Markup Languages

Lecture 3. XSL

XSL

- XSL stands for E**X**tensible **S**tylesheet **L**anguage.
- The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

CSS = Style Sheets for HTML

- HTML uses predefined tags, and the meaning of each tag is **well understood**.
- The <table> tag in HTML defines a table - and a browser knows **how to display it**.
- Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy with CSS.

XSL = Style Sheets for XML

- XML does not use predefined tags (we can use any tag-names we like), and therefore the meaning of each tag is **not well understood**.
- A <table> tag could mean an HTML table, a piece of furniture, or something else - and a browser **does not know how to display it**.
- XSL describes how the XML document should be displayed!

XSL - More Than a Style Sheet Language

- XSL consists of three parts:
  - XSLT - a language for transforming XML documents
  - XPath - a language for navigating in XML documents
  - XSL-FO - a language for formatting XML documents

XSLT Introduction

- **XSLT** is a language for transforming XML documents into HTML documents or to other XML documents.
- Before you continue you should have a basic understanding of the following:
  - HTML
  - XML
  - XPath

XPath

- **XPath** is used to navigate through elements and attributes in an XML document.
- XPath is a major element in W3C's XSLT standard.

XPath Expressions

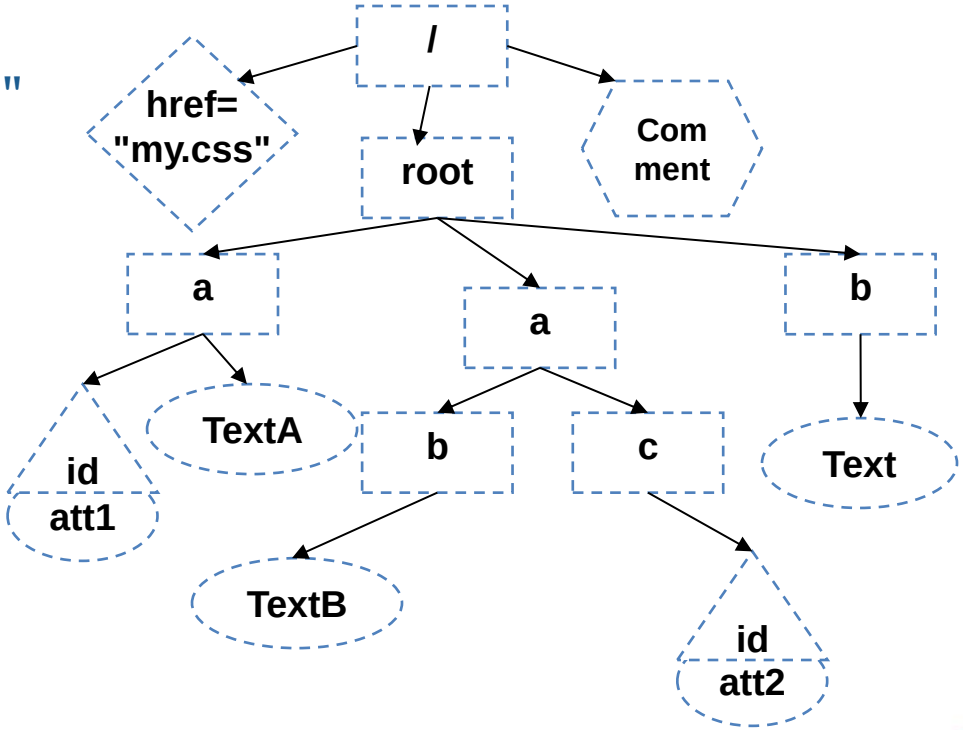
- XPath uses path expressions **to select nodes or node-sets** in an XML document.
- These path expressions look very much like the expressions you see when you work with a traditional computer file system.

XPath Terminology

- In XPath, there are seven kinds of nodes:
  1. element,
  2. attribute,
  3. text,
  4. namespace,
  5. processing-instruction,
  6. comment,
  7. document nodes

Document Tree

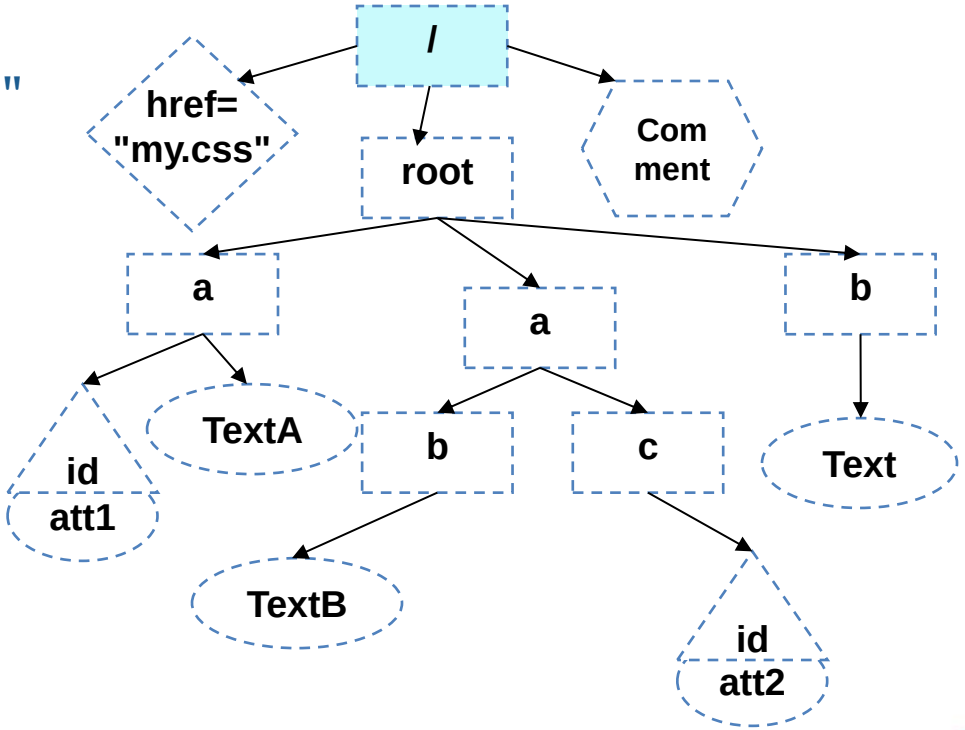
```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```



XML documents are treated as trees of nodes.

Document Nodes

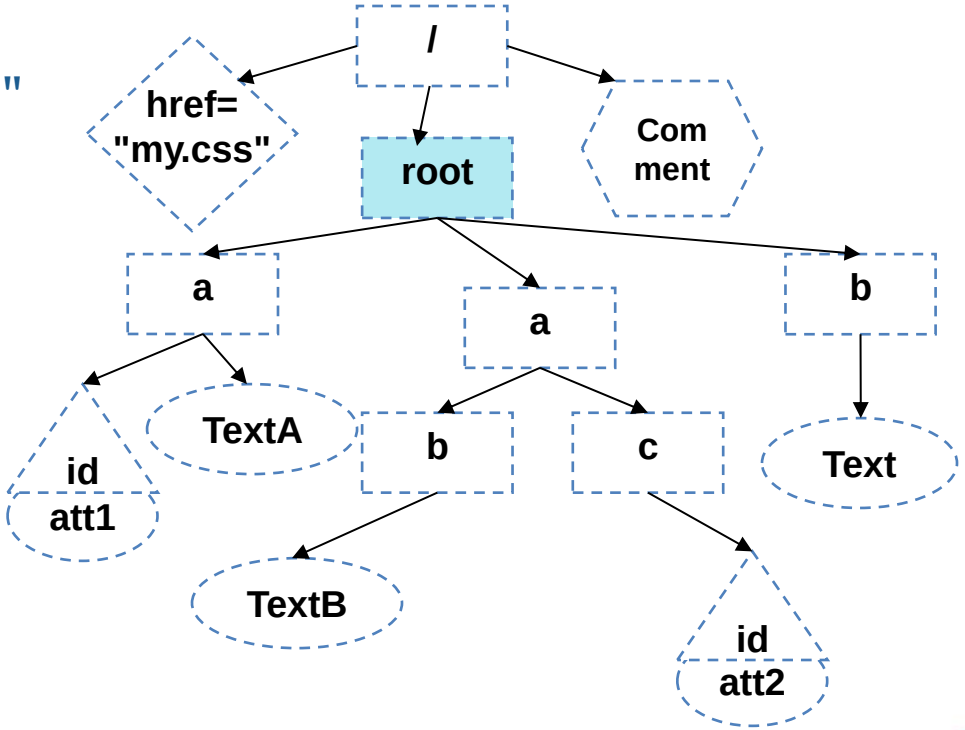
```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```



document nodes ≠ root element

Root Element

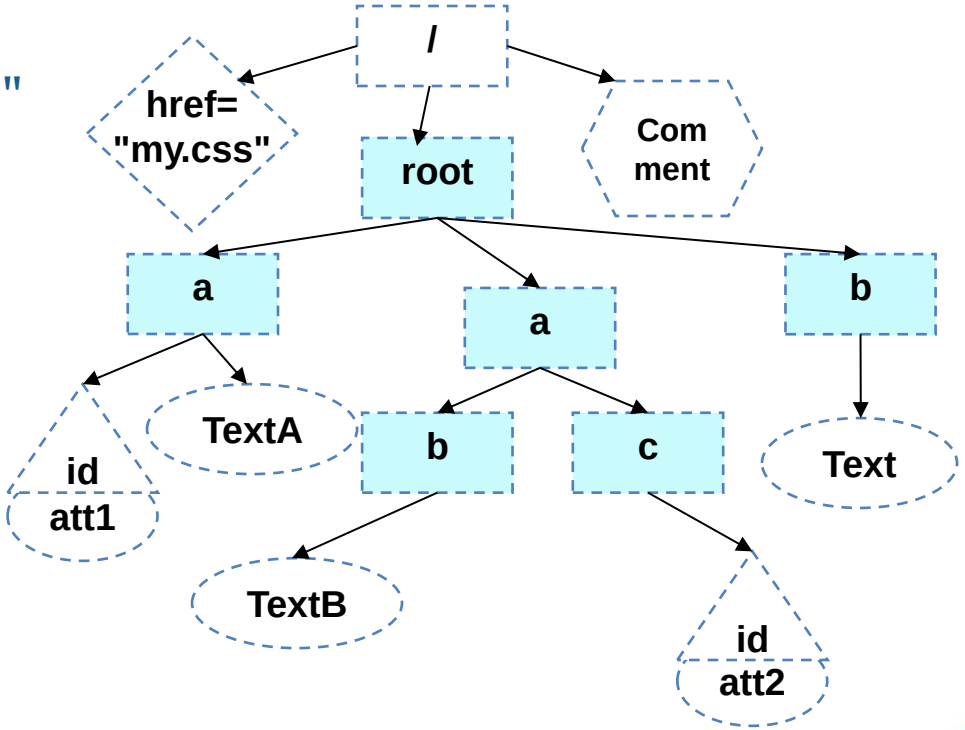
```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```



The topmost element of the tree is called the root element.

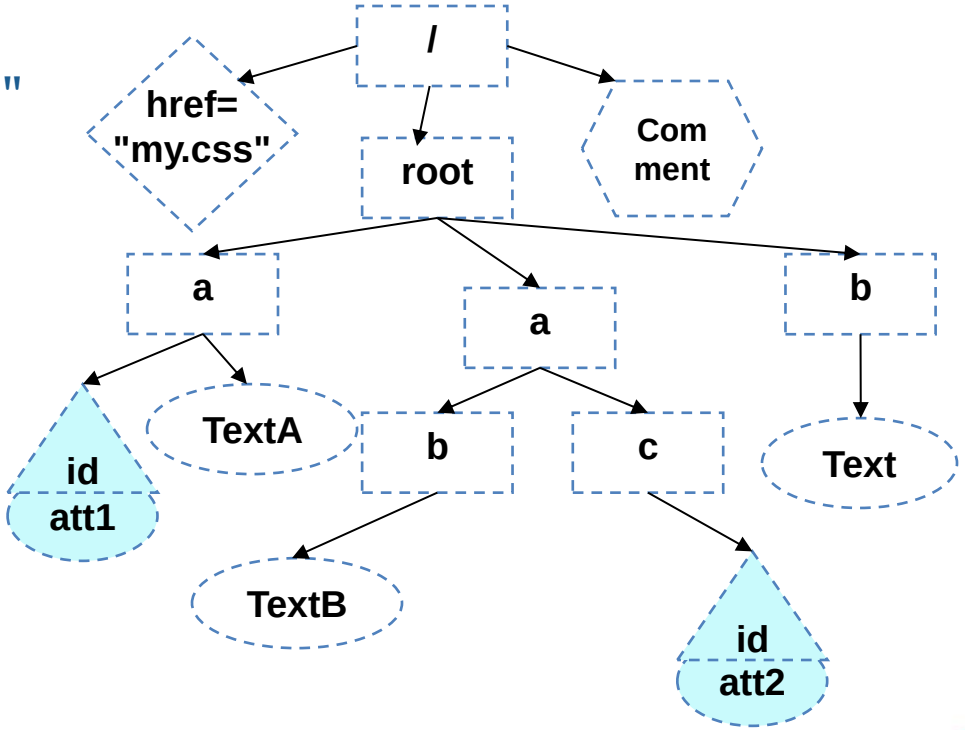
Elements

```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```



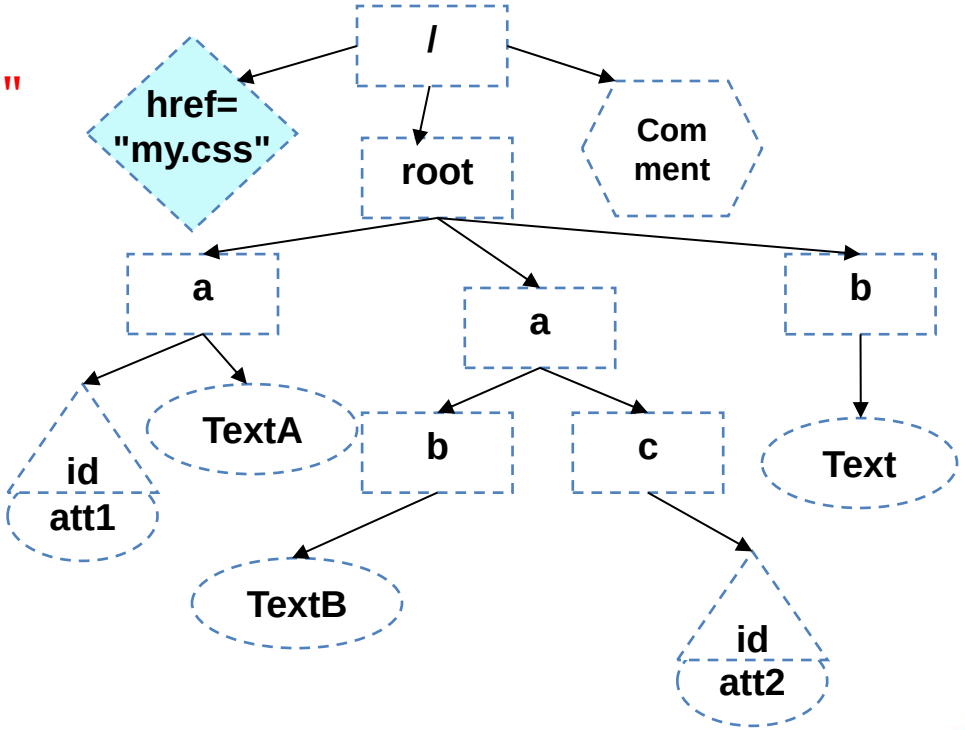
Attributes

```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```



Processing-instruction

```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```

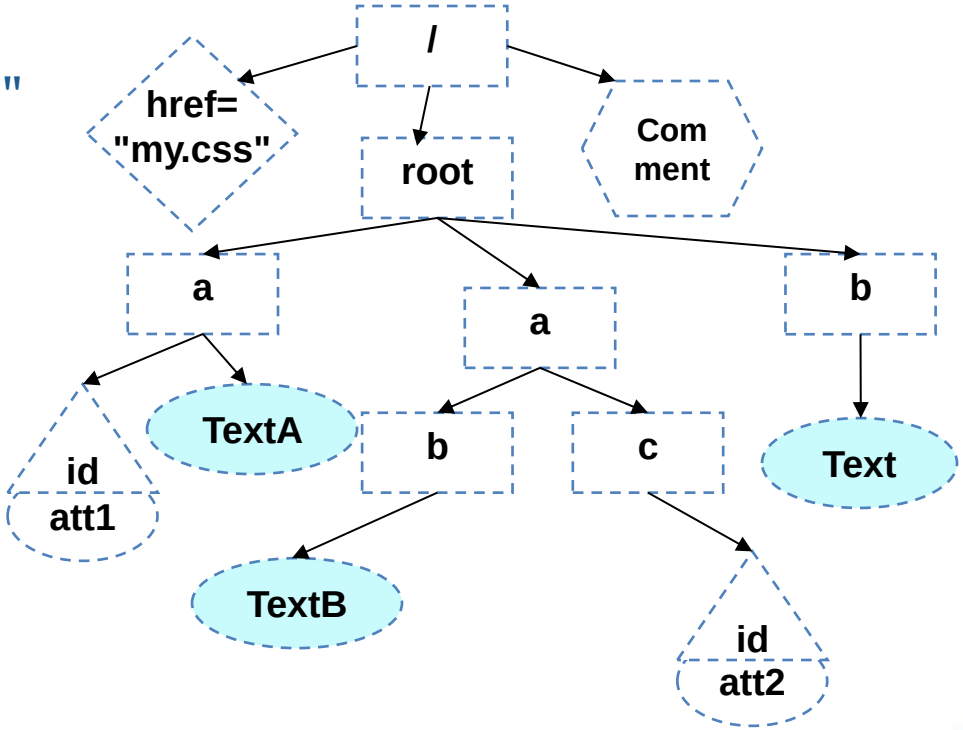


Processing-instructions are elements with <? … ?>

XML Declaration <?xml version="1.0"?> is not processing-instruction.

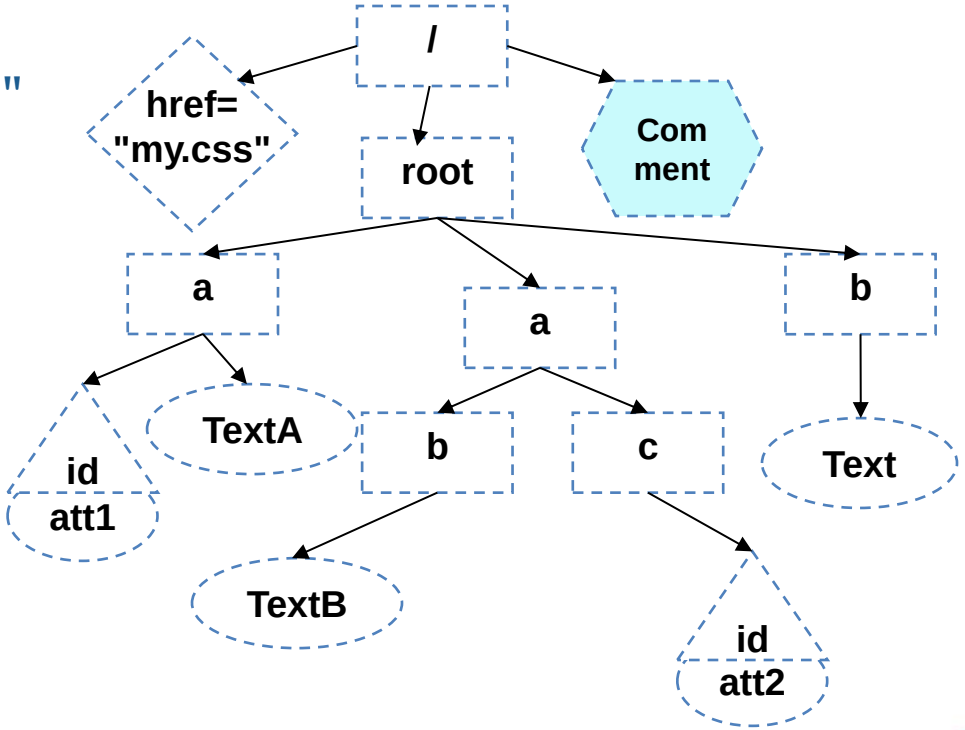
Text

```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```

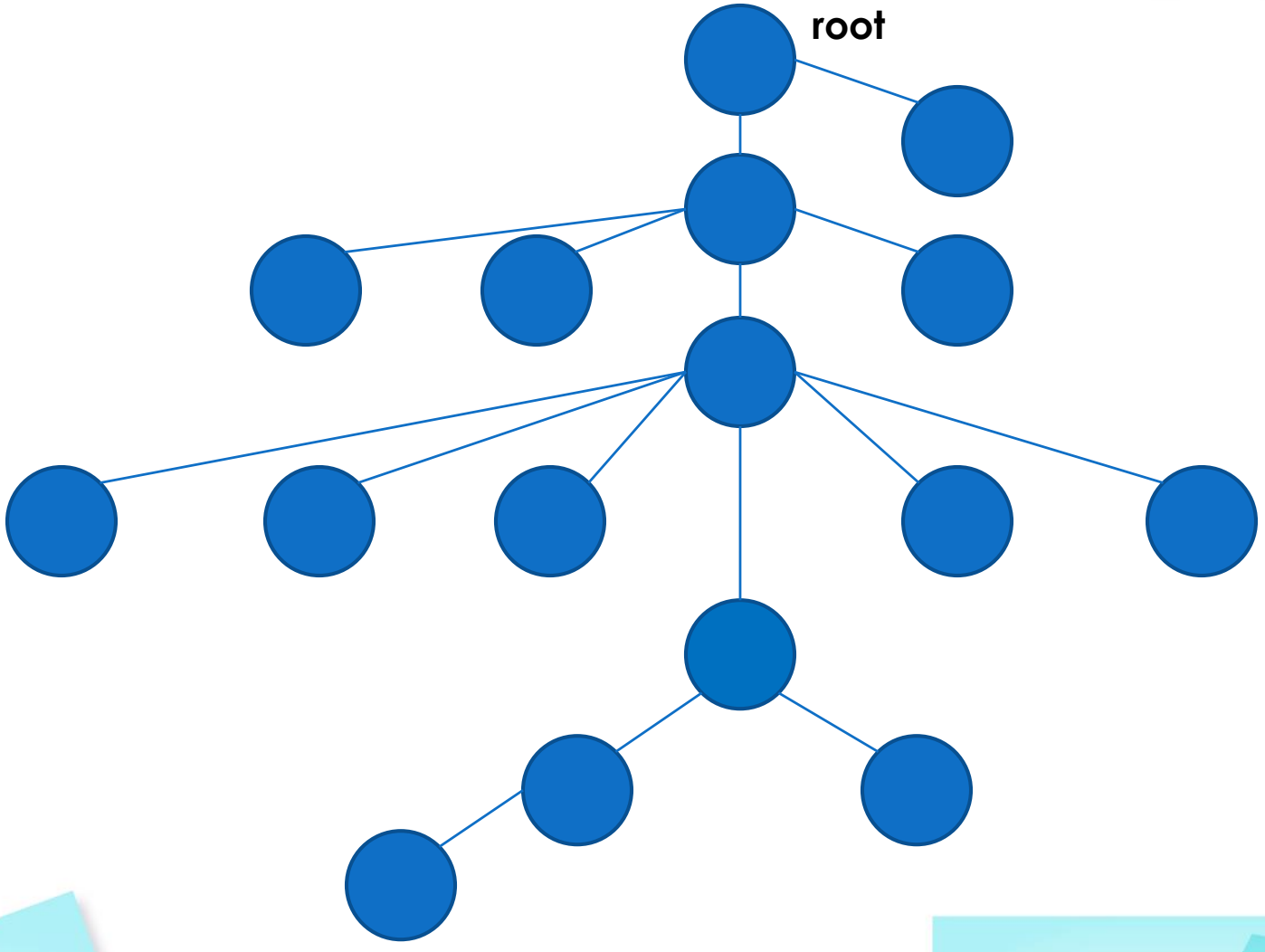


Comment

```
<?xml version="1.0"?>
<?xml-stylesheet href="my.css"
   type="text/css"?>
<!-- Comment -->
<root>
  <a id="att1">TextA</a>
  <a>
     <b>TextB</b>
     <c id="att2" />
  </a>
  <b>TextB</b>
</root>
```

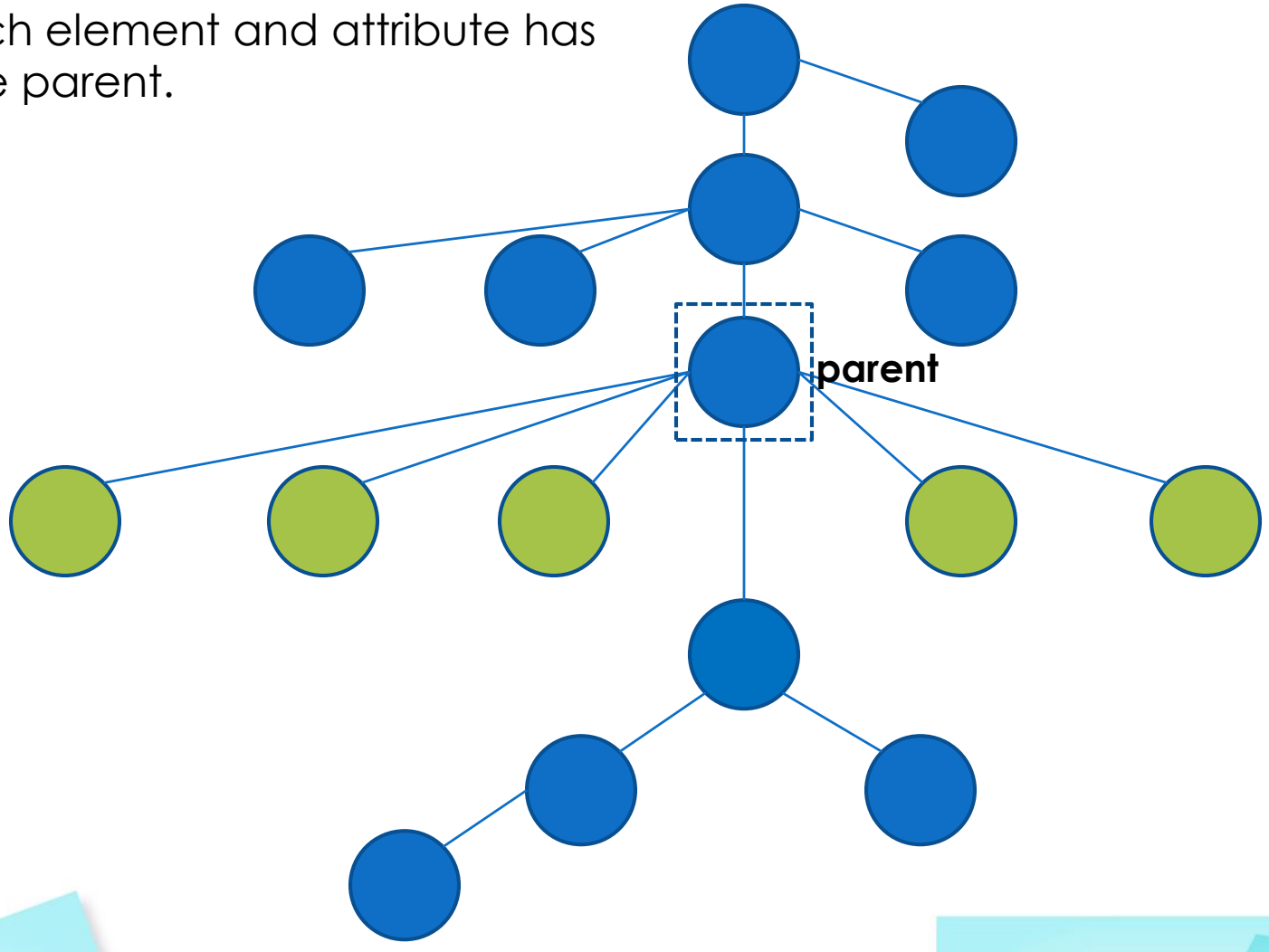


Relationship of Nodes



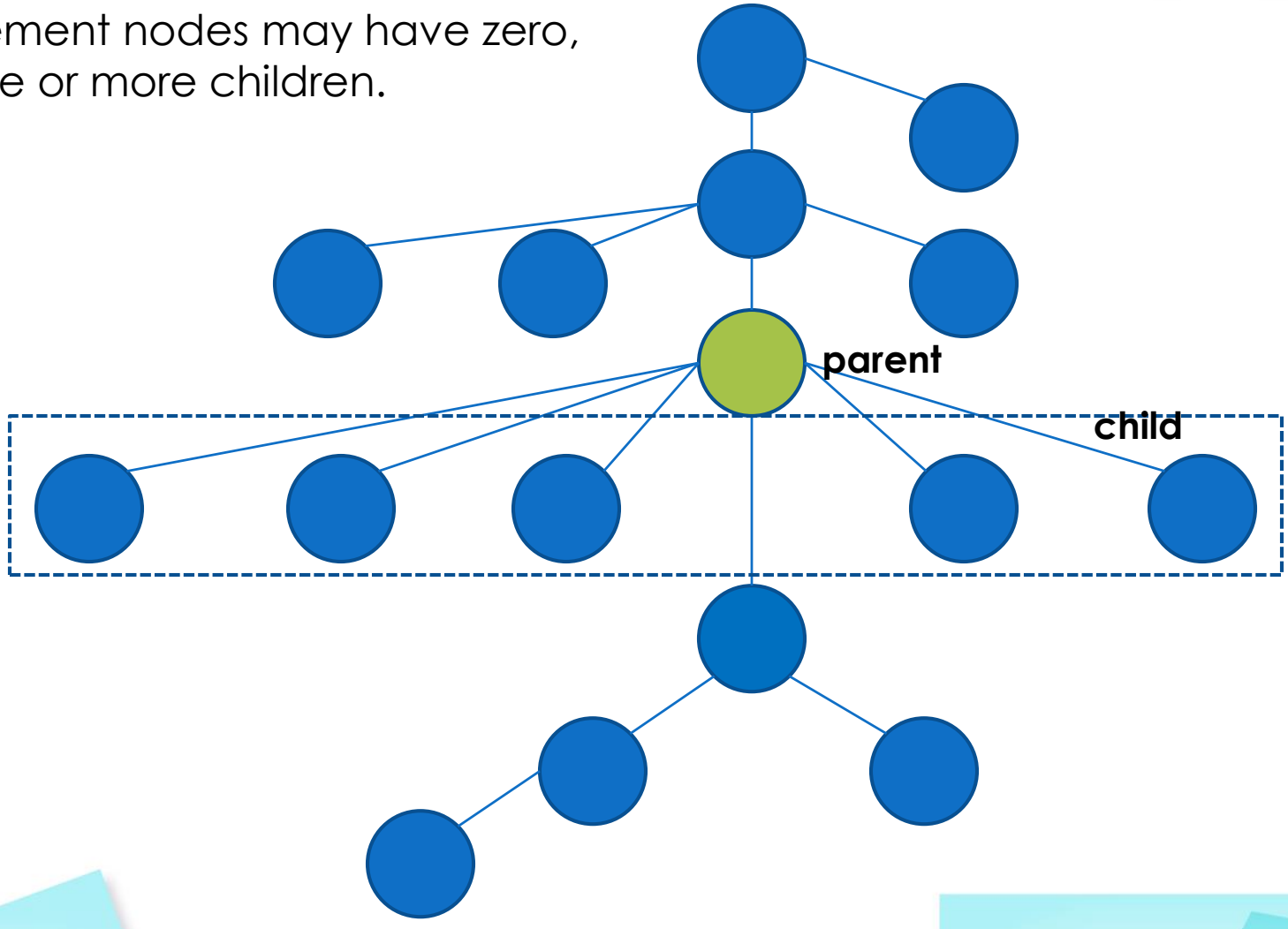
Relationship of Nodes

Each element and attribute has one parent.



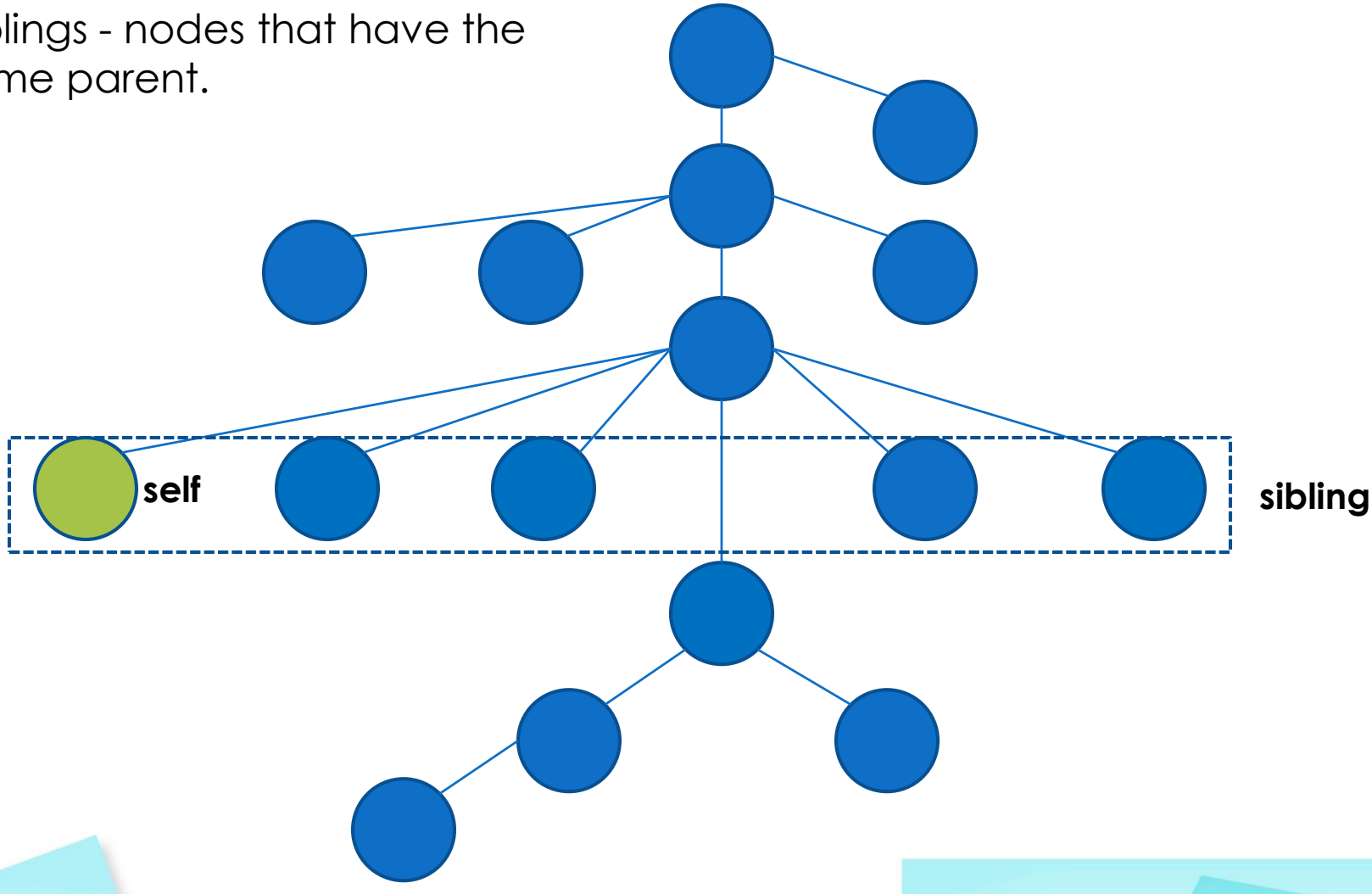
Relationship of Nodes

Element nodes may have zero, one or more children.



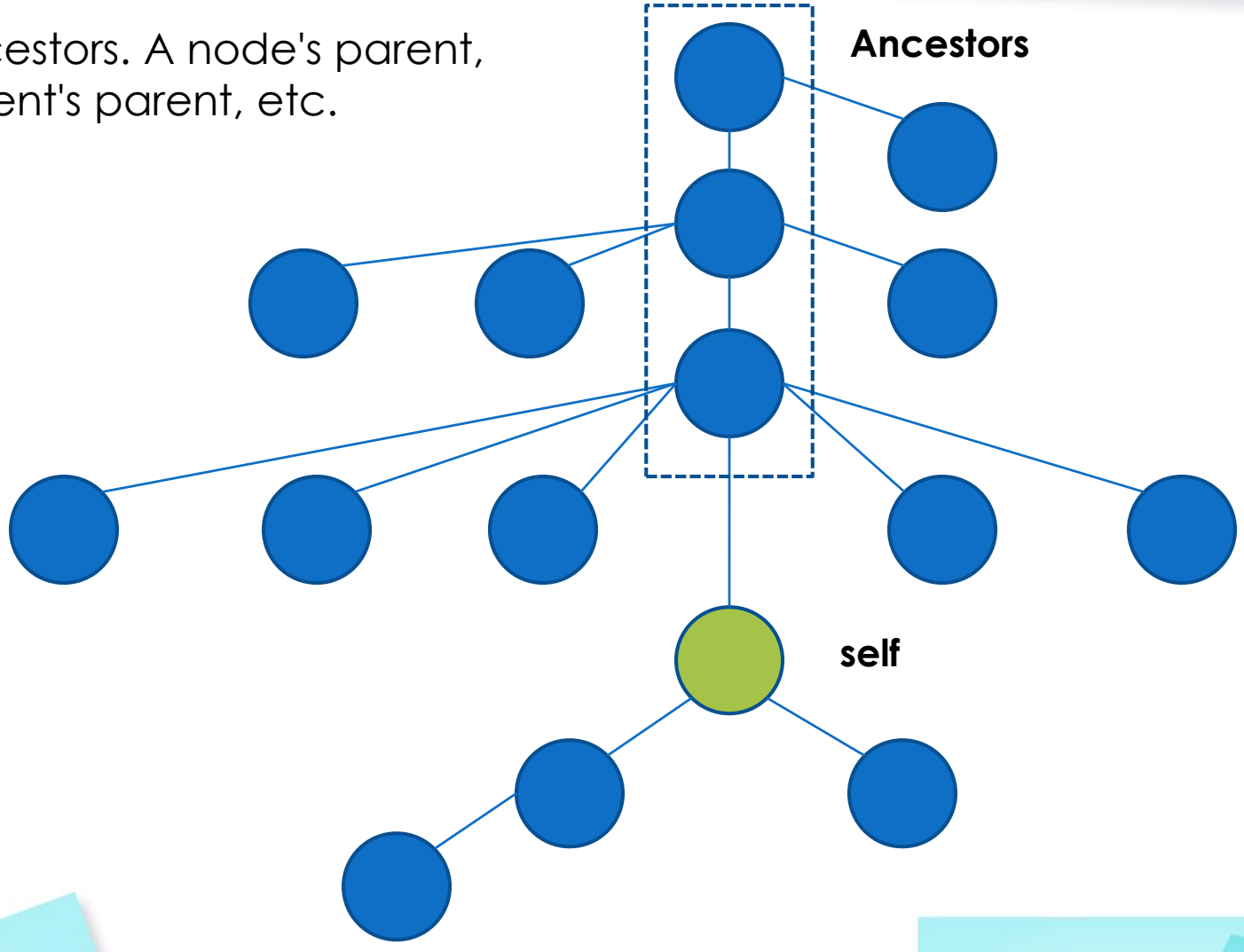
Relationship of Nodes

Siblings - nodes that have the same parent.



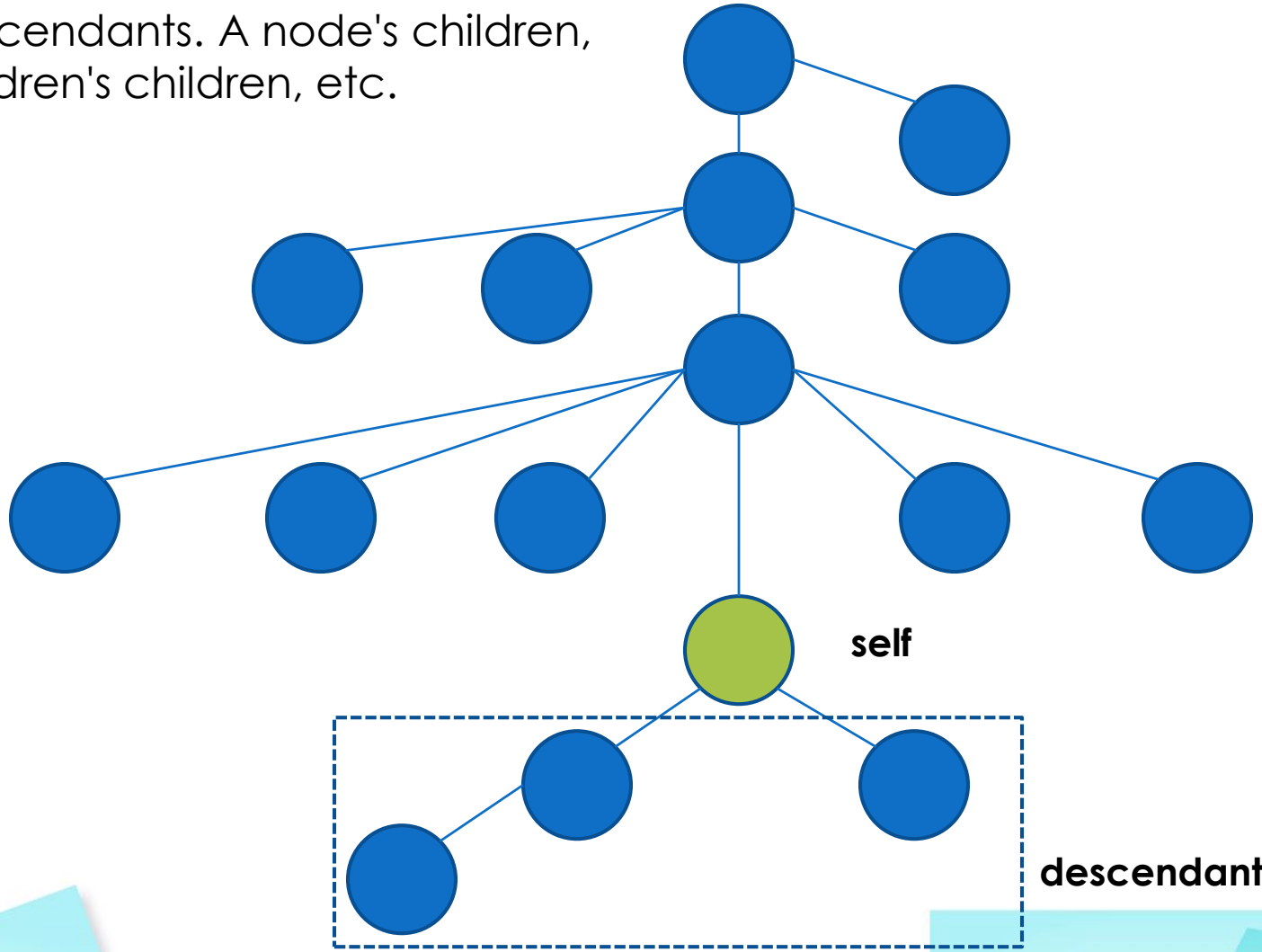
Relationship of Nodes

Ancestors. A node's parent, parent's parent, etc.



Relationship of Nodes

Descendants. A node's children, children's children, etc.



XPath Syntax

- XPath uses **path expressions** to select nodes or node-sets in an XML document. The node is selected by following a **path** or **steps**.

- A step consists of:
  - an axis (defines the tree-relationship between the selected nodes and the current node)
  - a node-test (identifies a node within an axis)
  - zero or more predicates (to further refine the selected node-set)

Axis::Select nodes[Predicate]

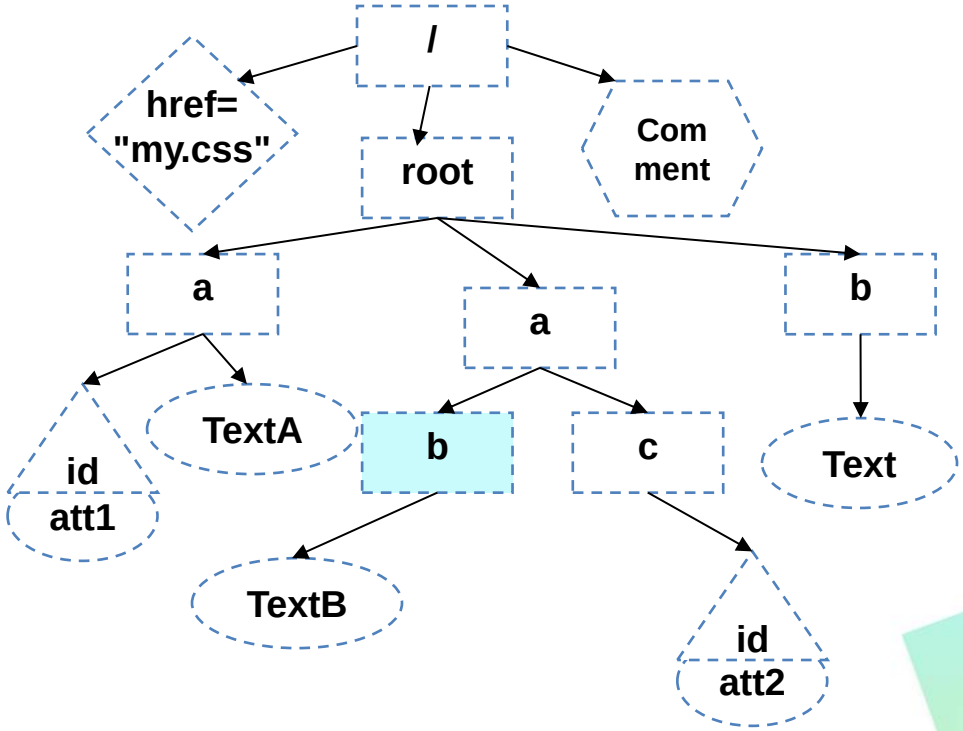
- Example: /child::html/child::a[@id]

XPath Syntax

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects **from the root node**
//	Selects nodes in the document from the current node that match the selection **no matter where they are**
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

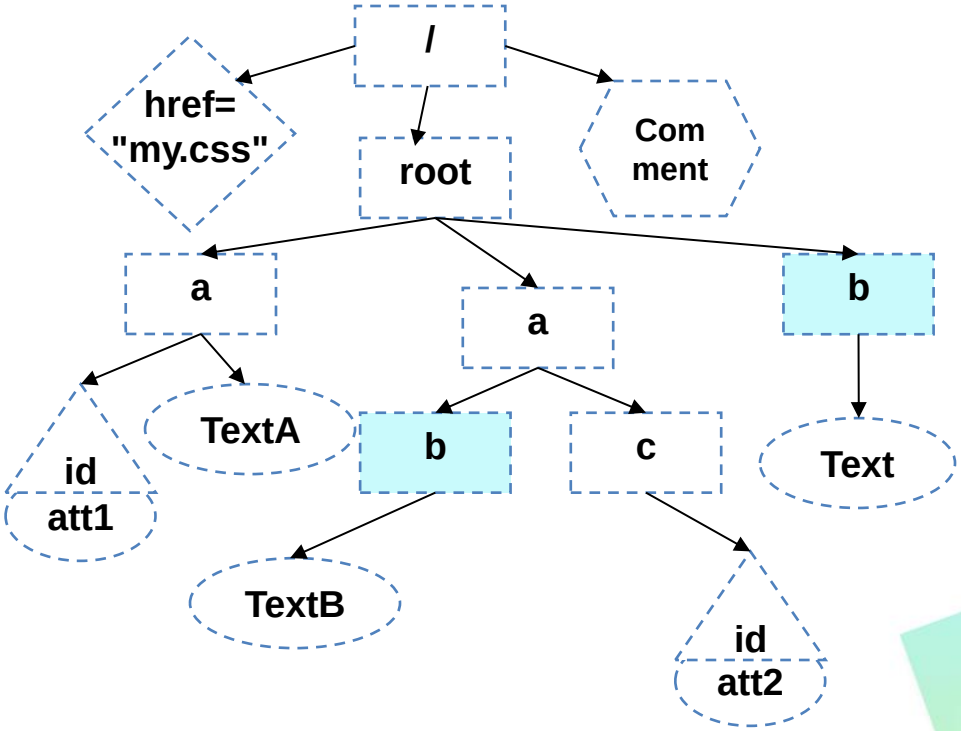
Example

- /root/a/b



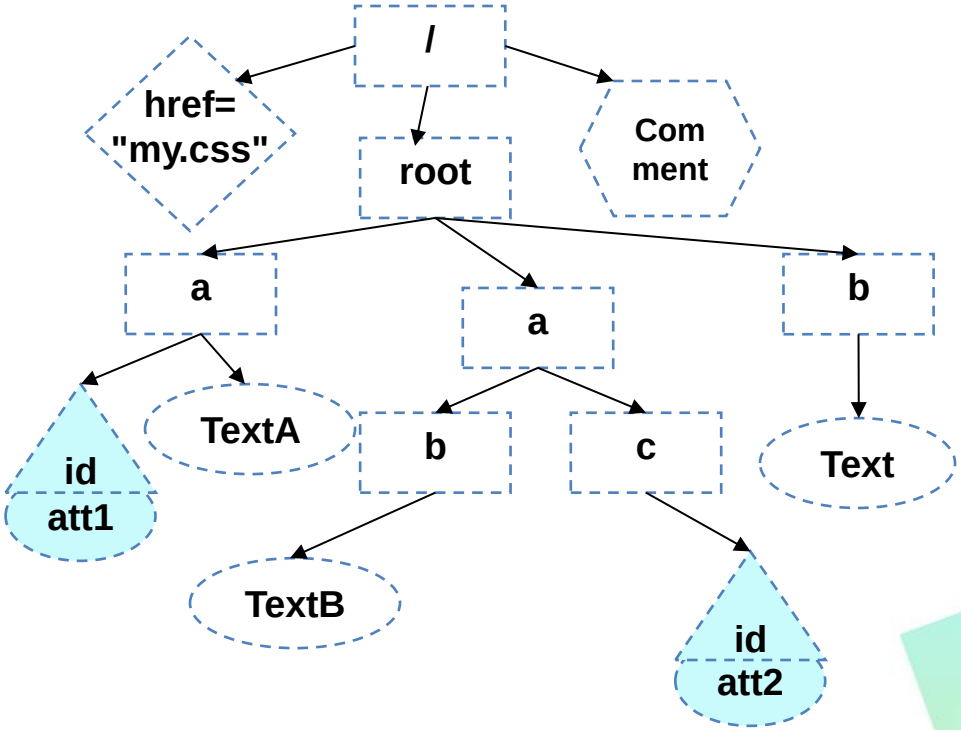
Example

- //b


/
href=
"my.css"
root
Com
ment
a
a
b
id
att1
TextA
b
c
Text
TextB
id
att2

Example

- //@id


/
href=
"my.css"
root
Com
ment
a
a
b
id
att1
TextA
b
c
Text
TextB
id
att2

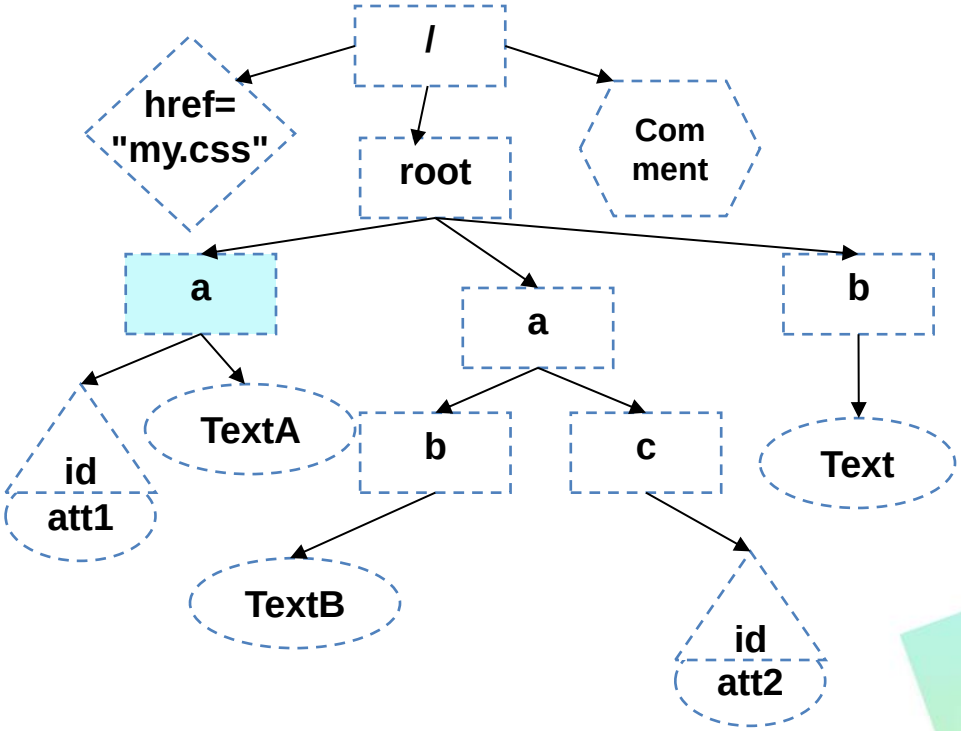
Predicates

- Predicates are used to find a specific node or a node that contains a specific value. Predicates are always embedded in square brackets.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element.
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Example

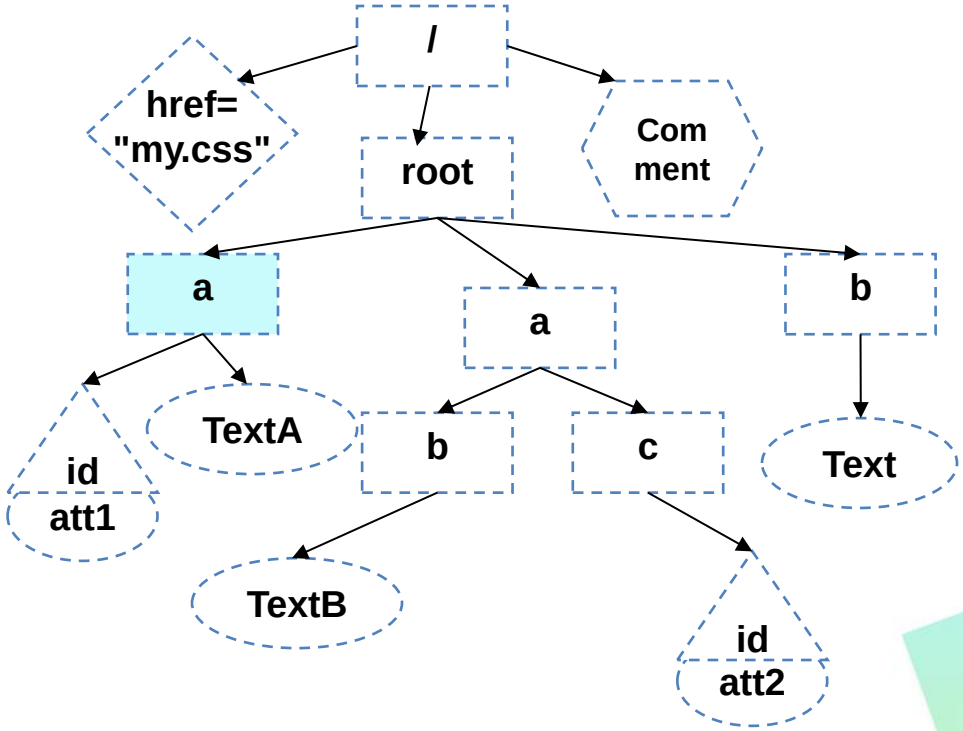
- //a[1]


/
href=
"my.css"
root
Com
ment
a
a
b
id
att1
TextA
b
c
Text
TextB
id
att2

Example

- //a[@id]

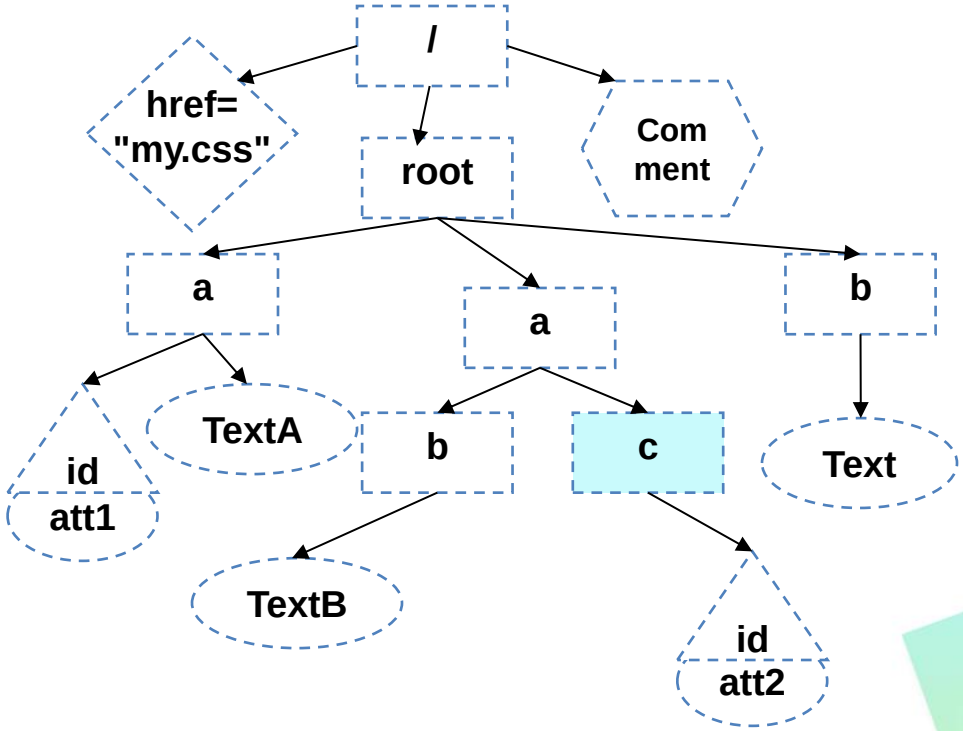
  - all elements ‘a’ with attribute ‘id’



Example

- //c[@id='att2']

  - all elements 'c' with attribute 'id', 'id' have to value 'att2'



XPath Standard Functions

- XPath includes over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node manipulation and more.

Example of XPath Functions

- **position**() - Returns the index position of the node that is currently being processed
  Example: //book[position()<=3]
  Result: Selects the first three book elements
- **last**() - Returns the number of items in the processed node listExample: //book[last()]
  Result: Selects the last book element
- **count**(node-set) - Returns the count of nodes в node-set
- **normalize-space**(string) - Removes leading and trailing spaces from the specified string, and replaces all internal sequences of white space with one and returns the result. If there is no string argument it does the same on the current node.

  Example: normalize-space(' The XML ')
  Result: 'The XML'

Predicates

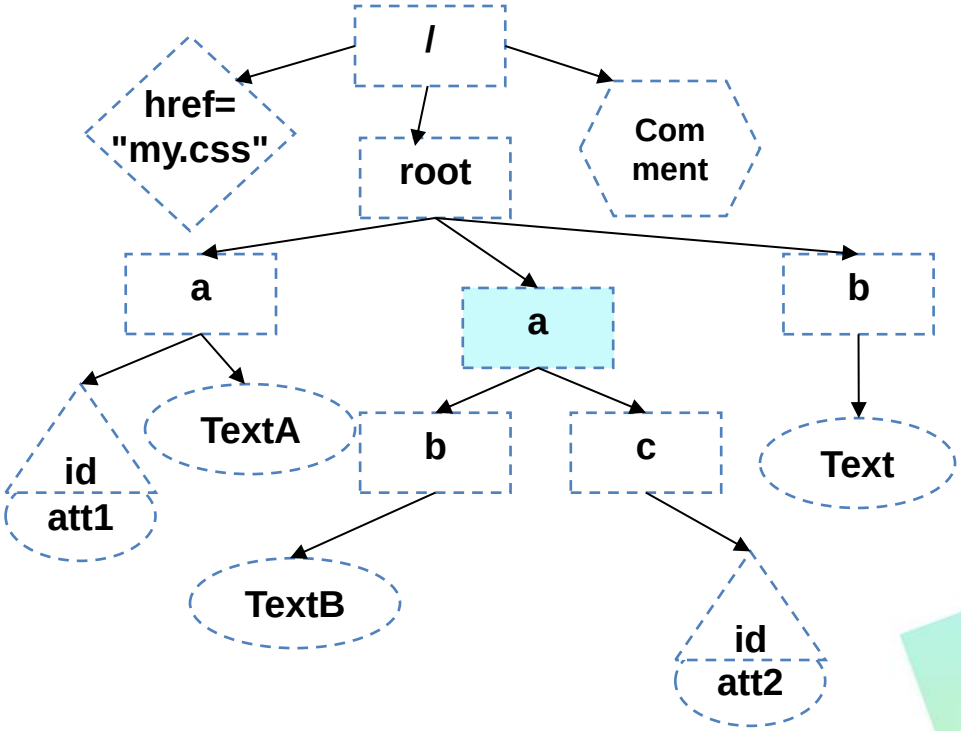
- Predicates are used to find a specific node or a node that contains a specific value. Predicates are always embedded in square brackets.

Path Expression	Result
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00] /title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Example

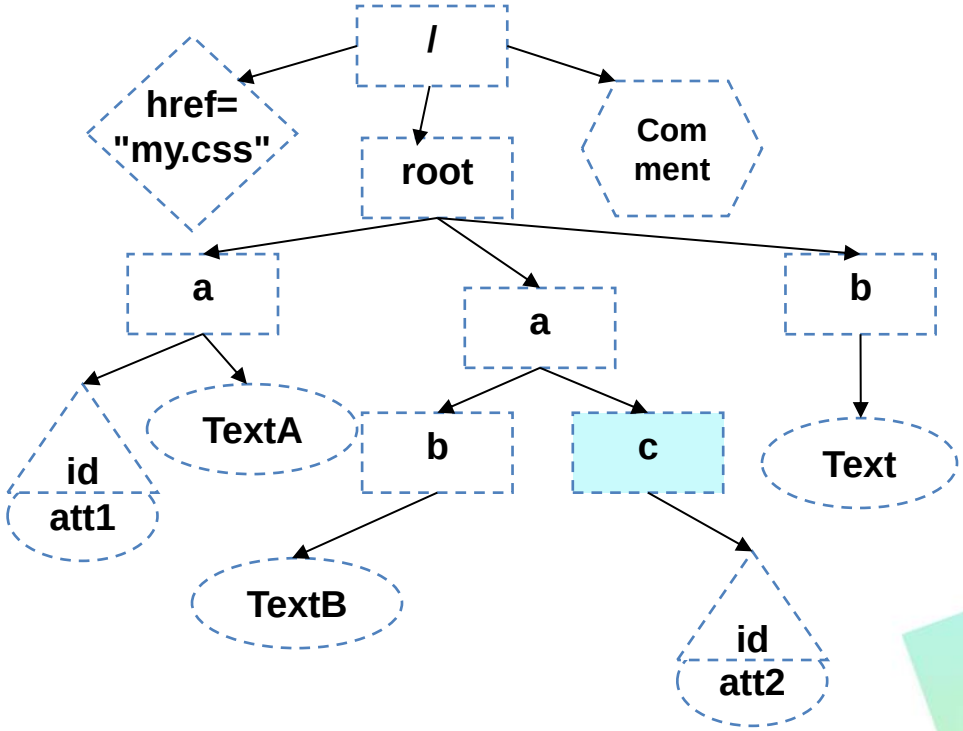
- /root/a[last()]

  - last element ‘a’ into ‘root’



Example

- //c[normalize-space(@id)= ‘att2’]



Selecting Several Paths

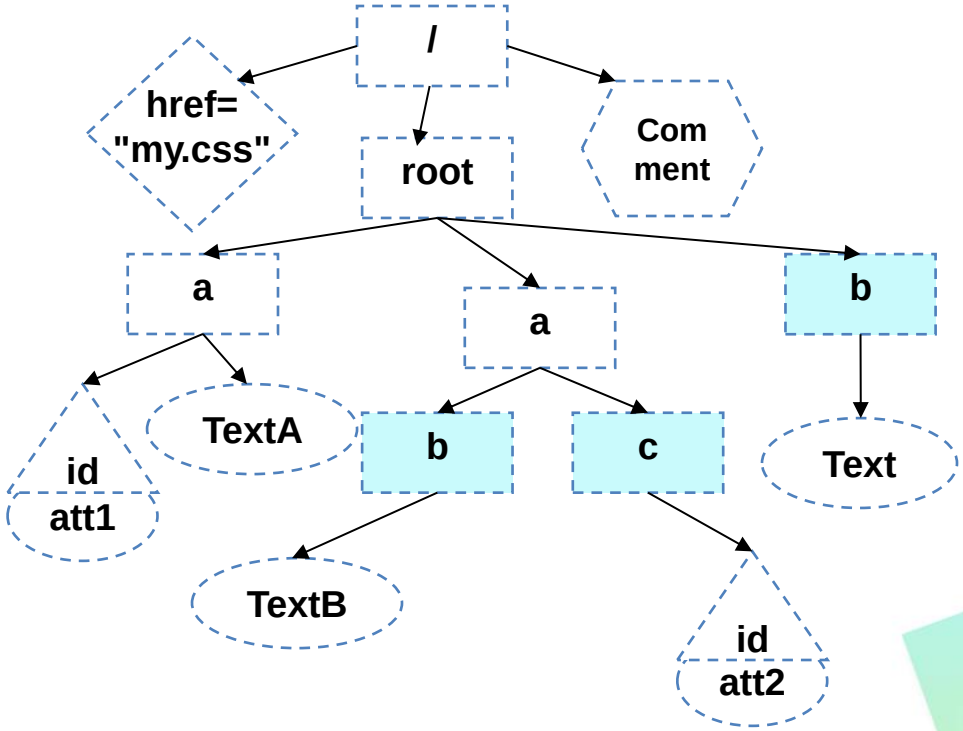
- By using the | operator in an XPath expression you can select several paths.

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

Example

- //b|//c

  - all elements
    ‘//b’ AND ‘//c’



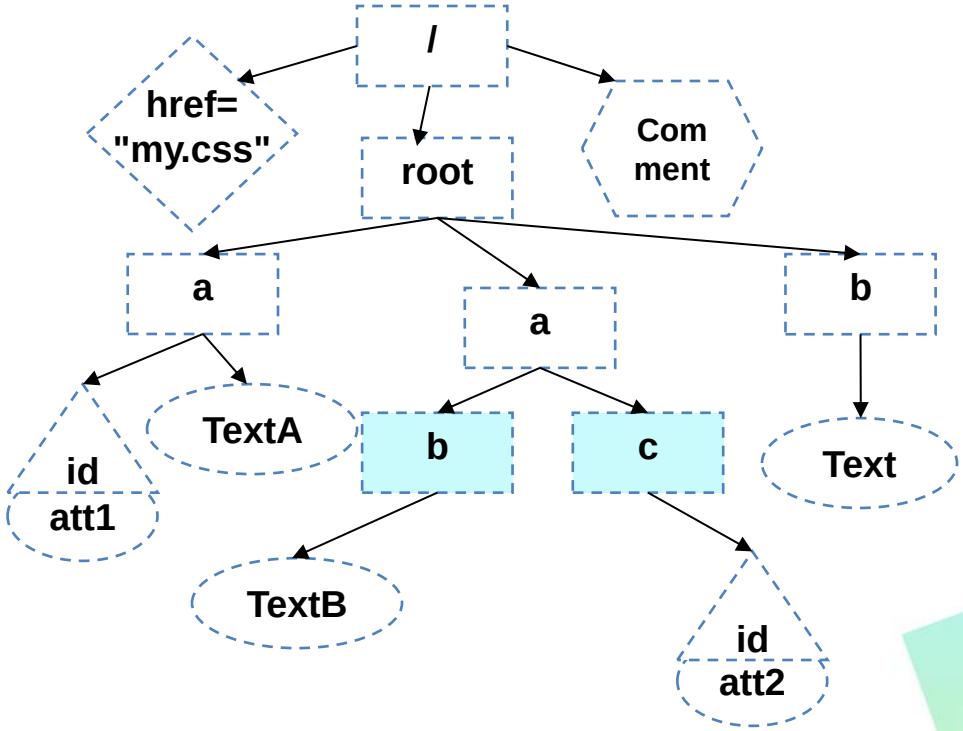
Selecting Unknown Nodes

- XPath wildcard can be used to select unknown XML elements.

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Example

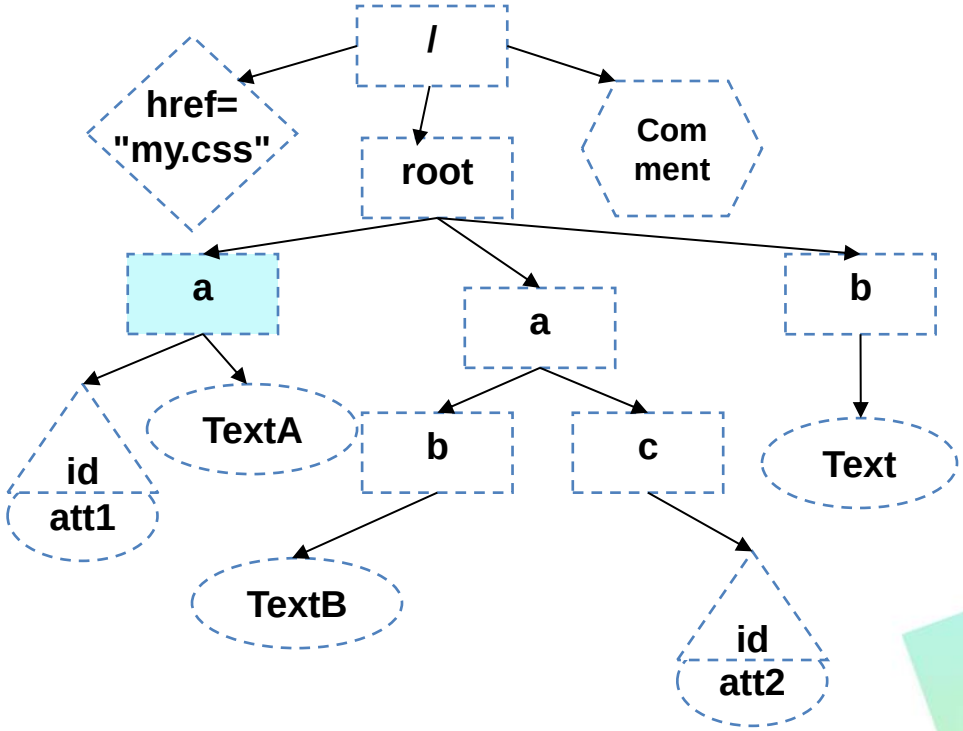
- /root/a/*



Example

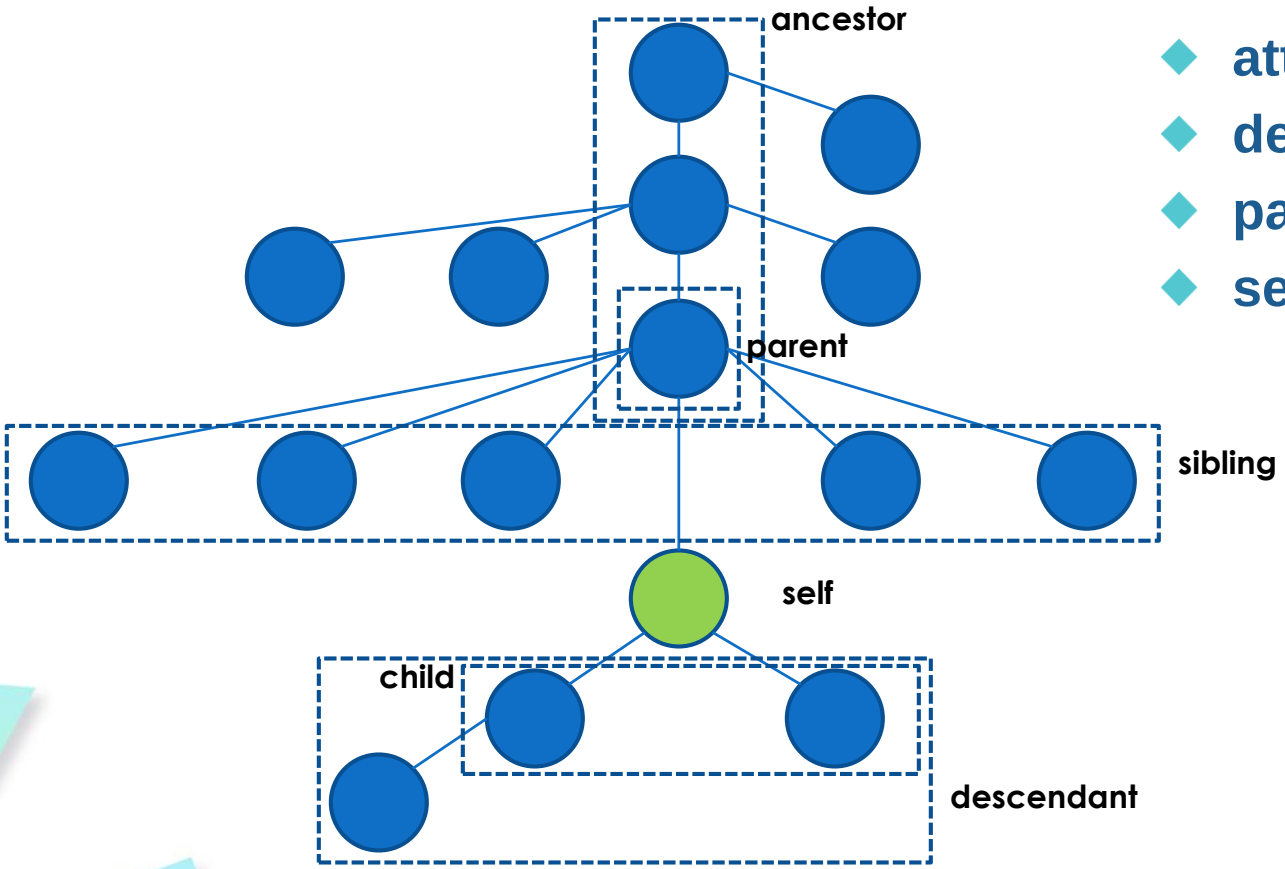
- //a[@*]

  - all elements ‘a’ with any attribute



XPath Axes

- An axis defines a node-set relative to the current node.



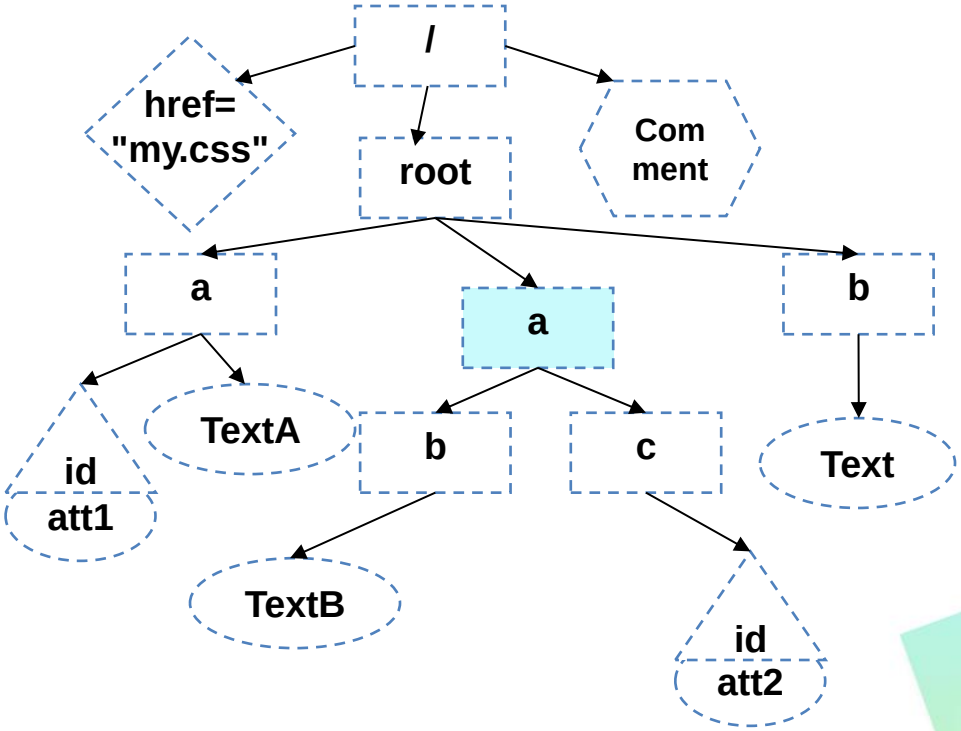
- **attribute:: = @**
- **descendant:: = .//**
- **parent:: = ..**
- **self:: = .**

Axes

- attribute:: = @
- descendant:: = .//
- parent:: = ..
- self:: = .

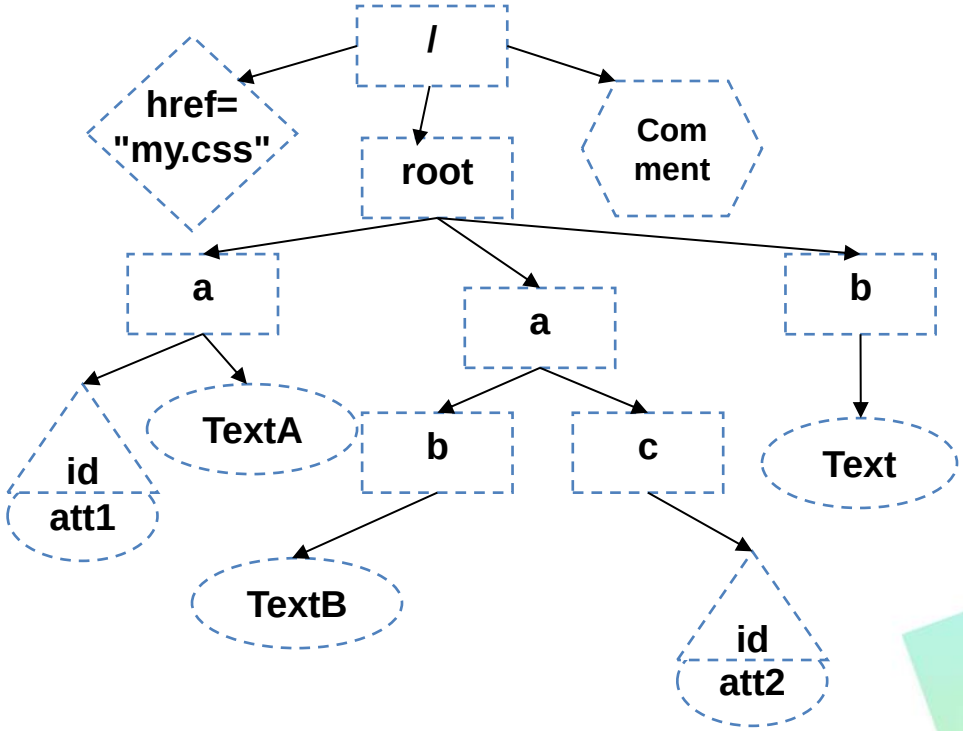
Example

- //parent::c



Examples

- //*[@id]
- //parent::b
- //descendant::a
- /root/c
- //*[count(@*)=0]
- //b[1]



Create an XSL Style Sheet

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


...


</xsl:stylesheet>
```

Link the XSL Style Sheet to the XML Document

```
<?xml version="1.0"?>

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

<root>
…
</root>
```

The <xsl:template> Element

- The <xsl:template> element is used to build templates.
- The **match** attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document.
- The value of the match attribute is an XPath expression (i.e. **match="/" defines the whole document**).

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

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

</xsl:stylesheet>
```