Sisukord

CSS **5**

CSS LINKIMINE HTML DOKUMENDIGA **6**

VÄLISE STIILILEHE RAKENDAMINE 6
SISEMISE STIILI RAKENDAMINE 7
REASTIILI RAKENDAMINE 7

CSS ÕIGEKIRI **7**

EESLIITED 8
ERINEVAD SELEKTORID 9
Selektoritega määratud stiilide hierarhia (prioriteedid) *9*
Mistahes element *9*
Elemendi identifikaatoriga (id) selektorid *9*
Mitu selektorit korraga *10*
Klassid *10*
Kujundus vastavalt elementide sisaldumisele üksteise sees *11*
Kujundus vastavalt elementi sisaldavale elemendile (parent) *12*
Kujundus vastavalt elementide otsesele järgnevusele *12*
Atribuudi selektorid *12*
Pseudoklassid ja pseudoelemendid *13*
Pseudoklassid 13
Kasutaja sisendiga seotud pseudoklassid 14
Elementide järjestusega seotud pseudoklassid 15
Ainsa tütarelemendi pseudoklassid 16
Pseudoelemendid 17
Esimese tekstirea ja tähemärgi pseudoelemendid 17
Pseudoelemendid lisamaks sisu elementide ette ja järele 17
Kasutaja poolt valitud osa 18
CSS VÄRVID 18
Värvikonstandid *18*
Värvid HEX väärtustena *19*
Värvid RGB ja RGBA väärtustena *19*
Värvid HSL ja HSLA väärtustena *19*
MITME CSS FAILI ÜHENDAMINE 20
KALKULATSIOONID MÕÕTÜHIKUTEGA 20

CSS VAHENDID **20**

VEEBILEHE ELEMENTIDE TAUST 21
Taustavärv *21*
Taustapilt ja selle paigutus *21*
Tausta stiil lühikeselt kirjutatuna *22*
Taustapildi suurus *22*
Taustapildi koordinaatide alguspunkt *23*
Tausta kaetav ala *23*
Mitu taustapilti *24*
Astmiktäide (gradient) *24*
Lineaarne astmiktäide 25
Lineaarse astmiktäite suund 25
Lineaarse astmiktäite värvipositsioonide fikseerimine 26
Korduv lineaarne astmiktäide 26
Kooniline astmiktäide 27

Koonilise astmiktäite nurk ja keskpunkt ... 27
Koonilise astmiktäite värvipositsioonide fikseerimine ... 28
Korduv kooniline astmiktäide ... 29
Radiaalne astmiktäide ... 30
Radiaalse astmiktäite värvipositsioonide fikseerimine ... 30
Radiaalse astmiktäite kuju ... 31
Radiaalse astmiktäite keskpunkt ... 31
Radiaalse astmiktäite suurus ... 31
Korduv radiaalne astmiktäide ... 32
TEKSTI KUJUNDUSVAHENDID ... 32
Teksti värv ... *32*
Lõigu kujundus ... *33*
Lõigu joondus ... 33
Taandrida ... 33
Reasamm ... 33
Font ja sellega seonduv ... *33*
Font ... 34
Kirja suurus ... 36
Kaldkiri ... 36
Paks kiri ... 37
Teksti joonimine, allajoonimine, läbikriipsutamine ... 37
Teksti joonimise värv ... 37
Teksti joonimise stiil ... 38
Suur- ja väiketähed ... *38*
Tähe- ja sõnade vahe ... *38*
Teksti „murdmine“ ... *39*
Teksti suund ... *39*
Tühja ruumi haldus ... *39*
Loendid ... *40*
Nummerdatud loend ... 40
Täpploend ... 41
Numbri/täpi paigutus ... 41
Loendi stiil lühidalt ... 42
Tsitaadid ... *42*
Tekstiefektid ... *43*
Teksti vari ... 43
Tekst „üle serva“ ... 43
Tekst mitmes veerus ... 44
Veergude vahekaugus ... 44
Joon veergude vahel ... 44
Teksti jagunemine veergude vahel ... 45
Teksti laotus üle mitme veeru ... 46
LINGID ... 46
RAAMJOONED ... 47
Raamjoone stiil ... *47*
Raamjoone paksus ... *47*
Raamjoone värv ... *48*
Raamjooned elemendi erinevatel külgedel ... *48*
Raamjoone lühendatud kirjeldus ... *49*
Ümardatud nurgad ... *49*
Raamjoon pildist ... *50*
Vari ... *52*
Kontuur ... *54*
POLSTERDUS ... 55
VEERISED ... 55
TABEL ... 56

Tabeli raamjoonte seaded ... *57*
Tabeli pealdis ... *58*
OBJEKTI NÄHTAVUS, LÄBIPAISTVUS JA FILTRID ... 58
Elemendi nähtamatuks muutmine ... *58*
Objektide läbipaistvus ... *58*
KASUTAJALIIDES ... 58
Ploki suuruse muutmine ... *59*
Kursori kuju määramine ... *59*
Teksti valimise lubamine ... *60*
LOENDUR EHK AUTOMAATNE NUMMERDAMINE ... 60
OBJEKTIDE SUURUS JA PAIGUTUS ... 61
Suurus ... *61*
Kasti mudel ... 61
Maksimaalsed ja minimaalsed mõõdud ... 62
Elemendi mõõdud väiksemad kui tema sisu ... 62
Elemendi kärpimine ... *63*
Elemendi kuvamisviis ... *64*
Objektide paigutus ... *65*
Objekti paigutus z-teljel ... *66*
Objekti „hõljumine“ vasakul/paremal ... *66*
Vertikaalne joondus ... *67*
KOHANDUV DISAIN ... 67
Meediapäringud ... *68*
Vaateava ehk viewport ... *69*
Flexbox ... *70*
Flex-konteineri omadused ... 70
Flex konteineri sisu paigutamise suund ... 71
Flex elementide joondamine ... 71
Flex ridade/veergude murdmine ... 72
Flex ridade joondus ... 73
Flex-konteineri sisu paigutamise suund ja ridade/veergude murdmine koos ... 73
Flex-elementide omadused ... 74
Flex-elemendi järjestuse muutmine ... 74
Flex-elemendi joondamine ... 74
Flex-elemendi suhteline suurus ... 74
Grid ... *75*
Grid-konteineri omadused ... 76
Võrestiku defineerimine ... 76
Veergude ja ridade defineerimine ... 76
Võrestiku defineerimine lühikesel kujul ... 78
Tühi ruum ridade ja/või veergude vahel ... 78
Piirkondade defineerimine grid-elementide sidumiseks võrestikuga ... 78
Võrestiku joondamine ... 79
Võrestiku joondamine horisontaalselt ... 80
Võrestiku joondamine vertikaalselt ... 80
Grid-elementide sisu joondamine ... 81
Elementide sisu joondamine horisontaalselt ... 81
Elementide sisu joondamine vertikaalselt ... 82
Veergude ja ridade automaatne suurus ... 82
Elementide automaatne paigutus võrestikul ... 83
Võrestiku defineerimine ja kujundamine lühidalt ... 84
Grid-elementide omadused ... 84
Grid-elementide paigutamine ridadele ja veergudele ... 84
Grid-elemendi määramine veeru-joonte järgi ... 85
Grid-elemendi määramine rea-joonte järgi ... 86
Kompaktsemad omadused elementide rea-joonte ning veeru-joonte järgi paigutamiseks ... 86

Grid-elementide nimetamine sidumiseks piirkondadega ... 87
Grid-elemendi sisu joondamine ... 87
Grid-elemendi sisu joondamine horisontaalselt ... 87
Grid-elemendi sisu joondamine vertikaalselt ... 88
TRANSFORMEERIMINE ... 88
Läbipaistvus ... *88*
2D Transformatsioonid ... *89*
3D transformatsioonid ... *90*
3D teisendused ... 90
3D vaate omadused ... 91
Ruumilise asendi näitamine ... 91
Perspektiiv ... 92
Vaatepunkt ... 92
Elemendi nähtavus tagantpoolt vaadates ... 93
ANIMEERIMINE ... 94
Siire (transition) ... *94*
Omadus, mille siire määratakse ... 94
Siirde kestus ja viivitus ... 94
Siirde kiirendusfunktsioon ... 95
Siirde määramine lühendatult, kasutamine ... 95
Võtmekaadrid ... *95*
Animatsioon ... *96*
Võtmekaadrite sidumine elemendiga ja animatsiooni kestus ... 96
Animatsiooni kiirendusfunktsioon ... 97
Viivitus enne animatsiooni algust ... 97
Animatsiooni kordamine ... 97
Animatsiooni kulgemise suund ... 98
Elemendi omadused enne ja pärast animatsiooni ... 98
Animatsiooni olek ... 98
Animatsiooni määramine ühe omadusena ... 99
Mitme animatsiooni korraga kasutamine ... 99
Animatsiooni näide ... 99

CSS

1990-ndate aastate keskpaigaks oli veeb plahvatuslikult arenenud ja levinud, esile kerkis veebilehtede kujundamise probleem. Jaanuaris 1997. kuulutati W3C poolt välja HTML 3.2, mille raames lisati HTML keelele terve hulk vormindusega seotud atribuute (*color, bgcolor* jpt) ning näiteks ka <font> element. See lisas küll võimalusi kuid muutis suurte veebilehestike arendamise väga vaevarikkaks, sest info teksti ja värvide kohta tuli lisada igale üksikule lehele.
Detsembris 1997 kuulutas W3C välja HTML 4 standardi, milles hakati kujunduselementidest loobuma ning kujundust veebilehe sisust täielikult eraldama. Selleks võeti kasutusele CSS, mis oli esmakordselt avalikustatud juba 17. detsembril 1996. CSS-i eelkäijaks oli *Standard Generalized Markup Language* (SGML).

CSS (*Cascading Style Sheets*) – astmelised stiililehed ehk kaskaadlaadistik ehk stiililehed on keel, milles märgitakse üles veebilehtede kujundus. Vastavate reeglite järgi pannakse kirja, kuidas erinevaid HTML elemente peab näitama (värvid, teksti font, suurus jne). CSS-i võib kasutada html faili sees aga ka välise failina (laiendiga css).
Kasutades välist stiililehte (css fail) saab veebilehe sisu ning kujunduse lahus hoida, mis lihtsustab veebilehe loomist ja hilisemat muutmist. Veebilehe kujunduse muutmiseks pole hiljem tarvis üle vaadata terve HTML dokumendi kõiki elemente vaid piisab eraldi CSS faili muutmisest või asendamisest.

Sama stiililehte võib kasutada terve veebilehestiku erinevate lehtede jaoks ning siis saab terve veebilehestiku kujunduse vaid ühe faili muutmisega ringi teha.

Nimetus kaskaad (*cascading*) tuleb sellest, et mitu erinevat stiili kogutakse kokku üheks.

Stiile võetakse arvesse järgmises järjekorras, kus viimane on kõrgeima prioriteediga:

1. Veebilehitseja vaikestiil (*Browser Default*). Ka veebilehitseja sätetes on määratud kuidas veebilehe elemente näidatakse, kui veebilehe autor pole ise midagi määranud, siis kasutataksegi veebilehitseja vaikestiili.
2. Väline stiil ehk stiilileht ehk lingitud CSS fail (*External Style Sheet*).
3. Sisemine stiil ehk HTML elemendi <head> osas defineeritud stiil (*Internal Style Sheet*).
4. Reastiil, konkreetse HTML elemendi sees *style* atribuudiga määratud stiil (*Inline style*).

Nagu HTML ja teised keeled ning tehnoloogiad, nii on ka CSS edasi arenenud, praeguseks on kasutusel viimane ametlik versioon 2.1. Alates 1998. aastast töötatakse versiooni CSS3 kallal, arendus sai tõelise hoo sisse 2009 aastal.

Joonis 1 CSS3 logo

CSS3 on moodulpõhine. Erinevad moodulid saavad W3C soovituse individuaalselt, sõltumata ülejäänud moodulite arendusstaadiumist. Erinevate moodulite tööjärge saab jälgida aadressil: http://www.w3.org/Style/CSS/current-work
Millist funktsionaalsust erinevad veebilehitsejad toetavad on hea vaadata aadressil: http://w3schools.com/cssref/css3_browsersupport.asp

Sarnaselt HTML-dokumentidele saab ka CSS faile valideerida! Selleks on vahend näiteks aadressil: http://jigsaw.w3.org/css-validator/

CSS linkimine HTML dokumendiga

Nagu eespool mainitud, saab kasutada välist stiililehte (css fail), HTML dokumendi sisemist stiili ja reastiili. Järgnevalt vaatame, kuidas neid veebilehega seotakse.

Välise stiililehe rakendamine

Välise stiili (*External Style Sheet*) tarvitamine on kasulik, kui on tarvis kujundada tervet veebilehestikku (mitmeid erinevaid lehti).
Välise stiili kasutamiseks tuleb HTML-dokumendi päisesse (elemendi <head> sisse) lisada kindlal kujul link vajalikule CSS failile:

```
<link rel="stylesheet" type="text/css" href="stiililehe_URL">
```

Seega võiks vastav element välja näha näiteks:

```
<link rel="stylesheet" type="text/css" href="omastiil.css">
```

Kõik css failis loodud stiilid erinevatele html elementidele rakenduvad automaatselt kõigile vastavatele elementidele, erandiks vaid need, millele on määratud kindel kujundusklass! Kujundusklasside (*class*) kasutamisel peab HTML dokumendis soovitud elementide algusmärgendisse soovitud klassi kirja panema. Näiteks:

```
<span class="markeriga">Rõhutamist vajav fraas</span>
```

Selleks, et ühele html-elemendile rakendada korraga mitu defineeritud klassi, selleks tuleb nad jutumärkide vahele kirjutada! Näiteks:

```
<p class="keskel paks">
  See on lõik.
</p>
```

Selles näites on lõigule rakendatud klassid keskel ja paks!
HTML-elementidele võib näiteks hiiresündmustele reageerides erinevaid klasse (kujundusi) rakendada! Selleks lisatakse html-koodis elemendile vastavad väärtused sündmuste atribuutidele. Näiteks määrame pildielemendile erinevad klassid hiirega pildile liikumisel ja pildilt väljumisel:

```
<img src=“pilt.jpg“ alt=“ilus pilt“ onmouseover="this.className='klass1'"
onmouseout="this.className='klass2'">
```

NB! Klassi nimi on paigutatud ülakomade vahele, sest jutumärgid ümbritsevad tervet atribuudi väärtust!

Sisemise stiili rakendamine

Olukorras, kus on tarvis kujundada vaid üks konkreetne veebileht, võib kasutada sisemist stiili (*Internal Style Sheet*). Selleks lisatakse HTML-dokumendi päisesse (elemendi <head> sisse) element <style>:

```
<style type="text/css">
  veebilehe erinevate elementide kujundusreeglid
</style>
```

Kujundusklasside kasutamine toimub sisemise stiili puhul samamoodi kui välise stiililehe korralgi!

Reastiili rakendamine

Harvematel juhtudel, kui on tarvis mingit ühte, konkreetset veebilehe elementi eriliselt kujundada, kasutatakse reastiili (*Inline Style*).

Reastiili kasutamisel läheb palju CSS võimalusi raisku, sest stiil luuakse konkreetse HTML elemendi algusmärgendisse.

Reastiili kasutamiseks tuleb vastava HTML elemendi algusmärgendisse lisada argument style järgmisel kujul:

```
<elemendinimi style="css stiil">
```

Näiteks määrame ühe tekstilõigule erilise värvi:

```
<p style="color: sienna">
  See on üks lõik
</p>
```

CSS õigekiri

CSS faili võib nagu HTML dokumendigi luua mistahes tekstiredaktorit kasutades.
Nagu HTML dokumendis, või mistahes programmeerimiskeeleski, on kasulik kasutada kommentaariridu. CSS kommentaar kirjutatakse kujul:

```
/* See on kommentaar */
```

CSS abil veebilehe elementidele kujunduse kirjutamisel pannakse kirja kolm tähtsat osa: selektor (*selector*, mis on reeglina tavaline HTML-elemendi nimetus), omadus (*property*) ja selle väärtus (*value*). Kõik see pannakse kirja järgmisel kujul:

```
selector {property: value}
```

Näiteks:

```
body {color: black}
```

NB! CSS failis ei tohi kasutada html-märgendeid! See tähendab, et näiteks ei tohi kirjutada <body>!

Kui väärtus koosneb mitmest sõnast, siis tuleb see jutumärkide vahele paigutada! Näiteks:

```
p {font-family: "sans serif"}
```

Mitmete omaduste väärtuste puhul tuleb kirja panna ka mõõtühik. Sellisel juhul ei tohi väärtuse ja selle mõõtühiku vahele tühikut jätta! Näiteks:

```
p { font-size: 36pt;}
```

Korraga võib määrata väärtused ühe HTML elemendi (selektori) mitmele omadusele, selleks tuleb need lihtsalt semikoolonitega eraldada! Näiteks:

```
p {text-align:center;color:red}
```

Et stiili kirjeldust kergemini loetavaks muuta, on kasulik iga omadus eraldi reale paigutada! Näiteks:

```
p {
  text-align:center;
  font-family: arial;
  color:red;
}
```

NB! Tüüpilise veana unustatakse mõne omaduse rea lõppu semikoolon ja/või stiili lõppu loogeline sulg lisada. Sellisel juhul stiililеht ei toimi ja kujundus veebilehele ei rakendu!

Eesliited

Kuna CSS3 pole veel standardina kinnitatud, siis ei toeta kõik veebilehitsejad veel kõiki võimalusi või ei toeta neid ühte moodi. Mitmete vahendite kasutamisel tuleb üht ja sama rida kirjutada vastavate eesliidetega iga veebilehitseja jaoks eraldi!

Eesliide algab reeglina sidekriipsuga „-“ või allkriipsuga „_“ millele järgneb vastava tootja või projekti nime lühend.

- -moz- Mozilla;
- -webkit- Webkit – Safari ja Chrome;
- -ms- MS Internet Explorer, AvantBrowser;
- -icab- iCab;
- -khtml- Khtml (Konqueror);
- -o- Opera.

Üheks näiteks oleks teksti paigutamine veergudesse:

```
div.veergudes {
  column-count: 3;
  -webkit-column-count: 3;
  -moz-column-count: 3;
  -ms-column-count: 3;
  -o-column-count: 3;
}
```

Erinevad selektorid

Kõige tavalisemaks selektoriks ongi HTML elementide nimed nagu body või h1 või p. Selektorid võivad aga olla elemendi nimega (atribuudi id väärtus) seotud, pseudoelemendid, selektoreid saab omavahel kombineerida.

Selektoritega määratud stiilide hierarhia (prioriteedid)

Ühele ja samale veebilehe elemendile saab rakendada kujundust erinevaid selektoreid kasutades. Need kujundused võivad olla erinevad ning siis jääb peale see, millel on kõrgeim prioriteet ehk spetsiifilisus (*specificity*).
Selektorite prioriteetide hierarhia on järgmine:

- Elemendid ja pseudoelemendid – madalaim prioriteet;
- Klassid, pseudoklassid, atribuudi selektorid – võidavad eelmist taset;
- Elemendi identifikaatori (id) selektor – võidab eelmiseid tasemeid.

NB! Reastiil (*inline style*) ehk elemendile style-atribuudiga määratud stiil on kõrgeima prioriteediga!

NB! Hierarhiat saab ignoreerida lisades kujunduses omadustele **!important** reegli (*rule*)! See muudab konkreetse omaduse kõige prioriteetsemaks.

Näiteks on järgnev kujundus madalaima prioriteediga selektoriga kuid omadus rakendatakse igal juhul tänu !important reeglile.:

```
p {
   color: red !important;
}
```

Mistahes element

Vahel harva võib vaja minna ka võimalust luua kujundus korraga kõigile HTML elementidele, sellisel juhul on selektoriks „*“ (tärn).
Näiteks:

```
* {color: black}
```

NB! Tavaliselt viidatakse kõigile mingi elemendi sees olevatele tütarelementidele.

Elemendi identifikaatoriga (id) selektorid

Kujunduse saab luua ka kindlat identifikaatorit omavale objektile (mille algusmärgendis on id atribuut). Sellisel juhul defineeritakse kujundus järgmisel kujul:

```
#atribuudi_id_väärtus {kujundus}
```

Näiteks:

```
#pildikoht {height: 100%}
```

NB! Elemendi id atribuudi järgi määratud kujundus on kõrgema prioriteediga kui näiteks klass!

NB! Kujunduse määramisel tuleks eelistada klasside kasutamist!

NB! Ühegi HTML elemendi id atribuudi väärtust ei tohi alustada numbriga!

Sarnaselt võib luua kindlat tüüpi ja kindlat identifikaatorit omavale objektile. Näiteks kujundus tekstilõigule (html element <p>), millel nimeks "oluline":

```
p#oluline {
  text-align: center;
  color: red
}
```

Mitu selektorit korraga

Ühesugust kujundust saab määrata korraga mitmele erinevale veebilehe elemendile! Selleks tuleb soovitud HTML elemendid üheks selektoriks grupeerida ehk nad lihtsalt komadega eraldatult üles loetleda. Näiteks:

```
p, h1, h2, h3, h4, h5 {
  text-align:center;
  font-family: arial;
  color:red
}
```

Klassid

Väga sageli on tarvis sama tüüpi HTML elemendile veebilehe erinevates osades erinevat kujundust. Sellisel juhul võetakse appi klassid (*class*). Selektori nime järgi lisatakse punkt ning klassi nimi:

```
selector.class {property: value}
```

Näiteks on osa tekstilõike veebilehel joondada paremale, osa aga keskele:

```
p.parem {text-align: left}
p.keskel {text-align: center}
```

NB! Klassi nime ei tohi alustada numbriga! Selline klass ei tööta Mozilla Firefox brauseri kasutamisel!

Nende klasside (ehk kujunduse) rakendamiseks soovitud elementidele HTML dokumendis, tuleb nende elementide algusmärgendisse lisada atribuut class:

```
<p class="klassinimi">
```

Näiteks meie eelpool loodud klasside puhul:

```
<p class="parem">See lõik on paremale joondatud.</p>
<p class="keskel">See lõik on keskele joondatud.</p>
```

Korraga võib ühe elemendi kujundamiseks rakendada ka **mitut klassi**, selleks tuleb atribuudi class väärtusena kirjutada tühikuga eraldatult soovitud klasside nimed!

Näiteks:

```
<p class="parem pakskiri">See lõik on paremale joondatud ja ilmselt ka paksus kirjas.</p>
```

Võib luua ka universaalseid, kujunduses kindlaks määramata html elemendile mõeldud klasse. Sellisel juhul tuleb klassi nime ette html elemendi nimi kirja panemata jätta:

```
.center {kujundus}
```

Loomulikult võib ka mitmele klassile korraga nende ühiseid omadusi kirja panna, selleks kirjutatakse nagu mitme selektori kasutamisel soovitud klassid komadega eraldatult üksteise järele, näiteks:

```
.nali, .vahetekst {
  font-family:"Comic Sans MS", Verdana;
}
```

Kujundus vastavalt elementide sisaldumisele üksteise sees

Stiile saab adresseerida konkreetsele elemendile vastavalt tema nimele ja paiknemisele teiste elementide sees (hierarhia). Tuleb lihtsalt elemendid üksteise sisaldamise järjekorras tühikutega eraldatult üles lugeda.

Näiteks kui HTML dokumendis on elemendid üksteise sees järgmiselt:

```
<div id=”kast”>
  <div id=”vasak”>
    <img id=”pilt1_1” ... />
    <img id=”pilt1_2” ... />
  </div>
  <div id=”parem”>
    <img id=”pilt2_1” ... />
  </div>
</div>
```

Siis saab stiile kirjutada näiteks nii:

```
#kast #vasak img { css stiil}
```

mille puhul rakendatakse stiili kõikidele piltidele, mis sisalduvad elemendi „kast“ sees olevas elemendis „vasak“.
Või:

```
#kast #vasak #pilt1_1 { css stiil}
```

mille puhul või rakendatakse stiili elemendis „kast“ sisalduva elemendi „vasak“ sees olevale elemendile „pilt1_1“.
Või näiteks:

```
div img { css stiil}
```

mille puhul rakendatakse stiili kõikidele piltidele, mis on paigutatud mõne elemendi sisse.

See annab mugava ja paindliku võimaluse veebilehe erinevate osade elementidele erinevaid ja/või sarnaseid kujundusi luua.

Kujundus vastavalt elementi sisaldavale elemendile (*parent*)

Vahel on tarvis kirjutada kujundus elementidele, mis sisalduvad mingit kindlat tüüpi elemendis. Sellisel juhul pannakse selektor kirja järgmisel kujul:

```
suurelement>väikeelement
```

Näiteks võib luua kujunduse kõigile piltidele, mis on paigutatud tabeli lahtritesse (HTML elemendid <td>):

```
td>img {
  margin: 5px;
}
```

Kujundus vastavalt elementide otsesele järgnevusele

Kui on tarvis määrata kujundust elementidele, mis järgnevad mingisugusele kindlale elemendile, siis pannakse selektor kirja järgmiselt:

```
esimeneelement+järgnevelement
```

Näiteks loome kujunduse tekstilõikudele (HTML element <p>), mis järgnevad suurtele pealkirjadele (HTML element <h1>):

```
h1+p {
  font-size: 20px;
}
```

NB! Saab kirja panna ka pikema järgnevuse jada, näiteks lõigule, mis järgneb h2 pealkirjale, mis omakorda järgneb otse peakirjale h1 ning vajadusel veelgi pikemalt!

Näiteks:

```
h1+h2+p {
  font-size: 20px;
}
```

Sarnaselt saab viidata kõikidele kindlat tüüpi elementidele, millele eelneb konkreetne element. Selektor kirjutatakse siis kujul:

```
esimeneelement~järgnevelement
```

Näiteks loome kujunduse kõigile lõikudele, mis järgnevad kolmanda taseme pealkirjadele (HTML element <h3>):

```
h3~p {
  color: gray;
}
```

Atribuudi selektorid

CSS3 võimaldab kasutada selektoreid vastavalt HTML elementide atribuutidele, nende väärtustele jne. Sellisel juhul kirjutatakse selektor tervenisti kandilistesse sulgudesse:

```
[atribuut_ja_selle_väärtus_vms]
```

- Kujundust võib luua vastavalt mõne atribuudi olemasolule.

Näiteks loome kujunduse kõigile elementidele, millel on olemas atribuut „lang“:

```
[lang] {
  font-famly: verdana;
}
```

- Kujunduse saab luua vastavalt mingi atribuudi väärtusele.

Näiteks loome kujunduse kõigile neile elementidele, mille atribuudi „title“ väärtus on „selgitus“:

```
[title=selgitus] {
  font-style: italic;
}
```

Atribuudi väärtuse kasutamisel võib viidata ka mingile osale väärtusest (string)!

- [atribuut^=string] – viitab elemendile, mille atribuudi väärtus algab vastava stringiga;
- [atribuut$=string] – viitab elemendile, mille atribuudi väärtus lõppeb vastava stringiga;
- [atribuut*=string] – viitab elemendile, mille atribuudi väärtus sisaldab vastavat stringi.

NB! String tuleb selektoris paigutada jutumärkide vahele!

Näiteks loome stiili kõigile elementidele, mille atribuudi „src“ väärtus sisaldab stringi „tlu.ee“:

```
[src*="tlu.ee"] {font-family: courier;}
```

- Eraldi on olemas keeleatribuudiga seotud pseudoklass :lang.

Näiteks loome kujunduse kõigile elementidele <h2>, millel kasutatakse inglise keelt:

```
h2:lang(en) {
  font-style: italic;
}
```

Pseudoklassid ja pseudoelemendid

Pseudoklasse ja pseudoelemente kasutatakse veebilehe elementidele veelgi paindlikumalt stiilide määramiseks.

Pseudoklassid

Pseudoklasse kasutatakse veebilehe elementide erinevate olekute (*state*) defineerimiseks. Nende puhul lisatakse tavapärasele selektorile kooloni järele veel nn pseudonimi järgmisel moel:

```
selektor:pseudonimi{kujundus}
```

Kõige tuntumad pseudoelemendid on ilmselt linkidega seotud:

- a:link – külastamata link;

- a:visited – külastatud link;
- a:hover – hiirekursor on liikunud lingi peale;
- a:active – hiire vasak nupp on lingil alla vajutatud (klõpsamine).

Näiteks:

```
a:hover {
  font-weight: bold;
}
```

NB! Nendest mainitutest saab pseudonime :hover kasutada ka teistel elementidel ning sedasi luua eriefekte hiirega elementidele liikumisel.

Olemas on ka pseudoklass, mis viitab mingi lingi sihtelemendile (*target*), mis on ka hetkel aktiivne. Selleks on pseudoklass :target. Näiteks kujundame elemendi nimega „portree“, mis oli klikitud URL-i väärtuseks ja mis on aktiivne:

```
#portree:target {
  background-color: yellow;
}
```

Üks huvitav pseudoklass tähistab kõiki nimetatud elemendist erinevaid elemente – selleks on :*not* pseudoklass.
Näiteks:

```
:not(h1) {
  text-indent: 20px;
}
```

Kogu veebilehe juurelemendi (*root*) jaoks on pseudoklass :*root*. Näiteks:

```
:root {
  background:#ff0000;
}
```

Tühja elemendi jaoks on pseudoklass :*empty*. Näiteks loome kujunduse tühjadele tabeli lahtritele:

```
td:empty {
  background-color: pink;
}
```

Eesnimi	Perekonnanimi	Telefon
Mati	Karu	5555551
Kati	Karu	
Rein	Rebane	5555552
Juta	Jänes	

Joonis 2 Tabeli tühjad lahtrid on teistsuguse taustaga

Kasutaja sisendiga seotud pseudoklassid

Osa pseudoklasse on sellised, mida saab kasutada vaid nende elementide puhul, mis võimaldavad kasutajal midagi sisestada, näiteks klaviatuurilt.

- Kujundust saab luua elemendile, millel on fookus, selleks kasutatakse pseudonime :*focus*. Näiteks:

```
input:focus {
  background-color: #FFDDDD;
}
```

- Võib lähtuda sellest, kas <input> element on aktiveeritud (*enabled*), selleks on pseudonimi :*enabled*. Näiteks:

```
input:enabled {
  font-weight: bold;
}
```

- Vastupidiselt võib kasutada pseudonime :disabled, mis tähistab neid <input> elemente, mis on deaktiveeritud. Näiteks:

```
input:disabled {
  color: gray;
}
```

- Võib kujundada neid <input> elemente, mis on märkeruutudena märgitud, selleks on pseudonimi :*checked*. Näiteks:

```
input:checked {
  background-color: #DDDDDD;
}
```

Elementide järjestusega seotud pseudoklassid

Pseudoklasside abil saab kujundada ka elemente vastavalt nende asukohale neid sisaldavas elemendis (*parent*).

- Kindlat tüüpi elementi, mis on esimene tütarelement (*child*) teda sisaldavas elemendis saab kujundada pseudoklassi :*first-child* abil.

Näiteks:

```
h2:first-child {
  background-color: white;
}
```

- Kindlat tüüpi elementi, mis on viimane tütarelement (*child*) teda sisaldavas elemendis saab kujundada pseudoklassi :*last-child* abil.

Näiteks muudame lõigu, mis on viimane tütarelement, laiust:

```
p:last-child {
  width: 50%;
}
```

- Esimese kindlat tüüpi elemendi jaoks teda sisaldavas elemendis (*parent*) on pseudoklass :*first-of-type*.

Näiteks määrame mingis elemendis sisalduvale esimesele lõigule taandrea:

```
p:first-of-type {
  text-indent: 20px;
}
```

- Viimase kindlat tüüpi elemendi jaoks teda sisaldavas elemendis (*parent*) on pseudoklass :*last-of-type*.

Näiteks muudame mingis elemendis sisalduva viimase sektsiooni raamjoone alumised nurgad ümaraks:

```
section:last-of-type {
  border-bottom-left-radius: 20px;
  border-bottom-right-radius: 20px;
}
```

- Lugedes algusest saab kindlal positsioonil asuvale tütarelemendile (*child*) viidata pseudoklassiga :*nth-child*.

Näiteks loome kujunduse lõigule, mis on kolmas tütarelement:

```
p:nth-child(3) {
  font-family: arial;
}
```

- Elementidele saab viidata ka tagantpoolt lugema hakates, siis kasutatakse pseudoklassi :*nth-last-child*(n).

Näiteks loome kujunduse lõigule, mis on tagantpoolt kolmas tütarelement:

```
p:nth-last-child(3) {
  font-family: arial;
}
```

- Sarnaselt saab viidata kindlat tüüpi elementidele nii eestpoolt kui ka tagantpoolt lugedes, selleks on pseudoklassid :*nth-of-type*(n) ja :*nth-last-of-type*(n).

Näiteks loome kujunduse tagantpoolt teisele sektsioonile:

```
section:nth-last-of-type {
  border-width: 10px;
}
```

NB! Järjekorranumbritena saab kasutada ka viidet paaris või paaritutele arvudele vastavalt *even* või *odd*.

Näiteks:

```
p:nth-child(odd) {
  font-family: arial;
}
```

Ainsa tütarelemendi pseudoklassid

- Elemendile, mis on ainsaks tütarelemendiks mingis elemendis, määratakse kujundus kasutades pseudoklassi :*only-child*.

Näiteks määrame ainsaks tütarelemendiks olevale elemendile raamjoone:

```
div:only-child {
  border: 20px dashed blue;
}
```

- Elemendile, mis on ainus vastavat tüüpi tütarelement, määratakse kujundus kasutades pseudoelementi :*only-of-type*;

Pseudoelemendid

Pseudoelemendid viitavad veebilehe elemendi mingisugusele osale.

Pseudoelementide puhul on CSS3 puhul võetud kasutusele kirjaviis, kus on kaks koolonit. Eesmärgiks on rõhutada pseudoklasside ja pseudoelementide erinevus.

```
selektor::pseudoelement{kujundus}
```

NB! Ühildumaks vanemate veebilehitsejatega, töötab ka vanamoodi ühe kooloniga kirjaviis.

Esimese tekstirea ja tähemärgi pseudoelemendid

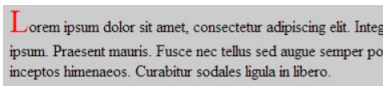
Teksti kujundamisel saab appi võtta pseudoelemendid, mis viitavad esimesele tähemärgile või esimesele tekstireale. Nendeks pseudoklassideks on: ::*first-line* ja ::*first-letter*.

Näiteks loome kujunduse kõikide lõikude esimestele tähemärkidele:

```
p::first-letter {
  font-size: xx-large;
  color: red;
}
```

Või loome kujunduse mingisuguse suure ploki esimese lõigu esimesele tähemärgile:

```
p::first-of-type:first-letter {
  font-size: xx-large;
  color: red;
}
```



Joonis 3 Lõigu esimene tähemärk on eraldi kujundatud

Pseudoelemendid lisamaks sisu elementide ette ja järele

Selleks, et lisada mingit kindlat sisu määratud elementide ette või järele on pseudoelemendid ::*before* ja ::*after*.

Sisu lisatakse omadusega *content*. Sellel omadusel on järgmised võimalikud väärtused:

- *none* – ei lisata mingit sisu;
- *normal* – (vaikeväärtus) seab selle omaduse väärtuse vaikimisi kasutatavale väärtusele „*none*“;
- *counter* – lisab sisule loenduri väärtuse (loenduri nimetus tuleb lisada sulgudesse);
- *attr*(atribuudinimi) – lisab sisule elemendi vastava atribuudi väärtuse;
- *string* – lisab sisule määratud stringi (teksti jutumärkides);
- *open-quote* – lisab sisule alustava jutumärgi;
- *close-quote* – lisab sisule lõpetava jutumärgi;
- *no-open-quote* – eemaldab alustava jutumärgi;
- *no-close-quote* – eemaldab lõpetava jutumärgi;
- *url*(aadress) – lisab sisule mingi meedia (pilt, heli- või videoklipp);
- *initial* – seab selle omaduse esialgsele väärtusele;

- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Näiteks lisame pealkirjadele <h2> ette sõna „Peatükk“ ning jutumärkide alguse ning järele jutumärkide lõpu:

```
h2::before {
  content: "Peatükk " open-quote;
}

h2::after {
  content: close-quote;
}
```

Loenduri (*counter*) loomise kohta saab lugeda peatükis „Loendur ehk automaatne nummerdamine“ leheküljel 60.

Kasutaja poolt valitud osa

Üks pseudoelement viitab veebilehe mingi elemendi kasutaja poolt valitud (*selected*) osale. Selleks on pseudoelement ::*selection*.
Näiteks:

```
p::selection {
  font-style: italic;
}
```

CSS värvid

Värvid on igasuguse kujundamise juures väga olulised ja seetõttu on neil ka CSS-is oluline roll.
Värve saab määrata väga mitmel moel:

- konstantidena – nimetustega;
- HEX väärtustena – kuueteistkümnendsüsteemi (*hexadecimal*) arvudena;
- RGB ja RGBA väärtustena – RGB värvimudeli järgi;
- HSL ja HSLA väärtustena – HSL värvimudeli järgi.

Näiteks:

	konstant	HEX	RGB	HSL
must	*black*	#000000	rgb(0,0,0)	hsl(0,0%,0%)
valge	*white*	#FFFFFF	rgb(255,255,255)	hsl(0,0%,100%)
punane	*red*	#FF0000	rgb(255,0,0)	hsl(0,100%,50%)
hall	*gray*	#808080	rgb(128,128,128)	hsl(0,0%,50%)

Värvikonstandid

Üsna tavaline on ka värvikonstantide ehk nimede kasutamine. CSS spetsifikatsioonis on kokku 147 värvi nime (17 standardvärvi ja 130 täiendavat). Standardvärvideks on: *aqua*, *black*, *blue*, *fuchsia*, *gray*, *grey*, *green*, *lime*, *maroon*, *navy*, *olive*, *purple*, *red*, *silver*, *teal*, *white* ja *yellow*.

Näiteks:

```
section {background-color: gray}
```

Värvikonstantide nimed leiab aadressil: https://www.w3schools.com/colors/colors_names.asp

Värvid HEX väärtustena

Kõige tavapärasem on CSS keeles värve kirja panna kuusteistkümmendsüsteemi (*hexa-decimal* ehk HEX), kus kõik arvud pannakse kirja numbritega 0 … 9 ja tähtedega A, B, C, D, E ja F (siin A = 10, B = 11 … F = 15) koodidena kujul #xxxxxx. Iga numbri ja/või tähepaar määrab RGB (*red, green, blue*) värvimudeli vastava värvikomponendi väärtuse.
Näiteks:

```
##FFB933
```

Need paarid saadakse RGB väärtustest kümnendsüsteemis jäägiga jagamisel. Kümnendsüsteemis väärtus jagatakse 16 –ga, täisarvukordne annab esimese „numbri“ ning jääk teise. Näiteks kui kümnendsüsteemis on väärtus 198, siis selle jagamisel 16-ga, saame jagatise täisosaks 12 ehk kuueteistkümmendsüsteemis C ning jäägiks 6, kokku C6.

Mõnele veebilehele saab niimoodi näiteks taustavärvi määrata:

```
body {background-color: #FFE799}
```

Põhjalikku infot värvide HEX koodide kohta leiab näiteks veebiaadressil: http://www.w3schools.com/css/css_colors.asp

Värvid RGB ja RGBA väärtustena

Kasutada võib ka RGB väärtuseid tavapärasemal kujul rgb(x,x,x), kus iga arv on vastava värvikomponendi väärtus kümnendkujul.
Näiteks:

```
rgb(255,135,22)
```

Kasutada saab rgba värve, kus a tähendab *alpha* väärtust ehk läbipaistvust (sarnane omadusega *opacity*)! *Alpha* väärtused on vahemikus 0 – 1.
Näiteks:

```
section {background-color: rgba(255, 0, 0, 0.2)}
```

Värvid HSL ja HSLA väärtustena

Kasutusele on võetud ka HSL värvimudel, millede puhul määratakse värv kolme komponendiga:

- *Hue* – värvitoon, väärtus vahemikus 0 – 360, mis vastab värvuse asukohale värvirattal (punane on väärtustega 0 ja 360);
- *Saturation* – küllastus, väärtus vahemikus 0 – 100%, näitab värvi küllastust (0 – värvi pole, 100% – puhas värv);

- *Lightness* – heledus, väärtus vahemikus 0 – 100%, näitab heledust (0 – maksimaalselt tume ehk must, 50% - normaalne värv, 100% - maksimaalselt hele ehk valge).

Näiteks:

```
section {background-color: hsl(120,100%,50%)}
```

Sarnaselt RGB mudeli läbipaistvusele on ka HSL puhul kasutusel uuendatud mudel alpha väärtusega – HSLA. *Alpha* väärtus on ikka kümnendmurruna vahemikus 0 – 1.
Näiteks:

```
section {background-color: hsla(120,100%,50%,0.2)}
```

Mitme CSS faili ühendamine

Üsna sageli on tarvis ühendada kujundus, mis pannakse kirja mitmes erinevas css failis. Selleks on võimalik lisada HTML-dokumenti viited mitmele CSS-failile kuid on võimalik ka importida ühe faili sisu teise! Selleks tuleks CSS-faili lisada import reegel (*rule*)!

```
@import url("teinecssfail.css")
```

NB! See käsk tuleb paigutada CSS-failis kõige esimeseks reegliks!

Lisada saab ka komadega eraldatud loendi erinevatest väljundseadmetest (*media queries*), mille jaoks vastav css-fail loodud on.

Näiteks:

```
@import url("suurekraan.css") projection, tv;
```

NB! Kui veebilehitseja ei toeta ühtki loetletud väljundseadet, siis vastavat css-faili üldse ei laetagi.

Kalkulatsioonid mõõtühikutega

Vahetevahel on kasulik erinevate omaduste väärtustena kasutada konkreetsete väärtuste asemel dünaamiliselt arvutatavaid väärtuseid. CSS3 pakub arvutusteks *calc*() funktsiooni.

Funktsiooni sulgudes pannakse kirja matemaatikaavaldis, mille tulemuseks on soovitud väärtus. Kasutada saab matemaatika põhitehteid (liitmine, lahutamine, korrutamine ja jagamine) ehk operaatoreid +, -, * ja /.

Näiteks:

```
width: calc(50% - 50px);
```

CSS vahendid

Järgnevalt kirjeldame veebilehtede põhiliste osade (elementide) olulisemaid kujundusvõimalusi CSS abil.

Veebilehe elementide taust

Mitmetel veebilehe elementidel saab määrata tausta omadusi. Terve veebilehe tausta määrab stiil, mis luuakse elemendile <body> kuid sarnaselt võib tausta määrata sektsioonidele <section> alajaotustele , tabelile ja/või tabeli osadele jne.

Taustavärv

Veebilehe taustavärv määratakse omadusega *background-color* järgmiselt:

```
background-color:värv
```

Näiteks määrame veebilehele taustavärviks helehalli:

```
body {background-color: #c0c0c0}
```

Taustapilt ja selle paigutus

Veebilehe elemendi (või terve veebilehe) taustana saab kasutada ka pilti. Üldiselt ei kasutata üht suurt, tervet veebilehe tausta katvat pilti, sest selle laadimine võtaks palju aega. Reeglina kasutatakse väikesemõõdulist pilti, mida siis mosaiigina üle terve tausta laotakse. Kasutatakse gif, jpg ja png vormingus pildifaile. Taustapildi lisamiseks kasutatakse omadust *background-image* järgmiselt:

```
background-image: url("pildifaili_URL")
```

Määrata saab ka seda kas ja kuidas kasutatavat pilti mosaiigina taustale laotakse. selleks on omadus *background-repeat*, millel on neli võimalikku väärtust:

- *repeat* – pilti laotakse mosaiigina nii horisontaalsuunal kui ka vertikaalsuunal (vaikeväärtus);
- *repeat-x* – pilti laotakse mosaiigina vaid horisontaalsuunal;
- *repeat-y* – pilti laotakse mosaiigina vaid vertikaalsuunal;
- *no-repeat* – pilti ei laota mosaiigina, näidatakse nii nagu on.

Taustapildile, mida mosaiigina ei laota saab määrata täpse asukoha veebilehitseja aknas. Selleks kasutatakse omadust *background-position*, mille väärtusteks võivad olla konstandid: *top left*, *top center*, *top right*, *center left*, *center center*, *center right*, *bottom left*, *bottom center*, *bottom right*, koordinaatide paar protsentides või koordinaatide paar pikselites.

Näiteks asetame veebilehe taustale pildi, ei lao seda mosaiigina ning asetame ta veebilehitseja aknas ülemisest vasakust nurgast 300 pikselit vasakule ja 200 pikselit allapoole:

```
body {
  background-image: url("taustapilt.gif");
  background-repeat: no-repeat;
  background-position: 300px 200px
}
```

Saab ka määrata, kas ja kuidas käitub taustapilt veebilehe kesrimisel. Kasutada on omadus *background-attachment*, millel on järgmised väärtused:

- *fixed* – pilt püsib veebilehitseja aknas ühel kohal;

- *scroll* – pilti keritakse koos elemendiga, mille taustal ta on;
- *local* – pilti keritakse koos elemendi sisuga.

Kokkuvõtlikult võib HTML elemendi <body> jaoks CSS abil kirjutada näiteks järgmise kujunduse, kus lisaks eelnevatele näidetele on taustapilt oma kohale fikseeritud:

```
body {
  background-color: #c0c0c0;
  background-image: url("taustapilt.gif");
  background-repeat: no-repeat;
  background-position: 300px 200px;
  background-attachment: fixed
}
```

Tausta stiil lühikeselt kirjutatuna

Tausta omadusi saab kirja panna ka lühidalt (*shorthand*) omadusena *background*, millele kõik erinevate omaduste väärtused tühikutega eraldatuna järjest kirja pannakse. Sellisel juhul tuleb kõik omadused kirja panna järgmises järjekorras:

- *background-color*
- *background-image*
- *background-repeat*
- *background-attachment*
- *background-position*

Kui mõne omaduse väärtust ei määrata (see puudub), siis ülejäänud peavad ikkagi seda järjekorda järgima!

Näiteks võib eespool toodud tausta stiili kirja panna järgmiselt:

```
body {background: #c0c0c0 url("taustapilt.gif") no-repeat fixed 300px 200px}
```

Taustapildi suurus

Ühe uuendusena saab võimalikuks taustapildi suuruse muutmine. Selleks on omadus *background-size*.

Taustapildi suurust saab määrata absoluutarvudega pikselites. Määrata võib ühe väärtuse (laius, kõrgus arvutatakse automaatselt ja proportsionaalselt) või siis kaks väärtust (laius, kõrgus). Näiteks:

```
#sisukast {background-size: 200px 100px}
```

Taustapildi suurust saab määrata ka protsentuaalselt elemendi suhtes, mille taustaks ta on. Määrata võib taas kas ühe väärtuse (laius, kõrgus arvutatakse automaatselt ja proportsionaalselt) või kaks väärtust (laius, kõrgus). Lisaks on võimalik kasutada konstante:

- *contain* – Taustapilt venitatakse maksimaalselt suureks nii, et ta laius ja kõrgus mahuvad mõlemad elemendi mõõtudesse.
- *cover* – taustapilt venitatakse nii suureks, et ta katab terve elemendi (võib ulatuda ka üle serva).

Näiteks:

```
#sisukast {background-size: cover}
```

Joonis 4 Taustapildi suurus protsentuaalselt: laius 100%, kõrgus 100%

Joonis 5 Taustapildi suurus: *contain*

Joonis 6 Taustapildi suurus: *cover*

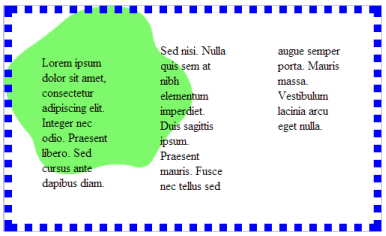
Taustapildi koordinaatide alguspunkt

Uue võimalusena saab määrata, kas taustapilt paigutatakse objekti raamjoone, polsterduse (*padding*) või sisu koordinaatidest lähtuvalt. Selleks on omadus *background-origin*, mille võimalikud väärtused on:

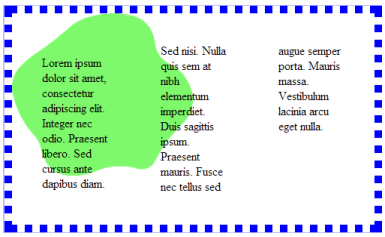
- *padding-box* – vaikeväärtus, taustapilt paigutatakse lähtudes raamjoonest (*border*);
- *border-box* – taustapilt paigutatakse lähtudes elemendi polsterdusest (*padding*);
- *content-box* – taustapilt paigutatakse lähtudes elemendi sisu piirkonnast.

Näiteks:

```
div {
   background-image:url('taustapilt.png');
   background-repeat:no-repeat;
   background-position:top left;
   background-origin:content-box;
}
```



Joonis 7 Tastapildi asukoht raamjoone suhtes (*border-box*)



Joonis 8 Tastapildi asukoht polsterduse suhtes (*padding-box*)

Joonis 9 Tastapildi asukoht sisu suhtes (*content-box*)

Tausta kaetav ala

Uus on ka võimalus määrata, millise ala elemendist taust katab. Omadus *background-clip* lubab määrata, kas kaetakse vaid sisu piirkond või kogu raamjoone sisse jääv ala. Võimalikud väärtused on:

- *padding-box* – vaikeväärtus, taust katab kogu elemendi alla jääva piirkonna, kaasaarvatud polsterduse ala ja raamjoon;

- *border-box* – taust katab elemendi sisu ja polsterduse (*padding*) alla jääva piirkonna;
- *content-box* – taust katab vaid elemendi sisu piirkonna.

Näiteks:

```
.piiratudtaust {background-clip:content-box}
```

Joonis 10 Taust katab kogu elemendi piirkonna (*border-box*)

Joonis 11 Taust katab sisu ja polsterduse ala, joone alla ei ulatu

Joonis 12 Taust katab vaid elemendi sisu ala

Mitu taustapilti

Uus oodatud võimalus on mitme taustapildi kasutamise võimalus! Selleks tuleb taustapildid komadega eraldatult üles lugeda! Sarnaselt tuleks komadega eraldatult üles lugeda kõik teised tausta omadused (samas järjekorras)!

Näiteks:

```
body {
  background-image: url("images/top_right.png"), url("images/bottom_right.png");
  background-size: 30%; /*see on mõlema taustapildi ühine omadus*/
  background-position: top right, bottom right;
  background-repeat: no-repeat; /*see on mõlema taustapildi ühine omadus*/
}
```

Astmiktäide (*gradient*)

CSS3 võimaldab veebi elementide taustal kasutada ka astmiktäidet (*gradient*).

Astmiktäide on realiseeritud taustapildina ning määrataksegi omaduse *background-image* väärtusena. Kasutada saab põhimõtteliselt kolme erinevat võimalust:

- lineaarne astmiktäide (*linear gradient*);
- radiaalne astmiktäide (*radial gradient*);
- kooniline astmiktäide (*conic gradient*).

Astmiktäide defineeritakse vähemalt kahe värvi abil (*color stop*).

Kõigil kolmel astmiktäite tüübil on olemas ka korduv variant (*repeating*).

Infot leiab näiteks aadressilt: https://www.w3schools.com/css/css3_gradients.asp

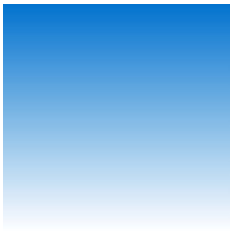
Astmiktäite loomine on mugavam mõnda veebipõhist tööriista kasutades, näiteks: https://cssgradient.io/).

Lineaarne astmiktäide

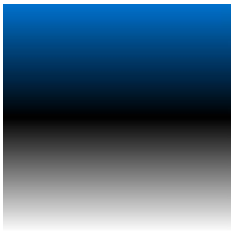
Lineaarse astmiktäite defineerimiseks kasutatakse funktsiooni *linear-gradient()*, lihtsaimal kujul on vaja määrata kaks värvi (*color stop*). Sellisel juhul tekib sujuv üleminek esimeselt värvilt teisele, suunaga alla (*to bottom*).

Näiteks:

```
background-image: linear-gradient(#0072CE, #FFFFFF);
```



Joonis 13 Lihtsaim lineaarne astmiktäide



Joonis 14 Kolme värviga lihtne astmikäide

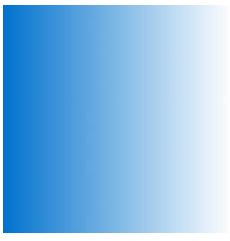
Lineaarse astmiktäite suund

Lineaarse astmiktäite suuna määramiseks tuleb enne värve kirja panna ka suuna väärtus. Kasutada saab eeldefineeritud väärtusi:

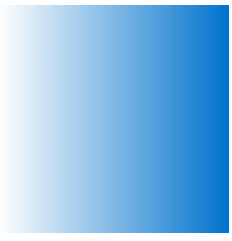
- *to bottom* – ülevalt alla (vaikeväärtus);
- *to topp* – alt üles;
- *to right* – vasakult paremale;
- *to left* – paremalt vasakule;
- *to top* right, *to top left*, *to bottom right*, *to bottom left* – diagonaalselt üles paremale, üles vasakule, alla paremale, alla vasakule.

Näiteks lineaarne astmiktäide suunaga diagonaalselt paremale üles [Joonis 17]:

```
background-image: linear-gradient(to top right, #0072CE, #FFFFFF);
```



Joonis 15 Astmiktäide suunaga paremale



Joonis 16 Astmiktäide suunaga vasakule

Joonis 17 Astmiktäide diagonaalselt üles paremale

Kui etteantud suunad ei paku soovitud tulemust, võib astmiktäite suuna anda ette ka täpse nurgana kasutades nurgakraade. Seejuurest tuleb arvestada, et nurk 0° on suund alt üles (*to top*), 90° on suund vasakult paremale (*to right*) jne.

Näiteks:

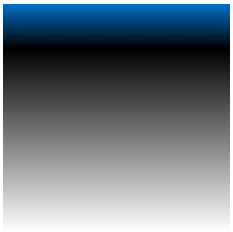
```
background-image: linear-gradient(20deg, #0072CE, #FFFFFF);
```

Lineaarse astmiktäite värvipositsioonide fikseerimine

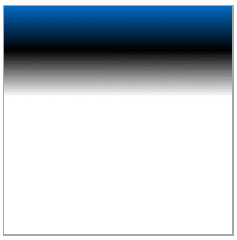
Astmiktäite värvidele saab positsiooni ka fikseerida ning selle kaudu määrata, kust alates ja kui palju mingi värv paistab. Selleks pannakse soovitud värvile juurde ka positsioon protsentidena (eraldajaks tühik).

Näiteks kolme värviga astmiktäide, kus keskmisel värvil on fikseeritud positsioon:

```
background-image: linear-gradient(#0072CE, #000000 20%, #FFFFFF);
```



Joonis 18 Kolme värviga astmiktäide, mustal fikseeritud positsioon 20%



Joonis 19 Kolme värviga astmiktäide, mustal fikseeritud positisoon 20% ja valgel 40%

Näiteks keskmisel värvil kaks fikseeritud positsiooni [Joonis 20]:

```
background-image: linear-gradient(#0072CE, #000000 20%, #000000 80%, #FFFFFF)
```

Määrates fikseeritud positsioonid kõigile värvidele, seejuures teiste vahele jäävatele kaks, saame selged triibud [Joonis 21].

Näiteks kõigil värvidel fikseeritud positsioon, keskmisel kaks:

```
background-image: linear-gradient(#0072CE 33%, #000000 33%, #000000 66%, #FFFFFF
66%);
```

Joonis 20 Kolme värviga astmiktäide, keskmisel kaks fikseeritud positsiooni

Joonis 21 Kolme värviga astmiktäide, kõigil fikseeritud positsioon, keskmisel kaks positsiooni

Korduv lineaarne astmiktäide

Korduva astmiktäite abil saab luua nn triibulisi taustasid. Korduva astmiktäite defineerimiseks kasutatakse funktsiooni *repeating-linear-gradient()*, triipude laiuse määramiseks tuleb vähemalt viimasele värvile fikseerida positsioon [Joonis 22].

Näiteks kolme värviga korduv lineaarne astmiktäide, mille ühe korduse laius on 15%:

```
background-image: repeating-linear-gradient(#0072CE, #000000, #FFFFFF 15%);
```

Täpselt nagu tavalise lineaarse astmiktäite korral, saab fikseeritud positsioone määrata kõikidele värvidele ning selgepiiriliste triipude saavutamiseks isegi kaks [Joonis 23].

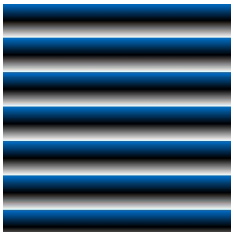
Näiteks kolme värviga korduv astmiktäide, mille kõigil värvidel on 2 fikseeritud positsiooni:

```
background-image: repeating-linear-gradient(#0072CE 0%, #0072CE 5%, #000000 5%,
#000000 10%, #FFFFFF 10%, #FFFFFF 15%);
```

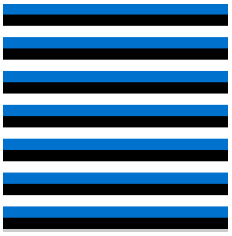
Ka saab analoogselt tavalise lineaarse astmiktäitega saab määrata ka astmiktäite suunda, seda isegi täpse nurga abil [Joonis 24].

Näiteks kolme värviga korduv astmiktäide 20° nurgaga, mille kõigil värvidel on kaks fikseeritud positsiooni:

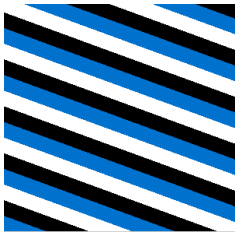
```
background-image: repeating-linear-gradient(20deg, #0072CE 0%, #0072CE 5%, #000000 5%,
#000000 10%, #FFFFFF 10%, #FFFFFF 15%);
```



Joonis 22 Kolme värviga korduv lineaarne astmiktäide, määratud valge positsiooniga 15%



Joonis 23 Kolme värviga korduv astmiktäide, kõigil värvidel 2 fikseeritud positsiooniga



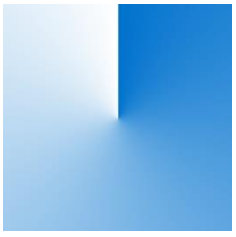
Joonis 24 Kolme värviga korduv astmiktäide 20° nurga all, kõigil värvidel 2 fikseeritud positsiooni

Kooniline astmiktäide

Koonilise astmiktäite (*conic gradient*) korral toimub värviüleminek ümber määratud keskpunkti.
Kooniline astmiktäite defineerimiseks kasutatakse funktsiooni *conic-gradient()*, millel lihtsaimal kujul määratakse kaks värvi (*color stop*), astmiktäide toimib ümber elemendi keskpunkti ning alustab ülalt keskelt (nurk 0°).

Näiteks lihtsaim kahevärviline kooniline astmiktäide:

```
background-image: conic-gradient(#0072CE, #FFFFFF);
```



Joonis 25 Kooniline astmiktäide kahe värviga

Koonilise astmiktäite nurk ja keskpunkt

Määrata saab nurka, millest kooniline astmiktäide algab. Selleks tuleb värvide ette lisada omadus *from* ning soovitud nurk kraadides [Joonis 26].

Näiteks kooniline astmiktäide, mis alustab nurgalt 45°:

```
background-image: conic-gradient(from 45deg, #0072CE, #FFFFFF);
```

Ka saab määrata koonilise astmiktäite keskpunkti, ümber mille värviüleminek toimub. Selleks lisatakse värvide ette omadus *at* ning horisontaal- ja vertikaalsuuna väärtused protsentides [Joonis 27].

Näiteks kooniline astmiktäide, millel on määratud keskpunkt:

```
background-image: conic-gradient(at 75% 25%, #0072CE, #FFFFFF);
```

Nurka ja keskpunkti saab ka koos määrata. Sellisel juhul tuleb need omavahel eraldada tühikuga [Joonis 28]!

Näiteks kooniline astmiktäide, millele on määratud nii alustamise nurk kui ka keskpunkt:

Joonis 26 Kooniline astmiktäide, mis alustab nurga 45° all

Joonis 27 Kooniline atmiktäide, mille keskpunkt on asukohas 75% 25%

Joonis 28 Kooniline astmiktäide, millel on alustamise nurk 45 ja keskpunkt asukoal 75% 25%

Koonilise astmiktäite värvipositsioonide fikseerimine

Koonilise astmiktäite värvidel saab määrata ka kindlaid positsioone nurgakraadides. See mõjutab nende värvide ulatust.

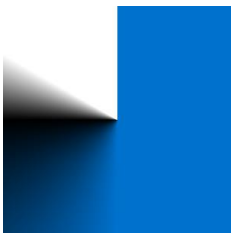
Näiteks kolme värviga kooniline astmiktäide, millel on kõigil värvidel fikseeritud positsioonid [Joonis 29]:

```
background-image: conic-gradient(#0072CE 180deg, #000000 270deg, #FFFFFF 300deg);
```

Igale värvile saab fikseerida kaks positsiooni ning sedasi saavutada selgepiirilised värvid [Joonis 30].

Näiteks:

```
background-image: conic-gradient(#0072CE 0deg, #0072CE 120deg, #000000 120deg,
#000000 240deg, #FFFFFF 240deg, #FFFFFF 360deg);
```



Joonis 29 Kolme värviga kooniline astmiktäide, mille värvdel on fikseeritud positsioonid vastavalt sinisel 180°, mustal 270° ja vagel 300°

Joonis 30 Kolme värviga kooniline astmiktäide, kõigil värvidel kaks fikseeritud positsiooni

Loomulikult saab fikseeritud värvipositsioonidega koonilise astmiktäite puhul määrata ka alustamise nurka ja/või keskpunkti (omavahel tühikuga eraldatult).

Näiteks kooniline astmiktäide kolme värviga, milledel kõigil on 2 fikseeritud positsiooni, mida alustatakse nurga 45° all keskpunktiga asukohal 25%, 25%:

```
background-image: conic-gradient(from 45deg at 25% 25%, #0072CE 0deg, #0072CE 120deg, #000000 120deg, #000000 240deg, #FFFFFF 240deg, #FFFFFF 360deg);
```

Korduv kooniline astmiktäide

Korduva koonilise astmiktäite defineerimiseks kasutatakse funktsiooni *repeating-conic-gradient()*, triipude laiuse määramiseks tuleb vähemalt viimasele värvile fikseerida positsioon [Joonis 31].

Näiteks korduv kooniline astmiktäide, mille triipude laiuse määrab kolmandale (valgele) värvile fikseeritud positsioon 60%:

```
background-image: repeating-conic-gradient(#0072CE, #000000, #FFFFFF 60deg);
```

Sarnaselt tavalisele koonilisele astmiktäitele saab määrata ka alustamise nurka ja/või keskpunkti (omavahel tühikuga eraldatult) [Joonis 32].

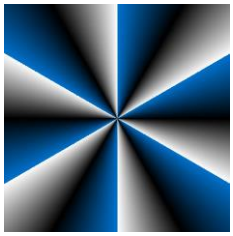
Näiteks kolme värviga korduv astmiktäide, mille kolmandal värvil (valgel) fikseeritud positsioon 60% keskpunktiga asukohal 75%, 25%:

```
background-image: repeating-conic-gradient(at 75% 25%, #0072CE, #000000, #FFFFFF 60deg);
```

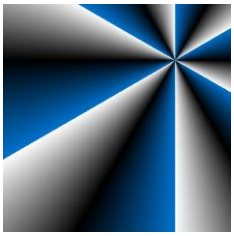
Sarnaselt tavapärasele koonilisele astmiktäitele saab igale värvile fikseerida ühe või kaks positsiooni. Kui kõigil värvidel on kaks fikseeritud positsiooni, siis saadakse selgepiirilised värvid [Joonis 33].

Näiteks korduv kooviline astmiktäide, mille kõigil värvidel on kaks fikseeritud positsiooni:

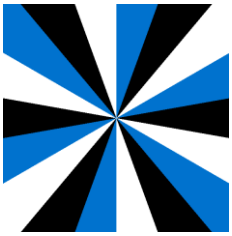
```
background-image: repeating-conic-gradient(#0072CE 0deg, #0072CE 20deg, #000000 20deg, #000000 40deg, #FFFFFF 40deg, #FFFFFF 60deg);
```



Joonis 31 Kolme värviga korduv kooniline astmiktäide, valge värvi fikseeritud positsioon 60%



Joonis 32 Kolme värviga korduv kooniline astmiktäide keskkohaga asukohas 75%, 25%



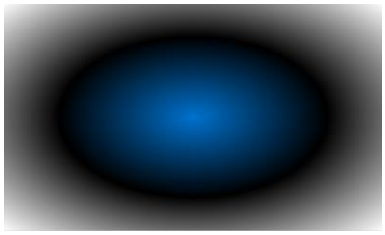
Joonis 33 Kolme värviga korduv astmiktäide, kõigil värvidel kaks fikseeritud positsiooni

Radiaalne astmiktäide

Radiaalse astmiktäite (*radial gradient*) puhul toimub värviüleminek radiaalselt ümber keskpunkti. Radiaalse astmiktäite defineerimiseks kasutatakse funktsiooni *radial-gradient()*. Lisaks värvidele saab määrata veel keskpunkti asukohta, astmiktäite suurust ning kuju (mis omavahel eraldatakse tühikutega).

Näiteks lihtsaim kolme värviga radiaalne astmiktäide:

```
background-image: radial-gradient(#0072CE, #000000, #FFFFFF);
```



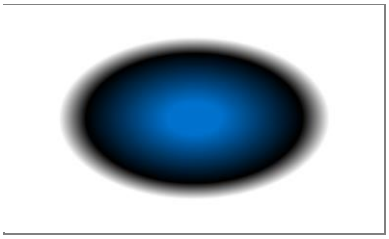
Joonis 34 Lihtsaim kolme värviga radiaalne astmiktäide

Radiaalse astmiktäite värvipositsioonide fikseerimine

Vaikimisi jaotatakse määratud värvid ühtlaselt kuid neil on võimalik ka kindlaid positsioone protsentides fikseerida.

Näiteks kolme värviga radiaalne astmiktäide, mille kõigil värvidel on fikseeritud positsioon:

```
background-image: radial-gradient(#0072CE 10%, #000000 40%, #FFFFFF 50%);
```



Joonis 35 Kolme värviga radiaalne astmiktäide, sinisel fikseeritud positsioon 10%, sinisel 40% ja valgel 50%

Fikseerides igale värvile kaks positsiooni, saame selgepiirilised rõngad.

Radiaalse astmiktäite kuju

Vaikimisi on radiaalne astmiktäide ellipsi (*ellipse*) kujuga. Kasutada saab aga veel ka ringi (*circle*) kuju.

Näiteks kolme värviga ringi kujuga radiaalne astmiktäide:

```
background-image: radial-gradient(circle, #0072CE 10%, #000000 40%, #FFFFFF 50%);
```

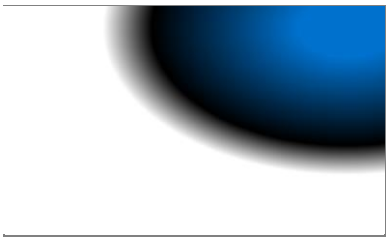
Joonis 36 Kolme värviga ringikujuline radiaalne astmiktäide, värvidel fikseeritud positsioonid

Radiaalse astmiktäite keskpunkt

Radiaalse astmiktäite jaoks saab määrata ka keskpunkti, ümber mille värviüleminek toimub. Selleks lisatakse värvide ette omadus *at* ning horisontaal- ja vertikaalsuuna väärtused protsentides.

Näiteks kolme värviga radiaalne astmiktäide, millel on määratud keskpunkt:

```
background-image: radial-gradient(at 90% 10%, #0072CE 10%, #000000 40%, #FFFFFF 50%);
```



Joonis 37 Kolme värviga radiaalne astmiktäide määratud keskpunktiga 90%, 10%, värvidel fikseeritud positsioonid

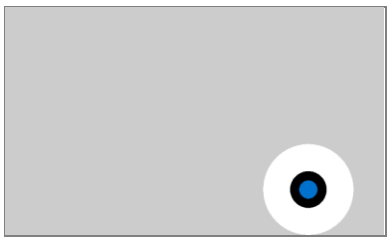
Radiaalse astmiktäite suurus

Radiaalse astmiktäite puhul saab määrata ka selle suurust. Kasutada saab nelja väärtus:

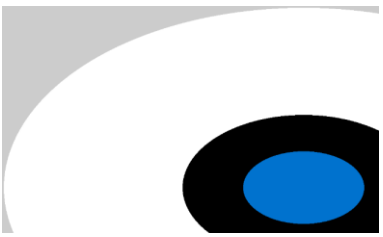
- *closest-side* – radiaalne astmiktäide ulatub selle keskpunktile lähima servani;
- *farthest-side* – radiaalne astmiktäide ulatub selle keskpunktile kaugeima servani;
- *closest-corner* – radiaalne astmiktäide ulatub selle keskpunktile lähima nurgani;
- *farthest-corner* – radiaalne astmiktäide ulatub selle keskpunktile kaugeima nurgani.

Näiteks radiaalne ringi kujuline astmiktäide, millel on fikseeritud keskpunkt ja suurus määratud keskpunktile lähima servani [Joonis 38]:

```
background-image: radial-gradient(circle closest-side at 80% 80%, #0072CE 0%, #0072CE
20%, #000000 20%, #000000 40%, #FFFFFF 40%, #FFFFFF 99%, #CCCCCC 99%);
```



Joonis 38 Radiaalne ringi kujuga astmiktäide fikseeritud keskpunktiga, suurus lähima servani



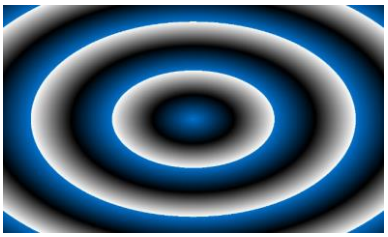
Joonis 39 Radiaalne ringi kujuga astmiktäide fikseeritud keskpunktiga, suurus kaugeima servani

Korduv radiaalne astmiktäide

Korduva radiaalse astmiktäite defineerimiseks kasutatakse funktsiooni *repeating-radial-gradient()*, rõngaste laiuse määramiseks tuleb vähemalt viimasele värvile fikseerida positsioon.

Näiteks kolme värviga korduv radiaalne astmiktäide mille rõngaste laiuse määrab valgele fikseeritud positsioon:

```
background-image: repeating-radial-gradient(#0072CE, #000000, #FFFFFF 30%);
```



Joonis 40 Kolme värviga korduv radiaalne astmiktäide, valgel fikseeritud positsioon 30%

Ka korduval radiaalsel astmiktäitel saab määrata kuju ja keskpunkti. Samuti võib igale värvile määrata kaks fikseeritud positsiooni, mille puhul on tulemuseks selgepiirilised rõngad.

Teksti kujundusvahendid

Teksti kujundamisvõtted on tekstitöötlusest tuttavad. Veebilehel tuleb nendega ettevaatlikult ümber käia, sest ekraanilt lugemine on raskem kui paberilt.
Teksti kujundamisvahendeid saab määrata kõigile teksti sisaldavatele veebilehe osadele (pealkirjad, lõigud, loendid, lingid jne).

Teksti värv

Vaikimisi kasutatav värv kõikide tekstiobjektide jaoks on must. Soovitud värvi määramiseks on omadus *color*! Veebilehele üldiselt saab teksti värvi määrata elemendile <body> loodud stiilis. Näiteks:

```
body{ color:white}
```

Selliselt määratud värvi kasutatakse kõigil tekstielementidel (pealkirjad, lõigud jms), millele pole eraldi värvi määratud.

Omaduse *color* võib lisada iga teksti sisaldava elemendi stiilile. Näiteks määrame pealkirja värvuseks oranži:

```
h1 {color: #FF8040}
```

Tekstiobjektidele saab määrata ka oma taustavärvi. Selleks tuleb soovitud elemendi kujundusele lisada tausta omadus *background-color*. Näiteks loome eraldi kujundusklassi lõikudele (element <p>), anname sellele nimeks "markeriga" ja määrame taustavärviks kollase:

```
p.markeriga {background-color: #FFFF00}
```

Lõigu kujundus

Tekstilõikude jaoks on veebis põhimõtteliselt samad vahendid nagu tavapärases tekstitöötluses.

Lõigu joondus

Üsna tavapärane on lõigu joonduse määramine, selleks on kasutatakse omadust *text-align*, millel võimalikeks väärtusteks on *left* (vasak, vaikimisi kasutatav), *right* (parem), *center* (keskel) ja *justify* (rööpselt). Näiteks joondame suure pealkirja (element <h1>) keskele:

```
h1 {text-align: center}
```

Taandrida

Taandrea loomiseks kasutatakse omadust *text-indent*, mille väärtus võib olla fikseeritud arv pikslites – px, punktides – pt, sentimeetrites – cm või ka em. Kasutada saab ka väärtust % selle elemendi laiusest, mille sisse käsitletav tekstiobjekt kuulub. Näiteks määrame tekstilõikudele (html element <p>) taandrea 20 pikselit:

```
p {text-indent: 20px}
```

NB! Terve lõigu taande jaoks tuleb kasutada omadust *margin*, mis on kirjeldatud peatükis “Veerised”!

Reasamm

Reasamm ehk rea kõrgus määratakse omadusega *line-height*, mille väärtusteks võivad olla: *normal* (vaikeväärtus), number (normaalkõrguse kordaja), fikseeritud suurus (mõõtühikutega px, pt või cm) või protsent teksti suurusest (*font size*).

Näiteks määrame tekstilõikudele reakõrguseks 125% teksti suurusest:

```
p {line-height: 125%}
```

Font ja sellega seonduv

Font tähistab tavaliselt kas konkreetset tähemärkide kujundust (font) või üldisemalt kõike, mida saab tähemärkidega seoses määrata (lisaks suurus, stiil jms).

Tähemärkide kujunduse võib kirja panna lühendatud kujul, loetledes kõigi võimalike omaduste väärtused järgmises järjekorras: *font-style font-variant font-weight font-size/line-height font-family*. Kohustuslikud siinjuures on *font-size* ja *font-family*, ülejäänud võib ära jätta, sellisel juhul kasutatakse nende vaikeväärtuseid.

Näiteks:

```
p {font:italic bold 14px/30px Arial, sans-serif}
```

Font

Nagu tekstitöötluseski, saab määrata kasutatavat fonti. Selleks kasutatakse omadust *font-family*, mille väärtusena tuleb kirja panna soovitud font. Fonte võib komadega eraldatult loetleda mitu! Sellisel juhul püüab veebilehitseja kõigepealt kasutusele võtta esimese, kui seda fonti süsteem ei toeta (seda pole arvutisse paigaldatud vms), siis järgmise jne. Näiteks määrame suurtes pealkirjades (element <h1>) kasutatava fondi:

```
h1 {font-family: trendy, joan, "comic sans ms"}
```

Järgnevad 12 fonti on turvalised, ehk nad on installeeritud nii PC kui ka MAC arvutitele: arial; ***arial black***; comic sans ms, courier, courier new, georgia, helvetica, **impact**, palatino, times new roman, trebuchet ms, verdana.

Paraku on alati võimalus, et ükski loetletud fontidest pole kättesaadav, sellisel juhul tuleks asendustoiminguteks pakkuda konkreetsete fontide nimede järel veel soovitud fondiperekonna ja lõpuks ka üldise perekonna nimetusi.

Fontide üldisteks ehk geneerilisteks (*generic*) perekondadeks on: *serif*, *sans-serif*, *cursive* ja *monospace*.

Näiteks:

```
h1 {font-family: Arial, "Trebuchet MS", Helvetica, sans-serif}
```

CSS3 võimaldab kasutada mistahes fonti ehk veebidisainer ei pea enam piirduma vaid 12 turvalise fondiga!

Kasutusel on uus *@font-face* reegel (*rule*), mis võimaldab määrata kasutajasõbraliku nime ja fondi faili URL-i. Disainer saab seega laadida fondi faili oma veebiserverisse ja see laetakse kliendile koos veebilehega.

Näiteks:

```
@font-face {
  font-family: omanimifondile;
  src: url(URL/FontFileName.nimelaiend) format(„tüüp“);
}
```

Ja seejärel võib teistes stiilides (teiste selektorite jaoks) seda kasutada nagu fonte tavaliselt kasutatakse, näiteks:

```
h1 {font-family: omanimifondile}
```

Fondi tüüpe on mitu:

- truetype – *True Type Fonts* (ttf), seda toetavad pea kõik brauserid;
- otf – *OpenType Fonts*, seda toetavad pea kõik brauserid;
- eot – *Embedded OpenType*, seda toetavad vaid Microsoft'i brauserid;
- woff – *Web Open Font Format*, sisuliselt ttf või otf font lisatud metaandmete ja kompressiooniga;
- woff2 – *Web Open Font Format version 2*, sama, mis woff kuid parema kompressiooniga, MS Edge ja Safari seda ei toeta;
- svg, svgz – *scalable-vector-graphic*, MS Edge ja Mozilla Firefox seda ei toeta.

Kui tarvis kasutada erinevate atribuutidega (*bold*, *italic*) kirju ja Teil on vastava kujundusega spetsiaalne font olemas, siis tuleb lisada veel teinegi samasugune @font-face reegel, milles kirjas sama fondi vastava atribuudiga versiooni URL ja lisaks vastav kujunduselement (*font-style*, *font-weight*, *font-stretch*)

NB! Internet Explorer 9 toetab vaid eot fonte! Kui tarvis, saab ttf fonte teisendada eot fontideks, selleks on olemas ka veebipõhised vahendid, näiteks: http://www.kirsle.net/wizards/ttf2eot.cgi

Ühildumaks kõigi veebilehitsejatega, võib oma fondi kirjeldada selliselt, et erinevate veebilehitsejate jaoks lisatakse komaga eraldatult eraldi URL-id.

Näiteks:

```
@font-face {
  font-family: omanimifondile;
  src: url("fondinimi.eot"), /*IE jaoks*/
  url("fondinimi.ttf") format("truetype");/*teised veebilehitsejad*/
}
```

Fondi allikana võib püüda kasutada ka lokaalset (kliendi arvutis paiknevat) fonti, mis säästaks vajadusest fonti serverist alla laadida. Selleks määratakse fondi allikas url-i asemel *local*. Sellisel juhul tuleks aga kindluse mõttes komaga eraldatult lisada viide ka samale fondile veebis.
Näiteks:

```
@font-face {
  font-family: minuEriFont;
  src: local(Gentium Bold),
  url(GentiumBold.woff);
}
```

NB! Fontide kasutamisel tuleb jälgida, et ei mindaks pahuksisse autoriõigustega! Veebis võib leida mitmeid leheküülgi, kus on saadaval ka tasuta fonte, mida saab allalaadida või siis nende lehel viidata.

Näiteks leiab fonte aadressil: http://webfonts.info/wiki/index.php?title=Fonts_available_for_%40font-face_embedding aga ka: http://www.theleagueofmoveabletype.com/, http://www.fonts.com/web-fonts ja http://www.fontsquirrel.com/

Kirja suurus

Kirja suuruse määramiseks kasutatakse omadust *font-size*, mille väärtus pannakse enamasti kirja pikselites (px) või isegi sentimeetrites (cm). Mõne veebilehitsejaga võib aga aeg-ajalt probleeme esineda, seetõttu kasutatakse sageli mõõtühikut em (elemendi kirja suurus, vaikimisi 1em = 16px), millel on ka W3C soovitus. Näiteks määrame suure pealkirja suuruseks 40 pikselit ja tekstile 1,25em (1,25 X 16px = 20px):

```
h1 {font-size: 40px}
p {font-size: 1.25em}
```

NB! Murdarvude kasutamisel tuleb komakoha eraldajana kirjutada punkt!

Suhteline suuruse mõõtühik on veel ka rem – veeblehe juurelemendi teksti suurus. Ning ka ch – sümboli „0“ laius.

Teksti suuruse võib määrata ka protsentuaalselt antud teksti sisaldava elemendi (*parent*) teksti suuruse suhtes. Näiteks:

```
.largertext {font-size: 125%}
```

Omaduse *font-size* väärtustena võib kasutada ka konstante:

- *xx-small* – extra extra väike;
- *x-small* – extra väike;
- *small* – väike;
- *medium* – keskmine;
- *large* – suur;
- *x-large* – extra suur;
- *xx-large* – extra extra suur;
- *smaller* – väiksem, kui antud teksti sisaldaval elemendil ette nähtud;
- *larger* – suurem, kui antud teksti sisaldaval elemendil ette nähtud

Näiteks:

```
.largertext {font-size: larger}
```

Kaldkiri

Loomulikult saab määrata, kas tegemist on kaldkirjaga või mitte. Selleks kasutatakse omadust *font-style*, millel on tavapärasteks väärtusteks on:

- *normal* (vaikeväärtus);
- *italic* (kaldkiri);
- *oblique* (ka kaldkiri).

Näiteks määrame kasutame teise taseme pealkirjadel (element <h2>) kaldkirja:

```
h2 {font-style: italic}
```

Paks kiri

Kirja paksus määratakse omadusega *font-weight*, mille väärtusteks on *normal*, *bold*, *bolder*, *lighter*, 100, 200, 300, 400 (sama mis *normal*), 500, 600, 700 (sama mis *bold*), 800, 900.

Näiteks määrame paksu kirjaga lõikude kujunduse (loome eraldi klassi elemendile <p>):

```
p.paks {font-weight: 800}
```

Teksti joonimine, allajoonimine, läbikriipsutamine

Võimalik on ka teksti allajoonida või läbi kriipsutada. Selleks teksti dekoreeritakse, milleks on omadus *text-decoration* ning mille võimalikud väärtused on:

- *none* (vaikeväärtus);
- *underline* - allajoonimine*;*
- *overline* – joon teksti kohal*;*
- *line-through* - läbikriipsutamine.

Näiteks määrame suurtele pealkirjadele allajoonimise:

```
h1 {text-decoration: underline}
```

Neid omadusi saab ka koos kasutada, selleks tuleb soovitud väärtused tühikutega eraldatult kirja panna.

Näiteks:

```
.imelikultJoonitud {text-decoration: underline line-through}
```

Väga imelikult joonitud tekst.

Joonis 41 Erinevad teksti joonimise võimalused

CSS3 puhul on uue omadusena loodud *text-decoration-line*, mis ilmselt tulevikus asendab seni kasutatud *text-decoration* omaduse.

Teksti joonimise värv

Täiesti uue omadusena pakutakse võimalust ka teksti joonimise värvi määrata. 2017. aasta juuni alguse seisuga on omadus toetatud Mozilla Firefox ja Google Chrome brauserites.

Omaduseks *text-decoration-color* ja selle väärtuseks on soovitud värv.
Näiteks:

```
span {text-decoration-color: #DD0000}
```

Väga imelikult joonitud tekst.

Joonis 42 Värviliselt joonitud tekst

Teksti joonimise stiil

Sarnaselt joonimise värviga pakub CSS3 ka joonimise võimalust. 2017. aasta juuni alguse seisuga on omadus toetatud Mozilla Firefox ja Google Chrome brauserites.

Omaduseks *text-decoration-style*, millel on järgmised võimalikud väärtused:

- *solid* – ühekordne pidevjoon (vaikeväärtus);
- *double* – kahekordne joon;
- *dotted* – punktiirjoon;
- *dashed* – kriipsjoon;
- *wavy* – laineline joon;
- *initial* – seab selle omaduse vaikeväärtusele;
- *inherit* – omadus päritakse vanem-elemendilt.

Näiteks:

```
.laineline {text-decoration-style: wavy}
```

Joonis 43 Teksti joonimise stiilid

Suur- ja väiketähed

Siinjuures on mõeldud võimalusi kasutada vaid väiketähti (*lowercase*), vaid suurtähti (*uppercase*) või sõnade alguses suurtähti (*capitalize*)
Tavapäraselt kasutatakse omadust *text-transform*, mille vaikeväärtuseks on *none* – teksti näidatakse nagu see kirjutati.

Näiteks:

```
p.suurtahed {text-transform: uppercase}
p.vaiketahed {text-transform: lowercase}
p.sonasuuretahega {text-transform: capitalize}
```

Suur- ja väiketähtedega on seotud ka omadus *font-variant* mis võimaldab lisaks vaikeväärtusele *normal* veel trükitähtede (kapiteelkiri ehk *small caps*) kasutamist.

```
p.trykitahed {font-variant: small-caps}
```

Tähe- ja sõnade vahe

Sarnaselt tekstitöötlusprogrammidele saab ka veebilehel tähtede omavahelist vahet muuta ning seeläbi näiteks hõrendatud teksti luua. Kasutusel on omadus *letter spacing* millel on võimalikud väärtused *normal*, *inherit* (mille korral kasutatakse antud tekstiobjekti sisaldava elemendi tähevahe) või väärtus pikselites või sentimeetrites (lisab tavapärasele tähevahele ruumi).
Näiteks:

```
p.horendatud {letter-spacing: 3px;}
```

NB! Lubatud on ka negatiivsed väärtused!

NB! Väärtus *inherit* ei ole toetatud kõigi IE versioonide poolt!

Võimalik on määrata ka sõnade vahel kasutatavat ruumi, selleks on kasutusel omadus *word-spacing*. Kasutada saab väärtuseid: *normal* (vaikeväärtus), *inherit* (vastav omadus päritakse tekstiobjekti sisaldavalt elemendilt) ja arvväärtus, mis lisab sõnade vahele ruumi ja mille mõõtühikutena võib olla px, pt, cm, em.

```
p.suuremsonavahe {word-spacing: 5px;}
```

NB! Arvväärtuste puhul saab kasutada ka negatiivseid väärtuseid.

Teksti „murdmine“

Microsofti ettepanekuna on CSS3-le lisatud ka teksti murdmine, mis võimaldab sõnu vajadusel ridade vahel poolitada! Selleks on kaks omadust *word-break* ja *word-wrap*.

Esimene neist, *word-break*, poolitab sõnu mistahes kahe tähe vahel ja tal on võimalikud väärtused:

- *normal* – sõnu poolitatakse tavapäraste reeglite järgi (vaikeväärtus);
- *break-all* – teksti rida võib murda mistahes kahe tekstimärgi vahel;
- *keep-all* – murdmine mistahes kahe tähe vahel on keelatud;
- *initial* – seab omaduse vaikeväärtusele;
- *inherit* – omaduse väärtus päritakse vanem-elemendilt.

Näiteks:

```
p {word-break: break-all}
```

Teine neist, *word-wrap*, poolitab pikki sõnu ja tal on järgmised omadused:

- *normal* – sõnu poolitatakse vaid lubatud kohtadel (vaikeväärtus);
- *break-word* – lubab poolitada sõnu, mida ei tohiks (*unbreakable words*);
- *initial* – seab omaduse vaikeväärtusele;
- *inherit* – omaduse väärtus päritakse vanem-elemendilt.

Teksti suund

Tavapärane teksti lugemissuund on vasakult paremale, kuid vajadusel saab teksti ka vastupidises suunas kirjutada. Selleks kasutatakse omadust *direction*, millel on kaks võimalikku väärtust: ltr (*left to right*) ja rtl (*right to left*).

```
div.pahupidi {direction: rtl}
```

Tühja ruumi haldus

CSS stiili abil saab määrata ka seda, kuidas käitutakse veebilehe autori poolt lisatud tühikute, reavahetuste (sisestusklahvi (*enter*) vajutused), tabulaatoriklahvi vajutuste jms „tühja ruumiga“ (*white space*).
Kasutada saab omadust *white-space*, millel on järgmised võimalikud väärtused:

- *normal* – kogu „tühi ruum“ (tühikute jada jms) pannakse kokku üheks tühikuks. Reavahetus (*text wrap*) toimub vastavalt vajadusele. See on vaikeväärtus.
- *nowrap* – kogu „tühi ruum“ (tühikute jada jms) pannakse kokku üheks tühikuks. Automaatset reavahetust ei toimu kunagi. Reavahetus toimub vaid HTML elemendi
 kasutamisel.
- *pre* – kogu „tühi ruum“ säilitatakse nii nagu see on. Reavahetus toimub seal, kus autor on selle määranud (sisestusklahviga (*enter*)). See väärtus vastab HTML elemendile <pre>!
- *pre-line* – kogu „tühi ruum“ (tühikute jada jms) pannakse kokku üheks tühikuks. Reavahetus toimub automaatselt vastavalt vajadusele ning ka kohtades, kus autor on määranud (sisestusklahviga (*enter*)).
- *pre-wrap* – kogu „tühi ruum“ säilitatakse nii nagu see on. Reavahetus toimub automaatselt vastavalt vajadusele ning ka kohtades, kus autor on määranud (sisestusklahviga (*enter*)).
- *inherit* – objekt pärib vastava omaduse teda sisaldavalt elemendilt (*parent*).

Näiteks:

```
p.naguoli {white-space: pre}
```

Loendid

Loendite (*lists*, html elemendid <ol> ja <ul>) jaoks kehtivad kõik eespool kirjeldatud teksti kujundusvahendid kuid lisaks saab neil määrata numbrite/täppide stiili. Selleks kasutatakse omadust *list-style-type*.
Selle omaduse ühised väärtused nii nummerdatud- kui täpploenditele on:

- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*));

Nummerdatud loend

Nummerdatud loendi puhul (element <ol>) saab omaduse *list-style-type* väärtustena kasutada järgmiseid väärtuseid:

- *decimal* – tavalised kümnendsüsteemi araabia numbrid, see on ka vaikeväärtus;
- *decimal-leading-zero* – araabia numbritele lisatakse polsterdusena ette 0;
- *lower-roman* – rooma numbrid väiketähtedega*;*
- *upper-roman* – rooma numbrid suurtähtedega (tavaline);
- *lower-alpha* –tähestiku väiketähed;
- *upper-alpha* –tähestiku suurtähed;
- *lower-latin* – ladina tähestiku väiketähed;
- *upper-latin* – ladina tähestiku suurtähed;
- *lower-greek*, – kreeka tähestiku väiketähed;
- *armenian* – traditsioonilised armeenia numbrid;
- *georgian* – traditsiooniline gruusia numeratsioon: an, ban, gan …;
- *cjk-ideographic* – ideograafilised numbrid;

- *hebrew* – traditsioonilised heebrea numbrid;
- *hiragana* – traditsioonilised Hiragana numbrid;
- *hiragana-iroha* – traditsioonilised Hiragana iroha numbrid;
- *katakana* – traditsioonilised Katakana numbrid;
- *katakana-iroha* – traditsioonilised Katakana iroha numbrid.

Näiteks:

```
ol {list-style-type: upper-roman}
```

Täpploend

Täpploendi puhul (element <ul>) saab omaduse *list-style-type* väärtustena kasutada väärtuseid:

- *disc* – täppideks on seest täidetud ringid, see on ka vaikeväärtus;
- *circle* – täppideks on seest tühjad ringid;
- *square* – täppideks on ruudud;
- *none* – täpid puuduvad.

Näiteks:

```
ul {list-style-type: square}
```

Täpploendi puhul saab täppidena kasutada ka pildifaili. Selleks kasutatakse omadust *list-style-image*, mille väärtuseks on soovitud pildifaili URL.

Näiteks kasutame täpploendis tavaliste täppide asemel pildifaili "leht.png":

```
ul {list-style-image: url("leht.png")}
```

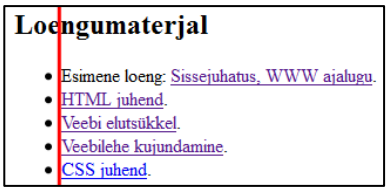
Joonis 44 Pilt järjestamata loendi täpina

Numbri/täpi paigutus

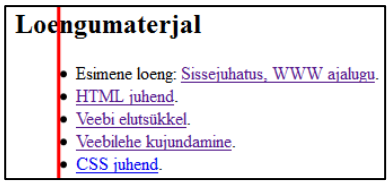
Loendite puhul saab määrata ka seda, kas loendielemendi (*list item* ehk html element <li>) marker (täpp või number) paigutatakse väljapoole sisu piirkonda (*content flow*) ehk ettepoole sisu vasakut serva või piirkonna sisse.
Selleks on kasutusel omadus *list-style-position*, mille võimalikud väärtused on:

- *outside* – täpp/number asub väljaspool sisu piirkonda, see on ka vaikeväärtus;
- *inside* – täpp/number asub sisu piirkonnas;
- *inherit* – omaduse väärtus päritakse loendit sisaldavalt elemendilt;
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:



Joonis 45 Loendielemendi marker väljaspool sisu piirkonda (*outside*)



Joonis 46 Loendielemendi marker sisu piirkonnas (inside)

Loendi stiil lühidalt

Loendite välimuse saab seada ka ainult ühe omadusega *list-style*. Selle väärtusena pannakse üksteise järel tühikutega eraldatult kirja *list-style-type*, *list-style-position* ja *list-style-image* väärused (selles järjestuses).

Näiteks:

```
.minuloend {list-style: circle inside url("leht.png")}
```

NB! Kui mõni kolmest väärtusest on kirja panemata siis kasutatakse vastava omaduse vaikeväärtust!

Tsitaadid

Ka tsitaatide (quotes) jaoks pakub css oma vahendeid, millega määratakse html elemendil <q> kasutatavad jutumärgid. Seejuures saab määrata erinevaid märke tsitaatidele, mis on omakorda tsitaadi sees.

Kasutatakse omadust *quotes*, mille väärtustena saab kasutada:

- *none* – keelab content omadusega (:*before* ja :*after* pseudoelementides) *open-quote* ja *close-quote* abil jutumärkide tekitamise;
- kasutatavad märgid (paarikaupa);
- *inherit* – omaduse väärtus päritakse loendit sisaldavalt elemendilt;
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
q {quotes: "\0022" "\0022"}
```

See näide tekitab "tavalised püstised jutumärgid".

```
q {quotes: "\2018 "\2019" "\0022" "\0022"}
```

See näide paneb tsitaadile ‘ühekordsed jutumärgid’ ning selle sees olevale tsitaadile omakorda eelmises näites nähtud "püstised jutumärgid".

Järgmine tabel sisaldab kõiki tsitaatidel kasutatavaid märke.

Märk	inglisekeelne nimetus	olemi number/kood
"	*double quote*	\0022
'	*single quote*	\0027
‹	*single, left angle quote*	\2039

Märk	inglisekeelne nimetus	olemi number/kood
›	*single, right angle quote*	\203A
«	*double, left angle quote*	\00AB
»	*double, right angle quote*	\00BB
‘	*left quote* (*single high*-6)	\2018
’	*right quote* (*single high*-9)	\2019
“	*left quote* (*double high*-6)	\201C
”	*right quote* (*double high*-9)	\201D
„	*double quote* (*double low*-9)	\201E

Tekstiefektid

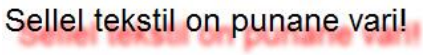
CSS3 pakub lisaks tavapärastele teksti vormindamisvahenditele ka efekte.

Teksti vari

Nüüd pole enam tekstile varju lisamiseks vaja kasutada fototöötluse abi ning teksti veebilehele pildina lisada! Kasutada on omadus text-shadow! Määrata tuleb kaks kohustuslikku parameetrit: horisontaalne nihe (*horizontal offset* või *h-shadow*), vertikaalne nihe (*vertical offset* või *v-shadow*). Lisaks saab määrata kaks valikulist parameetrit: hägustamine (*blur*) ja varju värv (*color*).

Näiteks:

```
h2 {text-shadow: 10px 10px 5px #ff0000}
```



Joonis 47 Varjuga tekst

NB! Määrates horisontaalse ja vertikaalse nihke väärtusteks 0px, saame sisuliselt kuma (*glow*) efekti!

Tekstile võib lisada mitu varju, selleks tuleb need lihtsalt komadega eraldatult järjest kirja panna!

Näiteks:

```
h2 {
  text-shadow: 10px 10px 5px #ff0000, 10px -10px 5px #ffbb00, -10px 10px 5px #ffff00;
}
```

Väga häid näiteid teksti varjuga loodud efektidest leiab näiteks aadressil: http://www.midwinter-dg.com/permalink-7-great-css-based-text-effects-using-the-text-shadow-property_2011-03-03.html

Tekst „üle serva“

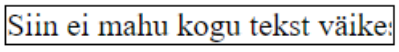
Sageli ei mahu tekst etteantud pinnale ära. Sellisel juhul võib nüüd anda visuaalse vihje, et tekst jätkub!

Selleks kasutatakse omadust *text-overflow*, millel on võimalikud väärtused:

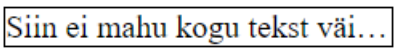
- *clip* – viimane sõna jääb poolikuks (vaikeväärtus),
- *ellipsis* – teksti jätkumise tunnuseks traditsioonilised 3 punkti;
- *string* – näidatakse etteantud teksti;
- *initial* – seab omaduse vaikeväärtusele;
- *inherit* – omaduse väärtus päritakse vanem-elemendilt.

Näiteks:

```
div {text-overflow: ellipsis}
```



Joonis 48 Üle serva jääv tekst lõigatakse ära (*clip*)



Joonis 49 Üle serva jäjäävale tekstile viitavad kolm punkti (ellipsis)

Tekst mitmes veerus

CSS3 lisab võimaluse teksti ajakirjanduslikus stiilis veergudesse (*columns*) paigutada!
Seda saab teha kolme erinevat omadust kasutades:

- *column-width* – määratakse minimaalne veeru laiuse väärtus pikselites, protsentides, em-ides jms;
  Kui pole võimalik mahutada vähemalt kahte veergu, siis loobutakse veergude kasutamisest.
- *column-count*, – määratakse soovitud veergude arv, väärtuseks positiivne täisarv.

Näiteks:

```
.newspaper {
  column-count: 3;
}
```

- *columns* – kombineerib kahte eelnevat määrates minimaalse laiuse veergudele ning maksimaalse veergude arvu.

Näiteks:

```
.newspaper {
  columns: 200px 3;
}
```

Veergude vahekaugus

Veergude vahekaugust saab määrata omadusega *column-gap*.
Selle omaduse väärtusteks võivad olla:

- *normal* (vaikeväärtus, tavaliselt 1em)
- positiivne arv pikselites, em-ides.

Joon veergude vahel

Veergude vahele saab lisada ka eraldusjoont, mida saab määrata mitme omaduse abil

Eraldusjoone paksuse määrab omadus *column-rule-width*, mille väärtuseks on positiivne arv tavapärastes mõõtühikutes.

Eraldusjoone stiil määratakse omadusega *column-rule-style*, mille võimalikud väärtused on samad, mis üldiselt raamjoontel:

- *none* – joon puudub, vaikeväärtus;
- *hidden* – raamjoont pole, enamasti sama, mis *none* kuid erinevusi on tabelite puhul;
- *solid* – pidevjoon;
- *dotted* – punktiirjoon;
- *dashed* – kriipsjoon;
- *double* – topeltjoon;
- *groove*, *ridge*, *inset* ja *outset*. – erinevad 3D jooned;
- *initial* – seab omaduse vaikeväärtusele;
- *inherit* – omaduse väärtus päritakse vanem-elemendilt

Eraldusjoone värvi saab määrata omadusega *column-rule-color*.

NB! Kui värvi pole määratud, siis kasutatakse eraldusjoonel teksti värvi.

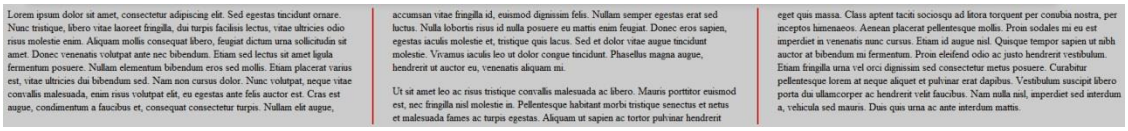
Näiteks:

```
.newsletterdeco {
   column-rule-width: 2px;
   column-rule-style: solid;
   column-rule-color: #AA0000;
}
```

Veergude vahelise eraldusjoone saab määrata ka ühe omadusega, mis kombineerib eelnevat kolme, selleks on *column-rule*, mille väärtusena pannakse kirja tühikutega eraldatult joone paksus, stiil ja värvus.

Näiteks:

```
div {
   width: 900px;
   column-count: 3;
   column-gap: 50px;
    column-rule: 2px solid red;
}
```



Joonis 50 Kolme veergu paigutatud tekst

Teksti jagunemine veergude vahel

Kui elemendile, mille sees on tekst mitmesse veergu jaotatud, on määratud ka kõrgus, siis võib juhtuda, et tekst täidab vaid esimese(d) veeru(d). Teksti jagunemist veergudesse saab reguleerida omadusega *column-fill*, millel on järgmised võimalikud väärtused:

- balance – tekst jaotatakse veergude vahel enam-vähem ühtlaselt (vaikeväärtus);

- auto – veerud täidetakse tekstiga terve kõrguse ulatuses ning osa veergusid võivad tühjaks jääda.

Näiteks:

```
.newsletter {
  column-fill: auto;
}
```

Teksti laotus üle mitme veeru

Jaotades teksti mitmesse veergu võib tekkida vajadus näiteks pealkirju laotada üle mitme veeru. Selleks on omadus column-span, mis tuleb määrata elemendile, mis jaotub üle mtme veeru.
Võimalikud väärtused on:

- *none* – element paigutatakse ühte veergu (vaikeväärtus);
- *all* – element laotatakse üle kõikide veergude;
- *initial*;
- *inherit*.

Näiteks:

```
.newspaper {
  columns: 200px 3;
}
h3 {
  column-span: all;
}
```

Lingid

Linkide tekst on vaikimisi allajoonitud, külastamata lingid on sinised, külastatud lillakad. Selline kujundus ei sobi väga paljude veebilehtede kujundusega.

Linkide kujunduse määramiseks tuleb stiil kirjutada HTML elemendile <a>. Kuna vajame erinevaid kujundusi samatüübilisele elemendile (külastamata, külastatud jne), siis on selektoriteks nn pseudoelemendid:

- a:link – külastamata link;
- a:visited – külastatud link;
- a:hover – hiirekursor on liikunud lingi peale;
- a:active – hiire vasak nupp on lingil alla vajutatud (klõpsamine).

Linkide kujundamiseks saab kasutada kõiki teksti kujundamiseks mõeldud omadusi (font, suurus, värv, taustavärv jms). Sellisel moel saab lingid muuta atraktiivsemaks, huvitavamaks, luua midagi animatsioonide sarnast.

Näiteks loome linkidele kujunduse, kus külastamata lingi värvus on must, külastatud lingil tumehall, aktiivne link on valge ja paksu kirjaga ning kui hiire kursoriga lingile liigutakse on link paksu tekstiga valgel taustal:

```
a:link {color:black; background: none; font-weight: normal}
```

```
a:visited {color: #808080; background: none; font-weight: normal}
a:hover {color:black; background-color: white; font-weight: bold}
a:active {color:white; background: none; font-weight: bold}
```

NB! Linkide kujundamiseks saab ka klasse luua!

Raamjooned

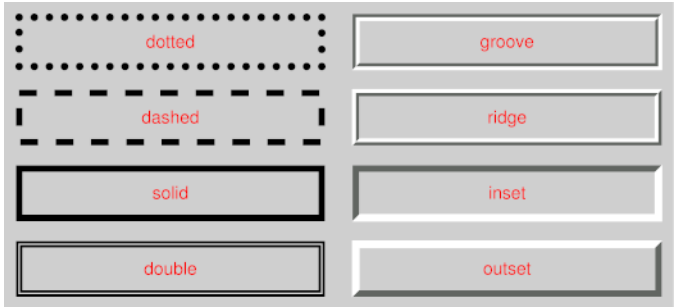
Raamjooned (*border*) on kasutusel peamiselt tabelitel kuid sageli ka piltidel, alajaotustel ning isegi tavalistel tekstilõikudel.
Määrata saab raamjoone stiili, paksust ja värvi.

NB! Paksuse ja värvi omadused ei tööta ilma, et eelnevalt poleks määratud stiil!

Raamjoone stiil

Raamjoone stiili määramiseks kasutatakse omadust *border-style*, millel on hulk võimalikke väärtuseid:

- *none* – raamjoont pole, vaikeväärtus;
- *hidden* – raamjoont pole, enamasti sama, mis *none* kuid erinevusi on tabelite puhul;
- *solid* – pidevjoon;
- *dotted* – punktiirjoon;
- *dashed* – kriipsjoon;
- *double* – topeltjoon;
- *groove*, *ridge*, *inset* ja *outset*. – erinevad 3D jooned;
- *initial* – seab omaduse vaikeväärtusele;
- *inherit* – omaduse väärtus päritakse vanem-elemendilt.



Joonis 51 Erinevad raamjoone stiilid

Näiteks määrame tabeli raamjoone stiiliks punktiirjoone:

```
table {border-style: dotted}
```

Raamjoone paksus

Raamjoone paksuse määramiseks kasutatakse omadust *border-width*, millel on kolm eeldefineeritud väärtust: *thin*, *medium* (vaikeväärtus) ja *thick*. Lisaks saab kasutada täpset väärtust pikselites.

Näiteks määrame tabeli joonepaksuseks 3 pikselit:

```
table {border-width: 3px}
```

Raamjoone värv

Raamjoone värvi määramiseks on omadus *border-color*, millele võib väärtusena kirja panna värvi nime, HEX või rgb väärtuse.
Näiteks määrame tabeli raamjoone värviks halli:

```
table {border-color: #c0c0c0}
```

Raamjooned elemendi erinevatel külgedel

Kõiki kolme kirjeldatud atribuuti saab määrata ka eraldi ülemisele, alumisele, vasakule ja paremale küljele, selleks tuleb kasutada omadusi, mille nimes vastav külg kirja pandud:

- Joone stiil:
  - *border-top-style*;
  - *border-bottom-style*;
  - *border-left-style*;
  - *border-right-style*.
- Joone paksus:
  - *border-top-width*;
  - *border-bottom-width*;
  - *border-left-width*;
  - *border-right-width*.
- Joone värv:
  - *border-top-color*;
  - *border-bottom-color*;
  - *border-left-color*;
  - *border-right-color*.

Näiteks määrame tabeli ülemise raamjoone stiiliks punktiiri ja alumisele kahekordse joone, vasaku joone paksuseks 5 pikselit, parema joone värviks punase:

```
table {
  border-top-style: dotted;
  border-bottom-style: double;
  border-left-width: 5px;
  border-right-color: #ff0000
}
```

Raamjoone paksuse ja värvi määramisel võib ühe korraga tühikutega eraldatult kirja panna kuni neli väärtust. Sellisel juhul määratakse neid omadusi järgnevalt:

- 1 väärtus – kõigile korraga;
- 2 väärtust – esimene ülemisele/alumisele, teine vasakule/paremale;
- 3 väärtust – esimene ülemisele, teine vasakule/paremale, kolmas alumisele;
- 4 väärtust – esimene ülemisele, teine paremale, kolmas alumisele ja neljas vasakule.

Raamjoone lühendatud kirjeldus

Eespool kirjeldatud atribuute saab määrata korraga kirjutades tühikutega eraldatult joone paksuse, stiili ja värvi väärtused. Kõigile raamjoontele näiteks:

```
table {border: medium double rgb(250,0,255)}
```

Eraldi ülemisele, alumisele, vasakule, paremale raamjoonele näiteks:

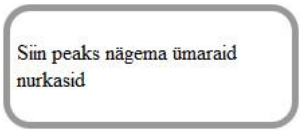
```
table {
  border-top: medium solid #ff0000;
  border-bottom: medium solid #ff0000;
  border-left: medium solid #ff0000;
  border-right: medium solid #ff0000
}
```

Ümardatud nurgad

Üks populaarsemaid uuendusi CSS3 juures on ümardatud nurgad. Üheks populaarsuse põhjuseks ilmselt asjaolu, et see on realiseeritud pea kõigi veebilehitsejate juures!

Ümarate nurkade saamiseks tuleb määrata omadus border-radius. Näiteks:

```
div {border-radius: 20px}
```



Joonis 52 Raamjoone ümarad nurgad

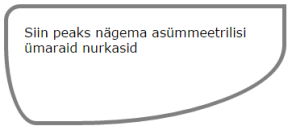
On võimalik määrata ümarus igale nurgale eraldi! Selleks saab kasutada omadusi:

- *border-bottom-left-radius*
- *border-bottom-right-radius*
- *border-top-left-radius*
- *border-top-right-radius*

Neile võib määrata väärtuse nii protsentides kui ka pikselites ning nad aktsepteerivad ka kahte väärtust, sellisel juhul on esimene horisontaalsuunaline (x) ja teine vertikaalsuunaline (y).

Näiteks:

```
.asym {
border: thick solid gray;
   border-bottom-left-radius: 40px 100px;
   border-bottom-right-radius:260px 100px;
   border-top-left-radius: 20px;
}
```



Joonis 53 Asümmeetriliste ümarustega raamjoon

NB! Määrates nurkade ümaruseks 50% saame ovaalse või lausa ringi kujulise raamjoone (ehk terve veebilehe elemendi)!

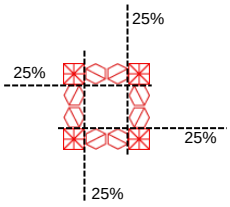
Raamjoon pildist

CSS3 võimaldab raamjoont luua ka pildi baasil! Selleks kasutatakse järgmiseid omadusi: *border-image-source*, *border-image-slice*, *border-image-width*, *border-image-outset* ja *border-image-repeat*!

Omadus *border-image source* väärtuseks on pildi URL ja see määrab kasutatava pildi. Näiteks üks võimalik rida:

```
border-image-source: url("pildid/pildiraam.png");
```

Pildi puhul tuleb määrata lõige (*slice*), ehk kui palju pildi servadest jäetakse ülemise, parema, alumise ja vasaku raamjoone jaoks. Vajadusel pannakse kirja neli erinevat väärtust alustades ülemisest ja liikudes edasi päripäeva. Väärtused määratakse protsentides või pikselites. Koos servadega saavad defineeritud ka nurgad!



Joonis 54 Pildi lõikumine raamjooneks

Pildi lõige määratakse omadusega *border-image-slice*.

Näiteks üks võimalik rida:

```
border-image-slice:20 20 20 20;
```

NB! Pikslites väljendatud väärtuste korral mõõtühikut (px) kirja ei panda!

NB! Kui osa või kõik väärtused on ühesugused, siis võib kirja panna ka näiteks ainult kaks või ühe väärtuse!

Vaikimisi on tekitatav raamjoon peenike – ei vasta kasutatava pildiosa mõõtmetele. Pildist loodava raamjoone paksuse määrab omadus *border-image-width*, millel on järgmised võimalikud väärtused:

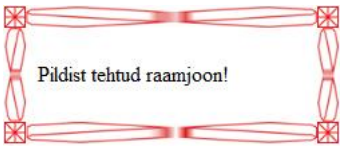
- väärtus pikslites – loodava raamjoone paksus pikslites, võib määrata 1, 2 3 või 4 väärtust – erinevatele servadele;
- kordaja – eelnevalt määratud raamjoone paksuse kordaja (vaikeväärtus on 1), võib määrata 1, 2 3 või 4 väärtust – erinevatele servadele;
- väärtus protsentides – loodava raamjoone paksus elemendi suhtes, mida ta ümbritseb, võib määrata 1, 2 3 või 4 väärtust – erinevatele servadele;
- *auto* – määrab raamjoone paksuse vastavalt lõike laiusele;
- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Näiteks üks võimalik rida:

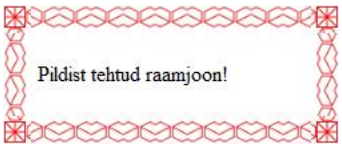
```
border-image-width: auto;
```

Samuti tuleb määrata venitus (*stretch*), mis määrab, kuidas raamjooned nurkade vahel pildiosaga täidetakse. Kasutatakse omadust *border-image-repeat*. Võimalikud väärtused on (ja neid võib määrata iga serva jaoks eraldi):

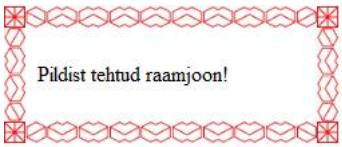
- *stretch* – pildiosa venitatakse üle terve serva (vaikeväärtus).
- *repeat* – pildiosa korratakse ja ülejääv serv lõigatakse maha;
- *round* – pildiosa korratakse ja suurus sobitatakse nii, et mahuks täpselt täisarv kordi;
- *space* – pildiosa korratakse, et terve serv saaks täidetud, kui täpselt ei jagu, lisatakse korduste vahele tühja ruumi;
- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*));



Joonis 55 Pildist raam, venitus: *stretch*



Joonis 56 Pildist raam, venitus: *repeat*



Joonis 57 Pildist raam, venitus: *round*

Näiteks üks võimalik rida rida:

```
border-image-repeat: round;
```

Määrata saab ka seda, kui palju pildist loodud raam laieneb üle border-box piiride. Selleks on omadus border-image-outset, millel on võimalikud väärtused:

- väärtus pikslites – loodava raamjoone paksus pikslites, võib määrata 1, 2 3 või 4 väärtust – erinevatele servadele;
- kordaja – eelnevalt määratud raamjoone paksuse kordaja (vaikeväärtus on 1), võib määrata 1, 2 3 või 4 väärtust – erinevatele servadele;
- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Lühidalt võib pildist raami panna kirja omadusega *border-image*, mille järel pannakse kirja eelpool vaadeldud omaduste väärtused järgmises järjestuses: *border-image-source*, *border-image-slice*, *border-image-width*, *border-image-outset* ja *border-image-repeat*! Omaduste puhul, mille väärtuseid kirja ei panda, kasutatakse nende vaikeväärtuseid.

Näiteks:

```
.raamiga {border-image: url("pildid/pildiraam.png") 20 auto round}
```

Kuna ainult pildist raamjoont luues jääb loodud raamjoon suuremas osas raamitava objekti piiridesse (fotode jms puhul isegi kujutise taha), siis on mõistlik enne pildist raami lisamist määrata ka tavaline raamjoon, mille laius ühtib pildist loodava raami paksusega. Nii on tagatud raamjoone olemasolu ka nendel juhtudel kui pildist loodavat ei toetata (mõned veebilehitsejad) ning ka see, et raamjoon tõesti objekti ümbritseb.

Näiteks:

```
.raamiga {
   border:20px solid #DD0000;
   border-image: url("pildid/pildiraam.png") 20 auto round;
}
```

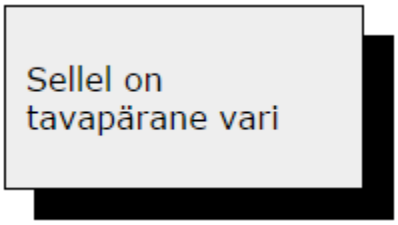
Vari

Üks oodatud uuendus CSS3-l on elementidele varju lisamise võimalus! Seejuures saab kasutada isegi mitut varju.
Varju lisamiseks kasutatakse omadust *box-shadow*. Kindlasti tuleb määrata kaks parameetrit järgnevas järjestuses:

- horisontaalne nihe (*horizontal offset*) – varju horisontaalne asukoht objekti suhtes, väärtus pikslites, kasutada saab ka negatiivseid väärtuseid (nihutab varju vasakule);
- vertikaalne nihe (*vertical offset*) – varju vertikaalne asukoht objekti suhtes, väärtus pikslites, kasutada saab ka negatiivseid väärtuseid (nihutab varju üles).

Näiteks lisame elementidele varju, mis on elemendist 15 pikslit paremale ja 15 pikslit allapoole nihutatud:

```
div {box-shadow: 15px 15px}
```



Joonis 58 Tavaline vari

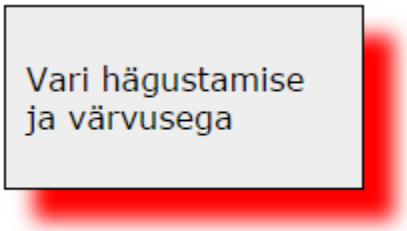
Lisaks saab määrata valikulisi parameetreid järgnevas järjestuses:

- hägustamise ulatus (*blur distance*) – väärtus pikslites, kui palju varju serva hägustatakse;
- ulatus (*spread*) – väärtus pikslites, määrab varju suuruse;

- värv – varju värv, vaikeväärtuseks on must.

Näiteks lisame varju, millel lisaks horisontaalsele ja vertikaalsele nihkele on määratud hägustamise ulatus 15 pikslit ning punane värv, varju ulatus on jäetud määramata:

```
div {box-shadow: 15px 15px 15px #F00}
```



Joonis 59 Hägustatud servaga punane vari

Võimalik on varju ka objektile sissepoole pöörata, selleks tuleb kõige viimaseks parameetriks lisada:

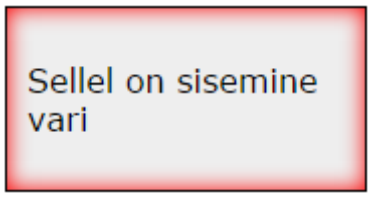
- *inset* (sisse panema) – muudab välise varju sisemiseks varjuks.

Loomulikult saab omadusel box-shadow kasutada ka standardseid väärtuseid:

- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Näiteks:

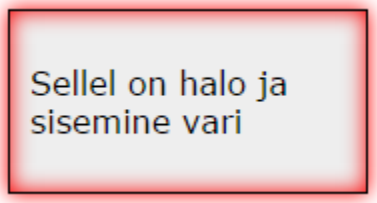
```
div {box-shadow: 0px 0px 15px #F00 inset}
```



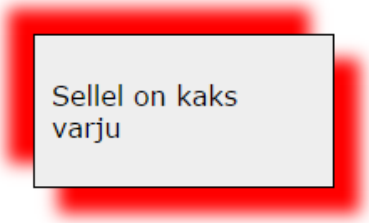
Joonis 60 Seesmine vari

Ühele elemendile saab lisada kaks varju, selleks tuleb need komaga eraldatult kirja panna. Näiteks lisame ühele elemendile nii välise varju (nihke väärtuseks 0px, siis on tegemist justkui haloga) ning seesmise varju:

```
div {box-shadow: 0px 0px 15px #F00, 0px 0px 15px #F00 inset}
```



Joonis 61 Element kahe varjuga (seesmine ja väline halo)



Joonis 62 Element kahe varjuga

Kontuur

Lisatud on võimalus plokielementidele kontuurjoon (*outline*) lisada. Kontuurjoon lisatakse väljapoole elemendi enda piiri (ja raamjoont (*border*)), eesmärgiks elemendi silmatorkavaks muutmine.
Määratakse kolm omadust:

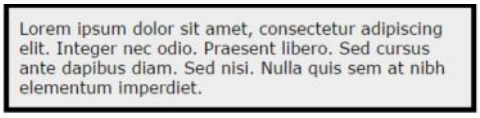
- *outline-color* – kontuurjoone värv;
- *outline-style* – kontuurjoone stiil (samad, mis raamjoone (*border*) puhul);
- *outline-width – kontuurjoone paksus.*

Kontuurjoone kirjeldus võib olla ka lühendatud,siis kirjutatakse see kujul:

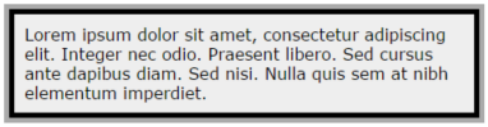
```
outline: värv stiil paksus
```

Näiteks:

```
div {outline:#DD0000 dotted medium}
```



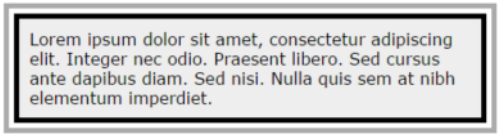
Joonis 63 Element ilma kontuurjooneta



Joonis 64 Element halli kontuurjoonega

Määrata saab ka– kontuurjoone kaugus elemendist, selleks on omadus *outline-offset*.
Näiteks:

```
div.example {
  border: 2px solid black;
  outline: 1px solid blue;
  outline-offset: 10px;
}
```



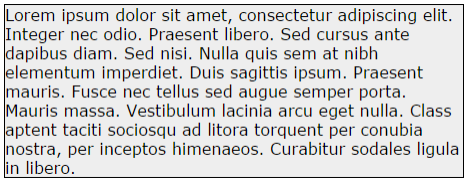
Joonis 65 Element kontuurjoonega, mis on eemale nihutatud

Polsterdus

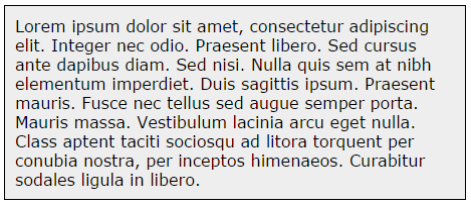
CSS võimaldab määrata ka polsterdust (*padding*) ehk ruumi veebilehe elemendi serva ja tema sisu vahel. Mõõtühikutena saab kasutada piksleid -- px, punkte – pt, sentimeetreid – cm ja protsente elementi sisaldava elemendi laiuse suhtes.

Näiteks:

```
section {padding: 20px}
```



Joonis 66 Elemendil polsterdus 0px



Joonis 67 Elemendil polsterdus 10px

Polsterdust saab määrata eraldi üleval, all, vasakul ja paremal, sellisel juhul lisatakse omaduse *padding* nimele liide *–top*, *-bottom*, *-left* või *-right*.

Näiteks:

```
table {
   padding-top: 5px;
   padding-bottom: 5px;
   padding-left: 10px;
   padding-right: 10px
}
```

Polsterdust saab määrata korraga mitmele küljele:

```
div {padding: 0.5cm 2.5cm 2cm 4cm}
```

Seejuures võib tühikutega eraldatult kirja panna mitu väärtust variantides:

- 2 väärtust – esimene ülemisele/alumisele, teine vasakule/paremale;
- 3 väärtust – esimene ülemisele, teine vasakule/paremale, kolmas alumisele;
- 4 väärtust – esimene ülemisele, teine paremale, kolmas alumisele ja neljas vasakule.

NB! Polsterdus lisatakse tavaliselt elemendi hõivatavale pinnale ehk mõõtmetele!

Veerised

CSS võimaldab määrata ka tühja ruumi veebilehe elementide ümber – veeriseid (*margin*). Veeriseid saab määrata tekstilõikudele, piltidele jne.
Näiteks:

```
p{ margin: 0px}
```

NB! Veerised lisatakse tavaliselt elemendi hõivatavale pinnale!

Veeriste arvväärtuse võib määrata pikslites – px, punktides – pt, sentimeetrites – cm ja protsentides elementi sisaldava elemendi laiuse suhtes.

Eraldi saab määrata veeriseid üleval, all, vasakul ja paremal! Sellisel juhul lisatakse omaduse *margin* nimele liide *–top, -bottom, -left* või *-right*.

Näiteks:

```
img {
  margin-top: 5cm;
  margin-bottom: 20pt;
  margin-left: 25%;
  margin-right: 5px
}
```

Võimalik on kasutada ka väärtust „*auto*“, mille puhul arvutab veeris(t)e suuruse veebilehitseja.

NB! Kui määrata vasaku ja parema veerise väärtusteks korraga auto, siis paigutatakse objekt keskele!

Näiteks:

```
img {margin-left:auto; margin-right:auto}
```

või

```
img {margin:auto}
```

Veeriseid saab määrata ühekorraga mitmele servale, siis kasutatakse omadust *margin* ja pannakse tühikutega eraldatult kirja üks kuni neli väärtust.

```
div.eraldatud {margin: 2cm 4cm 3cm 4cm}
```

Seejuures võib tühikutega eraldatult kirja panna mitu väärtust variantides:

- 1 väärtus – ümberringi ühesugune veeris
- 2 väärtust – esimene ülemisele/alumisele, teine vasakule/paremale;
- 3 väärtust – esimene ülemisele, teine vasakule/paremale, kolmas alumisele;
- 4 väärtust – esimene ülemisele, teine paremale, kolmas alumisele ja neljas vasakule.

Tabel

CSS võimaldab määrata ka tabelite välimust. Tabelite välimust määravad suuresti omadused, mis kasutusel ka teistel elementidel (taustavärv, tekstivärv, raamjooned, polsterdus jms) kuid on kasutusel ka mõned eriomadused.

Määrata, kuidas jaotatakse ruum tabeli lahtrite, ridade ja veergude vahel, selleks on omadus *table-layout*, mille väärtuseks on:

- *auto* – mille puhul tabeli veergude laius määratakse sisu järgi, vaikeväärtus;
- *fixed* – mille korral kehtivad autori poolt määratud lahtrite mõõdud;
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
table {table-layout: auto}
```

Tabeli raamjoonte seaded

Saab määrata, kuidas näidatakse raamjooni. Terve tabeli ja lahtrite jooned võivad olla kokkupandud (ühe joonena) või eraldi nähtavad.
Kasutatakse omadust *border-collapse*, millel on järgmised võimalikud väärtused:

- *collapse* – tabeli ja lahtrite raamjooni näidatakse koos, ühe joonena;
- *separate* – tabeli ja lahtrite raamjooni näidatakse eraldi;
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
table {border-collapse: collapse}
```

Oluline info algab siit		
Esimene veerg	**Teine veerg**	**Kolmas veerg**
Esimene vasak		Esimene parem
Teine vasak	Teine keskmine	Teine parem
Kolmas vasak	Kolmas keskmine	
Oluline info lõppeb siin		

Joonis 68 Tabeli jooned koos (*collapse*)

Oluline info algab siit		
Esimene veerg	**Teine veerg**	**Kolmas veerg**
Esimene vasak		Esimene parem
Teine vasak	Teine keskmine	Teine parem
Kolmas vasak	Kolmas keskmine	
Oluline info lõppeb siin		

Joonis 69Tabeli jooned eraldi (*separate*)

Kui kasutatakse väärtust „*separate*“, siis saab omadusega *border-spacing* määrata lahtrite ja tabeli raamjoonte omavahelist kaugust (px, cm vms).

Näiteks:

```
table {
  border-collapse: separate;
  border-spacing: 10px
}
```

Kasutades siin tühikuga eraldatult kahte väärtust, määrame erinevad kaugused vasaku/parema ja ülemise/alumise külje jaoks. Näiteks:

```
table {border-spacing: 10px 50px}
```

Sageli on tabelis tühjasid lahtreid. CSS-i abil saab määrata, kas neid näidatakse, selleks on omadus *empty-cells*, millel võimalikud väärtused:

- *show* – tühje lahtreid näidatakse;
- *hide* – tühje lahtreid ei näidata;
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
table {empty-cells: show}
```

Tabeli pealdis

Tabeli pealdise (*caption*) kasutamisel saab CSS-i abil määrata, kuhu see paigutatakse. Kasutusel omadus *caption-side*, millel võimalikud väärtused *top* ja *bottom*:

```
table {caption-side:bottom}
```

Objekti nähtavus, läbipaistvus ja filtrid

Pea kõiki veebilehe elemente saab vajadusel peita või muuta osaliselt või täielikult läbipaistvaks. CSS3 on lisanud võimaluse filtrite abil visuaalsete elementide valimust muuta (värve moonutada, kujutisi hägustada jne).

Elemendi nähtamatuks muutmine

Omadusega *visibility* saab määrata, kas objekt on nähtav või mitte.
Kasutada saab järgmiseid väärtuseid:

- *visible* – element on nähtav.
- *hidden*.—element on peidetud;
- *collapse* – kasutatakse tabelis, kus eemaldab rea või veeru kuid ei mõjuta tabeli struktuuri. Teiste elementide juures kasutades töötab nagu väärtus *hidden*!
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
p.peidus {visibility: hidden}
```

NB! Peidetud element pole nähtav kuid võtab siiski ruumi!

Kui on tarvis element nii peita, et ta vabastaks ka ruumi, siis tuleb kasutada omadust *display* väärtusega *none*! Näiteks:

```
img.peidetud {display: none}
```

Objektide läbipaistvus

CSS võimaldab ka veebilehe elementide (, <img> jms) läbipaistvust muuta, selleks kasutatakse omadust *opacity*, mille väärtus määratakse 0 (täiesti läbipaistev) kuni 1 (täiesti läbipaistmatu). Näiteks:

```
img.labipaistev {opacity: 0.5}
```

NB! Murdarv kirjutatakse punktiga!

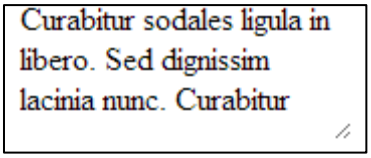
Kasutajaliides

Mitmeid uuendusi on ka kasutajaliidese kujundamiseks.

Ploki suuruse muutmine

Veebilehe elementidele on võimalik määrata võimalus nende suurust muuta. Selleks on omadus *resize*, mille võimalikud väärtused on *both*; *horizontal* ja *vertical*.
Koos omadustega *min-width*, *min-height*, *max-width ja max-height* saab luua hästi kontrollitavaid elemente.
Näiteks:

```
div {resize: horizontal}
```



Joonis 70 Element, mille suurust saab muuta (nupuke all paremal nurgas)

Kursori kuju määramine

CSS-i abil saab määrata ka seda, kuidas näeb välja kursor elemendile osutades. Selleks kasutatakse omadust cursor.

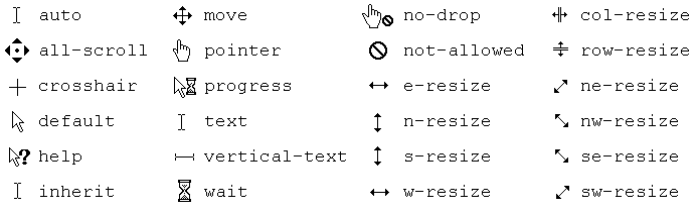
Kasutada saab järgmiseid suurt hulka väärtuseid, mõned neist:

- *default* – enamasti nool, vaikeväärtus;
- *auto* – brauser määrab kursori kuju;
- *help* – kursor viitab abiingo olemasolule (küsimärgiga);
- *pointer* – käsi, viitab hüperlingile;
- *wait* – liivakell;
- *progress* – nool liivakellaga, vaja oodata;
- *URL* – oma kursori pildifail;
- *alias* – viitab millelegi, mida luuakse;
- *all-scroll* – sisu saab igas suunas kerida;
- *move* – millegi liigutamine;
- *e-resize*, *ne-resize*, *nw-resize*, *se-resize*, *sw-resize*, *s-resize*, *w-resize* – erinevates suundades suuruse muutmine.

NB! Erinevate fraasi "*resize*" sisaldavate väärtuste korral tähistavad tähed ilmakaari: n – põhi, e – ida, s – lõuna ja w – lääs.

Näiteks:

```
h2 {cursor: crosshair}
```



Joonis 71 Näiteid erinevatest võimalikest kursoritest

Ülevaate erinevatest kursoritest saab näiteks:
http://www.w3schools.com/cssref/pr_class_cursor.asp
Näha saab kõiki näiteks:
http://www.w3schools.com/cssref/playit.asp?filename=playcss_cursor&preval=alias

NB! Kasutades erilisi kursoreid määrates nende url-i, tasub loetleda mitu võimalikku ja nimekirja lõppu mõni tavapärane juhuks kui ükski eriline ei tööta!

```
img {cursor : url("esimene.cur"), url("teine.cur"), pointer}
```

Kursoreid saab ise luua, selleks on olemas ka veebipõhised vahendid, näiteks:
http://www.rw-designer.com/online-cursor-editor

Teksti valimise lubamine

CSS3 pakub veebilehe autorile ka võimalust määrata, kas mingi elemendi sisuks olevat teksti saab kasutaja valida (*select*) või mitte.

Omadus *user-select* võib saada järgmised väärtused:

- *auto* – teksti saab valida, kui brauser seda lubab (vaikeväärtus);
- *none* – teksti valimine on keelatud;
- *text* – teksti valimine on lubatud;
- *All* – teksti valimine toimub ühe klõpsuga (topeltklõpsu asemel).

Näiteks:

```
thead {user-select: none}
```

Loendur ehk automaatne nummerdamine

Sarnaselt tekstitöötlusest tuntud automaatsetele nummerdamisvõimalustele (peatükid, joonised jms) saab nummerdada ka veebilehe elemente. Selleks saab kasutada omadust *counter*, mis põhimõtteliselt on nagu css-muutuja.

Iga loenduri ehk *counter*'i kasutamisel tuleb määrata, millisest veebilehe osast loenduri väärtus lähtestatakse (millise elemendi puhul algab numeratsioon uuesti algusest), millise elemendi juures loenduri väärtust suurendatakse ning lõpuks ka millise elemendi juures loenduri väärtust näidatakse.

Loenduri lähtestab omadus counter-reset, mille väärtusena pannakse kirja lähtestatava loenduri nimi ja vajadusel ka kasutatav algväärtus.

Näiteks lähtestame loenduri „joonised" elemendi body stiilis – numeratsioon on ühtne terve veebilehe jooksul:

```
body {counter-reset:joonised}
```

Kui on vaja alustada kindla väärtusega, siis tuleb lisada soovitud väärtus (vaikeväärtus on 0, mis tähendab, et esimene näidatav väärtus on 1).

Näiteks lähtestame loenduri „joonised" väärtusele 3:

```
body {counter-reset:joonised 2}
```

Loenduri väärtuse suurendamiseks kasutatakse omadust counter-increment.

Näiteks suurendame loenduri „joonised“ väärtust iga kord kui esineb element <figcaption>:

```
figcaption {counter-increment:joonised}
```

Vaikimisi suurendatakse loendurit alati 1 võrra. Kasutada tohib ka negatiivseid väärtuseid ning isegi 0-i. Näiteks suurendame loendurit „joonised“ kahe võrra:

```
figcaption {counter-increment:joonised 2}
```

Loenduri lisamiseks kasutatakse pseudoklasse ::before ja ::after, kus loenduri nimi lisatakse *content* ühe väärtusena.

Näiteks lisame numbrid ja sildi „Joonis“ kõigile piltidele, mis on elemendi <figure> tütarelementideks:

```
figcaption::before {content:'Joonis ' counter(joonised) ': '}
```

Objektide suurus ja paigutus

Tavapäraselt paigutatakse objektid veebilehel üksteise suhtes nii, nagu nad HTML dokumendis järjestatud ja üksteise sisse paigutatud on ning originaalsuuruses. Soovi korral saab seda aga muuta. Näiteks võib mõne elemendi kindlale kohale paigutada ning fikseerida ta selliselt, et ta püsib kohal isegi veebilehe kerimisel (*scroll*).

Suurus

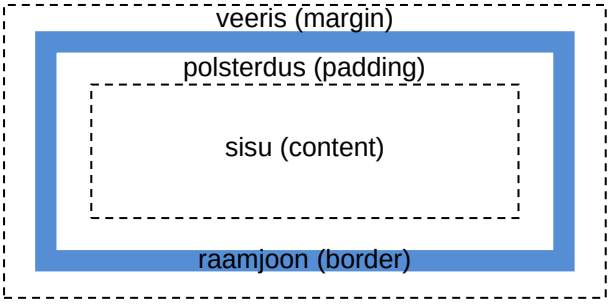
Objekti suuruse määramiseks on omadused *width* (laius) ja *height* (kõrgus), millede väärtusteks on arvud pikselites või protsentides (veebilehitseja akna mõõtude suhtes). Näiteks võime luua piltide jaoks kujundusklassi nimega "pisipilt", mille puhul on piltidel kindlad mõõtmed 100 X 75 pikselit:

```
img.pisipilt {width= 100px; height= 75px}
```

Kasti mudel

HTML elemente võib vaadelda kui kaste (*box*).

CSS-is kasutatakse elementide disainist ja paigutusest rääkides mõistet „kasti mudel“ (*box model*), mis kirjeldab elementi ümbritsevat kasti. Kast koosneb tegelikult neljast osast: sisu (*content*), polsterdus (*padding*), raamjoon (*border*) ja veerised (*margin*).



Joonis 72 Kasti mudel (box model)

NB! Tähtis on meeles pidada, et kasti ehk HTML elemendi tegelikud mõõtmed saadakse vaikimisi (nagu varasemate CSS versioonide puhul) kõigi nelja osa mõõtmete liitmisel!

CSS3 võimaldab seda põhimõtet muuta! Kasutada saab omadust box-sizing, millel on kaks võimalikku väärtust:

- content-box – suurust arvutatakse nagu vana, CSS 2.1 puhul;
- border-box – raamjoone paksus ja polsterdus arvatakse ploki mõõtmete sisse.

Näiteks:

```
box-sizing: border-box;
-moz-box-sizing: border-box;
```

NB! Lähenemine, kus elemendi mõõtmed sisaldavad ka joone paksust ning polstrit, on vajalik kohanduva (*responsive*) disaini vahendite *grid* ja *flexbox* kasutamisel!

Maksimaalsed ja minimaalsed mõõdud

Kasutada on ka omadused suurimate ning väiksemate võimalike mõõtmete jaoks: *max-width*; *max-height*; *min-width* ja *min-height*. Neid on kasulik tarvitada, kui ei soovita, et kasutaja poolt veebilehitseja akna mõõtude muutmisel objektid liialt paigast ära lähevad.

Näiteks määrame tekstilõikude minimaalseks pikkuseks 200 ja maksimaalseks laiuseks 600 pikselit, et tekstiread kunagi liialt lühikeseks või pikaks ei muutuks:

```
p {min-width: 200px; max-width: 600px}
```

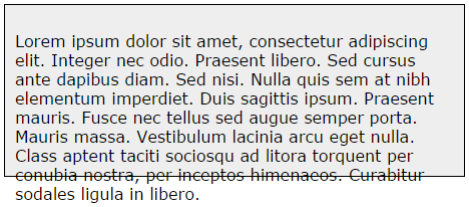
Elemendi mõõdud väiksemad kui tema sisu

Sageli juhtub, et objekti sisu on suurem kui määratud mõõdud (näiteks alajaotuse sisuks on suur hulk teksti või suuremõõduline pilt). Sellisel peab otsustama, mida teha mõõtudest välja jääva osaga.
Kasutada saab omadust *overflow*, millel on järgmised võimalikud väärtused:

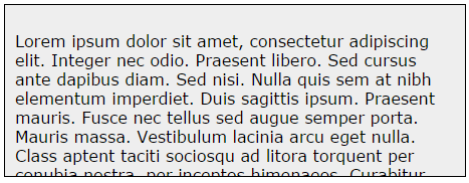
- *visible* – üle servade ulatuv sisu näidataksegi väljaspool elementi;
- *hidden* – üle servade jääv osa peidetakse lihtsalt ära;
- *auto* – lisab vajaduse korral kerimisribad (*scrollbar*);
- *scroll* –lisab kerimisribad.
- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Näiteks loome elemendile kujundusklassi nimega "aken", mille mõõtudeks on 300 X 300 pikselit ja üle servade jääva osa jaoks kasutatakse kerimisribasid:

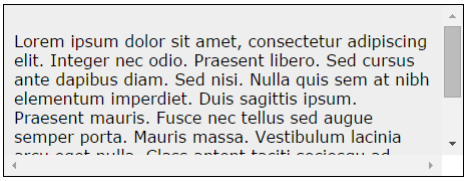
```
div.aken {
  width= 300px;
  height= 300px;
  overflow: auto
}
```



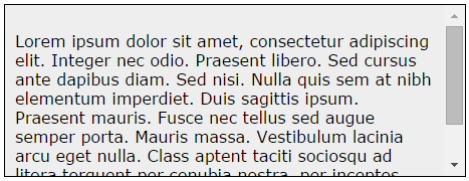
Joonis 73 Elemendi sisu näidatakse üle servade (*visible*)



Joonis 74 Üle servade jääv sisu osa peidetakse (*hidden*)



Joonis 75 Elemendile on lisatud kerimisribad (*scroll*)



Joonis 76 Kerimisriba lisatakse vajadusel (*auto*)

CSS3 lisab uute võimalustena omadused *overflow-x* ja *overflow-y*, mis võimaldavad sõltumatult kontrollida horisontaalseid ja vertikaalseid kerimisribasid!

- *visible* – paistab ka väljaspool plokki , üle serva;
- *no-display* – kui ei mahu, eemaldatakse kogu plokk;
- *no-content* – kui ei mahu, peidetakse kogu sisu.

Näiteks:

```
div.aken {
  width= 300px;
  height= 300px;
  overflow-y: auto;
  overflow-y: hidden
}
```

Elemendi kärpimine

Elementi saab ka kärpida, selleks kasutatakse omadust *clip*!

NB! Kärpida saab objekte, mille paigutus on absoluutselt määratud (omadus *position* väärtus *absolute* või *fixed*)!

Vaikimisi on selle omaduse väärtuseks *auto*, mille puhul elemendi kuju määratakse brauseri poolt. Kasutada saab veel väärtust *rect*, mis määrab kärpimise:

```
img.crop {
  position:absolute;
  clip:rect(50px 350px 250px 50px)
}
```

NB! Kärpimise väärtused määravad pildi vastavate servade asukoha järjekorras: ülemine, parem, alumine, vasak! Loomulikult on vaja teada pildi mõõtmeid!

Joonis 77 Originaal ja eeltoodud näitele vastavalt kärbitud pilt (originaalmõõdud 400X300 pikselit)

NB! Omadus clip asendatakse clip-path omadusega!

Elemendi kuvamisviis

Osa veebilehe elemente kasutavad ära terve vaba ruumi laiuse (neile eelneb ja järgneb reavahetus). Selliseid elemente nimetatakse plokielementideks (*block element*), tüüpilisteks näideteks on <h1>, <p> ja .

Osa elemente võtavaid vaid nii palju ruumi kui hädavajalik ning ei too kaasa kohustuslikke reavahetusi, neid nimetatakse reaelementideks (inline element). Tüüpilisteks näideteks on <a> ja .

Vajaduse korral on võimalik seda kuvaviisi muuta, reaelemente muuta plokielementideks ja vastupidi ning määrata veel teistsuguseidki kuvaviise!

Kasutada on omadus *display*, millel on järgmised võimalikud väärtused:

- *inline* – elementi kuvatakse reaelemendina (*inline element*) nagu näiteks ;
- *block* – elementi kuvatakse plokielemendina (*block element*) nagu näiteks h1;
- *inline-block* – elementi kuvatakse kui reaelementi, mille sisu vormindatakse kui plokielement;
- *run-in* – elementi kuvatakse rea- või plokielemendina vastavalt kontekstile;
- *list-item* – element käitub nagu loendi liige <li>;
- *table* – element käitub nagu tabel;
- *inline-table* – element käitub nagu reale paigutatud tabel;
- *table-caption* – element käitub nagu <caption> element;
- *table-column-group* – element käitub nagu <colgroup> element;
- *table-header-group* – element käitub nagu <thead> element;
- *table-footer-group* – element käitub nagu <tfoot> element;
- table-row-group – element käitub nagu <tbody> element;
- *table-cell* – element käitub nagu <td> element;
- *table-column* – element käitub nagu <col> element;
- table-row – element käitub nagu <tr> element;
- *flex* – elementi kuvatakse ploki tüüpi *flex*-konteinerina;
- *inline-flex* – elementi kuvatakse rea tüüpi *flex*-konteinerina;
- *grid* – elementi kuvatakse ploki tüüpi *grid*-konteinerina;
- *inline-grid* – elementi kuvatakse rea tüüpi *grid*-konteinerina;
- *subgrid* – kasutusel *grid*-konteineril, mis ise on *grid*-element, tema ridade ja veergude mõõdud võetakse vanemobjektilt (*parent*);
- *none* – elementi ei kuvata üldse;

- *initial* – seab selle omaduse esialgsele väärtusele;
- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt (*parent*)).

Näiteks:

```
span {display: block}
.valikud {display: flex}
```

Objektide paigutus

Objekte saab paigutada soovitud kohale ning seda mitme erineva nullpunkti (ankurobjekt) suhtes. Määramaks, mille suhtes objekti koordinaadid kirja pannakse, kasutatakse omadust *position*, millel on järgmised võimalikud väärtused:

- *static* – element paigutatakse nii nagu ta teiste suhtes normaalselt paigutuks (see on ka vaikeväärtus). Selline element ignoreerib igasuguseid koordinaatide määramisi.
- *relative* – element paigutatakse mingitele koordinaatidele oma originaalasukoha suhtes.
- *absolute* – element paigutatakse mingitele koordinaatidele teda sisaldava ploki suhtes (näiteks tabeli lahtri suhtes).
- *fixed* – element paigutatakse mingitele koordinaatidele veebilehitseja akna suhtes.

NB! Kasutades *position* omaduse väärtust *relative* hoitakse veebilehel elemendi originaalkoht originaalsuuruses ikka selle sama elemendi jaoks tühjana. Element ise nihutatakse teise kohta.

Objekti koordinaadid saab kirja panna erinevatest servadest lähtudes:

- *left* – elemendi vasaku serva kaugus teda sisaldava elemendi vasakust servast;
- *right* – elemendi parema serva kaugus teda sisaldava elemendi paremast servast;
- *top* – elemendi ülemise serva kaugus teda sisaldava elemendi ülemisest servast;
- *bottom* – elemendi alumise serva kaugus teda sisaldava elemendi alumisest servast;.

Näiteks määrame elemendile kujundusklass nimega "paremal" ülemise ja vasaku serva koordinaadid veebilehitseja akna suhtes:

```
div.paremal {
   position: fixed;
   rght:0px;
   top: 100px
}
```

NB! Kasutades neid vahendeid objektide paigutamiseks on mõistlik määrata veebilehele veerised ja polsterduse (et vältida visuaalseid erinevusi erinevate veebilehitsejate kasutamisel)!

Näiteks:

```
body {margin:0; padding:0}
```

NB! Selleks, et mingi elemendi sees saaks tema tütarelemente absoluutsete koordinaatidega paigutada, peaks elemendil olema position omaduse väärtus selline, mis laseb koordinaate kasutada, näiteks relative. Seejuure pole vaja talle mingeid koordinaate üldse määrata – siis ta asub ikka oma vaikimisi kohal.

Seda kasutatakse näiteks elementidel, mille sees luuakse CSS-animatsioon.

Objekti paigutus z-teljel

Kui on tarvis objekte üksteise peale või alla asetada, siis kasutatakse omadust *z-index* (nagu koordinaadid z-teljel). Väärtustena saab kasutada positiivseid aga ka negatiivseid täisarve, vaikeväärtuseks (*default*) on „auto“, mis sisuliselt on 0.

Näiteks loome piltide jaoks kujundusklassi nimega "esiplaan", mille abil asetame pildid teiste objektide suhtes kõrgemale kihile (näites on väärtuseks 10):

```
img.esiplaan {z-index: 10}
```

NB! Selle omaduse kasutamisel peab objektile olema määratud *position* omaduse väärtus *relative, absolute* või *fixed*!

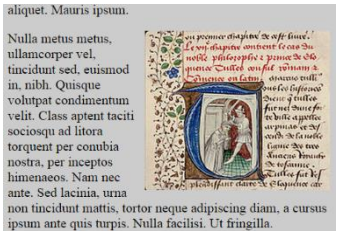
Objekti „hõljumine“ vasakul/paremal

Niinimetatud kastikujulisi elemente (*box elements*) saab lasta teiste elementide (näiteks teksti) kõrval vasakul või paremal serval „hõljuda“ (*float*). Selleks kasutatakse *float* omadust, millel on järgmised võimalikud väärtused:

- *none* – objekt ei „hõlju“, paikneb seal, kus ta tekstis paigutatud on, see on ka vaikeväärtus;
- *inherit* – objekt pärib vastava omaduse teda sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele;
- *left* – objekt „hõljub“ vasakul pool;
- *right* – objekt „hõljub“ paremal pool.

Näiteks:

```
img {float: right}
```



Joonis 78 Paremale poole paigutatud pilt

Elemendid hõljuvad nii kaugel servas kui võimalik. Kõik hõljuvale elemendile järgnevad HTML elemendid paigutatakse ümber tema. Hõljuvale elemendile eelnevaid elemente see omadus ei mõjuta.

Float omadusega on seotud omadus clear, mille väärtus määrab, kus mingi elemendi kõrval „hõljuvate“ objektide kasutamist keelatakse. Võimalikud väärtused on:

- *none* – „hõljuvaid“ objekte lubatakse mõlemale poole (vaikeväärtus);
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele;

- *both* – „hõljuvaid“ objekte ei lubata kummalegi poole;
- *left* – „hõljuvaid“ objekte ei lubata vasakule poole;
- *right* – „hõljuvaid“ objekte ei lubata paremale poole.

Näiteks:

```
ul {clear: both}
```

Vertikaalne joondus

Veebilehe elemente saab ka vertikaalselt joondada, näiteks keskele. Kasutatakse omadust *vertical-align*, millel on näiteks järgmised võimalikud väärtused:

- *baseline* – elemendi baasjoon joondatakse teda sisaldava elemendi (*parent*) baasjoonega (vaikeväärtus);
- *sub* – element joondatakse nagu oleks tegemist alaindeksiga (*subscript*);
- *super* – element joondatakse nagu oleks tegemist ülaindeksiga (*superscript*);
- *top* – elemendi ülemine serv joondatakse kõrgeima samal real asuva elemendi ülaserva järgi;
- *middle* – element joondatakse teda sisaldava elemendi keskele;
- *bottom* – element joondatakse kõige madalama samal real asuva elemendi järgi;
- *text-top* – element joondatakse teda sisaldava elemendi (*parent*) teksti fondi ülemise serva järgi;
- *text-bottom* – element joondatakse teda sisaldava elemendi (*parent*) teksti fondi alumise serva järgi;
- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*);
- *initial* – seab selle omaduse esialgsele väärtusele.

Näiteks:

```
img {vertical-align: middle}
```

Kasutada saab ka arvväärtuseid pikslite või protsentidena (*line-height* omaduse väärtuse suhtes). Sellisel juhul elementi kas kergitatakse (positiivsed väärtused) või nihutatakse allapoole (negatiivsed väärtused).

Näiteks:

```
span {vertical-align: -25%}
```

Kohanduv disain

Erinevate ekraanisuuruste aga ka erinevate seadmetega kohandumiseks pakutakse mitmeid erinevaid vahendeid.

Kohandumist võib jagada kahte kategooriasse:

- *adaptive* –adaptiivne ehk kohanev. Sellisel juhul muutub veebilehe elementide suurus ja paigutus astmeliselt, sammude kaupa. Määratakse justkui katkestuspunktid (*breakpoints*), mis määravad lävendid, mille puhul rakendub teistsugune kujundus. Sellise kohandamise peamine tööriist on meediapäringud (*media queries*).

Samuti sobib siia kategooriasse HTML <picture> element, mis muudab pildi kohanduvateks.

- *responsive* – kiirelt reageeriv, operatiivne. Sellisel juhul on kujundus sujuvalt muutuv (*fluid*).
  Kiirelt reageeriva (*responsive*) disaini elementide paigutamise jaoks pakub CSS3 kaht erinevat vahendite komplekti, millega veebilehe elemente paigutada: *flexbox* ja *grid*. *Flexbox* paigutab elemente ühesuunaliselt (horisontaalselt või vertikaalselt), *grid* aga kahesuunaliselt. Tõeliselt kohanduva kujunduse loomiseks kasutatakse neid koos.

Oluline vahend kohanduva disaini jaoks on ka vaateava (*viewport*), mida kasutatakse lehe elementide suuruse määramiseks.

Teksti suuruse määramiseks kohanduva disainiga lehtedel tuleks kasutada mõõtühikuid em (elemendi teksti suurus) ja rem (lehe juurelemendi teksti suurus).

Meediapäringud

Samuti nagu HTML võimaldab määrata erinevaid stiililehti (CSS faile) erinevate seadmete jaoks, saab ka CSS vahenditega defineerida erinevaid kujundusi erinevatel väljundseadmetel kasutamiseks.
Alates CSS2-st saab kasutada @media reeglit (*rule*). CSS3 laiendab varasemat meediatüüpide ideed keskendudes enam seadmete võimalustele (mitte enam niivõrd meediaseadmete tüüpidele).

Süntaks näeb ette meediatüübi ja/või ühe või mitme avaldise (loogika) kirjutamise

```
@media not|only mediatype and|not|only (meedia omadused) {
  CSS-kood;
}
```

CSS3 toetab järgmiseid **meediatüüpe**:

- *all* – kõik seadmed;
- *print* – printerid;
- *screen* – arvutiekraani, tahvelarvutid, nutitelefond jne;
- *speech* – kõnesüntees (nägemispuudega inimeste ekraanilugerid).

Näiteks:

```
@media print {
  body {background-color: white}
}
```

Meedia **omadustena** (*media features*) on kasutatavad järgmised omadused:

- *aspect-ratio* – väljundi laiuse/kõrguse suhe (näiteks 16/9, kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *color* – väljundseadme ekraani värvussügavus (kasutada saab eesliiteid „*min*-“ ja „*max*-“);

- *color-index* – värvide arv, mida väljundseadme ekraan suudab näidata (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *device-aspect-ratio* – väljundseadme/paberi laiuse/kõrguse suhe (näiteks 16/9, kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *device-width* – väljundseadme/paberi laius (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *device-height* – väljundseadme/paberi kõrgus (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *grid* – kas väljundseade on võrgustikuga (*grid*, kindel arv ridu ja sümboleid) või rastergraafika (*bitmap*) seade („1“ – *grid*, „0“ – muu variant).
- *height* – nähtava ala kõrgus (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *width* – nähtava ala laius (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *resolution* – väljundi punktitihedus dpi või dpcm (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *orientation* – väljundi orientatsioon („*portrait*“ või „*landscape*“);
- *monochrome* – bittide arv piksli kohta monokroomse väljundi puhul (*bits per pixel in a monochrome frame buffer*) (kasutada saab eesliiteid „*min*-“ ja „*max*-“);
- *scan* – kasutatava TV-ekraani realaotusmeetod („*progressive*“ või „*interlace*“);
- *update-frequency* – kui kiiresti meediaseade sisu muuta suudab, väärtused *none* (ei muudeta, näiteks paberil), *slow* (ei toeta sujuvaid animatsioone jms, näiteks e-ink ekraanid), *normal* (tavaline, kiiruspiiranguid pole, tavalised ekraanid jms).

Näiteks pealkirjade ette TLÜ tilluke logo:

```
@media screen and (min-width: 900px) {
  h1:before h2:before h3:before
  {
    content: url(„pildid/tlu_bullet.png“);
  }
}
```

Vaateava ehk viewport

Vaateava ehk *viewport* on veebilehe vaatajale nähtav ala. Selle suurus sõltub seadmest, mida veebilehe vaatamiseks kasutatakse.

Vaateava kasutamiseks tuleb veebilehe päises kirja panna vastav *meta* element.

Vaateavaga on seotud mõõtühikud, mis viitavad sajandikule (1%) vaateava mõõtudest ning mida kasutatakse kohanduva kujunduse puhul erinevate suuruste määramiseks veebilehel. Vaateava suurusega seotud mõõtühikud (*viewport-relative units*) on:

- *vw* – 1% vaateava laiusest (*viewport width*);
- *vh* – 1% vaateava kõrgusest (*viewport height*);

- *vmin* – 1% vaateava lühemast servast (*viewport minimum*), kas laiusest või kõrgusest olenevalt kumb on lühem;
- *vmax* – 1% vaateava pikemast servast (*viewport maximum*), kas laiusest või kõrgusest olenevalt kumb on pikem.

Pärast vaateava kasutuselevõttu on protsentuaalsed mõõdud veebilehe suuruse suhtes asendatud protsentidega vaateava suuruse suhtes.

Flexbox

Flexbox võimaldab mingi ploki piires elemente automaatselt paigutada, jaotada, nende suurust muuta jne.
Lugeda saab näiteks:

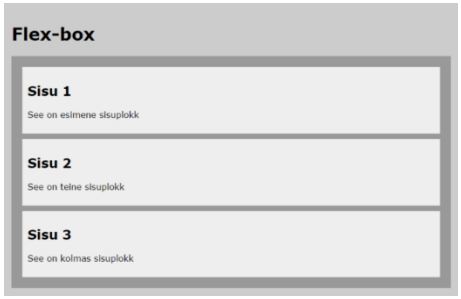
- https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes
- http://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox-i puhul määratakse omadusi elemendile, mille sees tütarelemente paigutama hakatakse ehk flex-konteinerile (*flex-container*) ja elementidele, mida paigutatakse ehk flex-elementidele (*flex-item*).

Selleks, et mingi element (näiteks) muuta flex-konteineriks, tuleb tema *display* omaduse väärtuseks määrata flex või *inline-flex*.
Näiteks:

```
.galleryarea {display: flex}
```

Järgnevatel joonistel on leht, mille sisu on ühe elemendi sees, alguses tavapäraselt ja siis on sama flex-konteinerina.



Joonis 79 Veebi sisu tavapäraselt

Joonis 80 Sama veebilehe sisu flex-konteineris

Järgnevalt anname ülevaate omadustest, millega määratakse flex-konteineri omadused ja konteineri sisse paigutatud flex-elementide omadused.

Flex-konteineri omadused

Flex-konteineri omadused määravad flex-elementide paigutuse ridadesse/veergudesse ja nende ridade/veergude omadused.

Flex konteineri sisu paigutamise suund

Kogu sellise flex-konteineri sisu paigutatakse vastavalt määratud reeglitele. Vaikimisi kasutatakse horisontaalset paigutust vasakult paremale.
Suunda saab muuta näiteks kogu veebilehele *direction* omadust määrates (näiteks rtl – *right to left*)

Flex-konteneri sees määrab elementide järjestuse suuna omadus *flex-direction*, millel on järgmised võimalikud väärtused:

- *row* – vaikeväärtus, elemendid paigutatakse vasakult paremale, ülevalt alla.
- *row-reverse* – elemendid paigutatakse paremalt vasakule.
- *column* – elemendid paigutatakse vertikaalselt.
- *column*-reverse – elemendid paigutatakse vertikaalselt, alt ülespoole.

Näiteks:

```
.suursisu {
  display: flex;
  flex-direction: row-reverse;
}
```

Joonis 81 Flex-konteiner elemendid reas vastupidi (*row-reverse*)

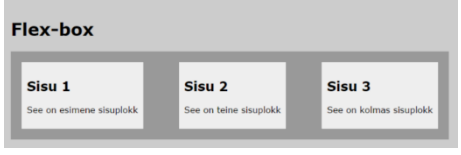
Flex elementide joondamine

Kui elemendid flex-konteineris ei kata real/veerus võimalikku ruumi, siis saab neid joondada kasutades omadust *justify-content*, millel on järgmised võimalikud väärtused:

- *flex-start* – elemendid paigutatakse konteineri algusesse (vaikeväärtus).
- *flex-end* – elemendid paigutatakse konteineri lõppu.
- *center* – elemendid paigutatakse konteineri keskele.
- *space-between* – elementide vahele paigutatakse tühja ruumi (nagu teksti rööpselt joondamine).
- *space-around* – elementide vahele, ette ja järele paigutatakse tühja ruumi.

Näiteks:

```
.suursisu {
  display: flex;
  justify-content: space-between;
}
```



Joonis 82 Flex-konteiner, elemendid ühtlaselt jaotatud (*space-between*)

Risti sisu paigutamise suunaga (*flex-direction*) saab elemente flex-konteineris joondada omadusega *align-items*, millel on järgmised võimalikud väärtused:

- *stretch* – elemendid venitatakse kõrgemaks, et täita kogu flex-konteineri kõrgus (vaikeväärtus).
- *flex-start* – elemendid paigutatakse konteineri ülaserva.
- *flex-end* – elemendid paigutatakse konteineri alumisele servale.
- *center* – elemendid paigutatakse konteinerisse vertikaalselt keskele.
- *baseline* – elemendid paigutatakse konteineri baasjoonele (*baseline*).

Näiteks:

```
.suursisu {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: flex-start;
}
```

Joonis 83 Flex elemendid vertikaalselt venitatud (*stretch*)

Joonis 84 Flex elemendid paigutatud konteineri ülaserva (flex-start)

Flex ridade/veergude murdmine

Juhuks, kui flex-konteiner pole piisavalt suur, et kõik elemendid mahuksid ühele reale või ühte veergu, saab määrata rea murdmist (wrap). Selleks on omadus *flex-wrap*, millel on järgmised võimalikud väärtused:

- *nowrap* – ridade/veergude murdmist ei toimu (vaikeväärtus).
- *wrap* – ridade veergude murdmine toimub vastavalt vajadusele.
- *wrap reverse* – ridade murdmine vastupidises järjestuses (hoopis alt üles vms).

Näiteks:

```
.suursisu {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap
}
```

Joonis 85 Flex elemendid ei mahu reale, toimub ridade murdmine (*wrap*)

Joonis 86 Flex elemendid ei mahu reale, toimub ridade murdmine tagurpidi järjestuses (*wrap-reverse*)

Flex ridade joondus

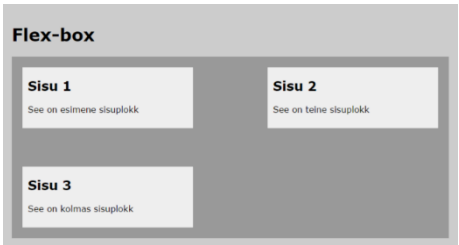
Sarnaselt üksikutele elementidele flex-konteineri reas/veerus, saab joondada ka terveid elementide ridu/veergusid. Selleks on omadus *align-content* järgmiste võimalike väärtustega:

- *stretch* – ridu venitatakse nii, et nad täidavad kogu olemasoleva ruumi (vaikeväärtus);
- *flex-start* – read paigutatakse flex-konteineri algusesse.
- *flex-end* – read paigutatakse flex-konteineri lõppu.
- *center* – read paigutatakse flex-konteineri keskele,
- *space-between* – ridade vahele paigutatakse lisaruumi, et katta kogu flex-konteiner.
- *space-around* – read paigutatakse ühtlaselt, ruumi lisatakse enne ja pärast ridu ning ridade vahele

Näiteks:

```
.suursisu {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  align-content: space-between;
}
```

Joonis 87 flex-ridade joondus *stretch*



Joonis 88 flex-ridade joondus *space-between*

Flex-konteineri sisu paigutamise suund ja ridade/veergude murdmine koos

Ajalooliselt on flex-konteineri jaoks välja mõeldud ka elementide paigutamisega seotud omaduse *flex-direction* ja ridade/veergude murdmise omaduse *flex-wrap* kombinatsioon – *flex-flow*.
Omadus pannakse kirja järgmiselt:

```
flex-flow: flex-direction flex-wrap;
```

Flex-elementide omadused

Flex-elementide omadused määravad käitumise.

Ühena tavalistest omadustest kasutatakse veeriseid ehk *margin*.

Flex-elemendi järjestuse muutmine

Kui on tarvis mõnd flex-elementi teiste suhtes ette või tahapoole nihutada, siis kasutatakse omadust *order*, mille väärtuseks on positiivne või negatiivne arv. Negatiivsed tõstavad elemendi ettepoole, positiivsed tahapoole.
Väärtus 0 jätab elemendi omale kohale.

Näiteks:

```
.ettepoole {order: -1}
```

Joonis 89 Teine sisuplokk tõstetud ühe koha võrra ettepoole (*order*: -1)

Flex-elemendi joondamine

Kui on vaja mõnd elementi flex-konteineris eriliselt joondada, siis saab kasutada omadust *align-self*, millel on samad võimalikud väärtused, nagu *align-items* omadusel.
Selle omaduse kasutamisel eiratakse flex-konteineri *align-items* väärtust.

Näiteks:

```
.ettepoole {
  order: -1;
  align-self: flex-start;
}
```

NB! See omadus on uus ja ei pruugi toimida;

Flex-elemendi suhteline suurus

Flex konteineris saab kõigile elementidele määrata ka nende suurust. Selleks kasutatakse omadust *flex-basis*, mille väärtuseks võib olla:

- *auto* – flex-elemendi suurus määratakse tema sisu järgi (vaikeväärtus).
- Täisarv – kasutada võib kõiki tüüpilisi mõõtühikuid (px, pt, cm, em, %).
- *initial* – seab väärtuse vaikeväärtusele.
- *inherit* – väärtus päritakse vanem-elemendilt.

Näiteks:

```
.pisike {flex-basis: 200px}
```

Kuna flex-konteineris elementide mõõdud vastavalt konteineri mõõtudele dünaamiliselt muutuvad, siis pakutakse ka võimalust määrata, kui palju elementide mõõdud teiste suhtes kasvavad või kahanevad.
Elemendi suurenemist reguleerib omadus *flex-grow*, mille väärtuseks on positiivne täisarv (vaikeväärtuseks on 0).

Näiteks:

```
.veerand {flex-grow: 1}
.pool{flex-grow: 2}
```

Sarnaselt reguleerib elementide kahanemist omadus *flex-shrink*. Erinevuseks aga, et vaikeväärtuseks on 1.

Näiteks:

```
.tava{flex-shrink: 2}
.kahanev{flex-grow: 4}
```

Siin kirjeldatud kolm omadust saab kirja panna lühemalt ühe omadusena *flex*. Selle puhul pannakse väärtused kirja järgmiselt:

```
flex: flex-grow flex-shrink flex-basis;
```

Vaikeväärtuseks on:

```
flex: 0 1 auto;
```

Vajadusel saab elementide suurust piirata tavapäraste *max-width* ja *min-width* (ka *max-height* ja *min-height*) abil.

Grid

Grid ehk võrestik on vahend veebilehe ülesehituse määramiseks, mis jagab lehe veergudeks ning ridadeks.
Elementi, mille sees võrestik on, nimetatakse grid-konteineriks (*grid-container*). Kõik grid-konteineri otsesed tütarelemendid on grid-elemendid (*grid-item*).
Tulemuse saavutamiseks määratakse omadusi grid-konteinerile ja selle tütarelementidele – grid-elementidele.

Selleks, et mingi element (näiteks) muuta grid-konteineriks, tuleb tema kujunduses *display* omaduse väärtuseks määrata *grid* või *inline-grid*. Kui element on ise suurema *grid*-konteineri tütarelemendiks, siis võib kasutada ka väärtus *subgrid*, et tema ridade ja veergude mõõdud võetakse vanemelemendi järgi.

Näiteks:

```
.main-container {display: grid}
```

NB! Grid'i kasutamiseks soovitatakse veebilehe elementide suuruse määramiseks kasutada lähenemist, kus elemendi raamjoone paksus ning polster sisalduvad elemendi mõõtmetes (*box-sizing: border-box*).

NB! Omadused *column*, *float*, *clear* ja *vertical-align* grid-konteineri puhul ei toimi!

Grid-konteineri omadused

Grid-konteineri omadused määravad, kuidas jaotatakse ruum konteineri sees. Kõige olulisem on võrestiku (*grid*) defineerimine.

Võrestiku defineerimine

Võrestik defineeritakse ridade ja veergude määramisega.

Veergude ja ridade defineerimine

Grid-konteineri sees ridade ja veergude defineerimiseks kasutatakse omadusi *grid-template-columns* ja *grid-template-rows*. Loodavad veerud ja read jäävad justkui nähtamatute joonte vahele, mida nimetatakse rea-joonteks (*row-line*) ja veeru-joonteks (*column-line*).

Nende joonte abil saab määrata grid-elementide paigutust (millisest joonest millise jooneni element ulatub). Selleks on grid-elementidel vastavad omadused.

Veergude ja ridade loomisel omadustega *grid-template-columns* ja *grid-template-rows* antakse väärtusteks tühikutega eraldatud loend veergude/ridade laiusest/kõrgusest (*track values*). Lisada võib ka rea-joonte ja/või veeru-joonte nimed, mis kirjutatakse kandiliste sulgude vahele.

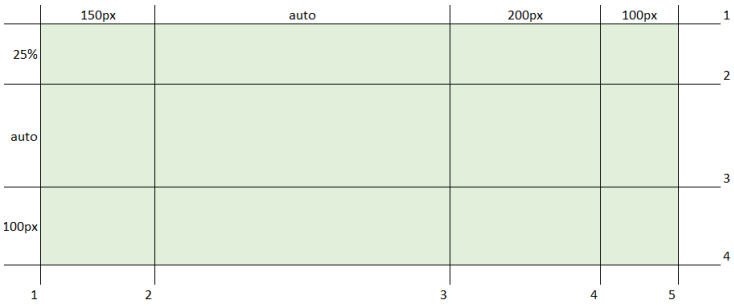
Veergude/ridade mõõtude määramiseks on järgmised võimalused:

- Väärtus puudub – veerud või read luuakse vajadusel automaatselt;
- *auto* – veeru või rea mõõdud määratakse grid-konteineri mõõtude ja grid-elementide sisu järgi (vaikeväärtus);
- *max-content* – veeru või rea suurus määratakse sellele paigutatud elementidest suurima järgi;
- *min-content* - veeru või rea suurus määratakse sellele paigutatud elementidest väikseima järgi;
- väärtus – erinevate mõõtühikutega (px, cm, jne) väljendatud täpne suurus;

NB! Kui veergudele/ridadele nimesid ei määrata, siis antakse neile automaatselt numbrilised nimed.

Näiteks määrame grid-konteinerile 4 veergu ja kolm rida:

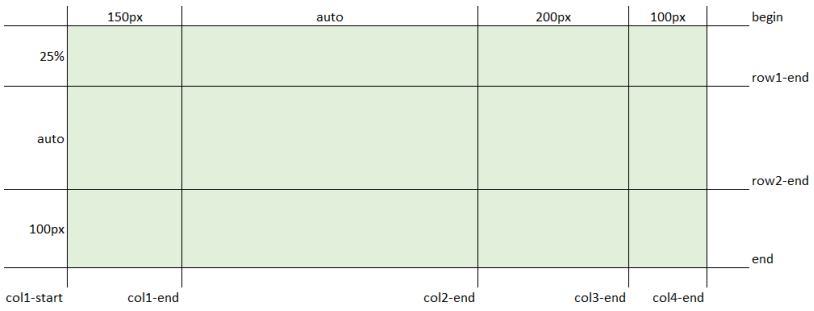
```
.main-container {
  display: grid;
  grid-template-columns: 150px auto 200px 100px;
  grid-template-rows: 25% auto 100px;
}
```



Joonis 90 Defineeritud grid, 4 veergu ja 3 rida, rea- ja veeru-jooned nimedeta

Näiteks määrame grid-konteinerile 4 veergu ja 3 rida nagu eespool aga anname kõigile nimed rea-joontele (*row-lines*) ja veeru-joontele (*column-lines*) nimed:

```
.main-container {
  display: grid;
  grid-template-columns: [col1-start] 150px [col1-end] auto [col2-end] 200px [col3-end] 100px
[col4-end];
  grid-template-rows: [begin] 25% [row1-end] auto [row2_end] 100px [end];
}
```



Joonis 91 Defineeritud grid, 4 veergu ja 3 rida koos rea- ja veeru-joonte nimedega

NB! Ridadel võib olla ka mitu nime! Need pannakse kirja tühikutega eraldatult.
Näiteks:

```
grid-template-rows: [toprow headerarea] 25% [contentrow] auto [bottomrow] 100px [end];
```

NB! Ridade või veergude määramisel saab kasutada kordust, *repeat*()!

Näiteks määrame neli veergu, milledest 3 esimest on ühesugused:

```
grid-template-columns: repeat(3, 150px) [contentcolumn] auto;
```

See on samaväärne järgmise näitega:

```
grid-template-columns: 150px 150px 150px [contentcolumn] auto;
```

Veergude ja ridade mõõduks võib määrata ka murdosa vabast ruumist. Selleks kasutatakse mõõtühikut fr (*fraction*).
See murdosa arvutatakse ruumist, mis jääb üle kõigist täpselt määratud osadest. Näiteks määrame neli veergu, ühe laiuseks 200 pikslit ja kõigi ülejäänute laiuseks on kolmandik järelejäänud ruumist:

```
grid-template-columns: 200px 1fr 1fr 1fr;
```

Võrestiku defineerimine lühikesel kujul

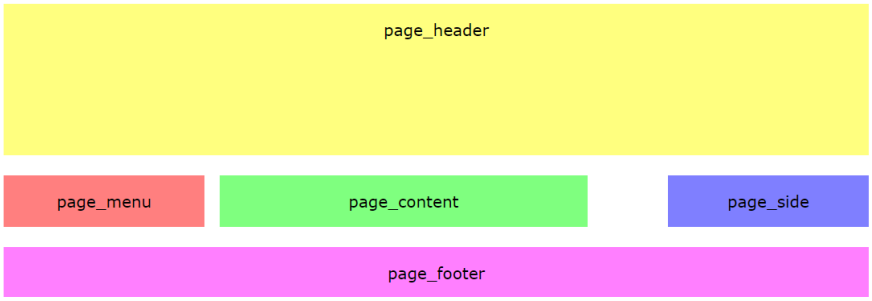
Võrestiku ridu, veergusid ja piirkondi saab defineerida ka ühe omadusega *grid-template*.

Tühi ruum ridade ja/või veergude vahel

Vajadusel saab ridade ja veergude vahele lisada tühja ruumi. Selleks on kasutusel omadused *grid-column-gap* ja *grid-row-gap*, millede väärtuseks on soovitud vahe suurus (*length*) mingites ühikutes (px, cm vms).

Näiteks:

```
.main-container {
  display: grid;
  grid-template-columns: 200px auto 50px 200px;
  grid-template-rows: 150px auto 50px;
  grid-column-gap: 15px;
  grid-row-gap: 20px;
}
```



Joonis 92 Grid veergude ja ridade vahel on lisatud tühi ruum

NB! Tühi ruum lisatakse vaid ridade ja veergude vahele, mitte väljapoole!

Tühja ruumi võib lisada ka ühe omadusega *grid-gap* korraga nii veergude kui ridade jaoks. Väärtustena pannakse kirja tühikutega eraldatuna vahe veergude vahel (*column-gap*) ja ridade vahel (*row-gap*).

Näiteks:

```
grid-gap: 15px 20px;
```

NB! Kasutada võib ka vaid üht väärtust, siis kasutatakse seda nii veergude kui ka ridade vahe määramisel.

Näiteks:

```
grid-gap: 20px;
```

Piirkondade defineerimine grid-elementide sidumiseks võrestikuga

Veergude ja ridadega defineeritud võrestikku saab jagada ka nimelisteks piirkondadeks (*area*). Selleks tuleb eelnevalt defineerida nimed grid-elementidele, kasutades grid-elemendi omadust *grid-area*.

Näiteks defineerime grid-elementide nimed: „page_header“, „page_menu“, „page_content“, „page_side“ ja „page_footer“.

Nimetuse saanud piirkonnad seotakse loodud võrestikuga (*grid*) omadusega *grid-template-areas*. Iga võrestiku rea (*row*) jaoks pannakse jutumärkides kirja mis piirkond (*area*) on seotud iga veeruga (*column*).
Iga veeru kohta pannakse kirja üks kolmest:

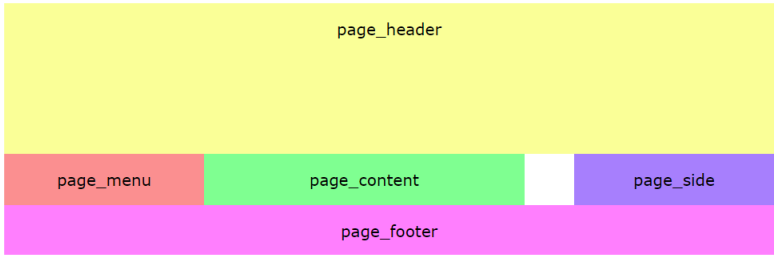
- piirkonna nimi – selle veeruga on seotud vastav piirkond, kui sama piirkond kordub mitmes veerus, siis ulatub (*span*) see piirkond üle kõigi nende veergude;
- . – (punkt) tähendab, et võrestiku lahter jääb tühjaks;
- *none* – võrestiku piirkonda pole defineeritud.

NB! Iga rea kohta tuleb kirja panna täpselt sama palju veergusid!

NB! Ridadele nimesid ei kirjutata kuid nad saavad nimed automaatselt (näiteks rida, millel algab piirkond „*content*“ saab nimeks „*content-start*“ ja rida, kus see piirkond lõppeb, saab nimeks „*content-end*“.

Näiteks:

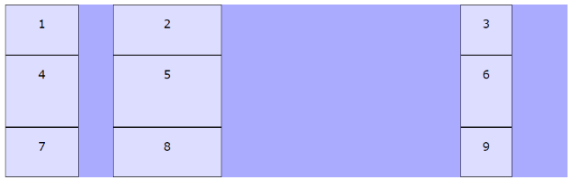
```
.main-container {
  display: grid;
  grid-template-columns: 200px auto 50px 200px;
  grid-template-rows: 150px auto 50px;
  grid-template-areas:
    "page_header page_header page_header page_header"
    "page_menu page_content . page_side"
    "page_footer page_footer page_footer page_footer";
}
```



Joonis 93 Grid võrestikuga on seotud defineeritud piirkonnad

Võrestiku joondamine

Vahel võivad grid-elemendid katta väiksema ala, kui võrestik. See juhtub tavaliselt, kui nende suuruse määramiseks kasutatakse absoluutseid ühikuid nagu näiteks px.



Joonis 94 Grid-elementide laius on väiksem, kui võrestiku veergudel

Joonis 95 Grid-elementide kõrgus on väiksem, kui võrestiku ridadel

Sellisel juhul saab tervet võrestikku grid-konteineris joondada.

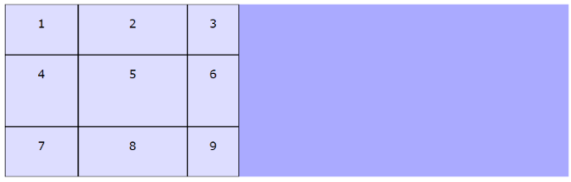
Võrestiku joondamine horisontaalselt

Terve võrestiku joondamiseks rea telje (*row axis*) järgi ehk horisontaalselt on omadus *justify-content*, millel on järgmised võimalikud väärtused:

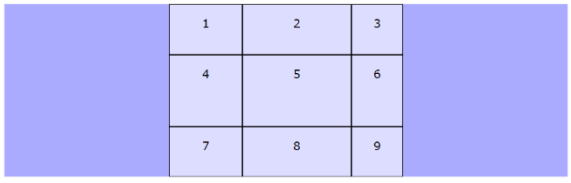
- *start* – võrestik joondatakse grid-konteineri vasakusse serva;
- *end* – võrestik joondatakse grid-konteineri paremasse serva;
- *center* – võrestik joondatakse grid-konteineri keskele;
- *stretch* – grid-elemente venitatakse laiemaks, et võrestik täidaks kogu grid-konteineri laiuse;
- *space-around* – lisab kõigi grid-elementide vahele võrdselt tühja ruumi ning rea algusesse ja lõppu pool sellest laiusest, et täita terve grid-konteiner;
- *space-between* – täidab kogu grid-konteineri laiuse lisades vaid grid elementide vahele võrdselt tühja ruumi;
- *space-evenly* – lisab terve grid-konteineri laiuse täitmiseks võrdselt tühja ruumi kõigi grid-elementide vahele ning ka rea algusesse ja lõppu.

Näiteks:

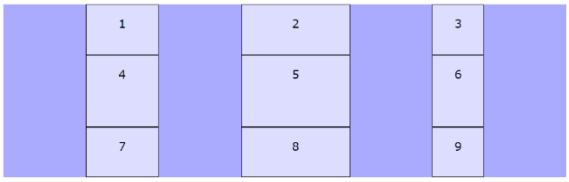
```
.main-container {
  justify-content: start;
}
```



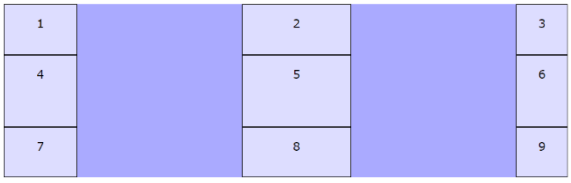
Joonis 96 Võrestik joondatud vasakule (*justify-content: start*)



Joonis 97 Võrestik joondatud keskele (*justify-content: center*)



Joonis 98 Võrestik joondatud ühtlaselt (*justify-content: space-evenly*)



Joonis 99 Võrestik joondatud ühtlaselt äärest ääreni (*justify-content: space-between*)

Võrestiku joondamine vertikaalselt

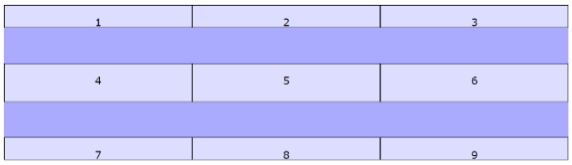
Terve võrestiku joondamiseks veeru telje (*column axis*) järgi ehk vertikaalselt on omadus *align-content*, millel on järgmised võimalikud väärtused:

- *start* – võrestik joondatakse grid-konteineri ülemisse serva;
- *end* – võrestik joondatakse grid-konteineri alumisse serva;
- *center* – võrestik joondatakse grid-konteineri keskele;

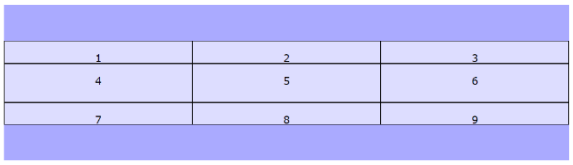
- *stretch* – grid-elemente venitatakse kõrgemaks, et võrestik täidaks kogu grid-konteineri kõrguse;
- *space-around* – lisab kõigi grid-elementide vahele võrdselt tühja ruumi ning veeru algusesse ja lõppu pool sellest kõrgusest, et täita terve grid-konteiner;
- *space-between* – täidab kogu grid-konteineri kõrguse lisades vaid grid elementide vahele võrdselt tühja ruumi;
- *space-evenly* – lisab terve grid-konteineri kõrguse täitmiseks võrdselt tühja ruumi kõigi grid-elementide vahele ning ka veeru algusesse ja lõppu.

Näiteks:

```
.main-container {
  align-items: space-between;
}
```



Joonis 100 Võrestik joondatud ühtlaselt üle kogu kõrguse (*align-content: space-between*)



Joonis 101 Võrestik joondatud vertikaalselt keskele (*align-content: center*)

Grid-elementide sisu joondamine

Tervele võrestikule (grid-konteiner) saab määrata ka omadusi, kuidas kõikide sisuelementide (grid-elemendid) sisu joondatakse.

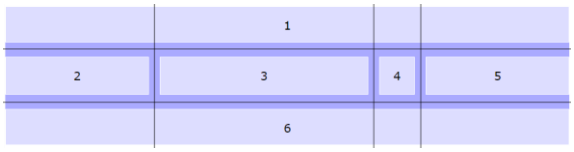
Elementide sisu joondamine horisontaalselt

Mööda ridade telgesid (*row axis*) ehk horisontaalselt joondamiseks on omadus *justify-items*, millel on järgmised võimalikud väärtused:

- *start* – kõikide grid-elementide sisu joondatakse vasakusse serva;
- *end* – kõikide grid-elementide sisu joondatakse paremasse serva;
- *center* – kõikide grid-elementide sisu joondatakse keskele;
- *stretch* – kõikide grid-elementide sisu täidab terve võimaliku ala (vaikeväärtus).

Näiteks:

```
.main-container {
  justify-items: end;
}
```



Joonis 102 Võrestiku kõik grid-elemendid on joondatud üle terve võimaliku ala (*justify-items: stretch*)

Joonis 103 Võrestiku kõik grid-elemendid on joondatud vasakule (*justify-items: start*)

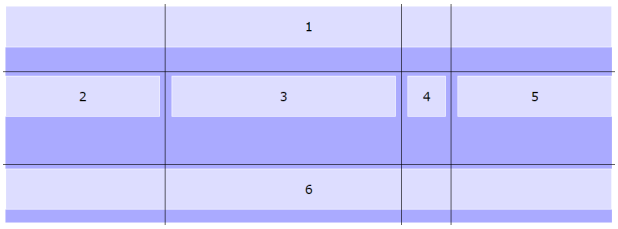
Elementide sisu joondamine vertikaalselt

Mööda veergude telgesid (*column axis*) ehk vertikaalselt joondamiseks on omadus *align-items*, millel on järgmised võimalikud väärtused:

- *start* – kõikide grid-elementide sisu joondatakse ülemisse serva;
- *end* – kõikide grid-elementide sisu joondatakse alumisse serva;
- *center* – kõikide grid-elementide sisu joondatakse vertikaalselt keskele;
- *stretch* – kõikide grid-elementide sisu täidab terve võimaliku kõrguse (vaikeväärtus).

Näiteks:

```
.main-container {
  align-items: start;
}
```



Joonis 104 Võrestiku kõik grid-elemendid on joondatud üles (*justify-items: start*)

Veergude ja ridade automaatne suurus

Võrestiku veergudele ja ridadele, millele pole mõõtu määratud, saab seda teha omadustega *grid-auto-columns* ja *grid-auto-rows*. Mõlemal saab kasutada järgmiseid võimalikke väärtuseid:

- *auto* – veeru või rea mõõdud määratakse grid-konteineri mõõtude ja grid-elementide sisu järgi (vaikeväärtus);
- *max-content* – veeru või rea suurus määratakse sellele paigutatud elementidest suurima järgi;
- *min-content* - veeru või rea suurus määratakse sellele paigutatud elementidest väikseima järgi;
- väärtus – erinevate mõõtühikutega (px, cm, % jne) väljendatud täpne suurus;

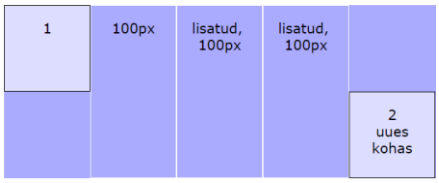
Neid omadusi kasutatakse peamiselt siis, kui osa veergusid või ridasid tekitatakse automaatselt.

Näiteks loome võrestiku kahe veeru ja kahe reaga ning määrame automaatselt loodavatele veergudele laiuse:

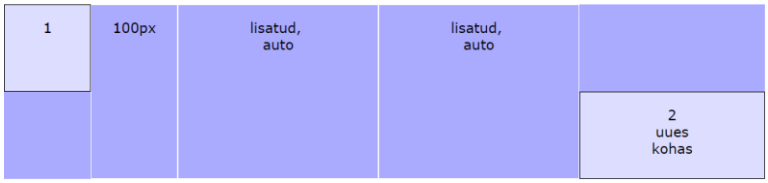
```
.main-container {
  display: grid;
  grid-template-columns: 100px 100px;
  grid-template-rows: 100px 100px;
  grid-auto-columns: 100px;
}
```

Sellele võrestikule paigutame kaks grid-elementi, ühe selliselt, et nõutud veeru-jooni pole olemas (viies veerg), mis ise ja sellele eelnevad kaks lisatud tühja veergu saavad kõik laiuseks 100px:

```
.item2 {
  grid-column: 5 / 6;
  grid-row: 2 / 3;
}
```



Joonis 105 Automaatselt lisatud veerud laiusega 100px



Joonis 106 Automaatselt lisatud veerud automaatselt valitud laiusega

Sama moodi saab vajadusel automaatselt ridu lisada, näiteks määrame automaatselt lisatavate ridade kõrguseks 100px, lisame grid-konteineri kujundusse rea:

```
grid-auto-rows: 100px;
```

paigutame ühe grid-elemendi kolmandasse ritta, mida pole olemas:

```
item2 {
  grid-column: 5 / 6;
  grid-row: 3 / 4;
}
```

Joonis 107 Automaatselt lisatud 1 rida kõrgusega 100px ja 3 veergu laisega 100px

Elementide automaatne paigutus võrestikul

Nende elementide jaoks võrestikul, mille jaoks pole asukohta määratud, saab kasutada automaatset paigutamist.
Kasutada saab *grid-auto-flow* omadust, millel on järgmised võimalikud väärtused:

- *row* – algoritm üritab järjest ära täita kõik võrestiku read (vaikeväärtus);
- *column* – algoritm üritab järjest täita võrestiku veerud;
- *row dense* – algoritm üritab ridu täita paigutades järgnevaid väiksemaid elemente eespool olevatele tühjadele kohtadele;

- *column dense* – algoritm üritab veergusid täita paigutades järgnevaid väiksemaid elemente eespool olevatele tühjadele kohtadele.

Näiteks:

```
.main-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
  grid-template-rows: auto auto;
  grid-auto-flow: row dense;
}
```

Võrestiku defineerimine ja kujundamine lühidalt

Omadusi *grid-template-rows*, *grid-template-columns*, *grid-template-areas*, *grid-auto-rows*, *grid-auto-columns* ja *grid-auto-flow* saab lühidalt kirja panna omaduse *grid* abil.

Grid-elementide omadused

Grid-elementide omadused määravad, kuidas need elemendid võrestikul paiknevad.

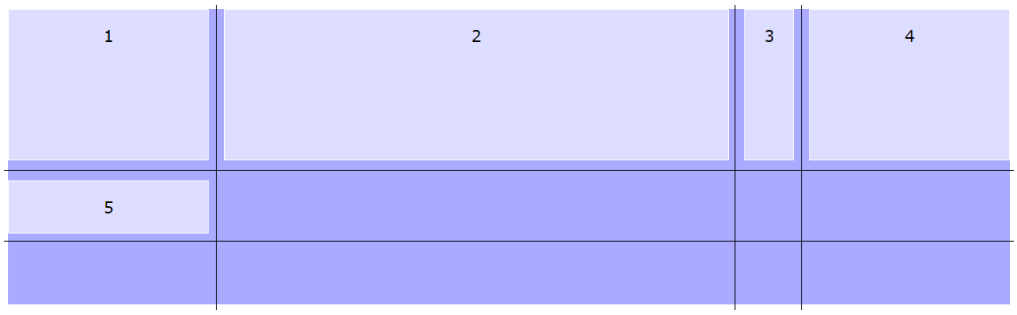
Grid-elementide paigutamine ridadele ja veergudele

Kui grid-elementidele ei määrata, kuidas nad paiknevad, siis paigutatakse nad automaatselt järjest võrestiku ridade ja veergude ristumisel moodustuvatesse „lahtritesse“.

Näiteks on järgmises näites defineeritud sõrestik 4 veeru ja 3 reaga aga HTML koodis vaid 5 grid-elementi.

```
.main-container {
  background-color: #AAAAFF;
  display: grid;
  grid-template-columns: 200px auto 50px 200px;
  grid-template-rows: 150px auto 50px;
  grid-column-gap: 15px;
  grid-row-gap: 20px;
}

.grid-item{
  background-color: rgba(255,255,255,.6);
  border: 1px solid white;  text-align: center;
  font-family: verdana;
  font-size: 1em;
}
```



Joonis 108 Grid-elemendid on paigutatud automaatselt

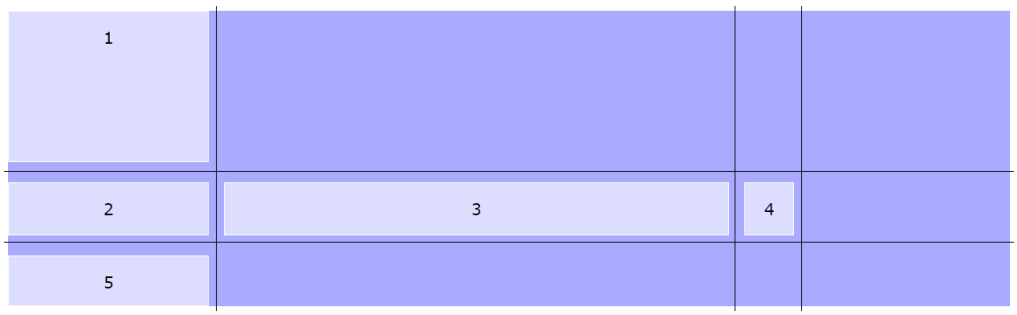
Grid-elemendi määramine veeru-joonte järgi

Määramaks grid-elemendi asukohta ja suurust veeru-joonte (*column-line*) järgi, saab kasutada omadusi *grid-column-start* ja *grid-column-end*. Neil mõlemal saab kasutada järgmiseid väärtuseid:

- joon – veeru-joone number või nimi, millest grid-element algab või millel lõppeb;
- *auto* – määrab automaatse paigutuse ja ulatuse veergude suhtes (vaikeväärtus).

Näiteks on esimesele, teisele ja kolmandale grid-elemendile määratud alustamine esimeselt veeru-joonelt:

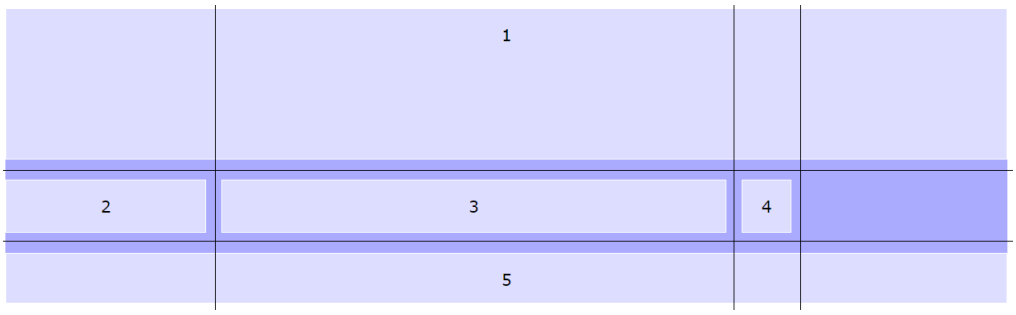
```
.item1,
.item2,
.item5 {
  grid-column-start: 1;
}
```



Joonis 109 Esimene, teine ja kolmas grid-element algavad esimesest veeru-joonest

Näiteks on esimesele ja viiendale elemendile määratud ulatumine üle kõigi veergude:

```
.item1,
.item5 {
  grid-column-start: 1;
  grid-column-end: 5;
}
```



Joonis 110 Esimesele ja viiendale grid-elemendile on määratud ulatumine (*span*) üle nelja veeru

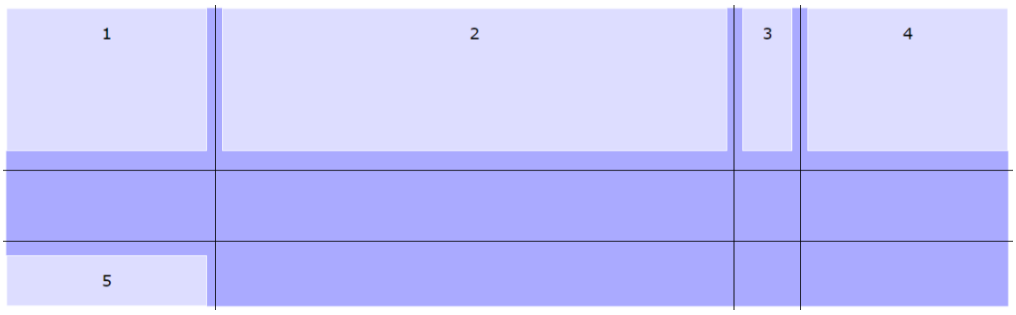
Grid-elemendi määramine rea-joonte järgi

Määramaks grid-elemendi asukohta ja suurust rea-joonte (*row-line*) järgi, saab kasutada omadusi *grid-row-start* ja *grid-row-end*. Neil mõlemal saab kasutada järgmiseid väärtuseid:

- joon – rea-joone number või nimi, millest grid-element algab või millel lõppeb;
- *auto* – määrab automaatse paigutuse ja ulatuse ridade suhtes (vaikeväärtus).

Näiteks on viiendale elemendile määratud alustamine kolmandalt realt ehk neljandalt rea-joonelt.

```
.item5 {
  grid-row-start: 4;
}
```



Joonis 111 Viies grid-element algab neljandalt rea-joonelt

Kompaktsemad omadused elementide rea-joonte ning veeru-joonte järgi paigutamiseks

Grid-elementide paigutamiseks veeru-joonte ja rea-joonte järgi saab kasutada ka kombineeritud omadusi *grid-column* ja *grid-row*, mille väärtustena antakse korraga nii jooned, millelt element algab kui ka jooned, millel element lõppeb. Alguse ja lõpu joone eraldajaks on kaldkriips „/“.

Näiteks elemendid 1 ja 5 paigutuvad eespool toodud joonisel (Joonis 110) ka järgmise stiiliga:

```
.item1,
item5 {
  grid-column: 1 / 5;
}
```

Grid-elementide nimetamine sidumiseks piirkondadega

Veebilehe sisuelemente, mis on grid-elementideks, saab loodud võrestikuga siduda ka nimede kaudu jagades võrestiku piirkondades.
Grid-elementidele nimede defineerimiseks kasutatakse omadust *grid-area*. Selle omaduse väärtuseks on reeglina tekst.

Näiteks defineerime CSS klassid piirkondade jaoks:

```
.area-header {grid-area: page_header;}
.area-menu {grid-area: page_menu;}
.area-content {grid-area: page_content;}
.area-side {grid-area: page_side;}
.area-footer {grid-area: page_footer;}
```

Neid nimetusi kasutatakse võrestiku enda omadustes ridade kaupa piirkondade määramisel.

Alternatiivselt võib omadust *grid-area* kasutada grid-elemendi paigutuse määramiseks veeru-joonte ja rea-joonte järgi lühemal kujul. Sellisel juhul pannakse kaldkriipsudena eraldatult kirja omaduste *grid-row-start*, *grid-column-start*, *grid-row-end* ja *grid-column-end* väärtused.

Näiteks eespool toodud joonisel (Joonis 93) saaks piirkonna „page_content“ siduda sõrestikuga ka nii:

```
.area-content {
  grid-area:2 / 2 /3 /3;
}
```

Grid-elemendi sisu joondamine

Grid-elemendi sisu saab joondada nii rea (*row axis*) telge mööda ehk horisontaalselt kui ka veeru (*column axis*) telge mööda ehk vertikaalselt.

NB! Kõigi võrestikule paigutatud grid-elementide sisu korraga joondamiseks võib kasutada ka grid-konteineri omadusi *justify-items* (horisontaalselt joondamiseks) ning *align-items* (vertikaalseks joondamiseks).

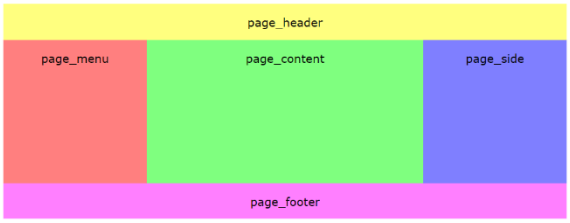
Grid-elemendi sisu joondamine horisontaalselt

Grid-elemendi joondamiseks horisontaalselt kasutatakse omadust *justify-self*, millel on järgmised võimalikud väärtused:

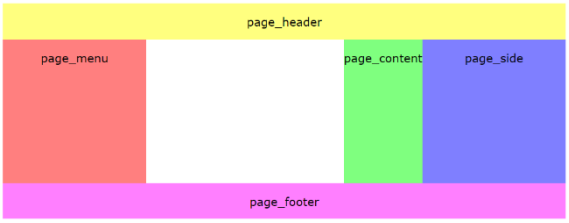
- *start* – sisu joondatakse elemendi vasakusse serva;
- *end* – sisu joondatakse elemendi paremasse serva;
- *center* – sisu joondatakse elemendi keskele;
- *stretch* – sisu täidab terve elemendi sisu (vaikeväärtus).

Näiteks:

```
.area-content {
  grid-area: page_content;
  justify-self: end;
}
```



Joonis 112 Grid-elemendi page_content sisu täidab terve võimaliku laiuse (*justify-self: stretch*)



Joonis 113 Grid-elemendi sisu on joondatud paremasse serva (*justify-self: end*)

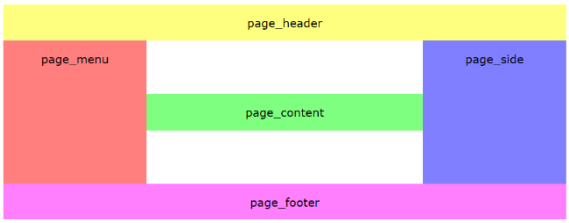
Grid-elemendi sisu joondamine vertikaalselt

Grid-elemendi joondamiseks horisontaalselt kasutatakse omadust justify-self, millel on järgmised võimalikud väärtused:

- *start* – sisu joondatakse elemendi ülemisse serva;
- *end* – sisu joondatakse elemendi alumisse serva;
- *center* – sisu joondatakse vertikaalselt elemendi keskele;
- *stretch* – sisu täidab terve elemendi sisu (vaikeväärtus).

Näiteks:

```
.area-content {
   grid-area: page_content;
   align-self: center;
}
```



Joonis 114 Grid-elemendi sisu on joondatud vertikaalselt keskele (*align-self: center*)

Transformeerimine

CSS3 võimaldab luua mitmesuguseid visuaalefekte, milliseid seni suures osas vaid Flashi, animeeritud GIF piltide vms tehnoloogia abil teha sai! Kasutada on nii 2D kui ka 3D-transformatsioonid.

Läbipaistvus

Tegelikult on kasutatav omadus *opacity* hoopis läbipaistmatus. Väärtuste vahemik on 0 – 1, kus 1 tähendab täielikku läbipaistmatust ja 0 olematut läbipaistmatust ehk läbipaistvust. Näiteks:

```
img:hover {opacity:0.5}
```

Huvitaval kombel arvati see omadus CSS 2.1 standardist välja aga see siiski toimis ja leidis laialt kasutatamist. CSS3 standardse omadusena on see toetatud kõigi veebilehitsejate jaoks ning kadunud on erinev süntaks IE jaoks!

2D Transformatsioonid

Tegemist on ühe huvitavama CSS3 omadusega, mis suudab muuta veebilehe elementide kuju, suurust ja asendit.

Transformeerimiseks on omadus *transform*. Kasutada saab järgmiseid teisendusi:

- *scale* – suurus, väärtuseks kümnendmurrud, väärtus 1 tähistab elemendi originaalsuurust;

NB! Negatiivne väärtus pöörab elemendi peegelpilti nagu tavapäraselt *flip* teisendus arvutigraafikas.

- *rotate* – pöördenurk, väärtused vahemikus 0 – 360, väärtusele lisatakse mõõtühikuna *deg* (lühend nurgakraadist *degree*), kui lähtepunkti (*origin*) pole määratud, siis toimub pööre ümber keskpunkti;
- *translate* – nihe mingis suunas, väärtusteks näiteks pikslid horisontaal ja vertikaalsuunal;
- *skew* – rööpnihe;
- *matrix* – kombineerib kõik eelmised, transformeerib elemendi 6 etteantud väärtuse järgi.

Süntaks on järgmine:

```
transform: scale(X[,Y]) rotate(Xdeg) translate(Xpx, Xpx) skew(Xdeg[, Ydeg]);
```

NB! Määrata võib ühe või mitu erinevat teisendust, eraldajaks on tühik! Seejuures ei pea järjekord vastama eelpool toodule ning tulemus sõltub teisenduste järjekorrast.

Huvitav on see, et *transform* rakendamisel ei mõjutata ülejäänud veebilehe elemente, ehk transformeeritud element hõivab endiselt oma originaalse suuruse ja asukohaga ala.

Lisaks saab määrata teisenduse lähtekoha, ehk siis punkti, mille suhtes transformatsioon toimub, selleks on omadus *transform-origin*, millel määratakse kaks tühikuga eraldatud väärtust – lähtekoht x-teljel ja lähtekoht y-teljel.

- X-telje jaoks on kasutada järgmised võimalikud väärtused:
  - *left* – lähtekohaks horisontaalselt vasak serv;
  - *center* – lähtekohaks horisontaalselt keskkoht;
  - *right* – lähtekohaks horisontaalselt parem serv;
  - arvväärtus pikslites alustades vasakust servast;
  - protsendid elemendi laiuse suhtes (alustatakse vasakult).
- Y-telje jaoks on kasutada järgmised võimalikud väärtused:
  - *top* – lähtekohaks vertikaalselt ülemine serv;

- *center* – lähtekohaks vertikaalselt keskkoht;
- *bottom* – lähtekohaks vertikaalselt alumine serv;
- arvväärtus pikslites alustades ülaservast;
- protsendid elemendi kõrguse suhtes (alustatakse ülevalt).

NB! Mõlema telje suhtes on vaikeväärtusena 50%!

Näiteks:

```
div {
  transform-origin: 50% 50%;
  transfom: scale(.5) rotate(45deg) skew(30deg 0deg);
}
```

3D transformatsioonid

Sarnaselt 2D transformatsioonidele on olemas ka 3D transformatsioonid. 3D puhul on lisaks teisendustele vaja kasutada ka omadusi, mis määravad, kuidas me objekte ruumis näeme.

3D teisendused

Pakutavad teisendused on:

- *scaleX* – suurus X-telje suunal, väärtuseks kümnendmurrud, väärtus 1 tähistab elemendi originaalsuurust;
- *scaleY* – suurus Y-telje suunal, väärtuseks kümnendmurrud, väärtus 1 tähistab elemendi originaalsuurust;
- *scaleZ* – suurus Z-telje suunal, väärtuseks kümnendmurrud, väärtus 1 tähistab elemendi originaalsuurust;

NB! Sarnaselt 2D transformatsioonidele on ka siin negatiivne väärtus peegeldus (*flip*).

- *rotateX* – pööre ümber X-telje;
- *rotateY* – pööre ümber Y-telje;
- *rotateZ* – pööre ümber Z-telje, põhimõtteliselt sama, mis 2D transformatsioonide rotate.

Sarnaselt 2D transformatsioonidele on pööramise puhul väärtused vahemikus 0 – 360, väärtusele lisatakse mõõtühikuna *deg* (lühend nurgakraadist *degree*). Kui lähtepunkti (*origin*) pole määratud, siis toimub pööre ümber keskpunkti.

- *translateX* – nihe X-telje suunal;
- *translateY* – nihe Y-telje suunal;
- *translateZ* – nihe Z-telje suunal (väärtused kasvavad vaataja suunas);
- *skewX* – rööpnihe X-telje suunal;
- *skewY* – rööpnihe Y-telje suunal;
- *skewZ* – rööpnihe Z-telje suunal;

Kõiki eespool loetletud transformatsioone kombineerib väärtus:

- *matrix3d*(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) – defineerib 3D transformatsiooni 4X4 maatriksi abil.

Täpselt samuti nagu 2D transformatsioonide puhul, saab määrata punkti, mille suhtes teisendused toimuvad. Kasutada on sama omadus *transform-origin*, millel antud juhul määratakse kolm tühikuga eraldatud väärtust – lähtekoht x-teljel, lähtekoht y-teljel ja lähtekoht z-teljel.

- X-telje jaoks on kasutada järgmised võimalikud väärtused:
  - *left* – lähtekohaks horisontaalselt vasak serv;
  - *center* – lähtekohaks horisontaalselt keskkoht;
  - *right* – lähtekohaks horisontaalselt parem serv;
  - arvväärtus pikslites alustades vasakust servast;
  - protsendid elemendi laiuse suhtes (alustatakse vasakult).
- Y-telje jaoks on kasutada järgmised võimalikud väärtused:
  - *top* – lähtekohaks vertikaalselt ülemine serv;
  - *center* – lähtekohaks vertikaalselt keskkoht;
  - *bottom* – lähtekohaks vertikaalselt alumine serv;
  - arvväärtus pikslites alustades ülaservast;
  - protsendid elemendi kõrguse suhtes (alustatakse ülevalt).
- Z-telje jaoks on kasutada vaid üks variant väärtuseid:
  - arvväärtus pikslites, eemaldudes väärtused kahanevad;

NB! Vaikeväärtusteks on 50% 50% 0!

Näiteks:

```
div {
  transform-origin: 50% 50% 150px;
  transfom: rotateY(45deg);
}
```

3D vaate omadused

Tulenevalt ruumiliste teisenduste iseärasustest on nende puhul kasutada ka omadused, mis määravad, kuidas me elemente ruumis näeme.

Ruumilise asendi näitamine

3D teisenduste puhul on võimalik määrata, kas mingi kindla elemendi tütarelemendid hoiavad oma asendit 3D ruumis või mittes.
Selleks on omadus *transform-style*, millel on järgmised võimalikud väärtused:

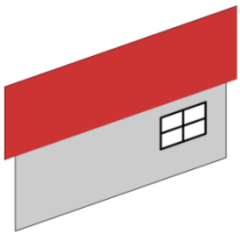
- *flat* – elemendi tütarelemendid ei hoia oma 3D asendit;

- *preserve-3d* – elemendi tütarelemendid hoiavad oma 3D asendit.

Näiteks:

```
div {
  transform-style: preserve-3d;
  transfom: rotateY(45deg);
}
```

Joonis 115 Tütarelemendid säilitavad asendi ruumis (*preserve-3d*)



Joonis 116 Tütarelemendid ei säilita asendit ruumis (*flat*)

Perspektiiv

3D teisenduste puhul saab määrata ka perspektiivi ehk kaugust vaatajast. Selleks kasutatakse omadust *perspective*, millel on kasutada järgmised väärtused:

- arvväärtus pikslites;
- *none* – samaväärne arvuga 0, perspektiiv on määramata, see on ka vaikeväärtus.

Näiteks:

```
div {
  transform-style: preserve-3d;
  perspective 500px;
}
```

Joonis 117 perspektiiv on määramata

Joonis 118 Perspektiivi väärtuseks 500px

NB! Selle omaduse määramisel mõjutatakse elemendi tütarelemente!

Vaatepunkt

Määrata saab ka punkti, mille suhtes elemente vaadatakse. Selleks on omadus *perspective-origin*, millele tuleb määrata tühikutega eraldatud väärtused x-telje ja y-telje suhtes.

- X-telje jaoks on kasutada järgmised võimalikud väärtused:

- *left* – vaatekoht horisontaalselt vasak serv;
- *center* – vaatekohaks horisontaalselt keskkoht;
- *right* – vaatekohaks horisontaalselt parem serv;
- arvväärtus pikslites alustades vasakust servast;
- protsendid elemendi laiuse suhtes (alustatakse vasakult).

- Y-telje jaoks on kasutada järgmised võimalikud väärtused:
  - *top* – vaatekohaks vertikaalselt ülemine serv;
  - *center* – vaatekohaks vertikaalselt keskkoht;
  - *bottom* – vaatekohaks vertikaalselt alumine serv;
  - arvväärtus pikslites alustades ülaservast;
  - protsendid elemendi kõrguse suhtes (alustatakse ülevalt).

NB! Mõlema telje suhtes on vaikeväärtusena 50%!

Joonis 119 Vaatepunkt vaikimisi

Joonis 120 Vaatepunkt 10% 20%

Joonis 121 Vaatepunkt 90% 20%

Elemendi nähtavus tagantpoolt vaadates

Määrata saab ka seda, kas elemendid paistavad ka tagantpoolt vaadatuna (kui neid ruumis pöörata). Omadusel *backface-visibility* on järgmised väärtused:

- *visible* – element on nähtavad ka tagantpoolt, vaikeväärtus;
- *hidden* – element pole tagantpoolt nähtav.

Näiteks:

```
div img {backface-visibility: hidden}
```

Joonis 122 Maja otsaseinal *backface-visibility* väärtuseks *hidden*

Animeerimine

CSS3 võimaldab ka animeerimist. See tähendab, et mingite veebilehe elementide omadusi saab mingite kindlate reeglite järgi teatud ajavahemiku jooksul muuta. Nii saab luua näiteks lihtsamaid bännereid.
Animatsioone võib CSS abil luua kaht moodi:

- lastes mingite omaduste väärtuseid muutuda mingi aja jooksul – siirded (*transition*), mis on sobilik lühikeste animatsioonide, näiteks kasutajaliidese efektide loomiseks;
- kasutades võtmekaadreid, pikemate ja/või tsükliliselt korduvate animatsioonide loomiseks.

Siire (*transition*)

Uue võimalusena saab luua sujuvaid üleminekuid muudetavate omaduste erinevate väärtuste vahel. Nii saab luua lausa animatsioone.
Ülemineku (*transition*) jaoks määratakse kasutatav aeg ja kiirendusfunktsioon (*ease*).

Omadus, mille siire määratakse

Määramaks omadust, mille ülemineku muutumist mõjutatakse, on omadus *transition-property*.

Seda omadust saab kasutada väga paljude omaduste puhul alates taustavärvist lõpetades mõõtmete, transformatsioonide jms!
Kui soovitakse ülemineku kiirust määrata kõigile võimalikele omadustele, siis kasutatakse omadusele viitamiseks sõna *all* (kõik)!
Korraga võib määrata ka mitu omadust, siis tuleb need lihtsalt komadega eraldada.

Näiteks:

```
div {transition-property: width, background-color}
```

Omadused, mida saab *transition* omadusega kasutada kasutada ehk animeerida on loetletud aadressil: http://www.w3schools.com/cssref/css_animatable.asp

Siirde kestus ja viivitus

Määramaks, kui kaua võtab määratud omaduse muutumine aega, kasutatakse omadust *transition-duration*, mille väärtuseks on aeg sekundites (mõõtühikuna lisatakse s).
Näiteks:

```
div {transition-duration: 3s}
```

Üleminek ei pea toimuma kohe, võib määrata ka viivituse enne alustamist, selleks on omadus *transition-delay*, mille väärtuseks on aeg sekundites (mõõtühikuna lisatakse s).
Näiteks:

```
div {transition-delay: 3s}
```

Siirde kiirendusfunktsioon

Üleminek ei pea toimuma ühtlase kiirusega, saab kasutada omadust *transition-timing-function*, millel on järgmised võimalikud väärtused:

- *ease* – (vaikeväärtus) üleminek algab aeglaselt, seejärel kiireneb ja lõppeb taas aeglaselt (vastab väärtusele *cubic-bezier*(0.25,0.1,0.25,1));
- *linear* – ühtlane kiirus (vastab väärtusele *cubic-bezier*(0,0,1,1));
- *ease-in* – aeglane algus, kiirenev (vastab väärtusele *cubic-bezier*(0.42,0,1,1));
- *ease-out* – aeglustuv (vastab väärtusele *cubic-bezier*(0,0,0.58,1));
- *ease-in-out* – alguses kiirenev, lõpus aeglustuv ((vastab väärtusele *cubic-bezier*(0.42,0,0.58,1));
- *cubic-bezier*(n,n,n,n) – kohandatud kiirusega, võimalikud väärtused vahemikus 0 … 1.

Näiteks:

```
div {transition-timing-function: ease-in-out}
```

Siirde määramine lühendatult, kasutamine

Neid nelja omadust võib kompaktselt koos kasutada, selleks on omadus *transition*, mille väärtuseks kirjutatakse tühikutega eraldatuna eelnevalt kirjeldatud omaduste väärtused järgmises järjekorras:

```
transition: mõjutatav_omadus aeg kiirendus viivitus_enne_muudatust
```

Korraga saab määrata mitut üleminekut, selleks tuleb need komadega eraldatult kirja panna. Näiteks:

```
transition: background-color 4s ease-in-out 0s, width 2s linear 1s;
```

NB! Kuna tegemist on üleminekuga kahe väärtuse vahel, siis on vaja mingisugust käivitumist, näiteks :*focus* või :*hover* sündmus (pseudoklass).

Näiteks:

```
#moonduvpilt:hover {
  transform-origin: 50% 50%;
  transfom: scale(.5) rotate(45deg) skew(30deg 0deg);
}
```

Võtmekaadrid

Animatsiooni loomiseks tuleb kõigepealt luua võtmekaadrid, mis määravad muutuvad omadused ja muutuste suurused ning seejärel seotakse need võtmekaadrid mingi veebilehe elemendiga.

Omadused, mida saab võtmekaadrites kasutada ehk animeerida on loetletud aadressil: http://www.w3schools.com/cssref/css_animatable.asp

Võtmekaadrid (*keyframes*) on animatsiooni töövahendid, millede abil fikseeritakse muudetavad omadused ja muutuste suurus erinevatel ajahetkedel.

Võtmekaadrite määramiseks kasutatakse *@keyframes* reeglit kujul:

```
@keyframes animatsiooni_nimi {
  võtmekaadrid
}
```

Võtmekaadrid seejuures on omaette CSS stiilid, kus selektoriks on võtmekaader (animatsiooni positsioon) ning sellele järgnevad muudetavad omadused ja nende väärtused antud hetkel. Võtmekaadri selektoriteks võivad olla väärtused 0% kuni 100% või siis *from* (sama mis 0%) ja *to* (sama mis 100%).

NB! Võtmekaadrite määramisel võib kasutada protsentuaalseid väärtuseid sajandiku täpsusega (kaks kohta pärast koma)!

Näiteks loome võtmekaadrid, mis muudavad elemendi taustavärvi alustades punasest, minnes üle roheliseks ja lõpetades sinisega:

```
@keyframes changebackcolor {
  0% {background-color: red;}
  50% {background-color: green;}
  100% {background-color: blue;}
}
```

Korraga võib muutuda mitu erinevat omadust, need tuleb võtmekaadrites CSS-ile tavapärasel moel semikoolonitega eraldatult kirja panna.

Näiteks loome võtmekaadrid, mis lisaks taustavärvile liigutavad elementi tema esialgsest asukohast alguses 200 pikslit paremale, siis 200 pikslit alla, siis 200 pikslit vasakule ja lõpuks esialgsele kohale tagasi:

```
@keyframes changebackcolor_move {
  0% {background-color: red; transform: translate(0px,0px);}
  25% {background-color: yellow; transform: translate(200px,0px);}
  50% {background-color: green; transform: translate(200px,200px);}
  75% {background-color: cyan; transform: translate(0px,200px);}
  100% {background-color: blue; transform: translate(0px,0px);}
}
```

Animatsioon

Määratud võtmekaadrid seotakse veebilehe elemendiga kasutades järgnevalt kirjeldatavaid omadusi.

NB! Selline lähenemine võimaldab ühte ja sama komplekti võtmekaadreid kasutada erinevatel veebilehe elementidel (ka erinevate seadetega).

Võtmekaadrite sidumine elemendiga ja animatsiooni kestus

Kasutatavad võtmekaadrid ehk võtmekaadritel määratud animatsiooni nimi määratakse omadusega *animation-name*. Selle väärtuseks on soovitud võtmekaadrite juures määratud nimetus.

Näiteks:

```
div {animation-name: pan}
```

Animatsiooni kestuse määrab omadus *animation-duration*. Selle omaduse väärtuseks on arv sekundites (mõõtühikuna lisatakse s), vaikimisi on väärtuseks 0s.

Näiteks:

```
div {animation-duration: 4s}
```

Animatsiooni kiirendusfunktsioon

Sarnaselt transformatsioonidele, saab määrata, kas on tegemist ühtlase kiirusega toimiva animatsiooniga või toimub ka kiirenemine, aeglustumine. Selleks on omadus *animation-timing-function*, millel on järgmised võimalikud väärtused:

- *ease* – (vaikeväärtus) üleminek algab aeglaselt, seejärel kiireneb ja lõppeb taas aeglaselt (vastab väärtusele *cubic-bezier*(0.25,0.1,0.25,1));
- *linear* – ühtlane kiirus (vastab väärtusele *cubic-bezier*(0,0,1,1));
- *ease-in* – aeglane algus, kiirenev (vastab väärtusele *cubic-bezier*(0.42,0,1,1));
- *ease-out* – aeglustuv (vastab väärtusele *cubic-bezier*(0,0,0.58,1));
- *ease-in-out* – alguses kiirenev, lõpus aeglustuv ((vastab väärtusele *cubic-bezier*(0.42,0,0.58,1));
- *cubic-bezier*(n,n,n,n) – kohandatud kiirusega, võimalikud väärtused vahemikus 0 … 1.

Näiteks:

```
div {animation-timing-function: ease-in-out}
```

Viivitus enne animatsiooni algust

Määramaks viivitust enne animatsiooni käivitumist (muidu käivitub kohe veebilehe laadimise järel), kasutatakse omadust *animation-delay*, mille väärtuseks on arv sekundites (mõõtühikuna lisatakse s), vaikimisi on väärtuseks 0s.

Näiteks:

```
div {animation-delay: 5s}
```

Viivitust saab kasutada näiteks mitmele elemendile identse animatsiooni rakendamisel, kui tahetakse, et elementide animatsioonid kulgeksid üksteise suhtes väikeses ajalises nihkes.

Animatsiooni kordamine

Animatsiooni võib panna korduma, selleks on omadus *animation-iteration-count*, millel on järgmised võimalikud väärtused:

- täisarv – määra, mitu korda animatsiooni esitatakse, vaikeväärtuseks 1;
- *infinite* – animatsiooni korratakse lõpmatult.

Näiteks:

```
div {animation-iteration-count: 6}
```

Animatsiooni kulgemise suund

Animatsiooni võib esitada otsast lõpuni, edasi tagasi või näiteks tagurpidi. Suuna määramiseks kasutatakse omadust *animation-direction*, millel on järgmised võimalikud väärtused:

- *normal* – animatsioon esitatakse tavapäraselt algusest lõpuni (vaikeväärtus);
- *reverse* – animatsioon esitatakse tagurpidi lõpust alguseni;
- *alternate* – animatsiooni esitatakse vaheldumisi õigetpidi ja tagurpidi ehk paaritutel kordadel õigetpidi;
- *alternate-reverse* – animatsiooni esitatakse vaheldumisi kuid alustatakse tagurpidi esitamisest ehk paariarvulistel kordadel õigetpidi.

Näiteks:

```
div {animation-direction: alternate}
```

Elemendi omadused enne ja pärast animatsiooni

Kui animatsioon lõppeb, siis vaikimisi taastub animatsiooni eelne olukord ehk kõik elemendid liiguvad tagasi algsetele kohtadele, taastuvad esialgsed värvid jne. Määramaks, mis toimub enne animatsiooni käivitumist või pärast selle lõppu, on omadus *animation-fill-mode*, millel on järgmised võimalikud väärtused:

- *none* – animatsioon ei määra veebilehe elemendile mingeid omadusi enne ega pärast esitust (vaikeväärtus);
- *forwards* – pärast animatsiooni lõppemist rakendatakse veebilehe elemendile need omadused, millega animatsioon lõppes (viimase võtmekaadri seaded);
- *backwards* – veebilehe elemendile rakendatakse juba enne animatsiooni algust (viivituse (*delay*) ajal) need omadused, millega algab esimene animatsiooni kordus;
- *both* – veebilehe elemendile rakendatakse enne ja pärast animatsiooni omadused, mis määratakse animatsiooni poolt vastavalt alustades ja lõpetades.

Näiteks:

```
div {animation-fill-mode: both}
```

Animatsiooni olek

Animatsiooni saab mängimise ajal ka peatada (paus), selleks kasutatakse omadust *animation-play-state*, mille väärtusteks on:

- *running* – animatsiooni esitatakse (vaikeväärtus);
- *paused* – animatsiooni esitus on peatatud.

NB! Omaduse *animation-play-state* väärtuse muutmiseks on vaja mingit sündmust (näiteks :*hover* pseudoklass) ning sellele reageerimist (näiteks JavaScript'i abil).

Näiteks:

```
img:hover {animation-play-state: paused}
```

Animatsiooni määramine ühe omadusena

Animatsiooni võib kirja panna ka lühidalt, ühe omadusega *animation*, mille väärtuseks tuleb tühikutega eraldatult loetleda eelnevalt kirjeldatud omaduste väärtused:

```
animation: name duration timing-function delay iteration-count direction fill-mode play-state;
```

Näiteks:

```
div {animation: pan 3s ease-in-out 2s 4 alternate}
```

NB! Kui mõne omaduse väärtust kirja ei panda, siis kasutatakse selle vaikeväärtust (eelmisel näitel *fill-mode* ja *play-state*).

Mitme animatsiooni korraga kasutamine

CSS3 lubab ühel ja samal objektil kasutada korraga ka mitut animatsiooni. Selleks tuleb luua eraldi võtmekaadrid ja siis animatsioonid ühe omadusena, komadega eraldatult kirja panna.

Näiteks:

```
.animated_object {
  animation: sway 2s ease infinite alternate, trip 10s linear infinite;
}
@keyframes sway {
  0%{transform: rotate(-5deg);}
  100%{transform: rotate(5deg);}
}
@keyframes trip {
  0%{left: 169px;}
  100%{left: 550px;}
}
```

NB! Jälgida tuleb, et erinevates animatsioonides ei kasutataks sama omaduse muutmist, sest siis rakendatakse selle omaduse väärtust viimasest animatsioonist!

Animatsiooni näide

Näiteks paigutame veebilehele pildi kõrgusega 200 pikslit ja laiusega 1200 pikslit, paigutame selle ühe elemendi sisse, mille laiuseks määrame 600 pikslit (pool pildi laiust) ning peidame üle serva ulatuva sisu (et pildist vaid pool paistaks). Paneme pildi edasi tagasi liikuma (600 pikslit vasakule), et kõike näha, lõpetame 5 korra järel (pildist jääb naha pool, mis algselt oli varjatud). Kui hiirega pildile liikuda (:*hover*) animatsioon peatub.
HTML koodis:

```
<div class="stage">
  <img src="panoraampilt.jpg" alt="panoraam" class="panoramic">
</div>
```

CSS koodis:

```
/*div jaoks*/
.stage {
   width: 600px;
   height:200px;
   overflow:hidden;
}

/*pildi jaoks*/
.panoramic {
   animation: pan 5s ease-in-out 2s 5 alternate forwards running;
}


.panoramic:hover {
   animation-play-state: paused;
}

/*animatsiooni võtmekaadrid*/
@keyframes pan {
   0% {transform: translate(0px,0px);}
   100% {transform: translate(-600px,0px);}
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/counters
https://www.w3schools.com/css/css_pseudo_elements.asp