



UNIVERSITÀ
DEGLI STUDI
DI PALERMO



Introduzione a XML

CORSO DI
PROGRAMMAZIONE WEB E MOBILE
a.a. 2021/2022

Prof. Roberto Pirrone

Sommario

- Origini e storia di XML
- La sintassi di XML
- Document Object Model
- Document Type Definition (DTD)
- Alcuni esempi di marcatura

Linguaggi di marcatura

- Un **linguaggio di marcatura** è un insieme di convenzioni per la marcatura di testi
- **Marcatura di documenti**
 - la marcatura (o etichettatura) permette di rendere **esplicita** un'interpretazione di un testo.
 - storicamente
 - annotazioni in un testo che descrivono al tipografo come stampare o comporre una parte del testo
 - oggi
 - **qualsiasi tipo di codice inserito in un testo in forma elettronica**

Tipi di marcatura

- Due tipi di marcatura
 - **marcatura procedurale**
 - descrive come processare il documento
 - postscript, rtf, ecc.

```
{\rtf1\ansi\ansicpg1252\uc1  
\deff0\deflang1040\deflangfe1040{\fonttbl{\f0\froman\fcharset0\prq2{\*\p  
anose 02020603050405020304}Times New Roman
```

- **marcatura descrittiva**
 - descrive la **struttura logica** del documento
 - HTML, SGML, XML

SGML - Standard Generalized Markup Language

- Il padre degli attuali linguaggi di marcatura
- È un **metalinguaggio di marcatura**, che permette di definire linguaggi di marcatura
 - estremamente espressivo e configurabile
 - l'alta espressività rende l'elaborazione automatica del testo complicata
 - utilizzato per grandi progetti di digitalizzazione del testo
 - non studiato espressamente per il Web

SGML - Standard Generalized Markup Language

- Manca di alcune caratteristiche fondamentali per il Web:
 - gestione dei link
 - gestione del conflitto sui nomi delle etichette
 - tutti i documenti devono essere *validi* oltre a essere *ben formati*
- È troppo complicato per poter essere adoperato come linguaggio di marcatura

HTML - HyperText Markup Language

- È un **linguaggio di marcatura** (non un metalinguaggio!)
 - definito in *direttamente in termini di SGML*
 - Insieme di etichette prefissato
- La marcatura non denota il “significato”, ovvero la struttura “logica” di un documento, *ma solo il suo formato*

HTML - HyperText Markup Language

- Studiato espressamente per il Web
 - collegamenti ipertestuali
 - immagini
 - marcatura finalizzata alla presentazione del documento come pagina Web
 - diversi tipi di titoli, tabelle, ecc.
- non c'è un legame tra marcatura e tipi di informazione rappresentati nel documento

Una pagina HTML

```
<html>
  <head>
    <title>Le avventure di Pinocchio</title>
  </head>
  <body>
    <h2>Carlo Collodi</h2>
    <h1>Le avventure di Pinocchio</h1>
    <p></p>
    <p>Capitolo I</p>
    <p><i>Come andò che Maestro
      Ciliegia, falegname, trovò un pezzo di legno, che piangeva e rideva come un
      bambino.
    </i></p>
  </body>
</html>
```

XML

eXtensible Mark-up Language

- La marcatura è dettata dalla **struttura logica** del documento
- L'insieme di etichette può cambiare in base l'applicazione
- Fondamentale il concetto di tipo di documento
 - specificato attraverso una **Document Type Definition** o **DTD** (parte dello standard XML)
 - permette di dichiarare la struttura che tutti i documenti di un certo tipo devono rispettare

XML - eXtensible Mark-up Language

- Naturale successore di HTML come linguaggio per il Web
 - più espressivo e flessibile
 - la visualizzazione del documento è **indipendente dalla sua organizzazione logica**
 - per lo stesso documento XML è possibile definire più modalità di visualizzazione attraverso fogli di stile *XSL – XML Stylesheet Language*

Esempio di documento XML

```
<libro>
  <intestazione>
    Le avventure di Pinocchio
  </intestazione>

  <autore>Carlo Collodi</autore>

  <titolo>Le avventure di Pinocchio</titolo>
  <capitolo>
    <intestazioneCapitolo>Capitolo I</intestazioneCapitolo>
    <titolo>Come andò che Maestro
      Ciliegia, falegname, trovò un pezzo di legno, che piangeva e rideva come un
      bambino.</titolo>
  </capitolo>
</libro>
```

Le origini di XML e HTML

- 1969
 - Charles Goldfarb (IBM) dirige lo sviluppo di GML
- 1974
 - Charles Goldfarb inventa SGML, il padre dei linguaggi di marcatura
- 1986
 - SGML diventa uno standard ISO (ISO 8879 ``Information Processing - Text and Office Systems - Standard Generalized Markup Language")
- 1989
 - Tim-Berners Lee (CERN di Ginevra) inventa HTML
- 1995
 - Fondazione del World Wide Web Consortium (W3C)

Le origini di XML e HTML

- 1995
 - HTML 2.0 diventa una raccomandazione del W3C
- 1996
 - Inizio dello sviluppo di XML presso il W3C
- 1997
 - HTML 3.2 diventa una raccomandazione del W3C
- 1998
 - XML 1.0 diventa una raccomandazione W3C (uno standard di fatto)
- 1999
 - HTML 4.01 diventa una raccomandazione W3C (*Strict* / *Transitional* / *Frameset*)

Le origini di XML e HTML

- 2000
 - *XHTML 1.0* diventa una raccomandazione W3C
- 2001
 - *XHTML 1.1* diventa una raccomandazione W3C
- 2002
 - XML 1.1 diventa una raccomandazione candidata W3C

WHATWG

- Web Hypertext Application Technology Working Group
 - Comunità di sviluppatori che curano lo sviluppo delle tecnologie web
- Fondato da Apple, Mozilla Foundation, Opera Software nel 2004 in contrapposizione alla linea di sviluppo XML based adottata dal W3C
- Ha generato lo standard HTML5, recepito poi dal W3C

Le origini di XML e HTML

- 2012
 - Il WHATWG si separa dal W3C e inizia a sviluppare lo *HTML Living Standard*, di fatto il nuovo HTML5.
- 2014
 - *HTML 5*, snapshot del Living Standard, diventa una raccomandazione candidata W3C
- 2016
 - *HTML 5.1* diventa una raccomandazione candidata W3C

I tratti caratterizzanti di XML

- **Marcatura dichiarativa**

- usa etichette di marcatura che indicano la **funzione astratta** della porzione di testo a cui si riferiscono

- **Marcatura strutturata**

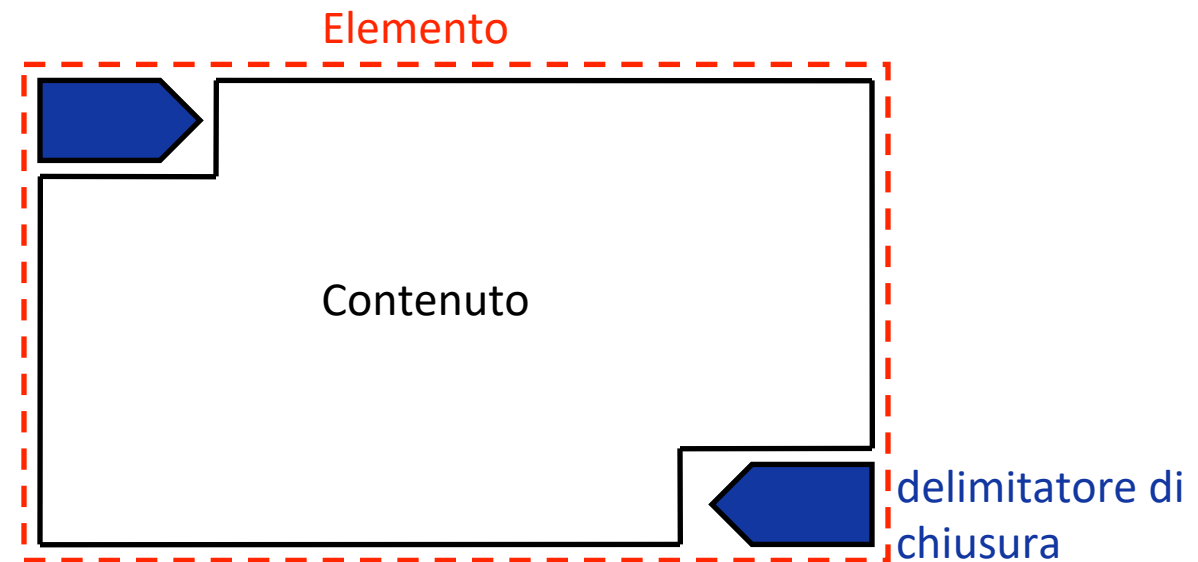
- permette di raggruppare porzioni del testo e di definirle come **unità strutturali complesse**, che riflettono l'organizzazione interna del testo

- **Marcatura gerarchica**

- le strutture identificate nel testo possono combinarsi in maniera **gerarchica**
 - un'unità strutturale del testo può a sua volta contenere altre strutture annidate
 - otteniamo una gerarchia di strutture definibili a livelli incrementali di dettaglio

I componenti della marcatura XML gli elementi

- Gli elementi rappresentano i blocchi costitutivi in cui si articola un testo
- Ogni elemento viene marcato in modo esplicito nel testo inserendo un delimitatore di apertura all'inizio dell'elemento e uno di chiusura alla fine
 - Es: **<autore>**Carlo Collodi**</autore>**



I componenti della marcatura XML

i nomi degli elementi

- Ogni tipo di elemento è identificato da un **nome** (**etichetta** o **tag**)
 - il nome associato a ogni tipo di elemento è chiamato identificatore generico (generic identifier o GI)
- XML è **case-sensitive**
 - l'identificatore generico deve essere sempre specificato con lo stesso tipo di carattere, maiuscolo o minuscolo:
`<tag>...</tag>`, `<TAG>...</TAG>`, `<Tag>...</Tag>`

`<tag>...</TAG>` ***errato***

I componenti della marcatura XML

i nomi degli elementi

- Norme per la sintassi del nome degli elementi
 - possono contenere solo lettere, cifre, ., -, _
 - possono iniziare solo con una lettera o con _

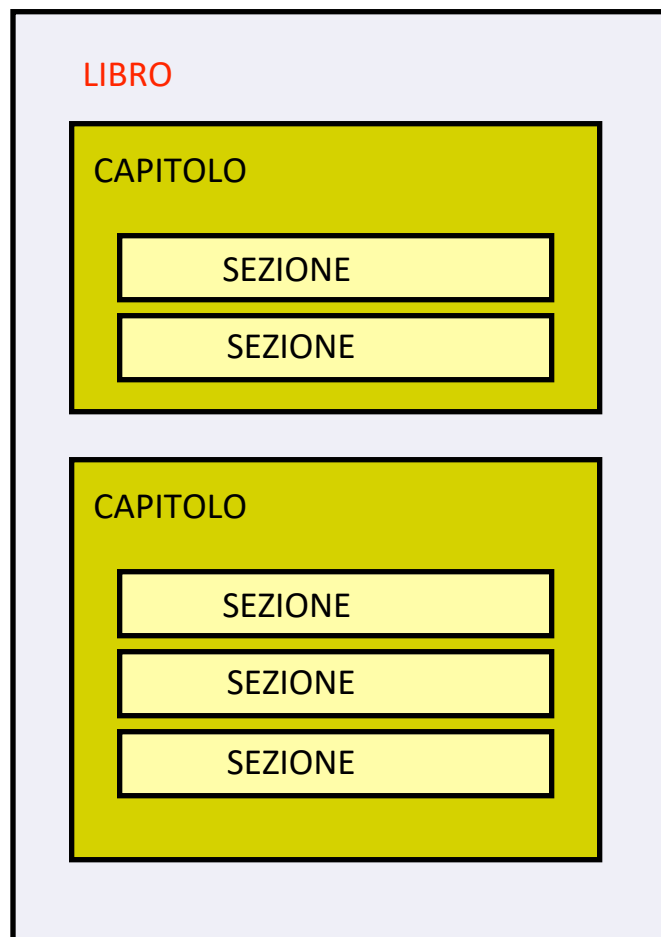
nomi consentiti: `<autore.libro>`, `<_autore>`, `<AUTORE-LIBRO>`, `<autore_1>`

nomi proibiti: `<1autore>`, `<autore libro>`, `<autore;@?libro>`

- non esiste un limite di lunghezza per il nome di un elemento

I componenti della marcatura XML

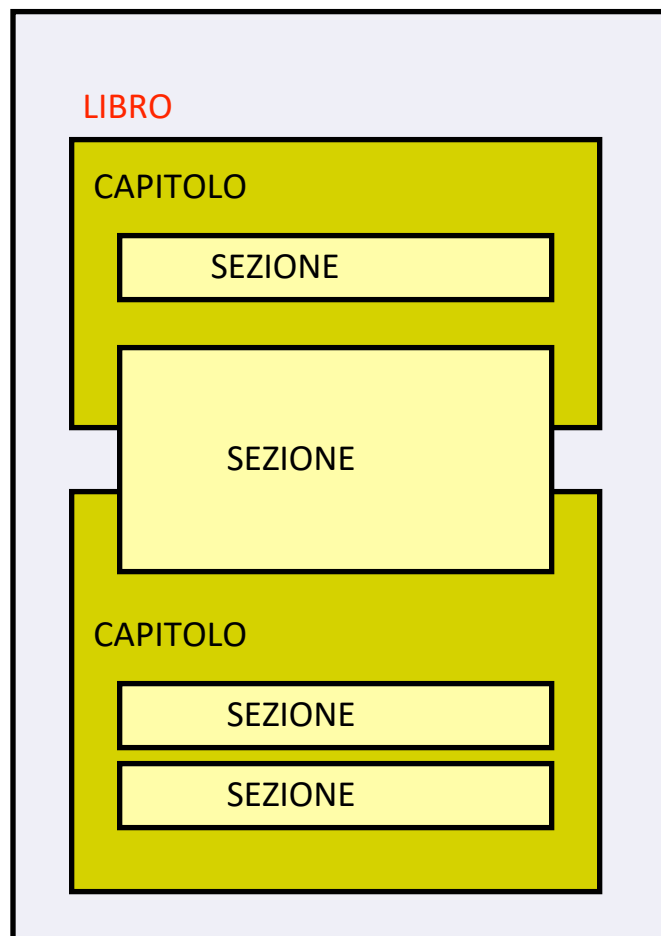
relazioni tra elementi



- due elementi XML possono essere annidati l'uno nell'altro
- l'elemento più esterno è detto elemento genitore, quello interno elemento figlio
- questo meccanismo di annidamento degli elementi permette la rappresentazione di strutture gerarchiche di profondità variabile

I componenti della marcatura XML

relazioni tra elementi



- struttura XML mal formata in quanto esiste un elemento “a cavallo” di due elementi (annidamento improprio)
- in XML non è consentita la sovrapposizione tra elementi
- un elemento figlio deve essere completamente incluso nell’elemento padre

I componenti della marcatura XML

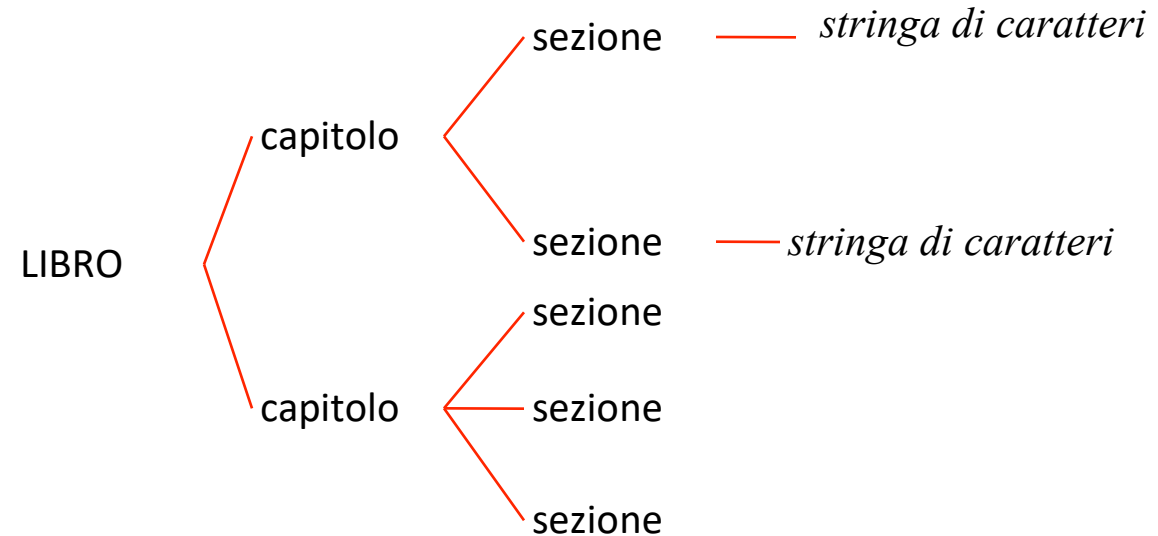
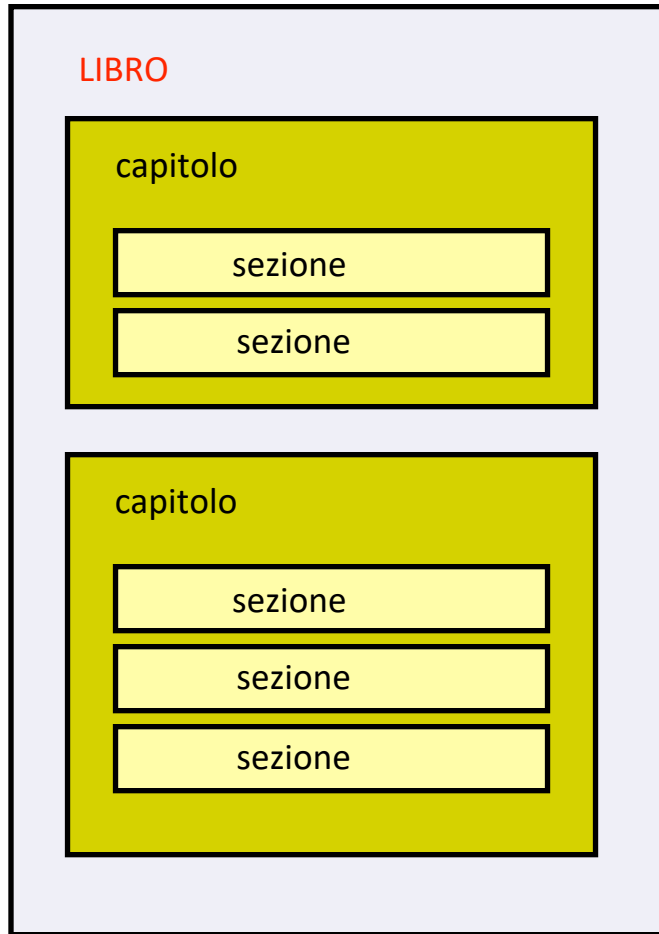
l'elemento radice



- ogni documento XML ben formato deve contenere un elemento che contiene tutti gli altri elementi (elemento radice)
- la figura rappresenta una struttura mal formata in quanto in XML non è possibile avere più elementi a livello di radice
- ogni documento XML deve contenere uno e uno solo elemento radice

I componenti della marcatura XML

il documento XML come albero



DOM

- Document Object Model
- Standard del W3C
- <http://www.w3.org/DOM/>
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

DOM

- Secondo il DOM, ogni cosa è un nodo
- Il DOM dice che:
 - L'intero documento è un nodo documento
 - Ciascun tag XML è un nodo elemento
 - I testi contenuti negli elementi XML sono nodi testo
 - Ogni attributo XML è un nodo attributo
 - I commenti sono nodi commento

DOM

- I nodi hanno tra di loro una relazione gerarchica *dettata dall'inclusione tra gli elementi nel testo del documento*
- Tutti i nodi di un documento XML formano l'albero del documento.
 - Ciascun elemento, attributo, testo, etc, di un documento XML rappresenta un nodo dell'albero.
 - L'albero comincia con il nodo documento e continua ad estendersi fino a quando si raggiungono tutti i nodi testo al livello più basso dell'albero.

DOM

- Alcuni nodi possono avere nodi figli, mentre altri possono non averne (nodi foglia).
- Poiché i dati XML sono strutturati a forma di albero, essi possono essere attraversati senza conoscere l'esatta struttura dell'albero o il tipo dei dati contenuti
 - Relazione padre-figlio
 - Relazione di *fratellanza* tra i nodi con lo stesso genitore

W3C DOM

- Descrive una API standardizzata per accedere e manipolare i documenti HTML e XML
 - Oggetto **document**
 - Classi **Node** e **Element**
- È diviso in tre parti:
 - **Core DOM**, che definisce un insieme standard di oggetti per qualunque documento strutturato
 - **XML DOM**, che definisce un insieme standard di oggetti per i documenti XML
 - **HTML DOM**, che definisce un insieme standard di oggetti per i documenti

HTML



UNIVERSITÀ
DEGLI STUDI
DI PALERMO



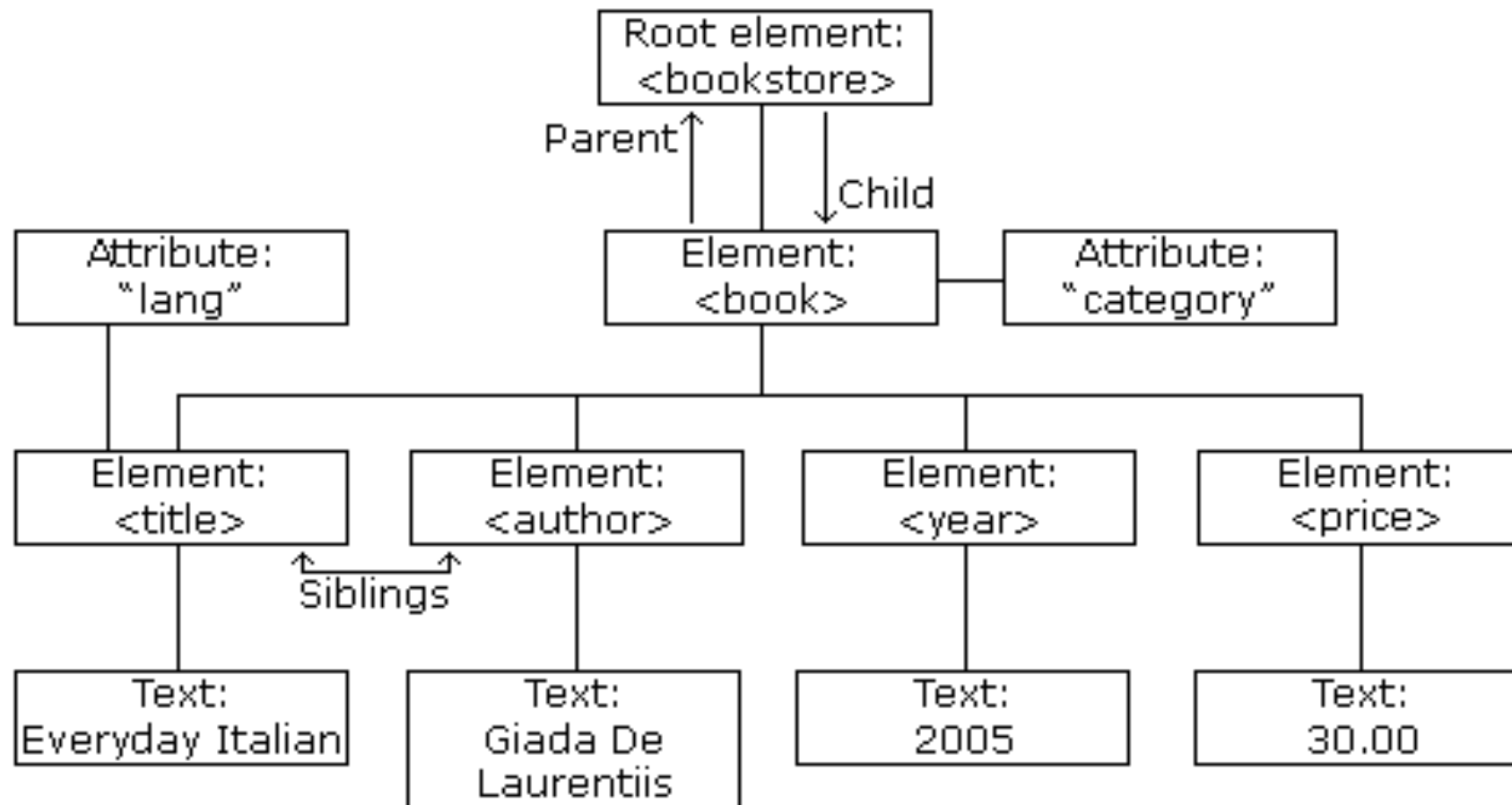
W3C DOM

- Esistono quattro livelli di specifica che definiscono gli oggetti della API con le relative interfacce
 - DOM level 1: specifiche base per Core, XML e HTML
 - DOM level 2: specifiche relative alla gestione degli eventi, dei *namespace XML*, l'accesso dinamico e l'attraversamento dell'albero nonché l'introduzione del metodo **getElementById()**
 - DOM level 3: specifiche relative a validazione, serializzazione e visita del documento usando *XPath*
 - DOM level 4: emesso nel 2015, "living standard" del **WHATWG**

DOM Esempio: bookstore.xml

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```


DOM Esemplio: bookstore.xml



I componenti della marcatura XML

processing instructions

- I dati contenuti in una istruzione di elaborazione vengono passate all'applicazione che usa il documento XML
- Solo delimitate dai caratteri `<?` e `?>`
- Sono formate da un *target* e da un *valore*

```
<?xml-stylesheet type="text/xsl" href="usage.xsl"?>
```

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

I componenti della marcatura XML

Intestazione del documento

- Tutti i documenti XML devono contenere una dichiarazione nel primo rigo

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

```
<myMessage>
```

```
    <message>hello</message>
```

```
</myMessage>
```

I componenti della marcatura XML

Intestazione del documento

- Anche l'intestazione è un esempio di Processing Instruction
- Essa indica la versione di XML in cui è stato scritto il documento
- A volte indica anche se il documento è da ritenersi come una risorsa a sé stante o se legata ad altre

I componenti della marcatura XML

contenuto di un elemento

- Il contenuto di un elemento può essere costituito da:
- testo libero non contenente altri elementi (dati di tipo carattere)
`<titolo>Le avventure di Pinocchio</titolo>`

- altri elementi (figli)

`<capitolo>`

`<titolo>Capitolo primo</titolo>`

`<capoverso>C'era una volta...</capoverso>`

`<capoverso>- Un re! - diranno subito i miei piccoli lettori.</capoverso>`

`</capitolo>`

I componenti della marcatura XML

contenuto di un elemento

- contenuto misto (elementi+ testo)

```
<titolo>Le avventure di Pinocchio  
  <sottotitolo>Storia di un burattino</sottotitolo>  
</titolo>
```
- Il contenuto di un elemento può essere “vuoto”
 - due modi di denotare un elemento vuoto:
 - coppia di delimitatori di apertura e chiusura

```
<salto_pagina></salto_pagina>
```
 - etichetta di elemento vuoto

```
<salto_pagina/>
```

I componenti della marcatura XML

gli attributi

- Gli elementi XML possono essere dotati di uno o più attributi
 - gli attributi rappresentano informazioni aggiuntive che specificano alcune caratteristiche dell'elemento (ma che non fanno parte del contenuto del testo)

`nome_attributo= "valore"`

`<capoverso num="1">C'era una volta...</capoverso>`

I componenti della marcatura XML

gli attributi

- Nomi degli attributi
 - stesse restrizioni definite per i nomi degli elementi
- i valori degli attributi devono sempre essere racchiusi tra virgolette (singole o doppie)
 - nel caso in cui un valore contenga al suo interno delle virgolette, allora diventa obbligatorio differenziarle da quelle più esterne
- un attributo può ricorrere al massimo una volta all'interno di un elemento
 - ma ci possono essere più attributi differenti
- Gli attributi possono comparire solo nei tag di apertura degli elementi

I componenti della marcatura XML

elementi vs. attributi

Elemento = “contenitore” e “classificatore” del dato testuale
Attributo = “glossa” associata al dato testuale

```
<parola pos="nome"  
  num="sing"  
  gen="masc"  
  lemma="legno">  
  <orto>legno</orto>  
</parola>
```

```
<parola>  
  <orto>legno</orto>  
  <pos>nome</pos>  
  <accordo>  
    <num>sing</num>  
    <gen>masc</gen>  
  </accordo>  
</parola>
```

I componenti della marcatura XML

elementi vs. attributi

- Non è sempre facile stabilire quando preferire una codifica in termini di elementi o di attributi
- Spesso è una questione di “stile di codifica”
- Elementi e attributi hanno delle differenze espressive che possono o meno avere rilevanza nella definizione della nostra marcatura.

I componenti della marcatura XML

elementi vs. attributi

Elementi	Attributi
Possono ricorrere <i>più volte</i> in un documento	Ricorrono <i>al massimo una volta</i> in un elemento
E' possibile specificare <i>l'ordine</i> degli elementi nel documento	Non è possibile stabilire l'ordine degli attributi
Un elemento può descrivere strutture complesse perché può contenere altri elementi	il valore di un attributo XML è semplicemente una stringa di caratteri

I componenti della marcatura XML

commenti

- Ogni documento XML può contenere uno o più commenti
 - Sono ignorati dalle eventuali applicazioni che processino il documento
 - Possono apparire in qualunque punto all'interno del testo con le seguenti eccezioni:
 - Non possono apparire all'interno di un delimitatore di apertura o di chiusura di un elemento
 - Non possono apparire all'interno di un commento
- <!-- questo è un commento XML -->**

I componenti della marcatura XML

namespace

- Chiunque può definire i propri tag
 - Conflitti di nomi
 - Posso definire dei namespace
 - Il tag assume la forma **<prefissonamespace:tag>**

```
<image:images xmlns:image="urn:deitel:imageInfo">  
  <image:file filename="bunny.jpg" />  
</image:images>
```

Caratteri e XML

- Tutti i file XML sono per default file di testo secondo la codifica Unicode UTF-8
 - indipendenza dei dati da piattaforme e totale interscambiabilità
 - è possibile specificare una codifica di caratteri diversa (nella Dichiarazione XML)

Caratteri e XML

- È possibile rappresentare qualsiasi carattere Unicode in un file XML con un riferimento a carattere:
 - `&#<codice decimale Unicode>;`
 - `&#x<codice esadecimale Unicode>;`

`"è" è è`

`"и" ш ш`

`<nome>РоссиИя</nome>`

`<nome>Россия</nome>`

XML - riferimenti a carattere

- Scelta raccomandata per qualsiasi carattere non ASCII Standard
 - `<frase> Pisa è una città </frase>`
 - `<frase> Pisa è una città </frase>`

à	à
é	é
è	è
ì	ì
ò	ò
ù	ù

Correttezza di un documento XML

- **Parsing** (o **analisi sintattica**)
 - il processo di analisi delle sequenze di token per determinarne la struttura grammaticale in relazione ad una data grammatica formale
 - Documento XML **ben formato** (well-formed) → sintatticamente corretto
- Parser non-validating → esegue **solo** il controllo sintattico per verificare se il documento è ben formato

Correttezza di un documento XML

- La correttezza del documento riguarda anche l'uso dei corretti *nomi di tag e attributi* nonché delle corrette *relazioni di contenimento tra i tag* e dei *valori degli attributi*
- La specifica di questo tipo di correttezza si ottiene attraverso la **Document Type Definition (DTD)**
- Parser validating: leggono la DTD e stabiliscono se un documento XML è ad essa conforme o meno
 - Documento **valido**: sintatticamente corretto e conforme alle specifiche DTD

Document Type Definition (DTD)

- Una Document Type Definition si può trovare nel prologo del documento XML

```
<!DOCTYPE prova [  
  <!ELEMENT miotag (#PCDATA) >  
>
```

- Gli elementi sono i componenti base di un documento XML
- PCDATA indica che il tag **miotag** racchiude del testo

Document Type Definition (DTD)

- Una Document Type Definition può far riferimento a dichiarazioni esterne al documento XML

```
<!DOCTYPE prova SYSTEM "myDTD.dtd" [  
  <!ELEMENT miotag (#PCDATA) >  
>
```

Document Type Definition (DTD)

- Una Document Type Definition può far riferimento a dichiarazioni esterne al documento XML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

Nome: *prefisso*//*proprietario*//*nome DTD*//*lingua*

prefisso: **ISO** (standard), + (approvato ISO), – (non approvato ISO)

URI di riferimento

Document Type Definition (DTD)

```
<!ELEMENT persons (person) >
```

```
<!ELEMENT person (#PCDATA) >
```

In questo caso un tag **persons** contiene un tag **person**, che a sua volta contiene del testo

```
< persons>
```

```
  < person>John Smith</person>
```

```
</persons>
```

Document Type Definition (DTD)

```
<!ELEMENT class (teacher, student) >  
<!ELEMENT teacher (#PCDATA) >  
<!ELEMENT student (#PCDATA) >
```

Il tag **class** contiene un tag **teacher** e un tag **student**. Questi due tag contengono testo

```
<class>  
  <teacher>prof</teacher>  
  <student>John</student>  
</class>
```

Document Type Definition (DTD)

```
<!ELEMENT dessert (glace|fruit) >  
<!ELEMENT glace (#PCDATA) >  
<!ELEMENT fruit (#PCDATA) >
```

Il tag **dessert** contiene un tag **glace** o un tag **fruit**. Questi due tag contengono testo

```
<dessert>  
  <fruit>peach</fruit>  
</dessert>
```

oppure

```
<dessert>  
  <glace>chocolate</glace>  
</dessert>
```


Document Type Definition (DTD)

```
<!ELEMENT compilation (song+) >
```

```
<!ELEMENT song (#PCDATA) >
```

Il tag `compilation` contiene **uno o più** tag `song`

```
<compilation>
```

```
  <song>Malo</song>
```

```
  <song>La Flaca</song>
```

```
  ...
```

```
</compilation>
```

Document Type Definition (DTD)

```
<!ELEMENT compilation (song*) >  
<!ELEMENT song (#PCDATA) >
```

Il tag `compilation` contiene **zero o più** tag `song`

```
<compilation >  
  <song>Malo</song >  
  <song>La Flaca</song >
```

...

```
</compilation>
```

oppure

```
<compilation/>
```

Document Type Definition (DTD)

```
<!ELEMENT compilation (song?) >  
<!ELEMENT song (#PCDATA) >
```

Il tag `compilation` contiene **zero o un** tag `song`

```
<compilation >  
  <song>Malo</song >  
</compilation>
```

oppure

```
<compilation/>
```

Document Type Definition (DTD)

- il contenuto di un elemento può essere di due tipo:
 - Vuoto (**EMPTY**)
 - Misto (**ANY**)
 - Solo PCDATA
 - solo altri elementi
 - PCDATA e altri elementi insieme

<!ELEMENT vuoto EMPTY>

<vuoto/>

Document Type Definition (DTD)

- A ciascun tag possono essere associati degli attributi
 - Inizialmente dichiaro un elemento
 - Quindi definisco gli attributi x e y

```
<!ELEMENT posizione EMPTY>  
<!ATTLIST posizione x CDATA #REQUIRED>  
<!ATTLIST posizione y CDATA #REQUIRED>
```

- **#REQUIRED** indica che l'attributo è obbligatorio
- **CDATA** indica che il valore dell'attributo può contenere qualunque carattere

```
<posizione x="1" y="2" />
```

Document Type Definition (DTD)

- Il valore di attributo può essere
 - **#IMPLIED** (non obbligatorio)
 - **#FIXED** (valore fissato)

```
<!ELEMENT indirizzo (via)>
```

```
<!ELEMENT via (#PCDATA)>
```

```
<!ATTLIST indirizzo cap #FIXED "90100">
```

```
<indirizzo cap="90100">
```

```
  <via>Via Libertà</via>
```

```
</indirizzo>
```

Document Type Definition (DTD)

- Gli attributi possono essere divisi per tipo:
 - Stringhe (CDATA)
 - Enumerati
 - Token

Document Type Definition (DTD)

- Gli attributi enumerati possono assumere solo uno dei valori elencati in una lista e separati da “|”

```
<!ELEMENT persona EMPTY>
```

```
<!ATTLIST persona sesso (M|F) "F">
```

In questo caso l'attributo può assumere **solo** il valore **M** o **F**.
F è il valore di default

```
<persona sesso="M" />
```

oppure

```
<persona /> → sesso vale automaticamente F
```


Document Type Definition (DTD)

- Gli attributi enumerati possono assumere solo uno dei valori elencati in una lista e separati da “|”

```
<!ATTLIST payment method (cash|credit|debit|paypal)  
#IMPLIED>
```

```
<payment>300.00</payment>
```

```
<payment method="paypal">250.00</payment>
```

XML Non valido:

```
<payment method="euro">150.00</payment>
```

Document Type Definition (DTD)

- I token si dividono in:
 - **ID** (identificatore unico)
 - **IDREF** (riferimento ad identificatore unico)
 - **IDREFS** (lista di riferimenti ad identificatori separati da spazio)
 - **ENTITY** (può assumere come valore un'entità)
 - **ENTITIES** (può assumere come valore una lista di entità separate da spazio)
 - **NMTOKEN** (nome XML valido, secondo la sintassi dei nomi in XML)
 - **NMTOKENS** (lista di nomi XML validi separati da spazio)
 - **NOTATION** (DTD NOTATION – riferimento a entità esterna *non XML*)
 - **xml:lang xml:space** (attributi predefiniti XML)

Document Type Definition (DTD)

```
<!ELEMENT catalogo (prodotto+, prezzo+)>
```

```
<!ELEMENT prodotto (#PCDATA)>
```

```
<!ELEMENT prezzo (#PCDATA)>
```

```
<!ATTLIST prodotto id ID #REQUIRED>
```

```
<!ATTLIST prezzo idref IDREF #REQUIRED>
```

```
<catalogo>
```

```
  <prodotto id="001">libro</prodotto>
```

```
  <prodotto id="002">penna</prodotto>
```

```
  ...
```

```
  <prezzo idref="001">15</prezzo>
```

```
  <prezzo idref="002">10</prezzo>
```

```
  ...
```

```
</catalogo>
```

Document Type Definition (DTD)

```
<?xml version="1.0"?>
<!DOCTYPE student_name [
<!ELEMENT student_name (#PCDATA)>
<!ATTLIST student_name student_no NMTOKEN #REQUIRED>
]>
<student_name student_no="9216735">
Jo Smith
</student_name>
```

Document Type Definition (DTD)

```
<?xml version="1.0"?>  
<!DOCTYPE secureDocument [  
  <!ELEMENT secureDocument EMPTY>  
  <!ATTLIST secureDocument authorizedUsers NMTOKENS  
    #REQUIRED>  
  
<secureDocument authorizedUsers="James.Bond M  
Miss.MoneyPenny" />
```

Document Type Definition (DTD)

```
<?xml version="1.0" standalone="yes"?>  
<!DOCTYPE document [  
  <!ELEMENT document (description,code)>  
  <!ELEMENT description (#PCDATA)>  
  <!ATTLIST description xml:lang NMTOKEN #FIXED "en">  
  <!ELEMENT code (#PCDATA)>  
  <!ATTLIST code xml:space (default|preserve) "preserve">  
>
```

(1/2)

Document Type Definition (DTD)

```
<document>
```

```
<description xml:lang="en">
```

The following section of code displays the menu of user choices and gets the user's request.

```
</description>
```

```
<code>
```

```
do    {  
    do    {  
        disp_menu();  
        scanf(" %d", &ans);  
    } while ((ans<1) || (ans>3));
```

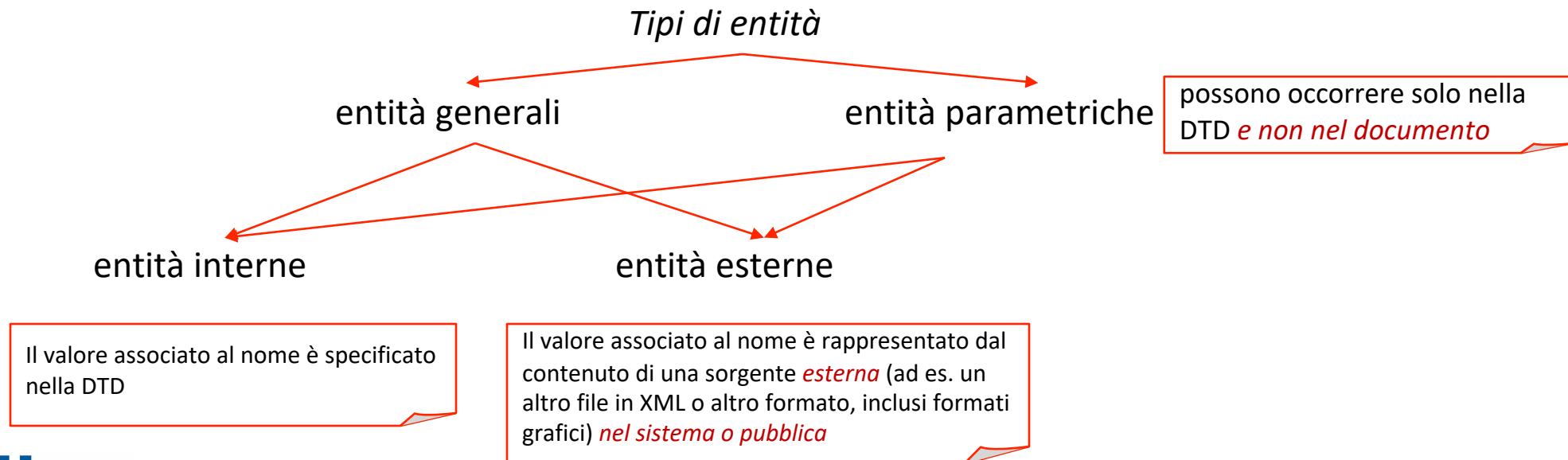
```
</code>
```

```
</document>
```

(2/2)

Entità e riferimenti a entità

- Le **entità** sono sequenze arbitrarie di byte (che vanno da una stringa di caratteri a un file intero) associate a nomi mnemonici
 - i riferimenti a entità usano questi nomi nei documenti XML come “segnaposto” del contenuto effettivo (valore) dell’entità



Entità generali – entità interne

- I riferimenti a entità generali hanno la forma **&nome_entità;**
- Sono da considerarsi entità di fatto anche i riferimenti ai caratteri usando direttamente il valore del punto di codice UTF-8 decimale o esadecimale
- Le entità interne sono associazioni tra un nome convenzionale e un frammento di testo, definite nella DTD e sostituite nel corpo del documento:

<!ENTITY JFK "John F. Kennedy">

<nome>&JFK;</nome> → <nome>John F. Kennedy</nome>

Entità generali – entità interne predefinite

- Riferimenti a entità predefinite
 - corrispondono a caratteri riservati di XML e devono essere sempre usati quando questi compaiono nel testo da codificare (e non come segni di marcatura)
 - non devono essere dichiarati nella DTD

" → **"**

& → **&**

' → **'**

< → **<**

> → **>**

Entità generali – entità esterne

- Possono essere **SYSTEM** (private, per gruppi di autori) o **PUBLIC**

```
<?xml version="1.0" standalone="no" ?>  
<!DOCTYPE copyright [  
  <!ELEMENT copyright (#PCDATA)>  
  <!ENTITY c SYSTEM  
    "http://www.xmlwriter.net/copyright.xml">  
]>  
<copyright>&c;</copyright>
```

Entità generali – entità esterne

- Possono essere **SYSTEM** (private, per gruppi di autori) o **PUBLIC**

```
<?xml version="1.0" standalone="no" ?>  
<!DOCTYPE copyright [  
  <!ELEMENT copyright (#PCDATA)>  
  <!ENTITY c PUBLIC "-//W3C//TEXT copyright//EN"  
    "http://www.w3.org/xmlspec/copyright.xml">  
]>  
<copyright>&c;</copyright>
```

Entità parametriche

- Sono dichiarate con % e vengono usate solo all'interno della DTD

```
<!ENTITY % p "(#PCDATA)">
```

```
<!ELEMENT student (id,surname,firstname,dob,(subject)*)>
```

```
<!ELEMENT id %p;>
```

```
<!ELEMENT surname %p;>
```

```
<!ELEMENT firstname %p;>
```

```
<!ELEMENT dob %p;>
```

```
<!ELEMENT subject %p;>
```

Notation e entità “unparsed”

- Una **NOTATION** definisce il formato dati di una *entità unparsed* cioè una entità che fa riferimento a *dati non XML*
- Anche in questo caso si possono avere entità unparsed interne ed esterne
- Anche le **NOTATION** possono essere **SYSTEM** e **PUBLIC**

Notation e entità “unparsed”

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE img [
  <!NOTATION jpg PUBLIC "JPG 1.0">
  <!NOTATION gif PUBLIC "GIF 1.0" "image/gif">
  <!NOTATION png SYSTEM "image/png">
  <!ENTITY companyLogo SYSTEM "http://www.liquid-
technologies.com/Content/images/liquid-logo.png" NDATA png>
  <!ELEMENT img EMPTY>
  <!ATTLIST img src ENTITY #REQUIRED>
]>

```

Linguaggi di markup personalizzati

- **Mathematical Markup Language (MathML)**
 - Sviluppato dal W3C per descrivere espressioni matematiche usando la sintassi XML

<math>

<msqrt>

<msup>

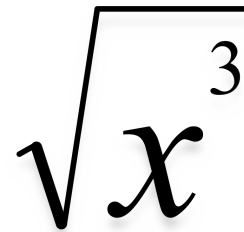
<mi>x</mi>

<mn>3</mn>

</msup>

</msqrt>

</math>


$$\sqrt{x^3}$$

Linguaggi di markup personalizzati

- **Chemical Markup Language (CML)**
 - Usato per rappresentare strutture chimiche e molecolari

```
<C:molecule id="ammoniaca">  
  <C:atomArray builtin="elsym">  
    N H H H  
  </C:atomArray>  
  ...  
</C:molecule>
```

Linguaggi di markup personalizzati

- **Geography Markup Language (GML)**
 - Sviluppato da OpenGIS Consortium
 - Descrive le informazioni geografiche
 - Le informazioni geografiche elementari sono dette features
 - Ogni feature possiede:
 - Proprietà
 - Entità geometriche

Linguaggi di markup personalizzati

- **eXtensible Business Reporting Language (XBRL)**
 - Permette di rappresentare dati di natura finanziaria, economica e amministrativa
- **Electronic Business Language (ebXML)**
 - Usato per lo scambio di informazioni commerciali e ed industriali
- **Commerce XML (cXML)**
 - Usato per descrivere dati di catalogo e svolgere transazioni elettroniche fra aziende che usano tali dati

Linguaggi di markup personalizzati

- LegalXML

- Ideato per ridurre la ridondanza di informazioni e documenti giudiziari nei sistemi di gestione di tali dati

- NewsML

- Utilizzato nei sistemi di gestione delle news

- Rich Site Summary (RSS)

- Per creare canali che distribuiscono automaticamente le informazioni. RSS consente agli autori Web di creare un link che i visitatori possono selezionare per ricevere un determinato canale

Linguaggi di markup personalizzati

- **eXtensible User Interface Language (XUL)**
 - Usato per descrivere le interfacce utente
 - Sviluppato dal progetto Mozilla
 - Cross-platform

Linguaggi di markup personalizzati

- Scalable Vector Graphics (SVG)
 - Permette di descrivere immagini vettoriali

```
<svg width="300" height="300">  
  <circle style="fill:green;fill-opacity:0.5"  
    cx="50" cy="150" r="50"/>  
</svg>
```

- *Il DOM SVG è parte di HTML5*

Linguaggi di markup personalizzati

- **eXtensible 3D Markup Language (X3D)**
 - Permette di descrivere scene in grafica 3D (si appoggia al sistema grafico del client)
 - Estensione di VRML

Linguaggi di markup personalizzati

- eXtensible 3D Markup Language (X3D)

```
<x3d width='500px' height='400px'>
  <scene>
    <shape>
      <appearance><material diffuseColor='1 0 0'></material>
      </appearance>
      <box></box></shape>
    <transform translation='-3 0 0'><shape>
      <appearance><material diffuseColor='0 1 0'></material>
      </appearance>
      <cone></cone></shape></transform>
    <transform translation='3 0 0'><shape>
      <appearance><material diffuseColor='0 0 1'></material>
      </appearance>
      <sphere></sphere></shape></transform>
    </scene>
  </x3d>
```


Linguaggi di markup personalizzati

- eXtensible 3D Markup Language (X3D)

- Il DOM X3D **non** è parte di HTML5
- Per visualizzarlo è necessario includere una *libreria Javascript* e un apposito *foglio di stile CSS*

```
<script type='text/javascript'  
src='https://www.x3dom.org/download/x3dom.js'></script>  
<link rel='stylesheet' type='text/css'  
href='https://www.x3dom.org/download/x3dom.css'></link>
```