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 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!

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!

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.

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

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

Elemendi nimega 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:

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

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

Kujunduse määramisel tuleks eelistada klasside kasutamiset!

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 ühele elemendile rakendada ka **mitme klassi kujundused**, 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 <1>):

```
h1+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 kolmandataseme 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 sulgdesse:

```
[atribuut_ja_selle_väärtus_vms]
```

- Kujundust võib luua vastavalt mõne atribuudi olemasolule.

Näteks 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“ (lang=“en“ – tekst on inglisekeelne):

```
[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 „id“ 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(e)
{
  font-style: italic;
}
```

Pseudoklassid/pseudoelemendid

Pseudoklasse ehk pseudoelemente kasutatakse mõningatele selektoritele eriefektide lisamiseks. Nende puhul lisatakse tavapärasele selektorile kooloni järele veel nn pseudonimi.

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).

Nendest mainitutest saab pseudonime :hover kasutada ka teistel elementidel ning sedasi luua eriefekte hiirega elementide liikumisel.
Olemas on ka pseudoklass, mis viitab mngi 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: gray;
}
```

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;
}
```

- On ka võimalus kujundada veebilehe osa, mille kasutaja on valinud (*selected*). Selleks on pseudoklass ::selection. Näiteks:

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

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 lõiguletaandrea:

```
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*) saab 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;
}
```

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;

Esimese tekstirea ja tähemärgi pseudoklassid

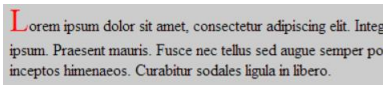
Teksti kujundamisel saab api võtta pseudoklassid, mis viitavad esimesele tähemärgile või esimesele tekstireale. Nendeks pseudoklassideks on: :first-line ja :first-letter.
Näteks loome kujunduse kõikide lõikude esimestele tähemärkidele:

```
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 erladi kujundatud

Pseudoklassid lisamaks sisu elementide ette ja järele

Selleks, et lisada mingit kindlat sisu määratud elementide ette või järele on pseudoklassid :before ja :after.
Sisu lisatakse omadusega content.
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;
}
```

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;
}
```

CSS värvid

Värvid pannakse CSS või ka HTML keeles tavaliselt kuusteistkümmendsüsteemi (*hexa-decimal* ehk HEX) (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 kus iga numbri ja/või tähepaar määrab RGB (*red, green, blue*) värvimudeli vastava värvikomponendi väärtuse.
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.

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

HEX	RGB	nimetus (name)	
#000000	rgb(0,0,0)	*black*	must
#FFFFFF	rgb(255,255,255)	*white*	valge
#FF0000	rgb(255,0,0)	*red*	punane
#808080	rgb(128,128,128)	*gray*	hall

Näiteks:

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

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*.

CSS3 lisab võimalusi värvide kasutamisel!

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

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

Kasutusele on võetud ka HSL värvid, 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%);
}
```

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

Värvikonstantide nimed leiab aadressil: http://w3schools.com/cssref/css_colornames.asp

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 käsk (*rule*)!

```
@import "teinecssfail.css";
```

NB! See käsk on kasulik paigutada CSS-faili kõige esimesele reale!

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
}
```

*Astmiktäide (*gradient*)*

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

Kasutada saab lineaarset (*linear*), radiaalset (*radial*) ja elliptilist (*elliptical*) astmiktäidet.

Lineaarse astmiktäite süntaks (selline toimib Mozilla Firefoxi puhul (prefiks –moz-):

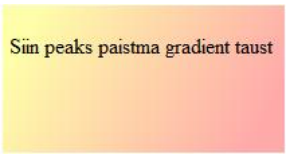
```
background:linear-gradient(pos, #AAA B, #XXX Y);
```

- pos – esimese värvi positsioon;
- #AAA = algusvärv;
- B = gradiendi algus (%)
- #XXX = lõppvärv;
- Y = gradiendi lõpp (%)

Webkit realisatsioon on pisut erineva süntaksiga kuid töötab samal moel!

Näiteks:

```
section
{
  background: -moz-linear-gradient(left top, #ffffaa 10%, #ffaaaa 90%);
  background: -webkit-gradient(linear, left top, left bottom, from(#ffffaa), to(#ffaaaa));
}
```



Joonis 4 Lineaarne astmiktäitega taust

Sarnaselt luuakse ka radiaalne astmiktäide. Sel puhul määratakse keskpunkt.
Näiteks:

```
section
{
  background: -moz-linear-gradient(50% 50%, farthest side, #ffffaa, #ffaaaa);
  background: -webkit-gradient(radial, 50% 50%, 0 , 50% 50%, 350, from(#ffffaa), to(#ffaaaa));
}
```

Joonis 5 Radiaalne astmiktäitega taust

Astmiktäite loomine on mugavam mõnda veebipõhist tööriista kasutades, näiteks: CSS *Gradient Background Maker* (http://ie.microsoft.com/testdrive/graphics/cssgradientbackgroundmaker/default.html).

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 koordinaatidepaar 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}
```

Huvitava võimalusena saab määrata, kas taustapilti keritakse (*scroll*) veebilehitseja aknas koos ülejäänud veebilehe sisuga või püsib ta fikseeritud kohas paigal. Selleks on omadus *background-attachment*, millel on kaks võimalikku väärtust: *scroll* – pilti keritakse; *fixed* – pilt püsib ühel kohal.
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 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 6 Taustapildi suurus protsentuaalselt: laius 100%, kõrgus 100%

Joonis 7 Taustapildi suurus: *contain*

Joonis 8 Taustapildi suurus: *cover*

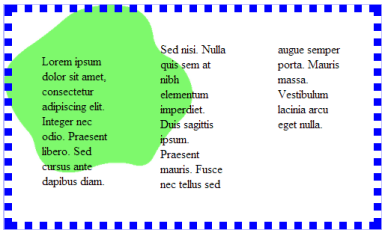
Taustapildi koordinaatide alguspunkt

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 9 Tastapildi asukoht raamjoone suhtes (*border-box*)

Joonis 10 Tastapildi asukoht polsterduse suhtes (*padding-box*)

Joonis 11 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 12 Taust katab kogu elemendi piirkonna (*border-box*)

Joonis 13 Taust katab sisu ja polsterduse ala, joone alla ei ulatu

Joonis 14 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*/
}
```

Tekst

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:

- ttf – *True Type Fonts*, seda toetavad Firefox, Opera, Safari ja Google Chrome;
- otf – *OpenType Fonts*, seda toetavad Firefox, Opera, Safari ja Google Chrome;

- eot – *Embedded OpenType*, seda toetab Internet Explorer;
- woff – *Web Open Font Format*;
- svg,svgz – *scalable-vector-graphic*.

Kui tarvis kasutada erinevate atribuutidega (*bold*, *italic*) kirju ja Teil on vastava kujundusega spetsiaalne font olemas, siis tuleb lisada veel teinegi samasugune @font-face selektor, 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 (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 eraldajaks kirjutada punkt!

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

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

Kaldkiri

Loomulikult saab määrata, kas tegemist on kaldkirjaga või mitte. Selleks kasutatakse omadust *font-style*, millel on tavapärasteks väärtusteks on: *normal* (vaikeväärtus), *italic* (kaldkiri) või *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}
```

Alla joonimine, läbikriipsutamine

Võimalik on ka teksti allajoonida või läbi kriipsutada. Selleks kasutatakse teksti dekoratsioone, millede määramiseks on omadus *text-decoration* ning mille võimalikud väärtused on *none* (vaikeväärtus), *underline*, *overline* ja *line-through*. Näiteks määrame suurtele pealkirjadele allajoonimise:

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

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.

```
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 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*.

Nummerdatud loendi puhul (element <ol>) saab selle omaduse väärtustena kasutada:

- *inherit* – loend pärib vastava omaduse teda sisaldavalt elemendilt);
- *decimal* – see on ka vaikeväärtus;
- *decimal-leading-zero* – numbritele lisatakse polsterdusena ette 0;
- *lower-roman*, *upper-roman* – rooma numbrid väike- või suurtähtedega;
- *lower-alpha*, *upper-alpha*, *lower-greek*, *lower-latin*, *upper-latin* – erinevad tähemärgid;
- *armenian* – traditsiooniline armeenia numeratsioon;
- *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}
```

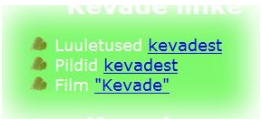
Täpploendi puhul (element <ul>) saab kasutada väärtuseid: *disc* (vaikeväärtus), *circle*, *square*, *none* (täpid puuduvad).

Näiteks:

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

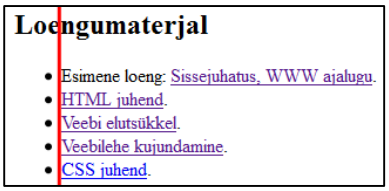
Täpploendi puhul saab täppidena kasutada ka pildifaili. Selleks kasutatakse omadust *list-style-image*, mille väärtuseks on soovitud pildifaili URL.
Näiteks kasutame täpploendis tavaliste täppide asemel pildifaili "leht.png":

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

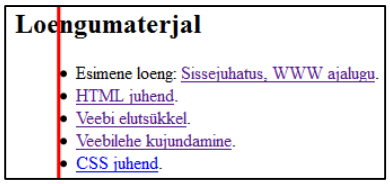


Joonis 15 Pilt järjestamata loendi täpina

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* (vaikeväärtus), *inside* ja *inherit* (omadus päritakse loendit sisaldavalt elemendilt). Näiteks:



Joonis 16 Loendielemendi marker väljaspool sisu piirkonda (*outside*)



Joonis 17 Loendielemendi marker sisu piirkonnas (inside)

Tekstiefektid

CSS3 pakub lisaks tavapärastele teksti vormindamisvahenditele ka efekte.

Teksti vari

Nüüd pole enam tekstile varju lisamiseks vaja kasutada fototöötluse abi ning teksti veebilehele pildinba 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;
}
```

Sellel tekstil on punane vari!

Joonis 18 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 *ellipsis* – teksti jätkumise tunnuseks traditsioonilised 3 punkti, *clip* – viimane sõna jääb poolikuks, *string* – näidatakse etteantud teksti.
Näiteks:

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

Teksti „murdmine“

Microsofti ettepanekuna on CSS3-le lisatud ka teksti murdmine (*word wrap*) mis võimaldab pikki sõnu vajadusel ridade vahel poolitada! Selleks ongi omadus *word-wrap*, millel on võimalikud väärtused *normal* ja *break-word*.
Näiteks:

```
p
{
  word-wrap: break-word;
}
```

Tekst mitmes veerus

CSS3 lisab võimaluse teksti ajakirjanduslikus stiilis veergudesse (*columns*) paigutada!
Seda saab teha kahel moel:

- andes ette veergude laiuse *column-width*, väärtus pikselites, protsentides, em-ides jms;
- või soovitud veergude arvu *column-count*, väärtuseks positiivne täisarv.

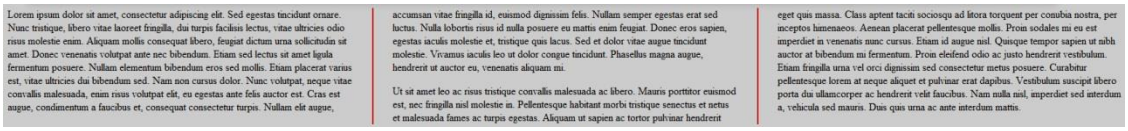
Mõlemal juhul saab määrata veel omadusi:

- *column-gap* – veergude vahekaugus (väärtuseks positiivne arv pikselites, em-ides jms);
- *column-rule* – veergusid eraldav vertikaalne joon (väärtuseks tüüpiline *border* omadus).

NB! Need võimalused on realiseeritud Mozilla ja Safari veebilehitsejatel!

Näiteks:

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



Joonis 19 Kolme veergu paigutatud tekst

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 pseudo-elemendid:

- 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*, mille võimalikud väärtused on: *none*, *hidden*, *dotted*, *dashed*, *solid* (vaikeväärtus), *double*, *groove*, *ridge*, *inset* ja *outset*.
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: *border-top-style*, *border-bottom-style*, *border-left-style*, *border-right-style*, *border-top-width*, *border-bottom-width*, *border-left-width*, *border-right-width*, *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:

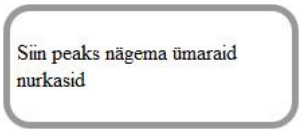
```
table
{border-top: medium solid #ff0000;
border-bottom: medium solid #ff0000;
border-left: medium solid #ff0000;
border-right: medium solid #ff0000;}
```

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 20 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).

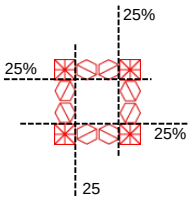
Raamjoon pildiga

CSS3 võimaldab raamjoont luua ka pildi baasil! Selleks kasutatakse omadust border-image, mille süntaks on järgmine:

```
border-image: url("pildi_URL") lõige venitus;
```

Lõige (*slice*) määrab kui palju etteantud pildi servadest jäetakse ülemise, parema, alumise ja vasaku serva jaoks (päripäeva)! Väärtused määratakse protsentides või pikselites. Vastavalt saavad määratud ka nurgad!

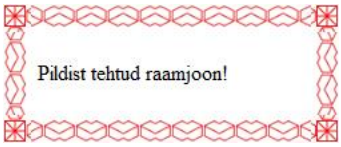
NB! Kui osa või kõik väärtused on ühesugused, siis võib kirja panna ka näiteks ainult kaks või ühe väärtuse!



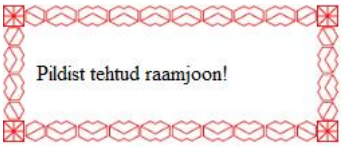
Joonis 21 Pildi lõikumine raamjooneks

Venitus (*stretch*) määrab, kuidas serv pildiosaga täidetakse. Võimalikud väärtused on (ja neid võib määrata iga serva jaoks eraldi):

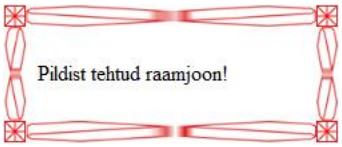
- *round* – pildiosa korratakse ja suurus sobitatakse nii, et mahuks täpselt täisarv kordi;
- *repeat* – pildiosa korratakse ja ülejääv serv lõigatakse maha;
- *stretch* – pildiosa venitatakse üle terve serva (vaikeväärtus).



Joonis 22 Pildist raam, venitus: *repeat*



Joonis 23 Pildist raam, venitus: *round*



Joonis 24 Pildist raam, venitus: *stretch*

Näiteks:

```
div
{
  border-image: url("pildid/raamiks_2.png") 25% round;
}
```

Vari

Üks oodatud uuendus CSS3-l on elementidele varju lisamise võimalus! Selleks kasutatakse omadust *box-shadow*. Määrata tuleb neli parameetrit: horisontaalne nihe (*horizontal offset*), vertikaalne nihe (*vertical offset*), hägustamise raadius (*blur radius*) ja varju värv.

```
div
{
  box-shadow: 15px 10px 5px #555;
}
```

Joonis 25 Varjuga raam

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;}
```

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:

```
img {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.

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. Tavaliselt kasutatakse seda elemendi (alajaotus) jaoks, mille sisse paigutatud objekte niiviisi veebilehel soovikohaselt paigutada saab. Näiteks võib selliselt pildi kindlale kohale paigutada ning fikseerida ta selliselt, et ta püsib kohal isegi veebilehe kerimisel (*scroll*).

Suurus

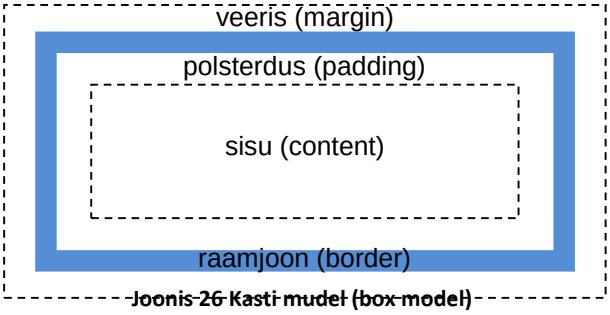
Objekti suuruse määramiseks on omadused *width* (laius) ja *height* (kõrgus), millede väärtusteks on arvud pikselites või protsentides (veebilehitseja akna mõõtude suhtes). Näiteks võime luua piltide jaoks kujundusklassi nimega "pisipilt", mille puhul on piltidel kindlad mõõtmed 100 X 75 pikselit:

```
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 26 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;
```

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*, mille väärtus *hidden* peidab üle servade jääva osa lihtsalt ära ning väärtus auto lisab vajaduse korral kerimisribad (*scrollbar*). Kasutada võib veel väärtus *scroll*, mis lisab kerimisribad.

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}
```

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

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.

Objekti koordinaadid saab kirja panna erinevatest servadest lähtudes: *left*, *right*, *top* ja *bottom*. Näiteks määrame elemendile kujundusklass nimega "paremal" ülemise ja vasaku serva koordinaadid veebilehitseja akna suhtes:

```
div.paremal {position: fixed;
left: 500px;
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;}
```

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*!

Ploki- ja reaelemendid

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

Osa elemente võtavad 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 reaelemente muuta plokielementideks ja vastupidi! Selleks kasutatakse omadust *display*, millel on kaks võimalikku väärtust: *inline* ja *block*!

Näiteks:

```
span {display: block}
```

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;
- *left* – objekt „hõljub“ vasakul pool;
- *right* – objekt „hõljub“ paremal pool;
- *inherit* – objekt pärib vastava omaduse teda sisaldavalt elemendilt (*parent*).

Näiteks:

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

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

- *inherit* – vastav väärtus päritakse objekti sisaldavalt elemendilt (*parent*).

Näiteks:

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

Vertikaalne joondus

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

- *top* – elemendi ülemine serv joondatakse kõrgeima samal real asuva elemendi ülaserva järgi;
- *text-top* – element joondatakse teda sisaldava elemendi (*parent*) teksti fondi ülemise serva järgi;
- *middle* – element joondatakse teda sisaldava elemendi keskele;
- *bottom* – element joondatakse kõige madalama samal real asuva elemendi järgi;
- *text-bottom* – element joondatakse teda sisaldava elemendi (*parent*) teksti fondi alumise serva järgi

Näiteks:

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

Võimalikke väärtuseid on veelgi (http://www.w3schools.com/Css/pr_pos_vertical-align.asp)!

Elemendi nähtamatuks muutmine

Omadusega *visibility* saab määrata, kas objekt on nähtav või mitte. Peitmiseks kasutatakse väärtust *hidden*. Näiteks:

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

NB! Element pole nähtav kuid võtab siiski ruumi!

Kasutada saab veel väärtuseid:

- *visible* – element on nähtav.
- *collapse* – kasutatakse tabelis, kus eemaldab rea või veeru kuid ei mõjuta tabeli struktuuri. Teiste elementide juures kasutades töötab nagu väärtus *hidden*!

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!

Siinkohal tuleb arvestada, et enamus veebilehitsejaid toetab standardset *opacity* omadust aga Internet Explorer ei toeta. Viimase jaoks tuleb kasutada omadust *filter* väärtusega *alpha*, mille arvväärtus on vahemikus 0 - 100!

Seega, et pilti kõikide veebilehitsejate jaoks läbipaistvaks muuta, tuleb stiil kirjutada kujul:

```
img.labipaistev {opacity: 0.5; filter:alpha(opacity=50)}
```

NB! CSS 2.1 standardis pole seda omadust enam määratud! See küll toimib veel aga css valideerimisel annab veateate!

Seda omadust ja html sündmuste atribuute kasutades saab luua ka animatsiooni:

```
<img src="pilt.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
```

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 omadus *table-layout*, mille vaikeväärtuseks *auto*, mille puhul tabeli veergude laius määratakse sisu järgi:

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

Teine võimalik väärtus on *fixed*, mille korral kehtivad autori poolt määratud lahtrite mõõdud või *inherit*, mille puhul see omadus määratakse tabelit sisaldava elemendi poolt.

Tabeli raamjoonte seaded

Saab määrata ka seda, kuidas näidatakse raamjooni. Terve tabeli ja lahtrite jooned võivad olla kokkupandud (ühe joonena):

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

või eraldi nähtavad:

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

Viimasel juhul saab määrata lahtrite ja tabeli raamjoonte omavahelist kaugust (px, cm vms):

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

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

```
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* ja *hide*:

```
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}
```

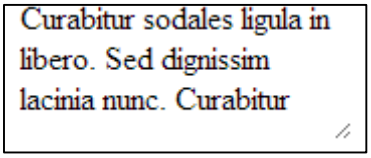
Kasutajaliides

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

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 (order) puhul);
- *outline-width – kontuurjoone paksus.*

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

```
outline: värv stiil paksus
```

Näiteks:

```
p
{
  outline:#DD0000 dotted medium;
}
```

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;
}
```

Kursori kuju määramine

CSS-i abil saab määrata ka seda, kuidas näeb välja kursor elemendile osutades. Näiteks:

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

Kasutada saab järgmiseid väärtuseid: *default* (enamasti nool), *auto* (brauser määrab kursori kuju), *pointer* (käsi), *move*, *e-resize*, *ne-resize*, *nw-resize*, *se-resize*, *sw-resize*, *s-resize*, *w-resize*, *text*, *wait* (liivakell), *progress* (nool liivakellaga), *help* (küsimärk) ja *url* (laseb määrata soovitud erikujulise kursori url-i), *inherit* (omadus päritakse elementi sisaldavalt elemendilt (*parent*)).
NB! Erinevate fraasi "*resize*" sisaldavate väärtuste korral tähistavad tähed ilmakaari: n – põhi, e – ida, s – lõuna ja w – lääs.

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}
```

NB! Mitmed brauserid ei toeta selle omaduse väärtusena url-i!

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

Nummerdamine

Kasutada saab omadust counter
counter-increment:id
counter-increment:id arv – vakimisi suurendaks 1 kaupa aga …

counter-reset:id
counter-reset:id arv – vaikimis alustaks 1-st, kui panna 1, siis kahest jne

Numbri ette lisamiseks tuleb kasutada omadust content, mida kasutatakse koos :before ja :arter pseudoklassidega

Value	Description
none	Sets the content, if specified, to nothing
normal	Sets the content, if specified, to normal, which default is "none" (which is nothing)
counter	Sets the content as a counter
attr*(attribute)*	Sets the content as one of the selector's attribute
string	Sets the content to the text you specify
open-quote	Sets the content to be an opening quote
close-quote	Sets the content to be a closing quote
no-open-quote	Removes the opening quote from the content, if specified
no-close-quote	Removes the closing quote from the content, if specified
url(*url*)	Sets the content to be some kind of media (an image, a sound, a video,

	etc.)
inherit	Specifies that the value of the content property should be inherited from the parent element

Transformeerimine

CSS3 võimaldab luua mitmesuguseid visuaalefekte, milliseid seni suures osas vaid Flashi, animeeritud GIF piltide vms tehnoloogia abil teha sai!

Läbipaistvus

CSS 2.1 standardi juurde mittekuuluv aga siiski toimiv ja laialt kasutatav omadus opacity on saanud CSS3 standardi osaks. Kadunud on erinevused süntaksis IE ja teiste veebilehitsejate jaoks!
Näiteks:

```
img:hover
{
opacity:0.5;
filter:alpha(opacity=50); /* IE8 ja varasemad */
}
```

2D Transformatsioonid

Tegemist on ühe huvitavama CSS3 omadusega, mis suudab muuta veebilehe elementide kuju ja asendit.
Kasutada saab järgmiseid teisendusi:

- *scale* – suurus, väärtused vahemikus 0 – 1;
- *rotate* – pöördenurk, väärtused vahemikus 0 – 360, kui lähtepunkti pole määratud (*origin*), siis toimub pööre ümber vasaku ülanurga;
- *translate* – nihe mingis suunas;
- *skew* – rööpnihe;
- *matrix* – kombineerib kõik eelmised, transformeerib elemendi 6 etteantud väärtuse järgi.

Lisaks saab määrata teisenduse lähtekoha *transform-origin*.

Süntaks on järgmine:

```
transform: scale(x) rotate([x]deg) translate(Xpx, Xpx) skew([x]deg, [y]deg);
transform-origin: X% X%;
```

Näiteks:

```
div
{
  transform-origin: 50% 50%;
  transfom: scale(.5) rotate(45deg) skew(30deg 0deg);
}
```

Üleminekuks kulutatav aeg

Uue võimalusena saab luua sujuvaid animatsioone, tuua efektide loomise juurde ka nende sooritamiseks kuluv aeg!
Kasutada saab omadust transition:

```
transition: mõjutatav_omadus aeg viivitus_enne_muudatust kiirendus
```

Seda omadust saab kasutada väga paljude omaduste puhul alates taustavärvist lõpetades mõõtmete, transformatsioonide jms!
Kiirenduse väärtus võib olla üks järgmistest: *linear* – ühtlane kiirus, *ease* – kiirendusega, *ease-in* – kiirenev, *ease-out* – aeglustuv; *ease-in-out*.
Näiteks:

```
transition: background-color 4s 0s ease-in-out;
```

Korraga saab määrata mitut üleminekut, näiteks:

```
transition: background-color 4s 0s ease-in-out, width 2s 1s linear;
div
{
  transition: width 2s, height 2s, transform 2s;
  -webkit-transition: width 2s, height 2s, -webkit-transform 2s;
}
```

NB! Kuna tegemist on üleminekuga kahe väärtuse vahel, siis on vaja mingisugust käivitumist, näiteks :focus või :hover sündmus.

Näiteks:

```
#moonduvpilt:hover
{
  transform-origin: 50% 50%;
  transfom: scale(.5) rotate(45deg) skew(30deg 0deg);
}
div
{
  transition-property: width;
  transition-duration: 1s;
  transition-timing-function: linear;
  transition-delay: 2s;
  /* Safari */
  -webkit-transition-property:width;
  -webkit-transition-duration:1s;
  -webkit-transition-timing-function:linear;
  -webkit-transition-delay:2s;
}
```