

Uppmärkningsspråk

TDP007 Konstruktion av datorspråk
Föreläsning 4

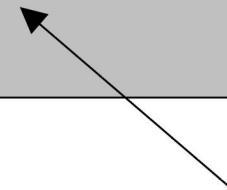


Från förra gången

- XML-dokument specificeras med DTD/XLS
- Två olika sätt att parsa XML-dokument:
 - *Strömparsning läser in dokumentet, i princip ett tecken i taget, och skickar signaler när något hittas (starttagg, sluttagg, etc)*
 - *Trädparsning*

Exempel på användning av DOM

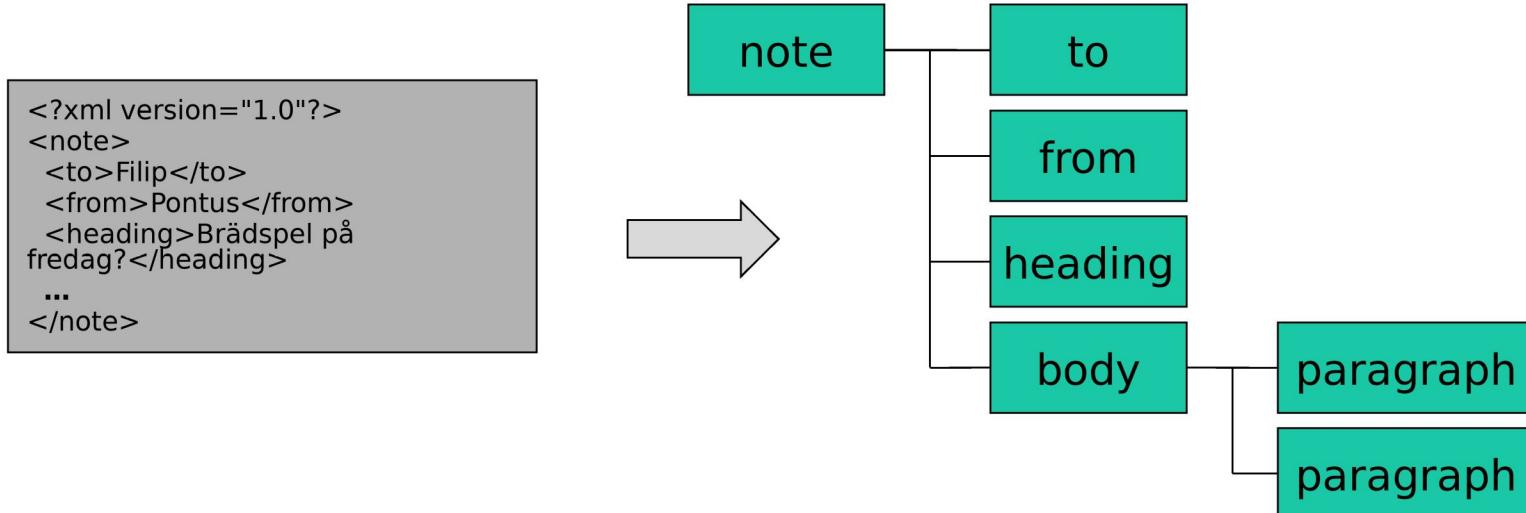
```
>> require 'rexml/document'  
=> true  
>> src = File.open "note2.xml"  
=> #<File:note2.xml>  
>> doc = REXML::Document.new src  
=> <UNDEFINED> ... </>
```



*Detta anrop parser hela XML-filen
och returnerat ett DOM-objekt.
Vad finns i detta objekt och hur
kommer man åt det?*

Trädparsning

Vid trädparsning läses hela XML-filen in på en gång och parsern returnerar ett objekt som motsvarar hela innehållet, enligt DOM (Document Object Model).

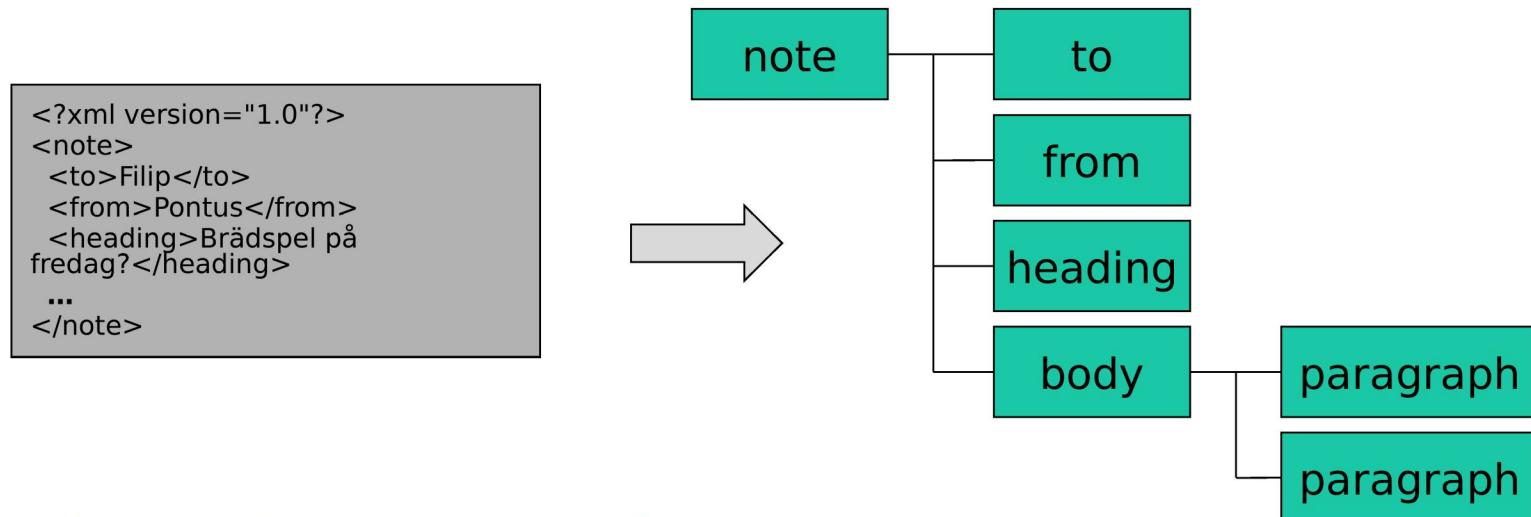


Officiell specifikation: www.w3.org/DOM

DOM

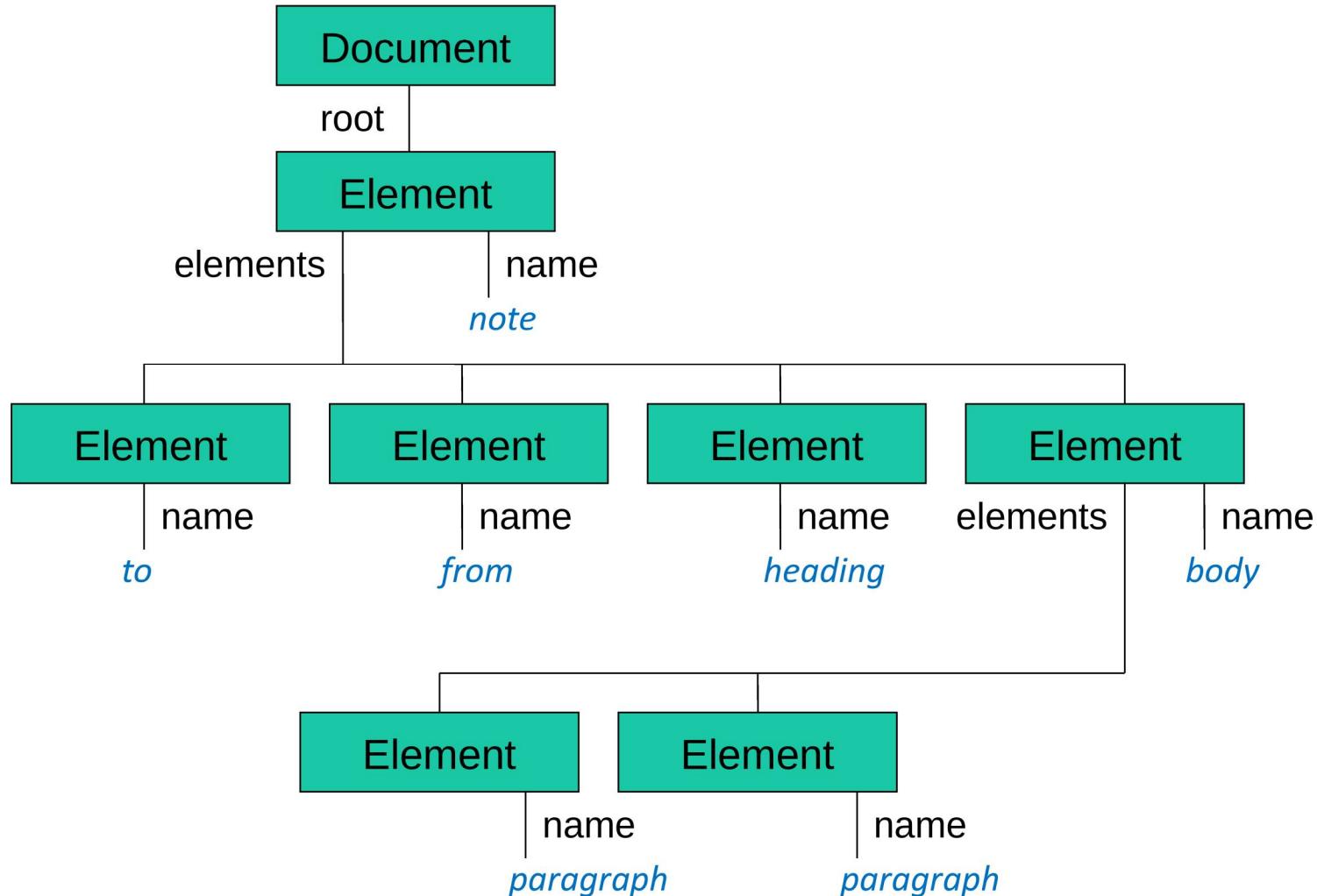
Denna modell representerar ett dokument som ett logiskt träd. Låter program göra intressanta saker med datan i HTML/XML.

Ursprungligen designat för Javascript och webben.



Officiell specifikation: www.w3.org/DOM

DOM-struktur för vårt XML-exempel



Exempel på metoder för DOM

```
>> doc = REXML::Document.new src  
=> <UNDEFINED> ... </>  
>> doc.class  
=> REXML::Document  
>> r = doc.root  
=> <note> ... </>  
>> r.class  
=> REXML::Element  
>> r.name  
=> "note"  
>> r.elements[1].name  
=> "to"  
>> r.elements[4].elements[1].text  
=> "Brädspel på fredag?"
```

Dokumentet är ett objekt av klassen **Document**. Det har ett attribut **root** som representerar rotelementet.

Varje element har en ”array” **elements** som innehåller ”barnen”. Observera att den har basen 1, inte 0.

Attributet **name** innehåller elementets namn.

Övning

- Ladda in filen **note2.xml** och plocka ut namnen på sändare och mottagare ur DOM-objektet.
- Ladda in följande funktion från **print_tree.rb**, kör den på det inlästa XML-dokumentet och förklara vad som händer.

```
def print_tree(elem, indent=0)
    elem.elements.each do |subelem|
        puts " "*2*indent + subelem.name
        print_tree(subelem, indent+1)
    end
end
```

Metoder att kunna **REXML::Document**

- ::new
- ::root
- ::parse_stream

REXML::StreamListener (template)

- Hur man utökar
- ::tag_start
- ::tag_end
- ::text
- ::comment
- ::cdata

REXML::Element

- `::elements`
- `::[]`
- `::text / ::texts`
- `::next_element / ::previous_element`
- `::each_element / ..with_attribute / ..with_text`
- `::add_attribute / ::add_text / ::add_element`

XPath

- Att plocka fram elementen ur dokument-objektet gör att man måste hårdkoda dokumentets specifikation i programmet. Det är inte så flexibelt att peka ut delar av dokumentet med uttryck som **r.elements[4].elements[2].text**.
- Ett lite enklare sätt är att använda XPath.
- XPath (XML Path Language) är ett språk för att välja ut delar av ett XML-dokument. Man kan tänka på det som ett sätt att formulera sökvägar till delar av dokumentet.
- Flera av funktionerna i REXML-paketet stödjer XPath-uttryck.
- XPath 1.0 är från 1999, XPath 2.0 är från 2007 och XPath 3.0 är från 2014.

Exempel på XPath-uttryck

Alla *paragraph*-noder inuti en *body*, inuti en *note* som är roten i dokumentet.



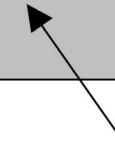
```
>> doc.elements.each("/note/body/paragraph") { |n| puts n.text }
```

Det har öppnat ett nytt holländskt ställe
på S:t Larsgatan. De lär ha en massa sorters holländsk
öl på fat. Vad sägs om att vi testar det på fredag kväll?
Förresten, stället heter De Klomp.

=> ...

```
>> doc.elements.each("//body/*") { |n| puts n.text }
```

Samma resultat som ovan...



Alla noder, vilka som helst (*), som ligger inuti en *body* som ligger någonstans
under roten, oavsett nivå.

Konstruktion av XPath-uttryck

- Alla XPath-uttryck har en **kontext** som tjänar som utgångspunkt (en nod i dokumentet).
- Från utgångspunkten går man med hjälp av tre saker (alla behöver inte vara med):
 - *en riktning, t.ex. barn eller förälder*
 - *ett nodtest, t.ex. att man söker noder med ett visst namn*
 - *ett predikat som talar om att noderna ska ha ytterligare egenskaper*
- Läs på om XPath och experimentera på egen hand! Det finns gott om bra introduktioner på nätet.

Fler exempel på XPath-uttryck

/note/from

Väljer noden *from* som finns under noden *note* som finns direkt under dokumentets rot.

//from

Väljer noden *from* som finns någonstans under dokumentets rot.

paragraph[2]

Väljer den andra *paragraph*-noden som finns under den aktuella noden.

..

Väljer den aktuella nodens föräldranod.

@version

Väljer attributet *version* i den aktuella noden.

para[@color="blue"]

Väljer den första *para*-noden under den aktuella noden som har attributet *color*...

Xpath-uttryck Wildcards

* Vilket element som helst

/key/* Alla barn i elementet key

@* Vilken attributenod som helst

//alt[@*] Alla element **alt** som har något attribut

Övning

Utgår från filen **foobar.xml**

Skriv en funktion som hämtar ut den totala lönen för alla personer som har rollen "leader" i division som heter "Development".

Lösningsförslag i **foobar.rb** om man vill se ett sätt att lösa problemet på.

Ett till Exempel

Utgår från filen **animals.xml**.

Xpath för att söka efter element på olika sätt

```
<?xml version="1.0"?>
<!DOCTYPE key SYSTEM "key.dtd">

<key header="Animals">

<node id="start" header="Classification of animals">
<alt dest="bird">Has wings and flies</alt>
<alt dest="fish">Lives in the water</alt>
<alt dest="land">Mostly walks on the ground</alt>
</node>

<node id="bird" header="Birds">
<alt dest="parus_major">...</alt>
<alt dest="cyanistes_caeruleus">...</alt>
</node>

...
</key>
```

Övning att göra själv

- Kopiera filen **animals.xml** och tillhörande DTD i filen **key.dtd**. Filen innehåller en bestämningsnyckel för djur.
- Skriv ett kort program som kan använda bestämningsnyckeln och ställa frågor till användaren. Se exempel på implementation och hur programmet är menat att köra i **key.rb** på kurssidan.
- Tips: Leta upp API-dokumentationen för REXML-paketet på nätet!

ruby-doc.org under *Standard Library API* eller via
<https://ruby.github.io/rexml/>

Microformat

- Microformats är en uppsättning små fria standarder för att representera vissa specifika typer av information.

Ett typiskt exempel är en överenskommelse om hur man kan bädda in kontaktinformation i en webbsida.

```
<span class="vcard"><span class="fn">Kalle  
Kula</span></span>
```

- Mer information finns på
www.microformats.org

Sammanfattning

- Tre sätt att bearbeta ett XML-dokument:
 - *Implementera egen SAX-lyssnare*
 - *Bygga DOM och använda metoder för element*
 - *Bygga DOM och använda XPath*
- Förkortningar vi har pratat om:
 - *API, DOM, DTD, HTML, SAX, SGML, XHTML, XML, XPath*

Seminarie 2 och dugga

- Övning 3 på seminarie 2
- Testbarheten
- Dugga, 4 uppgifter
 - 3 små (regexp, sax-xml, dom-xml)
 - 1 stor (xml)
- Tentan har 2 delar
 - Klara alla små uppgifter på duggorna, del 1 till godo. Klara små och stora, betyg 3 tillgodo

www.liu.se