UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

Open Hypermedia and the Web

COMP3220 Web Infrastructure

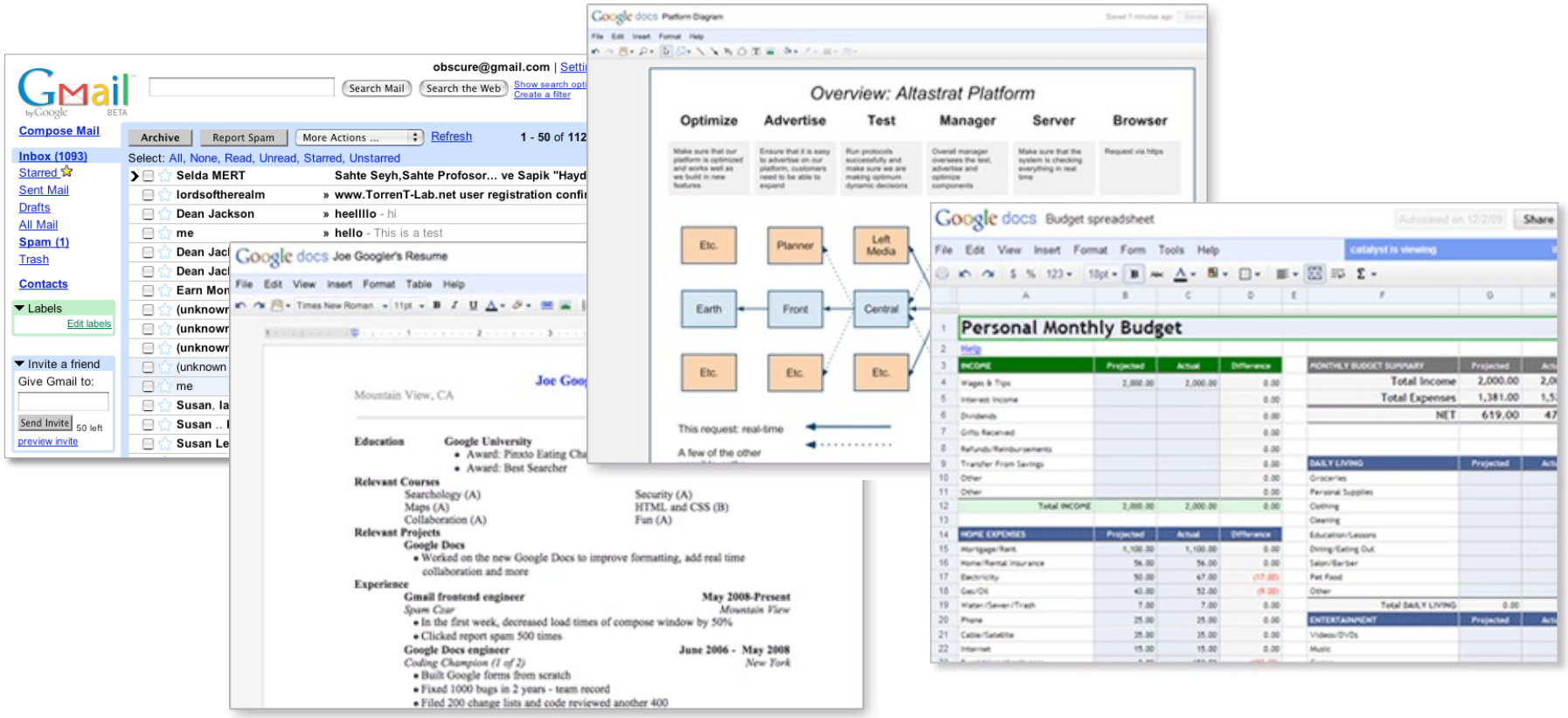
Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

There is no runner-up prize

If Microcosm and Hyper-G were so good,
why are we not all using them?

Open Hypertext on the Web

We lost the goal of hypertext across the entire desktop – is this still important?



Open Hypertext on the Web

HTML and DOM by themselves do not allow easy control of anchor positions

Still possible to separate links and annotations

Linkbase creation

Manual creation of links is possible (also desirable?), but time-consuming

Automatic creation of links based on natural language information extraction

1. Extract key phrases/themes from documents based on location and frequency
2. Create generic links to those documents based on the extracted themes
3. Use interactive linkbase editor to delete unwanted links

Effectively turning a search engine index into a linkbase

Link injection

How do we add links into representations?

Where/when do we add links into representations?

batch process documents

in origin server

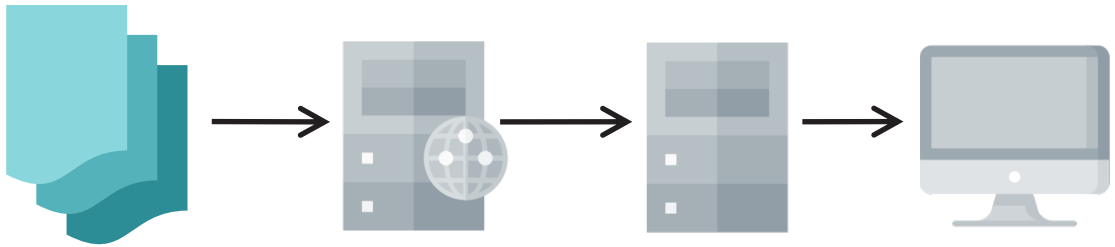
in proxy server

in user agent

Link injection: batch processing

Generate static HTML pages for use with any web server

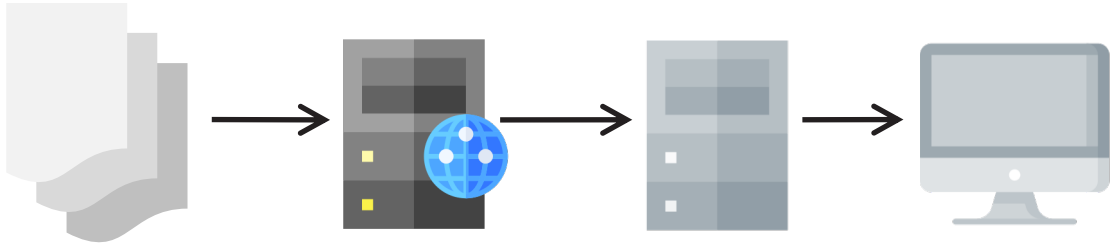
- Fastest way to serve pages - and most cache friendly approach
- Need to reprocess pages in the event of a linkbase update
- No opportunity to personalise content



Link injection: on demand in the origin server

Careful choice of data structures and algorithms needed to ensure good performance

- New material can immediately be viewed with hyperlinks from the linkbase
- User can choose linkbases to apply (stateful interaction)
- Higher run-time memory requirements, search capability can use same data

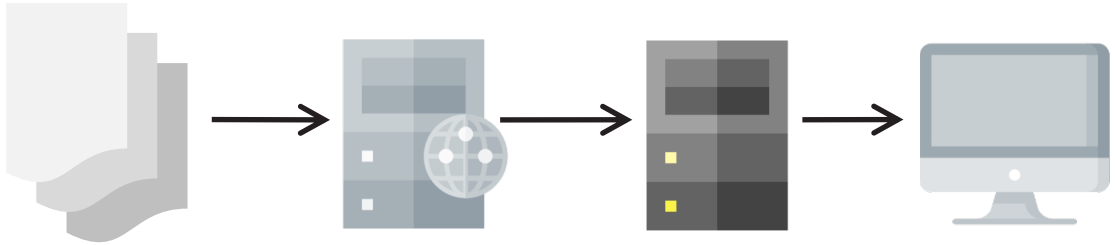


Approach taken by Active Navigation (commercial successor to Microcosm)

Link injection: using a proxy server

User configures their browser to proxy through the link injector

- Can apply link database to almost any content
- Can also perform other conversions e.g. Word -> HTML
- Casual site users won't want to reconfigure their browsers



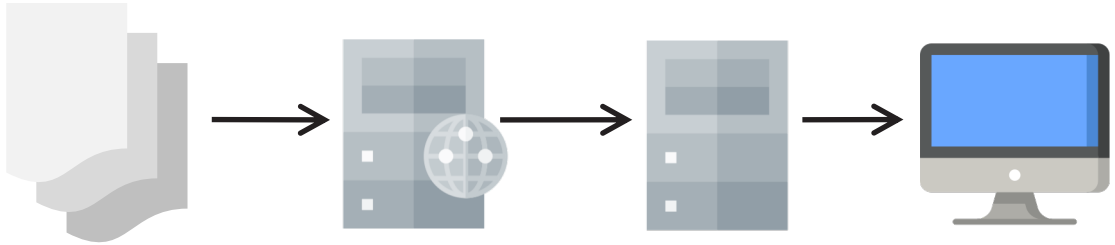
Approach taken by the Dynamic Link Service (DLS)

Link injection: in the user agent

Extend browser with a plugin that fetches links and rewrites documents

- Most modern browsers support add-on functionality
- Potentially expensive – need to provide plugins for many browsers

Approach taken by COHSE/Magpie



Link Presentation

Distinguish third-party (i.e. linkbase) links from authored (i.e. embedded) links

- Use different colours (etc)
- Consistent presentation across representations with variable styling
- Presentation of n-ary links and overlapping anchors: pop-up menus, interstitial pages

History of unsuccessful attempts ☹

- Microsoft Smart Tags in Office XP
- Third-party embedded advertising links (Infolinks, IntelliTXT, Vibrant)

UNIVERSITY OF Southampton

Open hypermedia on the Web

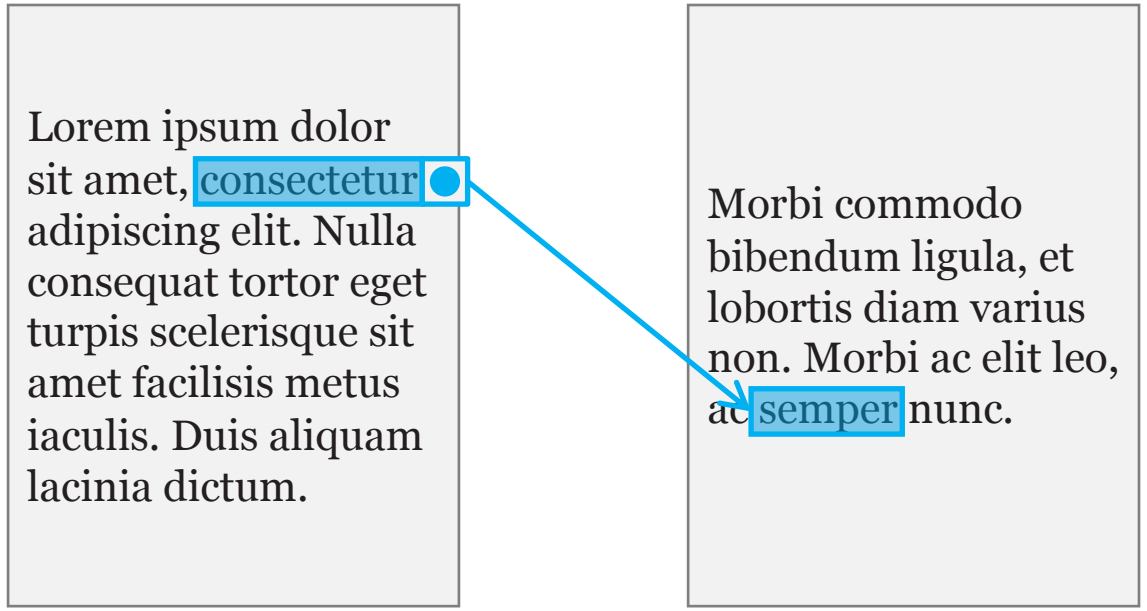
Limitations of HTML linking

Addressing

- URIs for identifying whole resources
- Elements with `name` or `id` attributes for identifying resource fragments

Closed tag set (`<a>`, `<img>`, etc)

- Fixed link traversal semantics
- Single-ended, unidirectional, embedded
- No stylesheet (or similar) to express link semantics



Can we do better using Web standards?

XPath

XPath provides path expressions for addressing nodes within an XML parse tree

We've used it so far to apply templates in XSLT

We could use it instead to specify the anchor of a link endpoint (i.e. as a locspec)

Robie, J. et al (2017) *XML Path Language (XPath) 3.1*. W3C Recommendation. Available at: https://www.w3.org/TR/xpath-31/

XPath abbreviated path expressions

`*`	Selects all children of the context node
`E`	Selects all `E` element children of the context node
`E[@att=val]`	Selects all `E` element children whose `att` attribute has a value of `val`
`E[n]`	Selects the nth `E` element child of the context node
`text()`	Selects the text node children of the context node
`id("x")`	Selects the element with id of x (i.e. the `id` attribute in HTML)
`/`	Selects from the root node
`//`	Selects nodes anywhere under the context node
`.`	Selects the context node
`..`	Selects the parent of the context node

XPath path expression examples

```
//main/p[1]
```

Selects the first paragraph in the main element

```
//p[@class="warning"]
```

Selects all paragraphs whose class attribute has the value “warning”

```
//footer/h2[text()="References"]
```

Selects all h2 elements in footers whose text is “References”

```
//a[starts-with(@href,"/"]/text()
```

Selects the text content of all a elements whose href attribute starts with “/”

```
//id("chapter1")
```

Selects the element with id "chapter1"

XPath and referential Integrity

XPath allows us to address arbitrary elements in an XML parse tree

- Resulting addresses may not be robust when the resource representation changes

Consider the difference between the following:

```
//id("chapter1")
//main/section[1]/p[3]/span
```

After editing, the path expression using `id()` is still likely to work

The second path expression may not point to the same part of the document

XPointer

XPath has some limitations:

- Can't identify arbitrary ranges within a document, only elements
- Doesn't identify which document the path is to be applied to

The XPointer framework allows XPath path expressions to be used as fragment identifiers in URIs

- `http://example.org/#xpointer(//main/section[1]/p[3]/span)`

Intended to be used with format-specific schemes

- Default scheme – xpointer() – is designed for use with XML-based languages

Grosso, P. et al (2003) *XPointer Framework*. W3C Recommendation. Available at: https://www.w3.org/TR/xptr-framework/

XPointer ranges

The xpointer() scheme extends XPath to include ranges in XML documents:

```
xpointer(string-range(//p,"squeamish ossifrage"))
```

- Indicates the string "squeamish ossifrage" in any p element
- Generic links!

```
xpointer(id("chapter1")/range-to(id("section1.1")))
```

- Indicates the portion of the document lying between the elements with ids "chapter1" and "section 1.1"

XLink

Generic framework for representing links in XML documents

- Used to define XML formats for linking – element names are arbitrary

Distinguishes between *simple links* and *extended links*

- Simple link: a link which associates exactly two resources, one of which is local (equivalent to an embedded HTML link using the <a> element)
- Extended link: a link which associates an arbitrary number of resources (i.e. a first-class link)

Defines a set of XML attributes which may be added to markup:

- `xlink:type`, `xlink:href`, `xlink:role`, `xlink:arcrole`, `xlink:title`, `xlink:label`, `xlink:show`, `xlink:actuate`

DeRose, S. (2010) *XML Linking Language (XLink) Version 1.1*. W3C Recommendation. Available at: https://www.w3.org/TR/xlink11/

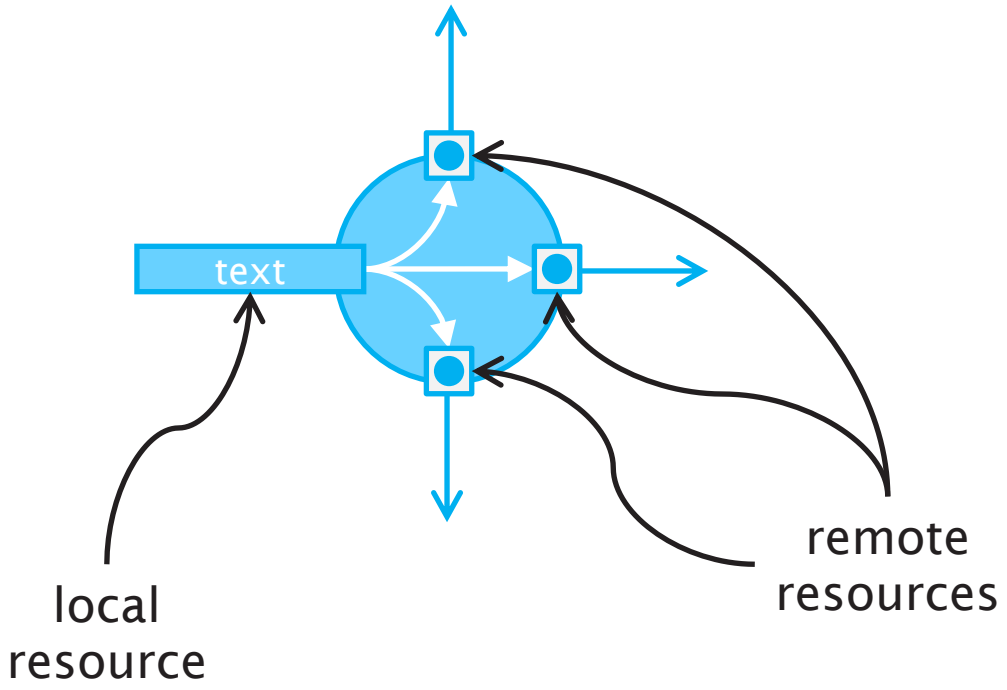
Anatomy of an XLink link

Links in Xlink are slightly different to other first-class links

Links are collections of *resources* (resource ≃ endpoint)

- Local resources
  (i.e. textual anchor for an embedded link)
- Remote resources specified by a *locator*
  (i.e. URI including optional fragment)

Links contain *arcs* which specify pairs of endpoints which may be traversed

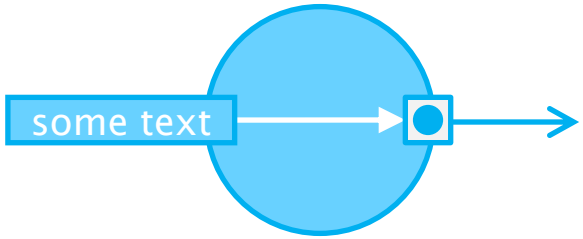


XLink simple links

```
<link xlink:href="http://example.org/">some text</link>
```

Note that `link` element name is not part of XLink.

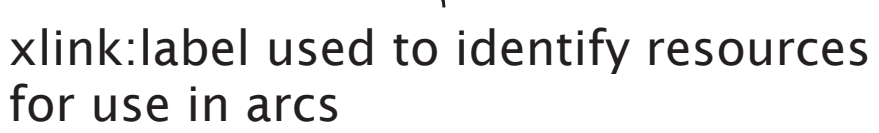
`xlink:type` attribute is optional on simple links, arc is implicit.



XLink extended links

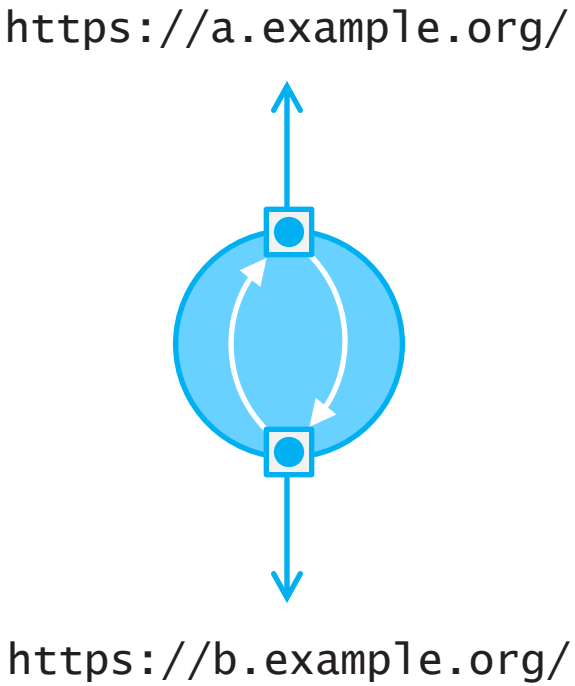
Consider an extended link version of the simple link on the previous slide:

```
<link xlink:type="extended">
  <anchor xlink:type="resource"
          xlink:label="source">some text</resource>
  <location xlink:type="locator"
            xlink:label="destination"
            xlink:href="http://example.org/"/>
  <transition xlink:type="arc"
              xlink:from="source"
              xlink:to="destination">
</link>
```



XLink bidirectional first-class link

```
<link xlink:type="extended">
  <location xlink:type="locator"
     xlink:label="end1"
     xlink:href="http://a.example.org/"/>
  <location xlink:type="locator"
     xlink:label="end2"
     xlink:href="http://b.example.org/"/>
  <transition xlink:type="arc"
     xlink:from="end1"
     xlink:to="end2">
  <transition xlink:type="arc"
     xlink:from="end2"
     xlink:to="end1">
</link>
```



XLink behaviours

Two orthogonal behaviours of links:

- Is a link traversed automatically, or must it be a deliberate act by the user?

`xlink:actuate="onLoad"`

- Automatically actuate the link when the page loads
- Behaves like the HTML `img` element

`xlink:actuate="onRequest"`

- Only actuate the link when the user requests
- Behaves like the HTML a element

XLink behaviours

Two orthogonal behaviours of links:

- Is a link traversed automatically, or must it be a deliberate act by the user?
- In what context is the link destination displayed?

`xlink:show="replace"`

- Display the destination in the current context (default behaviour of the HTML a element)

`xlink:show="new"`

- When traversing the link, display the destination in a new context (i.e. window, browser tab)
- Behaves like the HTML a element with `target="_blank"`

`xlink:show="embed"`

- Display the destination by embedding it in the current context
- Behaves like the HTML `img` element

XLink roles and titles

More general version of `rel/rev` attributes in HTML

- `xlink:role` used to express link type (i.e. meaning of a link as a whole)
- `xlink:arcrole` used to express the meaning of a specific arc within a link
- Both take URIs as values (compare with simple keywords used as link types in HTML)

`xlink:title` attribute used to provide human-readable labels

- Used for extended links, arcs and locators
- Can also use an element with `xlink:type="title"` to specify labels

Defining XLink vocabularies

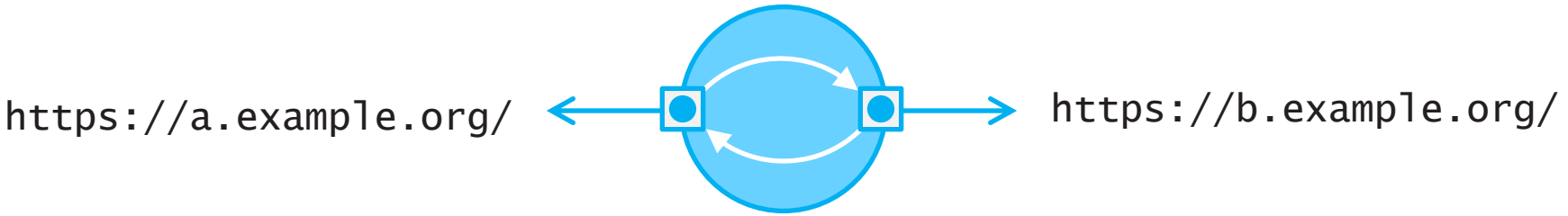
Use DTDs/XML Schema to simplify use of XLink:

```
<!ELEMENT link ((location|anchor|transition)*)>
<!ATTLIST link xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
               xlink:type (extended) #FIXED "extended">

<!ELEMENT location EMPTY>
<!ATTLIST location xlink:type (locator) #FIXED "locator"
                   xlink:href CDATA #REQUIRED
                   xlink:label NMTOKEN #REQUIRED>

<!ELEMENT anchor ANY>
<!ATTLIST anchor xlink:type (resource) #FIXED "resource"
                 xlink:label NMTOKEN #REQUIRED>

<!ELEMENT transition EMPTY>
<!ATTLIST transition xlink:type (arc) #FIXED "arc"
                     xlink:from NMTOKEN #REQUIRED
                     xlink:to NMTOKEN #REQUIRED>
```

Using XLink vocabularies

Consider the bidirectional link from earlier in this lecture:

```
<link>
  <location xlink:label="end1" xlink:href="http://a.example.org/"/>
  <location xlink:label="end2" xlink:href="http://b.example.org/"/>
  <transition xlink:from="end1" xlink:to="end2">
  <transition xlink:from="end2" xlink:to="end1">
</link>
```



Further reading

DeRose, S. et al (2010) XML Linking Language (XLink) Version 1.1. W3C Recommendation

https://www.w3.org/TR/xlink11/

Grosso, P. et al (2003) XPointer Framework. W3C Working Draft.

https://www.w3.org/TR/xptr-framework/

Robie, J. et al (2017) XML Path Language (XPath) 3.1. W3C Recommendation

https://www.w3.org/TR/xpath-31/

UNIVERSITY OF Southampton

Next Lecture: Richer Links