Navigating the Tree

Julia Flanders and Syd Bauman
Identifying the element to match: simple identification

We've already seen examples of the simple case where we name an element in the value of @match, and thereby we identify an element to be processed; here we say "please match the TEI element"

We're not talking here about the flow of the stylesheet: we're just saying "when my turn comes, if there's a XX, I match it..."
Identifying the element to match: multiples

Here's an example where naming an element actually identifies multiple instances of that element: in this case, all the `<div>` elements in the document.

We can also specify multiple different elements that we want to target; the or-bar means "this or that", "head or opener"
Identifying the element to match: context

And we can also specify particular contexts we're interested in: for instance, only the `<div>` elements that appear within `<front>`
Some other useful patterns:
attribute values, order of siblings

We can also select contexts based on particular attribute values (similar to CSS selectors): basically saying "I'm looking at all the divs, and of those I only want the ones that have this attribute, and this value"

Also based on which sibling we are talking about: e.g. we only want the first `<p>` child in a given context.
How to suppress parts of the input tree

We can also suppress whole branches of the input tree, basically telling the XSLT processor to ignore them.

We do this by providing a template that matches the parent element for that branch, but with no "apply templates" instruction.

```xml
<xsl:template match="teiHeader"></xsl:template>
```
More complex navigation: the context node

We've been doing some simple navigation so far, basically ignoring the question of where exactly we are in the tree.

However, to do more complex navigation and selection of nodes for processing, we have to have a more complex understanding of how we navigate, which involves knowing where the XSLT processor thinks we are (and hence, how to get to where we want to go from there).

In an XSLT template, when we match an element, whatever else happens inside that template happens relative to the location of that element (i.e. the matched element in the input tree); that element is "the context node".

```xml
<xsl:template match="div">
    <xsl:apply-templates select="p"/>
</xsl:template>
```
So for instance in this example, our template is selecting `<div>` as the context node, and as a result the `<p>` elements identified by the `@select` attribute are limited to those within the context node: the `<p>` children of the context node.

**Navigating from the root of the tree**

```xml
<xsl:template match="teiHeader">
  <xsl:apply-templates select="/TEI/text/front/div"/>
</xsl:template>
```

If we want to navigate from the root, rather than relative to our current position, start the expression with a slash.
Navigating the Tree

Parents, ancestors, descendants, and children

So far we've just been navigating down the tree in a very simple way: all of our matches and selections thus far have been done using just the element name. It's time to gloss this notation in a bit more detail: an element name X (used on the @select attribute, all by itself, means "the X children of the context node", "my X children"

If we want the descendants, not just the children, there's another notation: //

And if we want parents, there's another notation ../
And if we want ancestors, there's another (more verbose) notation

Following and preceding nodes

The idea of parents and ancestors, children and descendants, takes us up and down the tree; we can think of these relationships as traversing axes that radiate out from the context node. We can also go across the tree along horizontal axes: to identify preceding and following nodes.

A simple case: the following sibling...
Following and preceding nodes, more

The more general case of "following" nodes are a little trickier because you have to remember that an element's child doesn't "follow" it. To find "following" elements you have to go up one level, to the parent, as in this example: the "following" `<div>` elements are those that are children of any following siblings of the context node.
Navigating the Tree

Axes

For reference, a cribsheet for XPath

Let's take a step back and look at the axes themselves
For reference, a cribsheet for XPath