Hands-on TEI Publishing

Wolfgang Meier

Magdalena Turska

open source project developed since 2015

grass-roots community efforts coordinated by e-editiones

informal communication via Slack

source code on GitHub

eXist

eXist Core Features

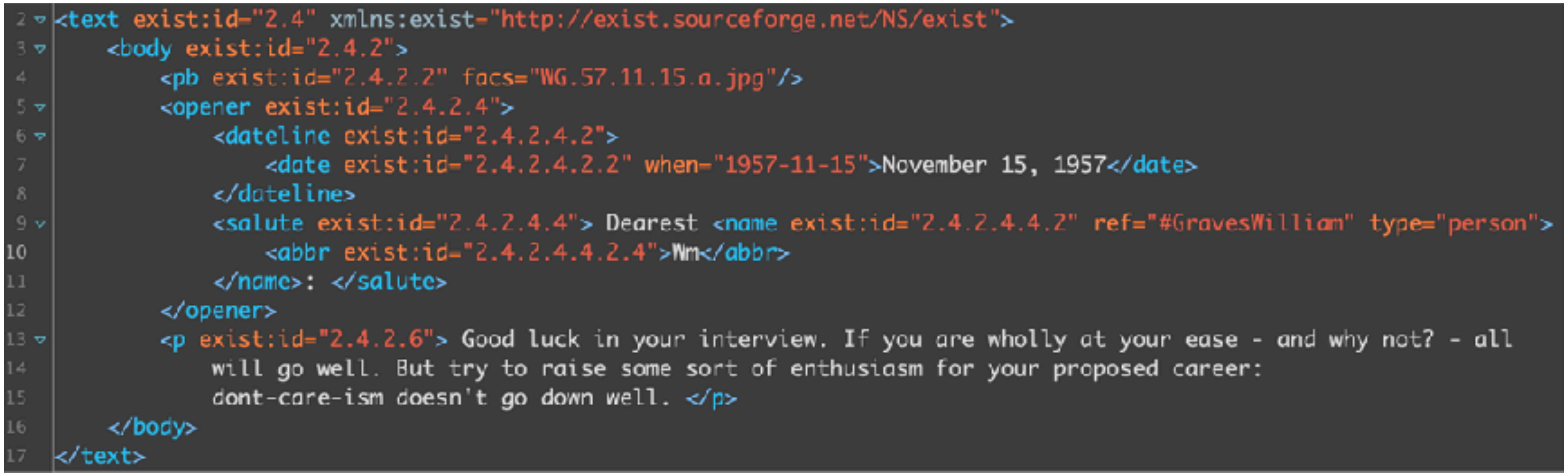
1. Accept any XML of any size or complexity
2. Directly locate any node in a huge collection
3. Avoid loading documents into memory
4. Evaluate XPath/XQuery via indexes, not tree traversals

1. Accept any XML

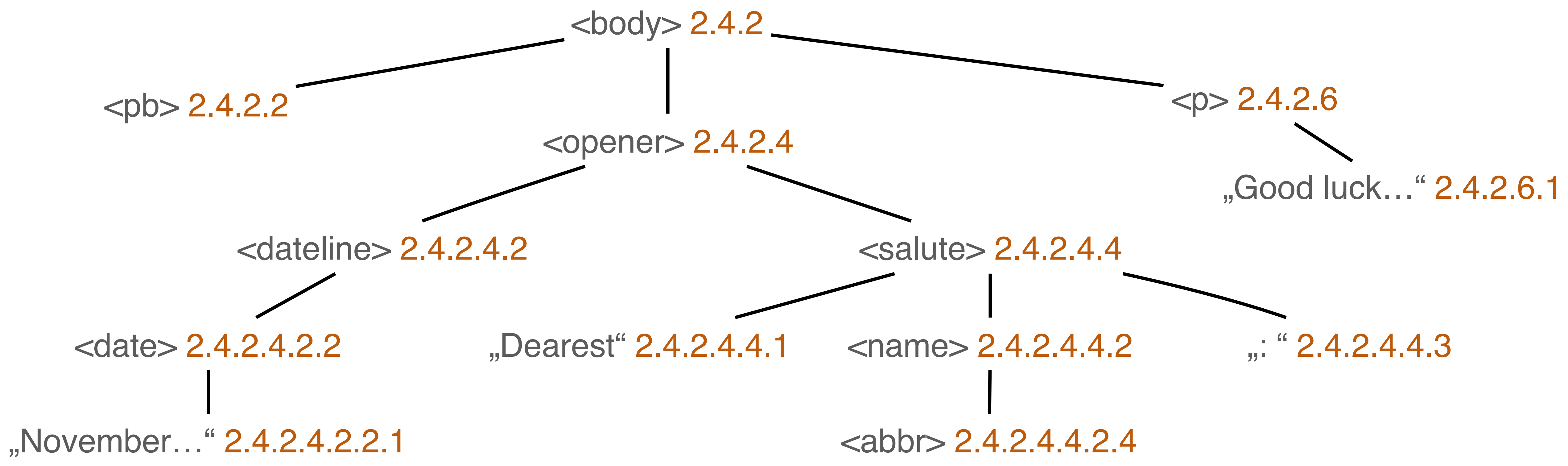
- XML should be designed to match the domain it describes, not the system to process it!
- Size of a single document should be irrelevant

2. Directly locate any node in a collection

- Every node in eXist has a unique address
- Immediately locate any node, no matter how deep it is nested …
- … or how large its parent document is



```
<text exist:id="2.4" xmlns:exist="http://exist.sourceforge.net/NS/exist">
    <body exist:id="2.4.2">
        <pb exist:id="2.4.2.2" facs="WG.57.11.15.a.jpg"/>
        <opener exist:id="2.4.2.4">
            <dateline exist:id="2.4.2.4.2">
                <date exist:id="2.4.2.4.2.2" when="1957-11-15">November 15, 1957</date>
            </dateline>
            <salute exist:id="2.4.2.4.4"> Dearest <name exist:id="2.4.2.4.4.2" ref="#GravesWilliam" type="person">
                <abbr exist:id="2.4.2.4.4.2.4">Wm</abbr>
            </name>: </salute>
        </opener>
        <p exist:id="2.4.2.6"> Good luck in your interview. If you are wholly at your ease - and why not? - all
            will go well. But try to raise some sort of enthusiasm for your proposed career:
            dont-care-ism doesn't go down well. </p>
    </body>
</text>
```



3. Avoid loading documents into memory

- eXist mostly operates with node ids, not the actual nodes
- Documents or fragments are never loaded into memory - unless user explicitly asks for it
- Avoid access to the actual node stored on disk

4. Evaluate XPath/XQuery via indexes

- XPath describes traversing a tree
- eXist uses indexes to take shortcuts everywhere
- Can determine the relationship between nodes based on node identifiers
- Uses fast set operations to compare node identifiers instead of traversing actual document tree

More than an XML Database

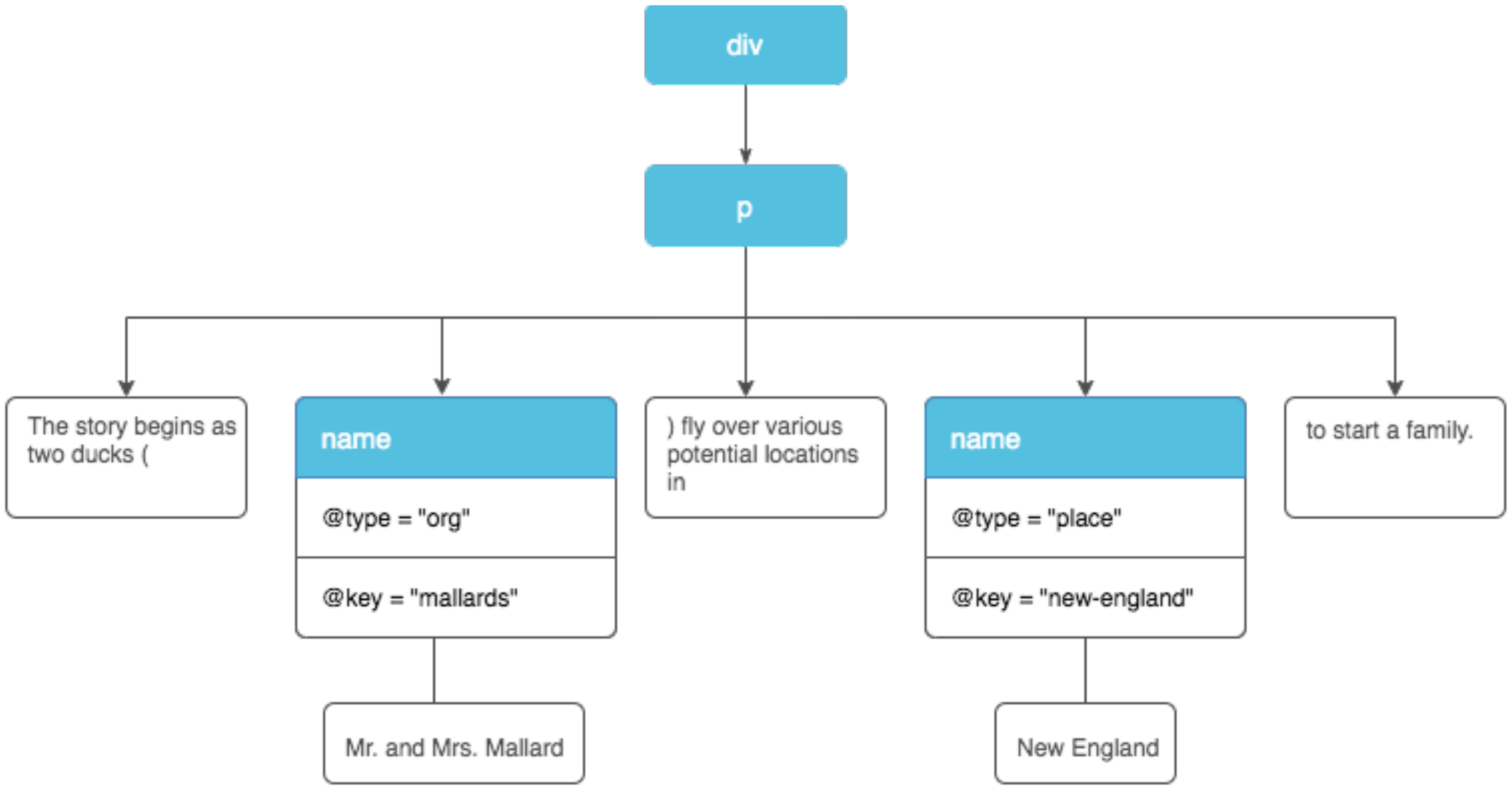
- eXist aims to provide an entire ecosystem for developing XML-based applications
- All our applications are written in XQuery: no other language needed
- Standardized packaging for apps and libraries
- Install apps with a mouse click

XPath

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<div xmlns="http://my.fantasy.namespace">
    <!-- This whole document is in a made-up namespace -->
    <p>The story begins as two ducks (<name type="org"
key="mallards">Mr. and Mrs. Mallard</name>)
        fly over various potential locations in <name
type="place" key=“new-england">New England</name> to start
a family.</p>
</div>
```

as a tree

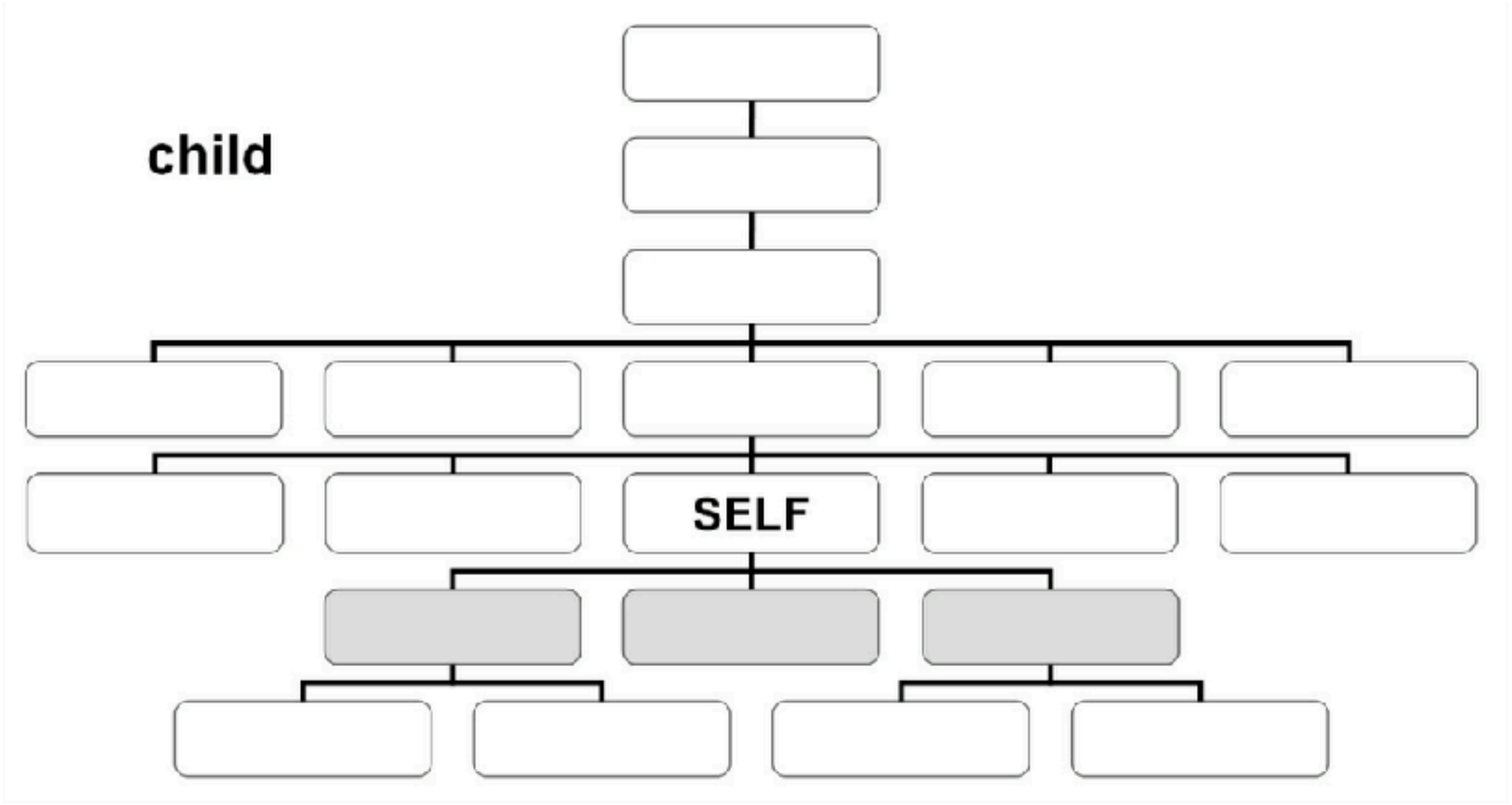

div
p
The story begins as two ducks (
name
@type = "org"
@key = "mallards"
Mr. and Mrs. Mallard
) fly over various potential locations in
name
@type = "place"
@key = "new-england"
New England
to start a family.

Node types in XML

element
attribute
text
namespace
processing-instruction
comment
document node

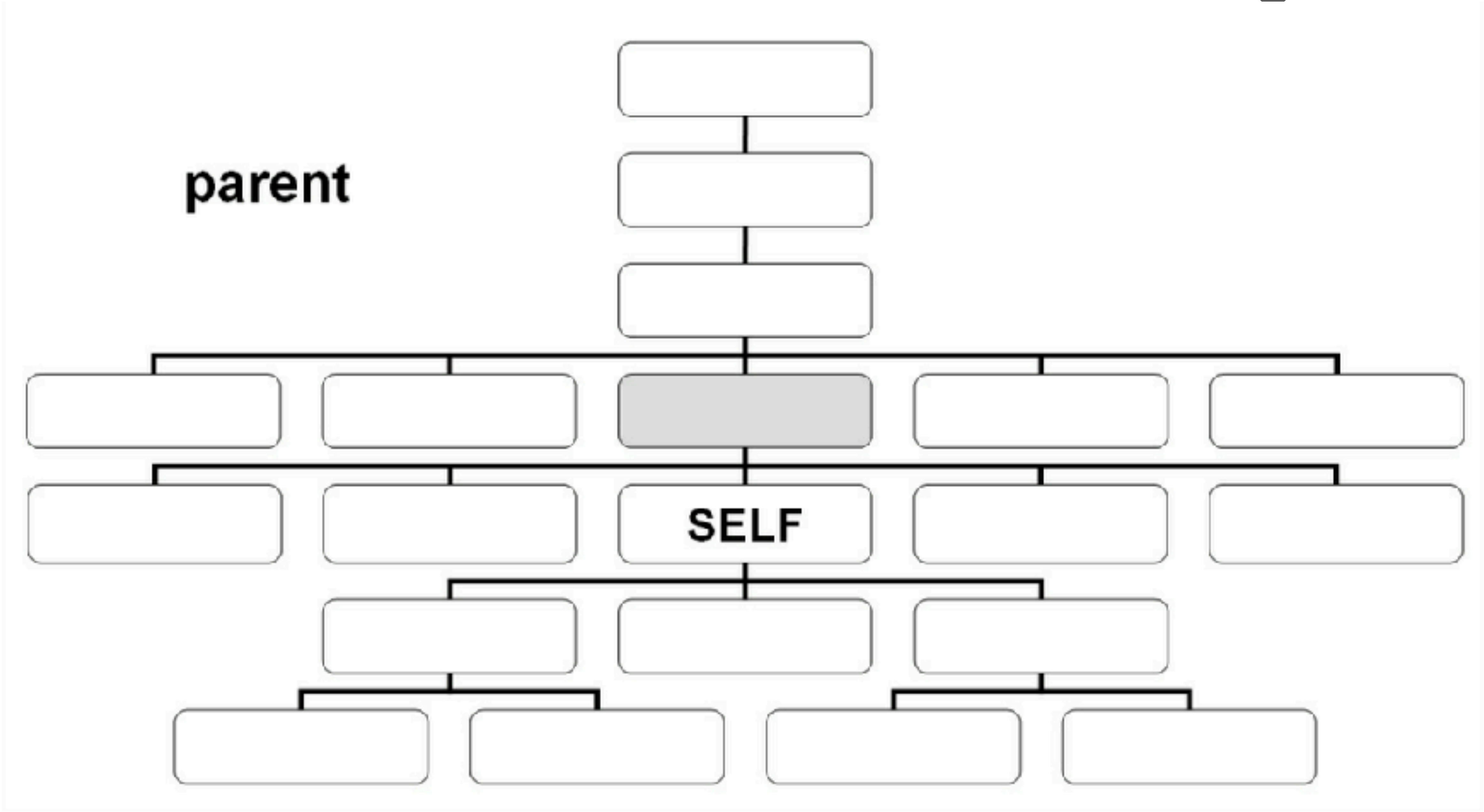
Family relationships

parent **child** ancestor descendant sibling



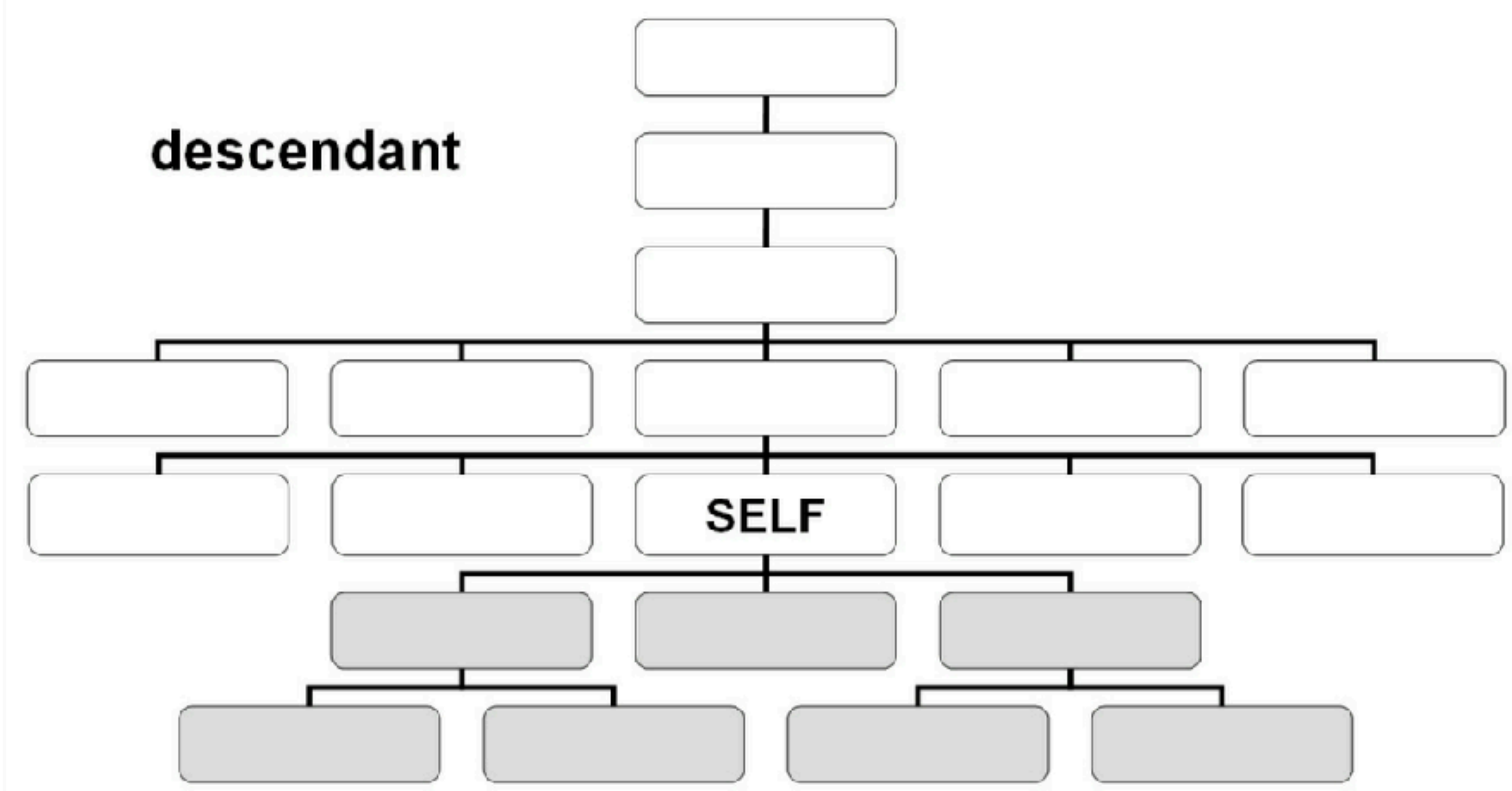
Family relationships

parent child ancestor descendant sibling



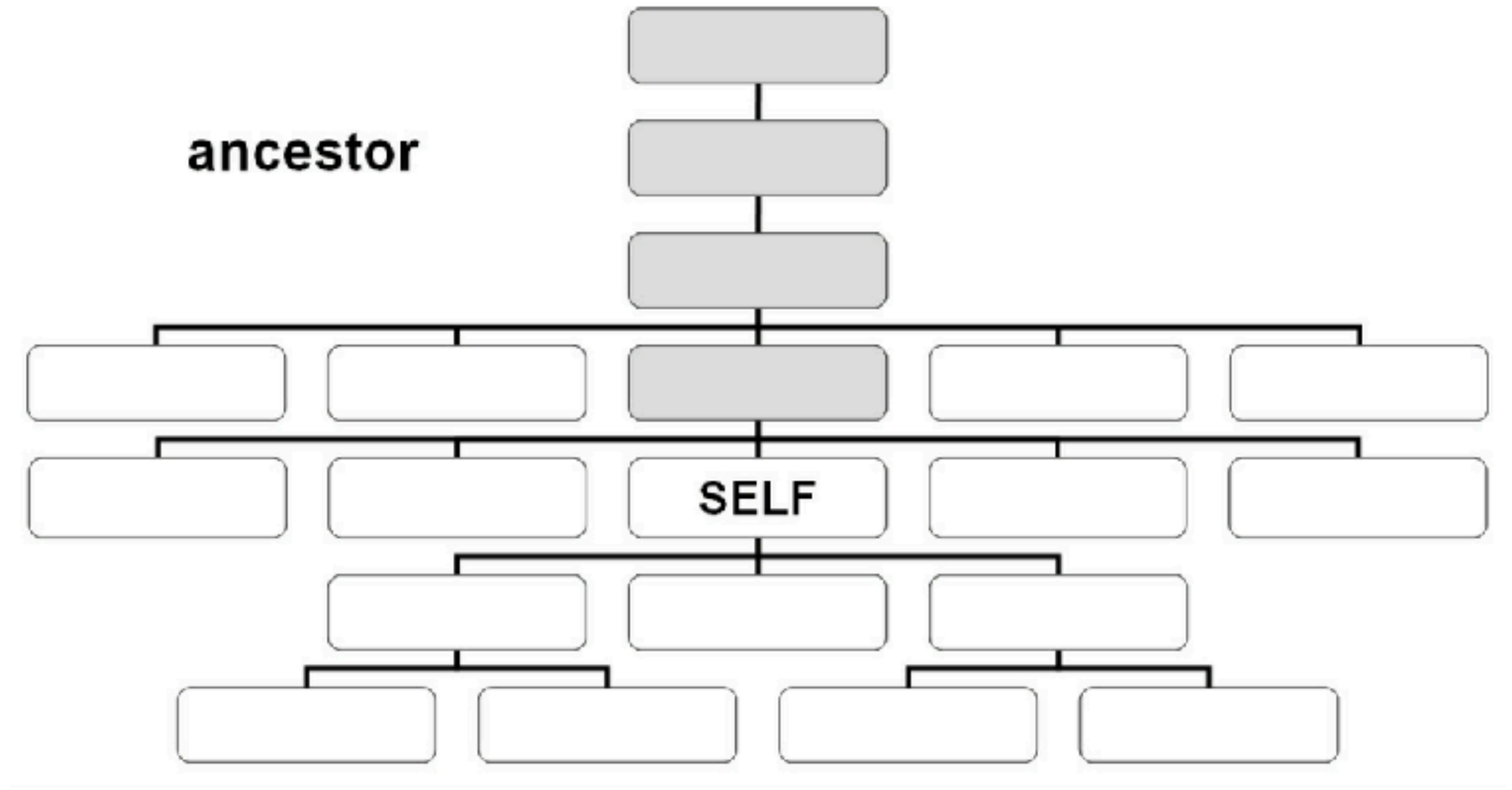
Family relationships

parent child ancestor **descendant** sibling



Family relationships

parent child **ancestor** descendant sibling



Simple path expressions

- navigate from a current location* to other nodes in the tree
- by default look for it among the children of the current node
- steps separated with a / (slash) character
- context node changes with each step

```
/TEI/teiHeader/fileDesc/titleStmt/title

text/p

div/p/persName/@ref

date/@when
```

*context node

Axes

- path steps by default look among children of the current node aka `child::` axis

```
text/p

text/child::p
```

- but it is possible to navigate between any parts of the document

```
div/descendant::persName

div//persName
```

Axes

- parent::

```
div/parent::text
```

- descendant::

```
div/descendant::name

div//name
```

- ancestor::

```
div/ancestor::text

div/ancestor::TEI
```

Establishing a context

context of an expression may be explicitly specified with

`doc()` and `collection()` functions

```
collection("/db/apps/tei-publisher/data/test")/text/p
```

```
div/p/persName
```

Predicates

Positional

```
text/div[1]
```

```
/TEI/teiHeader/revisionDesc/change[last()]
```

Filters

```
div[@type='chapter']
```

```
name[@type='person']
```

Functions

```
name[contains(., 'Kant')]

name[starts-with(., 'la')]

sp[count(l) > 10]

name/string()

date/substring(@when, 1, 4)

date/substring(@when, 6)

name/substring-after(@ref, '#')
```

Namespaces

full name of an element consists of its local name and namespace

to avoid using lengthy namespace URIs, we can define a namespace *prefix*

```
declare namespace tei="http://www.tei-c.org/ns/1.0";
```

```
tei:div

tei:div/descendant::tei:name

tei:div//tei:name
```

id() function

retrieves a node by its @xml:id

```
collection('/db/apps/tei-publisher')/id('F-rom-ben')/tei:persName[@type="standard"]
```

or

```
id('F-rom-ben', doc('/db/apps/tei-publisher/data/test/F-rom.xml'))
```

root() function

root: retrieves a document node for the current node

```
collection('/db/apps/tei-publisher')/id('F-rom-
ben')/tei:persName[@type='standard']/root()
```

or

```
root(id('F-rom-ben', doc('/db/apps/tei-
publisher/data/test/F-rom.xml')))
```