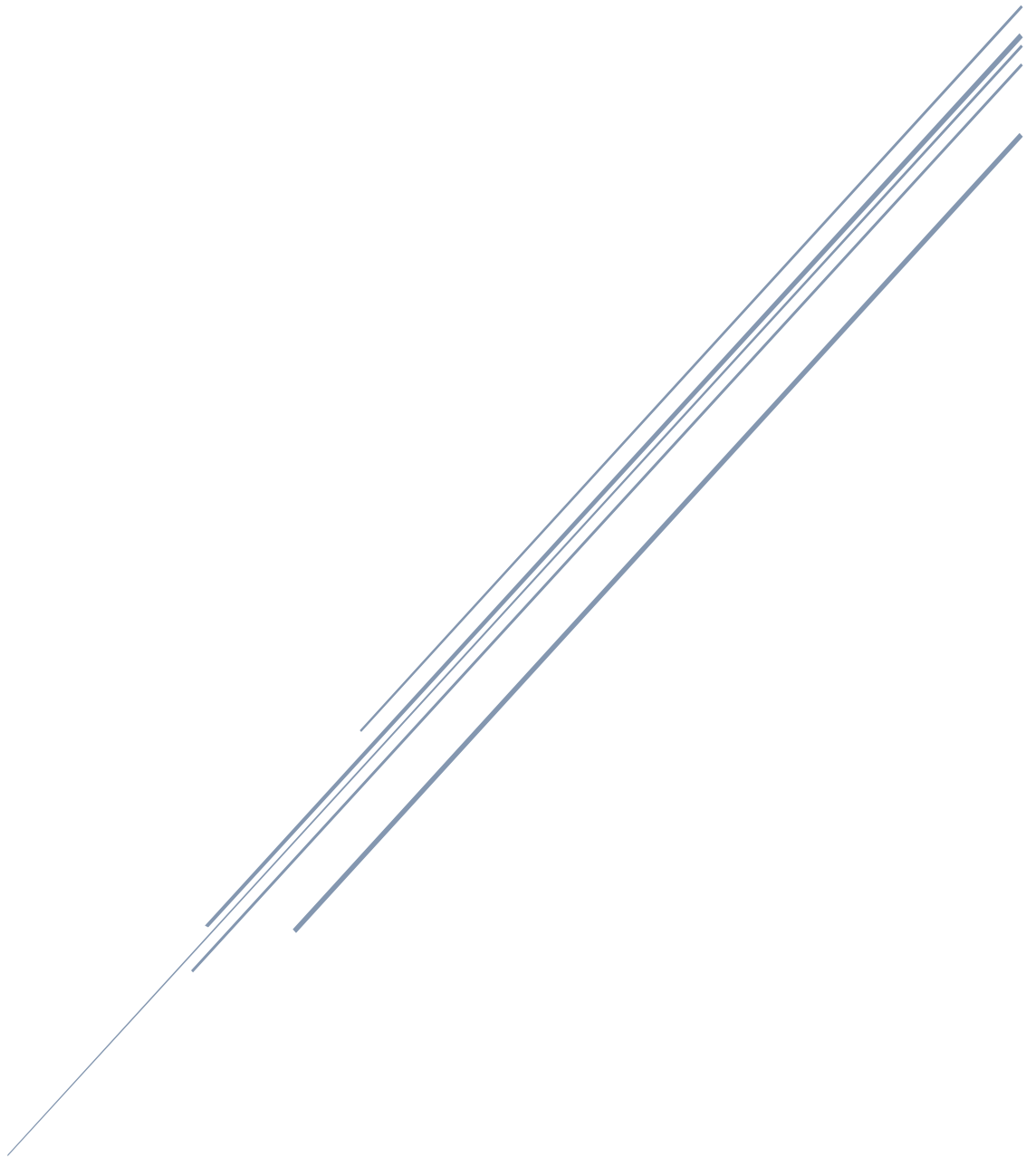


CSS



Mittuniversitetet
XML

Innehållsförteckning

1.1	Inledning	2
1.2	Grundläggande uppbyggnad	2
1.3	Länka CSS-filen till ett XML-dokument	4
1.4	Egenskapen display.....	5
1.5	Mer om selektorer.....	5
1.6	Arv i CSS	8
1.7	Värden	9
1.8	Egenskaper för färg och text	10
1.8.1	Färger.....	10
1.8.2	Text.....	10
1.9	Egenskaper för utrymme	12
1.9.1	Margin	12
1.9.2	Border.....	14
1.9.3	Padding.....	16
1.9.4	Size.....	16
1.10	Positionering.....	17
1.11	Lägga till bilder.....	18

1.1 Inledning

I denna lektion ska vi applicera stilmallar på våra XML-dokument. För detta ska vi använda CSS (Cascading Style Sheet). Lektionen börjar med att titta på hur ett CSS-dokument skapas, för att sen visa hur det knyts samman med ett XML-dokument. Dessutom går vi igenom en del egenskaper i CSS.

Genom att använda CSS kan vi skilja formateringen från semantiken och strukturen i ett XHTML- och XML-dokument. Detta gör vi genom att lägga information om hur data ska formateras i en CSS-fil och sen skapa en länk till denna fil i XHTML/XML-dokumentet. När det gäller XML-dokument kan vi även använda en teknik som heter XSL (Extensible Style Language) för att ange stilar. Detta är en betydligt mer avancerad teknik än CSS och som ger oss fler möjligheter till formatering. XSL kommer vi tillbaka till längre fram i kursen.

Fördelen med CSS är att standarden stöds av alla webbläsarna, men det kan vara så att webbläsare tolkar och visar standarden på lite olika sätt.

Några av er har kanske redan använt CSS i samband med HTML-filer. Tidigare lades information om hur varje enskilt element i en HTML-fil skulle presenteras direkt i HTML-filen, men World Wide Web Consortium (W3C) rekommenderar att skilja på presentation och innehåll. Syftet med en CSS-fil är därmed att berätta för webbläsaren hur innehållet i ett HTML/XHTML/XML-dokument ska visas.

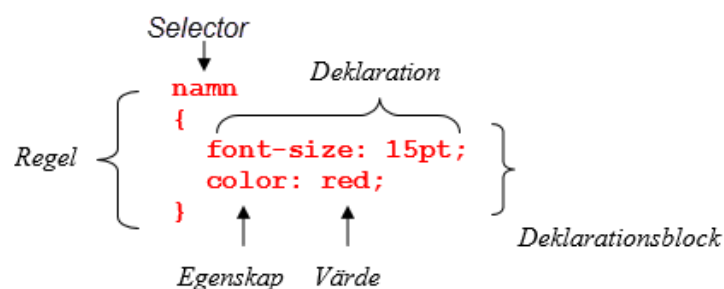
I HTML/XHTML finns som bekant en fastställd uppsättning av element med givna namn som används och webbläsaren som visar dokumentet vet hur dessa element ska presenteras. Detta gäller inte för XML-dokument där vi kan ha egna namn på elementen. Webbläsaren kan omöjligt veta vad ett `namn`-element innebär och hur det ska presenteras. Av denna anledning är det ännu viktigare att applicera en stilmall på ett XML-dokument.

Tillskillnad mot XML skiljer CSS inte mellan stora och små bokstäver. I kursen försöker vi uteslutande att använda små bokstäver i CSS-dokument.

I denna lektion kommer vi att gå igenom en hel del egenskaper i CSS där egenskapens namn listas tillsammans med dess möjliga värden och standardvärde. För en del egenskaper finns väldigt många olika värden och för dessa kommer inte alla värden att tas upp, utan endast de mest vanliga värdena.

1.2 Grundläggande uppbyggnad

En stilmall består av en uppsättning regler. Reglerna specificeras för de olika elementen i ett XHTML/XML-dokument och är uppbyggda på följande sätt:



I exemplet ovan visas regeln (namn) med röd text och består av en *selektor* som refererar till det element regeln gäller för. Regeln ovan bestämmer hur element med namnet `namn` ska formateras. Vidare har vi ett *deklarationsblock* i vilket formateringarna skrivs.

Deklarationsblocket börjar med ett `{` och avslutas med ett `}`. En *deklaration* består av en *egenskap* (till exempel `color`) och ett tillhörande *värde*. Ett kolon(`:`) används för att skilja mellan egenskapen och värdet. Här säger vi att innehållet i ett `namn`-elementet ska vara `röd` och ha `15 pt` som textstorlek. Ett deklarationsblock kan bestå av ingen, en eller flera deklarationer. Deklarationerna i ett deklarationsblock skiljs från varandra med ett semikolon (`;`). Den sista deklarationen i ett deklarationsblock behöver inte avslutas med ett semikolon.

I figuren ovan har vi en regel som gäller för ett element, men en regel kan även användas på flera element samtidigt. Selektorerna separeras då med ett kommatecken enligt nedan:

```
adress, postnr { color: blue; }
```

Det är även möjligt att lägga till kommentarer i en CSS-fil. Kommentarer startar med `/*` och avslutas med `*/`

Exempel

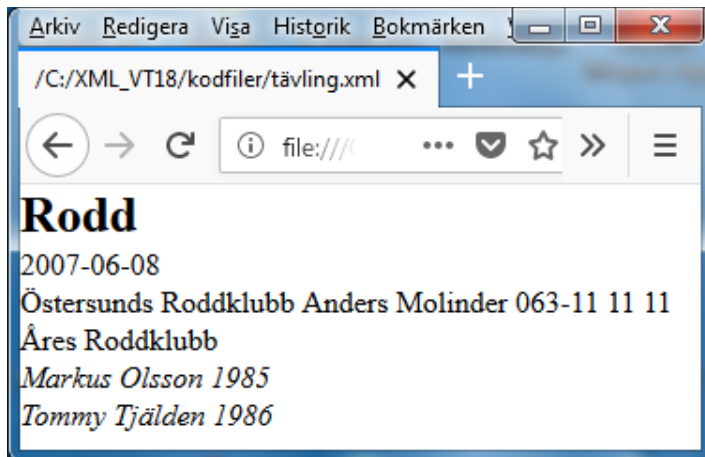
Vi ska nu titta på ett exempel som kommer att användas genom hela lektionen. Utgångspunkten är en tävling i rodd mellan flera lag och där varje lag består av två deltagare. Vi ska applicera en stillmall för att formatera innehållet i XML-dokumentet när det visas i en webbläsare. Nedan visas XML-dokumentet

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="tävling.css"?>
<tävling>
  <gren>Rodd</gren>
  <datum>2007-06-08</datum>
  <arrangör>
    <klubb>Östersunds Roddklubb</klubb>
    <kontakt>
      <namn>Anders Molinder</namn>
      <telefon>063-11 11 11</telefon>
    </kontakt>
  </arrangör>
  <lag id="10">
    <klubb>Åres Roddklubb</klubb>
    <deltagare>
      <namn>Markus Olsson</namn>
      <födelsedatum>1985</födelsedatum>
    </deltagare>
    <deltagare>
      <namn>Tommy Tjällden</namn>
      <födelsedatum>1986</födelsedatum>
    </deltagare>
  </lag>
  <!--Fler lag-->
</tävling>
```

Vi skapar även en enkel stilmall enligt nedan ([tävling.css](#)):

```
gren { display:
      block;
      font-size:
      20pt;
      font-
      weight:
      bold;
}
datum { display:
block; } arrangör {
display: block; }
lag { display:
block; } deltagare {
display: block;
font-style: italic;
}
```

Om vi nu öppnar [tävling.xml](#) i en webbläsare ser det ut så här (i Firefox)



Om ditt dokument inte formateras enligt ovan kan det innebära att du har skrivit något fel i CSS-filen. Några felmeddelanden visas inte om CSS-filen innehåller fel, utan det enda som sker när webbläsaren inte förstår vad som står i CSS-filen är att regeln i fråga ignoreras.

De vanligaste felen är felaktig användning av semikolon, `;`, och kolon, `:`, användning av parenteser `()` i stället för klamrar `{}`, eller ett fel med själva CSS-länken i XML-dokumentet. Kontrollera därför dessa saker om du får något problem.

1.3 Länka CSS-filen till ett XML-dokument

På rad 2 i XML-dokumentet ovan ([tävling.xml](#)) använder vi en processinstruktion för att länka stilmallen [tävling.css](#) till dokumentet:

```
<?xml-stylesheet type="text/css" href="tävling.css"?>
```

Denna rad ska stå i prologen, alltså före rootelementet. Som du kommer ihåg från förra lektionen betyder `xml-stylesheet` att en stilmall ska kopplas till XML-dokumentet. Stilmallen kan antingen vara ett CSS-dokument eller ett XSLT-dokument, något som anges av

attributet `type`. Här har attributet `type` värdet `text/css`. Det sista attributet, `href`, anger filnamnet till stilmallen. I exemplet står det bara ett filnamn här, något som betyder att CSS-filen måste ligga i samma mapp som XML-dokumentet. Om stilmallen ligger i en annan mapp måste du specificera vilken mapp detta är med antingen en relativ eller en absolut URL.

1.4 Egenskapen display

För att bestämma hur innehållet i de olika elementen ska visas kan bland annat egenskapen `display` användas:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>display</code>	<code>block</code> , <code>inline</code> , <code>list-item</code> , <code>none</code> (med flera)	<code>inline</code>

`display` kontrollerar om webbläsaren visar innehållet i ett element. Om egenskapen värdet `block` kommer en radbrytning att skrivas före och efter innehållet. Det kan jämföras med elementet `p` i XHTML.

Värdet `inline` skriver ut innehållet i elementet på samma rad. Detta värde används som standard om inget annat värde har angetts (det är till exempel därför födelseåret skrivs på samma rad som deltagarnas namn i bilden ovan). För att gömma hela innehållet kan värdet `none` användas.

1.5 Mer om selektorer

Ovan så vi att en selektor är det som står framför själva regeln. Nedan visas selektorn `namn`:

```
namn { color: blue; }
```

I detta kapitel ska vi titta närmare på en del sätt att använda selektorer. Var uppmärksam på att det kan förekomma viss skillnad av användandet i olika webbläsare. Nedan kommer tävlings-exemplet som beskrevs i kapitel 1.2 att användas.

Universell selektor

Det finns en selektor som kallas universell selektor, nämligen `*`. Denna kan användas helt ensam eller tillsammans med andra selektorer. Nedan används den universella selektorn för att allt innehåll i alla element i XML-dokumentet ska visas i blått:

```
* { color: blue; }
```

Underordnad selektor

En underordnad (decendant) selektor är en selektor med vilken vi närmare kan ange var ett element, som ska formateras, är placerat i dokumentet. Som exempel kan vi tänka oss att textfärgen på deltagarnas namn ska skrivas med blått. Nu finns

det flera `namn`-element i XML-dokumentet. Både namnet på en deltagare och namnet på arrangörens kontaktperson.

Om vi skapar en regel enligt `namn { color: blue; }` kommer både deltagarens namn och kontaktpersonens namn att visas i blått. Vi måste därför använda en underordnad selektor så att endast deltagarens namn påverkas. En underordnad selektor skrivs enligt följande form: `förälderelement barnelement { ... }`

För att nu endast ge deltagarens namn blå färg skapar vi följande regel:

```
deltagare namn { color: blue; }
```

Observera att detta påverkar alla `namn`-element som är ättlingar till `deltagare` och inte endast direkta barn till `deltagare`. I webbläsaren ser det nu ut så här:



Barnselektor

Om vi vill ange en stil för endast ett bestämt element som är ett direkt barn till ett annat element används symbolen `>` enligt: `förälder > barn { ... }`

I stället för att använda en underordnad selektor för att endast färglägga deltagarnas namn kan en barnselektor användas enligt nedan: `deltagare > namn { color: blue; }`

Samma resultat erhålls nu som bilden på förra sidan visar.

Syskonselektor

En annan typ av selektor är angränsande syskon-selektor (adjacent sibling). Denna selektor skrivs enligt formen:

```
element1 + element2 { ... }
```

Här är elementen `element1` och `element2` angränsande syskon med varandra. Med det menas att de har samma förälderelement och att `element1` omedelbart efterföljs av `element2`. Elementet `element2` kommer endast att påverkas av stilen om den direkt följer efter `element1`. Följande exempel illustrerar detta med angränsande syskon:

```
<stycke>Svensken <fet>William Nylander</fet> gjorde matchens enda mål när
```

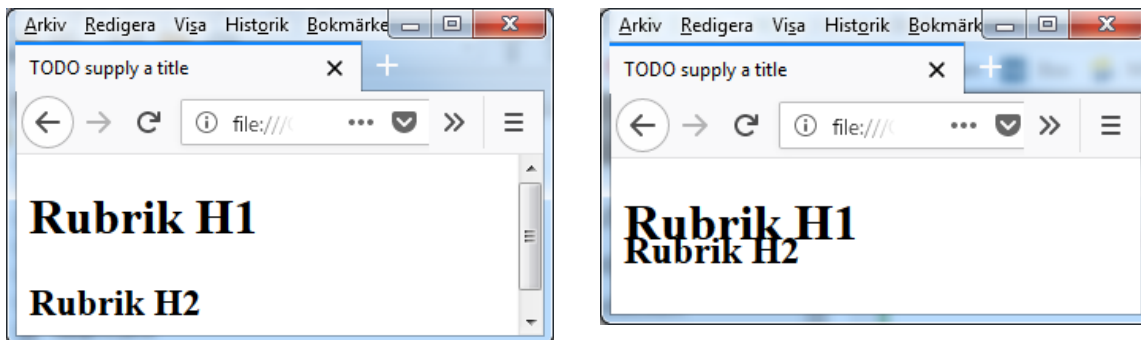
`<kursiv>Toronto</kursiv> tog ledningen med 3-0 i matcher mot Buffalo. Läs mer <länk url="http://www.aftonbladet.se">här</länk>.</stycke>`

Här finns tre olika element, `fet`-, `kursiv`-, och `länk`, som alla är syskon med varandra. Vi kan nu säga att:

- Elementen `fet` och `kursiv` är angränsande syskonelement
- Elementen `kursiv` och `länk` är angränsande syskonelement
- Elementen `fet` och `länk` inte är angränsande syskonelement

Ett exempel där syskonselektorer är användbara är bland annat vid rubriker. Det är vanligt att extra utrymme används före och efter en rubrik. Ibland kan det bli lite väl mycket utrymme vid tillfällen där XHTML's `h2`-element omedelbart följer efter ett `h1`-element. För att reducera detta utrymme mellan elementen kan vi skriva: `h1 + h2 { margin-top: -10mm; }`

Nedan visas hur resultatet ser ut i Firefox före (vänstra bilden) och efter (högra bilden) användningen av regeln ovan. Observera att `h2` endast påverkas om den direkt följer efter `h1`.



Attributselektor

Det finns även något som heter attributselektor. Denna används för att välja de element som har ett visst attribut eller ett visst värde på ett attribut. Nedan kommer vi att titta på två av dessa:

- Selektor där ett visst attribut måste finnas. Alltså en CSS-regel för de element som har ett visst attribut:

```
selektor [attribut] { egenskap: värde; }
```

Ett exempel är att vi vill att alla lag som har ett id registrera ska visas i

```
rött: lag [id] { color: red; }
```

- Selektor där ett visst värde till ett visst attribut måste finnas:

```
selektor [attribut="värde"] { egenskap: värde; }
```


Ett exempel här är att alla mobilnummer till en person ska visas med grönt. Vi kan då tänka oss att vi har ett element `telefon` med ett attribut som heter `type`. Attributet kan bland annat ha värdet `mobil`:

```
telefon [type="mobil"] { color: green; }
```

Tabellen nedan summerar de olika typer av selektorer som finns:

<i>Selektor</i>	<i>Betydelse</i>
<code>*</code>	Universell selektor. Gäller för alla element.
<code>a</code>	Enkel selektor. Gäller för alla <code>a</code> .
<code>a, b</code>	Multipel enkel selektor. Gäller för alla <code>a</code> och <code>b</code> .
<code>a b</code>	Underordnad selektor. Gäller för alla <code>b</code> som är ättlingar (inklusive barn) till <code>a</code> .
<code>a > b</code>	Barnselektor. Gäller för alla <code>b</code> som är ett barn till <code>a</code> .
<code>a + b</code>	Angränsande syskon-selektor. Gäller för alla <code>b</code> som direkt följer efter <code>a</code> .
<code>a[b]</code>	Attributselektor. Gäller för alla <code>a</code> som har attributet <code>b</code> . <code>*</code> kan användas i stället för <code>a</code> och gäller då alla element som har attributet <code>b</code> . <code>a[b="c"]</code>
	Attributselektor. Gäller för alla <code>a</code> som har attributet <code>b</code> vars värde är <code>c</code> . <code>*</code> kan användas i stället för <code>a</code> och gäller då alla element som har attributet <code>b</code> vars värde är <code>c</code> .

1.6 Arv i CSS

När en stilmall skapas kan en stilmall länkas till en annan stilmall. Flera olika stilmallar kan även länkas till ett och samma dokument. Det kan då hända att flera stilmallar för XML-dokumentet specificerar en regel för samma element. Det måste därför finnas något som avgör vilka regler som gäller framför andra. Det viktigaste är att mer specifika regler gäller före mer generella regler.

När en stilmall upprättas används en funktion som kallas arv. Det innebär att egenskaper som tillägnas ett visst element ärvs till alla underliggande element (ättlingar). Detta kan illustreras genom att för selektorn `lag` använda egenskapen `font-size` med värdet `14pt` för att ändra textstorleken:

```
lag { font-size: 14pt; }
```

Nu kommer alla ättlingar till elementet `lag`, som till exempel elementet `födelseår`, att skrivas ut med textstorleken `14pt`. Det är detta som kallas arv. Om vi nu vill att elementet `födelseår` inte ska ha textstorleken `14pt`, utan kanske `10pt` i stället, måste detta specificeras specifikt för detta element:

```
lag { font-size: 14pt; }  
födelseår { font-size: 10pt; }
```

Nu kommer elementet `födelseår` att ha en annan textstorlek än resterande ättlingar till `lag`. Detta visar att specifika regler gäller framför mer generella regler.

1.7 Värden

Från exemplet ovan ser vi att enheten `pt` anger hur stor texten ska vara. Det finns flera sätt att ange ett värde för en egenskap. Innan vi går in på de olika typer av egenskaper som finns i CSS är det viktigt att känna till några enheter och vad de betyder.

Normalt anges ett värde antingen som ett absolut eller ett relativt värde.

- **Absoluta värden.** Här sätts ett bestämt värde på egenskapen oberoende av vilka värden som gäller i resten av dokumentet. Vi behöver alltså inte räkna fram något värde utifrån de andra elementens värden. Ett exempel på detta kan vara när en viss färg anges. Säger vi att texten ska vara blå finns inga tvivel på vilken färgen ska vara. Andra exempel kan vara absoluta värden på längdenheter. Här följer en lista över enheter som används när absoluta värden anges:

`in`: tum (inches)

`cm`: centimeter

`mm`: millimeter

`pt`: points (i CSS2 är detta samma som 1/72 av en in) `pc`: picas (1 pi = 12 pt)

`px`: pixel

- **Relativa värden.** Specificerar ett värde i förhållande till andra värden. Ett exempel kan vara att en text ska vara `15%` större än en annan. Ett annat exempel är egenskaper där vi kan ange värdet `bold` (tjockare). Då kommer texten bli en storlek tjockare än det värde som redan finns. Nedan listas några enheter som är relativa:

`em`: "textstorleken" för den aktuella texten där `1em` är den textstorlek som förälderelementet har. Denna kan förändras relativt genom att skriva till exempel `2em`

(större) eller `0.5em` (mindre). `ex`: används på samma sätt som `em`, men gäller för höjden.

Procent kan även användas vid olika tillfällen. Detta kommer att vara ett relativt värde som måste beräknas i förhållande till det värde den aktuella egenskapen annars skulle haft.

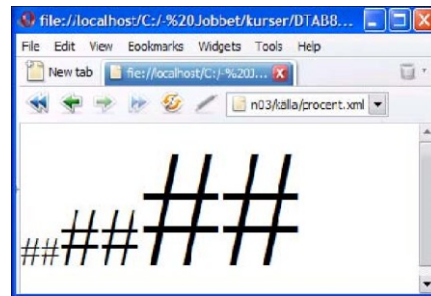
Ett värde kan både vara heltal och decimaltal. Om ett decimaltal ska användas måste en punkt (.) användas som avskiljare, till exempel `0.5` från ovan.

Om ingen enhet anges används `pixlar (px)` i CSS. Nedan visas ett exempel på ett dokument där procent används.

```
<stycke><hälften>##</hälften>##<dubbelt>##</dubbelt></stycke>
```

Låt oss nu appliceras följande stilmall på dokumentet ovan. Resultatet ser du i bilden nedan:

```
stycke { font-size: 50pt; }
dubbelt { font-size: 200%; }
hälften { font-size: 50%; }
```



1.8 Egenskaper för färg och text

Vi kommer nu att titta närmare på två egenskaper som finns i CSS. Det är egenskaper som styr färgsättning på texter och bakgrunder, samt hur typsnitt och storlek på texter ändras.

1.8.1 Färger

Det är två typer av färgsättning vi ska titta närmare på, nämligen färg på text och färg på bakgrunden. Nedan ser du specifikationen för var och en av dem:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>color</code>	Anges som namn på färg, RGB eller hexadecimalt tal	Skiljer mellan olika webbläsare
<code>background</code>	Anges som namn på färg, RGB- eller hexadecimalt, transparent	transparent

Värdet på färgen kan anges på flera sätt. Många av färgerna har namn knutna till sig och kan användas direkt med detta namn. Exempel på detta har vi sett ovan då vi gav texten färgen blå (`blue`). Alternativt kan vi använda färgens RGB-värde eller dess hexadecimala värde, där det anges ett värde för var och en av färgerna röd, grön och blå. Ett exempel på hexadecimalt värde är `#FF0000` (röd, observera att `#` ska anges före värdet) och för RGB-värde; `rgb(50,50,150)`.

Ett utmärkt verktyg för att experimentera med färger och färgpaletter samt att få reda på en färgs RGB-värde och hexadecimala-värden finns på följande hemsida.

https://www.w3schools.com/colors/colors_picker.asp

Med hjälp av egenskaperna `color` och `background` kan vi nu sätta bakgrundsfärgen i exemplet till gult och texten till blått:

```
tävling {
    display: block;
    color: #0000FF;
    background: yellow;
}
```

1.8.2 Text

Det finns flera egenskaper som justerar hur text ska visas. I ett ordbehandlingsprogram känner du till att det är möjligt att ange teckensnitt, teckenstorlek, stil och tjocklek. Detta är även möjligt i CSS. Nedan finner du specifikation för de egenskaper som justerar detta:

Egenskap	Möjliga värden	Standardvärde
font-family	serif, sans-serif, cursive, fantasy, monospace eller namn på teckenstilen som till exempel "Times New Roman"	Times New Roman
font-size	i punkter (pt), pixlar (px) eller ett av värdena: small, large, x-large, xx-small eller medium	12pt
font-weight	bold, normal eller bolder, lighter. Ett värde mellan 100-900	normal
font-style	italic, normal eller oblique	normal

Det finns flera sätt att använda dessa egenskaper på (med undantag för `font-style`).

För teckenstorlek (`font-size`) och tjocklek (`font-weight`) är det möjligt att använda både relativa och absoluta värden. Vi ska nu införa förändringar i tre regler i vår stilmall. De förändringar som kommer att göras är att `lag` ska skrivas ut med teckenstilen `sans-serif` och i storleken `medium`. Sen tidigare skrivs deltagarna ut i kursiv stil (`italic`). Här ska vi göra ett tillägg så att texten även blir tjockare (`bolder`). Observera att både deltagarnas namn och arrangörens namn kommer att påverkas av denna förändring. De regler som påverkas visas nedan:

```
lag { display:block;
      font-family: sans-serif; font-size: medium;
}
deltagare {
  display:block;
  font-style: italic;
}
namn {
  font-weight: bolder;
}
```



Tänk på att för `font-family` måste alla värden längre än ett ord, som `Times New Roman`, omslutas av dubbla citattecken: `"Times New Roman"`.

1.9 Egenskaper för utrymme

När webbläsaren visar ett XML-dokument placeras innehållet i de olika elementen i separata boxar. Precis som elementen i ett XML-dokument är nästlade, nästlas boxarna i webbläsaren. I CSS kan utrymmet för en box påverkas på olika sätt genom att påverka boxens egenskaper:

Margin – Sätter ut en genomskinlig kant runt boxen

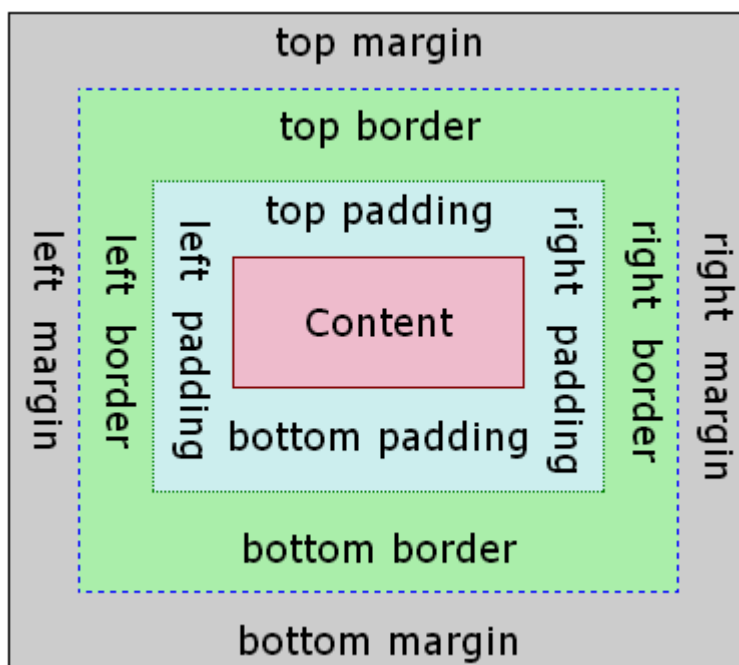
Border – Sätter ut en synlig kant runt boxen (kan döljas)

Padding – Lägger till ett tomt område utanför boxens innehåll, men innanför den synliga ramen (border)

Width – Anger boxens bredd (endast området elementets innehåll tar upp)

Height – Anger boxens höjd (endast området elementets innehåll tar upp)

Relativ och absolut positionering – Anger boxens placering på en sida



1.9.1 Margin

Margin är mellanrummet mellan ett elements "bounding box" och ett närliggande elements "bounding box". Det finns fem olika egenskaper som reglerar margin:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>margin,</code> <code>margin-top,</code> <code>margin-bottom,</code> <code>margin-left,</code> <code>margin-right</code>	px, em eller %	0

Som du ser kan dessa egenskaper ha både absoluta värden och relativa värden.

Vi kan lägga till egenskapen `margin-left` till regeln `lag` för att flytta texten något mot höger:

```
lag { display: block; font-  
      family: sans serif;  
      font-size: medium;  
      margin-left: 12px;  
}
```

Eventuellt kan vi använda ett relativt värde enligt:

```
lag { display: block;  
      font-family: sans  
      serif;  
      font-size: medium;  
      margin-left: 2em;  
}
```

Ovan gavs endast ett värde till en av sidorna. Det är givetvis möjligt att ge ett värde till alla sidor på en gång genom att använda egenskapen `margin`:

```
lag { display: block; font-  
      family: sans-serif;  
      font-size: medium;  
      margin: 20px;  
}
```

Vid vissa tillfällen vill vi kanske ha olika storlekar på marginalerna och vi kan då skriva:

```
lag { display: block;  
      font-family:  
      sans-serif; font-  
      size: medium;  
      margin: 20px  
      12px;  
}
```

Här anges två värden för egenskapen `margin`. Vid dessa tillfällen finns det speciella regler som säger vilka marginaler som får de olika värdena:

- När endast ett värde är satt gäller värdet för alla marginaler.
- När två värden anges används det första värdet för `margin-top` och `margin-bottom`, medan det andra värdet gäller för `margin-left` och `margin-right`.

När tre värden anges används första värdet för `margin-top`, det andra värdet för `margin-left` och `margin-right`, medan det sista värdet gäller för `margin-bottom`.

- När fyra värden anges gäller de i ordningsföljden: `top`, `right`, `bottom` och `left`.
- Dessa regler gäller även för egenskaperna `padding` och `border` (med flera).
- Det är viktigt att veta vad man gör när egenskapen `margin` används. Om vi nu lägger till en marginal för deltagarinformationen kommer den att ha dubbla vänstermarginaler:


```

lag { display: block;
      font-family: sans-
        serif;
      font-size: medium;
      margin: 20px 12px;
    }
deltagare{
  display: block;
  font-style:
    italic;
  margin: 14px;
}

```

Nu har vi lagt till marginaler runt hela `lag`-elementet och hela `deltagare`-elementet. Detta gör att deltagarnamnen totalt får en marginal på 34px i förhållande till sidans vänstra kant (men så klart 14px marginal i förhållande till boxen för `lag`-elementet):

1.9.2 Border

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>border-width</code>	px, em, pt eller ett av värdena thin, medium, thick	medium
<code>border-style</code>	none, solid, dotted, dashed, hidden, double, groove, ridge, inset, outset	none
 <code>border-color</code>	Anges som namn på färg, RGB eller hexadecimalt tal	Samma som färgen på elementet
<code>border</code> , <code>border-right</code> , <code>-left</code> , <code>-top</code> , <code>-bottom</code>	Kan ha ett värde för var och en av egenskaperna ovan	0

För att illustrera de olika egenskaperna har vi följande XML-dokument (`ram.xml`):

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="ram.css"?>
<test>
  <vitestar>Test av olika ramar</vitestar>
</test>

```

Vidare har vi CSS-filen `ram.css`:

```

vitestar { font-size: 24px;
          border-style: solid;
          }

```

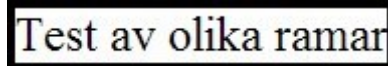
Här har vi angett värdet `solid` för egenskapen `border-style`. I exemplen nedan kommer vi att ändra värdet på denna egenskap för att se hur detta kommer att påverka visningen av XML-dokumentet.

Vidare har vi CSS-filen `ram.css`:

```
vitestar { font-size: 24px;
border-style: solid; }
```

Här har vi angett värdet `solid` för egenskapen `border-style`. I exemplen nedan kommer vi att ändra värdet på denna egenskap för att se hur detta kommer att påverka visningen av XML dokumentet.

`{ border-style: solid }` – ser ut så här i webbläsaren:



`{ border-style: dotted }` – ser ut så här i webbläsaren:

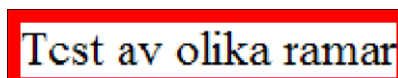


`{ border-style: inset }` – ser ut så här i webbläsaren:



Genom att använda egenskapen `border-width` kan vi ändra på ramens tjocklek. Kombinerar detta med `border-color` (ramens färg) och `border-style` kan vi bland annat få fram följande exempel:

```
vitestar
{ font-size: 24px;
border-style: solid;
border-width: thick;
border-color: red; }
```



Ett annat sätt att öka ramens tjocklek är att ange värdet i enheten `px` (pixlar):

```
vitestar {
font-size: 24px; border-
style: solid; border-
width: 10px; border-
color: red;
}
```



Precis som med `margin` kan vi även med `border` ange flera värden samtidigt. Ordningsföljden är:

```
border: border-width border-style border-
color Sista exemplet ovan kan därför skrivas som:
vitestar { font-size: 24px;
border: 15px solid
red; }
```


1.9.3 Padding

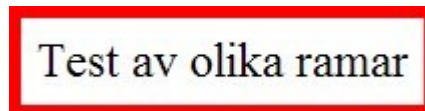
Egenskapen `padding` skapar ett mellanrum mellan elementets innehåll och ramen. Följande egenskaper finns:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>padding,</code> <code>padding-top,</code> <code>padding-bottom,</code> <code>padding-left,</code> <code>padding-right</code>	<code>px</code> , <code>em</code> eller <code>%</code>	0

Padding används på samma sätt som margin. Det är samma regler för hur egenskaper och värden anges.

Vi fortsätter med de exempel som användes i föregående kapitel (border). Vi kommer nu att lägga till egenskapen `padding` i regeln `vitestar` för att lägga in ett mellanrum mellan ramen och texten (visas med fet stil):

```
vitestar { font-size: 24px;  
  border-style: solid;  
  border-width: thick;  
  border-color: red;  
  padding: 8px; }
```



Den sista raden i exemplet ovan skapar mer avstånd runt texten. På samma sätt som med margin kan vi för padding ange olika värden för de olika kanterna. Detta får du själv testa.

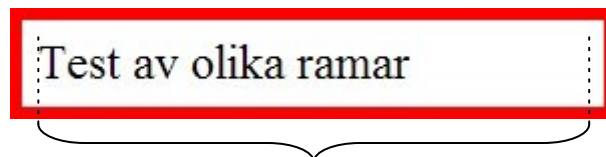
1.9.4 Size

För att ange höjd och bredd på en box används följande egenskaper:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>width,</code> <code>height</code>	<code>cm</code> , <code>in</code> , <code>auto</code> eller <code>%</code>	0

Vi fortsätter med samma exempel och anger att bredden ska vara 8 cm:

```
vitestar  
{ font-size: 24px;  
  border-style: solid;  
  border-width: thick;  
  border-color: red;  
  padding: 8px;  
  width: 8cm; }
```



Kom ihåg att när vi anger `width` och `height` för en box så är det innehållets höjd och bredd som påverkas. I exemplet ovan är den totala bredden inklusive den röda ramen större än 8cm.

1.10 Positionering

Fram till nu har vi använt egenskaperna `margin`, `border` och `padding` för att bestämma avståndet mellan olika element. Alternativt kan en position för de olika elementen anges:

- **Relativ position:** Anger positionen för en box relativt till dess normala position. Ett annat sätt att se det på är att elementets innehåll flyttas från var det skulle ha varit om du inte förändrat något.
- **Absolut position:** Här plockas boxen ut från hierarkin av boxar och ger den en egen position oberoende av hur andra boxar är placerade. Ett annat sätt att se det på är att boxen ges en fast position relativt webbläsarens sida. Är man inte försiktig är det lätt att boxarna flyttas så att innehållet ligger ovanpå varandra eller överlappar varandra. Fördelen är att boxen alltid kommer att vara positionerad på samma plats oberoende om användaren förstorar eller förminskar webbläsarens fönster.
- **Fixerad position:** Här ges boxen sin egen position som kommer att vara fast relativt webbläsarens fönster oberoende på hur användaren scrollar och hur innehållet förändras. Anges en fixerad position kommer innehållet att vara placerad på exakt samma ställe relativt webbläsarens fönster/kant även om användaren scrollar innehållet (fungerar inte i Internet Explorer).

Denna typ av positionering används inte alltför ofta. Som regel används `margin`, `border` och `padding` för att bestämma position.

För att använda ovanstående sätt att positionera används följande egenskap:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>position</code>	<code>static</code> , <code>relative</code> , <code>absolute</code> eller <code>fixed</code>	<code>static</code>

Värdet `static` innebär "vanlig" position medan de övriga har förklarats ovan. Egenskaperna ovan anger endast vilken typ av positionering som ska användas. Utöver detta måste även värdet för själva positionen anges. Detta görs med följande egenskaper:

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>top</code> , <code>right</code> , <code>bottom</code> , <code>left</code>	<code>in</code> , <code>cm</code> , <code>mm</code> , <code>pt</code> , <code>pc</code> , <code>px</code> , <code>em</code> , <code>ex</code> eller %	0

Vi fortsätter nu med tävlings-exemplet och utgår från koden så som den såg ut innan egenskapen `margin` användes. Nu kommer vi att sätta en absolut position på elementet `klubb` för de lag som ska tävla (inte för klubben som arrangerar tävlingen):

```
lag klubb{
    text-align: center;
```

```

position: absolute;
left: 200px;
}

```



Lägg märke till att elementet `klubb` nu har lyfts ut ur hierarkin av element så att texten i elementet `deltagare` flyttas upp.

1.11 Lägga till bilder

Det är inte ovanligt att man vill ha med bilder när ett XML-dokument visas. Det kan antingen vara en bakgrundsbild eller en vanlig bild som ska visas i sin helhet.

<i>Egenskap</i>	<i>Möjliga värden</i>	<i>Standardvärde</i>
<code>background-image</code>	<code>url</code> eller <code>none</code>	<code>none</code>

För att lägga till en bakgrundsbild används egenskapen `background-image`. Vi fortsätter med vårt tävlings-exempel (tar bort den gula bakgrundsfärgen samt marginaler först). I exemplet har vi elementet `lag` som gäller för varje lag som ska delta i tävlingen. Vi vill att bilden (`rodd.gif`) nedan ska läggas in som en bakgrundsbild för själva texten om laget.

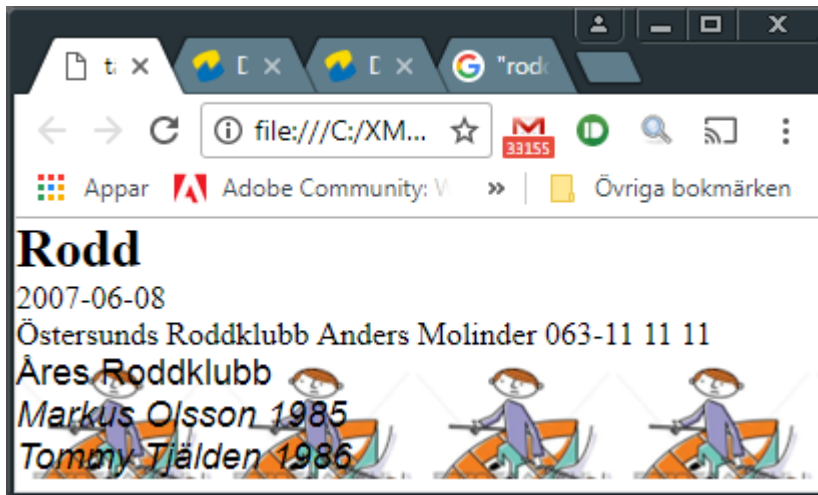
För att göra detta lägger vi till en ny egenskap i CSS-filen som bara gäller för elementet `lag`:

```

lag { display: block;
      font-family: sans-
      serif; font-size:
      large;
      background-image: url(rodd.gif);
}

```

Öppnar vi dokumentet i webbläsaren får vi följande:



Vi kan nu i stället prova att lägga bilden till höger om texten i elementet `lag`. Enklaste sättet att göra detta på är att lägga till en `tomelementstagg` i XML-dokumentet och därefter lägga till en regel för detta element i CSS-filen. Som första barnelement till elementet `lag` lägger vi in det nya elementet `bild`. Därefter uppdaterar vi CSS-filen med följande regel:

```
bild {
    display: block;
    background-image: url('rodd4.png');
    background-repeat: no-repeat;
    width: 100px; height: 97px;
```

} Som du ser ovan har en bredd och höjd angetts, detta är den storlek bilden har. Tar vi bort nu bort egenskapen `background-image` från `lag` ser det ut så här i webbläsaren:



Egenskapen `float` används för att låta ett innehåll i ett element "flyta" vid sidan om ett annat innehåll. Det kan vara både bilder och textinnehåll.

Vi har här gått lite grunder. Det finns betydligt mer att läsa och lära om CSS. Bästa sättet att lära är att prova och återigen prova alla olika egenskaper och ge dessa olika värden för att se hur det påverkar innehållet.

Den senaste versionen av CSS är CSS3 och den utökar möjligheterna ytterligare med bl.a responsiva sidor.