



XML-familjen Vad är Document Object Model (DOM)?





XML-familjen

**Vad är Document
Object Model (DOM)?**

Publikationen kan beställas från:

Statskontoret

Publikationsservice

Box 2280

103 17 Stockholm

Tfn: 08-454 46 43

Tfx: 08-454 46 45

E-post: publikations.service@statskontoret.se

Mer information om Statskontoret finns på Internet:

www.statskontoret.se

Kontaktpersoner Statskontoret:

Gunnar Hansson

gunnar.hansson@statskontoret.se

© STATSKONTORET

ISBN: 91-7220-430-3

Layout: Dreamforce Infomedia, Solna

Tryck: Novum Grafiska AB 2000

Innehållsförteckning

Inledning	5
Bakgrund	7
Historik	9
DOM	11
DOM:s objektmodell	11
Nodtyper och metoder	13
Exempel på nodtyper	13
Exempel på egenskaper	13
Exempel på metoder	14
Åtkomst av noder.....	14
Praktiskt fall med användning av DOM, script och ASP	17
Framtid	23
Statskontorets publikationer för 1999 och 2000	25

Inledning

Denna rapport riktar sig i första hand till IT-verksamma inom svensk offentlig sektor. Den ingår i rapportserien XML-familjen och syftar till att belysa det faktum att XML är en av ett antal tekniska specifikationer från World Wide Web Consortium (W3C) som tillsammans skapar förutsättningar för betydligt förbättrad hantering av information. Rapporterna vill ge kortfattade lättlästa beskrivningar av de nya teknikerna.

Denna rapport behandlar Document Object Model (DOM).

Rapporten är framtagen av Statskontoret inom ramen för arbetsområdet ärende- och dokumenthantering, projektledare Jan Lundh i samarbete med Daniel Björkman, Tero Ahola och Christoffer Andreasson på Improve.

I serien ingår:

Vad är XML?	Statskontoret 2000:30
Vad är XML Familjen?	Statskontoret 2000:31
Vad är XML Schema och XML DTD:er?	Statskontoret 2000:32
Vad är XML Formatmallar?	Statskontoret 2000:33
Vad är XML Länkar?	Statskontoret 2000:34
Vad är Document Object Model (DOM)?	Statskontoret 2000:35
Vad är XML Query Language (XQL)?	Statskontoret 2000:36

För aktuell information om de olika specifikationernas status se www.w3c.org.
För svenska aktiviteter kring XML se www.xmlakademin.nu.

Stockholm december 2000

Jan Lundh

Bakgrund

När man skall manipulera och hantera data i olika applikationer finns det ofta standardiserade sätt att göra detta. Man brukar skapa färdiga metoder för att hantera exempelvis grafik eller textmassor. Dessa metoder fungerar som programmeringsgränssnitt, API:er (Application Programming Interface), mot den aktuella informationen.

För att informationen vi lagrar med hjälp av XML skall komma till nytta, måste de applikationer vi använder kunna få tillgång till XML-dokumentet.

I syfte att göra detta möjligt har ett API mot XML-dokument utvecklats, Document Object Model (DOM).

DOM är framtaget av *World Wide Web Consortium* (W3C) för att tillhandahålla ett standardiserat, plattform- och programberoende gränssnitt. Med hjälp av detta programmeringsgränssnitt kan man använda olika script och programmeringsspråk för att dynamiskt manipulera HTML och XML-dokument.

Ett exempel på användning av DOM kan vara när man samlar in data från ett webbformulär och vill spara det i XML. De uppgifter som användaren matar in fångas upp med hjälp av ett scriptspråk, som javascript eller VB script. Dessa scriptspråk använder DOM:s programmeringsgränssnitt för att skapa ett XML dokument av informationen och sedan spara ner XML-dokumentet.

Denna skrift riktar sig till de inom offentlig sektor som hanterar och skapar informationslösningar. Skriften syftar till att ge en inblick i hur ett programmeringsgränssnitt för XML är uppbyggt och fungerar.

Historik

Tanken att använda ett programmeringsgränssnitt som är språkoberoende är inte ny. Ett exempel på ett känt API är *Microsoft Messaging Application Programming Interface* (MAPI) som används för att hantera mailfunktioner i olika applikationer i Windowsmiljö. Detta API är språkoberoende, det vill säga att metoderna som är definierade ser likadana ut vilket programspråk man än använder. DOM räknas som ett API, men har inte alltid varit det.

DOM är ingen nyhet för människor som sysslat med webbutveckling. Under rätt lång tid har DOM varit ett allmänt begrepp som betytt olika saker för olika människor. Begreppet uppstod i samband med att Netscape introducerade javascript i sin webbläsare. Denna tidiga version av javascript hade metoder för att komma åt exempelvis fönster och dokumentinstanser. Det var i dessa metoder som dokumentobjektmodellen började ta form. I detta stadium var DOM inte någon form av rekommendation, utan detta arbete tog fart först senare.

De två stora aktörerna, Microsoft och Netscape, utökade DOM på var sitt håll med nya metoder. Med varje version av webbläsarna dök det upp ytterligare funktioner. Detta ledde till att sättet man manipulerade ett dokument var osammanhängande och beroende av vilket verktyg man använde.

W3C påbörjade arbetet med en specifikation som försökte sammanställa dokumentobjektmodellen. Med denna specifikation samlade man upp de löst och allmänt hållna modellerna som fanns under ett tak. Många av de funktioner som fanns och fortfarande finns hamnade ej under detta tak, utan kallas numera utökningar av DOM. I oktober 1998 blev DOM Level 1 en rekommendation utfärdad av W3C.

Den 13 november 2000 antogs level 2 som rekommendation. DOM Level 2 är uppdelad på fem delar:

- Core Specification
- Style Specification
- Events Specification
- Views Specification
- Traversal and Range

Dessutom lades fram men underkändes efter kommentarer en specifikation för DOM Level 2 HTML. Den fick istället status som Last Call Working Draft. För beskrivning av W3Cs steg vid utveckling av specifikationer se rapporten, 2000:31 Vad är XML-familjen?

Arbete med DOM Level 3 har påbörjats.

DOM

DOM är tänkt att tillhandahålla ett standardiserat, plattform- och program-språksoberoende gränssnitt. Med hjälp av detta programmeringsgränssnitt kan man använda olika script och programmeringsspråk för att dynamiskt manipulera HTML och XML-dokument.

DOM utför inget själv utan är en metod för att generellt kunna använda olika programmeringsspråk för att manipulera XML-dokument på något sätt. Dess uppgift är till stor del att kunna kalla på elementen och attributen inuti ett XML-dokument.

DOM Level 1 innehåller två delar. Första delen innehåller gränssnittet som kan användas till olika typer av strukturerade dokument. Den andra delen innehåller utökade gränssnitt som behövs för dokument av formatet XML.

DOM Level 2 innehåller en objektmodell för style sheets och definierar funktioner för att manipulera layout-information som är kopplad till dokumentet. Den gör det också möjligt att klättra i dokumentstrukturen och definierar en händelsemodell. Nivå 2 ger också stöd för XML namespaces.

DOM Level 3. kommer ta upp laddning och lagring av dokument, DTD:er och Scheman samt stöd för validering av dokument.. Dessutom kommer vyer, formatering och ytterligare händelsefunktionalitet införas. Working drafts finns redan tillgängliga.

DOM:s objektmodell

DOM tillhandahåller en objektmodell för att dynamiskt kunna lägga till, ta bort, ändra element, attribut och textinnehåll i ett dokument. Denna modell har sitt ursprung i en objektorienterad miljö. I fallet med XML-dokument är det mestadels de olika elementen som ses som objekt.

Ser man på ett XML-dokument så har det en viss struktur. Det finns element som i sin tur innehåller andra element och på så vis utgör en trädstruktur. I trädstrukturen finns det olika relationer mellan element. Dessa relationer beskrivs som en familj. Ett element kan exempelvis ha barn eller syskon, eller så kan ett element vara förfader till andra element.

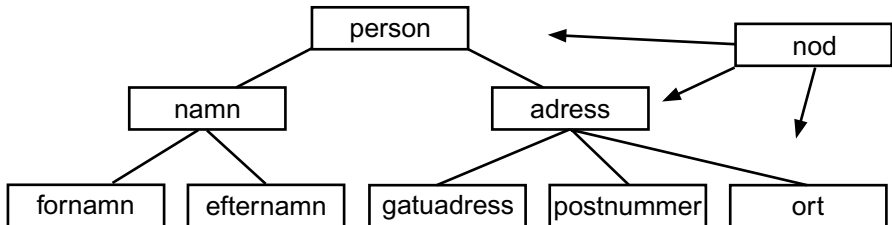
I exemplet nedan framgår det att elementet person innehåller andra element. Detta ser man tydligt när man tittar på start- och slutmärket för 'person'. Person är således förälder till elementet 'namn' och förfader till elementen 'for-namn' och 'efternamn'. Dessa är i sin tur syskon.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<person>
  <namn>
    <fornamn>Bo</fornamn>
    <efternamn>Ek</efternamn>
  </namn>
  <adress>
    <gatuadress>Gågatan 3</gatuadress>
    <postnummer>217 11</postnummer>
    <ort>Malmö</ort>
  </adress>
</person>

```

XML-dokumentet kan ritas upp som en informationsmodell där man placerar rotelementet, det vill säga 'person', överst i strukturen. Ett DOM objekt brukar beskrivas i formen av ett träd. Nedan presenteras hur en trädstruktur ser ut:



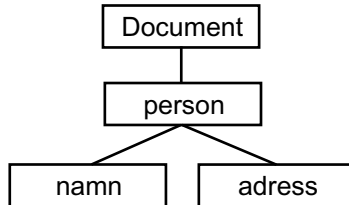
När man läser in ett XML-dokument i minnet, byggs en trädstruktur liknande den som presenteras ovan. De metoder som DOM har kan sedan användas för att manipulera trädet. Skillnaden mellan strukturen i informationsmodellen och i Document Object Model är själva benämningen av element, attribut och innehåll. I DOM betraktas alla delar av ett XML-dokument som noder. En nod finns på en viss nivå i trädet. Denna kan sedan förgrena sig till flera noder en nivå längre ner i trädet. I exemplet ovan finns det ett toppnivåelement även kallat för rotelementet som grenar ut sig i andra noder, som i sin tur förgrenar sig.

En nod är det sätt på vilket ett element i ett XML-dokument representeras i DOM. Sålunda blir elementet 'person' till noden 'person' när man använder DOM.

Ser vi på noden 'person', som i exemplet ovan är rotelement, så har den två barnnoder knutna till sig. Det är noden 'namn' och noden 'adress'. Noderna är syskon till varandra eftersom de har samma förälder. Dessa noder är i sin tur också föräldrar. Detta visar att en nod kan vara både barn och förälder beroende på vilken nivå i trädet man befinner sig.

Noderna längst ner i hierarkin har samma förfader (person). I syntaxen för DOM används engelska beteckningar för relationer mellan olika noder. Föräldranod kallas för parentnode, syskon för siblings, barn för children samt förfader för ancestor.

Det finns en nod i DOM som är viktig att känna till och det är 'document'. Det är denna nod som möjliggör att man exempelvis kan skapa nya element till strukturen. Läger vi till den noden i trädstrukturen ovan ser trädet ut så här:



Nodtyper och metoder

Noderna i ett träd har olika metoder och egenskaper som används när man skall manipulera data. Metoder är ett sätt att manipulera dokumentet och egenskaper är de olika värden som noderna kan anta eller hämta. Noderna kan vara av olika typer.

Exempel på nodtyper

Alla delarna i ett XML-dokument kan representeras som noder i DOM. De vanligaste delarna i ett XML-dokument är olika element, deras attribut och det textinnehåll som finns i ett element.

Exempel på egenskaper

En nod har ett antal egenskaper. Dessa beskriver dels noden och dels nodens förhållanden till resten av trädet:

nodeName – Hämtar namnet på noden, exempelvis <namn/>.

nodeValue – Hämtar värdet på noden, exempelvis innehåll i form av text.

parentNode – Hämtar föräldern till den aktuella noden.

childNodes – Nodlista innehållande alla barnen till noden. Exempelvis kan "for-namn" och "efternamn" ses som en nodlista för noden "namn" i vårt exempel.

firstChild – första barnet till den aktuella noden, exempelvis är noden "namn" första barnet till noden "person".

Exempel på metoder

När man skall manipulera data i ett XML-dokument används metoder. En metod förändrar ofta trädet på något sätt.

`InsertBefore (personnummer, namn)` – Noden personnummer sätts på platsen före noden namn.

`replaceChild (personnamn, namn)` – Noden personnamn ersätter noden namn.

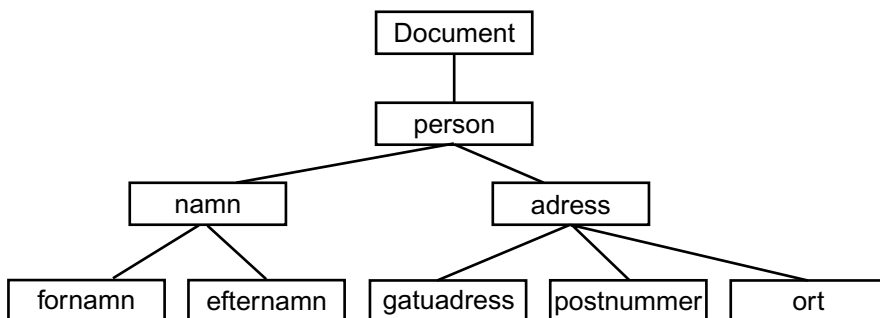
`removeChild (namn)` – Noden namn tas bort från trädet.

`createElement ("Anst.nr")` – Ett nytt element, Anst.nr skapas.

`AppendChild ("Anst.nr");` – Läger till ett nytt barn (Anst.nr) som sista barnet till den nuvarande noden.

Åtkomst av noder

Genom punktnotation skapas en sökväg till en specifik nod i DOM-trädet. Punktnotationen fungerar på liknande sätt som "backslash" fungerar när man orienterar sig i en dators filstruktur.



Om vi återigen tittar på exemplet med DOM-trädet så skulle man kunna orientera på följande sätt till elementet gatuadress:

```
var gAdress =  
MyXMLdoc.documentElement.childNodes.item(1).childNodes.item(0)
```

MyXMLdoc är namnet på Document. För att komma åt själva rotelementet så används `documentElement`. I vårt fall är rotelementet detsamma som elementet 'person'. Nästa del av uttrycket är `childNodes.item(1)` som navigerar till elementet 'adress'. När antalet barn räknas börjar man inte på ett, utan på noll. Adress blir ett eftersom man räknar barnen från vänster till höger. På samma sätt fungerar det för nästa del, `childNodes.item(0)`, som blir elementet 'gatuadress'.

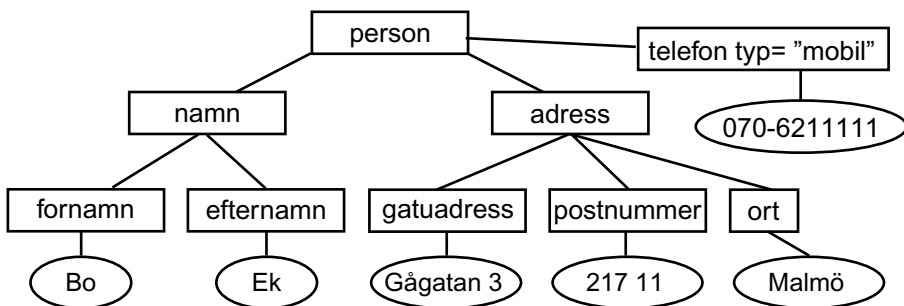
På detta sätt har vi nu orienterat oss ner i trädstrukturen till elementet gatuadress. Denna nod returneras och hamnar i variabeln gAdress. Ifrån denna variabel kan man sedan exempelvis manipulera innehållet.

```
gAdress.nodeValue = 'Storgatan';
```

Om vi ser på ett liknande exempel som innan, men med skillnaden att dokumentet även innehåller attribut, så kan det se ut så här:

```
<person>
  <namn>
    <fornamn>Bo</fornamn>
    <efternamn>Ek</efternamn>
  </namn>
  <adress>
    <gatuadress>Gågatan 3</gatuadress>
    <postnummer>217 11</postnummer>
    <ort>Malmö</ort>
  </adress>
  <telefon typ="mobil">070-621 11 11</telefon>
</person>
```

I exemplet vi tidigare använt har nu ett tillägg gjorts i form av att elementet 'telefon' tillkommit i strukturen med attributet 'typ'. Trädstrukturen ser likadan ut som tidigare med det undantaget att textnoder har lagts till. Detta symboliseras av cirklar i exemplet.



Vi börjar med att tilldela variabeln 'person' noden 'person'.

```
var person = myXMLdoc.documentElement;
```

Därefter hämtar vi noden för attributet 'typ'.

```
var telefontyp = person.lastChild.attributes.getNamedItem("typ");
```

Vi vet att elementet 'telefon' är det sista barnet till 'person' och använder därför egenskapen 'lastChild' för att hämta elementet. Därefter ber vi att få de attribut som tillhör 'telefon' genom att använda 'attributes'. Från det resultatet frågar vi efter det attribut som har namnet 'typ'. Denna operation ger oss noden för attributet 'typ' i variabeln telefontyp.

Man kan även manipulera trädet genom att ta bort och lägga till noder. Det är detta som ger en möjlighet att fylla på och reducera ett XML-dokument i realtid. Vi skapar och lägger till ett element. Därefter plockar vi bort det igen.

```
var personnummer = myXMLdoc.createElement("personnummer")
```

Man måste använda documentobjektet när man skall skapa en ny nod till ett träd. Detta görs med metoden 'createElement'. I parentesens anges vad den nya noden skall heta.

```
personnummer.nodeValue = '660101-4952';
```

Här tilldelas vår nyskapade nod ett textvärde.

```
var person = myXMLdoc.documentElement;  
person.insertBefore(personnummer, person.firstChild);
```

När en ny nod sätts in i trädet, görs det som barn till en annan nod. I vårt fall vill vi lägga in 'personnummer' före elementet 'namn'. Därför letar vi först upp föräldernoden som är 'person'. Argumenten säger att personnummer skall läggas in före noden 'persons' första barn, vilket är 'namn'.

```
person.removeChild(personnummer);
```

I likhet med när man lägger till en nod, så letar man även upp föräldern när man vill ta bort en nod. Argumentet avser i detta fallet det barn som man ej vill ha kvar.

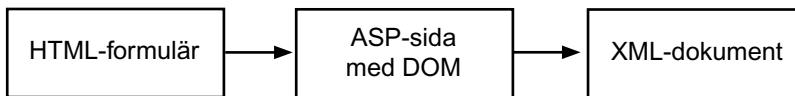
Praktiskt fall med användning av DOM, script och ASP.

DOM används bland annat när man vill bearbeta XML-dokument dynamiskt. Det kan vara på klienten eller på servern. När man använder sig av DOM på servern förutsätter detta att man använder sig av ASP-filer, eller någon annan liknande teknik.

ASP står för Active Server Pages och är en teknik för att utveckla serverbaserade webblösningar. När en fil är sparad som .asp vet servern om att filen skall bearbetas på servern. Vill man själva laborera med ASP-filer räcker det att man skaffar sig en personal-webbserver till sin dator.

Följande exempel visar hur man kan använda olika tekniker tillsammans för att göra dynamiska bearbetningar. Detta är ett praktiskt exempel som går att använda.

ASP-filen som finns med i slutet av dokumentet används till följande: När asp-filen aktiveras hämtar den inmatade värden ur ett htmlformulär. Dessa värden ligger till grunden för en XML-fil som skapas dynamiskt med DOM. Filen som skapas följer den struktur som använts i tidigare exempel.



Formuläret ser ut som följande. Man kan mata in namn och adressuppgifter. När man trycker på knappen 'skicka' aktiveras ASP-filen. Html-koden från formuläret kommer inte att gås igenom, utan kommer att presenteras i slutet av detta exempel.

Förnamn: Efternamn:

Gatuadress:

Postadress:

Allt innehåll i en asp-fil som man vill skall aktiveras från servern skriver man innanför tecknen <% och %>. Scriptspråket som används inuti asp-filen är javascript.

```
<%@ Language=javascript %>
<%//-----subrutin -----%>
```

Det första funktion som beskrivs skapar ett DOM-objekt som sedan skickas till funktionen 'nyperson'. Sist så sparas DOM-objektet ner till en XML-fil.

```
<%
function MakeXmlFile(){

    var aDOMObject = Server.CreateObject("Microsoft.XMLDOM");

    var pi = aDOMObject.createProcessingInstruction("xml", "version=\"1.0\"
encoding=\"ISO-8859-1\"");
    aDOMObject.appendChild(pi);
    nyperson(aDOMObject);
    aDOMObject.save(Server.MapPath("personuppgifter.xml"))
}
%>
```

Här påbörjas funktionen MakeXmlFile och i den skapar man sitt DOM-object. Detta görs med metoden 'CreateObject'. Denna metod ger åtkomst till externa objekt. I detta fall är det objektet MicrosoftXMLDOM som skapas och som sedan sparas i en variabel.

Det skapas också en bearbetningsinstruktion som placeras överst i XML-dokumentet. (<?xml version="1.0 encoding="ISO-8859-1"?> Denna bearbetningsinstruktion är nödvändig för att exempelvis Internet Explorer 5.0 skall kunna tolka dokumentet som ett XML-dokument. ISO-8859-1 är en teckenuppsättning som gör att man kan använda svenska tecken i XML-dokument.

Därefter ropar vi på funktionen 'nyperson' och skickar med det nyskapade DOM-objektet som argument.

```
<%
function nyperson(DOMObject)
{
    //Bygg trädet:
    var aPerson = DOMObject.createElement("PERSON");
    //skapa element till trädet:
    var aNamn = DOMObject.createElement("NAMN");
    var aFornamn = DOMObject.createElement("FORNAMN");
    var aEfternamn = DOMObject.createElement("EFTERNAMN");
```

```
var aAdress = DOMObject.createElement("ADRESS");
var aGatuadress = DOMObject.createElement("GATUADRESS");
var aPostnummer = DOMObject.createElement("POSTNUMMER");
var aOrt = DOMObject.createElement("ORT");
```

I första delen av funktionen 'nyperson' skapas olika element som sparas i variabler. Dessa kommer sedan att användas när XML-filen skall byggas ihop. Exempelvis så skapas det ett element med namnet 'ORT' som sparas i variabeln aOrt. Det är dessa objekt som blir märkord i vårt färdiga XML-dokument.

```
aFornamn.text = Request.Form.Item("Firstname");
aEfternamn.text = Request.Form.Item("Surname");
aGatuadress.text = Request.Form.Item("Street");
aPostnummer.text = Request.Form.Item("Code");
aOrt.text = Request.Form.Item("Town");
```

Denna del av scriptet känner av om man skrivit in värden i html-formuläret. Om en användare matat in värden i formuläret skall dessa hämtas genom uttrycket Request.Form.Item. Via detta kommando begär vi att få värdet från olika fält i vårt html-formulär. Värdet som återfinns inom citationstecken i parentes är namnet på ett fält i formuläret. Vi har tidigare skapat noden aFornamn och nu vill vi lagra värdet från fältet 'Firstname' här. I det här fallet följs aFornamn av 'text' som är samma sak som nodeValue.

```
//lägger in nya barnet som sista barnet till noden
aNamn.appendChild(aEfternamn);
aNamn.appendChild(aFornamn);
aAdress.appendChild(aGatuadress);
aAdress.appendChild(aPostnummer);
aAdress.appendChild(aOrt);
aPerson.appendChild(aNamn);
aPerson.appendChild(aAdress);
DOMObject.appendChild(aPerson);
}
%>
```

I denna del binds noderna ihop så att en korrekt trädstruktur skapas. Vi lägger helt enkelt till barn till respektive förälder. Exempelvis så skall 'förnamn' och 'efternamn' vara barn till 'namn'. Metoden 'appendChild' lägger till barnet sist, det vill säga längst till höger under respektive förälder. DOMObject, är den nod vi i tidigare exempel beskrivit som 'document'. Vad som sker på denna rad är att värdet ur variabeln aPerson sparas som den översta noden i trädet. Detta avslutar funktionen 'nyperson'.

<HTML>

ASP-filen 'nyxml.asp'som anropas av formuläret i html-sidan.

```
<%@ Language=javascript %>
<%//-----subrutin -----%>

<%
function MakeXmlFile(){

    var aDOMObject = Server.CreateObject("Microsoft.XMLDOM");
    var pi = aDOMObject.createProcessingInstruction("xml", "version='1.0'
encoding='ISO-8859-1'");
    aDOMObject.appendChild(pi);
    nyperson(aDOMObject);
    aDOMObject.save(Server.MapPath("testa.xml"))
}
%>

<%
function nyperson(DOMObject)
{
    //Bygg trädet:
    var aPerson = DOMObject.createElement("PERSON");
    //skapa element till trädet:
    var aNamn = DOMObject.createElement("NAMN");
    var aFornamn = DOMObject.createElement("FORNAMN");
    var aEfternamn = DOMObject.createElement("EFTERNAMN");
    var aAdress = DOMObject.createElement("ADRESS");
    var aGatuadress = DOMObject.createElement("GATUADRESS");
    var aPostnummer = DOMObject.createElement("POSTNUMMER");
    var aOrt = DOMObject.createElement("ORT");

    aFornamn.text = Request.Form.Item("Firstname");
    aEfternamn.text = Request.Form.Item("Surname");
    aGatuadress.text = Request.Form.Item("Street");
    aPostnummer.text = Request.Form.Item("Code");
    aOrt.text = Request.Form.Item("Town");

    //lägger in nya barnet som sista barnet till noden
    aNamn.appendChild(aEfternamn);
    aNamn.appendChild(aFornamn);
    aAdress.appendChild(aGatuadress);
    aAdress.appendChild(aPostnummer);
    aAdress.appendChild(aOrt);
    aPerson.appendChild(aNamn);
    aPerson.appendChild(aAdress);
    DOMObject.appendChild(aPerson);
}
%>
```

```
<HTML>
<HEAD>
</HEAD>
<BODY> <!--onLoad = "javascript:MakeXmlFile()"-->
<%
    MakeXmlFile();
%>
</BODY>
</HTML>
```

Framtid

DOM 1.0 blev rekommendation från W3C i oktober 1998. DOM nivå 2 blev rekommendation 13 december 2000. Skillnaden mellan dessa två, är att Nivå 2 är en utökad version av DOM som tillför ytterligare modeller till API:et. Bland dessa finns exempelvis en modell för CSS. Dessa modeller gör att en programmerare får tillgång till dokument, precis som i fallet med XML. Tanken är att man skall kunna komma åt och skapa ytterligare dynamik i sina lösningar.

DOM är under ytterligare utveckling. Denna kan enligt Robin Covers komma att specificera gränssnitt mot det underliggande fönsterhanteringssystemet. De kan också komma att innehålla gränssnitt för frågespråk och ta upp multithreading, synkronisering, säkerhet och repositories.

Statskontorets publikationer för 1999 och 2000

1999

- 99:1 Statskontorets publikationer utgivna under åren 1996-1998. GRATIS
- 99:2 Skiftet till år 2000 – läget i myndigheter. Lägesrapport 7. Slut finns i PDF.
- 99:3 E-post i förvaltningen – En rättslig översikt. 75:-
- 99:4 Samordning och styrning av smittskyddsverksamheten. GRATIS
- 99:5 Informationskampanjen Starta eget tillsammans. GRATIS
- 99:6 Utvecklingsgarantin för ungdomar – det första året. GRATIS
- 99:7 Regelförenkling i EU och Sverige. GRATIS
- 99:8 Användning av otraditionella medel i Skåne åren 1998 och 1999. GRATIS
- 99:9 Myndigheternas förutsättningar för deltagande i ett nytt offentligt rättsinformationssystem. GRATIS
- 99:10 Skiftet till år 2000 – Läget i myndigheter. Slut finns i PDF.
- 99:11 Läget i viktiga samhällsfunktioner. GRATIS
- 99:12 Tillstånd för näringsverksamhet. GRATIS
- 99:13 Intensifierat nordiskt samarbete – analys av det svenska styrsystemet. GRATIS
- 99:14 E-post i skolan. 150:-
- 99:15 Staten i omvandling 1999. 175:-
- 99:15A The Swedish Central Government in Transition 1999. 50:-
- 99:16 XML-verktyg. Ej utk.
- 99:17 Strukturer för hantering av certifikat och kryptonycklar i Sverige – Statskontorets förslag till vidare arbete. GRATIS
- 99:18 Hur mycket kostar Kunskapslyftet? – en analys av kommunernas kostnader och bidrag. GRATIS
- 99:19 Punktskatteorganisationen inom skatteförvaltningen. GRATIS
- 99:20 En samlad administration av lönegarantin. 100:-
- 99:21 Det viktiga valet av verkschef. En jämförande studie av verkschefsutnämningar. GRATIS

- 99:22 Skiftet till år 2000 läget i myndigheter. Lägesrapport 10.
Slut finns i PDF.
- 99:23 Enkla råd och tips vid E-posthantering. 75:-
- 99:24 Gamla län blir nya regioner? Slutrapport. GRATIS
- 99:25 Vad kostar ett tillstånd?
– förslag till enklare tillståndsgivning. GRATIS
- 99:26 SESAM Öppnade museisamlingarna? GRATIS
- 99:27 Läget i vissa samhällsfunktioner. Lägesrapport 11. GRATIS
- 99:28 Miljökrav vid offentlig upphandling
– samhällsekonomiska konsekvenser. GRATIS
- 99:29 Morgondagens kommission – en svensk angelägenhet. 75:-
- 99:29A Tomorrow's Commission – a Swedish Concern. GRATIS
- 99:30 Säkerhet med elektronisk identifiering. 100:-
- 99:31 Tjänster för elektronisk identifiering i offentlig förvaltning. GRATIS
- 99:32 Vem ser efter arbetslöshetskassorna? GRATIS
- 99:33 Förslag till uppföljning av de transportpolitiska målen. GRATIS
- 99:34 Konvergens mellan tele och data – en orientering. 100:-.
- 99:35 Skiftet till år 2000 – Läget i myndigheterna. Lägesrapport 12. GRATIS
- 99:36 Läget i viktiga samhällsfunktioner – Skiftet till år 2000.
Lägesrapport 13. GRATIS
- 99:37 IT-kostnader. Förstudie. 100:-
- 99:38 Organisations-, styr- och verksamhetsformer i kommuner och lands-
ting. GRATIS
- 99:39 Elektronisk upphandling under tröskelvärdena. GRATIS
- 99:40 Informationstjänster i fokus. 85:-
- 99:41 Från underutnyttjande av användarboom Online/webbtjänster,
CD-rom och betalsamtal i Sverige 1997 och 1998. 100 kr
- 99:42 Interaktiva medier – framtidsbransch med intelligens? 100 kr
- 99:43 Investeringar i digital information. GRATIS
- 99:44 Efterfrågan på tjänster via Internet. GRATIS
- 99:45 Förstudie – IP-telefoni. 100:-
- 99:46 Förstudie – Centrex-lösningar. 100:-
- 99:47 Mobiltelefoni – en förstudie. 100:-
- 99:48 UTGÅTT

- 99:49 Läget i myndigheter och vissa samhällsfunktioner – skiftet till år 2000. GRATIS
- 99:50 Framtidens servicefunktioner – en förstudie. 100:-
- 99:51 Att konsultera medborgarna. GRATIS
- 99:52 Hur kan Sveriges möjligheter att påverka EU:s samarbete förbättras? 130:-
- 99:53 Mittutvärderingar av strukturfonderna. GRATIS
- 99:53A Mittutvärderingar av strukturfonderna. Bilagedel. GRATIS

2000

- 00:1 Effektivare kustbevakning. 100:-
- 00:2 Compazisons of Character Sets – Utgåva 2. GRATIS
- 00:3 Öppenhet och insyn i myndigheternas EU-arbete. 75:-
- 00:4 Postrummet – ett sätt att öka säkerheten i e-post. 100:-
- 00:5 Intelligentia tjänster och elektroniska blanketter. 150:-
- 00:6 Sverige i världen – en utvärdering av svenskt deltagande i några internationella mellanstatliga organisationer. 150:-
- 00:7 Infrastruktur för säker elektronisk överföring till, från och inom statsförvaltningen. 100:-
- 00:8 Svenska myndigheter och euron – gemensamma frågor. GRATIS
- 00:9 Styrelser med fullt ansvar. 100:-
- 00:10 Intrångsdetekteringssystem
- 00:11 Statliga myndigheters rutiner för hantering av programvarulicenser. GRATIS
- 00:12 Telefonitrafik – vägledning och råd. GRATIS
- 00:13 SHS – en del av infrastrukturen för samhällets elektroniska tjänster. 150:-
- 00:14 Efter skiftet till år 2000, CD-rom. GRATIS
- 00:15 Staten i omvandling 2000.
- 00:16 Staten som kommersiell aktör. Del I Huvudrapport. 150:-
- 00:16A Staten som kommersiell aktör. Del II Fallstudier. GRATIS
- 00:17 Utvärdering och politik. GRATIS
- 00:18 Uppföljning och utvärdering av det regionala utvecklingsarbetet
En rapport till den parlamentariska regionkommittén (PARK). GRATIS

- 00:19 Resultatredovisning och resultatbedömning i budgetpropositionen år 2000. GRATIS
- 00:20 Fem år i EU – en utvärdering om statsförvaltningens medverkan i EU-samarbetet. Huvudrapport. 150:-.
- 00:20A EU-arbetet i Sverige, Danmark och Finland. Appendix I. GRATIS
- 00:20B Fallstudier av tre EU-intensiva politikområden. Appendix II. GRATIS
- 00:20C Genomförandekommittéer. En expendstudie om svenska departements- och myndigheters ansvar från antagande av gemenskapslagstiftning. Appendix III. GRATIS
- 00:21 24-timmarsmyndighet. Förslag till kriterier för statlig elektronisk förvaltning i medborgarnas tjänst. 100:-
- 00:22 Policymätning ett viktigt stöd för en bättre telefoni. 100:-
- 00:23 Administrationskostnader för regionala strukturfondsprogram. GRATIS
- 00:24 Slutredovisning – myndigheternas arbete med införandet av rättsinformationssystemet. GRATIS
- 00:25 En översyn av avgiftssystemet för inspektionen för strategiska produkter. GRATIS
- 00:26 Vårdmyndighet för nämndmyndigheter – utredning av förutsättningarna för utlokalisering av kanslistöd. GRATIS
- 00:27 Svenskundervisning för invandrare (sfi) – egen skolform eller del av komvux? GRATIS
- 00:28 Vinstandelsstiftelser. GRATIS
- 00:29 Att spara växel pengar. 100:-
- 00:30 Vad är XML? 125:-
- 00:31 Vad är XML Familjen? 125:-
- 00:32 Vad är XML Schema? 125:-
- 00:33 Vad är XML Formatmallar? 125:-
- 00:34 Vad är XML Länkar? 125:-
- 00:35 Vad är Document Object Model (DOM)? 125:-
- 00:36 Vad är XML Query Language? 125:-
- 00:37 Nummerinformation och katalogtjänster. 100:-
- 00:38 Vägledning – ramavtal för programvaror och tjänster. 100:- (*för stat, kommun och landsting) *GRATIS
- 00:39 Administration inom Regeringskansliet. Förstudie av Förvaltningsavdelningen. GRATIS

- 00:40 Elektroniska signaturer och elektronisk identifiering för myndigheters e-tjänst. 100:-
- 00:41 The 24/7 Agency – Criteria for 24/7 Agencius in the Networked Public Administration. GRATIS
- 00:42 Utlokalisering av sprängämnesinspektionen – Sammanslagning med Räddningsverket? 100:-
- 00:43 Förvaltningens utveckling – förvaltningspolitikens genomslag. GRATIS
- 00:44 Vägledning om ramavtal för dokumenthanteringssystem (DHS). 100:- (GRATIS för offentlig förvaltning)
- 00:45 Kan rättsväsendets information samordnas? – En utvärdering. GRATIS
- 00:46 Med världen i sikte. En studie av Sveriges internationella försvarsmaterielsamarbete. 100:-
- 00:47 Översyn av verksförordningen. En förstudie. GRATIS
- 00:48 Att verka i EU. En studie av hur medlemsländerna deltar i EG:s utvecklingssamarbete. GRATIS
- 00:49 Modernitet, effektivitet och förvaltningsförnyelse – en kartläggning och analys av Europeiska kommissionens vitbok om interna reformer. GRATIS



STATSKONTORET

Box 2280, 103 17 Stockholm.
Norra Riddarholmshamnen 1.
Telefon 08-454 46 00. Fax 08-791 89 72.
www.statskontoret.se