TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

# Paindliku veebidisaini arendamine weprint.ee näitel

Magistritöö

Üliõpilane: Renald Neero
Üliõpilaskood: 104741IAPMM
Juhendaja: Natalia Järv

Tallinn
2014

# Autorideklaratsioon

Kinnitan, et olen koostanud selle lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(*kuupäev*) (*allkiri*)

# Annotatsioon

Käesoleva magistritöö eesmärk on võrrelda paindliku veebidisaini loomise erinevaid lähenemisi: arendamine nullist ja arendamine, kasutades raamistikke. Lisaks eelmainitule soovitakse tuvastada ja analüüsida probleeme, mis tekivad erinevate lähenemiste kasutamisel. Töös käsitletavad metoodikad on autori poolt põhjalikult uuritud ning läbiproovitud.

Paindliku veebilehe loomiseks tuleb lahendada järgmised probleemid: veebilehe raamistiku loomine sõltuvalt ekraanisuurusest, küljenduse muutmine sõltuvalt ekraanisuurusest, veebilehele lisatud piltide muutmine sõltuvaks ekraanisuurusest ning veebilehe graafika kvaliteedi säilimine erinevate ekraanisuuruste korral.

Tööst saab analüüsitud ülevaate paindliku veebilehe loomisel tekkivatest probleemidest ning nende lahendamisest. Erinevate lähenemisviiside sobilikkust analüüsitakse lähtuvalt teostatava projekti eesmärkidest. Töö tulemiks on paindlikke põhimõtteid järgiva veebilehe weprint.ee valmimine ning analüüs raamistike kasutamise eelistest ja puudustest sellise veebilehe loomisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 59 leheküljel, 6 peatükki, 3 joonist ja 3 tabelit.

# Abstract

The aim of this Master's thesis is to compare different development approaches of responsive web design. There will be comparison between starting from scratch and development using frameworks. In addition to that there is problem analysis of different approaches. The methods used in this Master's thesis have been tested by the author.

The issues that need to be resolved for developing responsive web design are following: making website containers dependent on screen size, layout relating on screen size, making pictures responsive and sustaining quality of the graphics for different layouts and screen sizes.

This Master's thesis strives to give an analysed overview of creating a responsive website and tries giving answers to the questions that may appear. Suitability of different approaches are analysed regarding the aims of the project. The result of the project is weprint.ee website that has been developed considering the aspects and standards of responsive web design.

The thesis is in Estonian and contains 59 pages of text, 6 chapters, 3 figures and 3 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| **Paindlik veebidisain** | ***Responsive web design (RWD)***<br>Veebilehe loomine selliselt, et veebileht adapteeruks erinevate seadmete ekraane arvestavalt, pakkudes kasutajale head vaatamise kogemust. |
| **Mobiilne seade** | ***Mobile device***<br>Mobiilseteks seadmeteks peetakse tänapäeva mobiiltelefone ja tahvelarvuteid. |
| **Raamistik** | ***Framework***<br>Valmis loodud komponentide kogum, mida on võimalik korduvkasutada, et lihtsustada arendamist. |
| **Küljendus** | ***Layout***<br>Sektsioonide paiknemine veebilehel. |
| **Muutlik küljendus** | ***Fluid layout***<br>Küljendus, mille korral kasutatakse sektsioonide suuruse määramiseks protsente. |
| **Fikseeritud küljendus** | ***Fixed layout***<br>Küljendus, mille korral kasutatakse sektsioonide suuruse määramiseks piksleid. |
| **Elastne küljendus** | ***Elastic layout***<br>Küljendus, mille korral kasutatakse sektsioonide suuruse määramiseks *em*'e. |
| ***Media query*** | ***Media query***<br>CSS spetsifikatsioonis kirjeldatud avaldis, mille abil on võimalik kirjutada CSS reegleid sõltuvalt seadme omadustest. |
| **Suhtelised ühikud** | ***Relative sizes***<br>Suhtelised ühikud on sõltuvuses vaateala suurusest. Sellisteks ühikuteks |

| | |
|---|---|
| | on protsent ja *em*. |
| **Absoluutsed ühikud** | ***Absolute sizes*** Absoluutsed ühikud ei sõltu ülemelemendi suurusest. Selliseks ühikuks on piksel. |
| ***Foundation*** | ***Foundation*** Raamistik paindliku veebidisaini arendamiseks. |
| ***Bootstrap*** | ***Bootstrap*** Raamistik paindliku veebidisaini arendamiseks. |
| **Vaateala** | ***Viewport*** Ala suurus, mida veebilehitseja kasutab veebilehe kuvamiseks. |
| **Hüppeala** | ***Break point*** Vahemik, mille korral kehtivad kindlad *media query* reeglid. |
| **Mobiilile esmakohal** | ***Mobile first*** Lähenemine, mille puhul arendatakse veebileht, pidades esmakohal silmas mobiilseid seadmeid. |
| **Lauaarvutile esmakohal** | ***Desktop first*** Lähenemine, mille puhul arendatakse veebileht, pidades esmakohal silmas süle- ja lauaarvuteid. |
| **Jaotusel põhinev küljendus** | ***Grid-based layout*** Dünaamiline küljendus, mille korral on võimalik veebilehe sektsioone ümber paigutada. |
| **HTML keha** | ***HTML body*** HTML-i spetsifikatsioonis tutvustatud märgend, mis defineerib veebilehe sisu osa. |

# Jooniste nimekiri

Joonis 1. Fikseeritud küljendus ..... 15
Joonis 2. Muutlik küljendus ..... 16
Joonis 3. Suurendus sõltuvalt pikslitihedusest ..... 19

# Tabelite nimekiri

Tabel 1. Weprint.ee külastajate resolutsioon ja resolutsiooni kasutamise protsent.....21
Tabel 2. Weprint.ee kasutamise seadmete statistika.....22
Tabel 3. *Interchange*-nimetusega päringud.....44

# Sisukord

1. Sissejuhatus ..... 11
1.1 Taust ja probleem ..... 11
1.2 Ülesande püstitus ..... 11
1.3 Metoodika ..... 12
1.4 Ülevaade tööst ..... 12
2. Paindliku veebidisaini rakendamine ..... 13
2.1 Mis on paindlik veebidisain ..... 13
2.2 Paindliku veebidisaini põhitõed ..... 13
2.3 Küljenduse tüübid ..... 14
2.3.1 Fikseeritud küljendus ..... 14
2.3.2 Muutlik küljendus ..... 15
2.3.3 Elastne küljendus ..... 16
2.3.4 Hübriidküljendus ..... 17
2.4 Meta-märgend *viewport* ..... 17
2.5 Ekraani pikslitihedus ..... 18
*2.6 CSS media query* ..... 19
2.7 Lähenemised paindliku veebidisaini loomisel ..... 20
2.7.1 Mobiilile esmakohal ..... 20
2.7.2 Lauaarvutile esmakohal ..... 20
2.8 Olemasolevad raamistikud ..... 21
2.9 Veebilehe weprint.ee arendamise lähenemise valik ..... 21
3. Weprint.ee arendamine, kasutades HTML5-e ja CSS3-e võimalusi ..... 23
3.1 Resolutsioonist sõltuvate olekute kirjeldamine ..... 24
3.1.1 Avalehe olekud ..... 24
3.1.2 Alalehtede olekud ..... 25
3.2 Veebilehe raamistiku loomine ..... 26
3.3 Erinevate olekute loomine, kasutades *media query*'t ..... 29
3.4 Tulemuste ülevaade ..... 31
*4. Bootstrap* ..... 33
4.1 Kuidas on *Bootstrap* üles ehitatud ..... 33

4.2 Alustamise keerukus ..... 35
4.3 Kasutamise keerukus ..... 36
4.4 Võrdlus nullist lähenemisega ..... 37
*5. Foundation* ..... 40
5.1 Kuidas on *Foundation* üles ehitatud ..... 40
5.2 Alustamise keerukus ..... 45
5.3 Kasutamise keerukus ..... 45
5.4 Võrdlus *Bootstrap*'i ja nullist arendamisega ..... 47
6. Kokkuvõte ..... 49
Summary ..... 51
Kasutatud kirjandus ..... 53
Lisa 1. Avaleht vaateala laiusega 1366 px ..... 54
Lisa 2. Avaleht vaateala laiusega 766 px ..... 55
Lisa 3. Avaleht vaateala laiusega 320 px ..... 56
Lisa 4. Alalehe 1. küljendus vaateala laiusega 1366 px ..... 57
Lisa 5. Alalehe 2. küljendus vaateala laiusega 1366 px ..... 58
Lisa 6. Alalehe 3. küljendus vaateala laiusega 1366 px ..... 59

# 1. Sissejuhatus

Järjest enam inimesi kasutab nutitelefone ja tahvelarvuteid veebilehtede lehitsemiseks. Mobiilsete seadmete kasutamise trend on olnud kasvav ja prognoosid näitavad, et see nii ka jätkub (vt [1]). Sellest tulenevalt on muutunud väga oluliseks, et veebilehe sisu oleks mugavalt loetav ka palju väiksematel ekraanidel kui siiani kasutatud laua- ja sülearvutite puhul. Probleemi üheks lahenduseks on veebileht, mis suudab erinevate seadmete korral veebilehel olevat informatsiooni kuvada optimaalsel viisil. Lahendus on paindlik veebidisain ehk *responsive web design*.

## 1.1 Taust ja probleem

Veebilehe põhieesmärk on informatsiooni edastamine kasutajale. Järjest populaarsemate mobiilsete seadmete levik on tinginud olukorra, kus veebileht, mis on loodud ainult ühte ekraanisuurust arvestavalt, ei ole enam piisav, pakkumaks kasutajatele optimaalset veebilehitsemist. Väikese ekraani puhul ei ole kasutajal mugav veebilehte horisontaalse ja vertikaalse kerimisriba abil kasutada. Veebilehe kuvamisel väiksemate ekraanisuurustega suumivad veebilehitsejad veebilehe välja – tulemuseks on tekstide loetamatus. Mobiilsete seadmete puhul on andmemahu säästmise eesmärgil vajalik eristada olulist ebaolulisest.

Töö on kasulik kõigile, kes plaanivad hakata looma paindliku veebidisaini põhimõtteid järgivat veebilehte. Tööst leiavad kasulikku informatsiooni nii arendajad kui ka veebilehe tellijad.

## 1.2 Ülesande püstitus

Magistritöö eesmärk on analüüsida ja võrrelda paindliku veebidisaini loomise meetodeid. Arendatava veebilehe weprint.ee näitel soovitakse jõuda tulemuseni, kus erinevate ekraanisuurustega seadmetel esitatakse informatsiooni optimaalseimal viisil. Töös võrreldakse erinevate lähenemiste eeliseid ja puudusi ning analüüsitakse erinevate lähenemiste sobivust lihtsamate ja keerukamate infosüsteemide arendamisel. Weprint.ee veebilehe planeerimisel seati eesmärgiks toetada järgmisi seadmeid: nutitelefonid, tahvel-, süle- ja lauaarvutid.

## 1.3 Metoodika

Peatükis 1.2 mainitud veebilehe arendusel kasutatakse paindliku veebilehe loomise meetodeid. Sellist veebilehte saab luua nullist ehk kirjutada ise kogu HTML ja CSS või kasutada raamistikke. Täpsemalt kirjeldatakse nullist arendamise metoodikat peatükis 2 ja raamistike kasutamist peatükkides 4 ja 5. Töös analüüsitakse erinevate lähenemiste eeliseid ja puudusi ning analüüsitakse probleeme, mida tuleb lahendada sõltuvalt lähenemisest.

## 1.4 Ülevaade tööst

Peatükis 2 antakse ülevaade, mis on paindlik veebidisain ja mida peab teadma, kui sellist veebilehte soovitakse luua. Lisaks antakse praktilisi juhiseid, mida peab kindlasti tegema ja millele mõtlema, et saaks paindlikku veebilehte arendama hakata.

Peatükis 3 kasutatakse paindliku veebidisaini arendamise põhimõtteid weprint.ee näitel. Alustatakse veebilehe raamistiku loomisest ning jõutakse veebilehe erinevate küljenduste loomiseni. Lisaks tuuakse välja nullist arendamise eelised ja puudused.

Peatükis 4 tutvustatakse *Bootstrap*-raamistikku ning analüüsitakse *Bootstrap*'i kasutamise eeliseid ja puudusi paindliku veebidisaini rakendamisel. Weprint.ee näitel võrreldakse raamistiku kasutamise sobilikkust erinevate küljenduste loomiseks. Võrdlus toimub nullist arendamisega.

Peatükis 5 tutvustatakse *Foundation*-raamistikku ning analüüsitakse *Foundation*'i eeliseid ja puudusi paindliku veebidisaini rakendamisel. Erinevate küljenduste loomise võimalusi analüüsitakse weprint.ee näitel. Lisaks võrreldakse *Foundation*'i kasutamist *Bootstrap*'iga ja nullist arendamisega.

Peatükis 6 tuuakse välja magistritöö kõige olulisemad tulemused ning põhitulemuste loetelu, kirjeldatakse olulisemad järeldused ning tuuakse välja võimalikud edasiarendused.

# 2. Paindliku veebidisaini rakendamine

Paindliku veebidisaini rakendamiseks on vaja kõigepealt aru saada, mis see on. Rõhk on veebilehe sisu kuvamisel. Oluline on läbi mõelda küljendus erinevate ekraanimõõtmete korral. Väiksemate ekraanisuuruste korral ei ole võimalik kõike kuvada ja tuleb teha eelistus lähtuvalt kasutaja seisukohast – kuvada kõige olulisemat informatsiooni. Kõigest sellest räägitakse põhjalikumalt järgnevates alapeatükkides.

## 2.1 Mis on paindlik veebidisain

Paindliku veebidisainiga veebileht adapteerub ekraanisuurusega, millele see kuvatakse. Lihtsuse huvides kasutatakse selles töös mõistet „ekraanisuurus“, peatükis 2.4 tutvustatakse vaateala suuruse mõistet ja edaspidi kasutatakse erinevate seadmete kirjeldamiseks just seda mõistet. Põhirõhk on sisul ja selle kasutajale kuvamisel. Erinevatel ekraanidel paigutub sisu ümber ja mõnda sektsiooni ei kuvata. Paindlikku veebidisaini tutvustas esmakordselt 2010. aasta mais Ethan Marcotte oma asjakohases artiklis [2, p. 11]. Paindlik veebidisain koosneb kolmest peamisest osast [3, p. 9]:

1. paindlik, jaotusel põhinev küljendus (*grid-based layout*);
2. paindlikud pildid;
3. *media queries*, CSS3 spetsifikatsioonis kirjeldatud avaldis.

Neid kolme punkti rakendades saadakse paindliku veebidisainiga veebileht.

## 2.2 Paindliku veebidisaini põhitõed

Veebilehte kujundades tuleb kasutada võimalikult palju CSS-i võimalusi [4, p. 24], võimalusel tuleb vältida piltide kasutamist. See tagab kujunduse korrektse väljanägemise sõltumata ekraanisuurusest.

Kirja suuruse määramiseks tuleb kasutada absoluutsete ühikute (px ja pt) asemel suhtelisi ühikuid, kas *em*'e või protsente. Nii saab arendaja kirjasuurusi erinevate küljendusvaadete

jaoks muuta proportsionaalselt, muutes vaid baaskirjasuurust. Absoluutseid ühikuid kasutades tuleb kirjasuuruse muudatused teha ükshaaval ning jälgida, et proportsioonid jääksid samaks.

Sektsioonide suuruste määramiseks tuleb kasutada suhtelisi ühikuid. Veebilehe küljendamiseks ei tohi kasutada *table*-märgendit. Mõlema tingimuse täitmisel saab veebilehe küljendust kirjeldada täielikult CSS-i reeglitega.

Veebilehele pilte lisades tuleb tagada, et need sõltuksid ekraanisuurusest. Tuleb arvestada, et piltide suurus erinevatel seadmetel varieerub, seda kirjeldatakse täpsemalt peatükis 2.5. Algupärase pildi suuruse muutmisega kaasneb pildi kvaliteedi vähenemine.

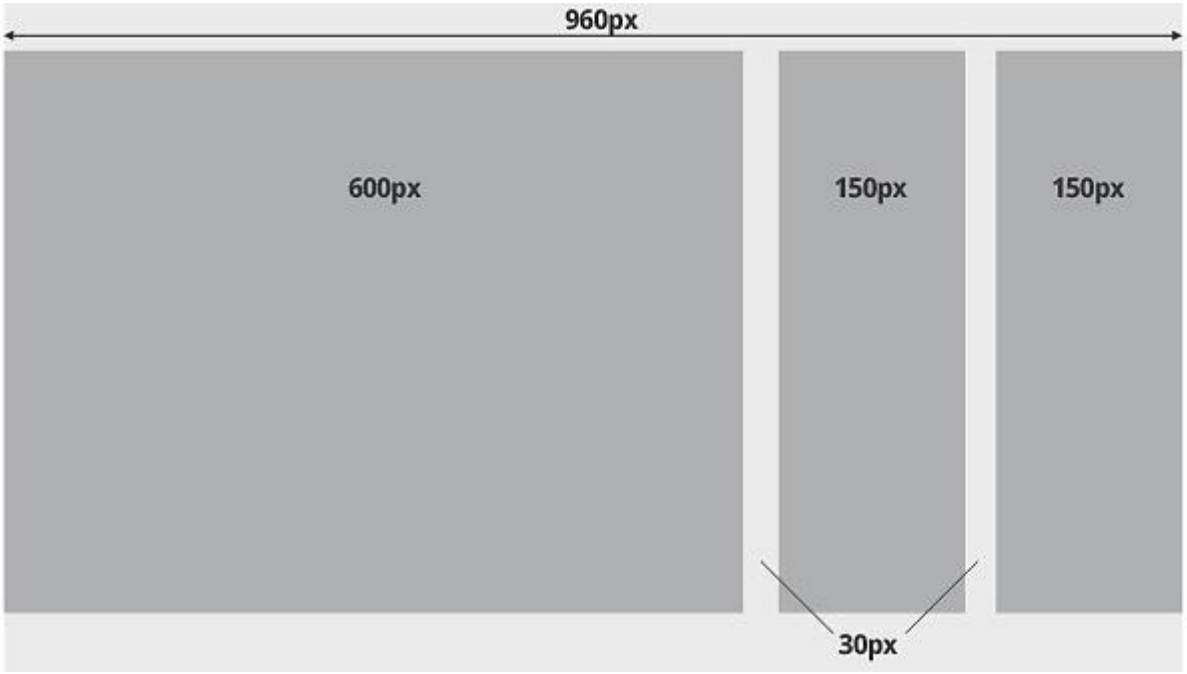
Paindlikku veebilehte luues saab tagada paindliku käitumise ka vanematel veebilehitsejatel. *Min-width* ja *max-width* CSS-i omadus ei ole kõikide veebilehitsejate puhul toetatud, kuid toe lisamiseks saab kasutada Respond.js'i (vt [5]). Kui vanemal veebilehitsejal ei ole JavaScript toetatud või see on välja lülitatud, siis paindlik veebileht ei tööta.

## 2.3 Küljenduse tüübid

Veebilehe alustalaks on küljendus ehk sisu jaotus. Läbimõtlemata või valesti valitud küljendus põhjustab veebilehe arendamisel suuri probleeme. Küljendus paneb paika aluse, mille peale veebilehte arendama hakatakse. Oluline on teada, milliseid sektsioone veebilehel kuvatakse, kuidas need omavahel käituvad ning kuidas need erinevate ekraanisuuruste korral ümber paiknevad. Erinevate ekraanisuurustega seadmeid on juba praegu palju, variatsioonide hulk kasvab pidevalt. Lisaks ekraanisuurusele tuleb arvestada, et kasutajad muudavad oma veebilehitseja suurust – enamikul juhtudel ei kata see kogu ekraani. Veebilehitseja veebilehele eraldatud ala suurust mõjutavad menüüd, otsingukastid, järjehoidjad ja erinevad tööriistaribad. Seega, ainult kindlate ekraanisuurustega arvestades ei ole võimalik saavutada kasutaja jaoks optimaalset vaatamiskogemust. Peatükis 2.1 nimetatud paindliku veebidisaini koostisosadest on küljenduse valikul väga suur osatähtsus, kuna sellest sõltub see, kui paindlikuks arendatav veebileht kujuneb.

### 2.3.1 Fikseeritud küljendus

Fikseeritud küljenduse korral on sektsioonide laiused määratud ära pikslitega. Kuna piksel on absoluutne suurusühik, siis kuvatakse erinevate ekraanisuuruste korral sektsioone fikseeritud suurustega, sellest tuleneb ka asjakohane nimetus.

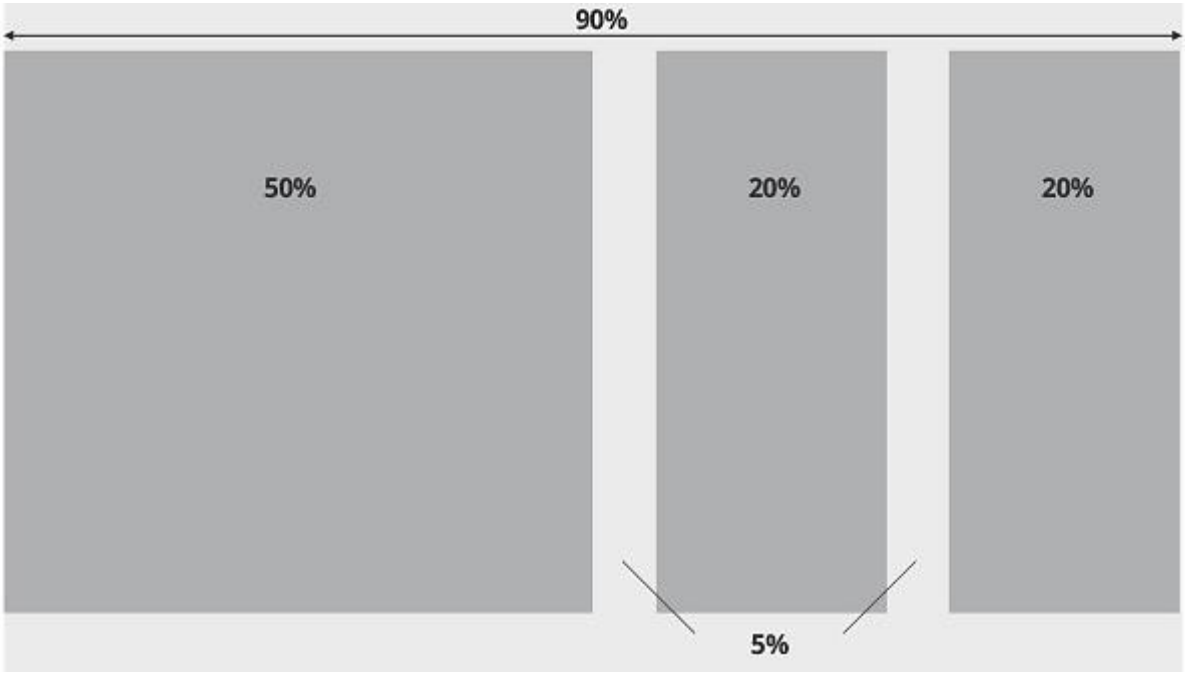



**Joonis 1. Fikseeritud küljendus**

Küljenduse puhul, mis on näha joonisel 1 „Fikseeritud küljendus“, tekib kasutajal horisontaalne kerimisriba, kui veebilehitseja maksimaalne ekraani laius on alla 960 px. Kui veebilehitseja ekraani laius on näiteks 1920 px, siis näeb kasutaja mõlemal pool 480 px laiust tühja ala – eeldusel, et konteiner on paigutatud veebilehe keskele. Fikseeritud küljendust kasutades saab luua paindlikku veebidisaini, pannes erinevate ekraanisuuruste vahemike korral veebilehte erinevalt käituma, sellel peatutakse lähemalt peatükis 2.6.

### 2.3.2 Muutlik küljendus

Muutliku küljenduse korral kasutatakse sektsioonide suuruse määramiseks suhtelisi ühikuid – protsente. Sellega tagatakse sektsiooni suuruse muutumine olenevalt kasutaja ekraanisuurusest. Nii ei teki horisontaalset kerimisriba, sõltumata sellest, millise ekraanisuurusega veebilehte vaadatakse.




**Joonis 2. Muutlik küljendus**

Joonise 2 „Muutlik küljendus“ korral on veebileht paigutatud sektsiooni, mille laius on 90% ekraani laiusest. Oluline on teada, et sektsiooni laius arvutatakse ülemsektsiooni laiuse suhtes ehk sisemise sektsiooni laius 50% tuleneb arvutusest (ekraani laius × 90% × 50%). Muutliku küljenduse korral on mõistlik kasutada CSS-i omadusi *min-width* ja *max-width*, andes väärtuse pikslites. *Min-width* määrab minimaalse laiuse, millest sektsioon mitte mingil juhul väiksemaks ei lähe. *Min-width* kasutamise korral tuleb arvestada sellega, et kui kasutaja ekraanisuurus on väiksem kui antud suurus, siis tekib kasutajal horisontaalne kerimisriba. *Max-width* määrab ära sektsiooni maksimaalse laiuse, millest sektsioon mitte mingil juhul suuremaks ei lähe. *Max-width* kasutamise korral kasutajal horisontaalset kerimisriba ei teki.

### 2.3.3 Elastne küljendus

Elastse küljenduse korral kasutatakse sektsioonide laiuse määramiseks suhtelisi ühikuid, *em*’e. *Em* sõltub kirjasuurusest. Näiteks kui HTML-i *body* ehk keha kirjasuurus on 16 px, siis 1 *em* vastab 16 px-le ja 2 *em*’i vastab 32 px-le [2, p. 26]. Selliselt sektsioonide suurusi määrates on võimalik tagada reas olevate sõnade arvu, et saavutada ideaalne loetavus. Muutes kirjasuurust, muutuvad ka sektsioonide suurused ja proportsioonid veebilehel jäävad paika. Elastse küljenduse korral võib tekkida horisontaalne kerimisriba. Kuna kirjasuurust on võimalik muuta, on veebilehe arendamine keerukam kui muutliku küljenduse korral.

### 2.3.4 Hübriidküljendus

Hübriidküljendus tähendab, et kasutatakse kahte või enamat eespool nimetatud küljendust. See võib osutuda vajalikuks sel juhul, kui mingi sektsioon veebilehest peab olema kindla laiusega ehk fikseeritud küljendusega. See võib olla vajalik näiteks reklaamide tootmiseks, et disainer oskaks kujundatava ala suurusega arvestada.

## 2.4 Meta-märgend *viewport*

Seoses mobiilsete seadmete leviku ja variatsioonide hüppelise kasvuga on muutunud veebilehtede kuvamise loogika. Kui laua- ja sülearvuti vaateala ehk *viewport* on veebilehte mahutava ala suurus, siis mobiilsete seadmete puhul ei ole see üksnes nii. Esimese iPhone'i ekraani laius oli 320 px, aga kui veebilehitseja veebilehte kuvas, siis arvestati vaateala laiuseks 980 px. See tingis olukorra, kus veebilehte vähendati nii palju, et see mahuks ekraanile, mille laius on 320 px. Tulemuseks on „kokku pressitud" veebileht, mida suumimata pole võimalik lugeda. Probleemi lahendamiseks võeti kasutusele meta-märgend *viewport*, mis defineerib, kui laialt tuleb veebilehte kuvada. Veebilehitseja „teab" seda märgendit lugedes, et arendaja on mõelnud ka mobiilsetele seadmetele. Paindliku veebidisaini eeltingimusena tuleb kasutada seda märgendit, et lehte ei „pressitaks kokku". Märgendi parameetris *content* saab defineerida konstantse suuruse, millega veebilehte kuvama peab, näiteks 640 px. Kui kasutaja vaatab sellist veebilehte mobiilse seadmega, mille ekraani laius on 320 px, siis sellisel juhul vähendatakse kõiki objekte 50% võrra. Kui kasutaja vaatab ekraaniga, mille laius on 1240 px, siis suurendatakse lehte 200%. Selline lähenemine ei ole soovitatav. Mõistlikum lahendus on öelda veebilehitsejale, et vaateala suuruseks on seadme laius. Selleks tuleb defineerida meta-märgend järgmiselt:

```
<meta name="viewport" content="width=device-width" />
```

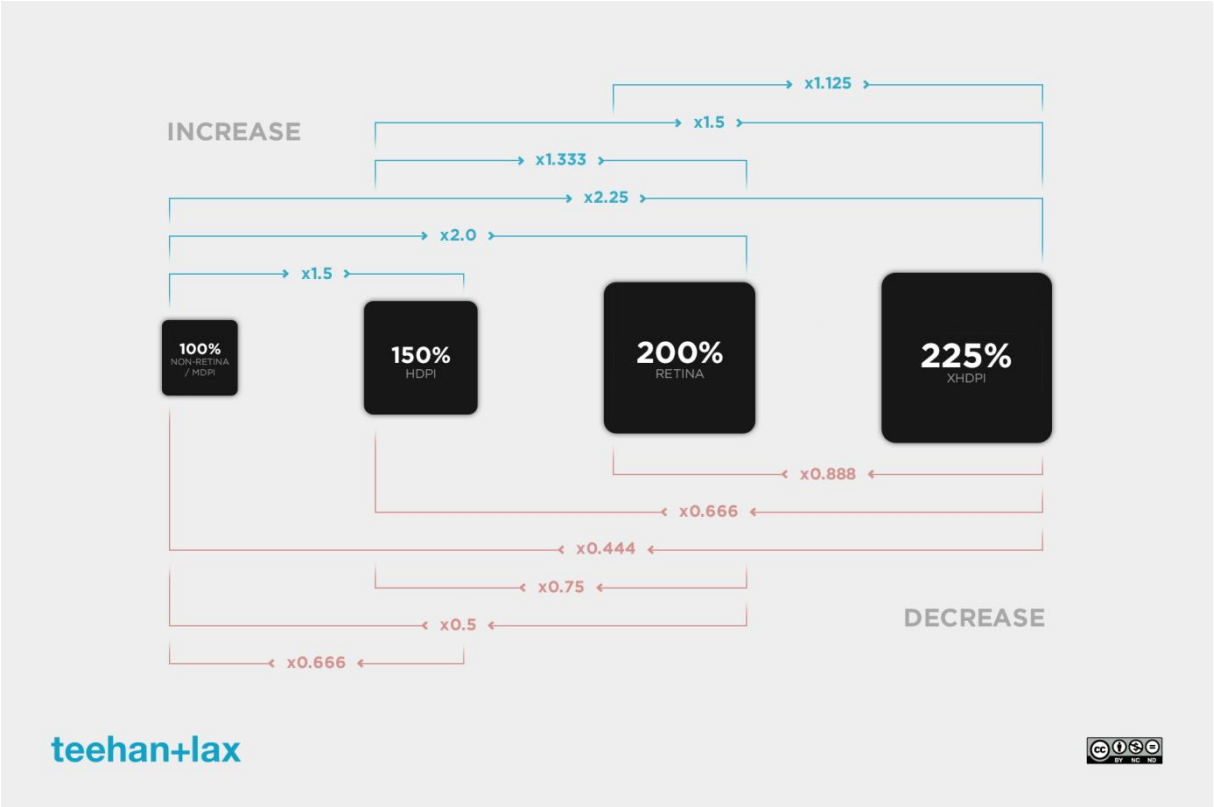
Selliselt defineerides on 320 px laiuse seadme korral vaateala suuruseks 320 px. Lisaks on võimalik veebilehitsejale öelda, kui suur on esialgne suurendus või vähendus. Selleks tuleb parameetrile *content* lisada *initial-scale*-väärtus, mis võib olla vahemikus 0,1 (10%) kuni 10,0 (1000%). Mõistlik on kasutada väärtust 1, mis tagab veebilehe näitamise suurenduseta ja vähenduseta. Soovitatav on kasutada mõlemat väärtust koos:

```
<meta name="viewport" content="initial-scale=1, width=device-width" />
```

Nii tagatakse veebilehe näitamine mobiilsetel seadmetel samamoodi, nagu oleks lauaarvutil veebilehitseja akna suurust muudetud väiksemaks. Tulemuseks on suure lehe mahutamine alale, mis võrdub seadme laiusega ja ei rakendata suurenduse muutmist. Märgendit selliselt defineerides muutsime mobiilsetel seadmetel olukorra esialgu halvemaks, kuid lahenduseks on CSS-i *media query*'de kasutamine, sellest kirjutatakse peatükis 2.6. Meta-märgendile *viewport* saab lisada lisaväärtusi, kuid need on vajalikud ainult erandkorras. Lisaväärtustest saab väga hea ülevaate viitest [2, p. 57]. Konkreetne peatükk põhineb peamiselt sellel viitel.

## 2.5 Ekraani pikslitihedus

Peatükis 2.4 käsitletud vaateala laiuse seadistamise korral tuleb arvestada ka ekraani pikslitihedusega. Erinevate seadmete ekraanisuurused varieeruvad, kuid nende resolutsioonid võivad olla samad. Sellest tulenevalt on pikslite arv tolli kohta (*PPI – pixels per inch*) erinev. Esimesel iPhone'il oli resolutsioon 320 x 800 pikslit 3,5" ekraanil ja pikslitihedus sellest tulenevalt 164 ppi. Google Nexus One'il oli resolutsioon 480 x 800 pikslit 3,7" ekraanil ja pikslitihedus 252 ppi [6, p. 110]. Pikslitihedus mõjutab ekraanilt paistva graafika suurust. Mida suurem on pikslitihedus, seda väiksem on iga piksel. Meta-märgend *viewport* hoolitseb selle eest, et vaateala laius määratakse vastavalt pikslitihedusele. Arendaja peab teadma, et suurema pikslitihedusega seadmete korral muudetakse vaateala väiksemaks: nii tagatakse kasutajale optimaalne loetavus. Oluline on teada, et graafika näitamise puhul suurendatakse pikslite arvu, seda demonstreerib joonis 3 „Suurendus sõltuvalt pikslitihedusest", see joonis on võetud viitest [7].




**Joonis 3. Suurendus sõltuvalt pikslitihedusest**

Veebilehele graafika lisamisel peab arvestama, millise suurusega pilte lisatakse. Kui aluseks võetakse 100%, siis *Retina* ekraani korral suurendatakse kõiki pilte 200%. Tulemuseks on „udused“ pildid. „Teravate“ piltide jaoks tuleb veebilehele lisada erinevate pikslitiheduste jaoks erineva suurusega pildid. Seda, millist pilti kasutajale kuvatakse, saab optimeerida, kasutades *CSS media query*'t, JavaScripti või serveri-poolse koodi abil.

## *2.6 CSS media query*

CSS3-s tutvustati esmakordselt *media query* avaldist. *Media query* abil on võimalik kirjutada CSS-i stiilireegleid, mis kehtivad ainult teatud tüüpi seadme omaduste korral. Peamiselt kasutatakse *max-width* ja *min-width* omadusi. *Media query* reeglid lisatakse CSS-i faili lõppu. Järgnevalt tuuakse näide *max-width* kasutamisest:

```
body {
    background-color: grey;
}
@media screen and (max-width: 960px) {
    body {
        background-color: red;
    }
}
```

Kasutaja, kelle vaateala laius on suurem kui 960 pikslit, näeb veebilehe tausta hallina. Kasutaja, kelle maksimaalne vaateala laius on kuni 960 pikslit, näeb veebilehe tausta punase värviga. *@media*'le järgnev *screen* (meediatüüp) tähendab, et reegel kehtib ekraani puhul. Teine populaarne meediatüüp on *print*, mis tähendab, et reegel kehtib veebilehte printides. *Media query* avaldises saab kasutada kõiki CSS spetsifikatsioonis kirjeldatud võimalusi. Nii kirjeldatakse, milliseid vormingumuudatusi vaateala suurust muutes esile kutsutakse.

## 2.7 Lähenemised paindliku veebidisaini loomisel

Veebilehe kasutajate seadmete omadustest ja seadetest sõltub, kuidas arendada paindlikku veebidisaini. Kui eelistused pole teada, tuleb uurida, millised on kasutajate sihtgrupi üldised harjumused. Peatükis 2.6 kirjeldatud *media query* kasutamisel on kaks erinevat lähenemisviisi: mobiilile esmakohal ja lauaarvutile esmakohal.

### 2.7.1 Mobiilile esmakohal

Kui veebilehe peamised külastajad kasutavad mobiilseid seadmeid, siis on mõistlik veebileht optimeerida just nende kasutajate jaoks. Sellise lähenemise korral kirjutatakse CSS mobiilseid seadmeid arvestades ja hiljem muudetakse *media query*'de abil lehe küljendust nii, et seda oleks meeldiv vaadata ka lauaarvutist. *Media query*'de reeglite seisukohalt tähendab see, et vaateala suurused lähevad reeglites järjest suuremaks. Kõigepealt kirjeldatakse näiteks ära reeglid, kui kasutaja maksimaalne vaateala on 480 pikslit, seejärel näiteks 960 pikslit jne. Kuna mobiilsete seadmete puhul on vaateala väiksem kui lauaarvutitel, siis peab olema veebilehe sisu kuvamine väga hoolikalt läbi mõeldud. Kui mobiilne kasutaja leiab veebilehelt kõik vajaliku lihtsalt üles, siis peaks leidma ka lauaarvuti kasutaja.

### 2.7.2 Lauaarvutile esmakohal

Lauaarvuti esmakohal lähenemise korral tehakse veebileht valmis suurele vaatealale ja siis hakatakse *media query*'de abil muutma veebilehte kasutajasõbralikuks ka mobiilsetel seadmetel. See tähendab, et *media query*'de reeglid lähevad järjest väiksemaks. Alustatakse näiteks *max-width* 1200 pikslit, siis lisatakse *max-width* 960 pikslit jne. Ka selle lähenemise korral on sisul väga oluline tähtsus. Kõik vajalik peab olema kättesaadav ka mobiilsetel seadmetel, muidu ei ole mõtet paindlikku veebidisaini rakendada.

## 2.8 Olemasolevad raamistikud

Kõike eelpool kirjeldatut on vaja teada, kui soovitakse rakendada paindlikku veebidisaini ise, nii-öelda nullist. Oluline on siinkohal ära märkida, et on olemas valmis raamistikke, mis võimaldavad paindlikku veebidisaini rakendada, kasutades juba valmis loodud komponente. Raamistikus on kirjeldatud hulk erinevaid küljendamisvõimalusi, mis arvestavad juba erinevate vaatealade suurustega. Kaks populaarsemat raamistikku, mida käsitletakse peatükkides 4 ja 5, on *Bootstrap* ja *Foundation*.

## 2.9 Veebilehe weprint.ee arendamise lähenemise valik

Töö autor otsustas weprint.ee veebilehte arendada raamistikke kasutamata, sest leiab, et just nii on võimalik paindliku veebidisaini arendamine endale kõige paremini selgeks teha. Võimalik, et järgmise veebilehe arendamiseks valitakse juba olemasolev raamistik. *Bootstrap*'i ja *Foundation*'i peatükkides tuuakse välja raamistike eelised ja puudused; võrreldakse raamistike kasutamist nullist tegemisega ning analüüsitakse raamistike võimalusi. Peatükkides 4 ja 5 teostatakse analüüs ja võrdlused läbiproovitud koodinäidete põhjal ehk lisaks nullist arendamisele on töö raames kõik küljendused loodud ka kasutades raamistikke.

Kuna arendatava veebilehe puhul on eesmärk vältida horisontaalse kerimisriba tekkimist seadmetel, mille vaateala laius on suurem kui 320 pikslit, otsustati kasutada muutlikku küljendust. Minimaalseks laiuseks valiti 320 pikslit, kuna enamikul veebilehe külastajatest on vaateala laius suurem. See info põhineb olemasoleva veebilehe Google Analytics statistikal (vt Tabel 1). Tabelis 1 on toodud kõik resolutsioonid, mille protsent on suurem või võrdne ühega.

**Tabel 1. Weprint.ee külastajate resolutsioon ja resolutsiooni kasutamise protsent**

| **Resolutsioon** | **Kasutamise protsent** |
|---|---|
| 1366 x 768 | 20,33% |
| 1920 x 1080 | 16,39% |
| 1280 x 800 | 13,66% |
| 1280 x 1024 | 9,26% |
| 1680 x 1050 | 8,80% |

| | |
|---|---|
| 1440 x 900 | 6,22% |
| 768 x 1024 | 3,79% |
| 1600 x 900 | 3,34% |
| 1024 x 768 | 2,73% |
| 2560 x 1440 | 2,28% |
| 1360 x 768 | 1,21% |
| 1536 x 864 | 1,21% |
| 1920 x 1200 | 1,21% |

Veebilehe loomisel kasutatakse lauaarvuti esikohal lähenemist, tuginedes samuti olemasolevale statistikale (vt Tabel 2). Veebilehe tekste koostades on lähtutud sellest, et informatsioon peab olema optimaalne ka mobiilsete seadmete jaoks. Pildid lisatakse veebilehele arvestades vaateala maksimaalset laiust. Siinkohal teadvustab töö autor, et väga suurte vaatealade korral (*Retina* ja sealt ülespoole) tekib olukord, kus pildid ei ole teravad. Teravuse probleem on lahendatav piltide dubleerimisega ja vastavate piltide näitamisega, kasutades *media query*'t või JavaScripti.

**Tabel 2. Weprint.ee kasutamise seadmete statistika**

| **Seadme tüüp** | **Kasutamise protsent** |
|---|---|
| Lauaarvutid | 90,90% |
| Tahvelarvutid | 5,01% |
| Mobiiltelefonid | 4,10% |

# 3. Weprint.ee arendamine, kasutades HTML5-e ja CSS3-e võimalusi

We Print on trükikoda, mis on spetsialiseerunud tekstiilile trükkimisele. Ettevõte tegeleb nii töö-, reklaam- kui ka spordiriietele trükkimisega. Lisaks mainitud põhisuunale pakutakse ka trükiteenuseid paberile (visiitkaardid, lendlehed, plakatid ja voldikud). Trükikoda soovib pakkuda täisteenust, mis algab kliendi vajaduste kaardistamisega ning lõpeb peale valmistoodete tarnimist kogu protsessile tagasiside saamisega.

Praegune weprint.ee veebileht on suunatud kasutajale, kes on teadlik erinevatest trükitehnikatest ja tunneb trükitehnikate kasutamise otstarvet. Terminoloogiat mitte tundev kasutaja ei saa aru, millega We Print trükikoda tegeleb. Kasutaja teab, millele ta soovib trükkida (näiteks T-särgile), kuid praegust veebilehte vaadates ei saa ta ülevaadet, kas We Print trükikoda on suuteline talle abiks olema. See on ka peamine põhjus, miks soovitakse teha uut veebilehte. Lisaks on praegune veebileht ajale jalgu jäänud: see on optimeeritud vaatealale, mille laius on 800 pikslit. Enamiku kasutajate vaateala on sellest laiusest suurem (vt Tabel 1). Seega ei ole veebilehe sisu optimaalselt loetav mobiilsetes seadmetes ega ka lauaarvutites.

Uue veebilehe planeerimisel võeti eesmärgiks paindliku veebidisaini loomine selliselt, et sõltumata vaateala suurusest ei tekiks horisontaalset kerimisriba. Paika sai pandud, et ei toetata väiksemat vaateala kui 320 pikslit. 320 pikslit sai valitud seetõttu, kuna praegust veebilehte ei ole külastatud sellest väiksema resolutsiooniga ja ei ole alust arvata, et seda tulevikus tegema hakatakse. Andmed pärinevad Google Analytics'i statistikast. Pigem on kasutamise tendents suuremate resolutsioonide kasuks. Eesmärgiks seati võimalikult palju kasutada CCS3-e võimalusi lehe kujundamiseks. Veebilehe avaleht otsustati teha ühe pika lehena, kus on neli erinevat sektsiooni: täisteenus, portfoolio, kontakt ja jalus. Eesmärk on anda kasutajale teada, et peamiselt tegeletakse riietele trükiga. Selle väljendamiseks kasutatakse täisteenuse taustana erinevaid riietele trükkimisega seotud pilte ja lisaks on täisteenuse esikohal tootepildina kasutatud ikooni, mis kujutab T-särki. Portfoolio eesmärk on anda kasutajale kiirelt teave, et lisaks tekstiilile on We Print trükikojas võimalik printida veel kruusidele, plätudele ja erinevatele paberitele. Lehe jaluses on erinevate grupeerimistega viidatud sisuliselt samadele mõistetele, et kasutaja leiaks just temale sobivat terminoloogiat

kasutades vajaliku sisulehe. Kasutaja ei pruugi teada, millist tehnoloogiat on vaja näiteks riietele trükkimiseks, kuid ta teab, et vaja on trükkida näiteks riidest kotile või särgile. Teise lähenemisena on välja toodud ka tehnoloogia järgi grupeerimine. See on mõeldud kasutajale, kes teab kindlat trükitehnoloogiat, kuid tahab saada ülevaadet, kas We Print pakub seda teenust.

Weprint.ee uue veebilehe graafiline disain ei ole lõputöö autori teostatud. Autori ülesanne oli paindliku veebilehe loomine, kasutades Photoshop'i disainifaili. Kõik töös toodud HTML-i ja CSS-i näited on autori töö tulem.

## 3.1 Resolutsioonist sõltuvate olekute kirjeldamine

Veebilehe planeerimisel otsustas töö autor kasutada viite erinevat niinimetatud hüppeala, mis järjest väiksemate vaatealade puhul kaotavad ära sektsioone või muudavad küljendust. Hüppealadeks sai valitud suurem võrdne kui 1150 px, 980 px – 1149 px, 768 px – 979 px, 481 px – 767 px ja kuni 480 px. Need viis hüppeala sai valitud sellepärast, et nii kaetakse ära väga lai vaatealade ulatus. Suurem vaateala kui 1150 px – sellega on kaetud kõik moodsamad laua- ja sülearvutid. Vaateala 980 px – 1149 px katab ära vanemad süle- ja lauaarvutid. Vaateala 768 px – 979 px katab ära tahvelarvutid külili olekus. Vahemik 481 px – 767 px katab tahvelarvutid püstises asendis ja moodsamad mobiiltelefonid. Viimasesse vahemikku kuni 480 px kuuluvad vanemad mobiiltelefonid. Küljenduste kirjeldamisel alustatakse vaatealast, mille laius on suurem või võrdne 1150 px. Erinevate küljenduste pildid on näidatud lisades 1–6 A4 formaadis.

### 3.1.1 Avalehe olekud

Lisades 1-3 on toodud avalehe küljendused kolme hüppeala korral. Paralleelselt lugemisega oleks parema arusaadavuse huvides mõistlik jälgida ka viidatud pilte. Küljenduse pildid on toodud lisades, kuna need ei oleks teksti vahele paigutatult jälgitavad.

Täisteenuse sektsiooni puhul tuleb tagada, et erinevate vaatealade korral oleksid ringid üksteise kõrval ja muutuksid koos vaateala muutumisega järjest väiksemaks. Eesmärk on sõltumata seadmest näidata seda sektsiooni kasutajale terviklikult, et kasutaja saaks kiire ülevaate.

Portfoolio sektsioon peab olema joondatud lehe keskele. Piltide arv reas peab sõltuma vaateala suurusest. Kõige väiksema vaateala vahemiku puhul (kuni 480 px) peab portfoolio piltide suurus muutuma väiksemaks, et kasutaja saaks parema ülevaate.

Kontakti sektsiooni küljendus peab olema selline, et ettevõtte kontaktandmed oleksid vasakul ja pankadega seotud info oleks kontakti kõrval paremal. Kaart ettevõtte asukohaga on paremas ääres. Kui kasutaja vaateala on väiksem ja kõik kolm ei mahu enam kõrvuti, siis esimesena peab muutuma nähtamatuks kaart. Nii jääb kontaktinfo lihtsasti kättesaadavaks. Teine variant oleks olnud kaardi allesjätmine ning pankadega seotud info liigutamine kontaktandmete alla, kuid sellisel juhul muutuks kontakti sektsiooni kõrgus. See ei ole hea, kuna sellisel juhul peab kasutaja veebilehte rohkem vertikaalselt kerima, et jõuda jaluseni: veebilehest kiire ülevaate saamine halveneb. Kui kontaktandmed ja pankadega seotud info ei mahu enam kõrvuti, siis peab pankadega seotud info liikuma kontaktandmete alla. Sellisel juhul kontakti sektsiooni kõrgus küll suureneb, kuid pankadega seotud info ärakaotamine ei olnud mõeldav variant, sest olulise info kättesaadavust ei tahaks kindlasti väikese vaateala korral vähendada.

Jaluses kuvatakse suurima vaateala korral nelja sektsiooni kõrvuti: „trükk“, „tehnikad“, „täisteenus“ ja „leia kiiresti“. Kui kasutaja vaateala on selline, kus neli sektsiooni ei mahu enam kõrvuti, siis peab küljendus muutuma nii, et kõrvuti jääks kaks sektsiooni. Vältida tuleb olukorda, kus kolm sekstiooni oleks ühel real ja üks sektsioon oleks eraldi teiste sektsioonide all. Jaluse sektsioonil on seega kaks küljendust: neli sektsiooni ja kaks sektsiooni reas. Oluline on, et kahe sektsiooni korral reas liiguksid alumisele reale viimased kaks sektsiooni ehk „täisteenus“ ja „leia kiiresti“.

### 3.1.2 Alalehtede olekud

Alalehtedel esineb kolm erinevat küljendust, mis siis vastavalt vaateala suurusele muudavad sektsioonide asetust. Erinevaid küljendusi kasutatakse vastavalt alalehe sisule. Erinevate hüppealade korral kasutatakse erinevat teksti suurust. Teksti suurus väheneb hüppealade muutumisel suuremast väiksemaks.

Esimene küljendus (vt lisa 4) on selline, kus on reas kaks pildiga sektsiooni. Küljenduse ülaosas võib olla tekst, mis on joondatud keskele. Pildiga sektsiooni ülaosas on pilt ning pealkiri ja tekst on joondatud sektsiooni keskele. Kui kaks pildiga sektsiooni ei mahu vaatealas kõrvuti ära, siis liigub sektsioon talle eelneva sektsiooni alla. Pildiga sektsiooni

suurus on sõltuvuses vaateala laiusest, seega muutuvad ka pildi mõõtmed. Sellist küljendust kasutatakse juhul, kui pildiga sektsioonid on võrdse tähtsusega ja ei olene sellest, millises järjekorras kasutaja neid sektsioone lugema peaks.

Teine küljendus (vt lisa 5) on selline, kus reas on üks sektsioon. Küljenduse ülaosas asuv tekst on joondatud vasakule. Igal pildiga sektsioonil on pealkiri, mis on joondatud vasakule ning ka sektsioonis olev tekst on joondatud vasakule. Pildiga sektsiooni pealkiri ja tekst on samal kaugusel pildist. Vaateala vähenemisel muutub pildile antava ala suurus, seega vähenevad pildi mõõtmed. Tekst kohandub vastavalt sellele antava ala suurusega: sõnad, mis ritta ära ei mahu, liiguvad järgmisele reale. Sellist küljendust kasutatakse juhul, kui sektsioonide lugemise järjekord on oluline. Näiteks täisteenuse puhul algab loogiline protsess toote valikust ja lõpeb transpordiga.

Kolmas küljendus (vt lisa 6) on selline, kus alalehel olev pilt ja tekst on joondatud lehe keskele. Tekstile lubatud ala suurus on sõltuvuses vaateala suurusest. Tekst ei tohi olla alalehel äärest ääreni. Pildi suurus ei ole muutuv, kuna pilt mahub alati täissuuruses vaatealale ära.

## 3.2 Veebilehe raamistiku loomine

Kasutades muutlikku küljendust, tuleb paindliku veebidisaini arendamisel sektsioonide mõõtmed paika panna sõltuvalt vaateala suurusest ehk kasutades protsente. Fikseeritud suuruste kasutamine tooks kaasa olukorra, kus looksime nii-öelda traditsioonilise ühele vaatealale optimeeritud veebilehe. Weprint.ee lehel on kasutatud mitme sektsiooni puhul taustavärvi, mis ulatub üle kogu vaateala. Kuidas saavutada seda, et veebilehe taust oleks äärest ääreni, kuid sisu sektsiooni sees oleks sõltuvuses vaateala suurusest? Selleks tuleb luua sektsioon, mille laius oleks 100%. HTML5-e spetsifikatsioonis on kirjeldatud selleks sobiv märgend nimega *section*. Sektsiooni sees oleva elemendi laiuse määramiseks tuleb kasutada protsenti. Kirjeldatud käitumise saavutamiseks kasutasin HTML-i:

```
<section class="section">
    <div class="inner">...</div>
</section>
```

CSS, mis antud HTML-märgistusele kehtib:

```
.section .inner {
    margin: 0 auto;
    max-width: 1150px;
    min-width: 320px;
    width: 90%;
}
```

Selline CSS-i reeglistik ütleb, et elemendi puhul, millel on klass nimega *section* ja mille sees on omakorda element klassiga *inner*, tuleb sisemine element joondada lehe keskele. Sisemise elemendi maksimaalne laius on 1150 pikslit ja minimaalne laius 320 pikslit. Sisemine element võtab omale laiuseks 90 protsenti vaateala suurusest, kuid ei ületa mitte mingil juhul minimaalset ega maksimaalset laiusele seatud piiri. Maksimaalse laiuse seadmine on selle veebilehe küljenduse puhul oluline seepärast, et kasutaja ei peaks suure vaateala korral tekste lugema ühest ekraani otsast teise. Paindliku veebidisaini metoodika ei sätesta, et peaks kasutama maksimaalset laiuse piirangut. Maksimaalse laiuse kasutamine või mittekasutamine sõltub veebilehe küljendusest. 90-protsendise laiuse määramisega luuakse veebilehe mõlemal poole 5-protsendiline ruum, seda juhul, kui vaateala laius on väiksem või võrdne 1150 piksliga. 5-protsendiline ruum on oluline, et veebilehe sisu ei algaks vaateala äärest. Ruumi määramine sõltub küljendusest ja seda saab määrata ka sisuelementide kirjeldamisel. Toodud HTML-i märgistust kasutatakse veebilehe weprint.ee kõikide sektsioonide alusena.

Eelpool kirjeldatud HTML-i märgistusele sisu loomisel peab arvestama, et elemendi klassiga *inner* maksimaalne laius on 1150 pikslit. Kui sisuosasse luua uus element, mille laius on 50%, siis see tähendab, et see element on maksimaalselt 1150 × 50% = 575 pikslit lai. Loodud elemendi minimaalne laius on 320 × 90% × 50% = 144 pikslit. Elemendi suuruse leidmiseks kehtib valem:

$$\mathbf{target \div context = result}$$

Toodud valemis on *target* planeeritava elemendi laius ja *context* ala suurus, mille sees asjakohane element paikneb [3, p. 31]. Suuruse määramisel protsendiga tuleb arvestada rekursiivselt ehk defineerides elemendi laiuse protsendiga, tähendab see laiuse sõltumist alati tema ülemelemendi laiusest. See on suurim erinevus võrreldes fikseeritud suuruse kasutamisega – fikseeritud suurus ei sõltu kunagi ülemelemendi laiusest. Selle loogikaga harjumine ja sellest arusaamine on paindliku veebidisaini arendamisel üks maailmavaatelisi erinevusi võrreldes varasemate arusaamadega.

Järgnevalt vaatleme, kuidas luua jaotusel põhinevat küljendust. Weprint.ee avalehel on selle lähenemise väga heaks näiteks jaluse käitumine. Kui neli sektsiooni ei mahu enam kõrvuti ära, siis näidatakse kasutajale kahte kõrvuti asetsevat sektsiooni. *Table*-märgendi kasutamine ei ole juba ammu veebidisaini küljenduse tegemiseks olnud soovitatav, kuna see muudab küljenduse muutmise tulevikus väga keeruliseks ja raskesti hallatavaks. Paindliku veebidisaini arendamisel on see väga selgesti arusaadav. Kujutame ette, et jaluse kirjeldamiseks on kasutatud *table*-märgendit. Tabeli kirjeldamiseks on võimalik luua ridu ja reas olevaid veerge. Kui luua tabel, millel on üks rida ja neli veergu, nii nagu jaluse sektsioonis oleks weprint.ee veebilehel vaja, siis ei oleks võimalik CSS-i reegleid kasutades kahte viimast veergu viia järgmisele reale. Soovitud käitumise loomiseks on vajalik defineerida elemendid, mis oleksid üksteisest sõltumatud. Elemendile määratakse laius protsentides. Sõltuvalt vaateala suurusest, muudetakse elemendi laiust: nii tagatakse, et ainult soovitud arv elemente mahuks vaatealas kõrvuti. Näiteks suure vaateala puhul määratakse elemendi laiuseks 20% ja väiksema vaateala korral 40%. Kui laiused on nii defineeritud, siis väiksema vaateala puhul ei mahu enam ritta rohkem kui kaks elementi ja ülejäänud kaks elementi liiguvad järgmisele reale. Elemendi laiuse määramist sõltuvalt vaatealast kirjeldatakse peatükis 3.3.

Piltide kasutamisel tuleb piltidele määrata *max-width* omadus, mille väärtus on 100%. Sellega tagatakse, et pildi laius ei ületa tema ülemelemendi laiust. Kui seda omadust ei määrata, siis kuvatakse pilti tema originaalmõõtmetega. Kui ülemelemendi laius on defineeritud vaatealast sõltuvalt ehk protsentidega ja pildil on kasutatud *max-width* omadust, siis sellisel juhul on ka pilt vaatealast sõltuv. Selline lähenemine on vajalik, et veebilehele lisatav pilt ei peaks olema täpse suurusega. Vajalik on see sellisel juhul, kui veebilehe haldaja saab ise pilte lisada ja ta ei oska arvestada pildi mõõtmetega. Sellega tagatakse, et sõltumata pildi suurusest ei lähe paindlik veebileht nii-öelda puruks.

Weprint.ee avalehel on täisteenuste sektsioon, kus valge ringi sees on kuvatud ikoon ja selle all tekst. Vajalik on, et ring, ikoon ja tekst oleksid sõltuvuses vaateala suurusest. Teksti suuruse puhul protsendi kasutamine tähendab sõltuvust dokumendi kehale määratud teksti suurusest, seega teksti suurus ei sõltu ülemelemendi suurusest. Pildi ja teksti vähendamine korraga ei toimi isegi siis, kui ülemelement on defineeritud protsendilise laiusega. Pilt on küll vaatealast sõltuv, kuid tekst ei ole. Variandiks on tekst panna pildi peale, seega väheneks üks pilt ja sellega oleks soovitud käitumine saavutatud. Sellise kasutamise juures tuleks pildile kindlasti lisada parameetrid *alt* ja *title*, et robotid, kes veebilehte vaatavad, saaksid aru, mida pilt kujutab. Sellise lähenemise peamiseks eeliseks on lihtsus. Teine variant on kasutada CSS-

i omadust *zoom*. *Zoom*'i omaduse väärtus on numbriline, kus number 1 tähendab, et elementi kuvatakse originaalsuuruses. *Zoom*'i määramisel 0.5 vähendatakse elemendi suurust poole võrra. Selline lahendus eeldab *zoom*'i väärtuse muutmist JavaScripti abil. JavaScriptis peab olema defineeritud ülemelemendi suuruse muutmist jälgiv kood, mis omakorda muudaks *zoom*'i väärtust. Selle lahenduse miinuseks on, et see eeldab veebilehe kasutajalt JavaScripti lubamist. Lisaks veel pidevat ülemelemendi suuruse jälgimist, mis aeglustab veebilehe toimimist, kui jälgitavaid elemente on palju. Kolmas variant on kasutada HTML-i märgendit *svg*. *Svg* võimaldab kasutada nii vektor- kui ka rastergraafikat. Selle lahenduse peamiseks miinuseks on defineerimise keerukus ja lisaks ei suuda veebilehitsejad sellist pilti vahemälust tagastada.

Eelpool kirjeldatud tehnikaid kasutades on võimalik luua veebilehe raamistik, mis toetaks vaatealast sõltumist. Nende põhimõtete järgimine on oluline sõltumata sellest, kas kasutatakse olemasolevat raamistikku või mitte.

## 3.3 Erinevate olekute loomine, kasutades *media query*'t

Peatükis 2.6 kirjeldatud *media query* kasutamiseks on vaja läbi mõelda loodavad hüppeala suurused, mille korral CSS-i reegleid kas lisatakse või kirjutatakse üle. Hüppealade defineerimisel ei ole ette antud mingeid reegleid, millised need olema peaksid. We Print küljenduse puhul otsustas töö autor kasutada viite hüppeala vahemikku. Peatükis 3.1 toodud hüppealad valiti peale erinevate seadmete resolutsioonide analüüsi. Lisaks uuris töö autor internetist populaarseimaid hüppealasid, selgitades välja, milliseid hüppealasid kasutatakse *Bootstrap*'is. Nende andmete põhjal pandi kokku just weprint.ee jaoks sobilikud hüppealad. Hüppealade defineerimise võlu seisneb selles, et määratakse ära, mis erinevate hüppealade puhul muutub. Kui tulevikus muutuvad suuremad vaatealad populaarsemaks, siis näevad kasutajad veebilehte ikka optimeeritud kujul, sest veebileht kuvatakse, kasutades suurema hüppeala reegleid.

Avalehel täisteenuse sektsioonis on kasutatud taustana kolme erinevat pilti, mis vahelduvad juhuslikkuse alusel. See tähendab, et serveri poolel lisatakse HTML-i erinev klass. Selleks on defineeritud HTML-i klassid: *home-1*, *home-2* ja *home-3*. Nendest kolmest taustast otsustas töö autor teha omakorda kolm erineva suurusega pilti. Erineva suurusega pildid said tehtud kahel põhjusel: esiteks, et parandada kasutaja arusaamist kuvatavast taustast; teine põhjus on andmemahu vähendamine. Väiksemate vaatealade korral ei oleks suurem osa suurele

vaatealale mõeldud pildist näha, siis ei ole mõtet ka sellist pilti kasutajale kuvada. Tulemuseks on, et väikseima ja suurima vaateala korral on pildi andmemahu erinevus vähemalt 1,6 korda. *CSS media query*'t kasutades:

```
@media (min-width: 1150px){
    #home.home-1 {background:url("home-1.jpg") no-repeat top center}
    #home.home-2 {background:url("home-2.jpg") no-repeat top center}
    #home.home-3 {background:url("home-3.jpg") no-repeat top center}
}
```

Toodud koodinäites laetakse pildid *home-1.jpg*, *home-2.jpg* ja *home-3.jpg* ainult vaateala korral, mis on suurem kui 1150 pikslit. Teiste hüppealade defineerimisel on kasutatud sarnast reeglit, kui kettal viidatakse pildile, mille nimi on teine ning lisaks on muudetud hüppeala suurust.

Jaluses kuvatakse suurima vaateala korral kõrvuti nelja sektsiooni. Iga sektsiooni laius on 20% ja sektsioonid on joondatud lehe keskele. See tähendab, et põhimõtteliselt mahuks ka 5 sektsiooni kõrvuti, kuid kuna HTML'is on defineeritud neli sektsiooni, siis esimese sektsiooni ees ja viimase sektsiooni järel on tühi ruum 10% ulatuses. Vaateala vahemiku 768 px – 979 px korral suurendatakse sektsiooni laiust 24 protsendile. Nii tagatakse, et jätkuvalt mahuks neli sektsiooni kõrvuti. Väiksema vaateala kui 768 px korral ei mahu sektsioonid kõrvuti üksteise otsa nihkumata ära. Kuna eesmärgiks oli kuvada kõrvuti kas nelja või kahte sektsiooni, siis tuleb suurendada väiksema vaateala kui 768 px korral sektsiooni laiust nii, et kõrvuti mahuks ainult kaks sektsiooni. Selleks määrati sektsiooni laiuseks 35%. Laiuseks sai valitud 35%, et sektsioonid ei oleks liialt üksteisest eraldatud ja mõlemale poole jääks vaba ruumi. Kõige väiksema vaateala puhul määrati sektsiooni laiuseks 45%, et sektsioon võtaks endale suurema ala vaatealast. Nii tagatakse jätkuvalt, et kõrvuti mahuks kaks sektsiooni. Sarnaselt on defineeritud ka alalehtedel küljenduse muutmised. Näide sektsiooni laiuse muutmisest vaateala vahemiku 481 px – 767 px korral:

```
@media (min-width: 481px) and (max-width: 767px){
    #footer .column{width:35%;}
}
```

*Media query* abil nullist küljenduse loomise puhul tuleb väga palju testida erinevate vaateala suuruste korral veebilehe käitumist ja probleemi esinemisel teha korrektuure ning uuesti testimist alustada. See on üks peamisi põhjusi, mis teeb paindliku veebidisaini arendamise keerukaks. Veebilehe arendamise puhul on olnud probleemiks erinevate veebilehitsejate erinev tõlgendus CSS-i reeglite osas. Paindliku veebidisaini rakendamisel lisandub veel

erinevate vaatealadega arvestamine. Tüüpiline probleem on see, et sektsioonid lähevad üksteisele liiga lähestikku ja seetõttu muutub ala ebaesteetiliseks. Lisaks võib tekkida olukord, kus valitud vaatealade vahemike abil ei ole võimalik saavutada soovitud tulemust. Näiteks weprint.ee arendamisel tekkis olukord, kus kontakti sektsioonis olev kaart ei olnud hästi nähtav vaatealavahemiku 1150 px – 1600 px korral. Probleemi lahendamiseks tuli defineerida lisahüppeala, mis muutis kaardi taustapildi positsiooni:

```
@media (min-width: 1150px) and (max-width: 1600px){
    #contact #map{background-position:-60px center;}
}
```

## 3.4 Tulemuste ülevaade

Paindliku veebidisaini küljendus peab olema oluliselt rohkem läbimõeldud kui fikseeritud, ühele vaatealale optimeeritud veebilehe loomise puhul. Küljenduse paika saamiseks on veebilehte vaja märgatavalt rohkem testida, seda just vaatealade erinevate suuruste korral. Erinevad vaatealad on nagu eraldi veebilehed, mis peavad olema optimeeritud konkreetsele vaateala suurusele. Nullist paindliku veebidisaini arendamisel tuleb lahendada palju probleeme, mida ei osata veebilehe planeerimisel ette näha. Probleemide lahendamise käigus õpitakse väga palju juurde, kuidas paindlikud veebilehed toimivad ja millele tuleb planeerimisel erilist tähelepanu pöörata. Kui on plaan hakata arendama paindlikku veebilehte, siis töö autor soovitab enda kogemusest lähtudes nullist lähenemist, kuna just nii õpib väga palju. Järgnevates peatükkides 4 ja 5 võrreldakse, kuidas olemasoleva raamistiku kasutamine lihtsustab ja samas ka muudab arendamise keerulisemaks ning milliseid probleeme raamistiku kasutamine kaasa toob.

Nullist lähenemise eelised:

- puhas kood – HTML on semantilise tähendusega;
- veebilehe kood on lihtsasti loetav ja hiljem paremini hallatav;
- puudub ebavajalik kood – ainult konkreetse projekti jaoks kirjutatud HTML ja CSS.

Nullist lähenemise puudused:

- palju testimist erinevate veebilehitsejate ja vaatealade korral;

- tüüpprobleemide korduv lahendamine;
- aeganõudev lähenemine.

# *4. Bootstrap*

*Bootstrap* on üks maailma populaarseimaid avatud lähtekoodiga raamistike. *Bootstrap*'ist on töö kirjutamise hetkel väljas versioon 3.1.1. Alates versioonist 3.0 kasutatakse mobiilile esimesena lähenemist. *Bootstrap* on loodud endiste Twitteri töötajate Mark Otto ja Jacob Thorntoni poolt. *Bootstrap*'i tutvustati avalikkusele esmakordselt augustis 2011 [8, p. 1]. Kõige uuem dokumentatsioon on leitav aadressilt http://getbootstrap.com. *Bootstrap* on väga hästi dokumenteeritud ning sobib kiiresti paindliku veebidisaini loomiseks. Lisaks olemasolevatele komponentidele on internetis väga palju lisakomponente, mida saab vajadusel kasutusele võtta. *Bootstrap* on konfigureeritav vastavalt kasutaja vajadustele, selleks on loodud eraldi alaleht http://getbootstrap.com/customize, kus saab kokku panna kasutaja vajadustele vastava CSS-i ja JavaScripti koodi. Komponentide kujundus on konfigureeritav. *Bootstrap* on välja arendatud, pidades silmas arendajate vajadusi – see tähendab veebilehe kiiret loomist kujundusele liialt keskendumata.

## 4.1 Kuidas on *Bootstrap* üles ehitatud

Alates versioonist 3.0 on *Bootstrap* välja arendatud lähenemisega mobiilile esimesena. Kasutamiseks on ette defineeritud hulk erinevaid komponente: vormid, nupud, tabelid, jaotussüsteemid ja ikoonid. Kõik komponendid on kujundatud, kasutades CSS-i. On jäetud võimalus kujundus üle kirjutada vastavalt kasutaja soovidele. Peamiselt on *media query*'d defineeritud nii:

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm-min) { ... }

/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md-min) { ... }

/* Large devices (large desktops, 1200px and up) */
@media (min-width: @screen-lg-min) { ... }
```

Toodud koodinäites tähistab *@screen-sm-min Less* markeeringut. *Less* lubab kasutajal defineerida CSS-is muutujaid, asjakohases töös *Less*'i funktsionaalsust täpsemalt ei kirjeldata, huvi korral soovitab töö autor tutvuda *Less*'iga veebilehel http://lesscss.org. Lisaks on

*Bootstrap*'i korral võimalik kasutada ka teist markeerimiskeelt nimega *Sass*. Kuna kasutatakse mobiilile esimesena lähenemist, siis *media query* reeglites vaateala suuruste reeglid järjest suurenevad. *Bootstrap*'i puhul on kasutatud nelja erinevat hüppeala.

Üldise praktika kohaselt paigutatakse veebilehe sisu elemendi sisse, millel on klass *container*. *Container*-klassi laius on kõige väiksemal, vaikimisi vaatealal, jäetud defineerimata. Vaateala korral suurem või võrdne 768 px on elemendi laius 750 px. Vaateala korral suurem või võrdne 992 px on elemendi laius 970 px ja vaateala korral suurem või võrdne 1200 px on elemendi laius 1170 px. Seega, sõltuvalt vaateala suurusest on elemendile defineeritud fikseeritud laius. Nullist lähenemise puhul kasutasin peatükis 3.2 sellise elemendi laiuse määramiseks protsenti. *Bootstrap*'is on defineeritud lisaks klass *container-fluid*, sellele elemendile ei ole määratud laiust, seega võtab see element 100% vaateala laiusest ning lisaks on tagatud, et enne ja pärast elementi oleks 15 px vaba ruumi. Selle klassi kasutamisest ei ole ka abi, seega tuleb defineerida uus klass *container-fluid2*, mis oleks defineeritud sarnaselt peatükis 3.2 kirjeldatule:

```
.container-fluid2 {
    margin: 0 auto;
    max-width: 1150px;
    min-width: 320px;
    width: 90%;
}
```

Erinevate küljenduste loomine käib ettedefineeritud klasside abil. HTML-i standardi järgi võib elemendile defineerida null kuni mitu klassi. Sellel loogikal baseerub *Bootstrap*'i küljendamine. *Bootstrap*'is on jaotussüsteemi jaoks defineeritud klassid *col-xs-*, *col-sm-*, *col-md-* ja *col-lg-*. Klassid on seotud hüppealade vahemikuga alustades väikseimast. Näiteks, kui soovitakse defineerida element, kus oleks reas neli sektsiooni, siis tehakse seda järgmiselt:

```
<div class="row">
    <div class="col-md-3">...</div>
    <div class="col-md-3">...</div>
    <div class="col-md-3">...</div>
    <div class="col-md-3">...</div>
</div>
```

Toodud näites kuvatakse kasutajale, kelle vaateala on suurem kui 992 px, nelja sektsiooni kõrvuti. Väiksema vaateala korral liiguvad kõik sektsioonid üksteise alla. Kui on soov, et kõige väiksema ja sellele järgneva vaateala korral oleks kõrvuti kaks sektsiooni, tuleb lisada klass *col-xs-6*.

```
<div class="row">
    <div class="col-xs-6 col-md-3">...</div>
    <div class="col-xs-6 col-md-3">...</div>
    <div class="col-xs-6 col-md-3">...</div>
    <div class="col-xs-6 col-md-3">...</div>
</div>
```

*Bootstrap*'i küljendamisel on aluseks võetud kaheteistkümnel tulbal põhinev jaotussüsteem. Klassi taga olev number määrab ära, mitme tulba laiune see element on. Klassi *col-md-3* kasutamine tähendab, et element võtab endale kolme tulba ruumi; seega maksimaalselt mahub kõrvuti neli sellist elementi. Lisaks kehtib see reegel vaatealale, mis on suurem või võrdne 992 px.

Elementide kuvamine ja mittekuvamine käib samuti klasside lisamise abil. Praegu toetatakse tabel- ja blokk-tüüpi elementide kuvamist ja mittekuvamist klasside abil. Kuvamiseks on defineeritud järgnevad klassid: *visible-xs*, *visible-sm*, *visible-md* ja *visible-lg*. Elementi näidatakse ainult konkreetse hüppeala korral. Mittekuvamiseks on defineeritud klassid: *hidden-xs*, *hidden-sm*, *hidden-md* ja *hidden-lg*. Elementi ei kuvata ainult konkreetse hüppeala korral. Kui on soov elementi kuvada mitme vaateala korral, tuleb elemendile panna mitu klassi. Sama loogika kehtib mittekuvamise korral. Lisaks on võimalik elementi kuvada või mitte kuvada veebilehe printversiooni korral. Selleks kasutatakse klasse: *visible-print* ja *hidden-print*.

Piltide muutmiseks paindlikust küljendusest sõltuvaks tuleb pildile lisada klass *img-responsive*. Sellega defineeritakse, et pildi maksimaalne laius on 100% ja kõrgus on automaatne. Klassi ei tule lisada, kui soovitakse, et pilt ei oleks sõltuvuses tema ülemelemendi suurusest.

```
<img src="..." class="img-responsive" alt="Responsive image" />
```

## 4.2 Alustamise keerukus

*Bootstrap*'iga alustamine on suhteliselt lihtne eeldusel, et omatakse teadmisi HTML-ist ja CSS-ist. Alustamist lihtsustab väga põhjalik dokumentatsioon, mis on lihtsasti jälgitav. Lisaks on valmis loodud näidisšabloone, mida saab aluseks võtta. Sobiva šablooni leidmisel piisab, kui lisatakse isikupärastamiseks pildid ja muudetakse sisu. *Bootstrap*'i raamistik on loodud selliselt, et see käitub kohe sõltuvalt vaatealast, seega väheneb testimisele kuluv aeg.

Valmiskomponentide käitumine erinevate vaatealade puhul on juba läbimõeldud ning testitud erinevate veebilehitsejate ja vaatealade korral.

## 4.3 Kasutamise keerukus

Raamistikuga tööd alustades tuleb palju selle kohta lugeda ja end selle põhimõtetega kurssi viia. Hilisemal kasutamisel tuleb uuesti üles otsida õige komponent ja vaadata, kuidas seda kasutama peab. Dokumentatsiooni hulk on suur, seega ei ole võimalik, et kogu süntaks jääks meelde juba esimesel lugemisel.

*Bootstrap*'is on palju valmiskomponente, mis lihtsustavad oluliselt veebilehe arendamist. Kõik valmiskomponendid arvestavad paindliku veebilehe arendamise põhimõtetega. Neid komponente kasutades on võimalik väga kiiresti veebilehte küljendada. Kõik komponendid on kujundatud ühtset stiili järgivalt. Kasutajale on jäetud võimalus kujundus üle kirjutada, kuid see ei ole alati vajalik. Komponentide defineeritus muudab kujunduse ülekirjutamise keeruliseks.

Keerulisem on olukord, kui soovitakse luua küljendust, kuid selleks ei ole olemasolevat komponenti. Sellisel juhul tuleb need ise nullist luua ehk see protsess on väga sarnane nullist lähenemisega. Sel juhul peab arvestama, et olemasolevat *Bootstrap*'i koodi kogemata üle ei kirjutataks. Nullist arendamisel tasub luua uued klassid ning neid klasse kasutades luua CSS-i stiilireeglid komponendi välimuse kujundamiseks.

Eelnevalt kirjeldatud klasside abil küljendamine tekitab HTML-i loetavuse probleemi. Üldise veebilehe arendamise tavapraktika on, et HTML kuvab informatsiooni ja lisab semantilist tähendust. *Bootstrap*'i kasutamisel seda tava ei järgita. Erinevate vaatealade käitumine on kirjeldatud HTML-i klasside abil. Keerulisemate küljenduste korral võib elemendil olla väga palju erinevaid klasse, mis kirjeldavad käitumist sõltuvalt vaateala suurusest. HTML-i lugedes võib väga keeruliseks muutuda arusaamine, mis erinevate vaatealade korral juhtuma hakkab. Mida rohkem on defineeritud klasse, seda mahukamaks muutub veebileht. Tulemuseks on suurenenud andmevahetus ning lisaks sellele peab veebilehitseja rohkem HTML-i töötlema, et kasutajale tulemust kuvada.

*Bootstrap*'ist hea ülevaate saamiseks tehti töö käigus weprint.ee'le erinevad küljendused, kasutades raamistiku võimalusi. Järgnevalt vaadeldakse, kuidas luua täisteenuse sektsioon,

portfoolio sektsioon, kontakti sektsioon ja alalehtede erinevad küljendused. Peatükis 4.1 on kirjeldatud, kuidas luua *Bootstrap*'iga jaluse küljendus.

Täisteenuse sektsiooni loomiseks ei ole *Bootstrap*'is vajalikku komponenti olemas. *Bootstrap*'i jaotussüsteem ei sobi vajaliku küljenduse loomiseks, kuna see koosneb kaheteistkümnest tulbast. Kaheteistkümne tulba kasutamisel ei ole võimalik jagada tulpasid nii, et moodustuks neljale ringile ja kolmele noolele võrdne ala. Võttes kasutusele üksteist tulpa, ei ole sektsioon enam veebilehe raamistiku sees äärest ääreni. Minimaalselt saab jaotussüsteemi ühe elemendi laius olla üks tulp, kuid see on proportsionaalselt liiga lai noolele planeeritud ala jaoks. *Bootstrap*'i kasutades tuleb täisteenuse sektsiooni loomiseks küljendus nullist luua.

Portfoolio sektsiooni loomiseks saab kasutada *Bootstrap*'i jaotussüsteemi. Jaotussüsteem võimaldab erinevate vaatealade korral kontrollida, mitu tulpa on kõrvuti. Selleks tuleb kasutada klasse: *col-xs-*, *col-sm-*, *col-md-* ja *col-lg-*. *Bootstrap*'is ei ole võimalust jaotussüsteemi tulpasid joondada veebilehe keskele, seega tuleb joondumine keskele nullist juurde kirjutada.

Kontakti sektsiooni küljenduse loomiseks sobib *Bootstrap* osaliselt. Võimalik on luua ettevõtte kontaktandmete ja pankadega seotud info küljenduse muutumine vastavalt vaatealale. Võimalik on kasutada elemendi peitmise funktsionaalsust kaardi peitmiseks väiksematel vaatealadel. Kaardi paiknemise küljendus tuleb luua nullist, kuna see on erilahendus. Kaart ületab kontakti sektsiooni ülemist piiri ja peitub osaliselt jaluse sektsiooni alla.

Peatükis 3.1.2 kirjeldatud alalehtede küljenduse loomiseks sobib *Bootstrap* samuti osaliselt. *Bootstrap*'i jaotussüsteemiga on võimalik sõltuvalt vaatealast muuta tulpade laiust ja seega ka tulpade ümberpaiknemist. *Bootstrap*'i jaotussüsteemis ei ole võimalik tulpa joondada veebilehe keskele, seega tuleb ühe küljenduse loomiseks olemasolevaid võimalusi täiendada.

## 4.4 Võrdlus nullist lähenemisega

*Bootstrap*'i kasutades on lihtsasti võimalik veebilehte luua, kui kasutatakse näidisšabloone. Sellisel juhul on kõik vajalik juba olemas. Veebilehele tuleb lisada pildid ja muuta sisutekstid ning paindlikku veebidisaini järgiv veebileht on valmis. Šabloone kasutamata tuleb veebileht luua, kasutades valmiskomponente, mida on suhteliselt palju ja mis on otsides lihtsasti

leitavad. Nullist lähenemise korral tuleb kõik ise kirjutada. Seega, nullist lähenemine on veebilehe küljenduse ja sisu paikasaamiseks kindlasti aeganõudvam. Lisaks tuleb nullist lähenemise korral rohkem testida. Nullist lähenemise peamiseks eeliseks on kõikide komponentide üle kontrolli omamine.

Mõlema lähenemise puhul tuleb erinevatest allikatest uurida parimaid lahendusi. *Bootstrap*'i eeliseks on see, et seda kasutatakse maailmas suhteliselt palju, seega, suure tõenäosusega on keegi juba probleemi lahendanud. Nullist lähenemise korral tuleb probleemid ise lahendada. Loomulikult on ka selle lähenemise korral võimalik leida valmislahendusi, kuid see ei pruugi sama lihtne olla.

Paindliku veebidisaini esmakordsel arendamisel on *Bootstrap*'i puhul õppimiskõver suurem. Selgeks tuleb teha paindliku veebidisaini praktikad, mida käsitleti peatükis 2, ning lisaks tuleb selgeks teha, kuidas *Bootstrap*'i kasutada. Omades mõlemast punktist teadmisi, saab *Bootstrap*'iga kiiresti toimiva veebilehe luua, samas, nullist lähenemisega tuleb komponente jälle uuesti defineerima hakata või neid varasemalt loodud projektidest kopeerida.

*Bootstrap* sobib väga hästi prototüüpimiseks. Olukorras, kus on kiiresti vaja luua toimiv veebileht, on see *Bootstrap*'i kasutades lihtsamalt teostatav kui nullist lähenemise korral. Erinevate küljenduste loomine on kiire, see võimaldab kiirelt AB testide loomist. Sellel põhjusel kasutavad palju idufirmad *Bootstrap*'i veebilehe loomiseks. Aja- ja seega ka rahafaktor on argumendid raamistiku kasuks otsustamisel. *Bootstrap* ei sobi väga hästi suuremate infosüsteemide loomiseks, kuna komponendid on liialt valmis kujundatud. Arendamise käigus peab veebilehe isikupärastamiseks palju lisatööd tegema.

*Bootstrap*'i kasutamise puhul on täheldatud usaldatavuse probleemi tekkimist. Valmiskomponente kasutava veebilehe arendamisel võib saada probleemiks omanäolisuse saavutamine. Kiireks veebilehe loomiseks kasutavad *Bootstrap*'i ka paljud petturid, kes üritavad näiteks krediitkaardiandmeid varastada või veebilehe külastajalt muud moodi raha välja petta. See on tinginud olukorra, kus paljud petturite veebilehed on väga sarnased seaduslike veebilehtedega. Eriti oluline on see probleem e-kaubanduse puhul, kus kasutaja usalduse võitmine on väga oluline. Ilma usalduseta ei soorita kasutaja ostu. Seega on mõistlik *Bootstrap*'i kasutamisel veebilehele anda unikaalne väljanägemine, see aga takistab kiiret loomist ja sarnaneb palju rohkem nullist lähenemisega.

Võrdlus *Foundation*'i kasutamisega on toodud peatükis 5, milles võrreldakse *Foundation*'i kasutamist *Bootstrap*'i ja nullist lähenemisega.

*Bootstrap*'i kasutamise eelised:

- kiirendab paindliku veebilehe arendamist;
- sisaldab moodsaid näidisšabloone;
- sobib kiireks prototüüpide loomiseks.

*Bootstrap*'i kasutamise puudused:

- usaldatavuse probleem ehk veebilehed on sarnased;
- raske luua omanäolist veebilehte;
- HTML-i klasside liigne kasutamine – HTML ei ole semantiline.

# 5. *Foundation*

*Foundation* on avatud lähtekoodiga raamistik, mis on arendatud firma Zurb poolt. Esimest korda tutvustati *Foundation*'it avalikkusele 2011. aasta septembris. Võrreldes *Bootstrap*'iga, toimus lansseerimine üks kuu hiljem. Töö kirjutamise ajal on *Foundation*'i viimase versiooni number 5.2.2. *Foundation 5*'e puhul on kasutusel mobiilile esimesena lähenemist. Kõige uuem dokumentatsioon on leitav aadressilt http://foundation.zurb.com. *Foundation*'i puhul on varasematele versioonidele ette heidetud dokumentatsiooni puudulikkust, kuid alates versioonist 5 on see muutunud. Dokumentatsioon on lihtsasti jälgitav ning sisaldab hulgaliselt koodinäiteid. Sarnaselt *Bootstrap*'iga on kasutamiseks loodud valmis šabloonid. Erinevalt *Bootstrap*'ist ei ole komponendid valmis kujundatud. Põhirõhk on pandud võimalikult paljude erinevate komponentide kirjeldamisele, et arendaja saaks olemasolevate komponentide seast valida sobivad. Sarnaselt *Bootstrap*'iga on võimalik kokku panna vajalikest komponentidest koosnev *Foundation*'i versioon, seda saab teha aadressil http://foundation.zurb.com/develop/download.html. Võrreldes *Bootstrap*'iga, on *Foundation*'i komponente võimalik palju vähem kujundada baaskonfiguratsiooni loomisel. See on ideoloogiline erinevus võrreldes *Bootstrap*'iga. *Foundation* on arendatud, pidades silmas disainerite vajadusi – palju erinevate küljenduste loomise võimalusi, millele vajalik kujundus luua.

## 5.1 Kuidas on *Foundation* üles ehitatud

Alates versioonist 5 on *Foundation* arendatud lähenemisega mobiilile esimesena. Kasutamiseks on valmis defineeritud hulgaliselt komponente, kuid need ei ole kasutaja jaoks detailideni valmis kujundatud. Seega, *Foundation*'i kasutamisel ei teki ohtu, et arendatav veebileht ei eristu teistest *Foundation*'i raamistikule arendatud veebilehtedest. *Foundation*'i puhul on *media query*'de defineerimiseks kasutatud *em*'e. *Media query*'d on defineeritud järgnevalt:

```
// Small screens
@media only screen { } /* Define mobile styles */

@media only screen and (max-width: 40em) { } /* max-width 640px, mobile-
only styles, use when QAing mobile issues */

// Medium screens
@media only screen and (min-width: 40.063em) { } /* min-width 641px, medium
screens */

@media only screen and (min-width: 40.063em) and (max-width: 64em) { } /*
min-width 641px and max-width 1024px, use when QAing tablet-only issues */

// Large screens
@media only screen and (min-width: 64.063em) { } /* min-width 1025px, large
screens */

@media only screen and (min-width: 64.063em) and (max-width: 90em) { } /*
min-width 1025px and max-width 1440px, use when QAing large screen-only is-
sues */

// XLarge screens
@media only screen and (min-width: 90.063em) { } /* min-width 1441px, xlarge
screens */

@media only screen and (min-width: 90.063em) and (max-width: 120em) { } /*
min-width 1441px and max-width 1920px, use when QAing xlarge screen-only
issues */

// XXLarge screens
@media only screen and (min-width: 120.063em) { } /* min-width 1921px,
xlarge screens */
```

*Foundation*'i puhul on kasutatud viite hüppeala. Lisaks on iga hüppeala kohta defineeritud vahemikud, kus saab ainult teatud tüüpi seadmete korral kirjutada lisareegleid. Seega on hüppealade definitsioon keerulisem kui *Bootstrap*'i puhul, kuid sellega kaasneb ka lisapaindlikkus.

*Foundation*'i puhul on kasutatud *Sass*'i. *Sass* võimaldab CSS-is defineerida muutujaid ning lisaks võimaldab see teha CSS-is matemaatilisi tehteid. Selline lisavõimalus muudab CSS-i kirjutamise ja haldamise suurte projektide puhul selgemaks ja lihtsamini hallatavaks. Selles töös *Sass*'i võimalusi ei käsitleta, huvi korral saab *Sass*'i kohta lugeda aadressilt http://sass-lang.com.

*Foundation*'is on loobutud *Bootstrap*'ist tuttava *container*-klassi kasutamisest. Klass *row* käitub küll sarnaselt, kuid selliselt defineerides hoitakse kokku üks liigne HTML-i märgend. Klass *row* on defineeritud järgmiselt:

```
.row {
    width: 100%;
    margin-left: auto;
    margin-right: auto;
    margin-top: 0;
    margin-bottom: 0;
    max-width: 62.5rem;
    *zoom: 1;
}
```

Element võtab endale 100 protsenti vaateala suurusest, kuid ei ole mitte mingil juhul suurem kui 62,5 *rem*'i ehk 1000 pikslit. *Rem* ühikut kasutatakse, kuna see ei ole sõltuvuses mitte ülemelemendi suurusest, vaid dokumendi kehale määratud suurusest. *Foundation*'il on vaikimisi määratud dokumendi kehale suurus 100% ehk 16 px, seega 16 × 62,5 = 1000 px. Element paikneb vaateala keskele.

Eelnevalt kirjeldatud klassi *row* abil kirjeldatakse *Foundation*'is jaotussüsteemi loomist. *Foundation*'i jaotussüsteem koosneb vaikimisi kaheteistkümnest tulbast, kuid see on laiendatav kuni kuueteistkümne tulbani. *Foundation*'is on jaotussüsteemi jaoks defineeritud klassid *small*, *medium* ja *large*. Sarnaselt *Bootstrap*'iga näitab klassi taga olev number, mitme tulba ruumi element endale võtab. Järgnevalt esitatakse näide, kuidas defineerida weprint.ee jaluse sektsioon nii, et kõrvuti oleks kas kaks või neli sektsiooni:

```
<div class="row">
    <div class="small-6 large-3 columns">...</div>
    <div class="small-6 large-3 columns">...</div>
    <div class="small-6 large-3 columns">...</div>
    <div class="small-6 large-3 columns">...</div>
</div>
```

*Foundation*'i puhul kasutatakse klasside eesliiteid *small*, *medium* ja *large*. Eesliite *small* kasutamisel kehtib reegel alates kõige väiksemast vaatealast kuni kõige suuremani, kui ei ole lisatud lisaklasse, mis kirjutaksid selle käitumise üle. Eesliide *medium* kehtib järgmise vaateala definitsiooni korral ning eesliide *large* võtab kokku kõik järgnevad vaatealad.

*Foundation*'is on loodud võimalus, kuidas element joondada lehe keskele. Selleks kasutatakse klasse: *small-centered*, *medium-centered* ja *large-centered*. Soovi korral saab sõltuvalt hüppeala suurusest muuta elemendi joondumist mitte keskele, seda tehakse klassidega: *small-uncentered*, *medium-uncentered* ja *large-uncentered*. Järgnevalt tuuakse näide, kuidas luua kuuest tulbast koosnev element, mis oleks joondatud vaateala keskele ainult kõige väiksema (*small*) ja talle järgneva (*medium*) vaateala korral:

```
<div class="row">
    <div class="small-6 small-centered large-uncentered columns">...</div>
</div>
```

*Foundation*'is on loodud võimalus, defineerimaks olukorda, kus kindlasti on vaja, et teatud arv elemente oleks järjest ning elemendid oleksid võrdse suurusega, võttes endale kogu võimaliku ruumi. Selleks on defineeritud klassid: *small-block-grid*, *medium-block-grid* ja *large-block-grid*. Number klassi järel näitab, mitu elementi on reas. Järgnevalt tuuakse näide, kus kõige väiksema vaateala korral on kaks elementi kõrvuti ja muul juhul on neli elementi kõrvuti:

```
<ul class="small-block-grid-2 medium-block-grid-4">
    <li>...</li>
    <li>...</li>
    <li>...</li>
    <li>...</li>
</ul>
```

Sarnaselt *Bootstrap*'iga on võimalik ka *Foundation*'it kasutades elemente näidata või peita sõltuvalt vaateala suurusest. Lisaks on *Foundation*'i kasutamisel võimalik elemente näidata või peita sõltuvalt kasutaja seadme asendist, seda siis mobiilsete seadmete korral, mis saavad aru seadme orientatsioonist. Näiteks on võimalik element peita või näidata püstise asendi korral. Selleks kasutatakse klasse: *show-for-portrait* ja *hide-for-portrait*. Sarnaselt saab muuta käitumist külili oleva seadme korral. Selleks kasutatakse klasse: *show-for-landscape* ja *hide-for-landscape*. Veel on võimalik elementi kuvada või mitte kuvada sõltuvalt sellest, kas kasutataval seadmel on võimaldatud ekraani puudutamine ehk *touch*. Selleks kasutatakse klasse: *show-for-touch* ja *hide-for-touch*. *Bootstrap*'ist tuttavat elemendi kuvamist või mittekuvamist veebilehe printversiooni korral *Foundation*'is ei toetata.

*Foundation*'i puhul ei tule piltidele lisaklasse määrata, et need muutuksid paindlikust kujundusest sõltuvaks. Kõikide piltide maksimaalne laius on määratud 100% ja kõrgus automaatseks. Seega, kõik pildid on kohe sõltuvad ülemelemendi suurusest. Fikseeritud suurusega pildi lisamiseks tuleb pildile lisada laius või kõrgus.

Kasutades komponenti nimega *interchange*, on võimalik laadida veebilehe sisu sõltuvalt kasutaja seadmest. Näiteks on võimalik suure vaateala korral kuvada kasutajale Google Maps'i kaarti, kuid väikese vaateala korral kuvada pilti. Sama loogikat kasutades on võimalik sõltuvalt kasutaja seadme omadustest laadida kõige sobilikuma suurusega pilt. Selleks tuleb märgendile *img* lisada atribuut *data-interchange*. Seda tehakse järgnevalt:

```
data-interchange="[image_path, (media query)], [image_path, (media query)]"
```

Kus *image_path* viitab pildi asukohale ja *media query*'ga määratakse ära, millisel juhul seda pilti kuvatakse. Märgendile *img* ei lisata atribuuti *src*, et vältida veebilehitseja topeltpäringut serverisse.

```
<img data-interchange="[/path/to/default.jpg, (default)], [/path/to/bigger-image.jpg, (large)]">
```

Toodud koodinäites laaditakse pilt nimega default.jpg, kui kasutaja vaateala suurus on kas *small* või *medium*. Vaateala suuruse *large* korral laaditakse pilt nimega bigger-image.jpg. *Media query* reegli võib ise kirjutada või kasutada ette defineeritud reegleid:

**Tabel 3. *Interchange*-nimetusega päringud**

| **Nimetus** | **Media Query** |
|---|---|
| default | only screen and (min-width: 1px) |
| small | only screen and (min-width: 1px) |
| medium | only screen and (min-width: 641px) |
| large | only screen and (min-width: 1024px) |
| landscape | only screen and (orientation: landscape) |
| portrait | only screen and (orientation: portrait) |
| retina | only screen and (-webkit-min-device-pixel-ratio:2),<br>only screen and (min--moz-device-pixel-ratio:2),<br>only screen and (-o-min-device-pixel-ratio: 2/1),<br>only screen and (min-device-pixel-ratio: 2),<br>only screen and (min-resolution: 192dpi),<br>only screen and (min-resolution: 2dppx) |

## 5.2 Alustamise keerukus

*Foundation*'iga alustamise keerukus on suhteliselt sarnane *Bootstrap*'iga alustamisega. Piisab raamistiku allalaadimisest ja saab kohe kasutama hakata, seda siis mõlema raamistiku CSS-versiooni korral (ilma *Less* ja *Sass* võimalusteta). *Foundation*'i CSS-versiooniga tuleb kohe kaasa index.html fail, kus on koodinäited erinevate komponentide kasutamisest. Lisaks on *Foundation*'il väga mahukas dokumentatsioon. Kindlasti tuleb läbi lugeda sektsioon, kus kirjeldatakse alustamist ja hiljem lugeda erinevate komponentide kasutamise kohta. Täielik ülevaade dokumentatsioonist on oluline, et olla teadlik olemasolevate komponentide võimalustest ja kasutamisjuhtudest. Lisaks on võimalik valida välja endale sobiv šabloon ja hakata veebilehte arendama valitud šabloonile. Šabloonid on erineva küljendusega, kuid ei sisalda sisutekste. Komponentide laialdane valik tagab selle, et ise ei pea arendamise ja testimise peale nii palju aega kulutama, kui seda tuleb teha nullist arendamise korral. *Foundation*'i puhul on olemasolevate komponentide arv suurem kui *Bootstrap*'i puhul. *Foundation*'i puhul ei ole komponendid liialt kujundatud, vaid on loodud baasvälimus, arendamise käigus tuleb komponente endal kujundada.

## 5.3 Kasutamise keerukus

Sarnaselt *Bootstrap*'iga on *Foundation*'i puhul vaja alguses palju lugeda, et viia ennast kurssi üldiste arendamise põhimõtetega. Kindlasti tuleb tutvuda erinevate komponentide võimalustega, et hilisema arendamise käigus saaks vajalikku komponenti kasutada.

*Foundation*'i olemasolevate komponentide baas on väga suur ja suure tõenäosusega on kõik vajalik küljendamise jaoks juba olemas. Arendajal piisab vajaliku komponendi üles otsimisest ja sellele vajaliku kujunduse loomisest. Juhul kui küljenduse jaoks soovitavat komponenti ei leidu, tuleb järgida sarnaseid põhimõtteid komponendi loomiseks, nagu seda tuli teha *Bootstrap*'i kasutamise korral peatükis 4.3.

*Foundation*'ist hea ülevaate saamiseks prooviti luua weprint.ee erinevaid küljendusi, kasutades raamistiku võimalusi. Järgnevalt vaadeldakse, kuidas luua täisteenuse sektsioon, portfoolio sektsioon, kontakti sektsioon ja alalehtede erinevad küljendused. Varem on kirjeldatud, kuidas luua jaluse käitumine (5.1).

Täisteenuse sektsiooni loomiseks ei ole *Foundation*'is vajalikku komponenti olemas. Kasutada ei saa kaheteistkümnest tulbast koosnevat jaotussüsteemi, kuna tulpade arv ei

jagune võrdselt ning lisaks oleks minimaalne ühe tulba laiune ala proportsionaalselt liiga suur noolele mõeldud alale. Seega tuleb luua nullist komponent, mis käituks täpselt nii, nagu soovitakse. Sellise komponendi arendamine on sarnane nullist lähenemisega, seega ei oleks abi *Foundation*'i raamistiku võimalustest.

Portfoolio sektsioon peab paiknema vaateala keskel. Piltide arv reas peab olema sõltuvuses vaateala suurusest. *Foundation*'is on keskele paiknemise määramiseks klass *small-centered*. Lisaks tuleb kasutada klasse *small-block-grid-* ja *large-block-grid-*. Soovitud küljenduse loomiseks tuleb HTML luua järgnevalt:

```
<div class="row">
    <div class="small-11 small-centered columns">
        <ul class="small-block-grid-3 large-block-grid-4">
            <li>...</li>
            <li>...</li>
            <li>...</li>
            <li>...</li>
        </ul>
    </div>
</div>
```

Klass *small-11* ütleb, et element võtab üheteistkümne tulba ruumi kaheteistkümnest võimalikust tulbast. Lisaklassi *small-centered* abil luuakse käitumine, et element paikneks alati vaateala keskel, sellega jäetakse elemendi mõlemale poole pool ühe tulba laiust vaba ruumi. Klass *small-block-grid-3* tagab, et väikseima vaateala korral on reas kolm elementi ning klassi *large-block-grid-4* abil näidatakse suurimate vaatealade korral nelja elementi reas. Kõik elemendid on alati võrdse suurusega. Selliselt defineerides on jäetud tähelepanuta, et kõige väiksema vaateala korral peaksid elemendid muutuma hüppeliselt väiksemaks, et võita ruumi lehe kõrguse arvelt. Tegelikult see ei ole probleem, kuna iga elemendi laius on sõltuvuses vaateala suurusest, seega muutub nii elemendi laius kui ka kõrgus.

Sarnaselt *Bootstrap*'iga tuleb kontakti sektsioonis *Foundation*'i puhul nullist luua kaardi paiknemine. Ettevõtte andmete ja pankadega seotud info küljendamiseks on kõik vajalik *Foundation*'is olemas. Lisaks on olemas võimalus, kuidas kaart väiksemate vaatealade korral peita.

Peatükis 3.1.2 kirjeldatud alalehtede küljenduste loomiseks sobib *Foundation*'i jaotussüsteem väga hästi. Sõltuvalt vaatealast on võimalik muuta elementide laiust ja seega ka paiknemist veebilehel. Elemendi paiknemiseks vaateala keskele on võimalus olemas. Seega saab kõik kolm alalehe küljendust luua raamistikus leiduvate komponentide abil.

## 5.4 Võrdlus *Bootstrap*'i ja nullist arendamisega

*Foundation*'i puhul on õppimiskõver sarnaselt *Bootstrap*'i kasutamisega suurem kui nullist arendamise puhul. Mõlemad raamistikud on alustamise ja kasutamise seisukohalt sarnase keerukusega. Raamistiku kasutamisel sõltub keerukus sellest, kas konkreetsel raamistikul on vajalik komponent eeldefineeritud. Komponendi puudumisel tuleb see sarnaselt nullist arendamisega endal luua.

Sarnaselt *Bootstrap*'iga on *Foundation*'is võimalik veebileht luua olemasolevale šabloonile. *Foundation*'i šabloonid erinevad küljenduse poolest. Arendamist lihtsustab sobiva küljendusega šablooni valimine. *Foundation*'i puhul ei sisalda šabloonid näidissisu. *Bootstrap*'i šabloonid on valmis veebilehed, kus tuleb sisutekstid kirjutada ja disaini vastavalt vajadustele täiendada. *Foundation*'it võib võrrelda tööriistakastiga, kus on kõik vajalikud tööriistad olemas. *Bootstrap* on võrdluse poolest moodulite komplekt, mis tuleb kokku panna.

Küljendamise seisukohalt on *Foundation* paremini läbi mõeldud kui *Bootstrap*. Jaotussüsteemi loomisel on lisavõimalustena olemas elemendi joondamine keskele ning ritta võrdse alaga elementide defineerimine. Mõlema raamistiku kasutamise korral võib tekkida probleem, et küljendust ei ole võimalik raamistiku komponentidega luua. Lahendus on küljenduse loomine nullist, mis on sarnane nullist arendamisega. Siinkohal on oluline teadvustada, et sellised probleemid tekivad raamistike kasutamisel, kui veebilehe disainimisel ei ole arvestatud raamistiku võimalustega või on kindel soov luua erilisem küljendus.

*Foundation*'i kasutamisel ei teki üldiselt probleemi, et loodav veebileht sarnaneks disainilt paljude teiste veebilehtedega, kuna *Foundation*'i puhul on komponendid vähesel määral disainitud. Arvestama peaks, et kindlasti tuleks komponentidele soovitud disain luua, kasutades selleks CSS-i võimalusi. See protsess sarnaneb nullist arendamisega. *Bootstrap*'i puhul tuleb lisaks CSS-i kasutades üle kirjutada olemasolev disain või teise võimalusena muuta komponentide baaskonfiguratsiooni. Mõlema variandi kasutamisel on tööd rohkem kui *Foundation*'i ja nullist arendamise puhul.

*Foundation*'is on loodud kasulikke komponente, mida ei ole *Bootstrap*'is. Üheks selliseks on *interchange*, mida kirjeldati peatükis 5.1. Sõltuvalt vajadusest võib tekkida olukord, kus kasutatavas raamistikus ei ole vajalikku komponenti, kuid mõnes teises raamistikus on see olemas. Vajaliku komponendi puudumisel tuleb nii nullist lähenemisel kui ka raamistiku kasutamisel komponent ise luua. Teise alternatiivina võib leida vajaliku komponendi

raamistiku kommuunist, kui ka see tulemuseni ei vii, on võimalik lahendus leida internetiotsingu abil. Vale on eeldada, et raamistiku kasutamisel ei tule komponente ise arendada. Probleemide ilmnemisel tuleb lahendusi otsida internetist, seda sarnaselt nullist lähenemisega.

*Foundation* sobib hästi veebilehe aluspõhja loomiseks, mis jälgiks paindliku veebidisaini põhimõtteid. Erinevalt *Bootstrap*'ist on *Foundation*'is põhirõhk komponentide baasi loomisel. Kiireks veebilehe loomiseks sobib *Bootstrap* paremini, kuna komponendid on disainitud. Lisaks on olemas näidissisuga šabloonid. *Foundation* sobib paremini keerukamate infosüsteemide loomiseks.

*Foundation*'i kasutamise eelised:

- lihtne lisada oma disain;
- sobib hästi erinevate küljenduste loomiseks;
- sisaldab palju valmis komponente.

*Foundation*'i kasutamise puudused:

- ei sobi kiireks veebilehe loomiseks;
- HTML-i ja CSS-i haldamine suurte projektide puhul võib muutuda probleemseks;
- HTML-i klasside liigne kasutamine – HTML ei ole semantiline.

# 6. Kokkuvõte

Magistritöö eesmärk oli analüüsida ja võrrelda paindliku veebidisaini loomise meetodeid, saavutamaks optimaalseim informatsiooni edastamine kasutajale. Variandid, mida kasutada tuleb, on järgmised: veebilehe küljenduse muutmine, sektsioonide kuvamine või kaotamine ja piltide muutmine vaatealast ning kasutatava seadme omadustest sõltuvaks. Toodud variantide loomiseks tuleb kindlasti kasutada *media query* võimalusi, et erinevate seadmete korral veebilehte muuta. Lisavõimalusena on JavaScripti kasutades võimalik saavutada sobivate piltide laadimine. Weprint.ee arendamisel ei kasutatud võimalust seadme omadustest sõltuvate sisupiltide laadimiseks. Kasutati ainult täisteenuse taustapildi laadimist sõltuvalt seadme omadustest, seega sisupilte oleks saanud kasutajale optimeeritumal viisil edastada. Kuna kasutati eelpool nimetatud variante, arendati veebileht, mida on optimaalne kasutada eesmärgiks seatud seadmetega: nutitelefonidega, tahvel-, süle- ja lauaarvutitega.

Töö olulisimaks tulemuseks on analüüsitud ja võrreldud, kuidas luua erinevaid paindliku veebidisaini põhimõtteid järgivat veebilehte. Lahendati erinevate küljenduste loomisel tekkivad probleemid ning näidati, kuidas muuta pildid vaatealast sõltuvaks. Toodi välja raamistike kasutamise sobilikkus olenevalt projekti eesmärkidest ning tuvastati, milliseid eeliseid ja puudusi raamistike kasutamine kaasa toob. Valmis paindlikke põhimõtteid järgiv veebileht weprint.ee. Järgnevalt tuuakse põhitulemuste loetelu:

- paindliku veebidisaini arendamise erinevate lähenemisviiside võrdlemine;
- nullist arendamisega weprint.ee küljenduste loomine;
- weprint.ee küljenduste loomine, kasutades *Bootstrap*- ja *Foundation*-raamistikke;
- raamistike kasutamise sobivuse analüüs sõltuvalt projekti eesmärkidest;
- piltide muutmine vaatealast sõltuvaks;
- paindlikke põhimõtteid järgiva veebilehe weprint.ee valmimine.

Paindliku veebidisaini arendamiseks ei ole kindlat parimat lähenemisviisi. Lähenemisviisi valikul tuleb arvestada projekti eesmärkidega ning otsustada, milliseid kompromisse ollakse valmis tegema. Nullist arendamine on kõige paindlikum, aga ka kõige aeganõudvam

võimalus. Analüüsitud raamistikud sobivad erinevat tüüpi projektide jaoks: *Bootstrap* sobib kiireks prototüüpimiseks ja *Foundation* sobib hästi erinevate küljenduste kiireks loomiseks. Raamistike peamiseks eeliseks on väiksem ajakulu testimisele, kuna raamistike valmiskomponendid on eelnevalt testitud erinevate vaatealade ja veebilehitsejate puhul.

Magistritöös seatud eesmärk saavutati. Nii nullist lähenemise kui ka raamistike kasutamisel analüüsiti ning võrreldi erinevaid viise, kuidas vaateala suurusest sõltuvalt muuta informatsiooni kuvamine kasutajale optimaalseimaks.

Ühe võimaliku magistritöö edasiarendusena tuleks luua erineva suurusega pildid kõikidest veebilehele lisatavatest piltidest. JavaScripti võimalusi kasutades tuleks laadida sobivaim pilt olenevalt seadme omadustest, millega veebilehte külastatakse. Teise võimaliku edasiarendusena võiks kaaluda serveripoolse koodi muutmist selliselt, et mobiilsete seadmete kasutajatele tagastataks väiksem hulk informatsiooni (näiteks e-poes tagastataks mobiilsete seadmete kasutajale 50 toodet, kuid süle- ja lauaarvutite puhul 100 toodet). See vähendaks mobiilsete seadmete puhul andmemahtu ning muudaks veebilehitsemise kiiremaks.

# Summary

The aim of this Master's thesis was to analyse and compare the development methods of responsive web design. The attributes that can be used are following: changing the layout, hiding or showing different sections, making pictures responsive. Media query should be used for creating previously mentioned attributes. As an option it is possible to load pictures using JavaScript. Content picture loading dependent on device parameters was not used on weprint.ee development. Different picture loading was only used on full service section background. Developed website was optimized for smart phones, tablets, laptops and desktop computers.

The most important result of this work is analyse and comparison of different methods developing responsive website. The problems that appeared constructing different layouts were solved. Suitability of different frameworks were analysed regarding the aims of the project. Pros and cons of using frameworks were brought out. Responsive website weprint.ee was developed. The main results gathered in the context of the Master's thesis:

- comparing different development approaches of responsive web design;
- creating weprint.ee layouts from the scratch;
- creating weprint.ee layouts using the frameworks of Bootstrap and Foundation;
- the analysis of the suitability for using frameworks;
- making pictures responsive;
- responsive website weprint.ee was developed.

There is no the best universal approach for developing responsive web design. The most suitable approach should be selected taking under consideration the aims of the project and the compromises that are accepted to be made. Developing from the scratch is definitely the most flexible, but also the most time consuming method. Bootstrap framework is suitable for easy prototyping and Foundation is suitable for setting up fast layouts. The main argument for using frameworks is the factor of saving time on testing, because framework components have previously been tested on different screen sizes and on different browsers.

The goal of Master's thesis was accomplished. Different development methods of responsive web design were compared and analysed. Developed website was optimized for displaying information on different screen sizes.

One possible development option for the Master's thesis would be making different size images of all the images used on the website. Using the possibilities of JavaScript would let the system to detect and print the most suitable image size for the device that the site is being visited from. The other possible development direction would be editing the server side code the way that visitors with mobile devices would receive less information than desktop computer users (e.g. web shop visitors would load 50 products while the visitors with desktop computers and laptops would receive 100 products). This would decrease the amount of data that needs to be processed on mobile devices.

# Kasutatud kirjandus


[1] D. Bosomworth, „Smart Insights,“ 24 märts 2014. [Võrgumaterjal]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Kasutatud 01 mai 2014].

[2] T. Kadlec, Implementing Responsive Design: Building sites for an anywhere, everywhere web, Berkeley: New Riders, 2013.

[3] E. Marcotte, Responsive Web Design, New York: A Book Apart, 2011.

[4] B. Frain, Responsive Web Design with HTML5 and CSS3, Birmingham: Packt Publishing Ltd. , 2012.

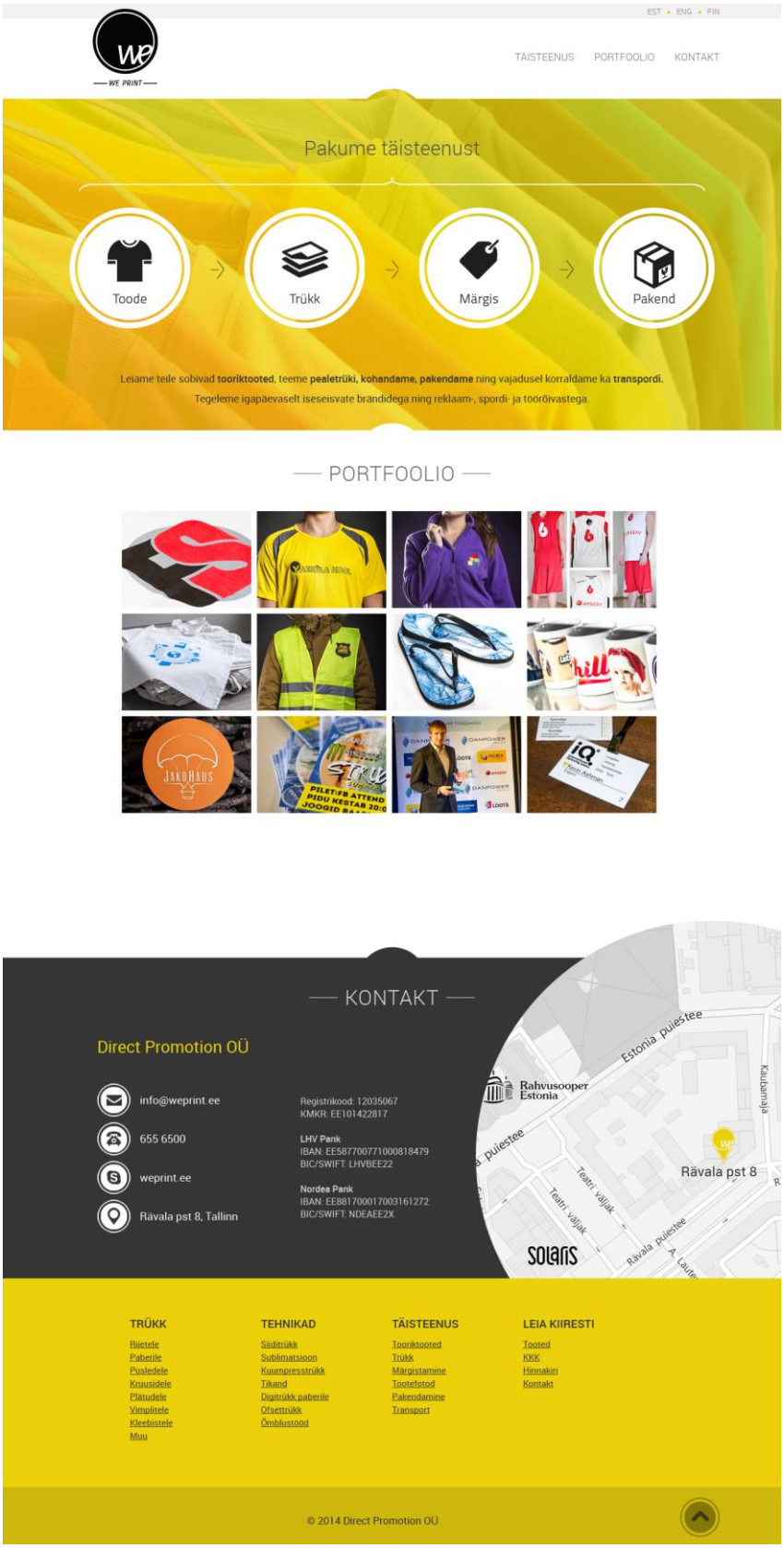
[5] S. Jehl, „GitHub,“ 2011. [Võrgumaterjal]. Available: https://github.com/scottjehl/Respond. [Kasutatud 1. aprill 2014].

[6] L. Wroblewski, Mobile First, New York: A Book Apart, 2011.

[7] T. Hines, „teehan+lax,“ 24 mai 2012. [Võrgumaterjal]. Available: http://www.teehanlax.com/blog/density-converter/. [Kasutatud 1. aprill 2014].

[8] J. Spurlock, Bootstrap, Sebastopol: O’Reilly Media, 2013.

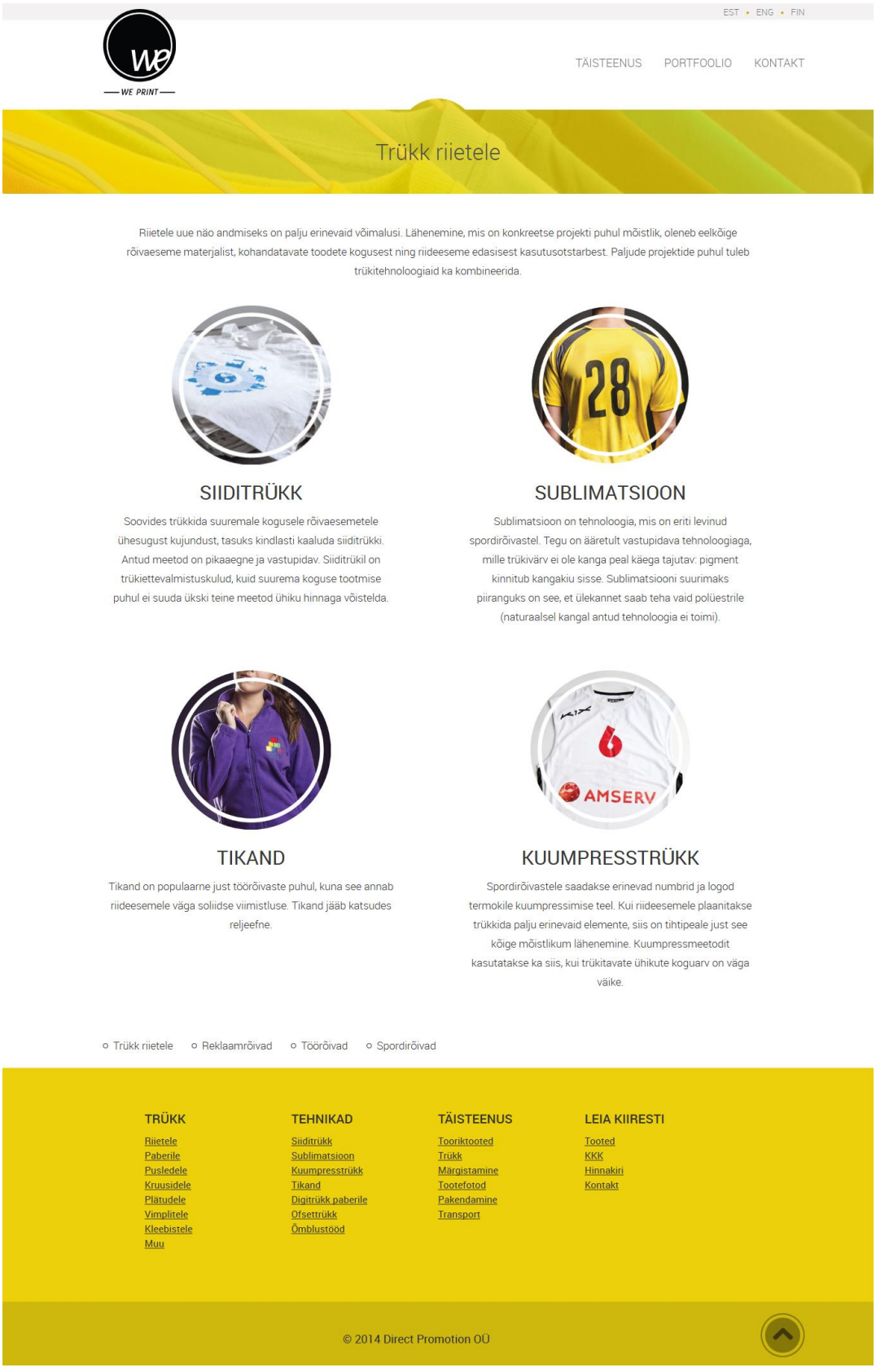
# Lisa 1. Avaleht vaateala laiusega 1366 px



# Lisa 2. Avaleht vaateala laiusega 766 px


# Lisa 3. Avaleht vaateala laiusega 320 px

# Lisa 4. Alalehe 1. küljendus vaateala laiusega 1366 px



# Lisa 5. Alalehe 2. küljendus vaateala laiusega 1366 px

# Lisa 6. Alalehe 3. küljendus vaateala laiusega 1366 px