Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

CÔNG NGHỆ WEB TIÊN TIẾN

Bài 04: Extensible Markup Language

Nội dung

- 1. Giới thiệu XML
- 2. Cú pháp của XML
- 3. Không gian tên của XML
- 4. Ứng dụng của XML
- 5. Document Object Model (DOM)
- 6. XML và CSS
- 7. Extensible Stylesheet Language (XSL)

1. Giới thiệu XML

- XML viết tắt của từ e*X*tensible *M*arkup *L*anguage
- Là tập các quy ước chuẩn về cách phân chia một tài liệu ra thành nhiều phần, đánh dấu từng phần theo đặc trưng của nó rồi ghép lại
- So sánh với HTML
  - XML: là ngôn ngữ dùng để định nghĩa dữ liệu, người dùng có thể tự định nghĩa các thẻ
  - HTML: là ngôn ngữ dùng để hiển thị nội dung dữ liệu thông qua các thẻ chuẩn

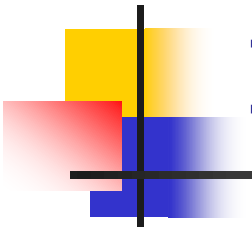
1. Giới thiệu XML

- Lịch sử ra đời
  - 1990, Tim Berners Lee đã đưa ra HTML
  - 1995, nhóm XML mong muốn
    - Không giới hạn ngôn ngữ
    - Đơn giản cho lập trình viên
    - Dễ dàng cho các Search Enginge
  - 11/1996, đặc tả đầu tiên của XML được phát hành
  - 2/1998, W3C phê chuẩn công nhận version 1.0 của XML

1. Giới thiệu XML

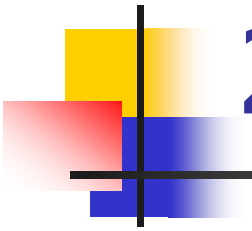
- Ví dụ về trang XML

```
<?xml version="1.0" standalone="no"
encoding="UTF-8"?>
<LexicalEntry>
     <HeadWord>an ninh</HeadWord>
     <Category>N</Category>
     <Subcategory>Na</Subcategory>
     <Definition>tình hình trật tự xã hội
 bình thường yên ổn, không có rối loạn
     </Definition>
</LexicalEntry>
```



1. Giới thiệu XML

- Các chức năng chính
  - Dễ dàng trao đổi dữ liệu
  - Tùy biến ngôn ngữ đa dạng
  - Dữ liệu mô tả
  - Dữ liệu có cấu trúc tích hợp



2. Cú pháp XML

- 2.1. Quy tắc chung trong XML
- 2.2. Tạo tài liệu XML hợp khuôn dạng

2.1. Quy tắc chung trong XML

- XML: tự do định nghĩa thẻ
- Xây dựng cách hiển thị chúng bởi CSS hoặc XSL
- Thẻ XML
  - Một phần tử XML bao gồm cặp thẻ mở đầu và kết thúc, bên trong là dữ liệu

    ```
    <HeadWord>an ninh</HeadWord>
    ```

  - Phần dữ liệu muốn thêm các kí tự đặc biệt (dấu >, <…) thì thực hiện giống như trong HTML

2.1. Quy tắc chung trong XML

- Thẻ XML
  - Nếu chỉ có một thẻ thì thêm dấu / phía cuối: <HeadWord/>
  - Trong mỗi thẻ có thể định nghĩa thêm các thuộc tính
- Tên thẻ
  - XML phân biệt chữ hoa và chữ thường
  - Bao gồm: chữ cái, chữ số, dấu "_", dấu "-", dấu ":" nhưng không được bắt đầu là chữ số, dấu "-"

2.1. Quy tắc chung trong XML

- Tạo khai báo XML
  - W3C khuyến cáo: mỗi tài liệu XML nên có duy nhất một khai báo

```
<?xml version="1.0" standalone="no"
encoding="UTF-8"?>
```

  - version="1.0": cho biết số hiệu phiên bản XML đang sử dụng
  - standalone="no": cho biết tài liệu này không có liên quan đến tài liệu khác
  - encoding="UTF-8": cho biết kiểu mã hóa ngôn ngữ

2.1. Quy tắc chung trong XML

- Tạo khai báo XML
  - Kết hợp dữ liệu của XML với CSS

```
<?xml-stylesheet type="text/css"
href="ten_file_css">
```

  - Tạo dòng ghi chú: tương tự HTML <!-- -->
  - Tạo thẻ gốc: mỗi file XML phải có một thẻ gốc
  - Tạo thuộc tính:
    - Dạng *ten_thuoc_tinh="gia_tri"*
    - Giá trị được bao trong nháy kép "" hoặc nháy đơn ''

2.1. Quy tắc chung trong XML

- Tạo khai báo XML
  - Thuộc tính xml:lang

    ```
    <text xml:lang="en">Hello</text>
    ```

  - Search engine sẽ sử dụng để nhận dạng ngôn ngữ sử dụng.
  - Tên quốc gia theo chuẩn ISO 693 gồm 2 kí tự tắt

2.2. Tạo tài liệu XML hợp khuôn dạng

- Các trình dịch XML thường yêu cầu rất nghiêm ngặt về kiểm tra cú pháp
- Một tài liệu XML được coi là đúng cú pháp khi nào hợp khuôn dạng
- Khuôn dạng: các quy tắc khai báo, thẻ, thuộc tính, đặt tên thẻ…

2.2. Tạo tài liệu XML hợp khuôn dạng

- Quy tắc cơ bản
  - Các khai báo XML cần đặt ở dòng đầu tiên
  - Mỗi tài liệu chỉ có một phần tử gốc
  - Thẻ đầy đủ bao gồm thẻ đóng và thẻ mở hoặc thẻ với dấu / ở cuối
  - Các thẻ lồng nhau phải có thẻ đóng, thẻ mở hợp vị trí
  - Tên thuộc tính trong một thẻ là duy nhất
  - Giá trị (kể cả số) nằm trong cặp dấu nháy đơn hoặc nháy kép

3. Không gian tên của XML

- XML cho phép tự định nghĩa và đặt tên các thẻ
- standalone="yes": cho biết tài liệu này có liên quan đến tài liệu khác

3. Không gian tên của XML

- Cần có sự phân biệt về phạm vi hay không gian tên mà mỗi thẻ có ý nghĩa
- Namespace: cho phép tạo và sử dụng các thẻ cùng tên độc lập mà không gây ra lỗi
- Sử dụng: thêm một *prefix* ở trước tên của thẻ và tên của thuộc tính

3. Không gian tên của XML

- Ví dụ

```
<LexicalEntry>
      <HeadWord>an ninh</HeadWord>
      <Category>N</Category>
      <Subcategory>Na</Subcategory>
      <Definition>tình hình trật tự xã hội
 bình thường yên ổn, không có rối loạn
      </Definition>
</LexicalEntry>
```

3. Không gian tên của XML

- Ví dụ: định nghĩa namespace là *dic*

```
<dic:LexicalEntry xmlns:dic="URL1">
      <dic:HeadWord>an ninh</dic:HeadWord>
      <dic:Category>N</dic:Category>
      <dic:Subcategory>Na</dic:Subcategory>
      <dic:Definition>tình hình trật tự xã
Hội bình thường yên ổn, không có rối loạn
      </dic:Definition>
</dic:LexicalEntry>
```

3. Không gian tên của XML

- Có thể bổ sung thêm các thẻ mới cùng tên

```
<dic:LexicalEntry
      xmlns:dic="URL1" xmlns:nmspc="URL2">
      <dic:HeadWord>an ninh</dic:HeadWord>
      <dic:Category>N</dic:Category>
      <dic:Subcategory>Na</dic:Subcategory>
      <dic:Definition>tình hình trật tự xã
Hội bình thường yên ổn, không có rối loạn
      </dic:Definition>
      <nmspc:HeadWord>composite word
      </nmspc:HeadWord>
</dic:LexicalEntry>
```

3. Không gian tên của XML

- Có thể đưa thêm thuộc tính vào thẻ trong không gian tên mới bằng cách thêm *"prefix:"* phía trước
- Thuộc tính *xmlns:prefix* có thể được đặt ở bất kỳ thẻ nào chứ không chỉ giới hạn ở phần tử gốc

=> Có thể đặt định nghĩa không gian tên đến nơi nó bắt đầu được sử dụng

3. Không gian tên của XML

```
<dic:LexicalEntry xmlns:dic="URL1">
     <dic:HeadWord>an ninh</dic:HeadWord>
     <dic:Category>N</dic:Category>
     <dic:Subcategory>Na</dic:Subcategory>
     <dic:Definition>tình hình trật tự xã
Hội bình thường yên ổn, không có rối loạn
     </dic:Definition>
     <nmspc:HeadWord xmlns:nmspc="URL2"
     nmspc:date="1/9/2009">composite word
     </nmspc:HeadWord>
</dic:LexicalEntry>
```

4. Ứng dụng của XML

- XML có thể tạo ra các ngôn ngữ con khác.
- Các ứng dụng thực tiễn của XML là một tập các thẻ XML hoạt động trong một lĩnh vực nào đó như toán học, tin học, kinh doanh…
- Mỗi lĩnh vực dựa trên đặc tả XML

4.1. MathML - Ngôn ngữ định dạng toán học

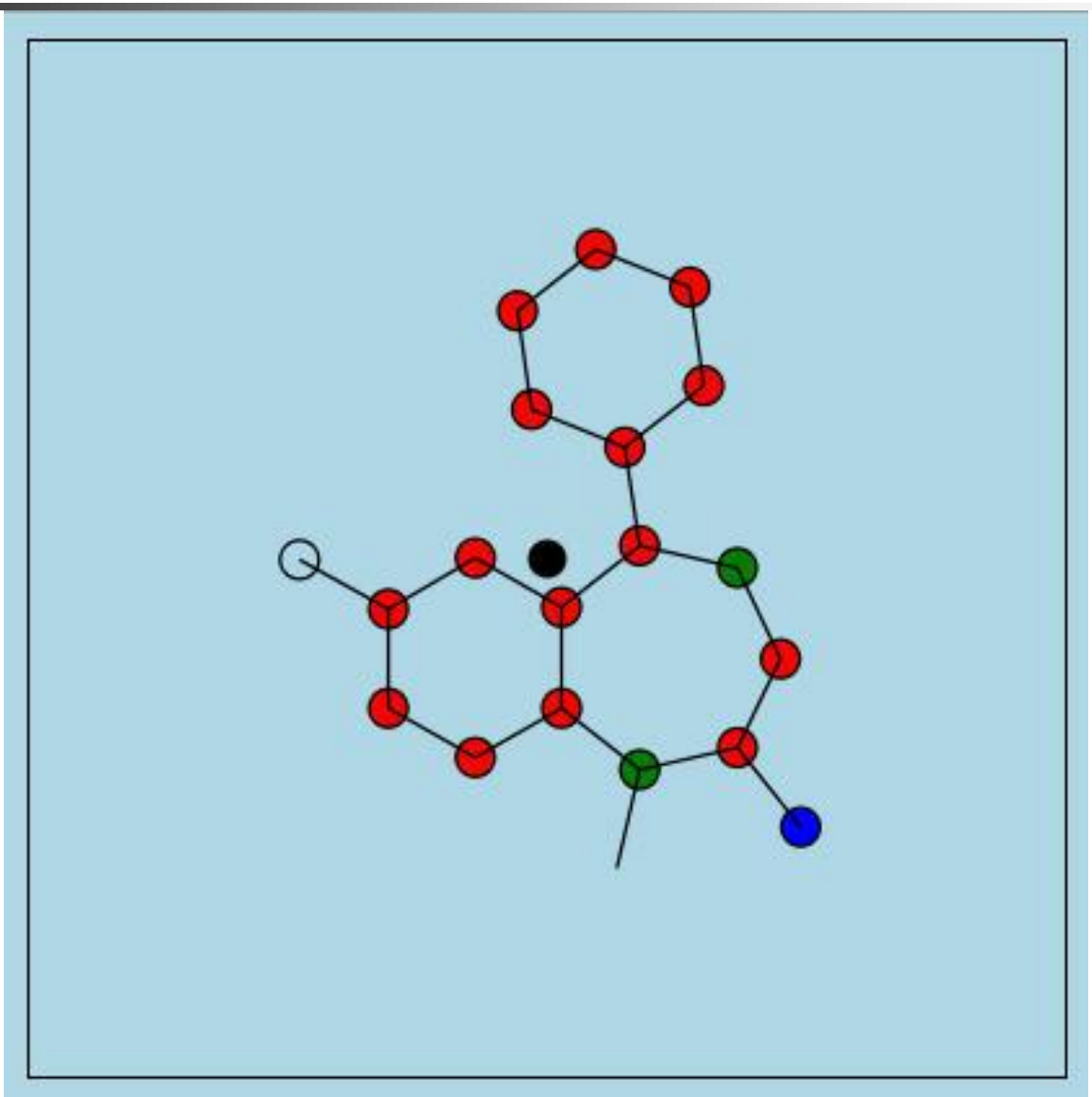
- Mục đích: hiển thị và định dạng các biểu thức toán học trên Web
- Giúp trình duyệt không chỉ hiển thị mà còn hiểu các biểu thức toán học
- Version 2.0 ra đời 21/3/2001
- Ví dụ công thức

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

4.2. CML – Ngôn ngữ định dạng hóa học

- CML mô tả và hiển thị nội dung của cấu trúc một phần tử hóa học
- Cung cấp danh sách công thức hóa học của các phần tử đã định nghĩa sẵn

4.2. CML - Ngôn ngữ định dạng hóa học



KML (Keyhole Markup Language)

- Geographic data in an Earth browser: Google Earth, Google Maps,
- Example: sample.kml

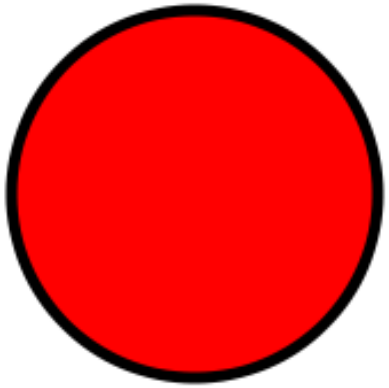
```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>HUT placemark</name>
    <description>Location of HUT</description>
    <Point>
      <coordinates>105.84413,21.00438,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

SVG (Scalable Vector Graphics)

- 2D vector graphics applications and images
- Firefox > 3.6
  - http://commons.wikimedia.org/wiki/SVG_examples
  - http://www.carto.net/papers/svg/samples/

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg"
  width="200" height="200">
  <circle cx="100" cy="100" r="50" stroke="black"
    stroke-width="5" fill="red" />
</svg>
```



Bài tập:

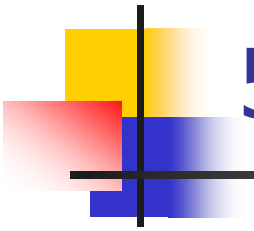
- Kiểm tra tính hợp khuôn dạng của các tài liệu XML sau và sửa lại nếu có sai sót:
- **Bài 1:**

```
<?xml version=1.0>
<DatHang>
        <NgayDat>25-12-2005<NgayDat>
        <KhachHang>Ly Thien My</khachhang>
        <SanPham>
                <MaSo>5
                <SoLuong>7</SoLuong>
        </sanpham>
</Dat+Hang>
```

Bài tập:

- Kiểm tra tính hợp khuôn dạng của các tài liệu XML sau và sửa lại nếu có sai sót:
- **Bài 2:**

```
<?xml version="1.0"?>
<KhachHang>
        <Ho>Nguyen</ho>
        <tenlot>Van <ten>Anh </tenlot>
        <Ten>
        <SoDienThoai vitri=Nha>9845345</SoDienThoai>
        <SoDienThoai vitri=CoQuan>994545345</sodienthoai>
</KhachHang>
```



5. Document Object Model (DOM)

- DOM là một API đối với HTML và XML, định nghĩa cấu trúc logic của tài liệu và các xử lý của chúng
- DOM được sử dụng để xử lý dữ liệu lưu trong XML
- Tài liệu XML sẽ là một cây bao gồm tập hợp các nút chứa: phần tử, dữ liệu, thuộc tính…

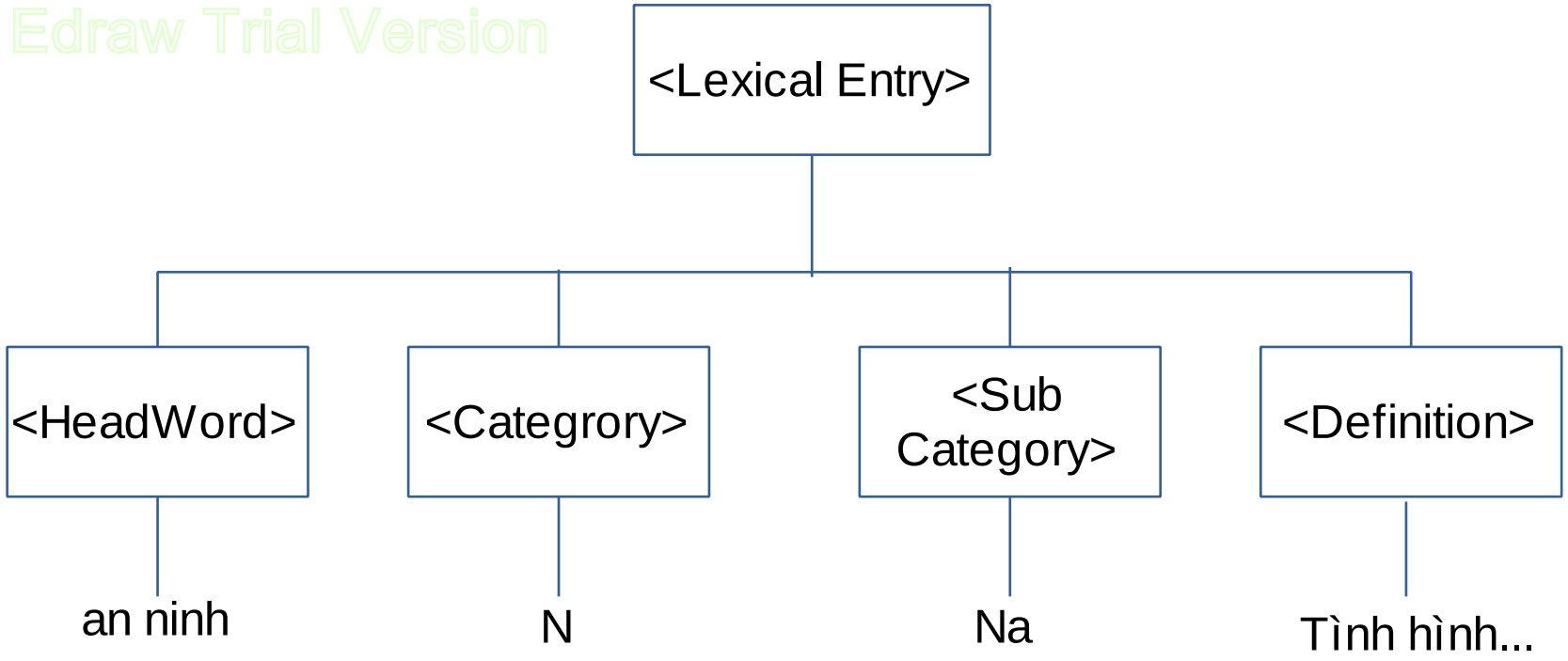
5. Document Object Model (DOM)

Nút	Mô tả
Element	Phần tử XML
Attribute	Thuộc tính
Text	Dữ liệu
Entity	Thực thể
Processing Instruction	Chỉ thị xử lý
Comment	Chú thích
Document	Tài liệu
Document Type	Kiểu tài liệu
Document Fragment	Đoạn tài liệu
Notation	Ghi chú

Các loại nút trong DOM

5. Document Object Model (DOM)

- Ví dụ

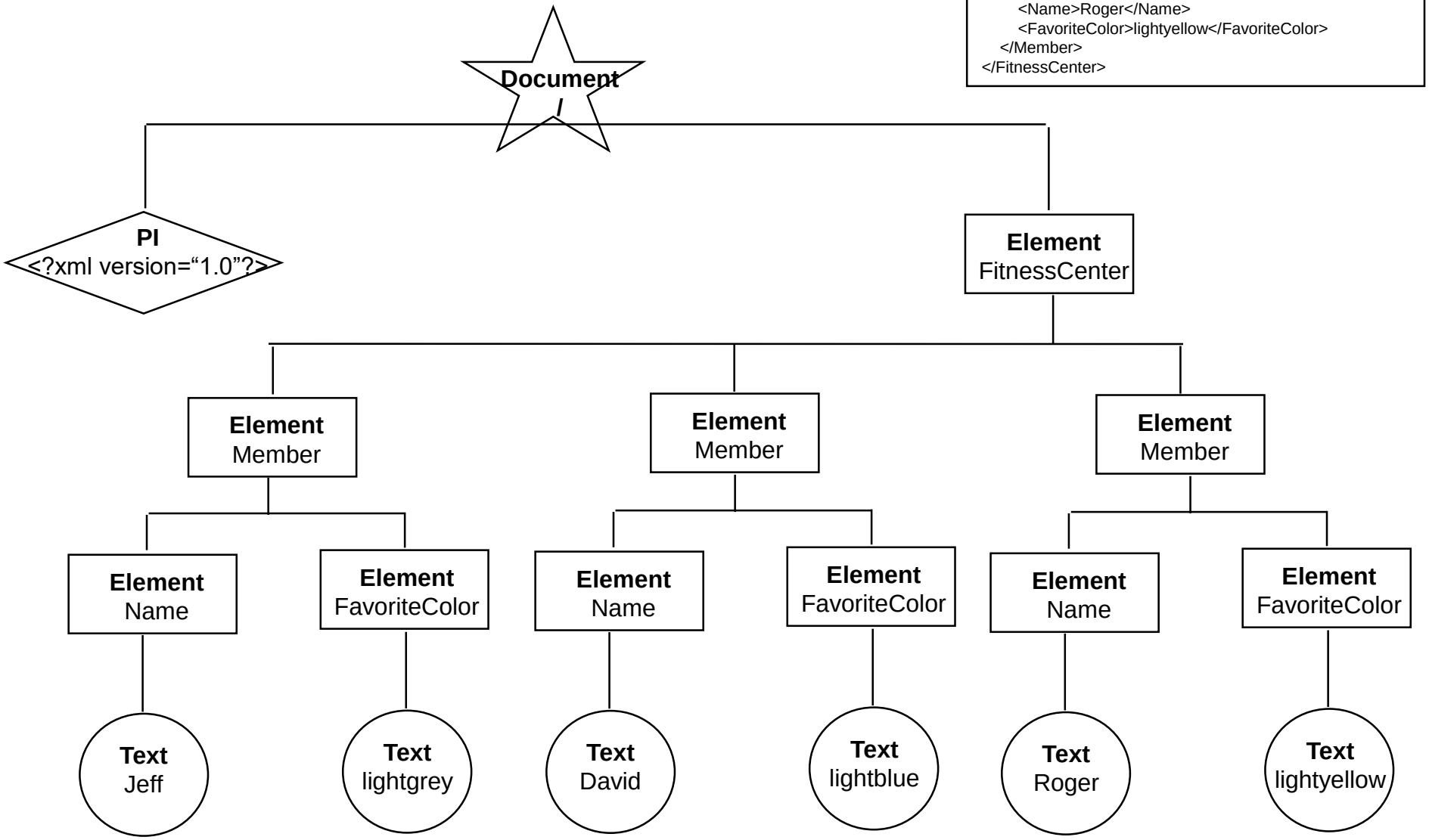


CẤU TRÚC CÂY CỦA VĂN BẢN XML

```
<?xml version="1.0"?>
<FitnessCenter>
    <Member>
        <Name>Jeff</Name>
        <FavoriteColor>lightgrey</FavoriteColor>
    </Member>
    <Member>
        <Name>David</Name>
        <FavoriteColor>lightblue</FavoriteColor>
    </Member>
    <Member>
        <Name>Roger</Name>
        <FavoriteColor>lightyellow</FavoriteColor>
    </Member>
</FitnessCenter>
```

CẤU TRÚC CÂY CỦA VĂN BẢN XML

```
<?xml version="1.0"?>
<FitnessCenter>
   <Member>
      <Name>Jeff</Name>
      <FavoriteColor>lightgrey</FavoriteColor>
   </Member>
   <Member>
      <Name>David</Name>
      <FavoriteColor>lightblue</FavoriteColor>
   </Member>
   <Member>
      <Name>Roger</Name>
      <FavoriteColor>lightyellow</FavoriteColor>
   </Member>
</FitnessCenter>
```



5. Document Object Model (DOM)

- Nếu các thẻ lồng nhau => mô hình phân cấp: nút cha, nút con, nút cháu…, dữ liệu
- W3C định nghĩa nhiều mức độ cho DOM
  - Level 0: đặc tả DOM khởi đầu, áp dụng trước đây trong NN 3.0, IE 3.0
  - Level 1: đặc tả DOM sử dụng phổ biến hiện nay
  - Level 2: kết hợp DOM với CSS hay XSL
  - Level 3: mức độ hoạch định

5. Document Object Model (DOM)

- Các đối tượng cơ bản của DOM

Đối tượng	Mô tả
Node	Một nút đơn trong kiến trúc
NodeList	Một tập hợp nút
NamedNodeMap	Một tập các nút, cho phép truy xuất theo tên cũng như chỉ số

5. Document Object Model (DOM)

- Một số phương thức trong DOM

Phương thức	Mô tả
childNodes	Trả về NodeList chứa các nút con
firstChild	Trả về nút con đầu tiên
lastChild	Trả về nút con cuối cùng
parentNode	Trả về nút cha
previousSibling	Trả về nút cùng mức đứng trước
nextSibling	Trả về nút cùng mức kế tiếp
nodeName	Trả về tên của nút
nodeValue	Trả về giá trị của nút

5. Document Object Model (DOM)

- Ví dụ
  - nodRoot trỏ tới nút gốc LexicalEntry
  - nodRoot.childNodes(0) => HeadWord
  - nodRoot.childNodes(1) => Category
  - nodRoot.childNodes(0).nextSibling() => Category
  - nodRoot.childNodes(0).parentNode() => LexicalEntry

6. XML và CSS

- Sử dụng CSS để định nghĩa cách hiển thị các thẻ XML
- Thực hiện
  - Tạo ra tài liệu XML
  - Tạo ra file CSS (kiểu External Style) có chứa định nghĩa các style của các thẻ trong tài liệu XML
  - Chèn đoạn mã khai báo việc sử dụng CSS

```
<?xml-stylesheet type="text/css"
href="ten_file_css"?>
```

Bài tập 0:

- Tạo một cấu trúc tài liệu XML được mô tả như sau:
- Thẻ gốc là Paragraph . Trong thẻ gốc chứa 3 thẻ con lần lượt là
  - Segment1 : chứa dữ liệu văn bản tùy ý .
  - Segment2: chứa 2 thẻ con là :
    - Red : chứa văn bản tùy ý
    - Green: chứa văn bản tùy ý
  - Segment3: chứa 3 thẻ con như sau:
    - BoldAndBlue : chứa văn bản tùy ý
    - UnderlineAndRed:chứa văn bản tùy ý
    - ItalicAndTahoma: chứa văn bản tùy ý

Bài giải

- Pagragraph.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="pagragraph.css" ?>
<Paragraph>
    <Segment1></Segment1>
    <Segment2>
        <Red>Van ban tuy y</Red>
        <Green>Van ban tuy y</Green>
    </Segment2>
    <Segment3>
        <BoldAndBlue>Van ban tuy y</BoldAndBlue>
        <UnderlineAndRed>Van ban tuy y</UnderlineAndRed>
        <ItalicAndTahoma>Van ban tuy y</ItalicAndTahoma>
    </Segment3>
</Paragraph>
```

Bài tập 2:

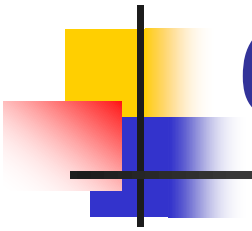
- Tạo File .css để định kiểu cho tài liệu XML trên với nội dung được mô tả như sau:
  - Tất cả các phần tử ở chế độ hiển thị
  - Phần tử Segment1 thể hiện văn bản ở giữa trang.
  - Phần tử Red thể hiện ở chế độ màu đỏ , font chữ tahoma và kích thước là 10pt
  - Phần tử Green thể hiện màu xanh, font chữ là Arial và kích thước là 15pt
  - BoldAndBlue : thể thiện chữ in đậm và màu xanh
  - UnderlineAndRed : thể hiện văn bản ở chế độ gạch dưới và chữ màu đỏ
  - ItalicAndTahoma: thể hiện chữ in nghiêng và font chữ tahoma

Bài giải

- Pagragraph.css

```
Paragraph {   display:on   }
Segment1 {   text-align:center   }
Red {  color:red;
        font-family:tahoma;
        font-size:10pt   }
Green {color:green;
        font-family:arial;
        font-size:15pt   }
BoldAndBlue {        font-weight:bold;
                     color:blue   }
UnderlineAndRed {   text-decoration:underline; color:red   }

ItalicAndTahoma {   font-style:italic;              font-
   family:tahoma   }
```



6. XML và DTD

Tại sao lại cần các DTD?

- Mọi thay đổi đối với cấu trúc dữ liệu XML phải được thực hiện trong ba nơi: định chuẩn, các ứng dụng nhận và gửi. Đây có thể là cách tiếp cận hiệu quả, năng suất cao, để điều quản dữ liệu XML trong một số tình huống hạn chế
- Tuy nhiên, trong các trường hợp ở đó dữ liệu XML được chia sẻ rộng rãi hơn việc duy trì các chức năng này trong mỗi và mọi ứng dụng đã trở thành cơn ác mộng bùng nổ
- Việc tách biệt phần mô tả dữ liệu XML với các ứng dụng riêng lẻ cho phép tất cả các ứng dụng hợp tác chia sẻ một phần mô tả dữ liệu đơn lẻ, có tên là **từ vựng XML**
- Một nhóm tư liệu XML chia sẻ một từ vựng XML chung được gọi là một **kiểu tư liệu**[document type], và mỗi tư liệu riêng lẻ tuân thủ một kiểu tư liệu được gọi là một **minh dụ tư liệu**[document instance]

Trình phân ngữ (Parsers)

- Trình phân ngữ XML (XML parser) là bộ xử lý XML để bảo đảm một tư liệu XML tề chỉnh:
  - Một bộ xử lý phổ biến hiện tại là DOM (Document Object Model) và SAX (Simple API for XML)
- Một trình phân ngữ phê chuẩn (validating parser) là bộ xử lý XML. Tiến trình phê chuẩn này được hoàn thành bằng cách so sánh nội dung của tư liệu XML với một tập mẫu kết hợp dưới dạng một DTD
  - Hầu hết các browser không dùng các validating parser

Một ví dụ về XML

```
<novel>
    <foreword>
        <paragraph>This is the great American novel.
          </paragraph>
    </foreword>
    <chapter number="1">
        <paragraph>It was a dark and stormy night.</paragraph>
        <paragraph>Suddenly, a shot rang out!</paragraph>
    </chapter>
</novel>
```

- Một tư liệu XML chứa (và DTD miêu tả):
  - Thành phần, là novel và paragraph, bao gôm thẻ và nội dung
  - Thuộc tính, chính là number = `1`, bao gồm tên và giá trị
  - Thực thể (không sử dụng trong ví dụ này)

Một ví dụ của DTD

```
<!DOCTYPE novel [
   <!ELEMENT novel (foreword, chapter+)>
   <!ELEMENT foreword (paragraph+)>
   <!ELEMENT chapter (paragraph+)>
   <!ELEMENT paragraph (#PCDATA)>
   <!ATTRIBUTE chapter number CDATA #REQUIRED>
]>
```

- Một novel bao gồm một foreword và một hay nhiều chapter, trong đó
- Một foreword bao gồm một hay nhiều paragraph
- Một chapter bap gồm một hay nhiều paragraph
- Một paragraph bao gồm parsed character data (văn bản không thể chứa bất kỳ một thành phần nào)
- Mỗi chapter phải có một thuộc tính number

Khai báo tổng quát kiểu tư liệu

- <!DOCTYPE root-name[DTD]>

Trong đó:

- root-name : là phần tử gốc

DTD là các định nghĩa cho các phần tử trong tài liệu

- Cấu trúc tổng quát của DTD

<!ELEMENT ElementName Content-model>

- ELEMENT là từ khoá và phải viết hoa

- ElementName là tên của phần tử do người sử dụng định nghĩa

- Content – model: mô hình nội dung

Tạo khai báo và định nghĩa DTD

```
<?xml version="1.0"?>
<name>
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>
```

→ Xây dựng DTD?

Tạo khai báo và định nghĩa DTD

```
<?xml version="1.0"?>
<name>
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>
```

→ Xây dựng DTD?

```
<!DOCTYPE name [
  <!ELEMENT name (first, middle, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT middle (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
]>
```

Tạo khai báo và định nghĩa DTD

Nếu ta đổi thành:

```
<?xml version="1.0"?>
<!DOCTYPE name [
  <!ELEMENT name (first, middle, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT middle (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
]>
```

Tạo khai báo và định nghĩa DTD

<name>
  <firstname>John</firstname>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>

→ Thực hiện kiểm tra tài liệu cho kết quả: "Element 'firstname' is unexpected according to content model of parent element 'name'. Expecting: first."

Nội dung chứa dữ liệu text

Cú pháp

<!ELEMENT ElementName (#PCDATA) >

Từ khoá ELEMENT và PCDATA phải viết hoa

Ví dụ

<!ELEMENT lastname (#PCDATA)>

<!ELEMENT firstname(#PCDATA)>

Khi đó chứa: <lastname>Nguyễn </lastname>

<firstname>Thanh</firstname>

Các khai báo kiểu thành phần (Element)

- Hậu tố (Suffixes):

?	tuỳ chọn	foreword?
+	Một hoặc nhiều	chapter+
*	zero hoặc nhiều	appendix*

- Toán tử (Separators):

,	tuần tự	foreword?, chapter+
|	chọn	section|chapter

- Nhóm (Grouping)

( ) grouping	(section|chapter)+

Phần tử chứa các phần tử con

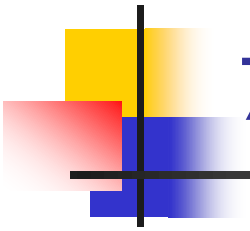
- Cú pháp:
- <!ELEMENT ElementName (child_e1 , child_e2, ...)>

Lí do sử dụng XML Schema

- XML Schema là một sự thay thế cho DTD
  - Hỗ trợ nhiều loại dữ liệu
  - Sử dụng cú pháp XML
  - Bảo toàn sự giao tiếp dữ liệu
  - Ràng buộc khóa và tham chiếu mạnh hơn DTD
  - Tích hợp với namespace

Ví dụ

```
<?xml version="1.0"?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.w3schools.com"
            elementFormDefault="qualified">
<xsd:element name="note">
   <xsd:complexType>
    <xsd:sequence>
            <xsd:element name="to" type="xsd:string"/>
            <xsd:element name="from" type="xsd:string"/>
            <xsd:element name="heading" type="xsd:string"/>
            <xsd:element name="body" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
</xsd:element>
</xsd:schema>
```



7. Extensible Stylesheet Language (XSL)

- XSL được dùng để định dạng tài liệu XML với mục đích có được kết quả như một trang HTML
- 2 loại
  - XSLT (XSL Transform): chuyên về chuyển dịch, trích rút dữ liệu XML đưa vào khuôn dạng HTML
  - XSL-FO (XSL Formatted Objects): chuyên về định dạng: font chữ, màu sắc…

Giới thiệu ngôn ngữ XPATH

- XPATH

  - Là các biểu thức dùng để truy xuất đến một hoặc nhiều thẻ trong tài liệu XML

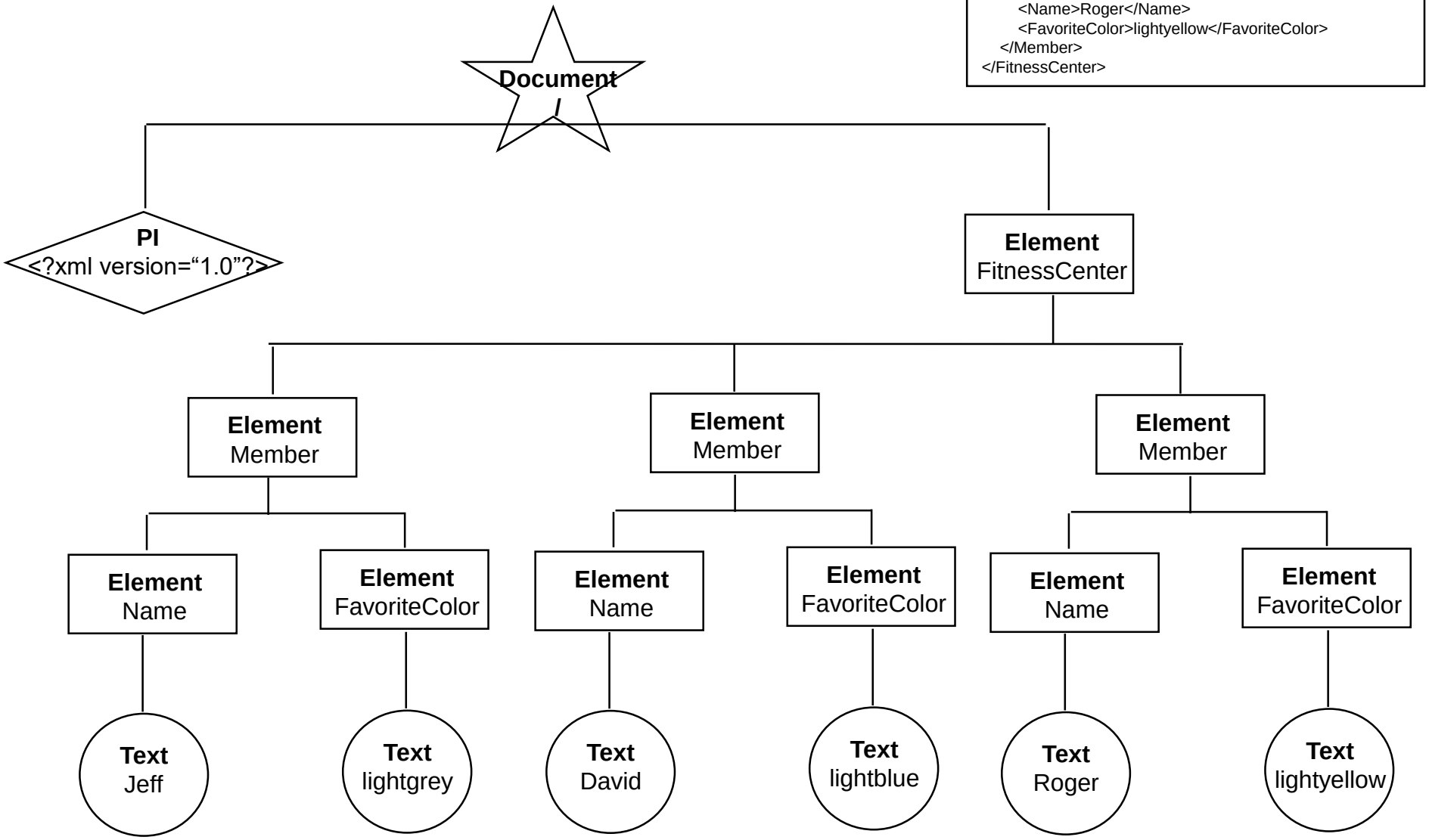
  - Được sử dụng trong XSLT để chuyển đổi cấu trúc nội dung tài liệu XML

CẤU TRÚC CÂY CỦA VĂN BẢN XML

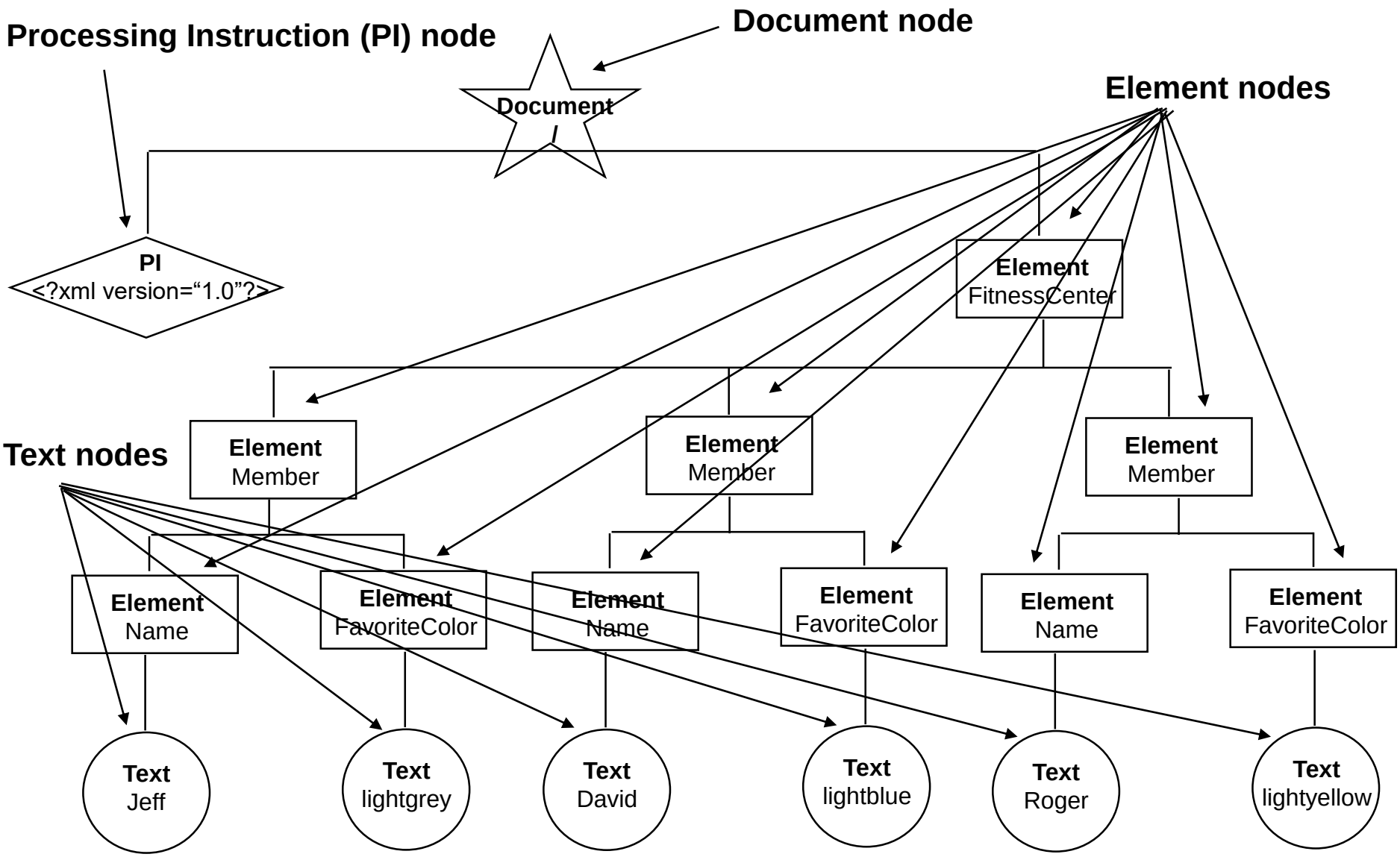
```
<?xml version="1.0"?>
<FitnessCenter>
    <Member>
        <Name>Jeff</Name>
        <FavoriteColor>lightgrey</FavoriteColor>
    </Member>
    <Member>
        <Name>David</Name>
        <FavoriteColor>lightblue</FavoriteColor>
    </Member>
    <Member>
        <Name>Roger</Name>
        <FavoriteColor>lightyellow</FavoriteColor>
    </Member>
</FitnessCenter>
```

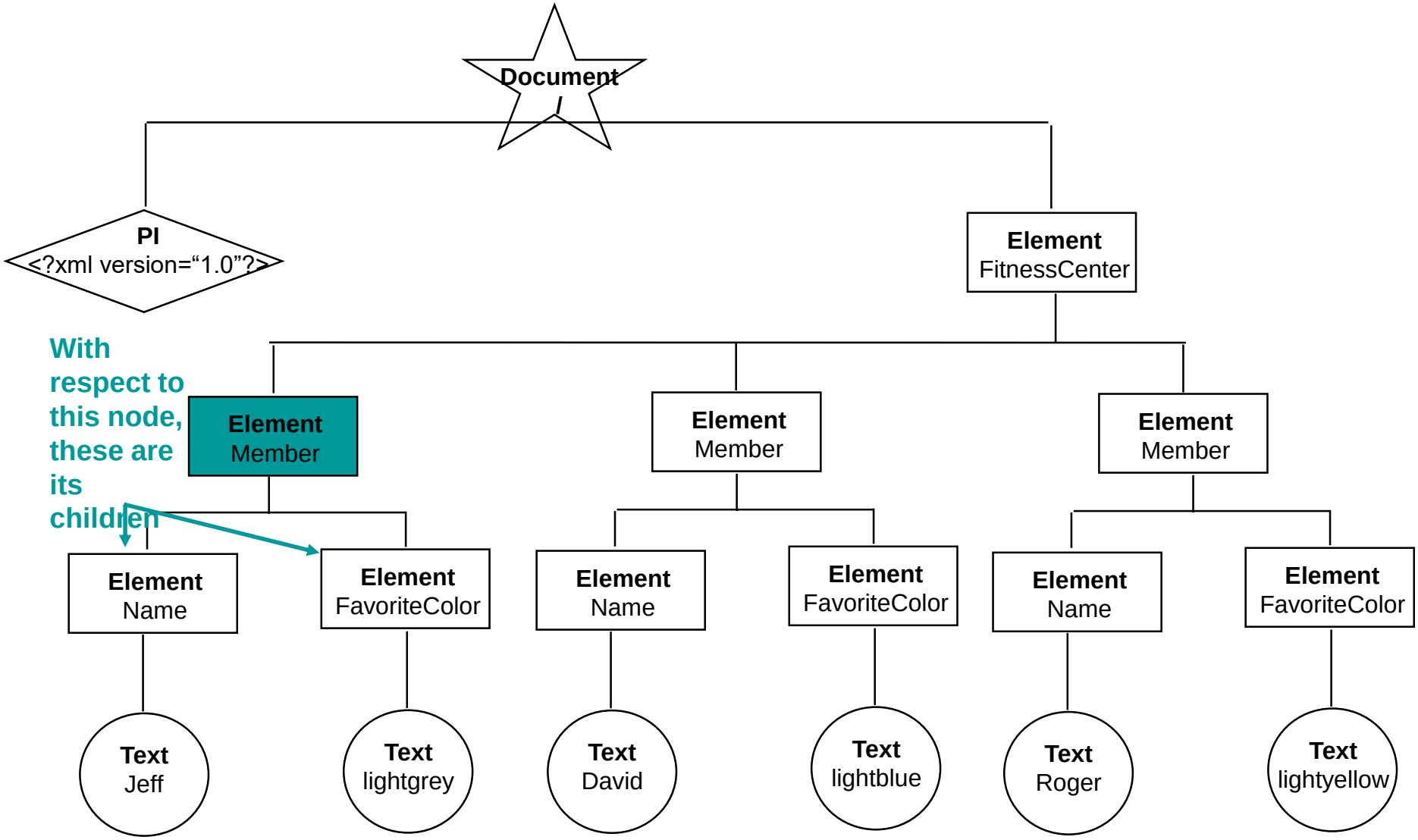
CẤU TRÚC CÂY CỦA VĂN BẢN XML

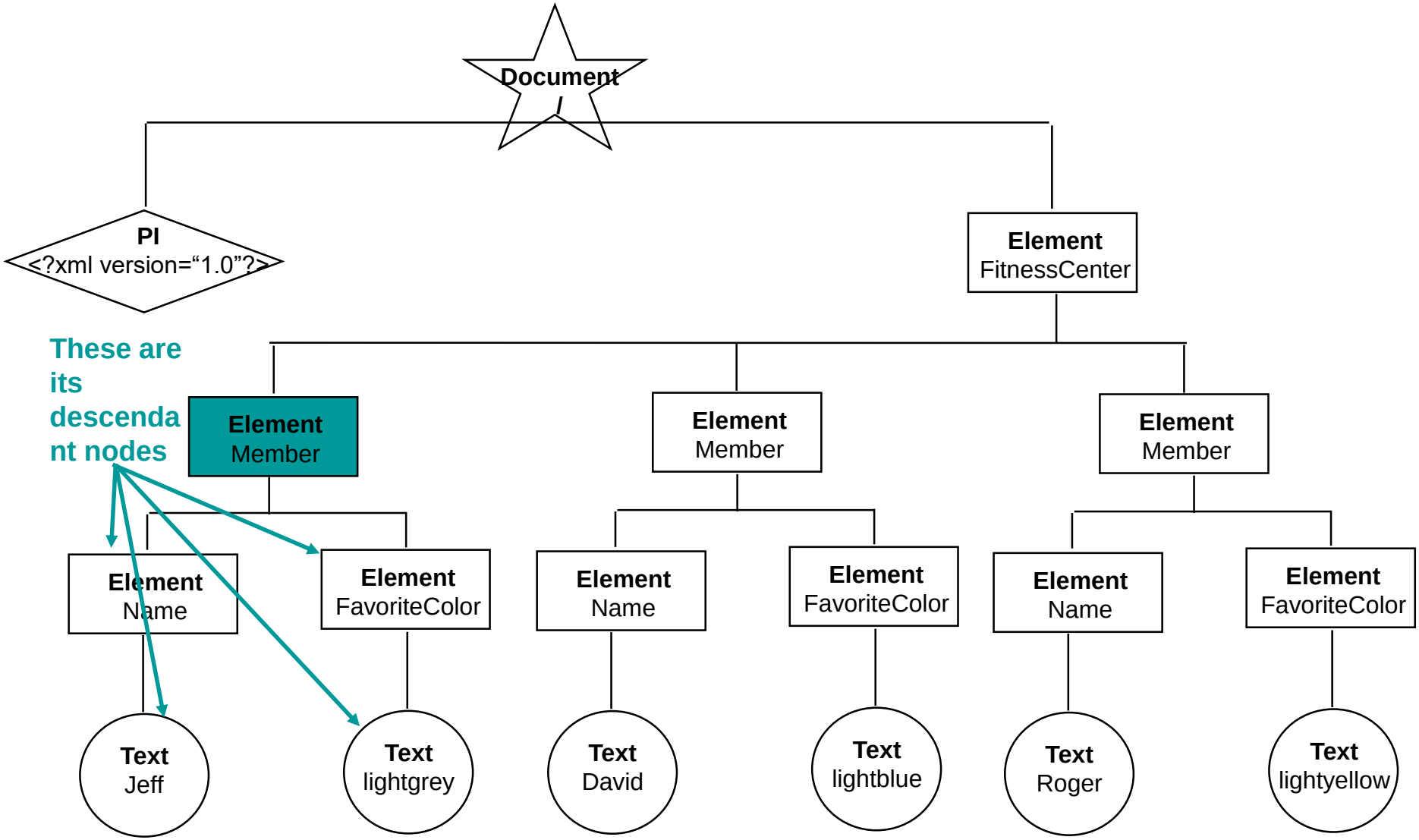
```
<?xml version="1.0"?>
<FitnessCenter>
   <Member>
      <Name>Jeff</Name>
      <FavoriteColor>lightgrey</FavoriteColor>
   </Member>
   <Member>
      <Name>David</Name>
      <FavoriteColor>lightblue</FavoriteColor>
   </Member>
   <Member>
      <Name>Roger</Name>
      <FavoriteColor>lightyellow</FavoriteColor>
   </Member>
</FitnessCenter>
```

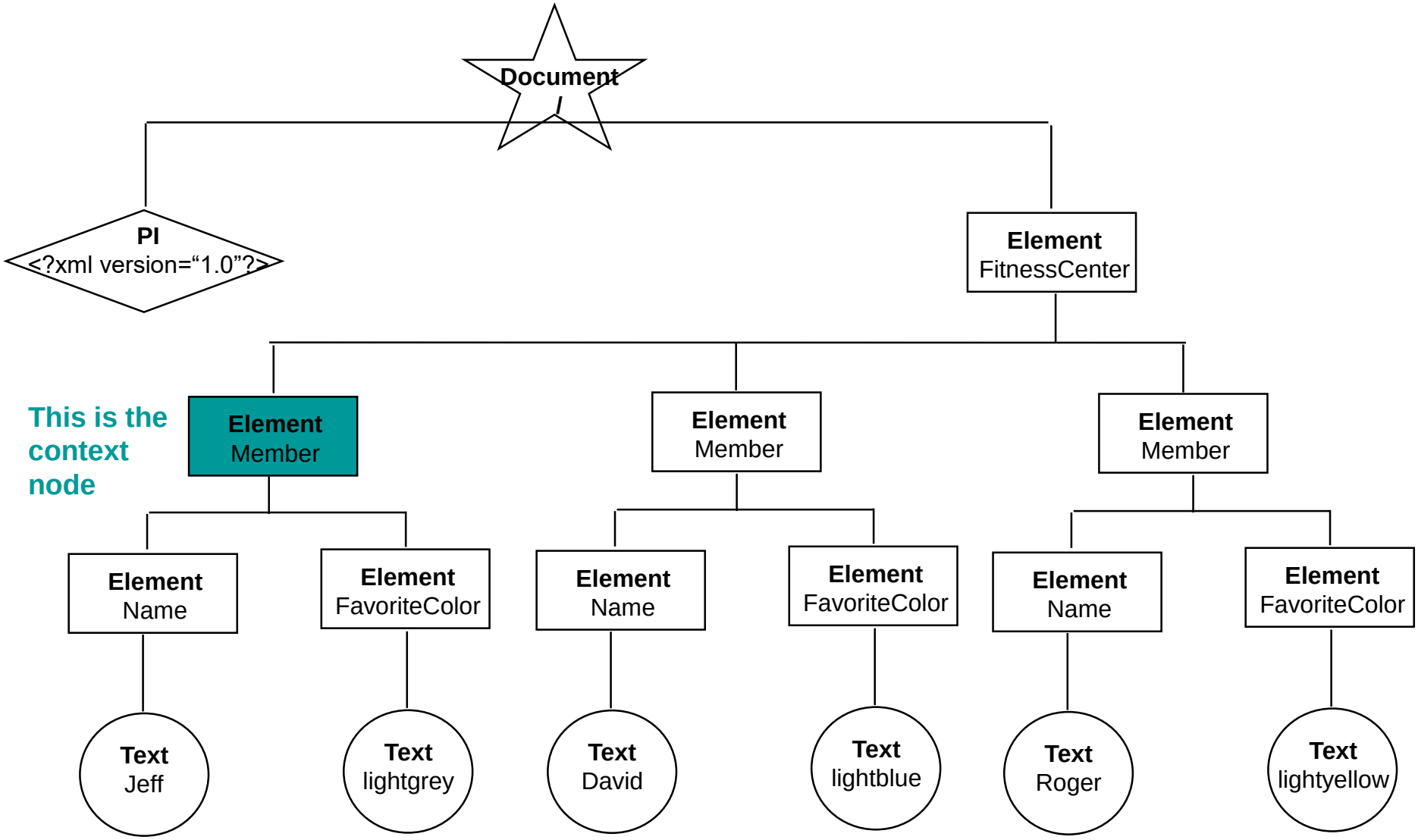


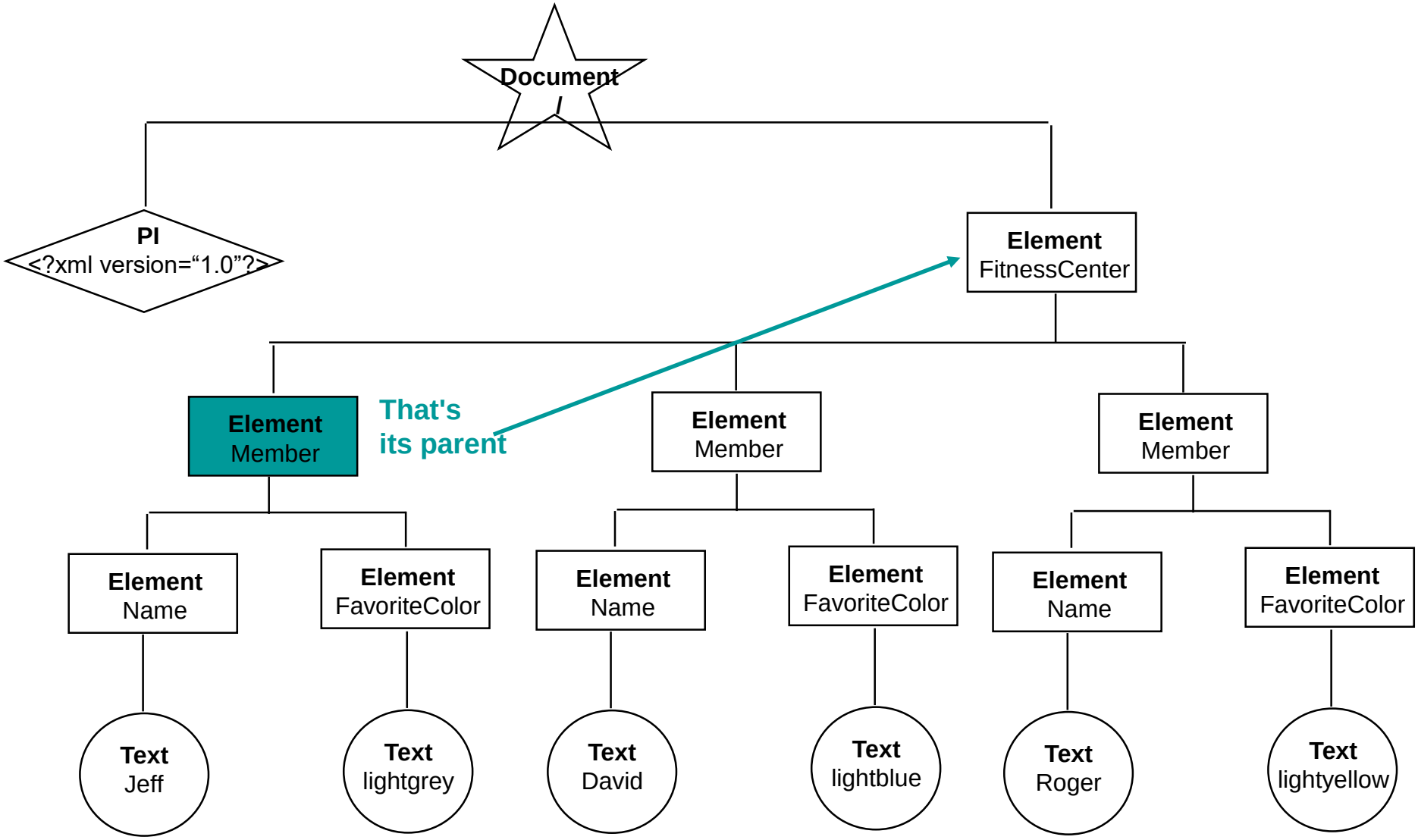
Node

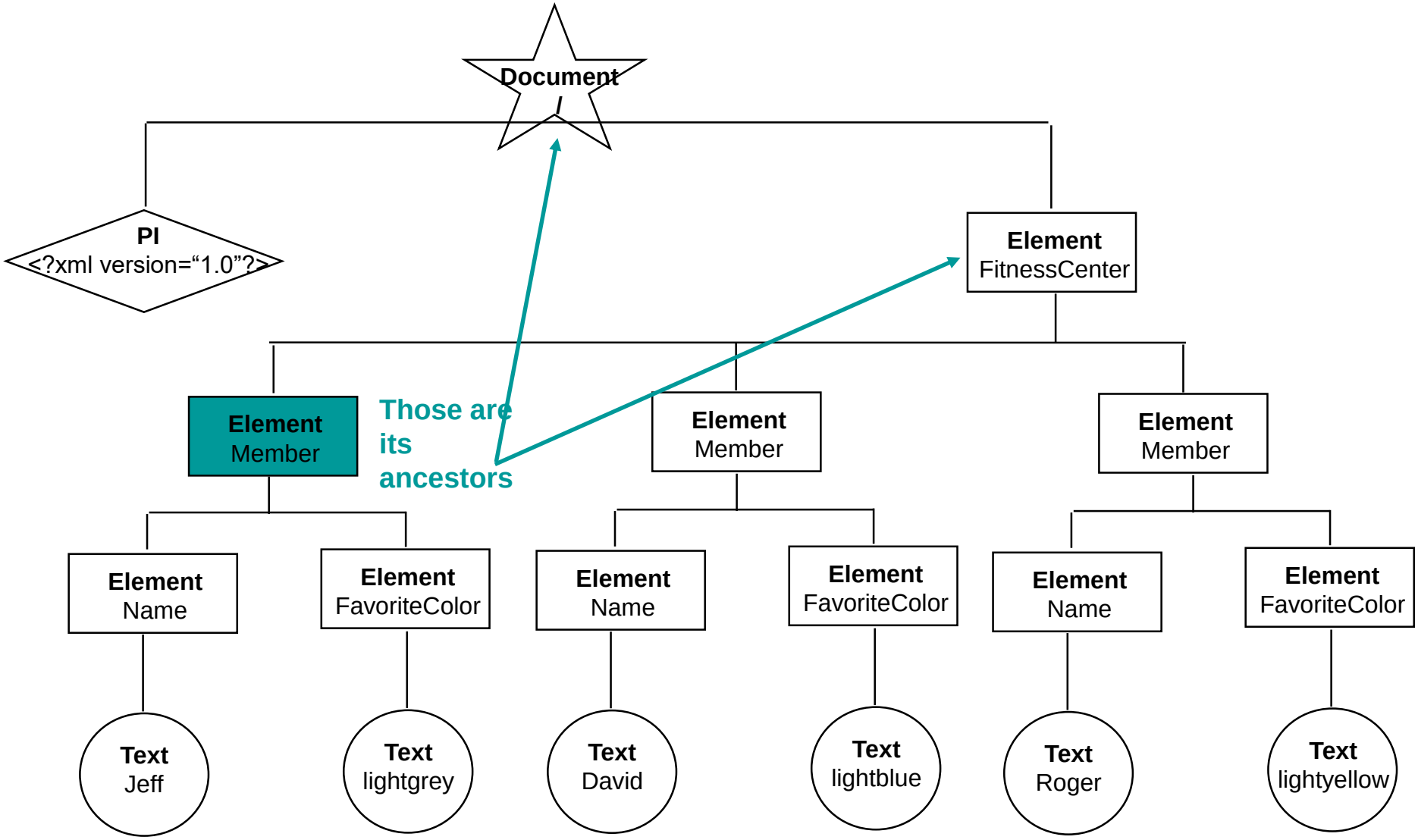

Processing Instruction (PI) node
Document node
Element nodes
Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Text nodes
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

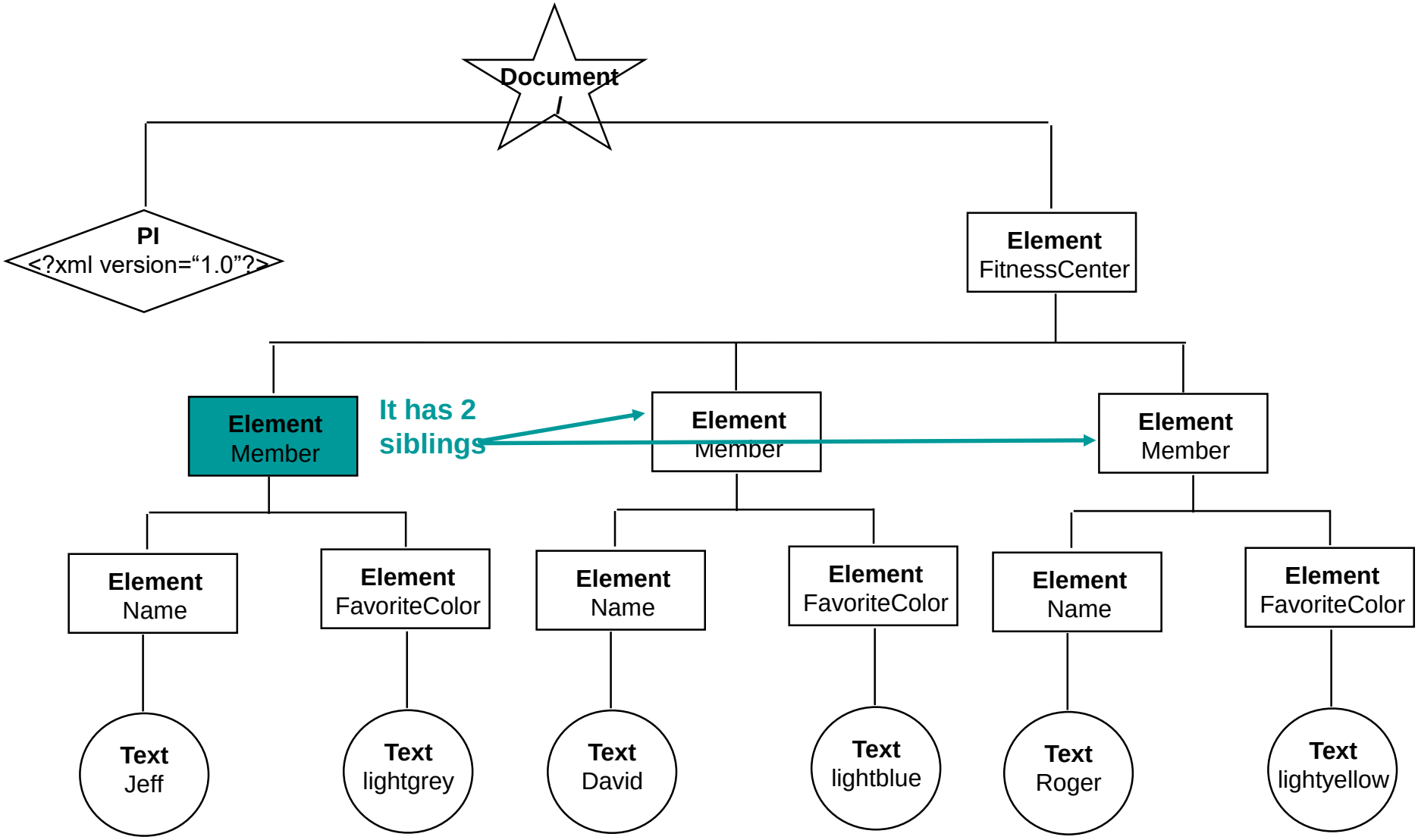

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
With respect to this node, these are its children
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

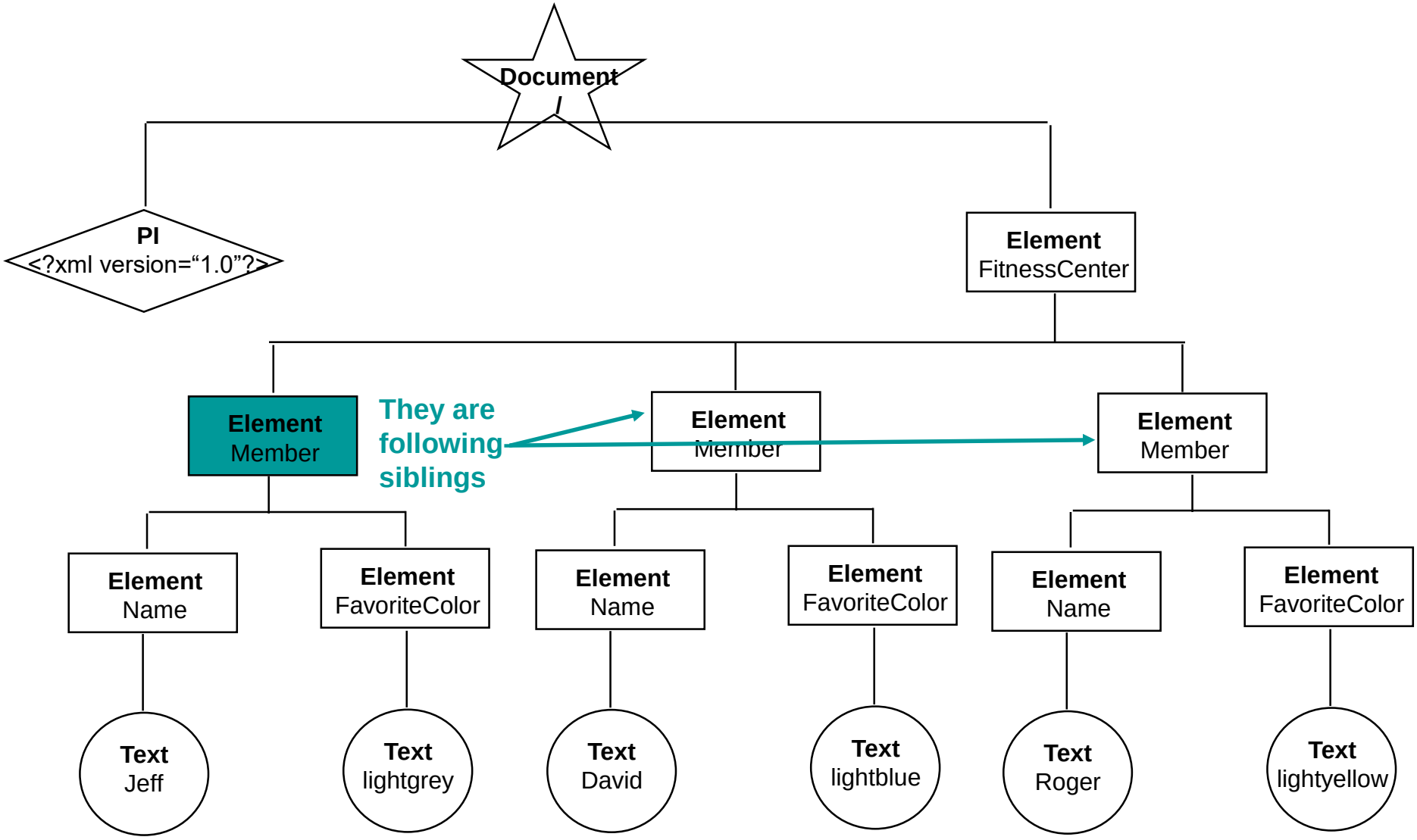

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
These are its descendant nodes
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

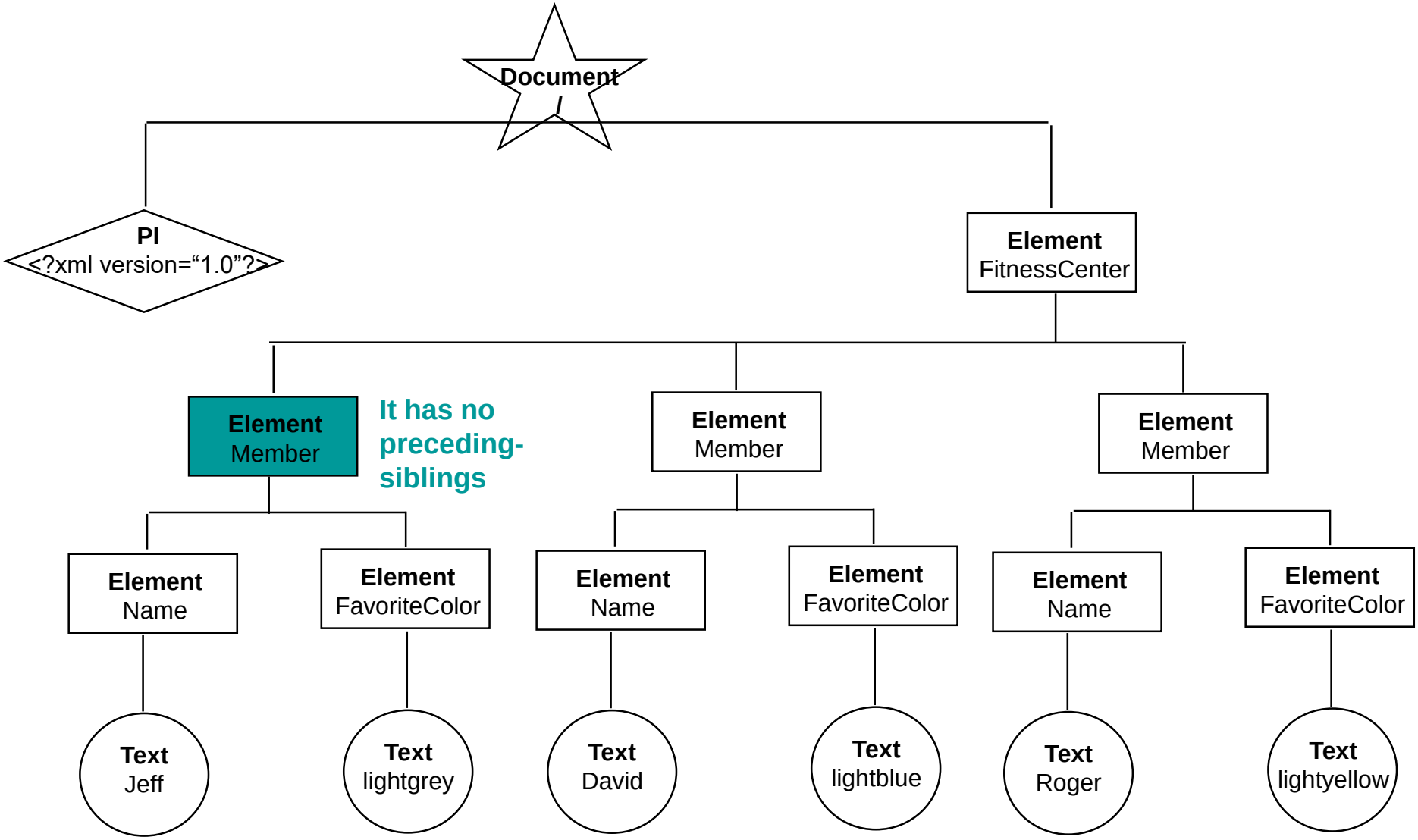

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
This is the context node
Element
Member
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
That's
its parent
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
Those are its ancestors
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

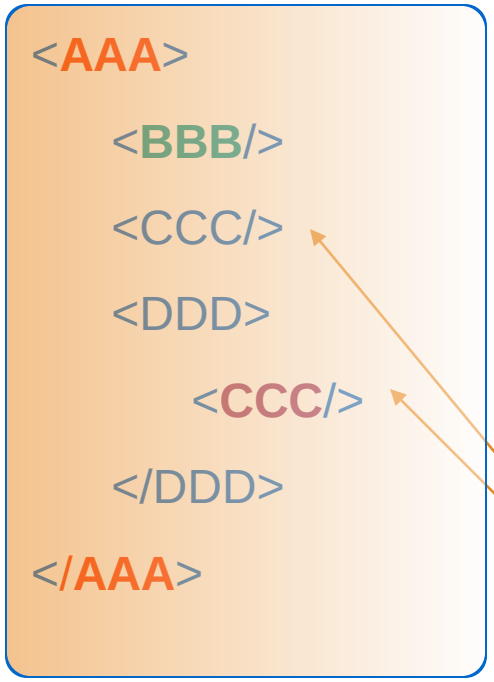

Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
It has 2
siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
They are
following
siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow


Document
PI
<?xml version="1.0"?>
Element
FitnessCenter
Element
Member
It has no preceding-siblings
Element
Member
Element
Member
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Element
Name
Element
FavoriteColor
Text
Jeff
Text
lightgrey
Text
David
Text
lightblue
Text
Roger
Text
lightyellow

XPATH

- **XPATH: Cú pháp cơ bản**
  - /: Đường dẫn tuyệt đối bắt đầu từ node gốc của tài liệu đến 1 node cụ thể



- /AAA: nút gốc
- /AAA/BBB: nút B là con của A
- /AAA/DDD/CCC: C – con D – con A (gốc)

  - //: Nút ở độ sâu bất kì

//CCC: nút CCC ở độ sâu bất kì

XPATH

- **XPATH: Cú pháp cơ bản**
  - *: Chọn tất cả các node với tên bất kỳ

```
<AAA>
    <BBB>
        <CCC/>
        <BBB>
            <CCC/>
        </BBB>
    </BBB>
</AAA>
```

- /AAA/*: Tất cả các nút con trực tiếp của AAA
- /*/BBB: Tất cả các nút B ở cấp thứ 2

//*: TẤT CẢ CÁC NODE

XPATH

- **XPATH: Cú pháp cơ bản**
  - [ ]: Truy xuất đến các node theo thứ tự nào đó hoặc theo chỉ mục.

    Ngoài ra, có thể dùng để chỉ định biểu thức điều kiện chọn lựa node

```
<AAA>
    <BBB/>
    <BBB/>
    <BBB>
        <CCC>
            content
        </CCC>
    </BBB>
    <BBB/>
</AAA>
```

- /AAA/BBB[1]: Nút BBB thứ 1 (con của AAA)
- /AAA/BBB[2]: Nút BBB thứ 2 (con của AAA)
- /AAA/BBB[**last()**]: Nút BBB cuối cùng
- /AAA/BBB[CCC=**“content”**]: Nút BBB thứ 3 (có nút con CCC với nội dung là **content**)

XPATH

- **XPATH: Cú pháp cơ bản**
  - @: Truy xuất đến thuộc tính

```
<AAA>
    <BBB id="b1"/>
    <BBB id="b2"/>
    <CCC name="ccc"/>
    <CCC/>
</AAA>
```

- /AAA/BBB[@id]:
  Những node BBB có thuộc tính **id**
- /AAA/CCC[@name="ccc"]:
  Những node CCC có giá trị thuộc tính **name = ccc**
- /AAA/CCC[@*]:
  Những node CCC có thuộc tính
- /AAA/CCC[**not**(@*)]: Những node CCC KHÔNG có thuộc tính

XPATH

- **XPATH: Cú pháp cơ bản**
  - count: hàm đếm

```
<AAA>
    <BBB>
        <DDD/>
        <DDD/>
    <BBB/>
    <CCC>
        <DDD/>
    <CCC/>
</AAA>
```

- //*[count(DDD)=2]:
  Tất cả các node có đúng 2 thẻ con DDD

- //*[count(*)=1]:
  Tất cả các node có đúng 1 thẻ con (tên thẻ con là gì cũng được)

XPATH

- **XPATH: Cú pháp cơ bản**
  - name(): Lấy tên thẻ, starts-with(), contains(): xử lý chuỗi

```
<AAA>
    <BBB>
        <DDD/>
    <BBB/>
    <ECC>
        <EEE/>
    <ECC/>
    <CFB/>
</AAA>
```

string-length(): chiều dài chuỗi

- //*[name()='DDD']:
  Những node có tên là DDD

- //*[starts-with(name(), 'E')]:
  Những node có tên bắt đầu bằng 'E'

- //*[contains(name(), 'B')]
  Những node mà tên có chứa 'B'

- //*[string-length(name())=3]
  Những node có chiều dài tên thẻ là 3

XPATH

- **XPATH: Cú pháp cơ bản**

```
<AAA>
    <BBB>
        <DDD/>
        <DDD>
            </EEE>
        </DDD>
    </BBB>
</AAA>
```

- Chọn ra tất cả thẻ “con cháu” của BBB ?

XPATH

- **XPATH: Axis – dùng để chọn các node trong mối quan hệ với node hiện thời**
  - **child::** : Chọn tất cả node con của node hiện thời

```
<AAA>
    <BBB>
        <DDD/>
    <BBB/>
    <ECC>
        <EEE/>
    <ECC/>
    <CFB/>
</AAA>
```

- /AAA/child::ECC
  ⟺ **/AAA/ECC**

- //BBB/child::*
  ⟺ **/AAA/BBB/DDD**

XPATH

- **XPATH: Axis**
  - descendant:: : Tất cả các thẻ "con cháu"

/AAA/child::*

```
<AAA>
    <BBB>
        <DDD>
            <EEE/>
        </DDD>
    </BBB>
</AAA>
```

/AAA/descendant::*

```
<AAA>
    <BBB>
        <DDD>
            <EEE/>
        </DDD>
    </BBB>
</AAA>
```

descendant-or-self:: = descendant:: + context node

XPATH

- **XPATH: Axis**
  - parent:: : Thẻ cha

```
<AAA>
    <BBB>
        <DDD/>
    <BBB/>
    <ECC>
        <EEE/>
    <ECC/>
    <CFB/>
</AAA>
```

- //BBB/parent::*

  Thẻ cha của BBB

- //EEE/parent::ECC

  Thẻ cha của ECC

XPATH

XPATH: Axis

- **ancestor::** : Tất cả các thẻ “cha ông”

/AAA/BBB/DDD/EEE/parent::*

```
<AAA>
    <BBB>
        <DDD>
            <EEE/>
        </DDD>
    </BBB>
</AAA>
```

/AAA/BBB/DDD/EEE/ancestor::*

```
<AAA>
    <BBB>
        <DDD>
            <EEE/>
        </DDD>
    </BBB>
</AAA>
```

ancestor-or-self:: = ancestor:: + context node

XPATH

- **XPATH: Axis**
  - following-sibling:: : Tất cả các thẻ “em”
  - preceding-sibling:: : Tất cả các thẻ “anh”

//EEE/following-sibling::*

```
<AAA>
    <BBB>
        <DDD/>
        <EEE/>
        <FFF/>
        <GGG/>
    </BBB>
</AAA>
```

//EEE/preceding-sibling::*

```
<AAA>
    <BBB>
        <DDD/>
        <EEE/>
        <FFF/>
        <GGG/>
    </BBB>
</AAA>
```

XPATH

- **XPATH: Axis**
  - following:: Tất cả các thẻ có thẻ mở xuất hiện sau thẻ đóng của thẻ hiện hành
  - preceding:: Tất cả các thẻ có thẻ đóng xuất hiện trước thẻ mở của thh

//BBB/following::*

```
<AAA>
    <BBB>
        <DDD/>
    </BBB>
    <CCC>
        <EEE/>
    </CCC>
</AAA>
```

//EEE/preceding::*

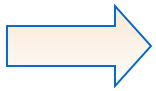
```
<AAA>
    <BBB>
        <DDD/>
    </BBB>
    <CCC>
        <EEE/>
    </CCC>
</AAA>
```

XPATH

- **XPATH: Axis**

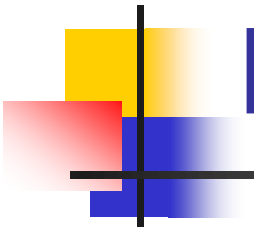
```
<AAA>
    <BBB>
        <DDD/>
    </BBB>
    <CCC>
        <EEE>
            <FFF/>
        </EEE>
        <GGG/>
    </CCC>
</AAA>
```

- //EEE/ancestor::*
- //EEE/descendant::*
- //EEE/following::*
- //EEE/preceding::*
- //EEE/self::*



5 axis trên tạo thành 5 phân hoạch của tài liệu xml

- Không overlap nhau
- “Or” (|) lại sẽ cho kết quả là tất cả các node của tài liệu



Kết luận

- Ngôn ngữ XML rất dễ dàng sử dụng và có ứng dụng đa dạng
- XML mô tả dữ liệu có cấu trúc nên được dùng trong tổ chức CSDL

Bài tập: tìm DTD cho tài liệu XML

```
<?xml version="1.0"?>
<DatHang maso="23456">
        <ngay>12/5/2006</ngay>
        <khachhang maso="237X">Cong ty TNHH Thanh Nghia</khachhang>
        <sanpham>
                <ma kho="kho so 11">F563-A34</ma>
                <mota>San xuat tai nhieu quoc gia</mota>
                <soluong>15</soluong>
        </sanpham>
</DatHang>
```

Xây dựng schema cho tài liệu XML

```
<?xml version="1.0"?>
<DatHang maso="23456">
        <ngay>12/5/2006</ngay>
        <khachhang maso="237X">Cong ty TNHH Thanh Nghia</khachhang>
        <sanpham>
                <ma kho="kho so 11">F563-A34</ma>
                <mota>San xuat tai nhieu quoc gia</mota>
                <soluong>15</soluong>
        </sanpham>
</DatHang>
```

Câu hỏi