Stiilid

Kui HTML markeerimiskeel 90. aastate alguses leiutati ning järgnevatel aastatel koos WWW ja HTTP protokolliga tööle rakendati, oli selle peamiseks eesmärgiks andmete struktureeritud kuvamine. Interneti meeletu menu tõi päevakorda aga disainimisvajaduse. Õigete reeglite puudumisel hakati kujunduse loomiseks ära kasutama HTML-keele võimalusi, mille tegelik eesmärk oli (ja on) vaid andmete struktureeritud kuvamine.

Nii paigutatakse praegugi enamik kujunduselemente HTML-tabelitesse, mis pole oma ülesehituselt mõeldud disainilahenduste loomiseks.

Atribuutide ja nendele antavate väärtuste abil on võimalik HTML-elementide mõju juhtida. Osad atribuudid omavad kindlaid väärtusi, osadele tuleb see ise omistada. Enamasti on atribuutidel olemas ka omad kindlad vaikeväärtused, mida kasutatakse siis, kui veebilehe autor pole elemendi atribuute täpsustanud.

Enamiku atribuutide väärtusi on võimalik kogu veebistruktuuri (st kõigi dokumentide) ulatuses paika panna ka stiililehtedega. ***Stiililehт*** on tavaline tekstidokument, mis sisaldab infot elementide põhiatribuutide ja nende väärtuste kohta. Muutes stiililehes infot kajastub see automaatselt kõigil dokumentidel, mis seda stiililehte kasutavad. Kui stiililehт ja elemendis sisalduv atribuut püüavad määrata sama asja, siis on suurem prioriteet atribuudil, st atribuudi puudumisel kasutatakse stiililehes antud väärtusi.

Stiililehtede kasutamist veebilehekülgede kujundamisel soovitab ka W3C (*World Wide Web Consortsium*) - organisatsioon, mille eesmärgiks interneti arendamine ning veebistandardite väljatöötamine.

Koostanud: Merike Hein

CSS

CSS (*Cascade Style Sheet*) on astmeline stiilileht. Igaüks, kes mingisuguselgi tasemel tegeleb veebilehtede ehitamise ja arendamisega, on pidanud seisma silmitsi probleemiga, et igale poole on tarvis kirjutada teksti kuvamise definitsioone, näiteks: <FONT SIZE="12" COLOR="#FFFFFF" FACE="ARIAL">...</FONT> jne. jne. Tihtipeale kujunevad seesugused read kuni saja tähemärgi pikkuseks. Selle asemel pakub CSS võimalust hoida kokku aega ja närve ning võimaldab ühte seesugust pikka rida märkida ülilühidalt HTML-keeles teadatuntud käskudega H1 - H6. Samuti jääb ära igasugune vanamoodne fontide defineerimine linkides, selle asemel defineeritakse CSS-i abil see vaid üks kord, iga eri lingi jaoks võetakse fondi parameetrid samast kohast. Ühesõnaga - CSS lihtsustab kõvasti iga veebiarendaja tööd ja lisab uusi võimalusi.

Astmelised on stiililehed seetõttu, et töötavad nn pärimis- ja pärandamismeetodil, st kui defineerida kõrgema taseme HTML-elemendi stiilivormingud (võtame nt <body>...</body> element), siis mitmed vormingutest kantakse üle madalama taseme HTML-elementidele, mis asuvad kõrgema taseme HTML-elementide mõjupiirkonnas (nt ... element asub <body>...</body> elemendi mõjuväljas või ... element asub ... elemendi mõjuväljas). Pärimine katkeb, kui madalama taseme elemendile samad reeglid eraldi defineerida.

CSS-i sisu

Astmelised stiililehed moodustuvad ühest või enamast reeglist, mis kontrollivad, kuidas valitud elemente veebilehel kuvada. Reegel koosneb elemendist (võib nimetada ka selektoriks) ja selle deklaratsioonist. Element asub alati deklaratsiooni ees. Deklaratsioon koosneb tunnusest ja selle väärtusest. Deklaratsioon(id) esitatakse loogelistes sulgudes. Ühele elemendile võib sätestada mitu deklaratsiooni, need eraldatakse semikoolonitega.

Enne, kui üldse hakatagi mõtlema CSS-i kirjutamisele ja kujundusele, tuleb paika panna dokumendi üldine struktuur. HTML-i loomisel ei tohiks teha mingeid eeldusi selle kohta, milline hakkab olema kujundus - veelgi enam - ideaalsel juhul ei nõua kujunduse muutmine mitte ainumastki muudatust HTML-is.

Lihtsaimal juhul koosneb veebileht ühestainsast HTML-failist. Alguses lehekülje pealkiri, siis natuke juttu, veel mõned pealkirjad ja veel veidi juttu. Kõik see jookseb ka kujunduslikus plaanis kenasti ühe ühise nivoona ülevalt alla kuni tekst lõpeb. Veebiajastu koidikul paljud lehed

Koostanud: Merike Hein

just nõnda lihtsad olidki, ent tänapäeval on sellised lehed pigem harulduseks saanud. Liigume edasi millegi reaalsema juurde...

Lihtsal juhul koosneb üks tänapäevane lehekülg sisust ning menüüst, viimane võimaldab navigatsiooni erinevate sisuosade vahel. Tõenäoliselt peame menüü eraldi identifitseerima, et saaksime seda eraldi ülejäänud tekstist kujundada. Kasulikuks võib osutuda ka sisuosa vastav tähistamine.

CSS-i abil võib optimeerida ka HTML-koodi, kui kasutada veebilehtede kujundamisel tabelite asemel CSS klass- või ID-elementidega määratletud kihte. Näiteks

```
<div id= ”menüü">Menüü</div>
```

Näiteks nõnda (selles ja järgnevates näidetes on toodud vaid dokumendi body osa):

```
<div id="menüü">
    <ul>
      <li><a href="pall1.htm">Pallike 1</a></li>
      <li><a href="pall2.htm">Pallike 2</a></li>
    </ul>
</div>
<div id="sisu">
    <h1>Elus eksida võime vaid korra</h1>
    <p>Elus eksida võime vaid korra -
    Sellest algabki tagasikäik.
    Pühi maha need mingikorrad
    Ja näita, mis on su õige läik.</p>
    <p>Pühi maha see krohvisegu,
    Et ma näeksin su murdejoont.
    Näen, sul ammugi juba on tegu
    Sirgelt järgida elujoont.</p>
    <p>Võta vastu üks sõbralik mükse,
    Nii saab seieril õigeks näit.
    Muidu elult saad vastu pükse -
    Lihtsalt algab ju tagasikäik.</p>
</div>
```

Kena, aga mis siis kui menüü osas pole mitte kaks vaid kolmkümmend linki? CSS-iga võime muidugi menüü asukohta muuta, aga need kasutajad, kelle brauser CSS-i ei toeta? Kas nad peaksid igal leheküljel enne tükk aega kerima, kuni jõuavad viimaks sisuni? Ning need kasutajad, kes ei näe? Kas nad peaksid kuulama, kuidas iga lehe alguses neile kogu menüü ette loetakse?

Ei! Kindlasti ei pea nad seda tegema. Nad võivad teie lehelt niisama kergesti lahkuda, kui nad sinna tulid. On sellele probleemile lahendus?

Loodetavasti on see küsimus juba ette vastuse leidnud: me paigutame menüü dokumendi lõppu. Nii lihtne see ongi. Ja kui peaks olema kaks menüüd, siis paigutame ka nad mõlemad

dokumendi lõppu. Tegelikult kehtib lihtne ja üldine reegel: me korraldame oma dokumendi sisu tähtsuse järjekorras.

Kui meil on näiteks uudisteleht, kus on viimaste uudiste nimekiri, menüü erinevate uudiste kategooriatega, viited teistele uudiste-lehekülgedele ning uudis ise, siis milline peaks olema järjekord? Kõigepealt kindlasti konkreetne uudis (avalehel päevauudis ja muudel kusagilt menüüst valitud uudis), siis viimaste uudiste nimekiri, siis uudised kategooriate kaupa ning lõpetuseks (kõige igavam ja vähem huvipakkuv) teised uudistelehed. Umbes nõnda:

```
<div id="uudis">
    <h1>Peaminister lendas nagu tuvi!</h1>
    <p>Täna võisime kõik nautida kuidas meie kallis peaminister
    võttis viimaks selga tiivad ja nii edasi.</p>
</div>
<div id="viimased_uudised">
    <h6>Viimased uudised</h6>
    <ul>
      <li><a href="peaminister.htm">Peaminister lendab</a></li>
      <li><a href="nato.htm">Õhuväebaas tuleb Narva</a></li>
      <li><a href="seks.html>Üleriigiline seksiskandaal</a></li>
    </ul>
</div>
<div id="kategooriad">
    <h6>Kategooriatena</h6>
    <ul>
      <li><a href="eesti.htm">Kodumaised</a></li>
      <li><a href="euro.htm">Euroopa Liit</a></li>
      <li><a href="maailm.htm">Muu maailm</a></li>
      <li><a href="imelik.htm">Imelikud uudised</a></li>
    </ul>
</div>
<div id="teised_lehed">
    <h6>Teised uudistelehed</h6>
    <ul>
      <li><a href="http://www.epl.ee">Eesti Päevaleht</a></li>
      <li><a href="http://www.postimees.ee">Postimees</a></li>
    </ul>
</div>
```

CSS annab meile võimaluse luua väljanägemiselt raamide sarnase funktsionaalsusega objekte. Näiteks määratakse lehekülje sisuosale kindel kõrgus ning juhul kui tekst ületab määratud kõrguse tekitatakse servale kerimisriba:

```
div#sisu {
    height: 400px;
    overflow: auto;
}
```

Välimuselt saavutatakse sama efekt, mis raamide kasutamisel, aga lehekülg jääb kasutatavaks ka siis, kui brauser CSS-i ei toeta, sest siis kuvatakse lihtsalt kogu tekst tema täies pikkuses. Seega ei teki ühegi brauseriga lehekülje kasutamisel selliseid probleeme nagu raamide puhul.

Peamiselt tekib vajadus selliste "raamide" järele siis, kui meil on tarvis, et tekst jääks teatud kindlatesse raamidesse (loogiline, kas pole). Näiteks kui meie veebileht esitab koodinäiteid, mille puhul reeglina reamurdmist ei kasutata, siis soovime end ehk relvastada olukorra vastu, kus mõni üksik pikk koodirida venitab meie lehe inetult kümme korda laiemaks. Selleks puhuks pole vaja muud, kui määrata näiteks elemendile „pre“ maksimaalne laius ning sellest üleminekul nõuda horisontaalse kerimisriba tekkimist.

Kuidas saab CSS-i rakendada?

CSS - faili rakendamiseks on kaks võimalust:

1. kirjutada CSS HTML-faili sisse
2. kirjutada eraldi CSS-fail, millest võetakse kõik definitsioonid

Esimesel juhul:

kirjutatakse CSS definitsioonid igasse konkreetsesse HTML-faili eraldi. Seda tehakse järgmiselt:

- CSS peab olema kirjutatud HTML-failis <HEAD> ja </HEAD> tagide vahele;
- peab olema ümbritsetud <STYLE TEXT="text/css"> ja </STYLE> tagidest;
- ja kuna ei soovita CSS-definitsioone lehekülje vaatajale kuvada, siis pannakse kogu CSS veel <!-- ja --> vahele.

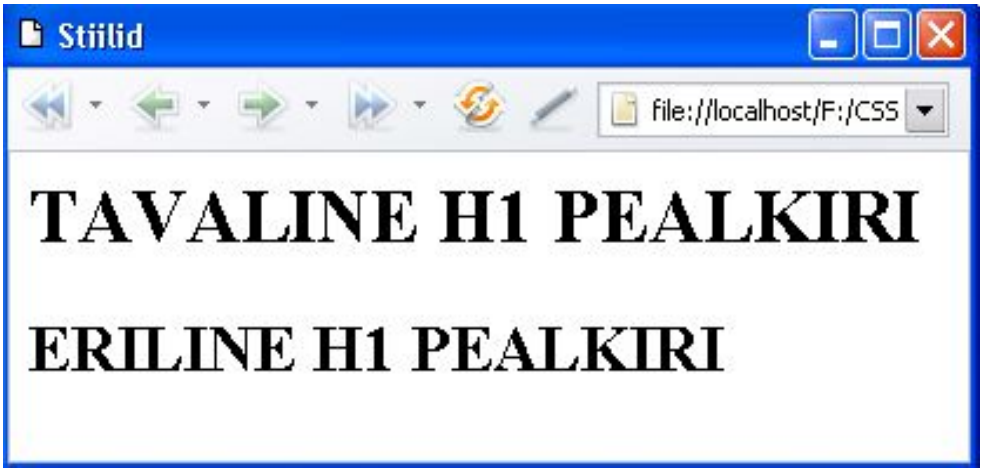
Lõpptulemus peaks välja nägema niisugune:

```
<HEAD>
<STYLE TYPE="text/css">
<!-- siia tulevad definitsioonid -->
</STYLE>
</HEAD>
```

Koodinäide 1: stiili määramine lehe päises

```
<HTML>
    <HEAD>
        <TITLE>Stiilid</TITLE>
        <STYLE>H1 {font:16pt}</STYLE>
    </HEAD>
    <BODY>
        <H1>TAVALINE H1 PEALKIRI</H1>
        <H1 STYLE="font-size: 20pt">ERILINE H1 PEALKIRI</H1>
    </BODY>
</HTML>
```

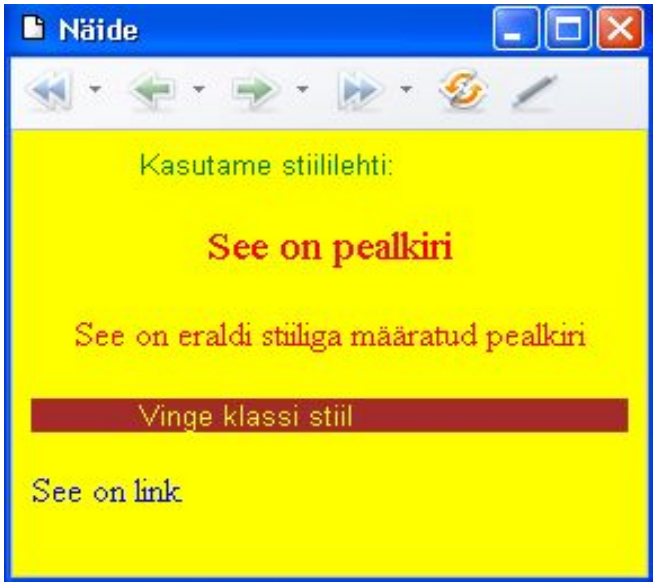
Nii näeb see kood välja testbrauseris:



Koodinäide 2: stiili määramine lehe päises ja konkreetses lõigus

```
<HTML>
    <HEAD>
        <TITLE>Näide</TITLE>
    </HEAD>
    <STYLE>
    BODY {background: yellow; color: darkblue}
    H1 {font: 14pt Arial bold; color:red; text-align:center}
    P {font: 10pt Arial; text-indent: 0.5in; color:green}
    A {text-decoration: none; color: blue}
    .vinge {font: Verdana Italic; color:yellow; background: brown}
    </STYLE>
    <BODY>
        <P>Kasutame stiililehti:</P>
        <H1>See on pealkiri</H1>
        <H1 STYLE="font-size:12pt">See on eraldi stiiliga määratud pealkiri</H1>
        <P CLASS="vinge">Vinge klassi stiil</P>
        <A HREF="#_self">See on link</A>
    </BODY>
</HTML>
```

Nii näeb see kood välja testbrauseris:



Koostanud: Merike Hein

Teisel juhul:

luuakse definitsioonide jaoks eraldi fail laiendiga *.css ning toimitakse järgmiselt:

- fail kirjutatakse sarnaselt esimesega, erinevus on selles, et puuduvad <HEAD> ja </HEAD> tagid;
- leheküljele, mida tehtud CSS mõjutama peaks, peab kirjutama <HEAD> ja </HEAD> tagide vahele järgmist:

```
<LINK REL=stylesheet HREF="failinimi.css" TYPE="text/css">
```

Milliseid HTML-i elemente saame CSS-iga muuta?

- heading - tagid (H1 - H6)
- paragrahv - tag (P)
- dokumendi nn. "keha" (BODY)
- kasutamata ja kasutatud lingid(A:link ja V:link)
- lehekülje jaotus (DIV)

Neljast esimesest elemendist on juba varem juttu olnud ja tuleb veel edaspidigi, kuid siinkohal peatuks pikemalt viimasel. -märgend on väga kasulik, sest tema abil saab lehekülje jaotada mõttelisteks osadeks. Näiteks:

DIV võimaldab mugavalt määrata tervele elementide plokile teatud omadusi - näiteks määrata ära kasutatav keel või stiil. Eriti mugav on kasutada DIV elementi koos CLASS, ID ja LANG üldatribuutidega. Kui näiteks dokumendi menüünuppude osa ümbritseda DIV elementidega, mille atribuudiks antakse CLASS=menüü, siis saab stiililehe abil mugavalt määrata ära stiiliklassi "menüü" taustavärvi (ja seda korraga kõikjal, kus menüünuppe kasutatakse!).

Järgnevas peatükis tuleb juttu lähemalt stiililehtede paigutusest.

Koostanud: Merike Hein

Valikuline selekteerimine

Tihtipeale jääb väheseks üksnes võimalusest omistada stiile HTML-elemendi nime järgi, näiteks muuta mõni konkreetne pealkiri allajoonituks või teha ühe lõigu piires kõik rõhutatud sõnad punaseks. Selleks ja veel palju enamaks on CSS-is mitmesugused selektorid. Täpsemalt järgmistes peatükkides.

Kõigi elementide selekteerimine

Sümbol „ * “ (tärn) selekteerib kõik elemendid. Esmapilgul võib tunduda, et selle efekt on täpselt sama, mis kasutades <body>..</body>-elementi, kuid neil kahel on siiski selgelt tuntav vahe. Mille poolest erinevad üksteisest näiteks järgnevad deklaratsioonid:

```
* { color: green }
body { color: green }
```

Mõlemad muudavad kogu teksti roheliseks, aga erinevus seisneb selles, et määrates värvi elemendile „body“, kandub see edasi kõigile temas sisalduvatele elementidele, välja arvatud nendele, millele on atribuut „color“ juba eelnevalt määratud. Seega muutub roheliseks kõik ülejäänud tekst peale linkide, sest neile on juba antud sinine värv. Kasutades tärni („ * “), muutub aga roheliseks kogu tekst, sealhulgas ka lingid. Toon siinkohal kaks näidet, kus ühes on kasutatud „ * “-elementi ja teises „body“-elementi ning vastavaid tulemusi:

Koodis:

```
<html>
<head>
<title>Näide</title>
</head>
<style>
* { color: green }
</style>
<body>
<h1>Paber kannatab kõike</h1>
<p>Paber kannatab kõike: Kastad vette - pärast väljavõtmist Kuivab varsti ära; Ajad värvi peale - Paber on alles, olgu küll teist värvi; Iga tuulega lendab kaasa,
Aga ta kestab.</p>
<a href="pall.htm">PALLIKE</a>
</body>
</html>
```

Brauseris:

Koostanud: Merike Hein

Koodis:

```
<html>
<head>
<title>Näide</title>
</head>
<style>
body { color: green }
</style>
<body>
<h1>Paber kannatab kõike</h1>
<p>Paber kannatab kõike: Kastad vette - pärast väljavõtmist Kuivab varsti ära; Ajad värvi peale - Paber on alles, olgu küll teist värvi; Iga tuulega lendab kaasa,
Aga ta kestab.</p>
<a href="pall.htm">PALLIKE</a>
</body>
</html>
```

Brauseris:

Alamelementide selekteerimine

Kõige lihtsamal juhul määratakse stiilid järgneva konstruktsiooniga:

ELEMENT { PARAMEETER: VÄÄRTUS }

kus

- **ELEMENT** on suvaline HTML-i element,
- **PARAMEETER** on suvaline CSS-i parameeter ja
- **VÄÄRTUS** on üks võimalikest antud parameetri väärtustest.

Näiteks parameetri **color** väärtuseks võib muuhulgas olla kas „red“, „blue“ või „green“. Seega võime anda elementidele „h1“ kuni „h5“ näiteks järgmised „color“ väärtused:

```
h1 { color: red }
h2 { color: blue }
h3 { color: blue }
h4 { color: green }
h5 { color: green }
```

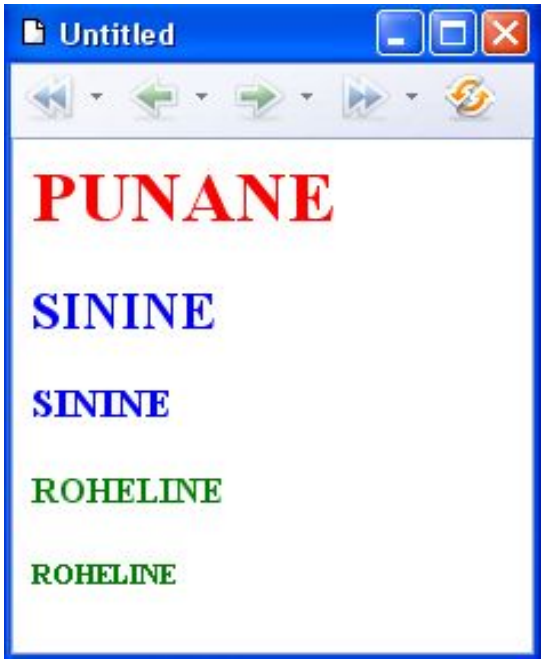
Koostanud: Merike Hein

Koodis:

```
<html>
 <head>
  <title>Näide</title>
  <style type="text/css">
h1 { color: red }
h2 { color: blue }
h3 { color: blue }
h4 { color: green }
h5 { color: green }
  </style>
 </head>
 <body>
  <h1>PUNANE</h1>
  <h2>SININE</h2>
  <h3>SININE</h3>
  <h4>ROHELINE</h4>
  <h5>ROHELINE</h5>
 </body>
</html>
```

Selle tulemusena näidatakse:

- „h1“ teksti punasena,
- „h2“ ja „h3“ teksti sinisena,
- „h4“ ja „h5“ teksti rohelisena.



Et laiendada värvigammat veelgi enam, lisame näiteks stiili **background-color** ja **font-style**:

Koodis:

```
<html>
    <head>
        <title>Näide</title>
    </head>
    <style>
h1 { color: red }
p { background-color: green; font-style: italic }
    </style>
    <body>
        <h1>Paber kannatab kõike</h1>
        <p>Paber kannatab kõike:
        Kastad vette - pärast väljavõtmist
        Kuivab varsti ära;
        Ajad värvi peale -
        Paber on alles, olgu küll teist värvi;
        Iga tuulega lendab kaasa,
        Aga ta kestab.</p>
    </body>
</html>
```

Selle tulemusena näidatakse:

- „h1“ teksti punasena,
- „p“ tekst kaldkirjas ja rohelise taustaga.

Koostanud: Merike Hein

Stiilid mitmele elemendile korraga

Kuna nii „h2“ ja „h3“ kui ka „h4“ ja „h5“ on täpselt sama stiiliga, siis võime need deklaratsioonid kokku võtta, eraldades elemendid komaga, mis on täiesti samaväärne eelnenuga.

```
h1 { color: red }
h2, h3 { color: blue }
h4, h5 { color: green }
```

Järgnevalt lisame uue parameetri **text-decoration**. See võib omada väärtusi „none“, „underline“, „overline“, „line-through“ või „blink“. Lisame mõningad „text-decoration“ parameetrid oma stiilidele:

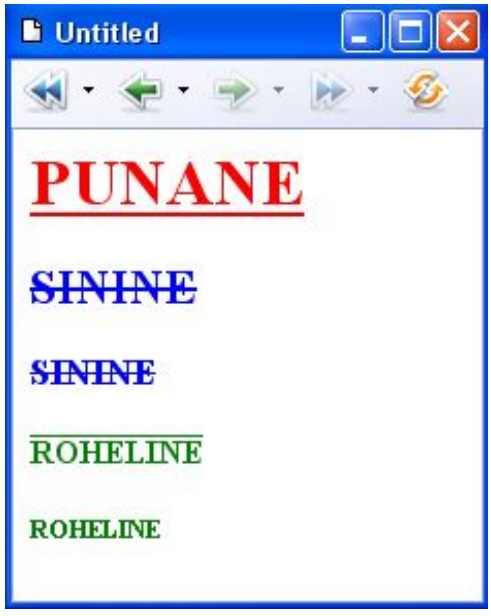
```
h1 { color: red }
h1 { text-decoration: underline }
h2, h3 { color: blue }
h2, h3 { text-decoration: line-through }
h4, h5 { color: green }
h4 { text-decoration: overline }
h5 { text-decoration: none }
```

Koodis:

```
<html>
 <head>
  <title>Näide</title>
  <style type="text/css">
h1 { color: red }
h1 { text-decoration: underline }
h2, h3 { color: blue }
h2, h3 { text-decoration: line-through }
h4, h5 { color: green }
h4 { text-decoration: overline }
h5 { text-decoration: none }
  </style>
 </head>
 <body>
  <h1>PUNANE</h1>
  <h2>SININE</h2>
  <h3>SININE</h3>
  <h4>ROHELINE</h4>
  <h5>ROHELINE</h5>
 </body>
</html>
```

Selle stiililehe rakendamise tulemus on:

- „h1“ on punane ja allajoonitud,
- „h2“ ja „h3“ on sinised ning läbijoonitud,
- „h4“ on roheline ja ülajoonitud,
- „h5“ on roheline ja ilma jooneta.



Koostanud: Merike Hein

Ka siinkohal saame oma stiililehte lihtsustada, koondades ühe elemendi kohta käivad parameetrid looksulgude sisse ja eraldades semikooloniga:

```
h1 { color: red; text-decoration: underline }
h2, h3 { color: blue; text-decoration: line-through }
h4, h5 { color: green }
h4 { text-decoration: overline }
h5 { text-decoration: none }
```

võib ka nii:

```
h1 { color: red; text-decoration: underline }
h2, h3 { color: blue; text-decoration: line-through }
h4 { color: green; text-decoration: overline }
h5 { color: green; text-decoration: none }
```

Vaikimisi väärtused

Kõigil HTML-i elementidel on juba eelnevalt defineeritud vaikimisi stiilid. „strong“ element on alati rasvane, „a“ element alla joonitud, „h1“ suur ja rasvane. Näiteks parameeter font-style on elemendil „em“ vaikimisi „italic“. Kui me soovime, et element „em“ oleks kaldkirjas võime me talle kirjutada järgmise stiili:

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

Üldjuhul me seda aga ei tee, sest „em“ on alati kaldkirjas. Pigem muudame me „em“-i parameetrit „font-style“ siis, kui soovime, et ta kindlasti kaldkirjas poleks.

Koostanud: Merike Hein

Kaskaadimine

Mis aga juhtub siis, kui me teeme sellise stiililehe:

```
p { text-decoration: underline }
p, em { text-decoration: overline }
h1, p { text-decoration: line-through }
```

Siinkohal avaldub CSS-i kaskaaduv iseloom. Brauser liigub läbi meie CSS-faili ülevalt alla ning kui ta kohtab samal elemendil sama omadust uuesti, siis ta lihtsalt jätab meelde viimase. Kaskaadumine on CSS-i üks peamisi omadusi, mille abil võib saavutada suurepäraseid tulemusi.

Koodis:

```
<html>
    <head>
        <title>Näide</title>
    </head>
    <style>
p { text-decoration: underline }
p, em { text-decoration: overline }
h1, p { text-decoration: line-through }
    </style>
    <body>
        <p>ALLA JOONITUD</p>
        <p>JOONTEGA</p>
        <em>JOONTEGA</em>
        <h1>PUNANE</h1>
        <p>PUNANE</p>
    </body>
</html>
```

Seda protsessi võib ette kujutada järgnevalt:

1. rida: element „p“ on alla joonitud,
2. rida: element „p“ on ülajoonitud, element „em“ on ülajoonitud,
3. rida: element „p“ on läbikriipsutatud, element „h1“ on läbikriipsutatud.

Lingitud stiililehed

Veelgi parem, kui stiilide paigutamine <STYLE>...</STYLE>-elementide vahele, on nende hoidmine hoopis eraldi failis. Selleks loome eraldi tekstifaili, millele paneme nimeks näiteks "eraldi.css" ning kirjutame sellesse oma stiilid:

```
h1 { color: red }
p { background-color: green; font-style: italic }
```

NB! faili kirjutatakse vaid stiilide kirjeldused, ümbritsevaid style-elemente nagu HTML-is sinna ei kirjutata. HTML-ist eemaldatud style-osa asemele paneme aga järgmise rea:

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

See märgib, et selle HTML-dokumendiga on ühendatud stiililleht, mis asub failis "eraldi.css", stiililehe tüüp on CSS formaadis tekst.

Koodis:

```
<html>
<head>
<title>Juku kodulehekülg</title>
<link rel="stylesheet" href="juku.css" type="text/css" />
</head>
<style type="text/css">
<!--
h1 { color: red }
p { background-color: green; font-style: italic }
-->
</style>
<body>
<h1>Teretulemast Juku kodukale</h1>
<p>Tere, mina olen Juku. Mul on suur rõõm tervitada
teid kõiki minu suurepärasel koduleheküljel, mida ma
olen otsustanud rikastada CSS-iga.</p>
</body>
</html>
```

Tulemus:

Koostanud: Merike Hein

Tungivalt soovitatav on eelistada sellist HTML-iga linkimise tehnikat eelpool kirjeldatud CSS-i paigutamisele HTML-i sisse. Linkimise eelisteks on:

- võimalus kasutada ühte CSS-i faili mitmes HTML-dokumendis,
- muutes kujundust ühes CSS-i failis muutuvad korraga kõik sellega seotud HTML-dokumendid,
- väheneb veebilehekülje maht, sest stiilid tuleb alla laadida vaid esimese lehekülje külastamisel, ülejäänute puhul kasutatakse juba olemasolevat faili.

Kõige enam kasutatavamat CSS - käsku

Kõik järgnev on esitatud kujul:

- funktsioon
- mida teeb?
- võimalused
- näide

Fondi/teksti definitsioonid

- **font-family**
- märgib fondi nägu e. shrifti
- võib kasutada kõikvõimalikke fonte
- H1 {font-family: arial, courier}

- **font-style**
- märgib fondi esitamise moodust
- normal, italic, small caps, oblique
- H1 {font-style: italic}

- **font-variant**
- märgib fondi varianti
- small caps (väiketähed), normal
- H1 {font-variant: small caps}

- **text-align**
- märgib teksti asetust lehel
- center, left, right, justify
- H1 {text-align: justify}

- **font-size**
- märgib fondi suurust
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- H1 {font-size: 20pt}

- **font-weight**
- märgib fondi "raskust"
- extra-light, light, demi-light, medium, demi-bold, bold, extra-bold
- H1 {font-weight: light}

Koostanud: Merike Hein

- **text-decoration**
- märgib teksti dekoreerivaid elemente
- italic, blink, underline, line-through, overline, none
- H1 {text-decoration: line-through}

- **text-indent**
- märgib äärejooni
- tollides(in), sentimeetrites(cm), pikslites(px)
- H1 {text-indent: 20px}

- **word-spacing**
- määrab ruumi sõnade vahel
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- H1 {word-spacing: 1in}

- **letter-spacing**
- määrab ruumi tähtede vahel
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- H1 {letter-spacing: 1cm}

- **text-transform**
- märgib teksti üldist vormi
- capitalize, uppercase, lowercase
- H1 {text-transform: uppercase}

- **color**
- märgib teksti värvi
- #värvikood
- H1 {color: #0000FF}

Äärejoonte/tausta definitsioonid

- **margin-left**
- määrab vasaku äärejoone
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px)
- BODY {margin-left: 1in} või P {margin-left: 1in}

Koostanud: Merike Hein

- **margin-right**
- määrab parema äärejoone
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px)
- BODY {margin-right: 1in} või P {margin-left: 1in}

- **margin-top**
- määrab ülemise äärejoone
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px)
- BODY {margin-top: 1in}
  või
- P {margin-left: 1in}

- **margin**
- määrab kolm äärejoont korraga
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px). Kõigepealt ülemine, parem ja siis vasak
- P {margin: 1in 3px 5cm}

- **line-height**
- märgib reavahet
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- H1 {line-height: 15pt}

- **background-color**
- määrab lehekülje tausta värvi
- #värvikood
- BODY {background-color: #000000}

- **background-image**
- määrab lehekülje taustaks kuvatava pildi
- URL ehk pildi aadress
- BODY {background-image: http://www.domeen.ee/pilt.gif"}

Koostanud: Merike Hein

- **background-repeat**
- määrab taustapildi korduvuse
- repeat-x, repeat-y, no-repeat
- BODY {background-repeat: repeat-y}

- **background-attachment**
- määrab tustapildi reageerimise kerimisele
- scroll, fixed
- BODY {background-attachment: fixed}

Positsiooni ja jaotuse definitsioonid

- **position**
- märgib objekti positsiooni leheküljel
- kirjutatakse objekti tagide sisse. absolute, relative
- <IMG STYLE="position:absolute" SRC="pilt.gif">

- **left**
- märgib objekti kauguse vasakust äärest
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- <IMG STYLE="position:absolute; LEFT: 20px;" SRC="pilt.gif">

- **top**
- märgib objekti kauguse ülemisest äärest
- punktides(pt), tollides(in), sentimeetrites(cm), pikslites(px), protsentides(%)
- <IMG STYLE="position:absolute; LEFT: 20px; TOP: 200pt" SRC="pilt.gif">

Kui kahtlete, kas CSS on ikka see, mida te vajate, siis võivad abiks olla järgnevad punktid, mis räägivad CSS-i kasuks.

- ***Sisu ja vorm on selgelt eraldatud*** - disainerid ja programmeerijad saavad töötada ilma et nad üksteist oma tööga segaksid. Kogu lehe disaini võib muuta ilma, et läheks tarvis ainsatki muudatust HTML-is või programmitekstis.
- ***Väheneb lehekülgede suurus*** - paigutades kogu kujunduse kohta käiva info ühte CSS-i faili, tuleb kasutajal kujundus alla laadida vaid esimesel korral, ülejäänud lehekülgede puhul kasutatakse juba varem allalaetud stiililehte. Brauserid, mis stiile ei näita, ei pea neid üleüldse alla laadima.
- ***Suureneb arusaadavus*** - CSS-ist on märksa kergem aru saada, kui vastavast HTML-is kirjutatud kujundusest, ühtlasi muutub paremini loetavaks ka HTML, kui sealt on kujunduse info eemaldatud.
- ***CSS on tulevik*** - varem või hiljem loobutakse HTML-i kasutamisest lehekülgede kujundamiseks ning minnakse üle puhtalt CSS-i põhistele lahendustele. See protsess on juba alanud.

Koostanud: Merike Hein