



CORSO DI LAUREA IN INFORMATICA

# PROGRAMMAZIONE WEB

XML

a.a 2015-2016

# Che cos'è XML?

- XML: **E**xensible **M**arkup **L**anguage:
  - è un linguaggio che consente la rappresentazione di documenti e dati strutturati su supporto digitale
  - è uno strumento potente e versatile per la creazione, memorizzazione e distribuzione di documenti digitali
  - la sua **sintassi rigorosa** e al contempo **flessibile** consente di utilizzarlo nella rappresentazione di dati strutturati anche molto complessi

# Le origini

- XML è stato sviluppato dal **World Wide Web Consortium**
- Nel **1996** è stato formato un gruppo di lavoro con l'incarico di definire un linguaggio a markup estensibile di uso generale
- Le specifiche sono state rilasciate come **W3C Recommendation** nel 1998 e aggiornate nel 2004
- XML deriva da **SGML (Standard Generalized Markup Language)**, un linguaggio di mark-up dichiarativo sviluppato dalla International Standardization Organization (ISO), e pubblicato ufficialmente nel 1986 con la sigla ISO 8879
- XML nasce come un sottoinsieme semplificato di SGML orientato all' utilizzo su World Wide Web
- Ha assunto ormai un ruolo autonomo e una diffusione ben maggiore del suo progenitore

# XML come linguaggio di markup

- Un linguaggio di markup è composto da istruzioni, **definite tag** o **marcatori**, che descrivono la struttura e la forma di un documento
  - Ogni marcatore (o coppia di marcatori) identifica **un elemento** o componente del documento
- I marcatori vengono inseriti all'interno del documento
  - Un documento XML è “leggibile” da un utente umano senza la mediazione di software specifico

# Esempio

- Un documento XML è leggibile ,chiaro, intuibile:

```
<documento>  
  <corpo>  
    Testo del mio primo documento  
  </corpo >  
</documento>
```

Marcatore di inizio

Marcatore di fine

- **Attenzione: XML è case sensitive**
  - nei **nomi dei tag** distingue fra maiuscole e minuscole

## Altro esempio

**<prenotazione>**

**<idVolo>PA321</idVolo>**

**<idCliente>PP2305</idCliente>**

**<data>22-10-2001</data>**

**<prezzo valuta="Euro">245</prezzo>**

**</prenotazione>**

## Esempio 3

**<utenti>**

**<utente>**

**<nome>**Luca**</nome>**

**<cognome>**Cicci**</cognome>**

**<indirizzo>**Milano**</indirizzo>**

**</utente>**

**<utente>**

**<nome>**Max**</nome>**

**<cognome>**Rossi**</cognome>**

**<indirizzo>**Roma**</indirizzo>**

**</utente>**

**</utenti>**

# Come può essere usato XML?

- XML separa i dati dalla presentazione
  - Non fornisce nessuna informazione su come i dati debbano essere visualizzati
  - Lo stesso XML può essere usato in differenti scenari di presentazione
- XML è spesso un complemento di HTML
  - *Spesso XML viene usato per memorizzare o trasportare i dati, mentre HTML è utilizzato per formattare e visualizzare i dati*
- XML separa i dati da HTML
  - Con XML, quando si visualizzano i dati non è necessario editare il file HTML se i dati cambiano
  - Con XML i dati sono memorizzati in file separati
  - Utilizzando JavaScript è possibile leggere un file XML e aggiornare i dati di una pagina HTML



# Dati per le transazioni

- Esistono migliaia di formati XML, in molte industrie differenti, per descrivere le transazioni di ogni giorno:
  - Stocks and Shares (titoli e azioni)
  - Financial transactions
  - Medical data
  - Mathematical data
  - Scientific measurements
  - News information
  - Weather services
  - ...

# Esempio: XML News

- È una specifica per scambiare news

```
<?xml version="1.0" encoding="UTF-8"?>
<nitf>
  <head>
    <title>Colombia Earthquake</title>
  </head>
  <body>
    <headline>
      <h1>143 Dead in Colombia Earthquake</h1>
    </headline>
    <byline>
      <bytag>By Jared Kotler, Associated Press Writer</bytag>
    </byline>
    <dateline>
      <location>Bogota, Colombia</location>
      <date>Monday January 25 1999 7:28 ET</date>
    </dateline>
  </body>
</nitf>
```

# XML: caratteristiche

- XML è **indipendente dal tipo di piattaforma** hardware e software su cui viene utilizzato
- Permette la rappresentazione di qualsiasi tipo di documento (e di struttura) indipendentemente dalle finalità applicative
- È **indipendente** dai dispositivi di archiviazione e visualizzazione:
  - può essere archiviato su qualsiasi tipo di supporto digitale
  - può essere visualizzato su qualsiasi dispositivo di output
  - può essere facilmente trasmesso via Internet tramite i protocolli HTTP, SMTP, FTP

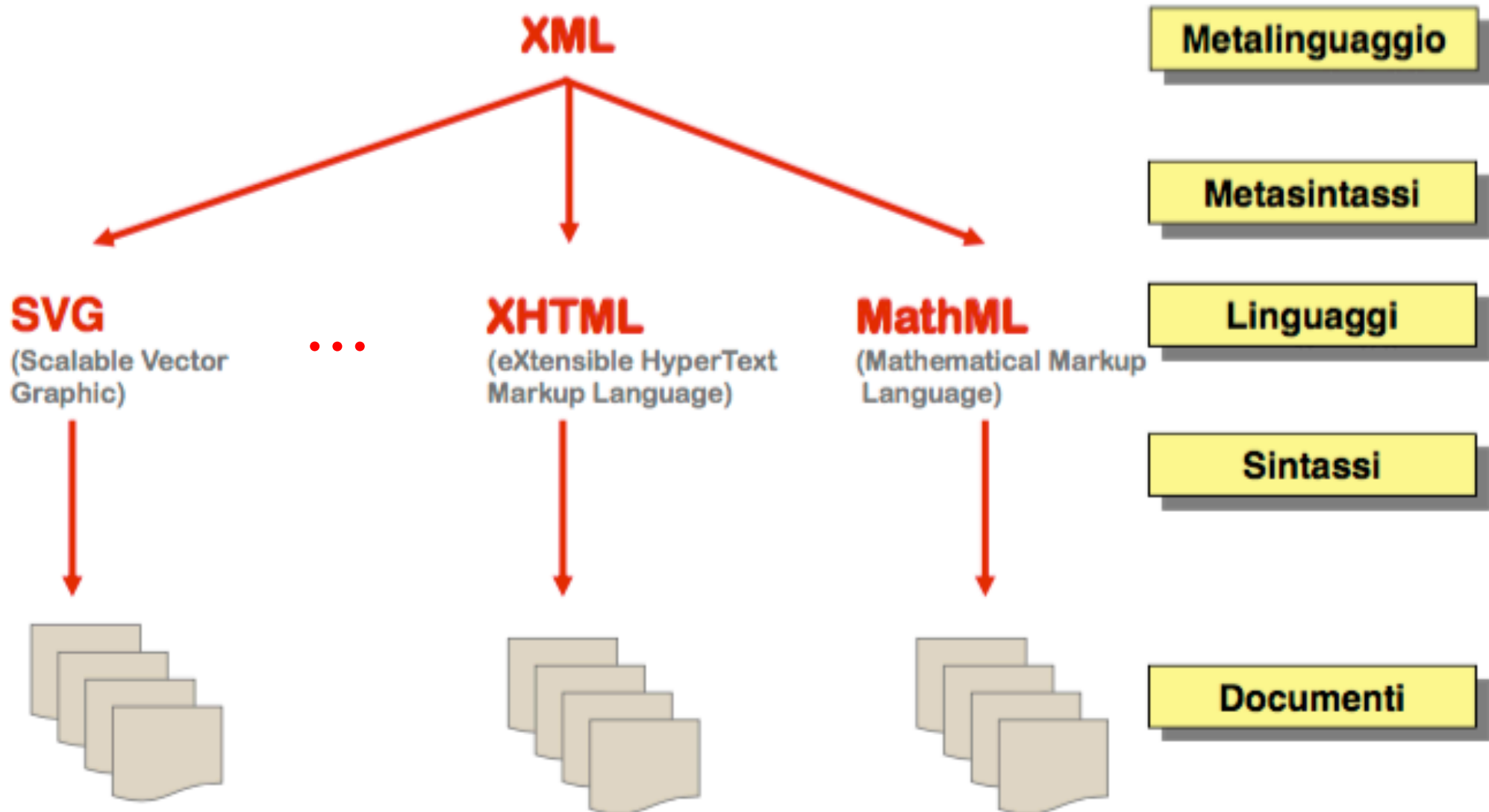
# XML: caratteristiche

- XML è uno **standard di pubblico dominio**
- Ogni software “*conforme a XML*” è in grado di gestire dati in formato XML
- Sono disponibili numerose applicazioni e librerie open source per la manipolazione di dati in formato XML basate su diversi linguaggi di programmazione (Java, C, C#, Python, Perl, PHP...)
- Una applicazione in grado di elaborare dati in formato XML viene definita **elaboratore XML**

# XML come metalinguaggio

- XML è **un metalinguaggio**
  - Definisce un insieme **regole** (meta-)sintattiche, attraverso le quali è possibile **descrivere formalmente un linguaggio di markup**, detto applicazione XML
- Ogni applicazione XML:
  - eredita un insieme di caratteristiche sintattiche comuni
  - definisce una sua sintassi formale
  - è dotata di una semantica

# Metalinguaggio e linguaggi



# Documenti ben formati e documenti validi

- In XML ci sono regole sintattiche (o meglio meta-sintattiche)
  - come dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere (ma non è obbligatorio) regole semantiche
  - cosa possiamo scrivere in un documento XML
- Un documento XML che rispetta le regole sintattiche si dice ben formato (**well-formed**)
- Un documento XML che rispetta le regole sintattiche e le regole semantiche si dice **valido**

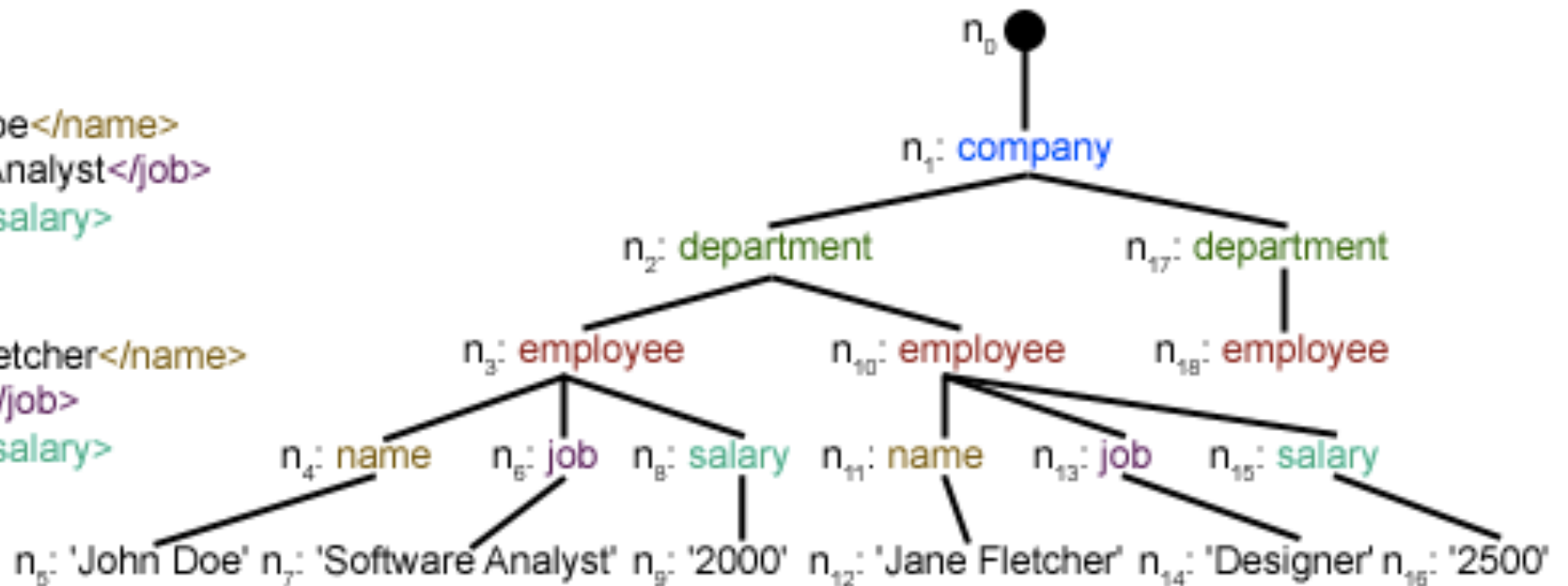
# Struttura **logica** di un documento XML

- Un **documento XML**
  - è strutturato in modo gerarchico
  - è composto da elementi
- Un **elemento**
  - rappresenta un componente logico del documento
  - può contenere un frammento di testo oppure altri elementi (sotto-elementi)
- Ad un elemento possono essere associate informazioni descrittive chiamate **attributi**
- Gli elementi sono organizzati ad albero con radice **root**
- Ogni documento XML può essere rappresentato come un albero
  - **document-tree**



# XML Document-Tree

```
<company>  
  <department>  
    <employee>  
      <name>John Doe</name>  
      <job>Software Analyst</job>  
      <salary>2000</salary>  
    </employee>  
    <employee>  
      <name>Jane Fletcher</name>  
      <job>Designer</job>  
      <salary>2500</salary>  
    </employee>  
  </department>  
  <department>  
    <employee>  
    </employee>  
  </department>  
</company>
```



# Ogni documento XML deve avere una radice

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

- Esempio con radice **<note>**

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

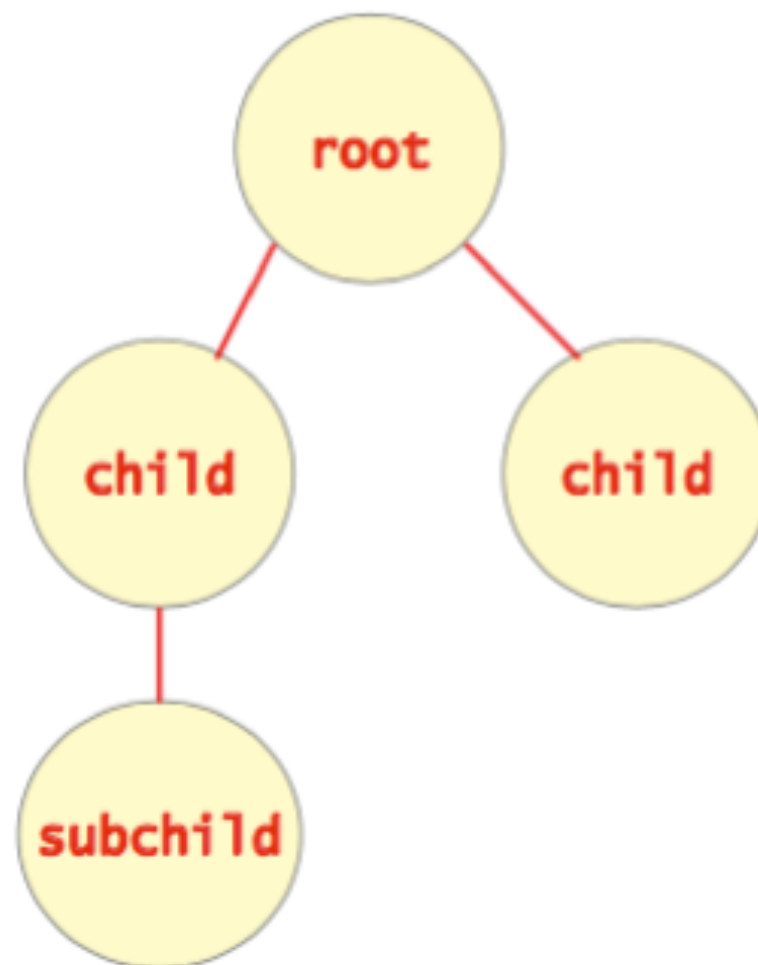
# Struttura **fisica** di un documento XML

- Un documento XML è un semplice file di testo (.xml)
- La struttura del documento viene rappresentata mediante marcatori (**markup**)
- Gli elementi sono rappresentati mediante **tag**
  - coppie di marcatori che racchiudono il contenuto dell'elemento
- I **sottoelementi** sono tag contenuti all'interno di un altro tag
- Gli **attributi** vengono rappresentati sotto forma di coppie *nome-valore* all'interno dei tag
- La **radice** è un tag che racchiude tutto il resto del documento (e quindi tutti gli altri tag)
- Un documento può inoltre contenere *spazi bianchi, a capo e commenti*

# Struttura logica e fisica

- Esiste una corrispondenza diretta fra struttura fisica e struttura logica (**tree**)

```
<root>  
  <child>  
    <subchild>  
    ...  
  </subchild>  
</child>  
<child>  
  ...  
</child>  
</root>
```



# Aspetti di sintassi

- Un documento XML è una stringa di caratteri ASCII o Unicode
- Nomi di elementi, attributi e entità sono **case-sensitive**
- Il mark-up è separato dal contenuto testuale mediante caratteri speciali:
  - < > &** (parentesi angolari e ampersand)
  - " '** (doppi apici e apici)
- I caratteri speciali non possono comparire come contenuto testuale e devono essere eventualmente sostituiti mediante i riferimenti a entità

**&lt;** (<), **&gt;** (>), **&amp;** (&), **&quot;** ("), and **&apos;** (')

# Struttura formale di un documento XML

- Un documento è costituito da due parti:
  - **Prologo:** contiene una dichiarazione XML ed il riferimento (opzionale) ad altri documenti che ne definiscono la struttura o direttive di elaborazione
  - **Corpo:** è il documento XML vero e proprio

```
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/css" href="gree.css"?>
```

Prologo

```
<root>  
  <!-- Questo è un commento -->  
  <child>  
    ...  
  </child>  
  <child>  
    ...  
  </child>  
</root>
```

Corpo

## Prologo: XML Declaration

- Ogni documento XML inizia con un prologo che contiene una **XML declaration**

- Forme di XML declaration:

`<?xml version="1.0"?>`

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

- Contiene informazioni su:
  - Versione: per ora solo 1.0
  - Set di caratteri (*opzionale*)

# Prologo: riferimenti a documenti esterni

- Il prologo può contenere riferimenti a documenti esterni utili per il trattamento del documento

- **Processing instruction:** istruzioni di elaborazione

Esempio. Rappresentazione mediante CSS:

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

- **Doctype declaration:** grammatica da utilizzare per la validazione del documento

- grammatica contenuta in un file locale

```
<!DOCTYPE book SYSTEM "book.dtd">
```

- grammatica accessibile ad un URL pubblico

```
<!DOCTYPE book PUBLIC "http://www.books.org/book.dtd">
```



# Commenti

- I commenti possono apparire ovunque in un documento XML (sia nel prologo che nel corpo)
- I commenti sono utili per:
  - spiegare la struttura del documento XML
  - commentare parti del documento durante le fasi di sviluppo e di test del nostro software
- I commenti non vengono mostrati dai browser ma sono visibili da parte di chi guarda il codice sorgente del documento XML

`<!-- Questo è un commento -->`

# Element e Tag

- Un **elemento** è un frammento di testo racchiuso fra uno start tag e un end tag
- Uno **start tag** è costituito da un nome più eventuali attributi racchiusi dai simboli '<', '>'

**<TagName attribute-list>**

- Un **end tag** è costituito da un nome (lo stesso dello start tag) racchiuso da '</','>':

**</TagName>**

- Un tag vuoto è rappresentabile come:

**<TagName attribute-list />**

- Equivale a

**<TagName attribute-list></TagName>**

- Attenzione: I tag non possono avere nomi che iniziano per XML, XML, Xml, xml...

# Elementi

- Un elemento può contenere testo, attributi, altri elementi, un mix delle voci precedenti. Esempio:
- `<title>`, `<author>`, `<year>`, and `<price>` have **text content** because they contain text (like Harry Potter and 29.99)
- `<bookstore>` and `<book>` have **element contents**, because they contain elements
- `<book>` has an **attribute** (`category="children"`)

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

# Attributi

- A ogni elemento possono essere associati uno o più **attributi** che ne specificano ulteriori caratteristiche o **proprietà** non strutturali
- Ad esempio:
  - la lingua del suo contenuto testuale
  - un identificatore univoco
  - un numero di ordine
  - ...
- Gli attributi XML sono caratterizzati da
  - un **nome** che li identifica
  - un **valore**

# Esempio di documento con attributi

```
<?xml version="1.0" ?>
<articolo titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo>Blocco di testo del primo
paragrafo</testo>
    <immagine file="immagine1.jpg"></immagine>
  </paragrafo>
  <paragrafo titolo="Titolo del secondo paragrafo">
    <testo>Blocco di testo del secondo
paragrafo</testo>
    <codice>Esempio di codice</codice>
    <testo>Altro blocco di testo</testo>
  </paragrafo>
  <paragrafo tipo="bibliografia">
    <testo>Riferimento ad un articolo</testo>
  </paragrafo>
</articolo>
```

# Elementi o attributi?

- Qualche regola per decidere:
  - Un **elemento è estendibile** in termini di contenuto (con elementi figli) e di attributi
  - Un **attributo non è estendibile**: può solo modellare una proprietà di un elemento in termini di valore
  - Un elemento è un'entità a se stante (un oggetto?)
  - Un attributo è strettamente legato ad un elemento
  - Un attributo può solamente contenere un valore "atomico"
- *In pratica non c'è una regola valida in assoluto*
- La scelta dipende da diversi fattori: leggibilità, semantica, tipo di applicazione, efficienza...

## Elementi o attributi: esempio

- Vediamo tre varianti dello stesso pezzo di documento che usano in modo diverso elementi e attributi

```
<libro isbn="1324AX" titolo="On the road" />
```

```
<libro isbn="1324AX">  
  <titolo>On the road</titolo>  
</libro>
```

```
<libro>  
  <isbn>1324AX</isbn>  
  <titolo>On the road</titolo>  
</libro>
```

# A Simple XML Document

```
<Article>
```

```
  <Author>gerhard weikum</author>
```

```
  <Title>the web in ten years</title>
```

```
  <Text>
```

```
    <Abstract>in order to evolve...</Abstract>
```

```
    <Section number="1" title="introduction">
```

```
      The <index>web</index> provides the universal...
```

```
    </Section>
```

```
  </Text>
```

```
</Article>
```



# A Simple XML Document

**Freely definable tags**

`<Article>`

`<Author>gerhard weikum</author>`

`<Title>the web in ten years</title>`

`<Text>`

`<Abstract>in order to evolve...</Abstract>`

`<Section number="1" title="introduction">`

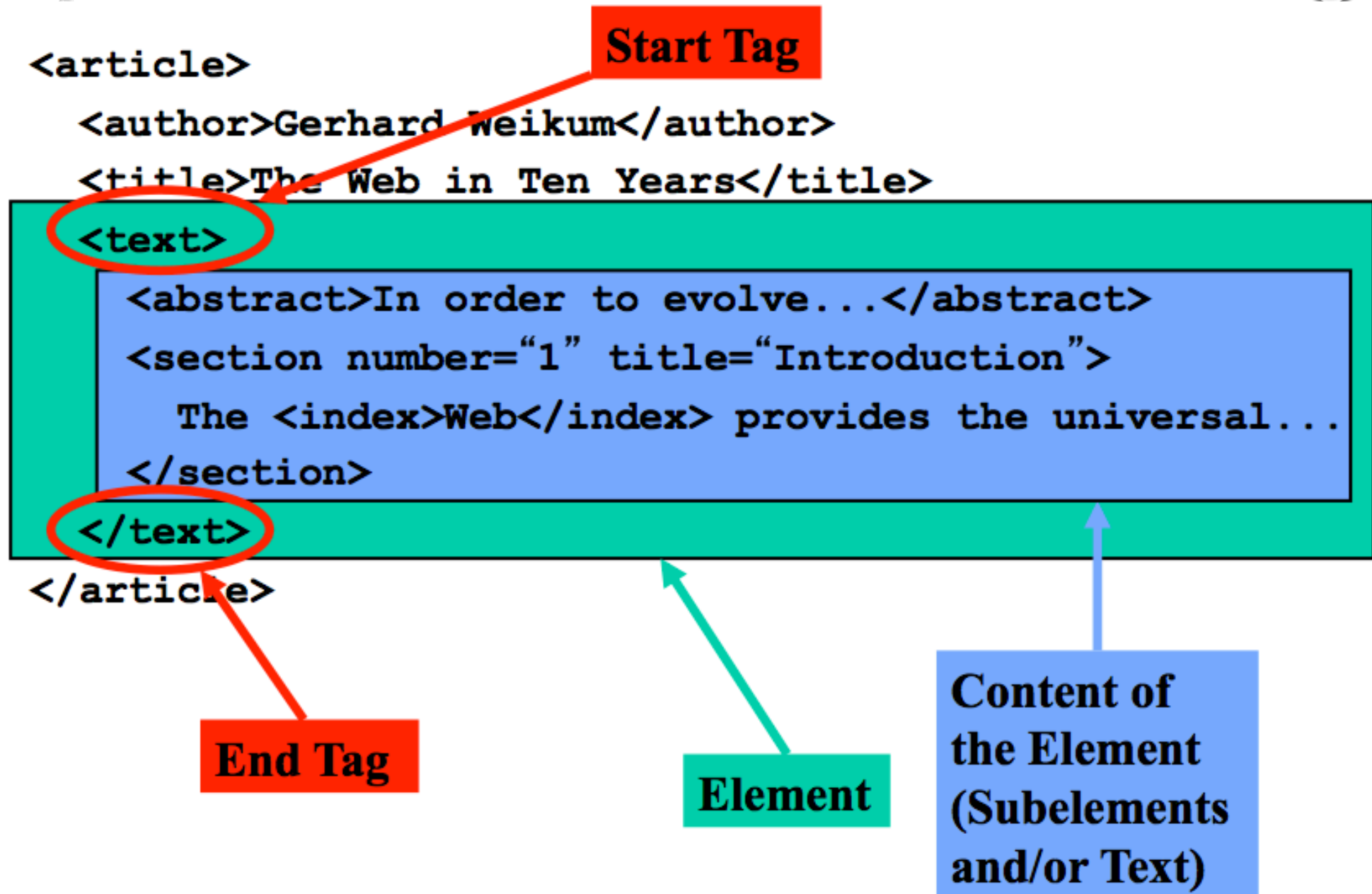
The `<index>web</index>` provides the universal...

`</Section>`

`</Text>`

`</Article>`

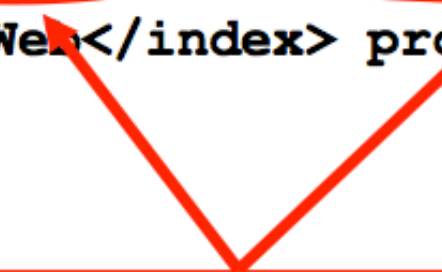
# Example of XML document



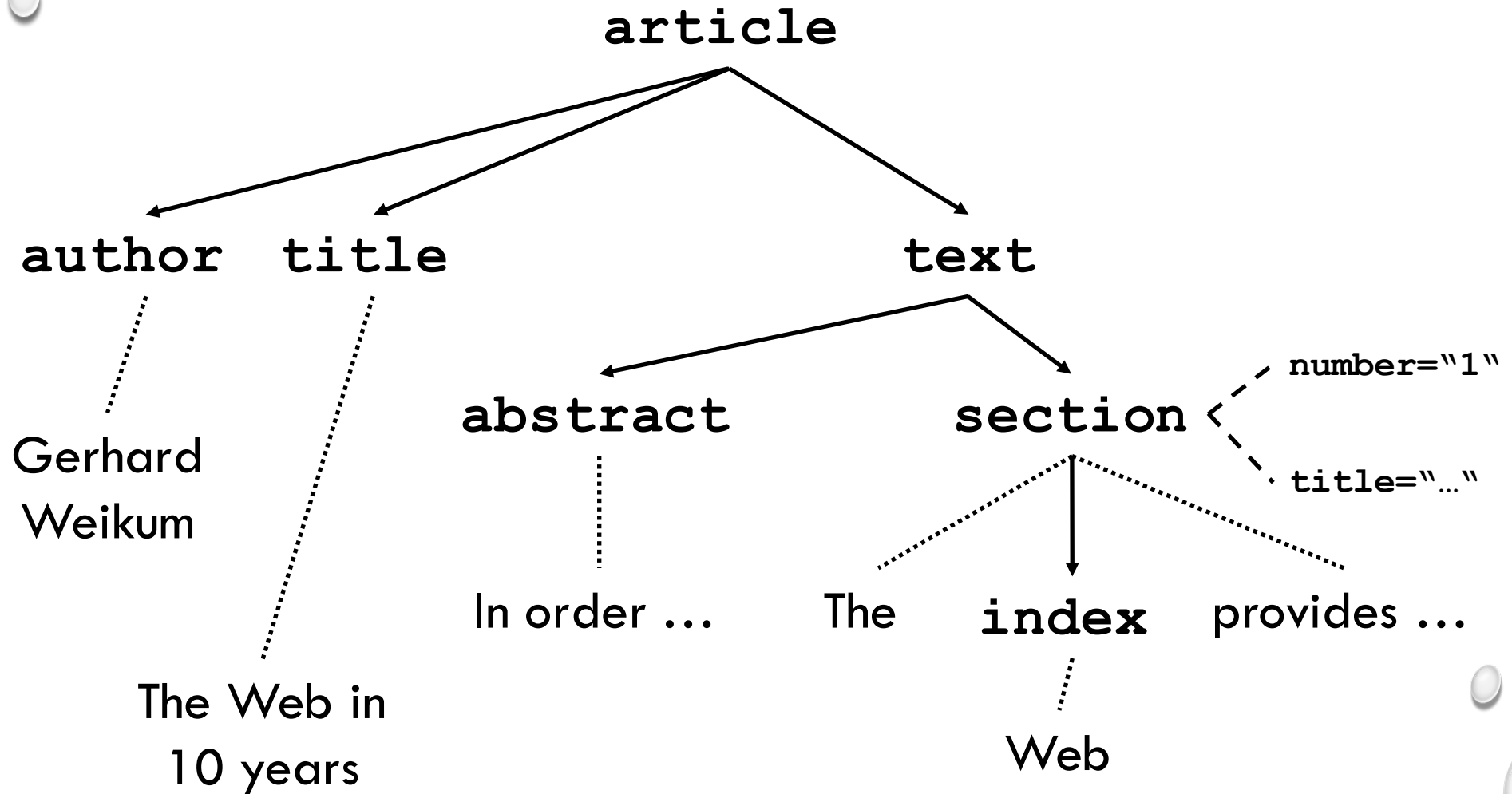
# Example of an XML document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

**Attributes with  
name and value**



# XML Documents as Ordered Trees



## Riferimenti ad entità

I **riferimenti ad entità** servono per rappresentare caratteri riservati (per esempio, **<** **>** o **&**)

Nome entità	Riferimento	Carattere
lt	&lt;	<
gt	&gt;	>
amp	&amp;	&
apos	&apos;	'
quot	&quot;	"

Oppure per rappresentare caratteri UNICODE mediante la notazione **&#XXXX**:

- **&#0189;** → **½**

## Sezione CDATA

- Per poter inserire brani di testo (porzioni di codice XML o XHTML) senza preoccuparsi di sostituire i caratteri speciali si possono utilizzare le sezioni **CDATA (Character Data)**
- Il testo contenuto in una sezione CDATA **NON viene analizzato dal parser**
- Una sezione CDATA può contenere caratteri “normalmente” proibiti
- Si utilizza la seguente sintassi:

**<![CDATA[** *Contenuto della sezione* **]]>**

- L'unica sequenza non ammessa è **]]** (chiusura)
- Esempi:

```
<E1> <![CDATA[ <<"' !] && ]]> </E1>
```

```
<E> <![CDATA[<Elemento/><A>Ciao</A>]]> </E>
```

## Conflitti sui nomi

- Capita abbastanza comunemente, soprattutto in documenti complessi, di dare nomi uguali ed elementi (o attributi) con significati diversi
- Ad esempio:

```
<libro>  
  <autore>  
    <titolo>Sir</titolo>  
    <nome>William Shakespeare</nome>  
  </autore>  
  <titolo>Romeo and Juliet</titolo>  
</libro>
```

# Esempi di conflitti sui nomi

- In XML i nomi degli elementi sono definiti dagli sviluppatori. Possono capitare dei conflitti quando si provano ad integrare documenti XML da applicazioni diverse
- Questo XML contiene informazioni di una tabella di dati (table):

```
<table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>
```

- Questo XML su di un tavolo (mobile):

```
<table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>
```



# Namespace

- Per risolvere il problema si ricorre al concetto di “**spazio dei nomi**” (**namespace**)
- Si usano **prefissi che identificano il vocabolario di appartenenza** di elementi ed attributi
- Ogni **prefisso** è associato ad un **URI** (Uniform Resource Identifier) ed è un alias per l’URI stesso
- L’URI in questione è normalmente un URL: si ha quindi la certezza di univocità
- È un meccanismo simile ai nomi lunghi delle classi in Java (i package definiscono un sistema di namespace):
  - Nome breve: **JButton**
  - Nome lungo: **javax.swing.JButton**

## Esempio di utilizzo di namespace

**Prefisso**

**Dichiarazione del prefisso  
e associazione all'URI**

**URI**

```
<lb:libro xmlns:lb="mysite.com/libri">
  <au:autore xmlns:au="mysite.com/autori">
    <au:titolo>Sir</au:titolo>
    <au:nome>William Shakespeare</au:nome>
  </au:autore>
  <lb:titolo>Romeo and Juliet</lb:titolo>
</lb:libro>
```

# Definizione di namespace

- Per **definire** un namespace si usa la seguente sintassi:  
***xmlns:NamespacePrefix="NamespaceURI"***
- La definizione è un attributo di un elemento e può essere messa ovunque all'interno del documento
- Lo **scope** del namespace è l'elemento all'interno del quale è stato dichiarato
  - Si estende a tutti i sottoelementi
  - Se si dichiara un namespace nell'elemento radice, il suo scope è l'intero documento
- L'URI può essere qualsiasi (il parser non ne controlla l'univocità) ma dovrebbe essere scelto in modo da essere effettivamente univoco

# Esempio

<DC:Docenti xmlns:DC="www.unisa.it/docenti" >

<DC:Docente codAteneo="112233">

<DC:Nome>Rita</DC:Nome>

<DC:Cognome>Francese</DC:Cognome>

<CR:Corso id="123" xmlns:CR="www.unisa.it/corsi">

<CR:Nome>Programmazione web</CR:Nome>

</CR:Corso >

<CO:Corso id="124" xmlns:CO="www.unisa.it/corsi">

<CO:Nome>Programmazione I</CO:Nome>

</CO:Corso >

</DC:Docente>

</DC:Docenti>

- **CR** e **CO** sono prefissi "collegati" allo stesso namespace
- Nel secondo elemento Corso è necessario ripetere la dichiarazione di namespace poiché ricade fuori dallo scope della prima dichiarazione
- Per evitare la seconda dichiarazione basta dichiarare il namespace in un elemento più in alto nella gerarchia

# Namespace di default

- È possibile definire un namespace di default associato al prefisso nullo
- Tutti gli elementi non qualificati da prefisso appartengono al namespace di default
- Attenzione: riduce la leggibilità di un documento

<Docenti **xmlns="www.unisa.it/docenti"**>

    <Docente codAteneo="112233">

        <Nome>Rita</Nome>

        <Cognome>Francese</Cognome>

        <**CR**:Corso id="123">

**xmlns:CR="www.unisa.it/corsi"**>

                <CR:Nome>Programmazione

web</CR:Nome>

        </CR:Corso >

    </Docente>

</Docenti>

# Ridefinizioni di prefissi

- Un **prefisso** di namespace (anche quello vuoto di default) può essere associato a diversi namespace all'interno di uno stesso documento
- È però preferibile evitare le ridefinizioni: *riducono la leggibilità del documento*

<PR:Docenti xmlns="www.unisa.it/docenti">

<Docente codAteneo="112233">

<PR:Nome>Rita</PR:Nome>

<PR:Cognome>Francese</PR:Cognome>

<PR:Corso id="123">

xmlns:PR="www.unisa.it/corsi">

<PR:Nome>Programmazione web</PR:Nome>

</PR:Corso >

</PR:Docente>

</PR:Docenti>

# Vincoli di buona formazione

- Affinché un documento XML sia **ben formato**:
  - Deve contenere una dichiarazione (XML Declaration) corretta
  - Il corpo deve avere un unico elemento radice
  - Ogni elemento deve avere un tag di apertura e uno di chiusura
    - se l'elemento è vuoto si può utilizzare la forma abbreviata (<nometag/>)
  - Gli elementi devono essere opportunamente nidificati, cioè i tag di chiusura devono seguire l'ordine inverso dei rispettivi tag di apertura
  - I nomi dei tag di apertura e chiusura devono coincidere
    - anche in termini di maiuscole e minuscole
  - I valori degli attributi devono sempre essere racchiusi tra singoli o doppi apici

# Documenti ben formati e documenti validi

- In XML ci sono **regole sintattiche**
  - come dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere **regole semantiche**
  - cosa possiamo scrivere in un documento XML
- Un documento XML che rispetta le regole sintattiche si dice **ben formato**
- Un documento XML che rispetta le regole sintattiche e le regole semantiche si dice **valido**



# Validazione: Document Type Definition (DTD)

- Un DTD è costituito da un elenco di dichiarazioni (**markup declaration**) che descrivono la struttura del documento
- Le dichiarazioni di un DTD definiscono:
  - gli elementi strutturali (**element**) di un documento mediante un identificatore generico
  - il modello di contenuto di ogni elemento (**content model**), ovvero gli elementi che contiene e le loro relazioni (un elemento può essere vuoto)
  - la lista degli **attributi** associati a ciascun elemento e il loro tipo