Innehållsförteckning

1 Document Object Model (DOM)

I tidigare lektioner har vi tittat på två olika sätt att visa XML-dokument i en webbläsare. Dessa sätt räcker inte alla gånger. Ibland finns även behov av att utföra förändringar i ett XML-dokument. I sådana tillfällen är DOM ett bra verktyg. I denna lektion tittar vi på vad DOM är och hur vi använder DOM med hjälp av JavaScript och lite Ajax.

1.1 Inledning

Fram till nu har kursen behandlat XHTML- och XML-dokument. Vi har använt oss av CSS och framför allt XSLT för att presentera dokumenten i en webbläsare. Från och till uppstår behov av att förändra de data som presenteras, men detta är inte möjligt i CSS eller XSLT. Förändringar kan vara av den typen att data, efter att den väl har presenterats, ska sorteras, flyttas ellerändras på. Till detta använder vi Document Object Model (DOM).
DOM tillåter att en användare kan läsa, söka i, ändra, manipulera, lägga till och ta bort data i ett XML-dokument. Därmed kan användarna i stort sett göra vad de vill med de data som presenteras, som till exempel att kombinera data från flera dokument. DOM är ett Application Programming Interface (API) för HTML-, XHTML- och XML-dokument. Detta innebär att man kan använda denna standard i många olika programmeringsspråk som Java, C, JavaScript och liknande. DOM är inte beroende av vare sig plattform eller programspråk.
Det har kommit ut fyra specifikationer:
- **DOM Level 1**. Här har man koncentrerat sig på kärnmodellen i HTML- och XMLdokument,
samt funktionalitet för dokumentnavigering och manipulering.
- **DOM Level 2**. Inkluderar en objektmall för stilmallar och definierar funktionalitet för att manipulera layout-information som är kopplat till dokumentet.
- **DOM Level 3**. Inkluderar laddning och lagring av dokument, schemaspråk som DTD och XML Schema samt validering av dokument.
- DOM Level 4. Lägger till Mutation Observers istället för Mutation Events
I denna lektion tittar vi på DOM Level 1. Lektionen börjar med en beskrivning av DOM och dess egenskaper och metoder. I andra halvan av lektionen går igenom flera exempel för vi använder DOM. I lektionen använder vi DOM tillsammans med JavaScript. JavaScript är enklare att använda ett programmeringsspråk som Java eller C och du behöver därför inga särskilda programmeringskunskaper för att förstå och kunna använda koden. Fokus vill vi ska ligga på DOM och inte på programmeringsspråket.

1.2 DOM Levels

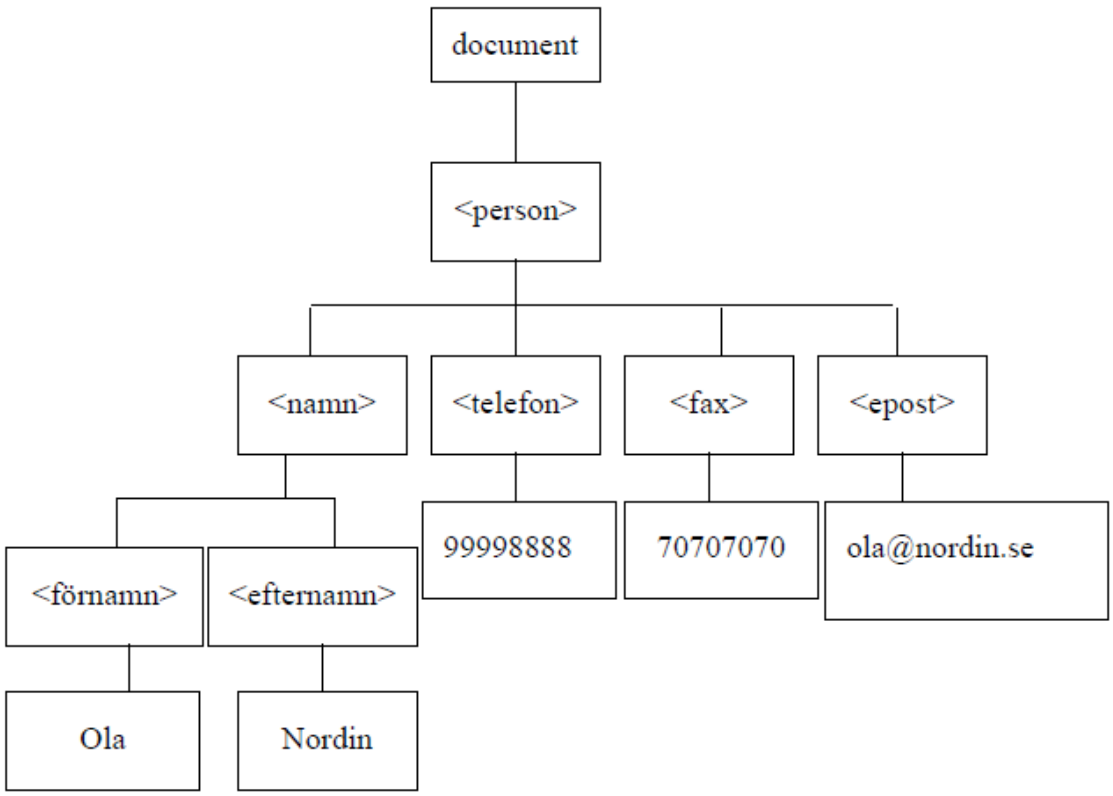
DOM baseras på den hierarkiska trädstrukturen XHTML- och XML-dokument har och de olika delarna representeras som en samling av objekt. För XML-dokument är det framför allt elementen som är objekt. Detta gör att man kan komma åt olika delar av ett dokument för att se på eller manipulera data. Det är värt att notera att hela dokumentet kommer att lagras i datorns minne under bearbetningen. Om storleken på dokumentet är 100 kB kommer datorn att behöva 1 MB för att lagra dokumentet i minnet under behandlingen (filens storlek x 10). Detta kan vara bra att veta om man arbetar med en dator som har begränsade resurser. I denna kurs kommer XML-dokumenten att vara så små att det inte leder till några problem.

1.3 DOM Noder

Tidigare har du lärt dig att se på ett XML-dokument uppbyggt som en trädstruktur. När DOM används för att manipulera XML-dokument kommer dokumentet först att tolkas. Under denna process kommer dokumentet att delas upp i separata delar som; element, attribut, entiteter, processinstruktioner och kommentarer. Alla dessa delar kommer att se på som noder. Utifrån dessa delar skapar DOM en representation av XML-dokumentet som ett nodträd och lagrar det i datorns minne.
Det används olika begrepp för att representera de olika delarna av nodträdet. För att illustrera detta ska vi börja med ett XML-dokument:

```
<?xml version="1.0" encoding=”UTF-8”?>
<person>
  <namn>
    <förnamn>Ola</förnamn>
    <efternamn>Nordin</efternamn>
  </namn>
  <telefon>99998888</telefon>
  <fax>70707070</fax>
  <epost>ola@nordin.se</epost>
</person>
```

Nodträdet ser då ut som figuren nedan:



Nodträdet ovan kan nu användas till att definiera olika begrepp och uttryck:

1. Alla nod-träd har en dokumentnod överst i trädet.
2. Rotnoden är `<person>` och den har fyra barn.
3. `<namn>` är det första barnet till `<person>`.

4. `<epost>` är det sista barnet till `<person>`.
5. `<namn>`, `<telefon>`, `<fax>` och `<epost>` är syskon, då de har samma föräldernod.
6. `<telefon>` är nästa syskon till det första barnet till `<person>`.
7. Det föregående syskonet till det sista barnet är till `<person>` är `<fax>`.
8. Det första barnet till rotnoden har två barn: `<förnamn>` och `<efternamn>`.
9. `<telefon>` har bara ett barn, som är dess innehåll (textnod).

Så här kan vi fortsätta med hela trädet. Som du ser finns det olika typer av relationer mellan de olika noderna. Uttrycken vi nämnde i listan ovan används mycket i DOM. Om du till exempel vill finna specifika element i ett XML-dokument, kan detta göras på flera sätt:
- Du kan skapa en lista över alla barn till rotelementet och därefter säga: `”finn det tredje barnet”`.
- `”Finn ett element med namnet telefon”`. Om det då finns flera `telefon`-element kommer du att få en lista över alla telefon-element (en nodlista).

Här används som sagt uttrycken för att finna specifika delar i ett XML-dokument. Med hjälp av DOM kan vi även förändra ett XML-dokument. Det kan till exempel vara att:
Radera det sista barnet till `<person>`.
- Ändra innehållet i `<telefon>` till `65656565`.
- Kopiera hela eller delar av XML-dokumentet och lägg det i ett annat XML-dokument.
- Skapa en ny `<telefon>` efter det andra barnet till rotelementet.

Varje nod i nodträdet representerar ett objekt med sina egna funktioner och egenskaper. Varje nodträd innehåller:
- Ingen eller en `DOCTYPE`-nod.
- En rotnod som innehåller en eller flera andra nodtyper.
- Ingen till många kommentarer eller processinstruktioner.

DOM definierar den logiska strukturen för dokumentet och sätter som sagt upp allt innehåll i XML-dokumentet som noder i ett notträd.

1.4 DOM Object

1. `Document`: representerar hela dokumentet.
Barnnoder: `Element` (rotelementet), `Processing Instruction`, `Comment` och `Document Type`
2. `Document Fragment`: en del av ett XML-dokument. Denna kan bestå av flera noder.
Barnnoder: `Element`, `Processing Instruction`, `Comment`, `Text`, `CDATA` och `Entity Reference`
3. `Document Type`: representerar `DOCTYPE`-deklarationer. Med hjälp av denna får man tillgång till entiteter och notationer i en DTD.
Barnnoder: `Notation` och `Entity`
4. `Entity Reference`: gäller för referensen till entiteter som står i XML-dokumentet.
Barnnoder: `Element`, `Processing Instruction`, `Comment`, `Text`, `CDATASection`, `Entity Reference`
5. `Element`: representerar element i ett XML-dokument.
Barnnoder: `Element` (vid nästling), `Text`, `Comment`, `Processing Instruction`, `CDATASection` och `Entity Reference`
6. `Attr`: representerar ett attribut i XML-dokumentet.
Barnnoder: `Text` och `Entity Reference`
7. `Processing Instruction`: för processinstruktioner.
Barnnoder: inga
8. `Comment`: kommentarer i dokumentet.
Barnnoder: inga
9. `Text`: innehållet i ett element eller attribut (som en textsträng).

Barnnoder: inga
10. `CDATASection`: för `CDATA`-sektioner.
Barnnoder: inga
11. `Entity`: för entiteterna som står i en DTD.
Barnnoder: inga
12. `Notation`: för `notation`-element i en DTD. Vi har inte använt denna hittills.
Barnnoder: inga
Vi ska nu illustrera dessa olika objekt med följande XML-dokument:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- File Name: person.xml -->
3 <!DOCTYPE person SYSTEM "person.dtd">
4 <?xml-stylesheet type="text/xsl" href="person.xsl"?>
5
6 <person>
7   <namn>Karin Nordin</namn>
8   <telefon type="hem">11111111</telefon>
9 </person>
```

Rad 1: XML-deklarationen. Denna tas inte med i nodträdet.
Rad 2: en kommentar som blir ett `Comment`-objekt.
Rad 3: specificerar dokumenttypsdefinitionen och är ett objekt av typen `Document Type`.
Rad 4: är en processinstruktion för att lägga till en stilmall. Denna är ett objekt av typen `Processing Instruction`.
Rad 6: rotelementet som är av typen `Element`. Denna har två barn som båda är av typen `Element`.
Rad 7: är ett element, med ett barn; ett textelement (med värdet `Karin Nordin`) av typen `Text`.
Rad 8: är ett element, med två barn; attributet `type` som är ett `Attr`-objekt och elementtexten `11111111` som är ett `Text`-objekt.
Rad 9: avslutar rotelementet.

1.5 Noders egenskaper och metoder

Alla DOM-noder har egenskaper och metoder som kan användas för att navigera i nodträdet. Dessa kan även användas för att komma åt och uppdatera noderna i nodträdet.

1.5.1 Nodegenskaper

Alla noder har en uppsättning med generella egenskaper som visas i tabellen nedan:

Namn	*Typ*	*Beskrivning*
`nodeName`	String	Representerar namnet på noden.
`nodeValue`	String	Representerar nodens värde. Om noden inte har ett värde kommer `null` att returneras. Kan användas på element, attribut, kommentarer, CDATA-sektioner och processinstruktioner.
`nodeType`	short integer	Varje typ av nod representeras av ett heltal enligt till exempel: Element = 1, Attribute = 2, Text = 3, Comment = 8, Document = 9.
`childNodes`	nodeList	Representerar en samling av barnnoder.
`attributes`	NamedNodeMap	Returnerar en speciell samling med all attribut till noden. För de noder som inte har några attribut returneras `null`.
`parentNode`	Node	Representerar föräldranoden.
`firstChild`	Node	Representerar första barnnoden.
`lastChild`	Node	Representerar sista barnnoden.
`nextSibling`	Node	Representerar nästa barnnod (till föräldernoden).
`previousSibling`	Node	Representerar föregående barnnod (till föräldernoden).
`ownerDocument`	Document Node	Returnerar hela dokumentet.

1.5.2 Nodmetoder

Nedan följer en lista med några generella nodmetoder:

Namn med parameterlista	*Returtyp*	*Beskrivning*
`appendChild(newChild)`	Node	`newChild` är en nod som kommer att läggas till i slutet av listan med barn för den aktuella noden.
`cloneNode(boolean)`	Node	Returnerar en exakt kopia av den nod metoden anropas på. Om parametern är `true` kommer även alla barnnoder att kopieras.
`hasChildNodes()`	Boolean	Returnerar `true` om noden har barnnoder.
`insertBefore(newNode, refNode)`	Node	Metoden lägger in noden `newNode` före noden `refNode`. Den nye noden returneras. Om `refNode` sätts till `null` kommer `newNode` att läggas till sist i barnlistan.
`removeChild(nodeName)`		Tar bort noden med namnet `nodeName`.
`replaceChild(newNode, oldNode)`	Node	Noden `oldNode` tas bort från barnlistan och noden `newNode` sätts in på samma position. `oldNode` returneras.

1.6 Att använda DOM med JavaScript och Ajax

I lektionens inledning nämnde vi några olika programmeringsspråk som kunde användas tillsammans med DOM. Bl.a JavaScript.
Vi kommer att läsa ett XML-dokument som finns lagrat på hårddisken och manipulera detta med JavaScript.
För dig som inte använt JavaScript och Ajax tidigare och vill ha en enkel introduktion finns det en
genomgång av språket på W3Schools hemsida: http://www.w3schools.com/js/default.asp. Du förväntas inte kunna något om JavaScript och Ajax utöver det som tas upp i denna lektion.

1.6.1 XMLHttpRequest

Alla moderna webbläsare har ett inbyggt **XMLHttpRequest**-objekt för att begära data från en server.

Med detta objekt kan du:
Uppdatera en webbsida utan att ladda om sidan
Begära data från en server - efter att sidan har laddats
Ta emot data från en server - efter att sidan har laddats
Skicka data till en server - i bakgrunden

XML-dokumentet som används

Nedanstående XML-dokumentet används i exemplet

```
<?xml version="1.0" encoding="UTF-8"?>

<Company>
    <Employee category="Technical" id="firstelement">
        <FirstName>Tanmay</FirstName>
        <LastName>Patil</LastName>
        <ContactNo>1234567890</ContactNo>
        <Email>tanmaypatil@xyz.com</Email>
    </Employee>
    <Employee category="Non-Technical">
        <FirstName>Taniya</FirstName>
        <LastName>Mishra</LastName>
        <ContactNo>1234667898</ContactNo>
        <Email>taniyamishra@xyz.com</Email>
    </Employee>
    <Employee category="Management">
        <FirstName>Tanisha</FirstName>
        <LastName>Sharma</LastName>
        <ContactNo>1234562350</ContactNo>
        <Email>tanishasharma@xyz.com</Email>
    </Employee>
</Company>
```

1.6.2 Dom och JavaScript exempel

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <div>
      <b>FirstName:</b> <span id="FirstName"></span><br>
      <b>LastName:</b> <span id="LastName"></span><br>
      <b>ContactNo:</b> <span id="ContactNo"></span><br>
      <b>Email:</b> <span id="Email"></span>
    </div>
    <script>
      //if browser supports XMLHttpRequest
      if (window.XMLHttpRequest)
      {
        //Firefox, Chrome, Opera, Safari
        xmlhttp = new XMLHttpRequest();
      }

      // sets and sends the request for calling "node.xml"
      xmlhttp.open("GET", "company.xml", false);
      xmlhttp.send();
      // sets and returns the content as XML DOM
      xmlDoc = xmlhttp.responseXML;
      //parsing the DOM object
      document.getElementById("FirstName").innerHTML =
```

```
            xmlDoc.getElementsByTagName("FirstName")[0].childNodes[0].nodeValue;
        document.getElementById("LastName").innerHTML =
            xmlDoc.getElementsByTagName("LastName")[0].childNodes[0].nodeValue;
        document.getElementById("ContactNo").innerHTML =
            xmlDoc.getElementsByTagName("ContactNo")[0].childNodes[0].nodeValue;
        document.getElementById("Email").innerHTML =
            xmlDoc.getElementsByTagName("Email")[0].childNodes[0].nodeValue;
    </script>
  </body>
</html>
```

Explorer

Internet Explorer använder ActiveXObject ("Microsoft.XMLHTTP") för att skapa en instans på ett XMLHttpRequest-objekt, andra webbläsare använder XMLHttpRequest () -metoden.

1.6.3 Få ut information med hjälp av noders egenskaper och metoder

Vi är nu intresserad av att hämta ut information om var och en av barnnoderna till `person`. Informationen ska bestå av nodens namn, typ och värde. Vi kommer då att använda DOM egenskaperna `nodeName`, `nodeType` och `nodeValue`. Det första vi måste göra är att få tag i rotelementet. Vi kan du med hjälp av den hämta alla barn och skriva ut information om dem. För att hämta ut rotelementet använder vi egenskapen `documentElement`:

```
root = xmlDoc.documentElement
```

Nu kommer `root` att innehålla en referens till själva rotnoden. Vi kan nu använda denna för att hämta ut alla barnnoder. Som du kanske minns (från kapitel 1.5) har en nod egenskapen `childNodes`, vilken vi använder enligt nedan:

```
rootList = root.childNodes
```

Variabeln `rootList` kommer nu att innehålla en lista över alla barnelement. När vi ska referera till ett bestämt barn i denna lista måste vi börja med värdet `0`:

`rootlist.item[0]` – första barnet
`rootlist.item[1]` – andra barnet

För att gå igenom alla barnnoder och skriva ut information om dem måste vi använda en loopfunktion.
Vi måste då veta hur många barnnoder vi ska gå igenom, vilket vi får reda på genom att hämta ut antalet barn i nodlistan:

```
len = rootList.length
```

Nu kan vi gå igenom alla barnnoder och skriva ut information om namn, typ och värde. Vi kommer att använda en `for`-loop för att gå igenom alla barn.
. Här är ett exempel på ett javascript-kod. Till detta använder vi variabeln `len`, som vi nyss gett ett värde. En for-loop:

```
for (i = 0; i < len; i++) {
        j = i + 1;
        document.write("Nod" + j + "<br />");
        document.write("Nodens namn är: " + rootList.item[i].nodeName + "<br />");
        document.write("Nodens typ är: " + rootList.item[i].nodeType + "<br />");
        document.write("Nodens värde är: " + rootList.item[i].nodeValue + "<br />");
    }
```

1.6.4 XML-fil byggs upp av text

Följande exempel visar hur man laddar XML-data med Ajax och Javascript när XML-innehåll mottas som XML-fil. Här får Ajax-funktionen innehållet i en xml-fil och lagrar den i XML DOM. När DOM-objektet har skapats, parsas det sedan.

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
  <head>
    <title>DOM</title>
    <script>
      // loads the xml string in a dom object
      function loadXMLString(t)
      {
        // for non IE browsers
        if (window.DOMParser)
        {
          // Skapa en instans för xml dom objektet
          parser = new DOMParser();
          xmlDoc = parser.parseFromString(t, "text/xml");
        }

        // kod för IE
        else
        {
          // Skapa en instans för xml dom objektet
          xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
          xmlDoc.async = false;
          xmlDoc.loadXML(t);
        }
        return xmlDoc;
      }
    </script>
  </head>
  <body>
    <script>
      // här byggs en variable upp med xml-kod
      var text = "<Employee>";
      text = text + "<FirstName>Tanmay</FirstName>";
      text = text + "<LastName>Patil</LastName>";
      text = text + "<ContactNo>1234567890</ContactNo>";
      text = text + "<Email>tanmaypatil@xyz.com</Email>";
      text = text + "</Employee>";

      // anropar funktionen loadXMLString() med variabeln "text" som argument och som sparas i
en variabel
```

```
        var xmlDoc = loadXMLString(text);

        //parsar DOM-objektet
        y = xmlDoc.documentElement.childNodes;
        for (i = 0; i < y.length; i++)
        {
           //Noderna i dokumentet skrivs ut
           document.write(y[i].childNodes[0].nodeValue);
           document.write("<br>");
        }
      </script>
   </body>
</html>
```

1.7 Ev fortsättning

Det finns många möjligheter genom att använda XML DOM, Javascript och Ajax.
Vi kommer inte just nu att gå djupare i den här kursen då javascript och Ajax i denna kurs, utan ni kommer bara tangera ämnet och kommer inte begäras kunna något mer än det vi gått igenom här. Däremot så tycker jag att ni bör studera dessa tekniker ytterligare själva, då det är mycket bra att behärska.