XSL Stylesheets - XPath

Παπαδάκης Γιάννης

29/11/2010

Τί θα μάθουμε

- Πως να σχεδιάζουμε XSL stylesheets έτσι ώστε να:
  - παρουσιάζουμε XML έγγραφα
  - μετασχηματίζουμε XML έγγραφα
- Πως να περιγράφουμε patterns σε XML έγγραφα μέσω XPath
- Μια εισαγωγή στα XSL formatting objects
- Διαθέσιμα εργαλεία για XSL

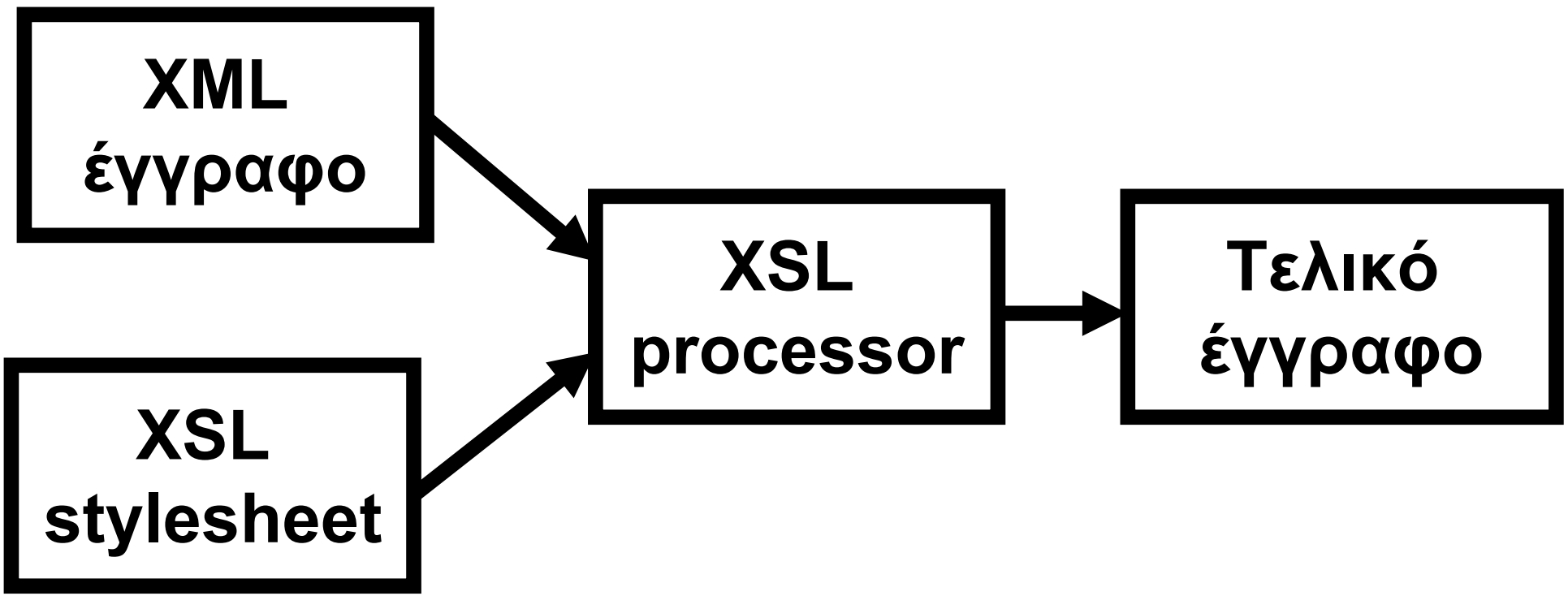
Τί είναι η XSL?

- **Η eXtensible Stylesheet Language είναι:**
  - **Μια γλώσσα για μετασχηματισμό XML εγγράφων: "XSLT"**
  - **Μια ορολογία για περιγραφή της παρουσίασης XML εγγράφων: "XSL formatting objects"**
- **Ενα XSL stylesheet περιγράφει πως θα μετασχηματιστεί ή παρουσιαστεί μια κλάση XML εγγράφων**
- **Η XSL είναι κωδικοποιημένη σε XML**
- **Η XSL αναπτύχθηκε από το World Wide Web consortium(http://www.w3.org/)**

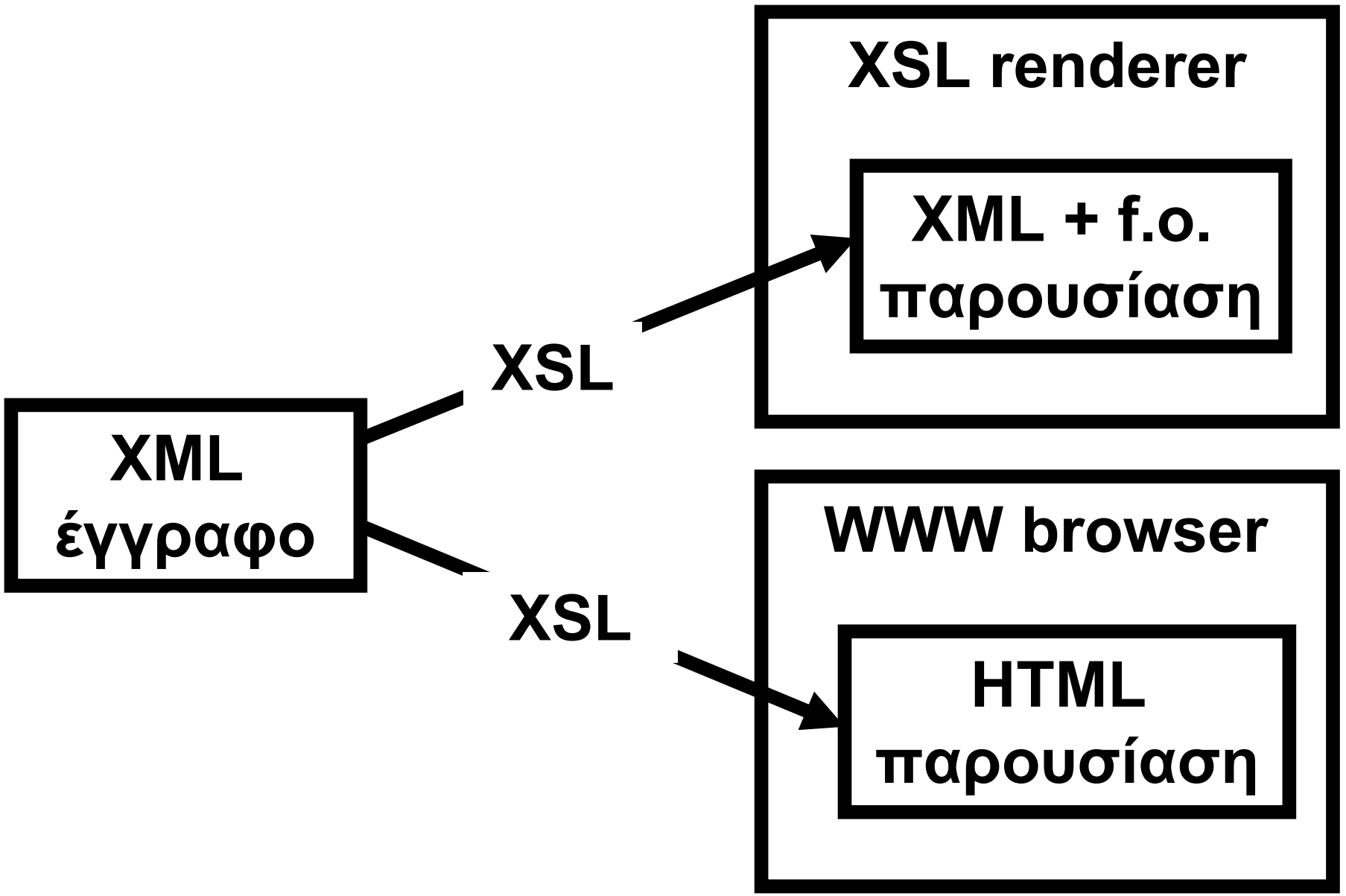
Γιατί να χρησιμοποιούμε XSL?

- διαχωρίζουμε παρουσίαση δεδομένων (XSL) από την αναπαράστασή τους (XML)
  - Ενα XML έγγραφο μπορεί να φαίνεται διαφορετικά ανάλογα με το stylesheet που του εφαρμόζεται
  - “Η XSL είναι για την XML ότι τα CSS για την HTML”
- Απλοποιούμε τη μετάφραση των δεδομένων από ένα XML format σε άλλο (XSLT)
  - Ενα XML έγγραφο από μια εφαρμογή μπορεί να “διαμορφωθεί” για μια άλλη μέσω XSL stylesheets αντί να χρησιμοποιήσουμε άλλου είδους κώδικα

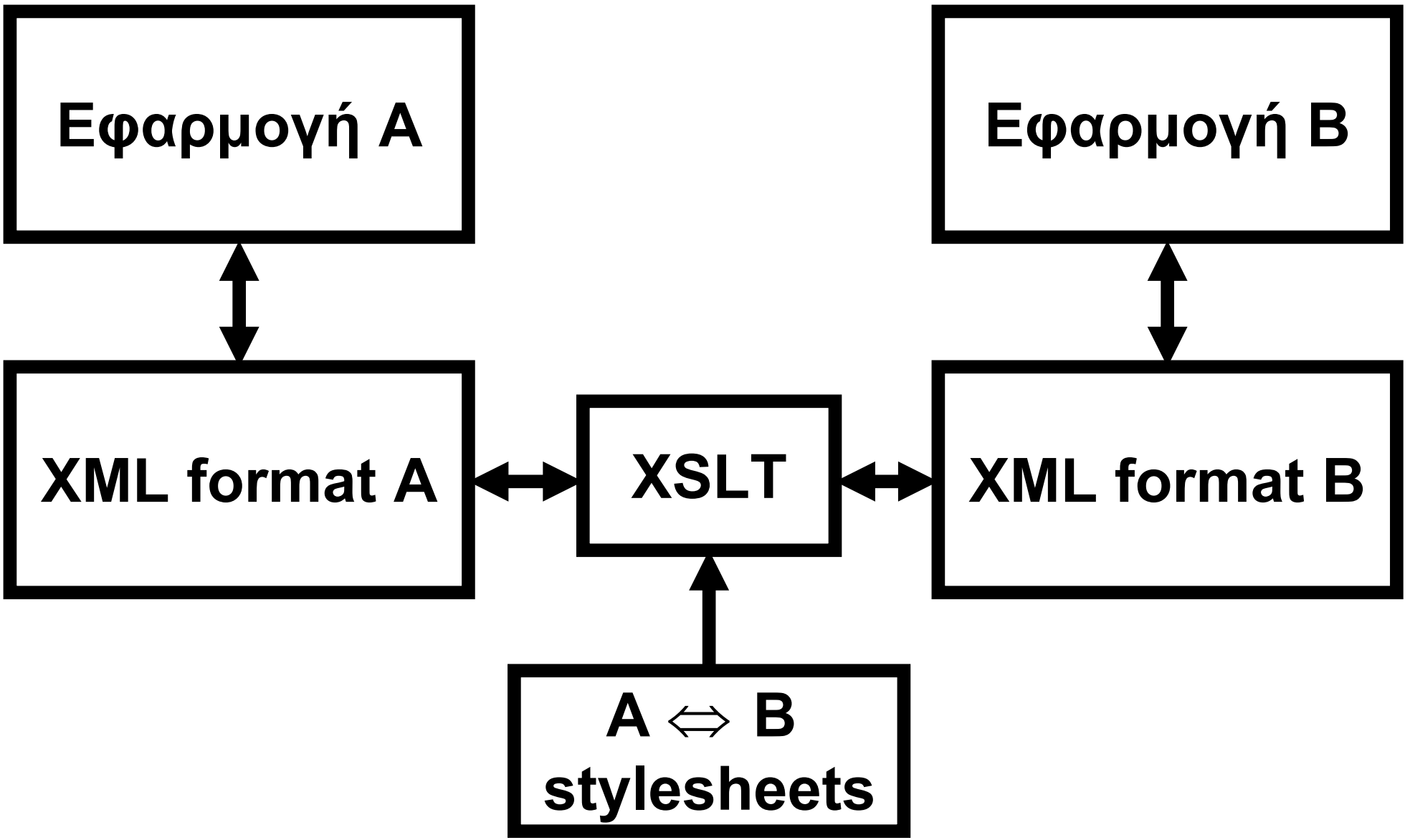
XSL Αρχιτεκτονική



XSL σενάρια χρήσης



XSL σενάρια χρήσης (συν.)



XSL αρχιτεκτονικές σημειώσεις

- Ενα XSL stylesheet μπορεί να χρησιμοποιηθεί για το μετασχηματισμό ή αναπαράσταση οποιουδήποτε instance (δηλ. .xml) του document type (δηλ. .xsd ή .dtd) για το οποίο κατασκευάστηκε
  - Χωρίς να χρειάζεται να συσχετιστούν άμεσα με κάποιο DTD ή xsd
- Μπορούν να υπάρχουν πολλά XSL stylesheets για το ίδιο document type
  - Πολλαπλές αναπαραστάσεις
  - Πολλαπλοί μετασχηματισμοί

Πως δουλεύει η XSL?

- Ο XSL processor διατρέχει-parses το XML αρχικό έγγραφο
- Templates στο stylesheet ταιριάζονται – matched με patterns στο αρχικό έγγραφο
- Τα templates επεκτείνονται, παράγοντας τμήματα του τελικού εγγράφου
- Η διαδικασία αυτή (συνήθως) εφαρμόζεται αναδρομικά μέχρι να εξαντληθούν όλα τα δυνατά ταιριάσματα (pattern-matches)
- Το ολοκληρωμένο τελικό έγγραφο παράγεται ως εξερχόμενο

XSL παράδειγμα

- **Εστω το παρακάτω XML έγγραφο ("`name.xml`"):**

```
<name>
   <first nick="Joe">Joseph</first>
   <last>Futrelle</last>
</name>
```

XSL παράδειγμα (συν.)

- Ορίζουμε το παρακάτω XSL stylesheet ("formal.xsl")

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="name">
    <xsl:value-of select="last"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="first"/>
  </xsl:template>
</xsl:stylesheet>
```

XSL παράδειγμα (συν.)

- Οταν το εφαρμόζουμε στο "`name.xml`", ο XSL processor παράγει το ακόλουθο κείμενο:

```
Futrelle, Joseph
```

XSL Παράδειγμα: Πώς δουλεύει?

- **Το template matches το `name` element στο έγγραφο.**
  - **Στο `formal.xsl`:**

```
<xsl:template match="name">
   ...
</xsl:template>
```

  - **Στο `name.xml`:**

```
<name>
   <first nick="Joe">Joseph</first>
   <last>Futrelle</last>
</name>
```

XSL Παράδειγμα: Πώς δουλεύει? (συν.)

- **Το template selects το sub-element με το όνομα `last`.**
  - **Στο `formal.xsl`:**

```
<xsl:template match="name">
  <xsl:value-of select="last"/>
  ...
```

  - **Στο `name.xml`:**

```
<name>
  <first nick="Joe">Joseph</first>
  <last>Futrelle</last>
</name>
```

XSL Παράδειγμα: Πώς δουλεύει? (συν.)

- **Ο XSL processor παράγει την «τιμή» του `last` (δηλ. Το περιεχόμενο κείμενο) ως εξαγόμενο.**
  - **Στο `formal.xsl`:**

```
<xsl:value-of select="last"/>
```

  - **Στο `name.xml`:**

```
<last>Futrelle</last>
```

  - **Στο τελικό έγγραφο:**

```
Futrelle
```

XSL Παράδειγμα: Πώς δουλεύει? (συν.)

- **Το template παράγει κείμενο.**
  - **Στο `formal.xsl`:**

```
<xsl:value-of select="last"/>
<xsl:text>, </xsl:text>
<xsl:value-of select="first"/>
```

  - **Στο τελικό έγγραφο:**

```
Futrelle,
```

XSL Παράδειγμα: Πώς δουλεύει? (συν.)

- Μια "value-of" οδηγία παράγει την τιμή του sub-element "first".
  - Στο formal.xsl:

```
<xsl:value-of select="first"/>
```

  - Στο name.xml:

```
<first nick="Joe">Joseph</first>
```

  - Στο τελικό έγγραφο:

```
Futrelle, Joseph
```

Ενα λιγότερο τυπικό παράδειγμα

- **Ας γράψουμε τώρα ένα άλλο template για το ίδιο XML έγγραφο: "casual.xsl"**

```
<xsl:template match="name">
  <xsl:value-of
        select="first/@nick"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="last"/>
</xsl:template>
```

Ενα λιγότερο τυπικό παράδειγμα (συν.)

- **Αν εφαρμόσουμε το "casual.xsl" στο "name.xml":**

```
<name>
   <first nick="Joe">Joseph</first>
   <last>Futrelle</last>
</name>
```

- **Ο XSL processor παράγει το εξής:**

```
Joe Futrelle
```

Ενα λιγότερο τυπικό παράδειγμα: Πως δουλεύει?

- "casual.xsl" είναι σχεδόν το ίδιο με το "formal.xsl", εκτός:
  - οι xsl:value-of οδηγίες για τα first και last ονόματα συμβαίνουν με διαφορετική σειρά
  - Η οδηγία xsl:text δεν περιέχει κόμμα
  - Η οδηγία value-of του first-name επιλέγει το nick attribute του first, αντί για την τιμή του, δηλ.

```
<first nick="Joe">Joseph</first>
```

αντί

```
<first nick="Joe">Joseph</first>
```

Ενα λιγότερο τυπικό παράδειγμα : Πως δουλεύει?

- Πώς επιλέγεις την τιμή ενός attribute?
  - Αντί

```
<xsl:value-of select="first"/>
```

  - χρησιμοποίησε

```
<xsl:value-of select="first/@nick"/>
```

- "`first/@nick`" σημαίνει «το `nick` attribute του `first`“»
- Αυτό είναι μια XPath έκφραση
- Η XSL χρησιμοποιεί XPath για ταιριάσματα – matching και επιλογές – selection

Ενα λιγότερο τυπικό παράδειγμα : Επιπλοκές

- Σχεδιάζοντας διαφορετικά XSL stylesheets, μπορούμε να αναπαραστήσουμε XML δεδομένα ποικιλοτρόπως
- Η XSL μας επιτρέπει να εξάγουμε μόνο την απαιτούμενη πληροφορία από το αρχικό έγγραφο (π.χ. Αγνοήσαμε το υποκοριστικό στο `formal.xsl`)
- Η XSL είναι σχεδιασμένη έτσι ώστε να διευκολύνει την επαναδόμηση ενός εγγράφου (π.χ. Αλλάξαμε τη σειρά των first και last name)
- Η XSL μπορεί να παράγει non-XML αποτέλεσμα το ίδιο εύκολα με XML

Εισαγωγή στην XPath

- Η XPath είναι μια σύνταξη για να αναφερόμαστε σε μέρη ενός XML εγγράφου
  - Περιγράφοντας μονοπάτια στην ιεραρχία του εγγράφου
  - Προσδιορίζοντας περιορισμούς προς ταίριασμα – match στη δομή του κειμένου
- Η XSL χρησιμοποιεί XPath εκφράσεις για
  - Να προσδιορίσει ποιά elements ταιριάζουν – match ένα template
  - Να επιλέξει κόμβους – nodes στους οποίους να εκτελέσει διάφορες διαδικασίες

Βασικά της XPath

- Οι εκφράσεις XPath μοιάζουν με τις διευθύνσεις του UNIX, π.χ.

```
poem/stanza/line
```

αναφέρεται σε «όλα τα `line` elements τα οποία είναι παιδιά των `stanza` elements τα οποία είναι παιδιά των `poem` elements"

- Οι εκφράσεις XPath εφαρμόζονται σχετικά με ένα "context node", ο οποίος αντιστοιχεί στον «τρέχων κατάλογο" στο UNIX ή στο DOS. Αυτή η XPath έκφραση είναι "."

Βασικά της XPath: ενα απλό παράδειγμα

- **Θεωρείστε το ακόλουθο XML έγγραφο:**

```
<poem>
  <title>Roses</title>
  <author>Ima Poet</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: ενα απλό παράδειγμα (συν.)

- **Η XPath "`poem/stanza/line`" επιλέγει**

```
<poem>
  <title>Roses</title>
  <author>I'm a Poet</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: wildcards

- **Η XPath "`poem/stanza/*`" επιλέγει**

```
<poem>
  <title>Roses</title>
  <author>Ima Poet</author>
  <stanza>
→ <line>Roses are red</line>
→ <line>violets are blue</line>
  </stanza>
  <stanza>
→ <line>I'm a poet</line>
→ <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: απόγονοι

- **Η XPath "`poem//punch`" επιλέγει:**

```
<poem>
  <title>Roses</title>
  <author>Ima Poet</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: σειριακά

- "poem/stanza/line[1]" επιλέγει:

```
<poem>
  <title>Roses</title>
  <author>John Doe</author>
  <stanza>
→   <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
→   <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: σειριακά (συν.)

- **"poem/stanza/line[position() = last()]" επιλέγει:**

```
<poem>
  <title>Roses</title>
  <author>John Doe</author>
  <stanza>
    <line>Roses are red</line>
→   <line>violets are blue</line>
  </stanza>
  <stanza>
→   <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: επιλέγοντας κόμβους κειμένου

- **"poem/author/text()" επιλέγει:**

```
<poem>
  <title>Roses</title>
  <author>John Doe</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: συνθήκες

- **"poem/stanza[punch]" επιλέγει:**

```
<poem>
  <title>Roses</title>
  <author>John Doe</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Βασικά της XPath: συνθήκες : ισότητα

- “//line[text()="I'm a poet"]”

```
<poem>
  <title>Roses</title>
  <author>John Doe</author>
  <stanza>
    <line>Roses are red</line>
    <line>violets are blue</line>
  </stanza>
  <stanza>
    <line>I'm a poet</line>
    <punch>and you're not!</punch>
  </stanza>
</poem>
```

Όλα για XPath

- Η XPath έχει πολλά περισσότερα χαρακτηριστικά από αυτά που έχουμε δει σήμερα
- Περισσότερα στο site του μαθήματος
- Επίσημος δικτυακός τρόπος
  http://www.w3.org/TR/xpath

Ταιριάσματα – Matching με XSL Templates

- Η XSL λειτουργεί ταιριάζοντας templates με XPath εκφράσεις, και μετά επεκτείνει τα templates, τα οποία αποτελούνται από XSL οδηγίες που εκτελούνται
- Τα Templates ορίζονται χρησιμοποιώντας
  - Την οδηγία `xsl:template`
  - Τα ονόματα της XSL οδηγίας βρίσκονται στο “xsl” namespace οπότε και ξεκινούν με "`xsl:`"
- Το `match` attribute του `xsl:template` περιέχει την έκφραση XPath προς ταίριασμα

Matching με XSL Templates (συν.)

- Για παράδειγμα, το template

```
<xsl:template match="item">
   <xsl:text>wow!</xsl:text>
</xsl:template>
```

θα ταιριάξει – match οποιοδήποτε element του τρέχοντα κόμβου με το όνομα "item" και θα εκτελέσει την οδηγία `xsl:text` για κάθε ένα

- Η οδηγία `xsl:text` χρησιμοποιείται για να προσθέσει κείμενο στο τελικό έγγραφο (σ' αυτήν την περίπτωση, "wow!")

Matching με XSL Templates (συν.)

- **Αν εφαρμόσουμε το template**

```
<xsl:template match="item">
    <xsl:text>wow!</xsl:text>
  </xsl:template>
```

- **σε όλους τους κόμβους του ακόλουθου XML εγγράφου:**

```
<menu>
  <item name="File"/>
  <item name="Edit">
    <item name="Cut"/>
    <item name="Copy"/>
  </item>
</menu>
```

- **Ο XSL processor τι θα παράγει?**

Μια συντόμευση για το xsl:text

- Οποιοδήποτε κείμενο υπάρχει μέσα σε ένα template αντιμετωπίζεται σαν να ήταν μια οδηγία `xsl:text`
- Οπότε, θα μπορούσαμε να ξαναγράψουμε το προηγούμενο παράδειγμα ως:

```
<xsl:template match="item">
   wow!
</xsl:template>
```

Λαμβάνοντας τιμές κόμβων με xsl:value-of

- Εστω ότι χρησιμοποιούμετο παρακάτω template στο παράδειγμα menu:

```
<xsl:template match="item">
   <xsl:value-of select="@name"/>
</xsl:template>
```

- Αν εφαρμόσουμε αυτό το template σε όλους τους κόμβους του εγγράφου menu, ο XSL processor θα παράγει (?):

```
FileEdit
```

Εφαρμόζοντας Templates με xsl:apply-templates

- Τα Templates μπορούν να εφαρμοστούν recursively χρησιμοποιώντας την οδηγία `xsl:apply-templates`
- Το "`select`" attribute της `xsl:apply-templates` είναι μια XPath έκφραση που προσδιορίζει ένα υποσύνολο του τρέχοντα κόμβου
- Αν κάποια templates στο stylesheet ταιριάζουν με τους κόμβους με αυτό το υποσύνολο, εφαρμόζονται
- Η τεχνική αυτή της «διάσχισης – walking down» του δένδρου του εγγράφου εφαρμόζοντας templates, είναι η πιο συνήθης για stylesheets

Εφαρμόζοντας Templates με xsl:apply-templates

- Αν περισσότερα του ενός template ταιριάζουν σε έναν κόμβο, υπάρχει προτεραιότητα εφαρμογής
  - Γενικά, εφαρμόζεται το πιο συγκεκριμένο
- Οταν εφαρμόζεται ένα template, ο κόμβος που ταίριαξε γίνεται ο τρέχων κόμβος
  - οπότε όλες οι εκφράσεις XPath που υπάρχουν στο template είναι σχετικές μ' αυτόν τον κόμβο

Εφαρμόζοντας Templates: Ενα παράδειγμα

- **Εστω το ακόλουθο XML έγγραφο:**

```
<poem>
  <stanza>
    <line>this stanza is short</line>
    <line>so take me to court</line>
  </stanza>
  <stanza>
    <line>this one's shorter</line>
  </stanza>
</poem>
```

Εφαρμόζοντας Templates: Ενα παράδειγμα (συν.)

- **Στο ακόλουθο απόσπασμα του stylesheet, stanzas και lines υπόκεινται επεξεργασία από 2 διαφορετικά templates:**

```
<xsl:template match="stanza">
  <blockquote>
    <xsl:apply-templates select="line"/>
  </blockquote>
</xsl:template>
<xsl:template match="line">
    <xsl:value-of select="."/>
    <br/>
</xsl:template>
```

Εφαρμόζοντας Templates: Ενα παράδειγμα (συν.)

- **Εφαρμόζοντας το πιο πάνω template στα "stanza" elements παράγει το ακόλουθο:**

```
<blockquote>
   this stanza is short<br/>
   so take me to court<br/>
</blockquote>
<blockquote>
   this one's shorter<br/>
</blockquote>
```

Επανάληψη με xsl:for-each

- **Η οδηγία `xsl:for-each` εφαρμόζει ένα σύνολο οδηγιών σε συνεχόμενους κόμβους σε μια ακολουθία. Εστω το ακόλουθο XML απόσπασμα:**

```
<list>
   <item>one</item>
   <item>two</item>
   <item>three</item>
</list>
```

- **Αν θέλαμε να το μετασχηματίσουμε σε:**

```
one, two, and three
```

Επανάληψη με xsl:for-each (συν.)

- **Θα μπορούσαμε σχεδόν να το επιτύχουμε με το ακόλουθο template:**

```
<xsl:template match="list">
  <xsl:for-each select="item">
    <xsl:value-of select="."/>
    <xsl:text>, </xsl:text>
  </xsl:for-each>
</xsl:template>
```

- **Που παράγει:**

```
one, two, three,
```

Συνθήκες με xsl:if

- **Μπορούμε να ραφινάρουμε το παράδειγμα εφαρμόζοντας διαφορετικές οδηγίες για διαφορετικά μέρη της σειράς των item:**

```
<xsl:for-each select="item">
  <xsl:value-of select="."/>
  <xsl:if test="position()!=last()">
    <xsl:text>, </xsl:text>
  </xsl:if>
  <xsl:if test="position()=last()-1">
    <xsl:text>and </xsl:text>
  </xsl:if>
</xsl:for-each>
```

Συνθήκες με xsl:choose

- Δεν υπάρχει `xsl:else` – Αντ'αυτού, χρησιμοποιείστε `xsl:choose`, `when`, και `otherwise`
- Αυτό είναι το "switch" της XSL. Για παράδειγμα:

```
<xsl:choose>
  <xsl:when test="position()=1">
    I'm number one!
  </xsl:when>
  <xsl:otherwise>
    I'm not number one!
  </xsl:otherwise>
</xsl:choose>
```

Η XSL είναι δυνατή

- **Η XSL συγκρίνεται με μια scripting language σε δυνατότητες**
  - Η XPath είναι εκφραστική και μπορεί να χρησιμοποιηθεί για περίπλοκα pattern-matching και information extraction
  - Το μοντέλο των templates, όταν χρησιμοποιείται αναδρομικά, μπορεί να διαιρέσει πολύπλοκες διαδικασίες μετασχηματισμού σε επαναχρησιμοποιούμενα συστατικά
  - Επανάληψη, συνθήκες, και περιορισμένα υπολογιστικά χαρακτηριστικά (π.χ. "`last()-1`" στο `xsl:if` παράδειγμα) επιτρέπουν στην XSL να εκφράσει πολλούς μετασχηματισμούς

Πρακτικά: Μετασχηματίζοντας XML σε XHTML

- **Η XSL μπορεί να χρησιμοποιηθεί για να παράγει HTML αναπαραστάσεις XML δεδομένων**
- **Αυτό μπορεί να γίνει είτε**
  - **στον server (από κάποιο servlet ή άλλο HTTP server plug-in):**

```
<?php
  $xslDoc = new DOMDocument();
  $xslDoc->load("EvdErgasia.xsl");
  $xmlDoc = new DOMDocument();
  $xmlDoc->load("EvdErgasia.xml");
  $proc = new XSLTProcessor();
  $proc->importStylesheet($xslDoc);
  echo $proc->transformToXML($xmlDoc);
  ?>
```

Πρακτικά: Μετασχηματίζοντας XML σε XHTML

- **Η XSL μπορεί να χρησιμοποιηθεί για να παράγει HTML αναπαραστάσεις XML δεδομένων**
- **Αυτό μπορεί να γίνει είτε**
  - **στον client από έναν XSL-ικανό browser όπως ο Internet Explorer 6.0 *(προσοχή! IE 7's XSLT υποστήριξη είναι ανεπίκαιρη!)***
- **Τα XSL stylesheets μπορούν να συσχετιστούν με XML έγγραφα χρησιμοποιώντας την οδηγία `xml-stylesheet`**

```
<?xml-stylesheet type="text/xsl" href="file.xsl"?>
```

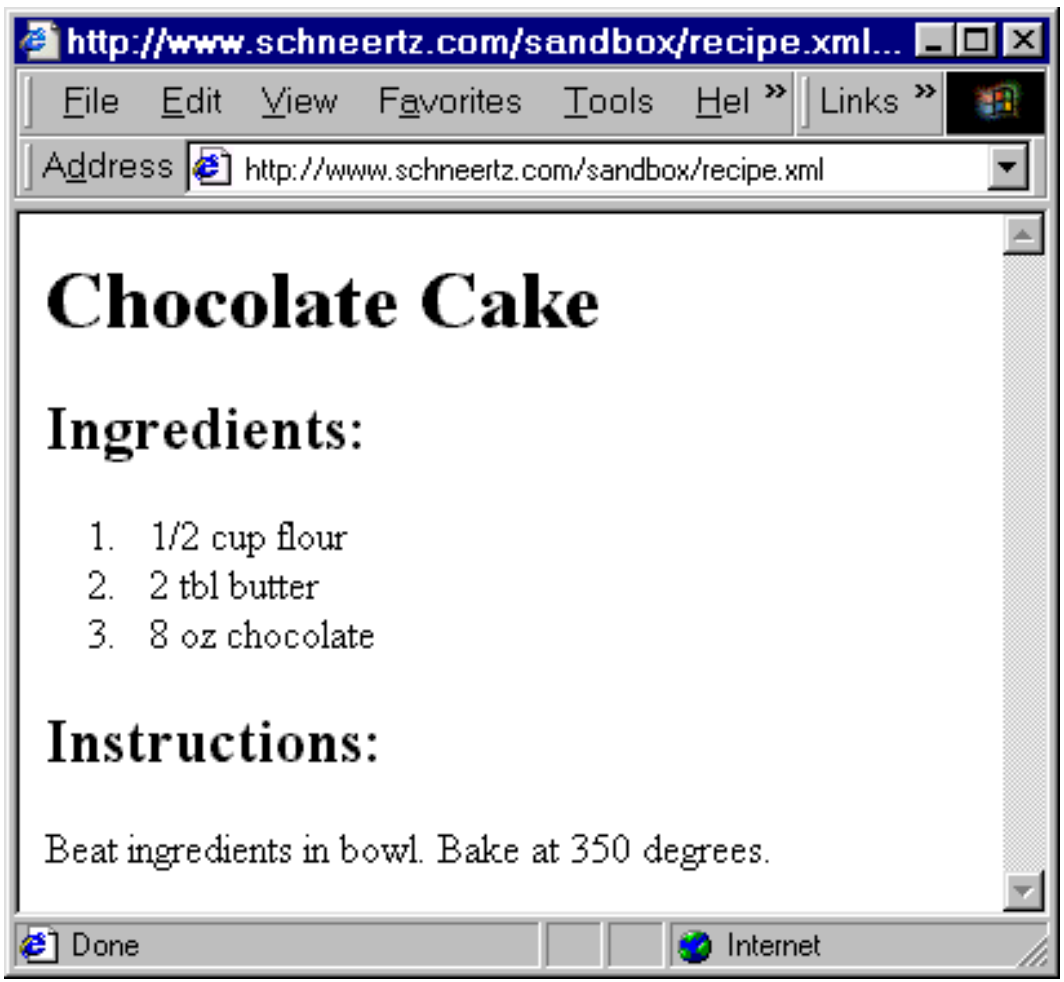
Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- **Εστω το ακόλουθο XML έγγραφο:**

```
<recipe name="Chocolate Cake">
  <ingredients>
    <item amount="1/2 cup">flour</item>
    <item amount="2 tbl">butter</item>
    <item amount="8 oz">chocolate</item>
  </ingredients>
  <instructions>Beat ingredients in
bowl. Bake at 350 degrees.
</instructions>
</recipe>
```

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- **Και έστω ότι θέλουμε να το κάνουμε να φαίνεται ως εξής:**



Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- **Αυτό απαιτεί κάτι σαν την ακόλουθη XHTML:**

```
<h1>Chocolate Cake</h1>
<h2>Ingredients</h2>
<ol>
  <li>1/2 cup flour</li>
  <li>2 tbl butter</li>
  <li>8 oz chocolate</li>
</ol>
<h2>Instructions</h2>
Beat ingredients in bowl. Bake at
350 degrees.
```

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- Γράφοντας το αντίστοιχο stylesheet: πρώτα γράφουμε το top-level template για το "recipe" element:

```
<xsl:template match="recipe">
  <h1><xsl:value-of select="@name"/></h1>
  <xsl:apply-templates/>
</xsl:template>
```

- Αυτό λέει στην XSL να εισάγει h1 ετικέτες εκατέρωθεν της τιμής του name attribute του recipe element
- Η κενή xsl:apply-templates οδηγία ισοδυναμεί με:

```
<xsl:apply-templates select="*"/>
```

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- Ας γράψουμε τώρα ένα template για να χειριστούμε το "ingredients" element:

```
<xsl:template match="ingredients">
  <h2>Ingredients</h2>
  <ol>
    <xsl:for-each select="item">
      <li>
        <xsl:value-of select="@amount">
        <xsl:text> </xsl:text>
        <xsl:value-of select="."/>
      </li>
    </xsl:for-each>
  </ol>
</xsl:template>
```

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- **Πως δουλεύει το "ingredients" template:**
  - το template παράγει HTML, συμπεριλαμβανομένου ενός heading και ενός "`ol`" start tag
  - Μέσα στο `ol` tag, ένας βρόγχος επαναλαμβάνεται στα "`item`" elements του αρχικού εγγράφου
  - Για κάθε item, το template παράγει ένα `li` tag
  - Μέσα στο `li` tag, γράφει την τιμή του "`amount`" attribute, ένα κενό, και την τιμή του "`item`" element
  - Το template γράφει το closing `ol` tag.

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

- Τελικά, θα γράψουμε ένα template για το "instructions" element:

```
<xsl:template match="instructions">
   <h2>Instructions</h2>
 <xsl:value-of select="."/>
</xsl:template>
```

- Τελειώσαμε! Το μόνο που μένει είναι να συμπεριλάβουμε τα 3 templates σε ένα xsl: stylesheet element

Πρακτικά: Μετασχηματίζοντας XML σε XHTML (συν.)

Το τελικό stylesheet:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="recipe">
    <h1><xsl:value-of select="@name"/></h1>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="ingredients">
    <h2>Ingredients</h2>
    <ol>
      <xsl:for-each select="item">
        <li>
          <xsl:value-of select="@amount"/>
          <xsl:text> </xsl:text>
          <xsl:value-of select="."/>
        </li>
      </xsl:for-each>
    </ol>
  </xsl:template>
  <xsl:template match="instructions">
    <h2>Instructions</h2>
    <xsl:value-of select="."/>
  </xsl:template>
</xsl:stylesheet>
```

Τα templates

Πρακτικά: Λίγα λόγια για Namespaces

- Το "`xmlns:xsl`" attribute στο `xsl:stylesheet` element πρέπει να περιέχει τη URL του xsl namespace.
- Υπάρχουν namespaces για διάφορες εκδόσεις της xsl. Η επίσημη και πιο πρόσφατη είναι η:

  http://www.w3.org/1999/XSL/Transform

- Οι XSL processors συμπεριφέρονται διαφορετικά για κάθε namespace!

XSL Formatting Objects

- **Η XSL αποτελείται από:**
  - **Μια γλώσσα για μετασχηματισμό XML εγγράφων (XSLT), και**
  - **Ένα λεξιλόγιο για περιγραφή του πώς παρουσιάζονται τα XML έγγραφα (XSL Formatting Objects)**
- **Τα Formatting Objects περιγράφουν ακριβώς πως θα πρέπει να φαίνεται ένα έγγραφο**
  - **Σε διαφορετικούς media types (print, browser, speech)**
  - **Σε διαφορετικές γλώσσες (υποστηρίζει right-to-left and top-to-bottom ροή κειμένου)**

XSL Formatting Objects

- Η διαδικασία επεξεργασίας XSL έχει ως εξής:
  - ένα XML έγγραφο
  - Μετασχηματίζεται με την XSLT σε ένα XML+FO έγγραφο
  - παρουσιάζεται από έναν XML+FO renderer (browser, printer, PDA, κλπ.)
- Κάποτε, browsers θα υποστηρίζουν XML+FO
- Τώρα, πειραματικοί renderers μετασχηματίζουν XML+FO σε PDF

The FO μοντέλο εγγράφου

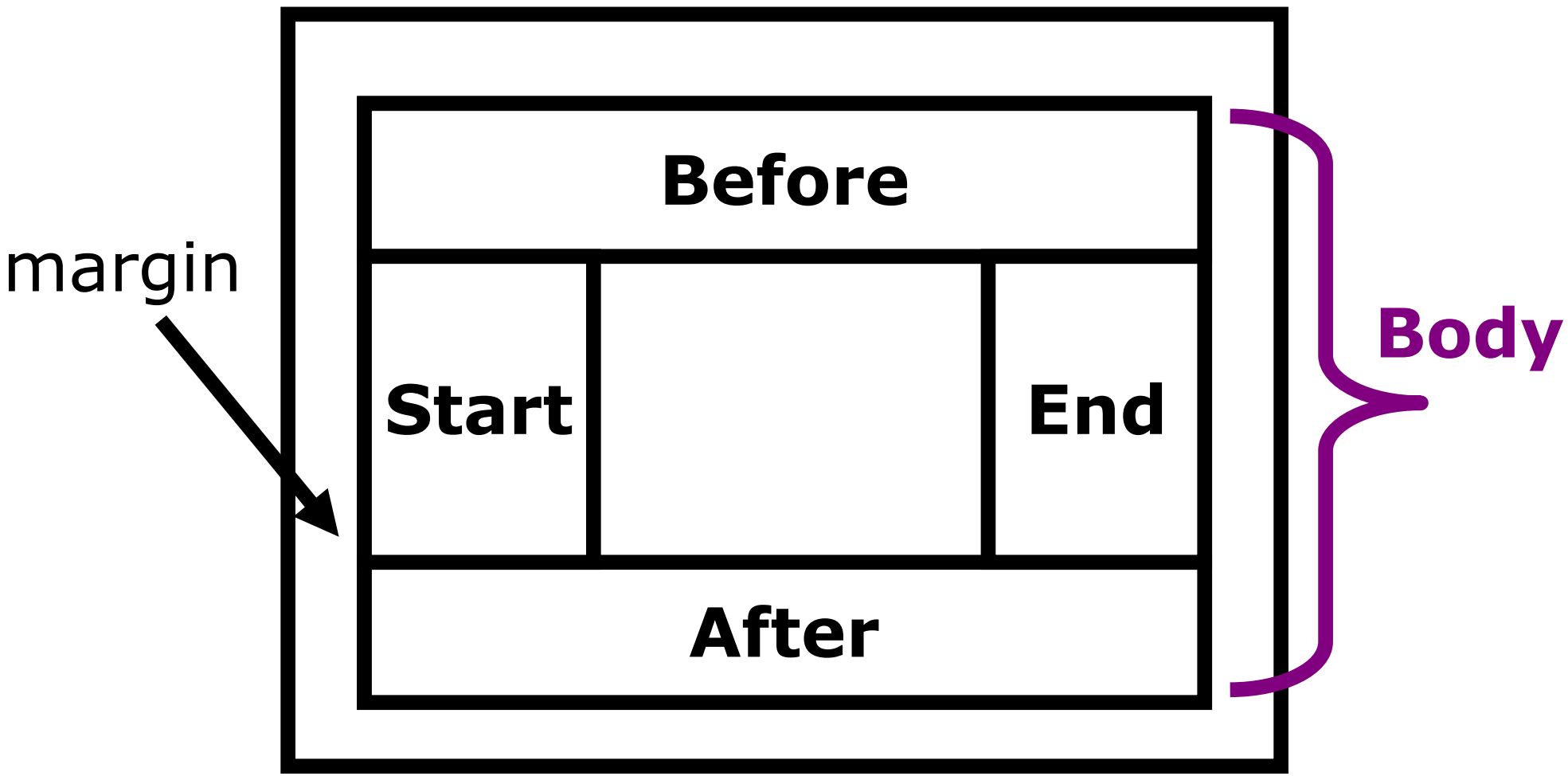
- Τα Formatting Objects περιγράφουν έγγραφα ως μια ιεραρχία από αντικείμενα που λέγονται
  - pages
  - regions
  - blocks
  - lines
  - inline areas (including characters)
- Πολλές ιδιότητες FO για κείμενο και άλλα αντικείμενα μοντελοποιούνται παρόμοια με τα CSS

Formatting Object Types

- Οι Pages έχουν πολλές ιδιότητες όπως
  - page masters
  - margins
  - "static" areas (αυτές που είναι ίδιες σε κάθε σελίδα)
  - page numbering
  - titles
- Οι Regions είναι block containers που τοποθετούνται σχετικά με τη σελίδα

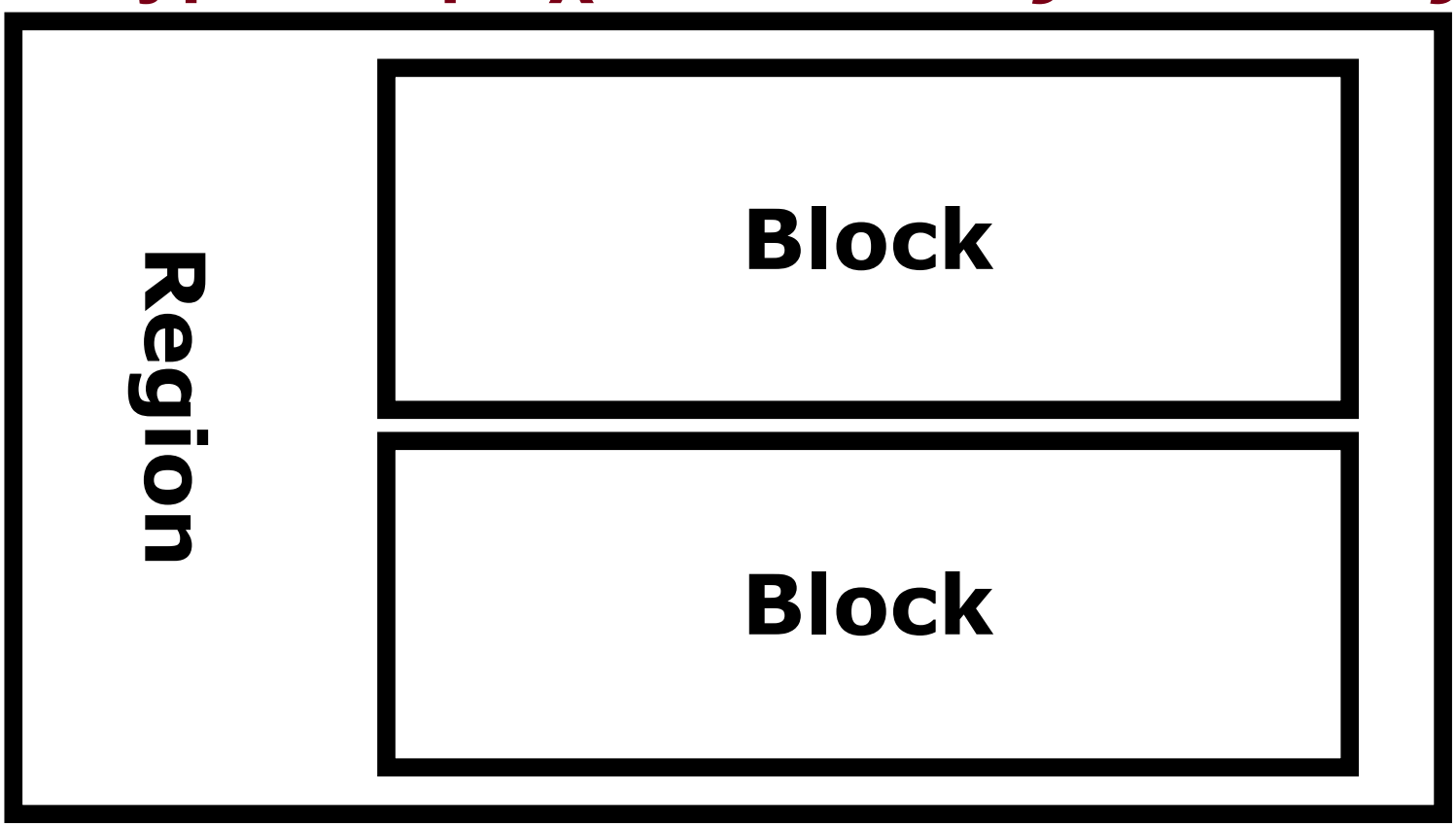
Formatting Objects: Pages και Regions

- **Η εξορισμού εμφάνιση σελίδας των XSL FO's είναι (default page layout)**
  - `fo:simple-page-master:`



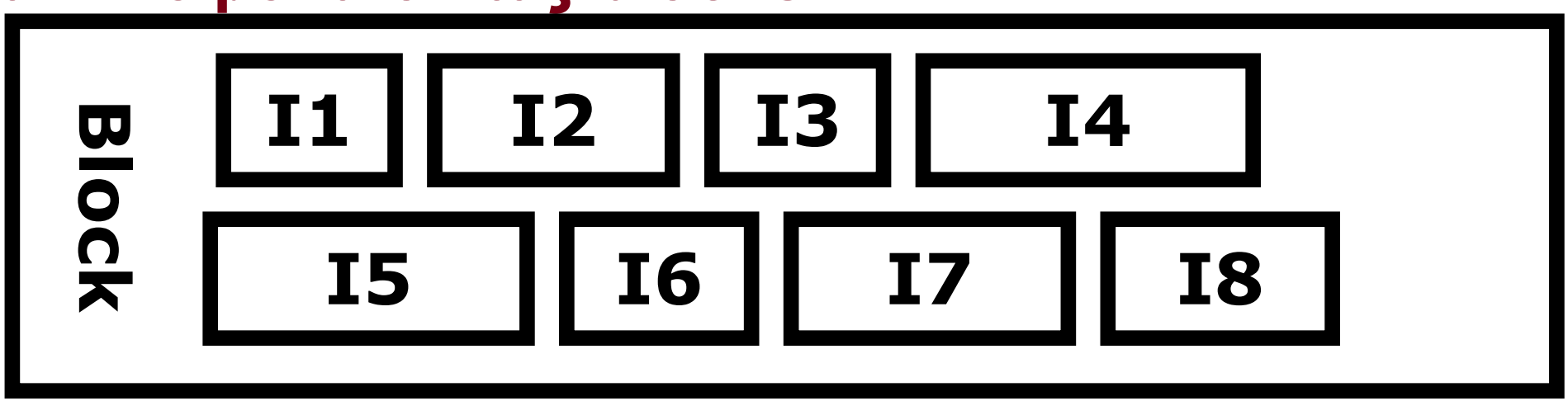
Formatting Objects: Blocks

- Τα Blocks είναι περιοχές που επιπλέουν σειριακά μέσα σε μια region, διαχωριζόμενα από αλλαγές γραμμής
- Block types περιέχουν πίνακες και λίστες



Formatting Objects: Inline

- Ροή από "start" σε "end" (left-to-right στα Αγγλικά) μέσα σε blocks και άλλα αντικείμενα, και περιτύλιξη – wrap
- Οι χαρακτήρες είναι inline αντικείμενα
- Τα Inline αντικείμενα μπορούν να χρησιμοποιηθούν ως containers για άλλα αντικείμενα όπως blocks



Formatting Objects: Out-of-line

- `fo:float` αντικείμενα «επιπλέουν» πάνω από άλλα layout elements και τα inline elements επιπλέουν γύρω τους
- `fo:footnote` και `fo:footnote-body` αντικείμενα επιτρέπουν footnotes και footnote αναφορές να τοποθετηθούν κατάλληλα
  - Η footnote αρίθμηση ή τα endnotes πρέπει να γίνουν στη φάση του XSLT

FO Properties

- υπάρχουν (200+!) formatting object properties που ελέγχουν
  - font-family, font, size, color, spacing, kerning
  - paragraph spacing, line spacing, indentation, page break rules, alignment, justification
  - hyphenation, capitalization
  - backgrounds, borders, padding, margins
  - overflow
  - speech properties
- Περικλείουν σε μεγάλο βαθμό τα CSS2

FO: Ενα απλό παράδειγμα

```
<fo:root xmlns:fo="http://www.w3.
org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="only">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-name="only">
    <fo:flow>
      <fo:block font-size="12pt">
        Hello, world!
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

FO: Καθεστώς υλοποίησης

- Δεν υπάρχει ολκληρωμένη αναφορά σε XSL Formatting Objects
- Τα "FOP" του Apache, που μετατρέπει XML+FO σε PDF, είναι ότι καλύτερο υπάρχει. Βλέπε:

http://xml.apache.org/fop/

- Κάποια στιγμή, browser και printer-based υλοποιήσεις θα είναι διαθέσιμες

Εργαλεία για υλοποίηση σε XSL

- XSLT processors:
  - MSXML, which currently supports the latest XSLT specification (native Win32)
  - Xalan from Apache (C++, Java)
  - James Clark's XT (Java)
- Editors and browsers
  - Internet Explorer 7.0
  - IBM Alphaworks XSL Editor
  - XML Spy (commercial)

Περαιτέρω διάβασμα

- **Στο WWW Consortium:**
  - **Η βασική XSL σελίδα:**
    
    http://www.w3.org/Style/XSL
  - **Οι XSLT προδιαγραφές:**
    
    http://www.w3.org/TR/xslt
  - **Οι XPath προδιαγραφές :**
    
    http://www.w3.org/TR/xpath
  - **Οι XSL προδιαγραφές, συμπεριλαμβανομένου και formatting objects:**
    http://www.w3.org/TR/xsl
    

Περαιτέρω διάβασμα:

- **Ενα καλό XSLT tutorial:**
  - **http://www.ibiblio.org /xml/books/bible/updates/14.html**
- **Ενα portal για πληροφορίες XSL:**
  - **http://www.xslt.com/**
- **Τα παραδείγματα της παρουσίασης:**
  - **http://www.ncsa.uiuc.edu /People/futrelle/ntuxsl**