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 loodi CSS.

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 stiililеht 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 ja praeguseks on põhiliselt kasutusel versioon 2.1.

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 tavalisem selektor on elemendi tüüp ehk HTML elemendi nimetus aga võimalikke selektoreid on veel mitmeid erinevaid.

Selektoriks elemendi nimi

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

```
#atribuudi_id_väärtus {kujundus}
```

Näiteks:

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

NB! id nime ei tohi alustada numbriga! See ei tööta Mozilla Firefox brauseri kasutamisel!

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

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

Selektoriks HTML elemendi atribuut

Selektorina võib määrata lisaks elemendi tüübile või atribuudi id väärtusele ka teiste atribuutide kaudu!

Määramaks kujundust kõigile, mingit kindlat atribuuti omavatele elementidele, tuleb selektorina kirja panna atribuudi nimi kandilistes sulgudes. Näiteks kujundus kõigile elementidele, millel on määratud atribuut *title*:

```
[title]{text-align:center}
```

Kasutada võib ka kindla väärtusega atribuute. Näiteks kujundus kõigile elementidele, millel on määratud keele atribuut väärtusega *lang=en*:

```
[lang|=en]{color:red}
```

Kuna näiteks keel võib olla täpsemalt määratletud, näiteks lang=en-us, siis võib selektori panna kirja pisut teisiti:

```
[lang|=en]{color:red}
```

Kui on tarvis luua kujundus kõigile elementidele, millede väärtus sisaldab teatud väärtust, saame kirjutada:

```
[title~=pilt]{width: 200px}
```

Selektoriks vormi (form) element

Sageli võib olla vaja kujundada vorme, sellisel juhul tuleb kasutada selektorina HTML elementi input koos atribuudi type väärtuse määramisega, näiteks:

```
input[type="text"]{color: red}
input[type="button"]{width: 60px}
```

Mitu erinevat selektorit

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
}
```

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*.

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

Erinevad kujundused sama tüüpi elemendile (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;
}
```

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 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 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}
```

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 vorming

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}
```

Taandrea loomiseks kasutatakse omadust *text-indent*, mille väärtus võib olla fikseeritud arv pikselites (mõõtühikuks px), punktides (mõõtühikuks pt) või ka sentimeetrites (cm). 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*!

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 normaalkõrgusest. Näiteks määrame tekstilõikudele reakõrguseks 125% normaalkõrgusest:

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

Lõikudele saab määrata ka veeriseid (*margin*), mille kirjeldus on peatükis „Veerised“.

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 ei toimi, sellisel juhul tuleks asendustoiminguteks pakkuda konkreetsete fontide nimede järel veel soovitud fondiperekonna ja lõpuks ka üldise perekonna nimetusi.

Fontide üldisteks (geneerilisteks) perekondadeks on: serif, sans-serif ja monospace.

Näiteks:

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

Kirja suurus

Kirja suuruse määramiseks kasutatakse omadust *font-size*, mille väärtus pannakse enamasti kirja pikselites (px). 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!

Kaldkiri, paks kiri, dekoratsioonid

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}
```

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}
```

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*.

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

Sarnane omadus on *font-variant* mis võimalda 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 ongi 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, lubatud on ka negatiivsed väärtused).

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

NB! Seda väärtust *inherit* ei toeta IE versioonid, IE8 kaasarvatud!

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. Arvväärtuste puhul saab kasutada ka negatiivseid väärtuseid.

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

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}
```

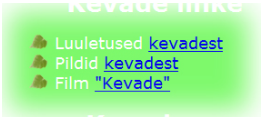
Täpploendi puhul (element <ul>) saab kasutada väärtuseid: *disc* (vaikeväärtus), *circle*, *square*, *none* (täpid puuduvad).

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")}
```

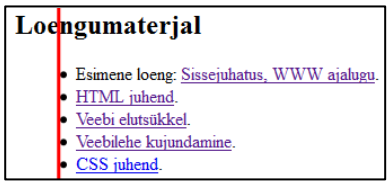


Joonis 1 Pilt järjestamata loendi täpina

NB! Pildifailide kasutamisel tuleb valida väikesemõõdulised pildid, näiteks 15X15 pikslit!

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 2 Loendielemendi marker väljaspool sisu piirkonda (*outside*)



Joonis 3 Loendielemendi marker sisu piirkonnas (inside)

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:

- 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 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 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 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}
```

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.

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;}
```

Polsterdus (*padding*)

CSS võimaldab määrata ka polsterdust (*padding*) ehk ruumi veebilehe elemendi ja tema raamjoone vahel (näiteks tabel või pilt). Polsterdust saab määrata eraldi üleval, all, vasakul ja paremal, mõõtühikutena px, pt, em ja cm:

```
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 elemendi mõõtmetele!

Veerised

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

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

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

```
img
{margin-top: 5cm;
margin-bottom: 2cm;
margin-left: 2cm;
margin-right: 5cm;}
```

Lisaks sentimeetritele võib kasutada ka protsente, näiteks:

```
img {margin-top: 25%;}
```

Võimalik on kasutada ka väärtus 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;}
```

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, sektsioon) 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

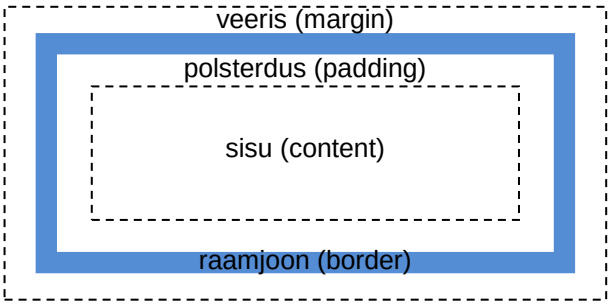
Objekti suuruse määramiseks on omadused *width* (laius) ja *height* (kõrgus), millede väärtusteks on arvud pikselites või protsentides (veebilehitseja akna mõõtude suhtes). Näiteks võime luua piltide jaoks kujundusklassi nimega "pisipilt", mille puhul on piltidel kindlad mõõtmed 100 X 75 pikselit:

```
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 4 Kasti mudel (box model)

Tähtis on meeles pidada, et kasti ehk HTML elemendi tegelikud mõõtmed saadakse kõigi nelja osa mõõtmete liitmisel!

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}
```

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 5 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 tema 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 0.

Näiteks loome piltide jaoks kujundusklassi nimega "esiplaan", mille abil asetame pildid teiste objektide suhtes kõrgemale kihile (meil 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 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 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 „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:

```
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}
```

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

Kokkuvõte

Käesolev CSS juhend on kokkuvõtlik ja katab vaid olulisemad, enimkasutatavad vahendid. Täieliku materjali CSS keele kohta leiab aadressil: http://www.w3schools.com

Näitena esitame siinkohal CSS stiililehe, milles:

- veebilehe taust kujundatud metalselt siniseks;
- pealkiri on fondiga Verdana, suurusega 36 pt ja tumesinine, kaldkirjas ja allajoonitud;
- tekst lõikudes ja loendites fondiga Verdana ja suurusega 24 pt, nende maksimaalseks laiuseks on 800 pikselit;
- täpploendis on täppidena kasutusel pilt "fotokas.png";
- tabeli raamjooned on sinised ja punktiiris;
- linkidele on loodud eespool näitena toodud kujundus;
- ühe lille nimetuse rõhutamiseks on loodud eraldi kujundusklass "markeriga" (elemendile), millel on valge taust ja paks kiri;
- veebilehele on lisatud alajaotus ja selle sees üks uus pilt. Alajaotus on kujundusklassi "pildiaken" abil paigutatud veebilehe sisu kõrvale paremale, kõige peale:

```
body      {background-color: #56A5EC}

h1        {font-family: verdana;
  font-size: 36px;
  color: #151B54;
  font-style: italic;
```

```
   text-decoration: underline}

p,ul,ol      {max-width: 800px;
   font-family: verdana;
   font-size: 24px}


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

table        {border-color: #153E7E;
   border-style: dotted}

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}

span.markeriga    {font-weight: bold;
   background-color: #FFFFFF}

div.pildiaken {position: fixed;
   left: 500px;
   top: 270px;
   z-index: 10}
```

Selle stiililehe rakendamisel saab meie HTML peatüki kokkuvõttes näitena toodud veebileht (millele on lisatud üks element ja lõppu üks alajaotus , mille sisuks lilleõitega pilt) järgmise välimuse:

Joonis 6 HTML näitena loodud veebileht CSS abil kujundatuna