Creare risorse digitali: formati, linguaggi, modelli

Francesca Tomasi francesca.tomasi@unibo.it

Creare risorse digitali

Il calcolatore ragiona in forma binaria e converte **sequenze di bit** in lettere, pixel, frammenti, a seconda della tipologia di oggetto (testo, immagini, audio, video).

Produrre un **contenuto digitale** significa gestire questioni di:

- **Linguaggio**. Scegliere il linguaggio formale più adeguato a seconda dello scopo, dell'obiettivo e della funzione del file generato
- **Formato**. Ogni tipo di oggetto ha un formato più adatto a seconda, di nuovo, dello scopo o della funzione che deve svolgere
- **Modello**. Ogni oggetto realizzato rispecchia un modello di riferimento che, chi l'ha prodotto, aveva in mente

Oltre a costruire file, creare risorse digitali significa provvedere alla **realizzazione di un ambiente di accesso e fruizione**

I formati dei file: estensione e *naming*

Il formato spiega l'applicazione (o le applicazioni) che ha generato il file e la/le applicazione/i che possono aprirlo, modificarlo ed utilizzarlo

I più comuni **formati per il web**:

Testo – CSV, TXT, HTML, CSS, JS

Immagini – JPG, PNG, GIF

Audio - MP3

Video – MP4

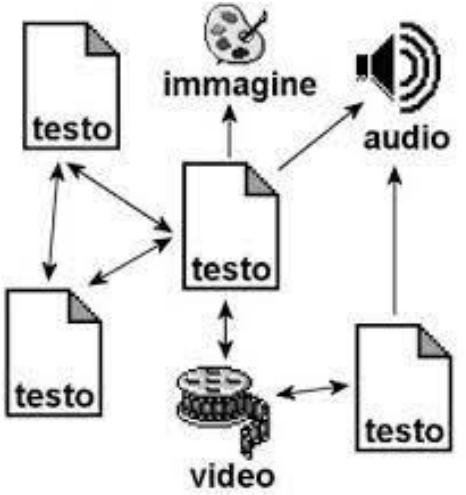
Il nome è fondamentale (l'estensione, in alcuni casi, si può cambiare!):

ilmiofile.estensione

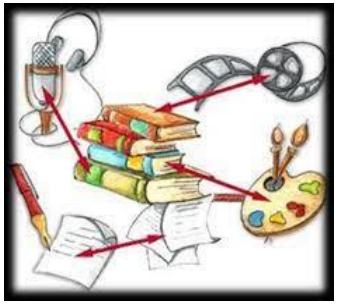
IlMioFile.estensione

il-mio-file.estensione

il_mio_file.estensione



Creare ipertesti (ipermedia) per il Web



Costruire un sito Web è prima di tutto una questione di **creare testi**, corredati di immagini, eventualmente audio o video (inseriti o richiamati nel flusso dei caratteri del testo).

In prima battuta è dunque necessario capire come si **produce il testo digitale** in termini di linguaggi, formati e modelli

Produrre un **testo per il Web** significa:

- Creare i contenuti che il sito ospiterà
- Scegliere (o realizzare) le componenti multimediali che si voglio integrare (inserire o richiamare)
- Creare l'ambiente per l'accesso e la navigazione di quei contenuti

Creare il testo digitale

Partiamo dalla codifica dei dati elementari. Il testo

La **codifica del testo** basata sulle sole **tavole dei codici**, insieme ai limiti di portabilità e compatibilità comporta un'ulteriore restrizione:

- ✓ essa consente di rappresentare nella memoria del computer solo la **sequenza dei segni grafici** che rappresentano il testo
- ✓ un testo, sia esso a stampa o manoscritto, come noto, contiene una serie di **informazioni**, a diversi livelli, che superano la mera sequenza di caratteri.

Distingueremo la codifica di 'basso livello' o **codifica dei dati elementari** da una codifica di 'alto livello' o rappresentazione dei dati a livello di 'strutture intermedie' che chiamiamo ***markup***.

La codifica dei caratteri

- ✓ ASCII (*American Standard Code for Information Interchange* http://www.asciitable.it).
  - 7 bit 128 caratteri ($2^7 = 128$)
  - 8 bit 256 caratteri ($2^8 = 256$)

Proliferazione=difficoltà di interscambio

- ✓ ISO (*International Organization for Standardization* http://www.iso.org/iso/home.html) 8859-n (ASCII 7 bit + 1)
- ✓ UNICODE (http://www.unicode.org). 16 bit, comprende ben 65.536 caratteri (2^{16}). Ultima versione 21 bit (2^{21}) oltre 1 milione di caratteri.
  - UTF-8 (ASCII 7 bit + 1)

Ruolo e funzione dei marcatori

Con *markup* ci riferiamo alla possibilità di aggiungere alla sequenza di caratteri che rappresentano il testo digitale, **altre stringhe di caratteri** denominate **marcatori**, utili a descrivere determinati aspetti funzionali alla produzione della risorsa digitale.

Questo processo di aggiunta di stringhe al flusso dei caratteri permette di **specificare determinate caratteristiche del testo**.

Siamo abituati all'uso dei *word processors* che ci consentono di effettuare questa operazione manipolando un'interfaccia grafica costituita da bottoni e menù: definire grassetti, realizzare corsivi, assegnare ad una porzione di testo uno stile, cambiare il tipo di carattere o le sue dimensioni.

Il *markup* come modellazione

La codifica dei caratteri prima e il *markup* poi rappresentano un **processo interpretativo**, risultato dell'analisi del testo.

Riguardano quindi la costruzione di un **modello** di quel testo più adeguato alle esigenze della rappresentazione digitale.

Codificare tramite **linguaggi formali di rappresentazione del testo** significa:

- ✓ effettuare un'**analisi** del testo, mirata ad individuarne le caratteristiche
- ✓ formulare un'**interpretazione** della fonte, sulla base delle *features* del documento
- ✓ in modo direttamente proporzionale agli **scopi** del trattamento informatico

Dal *layout* alla struttura

Il termine *markup* deriva dalla **stampa tipografica tradizionale** per riferirsi a quell'insieme di **simboli e annotazioni** che l'autore o l'editore aggiunge al manoscritto per istruire lo stampatore sulle caratteristiche del documento da destinare alla stampa (p.e. 'centrato', 'titolo', 'nota', 'grassetto', 'a capo').

Funzione dei linguaggi di *markup* è di fornire un insieme di **strumenti** che consentano di aggiungere **notizie** sul testo. Tali notizie possono riguardare due differenti livelli:

- l'**aspetto**: formattazione (stili, tipi di carattere, colore e dimensione dei font, ecc.) e disposizione degli elementi nella pagina (allineamento, organizzazione spaziale delle componenti testuali, disposizione di note e annotazioni, ecc.).
- la **struttura logica** (o astratta o formale): funzione dei blocchi di testo (paragrafi, titoli, note, capitoli, ecc.), cioè organizzazione delle porzioni secondo un sistema formale identificabile.

I sistemi WYSIWYG

Word-processors: programmi che consentono all'utente di effettuare operazioni di scrittura, correzione, lettura di un qualsiasi testo, permettendo altresì la preparazione del formato dello stesso testo al fine della stampa.

I sistemi di *text processing* (cioè di manipolazione o trattamento del testo) basati sull'impiego di *word processors* sono detti di tipo *WYSISYG* (*What You See Is What You Get*) cioè: ciò che vedi (sullo schermo dell'elaboratore) è ciò che ottieni (una volta stampato il documento).

Questi sistemi agevolano notevolmente il lavoro dell'utente, consentendogli di interagire tramite l'interfaccia grafica. Permettono quindi di manipolare la rappresentazione visibile sullo schermo, e gestire le attività di formattazione e impaginazione.

Il limite dei software proprietari

Il problema sostanziale è che questi sistemi legano l'elaborazione del testo ad un determinato **programma**, rendendo quindi problematica la portabilità tra ambienti hardware e software diversi.

Si pensi al programma di videoscrittura *Word* della casa produttrice Microsoft che non solo pone limitazioni alla portabilità del formato doc da un PC a un Macintosh, ma crea anche difficoltà di lettura da una versione all'altra del programma stesso.

Impiegando cioè, al fine della rappresentazione del testo, dei **caratteri di controllo invisibili**, immessi dentro il file di testo, questi linguaggi rendono il file leggibile esclusivamente dal sistema che l'ha generato.

È **l'applicativo che inserisce la marcatura**, impiegando un linguaggio proprietario, che lega il prodotto finale al software di creazione.

Estensione di formato dei file e visualizzazione della codifica

Dante online
Sito della Società Dantesca Italiana.
Di particolare interesse il censimento dei manoscritti danteschi della *Commedia*. Per molti dei codici descritti è possibile accedere alle immagini digitali dei manoscritti e alla trascrizione diplomatica.
<http://www.danteonline.it/italiano/home_ita.htm>

dante.txt - Blocco note

File Modifica Formato Visualizza ?

```
Dante online
Sito della Società Dantesca Italiana.
Di particolare interesse il censimento dei manoscritti danteschi della Commedia. Per
molti dei codici descritti è possibile accedere alle immagini digitali dei manoscritti
e alla trascrizione diplomatica.
<http://www.danteonline.it/italiano/home_ita.htm>
```

Dante online.rtf - Blocco note

File Modifica Formato Visualizza ?

```
{\rtf1\ansi\ansicpg1252\uc1\deff0\stshfdbch0\stshfloch0\stshfhich0\st
\deflangfe1040{\fonttbl{\f0\froman\fcharset0\fprq2{\*\panose 0202060
{\f223\froman\fcharset238\fprq2 Times New Roman CE;}
{\f224\froman\fcharset204\fprq2 Times New Roman Cyr;}{\f226\froman\f
Greek;}{\f227\froman\fcharset162\fprq2 Times New Roman Tur;}{\f228\f
New Roman (Hebrew);}
{\f229\froman\fcharset178\fprq2 Times New Roman (Arabic);}{\f230\fro
Roman Baltic;}{\f231\froman\fcharset163\fprq2 Times New Roman (Vietn
\green0\blue0;\red0\green0\blue255;\red0\green255\blue255;
\red0\green255\blue0;\red255\green0\blue255;\red255\green0\blue0;\red255\green255\blue0;\red255
\green255\blue255;\red0\green0\blue128;\red0\green128\blue128;\red0\green128\blue0;\red128\green0
```

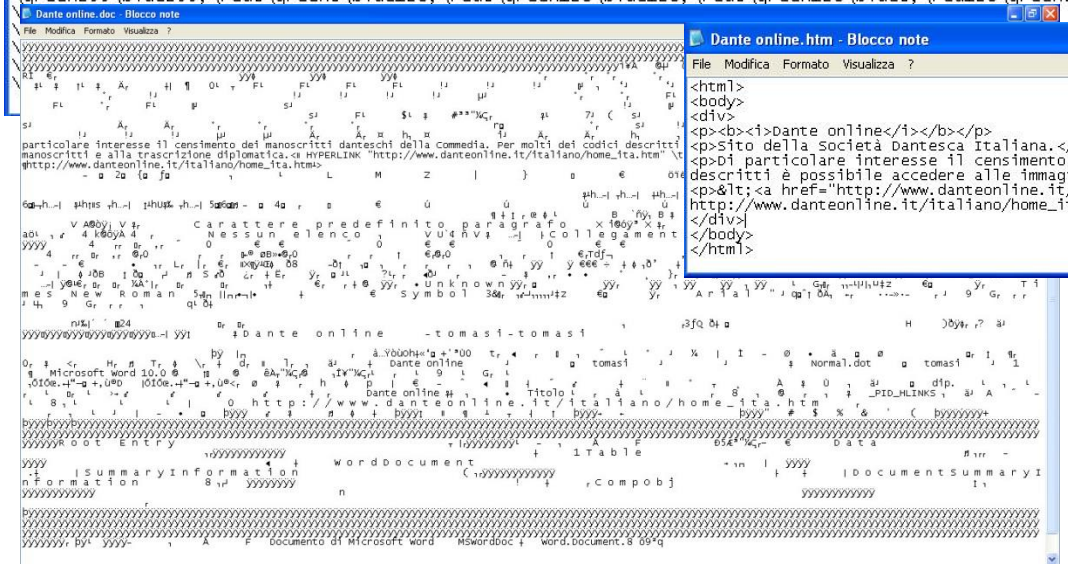
Dante online.xml - Blocco note

File Modifica Formato Visualizza ?

```
<corpo>
<sezione>
<titolo>Dante online</titolo>
<sottotitolo>Sito della Società Dantesca Italiana.</sottotitolo>
<contenuto>Di particolare interesse il censimento dei manoscritti danteschi della <opera>Commedia</opera>. Per
molti dei codici descritti è possibile accedere alle immagini digitali dei manoscritti e alla trascrizione
diplomatica.</contenuto>
<sito>&lt;<collegamento indirizzo="http://www.danteonline.it/italiano/home_ita.htm">
http://www.danteonline.it/italiano/home_ita.htm</collegamento>&gt;</sito>
</sezione>
</corpo>
```

Dante online.doc - Blocco note

File Modifica Formato Visualizza ?



Dante online.htm - Blocco note

File Modifica Formato Visualizza ?

```
<html>
<body>
<div>
<p><b><i>Dante online</i></b></p>
<p>Sito della Società Dantesca Italiana.</p>
<p>Di particolare interesse il censimento dei manoscritti danteschi della <i>Commedia</i>. Per molti dei codici
descritti è possibile accedere alle immagini digitali dei manoscritti e alla trascrizione diplomatica.</p>
<p>&lt;<a href="http://www.danteonline.it/italiano/home_ita.htm">
http://www.danteonline.it/italiano/home_ita.htm</a>&gt;</p>
</div>
</body>
</html>
```

Struttura (.xml, .htm) vs Layout (.rtf, ma anche .htm)
Codifica leggibile (.txt, .xml, .rtf, htm.) vs Non leggibile (.doc)

I linguaggi di markup

Markup languages

Se i *word processors* consentono una forma di *markup* del testo, si parla più propriamente di *markup languages*, detti in italiano linguaggi di marcatura del testo, per riferirsi a linguaggi che si basano su un **insieme di istruzioni e indicazioni** orientate alla **descrizione dei fenomeni di strutturazione, composizione, impaginazione** del testo stesso.

I marcatori si configurano come sequenze di caratteri visibili, e vengono immessi dentro il file, secondo una determinata **sintassi**, immediatamente accanto alla sequenza di caratteri a cui si riferiscono e cioè marcando direttamente i blocchi di testo cui intendono assegnare una determinata funzione.

Garantiscono in questo modo la **leggibilità** (che non necessariamente significa però completa comprensione) del codice introdotto.

Markup procedurale e *markup* dichiarativo

È possibile distinguere due diverse classi di linguaggi di *markup*, che differiscono per il tipo di indicazioni impiegate all'atto dell'operazione di marcatura ovvero per la **tipologia** e la **funzione** dei marcatori utilizzati:

- ✓ *markup* specifico (o procedurale)
- ✓ *markup* generico (o dichiarativo)

Linguaggi di *markup* dichiarativo

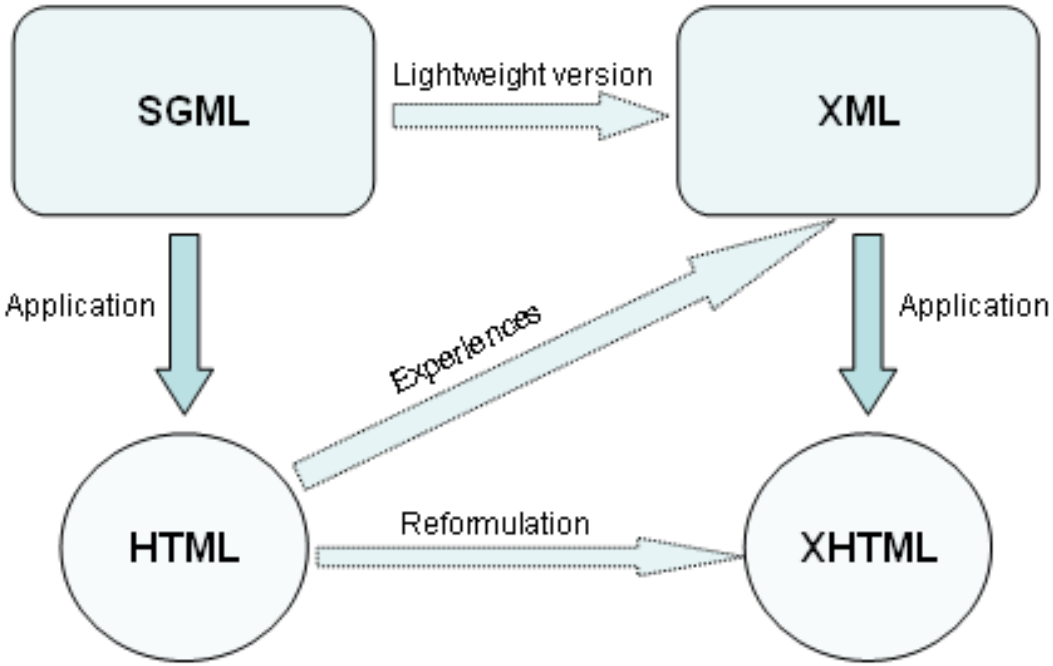
Principali linguaggi di *markup* dichiarativo:

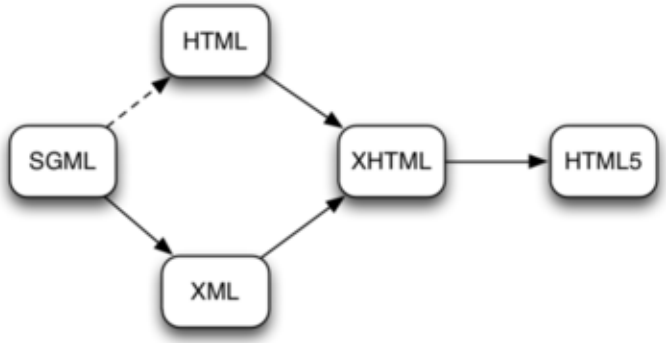
- ✓ **SGML** (1986), il capostipite
- ✓ **HTML** (1991), una sua diretta derivazione pensata per l'implementazione di ipertesti
- ✓ **XML** (1998), un sottoinsieme semplificato dell'SGML pensato per la realizzazione di testi elettronici portabili anche sul Web e a scopi di preservazione.

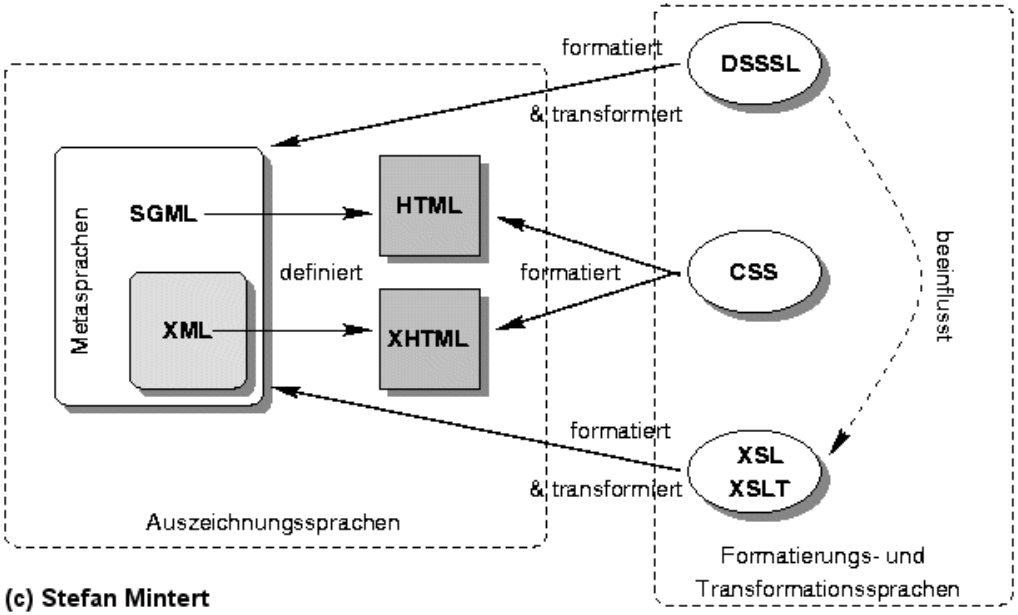
I linguaggi dichiarativi:

- ✓ si basano principalmente sulla descrizione della **struttura** logica del documento e sono quindi prevalentemente utilizzati a scopo **descrittivo**;
- ✓ il formato dei dati **non è proprietario**;
- ✓ i marcatori sono **leggibili** dall'utente;
- ✓ sono *platform-independent* perche basati esclusivamente su istruzioni espresse in formato **'solo testo'**.

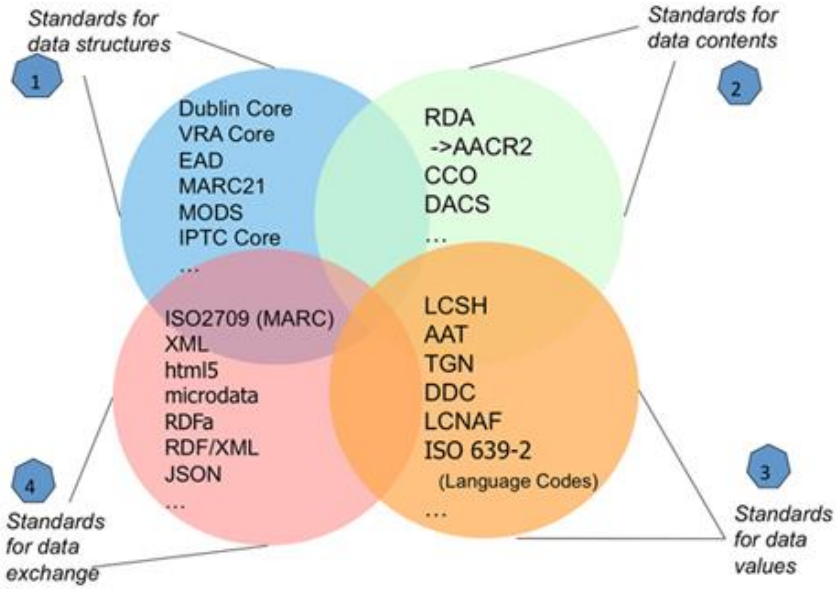
SGML, XML, HTML, XHTML, HTML5, CSS, DC… UN UNIVERSO DI STANDARD







(c) Stefan Mintert



Le origini: SGML

SGML (*Standard Generalized Markup Language*), GML prima di diventare standard, è stato elaborato nel 1986 da Charles Goldfarb con lo scopo di definire uno schema linguistico **standard a livello internazionale nell'ambito della codifica dei testi** e superare la moltitudine di sistemi di codifica fino ad allora sviluppati.

Obiettivo primario dello standard è consentire l'**interscambio** di documenti in formato elettronico tra ambienti hardware e software differenti e garantire quindi la **portabilità** dei dati.

La struttura gerarchica ad albero

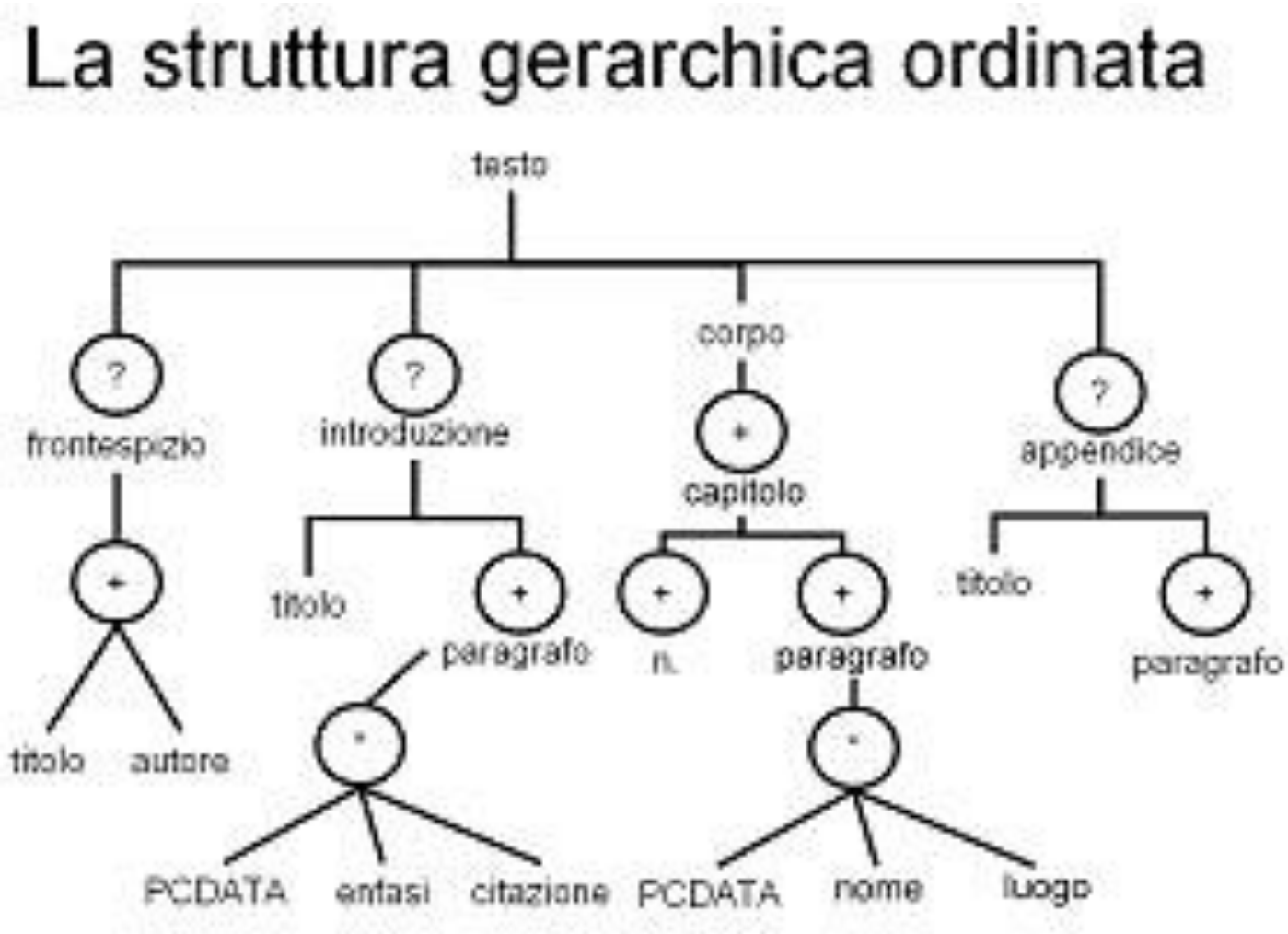
SGML (ma anche XML e tutti i linguaggi derivati come HTML) si basa su di un *markup* generico, orientato alla descrizione della struttura logica di un testo piuttosto che del suo aspetto.

Si fonda sull'idea che ogni documento sia dotato di una struttura astratta (o logica) definibile tramite una organizzazione rigidamente gerarchica dei suoi elementi costitutivi.

La struttura astratta di un **qualsiasi** documento viene identificata in una rappresentazione ad **albero** in cui:

- ✓ a ciascun nodo dell'albero corrisponde un elemento (e cioè ogni partizione logica della fonte)
- ✓ ai rami uscenti da ogni nodo corrispondono le relazioni tra elementi e sotto-elementi ad un dato livello (le relazioni tra elementi possono essere relazioni di **inclusione**, di **ordine** e di **ricorrenza**: per esempio potremmo dire che un elemento paragrafo è incluso in un elemento capitolo e può ricorrere molteplici volte, oppure che un elemento introduzione deve precedere un elemento capitolo, ecc.)
- ✓ alle foglie corrispondono gli elementi finali (generalmente i caratteri del testo).

Un esempio di albero gerarchico per un semplice documento è: **nodo radice** corrispondente al testo stesso; testo composto da un'eventuale **introduzione**, seguita da un certo numero di **capitoli**; a loro volta capitoli composti, ognuno, da un **titolo**, seguito da uno o più **paragrafi**; nodi terminali, o foglie, contenenti **stringhe di caratteri** costituenti il testo stesso (indicati con PCDATA) o altri **elementi marcati**, a seconda delle esigenze del *markup* (e.g. persone, luoghi, date, nomi, citazioni, etc.).



Il concetto di classe

Fondandosi su di una codifica di tipo dichiarativo, SGML (e XML) consente di **definire (o di dichiarare), in modo assolutamente personale ed autonomo, un insieme di marcatori (*tags*)** che adempiano al compito primario dello standard, che consentano cioè di fare il *markup* della struttura logica del documento.

Questo gruppo di marcatori, più precisamente, individua una **classe** di documenti testuali che presentano le medesime caratteristiche strutturali.

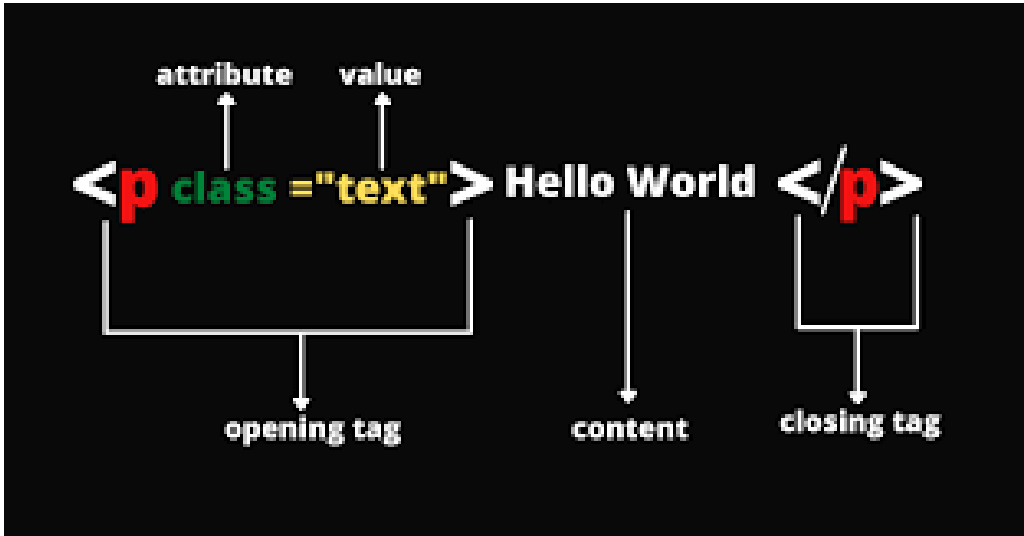
Con **classe intenderemo un insieme di documenti che condividono determinate proprietà**: i testi poetici, i testi narrativi, i testi drammatici sono per esempio tre classi che possiedono una specifica struttura logica; una poesia sarà di conseguenza marcata secondo le proprietà della classe testi poetici.

Elementi, attributi e valori

Ogni porzione testuale può essere dunque individuata e descritta tramite ricorso ad un nome convenzionale che prende il nome di **elemento** (p.e. ‘p’ per paragrafo, ‘n’ per nota ecc.) racchiuso tra due **delimitatori**; la porzione strutturale viene quindi rappresentata tramite un marcatore o *tag* di apertura ed uno di chiusura.

È possibile anche associare agli elementi degli **attributi**, che possiedono un determinato **valore**.

Per i *tags*, la sintassi di SGML ci suggerisce una forma grafica simbolica: parentesi ad angolo, o delimitatori, racchiudenti il nome dell’elemento, cioè l’identificatore dell’elemento, come *tag* d’apertura (`<elemento>`); stesse parentesi ma con l’identificatore preceduto dal simbolo dello slash / per il *tag* di chiusura: `</elemento>`).



Struttura sintattica di un documento conforme ad SGML (ma anche a XML)

```
<poesia>
  <titolo>Titolo della poesia</titolo>
  <strofa numero="1" tipo="terzina">
      <verso numero="1">Questo è il primo verso</verso>
      <verso numero="2">Questo è il secondo verso</verso>
      <verso numero="3">Questo è il terzo verso</verso>
   </strofa>
</poesia>
```

L'**indentazione** (rientro della riga rispetto al margine sinistro) mostra la **gerarchia**, ovvero il grado di annidamento di ogni elemento (elemento padre e elemento/i figlio/i). Nell'esempio:

Poesia è l'elemento radice

Titolo e strofa dipendono gerarchicamente da poesia

Verso è gerarchicamente dipendente da strofa

Il giorno dei morti

Io vedo (come è questo giorno, oscuro!)
vedo nel cuore, vedo un camposanto
con un fosco cipresso alto sul muro.

```
<poesia>
  <titolo>Il giorno dei morti</titolo>
  <strofa numero="1">
     <verso numero="1">Io vedo (come è questo giorno, oscuro!)</verso>
     <verso numero="2">vedo nel cuore, vedo un camposanto</verso>
     <verso numero="3">con un fosco cipresso alto sul muro.</verso>
  </strofa>
</poesia>
```

Funzioni di SGML

Più esattamente diremo che SGML è dotato di una sintassi astratta che spiega come operare il *markup* di un documento testuale, fornendo **regole che istruiscono l'utente su come aggiungere i marcatori** (cioè si dovranno porre una *tag* iniziale e una *tag* finale ad ogni porzione testuale; ogni *tag* sarà costituita da un nome convenzionale, che identifica l'elemento, racchiuso tra due delimitatori), **senza però fornire norme specifiche sui nomi per elementi ed attributi**. E questo vale anche per XML.

SGML non fornisce dunque nessuna indicazione circa il vocabolario per gli elementi, e nemmeno speciali caratteri in qualità di delimitatori (non obbliga cioè l'uso delle parentesi angolate, suggerisce solo questo tipo di notazione simbolica).

SGML e XML come metalinguaggi

Per questo è lecito affermare che SGML e XML più che linguaggi sono classificabili come **metalinguaggi**, e forniscono esclusivamente le **regole sintattiche** necessarie alla creazione di altri linguaggi di *markup* di testi.

Dal momento che ragionano per classi di documenti, diremo che ciascuno dei linguaggi derivati si configura come un peculiare **modello del testo** o, più esattamente, di un **insieme di testi aventi caratteristiche logico-strutturali analoghe**.

Il concetto di DTD

- Il metalinguaggio non fornisce dunque alcuna prescrizione relativamente alla tipologia, alla quantità o al nome dei marcatori, ma si occupa esclusivamente di **fornire precise regole sintattiche**, necessarie a definire un insieme di marcatori
- Il valore di tali marcatori va specificato in un **vocabolario di marcatura**, che è detto "definizione del tipo di documento", o DTD (*Document Type Definition*)
- La DTD è quindi la **grammatica del metalinguaggio** o anche il linguaggio impiegato per la rappresentazione di determinati parametri logico-strutturali di gruppi di documenti aventi le medesime caratteristiche
- Si pensi alla tradizionale suddivisione per il testo letterario fra testo in prosa, testo in versi, testo drammatico, testo parlato, ecc. Ciascuno di questo macro-raggruppamenti è un **tipo di documento**. Ma anche il testo letterario in sé può essere considerato tale.

DTD e Schema di codifica

La DTD si chiama anche lo **schema di codifica** (a partire da XML abbiamo Schemi invece di DTDs) cui un testo fa riferimento – o anche il **modello** – che ha il compito di definire da un lato i **diversi aspetti** della fonte che possono essere oggetto di intervento interpretativo e, dall'altro, **specificare il vocabolario** associato a ciascuno degli aspetti.

Testo da marcare

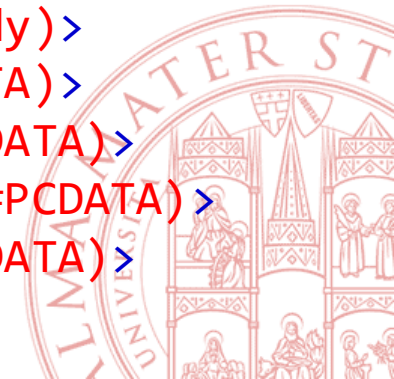
Tove Jani Reminder Don't forget me

File marcato secondo un certo modello

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me</body>
</note>
```

File note.dtd

```
<!ELEMENT note
(to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```



Funzioni della DTD

- i marcatori per gli **elementi** (*elements*) che identificano la serie delle proprietà di un testo o di una certa classe di documenti; per ciascuno di tali elementi è definito un nome convenzionale che qualifica la proprietà strutturale della porzione della fonte che andrà ad identificare: p.e. ‘t’ per testo, ‘intro’ per introduzione, ‘cap’ per capitolo, ‘p’ per paragrafo, ‘n’ per nota, ecc.;
- la descrizione del **contenuto di ogni elemento** (*content model*), quindi quali altri elementi possono apparire, con quale ordine e con quale frequenza (una sezione sarà suddivisa in paragrafi, ogni un paragrafo potrà contenere parole e le parole potranno contenere lettere);
- i marcatori per gli **attributi** (*attributes*) assegnabili ad un qualsivoglia elemento (un titolo potrà essere caratterizzato dalla sua tipologia di livello, un capitolo dal suo numero progressivo, ecc.);
- i simboli (stringhe di testo ascii/Unicode) per le **entità** (*entity*) che possono occorrere all’interno del documento e che rappresentano: caratteri non esistenti nel *code set* impiegato alla codifica, forme contratte che vanno estese in fase di *layout* (per esempio un’abbreviazione), oggetti esterni (altri file sgml o non- sgml). https://www.w3schools.com/charsets/

Un esempio di DTD relativa alla classe "quotidiani"

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE
(HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY TRADEMARK "#8482">
<!ENTITY TELEPHONE "#9742">
<!ENTITY COPYRIGHT "#169">

]>
```

HTML (e il ruolo dei CSS)

HTML come DTD

L'HTML (*Hyper Text Markup Language*) è un formato non proprietario basato sull'SGML, più precisamente è una **DTD SGML** che nasce quindi nel rispetto delle specifiche della sintassi dello standard e che prescrive un vocabolario legato a quella **classe** di documenti che sono gli **ipertesti.**

Nato agli inizi degli anni '90 e ideato da Tim Berners Lee, il padre fondatore del WWW. Specifica dal sito del consorzio W3C.

E' diventato la lingua franca per la creazione di **siti Web**, che possono essere scritti in formato "solo testo" e visualizzati con un browser.

Rapido excursus storico

- ✓ Ipertesti (anni ‘40 del 1900). Modello di rappresentazione dei collegamenti
  - Vannevar Bush, *As we may think* (*Memex)*, 1945
  - Ted Nelson, *Hypertext*, 1963 (1965)
- ✓ Internet (anni ‘70 del 1900). Interconnessione fra calcolatori
  - Rete Arpanet, 1969
  - Connessione fisica fra calcolatori
  - Protocollo TCP/IP
- ✓ Web (anni ‘90 del 1900). Ambiente di accesso agli ipertesti che usa la rete Internet
  - Linguaggio per scrivere ipertesti: HTML, 1991
  - Identificazione univoca risorse URL/URI
  - Protocollo HTTP
  - Il ruolo del browser
  - I sistemi di Web hosting

Suggerimento: se non si è letto il manuale, cercare info su Wikipedia

HTML Markup language

Non si tratta solo di diverse versioni ma di un modo di intendere il Web che si è evoluto ed è cambiato nel corso del tempo. Con l'evoluzione del Web sono cambiate le funzioni e le caratteristiche del linguaggio attraverso cui il Web si esprime.

Version	**Year**
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

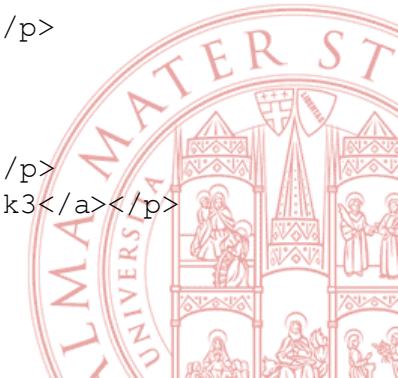
HTML 4.01: troppa enfasi sul layout. Deprecated!

```
<html>
<head>
<title>SCIENTIA RERUM</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="keywords" content="convegno, bologna, classici, scienze" />
    <meta name="description" content="scientia rerum la scienza di fronte ai classici convegno internazionale bologna" />
    <meta name="author" content="Francesca Tomasi" />
<link rel="stylesheet" href="stile.css" type="text/css">
</head>

<body>
<strong></strong>
<table width="80%" border="0">
  <tr>
    <td width="33%" height="154" rowspan="2" valign="top" bgcolor="#ced3e6"> <div align="center">
      </div>
      <div align="center">
        <p><font size="7" face="Georgia, Times New Roman, Times, serif"><img src="logoUNIBO.gif" width="182" height="48"></font><font color="#660033" size="1" face="Georgia, Times New Roman, Times, serif"><strong><br>
          </strong> <font color="#000000"><strong><br>
          <br>
          Centro Studi La permanenza del Classico</strong><br>
          Dipartimento di Filologia Classica e Medioevale Universit&agrave; degli
          Studi di Bologna <br>
          Via Zamboni, 32 - I - 40126 Bologna <br>
          Tel. 0512098539 - Fax 051228172<br>
          <br>
          <a href="http://www.classics.unibo.it/Permanenza">www.classics.unibo.it/Permanenza</a><br>
          <a href="mailto:permanenza@classics.unibo.it%20">permanenza@classics.unibo.it
          </a></font></font></p>
        </div></td>
    <td width="1%" height="154" rowspan="2"> <p> </p></td>
    <td width="66%" height="59" bgcolor="#f6f4f4">
      <div align="center">
        <p align="right"><em><font size="2" face="Geneva, Arial, Helvetica, sans-serif">Quid
          est bonum? Scientia rerum </font></em><font size="2" face="Geneva, Arial, Helvetica, sans-serif">(Seneca)</font></p>
      </div></td>
  </tr>
  <tr>
    <td height="157" bgcolor="#f6f4f4">
      <div align="center">
        <p><strong><font size="4" face="Georgia, Times New Roman, Times, serif">SCIENTIA
          RERUM</font><font size="7" face="Georgia, Times New Roman, Times, serif"><br>
          </font><font face="Georgia, Times New Roman, Times, serif">La <font color="#990000">scienza</font>
          di fronte ai <em>Classici</em></font></strong></p>
        <p><strong><font face="Georgia, Times New Roman, Times, serif"><strong>Bologna,
          29-30 settembre - 1 ottobre 2005<br>
          <font size="2">Aula Magna di Santa Lucia, via Castiglione 36 - BOLOGNA</font></strong></font></strong></p>
      </div></td>
  </tr>
  <tr>
    <td height="266" valign="top" bgcolor="#ced3e6">
      <div align="center">
        <p align="left"> </p>
        <p align="left"><font color="#000000"><b><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="programma.htm">Programma</a></font></b></font></p>
      </div>
      <p align="left"><font color="#000000" size="1" face="Georgia, Times New Roman, Times, serif"><strong><b><img src="../../images/freccia.gif" width="8" height="9"></b><a href="curricula.htm">Curricula
        dei relatori</a></strong></font> </p>
      <p align="left"><font color="#000000" size="1" face="Georgia, Times New Roman, Times, serif"><strong><b><img src="../../images/freccia.gif" width="8" height="9"></b><a href="handout.htm">Handout</a></strong></font>
      </p>
      <p align="left"><font color="#000000"><strong><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="comitato.htm">Comitato
        scientifico </a></font></strong></font></p>
      <p align="left"><font color="#000000"><strong><font color="#000000"><strong><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="patrocini sponsors.htm"
        e sponsors</a></font></strong></font><font size="1" face="Georgia, Times New Roman, Times, serif"><br>
        </font></strong></font></p></td>
    <td width="1%" height="266">
      <div align="center">
        <p> </p>
      </div>
      <p> </p></td>
    <td bgcolor="#f6f4f4"><div align="center">
```

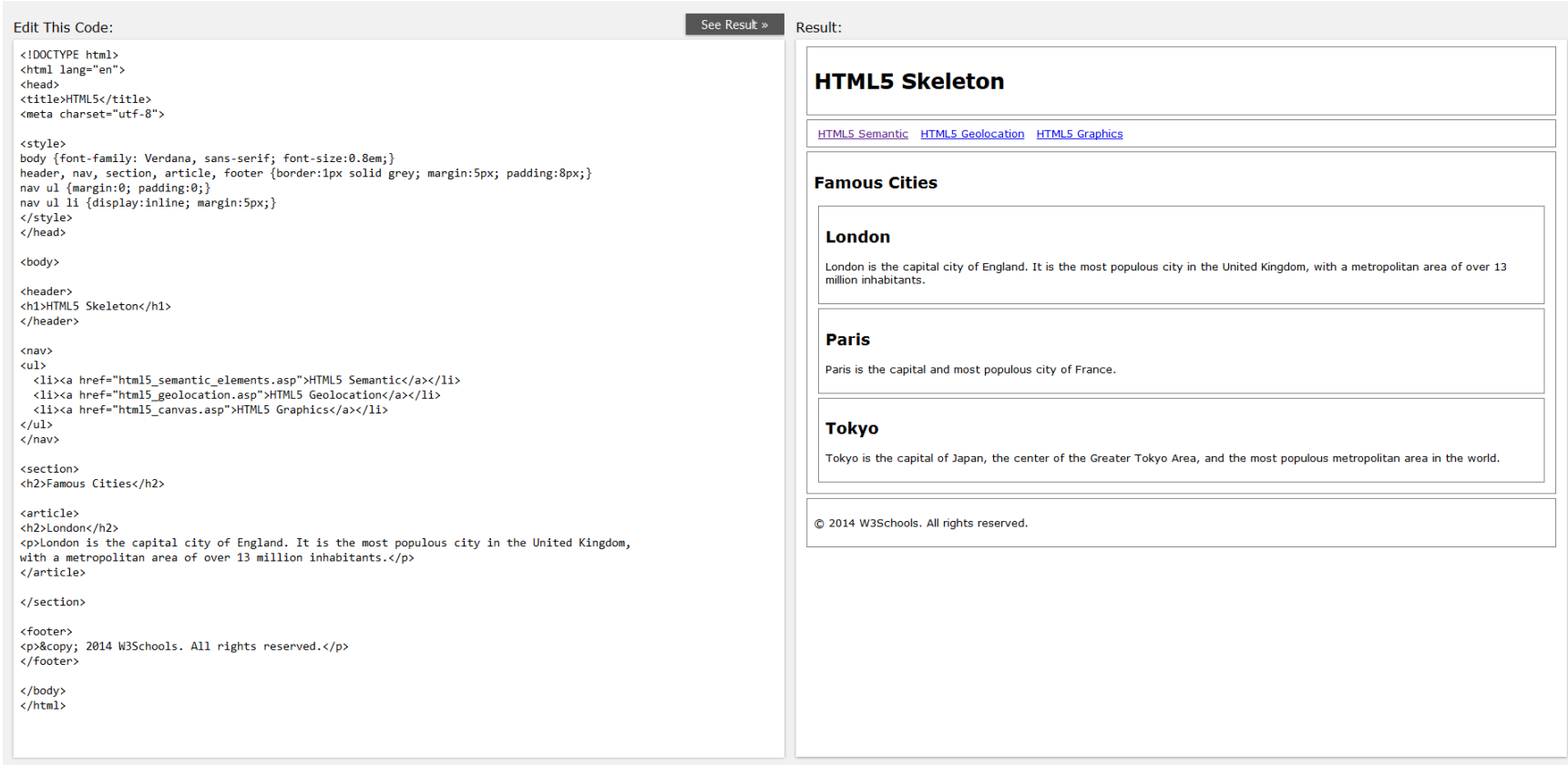
XHTML overview: verso la struttura. Il ruolo dell'elemento DIV

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Documento di prova</title>
        <link rel="stylesheet" href="stile.css"/>
    </head>
    <body>
        <div id="container">
            <div id="header">
               <h1>Testata del sito</h1>
            </div>
        <div id="menu-principale">
               <ul class="toc">
                   <li><a href="index.htm">Home</a></li>
                   <li><a id="activelink">Pagina attiva</a></li>
                   <li><a href="pagina1.htm">Pagina 1</a></li>
               </ul>
        </div>
             <div id="contenuto">
                  <hr/>
                <p>E qui metto il testo che voglio compaia come corpo centrale</p>
                <p>E magari voglio un'immagine <img src="immagine.png"/></p>
                <p>Altro paragrafo con <strong>una parola enfatizzata</strong></p>
            </div>
             <div id="piede-pagina">
                <p>&copy; Data, <span class="persona">Nome cognome</span></p>
                <p>E ancora altre informazioni: contatti, credits, validazione</p>
                <p><a href="">Link1</a> | <a href="">Link2</a> | <a href="">Link3</a></p>
            </div>
        </div>
    </body>
</html>
```



HTML5: l'enfasi sulla sola struttura

HTML5 verso la semantica della rappresentazione digitale, con fogli di stile CSS (*Cascading Style Sheets*) per la gestione dell'aspetto del documento.
Necessità di separare **struttura logica e contenuto** dall'**aspetto o layout** del documento



HTML5. Semantic elements. L'architettura d'interfaccia del sito Web

`<header>` - Defines a header for a document or a section

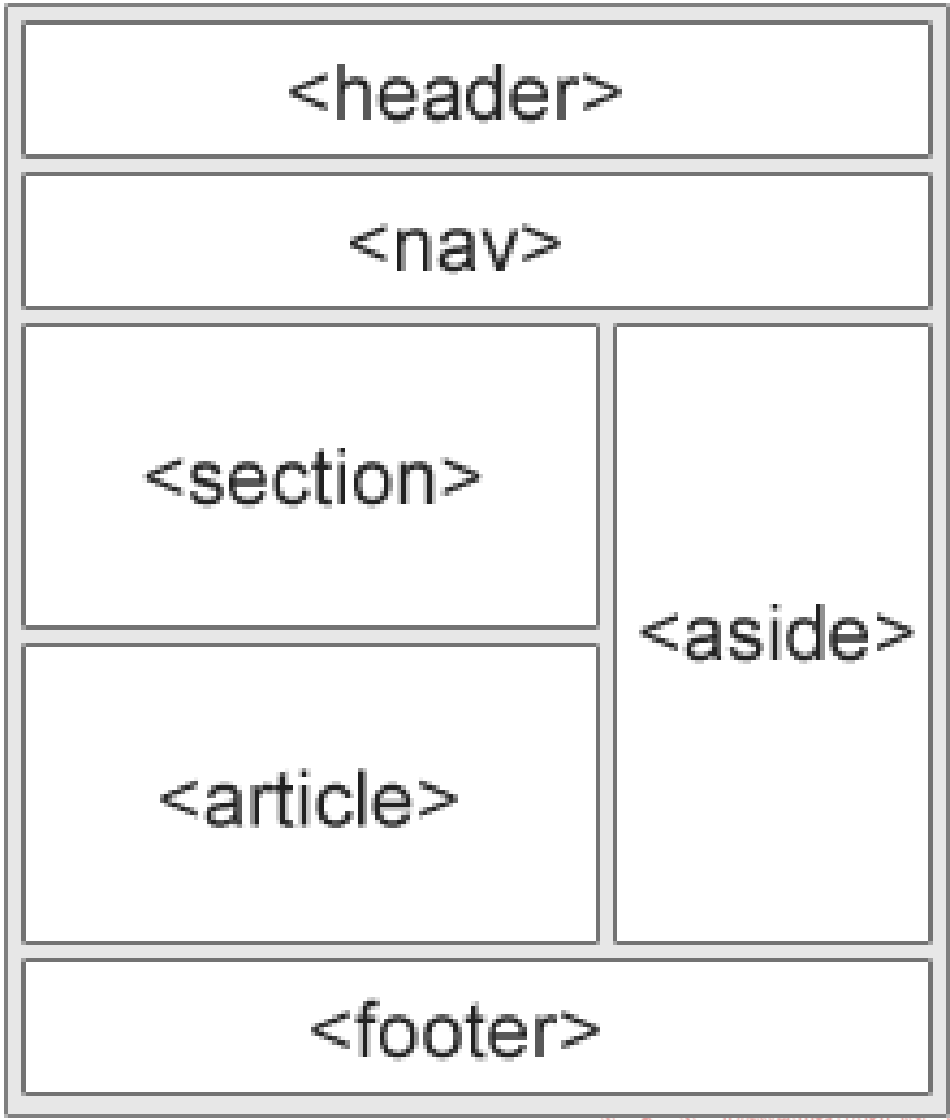
`<nav>` - Defines a set of navigation links

`<section>` - Defines a section in a document

`<article>` - Defines an independent, self-contained content

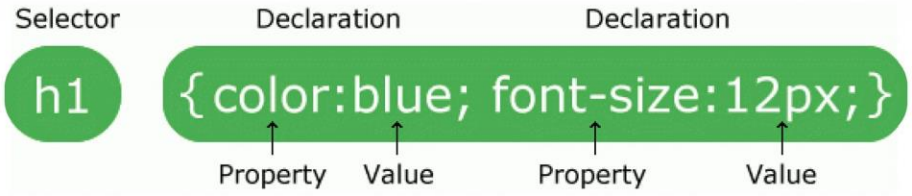
`<aside>` - Defines content aside from the content (like a sidebar)

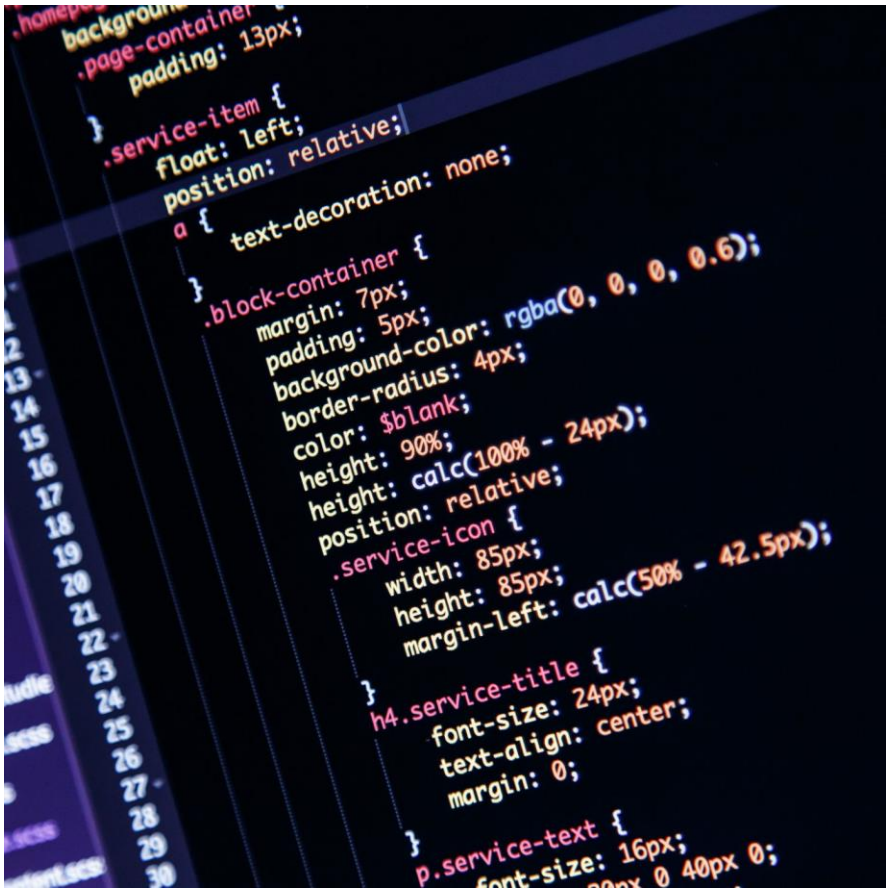
`<footer>` - Defines a footer for a document or a section



CSS. Non proprio un linguaggio di markup

- Il **CSS** (sigla di **Cascading Style Sheets**, in italiano **fogli di stile a cascata**), è un linguaggio usato per definire la formattazione e impaginazione di documenti HTML (ma anche XHTML e XML)
- Le regole per comporre il CSS sono contenute in un insieme di direttive (*Recommendations*) emanate a partire dal 1996 dal W3C.
- L'introduzione del CSS si è resa necessaria per **separare i contenuti delle pagine HTML dalla loro formattazione o**





Javascript

- Un linguaggio di **programmazione (non di markup!)** per rendere dinamici alcuni effetti sul layout e attivare certi comportamenti sui contenuti della pagina Web
- Impareremo solo a **copiare script** e **incollarli nella pagina** o richiamarli dal file HTML
- A seconda del tipo di script, a volte si richiama il file completo dall'<head> (file .js esterno) o lo si mette nell'<head>, o a volte nel <body>, sempre fra marcatore <script> di apertura e </script> di chiusura e si utilizza a volte richiamandolo come valore di alcuni speciali attributi (e.g. onclick); a volte si mette una parte nell'<head> (funzione) e una parte nel <body> (evento)

My First JavaScript

```
<!DOCTYPE html>
<html>
<body>

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo')
.innerHTML = Date()">
Click me to display Date and Time.
</button>

<p id="demo"></p>

</body>
</html>
```

Per chi vuole saperne di più, alcuni esempi precompilati dal sito W3Schools: https://www.w3schools.com/js/js_examples.asp

Parte pratica

Questo è quanto ci servirà per avviare il Laboratorio:

Verifichiamo di aver SublimeText 3 o di usare la versione *portable* (https://www.sublimetext.com/)

Per HTML5 si può fare riferimento a:
https://www.w3schools.com/html/

Per CSS:
https://www.w3schools.com/css/default.asp

Iniziamo con un semplice esercizio e poi continueremo con un Laboratorio ad hoc di 10 ore.