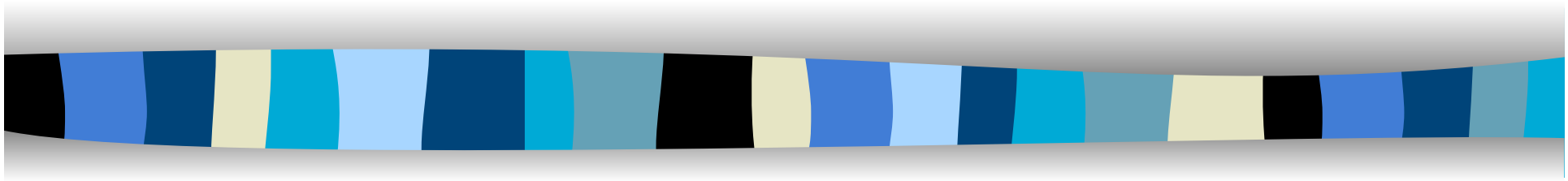
XML: sintassi



Cosa è XML – 1

- XML (Extensible Markup Language) è un linguaggio di markup
- È stato progettato per lo scambio e la interusabilità di documenti strutturati su Internet
- XML è un **metalinguaggio**, cioè un linguaggio che *permette di definire nuovi linguaggi*.
- XML deriva da **SGML**,
  - È stato *estremamente semplificato* ed *esteso* in certi aspetti.

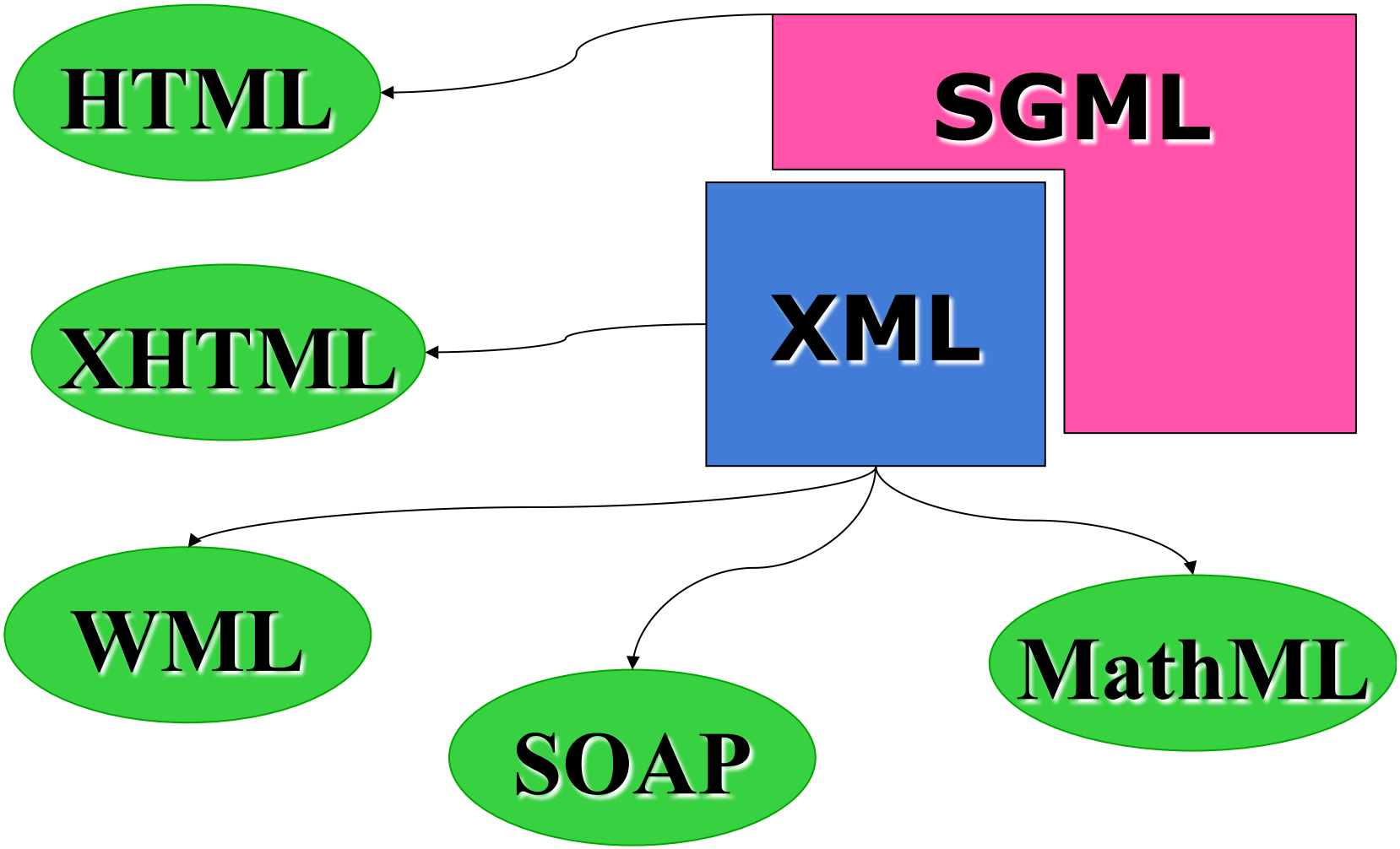
Cosa è XML – 2

- XML è un ambiente per definire linguaggi di markup
- Non c'è una collezione fissata di tag di markup, possiamo definire i tag a nostro piacimento, i tag sono ideati a seconda del tipo di informazione che devono rappresentare
- Ogni linguaggio XML è progettato per una particolare applicazione, ma tutti i linguaggi condividono molte caratteristiche
- Esiste un insieme di strumenti generici per elaborare i documenti XML

A cosa serve XML

- XML viene usato per
  - **definire tutti i linguaggi di tipo markup**
  - **ridefinire** in maniera più formale quelli già esistenti (e.g, XHTML)
  - **semplificare le specifiche** di linguaggi di markup originariamente scritte in SGML

La famiglia di XML



Esempio di documento XML

Mancano alcune dichiarazioni iniziali

```
<collection>
    <CD number="1">
        <song album="santana1" track="11">
            <title>African Bamba</title>
            <length>4:42</length>
        </song>
        <song album="santana1" track="9">
            <artist>Santana &amp;amp; Mana</artist>
            <title>Corazon Espinado</title>
            <length>4:36</length>
        </song>
        <album ID="santana1">
            <artist>Santana</artist>
            <title>Supernatural</title>
            <year>1999</year>
        </album>
    </CD>
</collection>
```

XML: Vantaggi – 1

- XML permette agli sviluppatori di creare facilmente **strutture ad-hoc** per contenere informazione strutturata.
- I **parser XML** sono diffusi su tutte le piattaforme e *free*. Gli sviluppatori possono utilizzare questi parser per decodificare e validare le strutture XML, limitandosi poi a gestire solo l'informazione contenuta usando API specifiche (**DOM**, **SAX**).

XML: Vantaggi – 2

- XML è completamente **text-based**, quindi leggibile anche dagli esseri umani e facilmente editabile anche a mano. Supporta **UNICODE**, quindi è adatto a ogni tipo di scrittura/lingua.
- Essendo dati testuali strutturati esattamente come HTML, i dati XML **possono essere trasportati usando il protocollo HTTP** anche attraverso *firewall* (**SOAP**).

XML: Svantaggi

- I documenti XML, a causa delle struttura testuale e dei *tag*, tendono ad essere **molto più ingombranti** di quelli in formato binario, quindi la loro trasmissione sulla rete non è ottimale.
- I **parser XML non sono veloci** come quelli scritti *ad-hoc* per formati specifici, soprattutto se binari.

XML: Applicazioni

- **Protocolli RMI**
  - SOAP
- **Matematica**
  - MathML
- **Web**
  - XHTML, WML
- **Multimedia**
  - SMIL
- **Musica**
  - MusicML

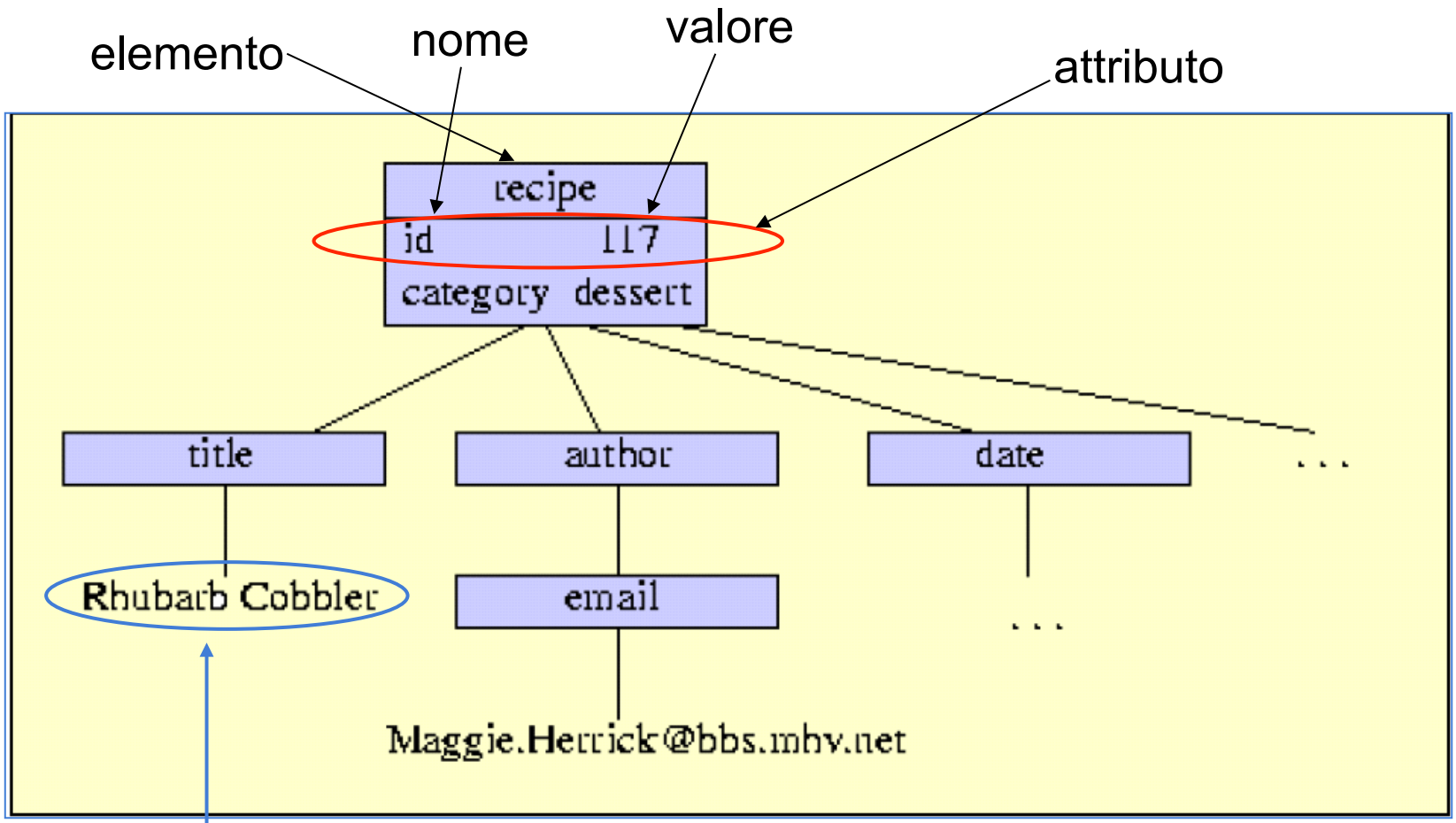
Documento XML – 1

- Un documento XML è un albero etichettato ed ordinato
- Esistono due tipi di nodi
  - Nodi dati (character data) -- (e.g., testo)
  - Nodi elementi (element) -- (e.g., tag html)
    - I nodi dati sono nodi foglia dell'albero
    - Contengono i dati del documento XML
    - In genere un nodo dato deve essere non vuoto e non adiacente ad altri nodi dati

Documento XML – 2

- I nodi elementi sono tutti etichettati con
  - Un nome (spesso chiamato il *tipo* dell'elemento)
  - Un insieme opzionale di attributi, ognuno consistente di un nome e di un valore
- Questi nodi possono avere nodi figli

Esempio


elemento
nome
valore
attributo
recipe
id 117
category dessert
title
author
date
. . .
Rhubarb Cobbler
email
. . .
Maggie.Herrick@bbs.mhv.net
nodo dato

Altri tipi di nodi

- Un albero/documento XML può contenere ulteriori tipi di nodi
  - **Processing Instructions**
    - Annotazioni per i programmi che elaboreranno il documento XML
  - **Dichiarazione del tipo di documento**
    - "i tipi di dati" che compongono il documento
  - **Commenti**
    - Come nei linguaggi di programmazione

Processing Instruction (PI)

- Le *Processing Instructions* (PI) vengono usate per passare informazioni **extra-markup** ai programmi che manipoleranno il file XML. Possono apparire ovunque, dopo la *dichiarazione XML*.

<?*target data* ?>

- ***target*** identifica il **destinatario della PI**.
- ***data*** è una **stringa di dati** per la PI. Non deve seguire alcuna regola XML, e può contenere anche caratteri riservati.

Esempio di PI

Tecnologie di Sviluppo per il web
HTML
XML
PHP
CSS
Javascript

- <?flubber pg=9 recto?>
- <?thingie?>
- <?xyz stop: the presses?>

- <title> Tecnologie di Sviluppo per il web <?lb?>HTML <?lb?> XML <?lb?>PHP <?lb?>CSS <?lb?>Javascript </title>

Prologo Documento XML

- Dichiarazione XML
  - Dichiarazione obbligatoria
  - Viene posta all'inizio del documento
- Dichiarazione DOCTYPE
  - Dichiarazione obbligatoria solo se il documento deve essere validato
  - Viene posta all'inizio del documento

Dichiarazione XML


```
<?xml version="1.0" encoding="UTF-8"
  standalone="yes" ?>
```

- Gli attributi sono:
  - ***version***: (*obbligatorio*) la **versione di XML** usata.
  - ***encoding***: (*opzionale*) nome della **codifica dei caratteri** usata nel documento.
    - Default: UTF-8 o 16
  - ***standalone***: (*opzionale*) se vale *yes* indica che il file **non** fa **riferimento ad altri file esterni** (e.g. DTD).
    - Default: no

Dichiarazione DOCTYPE

<!DOCTYPE *RootElement* *ExternalDTDReference*
[*InternalDTDSubset*]>

- ***RootElement*** (*obbligatorio*) è il nome dell'elemento radice del documento.
- ***ExternalDTDReference*** (*opzionale*) può essere:
  - SYSTEM "*uri*", dove *uri* punta a un **DTD esterno** contenente la definizione del linguaggio usato nel documento.
  - PUBLIC "*pubid*" "*uri*", dove *pubid* è un identificatore alternativo per il **DTD pubblico** puntato da *uri*.
- ***InternalDTDSubset*** (*opzionale*) è un DTD specificato *inline* al documento.

Esempi di DTD – PUBLIC

- <!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01//EN">

- <!DOCTYPE time-o-gram PUBLIC
  "-//LordsOfTime//DTD TimeOGram 1.8//EN"
  http://www.lordsoftime.org/DTDs/timeogram.dtd
  >

Esempi di DTD – SYSTEM

- <!DOCTYPE doc SYSTEM "http://www.dtds-r-us.com/generic.dtd">

- <!DOCTYPE press-release SYSTEM http://www.dtdland.org/dtds/reports.dtd
  [
  <!ENTITY bobco "Bob's Bolt Bazaar, Inc.">
  ]>

Schema XML

- Invece della DTD si utilizza uno schema
  - Documento XML per descrivere la struttura e la semantica di un documento XML

```
<card xmlns="http://businesscard.org"
      xmlns:xsi=
        "http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
        "http://businesscard.org http://…./business_card.xsd">

  …
</card>
```

url

Commenti in XML

- I commenti sono utili agli esseri umani, e vengono ignorati dai parser XML. Possono apparire ovunque tranne che all'interno degli attributi.

```
<!-- questo è un commento -->
```

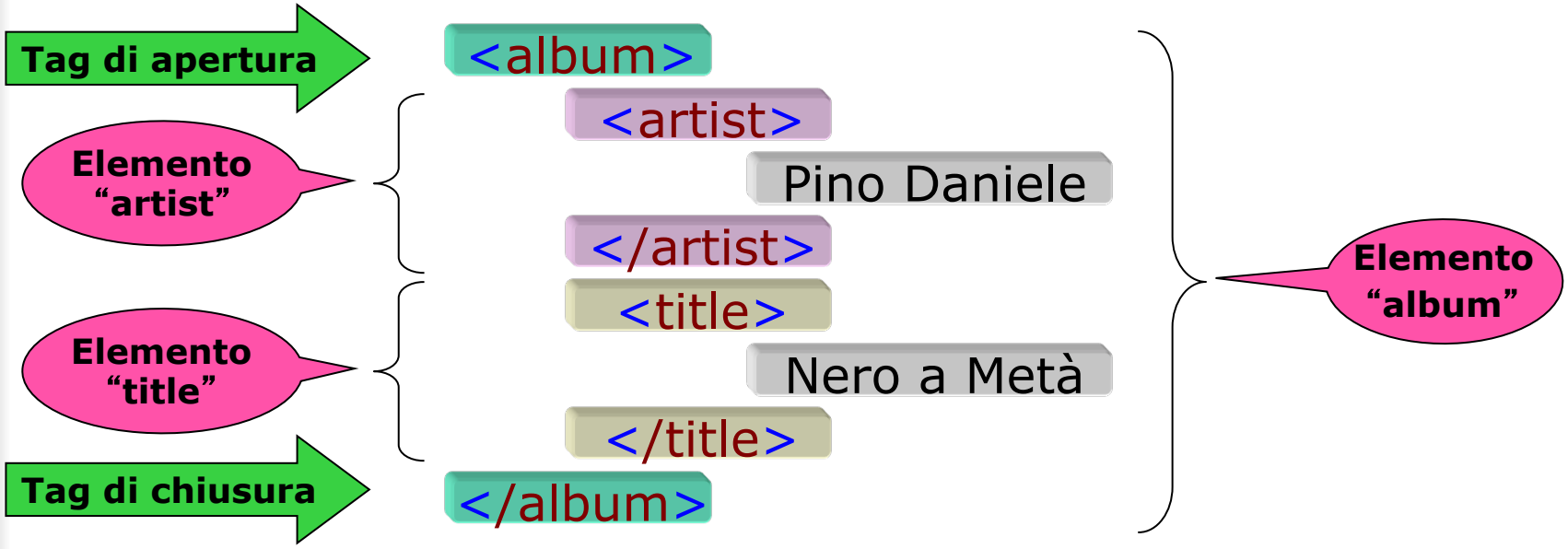
- Il commento si apre con un <!-- ed è chiuso da un -->, che quindi non può apparire nella stringa interna. Il contenuto non deve seguire regole e può anche contenere caratteri riservati.

Elementi XML

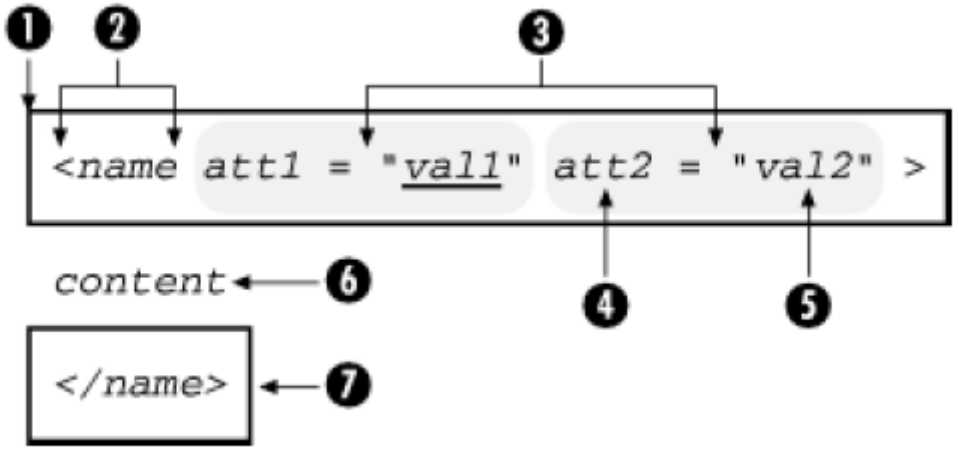
- Gli elementi sono la base di un linguaggio XML
  - Il documento è diviso in parti a cui sono associati degli elementi
- Gli elementi possono essere contenitori
  - Mix di testo ed altri elementi
- Gli elementi possono essere vuoti
  - Non hanno contenuto
- Gli elementi possono avere degli attributi
  - Aggiungono informazioni agli elementi

Elementi

- Un elemento è un frammento di dati, limitato ed identificato (tramite un nome) da un tag.

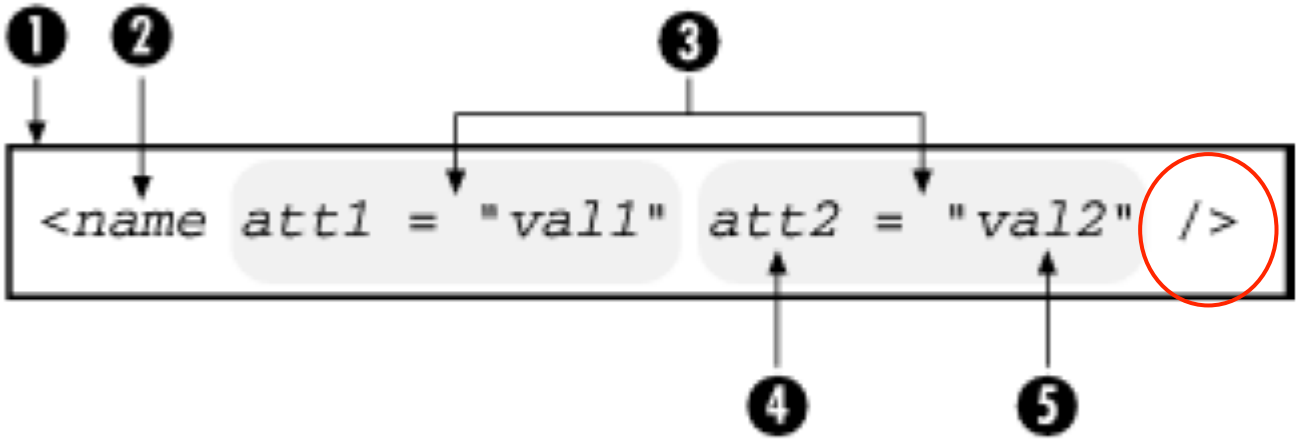


Sintassi per elemento contenitore



1. Tag di apertura
2. Nome contenitore
3. Lista di attibuti
4. Nome attributo
5. Valore attributo
6. Contenuto
7. Tag di chiusura

Sintassi per elemento vuoto



1. Tag di apertura
2. Nome contenitore
3. Lista di attibuti
4. Nome attributo
5. Valore attributo

   L' elemento termina con uno slash (/) ed una parentesi angolare chiusa

Elementi: regole

- I nomi degli elementi sono case-sensitive
- Ogni elemento contenitore aperto deve essere chiuso entro la fine del documento
- Gli elementi possono essere nidificati, in tal caso, vanno chiusi esattamente nell'ordine inverso a quello di apertura
- Un documento XML deve avere un unico elemento "radice", in cui tutti gli altri sono nidificati

Nota sull'elemento vuoto

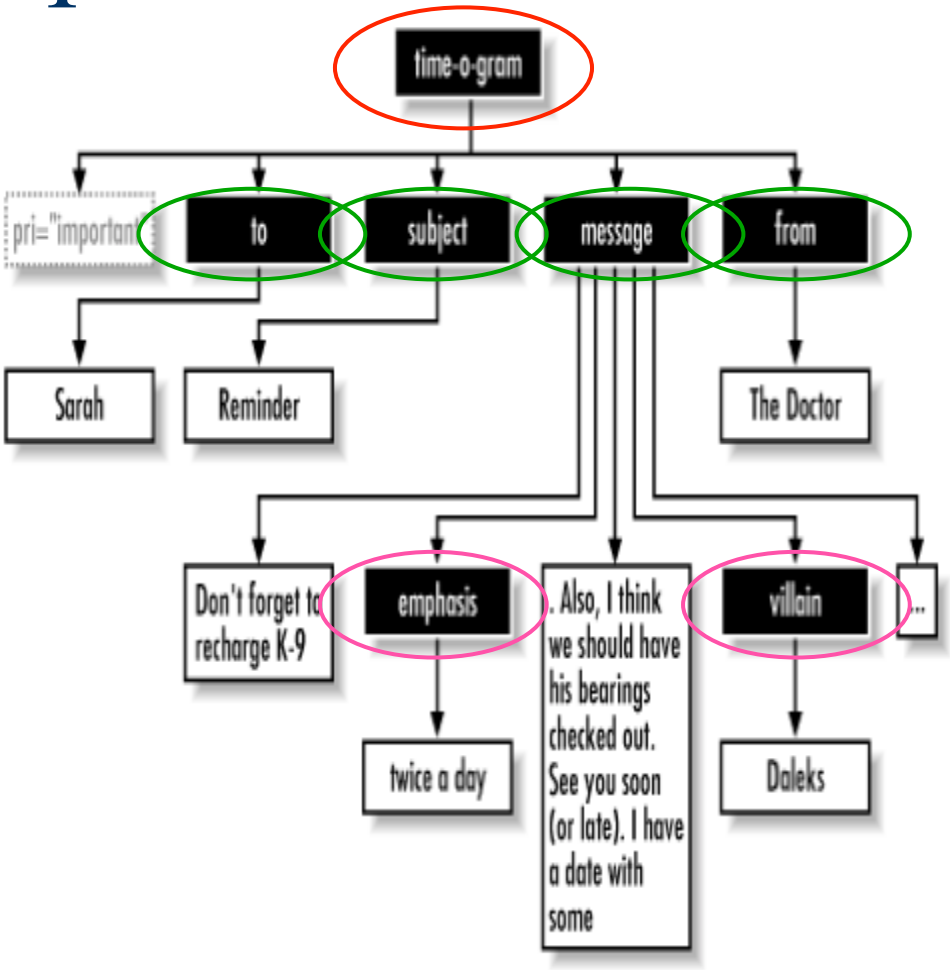
- Alcuni elementi possono essere privi di contenuto
- In questo caso è possibile omettere il tag di chiusura scrivendo quello di apertura come abbiamo già visto
- Il elemento <nome attributi /> è equivalente a <nome attributi> </nome>

Esempio codice XML – 1

```
<?xml version="1.0"?>
<time-o-gram pri="important">
    <to>Sarah</to>
    <subject>Reminder</subject>
    <message>Don't forget to recharge K-9
        <emphasis>twice a day</emphasis>.
        Also, I think we should have his
        bearings checked out. See you soon
        (or late). I have a date with some
        <villain>Daleks</villain>...
    </message>
    <from>The Doctor</from>
</time-o-gram>
```

mixed content

Esempio codice XML – 2



Albero del documento

- Gli elementi, nidificandosi, creano la struttura ad albero tipica dei documenti XML.
- All'interno di questa struttura si definiscono alcuni "*rapporti di parentela*" utili per individuare gli elementi
  - padre/figlio
  - fratelli
  - predecessore/discendente

Attributi

- Gli attributi permettono di specificare proprietà degli elementi come coppie nome-valore.
- Sono usati per definire proprietà che non possono o non si vogliono inserire nel contenuto dell'elemento.
- Vengono specificati all'interno dei tag di apertura degli elementi.
- Al contrario degli elementi, per gli attributi l'ordine di presentazione non è significativo.

Quando usare gli attributi

- Si possono usare quando non ha senso utilizzare un elemento figlio per specificare la proprietà che l'attributo indica
  - Abbiamo un documento XML che rappresenta una collezione di CD musicali

```
<CD number="1">
        <song album="santana1" track="11">
                <title> African Bamba</title>
                <length> 4:42</length>
        </song>
        <!-- altri elementi di tipo song>
        <album ID="santana1">
                <artist> Santana</artist>
                <title> Supernatural</title>
                <year> 1999</year>
        </album>
</CD>
```

Regola generale

- Usare elementi per dati che devono essere presentati ed attributi per dati di sistema

```
<chapter number="23"
         focus="Server-side programming">
   XML Processing with Java
</chapter>
```

```
<chapter>
<number> 23 </number>
<focus> Server-side programming </focus>
<title> XML Processing with Java </title>
</chapter>
```

Attributi: regole

- I nomi degli attributi sono **case-sensitive**
- Lo stesso tag non può contenere due attributi con lo stesso nome
- Non sono ammessi attributi senza valore (solo nome)
- Il valore degli attributi deve essere specificato tra virgolette semplici o doppie
- Il valore può contenere riferimenti ad entità
- Il valore non può contenere markup, sezioni CDATA, virgolette uguali a quelle iniziali

Sezione CDATA

- Permette di definire esplicitamente un'area in cui si trova solo testo semplice

  <!CDATA[<<solo testo!>>]]>

- È utile per impedire che il parser consideri markup delle stringhe che ne hanno solo la forma
- All'interno delle sezioni CDATA sono ammessi tutti i caratteri UNICODE, e anche le entità non sono riconosciute come tali.

Nota sul testo

- Il testo che possiamo inserire nei documenti XML comprende tutti i caratteri definiti in UNICODE
- È possibile inserire caratteri speciali o riservati tramite entità carattere
- È possibile inserire stringhe predefinite tramite entità generali
- Non è possibile usare esplicitamente i caratteri ‘>’, ‘<’ e ‘&’, per i quali è sempre necessario usare le corrispondenti entità carattere

Documento XML *ben formato*

- Un documento XML è *ben formato*
  - se rispetta le regole generali di sintassi viste nella parte precedente
- Un documento ben formato deve sottostare a limitazioni esclusivamente sintattiche

Documento XML *valido*

- Un documento XML è *valido*
  - se è ben formato e rispetta le regole sintattiche e semantiche contenute del DTD o nello Schema XML associato
  - Un documento senza DTD o Schema XML non è mai valido
- Un documento valido soddisfa dei vincoli imposti dalla semantica/struttura dei suoi elementi