UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

eXtensible Stylesheet Language

COMP3220 Web Infrastructure

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

The Problem

How do you maintain different versions of a document for presentation on different systems that:

- Have different presentation characteristics (screen size, etc)?
- Require different document formats?

XSL is a family of XML-based technologies designed to address this problem

eXtensible Stylesheet Language

XSL Transformations (XSLT)

- XML-based language for describing transformations from one XML-based language to another

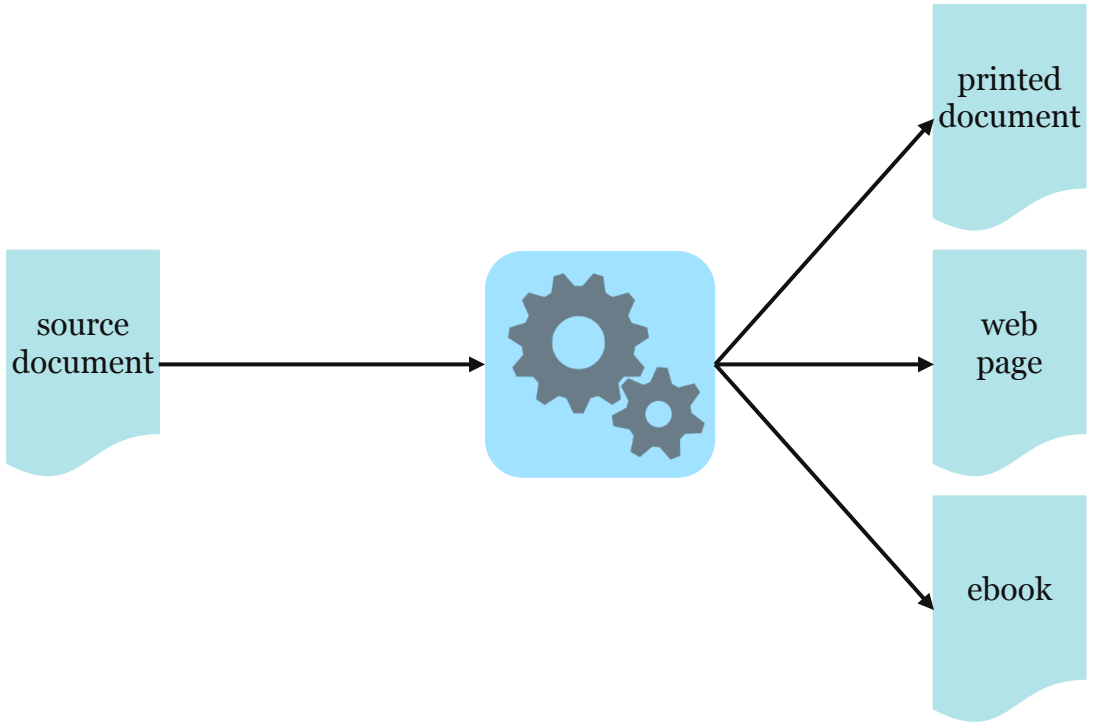
XML Path Language (XPath)

- Language for referring to parts of an XML document

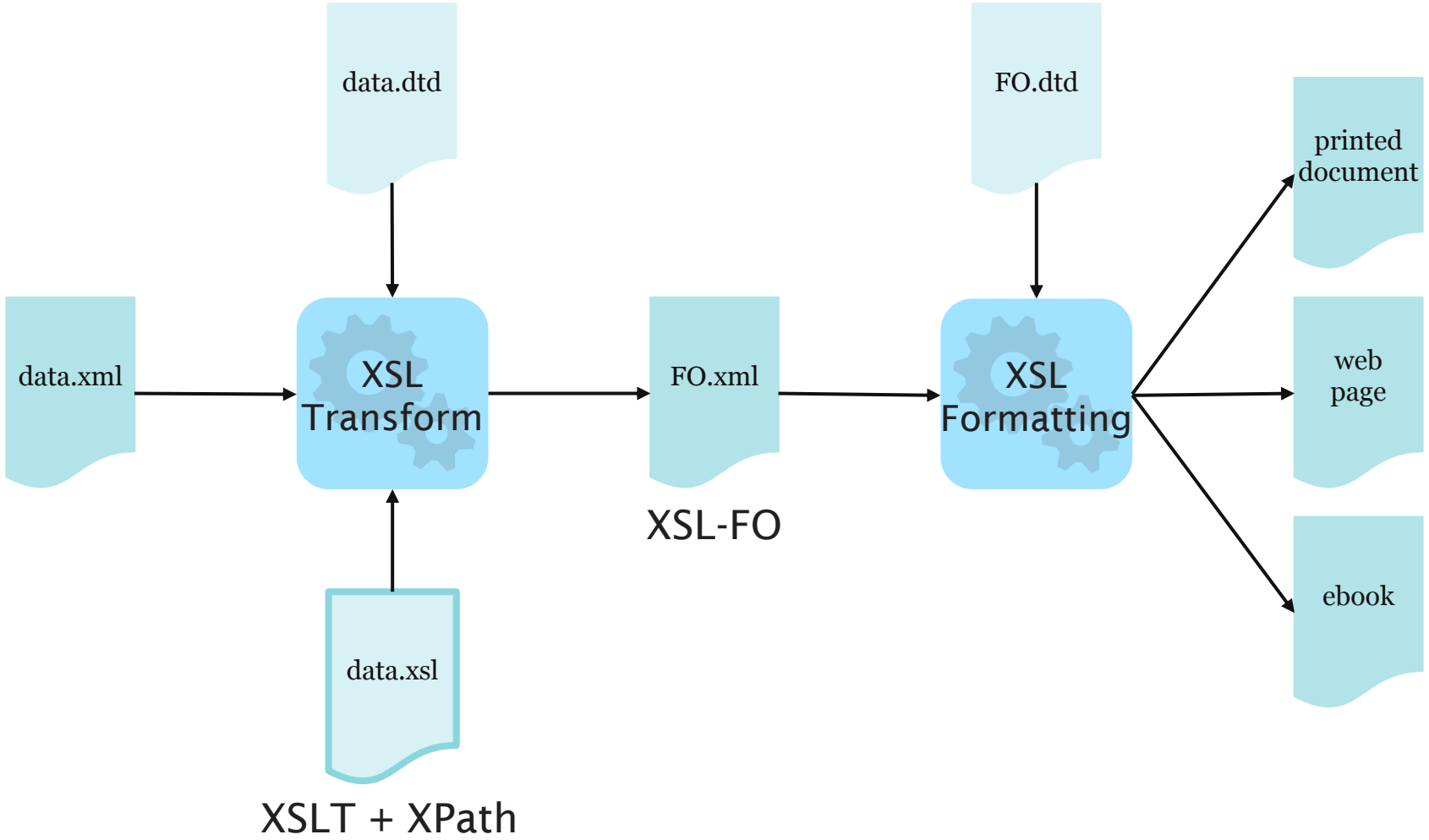
XSL Formatting Objects (XSL-FO)

- XML vocabulary for specifying formatting semantics

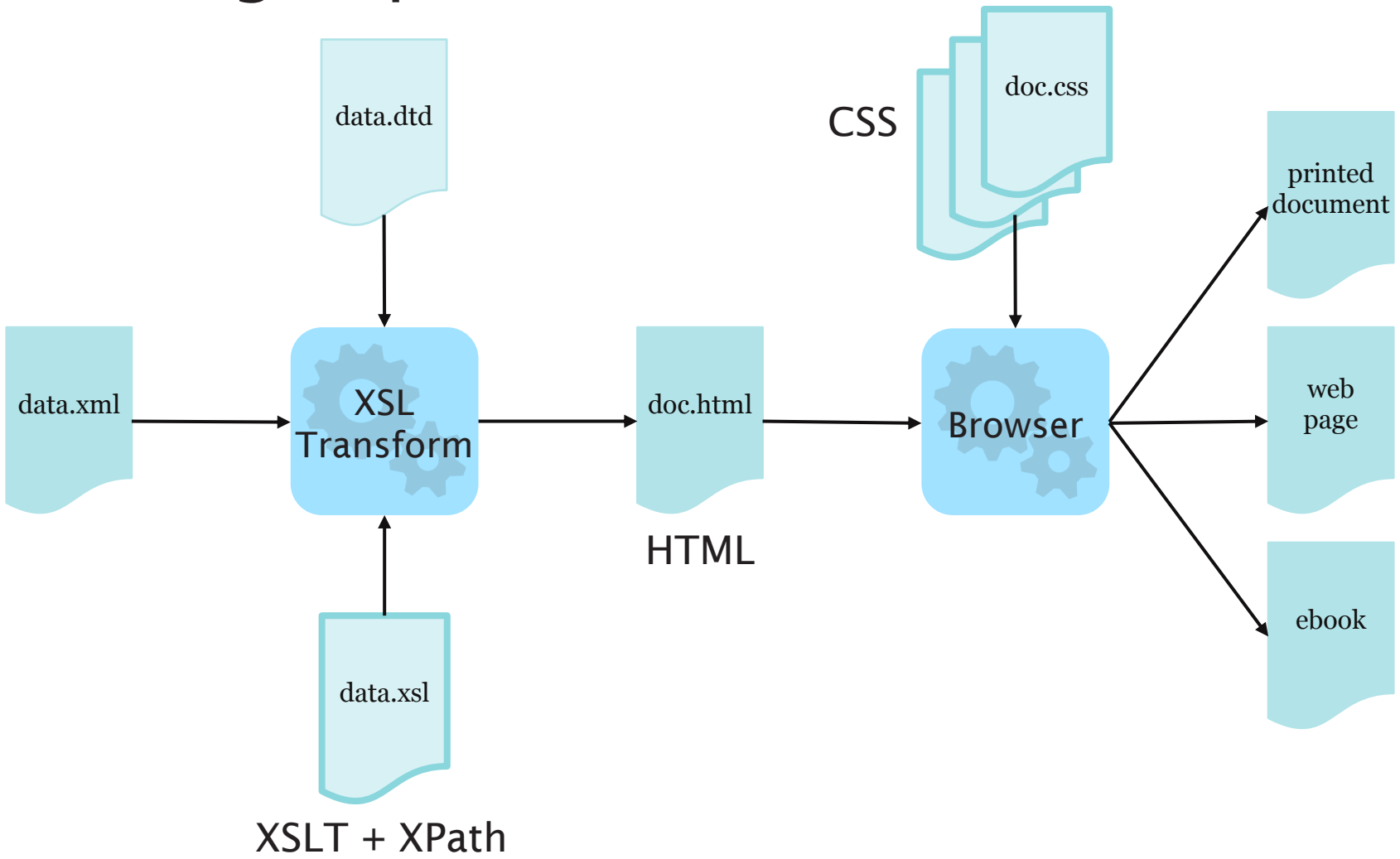
XML processing



XSL processing in theory



XSL processing in practice



Example input

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
  <name>My address list</name>
  <card>
    <name>Fred Foo</name>
    <email>fred@example.org</email>
    <tel type="work">01234567890</tel>
  </card>
  <card>
    <name>Jill Bar</name>
    <email>jill@example.org</email>
    <tel type="home">02345678901</tel>
  </card>
</contacts>
```

Example output

```
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>My address list</title>
  </head>
  <body>
    <h1>My address list</h1>
    <div>
      <h2>Fred Foo</h2>
      <p>Email:
        <a href="mailto:fred@example.org">fred@example.org</a></p>
      <p>Telephone: 01234567890</p>
    </div>
    <div>
      <h2>Jill Bar</h2>
       <p>Email:
         <a href="mailto:jill@example.org">jill@example.org</a></p>
       <p>Telephone: 02345678901</p></div>
  </body>
</html>
```

XSL stylesheets

An XSL stylesheet consists of a number of templates

Each template:

- Matches an element in the original document using XPath
- Specifies the new content with which the element is to be replaced

XPath

Expression language for specifying nodes (elements, attributes) within an XML document

Specifies nodes using a path through the hierarchy of the document

- Can be absolute (from the root) or relative (from current position)

XPath expressions

`E`	Selects all nodes with the name E
`/`	Selects from the root node
`//`	Selects nodes anywhere under the current node
`.`	Selects the current node
`..`	Selects the parent of the current node
`@`	Selects attributes

XPath expression examples

`contacts`	Selects all `contacts` elements
`/contacts`	Selects the root `contacts` element
`contacts/card`	Selects all `card` elements that are children of `contacts`
`//card`	Selects all `card` elements
`contacts//name`	Selects all `name` elements that are descendants of `contacts`
`//tel/@lang`	Selects `type` attribute on all `tel` elements

xsl:template

match attribute contains XPath expression that identifies an element or elements

- Content of element is either output markup, or other XSL directives

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

xsl:apply-templates

Used within body of a template

- Recursively applies templates to children of the element
- `select` attribute identifies target child node using XPath

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

xsl:for-each

Loops over every matching element

- `select` attribute identifies target node set

```
<xsl:for-each select=”...">

...

</xsl:for-each>
```

xsl:value-of

Extracts the value of an XML element and uses it in the output document

- `select` attribute used to identify element

```
<xsl:value-of select=”..."/>
```

XSLT example – input

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
  <name>My address list</name>
  <card>
    <name>Fred Foo</name>
    <email>fred@example.org</email>
    <tel type="work">01234567890</tel>
  </card>
  <card>
    <name>Jill Bar</name>
    <email>jill@example.org</email>
    <tel type="home">02345678901</tel>
  </card>
</contacts>
```

XSLT example – output

```
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>My address list</title>
  </head>
  <body>
    <h1>My address list</h1>
    <div>
      <h2>Fred Foo</h2>
      <p>Email:
        <a href="mailto:fred@example.org">fred@example.org</a></p>
      <p>Telephone: 01234567890</p>
    </div>
    <div>
      <h2>Jill Bar</h2>
       <p>Email:
         <a href="mailto:jill@example.org">jill@example.org</a></p>
       <p>Telephone: 02345678901</p></div>
  </body>
</html>
```

XSLT example – using stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="contacts.xsl" version="1.0"?>
<contacts>
  <name>My address list</name>
  <card>
    <name>Fred Foo</name>
    <email>fred@example.org</email>
    <tel>01234567890</tel>
  </card>
  <card>
    <name>Jill Bar</name>
    <email>jill@example.org</email>
    <tel>02345678901</tel>
  </card>
</contacts>
```

XSLT example – stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/TR/xhtml1/strict">

...

</xsl:stylesheet>
```

XSLT example – templates

```
<xsl:template match="tel">
  <p>Telephone: <xsl:value-of select="."/></p>
</xsl:template>
```

XSLT example – templates

```
<xsl:template match="email">
  <p>Email: <a href="mailto:{.}"><xsl:value-of select="."/></a></p>
</xsl:template>
```

XSLT example – templates

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

XSLT example – templates

```
<xsl:template match="/">
  <html>
    <head>
      <title><xsl:value-of select="/contacts/name"/></title>
    </head>
    <body>
      <h1><xsl:value-of select="/contacts/name"/></h1>
      <xsl:for-each select="/contacts/card">
        <div>
          <xsl:apply-templates select="name"/>
          <xsl:apply-templates select="email"/>
          <xsl:apply-templates select="tel"/>
        </div>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
```

XSL versus CSS

Considering CSS...

Pros

- Simple (relatively)
- Cascading recognises different needs of users and authors

Cons

- Unable to modify document structure

XSL versus CSS

Considering XSL:

Pros

- Able to modify and transform document structure
- Better for data

Cons

- Complex (relatively)
- Cumbersome for ordinary documents
- No consideration of differing needs of users versus authors

UNIVERSITY OF Southampton

Next Lecture: Optimising HTTP