



Tecnologie Web T

Il linguaggio HTML

Su Virtuale:

Versione 1 pagina per foglio = 1.04.HTML.pdf

Versione 2 pagine per foglio = 1.04.HTML-2p.pdf

WWW = URL + HTTP + HTML

- **HTML** è l'acronimo di **HyperText Markup Language**
- È il linguaggio utilizzato per descrivere le pagine che costituiscono i nodi dell'**ipertesto**
- È un **linguaggio di codifica del testo** del tipo a **marcatori** (markup)
- Un **linguaggio di codifica del testo** è un formalismo con cui rappresentare un documento su supporto digitale in modo che sia trattabile dall'elaboratore in quanto testo

Sistemi di codifica caratteri

- I formalismi più elementari per la codifica informatica del testo sono i **sistemi di codifica dei caratteri**
- In generale, ogni documento elettronico è costituito da una stringa di caratteri
- Come qualsiasi altro tipo di dato, anche i caratteri vengono rappresentati all'interno di un elaboratore mediante **codifica binaria**
- Per codificare i caratteri si stabilisce una **corrispondenza biunivoca tra** elementi di una collezione ordinata di **caratteri** e un insieme di **codici numerici**
- Si ottiene così un ***coded character set*** che di solito si rappresenta in forma di tabella (***code page*** o ***code table***)

Sistemi di codifica caratteri

- Per ciascun coded character set si definisce una **codifica dei caratteri (character encoding)**
- La codifica mappa una o più sequenze di byte (8 bit) a un **numero intero** che rappresenta un carattere in un determinato coded character set
- *Il numero di caratteri rappresentabili in un certo coded character set è determinato dal numero di bit utilizzati per codificare ogni singolo carattere*
- I più noti sono:
 - ASCII (7 bit)
 - Famiglia ISO 8859/ANSI (8 bit)
 - Unicode (8, 16 o 32 bit: **UTF-8**, UTF-16 e UTF-32)

Standard che sta prendendo piede come successore di ASCII, specie da quando è diventato la codifica principale di **Unicode** per Internet da parte del consorzio **W3C**

Linguaggi a marcatori

- La codifica dei caratteri non esaurisce i problemi di rappresentazione di un testo
- Un **testo** è un **oggetto complesso** caratterizzato da **molteplici livelli strutturali** che non si limitano alla sequenza di simboli del sistema di scrittura
- Si parla propriamente di **linguaggio di codifica testuale** solo in riferimento ai linguaggi che consentono la **rappresentazione o il controllo di uno o più livelli strutturali di un documento testuale**
 - Tali linguaggi vengono correntemente denominati **linguaggi a marcatori** (*mark-up language*)

Caratteristiche dei linguaggi a marcatori

- Un **linguaggio di mark-up** è composto da:
 - un **insieme di istruzioni** dette **tag** o **mark-up** (**marcatori**) che rappresentano le caratteristiche del documento testuale
 - una **grammatica** che regola l'uso del mark-up
 - una **semantica** che definisce il dominio di applicazione e la funzione del mark-up
- I marcatori vengono inseriti direttamente all'interno del testo cui viene applicato
- *Ogni tag è a sua volta costituito da sequenza di caratteri, preceduta da caratteri speciali che la delimitano e che permettono di distinguere facilmente il testo dai marcatori*

Classificazione

- Tradizionalmente i linguaggi di mark-up sono stati divisi in due tipologie:
 - **linguaggi procedurali o imperativi**
 - **linguaggi dichiarativi o descrittivi**
- Questa classificazione risale a Charles Goldfarb, il padre di Standard Generalized Mark-up Language (SGML)
- Nei **linguaggi procedurali** il mark-up specifica quali operazioni un dato programma deve compiere su un documento elettronico per ottenere una determinata presentazione (Tex, LaTeX)
- Nei **linguaggi dichiarativi** il mark-up descrive la struttura di un documento testuale identificandone i componenti (SGML, HTML, XML)

Linguaggi dichiarativi

In particolare viene descritta la struttura editoriale, costituita da componenti (**content object**) organizzati in modo gerarchico

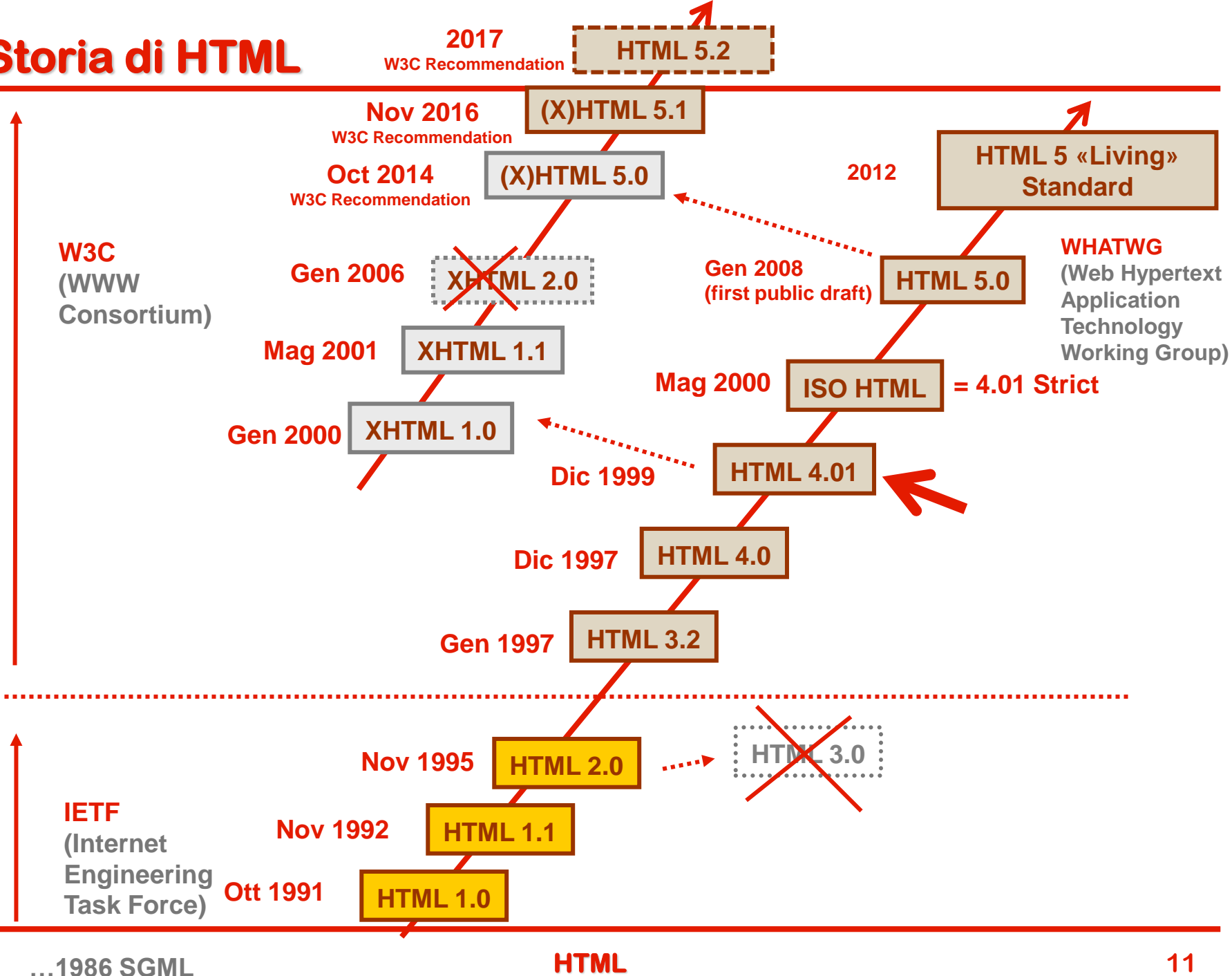
- *Frontespizio, introduzione, corpo, appendice...*
- *Capitoli, sottocapitoli, atti, scene, canti...*
- *Titoli, epigrafi, abstract...*
- *Paragrafi, versi, battute, entrate di dizionario...*
- *Enfasi, citazioni...*

- **SGML** = **S**tandard **G**eneralized **M**arkup **L**anguage
- È uno standard ISO (8879) pubblicato nel 1986
- È un meccanismo flessibile e portabile per rappresentare documenti elettronici proposto da *Charles Goldfarb*
- Un documento SGML comprende oggetti di varie classi chiamati **elementi**
 - capitoli, titoli, riferimenti, oggetti grafici, etc.
- SGML identifica gli estremi degli elementi tramite **tag iniziali** e **tag finali**
- *Non contiene sequenze di istruzioni di formattazione*
- **Gli elementi sono organizzati in una gerarchia**
 - un capitolo contiene un titolo ed una o più sezioni che a loro volta contengono altri elementi, ecc.

HTML e SGML

- HTML è un'**applicazione/semplicificazione di SGML**, ovvero un linguaggio per la rappresentazione di un tipo di documento SGML
- Ideato da *Tim Berners-Lee* a inizio '90
- Tramite HTML è possibile realizzare documenti con una struttura semplice contenenti:
 - testo, immagini e contenuti multimediali, oggetti interattivi e connessioni ipertestuali ad altri documenti
- Oltre a descrivere il contenuto, ***HTML associa anche significati grafici agli elementi che definisce!***
 - istruzioni più o meno precise su come rendere graficamente gli elementi che definisce
- Come vedremo questa commistione con elementi di formattazione crea diversi problemi...

Storia di HTML





- Proposta nata «ufficialmente» nel 2007 dal *Web Hypertext Application Technology Working Group* (**WHATWG**), formato inizialmente da Apple, Mozilla e Opera
 - **HTML5** si pone come **alternativa** al progetto di standardizzazione di **XHTML 2** proposto da W3C
- In reazione a questa iniziativa, il **gruppo W3C decide nel 2006 di abbandonare il progetto «XHTML 2» e di collaborare assieme a WHATWG al progetto congiunto di standardizzazione di HTML5**
 - primo draft di specifica «HTML 5» da parte di WHATWG nel 2008
- Tuttavia, **WHATWG e W3C hanno punti di vista diametralmente opposti** nei confronti del progetto e degli obiettivi da raggiungere...
- La collaborazione avrà presto termine (nel **2012**)



- **WHATWG** desiderava sviluppare il linguaggio **HTML** come “**Living Standard**”, una specifica perennemente in sviluppo
<http://www.whatwg.org/specs/web-apps/current-work/>
- Questa «instabilità» diventa la «caratteristica» che contraddistingue il linguaggio
 - Uno standard “living” si contraddistingue per essere **sempre aggiornato** e in **continua evoluzione**
 - Nuove feature possono essere aggiunte MA le vecchie funzionalità non possono essere rimosse
- **HTML5 Living Standard** (o più semplicemente **HTML**) è stato pubblicato nel **2012** da **WHATWG** ed è in continua evoluzione



- **W3C** desiderava invece sviluppare uno **standard HTML5 e XHTML definitivo**
- Lavora autonomamente giungendo alla pubblicazione
 - “standard” **HTML 5** («W3C Recommendation») il 28 ottobre **2014**
 - versione **HTML 5.1** («W3C Recommendation») il 1 novembre **2016**
 - versione **HTML 5.2** («W3C Recommendation») a fine **2017**
 - era in fase di definizione la versione **HTML 5.3** (2021, poi ritirata), ora si preme di nuovo maggiormente sulla idea di living standard (ultima edizione 28 settembre 2022)

HTML5: relazioni con HTML 4.01

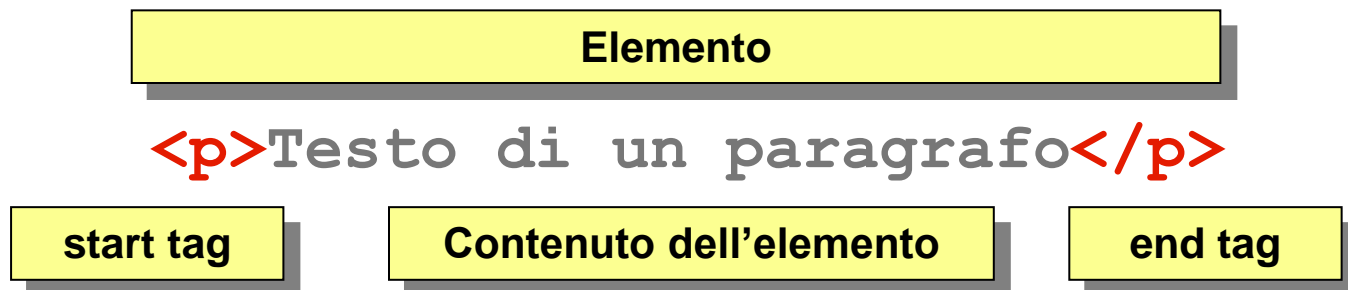
- HTML 4.01 è incluso in HTML 5!
- Molto di ciò che il linguaggio HTML 5 offre rispetto a HTML 4.01 è **opzionale**:
 - È quindi possibile utilizzare HTML 5, programmando con i costrutti HTML 4.01 e scegliendo le sole «feature» HTML 5 di cui si ha bisogno
- Questa **ortogonalità** dei «nuovi costrutti HTML 5» favorisce la proliferazione di documenti HTML 5 «100%»
 - in cui però la stragrande maggioranza della sintassi è puro HTML 4.01! 😊
 - A garanzia di piena retro compatibilità con l'esistente (hardware/software)

HTML5: partiamo da HTML 4.01

- Nel seguito faremo largamente riferimento allo **standard HTML 4.01** per poi trattare **HTML 5** nelle sue peculiarità principali
- HTML 4.01 risulta infatti ancora il linguaggio con cui sono scritte la maggioranza delle pagine Web attuali (assieme a XHTML di cui parleremo in seguito)
- Prevede tre varianti:
 - **Strict**, in cui gli elementi *deprecati* sono *vietati*
 - **Transitional**, in cui gli elementi *deprecati* sono *ammessi*
 - **Frameset**, in cui sono ammessi anche i *frame* e gli *elementi collegati*
- Nel maggio 2000 l'**HTML 4.01 Strict** è diventato **standard ISO/IEC** con il codice **15445:2000**

Tag

- I **tag HTML** sono usati per definire il **mark-up** di **elementi HTML**
- Sono preceduti e seguiti rispettivamente da due caratteri “<” e “>” (parentesi angolari)
- Sono normalmente accoppiati; un esempio è dato da: **<p>** e **</p>**, detti rispettivamente **start tag** ed **end tag**
- Il testo tra start tag ed end tag è detto **contenuto dell'elemento**
- Un documento HTML contiene quindi elementi composti da testo semplice delimitato da tag:



Grammatica poco rigorosa

- HTML rispetta in maniera poco rigorosa le specifiche SGML
 - Ammette elementi senza chiusura come `
`
 - I tag non sono case sensitive
 - L'apertura e chiusura di tag annidati può essere "incrociata"
`<i>Testo corsivo grassetto</i>`
- Esistono però delle buone pratiche che è bene rispettare e che diventano un obbligo in una versione più rigorosa del linguaggio chiamata **XHTML**
 - Chiudere sempre anche i tag singoli:
`
</br>` o in forma sintetica `
`
 - Tag in minuscolo
 - Apertura e chiusura senza incroci (in teoria non ammessi ma tollerati) `<i>...</i>`

Entity

HTML definisce un certo numero di entità (**entity**) per rappresentare i caratteri speciali senza incorrere in problemi di codifica:

- Caratteri riservati a HTML (<, >, &, “, ecc.)
- Caratteri non presenti nell'ASCII a 7 bit

<code>&amp;</code>	<code>&</code>	<code>&quot;</code>	<code>“</code>
<code>&lt;</code>	<code><</code>	<code>&gt;</code>	<code>></code>
<code>&reg;</code>	<code>®</code>	<code>&nbsp;</code>	<code>(non-breaking space)</code>
<code>&Aelig;</code>	<code>Æ</code>	<code>&Aacute;</code>	<code>Á</code>
<code>&Agrave;</code>	<code>À</code>	<code>&Auml;</code>	<code>Ä</code>
<code>&aelig;</code>	<code>æ</code>	<code>&aacute;</code>	<code>á</code>
<code>&agrave;</code>	<code>à</code>	<code>&auml;</code>	<code>ä</code>
<code>&ccedil;</code>	<code>ç</code>	<code>&ntilde;</code>	<code>ñ</code>

Attributi

- Un elemento può essere dettagliato mediante **attributi**
- Gli attributi sono **coppie** “**nome = valore**” contenute nello start tag con una sintassi di questo tipo
`<tag attrib1='valore1' attrib2='valore2'>`
- I valori sono racchiusi da apici singoli o doppi

Visualizzazione

NAME Ok

- Esempio:
`<input type='submit' value="NAME">Ok</input>`
- *! Gli apici possono essere omessi se il valore non contiene spazi*
- I valori «colori» vengono espressi con un nome o in formato RGB con la sintassi #RRGGBB



Red = "#FF0000"



Black = "#000000"



Blue = "#0000FF"



Yellow = "#FFFF00"

Tipi MIME

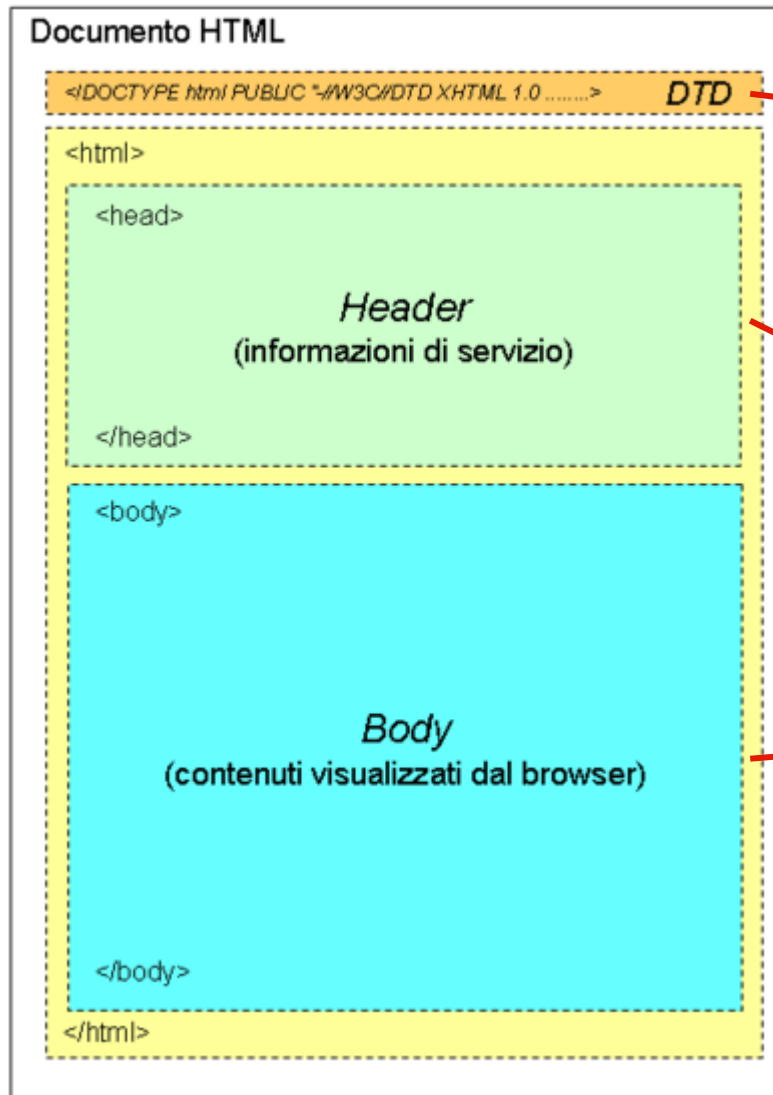
- Lo standard **MIME** (*Multipurpose Internet Mail Extensions*) è nato originariamente per poter allegare data file (audio, video, immagini, ...) ai messaggi di posta elettronica
- Oggi, noto anche come *Internet Media Type*, rappresenta il **tipo di contenuto di un messaggio** (es. HTTP request)
- Classifica i tipi di contenuto sulla base di una **logica a due livelli** ed è largamente utilizzata in ambito di HTML e delle tecnologie web in generale
- Un tipo MIME è espresso con questa sintassi:
tipo/sottotipo
- Esempi:
 - `text/plain`: testo semplice
 - `text/html`: testo HTML

Commenti

- È possibile inserire commenti in qualunque punto all'interno di una pagina HTML con la seguente sintassi:

`<!-- Questo è un testo di commento -->`

Struttura base di un documento HTML



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<html>
```

```
<head>  
  <title>Hello  
  document</title>  
</head>
```

```
<body>  
  Hello World!  
</body>
```

```
</html>
```

- Il primo elemento di un documento HTML è la definizione del tipo di documento (**Document Type Definition** o **DTD**):
- Serve al browser per identificare le **regole di interpretazione e visualizzazione** da applicare al documento
- Esempio: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>`
- È costituita da diverse parti:
 - **HTML** il tipo di linguaggio utilizzato è l'HTML
 - **PUBLIC** il documento è pubblico
 - - le specifiche non sono registrate all'ISO (altrimenti +)
 - **W3C** ente che ha rilasciato le specifiche
 - **DTD HTML 4.01 Transitional**: versione di HTML
 - **EN** la lingua con cui è scritta il DTD è l'inglese
 - **<http://...>** URL delle specifiche

Header

- È identificato dal tag **<head>**
- Contiene elementi non visualizzati dal browser (informazioni di servizio)
- **<title>** titolo della pagina (viene mostrato nella testata della finestra principale del browser)
- **<meta>** metadati informazioni utili ad applicazioni esterne (es. motori di ricerca) o al browser (es. lingua, codifica dei caratteri utile per la visualizzazione di alfabeti non latini)
- **<base>** definisce come vengono gestiti i riferimenti relativi nei link
- **<link>** collegamenti verso file esterni: CSS, script, icone visualizzabili nella barra degli indirizzi del browser
- **<script>** codice eseguibile utilizzato dal documento
- **<style>** informazioni di stile (CSS locali)

Elementi <meta>

- Gli elementi di tipo **<meta>** sono caratterizzati da una serie di attributi
- Esistono due tipi di elementi meta, distinguibili dal primo attributo: **http-equiv** o **name**
- Gli elementi di tipo **http-equiv** danno informazioni al browser su come gestire la pagina
- Hanno una struttura di questo tipo:
<meta http-equiv=nome content=valore>
- Gli elementi di tipo **name** forniscono informazioni utili ma non critiche
- Hanno una struttura di questo tipo:
<meta name=nome content=valore>

Elementi <meta> di tipo http-equiv

- **refresh**: indica che la pagina deve essere ricaricata dopo un numero di secondi definito dall'attributo content
`<meta http-equiv=refresh content=45>`
- **expires**: stabilisce una data scadenza (fine validità) per il documento
`<meta http-equiv=expires content="Tue, 20 Aug 1996 14:25:27 GMT">`
- **content type**: definisce il tipo di dati contenuto nella pagina (di solito il tipo MIME text/html):
`<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">`

Elementi <meta> di tipo name

- **author:** autore della pagina:
`<meta name=author content='John Smith' >`
- **description:** descrizione della pagina
`<meta name=description content="Home page UNIBO">`
- **copyright:** indica che la pagina è protetta da un diritto d'autore
`<meta name=copyright content="Copyright 2009, John Smith">`
- **keywords:** lista di parole chiave separate da virgole, usate dai motori di ricerca
`<meta name=keywords lang="en" content="computer documentation, computers, computer help">`
- **date:** data di creazione del documento
`<meta name="date" content="2008-05-07T09:10:56+00:00">`

Esempio di header

<head>

```
<meta http-equiv="Content-Type"  
  content="text/html;  
  charset=iso-8859-1">
```

```
<meta name="description"  
  content="Documentation about HTML">
```

```
<meta name="keywords"  
  content="HTML, tags, commands">
```

```
<title>Impariamo l'HTML</title>
```

```
<link href="style.css"  
  rel=stylesheet type="text/css">
```

</head>

Body

- Il tag **<body>** delimita il corpo del documento
- Contiene la parte che viene mostrata dal browser
- Ammette diversi attributi tra cui:
 - **background** = *uri*
Definisce l'URI di una immagine da usare come sfondo per la pagina
 - **text** = *color*
Definisce il colore del testo
 - **bgcolor** = *color*
In alternativa a background definisce il colore di sfondo della pagina
 - **lang** = *linguaggio*
definisce il linguaggio utilizzato nella pagina
es. **language="it"**

Un esempio di BODY

```
<body>
```

```
<h1>Titolo</h1>
```

```
<p>Questo è un  
paragrafo completo di un  
documento.</p>
```

```
<p>Un altro  
paragrafo<br>con un a  
capo</p>
```

```
<hr>
```

```
<p>Esempio di lista  
puntata, la lista della  
spesa:</p>
```

```
<ul>
```

```
<li>Pane</li>
```

```
<li>Latte</li>
```

```
<li>Prosciutto</li>
```

```
<li>Formaggio</li>
```

```
</ul>
```

```
</body>
```

Visualizzazione

Titolo

Questo è un paragrafo completo di un documento.

Un altro paragrafo
con un a capo

Esempio di lista puntata, la lista della spesa:

- Pane
- Latte
- Prosciutto
- Formaggio

Tipi di elementi del body

- **Intestazioni:** titoli organizzati in gerarchia
- **Strutture di testo:** paragrafi, testo indentato, ecc.
- **Aspetto del testo:** grassetto, corsivo, ecc.
- **Elenchi e liste:** numerati, puntati
- **Tabelle**
- **Form** (moduli elettronici): campi di inserimento, checkbox e radio button, menu a tendina, bottoni, ecc.
- **Collegamenti ipertestuali e ancore**
- **Immagini e contenuti multimediali** (audio, video, animazioni, ecc.)
- **Contenuti interattivi:** script, applicazioni esterne

Elementi blocco, elementi inline e liste

- Dal punto di vista del layout della pagina gli elementi HTML si dividono in 3 grandi categorie:
 - **Elementi “block-level”**: costituiscono un blocco attorno a sé, e di conseguenza «vanno a capo» (paragrafi, tabelle, form...)
 - **Elementi “inline”**: non vanno a capo e possono essere integrati nel testo (link, immagini,...)
 - **Liste**: numerate, puntate
- **Regole di composizione**:
 - Un elemento block-level può contenere altri elementi dello stesso tipo o di tipo inline
 - Un elemento inline può contenere solo altri elementi inline

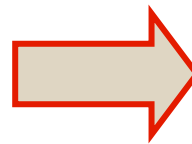
Elementi rimpiazzati e non rimpiazzati

- Un'altra distinzione da ricordare è quella tra **elementi rimpiazzati** (**replaced elements**) ed elementi **non rimpiazzati**
- Gli elementi **rimpiazzati** sono quelli di cui il browser conosce le dimensioni intrinseche
 - Sono quelli in cui **altezza e larghezza sono definite dall'elemento stesso e non da ciò che lo circonda**
 - L'esempio più tipico di elemento rimpiazzato è ****
 - Altri elementi rimpiazzati sono: **<input>**, **<textarea>**, **<select>**
- Tutti gli altri elementi sono in genere considerati non rimpiazzati

Heading

- I tag `<h1>`, `<h2>` ... `<h6>` servono per definire dei titoli di importanza decrescente (`<h1>` è il più importante)
- La "h" sta per "heading", cioè **titolo** e sono previste 6 grandezze; i titoli appaiono in grassetto e lasciano una riga vuota prima e dopo di sé (sono elementi di blocco)
- Ammettono attributi di allineamento:
`<h1 align = left|center|right|justify>`

```
<h1>titolo 1 </h1>
<h2>titolo 2 </h2>
<h3>titolo 3 </h3>
<h4>titolo 4 </h4>
<h5>titolo 5 </h5>
<h6>titolo 6 </h6>
```



titolo 1

titolo 2

titolo 3

titolo 4

titolo 5

titolo 6

Contenitori di testo: paragrafi

- Il **paragrafo** è l'unità di base entro cui suddividere un testo: è un elemento di tipo blocco
- Il tag **<p>** lascia una riga vuota prima della sua apertura e dopo la sua chiusura
- Se si vuole **andar a capo** all'interno di un paragrafo si usa l'elemento **
**
- Esempio: due paragrafi
<p>paragrafo 1</p>
<p>paragrafo 2</p>
- Vengono visualizzati così:

paragrafo1

paragrafo2

Esempio di paragrafi

`<p>Il romanzo si apre con il famoso incipit che introduce una realistica e minuziosa descrizione dell'ambiente in cui si svolgono i fatti:</p>
<p>Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte;
 e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni.</p>`

Il romanzo si apre con il famoso incipit che introduce una realistica e minuziosa descrizione dell'ambiente in cui si svolgono i fatti:

Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte;
e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni.

Allineamento

- È possibile definire l'allineamento di un paragrafo mediante l'attributo **align**

- Abbiamo 4 valori:

align = left

align = center

align = right

align = justify

Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a restringersi, e a prender corso e figura di fiume

Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a restringersi, e a prender corso e figura di fiume

Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a restringersi, e a prender corso e figura di fiume

Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a restringersi, e a prender corso e figura di fiume

- Se al posto di <p> si usa il tag **<div>** il blocco di testo va a capo, ma - a differenza del paragrafo - non lascia spazi prima e dopo la sua apertura
- *! È l'elemento di tipo blocco per eccellenza*
- Esempio: due <div>
<div>Blocco di testo 1</div>
<div>Blocco di testo 2</div>
- Vengono visualizzati così:
Blocco di testo 1
Blocco di testo 2

Span

- Lo **** è un **contenitore generico** che può essere annidato (ad esempio) all'interno dei **<div>**
- È un elemento **inline**, e quindi non va a capo ma continua sulla stessa linea del tag che lo include
- Esempio: due ****
Contenitore 1Contenitore 2
- Visualizzazione:
Contenitore 1Contenitore 2
- È un elemento molto utilizzato soprattutto insieme ai fogli di stile per dare un **aspetto particolare ad un pezzo di testo** in un blocco (per esempio, per evidenziare)
- *! Se non viene associato ad uno stile è invisibile*

Contenitori di testo: riepilogo

- Ricapitolando: `<p>`, `<div>` e `` sono tre diversi tipi di contenitori di testo
- Si comportano in modo diverso:
 - `<p>` è un elemento di blocco e lascia spazio prima e dopo la propria chiusura
 - `<div>` è un elemento di blocco, non lascia spazio prima e dopo la propria chiusura, ma va a capo
 - `` è un elemento inline e quindi non va a capo

Horizontal rule

- Il tag **<hr>** serve ad inserire una riga di separazione
- Attributi:
 - **align** = {*left / center / right*}
Allineamento della riga rispetto a ciò che la circonda
 - **size** = *pixels*
Altezza della riga
 - **width** = *length*
Larghezza della riga in modo assoluto o in percentuale delle dimensioni di ciò che la contiene
 - **noshade**
Riga senza effetto di ombreggiatura

```
<hr width="50%" align="center">  
<hr size="5" width="50%" align="center">  
<hr noshade size="5" width="50%" align="center">
```

Gli stili del testo

- Nella terminologia tipografica lo "stile di un testo" indica le possibili varianti di forma di un carattere: tondo (normale), neretto (grassetto), corsivo
- HTML consente di definire lo stile di un frammento di testo, combinando fra loro anche più stili
- I tag che svolgono questa funzione vengono normalmente suddivisi in **fisici** e **logici**:
 - **Tag fisici**: definiscono lo stile del carattere in termini grafici indipendentemente dalla funzione del testo nel documento
 - **Tag logici**: forniscono informazioni sul ruolo svolto dal contenuto, e in base a questo adottano uno stile grafico

Tag fisici

- `<tt>...</tt>` Carattere monospaziato
- `<i>...</i>` Corsivo
- `...` Grassetto
- `<u>...</u>` Sottolineato (deprecato)
- `<s>...</s>` Testo barrato (deprecato)

<code><tt>monospaced text</tt></code>	→	monospaced text
<code><i>italic text</i></code>	→	<i>italic text</i>
<code>bold text</code>	→	bold text
<code><u>underlined text</u></code>	→	<u>underlined text</u>
<code><s>stroke</s></code>	→	stroke

Tag logici

- `` Usualmente visualizzato in **grassetto**
- `` (emphasis) Usualmente visualizzato in **corsivo**
- `<code>/<pre>` **Codice**: usualmente monospaziato
- `<kbd>` **Keyboard**: monospaziato come code
- `<abbr>` **Abbreviazione** (nessun effetto)
- `<acronym>` **Acronimo** (nessun effetto)
- `<address>` **Indirizzo fisico o e-mail**: in corsivo
- `<blockquote>` **Blocco di citazione**: rientrato a destra
- `<cite>` **Citazione**: visualizzato in corsivo

Font

- Il tag **** permette di formattare il testo, definendo **dimensioni, colore, tipo** di carattere
- È l'esempio limite del mescolamento fra contenuto e rappresentazione
- È deprecato in HTML 4.01
- Attributi:
 - **size** = **[+|-]n**
Definisce le dimensioni del testo (1-7 o relative)
 - **color** = *color*
Definisce il colore del testo
 - **face** = *text*
Definisce il font del testo

Liste non ordinate

- Il tag **** (unordered list) permette di definire liste non ordinate (puntate)
- Gli elementi della lista vengono definiti mediante il tag **** (list item)
- L'attributo **type** definisce la forma dei punti e ammette 3 valori: disc, circle e square

```
<ul type="disc">  
  <li>Unordered information.</li>  
  <li>Ordered information.</li>  
  <li>Definitions.</li>  
</ul>
```

- Unordered information.
- Ordered information.
- Definitions.

Liste ordinate

- Il tag **** (ordered list) permette di definire liste ordinate (numerati)
- Gli elementi vengono definiti mediante il tag ****
- L'attributo **type** definisce il tipo di numerazione e ammette 5 valori: **1** (1,2,..), **a** (a,b,..), **A** (A,B,..), **i** (i,ii,..) e **I** (I,II,..)

```
<ol type="I">  
  <li>Unordered information.</li>  
  <li>Ordered information.</li>  
  <li>Definitions.</li>  
</ol>
```

```
I.    Unordered information.  
II.   Ordered information.  
III.  Definitions.
```


Liste di definizione

- Il tag **<dl>** (definition list) permette di definire liste di definizione
- Sono liste costituite alternativamente da **termini** (tag **<dt>**) e **definizioni** (tag **<dd>**)

```
<dl>
  <dt><strong>UL</strong></dt>
  <dd>Unordered List.</dd>
  <dt><strong>OL</strong></dt>
  <dd>Ordered List.</dd>
</dl>
```

UL

Unordered List.

OL

Ordered List.

Tabelle

```
<table border="1" >
  <caption align="top">
    <em>A test table with merged cells</em></caption>
  <tr>
    <th rowspan="2"></th>
    <th colspan="2">Average</th>
    <th rowspan="2">Red<br/>eyes</th>
  </tr>
  <tr><th>height</th><th>weight</th></tr>
  <tr><th>Males</th><td>1.9</td><td>0.003</td><td>40%</td></tr>
  <tr><th>Females</th><td>1.7</td><td>0.002</td><td>43%</td></tr>
</table>
```

A test table with merged cells

	Average		Red eyes
	height	weight	
Males	1.9	0.003	40%
Females	1.7	0.002	43%

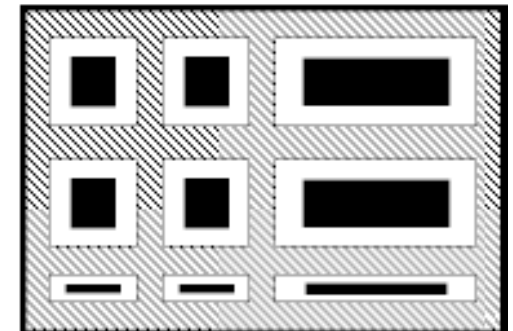
<table>

- Il tag **<table>** racchiude la tabella
- Attributi:
 - align** = "{left|center|right}"
allineamento della tabella rispetto alla pagina;
 - width**="n|n%"
larghezza della tabella (anche in percentuale rispetto alla pagina);
 - bgcolor**="#xxxxxx"
colore di sfondo della tabella;
 - border**="n"
spessore dei bordi della tabella (0 = tabella senza bordi);
 - cellspacing**, **cellpadding**

Cellspacing 

Cellpadding 

Cell content 



Righe

- **<tr>** è il tag che racchiude ciascuna riga della tabella
- Attributi:
 - **align** = “{left|center|right|justify}”
allineamento del contenuto delle celle della riga;
 - **valign** = “{top|middle|bottom|baseline}”
allineamento verticale del contenuto delle celle della riga;
 - **bgcolor**=“#xxxxxx”
colore di sfondo della riga

Testate e celle

- **<th>** e **<td>** sono i tag che racchiudono le celle
 - **<th>** serve per le celle della testata
 - **<td>** serve per le celle del contenuto
- Attributi:
 - Gli stessi di **<tr>**
 - **width, height** = {length|length%}
specifica le dimensioni (larghezza e altezza) della cella, dimensione assoluta (pixels) o valore percentuale;
 - **rowspan, colspan** = n
indica su quante righe/colonne della tabella si estende la cella

Tabelle e layout

- Le tabelle sono nate sostanzialmente per organizzare dati in modo ordinato; nel tempo si sono rivelate uno strumento indispensabile per definire layout grafici complessi
 - Permettono di costruire griglie in cui inserire i contenuti di un sito e per mezzo degli sfondi e dei margini è possibile riprodurre un'impostazione accattivante
 - Permettono di realizzare i cosiddetti layout “liquidi”, che si adattano cioè alla risoluzione del monitor dell'utente (grazie all'uso delle dimensioni in %)
- **La tendenza attuale è quella di superare questa tecnica, che presenta alcuni inconvenienti**
 - Mischia elementi di formattazione dei dati ai dati stessi
 - Appesantisce le pagine con molti elementi, rallentando lo scaricamento
- ***! Siamo comunque in una fase di transizione e l'impaginazione a tabelle è ancora molto, molto usata***

Link ipertestuali

- Il **link** è il costrutto di base di un ipertesto
- Caratterizza HTML come **linguaggio a marcatori per la descrizione di ipertesti**
- È una connessione fra una risorsa Web ed un'altra
- Un link è costituito da due estremi - detti **àncore** (**anchor**) - e da una direzione di percorrenza

Link = source anchor → destination anchor

- L'**àncora di origine** (**source anchor**) è un elemento contenuto nella pagina di partenza
- L'**àncora di destinazione** (**destination anchor**) è una qualsiasi risorsa web (un'immagine, un video, un eseguibile, un documento HTML o un elemento interno al documento)
- La **risorsa di destinazione** si ottiene «visitando» il link

Ancore

- In HTML le ancore, sia di origine che di destinazione, si esprimono utilizzando il tag **<a>**
- Le **àncore di origine** sono caratterizzate da un attributo, denominato **href**, che contiene l'indirizzo di destinazione (è un **URL**)
- Le **àncore di destinazione** sono invece caratterizzate dall'attributo **name**
- L'esempio più semplice di link è quello che collega due elementi all'interno di uno stesso documento
 - In questo caso l'attributo **href** dell'àncora di origine ha la forma ***#nome***
 - ***nome*** è il valore dell'attributo **name** dell'àncora di destinazione
 - Un elemento **#xxxx** posto alla fine di un URL viene chiamato **fragment**

Esempio di link all'interno di un documento

```
<p>  
  <a href="#section1">  
    Introduzione</a><br>  
  <a href="#section2">  
    Concetti di base</a><br>  
  <a href="#section2.1">  
    Definizione del problema</a><br>  
    ...  
</p>  
  
<h2><a name="section1">  
  Introduzione</a></h2>  
  ...sezione 1...  
  
<h2><a name="section2">  
  Concetti di base</a></h2>  
  ...sezione 2...  
  
<h3><a name="section2.1">  
  Definizione del problema</a></h3>  
  ...sezione 2.1...
```



Sommario

[Introduzione](#)
[Concetti di base](#)
[Definizione del problema](#)
...

Introduzione

...sezione 1...

Concetti di base

...sezione 2...

Definizione del problema

...sezione 2.1...

Ancore “implicite”

- Si può esprimere un'ancora di destinazione in forma “implicita”, cioè senza utilizzare il tag `<a>`
- È sufficiente assegnare l'attributo **ID** a un qualunque elemento della pagina
- È una forma più compatta anche se probabilmente meno facile da interpretare

Esempio con ancore di destinazione implicite

```
<p>  
  <a href="#section1">  
    Introduzione</a><br>  
  <a href="#section2">  
    Concetti di base</a><br>  
  <a href="#section2.1">  
    Definizione del problema</a><br>  
    ...  
</p>  
  
<h2 id="section1">  
  Introduzione</h2>  
  ...sezione 1...  
  
<h2 id="section2">  
  Concetti di base</h2>  
  ...sezione 2...  
  
<h3 id="section2.1">  
  Definizione del problema</h3>  
  ...sezione 2.1...
```

Sommario

[Introduzione](#)
[Concetti di base](#)
[Definizione del problema](#)
...

Introduzione

...sezione 1...

Concetti di base

...sezione 2...

Definizione del problema

...sezione 2.1...

Link a risorsa esterna

- Il caso più comune è quello di un link ad un **altro documento** (pagina HTML) o in generale ad un'altra **risorsa** (es. un'immagine)
- In questo caso il primo link non specifica un'ancora e quindi si “salta” all'inizio del documento **chapter2.html**

```
<body>
...
<p>Per maggiori informazioni leggete il
<a href="chapter2.html">capitolo 2</a>.
Guardate anche questa
<a href=" ../images/forest.gif">mapa della foresta
incantata.</a></p>
...
</body>
```

```
...
Per maggiori informazioni leggete il capitolo 2. Guardate anche questa mapa della foresta incantata.
...
```

Link completo

Il caso più completo è quello di un link ad un punto preciso di un documento (**àncora di destinazione**)

```
...  
<p>Per maggiori informazioni leggete il  
<a href="chapter2.html#section2 ">  
secondo paragrafo del capitolo 2</a>.  
Guardate anche questa  
<a href=" ../images/forest.gif">  
mappa della foresta incantata.</a></p>  
...
```

chapter1.html

```
...  
<h1>Capitolo 2</h1>  
<h2><a name="section1">Paragrafo 1</a></h2>  
<p>Testo del primo paragrafo...</p>  
<h2><a name="section2">Paragrafo 2</a></h2>  
<p>Testo del secondo paragrafo...</p>  
...
```

chapter2.html

URL relativi e assoluti

- Gli URL utilizzati nell'attributo HREF possono essere **assoluti** o **relativi**
- Se sono **relativi** si procede alla «risoluzione» utilizzando come base quella del documento corrente
*Se desidero un comportamento diverso, attivo l'attributo **<base>** dell'header*
- Per esempio, se l'URL completo del documento corrente è:
www.disi.unibo.it/docs/chapter2.html
e URL relativo messo in HREF è
chapter2.html
allora URL base sarà
www.disi.unibo.it/docs/
e la risoluzione porterà all'URL assoluto:
www.disi.unibo.it/docs/chapter2.html

Cosa succede quando si clicca su un'ancora di origine?

1. L'URL definito dall'attributo HREF viene «risolto»
2. Se è un URL HTTP, viene fatta una chiamata HTTP al server in cui si trova il documento;
chiamata di tipo GET per ottenere la risorsa descritta dall'URL
3. La pagina viene caricata e visualizzata dal browser
4. Se è stata definita anche la parte fragment (#xxxxxx) il browser si porta al punto della pagina specificato

Immagini

- Il tag **** consente di inserire immagini in un documento HTML con la sintassi:

- Attributi:
 - src = uri**
specifica l'indirizzo dell'immagine (required)
 - alt = text**
testo alternativo nel caso fosse impossibile visualizzare l'immagine
 - align = {bottom|middle|top|left|right}** (deprecato in HTML 4.01)
posizione dell'immagine rispetto al testo che la circonda
 - width,height = pixels**
larghezza e altezza dell'immagine in pixel
 - border = pixels** (deprecato in HTML 4.01)
spessore del bordo dell'immagine (0 = nessun bordo)

Form

- Un **form** (modulo) è una sezione di documento HTML che contiene **elementi di controllo** che l'utente può utilizzare per inserire dati o in generale per interagire
- I dati inseriti possono essere poi inoltrati al server dove un agente può processarli
- Gli elementi di controllo sono caratterizzati da un **valore iniziale** e da un **valore corrente**
- Gli **elementi di controllo** possono essere:
 - **Bottoni** di azione
 - **Checkbox** (caselle di spunta)
 - **Radio Button** (bottoni mutuamente esclusivi)
 - **Liste** di selezione (lista di opzioni)
 - **Caselle di inserimento di testo**
 - Oggetti nascosti (elementi valorizzati ma invisibili)

Il tag **<form>**

- Il tag **<form>** racchiude tutti gli elementi del modulo (è un elemento di tipo blocco)
- Attributi:
 - **action = uri**
URI dell'agente che riceverà i dati del form
 - **name = text**
specifica il nome del form
 - **method = {get|post}**
specifica il modo in cui i dati vengono inviati
 - **enctype = content-type**
se il metodo è POST specifica il content type usato per la codifica (encoding) dei dati contenuti nel form;
default application/x-www-form-urlencoded

```
<form action="http://site.com/bin/adduser" method="post">
  ...form contents...
</form>
```

Elementi input

- La maggior parte dei controlli viene definita mediante il tag **<input>**
- L'attributo **type** stabilisce il tipo di controllo
 - **text**: casella di testo monoriga
 - **password**: come text ma il testo non è leggibile (****)
 - **file**: controllo che consente di caricare un file
 - **checkbox**: casella di spunta
 - **radio**: radio button
 - **submit**: bottone per trasmettere il contenuto del form
 - **image**: bottone di submit sotto forma di immagine
 - **reset**: bottone che riporta tutti i campi al valore iniziale
 - **button**: bottone di azione
 - **hidden**: campo nascosto
- Tutti gli input possono essere disabilitati utilizzando l'attributo **disabled** nella forma **disabled = "disabled"**

Input text

- È un campo per l'inserimento di testo su una sola riga
- Attributi:
 - **name = text**
nome del controllo
 - **value = text**
eventuale valore iniziale
 - **size = n**
lunghezza del campo (numero di caratteri)
 - **maxlength = n**
massima lunghezza del testo (numero di caratteri)


```
<form action="http://site.com/bin/adduser" method="post">  
  <p>  
    Nome: <input type="text" name="firstname">  
  </p>  
</form>
```

Nome:

Input file

- Consente di fare l'upload di un file selezionandolo nel filesystem del client
- Attributi:
 - **name = text**
specifica il nome del controllo
 - **value = content-type**
lista di MIME types per l'upload
- Richiede una codifica particolare per il form (**multipart/form-data**) perché le informazioni trasmesse con il POST contengono tipologie di dati diverse: testo per i controlli normali, binario per il file da caricare

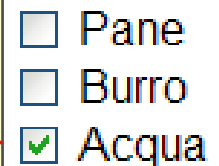
```
<form action="http://site.com/bin/adduser" method="post"
  enctype="multipart/form-data" >
  <p>
    <input type="file" name="attach">
  </p>
</form>
```



Checkbox

- Un input con tipo “checkbox” definisce una casella di spunta
- Attributi:
 - **name = text**
nome del controllo
 - **value = text**
valore restituito se la casella viene spuntata
 - **checked = “checked”**

```
<form action="http://site.com/bin/adduser" method="post">
  <p>
    <input type="checkbox" name="food" value="pane">
      Pane<br/>
    <input type="checkbox" name="food" value="burro">
      Burro<br/>
    <input type="checkbox" name="drink" value="acqua"
      checked="checked">
      Acqua<br/>
  </p>
</form>
```



Radio button

- Un radio button è una casella di spunta che serve per realizzare gruppi di scelta mutuamente esclusivi
- Tutti i controlli di questo tipo che condividono lo stesso nome sono esclusivi fra di loro
- Si può stabilire che un bottone è spuntato per default con l'attributo checked nella forma:

checked="checked"

```
<form action="http://site.com/bin/adduser" method="post">  
  <p>  
    <input type="radio" name="sex" value="M">Maschio<br/>  
    <input type="radio" name="sex" value="F">Femmina<br/>  
  </p>  
</form>
```

☐ Maschio
☐ Femmina

Bottoni

- Per i bottoni si utilizzano tre valori dell'attributo `type`
 - **submit** per il bottone che provoca la spedizione del form al server
 - **reset** per il bottone che riporta il contenuto dei campi al valore originale
 - **button** per un generico bottone di azione
- L'etichetta che compare nel bottone viene definita usando l'attributo **value**

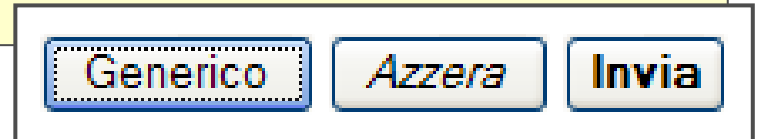
```
<form action="http://site.com/bin/adduser" method="post">
  <input type="submit" value="Conferma">&nbsp;&nbsp; 
  <input type="reset" value="Azzera">
</form>
```

Conferma

Il tag <button>

- In HTML 4 è stato introdotto il tag <button> che offre la possibilità di creare dei bottoni con un aspetto anche complesso
- Infatti <button> dà la possibilità di inserire il testo del bottone come contenuto del tag
- Questo consente di specificare anche codice HTML all'interno del tag: testo formattato ma anche immagini

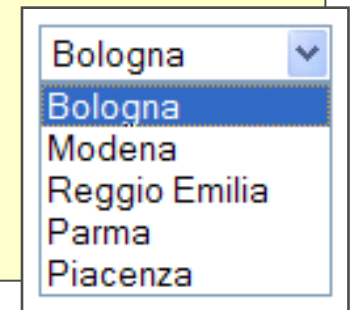
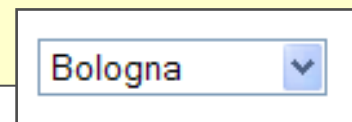
```
<form action="http://site.com/bin/adduser" method="post">  
  <button type="button">Generico</button>&nbsp;    
  <button type="reset"><i>Azzera</i></button>&nbsp;    
  <button type="submit"><b>Invia</b></button>  
</form>
```



Liste di opzioni

- Il tag **<select>** permette di costruire liste di opzioni
- Per definire le singole opzioni si usa il tag **<option>** ricorrendo all'attributo **value** si può attribuire il valore
- Con l'attributo **selected** si può indicare una scelta predefinita: **selected="selected"**
- L'aspetto di default è quello di un **combo box** (tendina a discesa)

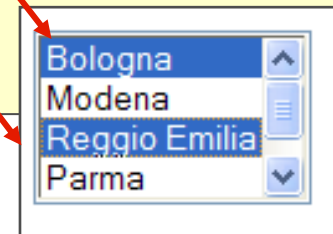
```
<form action="http://site.com/bin/adduser" method="post">  
  <select name="provincia" >  
    <option value="BO" selected="selected">Bologna</option>  
    <option value="MO">Modena</option>  
    <option value="RE">Reggio Emilia</option>  
    <option value="PR">Parma</option>  
    <option value="PC">Piacenza</option>  
  </select>  
</form>
```



Liste a scelta multipla

- Se si utilizza l'attributo `multiple` (nella forma `multiple="multiple"`) non abbiamo più un combo ma una lista sempre aperta
- Si può operare una scelta multipla tenendo premuto il tasto **[Ctrl]** durante la selezione
- L'attributo `size` determina il numero di righe mostrate

```
<form action="http://site.com/bin/adduser" method="post">  
  <select name="provincia" multiple="multiple">  
    <option value="BO" selected="selected">Bologna</option>  
    <option value="MO">Modena</option>  
    <option value="RE" selected="selected">Reggio Emilia</option>  
    <option value="PR">Parma</option>  
    <option value="PC">Piacenza</option>  
  </select>  
</form>
```



Gruppi di opzioni

- Con il tag **<optgroup>** è possibile organizzare la lista (sia a scelta singola che multipla) in gruppi
- Molto utile per liste lunghe

```
<form action="http://site.com/bin/adduser" method="post">
  <select name="provincia" multiple="multiple" size=7>
    <optgroup label="Capoluogo">
      <option value="BO" selected="selected">Bologna</option>
    </optgroup>
    <optgroup label="Emilia">
      <option value="MO">Modena</option>
      <option value="RE">Reggio Emilia</option>
      <option value="PR">Parma</option>
      <option value="PC">Piacenza</option>
    </optgroup>
  </select>
</form>
```

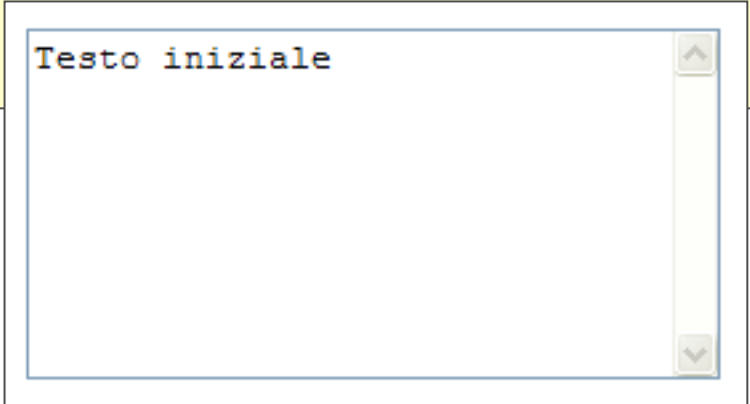


Capoluogo
Bologna
Emilia
Modena
Reggio Emilia
Parma
Piacenza

Textarea

- Il tag **<textarea>** consente di definire un campo di inserimento multiriga adatto a un testo lungo
- Il contenuto dell'elemento è il testo iniziale
- L'attributo **rows** indica il numero di righe della textarea,
cols il numero di caratteri (cioè di colonne) che ogni riga può contenere

```
<form action="http://site.com/bin/adduser" method="post">  
  <textarea name="testo" rows="8" cols="30">Testo iniziale  
  </textarea>  
</form>
```



Testo iniziale

Organizzare form complessi

Con il tag **<fieldset>** si possono creare gruppi di campi a cui è possibile attribuire un nome utilizzando il tag **<legend>**

```
<form action="http://site.com/bin/adduser" method="post">
<fieldset>
<legend>Nome e cognome</legend>
Nome: <input type="text" name="nome"><br>
Cognome: <input type="text" name="cognome">
</fieldset>
<fieldset>
<legend>Provincia</legend>
  <select name="provincia" multiple="multiple" size=7>
    <optgroup label="Capoluogo">
      <option value="BO" selected="selected">Bologna</option>
    </optgroup>
    <optgroup label="Emilia">
      <option value="MO">Modena</option>
      <option value="RE">Reggio Emilia</option>
      <option value="PR">Parma</option>
      <option value="PC">Piacenza</option>
    </optgroup>
  </select>
</fieldset>
</form>
```

Collegare le etichette ai controlli

Il tag **<label>** permette di associare un'etichetta ad un qualunque controllo di un form

- L'associazione può essere fatta in forma implicita inserendo il controllo nell'elemento label
- Oppure in forma esplicita tramite l'attributo **for** che deve corrispondere all'attributo **id** del controllo

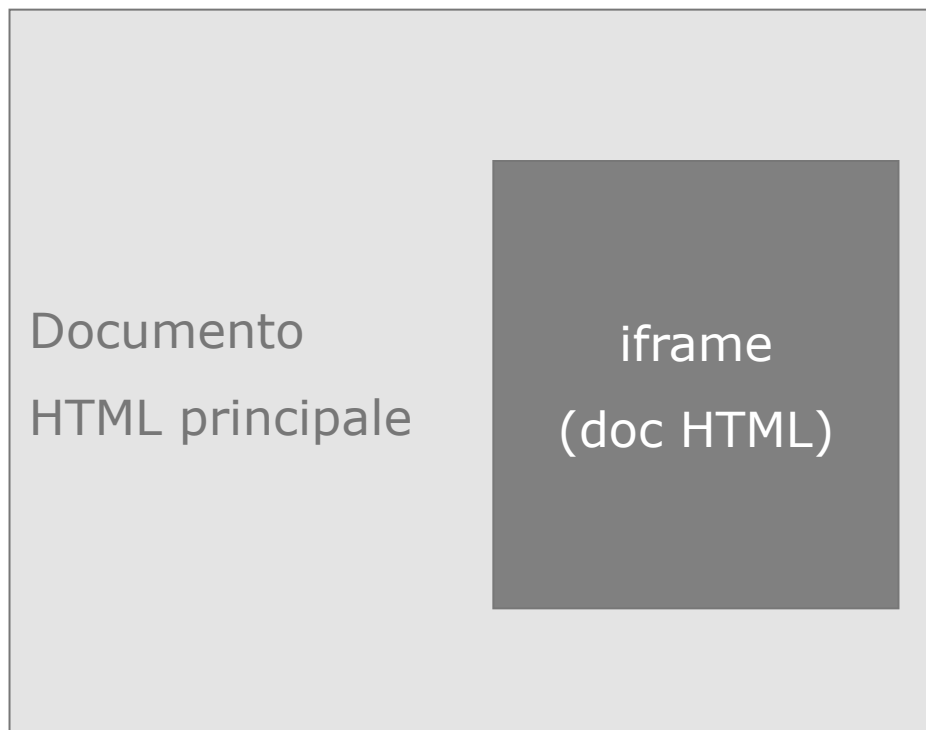
```
<form action="...">
  <label>Nome: <input type="text" id="nome"></label><br>
  <label>Cognome: <input type="text" id="cognome"></label><br>
</form>
```

```
<form action="...">
  <label for="nome">Nome: </label>
  <input type="text" id="nome"><br>
  <label for="cognome">Cognome: </label>
  <input type="text" id="cognome"><br>
</form>
```

Inline frames

- L'elemento **<iframe>** crea un frame inline che contiene un altro documento
- È deprecato in HTML 4.01 ma è ancora molto utilizzato (in certi casi indispensabile, es. applicazioni «cross domain»)

```
<iframe  
  src ="interno.html"  
  width="100%">  
</iframe>
```

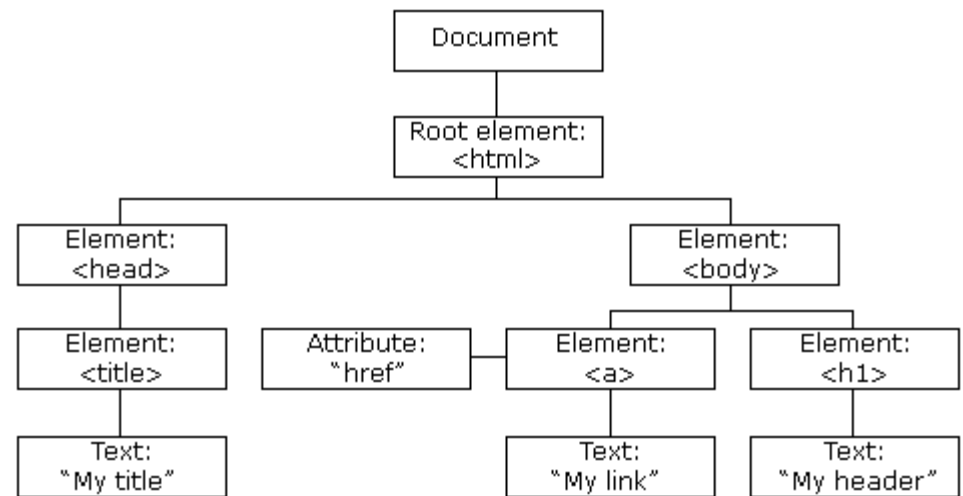


II DOM

- Una pagina HTML può essere rappresentata come una struttura ad albero
- Questa struttura logica prende il nome di **DOM**:
Document Object Model
 - ...analizzeremo dettagliatamente in seguito il comportamento di questo modello, anche collegato all'uso di eventi

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="...">MyLink</a>
    <h1>My header</h1>
  </body>
</html>
```

Testo HTML



DOM

!Quando un browser riceve una pagina HTML, ne fa il parsing e costruisce la struttura ad albero del DOM

- Come detto, molto di ciò che il linguaggio HTML 5 offre rispetto a HTML 4.01 è **opzionale**:
 - Possiamo dunque utilizzare il linguaggio HTML 5 programmando con costrutti HTML 4.01 e «attivando» le sole «feature» HTML 5 di cui abbiamo veramente bisogno
...ricordiamoci che HTML 4.01 è incluso in HTML 5!
- Questa **ortogonalità** dei «nuovi costrutti HTML 5» favorisce la proliferazione di documenti HTML 5 «100%»
 - nuova **formula semplificata** del doctype **<!DOCTYPE HTML>** in cui però la stragrande maggioranza della sintassi è puro HTML 4.01! 😊

HTML5: principali caratteristiche

- Tra le tante cose, HTML 5 estende notevolmente la possibilità di **integrazione di contenuti multimediali nella pagina** (embedded content)
- Elementi come **<audio>**, **<video>**, **<canvas>**, **<math>** permettono di includere contenuti con i quali, è possibile **interagire in modo avanzato**
 - *Questo è il vero punto di forza di HTML 5*
- Il modello ad eventi di **DOM** è esteso con eventi specifici che permettono la costruzione di applicazioni sofisticate *client-side*. Agli eventi si aggiungono nuove API per la manipolazione degli oggetti DOM
- Guardiamo in particolare:
 - Canvas
 - Embedding di contenuti audio e video

HTML5: <canvas>

- Una delle più importanti innovazioni di HTML 5 è la possibilità di **disegnare direttamente sulla pagina e interagire con gli oggetti multimediali**
- L'elemento **<canvas>** definisce un'area rettangolare in cui disegnare direttamente immagini bidimensionali e modificarle in relazione a eventi, tramite funzioni Javascript
 - La larghezza e l'altezza del canvas sono specificati tramite gli attributi **width** e **height** dell'elemento **<canvas>**
 - Le coordinate (0,0) corrispondono all'angolo in alto a sinistra
- Esempio:
<canvas id="esempio" width="196" height="96">
</canvas>

HTML5: <audio>, <video>

- HTML 5 permette di includere video in una pagina senza richiedere plug-in esterni (Flash, Real Player, Quicktime)
- L'elemento **<video>** specifica un meccanismo generico per il caricamento di file e stream video, più alcune proprietà DOM per controllarne l'esecuzione
- Ogni elemento **<video>** in realtà può contenere diversi elementi **<source>** che specificano diversi file, tra i quali il browser sceglie quello da eseguire
- L'elemento **<audio>** è usato allo stesso modo per i contenuti sonori

Non esiste tuttavia codifica universalmente obbligatoria e accettata ma è talora necessario codificare il video (o audio) in più formati, per renderlo realmente disponibile su diversi browser ☹

HTML5: <video>

- Esempio d'uso dell'elemento <video>:

```
<video id="sampleMovie" width="640"  
      height="360" preload controls="metadata">  
  <source src="movie.mp4" type="video/mp4"  
    codecs="avc1.42E01E, mp4a.40.2" >  
  <source src="..." type="..." codecs="...">  
  <source src="..." type="..." codecs="...">  
  ...  
</video>
```

- Da notare la parte di sintassi sui *codec* in grigio... capiremo il perché tra un minuto...

HTML5: <video> e il problema dei *codec*

I membri WHATWG non sono riusciti a trovare un accordo sul formato di *codec* da usare per i video

A titolo esemplificativo:

- *Apple* supporta *H.264* (supportato da iPhone, iPod e Mac OS X) anche perché detiene alcuni brevetti software
 - *Mozilla* spingeva *Ogg Theora* (supportato da Firefox 3.5+) perché è un formato libero, ad esempio senza licenza invece necessaria per H.264
 - *Chrome* usa *H.264* ma supportava un suo formato prop. *WebM*
 - Anche motivazione di brevetti su H.264...
 - *YouTube* (ora Google) critica la mancanza di uno standard unico, congiuntamente alla mancanza di strumenti di protezione dei contenuti
-
- **! La «soluzione pratica» ad oggi è non indicare un formato nelle specifiche HTML e permettere l'uso di diversi *codec***

HTML5: ...e ancora

- HTML 5 introduce anche utili API specifiche per servizi avanzati di **geo-localizzazione**
- Tra i metodi esposti:
 - **getCurrentPosition()**, restituisce latitudine e longitudine della posizione dell'utente
 - **watchPosition()**, restituisce la posizione corrente dell'utente e continua ad aggiornare la posizione dello stesso durante i suoi spostamenti (come in una applicazione di navigazione GPS)
 - **ClearWatch()**, termina il metodo **watchPosition()**
 - ...
- Analizzeremo esempi concreti di pagine HTML 5 durante le esercitazioni guidate in laboratorio...

Riferimenti

- **HTML 4.01 Specification:**
<http://www.w3.org/TR/html401/>
- **HTML5:**
<http://www.w3.org/TR/html5/>
- **Guida in inglese (molto completa)**
<http://www.w3schools.com/html/default.asp>
- **Corso su HTML in italiano:**
<http://xhtml.html.it/guide/leggi/51/guida-html/>