



2

Informazione, informatica e risorse culturali digitali sul web: guida all'uso dei linguaggi ipertestuali

UniMc - BBCC 10/11 – I semestre

Laboratorio di informatica
avanzata per i beni culturali

prof. Pierluigi Feliciati
pierluigi.feliciati@unimc.it

I'ARCHITETTURA del WEB

Gli aspetti che caratterizzano una pagina Web sono quattro:

1. il contenuto

che rappresenta l'informazione che si vuole comunicare all'utente, che come è facile intuire è l'aspetto che maggiormente contribuisce a determinare il valore che l'utente attribuisce alla pagina

2. la struttura

intesa sia come struttura di navigazione, costituita dall'insieme di link che consentono all'utente di raggiungere le varie sezioni di un sito, eventuali form di ricerca e login, ecc. sia come struttura logica del contenuto, cioè l'organizzazione di questo in paragrafi, intestazioni, elenchi puntati e numerati, ecc.

I'ARCHITETTURA del WEB

Gli aspetti che caratterizzano una pagina Web sono:

3. la presentazione

che rappresenta la modalità con cui la struttura viene presentata all'utente. La presentazione può essere di tipo visuale (grafica), testuale e non visuale, per gli utenti che usano tecnologie assistive ed alcuni browser alternativi

4. il comportamento

che consente di alterare la struttura e la presentazione in risposta ad eventi generati dall'utente (pressione di un tasto del mouse, movimento del puntatore, pressione di un tasto della tastiera, ecc.) o ad eventi e condizioni indipendenti dall'utente

L'ARCHITETTURA del WEB

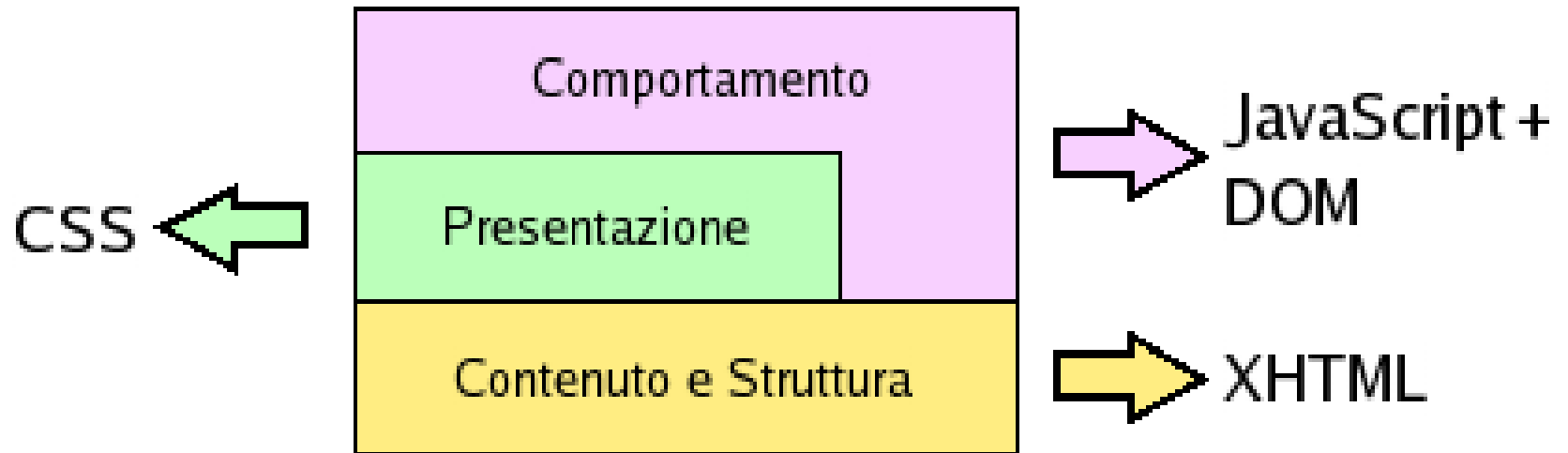
Gli aspetti che caratterizzano una pagina Web

I primi tre aspetti (**contenuto**, **struttura**, **presentazione**) definiscono una pagina come entità statica, l'ultimo (**comportamento**) ne determina la dinamicità.

Premesso che il contenuto ed un minimo di struttura logica sono sempre presenti, non tutte le pagine presentano tutti e quattro gli aspetti.

Molte pagine, infatti, non presentano nessuna struttura di navigazione, altre non presentano nessun comportamento ed altre ancora, per la verità piuttosto rare, non presentano alcuna presentazione.

l'architettura del web a strati



Lo strato di comportamento poggia sia su quello di presentazione, che su quello di contenuto e struttura in quanto agisce su entrambi quello di presentazione poggia unicamente su quello di contenuto e struttura. Ogni strato agisce esclusivamente su quello sottostante (quelli sottostanti nel caso del comportamento) e deve totalmente ignorare la presenza di quelli sovrastanti.

le risorse digitali e il WWW

I formati delle risorse

Sono la **modalità con cui i dati vengono codificati** digitalmente in modo che i programmi possano decodificarli, restituirli e elaborarli.

Per elaborare un file, un programma deve sapere come i dati vi sono “memorizzati” (es. html, jpg, avi, wmf)

Il formato specifica la **corrispondenza fra la rappresentazione binaria e i dati rappresentati**

I formati digitali si classificano **anche** come:

- **proprietary** (posseduta in esclusiva da un soggetto che ne mantiene segreto il funzionamento)
- **aperti** (è resa pubblica ed è documentata)
- **liberi** (libertà di utilizzare, studiare, modificare e redistribuire)

le risorse digitali e il WWW

Type:text

E' il primo e più universale dei formati secondo la classificazione standard MIME. Si tratta di dati codificati con combinazioni 1:1 di mappe caratteri – charset – anche per esprimere eventuali formattazioni.

Subtype and extensions:

plain	txt, text	Cfr. RFC 2046
html	htm, html	
sgml	sgml	http://www.w3.org/MarkUp/SGML/
xml	xml, dtd, x	http://www.w3.org/MarkUp/
rtf	rtf	www.soft-land.org/documenti/rtf.html

le risorse digitali e il WWW

I linguaggi di markup

Il termine **markup** (o **marcatura**) deriva dall'ambiente tipografico dove si usava marcare con annotazioni le parti del testo che andavano evidenziate o corrette.

La tecnica di composizione di un testo con l'uso di marcatori (o espressioni codificate) richiede una serie di convenzioni, ovvero di un **linguaggio a marcatori** di documenti. In generale un linguaggio di markup **descrive i meccanismi di rappresentazione** (strutturali, semantici o presentazionali) **del testo** che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti di lettura.

I *linguaggi di marcatura di tipo descrittivo* lasciano la scelta del tipo di rappresentazione da applicare al testo al software che di volta in volta lo riprodurrà

HTML

I linguaggi di markup: HTML

HTML è un linguaggio usato per scrivere i documenti ipertestuali per il Web, ossia **descrive il contenuto, testuale e non, di una pagina web**. E' un linguaggio standard, di pubblico dominio, la cui sintassi è stabilita dal **World Wide Web Consortium (W3C)**

Durante gli anni l'HTML ha subito molte revisioni e miglioramenti: attualmente l'ultima versione disponibile è la versione 4.01, resa pubblica il 24 dicembre 1999.

Da allora, da parte del W3C non è stata manifestata alcuna intenzione di apportare ulteriori modifiche all'HTML, poiché verrà presto sostituito dai nuovi linguaggi **XHTML** ed **XML** di cui si dirà.

HTML

Ogni documento HTML deve essere contenuto in un **file**, la cui **estensione** deve essere **.htm** o **.html**.

L'atomo principale della sintassi di questo linguaggio è **l'elemento**. Gli elementi sono le **strutture del linguaggio** a cui è delegata la funzione di formattare i dati o indicare al Web browser delle informazioni. Ogni **elemento** è racchiuso all'interno di **tag, uno di apertura ed uno di chiusura**, quest'ultimo, per certi elementi opzionale, è indicato apponendo il carattere **slash (/)** dopo la parentesi angolare aperta. I tag sono **marcatori (markup)** costituiti da una sequenza di caratteri racchiusa da due parentesi angolari, cioè i segni minore e maggiore.

HTML

la struttura dei documenti

Ogni pagina HTML, nella sua interezza, costituisce essa stessa **un elemento** di nome **HTML** e contiene **due sottoelementi distinti: HEAD e BODY**.

I tag iniziali e finali che delimitano tutti e tre i suddetti elementi sarebbero opzionali, ma è buona norma specificarli comunque.

<HEAD> contiene l'insieme di elementi che consentono di specificare informazioni generali sul documento che non ne costituiscono il contenuto informativo vero e proprio, insomma i suoi **metadati**.

<BODY> delimita gli elementi che costituiscono il **contenuto informativo** vero e proprio del documento.

HTML

HEAD e metadati

Oltre all'elemento **TITLE**, l'elemento **META** viene inserito nella sezione HEAD e consente di specificare informazioni (o proprietà) non specificabili mediante altri elementi. I METADATI, che possono essere estratte ed **utilizzate dai server, dai browser e dai motori di ricerca**. META riconosce i seguenti attributi:

NAME per specificare il valore della proprietà che si sta definendo
CONTENT per attribuire un nome alla proprietà che si sta definendo:

```
<META NAME="Author" CONTENT="Pierluigi Feliciati">
```

```
<META NAME="Description" CONTENT="Pagina che mostra l'uso  
del tag META">
```

```
<META NAME="keywords" CONTENT="HTML, META">
```

HTML

HEAD e metadati

Per non lasciare alla discrezionalità di ogni sito la definizione dei **NAME** dei **META**, è stato elaborato come abbiamo detto lo standard **DUBLIN CORE**. Vediamone un esempio di uso (si noti la dichiarazione dello schema con LINK) per l'home page dell'AIB:

```
<link rel="schema.DC" href="http://dublincore.org" />
```

```
<meta name="DC.Title" content="AIB-WEB (Associazione italiana biblioteche)" />
```

```
<meta name="DC.Creator" content="Associazione italiana biblioteche" />
```

```
<meta name="DC.Subject" content="Associazione italiana biblioteche, AIB, Italian library  
association" />
```

```
<meta name="DC.Description" lang="it" xml:lang="it" content="Home page dell'Associazione  
italiana biblioteche" />
```

```
<meta name="DC.Date.Created" content="1997-03-01" />
```

```
<meta name="DC.Language" scheme="RFC1766" content="it" />
```

HTML

BODY: i tag

I tag (elementi) di BODY possono essere **strutturali o formali**

Un esempio base di elemento formale è `<br>` che serve per indicare **un ritorno a capo**, che non ha bisogno di tag di chiusura. Quasi sempre **le informazioni su cui "agisce" il tag** devono essere racchiuse fra un tag di apertura ed uno di chiusura:

prima, `<b>`testo testo testo`</b>` e dopo

marca la parte di testo che deve essere visualizzato **in grassetto**:

prima, **testo testo testo** e dopo

ci sono molti altri elementi che agiscono a livello di fonte carattere, per ottenere corsivo `<i>`, sottolineato `<u>`, etc.

HTML

Gli attributi

Gli attributi che hanno lo scopo di **specificare le proprietà degli elementi**, se presenti, sono caratterizzati dal seguente formato:

nome_attributo = valore

Non tutti gli attributi necessitano di un valore. I valori, se presenti, possono essere racchiusi tra apici singoli (') o doppi (").

Esempio: l'elemento **SIZE** consente di impostare il livello di **dimensione dei caratteri** esplicitamente **<SIZE="n">** oppure come incremento **<SIZE="+n">** o decremento **<SIZE="-n">** rispetto al livello base

HTML

la struttura dei documenti HTML

Riassumendo, quindi, ogni pagina HTML dovrebbe avere questa struttura, che si raccomanda di definire prima dell'inserimento dei contenuti:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Titolo</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<P> Paragrafi... </P>
```

```
</BODY>
```

```
</HTML>
```

HTML

HTML

In HTML esistono **6 livelli di intestazioni** specificati mediante gli elementi **H?** (con ? che può assumere i valori da 1 a 6).

Ogni intestazione è visualizzata su una riga separata utilizzando una fonte carattere ed uno stile che dipendono dal browser.

La dimensione dei caratteri utilizzati, comunque, diminuisce al crescere del livello dell'intestazione.

Livello 1

Livello 2

Livello 3

Livello 4

Livello 5

Livello 6

HTML

Struttura: le liste

Sono previste **tre tipologie di liste** in HTML:

- **di definizioni**, che consentono di rappresentare elenchi di termini con le relative definizioni;
- **ordinate**, che consentono di rappresentare elenchi numerati;
- **non ordinate**, che consentono di rappresentare elenchi i cui elementi vengono contrassegnati con un simbolo grafico (generalmente un pallino riempito).

HTML

Le liste non ordinate vengono delimitate dall'elemento `<UL>`. I loro elementi, invece, vengono specificati mediante l'elemento `<LI>` (il cui tag finale è opzionale).

`<UL>`

`<LI>` Elemento 1

`<LI>` Elemento 2

`<LI>` Elemento 3

`<LI>` Elemento 4

`<LI>` Elemento 5

`</UL>`

- Elemento 1
- Elemento 2
- Elemento 3
- Elemento 4
- Elemento 5

HTML

Le liste ordinate vengono delimitate dall'elemento ****. I loro elementi, invece, vengono specificati mediante l'elemento **** (il cui tag finale è opzionale).

** Elemento 1**

** Elemento 2**

** Elemento 3**

** Elemento 4**

** Elemento 5**

1. Elemento 1
2. Elemento 2
3. Elemento 3
4. Elemento 4
5. Elemento 5

HTML

È consentito **annidare liste** anche di tipo diverso.

Quando si annidano liste dello stesso tipo il browser utilizzerà contrassegni diversi per gli elementi dei vari livelli e in ogni caso ogni livello sarà più indentato del precedente.

Verdura

 Lattuga

 Pomodori

 Carote

- Verdura
 - 1.Lattuga
 - 2.Pomodori
 - 3.Carote

HTML

i link

Ogni link è caratterizzato da **due estremità**, denominate **anchor**, ed **una direzione**. Un link ha **inizio dall'anchor di origine e punta a quello di destinazione** che può essere una qualsiasi risorsa individuabile mediante una URL.

Nella definizione di un link oltre ad indicarne l'anchor di destinazione è anche possibile specificare **il tipo di relazione tra questo e l'anchor di origine**.

I link – la base dell'ipertesto - vengono comunemente utilizzati per accedere mediante la **selezione di una componente testuale e/o grafica del documento alla risorsa a cui puntano**.

HTML

i link

L'elemento `<A>` viene utilizzato per definire link e anchor di destinazione all'interno del corpo dei documenti (sezione BODY). Per funzionare utilizza l'attributo `HREF` che consente di definire **un link il cui anchor di origine è il contenuto dell'elemento A e quello di destinazione è la URL che costituisce il suo valore.**

L'utente avrà accesso a tale URL selezionando il testo e/o le componenti grafiche che costituiscono il contenuto di A.

Per evidenziare la presenza del link **il testo viene visualizzato sottolineato** mentre **le immagini appaiono bordate**, utilizzando un colore specificabile mediante gli attributi di `<BODY>`.

HTML

i link

È possibile definire un link specificando un valore per **HREF**, ma non è consentito annidare un elemento **A** all'interno di un altro elemento dello stesso tipo.

Vediamo un esempio di definizione di un link in cui la pagina *prova.html* viene associata alla parola “selezionami”:

`<A HREF="ApproccioWeb.html">Selezionami</A>` per accedere ad una semplice pagina di prova che ti mostrerà l'uso dei link.

ottiene:

Selezionami per accedere ad una semplice pagina di prova che ti mostrerà l'uso dei link.

HTML

Link assoluti e link relativi

nella sintassi delle URL, è possibile separare idealmente la parte che si riferisce al nome del server (dominio) e al percorso gerarchico per arrivare al nome del file (detta **URL base**) dal nome del file. Ecco un link assoluto:

```
<A HREF="www.w3c.org/Consortium/Offices/role.html">
```

HTML consente di inserire nel documento link relativi, che **omettono la parte di indirizzo già dichiarata** al browser, oppure uguale a quella del documento. Se volessi rinviare al file `role2.html` che si trova nella stessa cartella dello stesso server, sarebbe infatti sufficiente:

```
<A href="role2.html">
```

HTML

dichiarazione URL base

è possibile dichiarare nella **<HEAD>** la **URL base** tramite l'elemento **<BASE>**.

Nel caso mancasse **BASE** la **URL base** sarà quella che individua la **directory contenente il documento** di origine dei link.

BASE riconosce ovviamente l'attributo **HREF** per specificare la URL:

```
<HEAD>
...
<BASE HREF="www.w3c.org/Consortium/Offices/">
</HEAD>
```

HTML

Link per l'invio di e-mail

Utilizzando sempre l'elemento `<A>` ma con schema **mailto** (mailto:nome_utente@macchina?subject="oggetto") possono essere utilizzate per creare un link la cui selezione fornisce all'utente la possibilità di **inviare un messaggio di posta elettronica ad un qualsiasi indirizzo e-mail** (come ad esempio quello dell'autore o del responsabile della pagina).

Per maggiori informazioni contattare: `<A`

`HREF="mailto:pierluigi.feliciati@unimc.it">`Pierluigi Feliciati

`</A>`

ottiene:

Per maggiori informazioni contattare: Pierluigi Feliciati

HTML

le immagini

Per inserire un'immagine (ovvero un **file immagine**) in una pagina HTML si utilizza l'elemento **IMG**, con l'attributo **SRC** che consente di specificare la URL dell'immagine da visualizzare e alcuni altri attributi per specifiche di posizione e formattazione dell'immagine. La sorgente:

Questo simbolo `<IMG SRC="smile.gif">` viene spesso utilizzato per sottolineare che una frase scritta e' uno scherzo.

restituisce (se **smile.gif** è un link corretto):

Questo simbolo  viene spesso utilizzato per sottolineare che una frase scritta e' uno scherzo.

HTML

le form

Le form sono maschere attraverso le quali l'utente può inserire dei dati. Questi potranno essere processati mediante una **CGI** (Common Gateway Interface), uno script scritto in uno dei linguaggi che è possibile "immergere" nelle pagine HTML (tipicamente JavaScript) o una combinazione dei due.

All'interno delle form, oltre agli elementi che ne definiscono le componenti (*controlli*), possono essere contenute tutte le entità HTML fatta eccezione per altre form. Non è possibile quindi annidare form, anche se una pagina ne può contenere più d'una.

La definizione di una form avviene mediante l'elemento **FORM**, con gli attributi **ACTION** e **METHOD**

HTML

un esempio di form

```
<FORM ACTION="" METHOD="POST">
```

```
  Nome: <INPUT TYPE="text" NAME="nome" SIZE="25" MAXLENGTH="30">
```

```
<P>Cognome: <INPUT TYPE="text" NAME="cognome" SIZE="25"
  MAXLENGTH="30">
```

```
<P>Indirizzo: <INPUT TYPE="text" NAME="indirizzo" SIZE="25" MAXLENGTH="30">
```

```
<P>Sesso: <INPUT NAME="sesso" TYPE=RADIO VALUE="f">femminile
```

```
  <INPUT NAME="sesso" TYPE=RADIO VALUE="m" CHECKED>maschile
```

```
<P>Che genere musicale preferisci?<BR><SELECT NAME="genere_musicale">
```

```
  <OPTION> rock
```

```
  <OPTION> pop
```

```
  <OPTION> musica classica </SELECT>    <P>
```

```
<INPUT TYPE=SUBMIT VALUE="Invia">
```

```
<INPUT TYPE=RESET VALUE="Cancella">
```

```
</FORM>
```

HTML

Ecco il nostro esempio di form

Nome:

Cognome:

Indirizzo:

Sesso: ☐ femminile ☒ maschile

Che genere musicale preferisci?



HTML

HTML e i fogli di stile

Ci siamo limitati qui a vedere e sperimentare le funzioni principali del metalinguaggio HTML, prevedendo che ogni pagina web debba avere, oltre ai contenuti, le indicazioni riguardo alla struttura e al formato di visualizzazione.

In realtà, da tempo si è superato questo metodo, preferendo **separare i contenuti dagli stili di visualizzazione**, attraverso i **fogli di stile a cascata (CSS - Cascading Style Sheets)**, una tecnica che permette di fissare gli stili (per es. tipo di carattere, colori e spaziature) da applicare ai documenti HTML e XHTML. Le regole per comporre i fogli di stile sono contenute in un insieme di direttive emanate a partire dal 1996 dal W3C.

HTML

HTML e i fogli di stile

L'inserimento di codice CSS nelle pagine web può essere effettuato in due modi:

- 1- Inserendo nel tag <head> della pagina un **collegamento ad un foglio di stile esterno**, cioè un file contrassegnato dall'estensione **.css** come nell'esempio seguente:

```
<html>
  <head>
    <title>Esempio</title>
    <link rel="stylesheet" type="text/css" href="foglio_di_stile.css">
  </head>
  <body> ... </body>
</html>
```

HTML

HTML e i fogli di stile

2 - Inserendo, sempre all'interno dell'<head> tra gli specifici tag **<style>** e **</style>** le dichiarazioni CSS:

```
<html>  
  <head>  
    <title>Esempio</title>  
    <style type="text/css">  
      comandi css  
    </style>  
  </head>  
  <body> ... </body>  
</html>
```

XML

oltre HTML: XML

XML, acronimo di **eXtensible Markup Language** è un metalinguaggio creato e gestito dal World Wide Web Consortium (W3C). È una semplificazione e adattamento dell'**SGML**, da cui è nato nel 1998, e permette di definire la grammatica di diversi linguaggi specifici derivati.

Rispetto all'HTML, l'XML ha uno scopo ben diverso: XML è un linguaggio utile allo **scambio dei dati**, quindi di **back-office** e **non di front-office**, o di rappresentazione che dir si voglia.

La sua versione 1.0 risale al 1998 e la versione italiana del linguaggio è in <http://www.xml.it:23456/XML/REC-xml-19980210-it.html>, la seconda specifica del 2000, la terza e la versione 1.1 (compatibile con Unicode 4) del 2004 <http://www.w3.org/TR/REC-xml>

XML

ecco un esempio tipico di file XML. Lo si può visualizzare con un browser qualsiasi semplicemente salvandolo in un file di testo con estensione **.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<utenti>
  <utente>
    <nome>Luca</nome>
    <cognome>Ruggiero</cognome>
  </utente>
  <utente>
    <nome>Max</nome>
    <cognome>Rossi</cognome>
  </utente>
</utenti>
</xml>
```

XML

La struttura vera e propria è composta dai **tag** creati dallo sviluppatore, che hanno due caratteristiche:

- devono essere comprensibili in funzione dello scopo (onde evitare di non capirci nulla ad una seconda lettura e per facilitare la comprensione agli altri utenti);
- devono rispettare delle regole, come la differenza tra maiuscolo e minuscolo, non possono iniziare con numeri o caratteri speciali e non possono contenere spazi.

Ogni **record**, se si vuol pensare ad XML come ad un **database**, viene chiamato **nodo** ed ogni tag può essere corredato da **attributi**, come per HTML..

XML

Ad esempio, la struttura precedente può essere modificata come segue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<utenti>
  <utente nome="Luca" cognome="Ruggiero" />
  <utente nome="Max" cognome="Rossi" />
</utenti>
</xml>
```

I tag che non hanno l'omonimo tag di chiusura vanno chiusi con uno spazio più **slash (/)** finale, prima della fine del tag stesso.

XML

La struttura di XML come apparirà ormai chiaro **non ha interfaccia propria** ma è accessibile da qualsiasi linguaggio di programmazione Web (sia lato server che lato client). Il rispetto delle sue semplici regole ed il fatto che i dati sono accessibili semplicemente insieme alla loro struttura a qualsiasi tipo di applicazione, locale o remota, fanno dell'XML uno standard che si caratterizza per la sua **portabilità**.

L'accesso ad una struttura XML può avvenire attraverso diversi sistemi, il più comune ma rudimentale dei quali è lo sfruttamento del *filesystem* (la gestione dei file in memoria) del server su cui gira il motore del linguaggio in uso.

XML

La semplicità, adattabilità e piena compatibilità di XML con gli altri linguaggi di marcatura, la quasi totale mancanza di caratteristiche opzionali, la leggibilità umana e la facilità per le macchine di elaborarlo ne hanno decretato il successo.

Lo sviluppo del linguaggio DTD prima, di XSD poi, l'adozione dei *NameSpace* (per la definizione delle strutture dati e il richiamo di diverse regole in uno steso file) e degli *stylesheet* (per la visualizzazione) hanno sostenuto questo percorso. L'unico difetto è la poca economicità in termini di spazio di occupazione, dovuta alla ridondanza di elementi oltre ai dati veri e propri.

Vediamo sinteticamente alcune regole sintattiche di base di XML, per poter meglio comprendere gli standard sul trattamento dei testi.

Sintassi XML

Le regole sintattiche di Xml sono poche:

- Ogni documento deve presentare un prologo per la dichiarazione della specifica Xml che si usa, le eventuali informazioni di codifica e il richiamo esplicito alla DTD in uso

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Questa prima riga, racchiusa tra i simboli `<?` e `?>` si dice *processing instruction* e dalla versione 1.1 di XML è divenuta obbligatoria. Se omessa, il documento diventerà automaticamente 1.0

Sintassi XML

Il contenuto di un documento XML deve corrispondere a una gerarchia di **tag** (o **elementi**) **non sovrapposti**, quindi i tag che si aprono per primi debbono chiudersi per primi.

Per elemento/tag (marcatore) si intende una coppia

<parola>(apertura) </parola> (chiusura) con parola=nome del tag

<DOCUMENTO>

<TITOLO>Titolo del documento</TITOLO>

<TESTO>

<PARAGRAFO>testo del primo paragrafo</PARAGRAFO>

<PARAGRAFO>testo del secondo paragrafo</PARAGRAFO>

</TESTO>

</DOCUMENTO>

Sintassi XML

- Deve esistere un tag che contenga tutti gli altri, che si dice **tag radice** o *root*.
- Tutti i tag devono **sempre essere aperti e chiusi** e se non contengono testo, per economia, possono essere presentati usando la cd. *chiusura implicita*:

<TAGVUOTO/> equivale a <TAGVUOTO></TAGVUOTO>

- I nomi dei tag sono **a nostro piacere**, rispettando però alcune regole:
 - Devono iniziare con un carattere o con un _ (*underscore*), mai con numeri, possono contenere un numero qualsiasi di lettere, numeri, trattini, punti e underscore
 - Non possono contenere spazi e sono case-sensitive: <TAG> e <tag> sono diversi, i tag di apertura e di chiusura devono essere uguali

Sintassi XML

- Possono essere associati **attributi** agli elementi (tag), rappresentandoli nel tag di apertura, racchiudendone **il valore tra apici o virgolette** e non ripetendoli più di una volta:

`<PREZZO valuta='euro'>15</PREZZO>`

`<PREZZO valuta="euro">15</PREZZO>`

- Per i nomi degli attributi valgono le stesse regole degli elementi
- Esistono **attributi riservati** che possono essere usati solo rispettandone il significato previsto, ad es.
 - `xml:lang="it | en | fr ..."` indica la lingua del tag
 - `xml:space="preserve | default"` indica se gli spazi presenti nel contenuto devono essere mantenuti o considerati superflui (quindi ev. eliminati)

Sintassi XML

- Nel caso sia necessario usare **all'interno del contenuto caratteri speciali** (come <, >, &, " etc.) si dovrà usare il cd. **Entity-name** corrispondente, composto da un &, da un nome e da un punto-e-virgola finale, ad es.

& corrisponde a &

< corrisponde a <

> corrisponde a >

" corrisponde a "

- Se si deve inserire nel contenuto una porzione di testo **scritta in un codice diverso** (ad es. HTML, JavaScript), si dovrà usare la tipologia di contenuto **CDATA section**

```
<contenuto>
  <![CDATA[
    <html><body> .... </body></html>
  ]]>
</contenuto>
```

XML e *namespace*

- Il rispetto della sintassi XML garantisce **documenti *well-formed*** (ben costruiti), leggibili da qualunque software su qualunque piattaforma hardware. I browser visualizzano i file XML ed evidenziano se ci sono errori.
- I software che leggono i documenti XML si dicono ***parser*** perché scandiscono riga per riga il codice e lo interpretano.
- A volte è necessario poter usare dentro al proprio documento XML alcuni elementi che hanno **una particolare provenienza**, vale a dire che sono **già stati definiti in un qualche schema** di codifica. Lo strumento per andare incontro a questa esigenza è chiamato ***namespace***.

XML e *namespace*

Un *namespace* è indicato da un **prefisso** distintivo.

Per usarlo è necessario **dichiararlo** all'interno di un qualsiasi elemento di un documento XML, usando un attributo standard, **xmlns** (abbreviazione di XML NameSpace) seguito da un due punti, dal prefisso distintivo del namespace da adottare e dalla URL dove si trova lo schema cui il namespace fa riferimento.

Se, per esempio, vogliamo usare alcuni elementi definiti dallo schema *Dublin Core* per codificare una citazione bibliografica, basterà usare il **namespace dc** nel modo che segue:

```
<citazione xmlns:dc="http://purl.org/dc/elements/1.1/">  
  <dc:creator>Carlo Dionisotti</dc:creator>  
  <dc:title>Geografia e storia della letteratura italiana  
  </dc:title>  
  <dc:publisher>Einaudi</dc:publisher>  
</citazione>
```

XML e *namespace*

..che è più funzionale, internazionale e *interoperabile* rispetto a questo codice:

<citazione>

<autore>Carlo Dionisotti</autore>

<titolo>Geografia e storia della letteratura italiana</titolo>

<editore>Einaudi</editore>

</citazione>

perché si adotta **uno standard usato da molti** e che quindi molti (intendendo soprattutto le macchine) possono decodificare facilmente.

XML: definizione degli schemi

Ci sono diversi modi per **dichiarare uno schema di codifica**, vale a dire la grammatica che stabilisce quali elementi sarà possibile usare all'interno di un documento XML e in che modo tali elementi si potranno relazionare fra di loro.

Il modo più conosciuto per dichiarare uno schema di codifica è attraverso una **Document Type Definition** (o **DTD**), una sintassi ereditata direttamente dal linguaggio SGML, da cui l'XML deriva. La DTD è costituita da una sintassi estremamente semplice, ma con varie limitazioni.

Recentemente a questo metodo si sono aggiunti alcuni linguaggi per scrivere **XML Schema** (o **XSD**). Lo schema più usato è appunto quello del W3C

<http://www.w3.org/TR/xmlschema-0/>.

XHTML

da XML a XHTML

L'HTML non è un vero standard data la sua eccessiva flessibilità ed il suo funzionamento immutato anche in presenza di errori semantici, sintattici e grammaticali.

Ad affiancarlo c'è **XHTML (eXtensible HyperText Markup Language)** ovvero l'HTML tradizionale basato su XML, con la sua struttura più definita e con le sue stesse regole: : un file XHTML si può definire come un file XML scritto con i termini dell'HTML.

Il linguaggio prevede **un uso più restrittivo dei tag HTML**; solo la struttura della pagina è scritta in XHTML, mentre il **layout è imposto** dai fogli di stile a cascata (CSS).

XHTML

La maggior parte dei browser attualmente più diffusi è in grado di rendere correttamente i documenti XHTML, poiché questo linguaggio è in buona parte un **sottoinsieme dell'HTML**. Lo stesso vale anche in senso inverso: quasi tutti i browser compatibili con l'XHTML renderizzano correttamente i documenti HTML.

Secondo un'opinione diffusa, questo **alto grado di compatibilità** sta rallentando il passaggio da HTML a XHTML. Per sfruttare appieno le potenzialità dell'XHTML è necessario usarlo **in abbinamento ai fogli di stile**.

XHTML

La differenza più importante tra HTML e XHTML è che tutti i tag devono essere **ben strutturati**, cioè obbedire ad una serie di regole che ne assicurino la coerenza reciproca.

Inoltre: i tag devono essere sempre scritti **in lettere minuscole**, convenzione in contrasto con l'abitudine invalsa a partire dalla versione 2.0 di HTML;

tutti gli attributi (compresi quelli numerici) devono essere scritti **fra virgolette**, cosa facoltativa in SGML e HTML;

tutti gli elementi del linguaggio devono essere **terminati**, compresi quelli vuoti (ad esempio `img` e `br`). Per eseguire la terminazione in modo implicito si può aggiungere una `"/` di chiusura (es: `<img ... />` e `<br />`).

Verso il WWW semantico: RDF

RDF e la marcatura semantica

RDF - Resource Description Framework è un framework per la descrizione della conoscenza nel web. Esso è stato specificatamente creato, secondo una recommendation del W3C, per la **descrizione dei metadati relativi alle risorse**. Questo framework è alla base del cosiddetto **web semantico**, e permette la condivisione di conoscenza sul web. L'RDF si basa su tre principi chiave:

1. Qualunque cosa può essere identificato da un URI
2. Il *least power*: è da utilizzare un linguaggio meno espressivo possibile per definire qualunque cosa
3. Qualunque cosa può dire qualunque cosa su qualunque cosa

Verso il WWW semantico: RDF

Qualunque cosa descritta da RDF è detta **risorsa**

Ogni risorsa è identificata da un **URI**.

Il modello di dati RDF è formato da **risorse, proprietà e valori**.

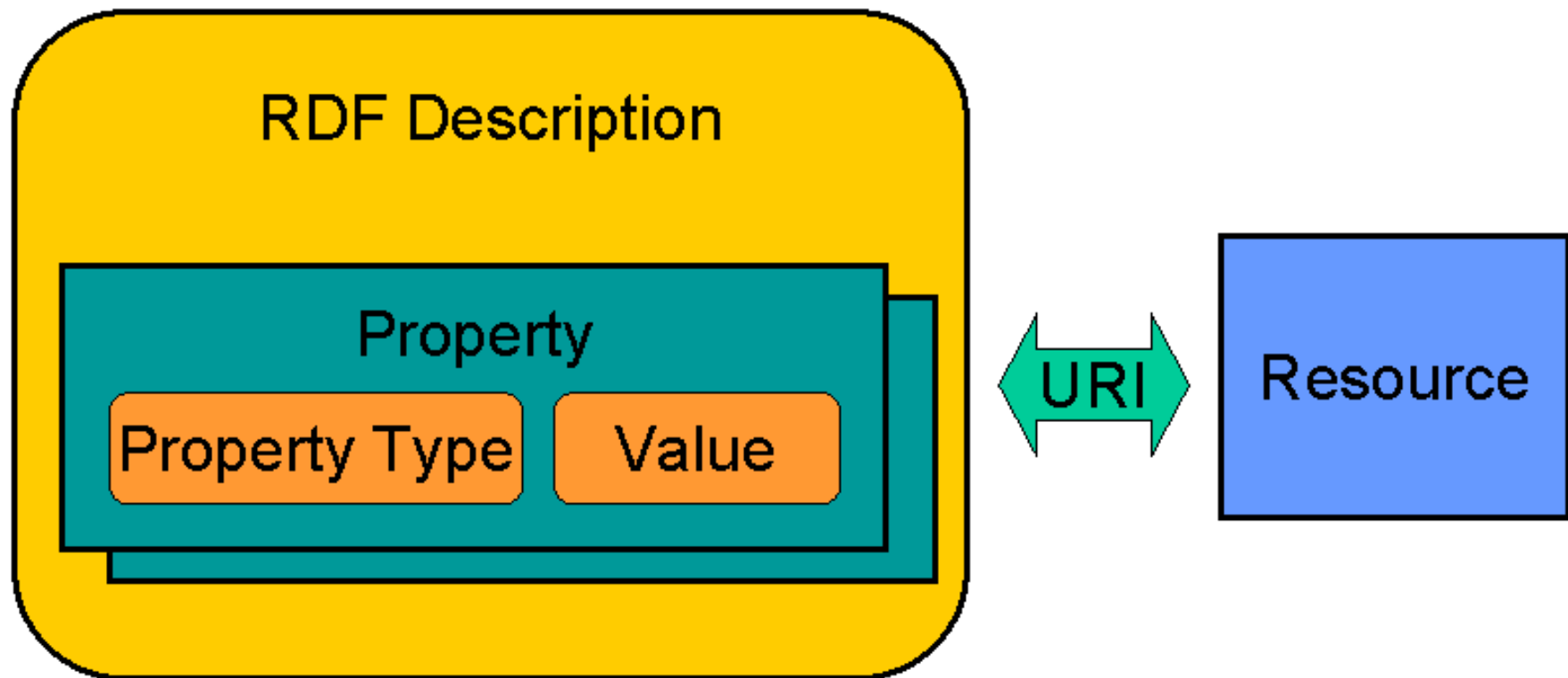
Le **proprietà** sono delle **relazioni che legano tra loro risorse e valori**, e sono anch'esse identificate da URI.

Un **valore**, invece, è **una risorsa o è un dato** primitivo.

L'unità base per rappresentare un'informazione in RDF è una tripletta del tipo: **Soggetto – Predicato – Oggetto** dove il soggetto è una risorsa, il predicato è una proprietà e l'oggetto è un valore.

Verso il WWW semantico: RDF

Una rappresentazione di conoscenza in RDF



Verso il WWW semantico: RDF

Le principali modalità adottabili per esprimere RDF sono:

XML: con metodo classico o abbreviato, più leggibile per l'uomo

N3: si descrive una risorsa e tutte le sue proprietà

“Alessandro_Manzoni” “è_autore_di”

“I_promessi_sposi” in RDF/XML si esprime così:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:au="http://description.org/schema/">
  <rdf:Description about="http://www.book.it/I_promessi_sposi/">
    <au:author>Alessandro_Manzoni</au:author>
  </rdf:Description>
</rdf:RDF>
```