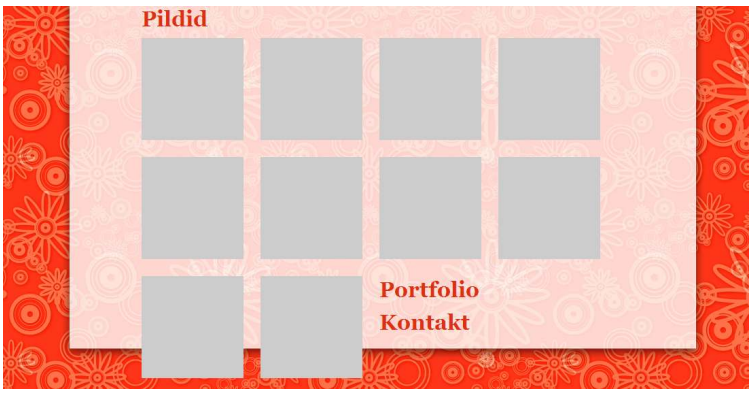
„Minust” teksti lisamist kirjeldav HTML kood on toodud Lisas 1 kaustas „isiklik” failis *index3-1.html* ja CSS kood failis *style3-1.css*.

Pildigalerii

Kodulehe pildigaleriisse lähevad minipildid, mille jaoks on loodud 150x150px konteinerid klassiga `images`, mis on omakorda alamjaotuse `images_container` sees. Viimane on vajalik selleks, et piltide ümber oleks paremal ja vasemal sama suured äärised, nagu tekstil on. Konteinerid `images` on joondatud vasakule, kuna siis paigutatakse neid ühele reale kõrvuti nii palju, kui mahub, ning ülejäänud lükatakse järgmisele reale. Näites on esialgu tehtud 10 sellist konteinerit, kuhu hiljem pilt sisse lisada:

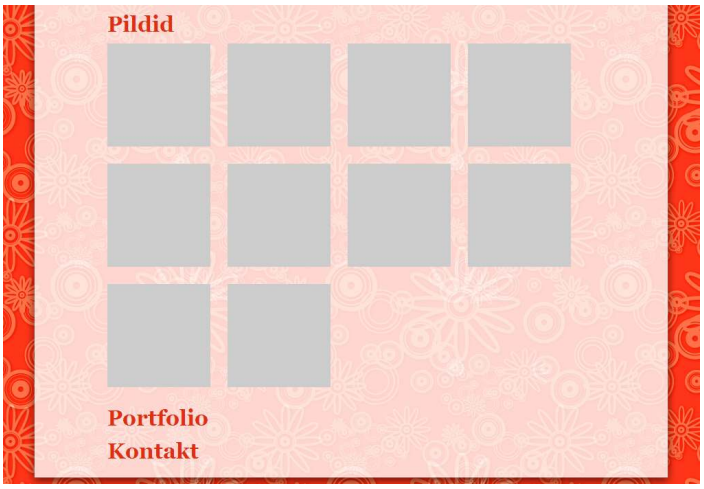
```
<div id=”images_container”>
    ...
    <div class=”images”></div>
    ...
</div>
```

Galerii loomisel tekib probleem: üksteise kõrvale mitte mahtuvad konteinerid lükatakse küll uuele reale, kuid viimase konteineri järel ei lükata järgnevaid pealkirjasid uuele reale. Olukord on kujutatud joonisel 13.



Joonis 13. Konteinerite vale paigutus galeriis.

Lahendus: konteinerile `images_container` tuleb lisada CSS reegel `float: left;`, siis nihkuvad sellele järgnevad elemendid (antud juhul pealkirjad) uuele reale (joonis 14).



Joonis 14. Konteinerite õige paigutus galeriis.

Pildigalerii loomise HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index3-2.html* ja CSS kood failis *style3-2.css*.

Tehtud tööd

Tehtud tööde kuvamiseks loodud alamjaotused, mille sisse hiljem lisatakse pilt ja tekst, on kujutatud joonisel 15.

Joonis 15. Tehtud tööde kuvamiseks loodud alamjaotised.

Joonisel 15 kujutatud alamjaotiste HTML kood on järgmine:

```
<div id="portfolio_container">
    <div class="works"></div>
    <div class="works"></div>
    <div class="works"></div>
</div>
```

Järgmisena lisatakse igasse konteinerisse tehtud töö pealkiri koos kuupäevaga, selle alla vasakule pilt ning paremale kirjeldav tekst. Konteineri kõrgus on automaatselt muutuv, kuna piltide kõrgused ning kirjeldavate tekstide pikkused võivad varieeruda. Lisatud pidi ning teksti esialgne asetsus on toodud joonisel 15.

Joonis 15. Teksti ebakorrektne paigutus pildi suhtes.

Joonisel 15 ilmneb probleem - nimelt tekst ei asetse pildi kõrval, vaid all. Lahenduseks tuleb pildid CSS koodiga vasakule joondada - `float: left;`, siis asetseb tekst pildi suhtes õigesti (joonis 16).

Joonis 16. Teksti õige paigutus pildi suhtes.

Kuid ka see pole päris soovitud tulemus. Kuna viimase näite kirjelduse tekst on pildi kõrgusest pikem, siis algab viimane lause üksikuna pildi alt vasakult. See peaks aga algama samalt realt ülejäänud tekstiga. Seetõttu tuleb kirjelduse paragrahvile `p` lisada CSS reegel `overflow: hidden;`. Siis näeb portfolio see osagi välja just selline, nagu soovitud (joonis 17).

Joonis 17. Portfolio korrektne kujundus.

Portfolio loomist kirjeldav HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index3-3.html* ja CSS kood failis *style3-3.css*.

Kontaktivorm

Selle sammu eesmärk on luua kontaktivorm, mille kaudu saab lehe omanikuga ühendust võtta. Esialgu kujundamata kontaktivorm on näidatud joonisel 18.

Joonis 18. Kujundamata kontaktivorm.

Pärast vormi kujundamist näeb kontaktivorm välja selline, nagu toodud joonisel 19.

Joonis 19. Kujundatud kontaktivorm.

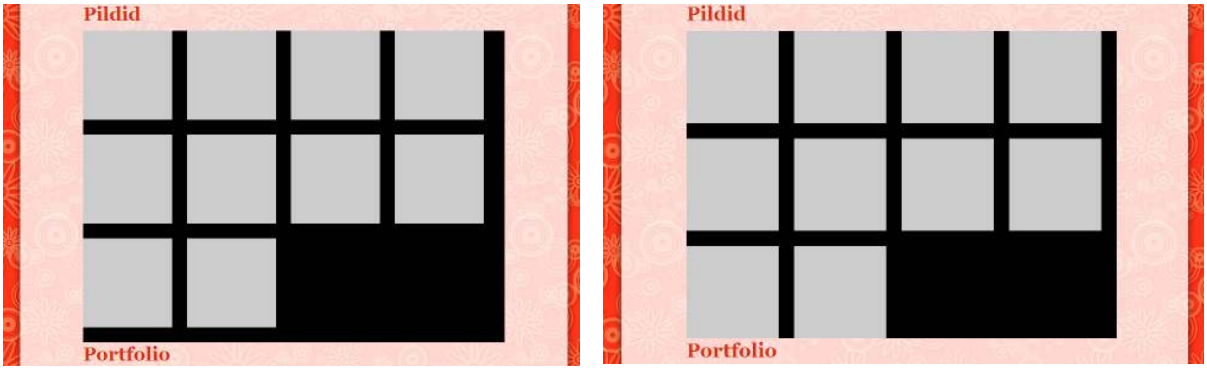
Kontaktivormi loomise HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index3-4.html* ja CSS kood failis *style3-4.css*.

4.1.4. Brauseritevaheliste probleemide likvideerimine

Veebilehitsejas Mozilla Firefox 3.6.3, Google Chrome 4.1.249.1064, Safari 4.0.3, Opera 10.53 ja Internet Explorer 8 näeb koduleht välja korrektne, ühtki kujundusviga ei esine.

Internet Explorer 7

Veebilehitsejas Internet Explorer 7 leidub mõningaid probleeme. Nimelt on galeriis iga pildikonteineri images alumiseks veeriseks määratud 25px, kuid Internet Exploreri versioonides 7 ja 6 ei ole viimastel images konteineritel alumist veerist. See tähendab, et pildigalerii ja pealkirja „Portfolio" vahel on vähem ruumi, kui peaks. Joonisel 20 on musta taustaga näidatud konteiner `images_container` ning halli taustaga selle sees asetsevad konteinerid `images`.



Joonis 20. Vasakul pildikonteiner brauseris Mozilla Firefox, paremal brauseris Internet Explorer 7.

Kirjeldatud probleemi näol on tegemist veaga, mis ongi veebilehitseja Internet Explorer versioonidele 6 ja 7 omane ning selle parandus on teostatud Internet Exploreri versioonis 8. Probleem tekib sellest, et joondatud elementidel (*floated elements*) läheb alumine veeris (*margin-bottom*) kaduma. [18]

Kuna ühest lahendust sellele probleemile ei leidu, siis tuleb see parandada näiteks tingimusliku lausega (*if conditional*). Antud juhul tähendab see, et kui tegemist on veebilehitseja Internet Explorer versiooniga 6 või 7, siis muudetakse CSSis alamjaotuse `images_container` alumist vooderdust. Sellega jääb kõik muu paika, kuid lisaks tekitatakse konteineri alumisse äärde 25px kõrgune vahe, misjärel näeb tulemus välja samasugune, nagu teistes brauseriteski.

Lahenduse CSS kood on järgmine:

```
<!--[if lte IE 7]>
    <style type="text/css">
        #images_container { padding-bottom: 25px; }
    </style>
<![endif]-->
```

Internet Explorer 7 paranduste HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index4-1.html* ja CSS kood failis *style4-1.css*.

Internet Explorer 6

Internet Exploreri versioonis 6 leidub veelgi rohkem kujundusvigu, kui versioonis 7. Internet Explorer 6 suurimaks probleemiks on fakt, et see veebilehitseja versioon ei kuva läbipaistvaid *png*-laiendiga pilte läbipaistvana, vaid tekitab hallika tausta. [24] Sellel probleemil on kolm võimalikku lahendust:

1. salvestada läbipaistvad *png*-laiendiga pildid ümber *gif*-laiendiga failideks;
2. kasutada JQuery raamistiku JavaScripti faili, mis peaks läbipaistvad *png*-laiendiga failid muutma läbipaistvaks ka Internet Exploreri versioonis 6; [25]
3. muuta *png*-laiendiga pildid Internet Explorer 6 jaoks läbipaistvaks *htc*-failiga. [17]

Neist esimene variant antud juhul ei sobi, kuna pildid ei jää läbipaistvad (joonis 21).

Joonis 21. Taustapilt *gif*-laiendiga pildifailiks ümber salvestatult.

Ka JavaScripti kasutamise variant ei sobi konkreetse veebilehe korral, kuna venitab mõned pildid välja ning need taustapildid, mida korratakse, kaovad ära (joonis 22).

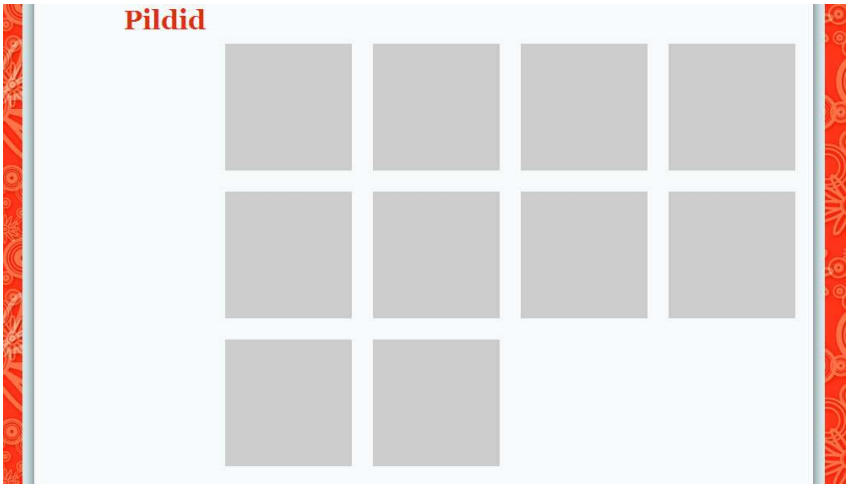
Joonis 22. Veebilehe välimus *png*-failide läbipaistvaks muutmiseks JavaScripti kasutamisel.

Samuti ei sobi ka kolmas lahenduse variant, kuna sarnaselt eelmisele kaovad korratavad taustapildid ära (joonis 23).

Joonis 23. Veebilehe välimus *png*-failide läbipaistvaks muutmiseks *htc*-faili kasutamisel.

Mööndusega, et kõigis teistes veebilehitsejates kuvatakse lehte õigesti ning probleem esineb vaid veebilehitsejas Internet Explorer 6, jääb probleem lahendamata.

Lisaks *png*-laiendiga piltide läbipaistmatusele leidub Internet Exploreri versioonis 6 teisigi probleeme. Näiteks Internet Exploreri versioonile 6 omane topeltveerise viga - piltide konteinerist vasakul on kaks korda rohkem ruumi, kui peaks. Pildikonteinerite paigutus veebilehitsejas Internet Explorer 6 on näidatud joonisel 24.



Joonis 24. Veebilehitseja Internet Exploreri topeltveerise viga.

Lahendus topeltveerise kaotamiseks on lisada alamjaotusele `images_container` CSS reegel `display: inline;`.

Veel üks väiksem erinevus on teksti paigutus portfolio osas, kus pildi kõrval oleva teksti viimased sõnad nihkuvad pildi alla, nagu on näidatud joonisel 25.

Joonis 25. Portfolio teksti asetsus veebilehitsejas Internet Explorer 6.

Joonisel 25 toodud probleemi vastu aitab tekstilõigule CSS reegli `height: 100%;` lisamine.

Järgmine probleem on vale vormi kujundus - nupp on tavaline, mitte pilt; tekstialade piirid (*border*) on valet värvi - joonis 26.

Joonis 26. Vormi kujundus veebilehitsejas Internet Explorer 6.

Tekstikastide piirete värvi erinevus tekib seetõttu, et see on CSS failis määratud tekstikastidel, mille tüübiks on *text* - `input[type="text"]`. Internet Explorer 6 aga ei toeta sellist pöördumist. Seetõttu tuleb stiil elemendi `input` külge panna ilma täpsustuseta:

```
input {
    border-top: 1px solid #FFA989;
    border-right: 1px solid #FF724A;
    border-bottom: 1px solid #FF724A;
    border-left: 1px solid #FFA989;
}
```

Seejärel on tekstikastide piirete värv õige.

Ka „Saada" nupp on vale kujundusega just CSS valija tüübi *submit* määramise tõttu. Seetõttu tuleb `input[type="submit"]` asemel valijaks kirjutada `input.button`, olles eelnevalt lisanud HTML koodis nupu elemendile klassi *button*. Seejärel on ka „Saada"-nupu kujundus korras.

Internet Explorer 6 vigade parandustega HTML kood on toodud Lisas 1 kaustas „isiklik" failis *index4-2.html* ja CSS kood failis *style4-2.css*.

4.1.5. Kokkuvõtvad andmed

Selleks, et hiljem ilma CSS raamistikuta ning CSS raamistikuga kodeeritud veebilehti oleks võimalik võrrelda, on uuritud mõningaid näitajaid - HTML ja CSS failide suurused, HTTP-päringute arv ja maht, keskmine lehe laadimise kiirus ning koodi valideeruvus.

Ilma CSS raamistikuta kodeeritud isikliku veebilehe puhul on need näitajad järgmised:

- HTML faili suurus: 5,2KB
- CSS faili suurus: 3.0KB
- HTTP-päringud: 12 päringut kogumahuga 254.2KB tühja vahemälu korral
- lehe laadimise kiirus: keskmiselt 0.269s tühja vahemälu korral (lehe 10 järjestikuse värskendamise ajad tühja vahemälu korral on toodud Lisas 2)
- koodi valideeruvus: nii HTML kui CSS kood valideeruvad

Ilma CSS raamistikuta kodeeritud personaalne veebileht on toodud Lisas 1 kaustas „isiklik" - vastav HTML kood on failis *index.html* ja CSS kood failis *style.css*.

4.2. Kodulehe loomine Blueprint raamistikuga

Käesolevas peatükis kirjeldatakse isikliku veebilehe loomist vabavaralise CSS raamistiku Blueprint põhjal. Blueprint CSS raamistiku kasutamiseks tuleb raamistiku CSS failid lisada sobivasse kausta (näiteks kaust „css"). Seejärel tuleb vajalikud stiilifailid kaasata loodava veebilehe HTML faili päises.

Blueprint CSS raamistikus on kuus erinevat stiililehte (*reset.css*, *grid.css*, *typography.css*, *forms.css*, *ie.css*, *print.css*). Antud juhul jääb printimise stiililehte *print.css* projekti skoobist välja. Internet Exploreri jaoks vajalik stiilifail *ie.css* kaasatakse lehele tingimuslausega vaid siis, kui veebilehitsejana kasutatakse Internet Explorerit. Ülejäänud nelja stiilifaili ei pea eraldi kaasama, kuna Blueprint CSS raamistiku failide hulgas on ka kompaktne *fail screen.css*, kuhu on kokku koondatud kõik neli stiilifaili. Seega piisab, kui HTML lehe päises kaasata kolm CSS faili - raamistiku failid *screen.css* ja *ie.css* ning enda loodud stiilifail *style.css*:

```
<link rel="stylesheet" type="text/css" href="css/screen.css" media="screen" />
<link rel="stylesheet" type="text/css" href="css/style.css" media="screen" />
<!--[if lte IE 8]>
    <link rel="stylesheet" type="text/css" href="css/ie.css" />
<![endif]-->
```

4.2.1. Põhilise ülesehituse loomine

Isikliku veebilehe põhilise ülesehituse HTML kood CSS raamistikku kasutades on järgmine:

```
<div id="outer_container">
    <div class="container">
        <div id="menu" class="span-24 last">MENÜÜ</div>
        <div id="content" class="span-24 last">SISU</div>
    </div>
</div>
```

Raamisrtiku poolt fikseeritud klass `container` määrab veebilehitseja keskele 950px laiuse ala. Klassid `span-x` on CSS raamistikus alamjaotuse laiuse määramiseks (näiteks `span-24` tähistab laiust 950px).

Põhilise paigutuse HTML kood on toodud Lisas 1 kaustas „isiklik_raamistikuga" failis *index1.html* ja CSS kood failis *style1.css*.

4.2.2. Põhikujunduse paika sättimine

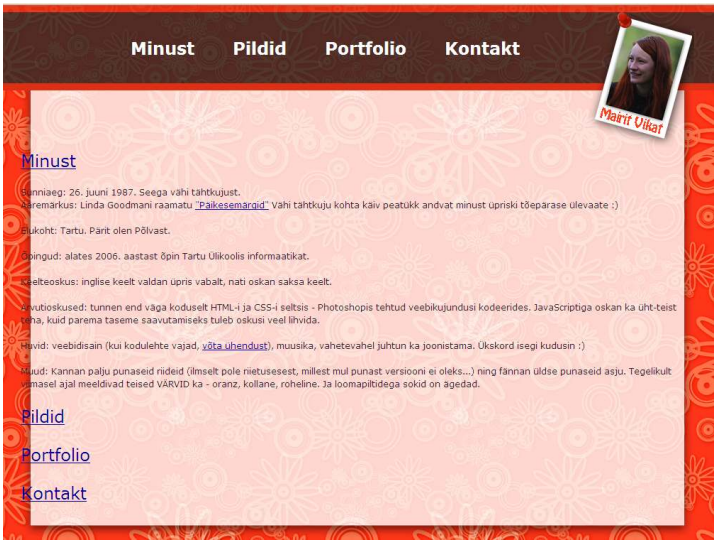
Põhikujunduse paika seadmisel Blueprint CSS raamistikku kasutades olulisi erinevusi ilma raamistikuta kodeerimisega võrreldes ei ole.

Põhikujunduse paika seadmise HTML koodid on toodud Lisas 1 kaustas „isiklik" failides *index2-1.html* ja *index2-2.html* ning CSS koodid failides *style2-1.css* ja *style2-2.css*.

4.2.3. Sisu lisamine ning selle kujundamine

„Minust" tekst

„Minust" teksti lisamise järel näeb teksti kujundus ja paigutus välja selline, nagu kujutatud joonisel 27.



Joonis 27. Teksti esialgne kujundus.

Pärast vajalike kujundusstiilide lisamist tekib probleem, mis on kujutatud joonisel 28.

Joonis 28. Osa teksti ebakorrektne kujundus.

Nimelt on lause „Ääremärkus: ...” kujundamiseks lisatud sellele klass `quiet`, mis peaks antud lause kirjastiili muutma kaldkirjaks ning kirjatüübi suuruse väiksemaks (90% tavapärasest kirjasuurusest). Blueprint raamistikus aga on juba klass `quiet` olemas ning see seab teksti värviks halli värvi. Seega, et mainitud lause õigesti kujundatud oleks, tuleb lisaks klassi `quiet` kirjastiili ja -suuruse reeglitele üle kirjutada raamistiku reegel `color: #666;`. See tähendab, et klassile `quiet` tuleb lisada reegel `color: #000;`.

Samuti tuleb üle kirjutada reegel, mis käib viitade kohta - raamistikus on määratud hiirega viida peale liikudes värviks must, kuid antud juhul peab see olema oranž.

Pärast muudatusi näeb kogu teksti kujundus välja korrektne.

„Minust” teksti lisamise osa on kajastatud Lisas 1, kus HTML kood on toodud kaustas „isiklik_raamistikuga” failis *index3-1.html* ja CSS kood failis *style3-1.css*.

Galerii

Galerii piltide jaoks on 150x150px konteinerid klassiga `images`, mis on alamjaotuse `images_container` sees - samamoodi, nagu ilma raamistikuta kodeerideski.

Vastav HTML kood:

```
<div id=”images_container” class=”span-18”>
    ...
    <div class=”span-4 images”></div>
    ...
</div>
```

Raamistikuga kodeerides saab alamjaotuse `images_container` laiuse 710px määrata HTML koodis klassiga `span-18` ning konteinerite `images` laiuse 150px klassiga `span-4`.

Vastupidiselt ilma raamistikuta kodeerimisele paigutuvad Blueprint raamistiku korral pildikonteinerid õigesti. Seejuures pole vaja konteineritele `images_container` ega `images` CSS koodis lisada vasakule joondamist (`float: left;`), piisab vaid konteinerite kõrguse, taustavärvi ja veeriste määramisest:

```
#images_container {
    margin: 0 120px;
}
.images {
    height: 150px;
    margin: 0 25px 25px 0;
    background: #ccc;
}
```

Galerii loomise HTML kood on toodud Lisas 1 kaustas „isiklik_raamistikuga” failis *index3-2.html* ja CSS kood failis *style3-2.css*.

Tehtud tööd

Tehtud tööde esitamise konteineritesse pildi ja teksti lisamisel tekivad samad probleemid pildi ja teksti paigutuses, mis ilma raamistikutagi. Ka lahendus neile on samasugune, nagu eelnevalt ilma raamistikuta kodeerides.

Lisaks sellele on aga probleem ka pealkirjas sisalduva valmimiskuu kujundusega. Selle kujundamiseks kasutatakse klassi `quiet`, mis peaks teksti muutma kaldkirjaks ning

väiksemaks. Eelnevalt oli probleem klassi `quiet` kirjavärviga, mille lahendamiseks sai lisatud reegel `color: #000;`. See nüüd probleemi tekitabki - antud juhul peaks teksti värv olema oranž.

Õige tekstivärvi saavutamiseks tuleb CSS faili lisada järgmine lõik:

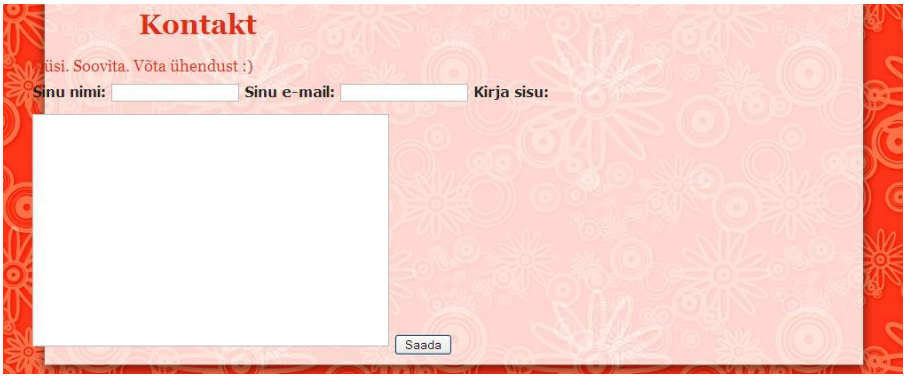
```
h3 span.quiet {
    color: #DE3014;
}
```

Selle ning teiste muudatuste järel on portfolio kujundus korras.

Selle sammu HTML kood on toodud Lisas 1 kaustas „isiklik_raamistikuga" failis *index3-3.html* ja CSS kood failis *style3-3.css*.

Kontaktivorm

Kontaktivormi esialgne väljanägemine Blueprint raamistiku põhjal on toodud joonisel 29.



Joonis 29. Kontaktivormi välimus Blueprint raamistiku põhjal.

Joonisel 29 nähtavad „Sinu nimi" jms on paksus kirjas Blueprint raamistiku stiili tõttu, seega tuleb üle kirjutada siltidele (`label`) mõjuv reegel `font-weight: bold;`, et selle asemel oleks `font-weight: normal;`. Pärast muudatuste sisseviimist näeb vormi kujundus välja selline, nagu näidatud joonisel 30.

Joonis 30. Vormi väljanägemine pärast mõningaid CSS muudatusi.

Kuigi CSS koodis on lisatud kõik samad stiilid, nagu ilma raamistikuta kodeeritud lehe puhulgi, siis on näha, et antud juhul ei mõju teatud omadused tekstikastidele. Põhjuseks see, et need omadused on juba raamistikus defineeritud. Selleks, et mõjuks enda kirjutatud stiil, tuleb nendele omadustele, mida praegu valesti kuvatakse, juurde kirjutada `!important`. Näiteks:

```
input, textarea {
    ...
    margin: 0 !important;
    background: none !important;
    border-top: 1px solid #FFA989 !important;
    border-right: 1px solid #FF724A !important;
    border-bottom: 1px solid #FF724A !important;
    border-left: 1px solid #FFA989 !important
    ...
}
```

Nende muudatuste järel näeb kontaktivormi kujundus korrektne välja.

Kontaktivormi loomise failid on Lisas 1 - HTML kood kaustas „isiklik_raamistikuga" failis *index3-4.html* ja CSS kood failis *style3-4.css*.

4.2.4. Brauseritevaheliste probleemide likvideerimine

Veebilehitsejates Mozilla Firefox 3.6.3, Google Chrome 4.1.249.1064, Safari 4.0.3, Opera 10.53 ja Internet Explorer 8 on veebilehe kujundus korrektne.

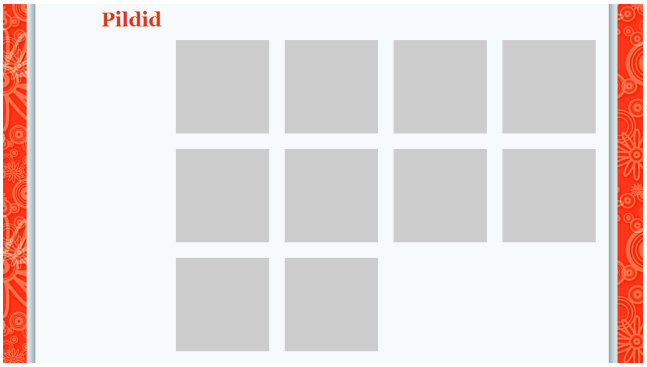
Internet Explorer 7

Veebilehitsejas Internet Explorer 7 ilmneb sama probleem, mis ilma raamistikuta kodeerimisel - pildigalerii ja sellele järgneva pealkirja vahel on väiksem ruum, kui peaks. Kuna tegemist on Internet Exploreri versioonidele 6 ja 7 omase probleemiga, siis on ka lahendus samasugune, nagu eelnevaltki.

Internet Explorer 7 parandustega HTML kood on toodud Lisas 1 kaustas „isiklik_raamistikuga" failis *index4-1.html* ja CSS kood failis *style4-1.css*.

Internet Explorer 6

Veebilehitsejas Internet Explorer 6 on lehel esialgu kaks probleemi. Esiteks ei asetse kogu kujundus lehe keskel, vaid vasakus ääres, ning teiseks on pildikonteineritest vasakul liiga palju ruumi (Internet Explorer 6 topeltveerise viga) - joonis 31.



Joonis 31. Pildikonteinerite paigutuse topeltveerise viga veebilehitsejas Internet Explorer.

Kui aga HTML lehel kaasata Internet Exploreri jaoks mõeldud stiililleht *ie.css*, siis leiavad need probleemid lahenduse ning kujundus näeb välja nii, nagu peab.

Peale eelmainitud probleemide jääb veel vaid *png*-laiendiga piltide läbipaistmatuse probleem, kuid see jääb ka sel korral lahendamata.

Internet Explorer 6 paranduste HTML kood on toodud Lisas 1 kaustas „isiklik_raamistikuga" failis *index4-2.html* ja CSS kood failis *style4-2.css*.

4.2.5. Kokkuvõtvad andmed

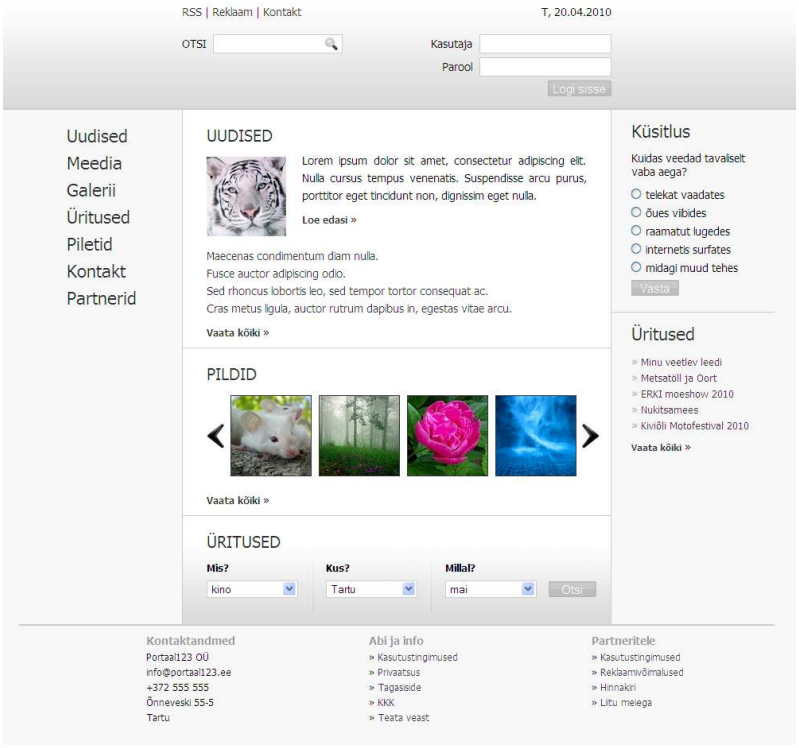
Blueprint CSS raamistikuga kodeeritud isikliku veebilehe näitajad:

- HTML faili suurus: 5.1KB
- CSS failide suurus: 15.2KB
- HTTP-päringuid: 13 päringut kogumahuga 266.3KB tühja vahemälu korral
- lehe laadimise kiirus: keskmiselt 0.301s tühja vahemälu korral (lehe 10 järjestikuse värskendamise ajad tühja vahemälu korral on toodud Lisas 3)
- koodi valideeruvus: nii HTML kood kui CSS kood on valideeruvad

Blueprint CSS raamistikuga kodeeritud isiklik veebileht on toodud Lisas 1 kaustas „isiklik_raamistikuga" - vastav HTML kood on failis *index.html* ja CSS kood failis *style.css*.

5. Keerulisema struktuuriga veebileht

Teise veebilehe näitena luuakse portaali veebileht, mille kujundus on toodud joonisel 32.



Joonis 32. Näitelehena loodud portaali kujundus.

Portaali veebileht luuakse samuti esmalt ilma raamistikuta ning seejärel vabavaralist CSS raamistikku Blueprint kasutades.

Töö protsess kummalgi juhul on sarnane isikliku veebilehe loomisele:

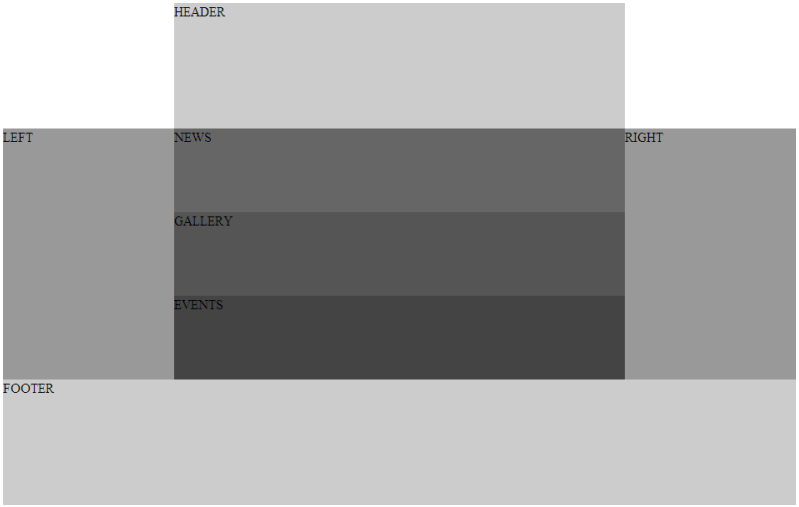
1. Põhilise ülesehituse loomine - päis, külgmenüüd, sisuala, jalus.
2. Põhikujunduse paika sättimine.
3. Sisu lisamine ning selle kujundamine.
4. Brauseritevaheliste erinevuste likvideerimine.
5. Kokkuvõtvate andmete kirjeldamine.

5.1. Kodulehe loomine raamistikuta

Sarnaselt isikliku veebilehe loomisele ilma kujundusraamistikuta kirjutatakse ka portaali veebilehe jaoks vajalikud stiilid omaloodud CSS faili, mis lisatakse kausta „css" ning kaasatakse veebilehele HTML lehe päises.

5.1.1. Põhilise ülesehituse loomine

Esimese sammuna luuakse põhiline ülesehitus - veebilehitseja akna keskel asetsev sisukonteiner laiusega 950px, 205px laiune vasak külgmenüü, 540px laiune sisuosa ning 205px laiune parem külgmenüü. Esmane ülesehitus on kujutatud joonisel 33.



Joonis 33. Portaali veebilehe ülesehitus.

Veebilehe „selgroo" HTML kood:

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="content_container">
        <div id="left_sidebar">LEFT</div>
        <div id="content">
            <div id="news">NEWS</div>
            <div id="gallery">GALLERY</div>
            <div id="events">EVENTS</div>
        </div>
```

```
        <div id="right_sidebar">RIGHT</div>
    </div>
    <div id="footer">FOOTER</div>
</div>
```

Portaali veebilehe ülesehitust sisaldav HTML kood on toodud Lisas 1 kaustas „portaal" failis *index1.html* ja CSS kood failis *style1.css*.

5.1.2. Põhikujunduse paika sättimine

Eelnevalt on sisuosa laiuseks määratud 540px ning kummagi külgmenüü laiuseks 205px - kokku 950px. Tekib probleem, kuna kujunduselemendina lisatakse sisuosale kummalegi poole 1px laiused äärised (*border*). Need aga lisavad laiusele juurde 2px ning kujunduse senine paigutus läheb paigast (joonis 34).

Joonis 34. Veebilehe alamjaotiste vale paigutus.

Seetõttu tuleb sisuosa laiust vähendada 2px võrra ehk sisuosa laiuseks on nüüdsest 538px. Loodud kujunduspõhi on nähtav joonisel 35.

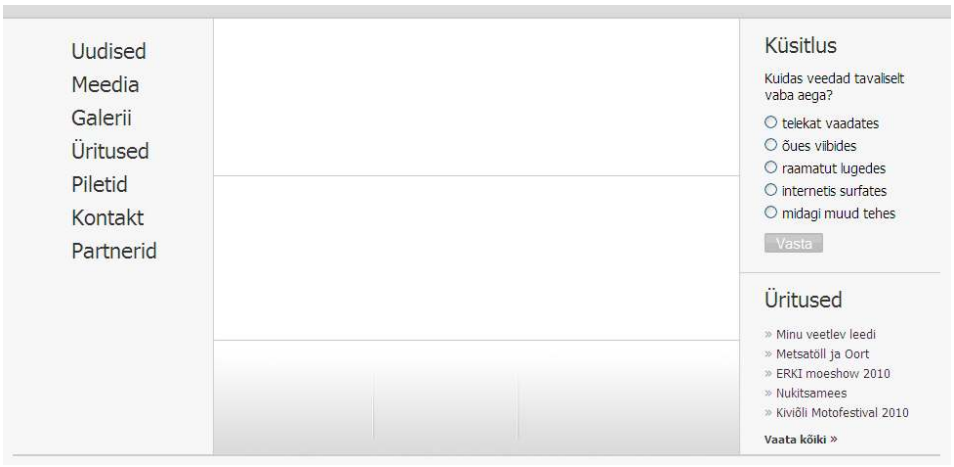
Joonis 35. Portaali veebilehe kujunduspõhi.

Põhikujunduse HTML kood on toodud Lisas 1 kaustas „portaal" failis *index2.html* ja CSS kood failis *style2.css*.

5.1.3. Sisu lisamine ning selle kujundamine

Külgmenüüd

Vasak külgmenüü kujutab endast navigatsiooni portaali eri alamlehtede vahel liikumiseks. Parem külgmenüü sisaldab küsitlust ning lühikest ülevaadet mõningatest lähiajal toimuvatest üritustest. Külgmenüüde kujundus on näidatud joonisel 36.

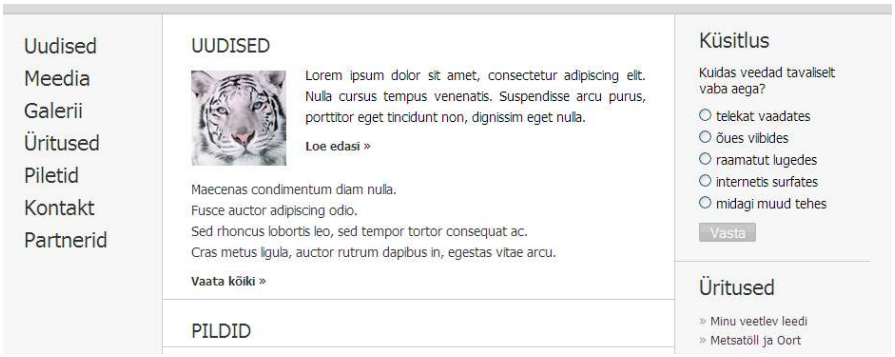


Joonis 36. Külgmenüüde kujundus.

Külgmenüüde HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-1.html* ja CSS kood failis *style3-1.css*.

Uudised

Uudiste rubriigi all kuvatakse ühest uudisest väike lõik koos pildiga ning veel viie uudise pealkirjad. Uudiste rubriigi kujundus on joonisel 37.



Joonis 37. Uudiste rubriigi kujundus.

Uudiste rubriigi HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-2.html* ja CSS kood failis *style3-2.css*.

Pildid

Piltide rubriik näitab ülevaadet mõningatest viimati lisatud piltidest, mida saab nuppudega edasi-tagasi kerida. Piltide rubriigi nuppude esialgne kujundus on näha joonisel 38.

Joonis 38. Piltide rubriigi nuppude esialgne kujundus.

Jooniselt 38 on näha, et eelmiste ja järgmiste piltide kerimise nupud ei ole õiged.

Nuppude kujundus on lahendatud nii, et viida sisuks on teksti kujul vastavalt kas „<” või „>” ning taustapildiga määratakse õige välimus. Üldjuhul ei saa aga viitade kõrgust ja laiust määrata ning seetõttu kuvataksegi taustapildist vaid nii palju, kui viita kirjutatud teksti tagant näha jääb.

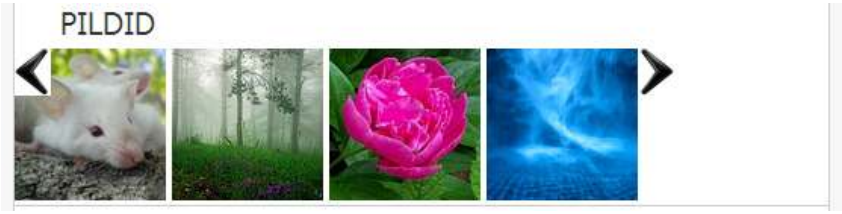
Kõigepealt on vaja teksti kujul kirjutatud nooled („<” ja „>”) ära peita ning seejärel taustapiltidel olevad nooled õige suurusega kuvada.

Tekstikujul noolte ära peitmiseks tuleb vastavatele viitadele lisada CSS reeglid `position: absolute;` ja `text-indent: -9999px;`. Neist esimene reegel muudab noole asukoha talle vastava mittestaatilise positsiooniga vanemelemendi suhtes absoluutseks ning viitasid kuvatakse neile määratud laiuse ja kõrgusega. Teine reegel määrab viitade tekstiosa taandeks -9999px, tänu millele jäävad tekstikujul viidad veebilehe nähtavast osast välja. Taande suurus ei pea olema just -9999px, vaid võib olla ka muu piisavalt väike arv, et tekstid veebilehel paistma ei jääks (ka veebilehe vaatamisel väga laia kuvariga).

Vastav CSS kood:

```
a.prev, a.next {
    width: 24px;
    height: 31px;
    position: absolute;
    text-indent: -9999px;
}
```

Nende reeglite lisamise järel on tekstikujul nooled peidetud ning õiged nooled nähtavad (joonis 39).



Joonis 39. Piltide rubriigi nuppude vale paigutus.

Järgmine samm on pildikujul nooled õigesse kohta paigutada. Seda saab teha absoluutselt positsioneeritud elemendi korral CSS reeglitega `top` ja `left`. Ehk antud juhul näeks nuppudele lisatud stiil välja järgmine:

```
a.prev, a.next {
    ...
    position: absolute;
    text-indent: -9999px;
    top: 75px;
}
a.prev {
    ...
    left: 30px;
}
a.next
    ...
    left: 500px;
}
```

Nende reeglite lisamise ning mõningate teiste stiilimuudatuste tegemise järel on pildigalerii kujundus paigas (joonis 40).

Joonis 40. Piltide rubriigi korrektne kujundus.

Pildigalerii kujundamise osa HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-3.html* ja CSS kood failis *style3-3.css*.

Üritused

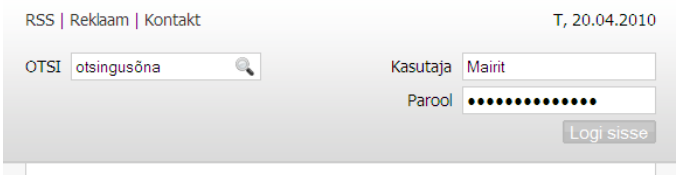
Ürituste rubriigis (joonis 41) on rippmenüüd ürituse tüübi, koha ja aja valimiseks.

Joonis 41. Ürituste rubriigi kujundus.

Ürituste rubriiki kirjeldav HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-4.html* ja CSS kood failis *style3-4.css*.

Päis

Loodud meelelahutuskeskkonna veebilehe päis (joonisel 42) sisaldab otsingukasti ning sisselogimisvõimalust.



Joonis 42. Päise kujundus.

Päise HTML kood on järgmine:

```
<div id="header">
    <div id="search">
        <p><a href="#">RSS</a>|<a href="#">Reklaam</a>|
           <a href="#">Kontakt</a></p>
        <span id="search_heading">Otsi</span>
        <input type="text" id="search_field" />
    </div>
    <div id="login">
        <p>T, 20.04.2010</p>
        <label>Kasutaja</label>
```

```
        <input type="text" id="login_user" name="login_user" /><br />
        <label>Parool</label>
        <input type="password" id="login_password" />
        <input type="submit" value="Logi sisse" class="button" />
    </div>
</div>
```

Otsingu ning sisselogimise eraldi konteineritesse asetamine on vajalik selleks, et neid teineteise kõrvale paigutada. Seejuures on oluline joondada sisselogimise konteiner ning selles sisalduv tekst paremale. Selleks vajalikud CSS reeglid:

```
#login {
    ...
    float: right;
    text-align: right;
}
```

Päise kujundamise osa HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-5.html* ja CSS kood failis *style3-5.css*.

Jalus

Veebilehe jalus sisaldab kontaktinfot, abi ja info viiteid ning partneritele vajalikke viiteid. Jaluse kujundus on joonisel 43.

Joonis 43. Jaluse kujundus.

Jaluse kujunduse HTML kood on toodud Lisas 1 kaustas „portaal" failis *index3-6.html* ja CSS kood failis *style3-6.css*.

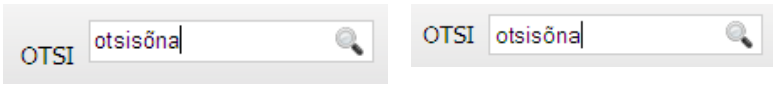
5.1.4. Brauseritevaheliste probleemide likvideerimine

Veebilehitsejas Mozilla Firefox 3.6.3, Google Chrome 4.1.249.1064, Safari 4.0.3, Opera 10.53 ja Internet Explorer 8 on portaali veebilehe kujundus korras.

Internet Explorer 7

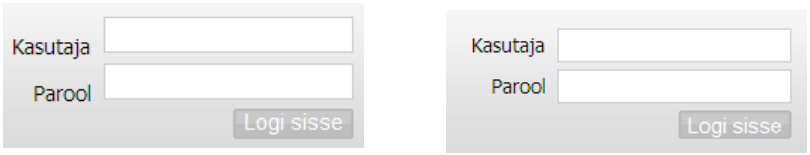
Veebilehitsejas Internet Explorer 7 leidub portaali veebilehe kujunduses viis ebakorrektsust:

1. Otsingukasti ees olev sõna „Otsi" on otsingukasti suhtes liiga all (joonis 44).



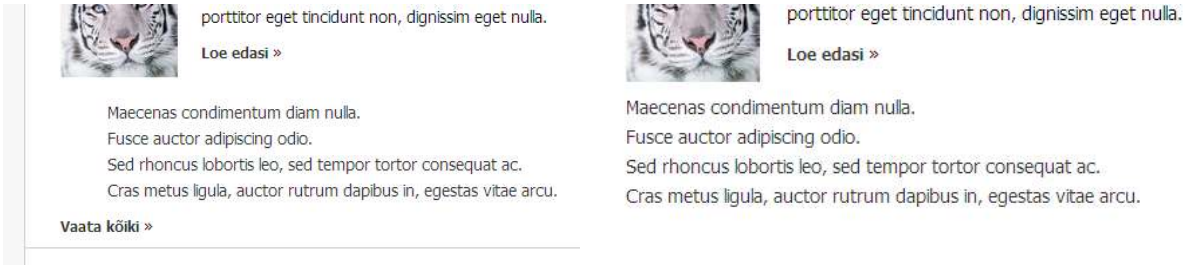
Joonis 44. Vasakul otsinguala brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

2. Sisselogimisala sildid „Kasutaja" ja „Parool" on tekstikastidega võrreldes liiga all (joonis 45).



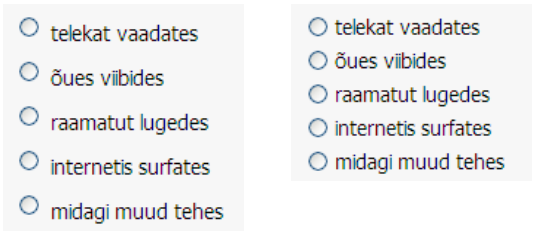
Joonis 45. Vasakul sisselogimisala brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

3. Uudisteploki nimekirjast vasakul on liiga palju ruumi (joonis 46).



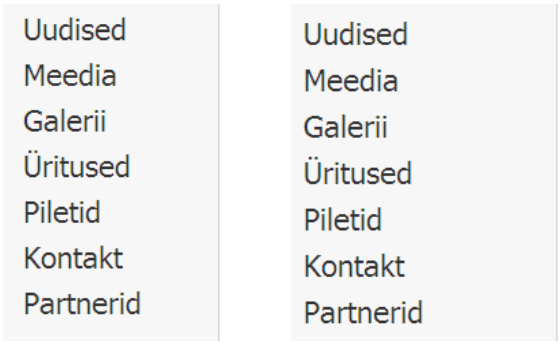
Joonis 46. Vasakul uudisteplokk brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

4. Küsitluse vastusevariandid asetsevad raadionupuga võrreldes liiga all ning nende vahel on liialt palju ruumi (joonis 47).



Joonis 47. Vasakul küsitlus brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

5. Navigatsioonimenüü on sisuosaga võrreldes liiga vasakul (joonis 48).



Joonis 48. Vasakul menüü brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

Esimesed kaks probleemi on võimalik lahendada järgmiselt:

```
<!--[if lte IE 7]>
    <style type="text/css">
        label, #search_heading { vertical-align: 4px; }
    </style>
<![endif]-->
```

Kolmanda probleemi lahendus: uudiste nimekirjale CSS reegli `margin-bottom: 0;` asemel `margin: 0;` kirjutamine kaotab ära liigse vaba ruumi nimekirjast vasakul.

Neljanda, küsitluse vastusevariantide paiknemise probleemi lahendamiseks tuleb muuta Internet Explorer 7 jaoks raadionuppude veeriseid:

```
<!--[if lte IE 7]>
    <style type="text/css">
        input[type="radio"] { margin: 2px 2px 1px 22px; }
    </style>
<![endif]-->
```

Eelnevalt oli CSS koodis kõigi sisendite veerised määratud lõiguga `input { margin: 5px 5px 5px 25px; }`.

Navigatsioonimenüü paiknevuse erinevus on tingitud menüü vooderduse vaikesätte erinevusest brauserites. Ülejäänud veebilehitsejates on vaikesättena menüüst vasakul 40px laiune vooderdus, Internet Explorer 7 aga mitte. Seetõttu tuleb menüü nimekirjale `#left_sidebar ul` lisada CSS reegel `padding-left: 40px;`.

Internet Explorer 7 jaoks tehtud muudatustega HTML kood on toodud Lisas 1 kaustas „portaal" failis *index4-1.html* ja CSS kood failis *style4-1.css*.

Internet Explorer 6

Internet Exploreri versioonis 6 leidub kaks üpriski silmatorkavat viga.

1. Pildigalerii edasi-tagasi kerimise nupud on paigast (joonis 49).

Joonis 49. Vasakul pildigalerii nupud brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

2. Ürituste rubriigi rippmenüüdest vasakul on liialt palju ruumi ning seetõttu paikneb ka otsingu nupp vales kohas (joonis 50).

Joonis 50. Vasakul ürituste rubriik brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

Esimese probleemi lahendab, kui Internet Explorer 6 puhul vähendada kummagi noole kaugust vasakult poolt 30px võrra ehk:

```
<!--[if lte IE 6]>
    <style type="text/css">
        a.prev { left: 0; }
        a.next { left: 470px; }
    </style>
<![endif]-->
```

Teise probleemi näol on tegemist Internet Explorer 6 topeltveerise veaga. Selle lahendamiseks tuleb rippmenüüsid sisaldavatele konteineritele (`events_sel`) lisada CSS reegel `display: inline;`.

Internet Explorer 6 muudatustega HTML kood on toodud Lisas 1 kaustas „portaal" failis *index4-2.html* ja CSS kood failis *style4-2.css*.

5.1.5. Kokkuvõtvad andmed

Portaali veebilehe näitajad:

- HTML faili suurus: 5.8KB
- CSS failide suurus: 5.3KB
- HTTP-päringuid: 14 päringut kogumahuga 148.6KB tühja vahemälu korral
- lehe laadimise kiirus: keskmiselt 0.325s tühja vahemälu korral (lehe 10 järjestikuse värskendamise ajad tühja vahemälu korral on toodud Lisas 4)
- koodi valideeruvus: nii HTML kood kui CSS kood on valideeruvad

Ilma CSS raamistikuta kodeeritud portaali veebileht on toodud Lisas 1 kaustas „portaal” - vastav HTML kood on failis *index.html* ja CSS kood failis *style.css*.

5.2. Kodulehe loomine Blueprint raamistikuga

Portaali veebilehe kodeerimine Blueprint raamistiku põhjal käib sarnaselt isikliku veebilehe kodeerimisele raamistikuga. HTML lehele kaasatakse raamistiku CSS failid *screen.css* ning veebilehitseja Internet Explorer korral *ie.css*, lisaks endakirjutatud stiililleht *style.css*.

5.2.1. Põhilise ülesehituse loomine

Vasak ning parem külgmenüü on kumbki 205px laiad ning sisuosa 540px lai. Et aga Blueprint CSS raamistikus täpselt selliste laiuste määramiseks vastavaid klasse ei leidu, tuleb mõningate konteinerite laiused määrata siiski isekirjutatud CSS koodiga. Seejuures päise nihutamiseks sisuosa keskele ei piisa `margin: 0 auto;` lisamisest, vaid selle asemel tuleb kasutada `margin-left: 205px;`.

Veebilehe "selgroo" HTML-kood:

```
<div class=”container”>
    <div id=”header” class=”span-24 last”>HEADER</div>
    <div id=”content_container” class=”span-24 last”>
        <div id=”left_sidebar”>LEFT</div>
        <div id=”content”>
            <div id=”news”>NEWS</div>
            <div id=”gallery”>GALLERY</div>
            <div id=”events”>EVENTS</div>
        </div>
        <div id=”right_sidebar”>RIGHT</div>
    </div>
    <div id=”footer” class=”span-24 last”>FOOTER</div>
</div>
```

Portaali kujunduse ülesehituse HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga” failis *index1.html* ja CSS kood failis *style1.css*.

5.2.2. Põhikujunduse paika panemine

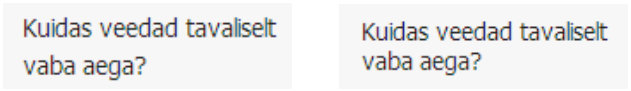
Portaali veebilehe põhikujunduse paika seadmine kulgeb sarnaselt ilma raamistikuta kodeerimisele.

Põhikujunduse HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga" failis *index2.html* ja CSS failis *style2.css*.

5.2.3. Sisu lisamine ning selle kujundamine

Külgmenüüde kodeerimisel Blueprint CSS raamistikku kasutades ilmneb neli erinevust.

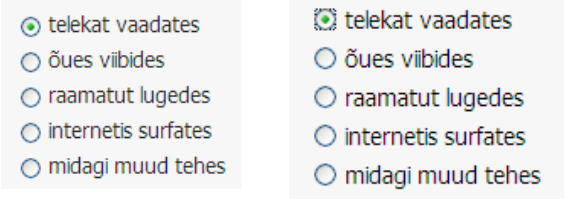
1. Et tekstide reavahe oleks ilma raamistikuta kodeerimisega võrreldes sama suur, tuleb lõigu reakõrguseks määrata 150% (`body{ line-height: 150%; }`), Blueprint raamistikus on selleks määratud 1.5. Joonisel 51 on võrreldud teksti väljanägemist reakõrgusega 150% ning 1.5.



Joonis 51. Vasakul tekst reakõrgusega 1.5, paremal tekst reakõrgusega 150%.

2. Parema külgmenüü ürituste nimekirja ridade ees on helehalli värviga noolekesed. Nende värvus on määratud klassiga quiet (`.quiet { color: #999; }`). Raamistikus on aga klass `quiet` juba olemas, kuid selle värvuseks on toon värvikoodiga #666, mistõttu tuleb ka raamistiku korral kirjutada ise klassi `quiet` õige tekstivärv.

3. Küsitluse vastusevariantide raadionupud asetsevad teksti suhtes liiga all (joonis 52). See tuleneb Blueprint raamistiku stiilist, kus raadionupule on määratud reeglid `input[type="radio"] { position: relative; top: .25em; }`. Kui kirjutada neist viimane üle reegliga `top: 1px;`, siis asetsevad raadionupud teksti suhtes õigesti.



Joonis 52. Vasakul raadionuppude vaikimisi paigutus CSS raamistikuga, paremal raadionuppude õige asetus.

4. Navigatsioonimenüü elementidest jääb vasakule mõnevõrra rohkem ruumi, kui ilma raamistikuta kodeerides. Põhjuseks see, et ilma raamistikuta kodeerides on nimekirja (`ul`) vasakpoolseks vooderduseks 40px, raamistiku korral aga 3.333em ehk 53px. Seetõttu tuleb nimekirja kohta käiv Blueprint raamistiku CSS reegel `padding-left: 3.333em;` üle kirjutada reegliga `padding-left: 40px;`.

Külgmenüüde HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga" failis *index3-1.html* ja CSS kood failis *style3-1.css*.

Sisuplokid

Uudisteploki, piltide rubriigi ning ürituste rubriigi kodeerimisel Blueprint CSS raamistiku põhjal olulisi erinevusi võrreldes ilma raamistikuta kodeerimisega ei tekkinud. Neid kolme sammu kirjeldavad HTML koodid on vastavalt toodud Lisas 1 kaustas „portaal_raamistikuga" failides *index3-2.html*, *index3-3.html*, *index3-4.html* ja CSS koodid failides *style3-2.css*, *style3-3.css*, *style3-4.css*.

Päis

Blueprint raamistikuga veebilehe päise kodeerimisel ei pea otsingu ja sisselogimise konteinerite laiuseid ise määrama, kuna CSS raamistikus on olemas sobiliku laiusega klassid, mida saab laiuse määramiseks HTML koodis kasutada. Päise HTML kood raamistikuga:

```
<div id="header" class="span-24 last">
    <div id="search" class="span-7 last">
        <p><a href="#">RSS</a>|<a href="#">Reklaam</a>|
           <a href="#">Kontakt</a></p>
        <span id="search_heading">Otsi</span>
        <input type="text" id="search_field" />
```

```
    </div>
    <div id="login" class="span-7 last">
        <p>T, 20.04.2010</p>
        <label>Kasutaja</label>
        <input type="text" id="login_user" /><br />
        <label>Parool</label>
        <input type="password" id="login_password"  />
        <input type="submit" value="Logi sisse" class="button" />
    </div>
</div>
```

Päise kujundamise osa HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga" failis *index3-5.html* ja CSS kood failis *style3-5.css*.

Jalus

Veebilehe jaluse kujundamisel Blueprint raamistikuga olulisi erinevusi võrreldes ilma raamistikuta kodeerimisega ei leidu.

Jaluse HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga" failis *index3-6.html* ja CSS kood failis *style3-6.css*.

5.2.4. Brauseritevaheliste probleemide likvideerimine

Veebilehitsejates Mozilla Firefox 3.6.3, Google Chrome 4.1.249.1064, Safari 4.0.3, Opera 10.53 ja Internet Explorer 8 on Blueprint CSS raamistiku põhjal kodeeritud portaali veebilehe kujundus korras.

Internet Explorer 7

Brauseris Internet Explorer 7 leidub neli kujunduslikku viga. Kolm probleemi on sarnased, nagu ilma raamistikuta kodeerideski:

1. Otsingukasti ees olev sõna „Otsi" on otsingukastiga võrreldes liiga all.
2. Sisselogimisala sildid „Kasutaja" ja „Parool" on tekstikastidega võrreldes liiga all.
3. Küsitluse vastusevariandid asuvad raadionupu suhtes liiga all ning nende vahel on liiga palju ruumi.

Lahendused kõigile neile probleemidele on samasugused, nagu ilma raamistikuta kodeerides.

Eelmainitud kolmele probleemile lisandub aga ka neljas - ürituste plokis asetseb otsingunupp rippmenüüdega võrreldes liiga all (joonis 53).

Joonis 53. Vasakul otsingunupp brauseris Internet Explorer 7, paremal brauseris Mozilla Firefox.

Lahendus on muuta veebilehitseja Internet Explorer 7 korral nupu ülemist veerist:

```
<!--[if lte IE 7]>
    <style type=”text/css”>
        #events input.button { margin-top: 29px; }
    </style>
<![endif]-->
```

Internet Explorer 7 jaoks tehtud muudatustega veebilehe HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga” failis *index4-1.html* ja CSS kood Lisas failis *style4-1.css*.

Internet Explorer 6

Kodeerimisel Blueprint raamistikku kasutades on brauseris Internet Explorer 6 täpselt samad kaks viga, mis ilma raamistikutagi. Ka lahendused neile probleemidele on samad.

Internet Explorer 6 parandustega HTML kood on toodud Lisas 1 kaustas „portaal_raamistikuga” failis *index4-2.html* ja CSS kood failis *style4-2.css*.

5.2.5. Kokkuvõtvad andmed

Blueprint CSS raamistiku põhjal loodud portaali veebilehe näitajad:

- HTML faili suurus: 6.2KB
- CSS failide suurus: 17.1KB
- HTTP-päringuid: 15 päringut kogumahuga 160.9KB tühja vahemälu korral
- lehe laadimise kiirus: keskmiselt 0.3529s (lehe 10 järjestikuse värskendamise ajad tühja vahemälu korral on toodud Lisas 5)

- koodi valideeruvus: nii HTML kood kui CSS kood on valideeruvad

Blueprint CSS raamistikuga kodeeritud meelelahutuskeskkonna veebileht on toodud Lisas 1 kaustas „portaal_raamistikuga" - vastav HTML kood on failis *index.html* ja CSS kood failis *style.css*.

6. Võrdlus

Käesolevas peatükis võrreldakse eelnevates peatükkides loodud veebilehtede näitajaid ilma kujundusraamistikuta ning Blueprint kujundusraamistikuga kodeerimisel - tekkinud probleemid, brauseritevahelised erinevused, koodi valideeruvus, HTML ja CSS failide suurus ning HTTP-päringute arv ja maht lehe laadimisel, lehe laadimise kiirus. Loodud näitelehtede kohta koostatud võrdluse põhjal tehakse kokkuvõte, kas kujundusraamistiku kasutamine veebilehe loomisel muudab veebilehe failide mahu oluliselt suuremaks ning seetõttu lehe laadimise aeglasemaks, või ei mõjuta see oluliselt veebilehe laadimise kiirust.

Loodud kahe veebilehe kummagi versiooni (nii ilma kujundusraamistikuta kui kujundusraamistikuga) HTML ning CSS failide koodid on valideeruvad. See tähendab, et süntaksivigadest tingitud probleeme veebilehtede visuaalses esituses ei leidunud. Küll aga esines nii isikliku kui portaali kodulehe loomise protsessi jooksul teisi probleeme.

Ilma raamistikuta kodeerides olid põhilisteks komistuskivideks mõningad pisivead (vajadus kasutada õige paigutuse saavutamiseks CSS reeglit `float: left;`, mis kippus ära ununema; teatud elementide veeriste-vooderduste muutmise vajadus) ning brauseritevahelised erinevused (Internet Exploreri topeltveerise viga).

Veebilehe loomisel kujundusraamistikku Blueprint kasutades esines sarnaseid probleeme, kuid lisaks tekkis kohati raamistiku poolt defineeritud CSS reeglite ülekirjutamise vajadus. See aga tähendas koodi mõnevõrra dubleerimist. Samas lahendas Blueprint CSS raamistiku Internet Exploreri stiilifaili lisamine mõningad sellele veebilehitsejale omased probleemid, mille lahendamiseks ilma raamistikuta rohkem vaeva pidi nägema. Portaali veebilehe loomisel Blueprint raamistikuga on aga juba näha, et väga eriliste lahenduste jaoks ei ole CSS raamistik ilmselt kõige otstarbekam - näiteks kui ülesehituse ühegi alamjaotise laiust ei ole võimalik raamistiku võrestiku klassiga määrata. Sellisel juhul tasub kaaluda raamistikust vaid mõne konkreetse osa kasutamist (näiteks veebilehitsejate vaikesätete nullimised).

Kummagi veebilehe loomisel raamistikuga ning raamistikuta on võrreldud järgmisi näitajaid - HTML ja CSS failide suurust ning lehe laadimise kiirust.

Isikliku veebilehe HTML failide suuruse erinevus ilma raamistikuta ning Blueprint CSS raamistikuga kodeerides on minimaalne - HTML failide suurused on vastavalt 5.2KB ja

5.1KB. CSS failide mahu erinevus on märkimisväärsem - ilma raamistikuta kodeerides on stiilifaili suurus 3.0KB, Blueprint raamistikku kasutades on stiilifailide suurus kokku 15.2KB. Seega on Blueprint CSS raamistiku põhjale loodud isikliku veebilehe stiilifailide kogumaht ligikaudu viis korda suurem kui ilma raamistikuta kodeerides. CSS failide suuruse erinevus mõjutab mõnevõrra ka HTTP-päringute arvu ning mahtu. Ilma CSS raamistikuta loodud isikliku veebilehe laadimisel tehakse tühja vahemälu korral veebilehitsejas Mozilla Firefox 12 HTTP-päringut kogumahuga 254.2KB, kujundusraamistiku Blueprint põhjal loodud veebilehe puhul aga 13 HTTP-päringut kogumahuga 266.3KB. Lehe laadimise kiirust eelmainitud tegurite erinevused aga oluliselt ei mõjuta - ilma kujundusraamistikuta kodeeritud isiklik veebileht laetakse tühja vahemälu korral keskmiselt 0.269s jooksul ning Blueprint raamistikuga kodeeritud leht keskmiselt 0.301s jooksul. See tähendab, et lehe laadimise keskmiste kiiruste erinevus on inimsilmale praktiliselt märkamatu.

Sarnaselt isiklikule veebilehele ei erine omavahel oluliselt ka meelelahutuskeskkonna veebilehe HTML failide suurused - ilma raamistikuta kodeerides 5.8KB ning Blueprint raamistikku kasutades 6.2KB. CSS failide suurused on vastavalt 5.3KB ning 17.1KB - Blueprint raamistikuga kodeerides on CSS failide kogumaht ligikaudu kolm korda suurem. HTTP-päringute arv lehe laadimisel tühja vahemälu korral erineb ühe võrra - ilma raamistikuta tehakse 14 päringut kogumahuga 148.6KB, raamistikuga 15 päringut kogumahuga 160.9KB. Sarnaselt esimesele näitele mõjutavad need erinevused lehe laadimise kiirust üsna minimaalselt - ilma raamistikuta valmistatud veebileht laetakse keskmiselt 0.325s jooksul, Blueprint raamistiku põhjal loodud leht 0.3529s jooksul. Ka portaali veebilehe korral on andmete laadimise kiiruste erinevus vaatajale praktiliselt märkamatu.

Eelnevalt välja toodud andmete võrdluse põhjal on võimalik ümber lükata väide, et CSS raamistiku kasutamine aeglustab rohkemate HTTP-päringute ning failide suurema kogumahu tõttu veebilehe laadimist. Seevastu veebilehe kodeerimise kiirusele aitab CSS raamistiku kasutamine kaasa - tänu nullimiste ning Internet Exploreri stiilifailidele jäävad mitmed probleemid tekkimata, seega ei kulu ka nende parandamisele aega. See tähendab, et veebilehe kodeerijad võivad töö kiiremaks ja mugavamaks muutmise nimel julgesti kasutada CSS raamistikku, kartmata, et see failide mahu nii suureks ajab, et veebilehe laadimine oluliselt aeglasemaks muutub.

Kokkuvõte

Käesolevas töös esitati ülevaade veebilehtede kujundamiseks mõeldud CSS keelest ning CSS raamistikest. Kirjeldati CSS raamistike olemust ja võimalusi, eeliseid ja puuduseid.

Töö eesmärk oli praktiliste näidete põhjal välja selgitada võimalikud erinevused ja probleemid veebilehe loomisel ilma kujundusraamistikuta ning kujundusraamistikuga. Selleks loodi bakalaureusetöö raames kaks näitelehte - üks lihtsama ning teine keerulisema ülesehitusega. Mõlemad veebilehed kodeeriti esmalt ilma CSS raamistikuta ning seejärel vabavaralist CSS raamistikku Blueprint kasutades. Eelnevalt fikseeriti ka nõuded, millele loodavad veebilehed vastama pidid.

Näitelehtede loomisel kirjeldati tekkinud olulisemaid probleeme ning otsiti neile lahendused. Valminud veebilehtedel uuriti HTML ja CSS failide suurust ning failide HTTP-päringute kogumahtu ja lehe laadimise kiirust. Seejärel võrreldi omavahel ilma raamistikuta ning raamistikuga valmistatud veebilehtede andmeid.

Bakalaureusetöö käigus loodud näitelehtede kohta koostatud võrdluse põhjal sai teha järeldused, et CSS raamistik ei muuda veebilehe andmete mahtu nii palju suuremaks, et see lehe laadimise kiirust mõjutaks. See tähendab, et kui on huvi või vajadus veebilehe loomisel CSS raamistikku kasutada, siis ei pea muretsema, et see andmetehulga väga suureks või lehe laadimise aeglaseks muudab.

Using a CSS framework for designing a website

Bachelor thesis (6EAP)

Mairit Vikat

Abstract

The present Bachelor's thesis presents an overview of the creation of websites using HTML and CSS languages and the freeware CSS framework Blueprint. The purpose of the thesis was to examine the problems that may occur during the creation of a website with and without a CSS framework.

To that end, two examples of websites were created – a personal website with a simple layout and another website with a more complex layout. Both websites were first created without a CSS framework and then by the aid of Blueprint CSS framework.

After creating these websites, some parameters of each website were analysed – the size of the HTML and CSS files, the total volume of HTTP requests and the loading speed of the page.

Then the parameters of websites made without the framework and with Blueprint CSS framework were compared.

The comparison indicates that a CSS framework does not change the total volume of HTTP requests enough to affect the loading speed of a website. However it does speed up the process of coding a website's design. This means that the coders may use a CSS framework to change the process of coding a website's design quicker and more effective without having to worry that the use of a framework would make the website load slower.

Kasutatud kirjandus

1. 960 Grid System http://960.gs/ (1.juuni 2010)

2. A. Van Wagener. *The Grid: The Structure of Design*, 2003. http://www.poynter.org/column.asp?id=47&aid=37529 (1.juuni 2010)

3. Blueprint: A CSS Framework http://www.blueprintcss.org/ (1.juuni 2010)

4. Cascading Style Sheets http://en.wikipedia.org/wiki/Cascading_Style_Sheets (1.juuni 2010)

5. C. Chapman. *Choosing The Best CSS Framework: A Complete Guide*, 2010. http://devsnippets.com/article/complete-guide-to-css-frameworks.html (1.juuni 2010)

6. C. Coyier. *IE CSS Bugs That'll Get You Every Time*, 2008. http://css-tricks.com/ie-css-bugs-thatll-get-you-every-time/ (1.juuni 2010)

7. C. Coyier. *CSS Frameworks Roundup (and some thoughts)*, 2007. http://css-tricks.com/css-frameworks-roundup-and-some-thoughts/ (1.juuni 2010)

8. C. Coyier. *What Are The Benefits of Using a CSS Framework?,* 2008. http://css-tricks.com/what-are-the-benefits-of-using-a-css-framework/ (1.juuni 2010)

9. C. Dang. *Pros and Cons of Using CSS Framework in DotNetNuke*, 2009. http://dnngallery.net/blog/articletype/articleview/articleid/140/pros-and-cons-of-using-css-framework-in-dotnetnuke (1.juuni 2010)

10. CSS Framework http://en.wikipedia.org/wiki/CSS_framework (1.juuni 2010)

11. CSS Validations http://www.tutorialspoint.com/css/css_validations.htm (1.juuni 2010)

12. Emastic – CSS Framework http://code.google.com/p/emastic/ (1.juuni 2010)

13. E. Meyer. *Reset Reasoning*, 2007. http://meyerweb.com/eric/thoughts/2007/04/18/reset-reasoning/ (1.juuni 2010)

14. Grid (page layout) http://en.wikipedia.org/wiki/Grid_%28page_layout%29 (1.juuni 2010)

15. H. Jeret. *CSS raamistikud*, 2009. http://weebomagazine.com/htmlcss-raamistikud-frameworks/ (1.juuni 2010)

16. HTML http://en.wikipedia.org/wiki/HTML (1. juuni 2010)

17. IE PNG Alpha Fix v2.0 Alpha 4 http://www.twinhelix.com/css/iepngfix/demo/ (1.juuni 2010)

18. IE 7 Quirks: Floats and Margins, Here We Go Again
http://www.maratz.com/blog/archives/2006/11/11/ie-7-quirks-floats-and-margins/ (1. juuni 2010)

19. J. Christopher. *Beautify Your Print CSS*, 2006.
http://mondaybynoon.com/2006/05/01/beautify-your-print-css/ (1.juuni 2010)

20. J. Storimer. *A Closer Look At the Blueprint CSS Framework*, 2008.
http://net.tutsplus.com/tutorials/html-css-techniques/a-closer-look-at-the-blueprint-css-framework/ (1.juuni 2010)

21. Media types http://www.w3.org/TR/CSS2/media.html (1.juuni 2010)

22. N. Smith. *960 Grid System*, 2008.
http://sonspring.com/journal/960-grid-system/ (1. juuni 2010)

23. Page Layout http://en.wikipedia.org/wiki/Page_layout (1.juuni 2010)

24. PNG Files Do Not Show Transparency in Internet Explorer
http://support.microsoft.com/kb/294714 (1.juuni 2010)

25. PNG-Transparency for Windows IE 5.5 & 6
http://jquery.andreaseberhard.de/pngFix/ (1.juuni 2010)

26. Quick-Start Tutorial
http://wiki.github.com/joshuaclayton/blueprint-css/quick-start-tutorial (1.juuni 2010)

27. R. Andrew. *The CSS Anthology: 101 Essential Tips, Tricks & Hacks, 3rd Edition*, Sitepoint, 2009.

28. R. Dash. *Which CSS Grid Framework Should You Use for Web Design?*, 2008.
http://net.tutsplus.com/html-css-techniques/which-css-grid-framework-should-you-use-for-web-design/ (1.juuni 2010)

29. The IE5/6 Doubled Float-Margin Bug
http://www.positioniseverything.net/explorer/doubled-margin.html (1.juuni 2010)

30. W3Counter – Global Web Stats
http://www.w3counter.com/globalstats.php?year=2009&month=12 (1.juuni 2010)

31. Yet Another Multicolumn Layout – An (X)HTML/CSS Framework
http://www.yaml.de/en/ (1.juuni 2010)

Lisad

Lisa 1 – Koodinäited (CD-plaadil)

Lisa 2 – Raamistikuta loodud isikliku kodulehe laadimise ajad

Lisa 3 – Raamistikuga loodud isikliku kodulehe laadimise ajad

Lisa 4 – Raamistikuta loodud portaali kodulehe laadimise ajad

Lisa 5 – Raamistikuga loodud portaali kodulehe laadimise ajad

Lisa 1 – Koodinäited

Tööga on kaasas CD-plaadil olevad koodinäited.

Kaustas „isiklik” on toodud lihtsa struktuuriga kodulehe ilma raamistikuta loomise osas (peatükk 4.1) loodud failinäited ning lõplik tulemus on failides *index.html* ja *style.css*.

Kaustas „isiklik_raamistikuga” on toodud personaalse kodulehe Blueprint raamistikuga loomise osas (peatükk 4.2) loodud failinäited. Lõplik tulemus on failides *index.html* ja *style.css*.

Kaustas „portaal” on toodud keerulisema struktuuriga kodulehe ilma raamistikuta loomise osas (peatükk 5.1) loodud failinäited. Lõplik tulemus on failides *index.html* ja *style.css*.

Kaustas „portaal_raamistikuga” on toodud keerulisema struktuuriga kodulehe Blueprint raamistikuga loomise osas (peatükk 5.2) loodud failinäited ning lõplik tulemus on failides *index.html* ja *style.css*.

Lisa 2 – Raamistikuta loodud isikliku kodulehe laadimise ajad

Ilma CSS raamistikuta loodud lihtsama struktuuriga veebilehe laadimise keskmise kiiruse arvestamise jaoks (peatükk 4.1.) mõõdetud ajad lehe kümnel järjestikusel värskendamisel veebilehitsejas Mozilla Firefox tühja vahemälu korral olid vastavalt:

1. 0.349s
2. 0.23s
3. 0.233s
4. 0.244s
5. 0.234s
6. 0.249s
7. 0.289s
8. 0.262s
9. 0.277s
10. 0.252s

Lisa 3 – Raamistikuga loodud isikliku kodulehe laadimise ajad

Blueprint CSS raamistikuga loodud lihtsama ülesehitusega veebilehe laadimise keskmise kiiruse arvestamise jaoks (peatükk 4.2.) mõõdetud ajad lehe kümnel järjestikusel värskendamisel veebilehitsejas Mozilla Firefox tühja vahemälu korral olid vastavalt:

1. 0.319s
2. 0.29s
3. 0.283s
4. 0.307s
5. 0.308s
6. 0.277s
7. 0.303s
8. 0.313s
9. 0.301s
10. 0.309s

Lisa 4 – Raamistikuta loodud portaali kodulehe laadimise ajad

Ilma CSS raamistikuta loodud keerulisema ülesehitusega veebilehe laadimise keskmise kiiruse arvestamise jaoks (peatükk 5.1.) mõõdetud ajad lehe kümnel järjestikusel värskendamisel veebilehitsejas Mozilla Firefox tühja vahemälu korral olid vastavalt:

1. 0.366s
2. 0.306s
3. 0.301s
4. 0.318s
5. 0.346s
6. 0.377s
7. 0.36s
8. 0.217s
9. 0.318s
10. 0.346s

Lisa 5 – Raamistikuga loodud portaali kodulehe laadimise ajad

Blueprint CSS raamistikuga loodud keerulisema struktuuriga veebilehe laadimise keskmise kiiruse arvestamise jaoks (peatükk 5.2.) mõõdetud ajad lehe kümnel järjestikusel värskendamisel veebilehitsejas Mozilla Firefox tühja vahemälu korral olid vastavalt:

1. 0.268s
2. 0.361s
3. 0.344s
4. 0.351s
5. 0.408s
6. 0.222s
7. 0.412s
8. 0.382s
9. 0.4s
10. 0.381s