NPRG036

XML Technologies

Lecture 3

XPath

2. 3. 2020

Author: **Irena Holubová**
Lecturer: **Martin Svoboda**

http://www.ksi.mff.cuni.cz/~svoboda/courses/192-NPRG036/

Lecture Outline

- **XPath**
  - Data model
  - Path expressions
  - Axes, node tests, predicates

Query Languages for XML Data

- Aims: querying, views, transformations, actualization, …
  - Since 1998 XML-QL, XQL, …
  - The development stabilized in W3C in languages XSLT, XPath, XQuery
    - XSLT is a language for data transformation
      - Exploits XPath for targeting parts of XML document
      - Has XML syntax
    - XQuery is more suitable for querying – user-oriented
      - Exploits XPath for targeting parts of XML document
- Today: XPath 1.0
  - Note: XPath 2.0 $\subset$ XQuery

What is XPath?

- Basic language for querying XML data
  - Selecting parts of XML documents
- The idea resembles navigation in a file system
- XPath does not have XML syntax
- XPath is exploited in multiple other XML technologies
  - XSLT, XQuery, XML Schema, XPointer, …

XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```

XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```

document

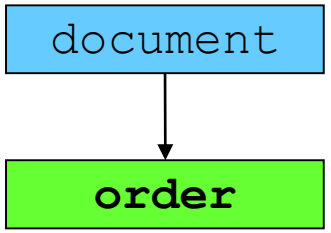
XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```

document

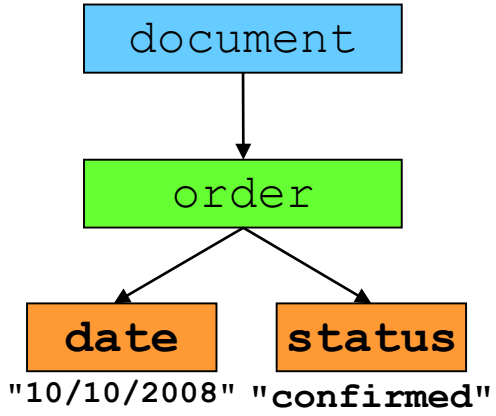
XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```



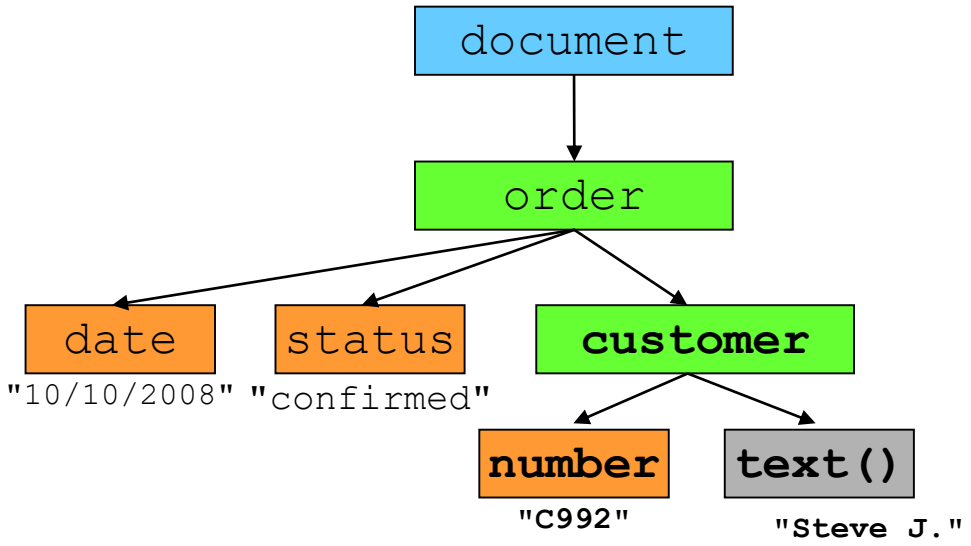
XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```

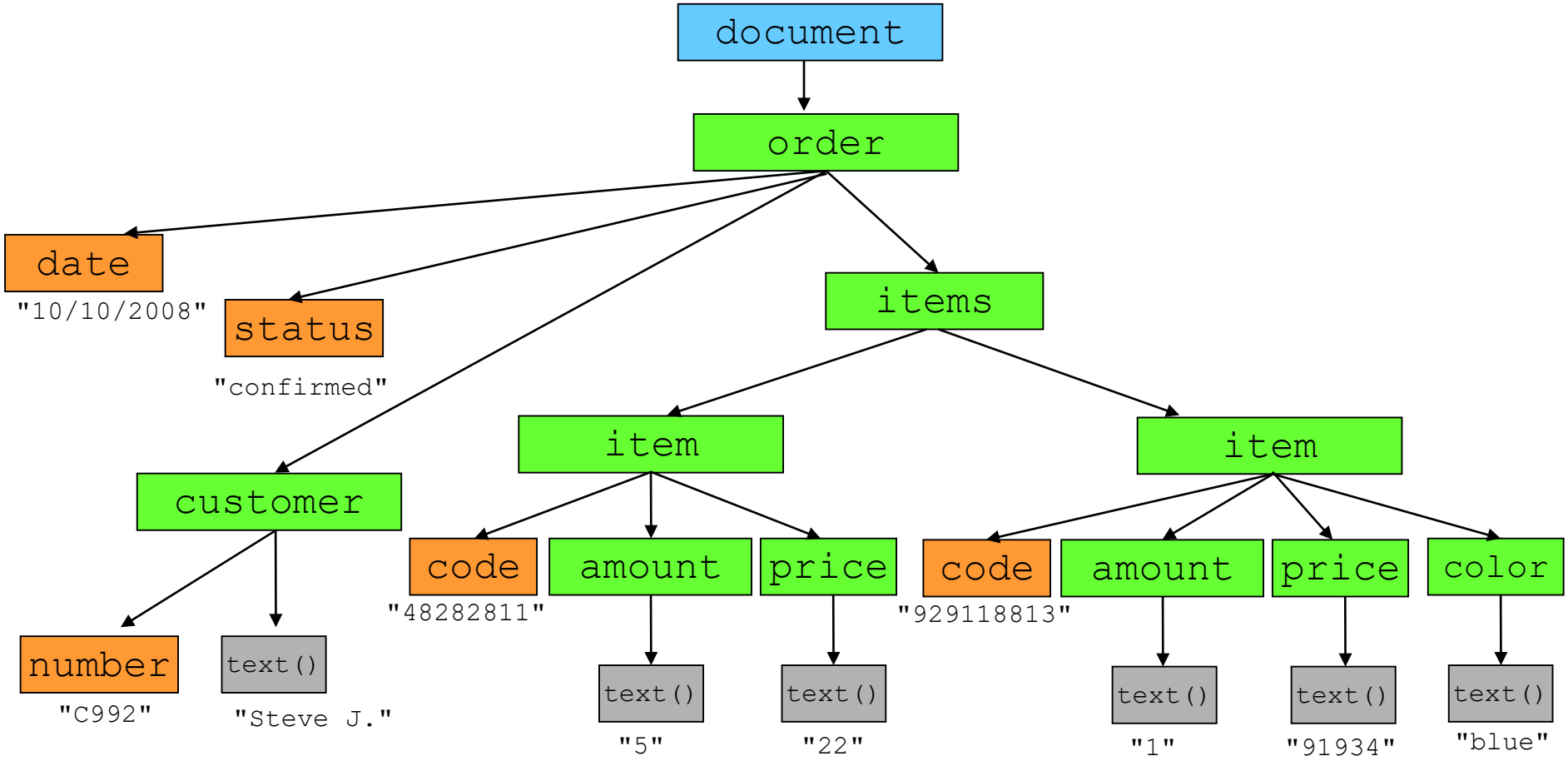


XPath Data Model

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order date="10/10/2008" status="confirmed">
 <customer number="C992">Steve J.</customer>
 <items>
  <item code="48282811">
   <amount>5</amount>
   <price>22</price>
  </item>
  <item code="929118813">
   <amount>1</amount>
   <price>91934</price>
   <color>blue</color>
  </item>
 </items>
</order>
```



XPath Data Model



XPath Data Model

XML Infoset?

- Types of nodes in the model
  - Root node
  - Element node
  - Text node
  - Attribute node
  - Comment
  - Processing instruction
  - Namespace
- What is not included: CDATA section, entity reference, DTD

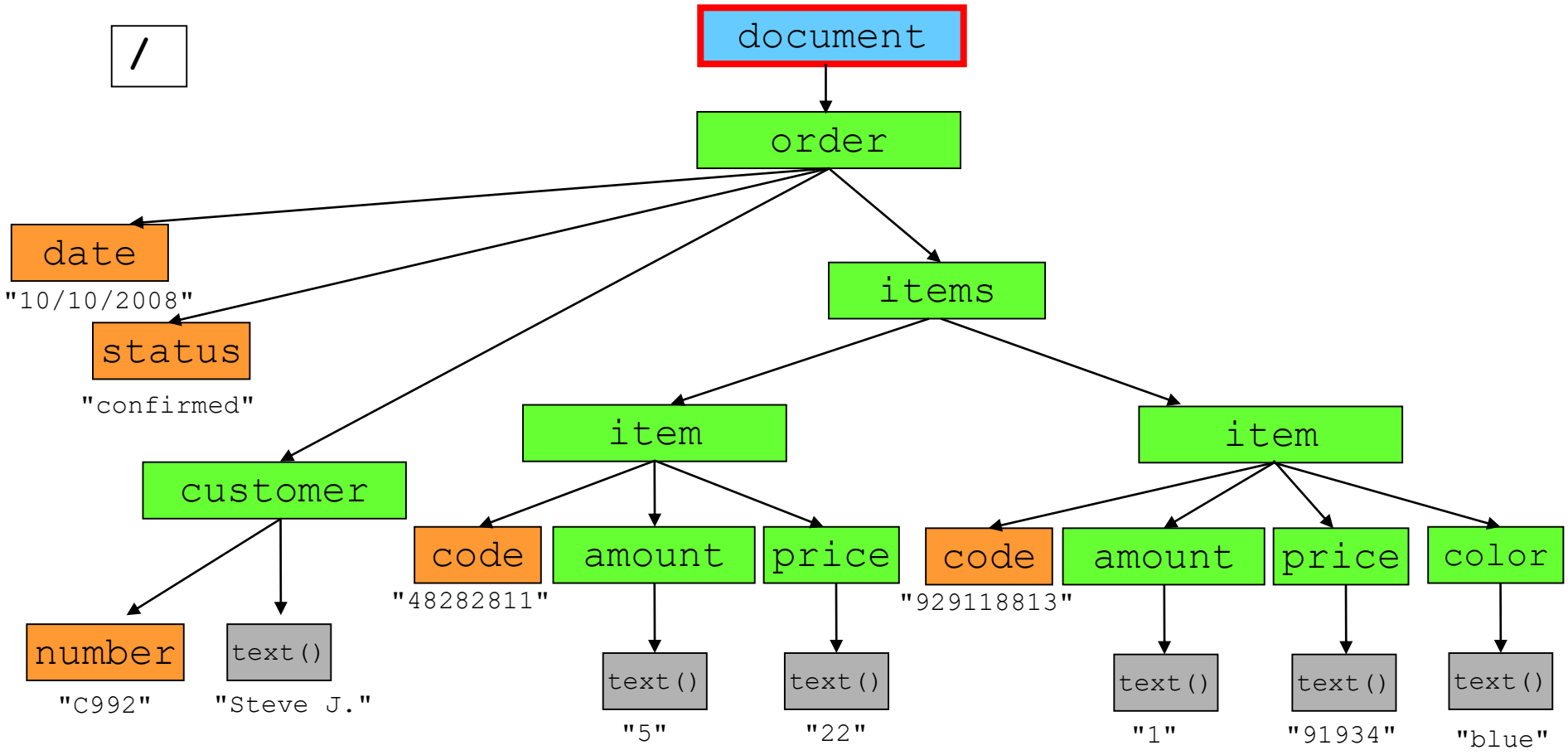
XPath Data Model

- Specifics
  - Attributes are not a part of list of child nodes of a node
  - Namespace declarations are not considered as attributes, but special types of nodes
    - xmlns:bib="…"
  - The root node does not represent root element but the whole document
    - Root element is represented as a child of root node

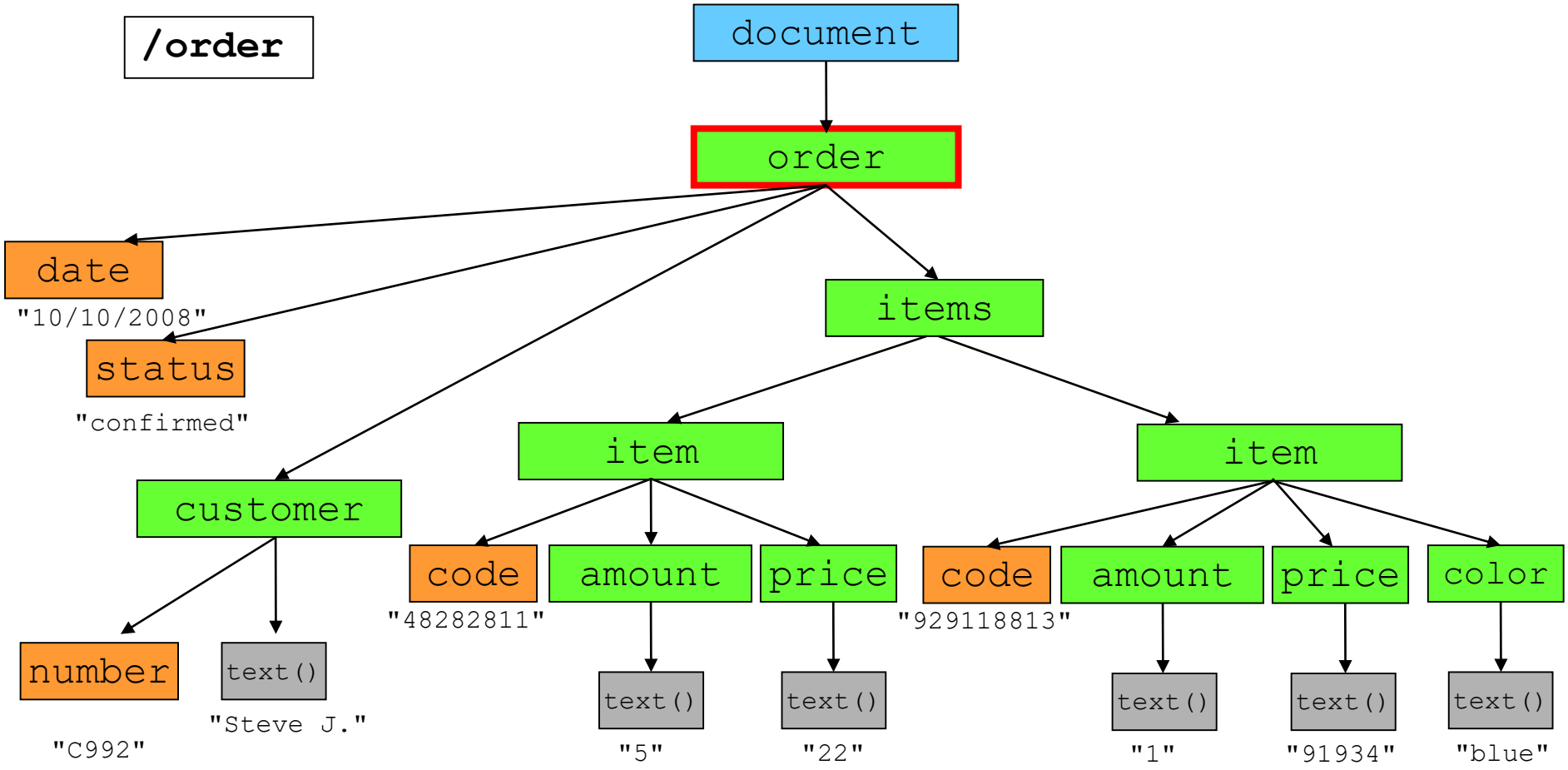
XPath Expression

- XPath expression is a path
- Path consists of steps
  - Absolute path:
    - `/Step1/Step2/…/StepN`
  - Relative path:
    - `Step1/Step2/…/StepN`

XPath Expressions – Examples

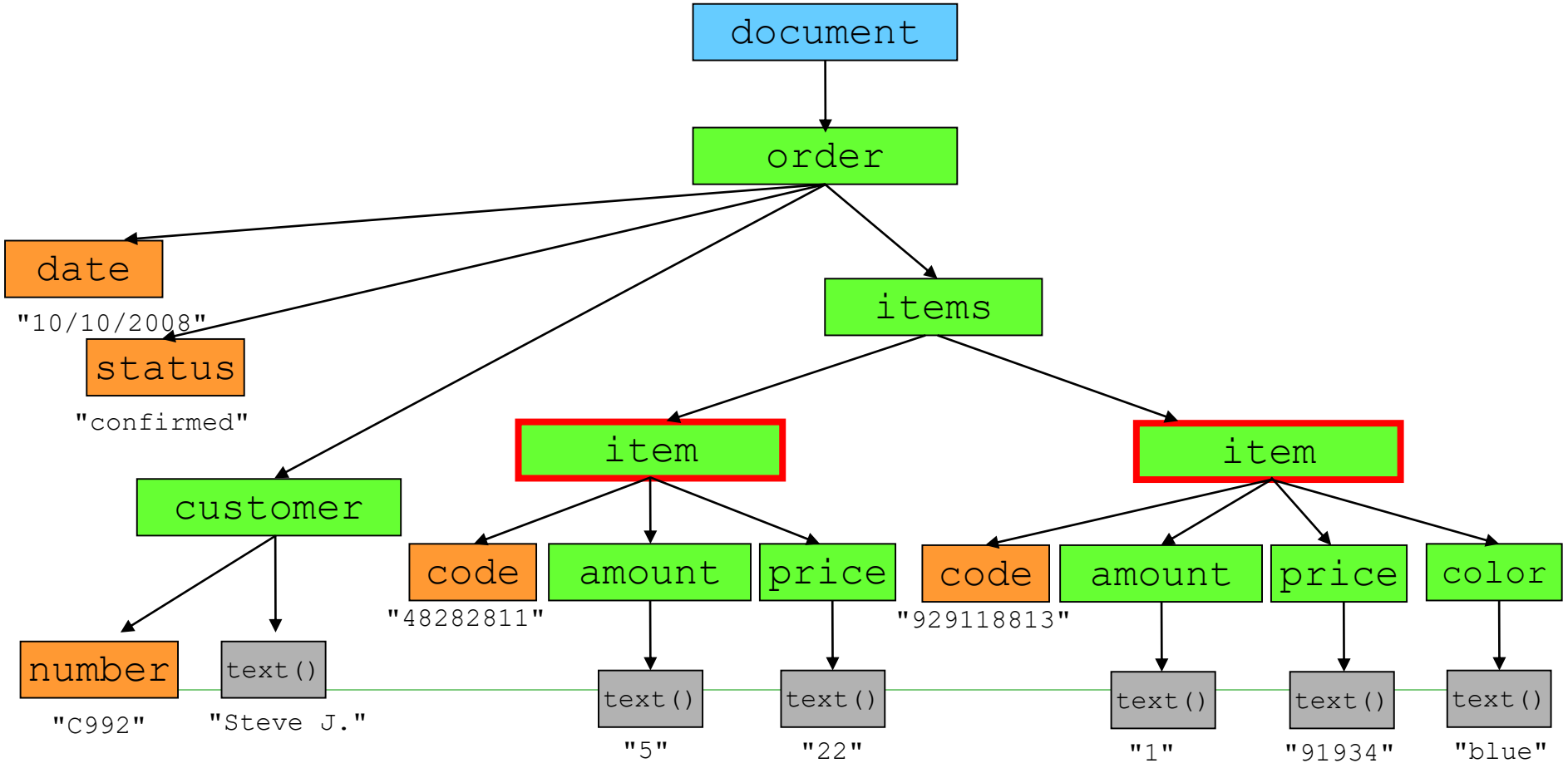


XPath Expressions – Examples



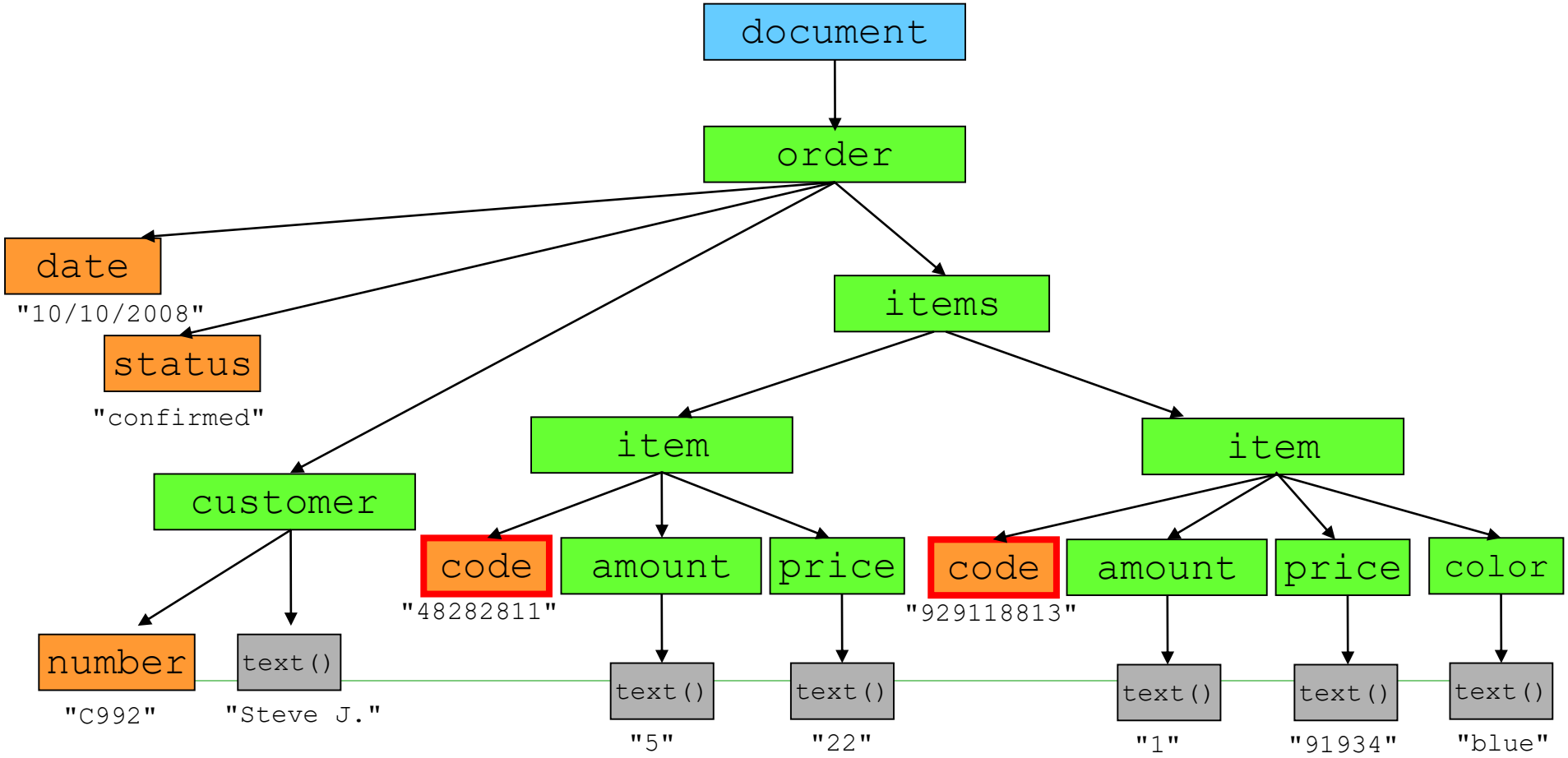
XPath Expressions – Examples

/order/items/item

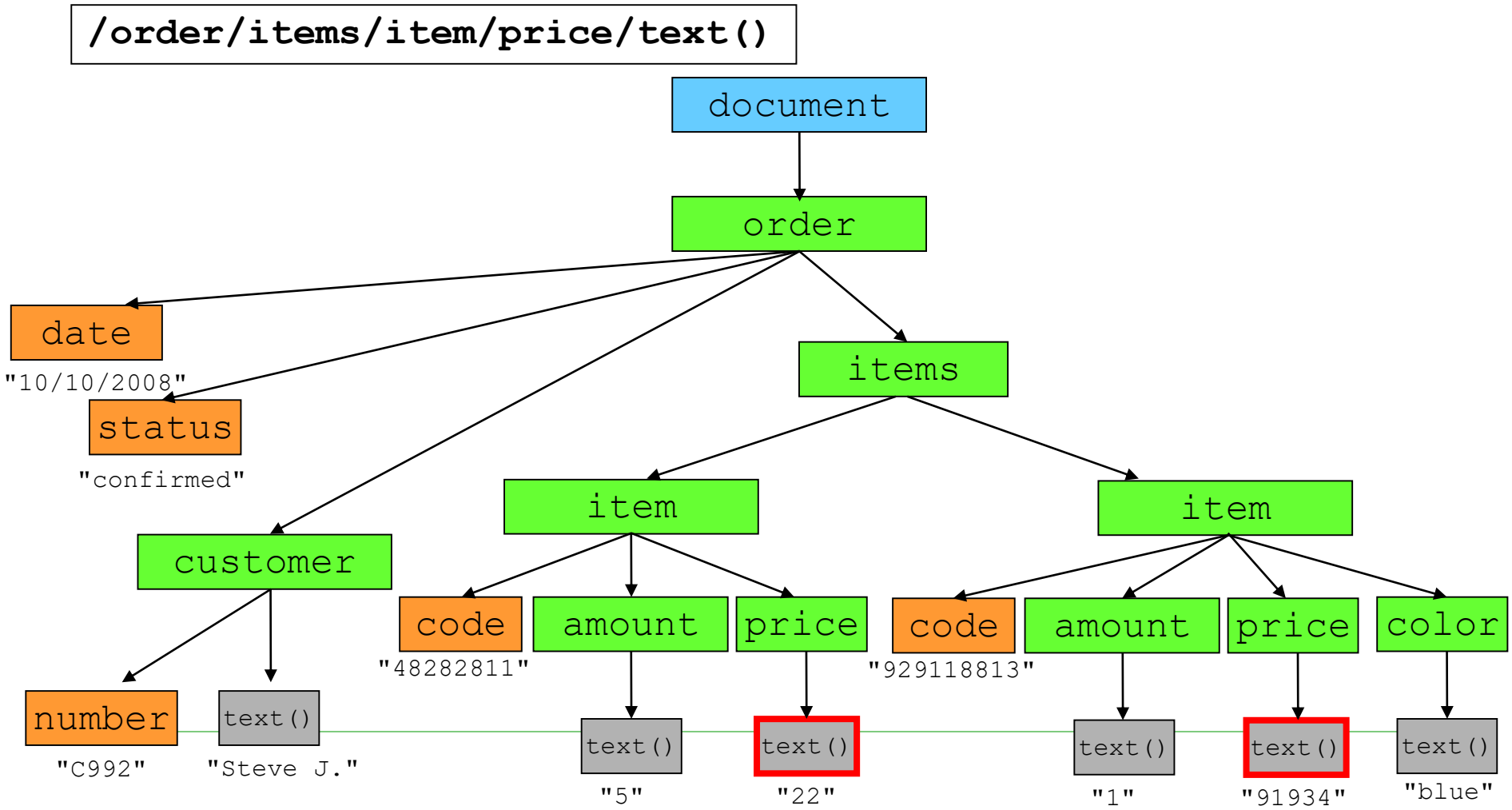


XPath Expressions – Examples

/order/items/item/@code

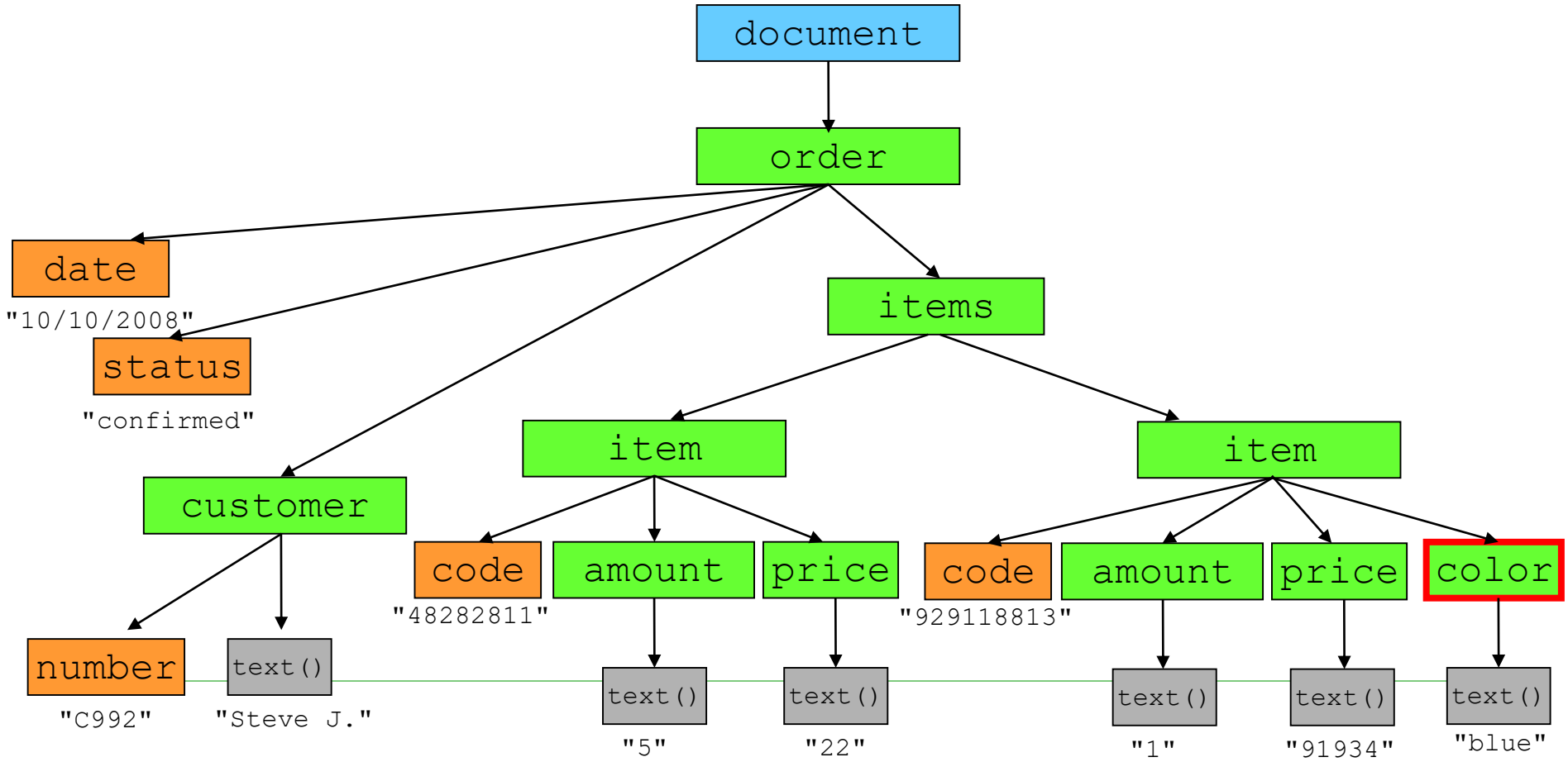


XPath Expressions – Examples



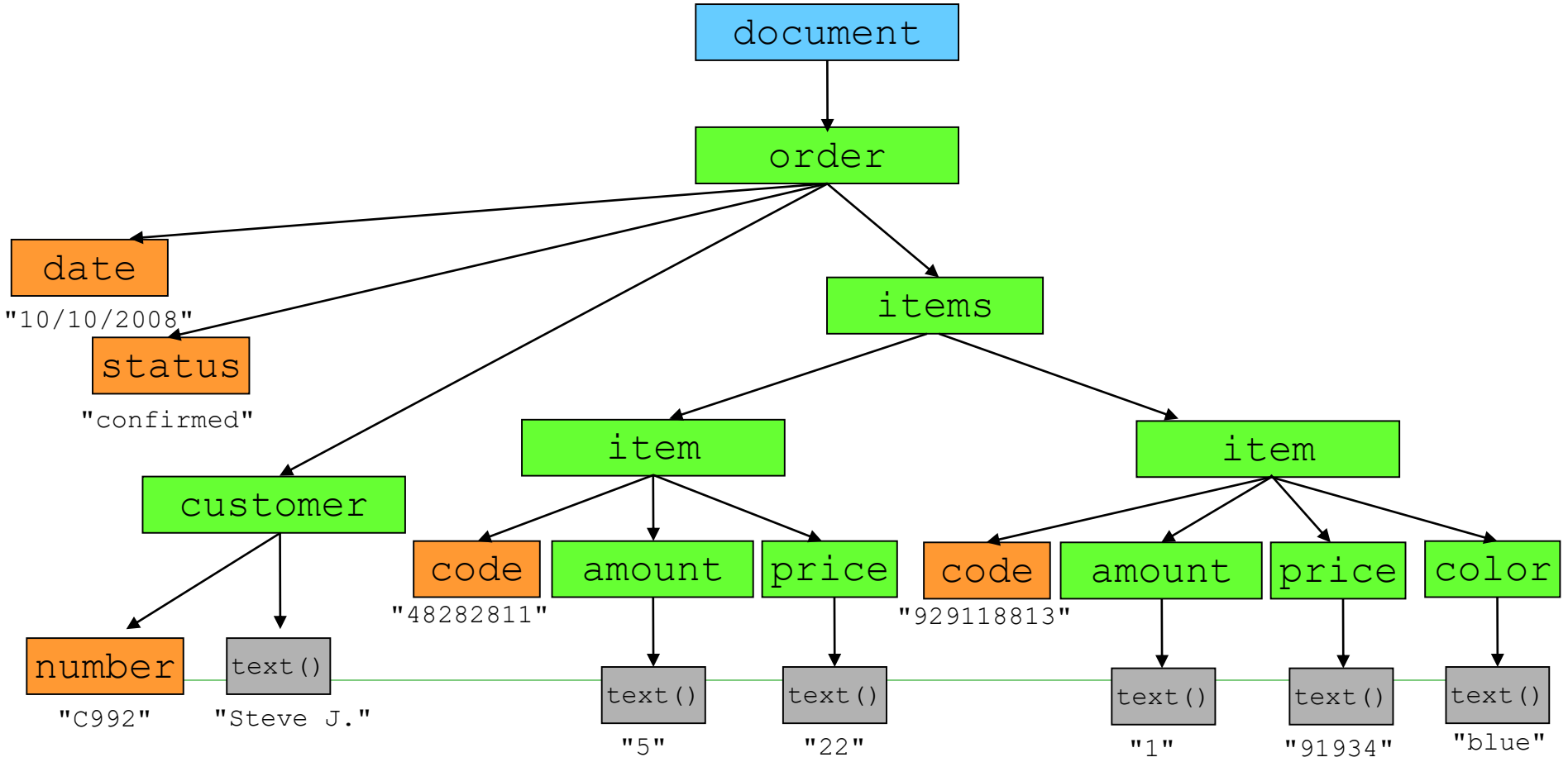
XPath Expressions – Examples

/order/items/item/color



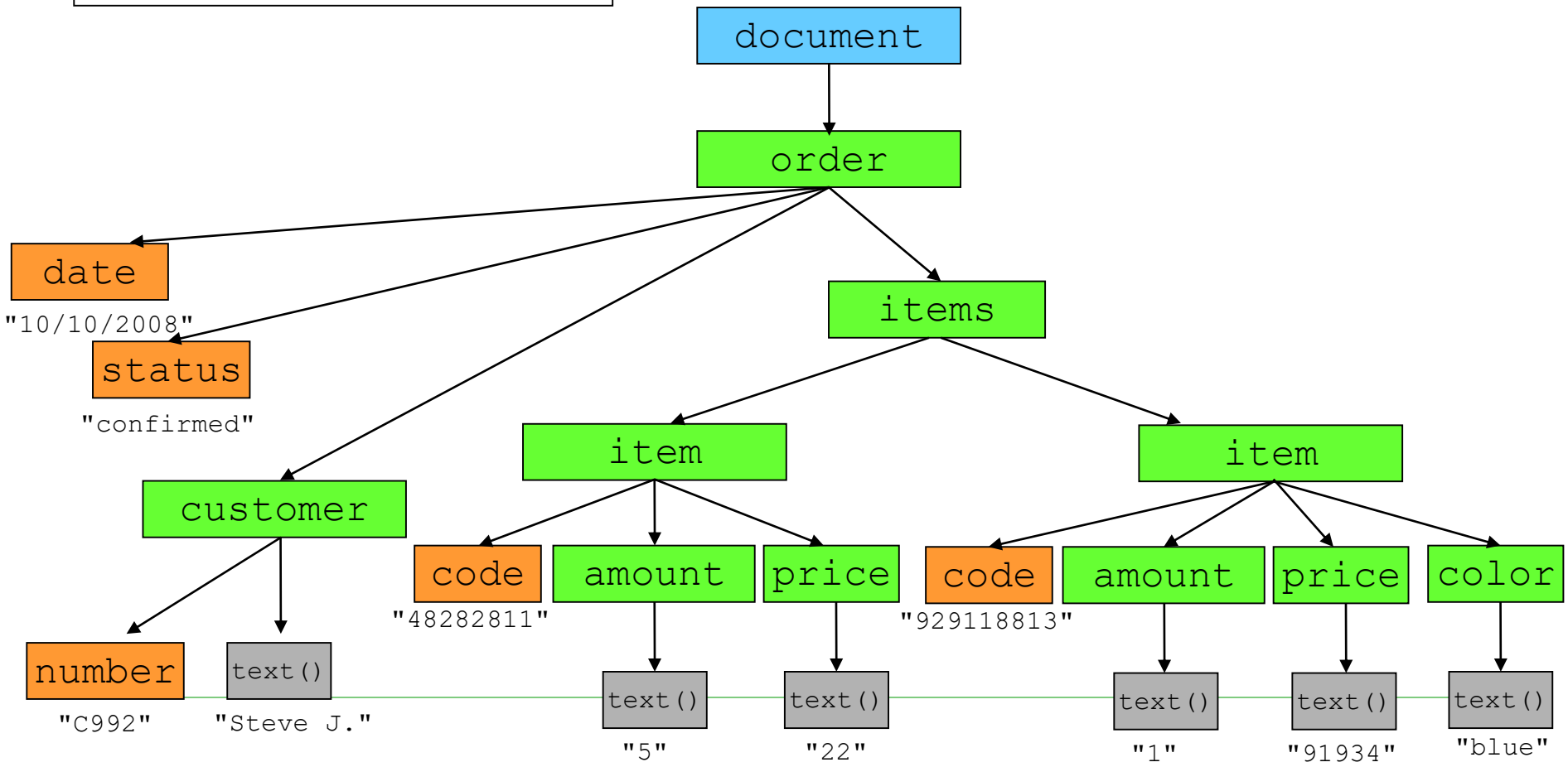
XPath Expressions – Examples

/order/customer/name



XPath Expressions – Examples

/order/item-list/item



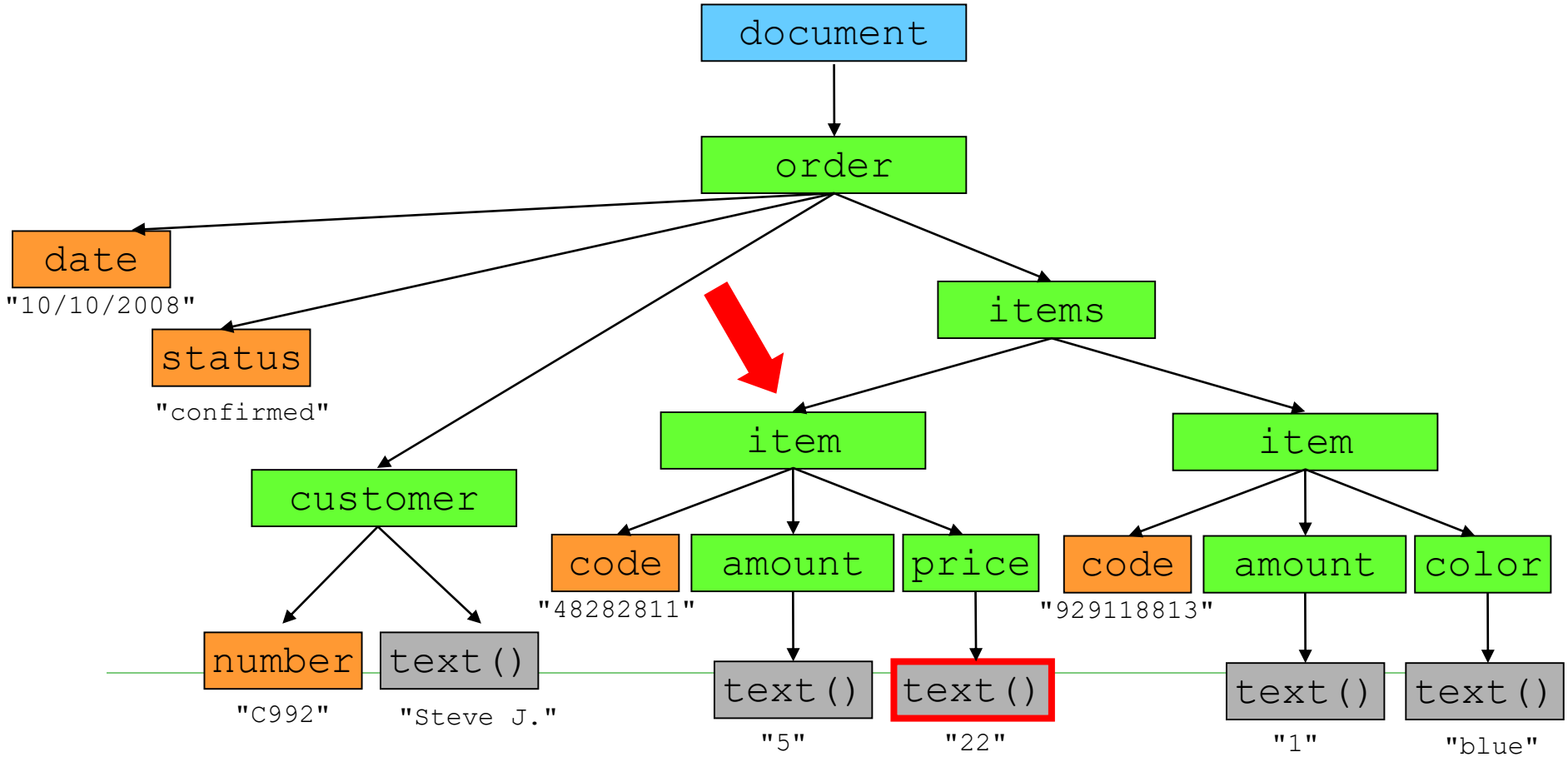
XPath Expressions – Examples

```
price/text()
```

- A relative path itself cannot be evaluated separately
  - It does not make sense, because we do not know where to start
  - The input must involve both the relative path and one or more nodes of XML documents where the evaluation should start
    - so-called context set

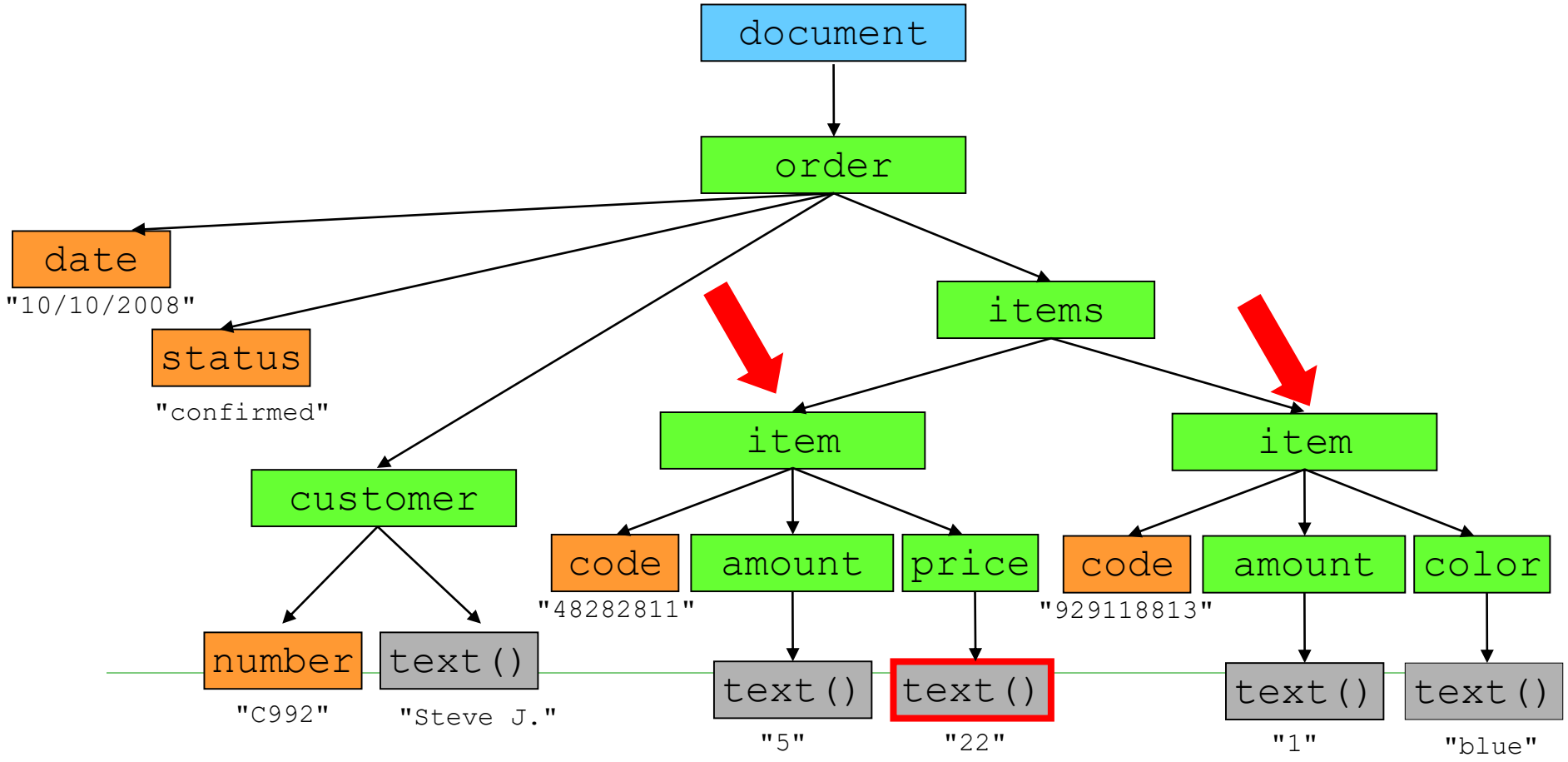
XPath Expressions – Examples

price/text()



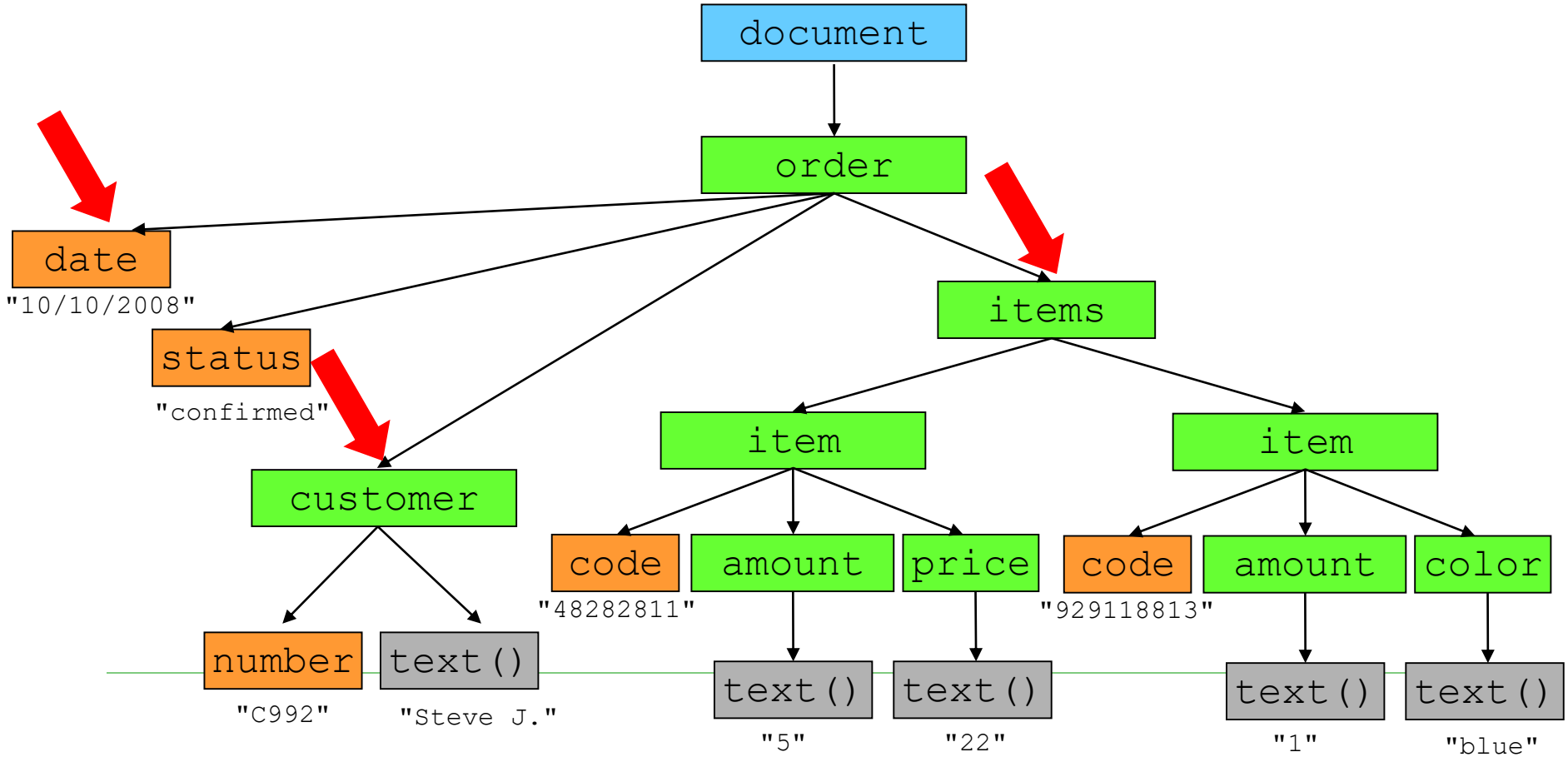
XPath Expressions – Examples

price/text()



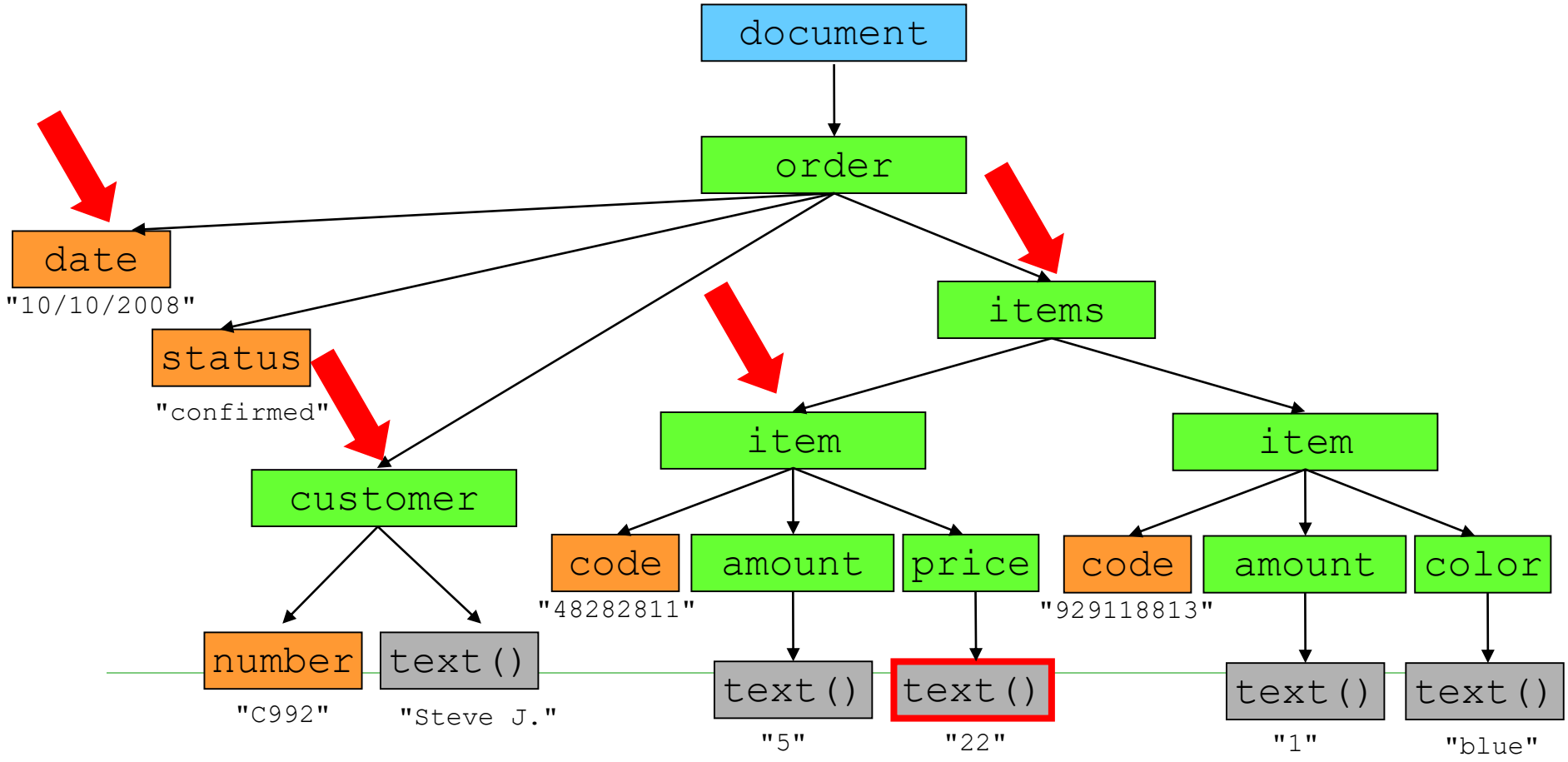
XPath Expressions – Examples

price/text()



XPath Expressions – Examples

price/text()



Evaluation of XPath Expression

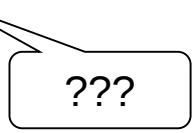
- Let P be an XPath path
- Let C be the context set of nodes for evaluation of P
  - If P is absolute, then C contains the root node of the document
  - If P is relative, then C must be specified explicitly
- If P is empty, then the result of evaluation corresponds to C
- Else, P is evaluated with regards to C as follows:
  - Let S be its first step and P' is the rest of the path, i.e. P = S/P'
  - C' = {}
  - For each node u from C evaluate S and add the result to C'
  - Evaluate P' with regard to C'

XPath Paths Formally

- XPath step is formally the following expression:

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

- Axis, node test and list of predicates
  - Predicates are optional
- So far we saw only node tests
  - The list of predicates was empty
  - Axes were abbreviated



XPath Axes

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

- Axis specifies the relation of nodes selected in this step with regard to each node u from context set C

```
child
```

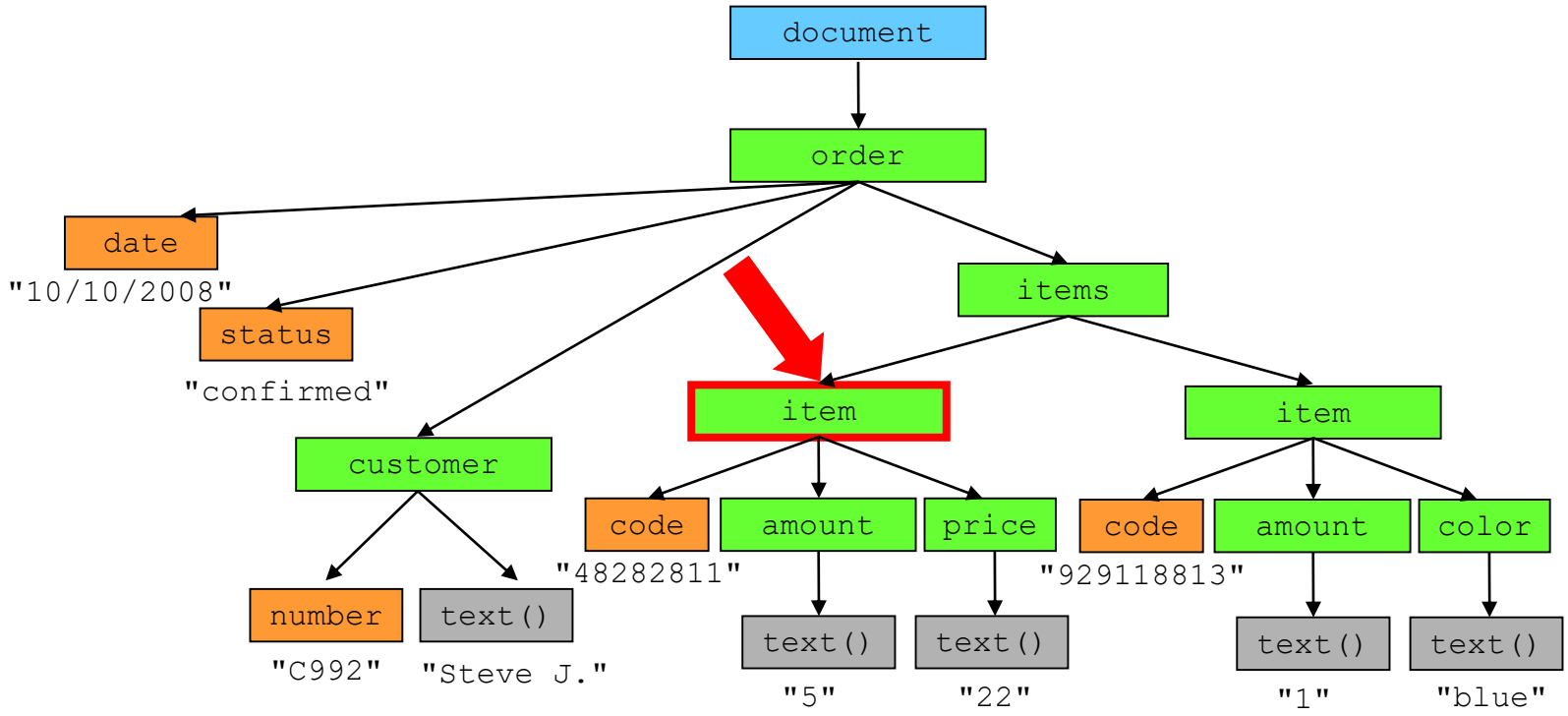
  - Selected nodes are child nodes of node u
  - Most common axis

```
/order/customer ↔
      /child::order/child::customer
```

abbreviation

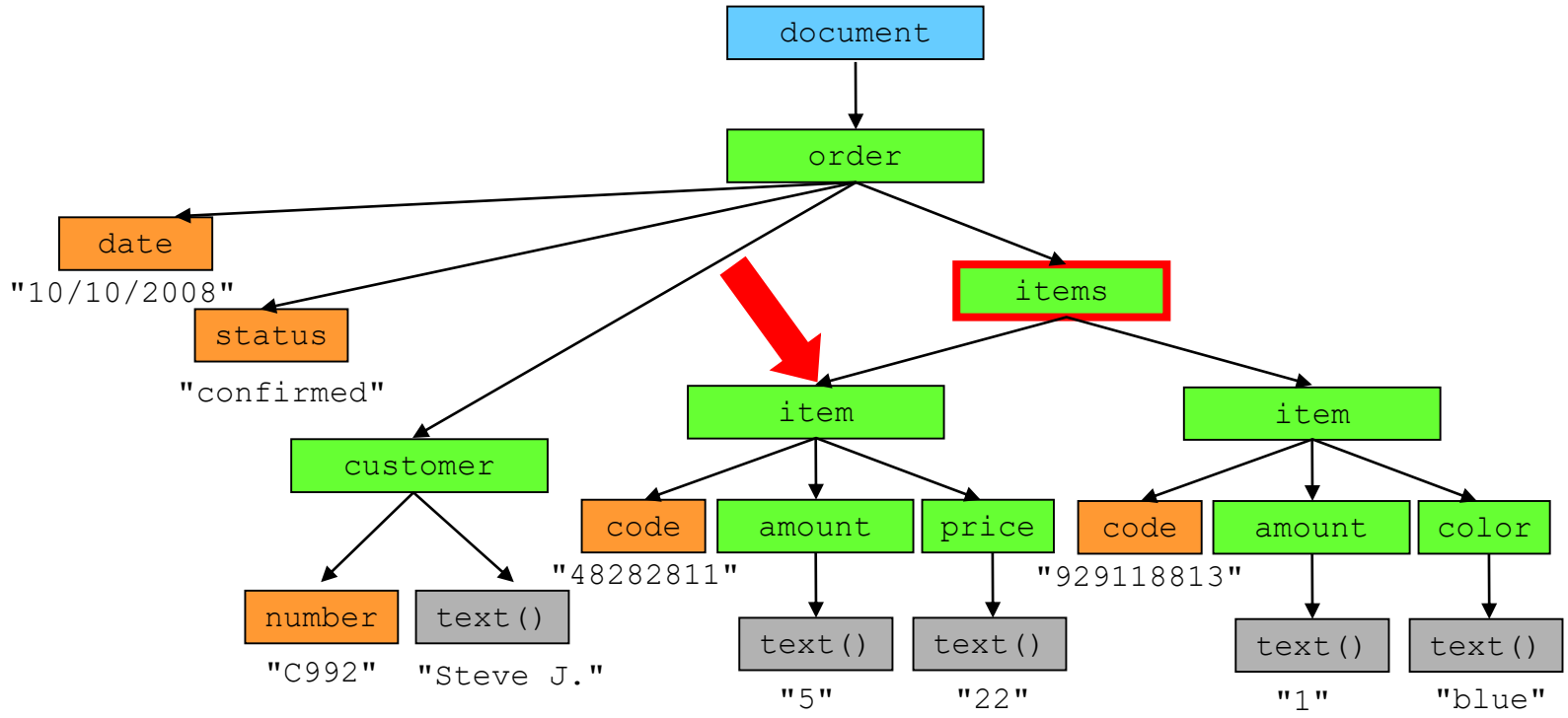
XPath Axis self

- The selected node is u itself



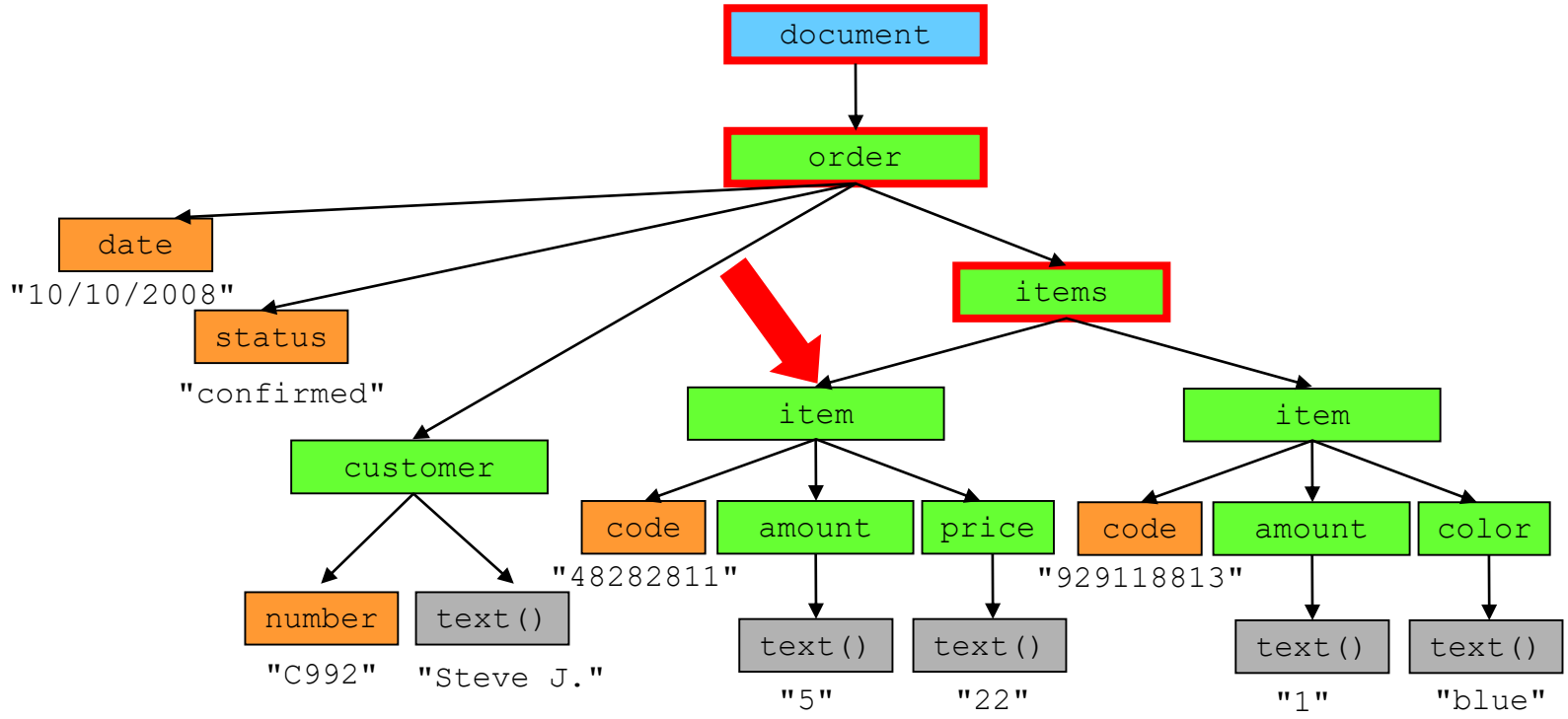
XPath Axis parent

- Parent node of node u



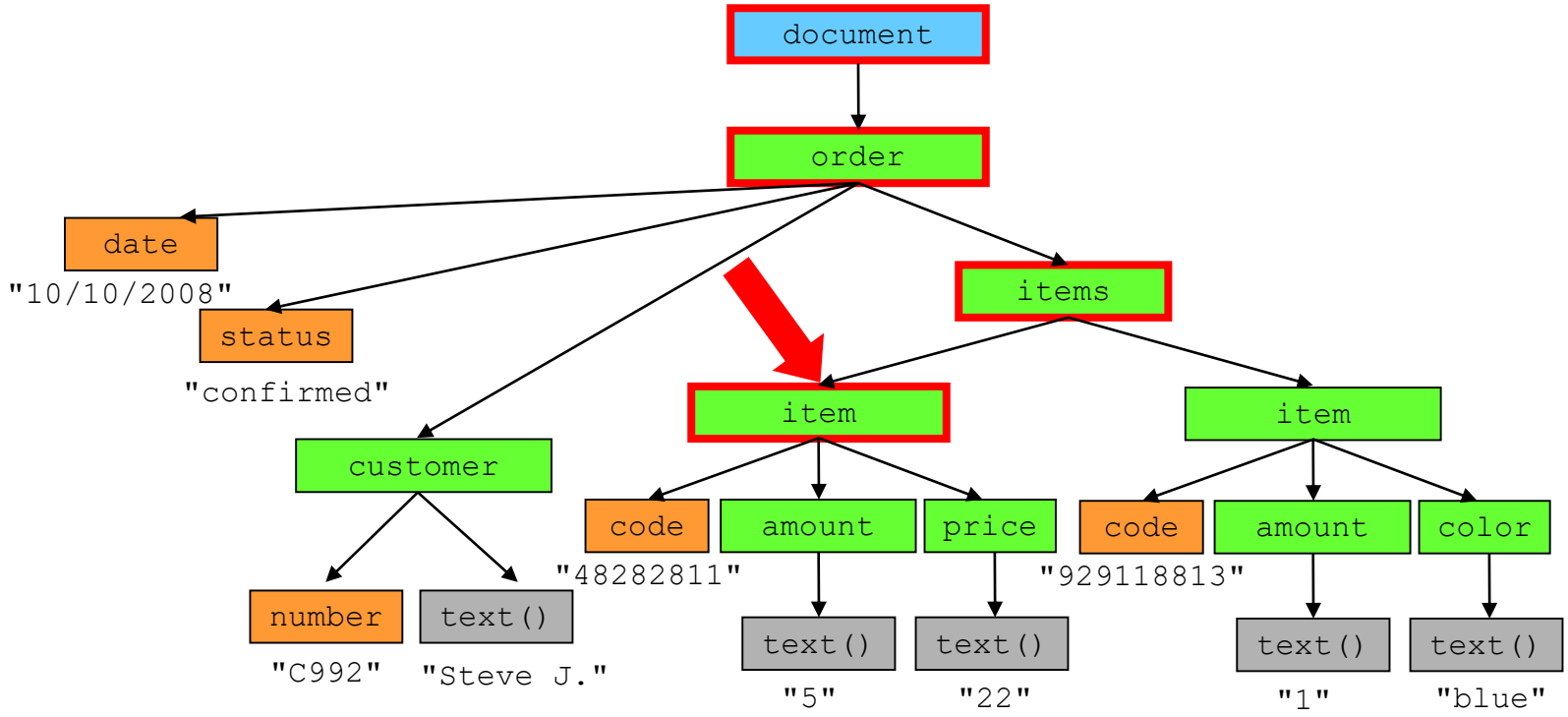
XPath Axis ancestor

- All ancestor nodes of node u
  - All nodes on the path from u to root node



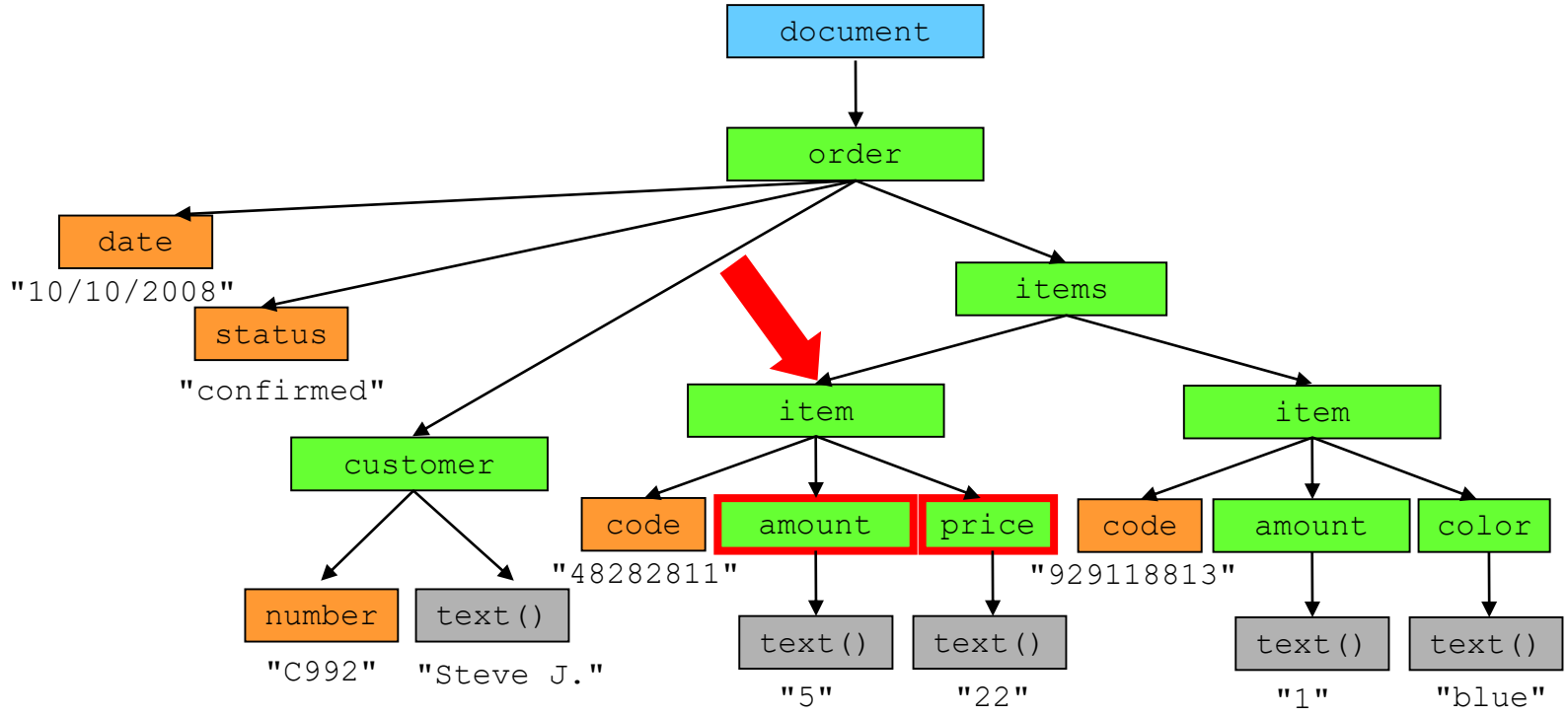
XPath Axis ancestor-or-self

- All ancestor nodes of node u including u



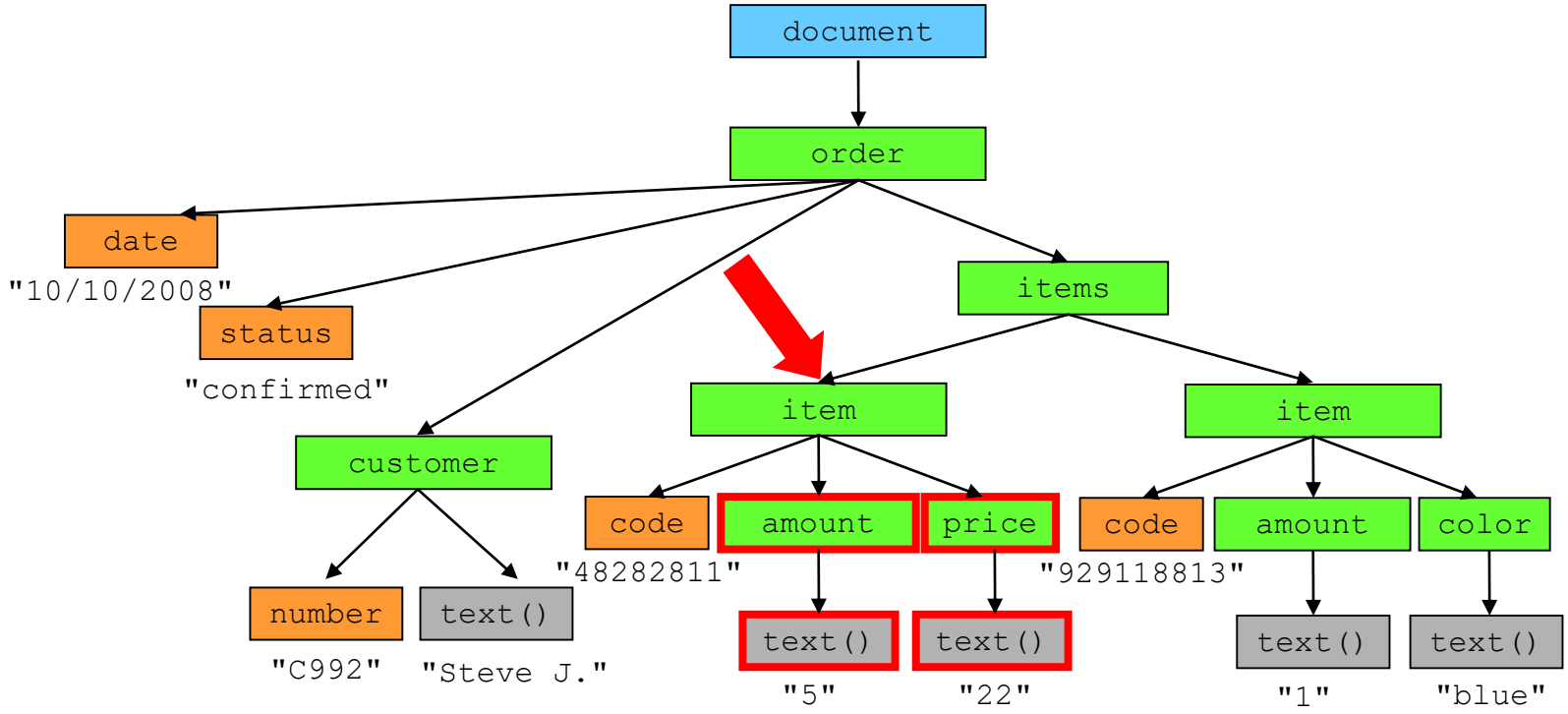
XPath Axis child

- All child nodes of node u



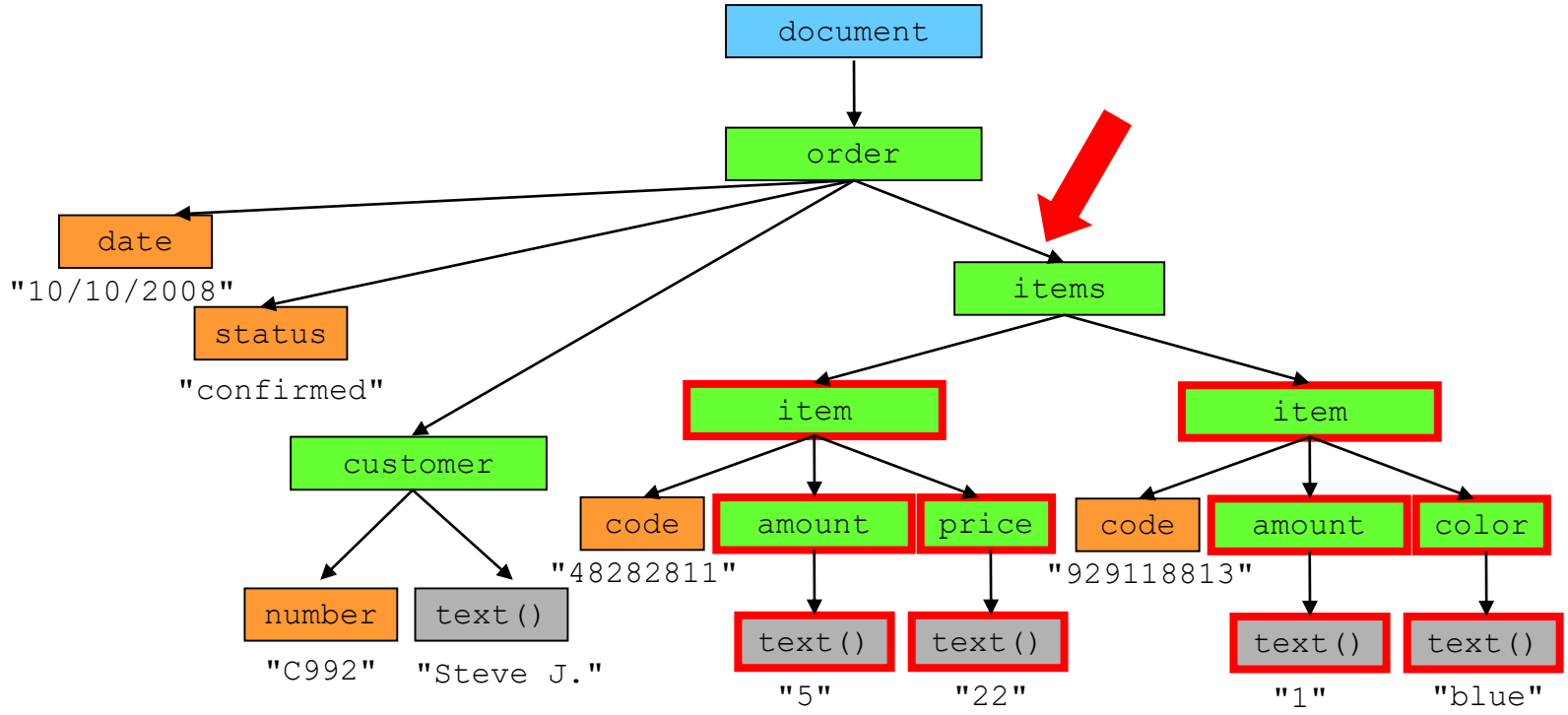
XPath Axis descendant

- All descendant nodes of node u



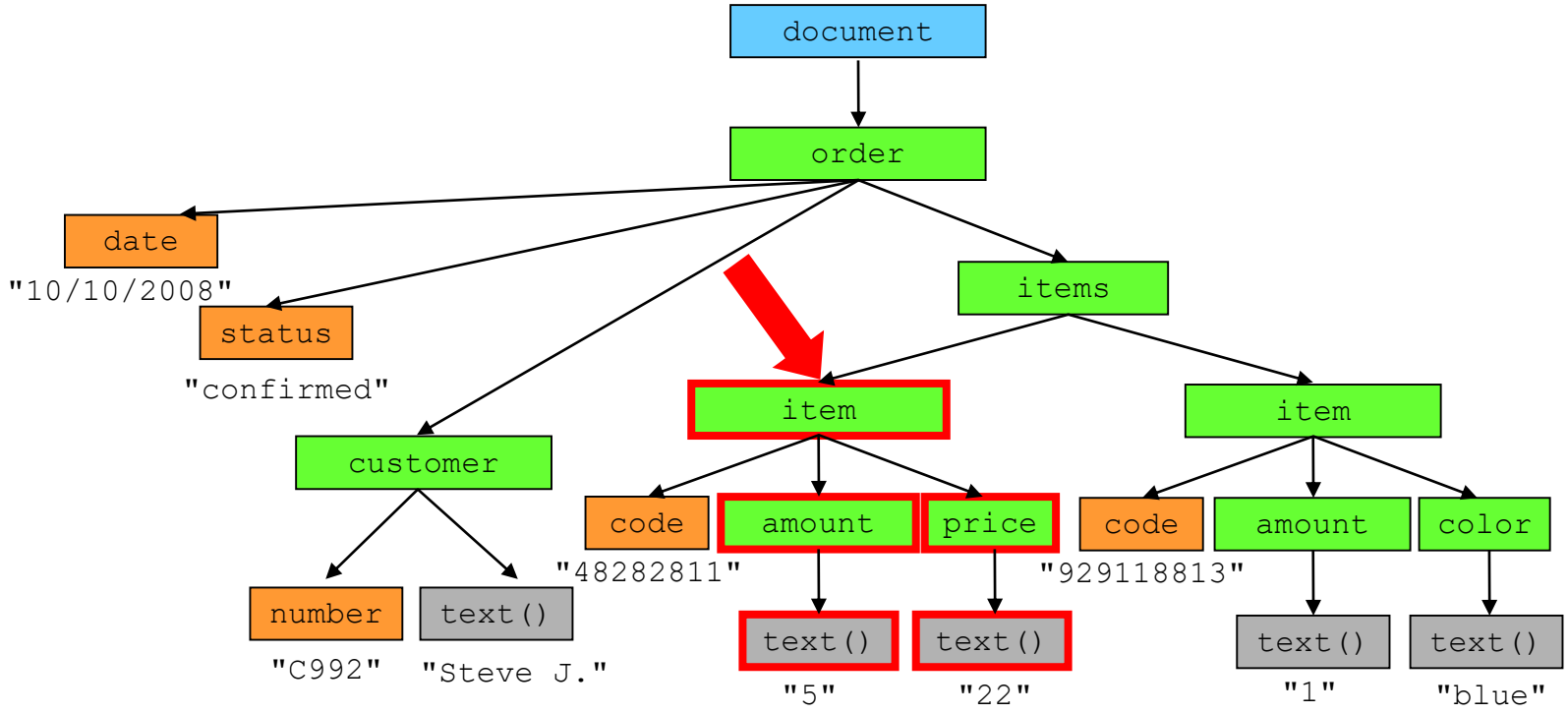
XPath Axis descendant

- All descendant nodes of node u

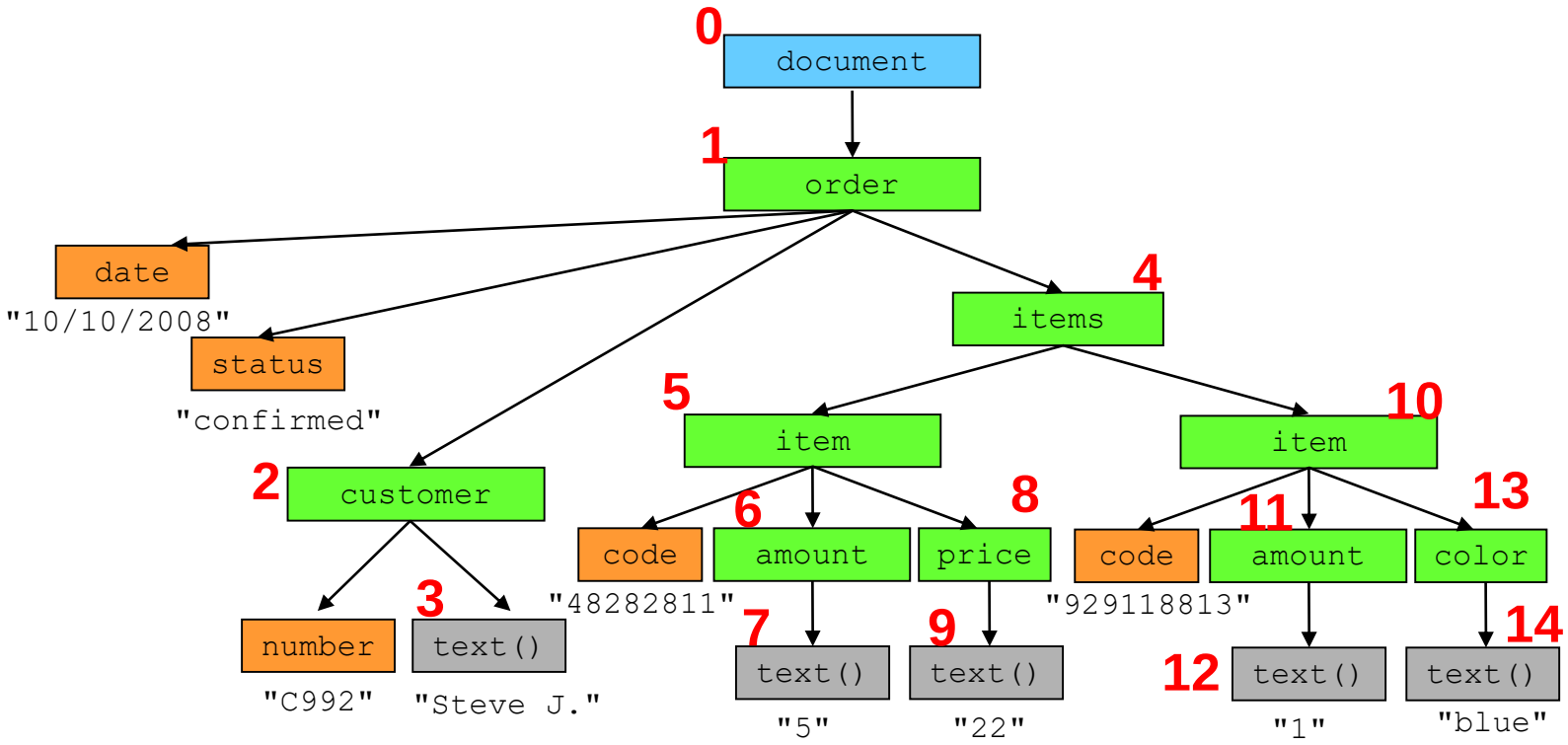


XPath Axis descendant-or-self

- All descendant nodes of node u including u

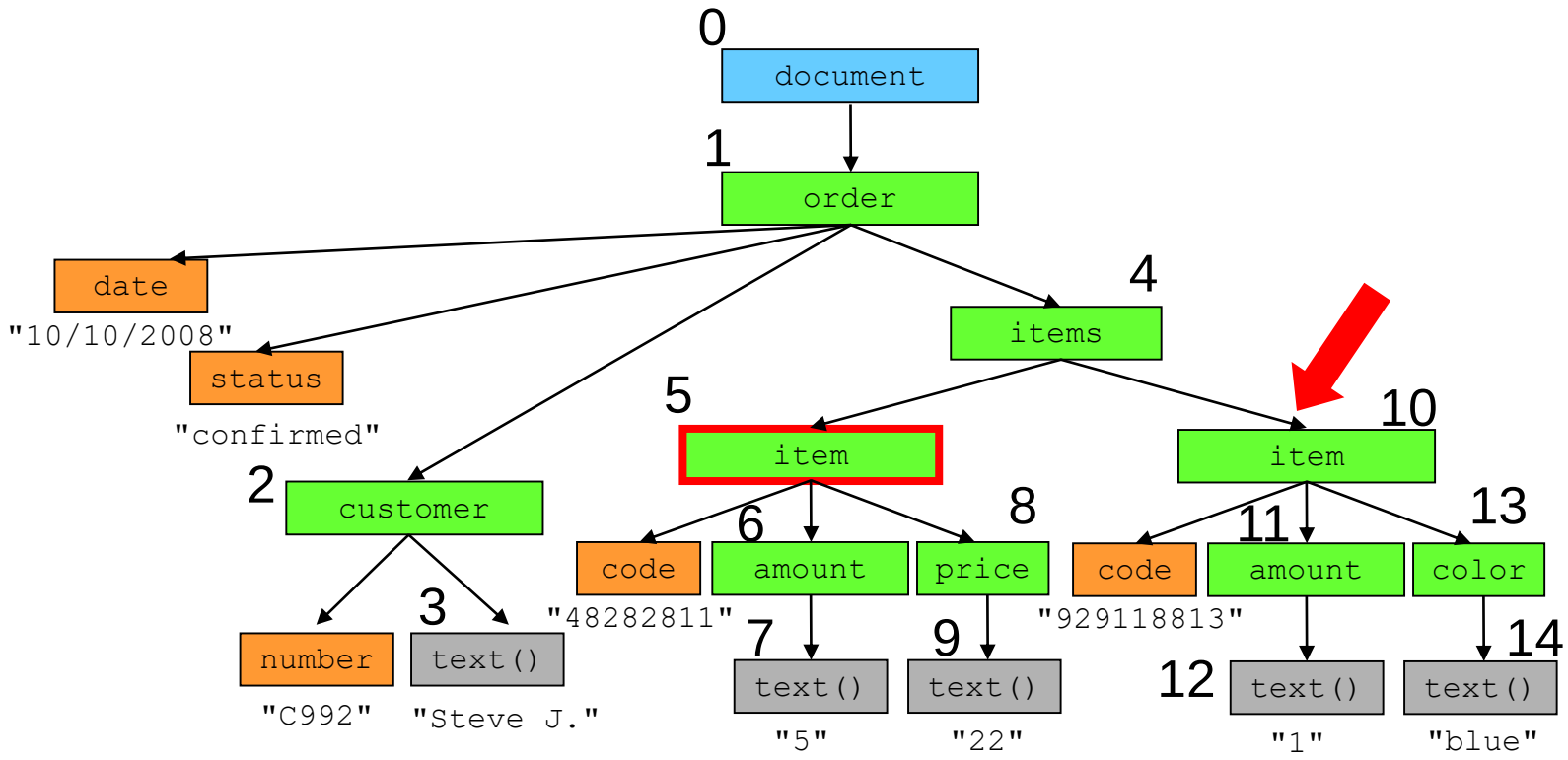


Tree Traversal



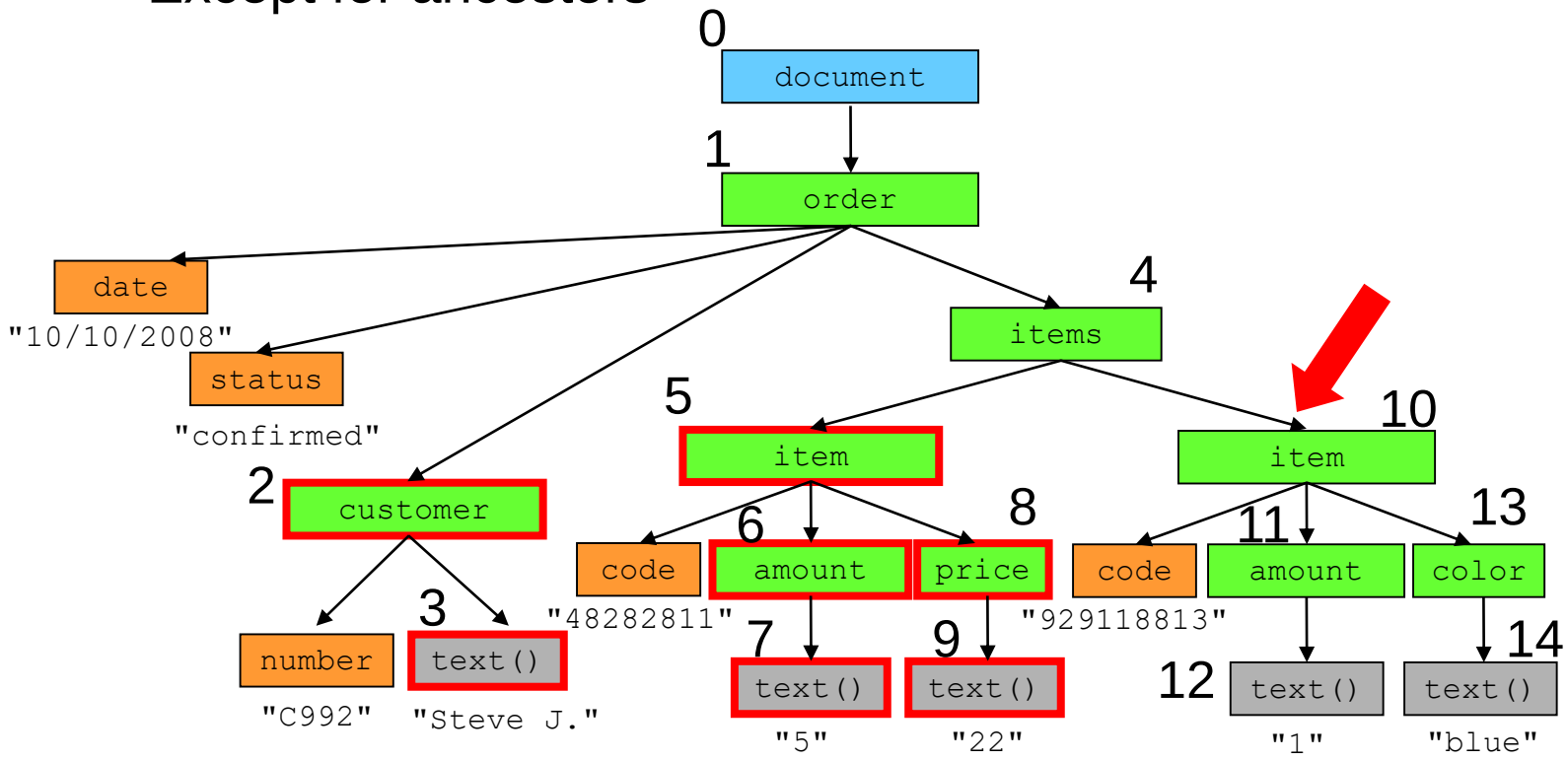
XPath Axis preceding-sibling

- All siblings of u which precede it in tree traversal



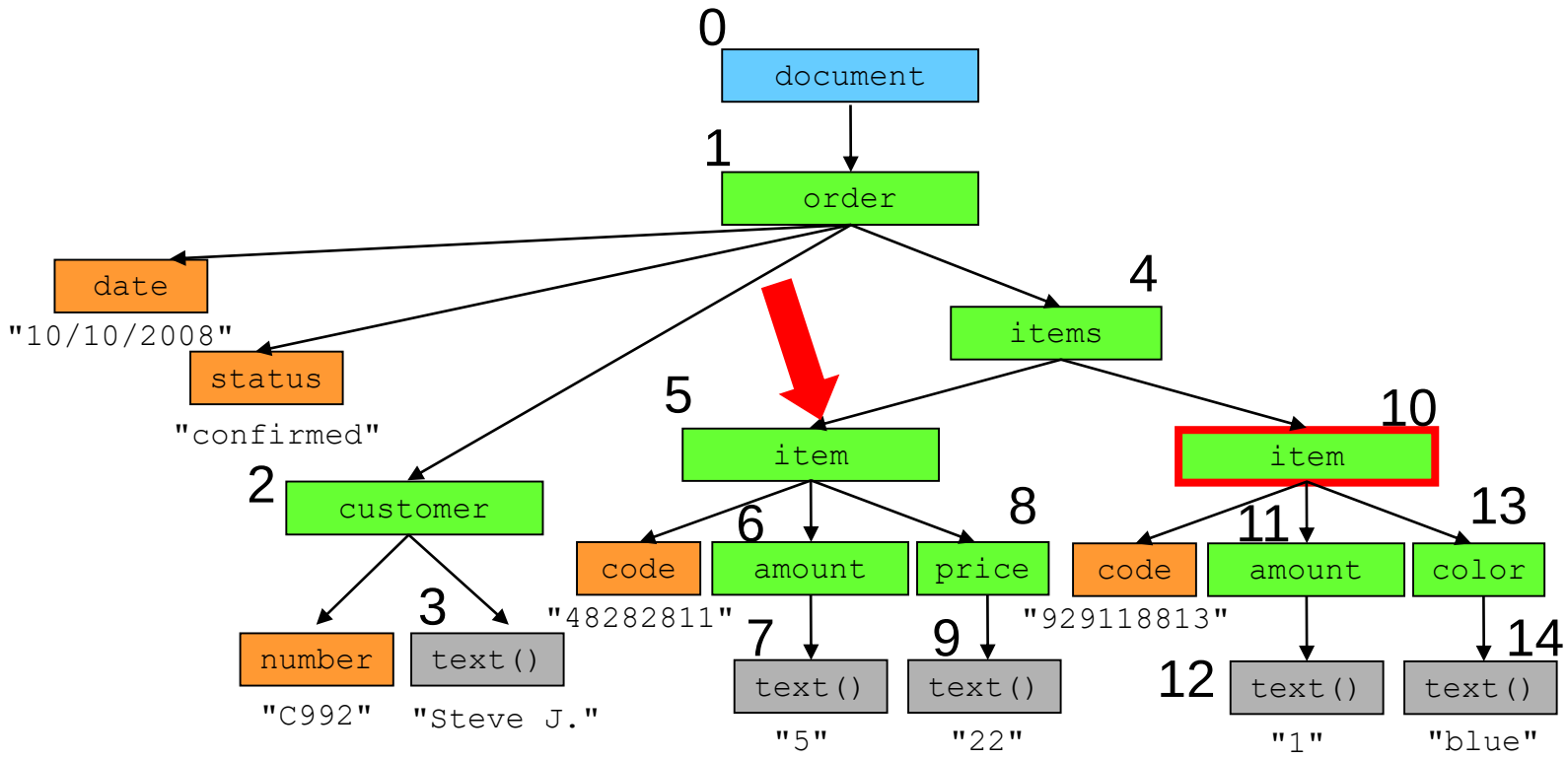
XPath Axis preceding

- All nodes which precede u in tree traversal
  - Except for ancestors



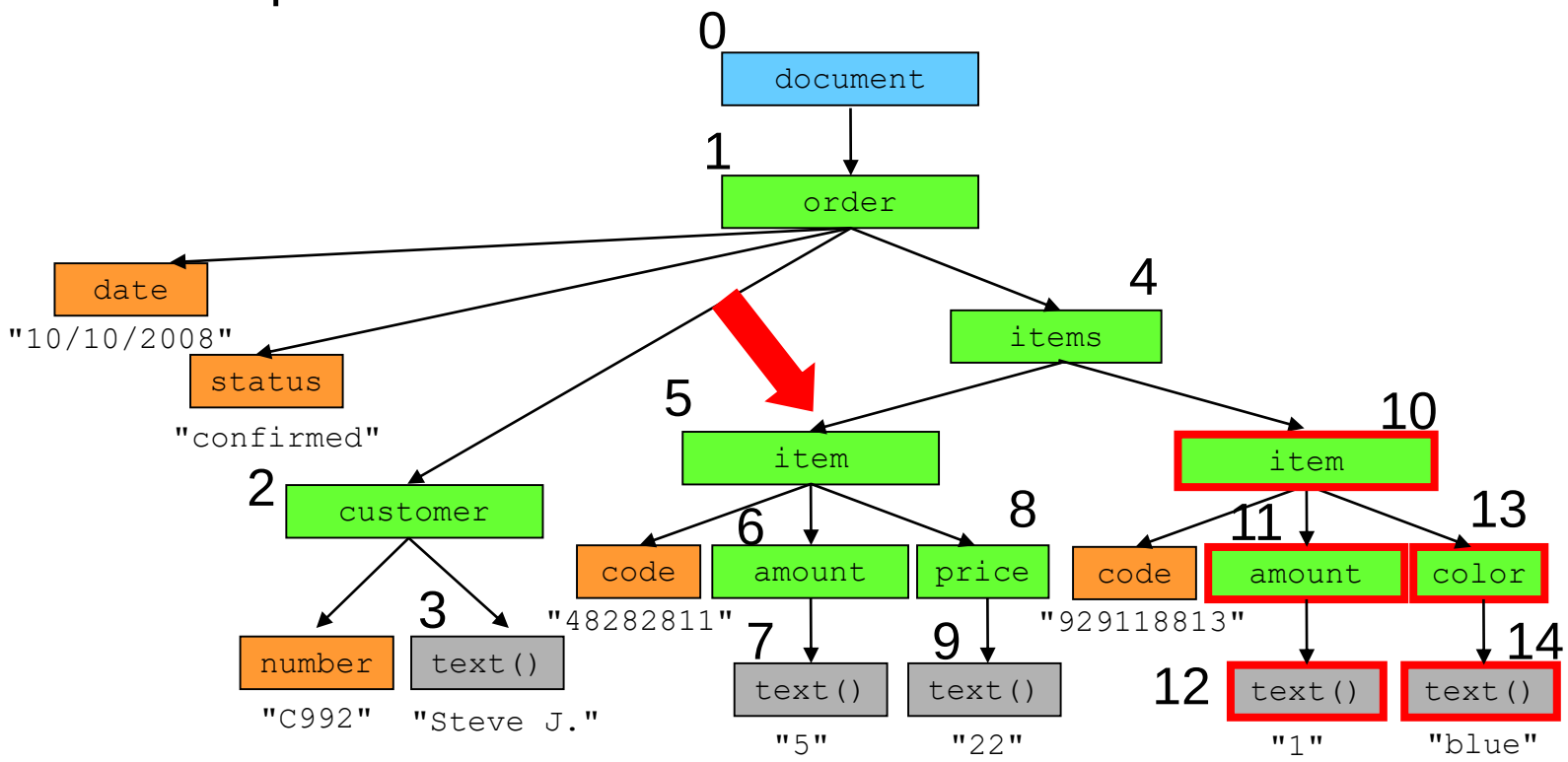
XPath Axis following-sibling

- All siblings of u which follow it in tree traversal



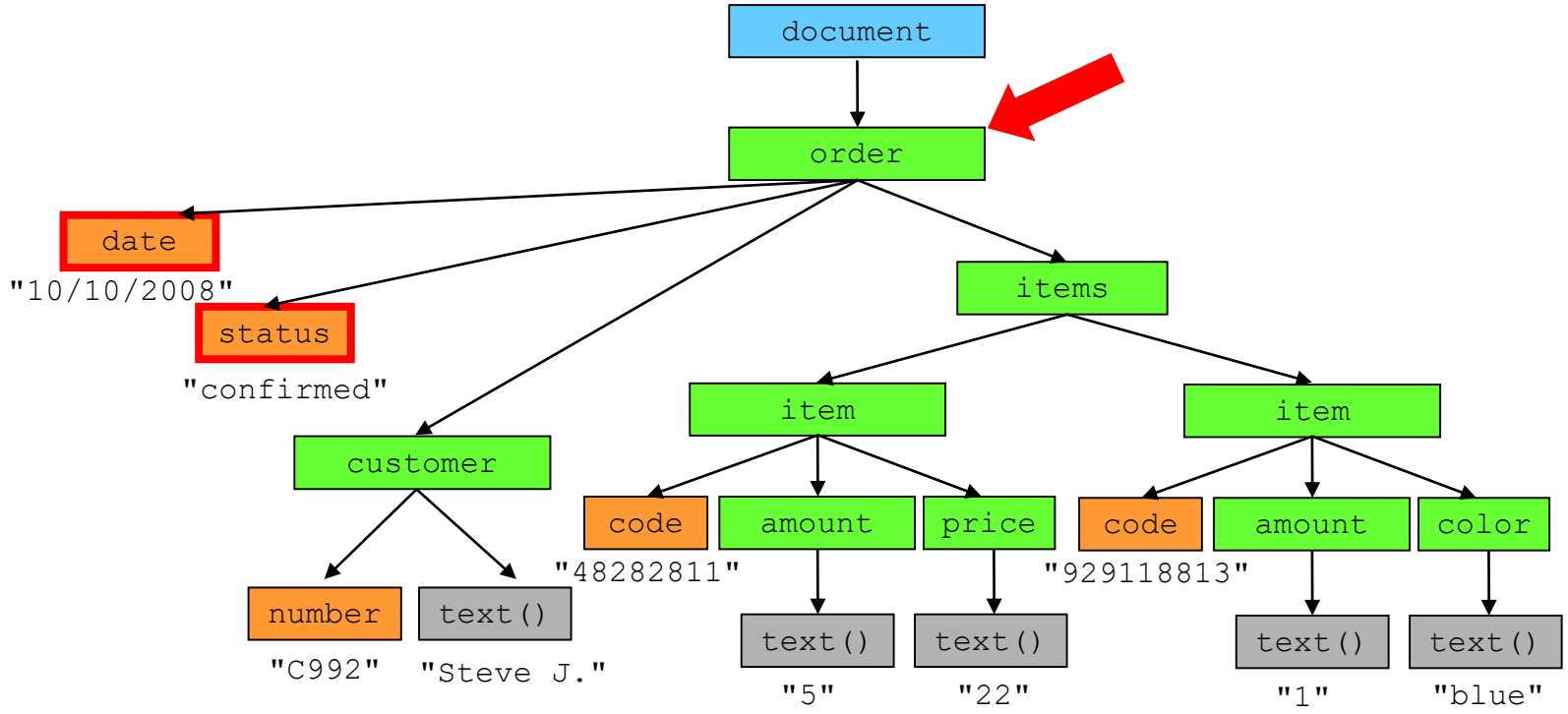
XPath Axis following

- All nodes which follow u in tree traversal
  - Except for ancestors



XPath Axis attribute

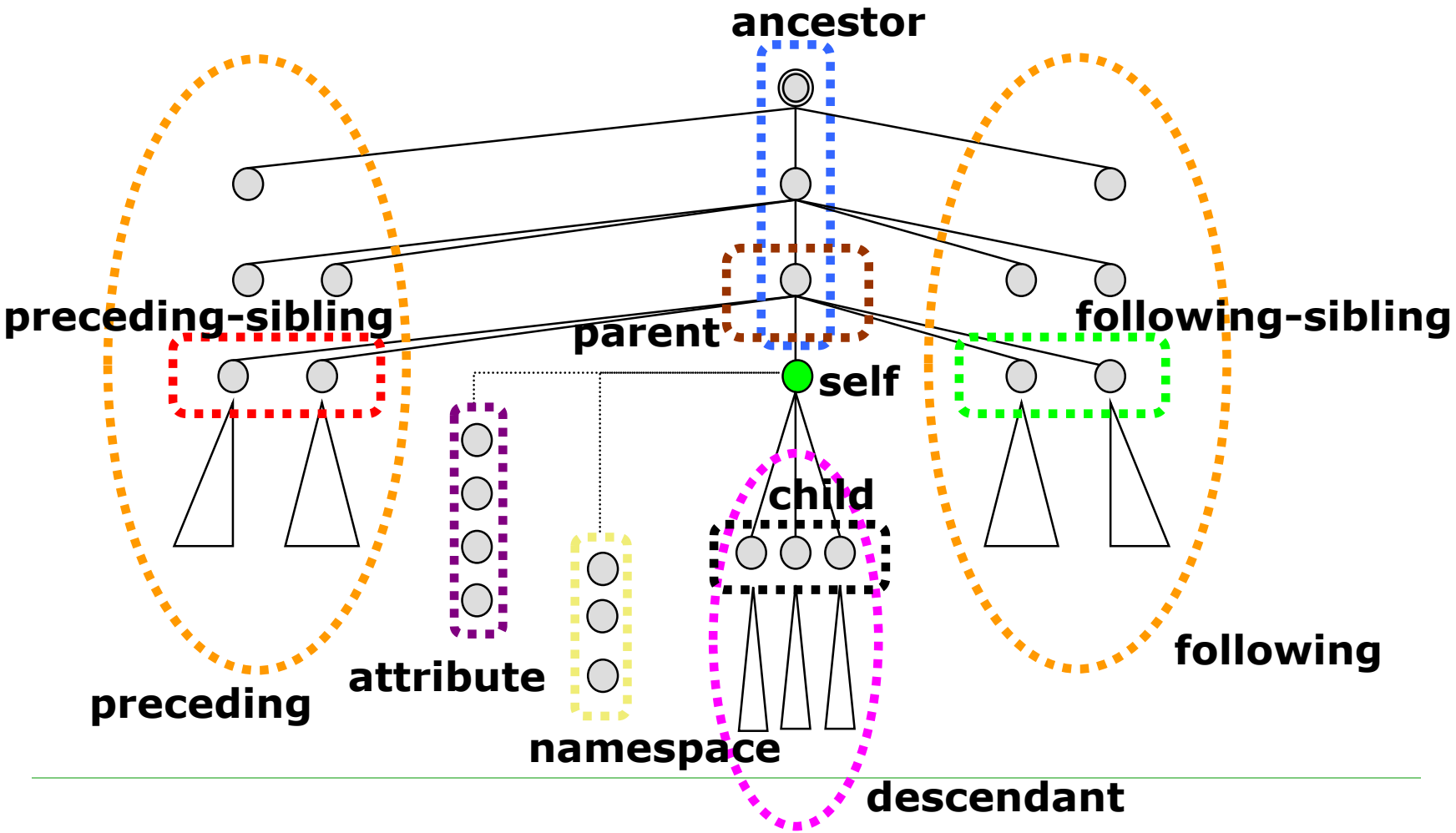
- All attributes of node u



XPath Axis namespace

- All namespace declarations of node u
  - Similarly to axis attribute it selects nodes of a special type

All XPath Axes



XPath Node Test

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

- Tests nodes selected by the axis
  - Node type, node name

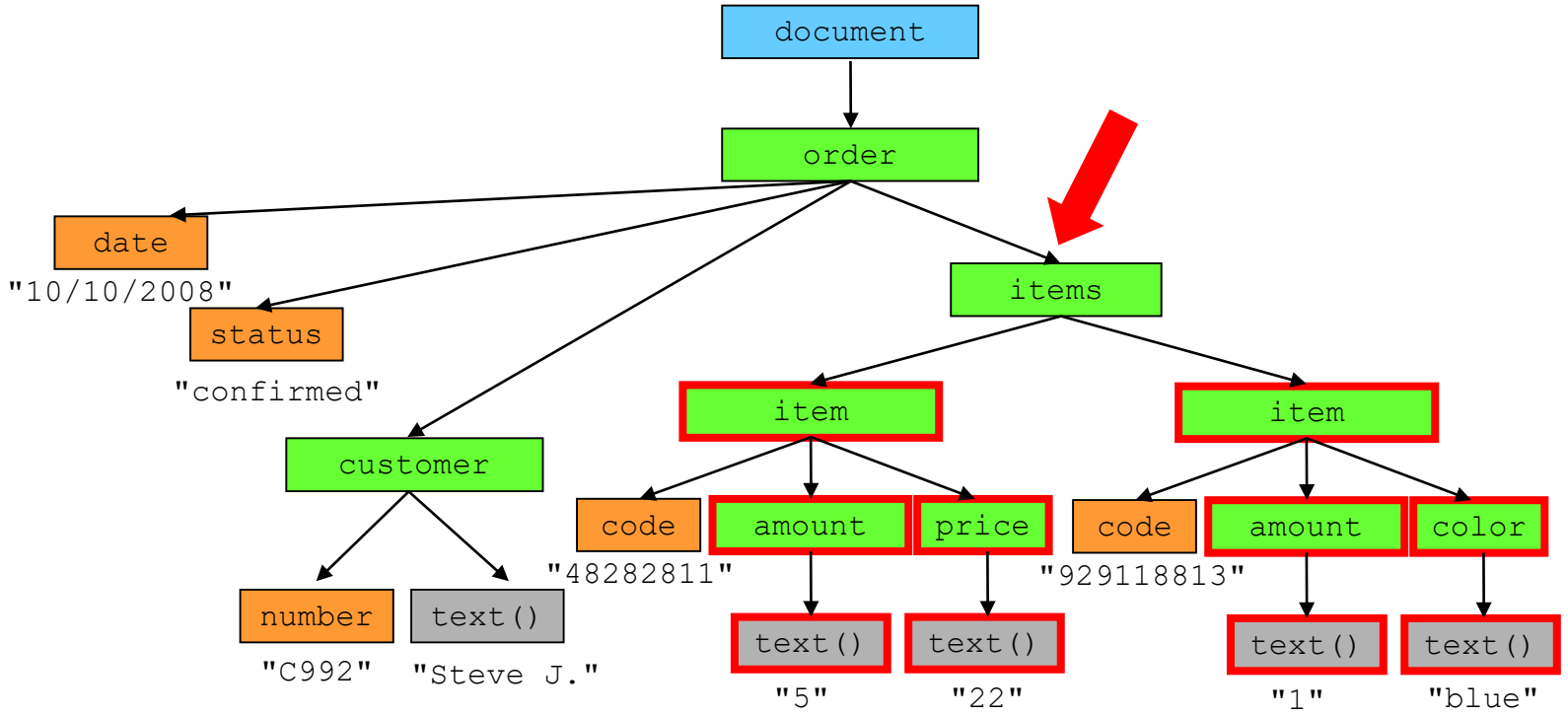
XPath Node Test

```
axis::node() predicate1 ... predicateN
```

- All nodes selected by the axis

XPath Node Test

descendant::node()



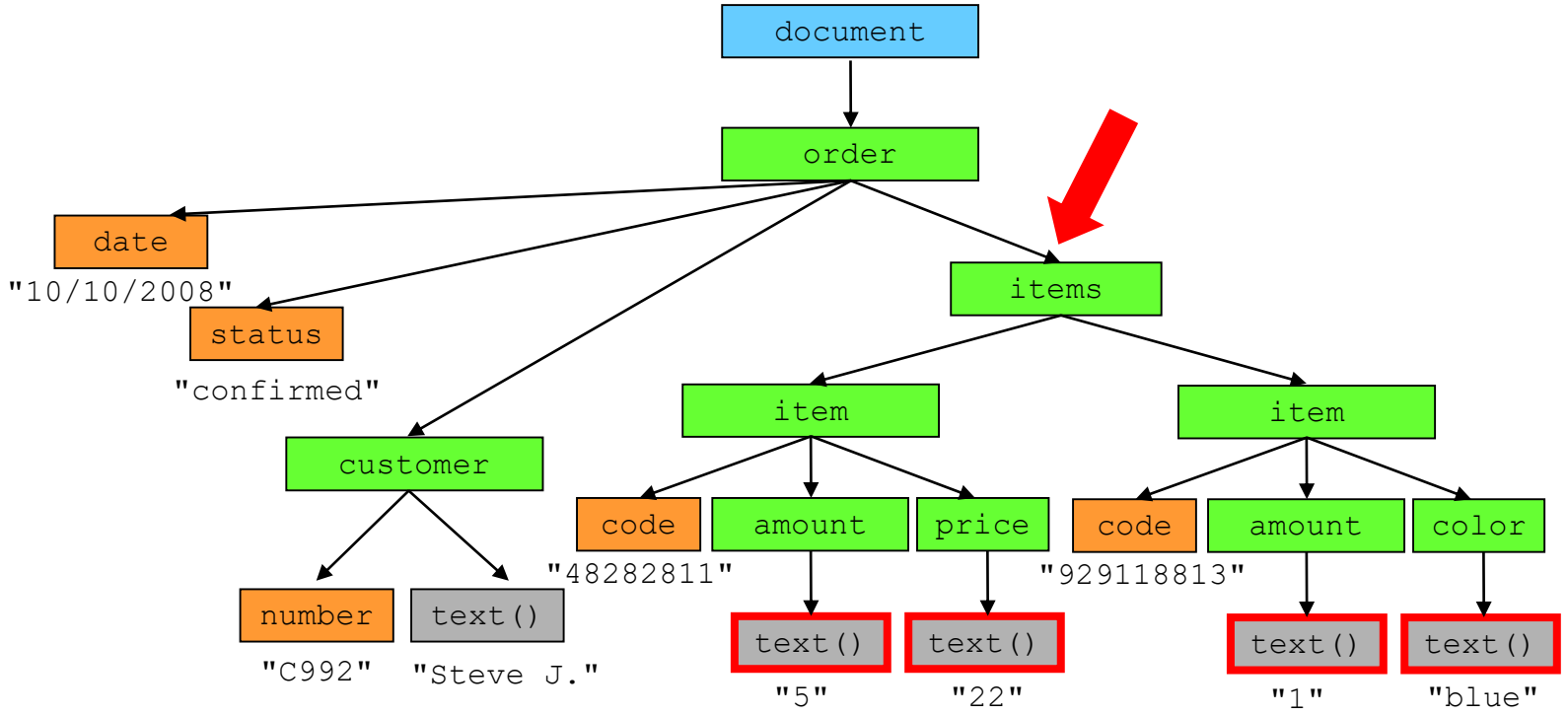
XPath Node Test

```
axis::text() predicate1 ... predicateN
```

- All text nodes selected by the axis

XPath Node Test

descendant::text()



XPath Node Test

```
axis::* predicate1 ... predicateN
```

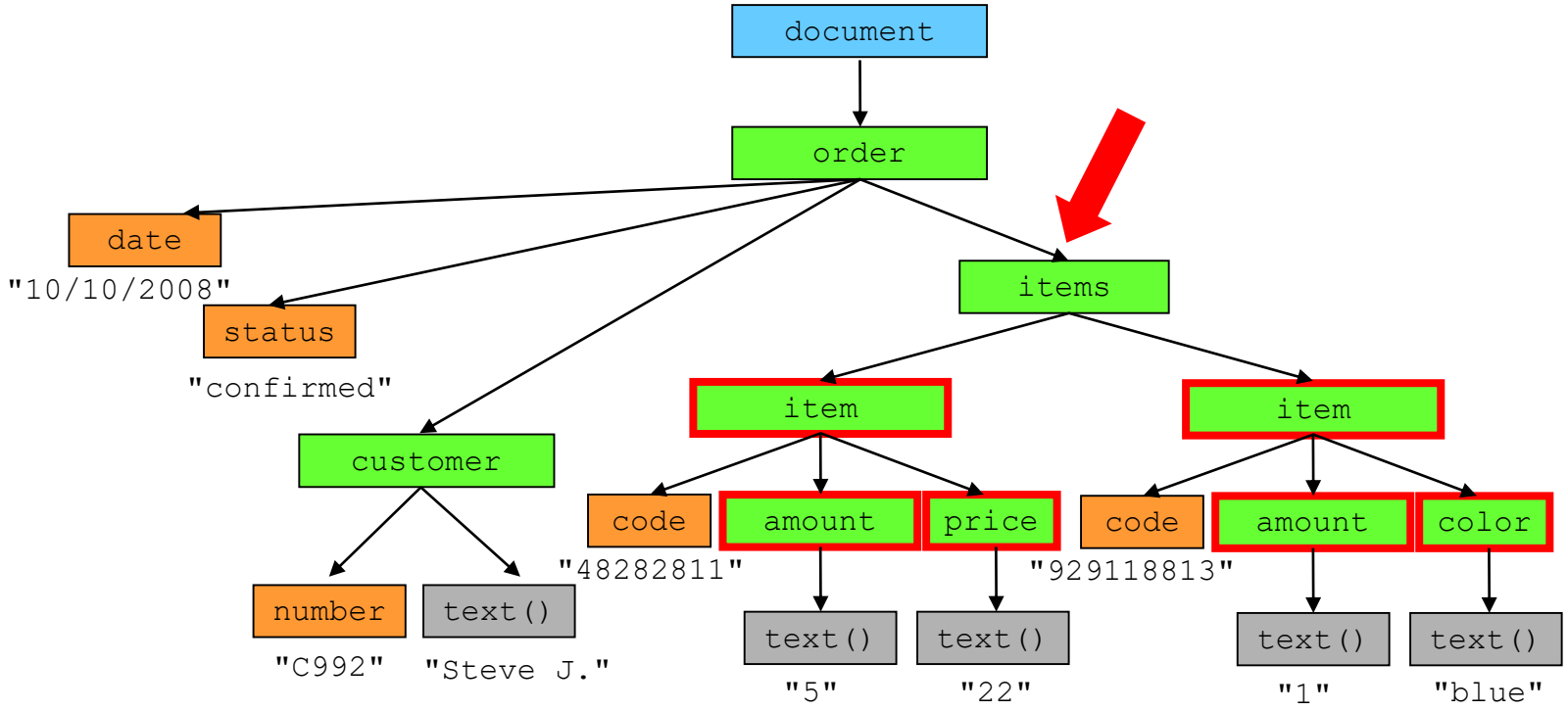
- All nodes selected by the axis which have a name
  - Name can have an element or an attribute
  - Note: there exists no axis that enables to selected elements and attributes at the same time

```
axis::name predicate1 ... predicateN
```

- All nodes with the specified name

XPath Node Test

descendant::*



XPath Node Test

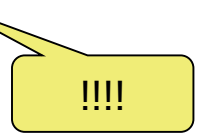
- `axis::comment()`
- `axis::processing-instruction()`
- `axis::processing-instruction("php")`

XPath Axes and Node Test Abbreviations

- For the most commonly used axes and node tests

```
…/…    <=> …/child::…
…/@…   <=> …/attribute::…
…/.…   <=> …/self::node()…
…/..…  <=> …/parent::node()…
…//…   <=> …/descendant-or-self::node()/…
```

  - `//customer` selects all elements `customer` in XML document



XPath Predicates

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

- A predicate enables to specify advanced conditions for nodes which were selected by the axis and node test
  - For context node u we find all nodes selected by the axis from node u
  - On input we put those which satisfy node test and all predicates

```
predicate ::= '[' condition ']'
condition ::= 'not(' condition ')' |
              condition 'and' condition |
              condition 'or' condition
```

```
[condition1][condition2] <=> [condition1 and condition2]
```

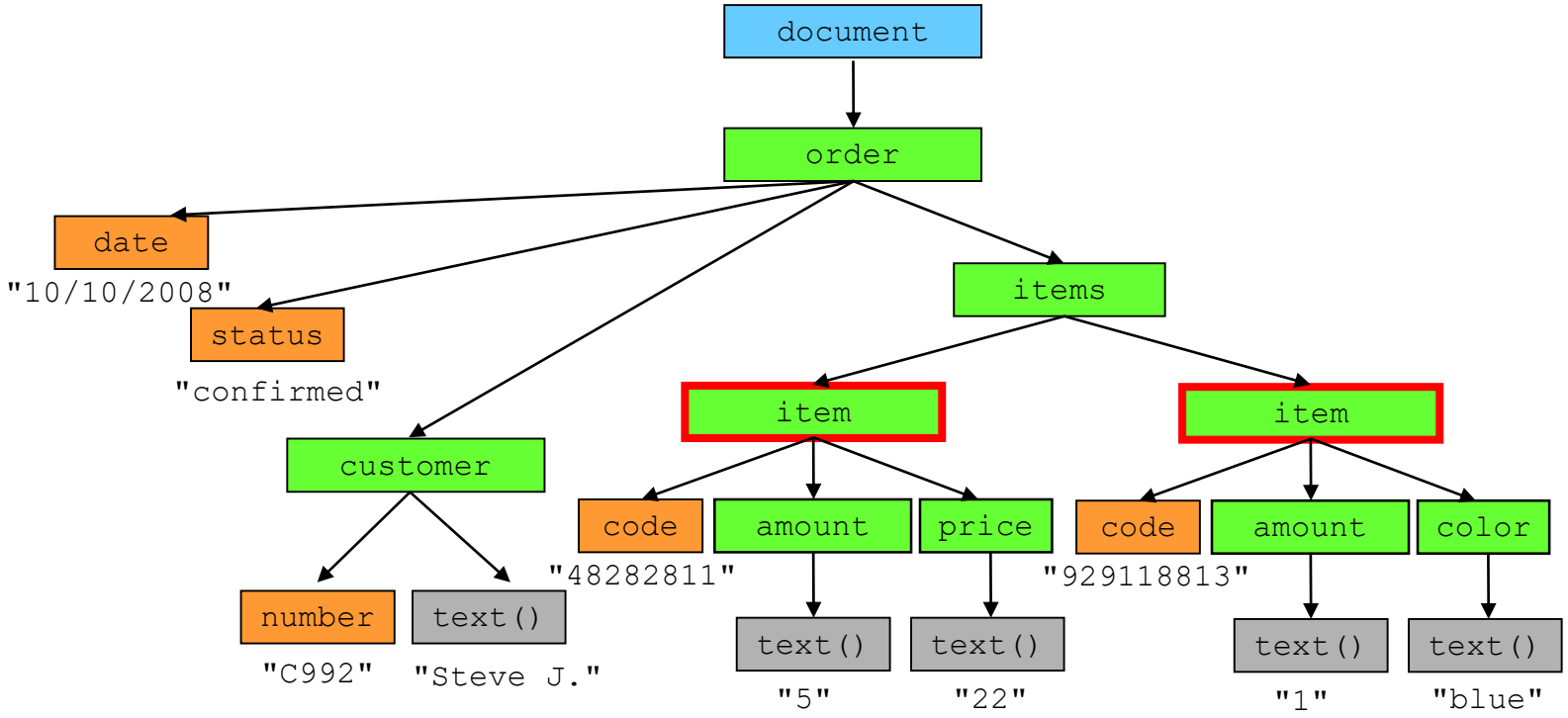
XPath Predicates

- Predicate condition can be a relative XPath path P
  - For node u it returns true if the set of nodes returned by path P from u is non-empty
- Predicated condition can be an absolute XPath path P
  - It returns true if the set of nodes returned by path P is non-empty

XPath Predicates

//item[@code]

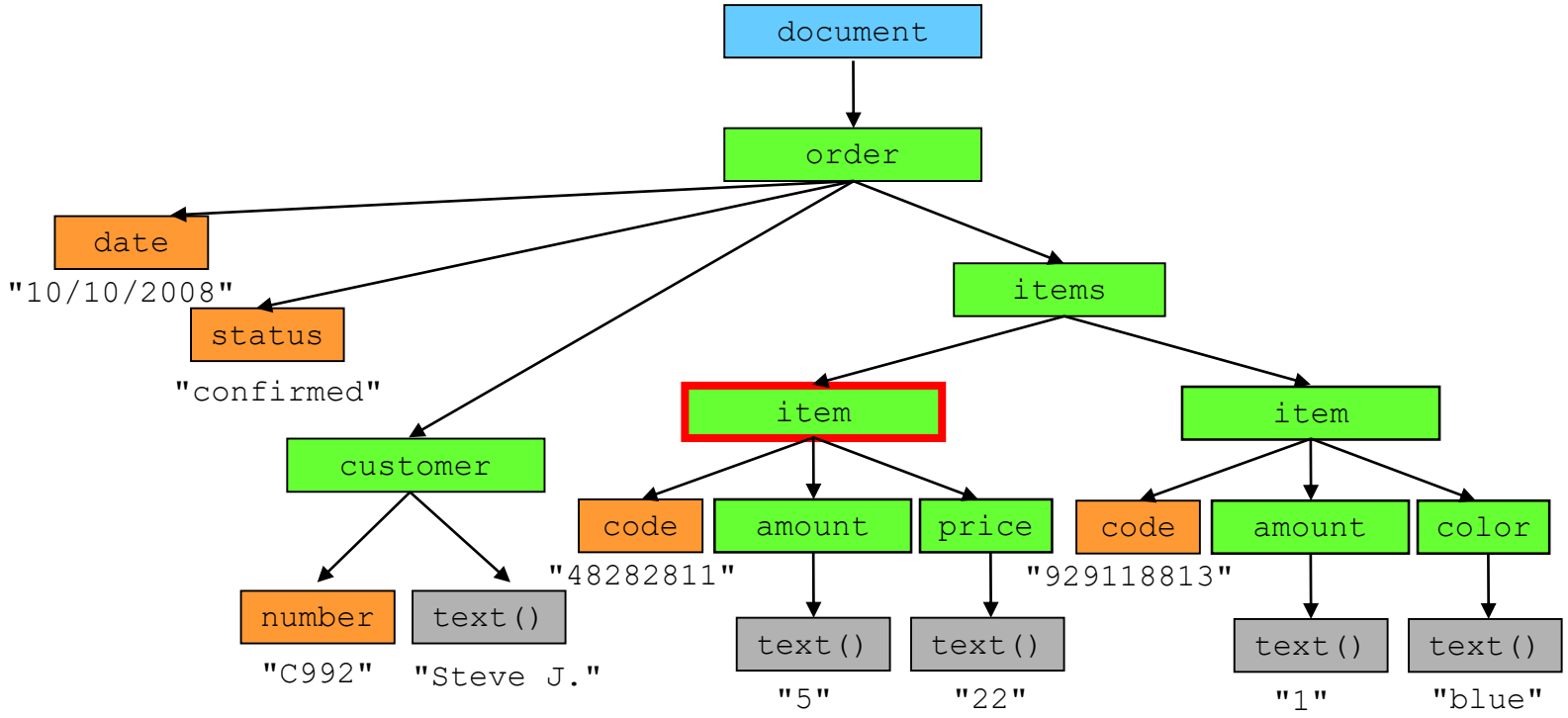
Relative path: **attribute::code**



XPath Predicates

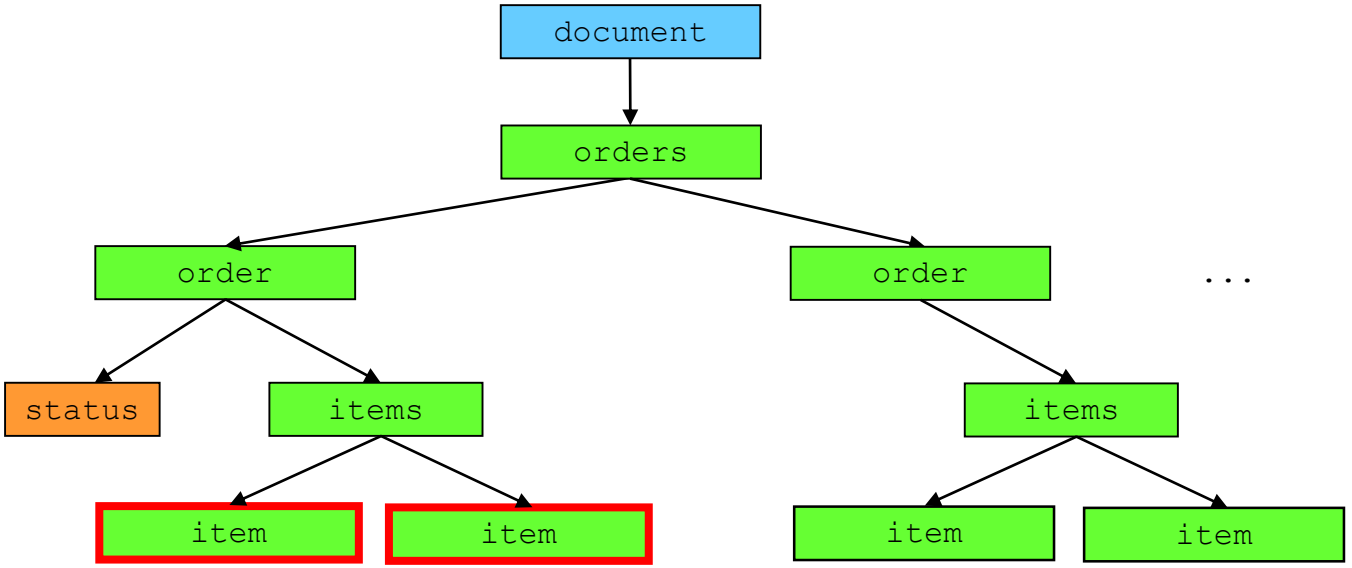
//item[price]

Relative path: **child::price**



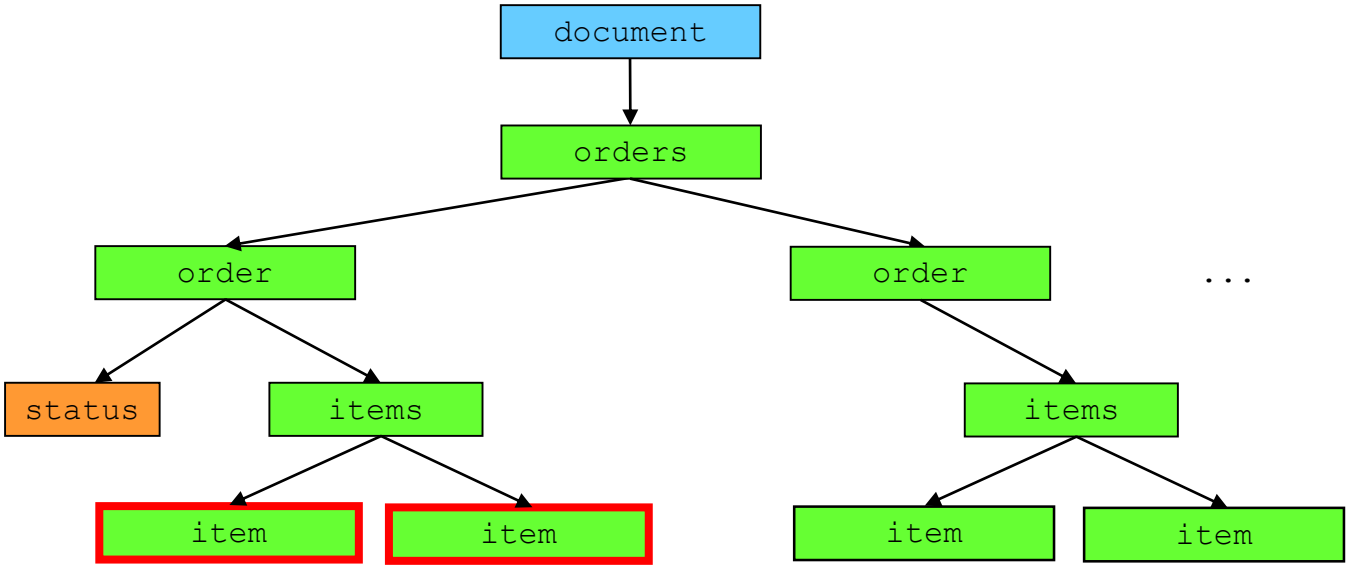
XPath Predicates

```
//item[../../@status]
```



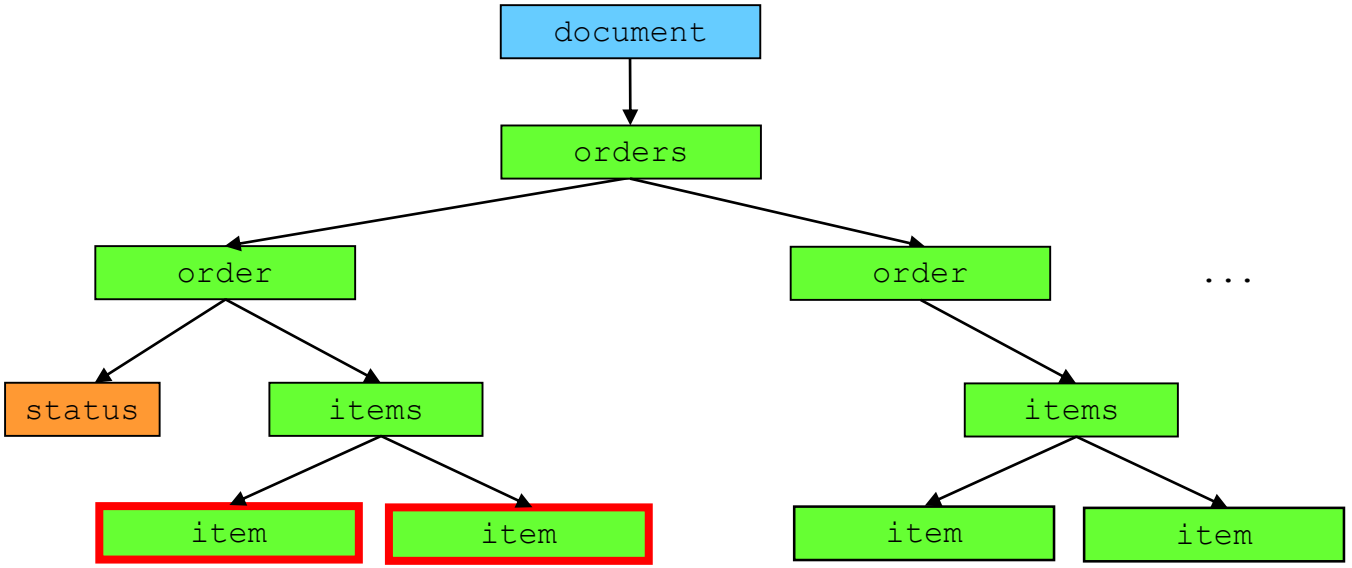
XPath Predicates

```
//item[ancestor::order/@status]
```



XPath Predicates

```
//order[@status]//item
```

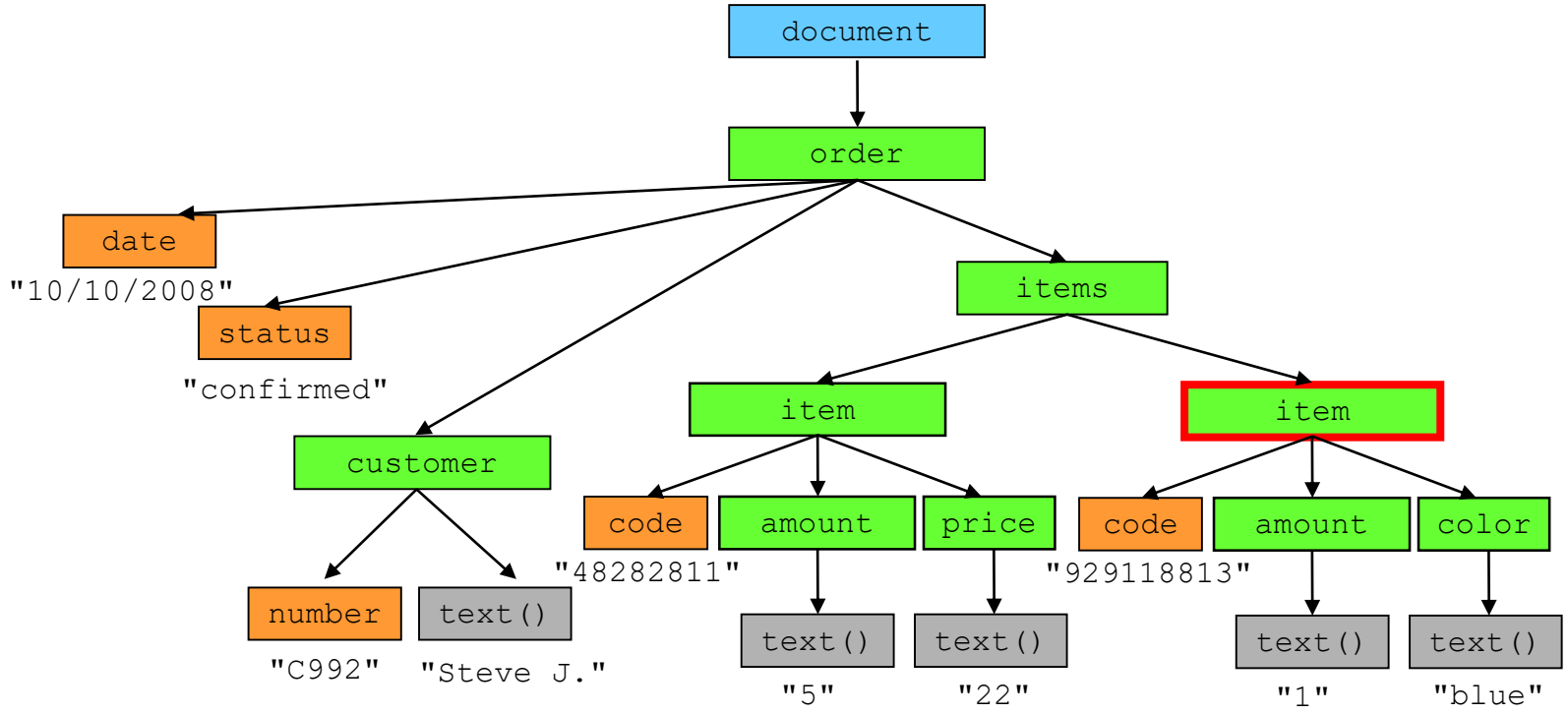


XPath Predicates

- The condition can involve comparison of two operands
  - Operands are XPath expressions
    - XPath path, value, ...
  - Operators are = != < > <= >=
  - String value of node
    - Attribute – normalized value
    - Element – concatenation of text nodes in its subtree

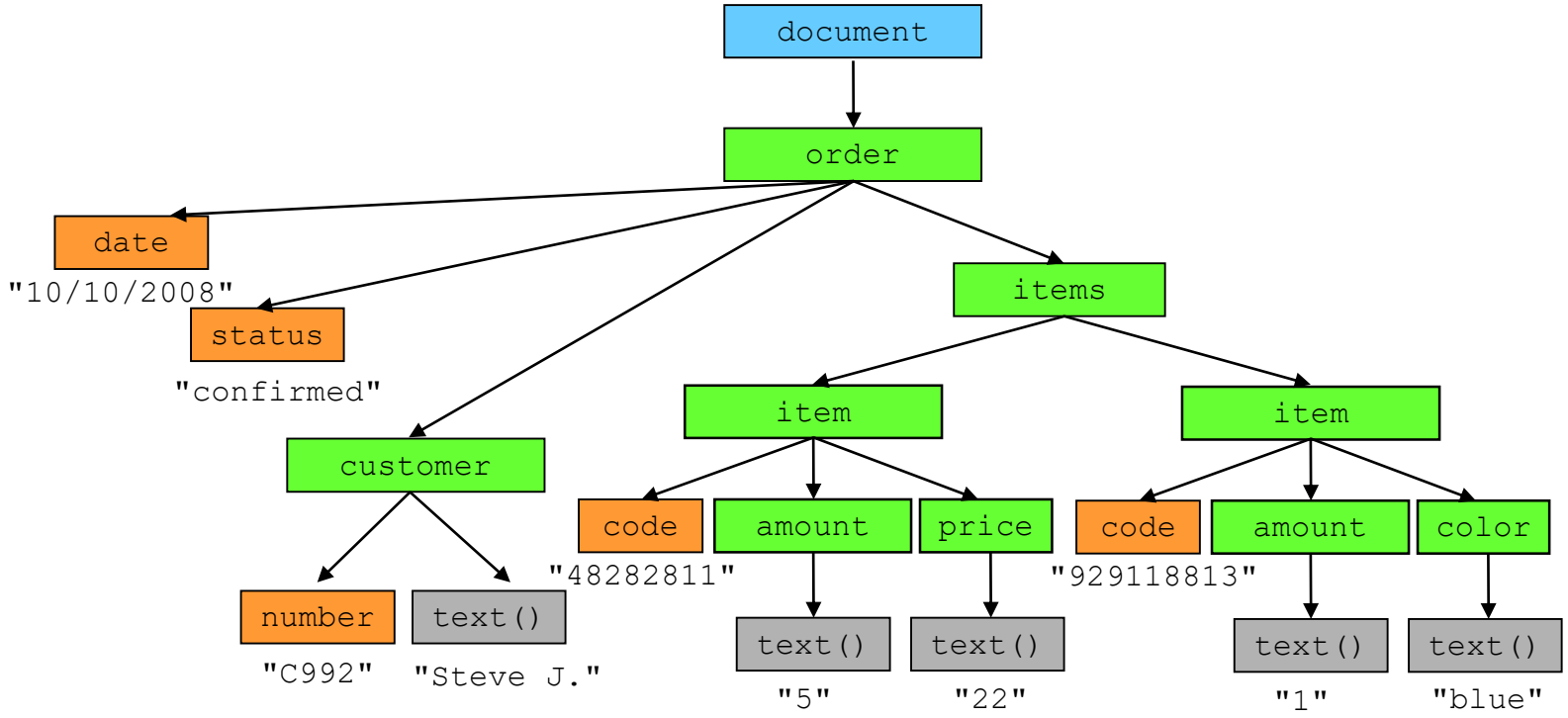
XPath Predicates

```
//item[color = "blue"]
```



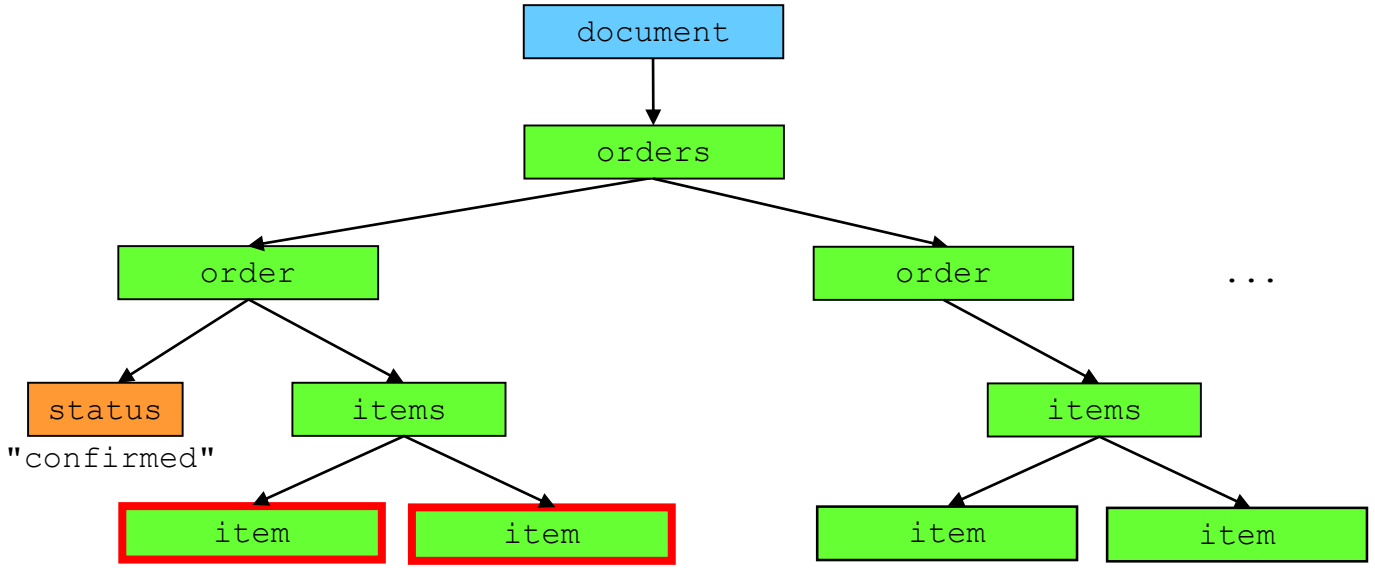
XPath Predicates

```
//item[price > 30]
```



XPath Predicates

```
//order[@status = "confirmed"]//item
```



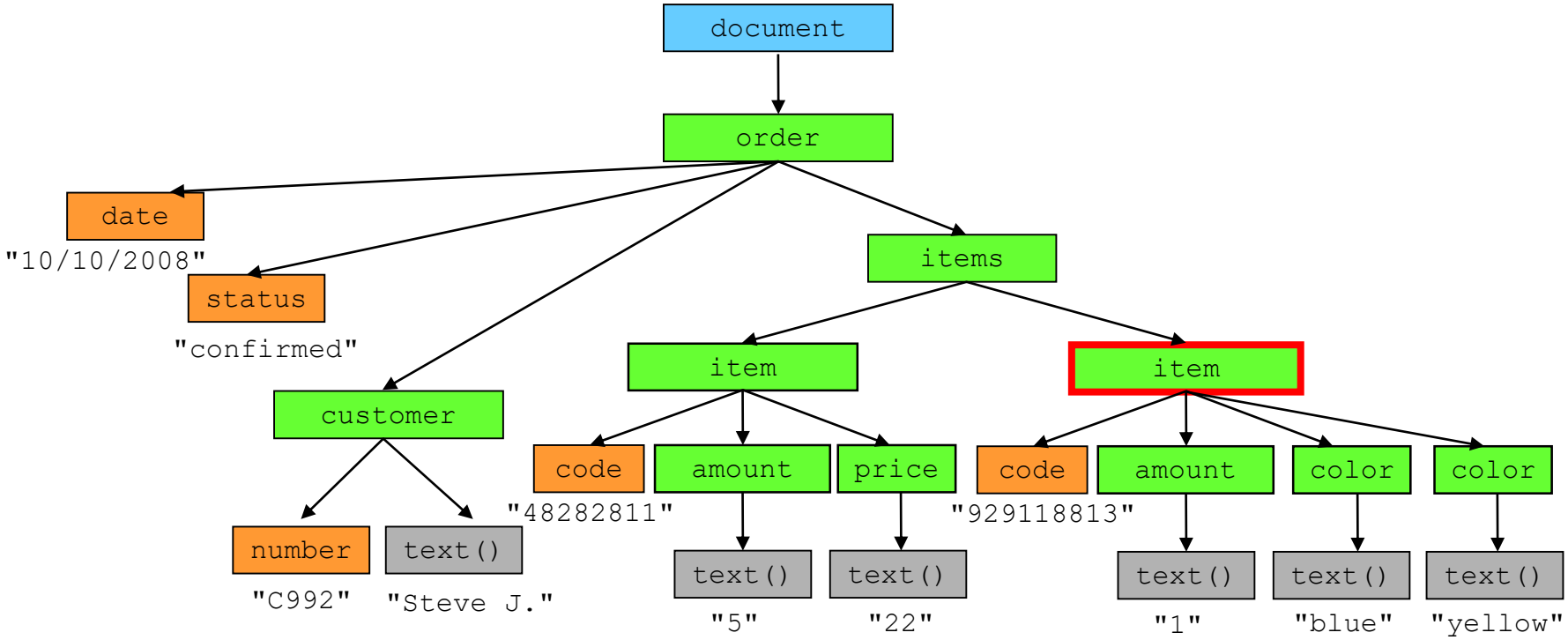
XPath Predicates

- Operators = != ...
  - Operands are sets of values/nodes
  - Evaluated as true if there exists a value/node in the left operand and a value/node in the right operand for which the operator evaluates as true

- Consequences:
  - Expression with `=` and `!=` can return the same result!
  - `x="foo"` is **not** the same as `not(x!="foo")`
    - There exists a node in x with string value `foo`
    - All nodes in x have string value `foo`

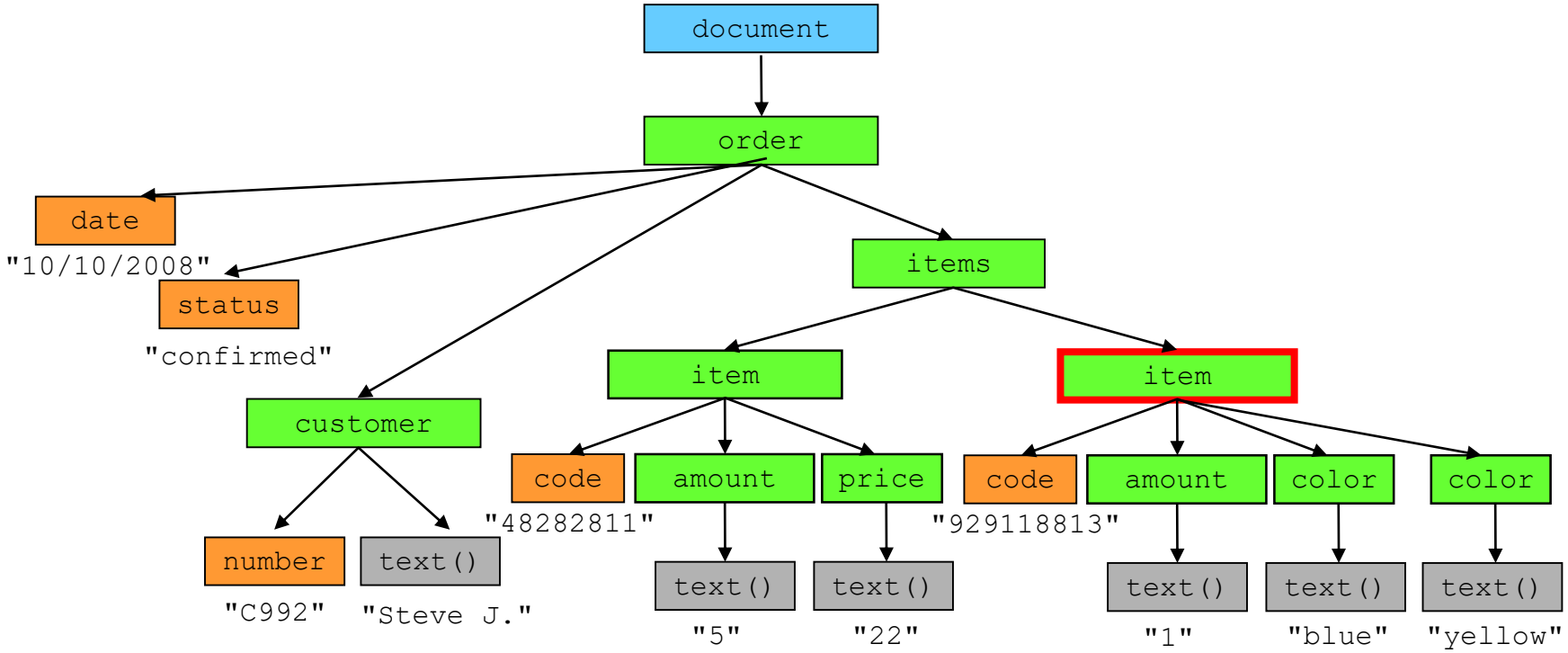
XPath Predicates

```
//item[color = "blue"]
```



XPath Predicates

```
//item[color != "blue"]
```

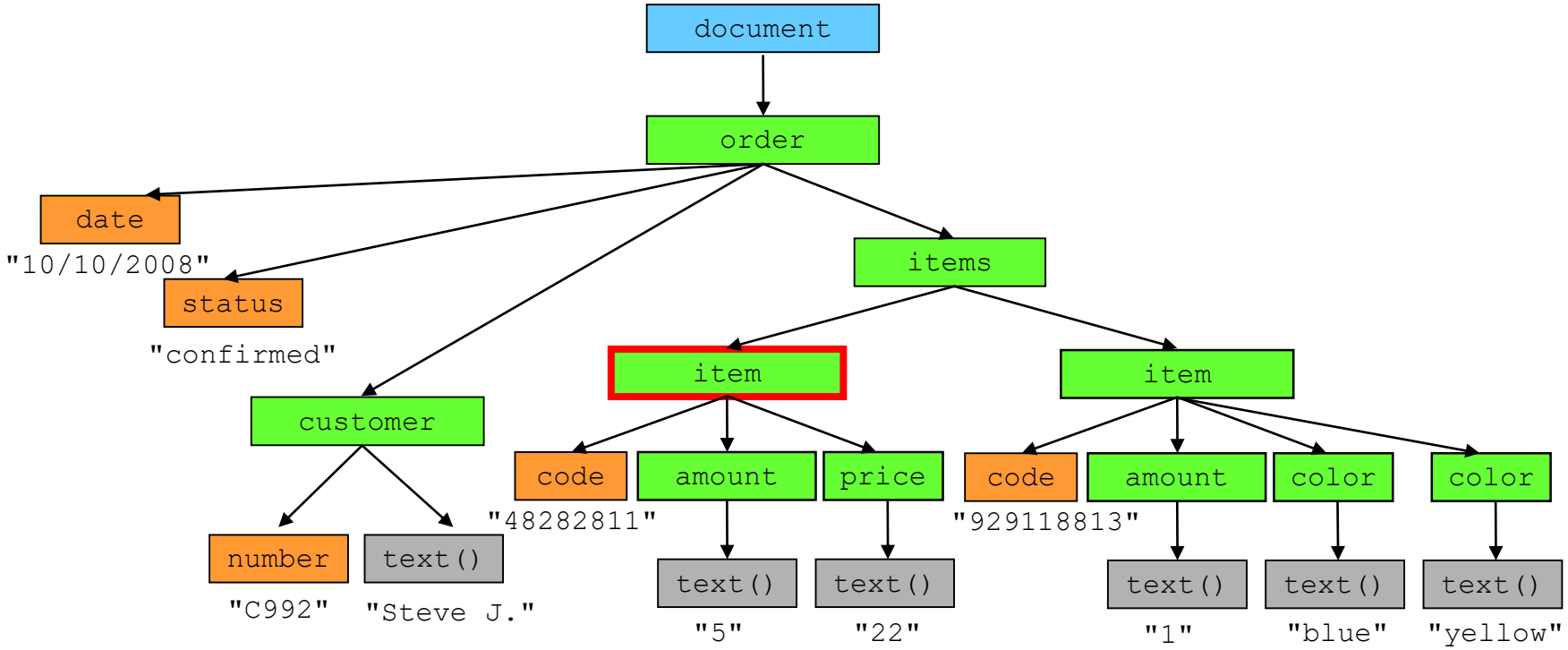


Built-in Functions

- Testing of position
  - Each node in a context set has a position
    - Determined by its position in document and the (direction of a) particular path
  - `position()`
    - Returns the position of node in a context set
  - `last()`
    - Returns the number of nodes in a context set

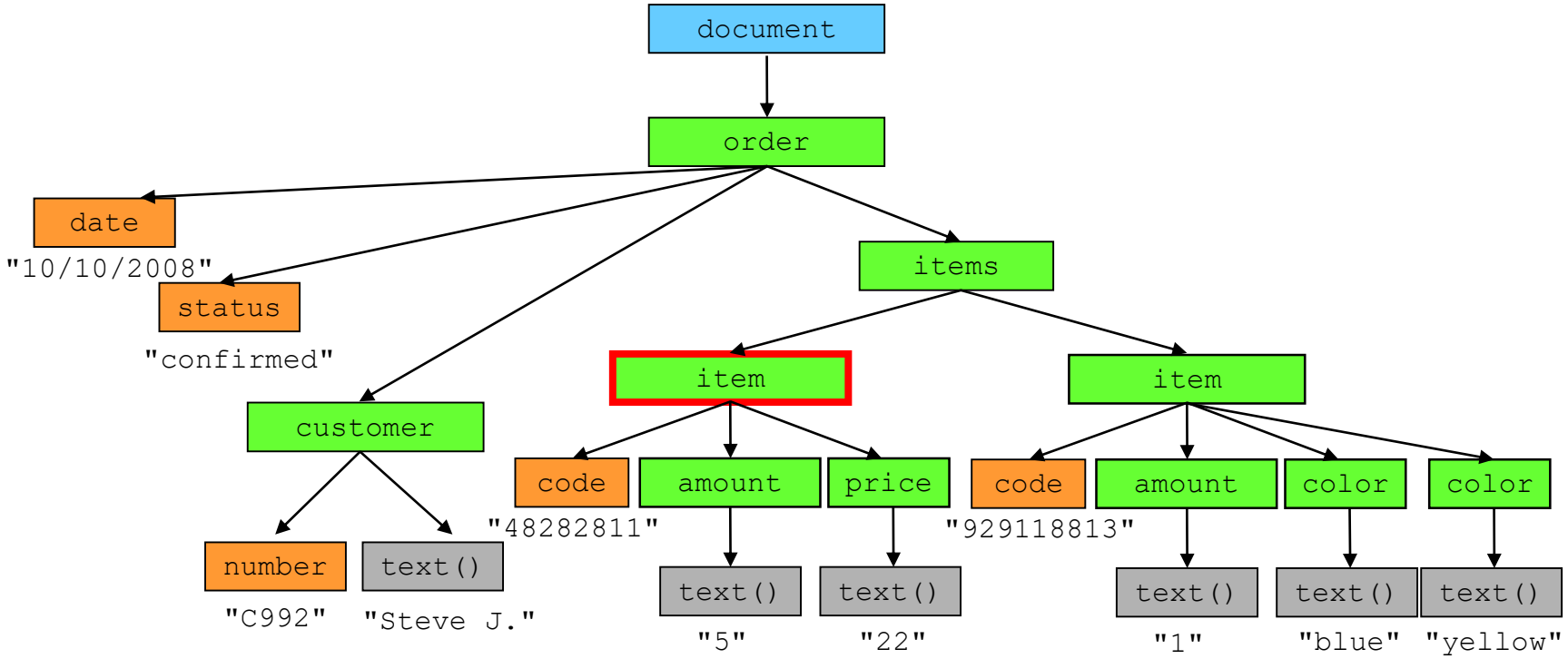
Built-in Functions

```
//items/item[position() = 1]
```



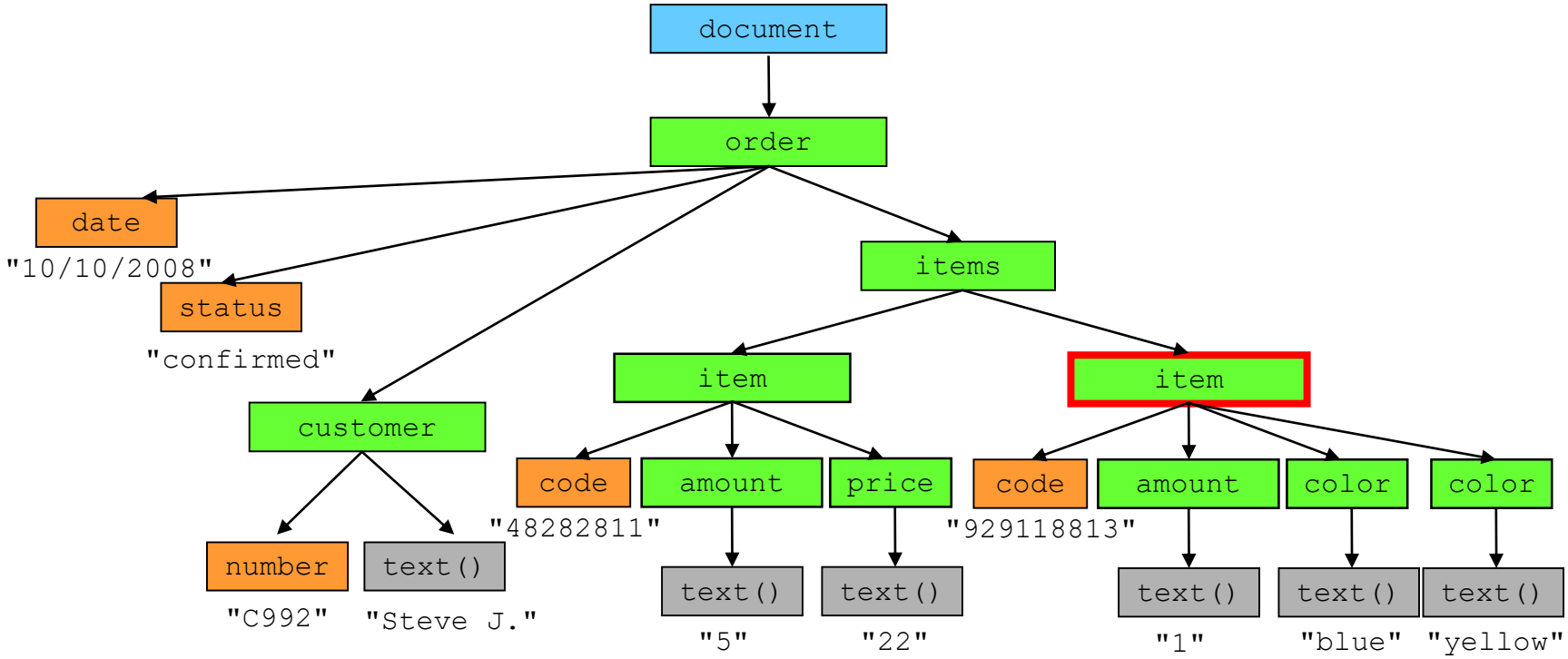
Built-in Functions

```
//items/item[1]
```



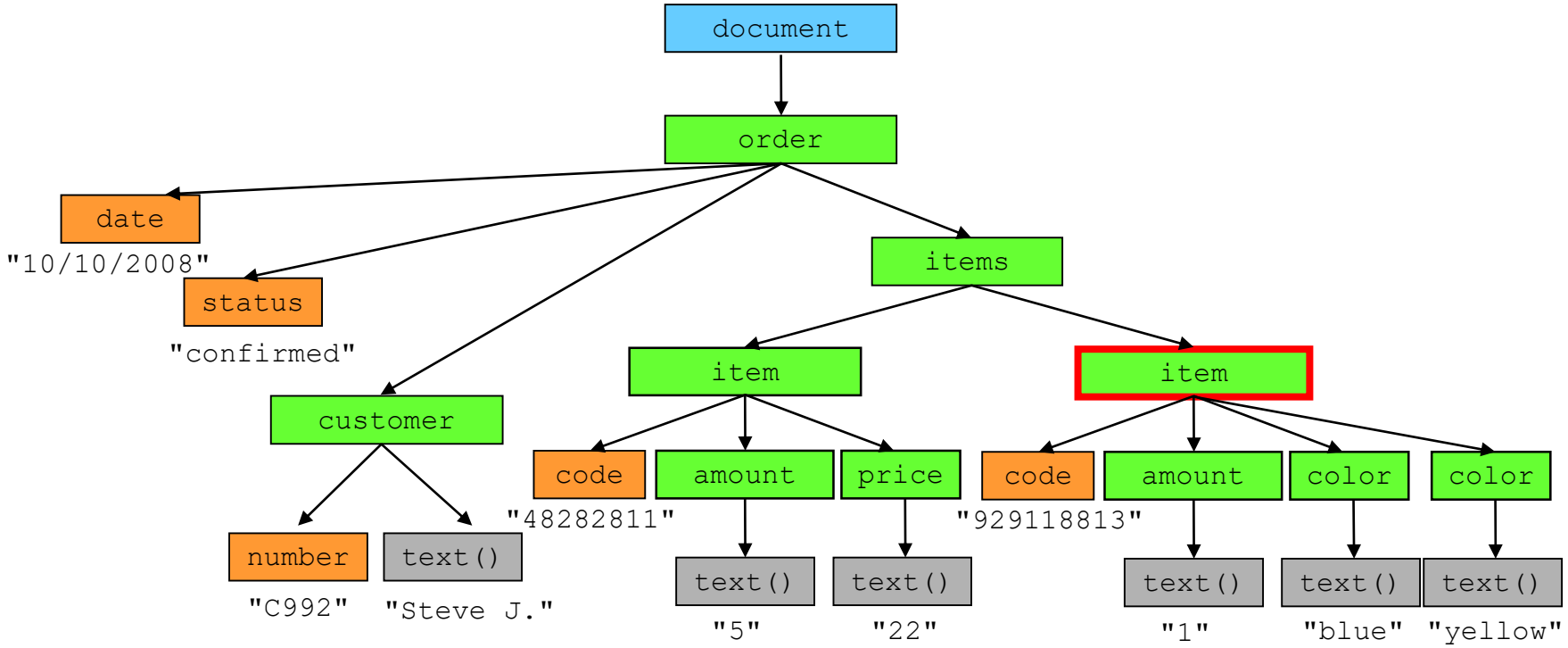
Built-in Functions

```
//items/item[position() = last()]
```



Built-in Functions

```
//items/item[last()]
```



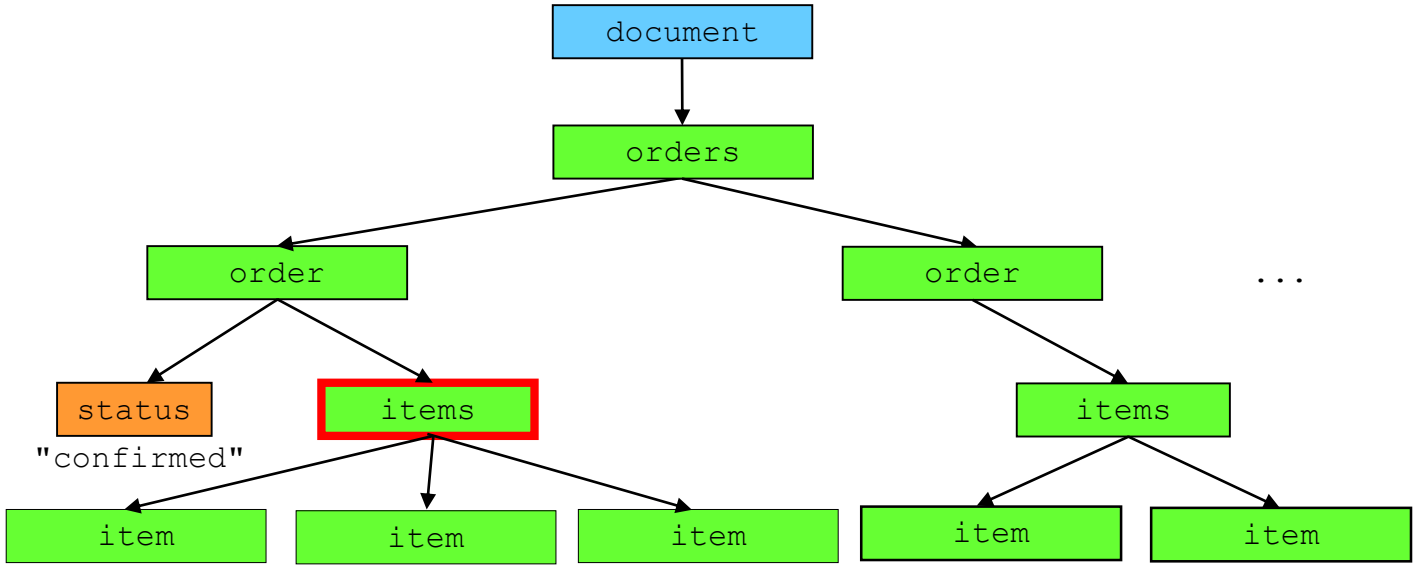
Built-in Functions

```
count(expression)
```

- Returns the number of results given by `expression`

Built-in Functions

```
//items[count(item) > 2]
```



Built-in Functions

- `+, -, *, div, mod`
- `name(), id()`
- `concat(), starts-with(), contains(), substring-after(), substring-before(), substring(), ...`
- `sum(), floor(), ceiling(), ...`