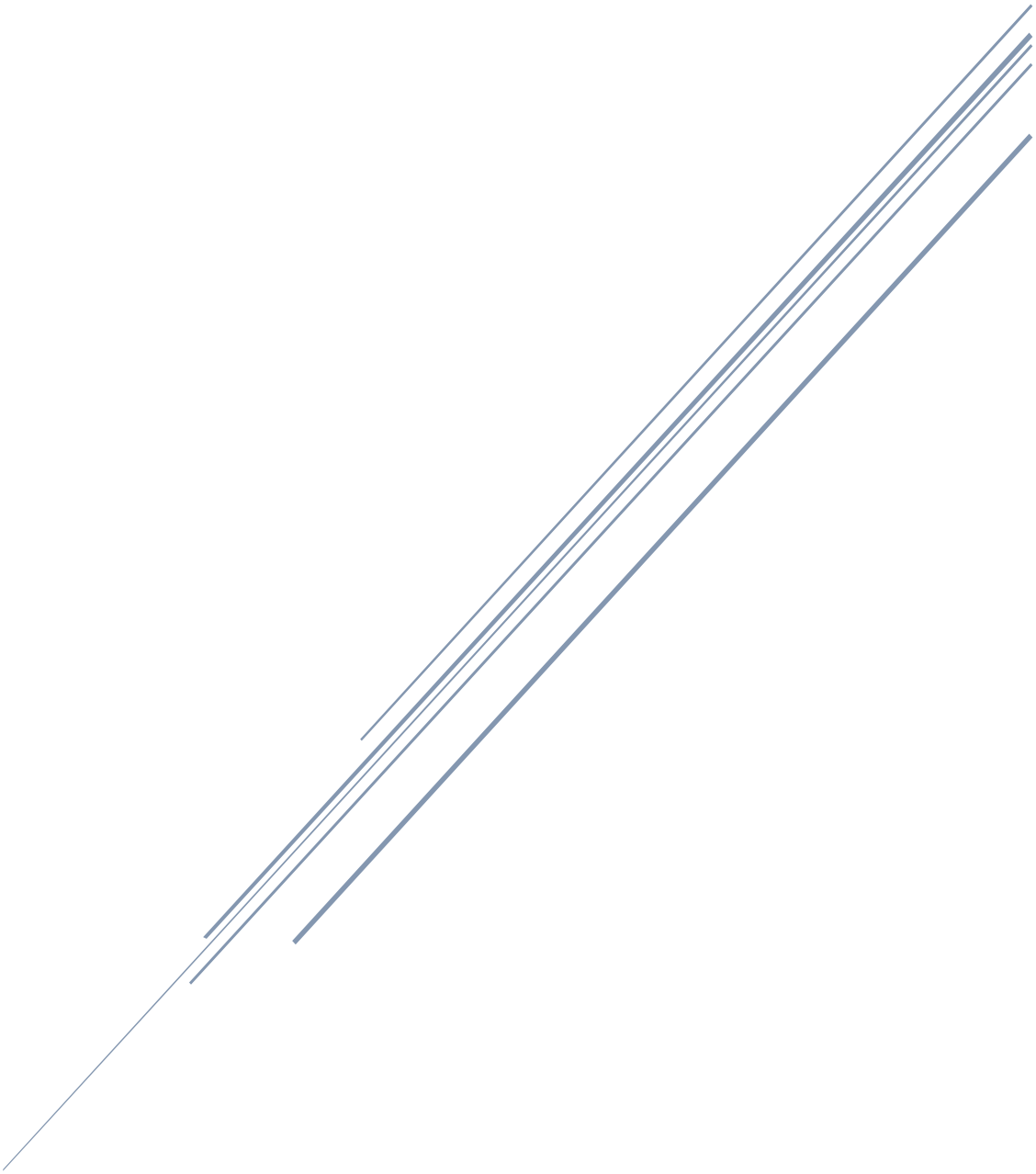
LÄSNING MOMENT 1



Mittuniversitetet
XML

1 Skapa och visa ett första XML-dokument

Sammanfattning: I kursens första lektion går vi igenom grunderna i XML. Du kommer att lära dig att skapa ett enkelt XML-dokument och vilka regler som gäller när detta görs. Vidare kommer du bland annat att lära dig hur attribut används och hur tecken kan tolkas olika beroende på språk.

1.1 Inledning

XML står för Extensible Markup Language och namnet säger ganska mycket om vad XML innebär. Extensible (uttänjbart) betyder att man kan skapa de element man själv vill använda, i motsatts till HTML där det finns ett fastställt antal element som kan användas.

HTML är dålig på att beskriva en hel del olika typer av dokument, som till exempel:

- Dokument som inte består av de typiska HTML-taggarna som `head`, `h1`, `ul`, `table` och liknande. Exempel kan vara dokument för att presentera matematiska formler.

- Dokument som du önskar att presentera i en trädliknande struktur. Detta kan till exempel vara aktuellt för att organisera en bok med innehållsförteckning, kapitel och liknande.

Detta är svårt att beskriva med hjälp av HTML på grund av nästling av element och text.

Här kommer XML att fungera långt bättre då XML kan representera alla typer av dokument på det sätt man själv önskar.

XML kan ses på som en plats att lagra data. XML kan också kallas en ”databank”, där information i XML-dokumentet kan presenteras på många sätt och i olika format. Data i ett

XML-dokument kan användas i allt från mer avancerade program till enkla webbsidor.

XML-dokument kan också användas till att utväxla data mellan olika system. XML blir då ett systemoberoende överföringsformat.

Ofta är det tre olika typer av filer som används när XML-data ska användas i ett program, som till exempel när data ska visas i en webbläsare:

1. XML-dokumentet (.xml) som innehåller data (innehållet). Detta representerar den semantiska delen.

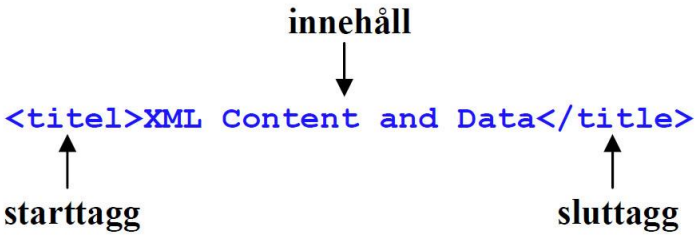
2. En stillmall, som till exempel Cascading Stylesheet (CSS), som bestämmer hur data ska formateras och presenteras. Många olika stilmallar kan användas till samma XMLdokument så att samma data kan visas på flera olika sätt eller på olika typer av utrustning (mobiltelefoner, handhållna datorer, webbläsare, skärmar och liknande). Det finns flera olika standarder som har med stilmallar att göra. I kursen kommer vi att titta på CSS (lektion 3) och XSL, som är en mer avancerad standard för presentation av XML-dokument, tittar vi på i lektion 7.

3. En dokumenttypsdefinition som Document Type Definition (DTD) eller XML Schema (XSD) används för att specificera regler för hur element, attribut och innehåll ska struktureras och relateras. DTD tittar vi mer på i framför allt lektion 5 och 6.

Ett XML-dokument (.xml-filen) behöver inte nödvändigtvis knytas till varken en stilmall eller ett schema. För att utnyttja XML-teknologin fullt ut kan XML-dokumentet dock vid behov knytas till ett schema (en DTD eller XSD), samt en eller flera stilmallar.

1.1.1 Element i ett XML-dokument

Element är den viktigaste delen i ett XML-dokument och är uppbyggt på samma sätt som i ett HTML/XHTML-dokument, som du mest troligt känner till sen förr. Ett element består av en *starttagg*, ett *innehåll* och en *sluttagg*:



Innehållet i ett element beskrivs med namnet på taggen (start- och sluttagg). Därför sägs det ofta att XML är självbeskrivande och precis detta gör XML så praktiskt att använda för till exempel utbyte av data. Namnet på elementet kallas kort och gott för elementnamn. Behövervi lagra data om en person kan vi till exempel använda elementnamnen person, namn eller telefon.
I XML kan vi också använda oss av element utan innehåll, precis som i HTML där till exempel br eller hr kan användas. Ett sådant element kallas för tomelementstagg (empty element tag). Dessa element skrivs enligt: <elementnamn />

För att strukturera data i ett XML-dokument kan elementen nästlas med varandra (kapsla in element i varandra) enligt:

```
<bok>
     <titel>XML Content and Data</titel>
     <författare>Kelly Carey</författare>
     <författare>Stanko Blatnik</författare>
     <sidor>408</sidor>
     <pris>410 kronor</pris>
</bok>
```

1.2 Skapa ett XML-dokument

1.2.1 Tillvägagångssätt

Nu är du redo att skapa ditt första XML-dokument och prova hur ett XML-dokument visas i en webbläsare. I kapitel 1.2.2 går vi närmare in på själva skapandet av dokument. Börja med att skriva in följande kod i en texteditor

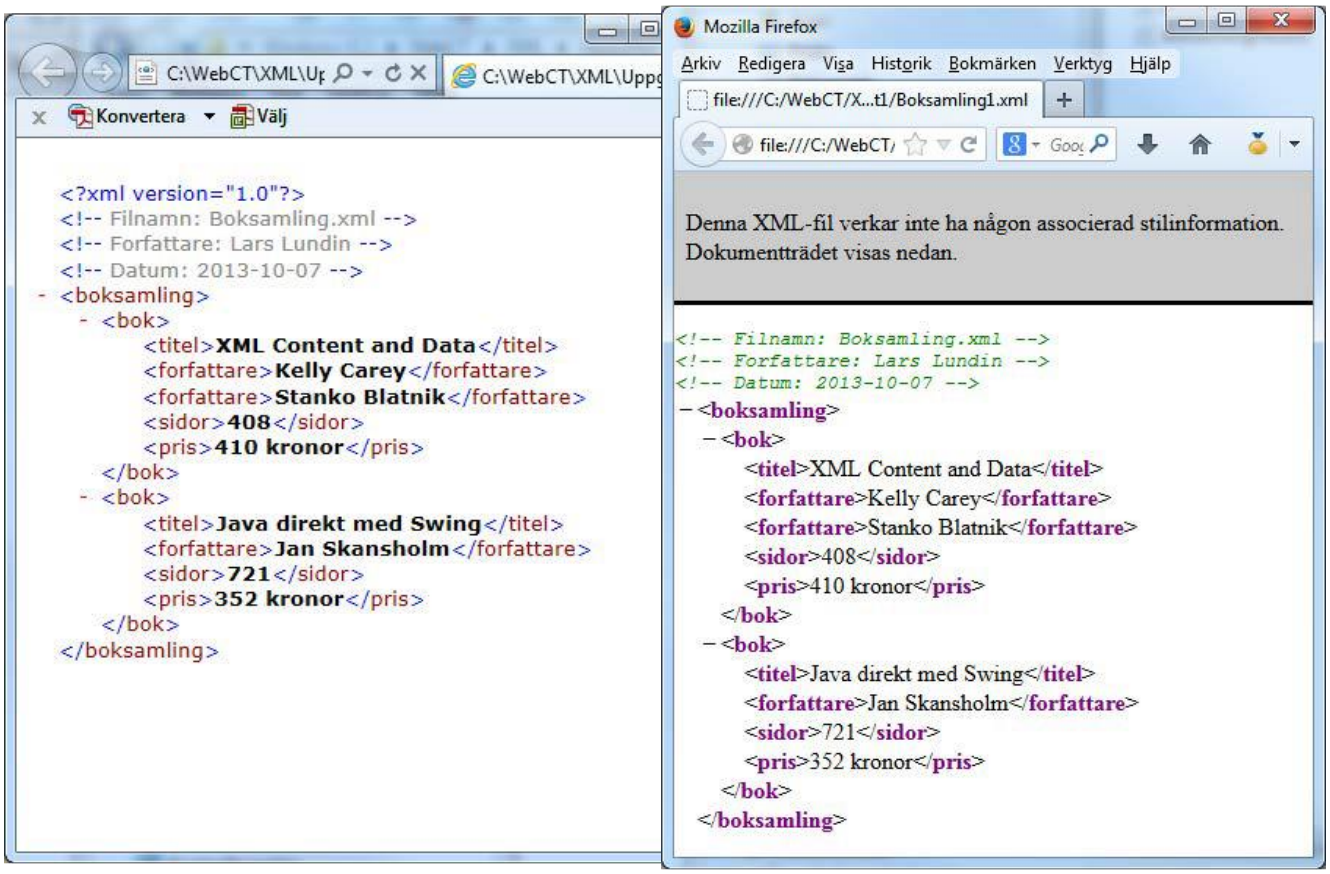
```
<?xml version="1.0"?>
<!-- Filnamn: Boksamling.xml -->
<!-- Författare: Namn Efternamn -->
<!-- Datum: 2018-03-03 -->

<boksamling>
   <bok>
      <titel>XML Content and Data</titel>
      <författare>Kelly Carey</författare>
      <författare>Stanko Blatnik</författare>
      <sidor>408</sidor>
      <pris>410 kronor</pris>
   </bok>
```

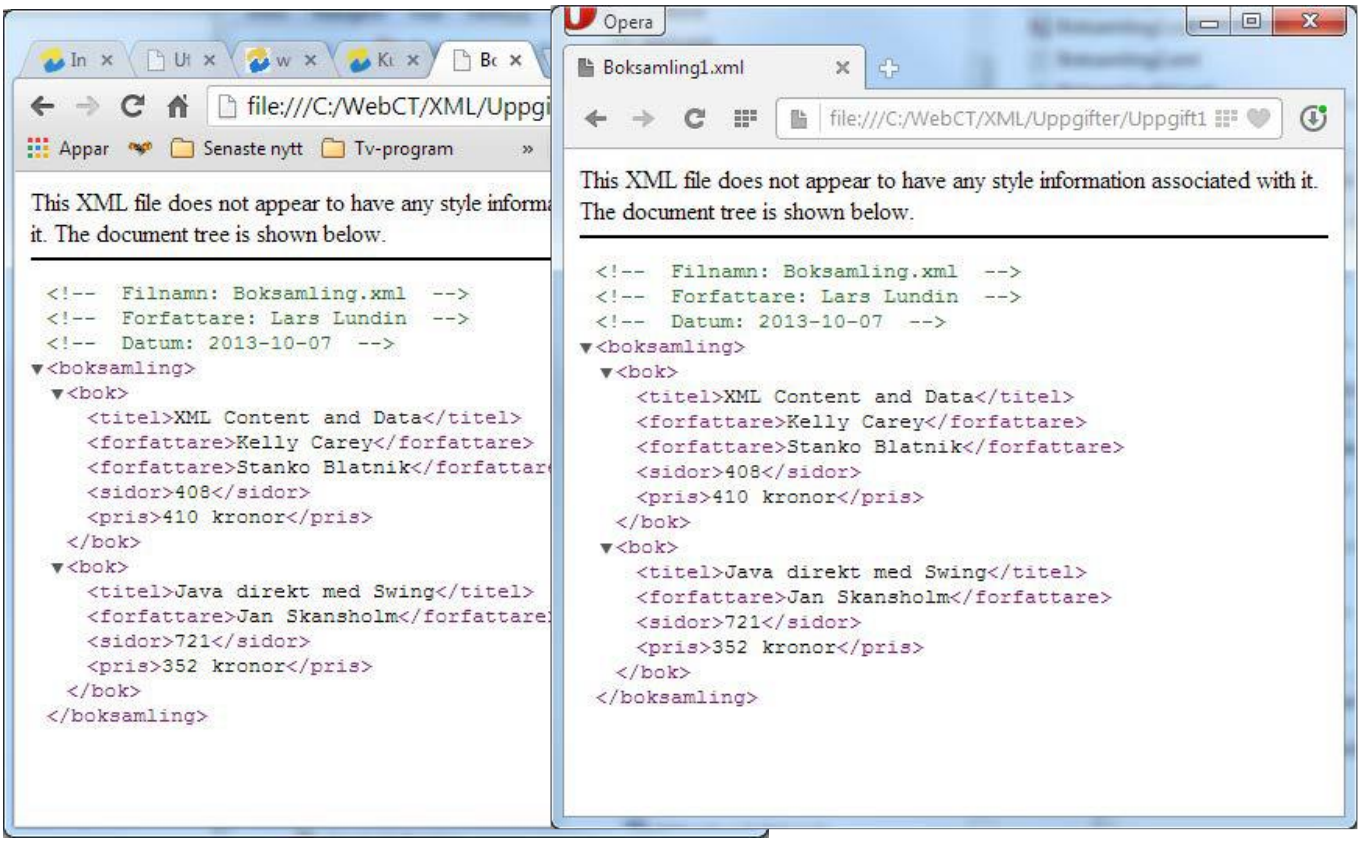
```
    <bok>
        <titel>Java direkt med Swing</titel>
        <författare>Jan Skansholm</författare>
        <sidor>721</sidor>
        <pris>352 kronor</pris>
    </bok>
</boksamling>
```

Spara dokumentet som Boksamling.xml. Nu kan du öppna XML-dokumentet i din Webbläsare valfri editor som innehåller möjlighet att visa xml-träd. Om du inte har gjort några skrivfel och om du har följt alla regler för ett välutformat XML-dokument (kapitel 1.3) kommer innehållet att visas i webbläsaren.

Observera att svenska tecken som å, ä och ö är tillåtna i elementnamn, men att webbläsare kan ha problem med sådana namn, speciellt om felaktig teckenkodning används. Ändra i så fall i Boksamling.xml så att författare skrivs som forfattare.
Eftersom vi ännu inte har knutit någon stilmall till XML-dokumentet kommer olika webbläsare att visa dokumentet på lite olika sätt. Nedan till vänster ser du hur Boksamling.xml visas i de vanligaste webbläsarna. Här visas det hur filen ser ut i Internet

Explorer 10, Mozilla Firefox 24, Google Chrome 30 och Opera 16.





Du väljer givetvis själv vilken webbläsare du vill använda, men du rekommenderas att använda den senaste versionen av webbläsaren. Det är viktigt att om du åtminstone använder version 8 av Internet Explorer. Det finns fler webbläsare utöver de fyra som nämnts ovan. Använder du någon annan är det viktigt att du kontrollerar att den har stöd för XML och XSLT.
Var uppmärksam på att de olika webbläsarna visar XML-dokument på olika sätt. Detta ser vi exempel på i bilderna ovan, men det kan även gälla när vi till exempel använder en stilmall som CSS.

1.2.2 Ett XML-dokuments uppbyggnad

Ett XML-dokument består av en *prolog* och ett *rootelement* (och eventuellt andra saker efteråt). Vi kommer nu att titta närmare på dessa delar.

Prolog

Prologen i exemplet på föregående sidor består av:
- **XML-deklarationen** (<?xml version="1.0"?>). Denna deklaration måste alltid finnas först i XML-dokumentet. Deklarationen visar först och främst att detta är ett XMLdokument, men berättar även vilken version av XML som används samt eventuellt vilken teckenkodning som används och en referens till schemat.
- **Kommentarer** (<!-- Filnamn: Boksamling.xml -->)
- **Blanka rader**

Alla delar i listan ovan kan uteslutas helt, men det rekommenderas starkt att använda dem. XML-deklarationen är nödvändig om vi vill använda oss av svenska tecken (kommer mer om detta i kapitel 1.6).
Alla rader som börjar med <!-- och avslutas med -->, ska uppfattas som kommentarer. Kommentarerna ignoreras helt av XML-tolken2 och har som uppgift att hjälpa programmeraren att läsa och förstå innehållet i XML-dokumentet. Kommentarer om vem som har skapat filen, när den skapades, filnamnet och liknande bör alltid tas med i en prolog. Blanka rader används för att skilja mellan de olika delarna i prologen och gör att dokumentet lättare för oss att läsa. På samma sätt som med kommentarer ignoreras de av XML-tolken.
Prologen kan även innehålla följande delar:
- **DOCTYPE-deklaration**. Dokumenttypsdeklarationen används för att referera till ett schema (till exempel en DTD). Ett exempel kan se ut så här:

```
<!DOCTYPE boksamling SYSTEM "boksamling.dtd">
```

Mer om detta i moment 4
- **Processinstruktioner**. Används för att ge mer information till XML-tolken, till exempel information om var radbrytningar ska infogas och vilken stilmall som ska användas. Vi kommer att titta närmare på detta i nästa lektion.

Rootelelement

Rootelementet kallas även för dokumentelementet. Alla XML-dokument består av **ett**, och endast ett, rootelement. Alla andra element kommer att kapslas in i detta element.
I exemplet på tidigare sidor är det elementet boksamling som är rootelementet. Innanför detta element nästlas sen alla andra element. Det är endast tillåtet med ett rootelement i ett XMLdokument.
Det är med andra ord inte möjligt att i tidigare exempel använda sig av två element med namnet boksamling.

1.3 Regler för ett välutformat XML-dokument

Det finns ett stort antal regler som måste följas när ett XML-dokument skapas om dokumentet ska anses vara välutformat. Här följer en lista på dessa regler:
1. Namn på element och attribut måste börja med en bokstav eller underscore (_). Därefter kan namnet bestå av bokstäver, tal, bindestreck, punkt, kolon eller fler underscore. Längden på ett namn är obegränsat.
2. Tomrumstecken är inte tillåtna i namn.
3. Kolon kan bara användas i namn när en namnrymd har specificerats. Detta beskrivs närmare i nästa lektion.
4. Inget egendefinierat elementnamn kan börja med xml eller XML. Dessa ord är reserverade.
5. Alla XML-dokument får ha ett, och endast ett, rootelement.
6. XML skiljer på versaler och gemener. NAMN ä inte samma som Namn eller namn.
7. Alla element som innehåller data (som inte är en tomelementstagg) måste ha en start- och sluttagg.
8. Tomelementstaggar måste avslutas med en slash (/) enligt följande sätt:

```
<bild src="xml.gif" />
```

9. Alla taggar måste nästlas korrekt:

```
<adress><gata>...</gata></adress>
```

Ett element kan inte avslutas innan dess att alla barnelement har avslutats. Följande exempel är inte korrekt nästlat:

<adress><gata>...</adress></gata>
Om reglerna ovan följs säger man att vi har ett **välutformat** XML-dokument. Alla XML-dokument måste vara välutformade. Finns ett fel i XML-dokumentet kommer till exempel en webbläsare att visa inte kunna visa filen på ett korrekt sätt.

Utöver välutformade dokument finns även så kallade **giltiga** XML-dokument. Detta är XMLdokument som knutits samman med ett schema som DTD eller XML Schema. Ett schema bestämmer bland annat vilka element och attribut ett XML-dokument kan bestå av och hur elementen kan nästlas. Tilläggas kan att alla element och attribut för ett HTML-dokument definieras i en DTD. Det finns inga krav på att ett XML-dokument måste vara giltigt, men det är ett val programmeraren har. XML schema och DTD tas upp i lektion 5
och 6.

1.4 Attribut

Attribut kan användas för att ytterligare beskriva information om ett element. Attributet består
av ett namn och ett värde och måste stå i ett elements starttagg:
<bok inbindning="mjuk pärm">...</bok>
Attributets värde måste omges av " eller '. Det finns stor oenighet om hur och när attribut ska användas. Några menar på att det är bättre att lägga all data i element eftersom det ger större flexibilitet. Andra menar på att attribut ibland är att föredra då antalet element i ett XML-dokument kan minska drastiskt.
Elementet bok i tidigare exempel hade vi kunna skriva genom att enbart använda attribut:
<bok titel="XML Content and Data" författare="Kelly Carey" sidor="408"
pris="410 kronor" />
Observera att det inte är möjligt att ha flera attribut med samma namn i ett element. Av denna anledning är endast en författare med i exemplet ovan, men att det ursprungligen fanns med två författare-element. Detta är en nackdel med attribut. Nästan allt som kan skrivas som attribut kan i stället vara element, men tvärtom gäller inte alltid.

1.5 PCDATA och CDATA

Alla XML-dokument innehåller teckendata, det vill säga den text eller data som är uppmärkt av element och attribut. Teckendata delas in i två kategorier:
- **PCDATA** (Parsed Character Data). Teckendata som måste tolkas.
- **CDATA** (Character Data). Teckendata som inte tolkas.
Ett elements innehåll består av PCDATA, medan attribut består av CDATA.
Eftersom PCDATA måste tolkas finns det en del tecken som inte är tillåtna då tecknen har en viss betydelse för tolken. Om till exempel tecknet < används i ett elements innehåll tror tolken att detta indikerar början på en ny tagg. Detta tecken används som bekant i samband med uppmärkning (start- och sluttaggar) och kan därför inte vara en del i innehållet till ett element.
Andra tecken som används i samband med uppmärkning är >, ”, ’ och &.

1.5.1 Entiteter

Lösningen på problemet ovan kan vara att använda ett teckennummer eller en fördefinierad entitet i stället för själva tecknet. De fem tecknen som nämndes ovan finns alla som fördefinierade entiteter.

Tecken	*Fördefinierat namn*	*Teckennummer*
<	lt	60
>	gt	62
"	quot	34
'	apos	39
&	amp	38

För att använda dessa tecken måste de skrivas på ett speciellt sätt. Används de fördefinierade
entiteterna måste det stå & före namnet och ; efter namnet. Tänk dig att vi vill ha följande text i ett element med namnet uttryck:
a < b & c > d
För att detta innehåll ska tolkas korrekt måste de ogiltiga tecknen bytas ut mot de fördefinierade namnen från tabellen ovan. I XML-dokumentet kommer det att se ut så här:
<uttryck>a **<** b **&** c **>** d</uttryck>

Alternativt kan en teckenreferens användas där teckennumren från tabellen ovan användas. Vi börjar då med &# före själva teckennumret och avslutar med ett ;:
<uttryck>a **<** b **&** c **>** d</uttryck>
Observera skillnaden på användning av de fördefinierade entiteterna och teckennummer. Teckennummer använder ett extra #. Teckennumren ovan uttrycks decimalt, men kan även uttryckas som hexadecimalt. I stället för &# används då &#x.
Om du kommer att ha ett innehåll som liknar uppmärkning, som nedan där <inledning> ingår i innehållet för elementet test,
<test><inledning>text</test>
Måste du med entiteterna ovan skriva enligt
<test><inledning>text</test>

Vi kan även skapa egna namn (entitetsreferenser) för andra speciella tecken. I Sverige är det vanligt att å, ä och ö förekommer som innehåll i XML-dokument. För att använda dessa bör vi dock skapa egna entiteter eller använda oss av en speciell teckenkodning i XML-deklarationen.
Entiteter kommer vi titta närmare på i nästa lektion. Specificering av teckenkodning tas upp i kapitel 1.6.

1.5.2 CDATA-sektioner

Lösningen ovan fungerar utmärkt om innehållet inte består av allt för många ogiltiga tecken. Tänk dig att vi som exempel ska lagra kod skrivet i JavaScript (eller liknande) som innehåll till ett element. I sådana tillfällen kommer antalet ogiltiga tecken att vara väldigt många. Låt säga att vi vill lagra nedanstående kod som innehåll till ett element med namnet jscript:

```
text="Hej alla"
if (a < b && c == d) {
  document.write("<br/>" + text + "<br/>")
}
```

Innehållet består av många ogiltiga tecken som inte tillåts som PCDATA. I stället för att använda teckennummer eller fördefinierade entiteter kan vi skapa en egen CDATA-sektion. Teckendata som förekommer i sådana sektioner kommer inte att tolkas, något som gör att alla tecken användas. CDATA-sektioner skrivs enligt följande form:

```
<![CDATA[ … ]]>
```

JavaScript-koden ovan kan läggas i en CDATA-sektion enligt:

```
<jscript><![CDATA[

text="Hej alla"
if (a < b && c == d) {
    document.write("<br/>" + text + "<br/>")
}

]]></jscript>
```

Lägg märke till att CDATA-sektionen läggs direkt efter starttagen. Nu kommer allt innehåll i elementet att uppfattas som CDATA och inte som PCDATA.

1.6 Teckenkodning

I föregående kapitel nämndes en del tecken som var ogiltiga att använda som innehåll till element. Många länder har tecken som är speciella för just sitt språk. Därför har det upprättats olika teckenkodningar för att täcka de olika ländernas tecken och bokstäver. Kravet för en XML-tolk är att minst stödja Universal Characterset Transformastion Format (UTF); UTF-8 och UTF-16 Unicode. Här i Sverige användes ISO Latin1 (ISO-8859-1), som bland annat täcker våra särsvenska tecken. Numera kan vi använda UTF-8 som kan hantera betydligt fler tecken. Eftersom UTF-8 kodar på ett visst sätt (första byten för ett tecken utanför ASCII har C0-F7 (hex), övriga har 80-BF), kan en modern texteditor eller webbläsare i många fall se i själva filen att den är kodad i UTF-8 och därmed Unicode, och tolka tecknen rätt. Detta är en stor fördel mot nästan alla äldre kodningar, då det var svårt eller omöjligt att räkna ut kodningen om den inte angavs explicit. Också i text som i huvudsak består av ASCII är det osannolikt att enskilda andra tecken av en slump blir giltig UTF-8.
För att ange vilken teckenkodning som ska användas i ett XML-dokument används attributet encoding i XML-deklarationen. För att hantera våra svenska tecken skrev vi:
`<?xml version="1.0" encoding="ISO-8859-1"?>` och numera `<?xml version=”1.0” encoding=”UTF-8”?>`
För att demonstrera vad som händer om detta attribut inte anges kan vi visa följande XML-dokument i en webbläsare:

```
<?xml version="1.0"?>
<text>Örnen svävar högt över älven.</text>
```

Man kan alltså säkerställa detta genom att använda UTF-8 istället. MEN det räcker inte att ange att man använder sig av UTF-8, man måste också SPARA själva dokumentet i UTF-8.

Attributet standalone

XML-deklarationen kan innehålla ett tredje attribut utöver attributen version och encoding, nämligen standalone. Detta attribut kan anta värdena yes eller no. Attributet används för att ange om någon relation finns mellan XML-dokumentet och andra dokument. Om XML-dokument används helt ensamt och inte får någon ytterligare information från till exempel en extern DTD (DTD ligger i en egen fil), säger man att XML-dokumentet är fristående. Vi sätter då värdet på attributet standalone till yes. I exemplet ovan kan vi sätta värdet till yes då ingen extern information ska användas:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<text>Örnen svävar högt över älven.</text>
```

Behöver vi använda oss av tecken-kodning idag? De flesta läsarna klarar ju av att visa rätt tecken ändå. Jo, det behöver vi. För att säkerställa att rätt tecken kommer visas i webb-läsaren överallt!
Svenska webbsidor läses ju även i utlandet där samma tecken-kodning inte används. Här är en länk om tecken-kodning: http://www.w3.org/International/questions/qa-what-is-encoding.