Hierarchical data formats: XPath, XSLT

Jakub Klímek

XML Path Language - XPath

XPath - example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title xml:lang="en">My catalog</title>
    <title xml:lang="cs">Můj katalog</title>
    <description xml:lang="en">This is my dummy catalog</description>
    <description xml:lang="cs">Toto je můj falešný katalog</description>
    <contact-point>
        <name xml:lang="en">John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title xml:lang="en">Bikesharing in Brno</title>
            <title xml:lang="cs">Sdílení kol v Brně</title>
            <distributions>
                <distribution>
                    <media-type>application/xml</media-type>
                    <downloadURL>http://brno.cz/myfile.xml</downloadURL>
                </distribution>
                <distribution>
                    <accessService>
                        <endpointURL>https://brno.cz/myAPI</endpointURL>
                        <title xml:lang="en">My API</title>
                    </accessService>
                </distribution>
            </distributions>
        </dataset>
        <dataset>
            <title xml:lang="en">Bikesharing in Prague</title>
            <title xml:lang="cs">Sdílení kol v Praze</title>
            <distributions>
                <distribution>
                    <title xml:lang="en">CSV</title>
                    <media-type>text/csv</media-type>
                    <downloadURL>http://praha.eu/myfile.csv</downloadURL>
                </distribution>
            </distributions>
        </dataset>
    </datasets>
</catalog>
```

`/catalog/datasets/dataset/title`

- Element: title
- Element: title
- Element: title
- Element: title

`/catalog/datasets/dataset/title/text()`

- Text: Bikesharing in Brno
- Text: Sdílení kol v Brně
- Text: Bikesharing in Prague
- Text: Sdílení kol v Praze

`/catalog/datasets/dataset/title[@xml:lang="en"]/text()`

- Text: Bikesharing in Brno
- Text: Bikesharing in Prague

XPath - specifications

- XML Path Language (XPath) 1.0
  - W3C Recommendation **1999**
  - what we will cover mostly
- XML Path Language (XPath) 2.0 (Second Edition)
  - W3C Recommendation **2010**
  - most widely implemented
- XML Path Language (XPath) 3.1
  - W3C Recommendation **2017**

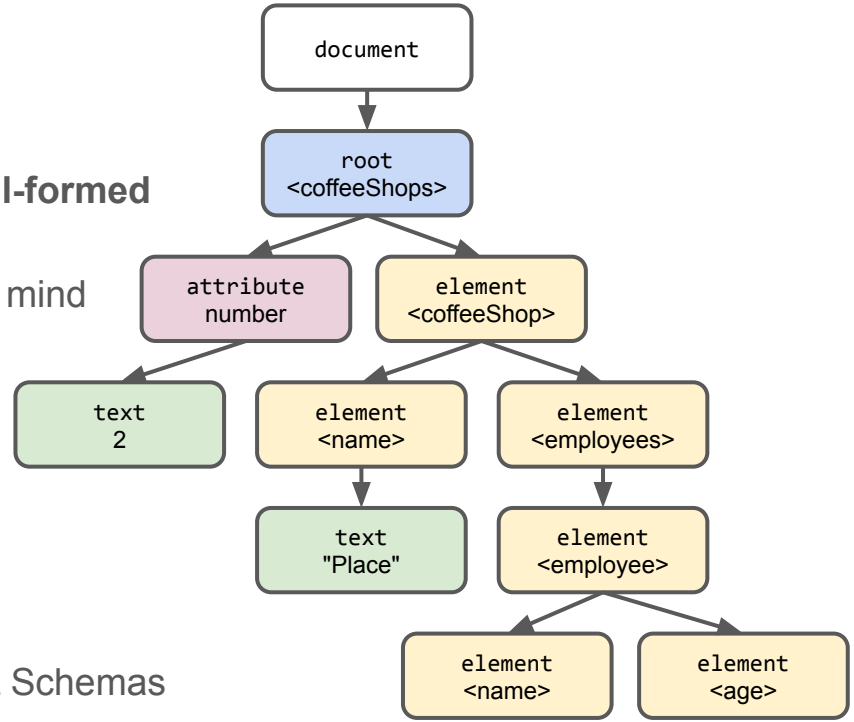
XML Infoset and XPath Data Model

XML Information Set (Second Edition)

- W3C Recommendation (Second Edition) **2004**
- Set of definitions for referring to information in **well-formed** XML documents
- Created with no particular processing language in mind

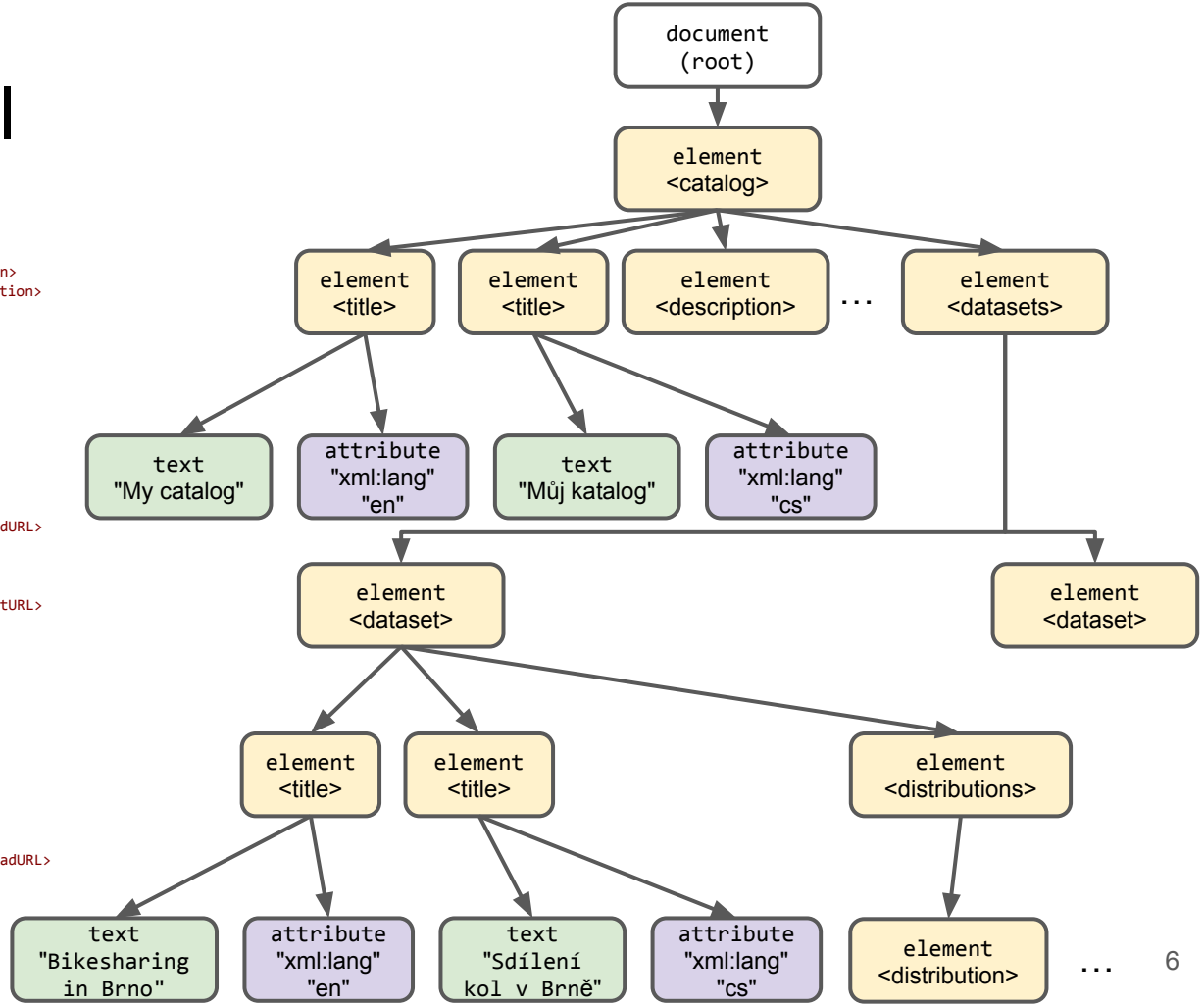
XQuery and XPath Data Model 3.1

- W3C Recommendation **2017**
  - For XPath 2.0 **2001**
- Based on XML Infoset
- Created for use in XPath, XSLT and XQuery
- Includes support for information coming from XML Schemas
  - e.g. types of text values



XPath Data Model

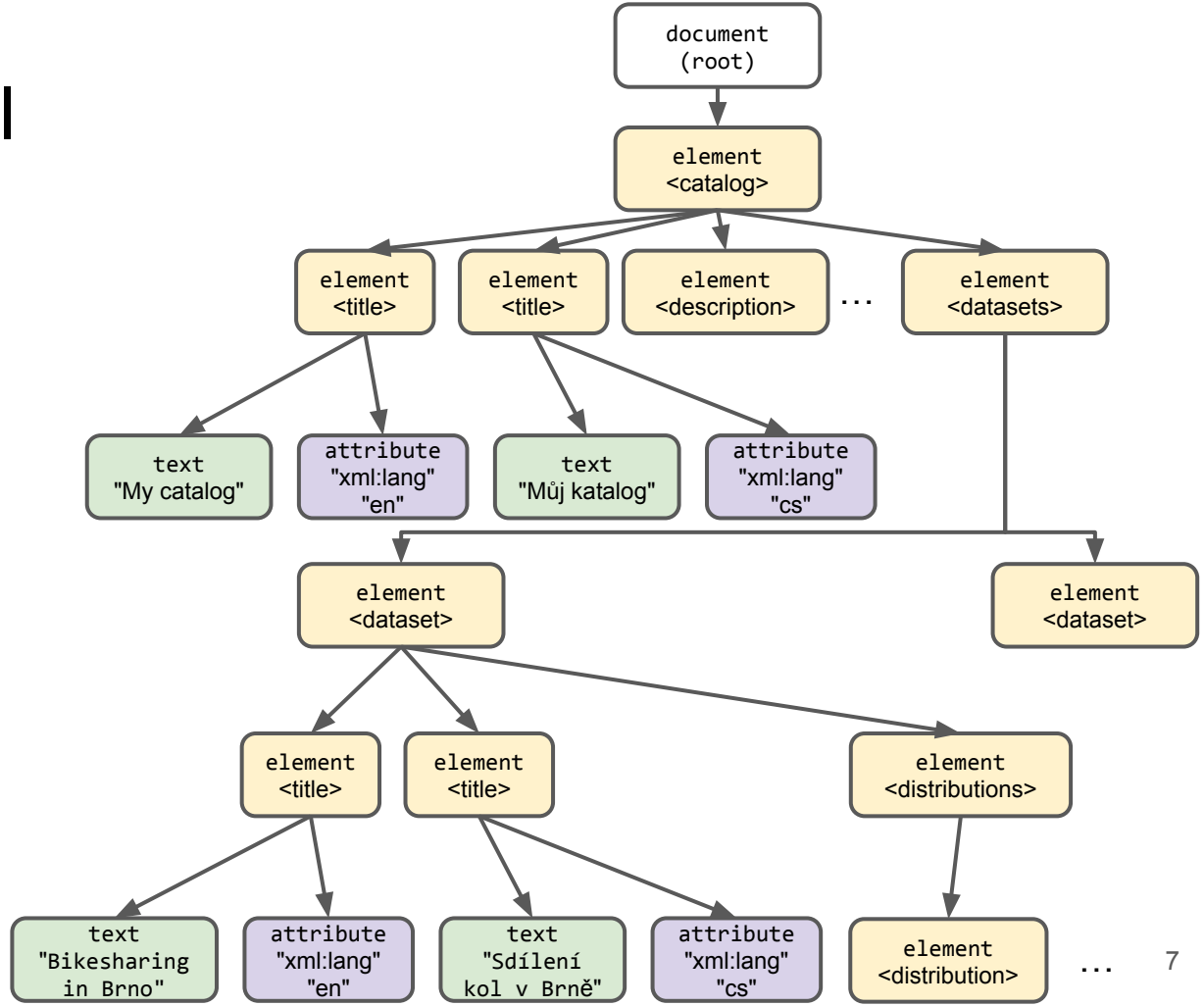
```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title xml:lang="en">My catalog</title>
    <title xml:lang="cs">Můj katalog</title>
    <description xml:lang="en">This is my dummy catalog</description>
    <description xml:lang="cs">Toto je můj falešný katalog</description>
    <contact-point>
        <name xml:lang="en">John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title xml:lang="en">Bikesharing in Brno</title>
            <title xml:lang="cs">Sdílení kol v Brně</title>
            <distributions>
                <distribution>
                    <media-type>application/xml</media-type>
                    <downloadURL>http://brno.cz/myfile.xml</downloadURL>
                </distribution>
                <distribution>
                    <accessService>
                        <endpointURL>https://brno.cz/myAPI</endpointURL>
                        <title xml:lang="en">My API</title>
                    </accessService>
                </distribution>
            </distributions>
        </dataset>
        <dataset>
            <title xml:lang="en">Bikesharing in Prague</title>
            <title xml:lang="cs">Sdílení kol v Praze</title>
            <distributions>
                <distribution>
                    <title xml:lang="en">CSV</title>
                    <media-type>text/csv</media-type>
                    <downloadURL>http://praha.eu/myfile.csv</downloadURL>
                </distribution>
            </distributions>
        </dataset>
    </datasets>
</catalog>
```



XPath Data Model

XPath types of **nodes**

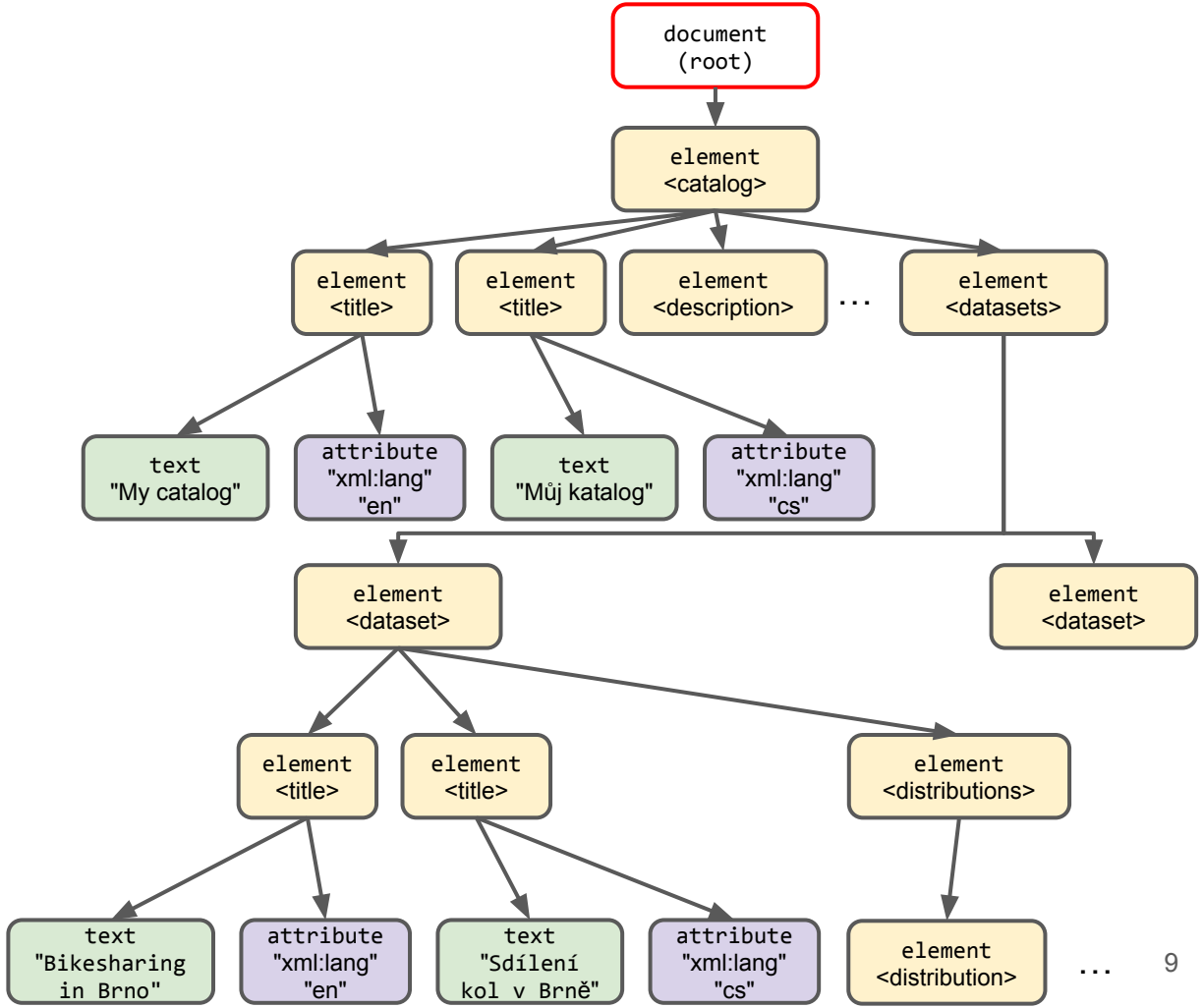
- document (root)
- element
- attribute
- text
- comment
- namespace
- processing instruction



XPath basic examples

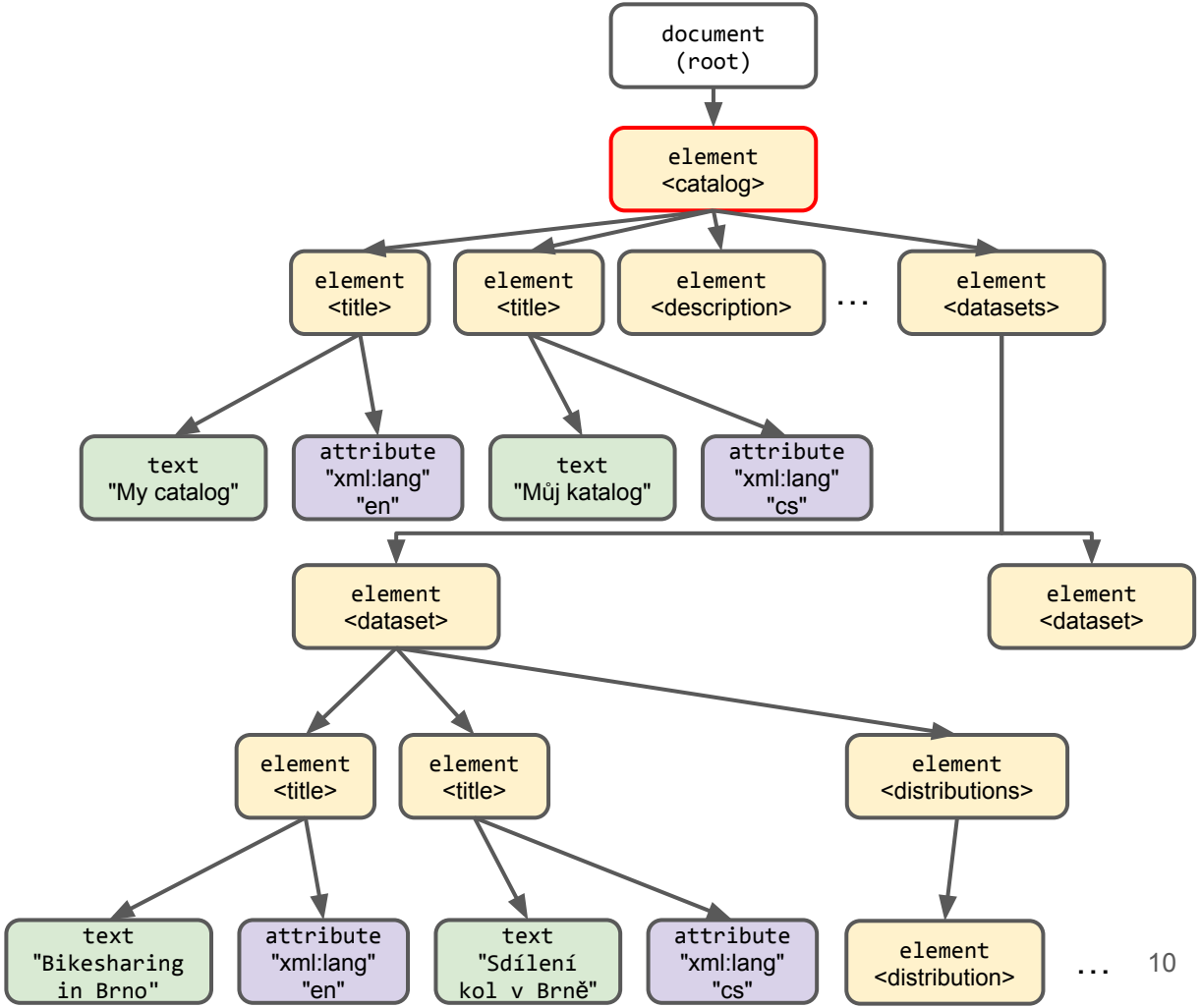
XPath - example

/



XPath - example

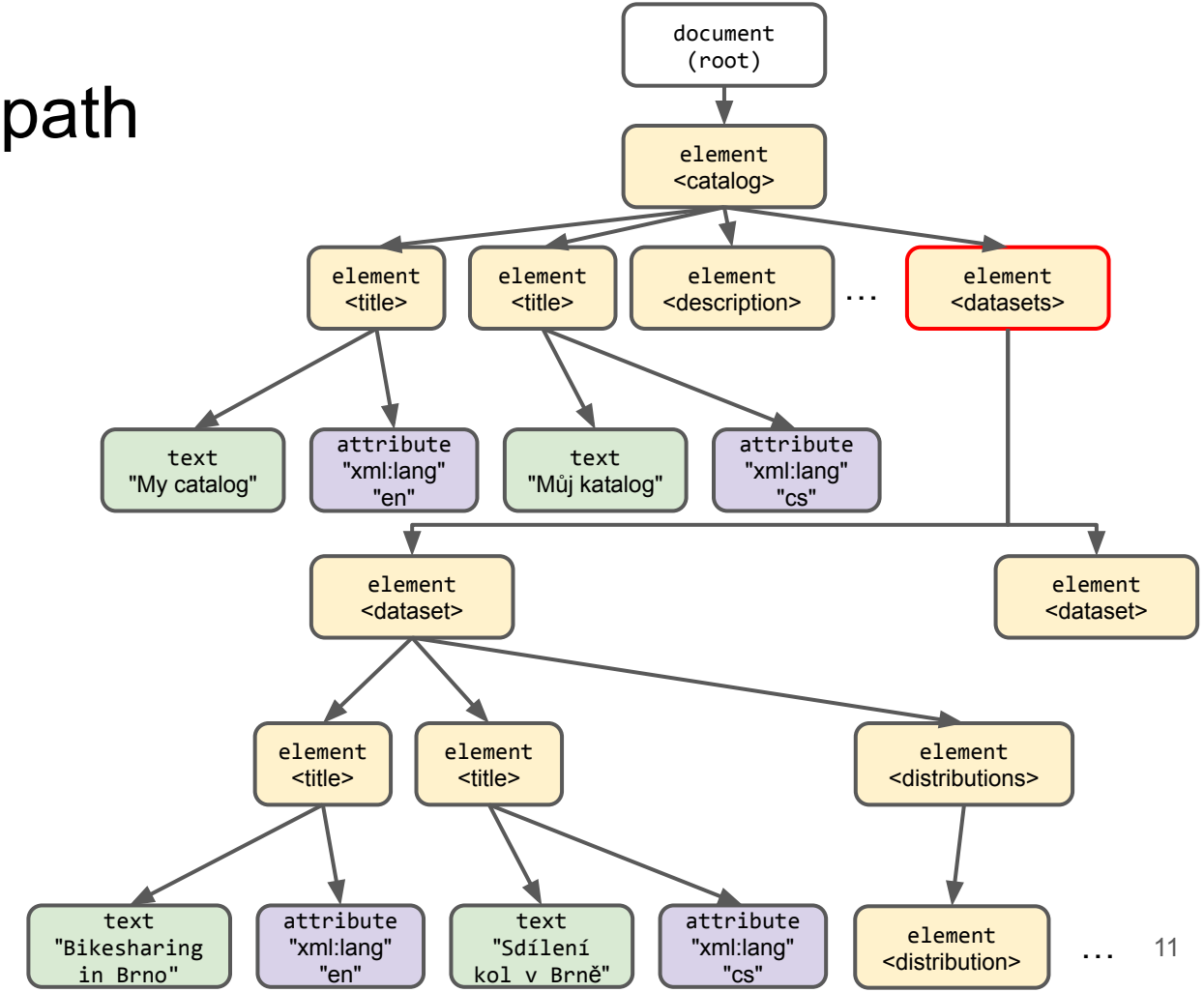
`/catalog`



XPath - absolute path

`/catalog/datasets`

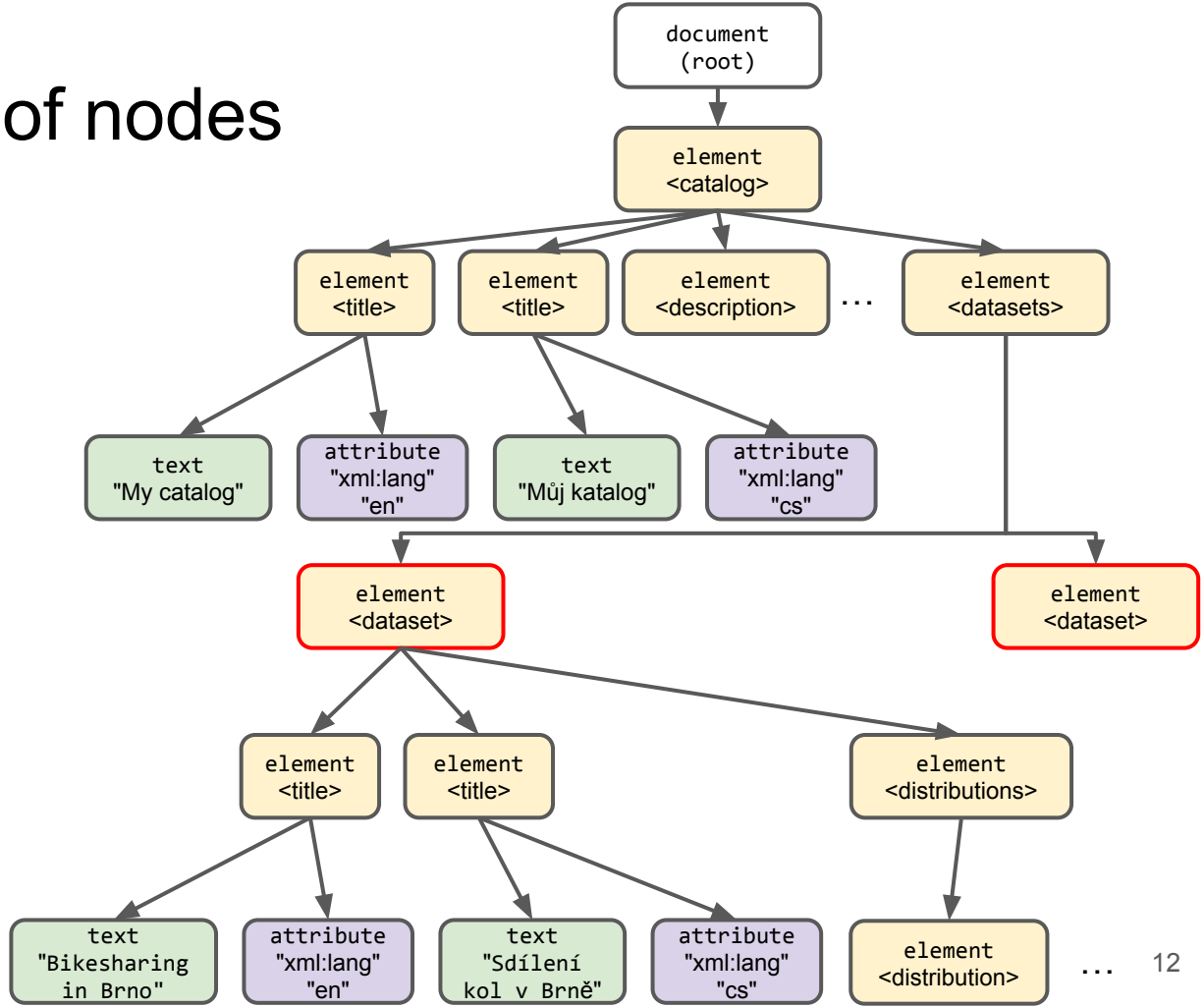
Absolute path
`/step1/.../stepN`



XPath - result set of nodes

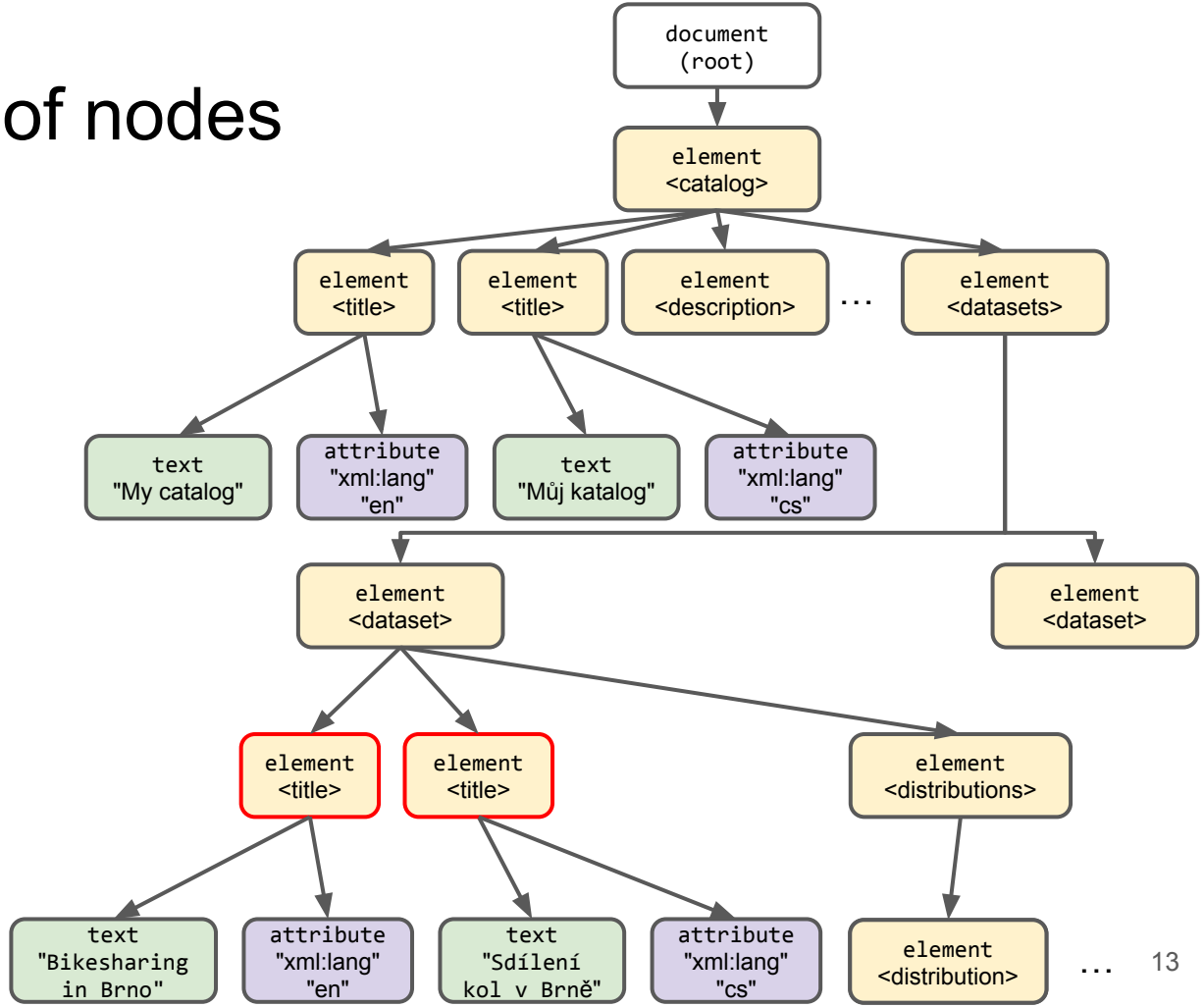
`/catalog/datasets/dataset`

Result is a set of nodes
(in no explicit order)



XPath - result set of nodes

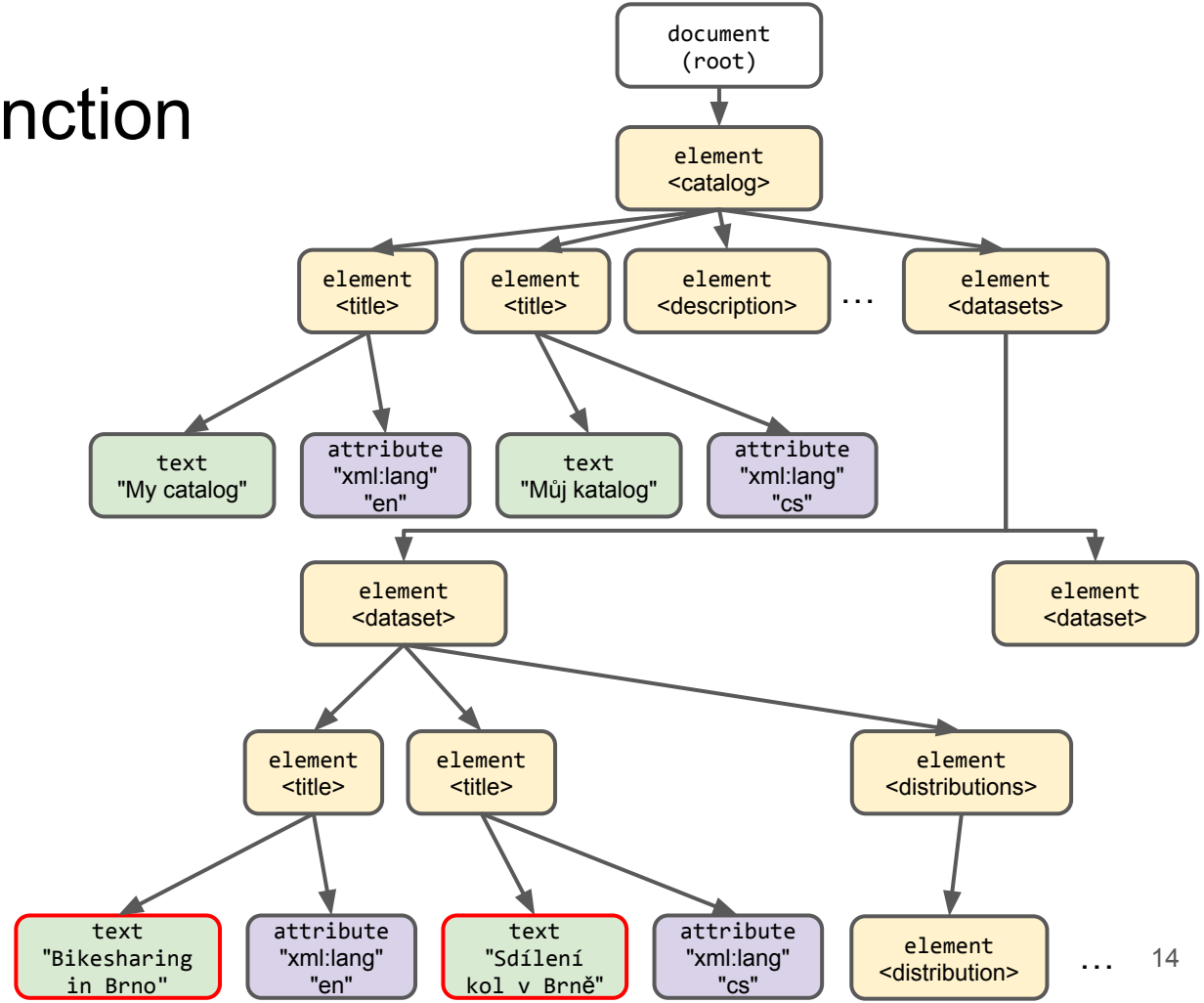
/catalog/datasets/dataset/title



XPath - access function

```
/catalog/datasets/dataset/title/text()
```

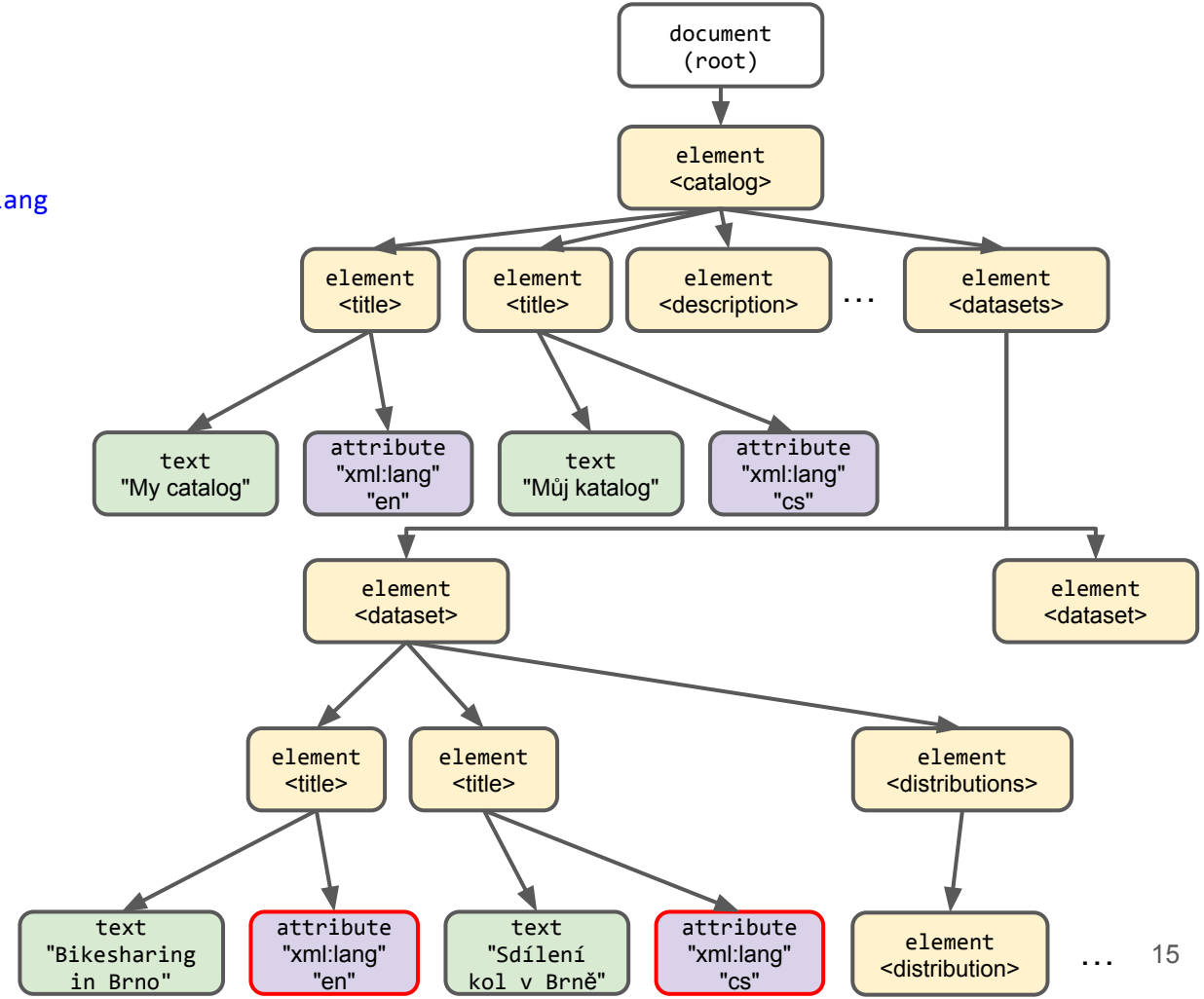
access function
`text()`



XPath - attribute

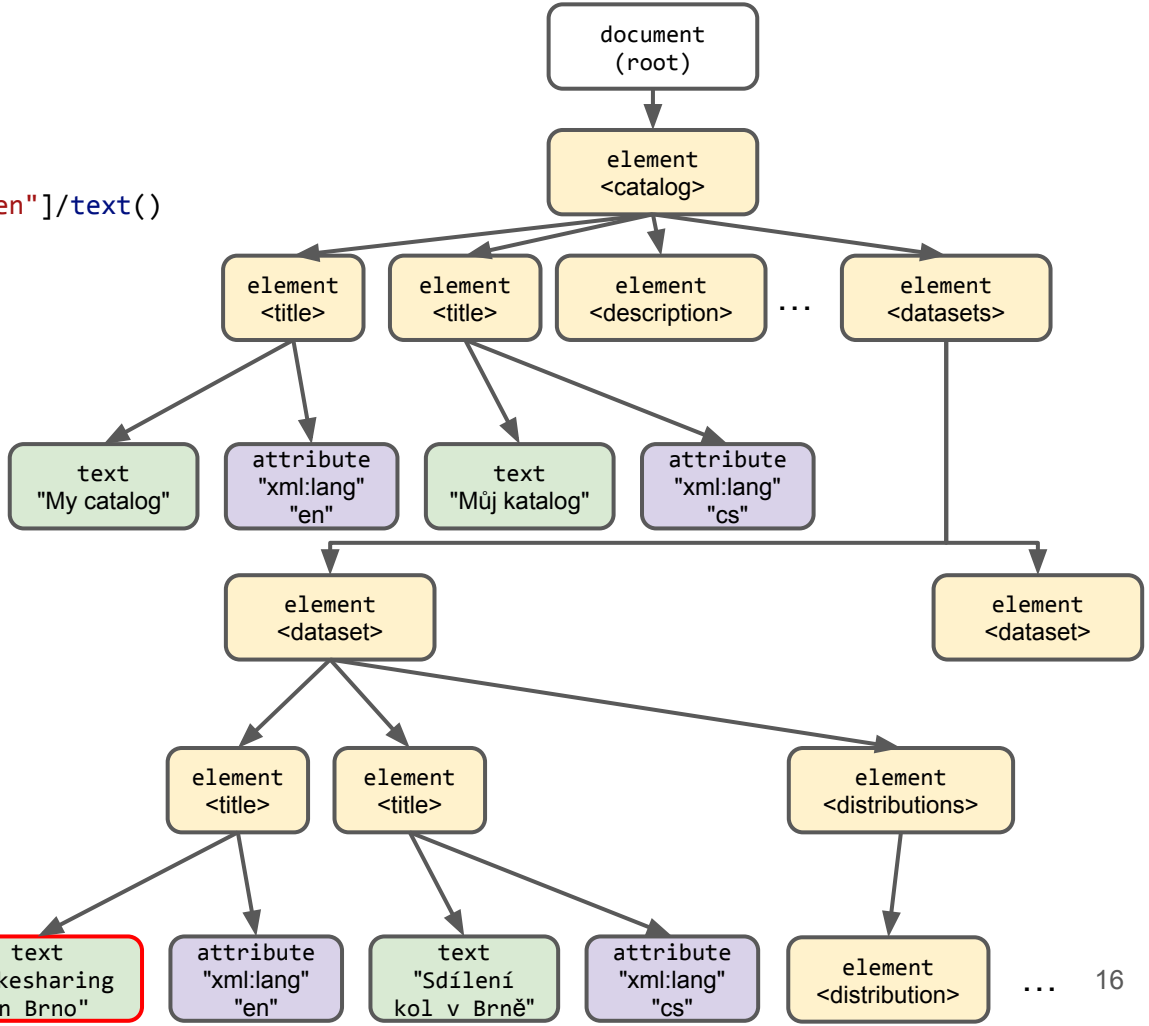
/catalog/datasets/dataset/title/@xml:lang

@attribute



XPath - predicate

```
/catalog/datasets/dataset/title[@xml:lang="en"]/text()
```



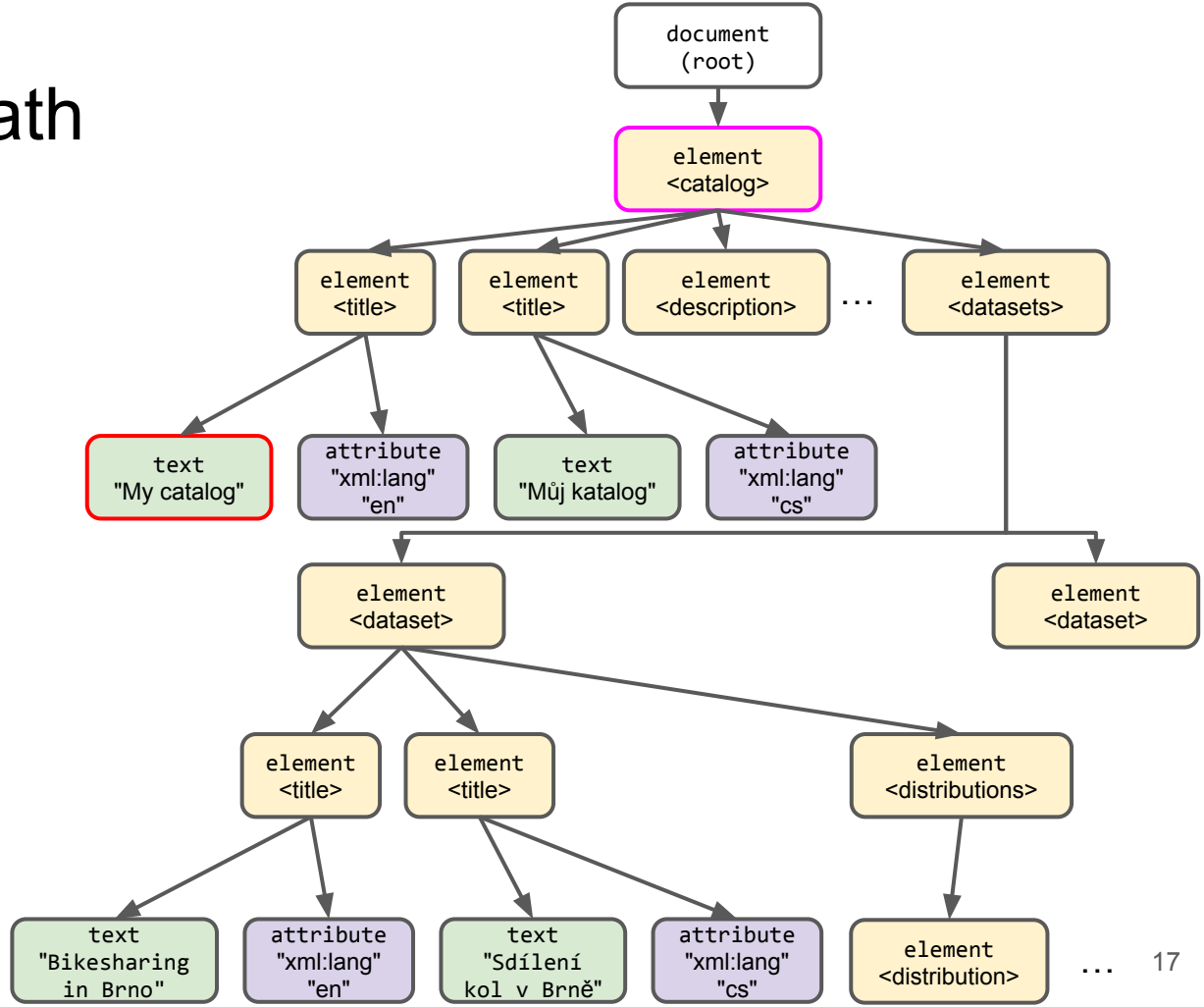
[predicate]

logical expression

XPath - relative path

```
title[@xml:lang="en"]/text()
```

Relative path
`step1/.../stepN`
Needs a starting point



XPath - axes - child

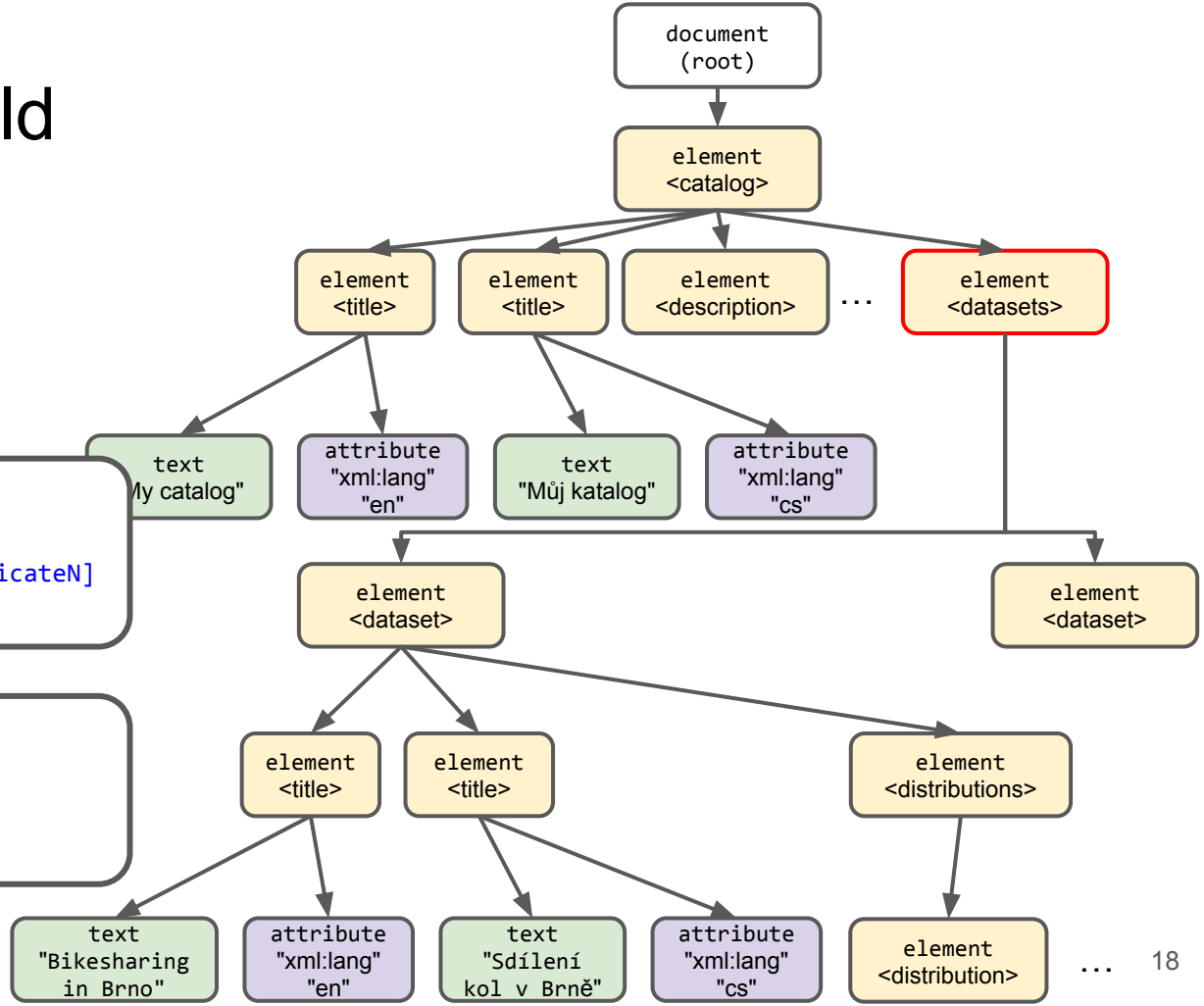
```
/catalog/datasets
/child::catalog/child::datasets
```

XPath path step

```
axis::node-test [predicate1] ... [predicateN]
```

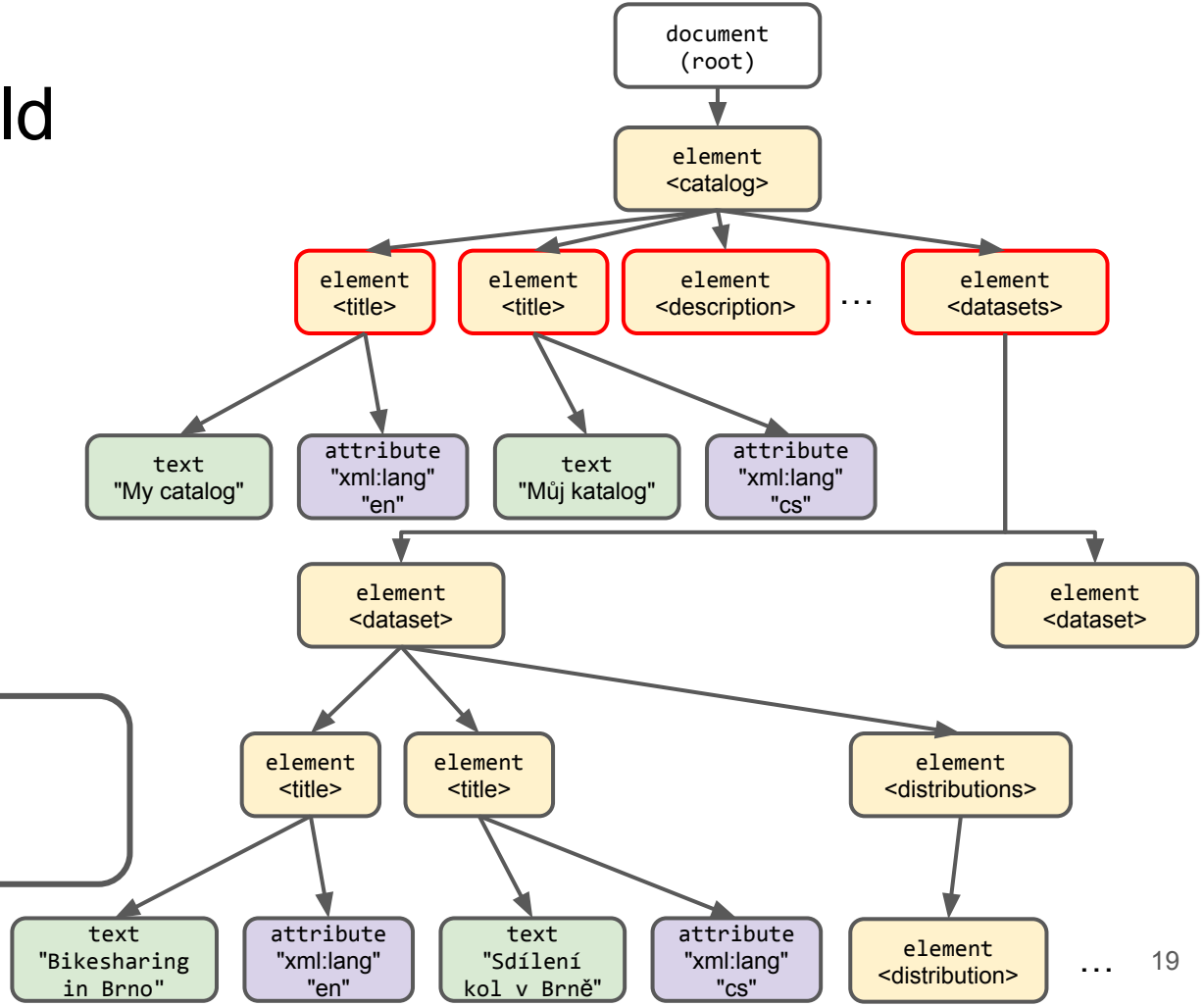
Default axis can be omitted

```
child
```



XPath - axes - child

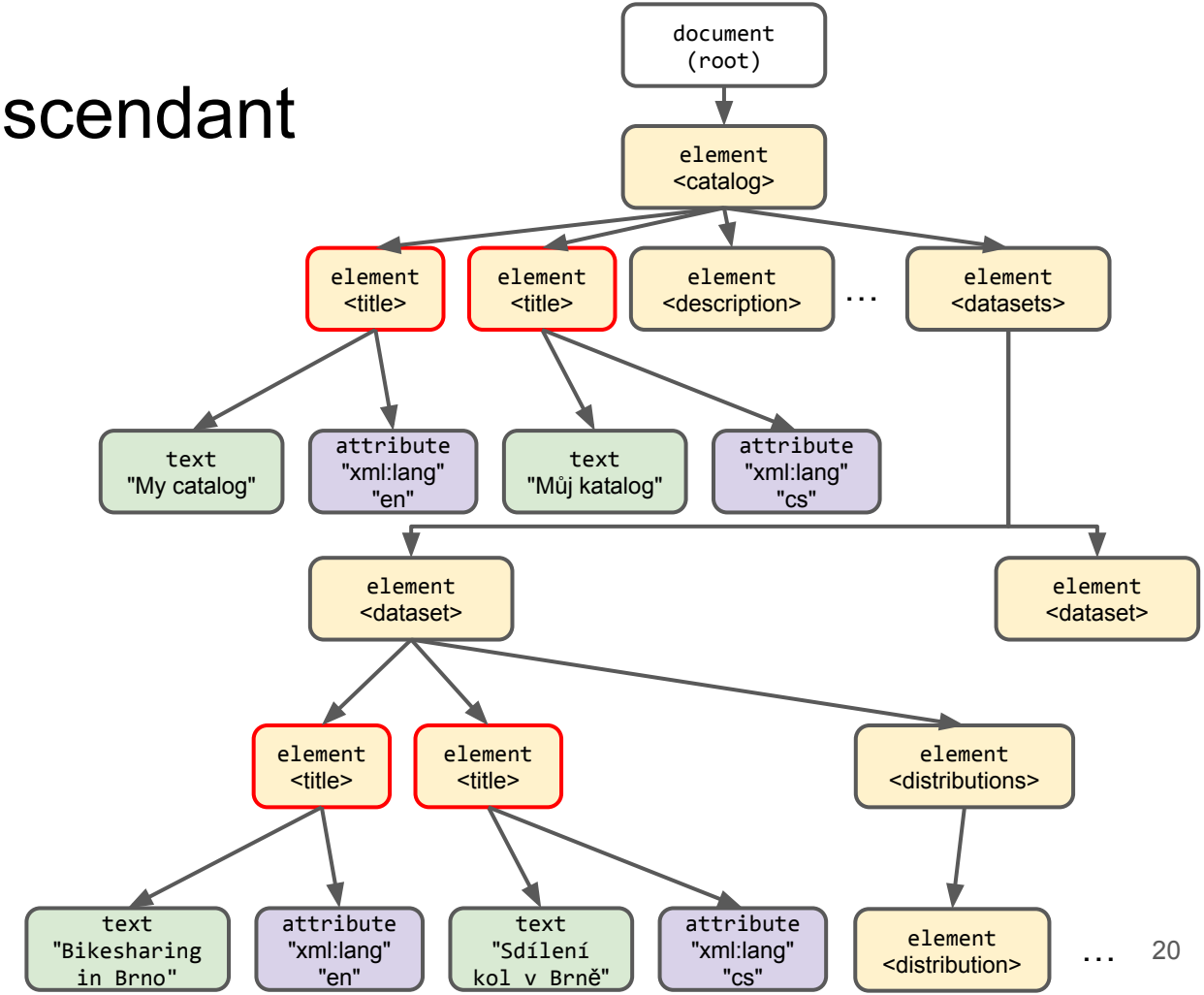
```
/catalog/child::*
/catalog/*
```



Children do not contain attributes or text nodes

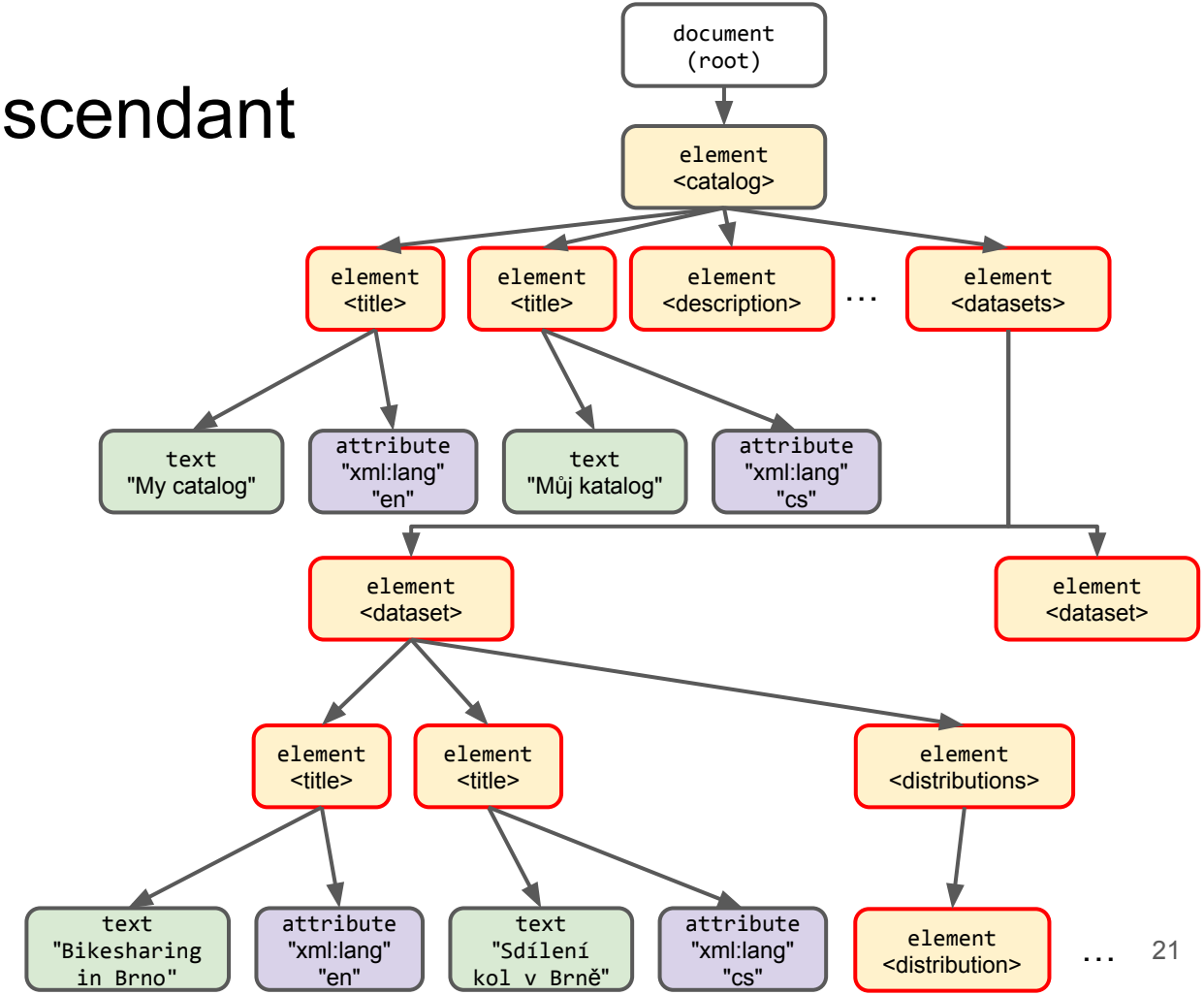
XPath - axes - descendant

```
/catalog/descendant::title
```



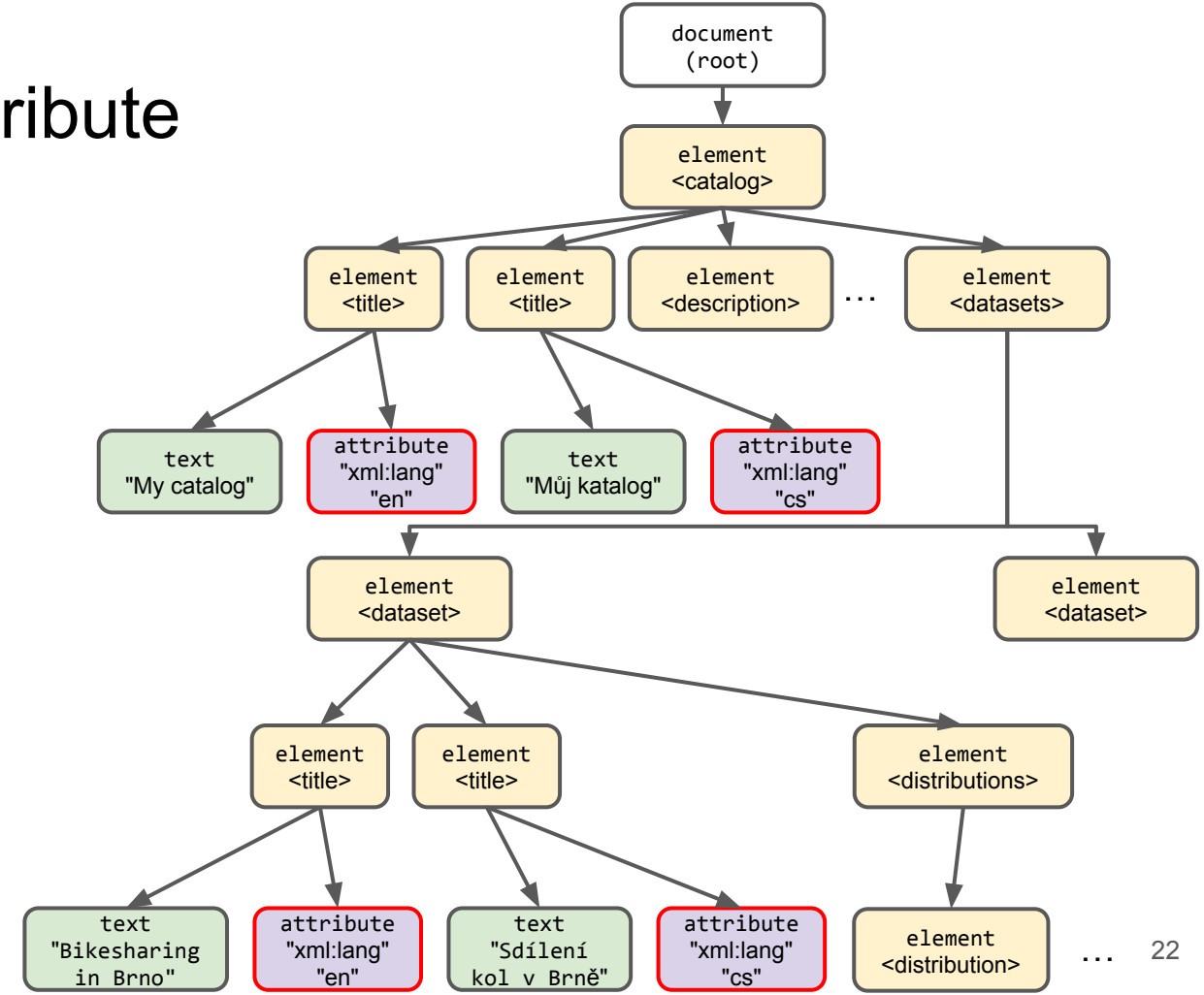
XPath - axes - descendant

```
/catalog/descendant::*
```



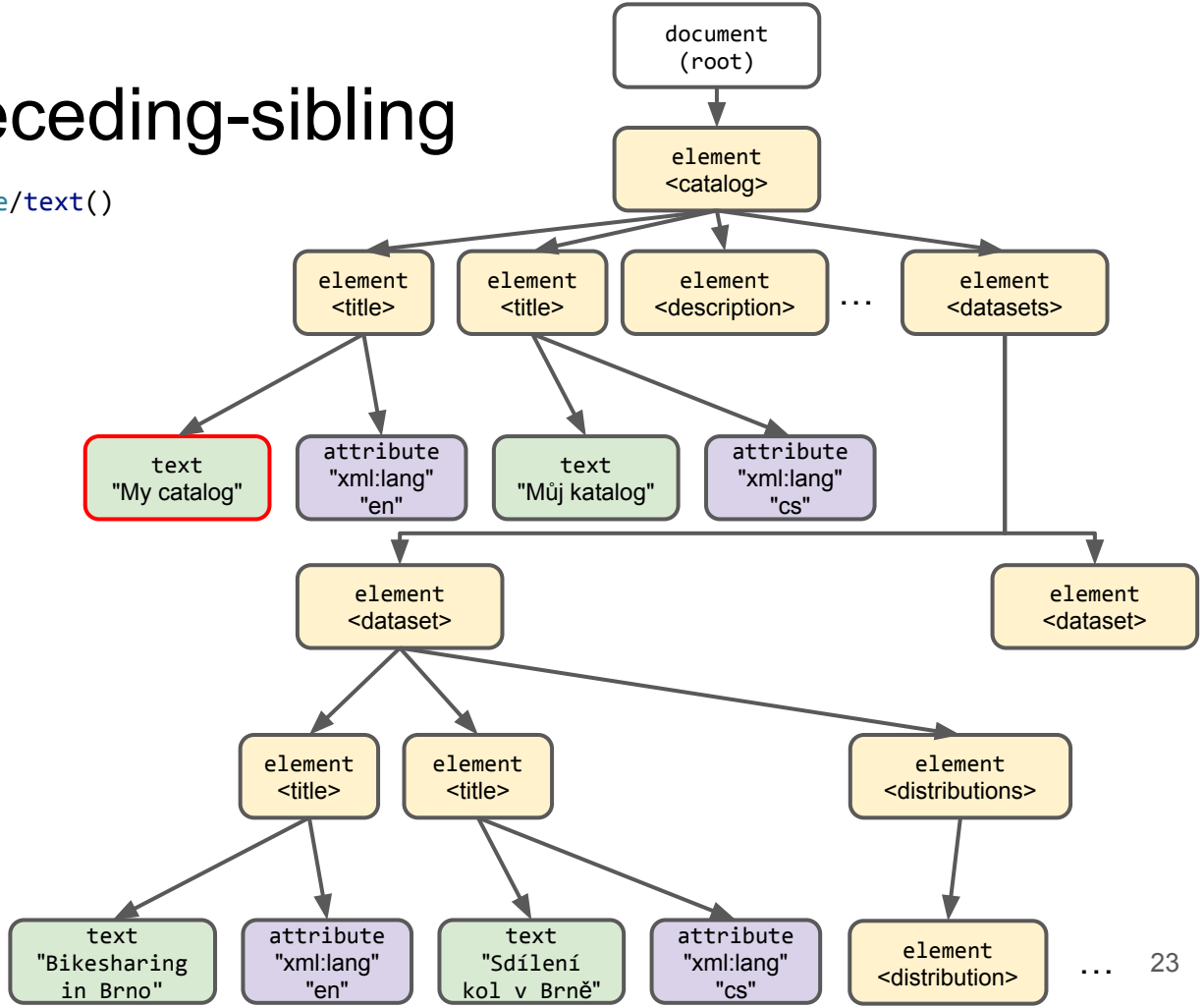
XPath - axes - attribute

```
/catalog/descendant::*/attribute::*
/catalog/descendant::*/@*
```



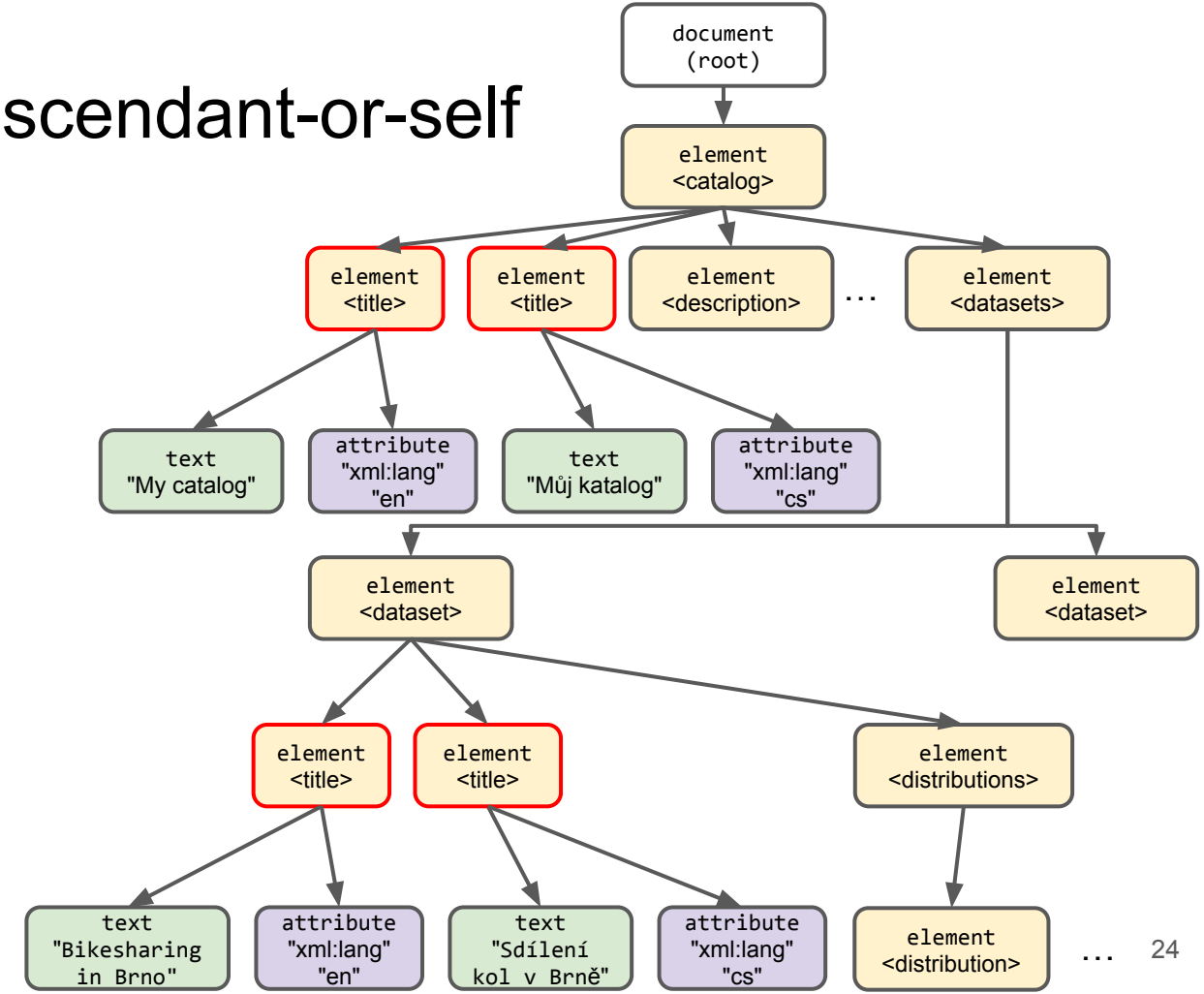
XPath - axes - preceding-sibling

```
/catalog/title/preceding-sibling::title/text()
```



XPath - axes - descendant-or-self

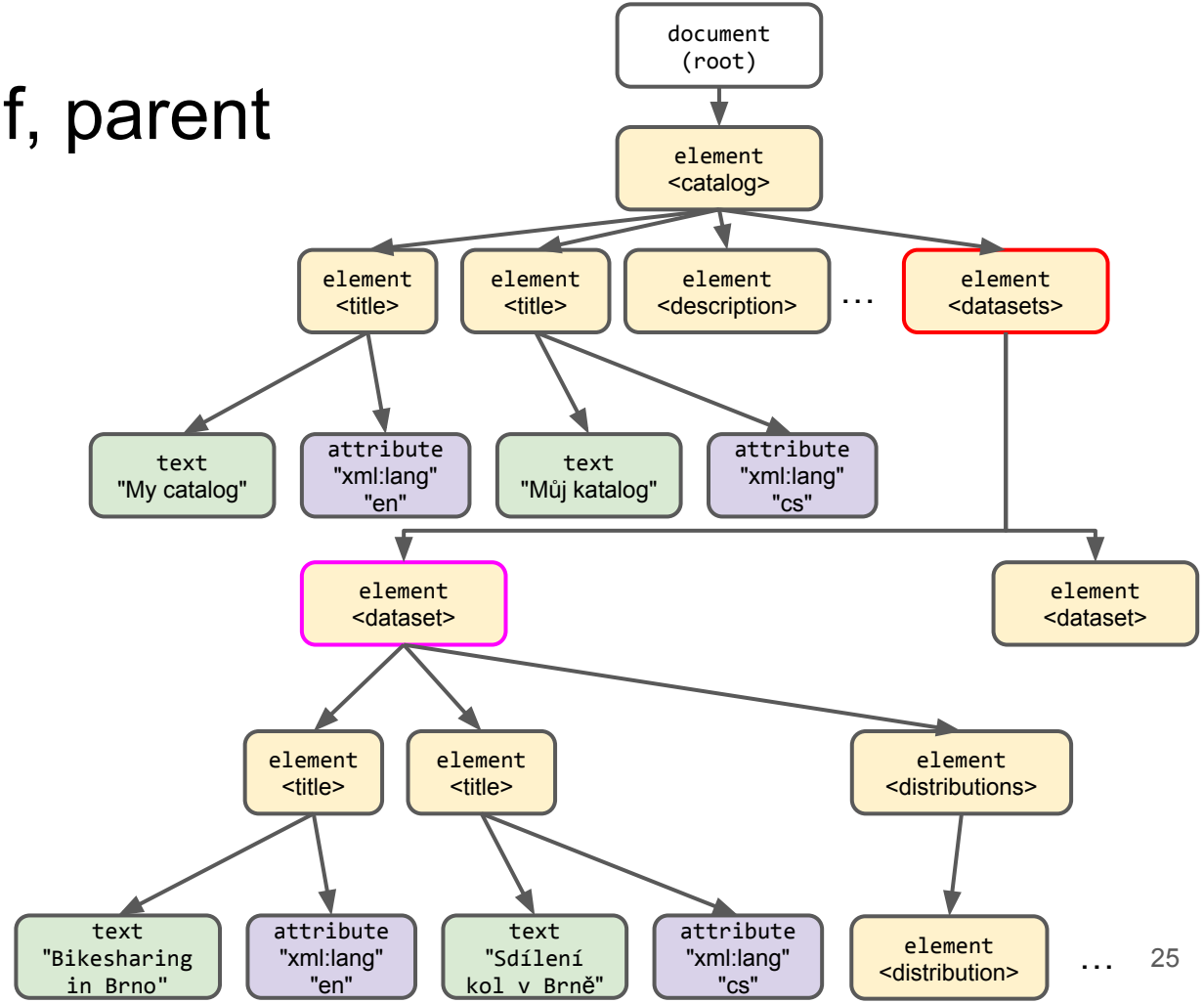
```
/catalog/descendant-or-self::title
/catalog//title
```



XPath - axes - self, parent

Starting in dataset

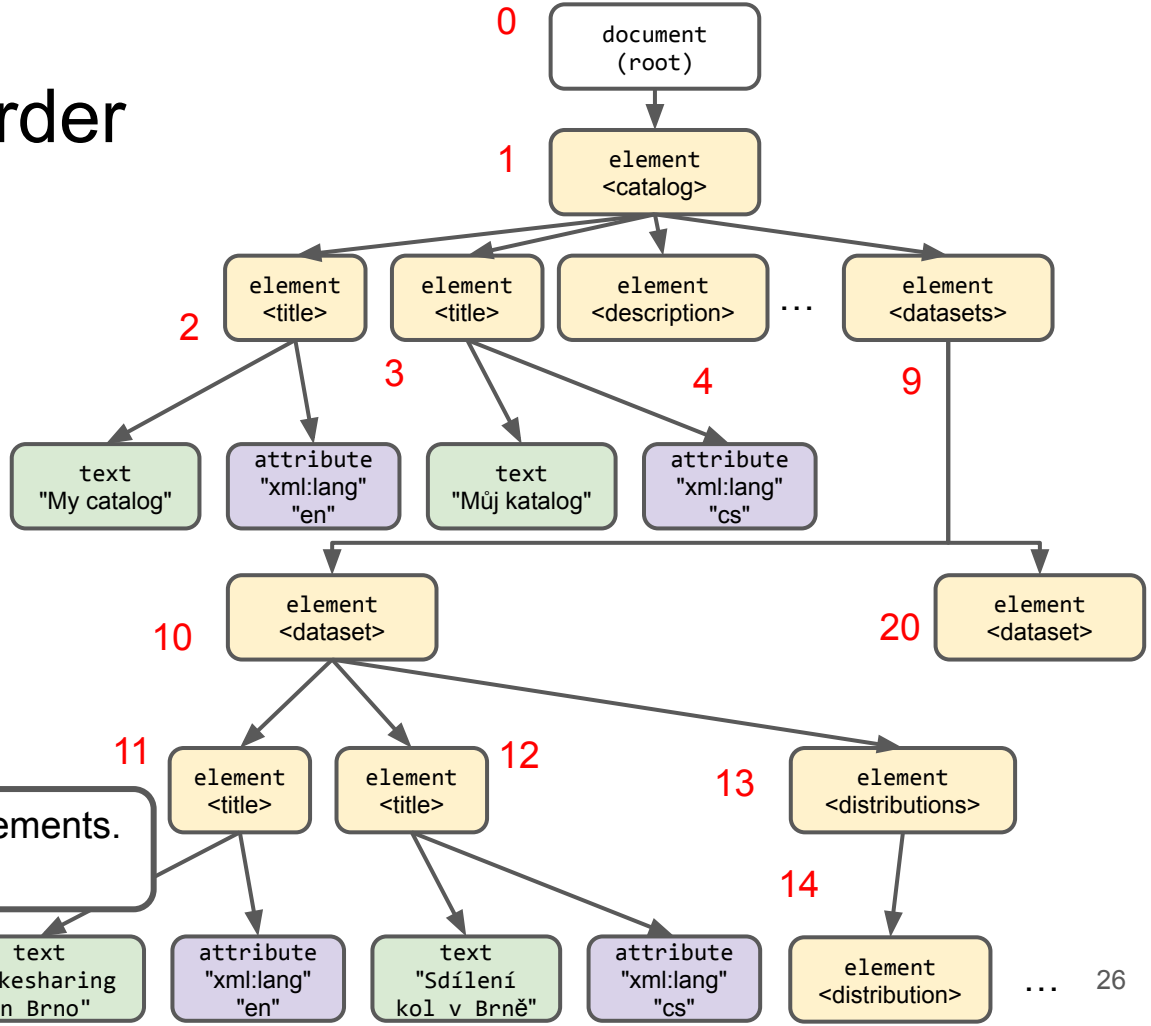
```
.
self::node()

..
parent::node()
```



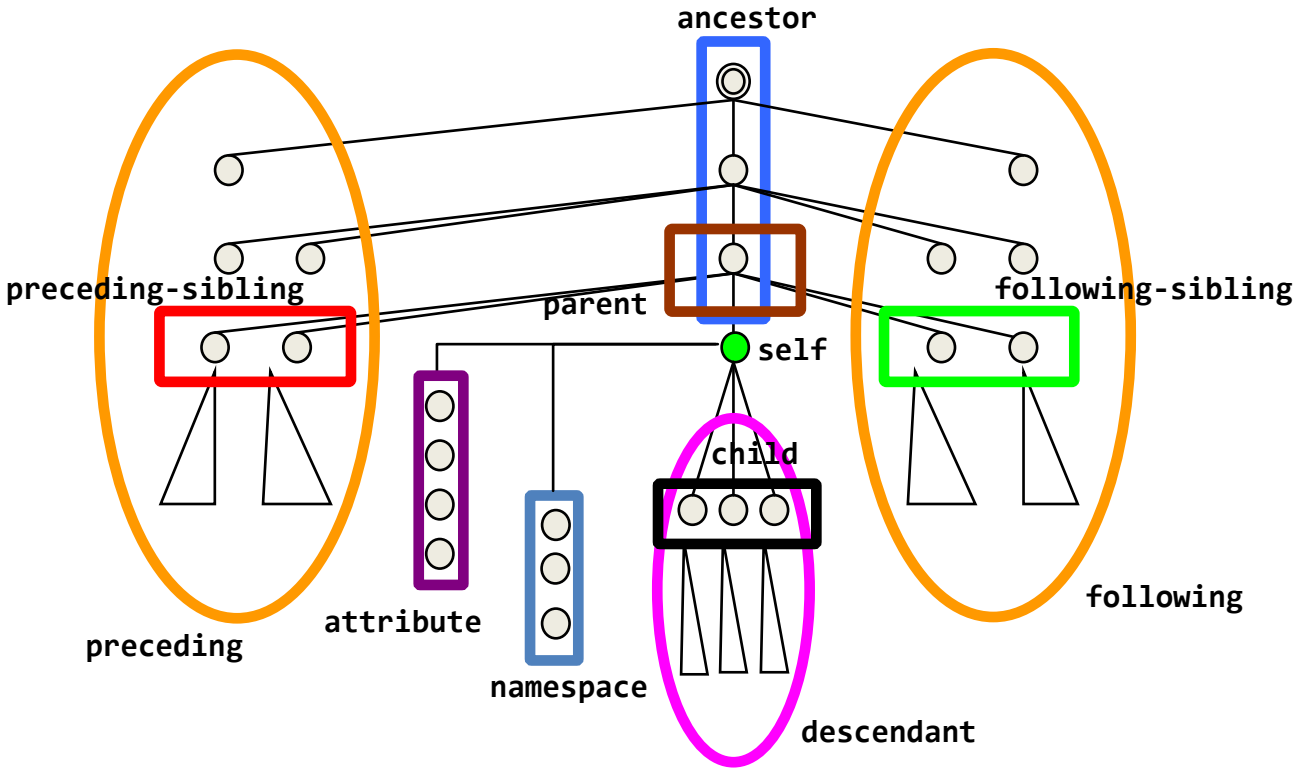
XPath - document order

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title xml:lang="en">My catalog</title>
    <title xml:lang="cs">Můj katalog</title>
    <description xml:lang="en">This is my dummy catalog</description>
    <description xml:lang="cs">Toto je můj falešný katalog</description>
    <contact-point>
        <name xml:lang="en">John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title xml:lang="en">Bikesharing in Brno</title>
            <title xml:lang="cs">Sdílení kol v Brně</title>
            <distributions>
                <distribution>
                    <media-type>application/xml</media-type>
                    <downloadURL>http://brno.cz/myfile.xml</downloadURL>
                </distribution>
                <distribution>
                    <accessService>
                        <endpointURL>https://brno.cz/myAPI</endpointURL>
                        <title xml:lang="en">My API</title>
                    </accessService>
                </distribution>
            </distributions>
        </dataset>
        <dataset>
            <title xml:lang="en">Bikesharing in Prague</title>
            <title xml:lang="cs">Sdílení kol v Praze</title>
            ...
            </distributions>
        </dataset>
    </datasets>
</catalog>
```

according to the position of start tags of elements.

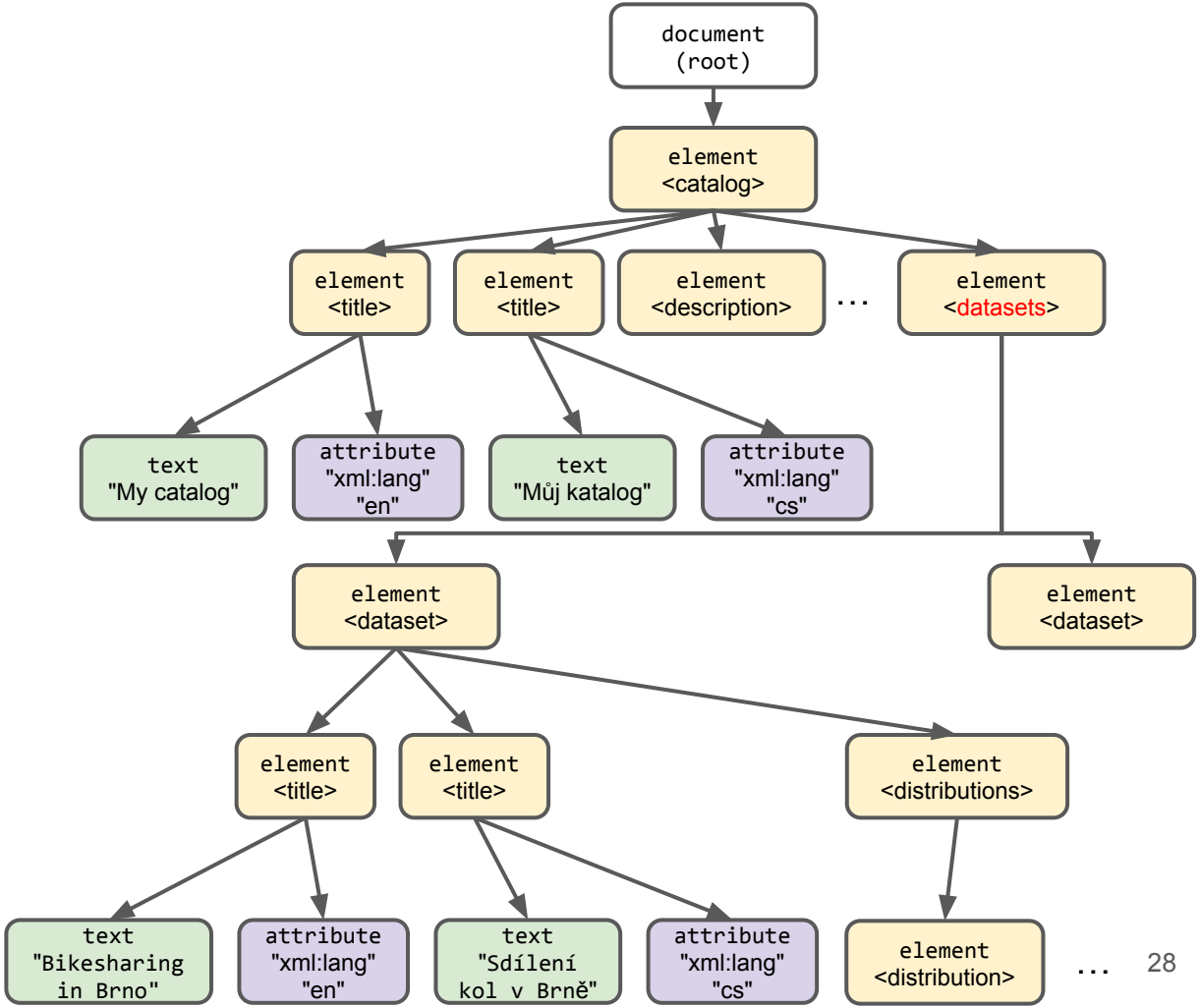


XPath - all axes



XPath - name()

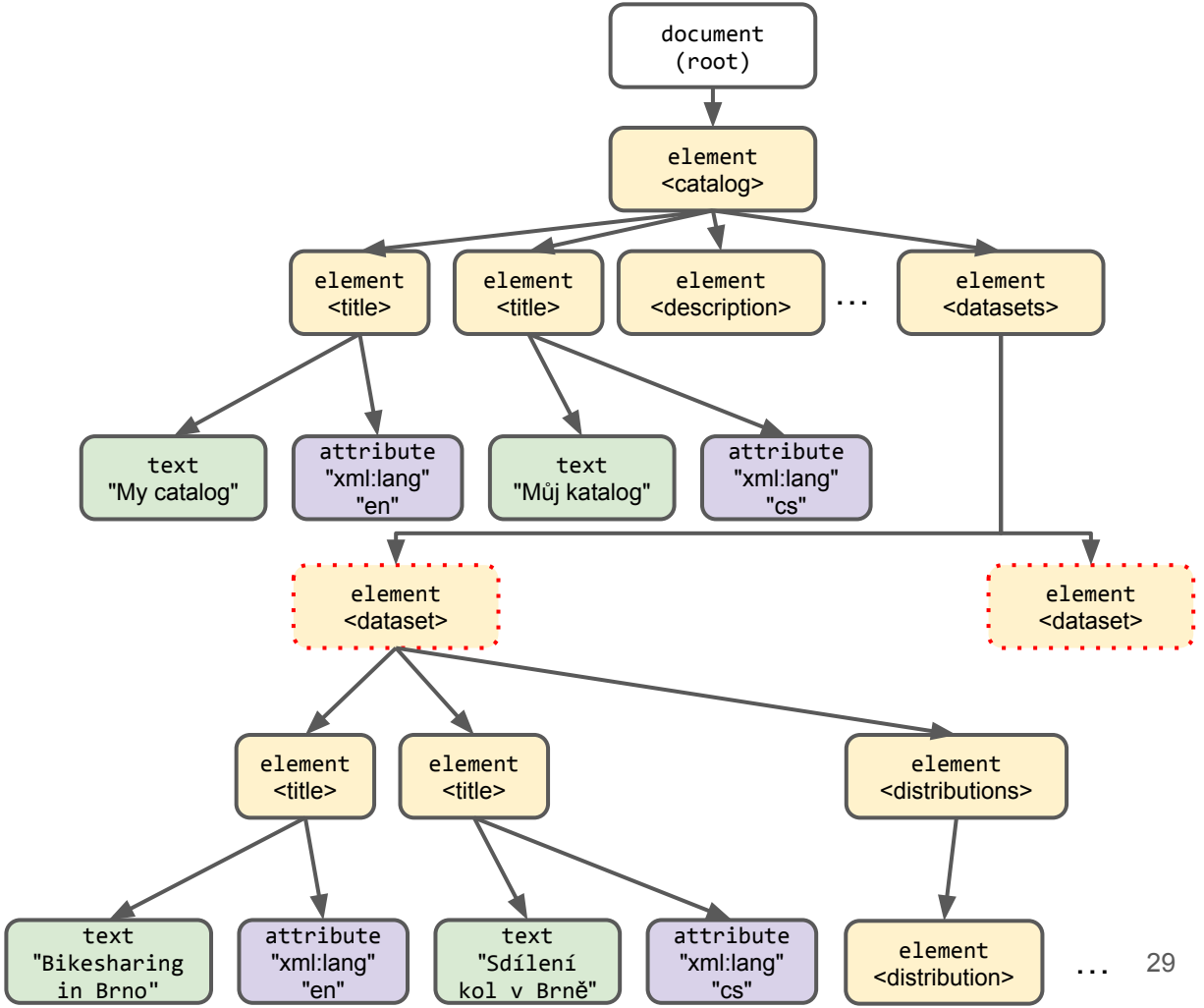
```
/catalog/datasets/name()
```



XPath - position()

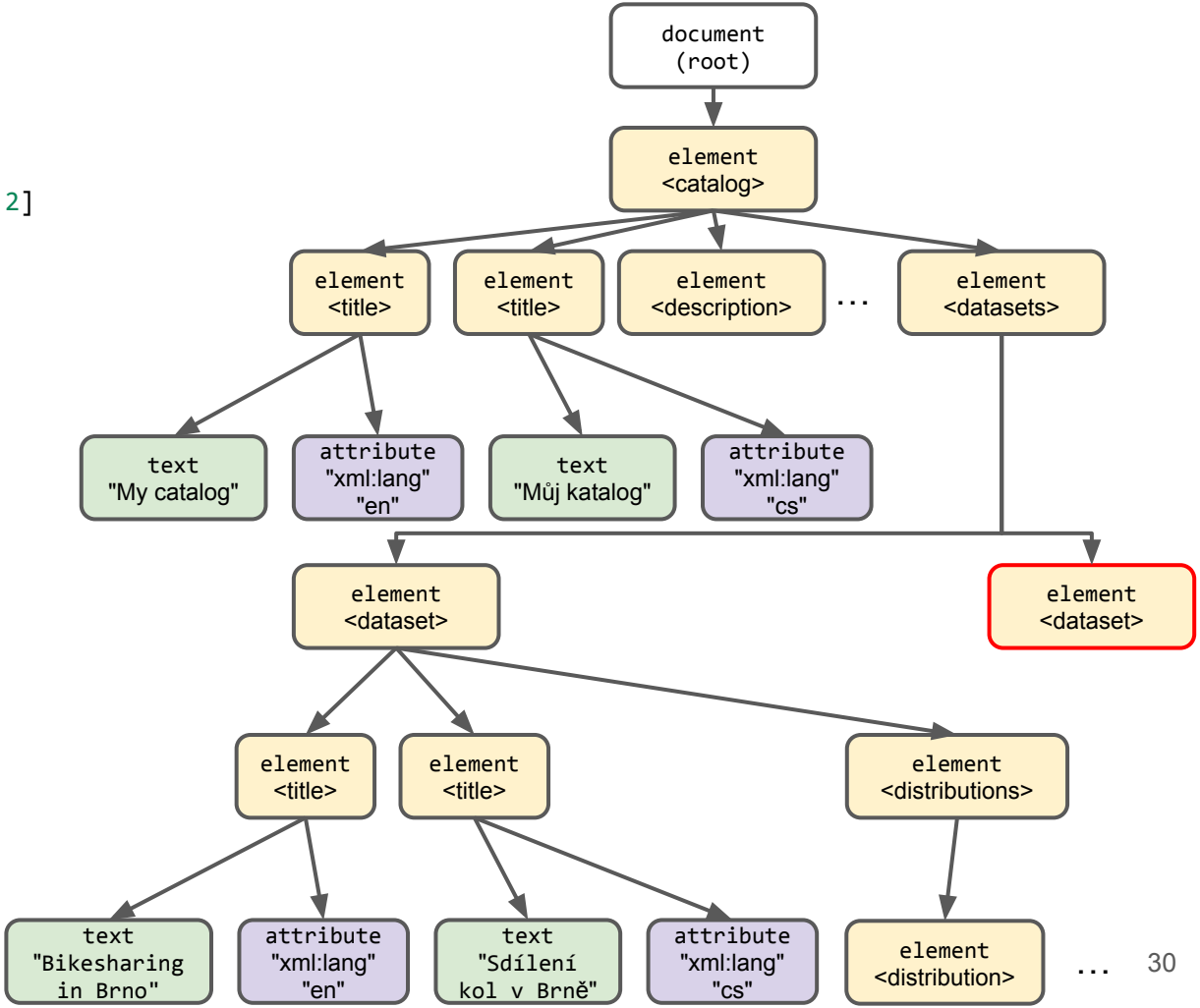
`/catalog/datasets/dataset/position()`

- 1 (xs:integer)
- 2 (xs:integer)



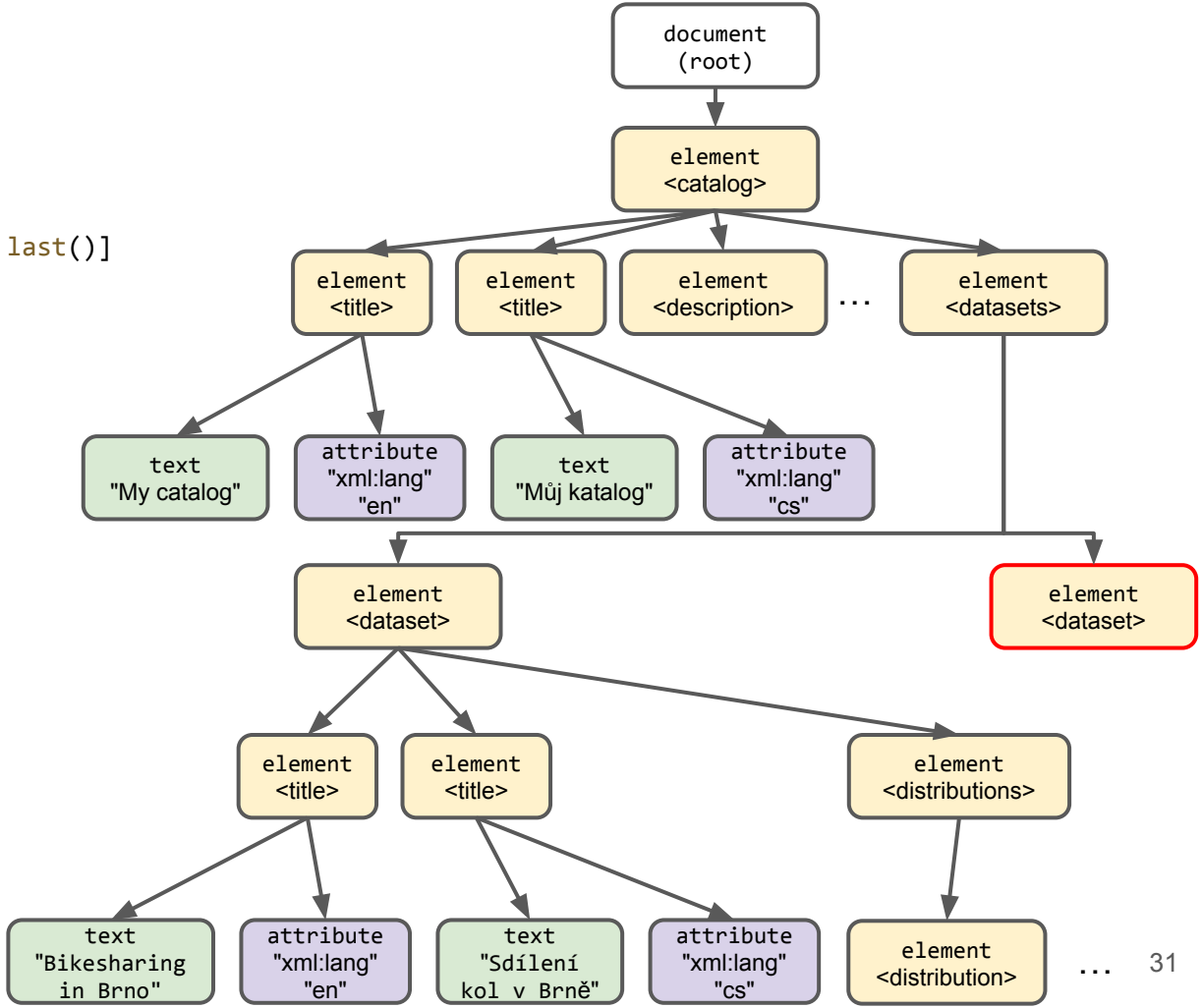
XPath - position()

```
/catalog/datasets/dataset[position() = 2]
/catalog/datasets/dataset[2]
```



XPath - last()

```
/catalog/datasets/dataset[last()]
/catalog/datasets/dataset[position() = last()]
```



Axes
self
child
descendant
parent
ancestor
attribute
following(-siblings)
preceding(-siblings)

Node tests	
name	node with particular name
*	node with arbitrary name
node()	any node
text()	any text node

Functions	
position()	position of node in the result
last()	position of the last node in the result
count()	number of nodes in the result

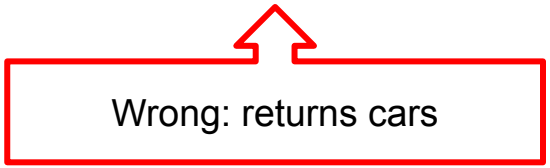
normalize-space()	normalization of white spaces
name()	name of node

Abbreviations	
.../...	.../child::...
.../@...	.../attribute::...
.../....	.../self::node()...
.../.....	.../parent::node()...
.....//...	.../descendant-or-self::node()/...

XPath - common errors

“Select car rental companies in Hawaii which offer at least one cabrio"

```
/rental[state="Hawaii"]/offer/car[type="cabrio"]
```



Correct:

```
/rental[state="Hawaii" and offer/car[type="cabrio"]]
```

XPath - common errors

“Select the last section in the book."

```
//section[last()]
/descendant-or-self::node()/section[last()]
```

Wrong: returns the last section in each chapter / section

Correct: `/descendant::section[last()]`

```
<book>
 <chapter>
  <section>...</section>
 </chapter>
 <chapter>
  <section>
   <section>...</section>
   <section>
    <section>...</section>
   </section>
  </section>
  <section>...</section>
 </chapter>
</book>
```

Some XPath 2.0 features

- result is a sequence (ordered)
  `("a", 2, "c")[3]` results in `"c"`
  `(1 to 10)[7]` results in `7`

- for cycles
```
for $dataset in //dataset return
count($dataset//distribution)
```

- conditional expressions
```
if (count(//dataset) > 1) then
"Datasets" else "Dataset"

//dataset[some $title in title
satisfies $title/@xml:lang="en"]
```

- comments
```
(:comment:)
```

Some XPath 3.0/3.1 features

- mapping operator !

```
//dataset ! count(descendant::distribution)
```

- 1
- 2

- functions chaining (to avoid deep nesting)

```
upper-case(/descendant::dataset[1]/title[1])
/descendant::dataset[1]/title[1] => upper-case()
```

- string concatenation operator ||

```
//dataset[1]/title[@xml:lang="en"] || //dataset[1]/title[@xml:lang="cs"]
concat(//dataset[1]/title[@xml:lang="en"], //dataset[1]/title[@xml:lang="cs"])
```

XSL Transformations - XSLT

XSLT - example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title xml:lang="en">My catalog</title>
    <title xml:lang="cs">Můj katalog</title>
    <description xml:lang="en">This is my dummy catalog</description>
    <description xml:lang="cs">Toto je můj falešný katalog</description>
    <contact-point>
        <name xml:lang="en">John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title xml:lang="en">Bikesharing in Brno</title>
            <title xml:lang="cs">Sdílení kol v Brně</title>
            <distributions>
                <distribution>
                    <media-type>application/xml</media-type>
                    <downloadURL>http://brno.cz/myfile.xml</downloadURL>
                </distribution>
                <distribution>
                    <accessService>
                        <endpointURL>https://brno.cz/myAPI</endpointURL>
                        <title xml:lang="en">My API</title>
                    </accessService>
                </distribution>
            </distributions>
        </dataset>
        <dataset>
            <title xml:lang="en">Bikesharing in Prague</title>
            <title xml:lang="cs">Sdílení kol v Praze</title>
            <distributions>
                <distribution>
                    <title xml:lang="en">CSV</title>
                    <media-type>text/csv</media-type>
                    <downloadURL>http://praha.eu/myfile.csv</downloadURL>
                </distribution>
            </distributions>
        </dataset>
    </datasets>
</catalog>
```


```xml
<html xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
        <h2>Bikesharing in Brno</h2>
        <p>Number of distributions: 2</p>
        <h2>Bikesharing in Prague</h2>
        <p>Number of distributions: 1</p>
    </body>
</html>
```

My catalog

Bikesharing in Brno

Number of distributions: 2

Bikesharing in Prague

Number of distributions: 1

XSLT - example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title xml:lang="en">My catalog</title>
    <title xml:lang="cs">Můj katalog</title>
    <description xml:lang="en">This is my dummy catalog</description>
    <description xml:lang="cs">Toto je můj falešný katalog</description>
    <contact-point>
        <name xml:lang="en">John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title xml:lang="en">Bikesharing in Brno</title>
            <title xml:lang="cs">Sdílení kol v Brně</title>
            <distributions>
                <distribution>
                    <media-type>application/xml</media-type>
                    <downloadURL>http://brno.cz/myfile.xml</downloadURL>
                </distribution>
                <distribution>
                    <accessService>
                        <endpointURL>https://brno.cz/myAPI</endpointURL>
                        <title xml:lang="en">My API</title>
                    </accessService>
                </distribution>
            </distributions>
        </dataset>
        <dataset>
            <title xml:lang="en">Bikesharing in Prague</title>
            <title xml:lang="cs">Sdílení kol v Praze</title>
            <distributions>
                <distribution>
                    <title xml:lang="en">CSV</title>
                    <media-type>text/csv</media-type>
                    <downloadURL>http://praha.eu/myfile.csv</downloadURL>
                </distribution>
            </distributions>
        </dataset>
    </datasets>
</catalog>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="http://www.w3.org/2005/xpath-functions">
    <xsl:output method="html" encoding="UTF-8" indent="yes"/>
    <xsl:template match="catalog">
        <html>
            <head>
                <title>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </title>
            </head>
            <body>
                <h1>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </h1>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="dataset">
        <h2>
            <xsl:value-of select="title[@xml:lang='en']"/>
        </h2>
        <p>Number of distributions: <xsl:value-of select="count(descendant::distribution)"/>
        </p>
    </xsl:template>

    <xsl:template match="text()">
        <xsl:apply-templates/>
    </xsl:template>
</xsl:stylesheet>
```

XSLT - Specifications

- XSL Transformations (XSLT) Version 1.0
  - W3C Recommendation, **1999**
  - what we will cover mostly
- XSL Transformations (XSLT) Version 2.0
  - W3C Recommendation, **2007**
  - most widely implemented
- XSL Transformations (XSLT) Version 3.0
  - W3C Recommendation, **2017**

XSLT principles - stylesheet, template, processor

Input

- one or more XML documents

Output

- one or more text files
  - XML, HTML
  - RDF Turtle
  - TXT
  - ...

XSLT **stylesheet**

- is an XML document
- `stylesheet` root element
- set of `templates`

XSLT **template**

- matches part of input XML document using XPath expressions
- produces output text

XSLT **processor**

- goes through an input XML document
- tries to match templates

XSLT - empty stylesheet

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />

</xsl:stylesheet>
```

`version` attribute - version of XSLT used

`xsl:output` - specifies the output behavior of the XSLT processor

- method
  - `html`, `xhtml`, `xml` - produces well-formed documents
  - `text` - pure text output
- indent
  - `yes` - generates correct indentation for `xml`, `html`
  - `no` - only explicitly generated whitespace included in output

XSLT - first template

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">
      <html>
        <head>
            <title>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </title>
        </head>
        <body>
            <h1>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </h1>
        </body>
      </html>
    </xsl:template>

</xsl:stylesheet>
```

`match` - contains XPath expression which needs to match

`xsl:template`

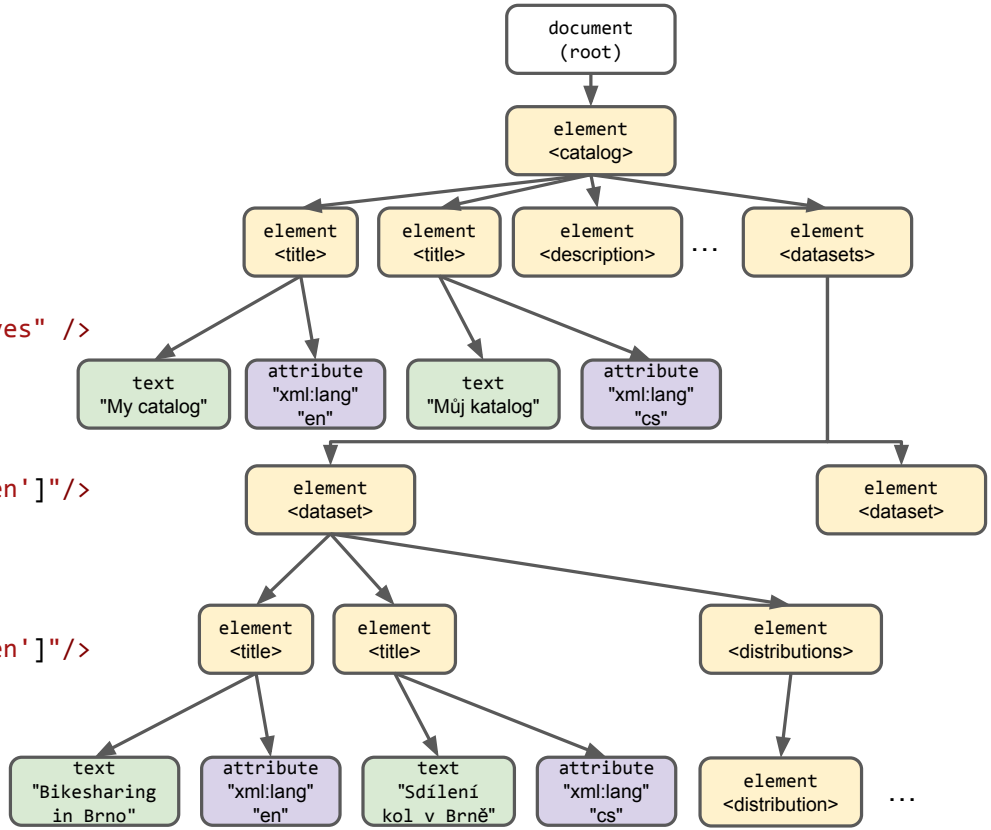
- content goes to the output
  - here, we generate the HTML stub
- `xsl:` elements get processed

e.g `xsl:value-of`

- `select` attribute contains XPath expression
- result of the expression replaces the `xsl:value-of` element in the output

XSLT - first template

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">
      <html>
        <head>
            <title>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </title>
        </head>
        <body>
            <h1>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </h1>
        </body>
      </html>
    </xsl:template>

</xsl:stylesheet>
```



XSLT - first template

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">
      <html>
        <head>
            <title>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </title>
        </head>
        <body>
            <h1>
                <xsl:value-of select="title[@xml:lang='en']"/>
            </h1>
        </body>
      </html>
    </xsl:template>

</xsl:stylesheet>
```

```xml
<html
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
    </body>
</html>
```

- output is indented
- whitespace is normalized
- encoding indicated in the head

XSLT - second template

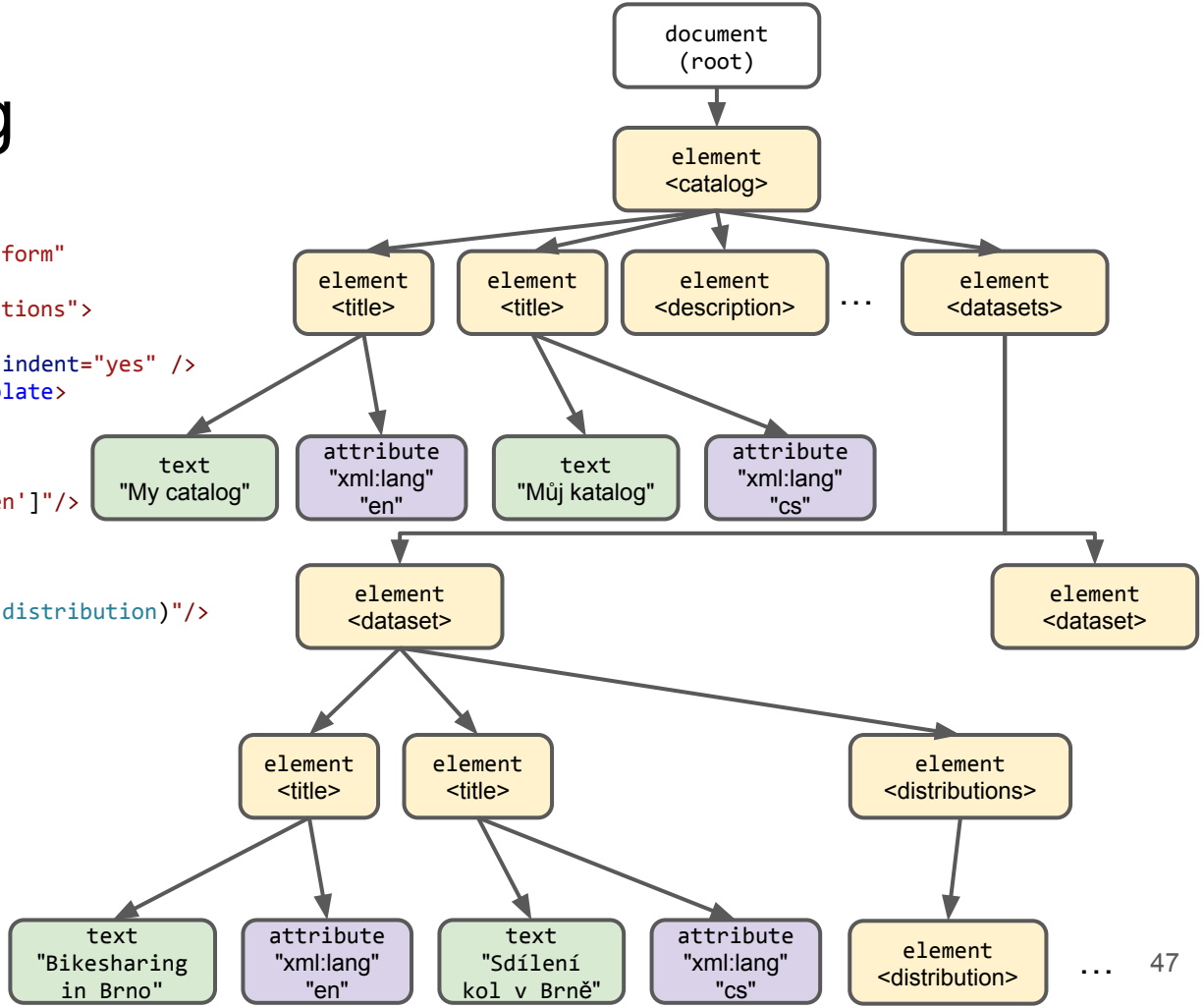
```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">...</xsl:template>

    <xsl:template match="dataset">
        <h2>
            <xsl:value-of select="title[@xml:lang='en']"/>
        </h2>
        <p>
            Number of distributions:
            <xsl:value-of select="count(descendant::distribution)"/>
        </p>
    </xsl:template>

</xsl:stylesheet>
```

```xml
<html
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
    </body>
</html>
```

Nothing new in the output… why?

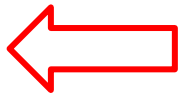
XSLT - processing

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions">

  <xsl:output method="html" encoding="UTF-8" indent="yes" />
  <xsl:template match="catalog">...</xsl:template>

  <xsl:template match="dataset">
    <h2>
     <xsl:value-of select="title[@xml:lang='en']"/>
    </h2>
    <p>
     Number of distributions:
     <xsl:value-of select="count(descendant::distribution)"/>
    </p>
  </xsl:template>

</xsl:stylesheet>
```

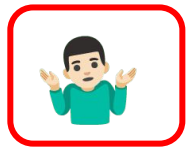


XSLT - apply templates

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
     <xsl:template match="catalog">
       <html>
         <head>
           <title>
             <xsl:value-of select="title[@xml:lang='en']"/>
           </title>
         </head>
         <body>
           <h1>
             <xsl:value-of select="title[@xml:lang='en']"/>
           </h1>
           <xsl:apply-templates/>
         </body>
      </html>
     </xsl:template>

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

</xsl:stylesheet>
```

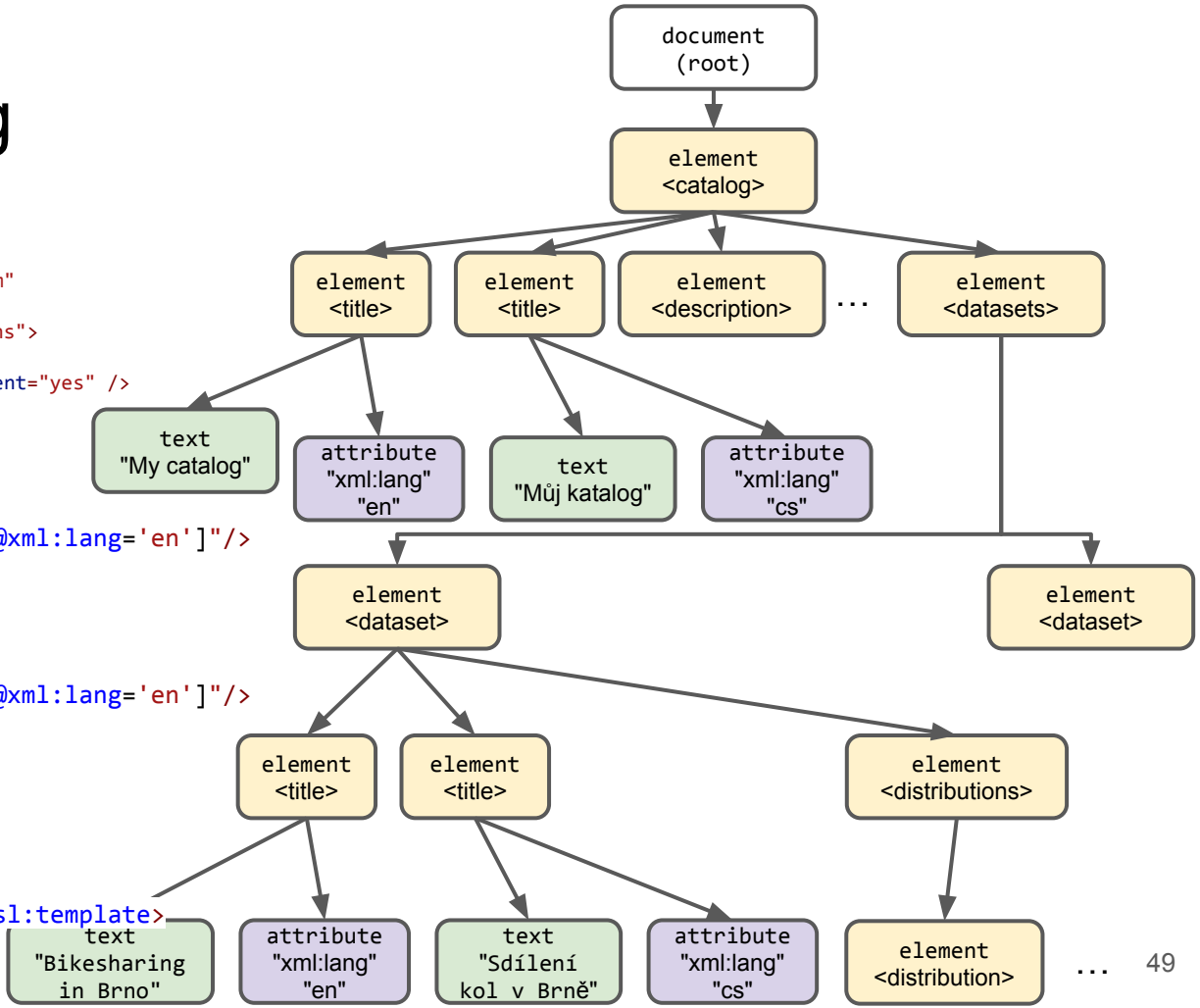


```
<html
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
    My catalog
    Můj katalog
    This is my dummy catalog
    Toto je můj falešný katalog

        John Doe
        mailto:john@doe.org


        <h2>Bikesharing in Brno</h2><p>Number of
distributions: 2</p>
        <h2>Bikesharing in Prague</h2><p>Number of
distributions: 1</p>

</body>
</html>
```



XSLT - processing

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:output method="html" encoding="UTF-8" indent="yes" />
     <xsl:template match="catalog">
       <html>
         <head>
           <title>
             <xsl:value-of select="title[@xml:lang='en']"/>
           </title>
         </head>
         <body>
           <h1>
             <xsl:value-of select="title[@xml:lang='en']"/>
           </h1>
           <xsl:apply-templates/>
         </body>
      </html>
     </xsl:template>

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

</xsl:stylesheet>
```



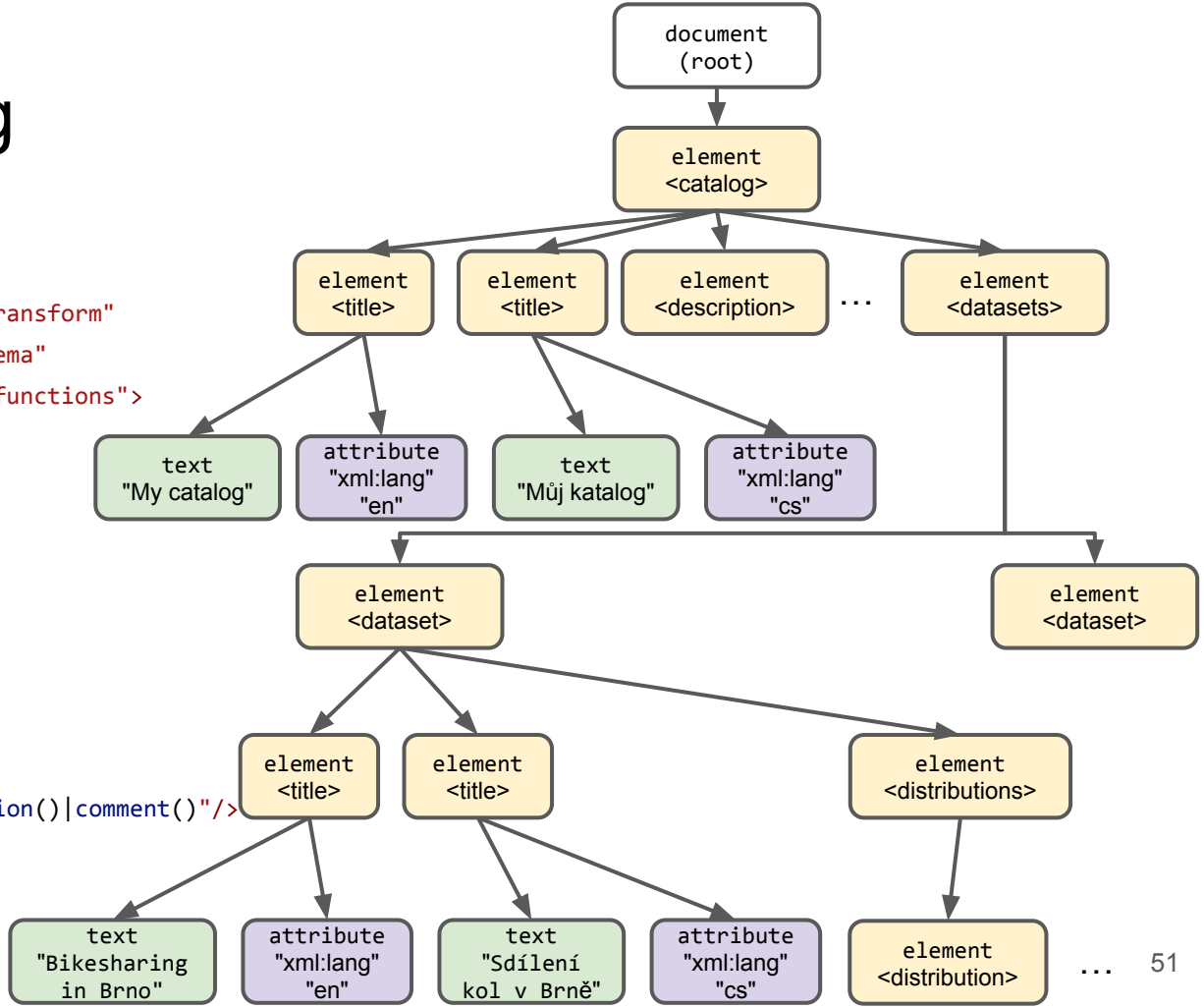
XSLT - implicit templates

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">

    <xsl:template match="*|/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="text()|@*">
        <xsl:value-of select="."/>
    </xsl:template>

    <xsl:template match="processing-instruction()|comment()"/>

</xsl:stylesheet>
```

- Present implicitly - need to be overridden
- Result in text from elements and attributes to be copied to output

XSLT - processing

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">

  <xsl:template match="*|/">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="text()|@*">
    <xsl:value-of select="."/>
  </xsl:template>

  <xsl:template match="processing-instruction()|comment()"/>

</xsl:stylesheet>
```



XSLT - implicit templates

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">
        <html>
            <head>
                <title>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </title>
            </head>
            <body>
                <h1>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </h1>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="dataset">
        <h2>
            <xsl:value-of select="title[@xml:lang='en']"/>
        </h2>
        <p>Number of distributions: <xsl:value-of select="count(descendant::distribution)"/>
        </p>
    </xsl:template>

    <xsl:template match="text()"/>
</xsl:stylesheet>
```

override the implicit template

```
<html
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
        <h2>Bikesharing in Brno</h2>
        <p>Number of distributions: 2</p>
        <h2>Bikesharing in Prague</h2>
        <p>Number of distributions: 1</p>
    </body>
</html>
```

XSLT - apply templates - select which

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
    <xsl:output method="html" encoding="UTF-8" indent="yes" />
    <xsl:template match="catalog">
        <html>
            <head>
                <title>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </title>
            </head>
            <body>
                <h1>
                    <xsl:value-of select="title[@xml:lang='en']"/>
                </h1>
                <xsl:apply-templates select="datasets/dataset"/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="dataset">
        <h2>
            <xsl:value-of select="title[@xml:l
        </h2>
        <p>Number of distributions: <xsl:value
        </p>
    </xsl:template>

    <xsl:template match="text()"/>
</xsl:stylesheet>
```

XPath selecting nodes to which the templates will be applied next.
Default: `child::node()`

```xml
<html
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <head>
        <title>My catalog</title>
    </head>
    <body>
        <h1>My catalog</h1>
        <h2>Bikesharing in Brno</h2>
        <p>Number of distributions: 2</p>
        <h2>Bikesharing in Prague</h2>
        <p>Number of distributions: 1</p>
    </body>
</html>
```

XSLT - named templates and parameters

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions">
     <xsl:output method="html" encoding="UTF-8" indent="yes" />
     <xsl:template match="catalog">
         <html>
                 <head>
                     <xsl:call-template name="processTitle">
                         <xsl:with-param name="element">title</xsl:with-param>
                         <xsl:with-param name="lang">cs</xsl:with-param>
                     </xsl:call-template>
                 </head>
                 <body>
                     <xsl:call-template name="processTitle">
                         <xsl:with-param name="element">h1</xsl:with-param>
                         <xsl:with-param name="lang">en</xsl:with-param>
                     </xsl:call-template>
                     <xsl:apply-templates select="datasets/dataset"/>
                 </body>
         </html>
     </xsl:template>

     <xsl:template match="dataset">
           <xsl:call-template name="processTitle">
               <xsl:with-param name="element">h2</xsl:with-param>
               <xsl:with-param name="lang">en</xsl:with-param>
           </xsl:call-template>
          <p>Number of distributions: <xsl:value-of select="count(descendant::distribution)"/>
          </p>
     </xsl:template>

     <xsl:template match="text()"/>

      <xsl:template name="processTitle">
          <xsl:param name="element" required="yes"/>
          <xsl:param name="lang" required="yes"/>
          <xsl:element name="{$element}">
              <xsl:value-of select="title[@xml:lang=$lang]"/>
          </xsl:element>
      </xsl:template>

</xsl:stylesheet>
```

Named templates

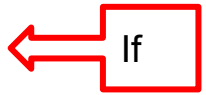
- `name` attribute instead of `match` attribute
- accept parameters
  - `xsl:param` - definition in named template
  - `$variable` - access to variable value in XPath
  - `{$variable}` - access to variable value elsewhere
- called using `xsl:call-template`
  - does not change the currently processed node set
  - xsl:with-param - values passed when calling

`xsl:element`

- creates an element on the output
- name can be constant or `{$variable}`

XSLT - global variables, modes, if

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
    <xsl:output method="html" encoding="UTF-8" indent="yes" />
      <xsl:variable name="lang">en</xsl:variable>

    <xsl:template match="catalog">
        <html>
            <head>
                    <xsl:apply-templates mode="head"/>
            </head>
            <body>
                    <xsl:apply-templates mode="catalog"/>
            </body>
        </html>
    </xsl:template>

     <xsl:template match="dataset" mode="head"/>

     <xsl:template match="dataset" mode="catalog">
         <xsl:apply-templates mode="dataset"/>
       <p>Number of distributions: <xsl:value-of select="count(descendant::distribution)"/>
       </p>
    </xsl:template>

     <xsl:template match="title" mode="head">
         <xsl:if test="@xml:lang=$lang">
             <xsl:element name="title">
                 <xsl:value-of select="text()"/>
             </xsl:element>
         </xsl:if>
     </xsl:template>

     <xsl:template match="title" mode="catalog">
         <xsl:if test="@xml:lang=$lang">
             <xsl:element name="h1">
                 <xsl:value-of select="text()"/>
             </xsl:element>
         </xsl:if>
     </xsl:template>
...
     <xsl:template match="text()" mode="#all"/>
</xsl:stylesheet>
```



Global variable

- defined in the `xsl:stylesheet` root element using `xsl:variable`
- accessible in the whole stylesheet
  - e.g. `$lang`

Mode

- ability to process the same nodes in different ways
  - different templates with the same `match`
- specified in `xsl:apply-templates`
- used in unnamed `xsl:template`
  - `#all` matches all modes

Some remaining XSLT 1.0 features

"Switch"

```
<xsl:choose>
    <xsl:when test='$level=1'>
        <xsl:number format="i"/>
    </xsl:when>
    <xsl:when test='$level=2'>
        <xsl:number format="a"/>
    </xsl:when>
    <xsl:otherwise>
        <xsl:number format="1"/>
    </xsl:otherwise>
</xsl:choose>
```

For each

```
<xsl:for-each select="item">
    <xsl:sort select="."/>
    <p>
      <xsl:number value="position()" format="1. "/>
      <xsl:value-of select="."/>
    </p>
</xsl:for-each>
```

Include - XML-based inclusion

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

Import - templates in importing stylesheet take precedence over imported templates

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

Some XSLT 2.0 and 3.0 features

Grouping of data

```
<xsl:for-each-group select="cities/city" group-by="@country">
    <tr>
      <td><xsl:value-of select="@country"/></td>
      <td>
        <xsl:value-of select="current-group()/@name" separator=", "/>
      </td>
      <td><xsl:value-of select="sum(current-group()/@pop)"/></td>
    </tr>
</xsl:for-each-group>
```

Multiple output documents

```
<xsl:result-document href="foo.html">
 <!-- add instructions to generate document content here -->
</xsl:result-document>
```

Regular expressions

```
<!--This example transforms dates of the form "12/8/2003" into ISO 8601
standard form: "2003-12-08".-->
<xsl:analyze-string select="$date" regex="([0-9]+)/([0-9]+)/([0-9]{{4}})">
  <xsl:matching-substring>
    <xsl:number value="regex-group(3)" format="0001"/>
    <xsl:text>-</xsl:text>
  </xsl:matching-substring>
</xsl:analyse-string>
```

Streaming

```
<xsl:template match="/">
    <xsl:stream href="books.xml">
        <xsl:iterate select="/books/book">
            <xsl:result-document href="{concat('book', position(),'.xml')}">
                <xsl:copy-of select="."/>
            </xsl:result-document>
            <xsl:next-iteration/>
        </xsl:iterate>
    </xsl:stream>
</xsl:template>
```

Higher-order functions

```
<xsl:value-of select="$f1(2)"/>
```

Text processing: CSV, JSON, … on input

```
<xsl:variable name="header" select="tokenize(unparsed-text-lines($csv)[1], $sep)"/>
```

Examples

XSLT Example - IANA registry - generating HTML

Open https://www.iana.org/assignments/media-types/media-types.xml in a web browser

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="media-types.xsl"?>
<?oxygen RNGSchema="media-types.rng" type="xml"?>
<registry xmlns="http://www.iana.org/assignment
id="media-types">
  <title>Media Types</title>
  <category>Multipurpose Inte
Media Types</category>
  <updated>2021-03-10</update


  <registration_rule>Expert Review for Vendor and Personal
Trees.</registration_rule>
  <expert>Ned Freed, Alexey Melnikov, Murray Kucherawy
(backup)</expert>
  <xref type="rfc" data="rfc6838"/>
  <xref type="rfc" data="rfc4855"/>
...
```

Link to XSLT stylesheet transforming XML to HTML

Media Types

Last Updated
2021-03-10
Registration Procedure(s)
Expert Review for Vendor and Personal Trees.
Expert(s)
Ned Freed, Alexey Melnikov, Murray Kucherawy (backup)
Reference
[RFC6838][RFC4855]
Note
Per Section 3.1 of [RFC6838], Standards Tree requests made through IETF documents will be reviewed and approved by the IESG, while requests made by other recognized standards organizations will be reviewed by the Designated Expert in accordance with the Specification Required policy. IANA will verify that this organization is recognized as a standards organization by the IESG.
Note
[RFC2046] specifies that Media Types (formerly known as MIME types) and Media Subtypes will be assigned and listed by the IANA.

Procedures for registering Media Types can be found in [RFC6838], [RFC4289], and [RFC6657]. Additional procedures for registering media types for transfer via Real-time Transport Protocol (RTP) can be found in [RFC4855].

The following is the list of Directories of Content Types and Subtypes. If you wish to register a Media Type with the IANA, please see the following for the online application:

[Application for registration of Media Types]

Other Media Type Parameters: [IANA registry *media-types-parameters*]
Media Type Sub-Parameters: [IANA registry *media-type-sub-parameters*]
Provisional Standard Media Type Registry: [IANA registry *provisional-standard-media-types*]
Available Formats
XML HTML Plain text

Registries included below

XSLT Example - Generating RDF Turtle

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
    <xsl:output method="text" encoding="UTF-8" />
    <xsl:variable name="prefix">https://ex.org/resource/</xsl:variable>
    <xsl:variable name="catalogIRI" select="concat($prefix, 'Catalog')"/>

    <xsl:template match="catalog">
    @prefix dcat: &lt;http://www.w3.org/ns/dcat#&gt; .
    @prefix dcterms: &lt;http://purl.org/dc/terms/&gt; .

    &lt;<xsl:value-of select="$catalogIRI"/>&gt; a dcat:Catalog .
        <xsl:apply-templates>
            <xsl:with-param name="currentIRI" select="$catalogIRI"/>
        </xsl:apply-templates>
    </xsl:template>

    <xsl:template match="dataset">
    <xsl:variable name="datasetIRI" select="concat($prefix, 'dataset/', fn:position())"/>
    &lt;<xsl:value-of select="$catalogIRI"/>&gt; dcat:dataset &lt;<xsl:value-of
select="$datasetIRI"/>&gt; .
    &lt;<xsl:value-of select="$datasetIRI"/>&gt; a dcat:Dataset .
        <xsl:apply-templates select="title">
            <xsl:with-param name="currentIRI" select="$datasetIRI"/>
        </xsl:apply-templates>
    </xsl:template>

    <xsl:template match="title">
        <xsl:param name="currentIRI"/>
        &lt;<xsl:value-of select="$currentIRI"/>&gt; dcterms:title &quot;<xsl:value-of
select="text()"/>&quot;@<xsl:value-of select="@xml:lang"/> .
    </xsl:template>

    <xsl:template match="text()" mode="#all"/>
</xsl:stylesheet>
```

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<https://ex.org/resource/Catalog> a dcat:Catalog .
    <https://ex.org/resource/Catalog> dcterms:title "My catalog"@en .
    <https://ex.org/resource/Catalog> dcterms:title "Můj katalog"@cs .

<https://ex.org/resource/Catalog> dcat:dataset <https://ex.org/resource/dataset/2> .
<https://ex.org/resource/dataset/2> a dcat:Dataset .
    <https://ex.org/resource/dataset/2> dcterms:title "Bikesharing in Brno"@en .
    <https://ex.org/resource/dataset/2> dcterms:title "Sdílení kol v Brně"@cs .

<https://ex.org/resource/Catalog> dcat:dataset <https://ex.org/resource/dataset/4> .
<https://ex.org/resource/dataset/4> a dcat:Dataset .
    <https://ex.org/resource/dataset/4> dcterms:title "Bikesharing in Prague"@en .
    <https://ex.org/resource/dataset/4> dcterms:title "Sdílení kol v Praze"@cs .
```

Literature

Jiří Kosek - XML pro každého (2004!) - https://www.kosek.cz/xml/index.html (in Czech)