Come costruire un sito web

Indice del Corso

1. **Fondamenti per scrivere una pagina web**
   - Introduzione al linguaggio di marcatura
   - Contenuto, struttura, presentazione
   - Elementi e attributi
   - Proviamo a scrivere una pagina HTML
   - Aggiungere intestazioni e paragrafi
   - Rendere le vostre pagine più interessanti con le immagini
   - Aggiungere collegamenti verso altre pagine
   - Scrivere un indirizzo e-mail
   - I percorsi assoluti e relativi
   - Fare una lista
   - Commentare il codice
   - I caratteri ammessi

2. **Una struttura rigorosa per i contenuti**
   - Introduzione al linguaggio XHTML
   - Cenni su XML
   - Quali sono i vantaggi di XHTML rispetto a HTML
   - La versione 1.0 di XHTML
   - Perché è meglio usare la DTD Srict
   - Le regole base di un documento XHTML
   - La validazione del codice
   - La struttura di una pagina XHTML
   - L'elemento DOCTYPE
   - L'elemento radice
   - La testata del documento
   - Il corpo del documento
   - Gli elementi blocco
   - Gli Elementi inline
   - Usare i tag per il loro scopo e significato

3. **La presentazione di una pagina web**
   - Introduzione ai fogli di stile CSS
   - A cosa servono i CSS
   - I vantaggi
   - Il supporto dei browser
   - Collegare i CSS ad una pagina XTHML
   - La struttura di un CSS
   - Come fatta una regola
   - I selettori principali
   - Le pseudo-classi
   - L'ereditarietà
   - Il testo: proprietà di base
   - Il box model
   - Il posizionamento

4. **Progettiamo un layout con i CSS**
   - La struttura logica e il layout
   - Il layout
   - Layout fissi e layout fluidi
   - Esempio pratico: layout a due colonne con float
   - Esempi di modelli XHTML/CSS

Modulo 1: Fondamenti per scrivere una pagina web

Indice del modulo

Lezione 1: Introduzione al linguaggio XHTML

Per pubblicare informazioni destinate ad Internet è necessario usare un linguaggio universalmente compreso, una specie di madre lingua per l'editoria comprensibile da tutti i computer.

Il linguaggio di pubblicazione usato dal web è l'HTML (da **H**yper**T**ext **M**arkup **L**anguage), un **linguaggio di marcatura ipertestuale**.

< NOTA BENE >

L'HTML si è evoluto nel corso del tempo. Il World Wide Web Consortium (detto W3C) è l'organizzazione che, fra le varie attività, si occupa di standardizzare e sviluppare la sintassi di questo linguaggio. Dopo la versione HTML 4.01 non ha più rilasciato nuove versioni, ma ha compiuto un'opera di ridefinizione, facendo evolvere l'HTML in XHTML (ne parleremo nel prossimo modulo).

HTML dà agli autori i mezzi per:

- **pubblicare** documenti sulla rete con titolazioni, testo, tabelle, elenchi, foto, ecc;
- **recuperare** informazioni dalla rete con collegamenti ipertestuali, cliccando un pulsante;
- **progettare** moduli per eseguire transazioni con servizi remoti, utili per cercare informazioni, fare prenotazioni, ordinare prodotti, ecc;
- **includere** fogli elettronici, sequenze video, sequenze audio ed altre applicazioni, direttamente nei propri documenti.

L'HTML è un linguaggio di marcatura. Gli autori marcano i loro documenti rappresentando, accanto al contenuto, informazioni strutturali.

Ecco un esempio di documento HTML

```
<html>
   <head>
      <title>Il mio primo documento HTML</title>
   </head>
   <body>
      <p>Ciao mondo!</p>
   </body>
</html>
```

Una pagina HTML è un semplice file di testo in cui alcuni caratteri svolgono le funzioni di **marcatori**, definendo la struttura del documento.

I marcatori [o *tag*] sono racchiusi fra i simboli < e >. I marcatori sono di due tipi: di apertura e di chiusura. Quelli di chiusura sono identici agli altri, ma preceduti dal simbolo /. Ogni marcatore indica una determinata informazione di formattazione, che si applica a ciò che è racchiuso fra quello iniziale e quello finale.

Esempio
Supponiamo di voler creare una pagina con questa frase:
"Ieri pomeriggio sono andato allo *stadio*".

La parola stadio è in formato *italico* (*corsivo*), la relativa istruzione HTML quindi è:
Ieri pomeriggio sono andato allo ****stadio****.

Analizziamo il significato dei simboli:

- < segnala al browser l'esistenza di un'istruzione html
- **em** comunica al browser che il nome dell'istruzione è italic
- > segnala al browser la chiusura dell'istruzione HTML. Quindi
  - **** è l'istruzione che segnala l'inizio del testo in stile italico
  - **** è l'istruzione che segnala la fine del testo in stile italico.

Lezione 2: Contenuto, struttura, presentazione

Prima di approfondire come utilizzare il linguaggio di marcatura è importante capire la differenza tra **contenuto**, **struttura** e **presentazione** di una pagina web.

1. **Contenuto**
   Per contenuto si intendono testo, immagini, suoni, filmati e animazioni. È fornito attraverso il linguaggio di marcatura che caratterizza i vari elementi della pagina web.

2. **Struttura**
   Rappresenta l'organizzazione logica del contenuto (ad esempio capitoli, titoli, sottotitoli). Il linguaggio di marcatura ha il compito di realizzare la struttura in cui inserire i contenuti.

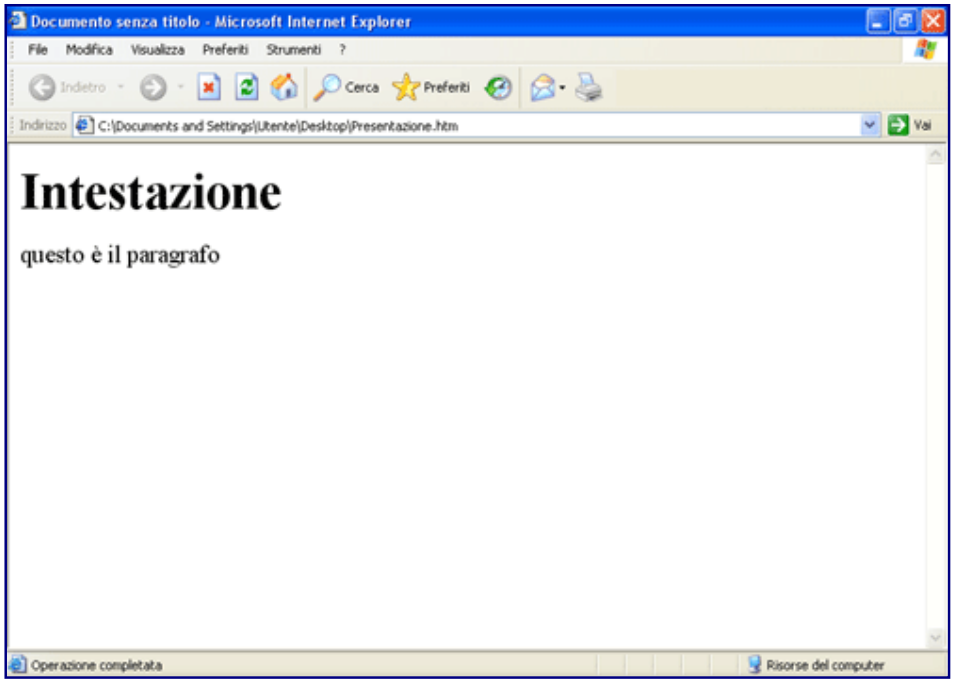
3. **Presentazione**
   È il modo in cui una pagina web è riprodotta, quindi implica la formattazione del carattere, la posizione degli oggetti, le scelte di layout, la grafica.

Esempio

Visualizza la **struttura** (in bordeaux) e il **contenuto** (in nero).

```
<div>
      <h1>Intestazione</h1>
      <p>questo è il paragrafo</p>
</div>
```

Visualizza la **presentazione**



Lezione 3: Elementi e attributi

L'HTML include **elementi** che servono per rappresentare paragrafi, collegamenti ipertestuali, elenchi, tabelle, immagini, ecc.

L'elemento compare all'interno dei marcatori secondo la seguente struttura:
<nome-elemento> contenuto **</nome-elemento>**

Ad esempio, per rappresentare un paragrafo utilizziamo l'elemento **p** e scriviamo:

```
<p> questo è un paragrafo </p>
```

Alcuni elementi non hanno contenuto. Per esempio **br** serve per mandare a capo un testo:

```
questo è un testo <br />
questo testo è scritto a capo
```

Gli elementi possono essere associati a proprietà, chiamate **attributi**, che possono avere dei *valori*.

Le coppie *attributo/valore* si trovano dentro al marcatore di apertura di un elemento.
`<nome-elemento attributo="valore"> testo </nome-elemento>`

Esempio
Per inserire un collegamento ipertestuale con le pagine di Cineca, bisogna scrivere:

```
<a href="http://www.cineca.it/">Cineca</a>
```

In cui:

- **<** segnala al browser l'esistenza di un'istruzione html
- **a** è il nome dell'elemento
- **href** è l'attributo
- **http://www.cineca.it/** è il valore dell'attributo
- **>** segnala al browser la chiusura dell'istruzione HTML
- **** è il marcatore finale dell'elemento **a**

Lezione 4: Proviamo a scrivere una pagina HTML

Per scrivere una pagina HTML basta utilizzare un programma editore di testo, ad esempio **Notepad** (per Windows), **TextEdit** (per Mac) o **KEdit** (per KDE). Compresi i principi si può passare a programmi più avanzati, o anche commerciali, come Style Master, Dreamweaver o GoLive.

Per prima cosa bisogna aprire l'editore di testo, cominciare con una finestra vuota e scrivere:

```
<html>
   <head>
      <title> Il mio primo documento in HTML </title>
   </head>
   <body>
      Corpo del documento
   </body>
</html>
```

Ora, selezionate il comando “Salva con nome…” dal menu File, andate nella cartella dove volete metterlo e salvate il file come “miapagina.html”. Non chiudete l'editor, perché serve ancora.

Approfondimento: Per chi usa TextEdit per Mac
Innanzitutto bisogna dire a TextEdit che il testo è puro testo. Andando sul bottone Preferenze si può impostare di default che i file creati siano di solo testo. Quando si salva e TextEdit propone di aggiungere l'estensione “.txt” al file, basta eliminare il flag dall’opzione “Se non è indicata alcuna estensione…” e aggiungere a mano l’estensione .htm al nome.

Aprite il file in un browser, ossia:

1. andate nella cartella in cui lo avete salvato;
2. cliccate sul file “miapagina.html”.

Dovrebbe aprirsi nel browser di default (se non accade, aprite una finestra del browser e trascinate il file su di essa).

Il **titolo** (definito dall’elemento `title`) non compare nella pagina, ma viene visualizzato nell’intestazione della finestra del browser. La pagina è piuttosto scarna… viene visualizzato solo il testo “Corpo del documento”.

Lezione 5: Aggiungere intestazioni e paragrafi

Se utilizzate Microsoft Word vi saranno già familiari gli stili predefiniti per intestazioni di differente importanza. In HTML esistono **6 livelli** di intestazioni. `H1` è il più importante, `H2` un po’ meno e così via, fino ad `H6`, il meno importante di tutti.

Ecco come aggiungere un'intestazione importante:

```
<h1>Un'intestazione importante</h1>
```

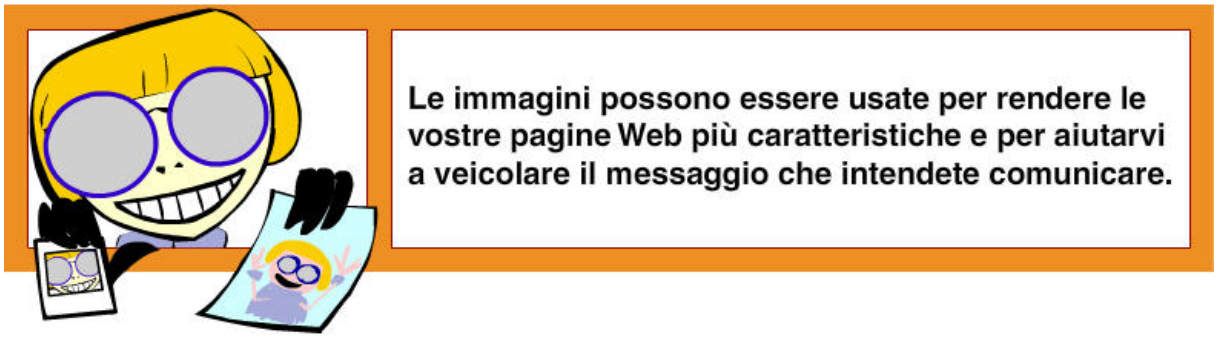
ed ecco un'intestazione leggermente meno importante:

```
<h2>Un'intestazione leggermente meno importante</h2>
```

Ogni paragrafo che scrivete deve cominciare con il marcatore ** ** e terminare con il marcatore ** ** . Per esempio:

```
<p>Questo è il primo paragrafo.</p>
<p>Questo è il secondo paragrafo.</p>
```

Lezione 6: Rendere le vostre pagine più interessanti con le immagini



Il modo più semplice di aggiungere un'immagine è usare il marcatore **<img>.** Poniamo che vi sia un file immagine chiamato "pietro.jpg" nella stessa cartella/directory del vostro file HTML. Esso misura 200 pixel di larghezza per 150 di altezza.

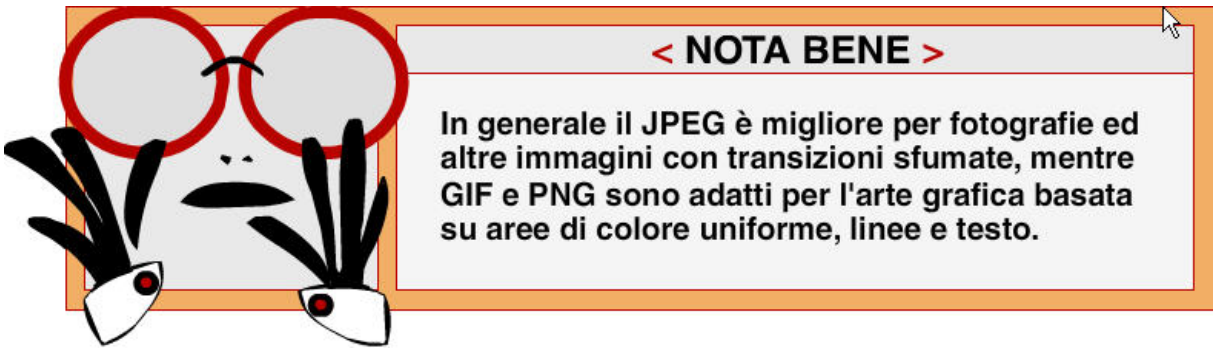
```
<img src="pietro.jpg" width="200" height="150">
```

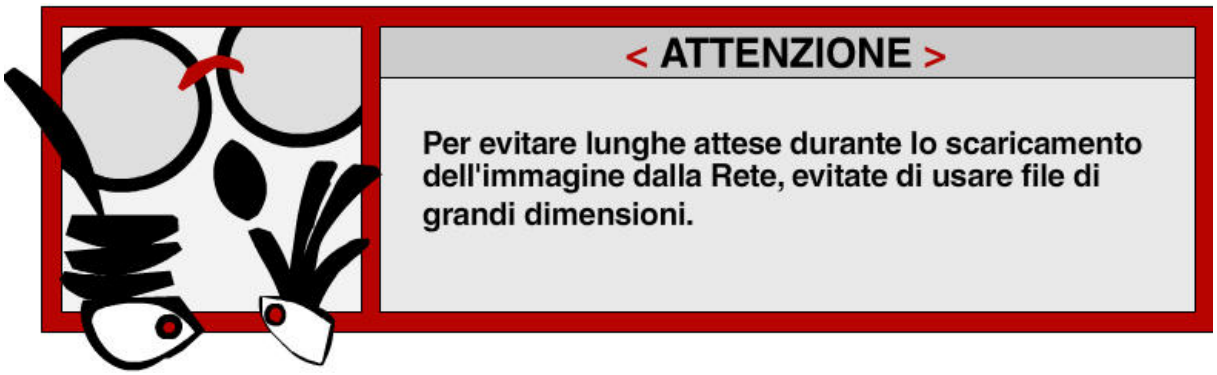
L'attributo **src** richiama il file immagine. La larghezza e l'altezza non sono strettamente necessarie ma aiutano a velocizzare la visualizzazione della vostra pagina Web.

Le persone che non possono vedere l'immagine hanno bisogno di una descrizione da leggere in sua assenza. Potete aggiungere una breve descrizione utilizzando l'attributo **alt**:

```
<img src="pietro.jpg" width="200" height="150"
 alt="Il mio amico Pietro">
```

La maggior parte dei browser sono in grado di visualizzare i formati grafici GIF e JPEG, quelli più recenti supportano anche il PNG.

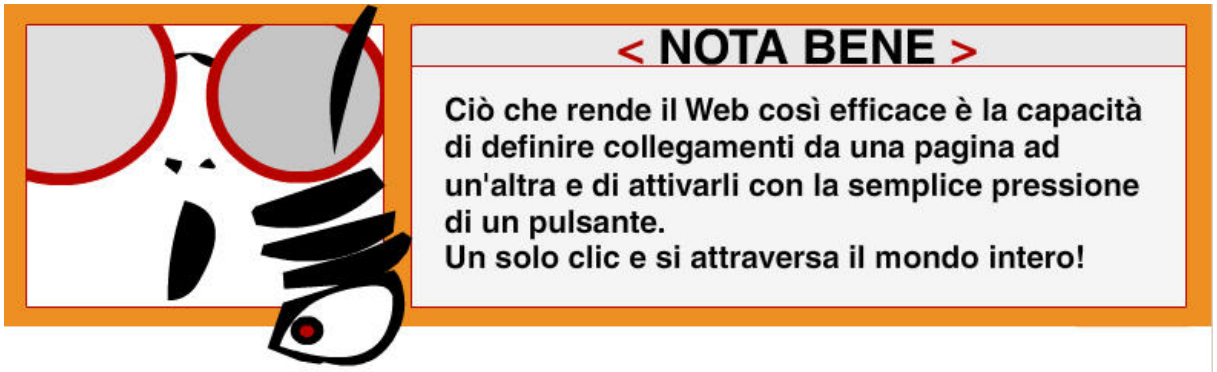




In linea di massima, infatti, per scaricare una pagina html che “pesa” 50 KB con un modem a 56 Kb/s occorrono circa 15 secondi (1 secondo con un ADSL da 640 Kb/s).

Il "peso" di una pagina html si ottiene sommando le dimensioni del documento e dei contenuti multimediali da esso richiamati (immagini, sfondi, suoni, applet ecc).

Lezione 7: Aggiungere collegamenti verso altre pagine



I collegamenti sono definiti per mezzo del marcatore **<a>**. Creiamone ora uno alla pagina descritta nel file "pietro.html":

```
Questo è un collegamento <a href="pietro.html">
alla pagina di Pietro</a>.
```

Il testo tra **<a>** e **** è usato come didascalia per il collegamento. Essa appare di solito come un testo blu sottolineato.

Per fare un collegamento ad una pagina su un altro sito bisogna fornire l'indirizzo completo (detto **URL**), per esempio per collegarsi a http://www.w3c.org dovete scrivere:

```
Questo è un link al <a href="http://www.w3.org/">W3C</a>.
```

Potete anche trasformare un'immagine in un collegamento ipertestuale, per esempio quello che segue vi consente di cliccare sull'immagine per andare alla home page:

```
<a href=" http://www.cineca.it/"><img src="immagine.gif"
alt="home page Cineca"></a>
```

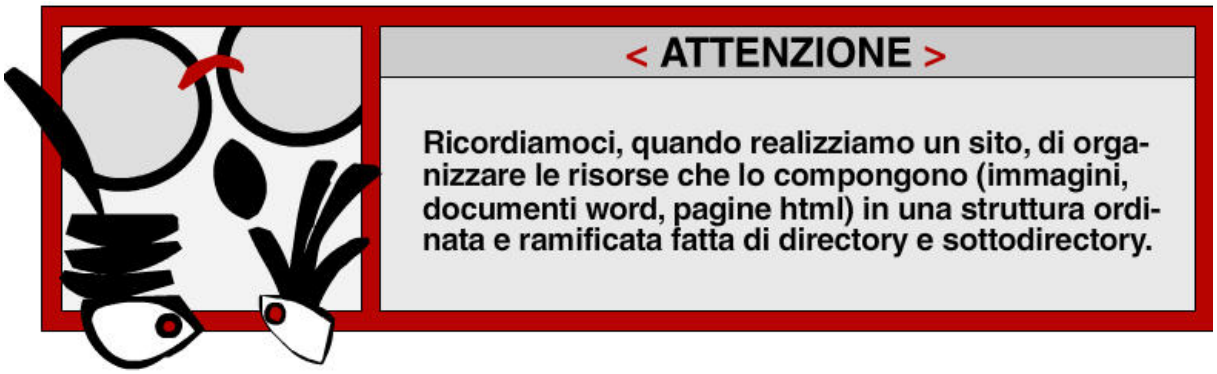
Lezione 8: Scrivere un indirizzo mail

Potete specificare un indirizzo e-mail. In questo caso si aprirà direttamente il client di posta dell'utente con l'indirizzo pre-impostato.

La sintassi da utilizzare è la seguente:

```
<a href="mailto:Mail@nomeSito.it">Invia una mail</a>.
```

Lezione 9: I percorsi assoluti e relativi



Dobbiamo imparare a muoverci tra i vari file che costituiscono il sito, per essere in grado di creare collegamenti tra le varie risorse della nostra creazione.

Per farlo esistono due tecniche:

- indicare un **percorso assoluto**;
- indicare un **percorso relativo**.

I **percorsi assolut**i si usano generalmente per fare riferimento a risorse situate in altri siti.
Per creare un collegamento assoluto basta fare riferimento all'URL che di solito si trova nella barra degli indirizzi.

Ad esempio, se dal sito del Cineca vogliamo creare un link alla pagina del portale della ricerca italiana che tratta della "qualità della vita "dobbiamo usare la seguente sintassi:

```
Leggi le informazioni sulla
<a href="http://www.ricercaitaliana.it/rassegna/rassegna.htm">
qualità della vita </a>
```

In cui

- **`http://`** indica al browser di utilizzare il protocollo per navigare nel web (l'http);
- **`www.ricercaitaliana.it`** indica di fare riferimento a tale sito;
- **`rassegna/`** indica che la risorsa indicata si trova all'interno della cartella "rassegna";
- **`rassegna.htm`** indica che il file da collegare è quello chiamato "rassegna.htm".

I **percorsi relativi** indicano la posizione degli altri file rispetto al documento in cui ci troviamo ora. Per far riferimento a un file che si trovi all'interno della stessa directory basta linkare il suo nome. Per far riferimento a un file contenuto in una cartella di livello inferiore alla posizione corrente, basta nominare la cartella, seguita dallo "slash", e poi il nome del file.

Esempio

Immaginiamo di aver strutturato parte delle risorse del nostro sito in cartelle. Dalla pagina "index.html" vogliamo far riferimento al file "pagina1.html", che si trova all'interno della directory "comunicati", che a sua volta si trova all'interno della directory "stampa".

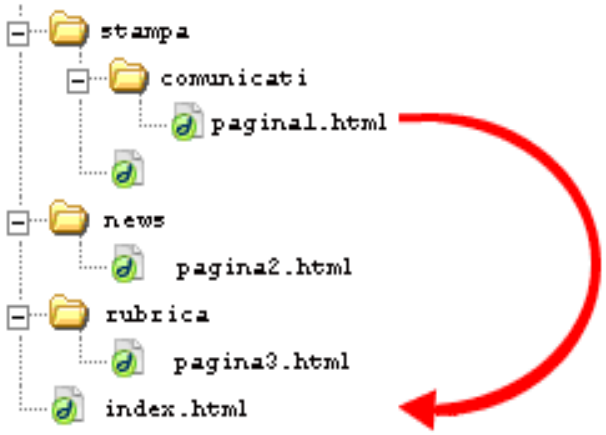
La sintassi è la seguente:

```
<a href="stampa/comunicati/pagina1.html">Visita la prima pagina</a>
```

Per far riferimento a un file contenuto in una cartella di livello superiore, è sufficiente utilizzare la notazione: **`../nomeFile.html`**

Esempio

Adesso dalla “pagina1.html” vogliamo far riferimento a una "index.html" che si trova più in alto di due livelli:



La sintassi è la seguente:

```
<a href="../../index.html">Visita la pagina index</a>
```

Se dalla “pagina1.html” vogliamo fare un riferimento a “pagina2.html” che si trova nella cartella “news” scriviamo:

```
<a href="../../news/pagina2.html">Visita la pagina 2</a>
```

Lezione 10: Fare una lista

Un tipo di lista è l'elenco puntato, definito spesso **lista non ordinata**. Usa i marcatori **<ul>** e **<li>**, per esempio:

```
<ul>
  <li>il primo elemento della lista</li>

  <li>il secondo elemento della lista</li>

  <li>il terzo elemento della lista</li>
</ul>
```

Un altro è l'elenco numerato, detto anche **lista ordinata**. Usa i marcatori **<ol>** e **<li>**. Per esempio:

```
<ol>
  <li>il primo elemento della lista</li>

  <li>il secondo elemento della lista</li>

  <li>il terzo elemento della lista</li>
</ol>
```

Notate che le liste possono essere annidate l'una dentro l'altra.
Per esempio:

```
<ol>
  <li>il primo elemento della lista</li>
  <li>
    il secondo elemento della lista
    <ul>
      <li>primo elemento annidato</li>
      <li>secondo elemento annidato</li>
    </ul>
  </li>
  <li>il terzo elemento della lista</li>
</ol>
```

Potete anche utilizzare paragrafi e intestazioni per elementi di lista più lunghi.

Lezione 11: Commentare il codice

Per ottenere una migliore consultazione del file HTML è meglio aggiungere commenti per migliorare la lettura del codice.

I commenti sono ignorati dal browser, quindi non appaiono nella presentazione della pagina.

Il testo da commentare va inserito tra le istruzioni **<!--** e **-->**, ad esempio:

```
<!-- questo paragrafo serve per spiegare l'HTML -->
<p> L'obiettivo del corso è spiegare l'uso dell'HTML…</p>
```

Lezione 12: I caratteri ammessi

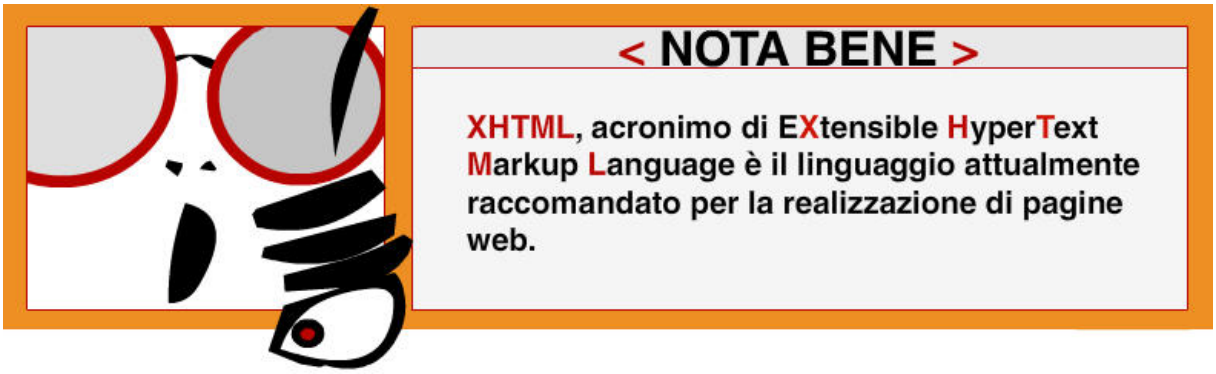
I documenti in formato HTML possono contenere qualsiasi carattere definito nello standard ISO Latin-1. Per evitare problemi nello scambio di documenti internazionali, ad esempio caratteri particolari come le lettere accentate, è meglio consultare la tabella di riferimento che trovate in allegato al corso.

Ci sono poi alcuni caratteri speciali che possono essere utili nella redazione della pagine HTML. Uno di questi è lo **Spazio unificatore** (Non-breaking space), che viene utilizzato quando uno spazio che separa due parole non deve essere interrotto dall'andata a capo causata dalla formattazione del paragrafo (es. nomi propri, sigle, ecc.). Questo carattere viene anche usato impropriamente per inserire forzatamente degli spazi tra due parole.
Il codice html di questo carattere è: ` `

Modulo 2: Una struttura rigorosa per i contenuti

Indice del modulo

Lezione 1: Introduzione al linguaggio XHTML



Chi conosce già l'HTML non deve imparare un nuovo linguaggio. I tag sono essenzialmente gli stessi; ciò che cambia è il modo di scriverli: più rigoroso, perché basato sulle regole di **XML**, eXtensible **M**arkup **L**anguage (ecco spiegata la **X** davanti a HTML!) .

HTML + XML = XHTML

Ad esempio, in base alle regole dell'XHTML, ai tag iniziali deve sempre corrispondere un tag finale e i valori degli attributi devono essere specificati tra doppi apici (“ ”).

XHTML si basa quindi su XML e ne condivide regole di base e potenzialità, rispondendo così a due esigenze fondamentali:

- portare HTML nella famiglia **XML**, con i benefici che ciò comporta in termini di estensibilità e rigore sintattico: il documento XHTML è un documento XML
- mantenere la **compatibilità** con i browser che supportano HTML.

Lezione 2: Cenni su XML

XHTML è quindi una combinazione di HTML e XML, costituito da tutti gli elementi di HTML versione 4.01 combinati con la sintassi XML.

Ma cos'è XML?

- XML è un **metalinguaggio** che permette di creare dei linguaggi personalizzati di marcatura
- potente, flessibile e rigoroso, è alla base di tutte le nuove specifiche tecnologiche rilasciate dal W3C
- in un documento XML possono essere definiti **nuovi tag** ed **attributi**

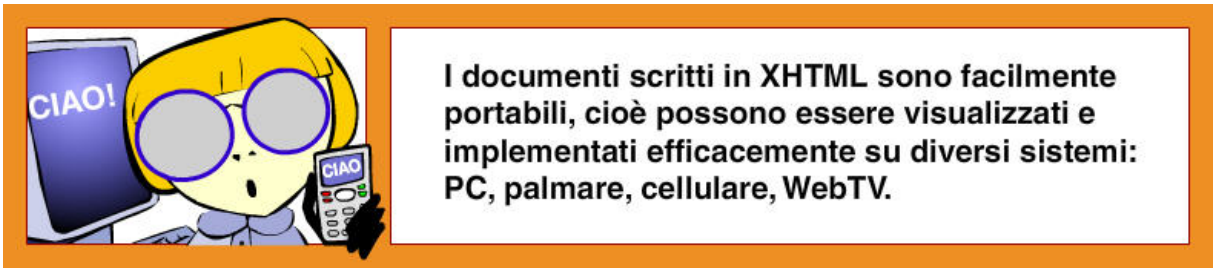
- la struttura di un documento XML può essere vista **in modo gerarchico**, nidificando i tag in ogni livello di complessità (**struttura ad albero**)
- il linguaggio XML ha la capacità di **fornire una struttura ai documenti** e di rendere i dati **autodescrittivi**.

Esempio di tag XML

```
<rubrica>
   <contatto>
        <nome>Marco</nome>
        <cognome>Rossi</nome>
   </contatto>
</rubrica>
```

Lezione 3: Quali sono i vantaggi di XHTML rispetto a HTML?

L'utilizzo di XHTML comporta notevoli benefici, primo fra tutti **vantaggi di gestione** per lo sviluppatore: un **codice pulito e ben strutturato** significa pagine più leggere, più gestibili, più facili da mantenere, quindi meno lavoro di produzione.



Infatti potendo contare su un linguaggio essenziale, centrato sulla struttura (titolo, intestazione, paragrafo, lista per gli argomenti...), è sufficiente applicare al documento XHTML un foglio di stile adeguato per presentare gli stessi dati su dispositivi differenti: ad esempio uno con grafica essenziale se si tratta di un cellulare.

Software come i **lettori di schermo**, utilizzati da persone non vedenti, o *browser vocali e testuali* gestiscono meglio una pagina con un codice pulito e ben strutturato.

La recente "legge Stanca" sull'**accessibilità** – *Disposizione per favorire l'accesso dei soggetti disabili agli strumenti informatici* – richiede che sia utilizzato XHTML 1.0.

Infine, visto che XHTML è un documento XML, diventa **estensibile**. Ciò vuol dire che si potrà incorporare nel documento del codice scritto in uno dei tanti linguaggi della famiglia XML.

Lezione 4: La versione 1.0 di XHTML

La versione 1.0 di XHTML è la riscrittura in XML di HTML 4.0. Ogni documento deve contenere l'indicazione della versione XHTML (**versione 1.0**, che è quella **raccomandata**, la versione 1.1, ecc.) e della DTD utilizzata.

Ma cos'è la DTD?
Nella **DTD**, acronimo di **D**ocument **T**ype **D**efinition, cioè "definizione del tipo di documento" sono definite le regole di applicazione di tutti gli elementi e gli attributi utilizzati nel documento: contiene la lista degli elementi e attributi consentiti, il loro significato e come utilizzarli. Ad esempio è specificato che gli elementi header **`h1, h2, h3, h4, h5, h6`**, identificano sei livelli di titolo, che **`h1`** è il più importante e **`h6`** il meno.
Le DTD per XHTML 1.0 sono 3:

- **DTD Strict**
  All'interno del documento XHTML appare questa informazione:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Con la DTD Strict si ottiene una totale separazione tra i contenuti e la presentazione. La pagina XHTML è centrata solo sulla struttura del documento, quindi non utilizza elementi e attributi per gestire l'aspetto della pagina. Il foglio di stile CSS riguarda la presentazione.

- **DTD Transitional** POP UP 2
  La DTD Transitional è definita nel documento da questa stringa:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Supporta tutti gli elementi e gli attributi di presentazione di HTML 4.0. Ciò però significa mescolare ai contenuti della pagina, in modo inestricabile, una quantità di codice che serve solo a definire la presentazione visuale dei contenuti. Per questo motivo l'XHTML Transitional è sconsigliato per realizzare nuovi siti.

- **DTD Frameset**
  La DTD Frameset è definita da questa stringa:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

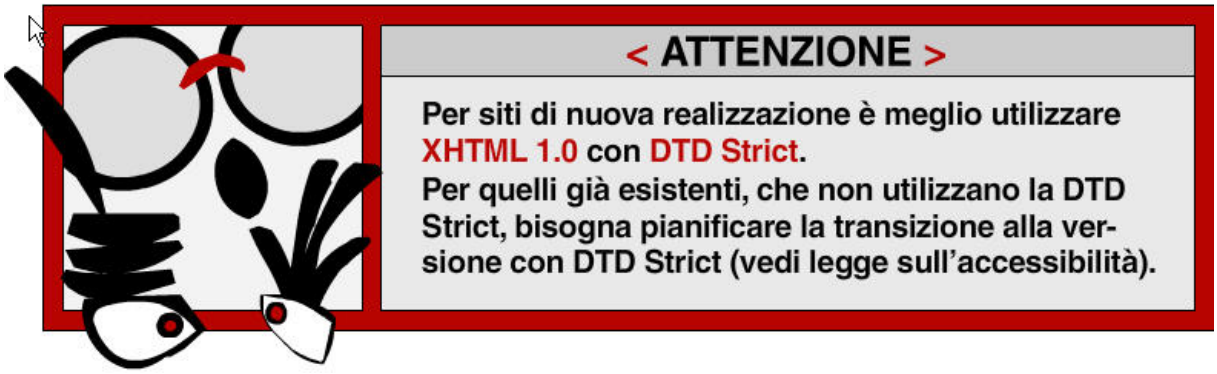
Serve per i frame, in ogni caso con parsimonia: il loro utilizzo è sconsigliato per l'accessibilità.

Lezione 5: Perché è meglio usare la DTD Strict

Con XHTML 1.0 Strict si ottiene una **totale separazione fra struttura e contenuto** da un lato e **presentazione** dall'altro, quindi aumenta la possibilità di accesso ai contenuti da parte di diversi programmi utente.

Per utilizzare la DTD Strict bisogna riportare la seguente dicitura in testa al documento XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```



Consulta l'allegato del corso "Requisito 1 della legge Stanca."

La pagina XHTML è utilizzata solo per i **contenuti** e la **struttura**, eliminando alcuni elementi e attributi di presentazione.

Approfondimento: elementi e attributi di presentazione
Per elemento di presentazione si intende un elemento che specifica la presentazione del documento (es. `<b>`, `<font>`, `<center>`). Analogamente, un attributo di presentazione specifica un modo di presentare un certo contenuto (ad esempio "`align`", "`height`", "`bgcolor`", "`size`", "`face`", "`color`" e simili).

La **presentazione** è gestita completamente dal foglio stile CSS. Se si inseriscono elementi non supportati (ad esempio l'elemento `font`, l'attributo `align` per le tabelle, ecc.) il documento non è valido.

Lezione 6: Le regole base di un documento XHTML

1. **Il documento deve essere ben formato**
   Ben-formato è un concetto introdotto da XML. Sostanzialmente significa che tutti gli elementi devono avere il tag di chiusura o essere scritti in una forma speciale e che, delimitati da tag-di-inizio e tag-di-fine, devono essere annidati in modo appropriato.

   Esempio corretto: elementi annidati

   ```
   <strong>questo testo è grassetto, <em>questo corsivo</em></strong>
   ```

   Esempio sbagliato: elementi sovrapposti

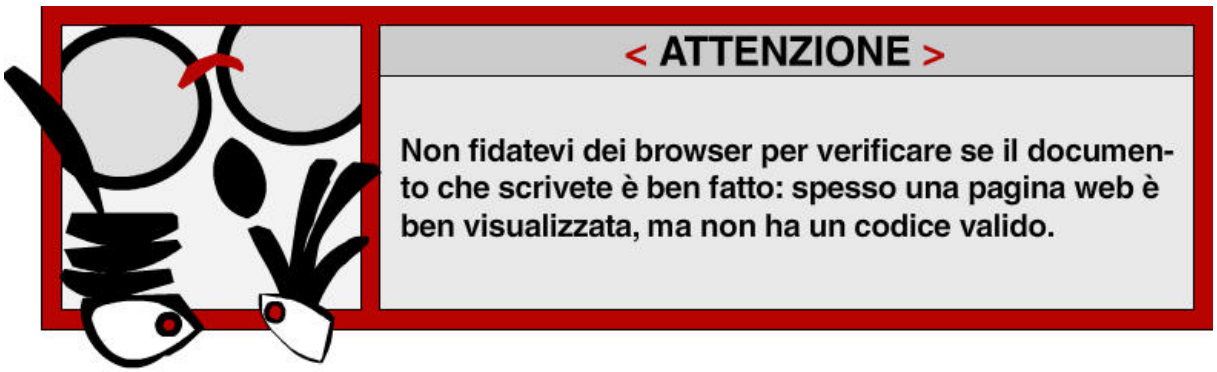
   ```
   <strong>questo testo è grassetto, <em>questo corsivo</strong></em>
   ```

   Consulta l'allegato sulle differenze con HTML 4.0.

2. **Il documento deve essere valido**
   Un documento è valido solo se:
   - al suo inizio è **dichiarata la DTD utilizzata**;
   - gli **elementi** e gli **attributi** utilizzati rispettano alla lettera la **sintassi** per loro definita nella DTD dichiarata all'inizio.

Lezione 7: La validazione del codice

La validazione del codice controlla che il documento sia **valido** e **ben formato**: in XHTML la correttezza formale è fondamentale per ottenere dei veri vantaggi.



Il controllo di validità del codice può essere effettuato ricorrendo a software specifici:

- **MarkUp Validation Service**
  È un software automatico del W3C; ha solo il limite di effettuare il controllo una pagina alla volta. Si può scaricare alla pagina http://validator.w3.org/

- **WDG HTML Validator**
  Il WDG HTML Validator ha il pregio di controllare in una volta sola l'intero sito.
  Si può scaricare alla pagina http://www.htmlhelp.com/tools/validator/

Entrambi i validatori automatici forniscono la riga del listato della pagina in cui c'è l'errore e ne descrivono il genere.

La maggior parte degli errori riscontrati riguardano:

- DTD non dichiarata;
- uso di elementi e attributi non consentiti dalla DTD adoperata;
- elementi aperti e non chiusi o viceversa;
- elementi non correttamente annidati
  (p.es. `<strong><em>` ... `</strong></em>` , invece di `<strong><em>` ... `</em></strong>`);
- uso di caratteri speciali, ad esempio '&', che va sostituito con l'entità carattere `&amp;` ;
- uso di valori di attributo non consentiti .

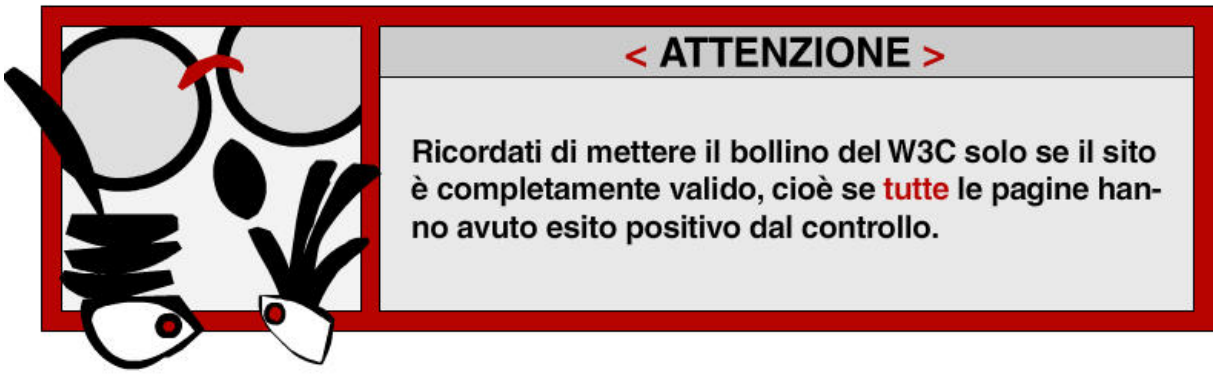
Purtroppo il controllo di validità del codice è spesso trascurato dagli sviluppatori, anche per i siti più importanti. Se proviamo, ad esempio, a far analizzare al validatore automatico del W3C la prima pagina di www.trenitalia.it otteniamo **17 errori di codice** (prova eseguita il 22/03/2006).

L’esito positivo del validatore dà diritto ad inserire sul sito il cosiddetto “bollino W3C”, che certifica la validità delle pagine.

Questo è il bollino W3C

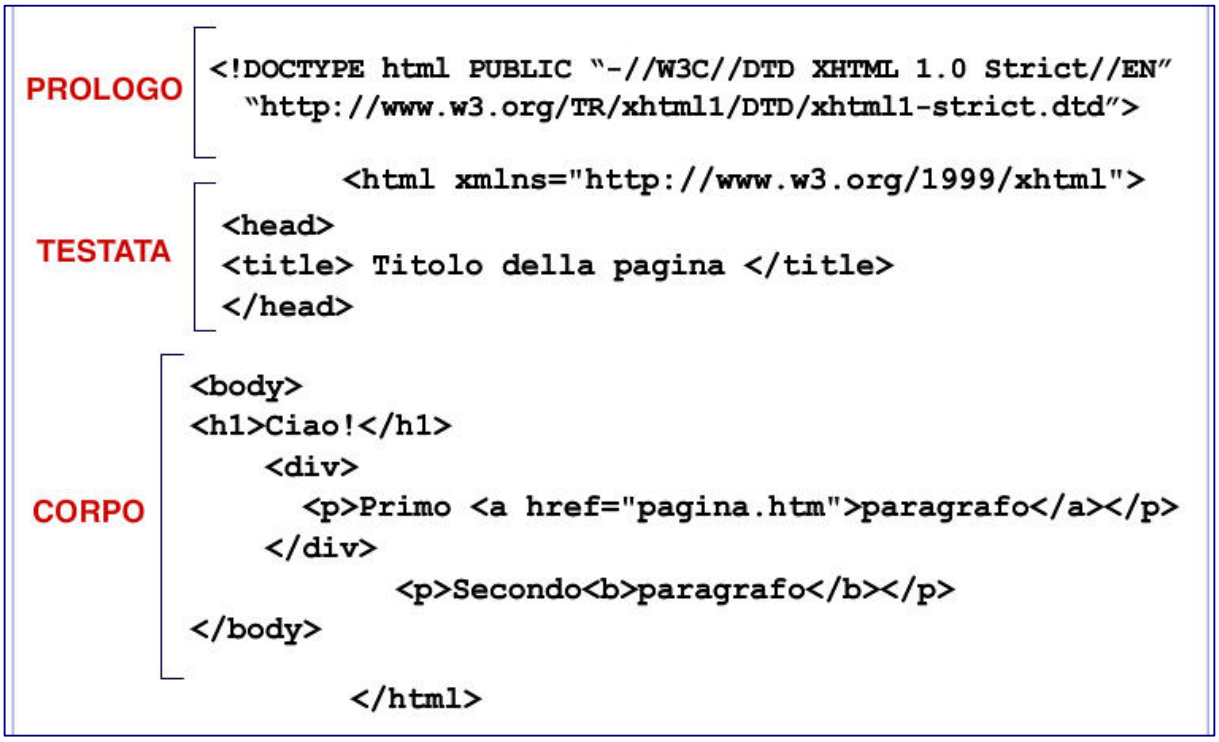
Per inserirlo su una pagina del sito basta scrivere alcune righe di codice:

```
<p>
   <a href="http://validator.w3.org/check/referer">
      <img src="http://www.w3.org/Icons/valid-xhtml10"
     alt="Valid XHTML 1.0!" height="31" width="88" /></a>
</p>
```



Lezione 8: La struttura di una pagina XHTML

Cominciamo a creare la nostra prima pagina XHTML e analizziamo subito la struttura:



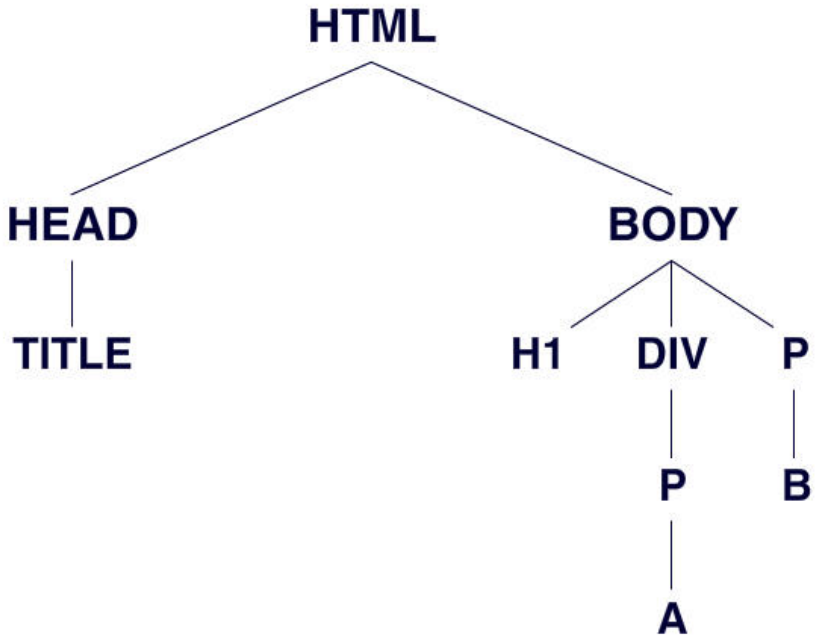
Tutti gli elementi hanno tra loro una relazione del tipo **genitore-figlio** (*parent-child* in inglese):

- un elemento è **genitore** quando contiene altri elementi
- un elemento è **figlio** quando è racchiuso in un altro elemento.

Ad esempio:

- **`body`** è figlio di **`html`**
- **`body`** è genitore di **`h1`**, **`div`** e **`p`**
- **`p`** è genitore di **`b`**.

Il frammento di codice XHTML è rappresentabile con una struttura ad albero che evidenzia le relazioni fra le varie componenti.



Lezione 9: L'elemento DOCTYPE

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

L'elemento **DOCTYPE** deve essere il primo ad aprire il documento XHTML: in questo esempio è dichiarato che si utilizza XHTML 1.0 con DTD Strict.

L'elemento **DOCTYPE** consente ai browser di processare correttamente il documento e al validatore di controllare il codice in base alla DTD dichiarata.
La lista completa dei **DOCTYPE** è curata dal W3C ed è disponibile all'indirizzo http://www.w3.org/QA/2002/04/valid-dtd-list.html .

Lezione 10: L'elemento radice

```
<html xmlns="http://www.w3.org/1999/xhtml">
   ……
   ……
   ……
</html>
```

In un documento XHTML ci deve essere l'elemento **<html>,** che deve a sua volta contenere tutti gli altri e avere il tag di chiusura **</html>**, la fine di tutto il documento.

L'elemento **<html>** può avere diversi attributi, ma l'unico obbligatorio è **xmlns**. **xmlns** specifica il *namespace* (spazio di nomi) predefinito per XHTML, la cui necessità deriva da XML, un linguaggio estensibile. È possibile, infatti, allargare il set di tag di XHTML con elementi di altri linguaggi, anche creati di persona. Specificare uno o più *namespace* evita conflitti tra i tag.

Lezione 11: La testata del documento

```
<head>
   <title> Titolo della pagina </title>
</head>
```

La funzione principale della testata (**<head>...</head>**) è contenere informazioni molto importanti che non vengono direttamente visualizzate nella pagina.

Gli elementi principali che si usano sono il **title** e i **META.**

Elemento `title`
Il contenuto dell'elemento **title** fornisce il titolo ed è visualizzato nell'intestazione di pagina. È letto dal browser vocale prima di procedere alla scansione del contenuto ed è salvato come titolo di pagina quando si usa lo strumento "Preferiti" del browser.

L'elemento **title** deve identificare in modo chiaro i contenuti di un documento anche al di fuori del contesto del sito, ad esempio "Le gite scolastiche", va modificato con "Nome_della_Scuola - Le gite scolastiche".

Elemento META
I **META** sono stringhe di codice in testa al documento, che vengono lette per prime dai motori di ricerca. Forniscono meta-informazione alla pagina web: le *keywords* (parole chiave che descrivono sinteticamente il contenuto), l'indicazione dell'autore, il titolo che apparirà alla fine della ricerca, ecc.

Ad esempio per specificare l'autore di un documento, si usa l'elemento **META** come segue:

```
<meta name="Author" content="Mario Rossi">
```

L'elemento **META** specifica una proprietà (in questo caso **"Author"**) e le assegna un valore (in questo caso **"Mario Rossi"**).

Lezione 12: Il corpo del documento

```
<body>
      <h1>Ciao!</h1>
  <div>
       <p>Primo <a href="pagina.htm">paragrafo</a></p>
  </div>
          <p>Secondo<strong>paragrafo</strong></p>
</body>
```

Il corpo del documento è la sezione in cui si sviluppa il contenuto. È racchiusa tra i tag **<body>...</body>.** Gli elementi che possono comparire all'interno del corpo sono in genere suddivisi in due categorie: **elementi blocco** ed **elementi inline**.

Lezione 13: Gli elementi blocco

Gli elementi blocco sono quelli che **definiscono la struttura** del documento.

Possono contenere altri elementi blocco, inline (che vedremo nella lezione seguente) o testo. Quando sono inseriti danno origine ad una nuova riga nel flusso del documento.

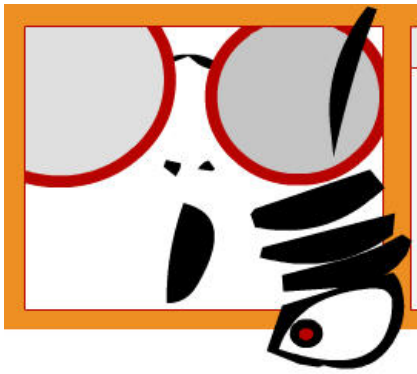
Esempio

Il testo "Paragrafo" si troverà su una nuova riga, in quanto abbiamo inserito un nuovo elemento blocco (**<p>**).

```
<h1>Titolo</h1>
<p>Paragrafo</p>
```

Consulta l'allegato sugli elementi blocco

Lezione 14: Gerarchie e annidamento degli elementi blocco



< NOTA BENE >

Nella strutturazione del documento dobbiamo rispettare alcune semplici regole relative alla gerarchia e all'annidamento degli elementi bloc-co.

Il primo elemento della gerarchia è ** **, che definisce una sezione del documento. Al suo interno si trovano gli altri. È importante evitare annidamenti sbagliati, che i browser fanno passare senza problemi, ma che il validatore segnala come documento non *ben-formato*.

Ad esempio, scriviamo questa porzione di codice:

```
<p><div>Qui inserisco il mio testo</div></p>
```

Se la inseriamo in un documento, visualizzeremo regolarmente il testo. Ma non bisogna fidarsi solo del browser, perché in questo caso il documento non è valido: **<p>** non può contenere altri elementi blocco.

L'annidamento giusto è:

```
<div><p>Qui inserisco il mio testo</p></div>
```

Lezione 15: Elementi inline

Gli elementi inline si distinguono da quelli di tipo blocco per due motivi: quando sono inseriti non danno origine a una nuova riga e possono contenere solo dati (essenzialmente testo) o altri elementi inline.

Esempio:

```
<p>Questo testo è <strong>grassetto</strong></p>
```

La parte delimitata dai tag **...** non viene posta su una nuova riga. Anche per gli elementi inline ci vuole molta attenzione nell'annidamento.

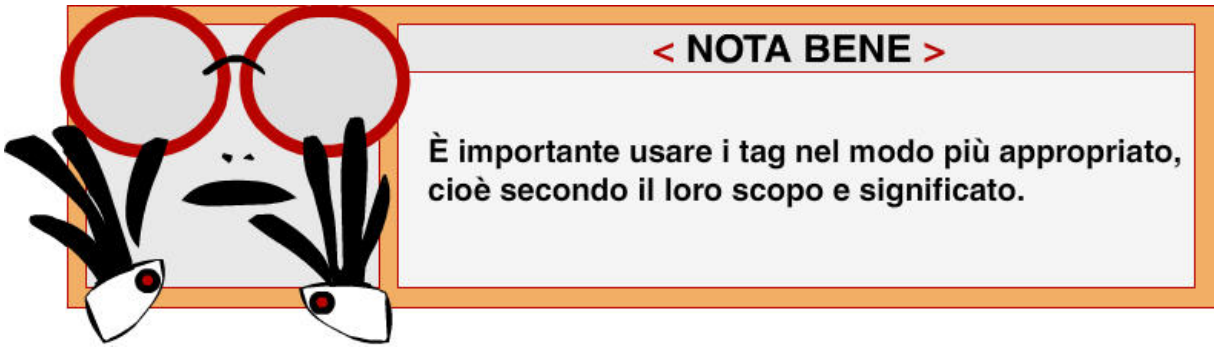
Esempi come questo:

```
<strong><p>Testo in grassetto</p></strong>
```

sono tollerati dai browser, ma non reggono alla validazione: un elemento inline non può contenerne uno di tipo blocco.

Consulta l'allegato sugli elementi inline.

Lezione 16: Usare i tag per il loro scopo e significato



Per capire, vediamo gli elementi strutturali più importanti e il loro uso:

I titoli

I titoli servono per dichiarare pagine, sezioni di pagina e paragrafi.
Il titolo principale di una pagina **<h1>** di solito serve ad indicare il nome del sito. Il tag **<h2>** generalmente descrive una pagina o una macro-sezione, mentre il tag **<h3>** introduce una sotto-sezione. E così via.

I paragrafi

I paragrafi servono a contenere frasi.
Generalmente un paragrafo contiene al massimo tre-quattro frasi logicamente correlate. A tal proposito, usare più tag **<p>** in una sezione di testo serve per separare i paragrafi e facilitarne la lettura.

Il tag <br /> per mandare a capo il testo

L'uso del tag **
** andrebbe limitato al minimo. L'uso di più **
** consecutivi, per separare sezioni di testo e aumentare la lunghezza di una sezione, andrebbe sempre evitato: i margini impostati via CSS servono proprio a questo.

L'enfasi è un aspetto essenziale nelle sezioni di testo. Si può ottenere con i tag **** e ****, che sono resi rispettivamente *in corsivo* e **in grassetto**. **<b>** e **<i>** danno la stessa resa visuale: sono però **tag sconsigliati** perché sono esclusivamente presentazionali e non hanno nessun contenuto semantico. Usare **em** e **strong** serve a focalizzare la lettura su parole o sezioni di testo in rilievo, dare una certa profondità al testo e facilitare la lettura.

Le liste non ordinate

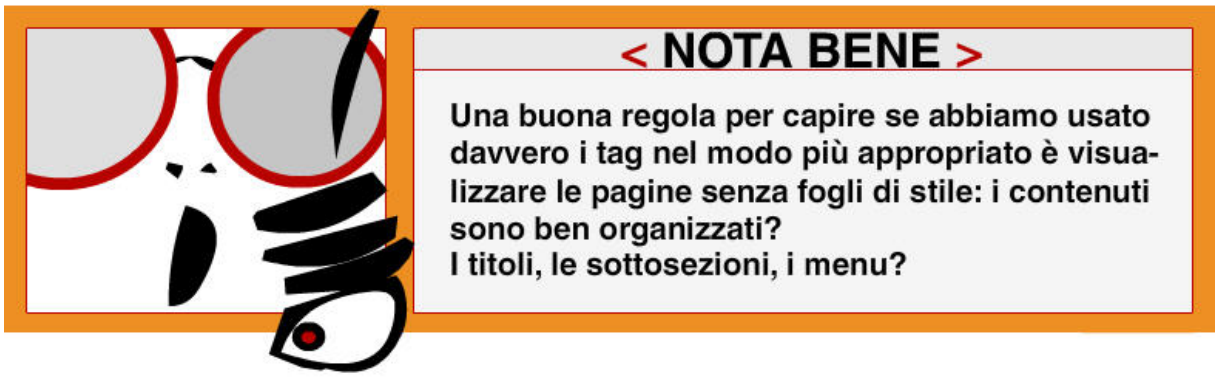
Le liste non ordinate **<ul>**, oltre che per elenchi semplici, vanno utilizzate per menù di navigazione, elenchi di notizie e gallerie di immagini. (vedi anche il modulo sui CSS)

Il tag

I ** ** sono i contenitori generici per eccellenza. Vanno usati per definire sezioni di pagina e contenere elementi correlati (esempio: titolo + paragrafi relativi).

Lo

Lo **** è un elemento in linea totalmente neutro, da personalizzare con i CSS, che non ha contenuto semantico effettivo: per questo andrebbe usato con molta parsimonia.



Analizziamo ora alcuni esempi di codice non semantico (tag non usati per il loro scopo e significato) con la corrispettiva versione semantica.

- **Come scrivere il titolo**
  Questa è la versione non semantica per un titolo:

```
<p class="titolo">Questo è il titolo di un paragrafo</p>
```

  E la corrispettiva versione semantica:

```
<h2>Questo è il titolo di un paragrafo</h2>
```

- **Come scrivere un menu di navigazione**

Vediamo un esempio di codice non semantico per un menu di navigazione:

```
<div id="menu">
    <p><a href="home.html">Home page</a></p>
    <p><a href="prodotti.html">Prodotti</a></p>
    <p><a href="contatti.html">Contatti</a></p>
</div>
```

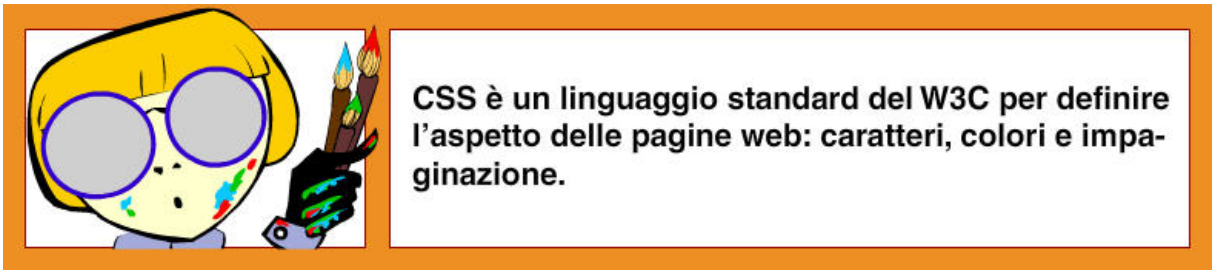
I paragrafi servono a contenere delle frasi, mentre i div sono contenitori generici. Se si pensa che un menù di navigazione è un elenco (quindi una lista) di link, il codice sopra diventa per logica:

```
<ul id="menu">
        <li><a href="home.html">Home page</a></li>
    <li><a href="prodotti.html”>Prodotti</a></li>
    <li><a href="contatti.html”>Contatti</a></li>
</ul>
```

Modulo 3: La presentazione di una pagina web

Indice del modulo

Lezione 1: Introduzione ai fogli di stile CSS



I CSS (**C**ascading **S**tyle **S**heet, Fogli di stile a Cascata) sono un **insieme di regole** che istruiscono il browser su come rappresentare i contenuti del documento XHTML.

Ma cosa gestiscono i CSS?
I CSS gestiscono:

- il colore;
- le famiglie di caratteri;
- l'interlinea;
- la spaziatura di lettere e parole;
- gli stili per titoli e paragrafi;
- la giustificazione;
- i margini;
- i bordi;
- gli sfondi;
- l'impaginazione.

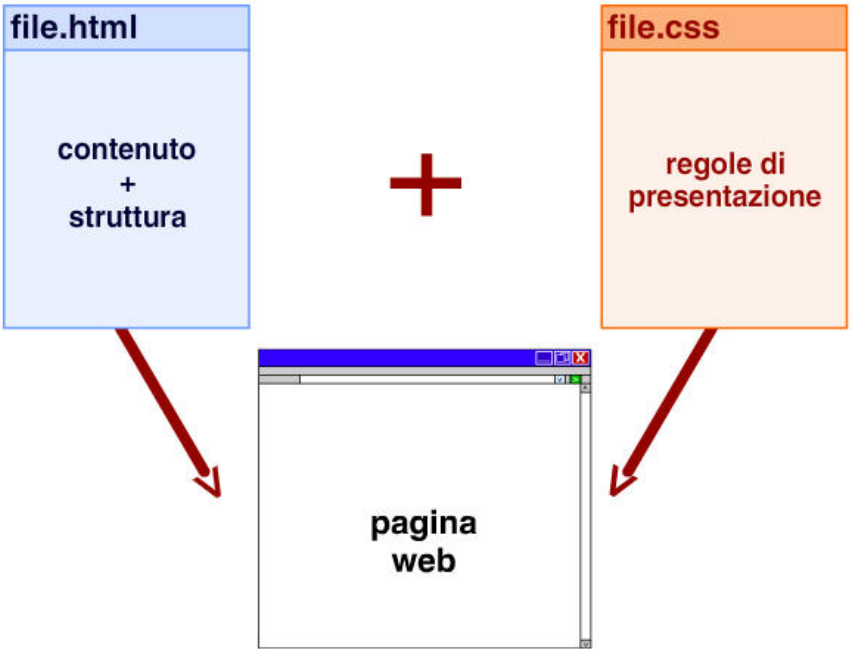
La specifica di riferimento è il CSS 2, approvata nel 1996. Il CSS3 è in fase di sviluppo.

La normativa sull'accessibilità varata di recente richiede l'uso del linguaggio **XHTML 1.0 Strict** associato a **CSS2.** È la soluzione migliore disponibile oggi: **garantisce la separazione** tra i **contenuti** veri e propri della pagina e la **presentazione**.

> < L'XHTML si occupa del contenuto, il CSS della presentazione >

Ci sono diverse sintassi per specificare l'uso di uno (o più) fogli CSS nel documento XHTML. In questo corso parleremo solo dei CSS esterni (**linked**). Nei linked CSS il foglio di stile è definito in un file separato dal documento XHTML che lo usa.

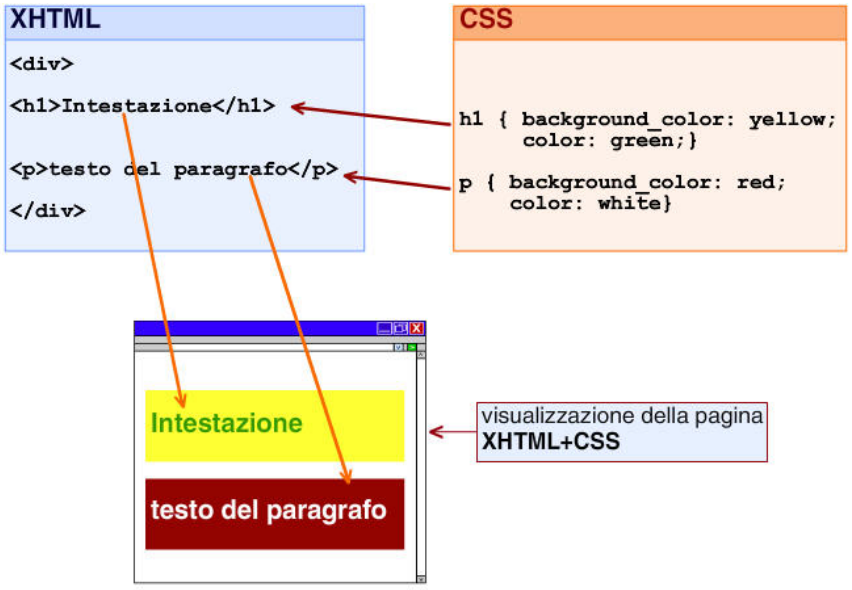
Ogni documento CSS esterno è un file con estensione **.css** con le regole per la presentazione dei contenuti definiti nel file **.html**.



Lezione 2: A cosa servono i CSS

Tramite i CSS si può definire in che modo il browser deve rappresentare i contenuti: i diversi elementi e le sezioni di una pagina web, definiti con XHTML.

Esempio



Lezione 3: I vantaggi

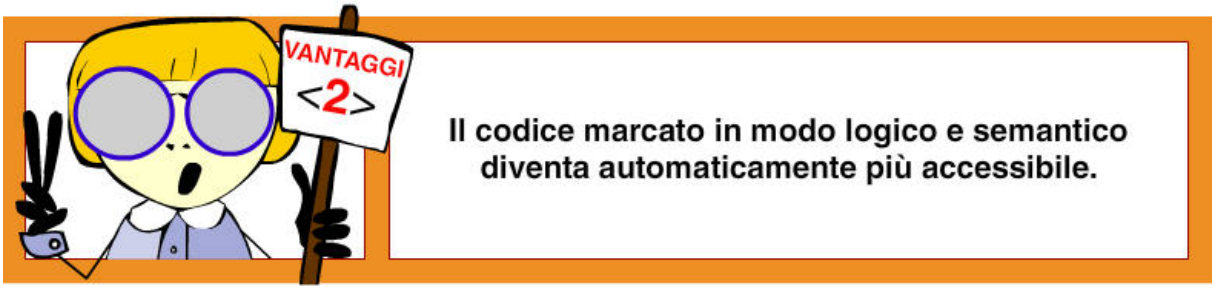
Separare il codice (struttura e contenuto) dalle regole di rappresentazione (stili), con il linguaggio XHTML insieme ai CSS, presenta diversi vantaggi.

Separare le regole di presentazione dall'XHTML per raggrupparle in un foglio di stile esterno significa semplificare il codice della pagina (che risulta di molte righe in meno).

Collegare più pagine XHTML allo stesso foglio di stile significa migliorare le performance di un sito, dal momento che il foglio è caricato una sola volta e **rimane pre-caricato** nella **cache** del **browser** .

Pagine più leggere e fogli di stile pre-caricati comportano:

- per gli utenti (in particolare con connessioni lente) la riduzione dei tempi/costi di connessione necessari per lo scaricamento;
- per i gestori dei siti la riduzione dei costi di hosting, poiché viene ridotta l'occupazione di banda (cioè la quantità di byte trasmessi dal server in un tempo dato).



Per l'accessibilità è fondamentale che i diversi tipi di presentazione di una pagina riescano a fornire, se non proprio lo stesso contenuto, almeno un suo valido equivalente.

Il primo passo, per evitare vincoli ad adeguate presentazioni alternative di un documento, consiste nell'**eliminare** dall'XHTML **gli elementi e gli attributi di presentazione**. Un utilizzo corretto degli elementi di strutturazione fa sì che i browser garantiscano una migliore navigazione della pagina.

L'uso dei CSS consente di creare presentazioni alternative di una stessa pagina, ciascuna con caratteristiche diverse che soddisfano diverse esigenze di visualizzazione, lasciando del tutto invariato il contenuto del documento XHTML.

Utilizzare diversi fogli stile CSS per ogni presentazione alternativa permette di personalizzare il layout di un sito senza doverne creare diverse versioni.

Lezione 4: Il supporto dei browser

I CSS non sono compatibili con browser datati come Netscape 4 o Internet Explorer 4. Alcuni, inoltre, supportano i CSS in maniera imperfetta o errata, fornendo un aspetto della pagina a volte confuso e poco usabile. Questo perché non tutti i browser sono stati sviluppati tenendo conto degli standard web del W3C, di conseguenza non supportano bene i CSS.

Come deve comportarsi lo sviluppatore?

È importante sapere che i siti che utilizzano i CSS non sono davvero incompatibili con i vecchi browser: i contenuti della pagina vengono sempre visualizzati. **È il design che non è compatibile.** Questo comporta una cattiva resa, ma una completa accessibilità dei contenuti, anche se in forma a volte brutta o poco usabile.

Bisogna precisare che ci sono due approcci progettuali per gestire l'impaginazione con i CSS:

- "**approccio di transizione**"
  utilizza tabelle html a scopo di impaginazione, misto a CSS, e mantiene la compatibilità del design anche con browser più datati

- "**approccio radicale**"
  più rigoroso, abbandona completamente le tabelle html per scopi di impaginazione, utilizzando totalmente le possibilità offerte dai fogli stile.

Gli esempi di questo corso, come i modelli XHTML/CSS messi a disposizione per le scuole, seguono l'approccio radicale, cioè senza tabelle di impaginazione. Come già detto, non viene assicurata la compatibilità del design nei browser più datati, ma l'importante è che i contenuti siano comunque accessibili, non che l'estetica sia identica in tutti i browser.

Lezione 5: Collegare i CSS a una pagina XHTML

I fogli di stile esterni sono richiamati, nel documento XHTML, tramite l'uso dell'elemento **<link>** e del suo attributo **rel**.

L'elemento **<link>** si trova all'interno della sezione **<head>** del documento.

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

L'attributo **rel** può avere due valori:

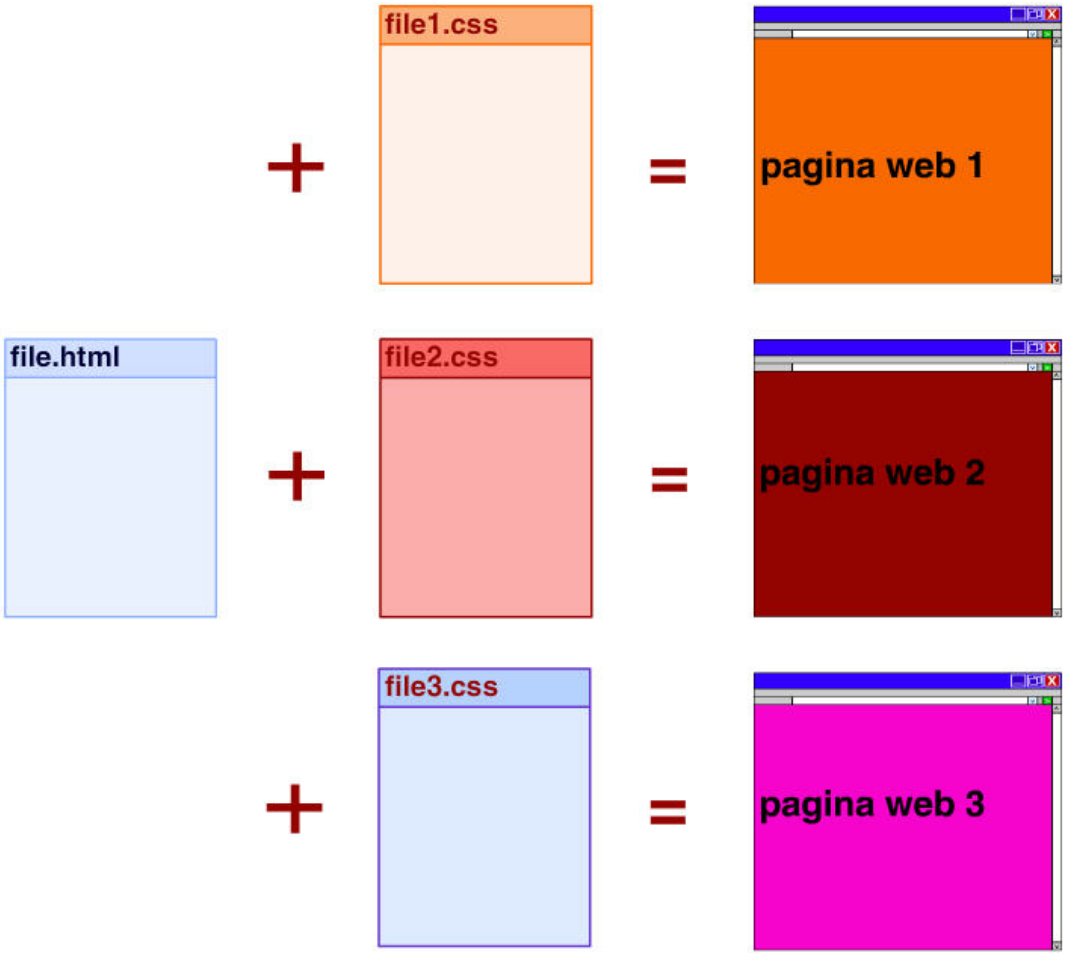
- **stylesheet**
  Identifica il foglio di stile preferito: è attivo quando viene caricata la pagina.
  Viene disattivato quando l'utente cambia la visualizzazione e sceglie uno stile alternativo

- **alternate stylesheet**
  Identifica il foglio di stile alternativo a quello preferito: quando la pagina viene caricata è disattivo. Viene attivato quando l'utente lo seleziona.

Esempio

```
<head>
<link rel="stylesheet" type="text/css" href="stile_grafica.css" />
<link rel="alternate stylesheet" type="text/css"
href="stile_altaleggibilità.css" />
<link rel="alternate stylesheet" type="text/css"
href="stile_solotesto.css" />
</head>
```

Quindi è l'elemento **<link>** e il suo attributo **rel** che possono collegare a un file XHTML diversi stili alternativi. Ciò consente all'utente di scegliere tra diverse visualizzazioni del file XHTML.



Lezione 6: La struttura di un CSS

Un foglio di stile è un file con estensione **.css** che elenca:

- un insieme di regole di rappresentazione;
- note di commento (non necessarie).

I commenti devono essere delimitati tra questi segni:

`/*` (come segno di apertura)

`*/` (come segno di chiusura).

Esempio

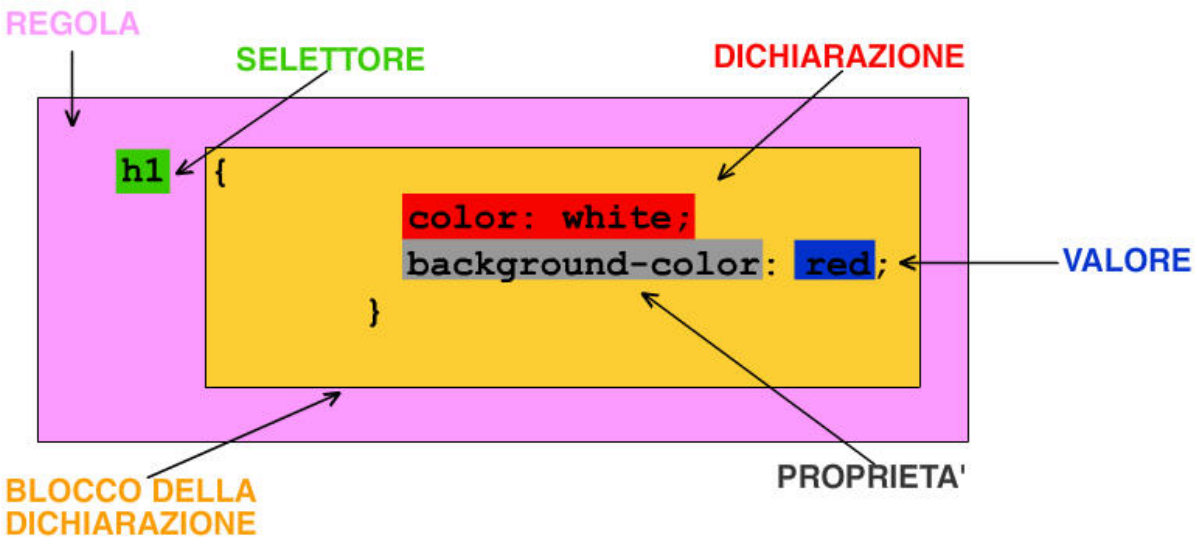
commento

```
/* Definisco le regole per il titolo di primo livello */
```

regola

```
h1    {
          color: red;
}

/* Paragrafo generico */

p     {
          color: green;
          font-size: 0.9 em;
}
```

Lezione 7: Come è fatta una regola



Il **selettore** serve a definire la parte del documento XHTML a cui verrà applicata la regola.

La **proprietà** definisce un aspetto dell'elemento da modificare (margini, colore di sfondo, ecc.) secondo il valore (o i valori) espressi.

Una **regola** è costituita da uno o più **selettori** cui associare il **blocco delle dichiarazioni**.
(che è delimitato da due parentesi graffe **{}**)

Ogni **dichiarazione** è composta da una coppia: **proprietà** e **valore**.
(le varie dichiarazioni sono separate da un punto e virgola **;**)
(proprietà e valore devono essere separati dai due punti **:**)

Lezione 8: I selettori principali

Come abbiamo visto, i selettori indicano al browser a quali elementi della pagina si dovranno applicare le dichiarazioni della regola.

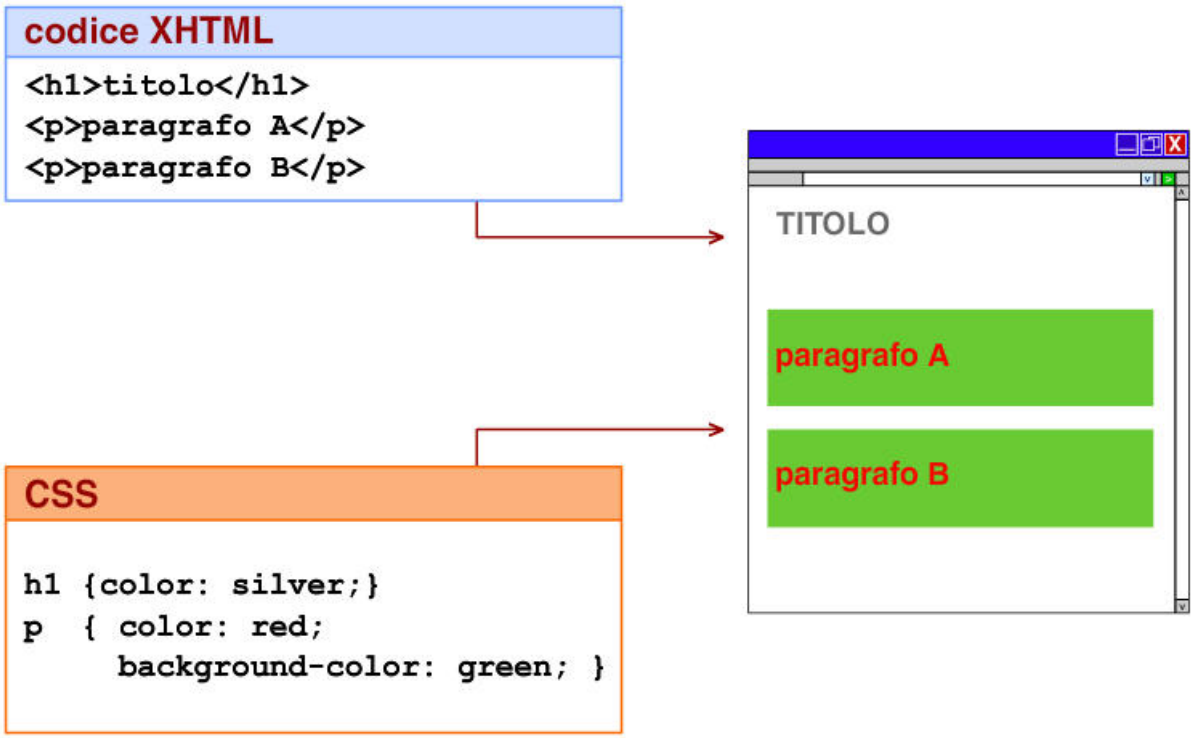
I principali tipi di selettori sono:

- **selettori di elementi**
  Esempio: `h1 {color: green;}`
- **selettori classe**
  Esempio: `.testoverde {color: green;}`
- **selettori id**
  Esempio: `#testata {color: green;}`
- **selettori del discendente**
  Esempio: `#testata p {color: green;}`

Selettori di elementi

È il più semplice e generico. È costituito da uno qualunque degli **elementi di XHTML** e specifica che la regola definita deve essere applicata a tutti gli elementi del tipo indicato.

Esempio



È possibile raggruppare nei CSS diversi selettori per semplificare il codice. Tutti quelli raggruppati vanno separati da una **virgola**.

Esempio

Codice XHTML

```
h1, h2, h3 {background: green;}
```

Selettori classe

Il selettore di tipo classe definisce le regole di presentazione di qualsiasi elemento XHTML (ad esempio **<p>**) specificandone il nome attraverso l'attributo class.

Nel codice XHTML l'attributo **class** è applicabile a qualsiasi elemento.

Esempio

Codice XHTML

```
<h1>titolo</h1>
<p class=”testogiallo”>paragrafo A</p>
<p>paragrafo B</p>
```

Il valore dell'attributo **`class (”testogiallo”)`** deve trovare una corrispondenza in un foglio di stile. È quindi necessario creare un selettore di questo tipo nel foglio di stile, utilizzando il valore assegnato all’attributo **`class (”testogiallo”)`**, preceduto da un punto “.” : **`.testogiallo`**

CSS

```
h1 {
    color: silver;
   }

 p {
    color: red;
    background: green;
    }

.testogiallo {
    color: yellow;
    background-color: black;
    }
```

Ecco il risultato

Selettori id

L'attributo **id** è usato per identificare in **modo univoco** un elemento. Quindi se assegniamo ad un elemento l'id "menu" non sarà più possibile utilizzare questo valore nel resto della pagina. Questo attributo è applicabile a qualsiasi elemento.

Creando un selettore **id** nel foglio di stile è possibile definire le regole di presentazione di qualsiasi elemento (es. <**div**>), specificandone il nome attraverso l'attributo**.**

Esempio

Codice XHTML

```
<h1>titolo</h1>
<div id="menu">
  <ul>
    <li><a href="#">voce 1</a></li>
    <li><a href="#">voce 2</a></li>
  </ul>
</div>
<p>paragrafo B</p>
```

Il valore dell'attributo **`id ("menu")`** deve trovare una corrispondenza in un foglio di stile. Quindi bisogna creare un selettore di questo tipo nel foglio, utilizzando il valore assegnato all'attributo **`id ("menu")`**, preceduto dal simbolo "**`#`**" : **`#menu`**

CSS

```
h1 {
    color: silver;
   }

p   {
    color: red;
    background: green;
    }

#menu {
    color: white;
    background-color: black;
    }
```

Ecco il risultato

Selettore del discendente

Il selettore di discendenza permette di selezionare un elemento che è discendente di un altro specificato nella regola. Il selettore va letto da destra a sinistra.

Un elemento è discendente di un altro se contenuto al suo interno, a qualsiasi livello.

Nell'esempio che segue, la regola definisce che tutti i paragrafi **<p>** contenuti nell'elemento ** ** siano scritti in bianco su sfondo nero (indipendentemente dal livello della relazione di discendenza).

Esempio

Codice XHTML

```
<h1>titolo</h1>
<p>paragrafo A</p>
<div id="footer">
   <h2>sottotitolo></h2>
   <p>paragrafo B</p>
 </div>
```

Questa regola definisce che tutti i paragrafi **<p>** contenuti all'interno di un elemento con **<id="footer">** siano scritti in bianco su sfondo nero.

CSS

```
h1 {
    color: silver;
   }

p  {
    color: red;
    background: green;
    }

#footer {
     color: black;
     background-color: silver;
    }

#footer p {
    color: white;
    background-color: black;
    }
```

Ecco il risultato

Lezione 9: Le pseudo-classi

Con le **pseudo-classi** si può impostare uno stile per un elemento al verificarsi di certe condizioni.

Codice XHTML

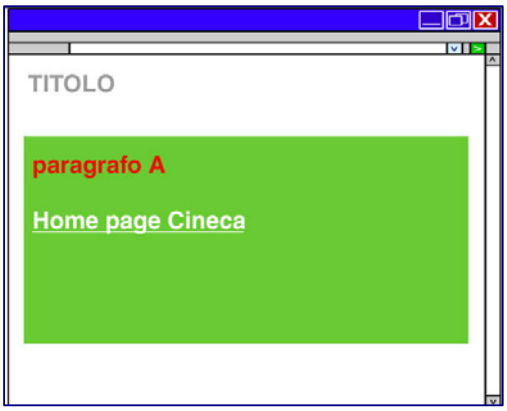
```
<h1>titolo</h1>

<p>paragrafo A <br />
  <a href=”http://www.cineca.it">Home page Cineca</a> </p>
```

Queste regole definiscono che i link della pagina (**a**) non visitati (**:link**) siano bianchi, mentre quelli visitati (**:visited**) siano di colore nero. Quando il cursore è posizionato sul link (**:hover**), il suo colore è giallo.

CSS

```
h1 {
    color: silver;
    }

p   {
    color: red;
    background: green;
    }

a:link {
    color: white;
    }

a:visited {
    color: black;
    }

a:hover {
    color: yellow;
    }
```

Ecco il risultato



Lezione 10: L'ereditarietà

Una caratteristica fondamentale dei CSS è l'**ereditarietà**: le regole applicate a un elemento valgono anche per i suoi discendenti fino a quando, per elementi discendenti, non ne vengono impostate altre con proprietà stilistiche differenti.

In questo esempio definiamo il colore rosso per gli elementi **`<div>`**. Tutti i suoi discendenti **`<h1>`** e **`<p>`** ereditano il testo scritto nello stesso colore. Associando la classe **`.testo_verde`** a **`<p>`** il "paragrafo A" risulterà in verde.

Codice XHTML

```
<div>
<h1>Titolo</h1>
<p class="testo_verde">paragrafo A</p>
<p>paragrafo B</p>
</div>
```

CSS

```
div {color: red;}
.testo_verde {color: green;}
```

Ecco il risultato

Lezione 11: Il testo: proprietà di base

Le proprietà dei CSS2 che permettono di gestire l'aspetto del testo sono numerose.
Vediamo le principali.

- **color**

Definisce il colore del testo di un elemento. Si applica a tutti ed è ereditata.
Per definire lo sfondo si utilizza la proprietà background.

Esempio

```
p { color: red;
 background-color: black;}
```

- **font-size:**

Definisce la dimensione del carattere.
Si applica a tutti gli elementi ed è ereditata.

I valori delle dimensioni del font possono essere espressi in vari modi.

La dimensione espressa con **valori assoluti** non dipende da nessun altro elemento ed è quella definita dall'unità di misura usata.

Esempio

```
p {font-size: 12px;}
```

In **valori relativi** significa che viene calcolata in base alla dimensione del testo dell'elemento parente (è sempre preferibile utilizzare questa tipologia di valori).

Esempio

```
      h2 {font-size: 1.2em;}
```

- **font-family:**

Definisce il tipo di carattere.
È possibile così definire un elenco di caratteri alternativi.
Si applica a tutti gli elementi ed è ereditata.

Esempio

```
h2 {font-family: Arial, Verdana, sans-serif;}
```

Quando la pagina viene caricata, il browser tenta di usare il primo font della lista. Se non è disponibile usa il secondo. In mancanza anche di questo viene utilizzato il font principale della famiglia sans-serif presente sul sistema.

Quindi **non bisogna mai dimenticare** di indicare una delle 5 famiglie generiche:

- `serif (Times New Roman)`
- `sans-serif (Arial)`
- `cursive (Comic Sans)`
- `fantasy (Allegro BT)`
- `monospace (Courier)`

- **font-variant:**

Consente di trasformare il testo in maiuscoletto.
La proprietà è ereditata.

Esempio

```
h2 {font-variant: small-caps;}
```

- **font-style:**

Si applica a tutti gli elementi ed è ereditata.
Imposta le caratteristiche del testo in base ad uno di questi valori:
normal, il testo mantiene il suo aspetto normale
italic, formatta il testo in corsivo

Esempio

```
h2 {font-style: italic;}
```

- **font-weight:**

Definisce la consistenza o "peso" visivo del testo.
Si applica a tutti gli elementi ed è ereditata.

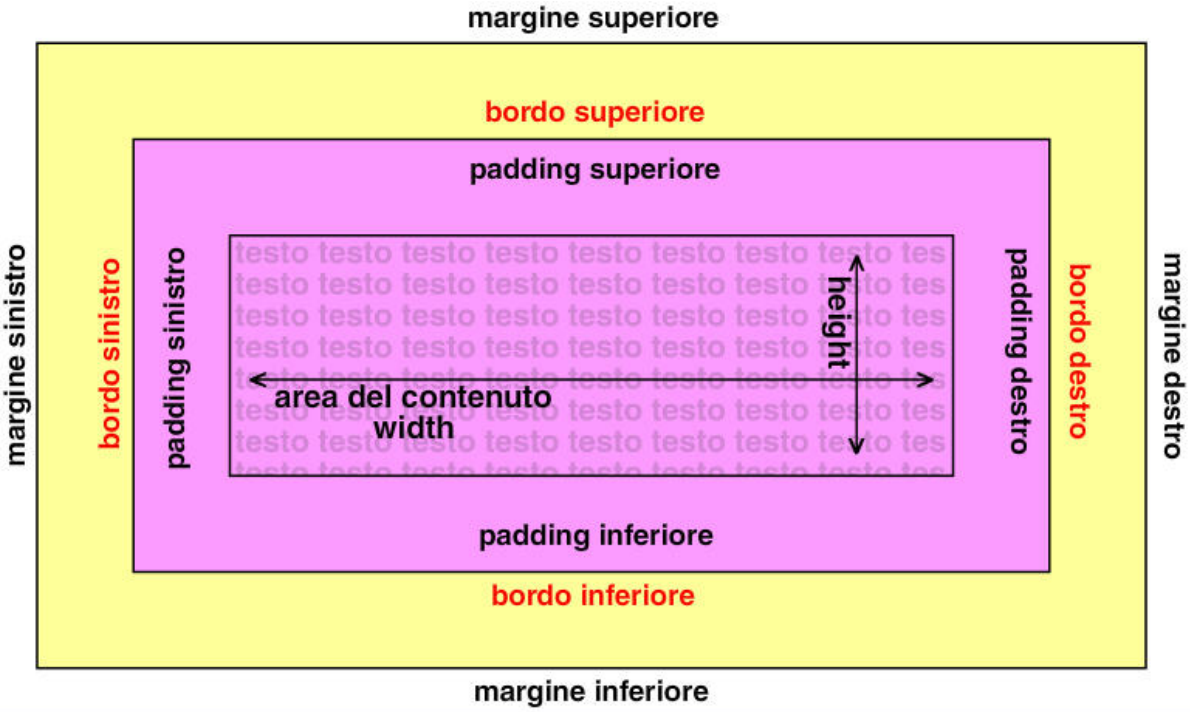
Esempio

```
div {font-weight: bold;}
```

Lezione 12: il box model

Una pagina XHTML è definibile come un insieme di box (elementi blocco e/o di elementi inline).

Per ogni elemento xhtml viene creata idealmente una scatola (box), che lo contiene e che ha alcune proprietà, tutte gestibili attraverso il foglio di stile. Si possono definire i bordi, il padding, il margine, l'ampiezza, ecc. Per capire l'insieme di regole che gestisce l'aspetto visuale della scatola, facciamo riferimento al **box model.**



Ogni box è caratterizzato da:

- **width**
  È la larghezza dello spazio destinato ai contenuti. Se non viene specificata il box si allarga (margini compresi), per riempire tutto lo spazio a sua disposizione.

- **height**
  È l'altezza dello spazio destinato ai contenuti. Se non viene specificata il box si espande in verticale il minimo possibile, per ospitare il contenuto.

- **padding**
  È uno spazio vuoto, che può essere creato tra l'area del contenuto e il bordo del box.

- **border**
  È una linea che circonda la zona del padding e l'area del contenuto. I bordi sono definiti da tre aspetti: lo stile, il colore e lo spessore. Se non specificato, il bordo non viene visualizzato.
  Il colore, se non specificato, coincide con quello impostato per il testo interno al box.

- **margin**
  È uno spazio fra il bordo e gli altri oggetti, che separa un dato elemento da quelli adiacenti.

Esempio

Codice XHTML

```
<div>
<p>testo contenuto nel box</p>
</div>
```

CSS

```
div {
    color: red;
    background-color: yellow;
    width: 300px;
    height: 200px;
    padding: 40px 20px 40px 20px;
    border:4px solid green;
    margin: 10px;
    }
```

Ecco il risultato

Lezione 13: posizionamento

Tipi di elementi xhtml e loro comportamento

Per realizzare il layout di pagine web è fondamentale la definizione del posizionamento degli elementi (dei box) all'interno della pagina e il loro comportamento in relazione reciproca.

Vediamo i principali tipi di elementi:

- **elementi blocco XHTML**, ad esempio `<p>, <h1>,<div>, …`(box a livello di blocco)
  Si dispongono in verticale uno dopo l'altro, distanziandosi in base ai margini. Gli elementi vanno a capo e mandano a capo ciò che segue.

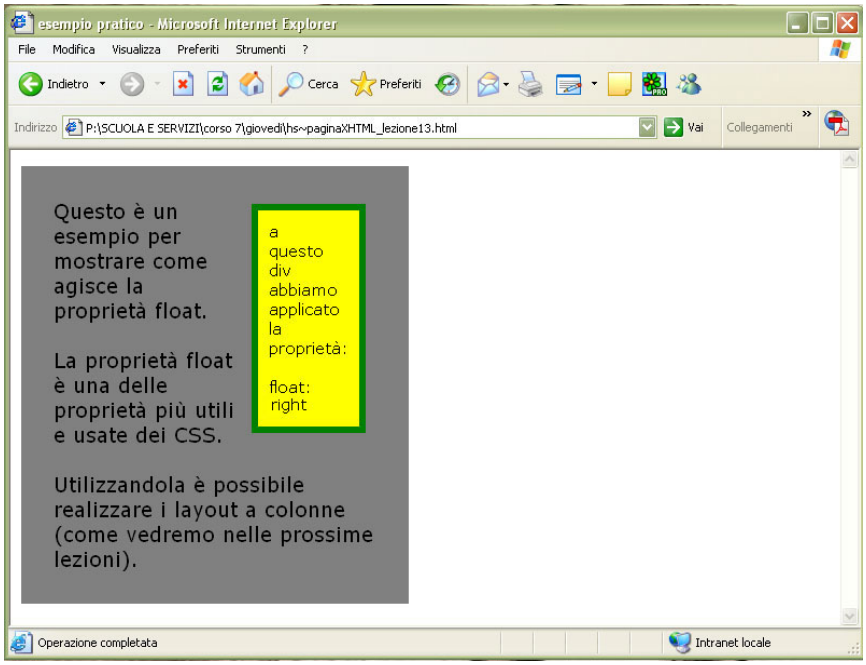
- **elementi in line XHTML** , ad esempio `<b>, <a>, <span>, …`(box in linea)
  Si dispongono orizzontalmente uno di seguito all'altro. La loro spaziatura è determinata anche da margini, bordi e padding. Non vanno a capo e non mandano a capo ciò che segue.

Se non è specificato nel foglio stile, il posizionamento dei box assume le caratteristiche sopra citate.

La proprietà float

Attraverso la proprietà **float** si può rimuovere un elemento dal normale flusso del documento e spostarlo su uno dei lati (destro o sinistro) del suo box contenitore. Il contenuto che lo circonda gli scorre intorno sul lato opposto a quello indicato come valore di float.

Il nostro obiettivo è realizzare il design di questa figura:



La proprietà **float** può assumere i seguenti valori:

- **left**.
  L'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra.

- **right**.
  L'elemento viene spostato sul lato destro, il contenuto scorre a sinistra.

- **none**.
  Valore iniziale, in mancanza di una dichiarazione l'elemento mantiene la sua posizione normale.

Con la proprietà **clear** si può impedire che al fianco di un elemento ne compaiano altri con il float.

Esempio

Codice XHTML

```
<div class="esempio">
<div class="box_destro">
   <p>
     a questo div abbiamo applicato
     la proprietà: <br />
           float:  right
   </p>
</div>
      <p>
        Questo è un esempio per mostrare come agisce
        la proprietà float. <br /><br />
        La proprietà float è una delle proprietà più
        utilizzate e usate dei CSS.<br /><br />
        Utilizzandola è possibile realizzare i layout
        a colonne (come vedremo nelle
        prossime  lezioni).
      </p>
</div>
```

CSS

```
.esempio {
   font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
       font-size: 1.2em;
   background-color: Gray;
       color: Black;
       width: 40%;
       margin: 0px;
       padding: 30px;
}
.box_destro{
       font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
       font-size: 0.8em;
       float: right;
       width: 20%;
       background-color:Yellow;
       color: Black;
       margin: 5px;
       padding: 10px;
       border: 4px solid Green;
}
```

La proprietà position

La proprietà **position** è fondamentale per la gestione della posizione degli elementi.
Può assumere i seguenti valori:

- **static**
  È il valore predefinito per tutti gli elementi per i quali non è stata specificata una posizione. Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.

- **absolute** Con questo valore è possibile rimuovere completamente il box dell'elemento dal flusso e posizionarlo in punto esatto della pagina. Il box posizionato in modo assoluto non ha alcuna influenza sulla posizione degli altri. La posizione viene definita in base alle coordinate fornite con le proprietà **top**, **left**, **right** o **bottom** a partire da un punto di riferimento iniziale. Il box di riferimento è quello dell'elemento contenitore (quello dell'elemento 'padre').

- **fixed** Usando questo valore, il box dell'elemento viene sottratto al normale flusso del documento. Per il box posizionato con fixed il contenitore da utilizzare come punto di riferimento è sempre il cosiddetto *viewport*, cioè la finestra principale del browser, l'area del contenuto. La posizione viene definita con le proprietà **top, left, bottom, right** . Il box così posizionato non scorre con il resto del documento: rimane fisso al suo posto.

- **relative**
  Con questo valore è possibile alterare il normale flusso della pagina. L'elemento viene posizionato rispetto al suo box contenitore. La posizione viene definita con top, left, bottom, **right** che indicano quant'è lo spostamento in senso orizzontale e verticale rispetto al contenitore.

Modulo 4:Progettiamo un layout con i CSS

Indice del modulo

Lezione 1: La struttura logica e il layout

Il primo passo per realizzare una pagina web è strutturare logicamente i contenuti, creando una gabbia (modello) che costituisca la struttura.

Quindi per progettare una pagina bisogna individuarne concettualmente le principali sezioni logiche:

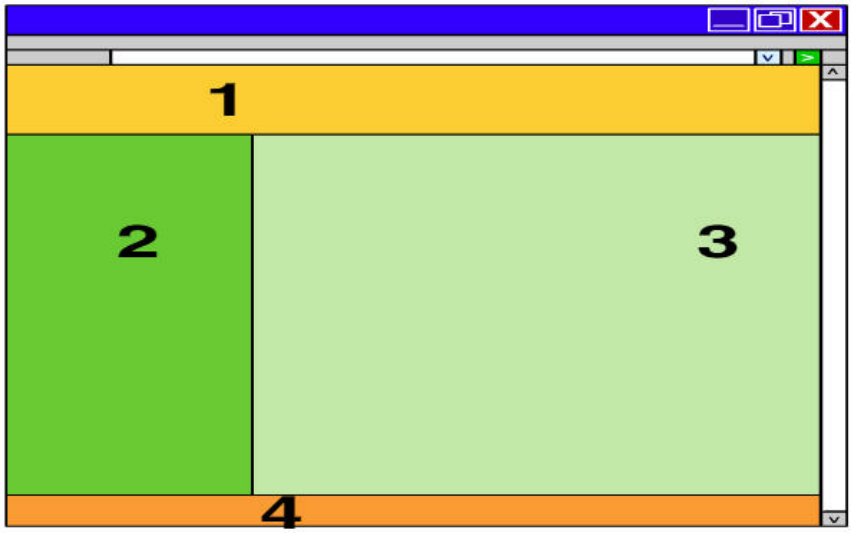
- **testata** (header): in questa sezione di solito ci sono il nome del sito e il logo
- **navigazione**: comprende i menù di navigazione
- **contenuti**: è la parte principale della pagina
- **piè di pagina** (footer): è sezione disposta a fondo pagina.

Solo dopo aver progettato i contenuti in sezioni logiche si può scegliere il tipo di layout.

Lezione 2: I layout

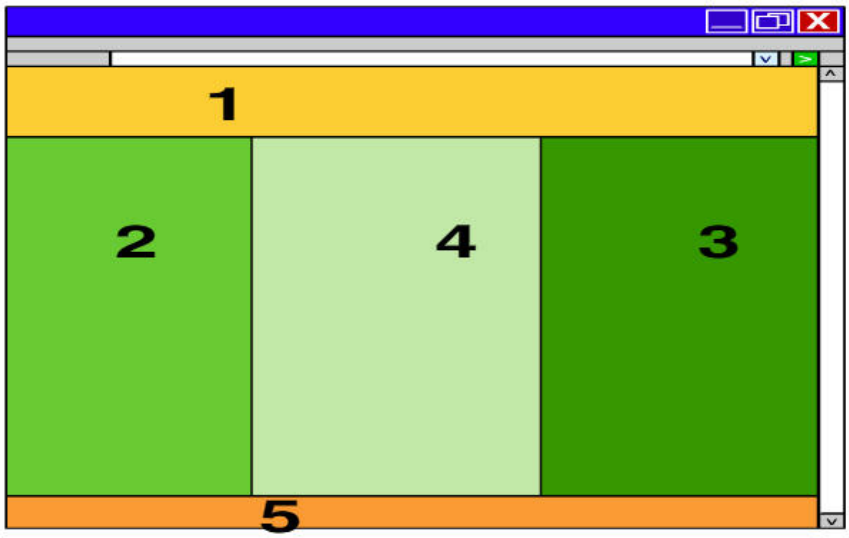
Le gabbie (o modelli) standard più diffuse sono:

- **il Layout a 2 colonne (con testata e piè di pagina)**

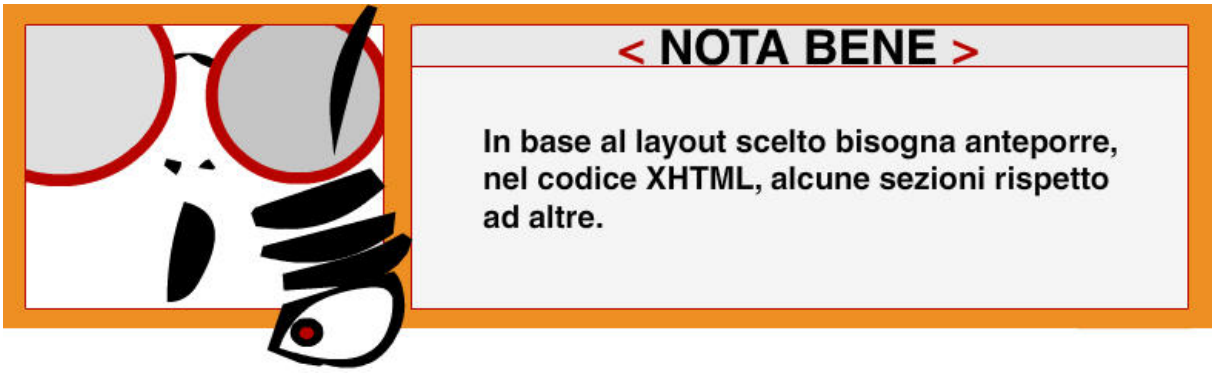


L'ordine delle sezioni nel codice XHTML coincide con quello di visualizzazione.

- **il Layout a 3 colonne (con testata e piè di pagina)**



L'ordine delle sezioni nel codice XHTML è differente in quello di visualizzazione.



Lezione 3: Layout fissi e layout fluidi

Per ogni layout ci sono due categorie principali, a seconda della sua estensione orizzontale:

- **layout fisso**
  il contenitore principale ha la larghezza dimensionata in pixels

- **layout fluido**
  sono quelli che variano al variare della larghezza della finestra del browser.

L'effetto fluido si può ottenere in svariati modi, ad esempio non specificando in nessun modo la larghezza del contenitore principale né usando margini, bordi o padding. In tal caso si ha un layout fluido totale.

Lezione 4: Esempio pratico: Layout a 2 colonne con float

Grazie all'uso della proprietà **float** si può realizzare un layout a colonne, posizionando i box uno di fianco all'altro. Vediamo come creare un layout a due colonne con i float.

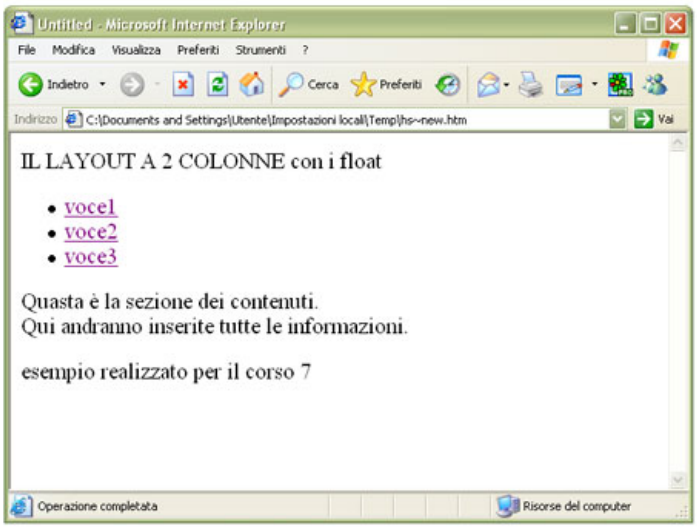
Codice XHTML di base (gabbia)

```
<body>
    <div id="pagina">
        <div id="testata"></div>
        <div id="colonna1_navigazione"></div>
        <div id="colonna2_contenuti"></div>
        <div id="piede_pagina"></div>
    </div>
</body>
```

Come si può notare, la struttura rispecchia le sezioni logiche della nostra pagina:

- testata;
- colonna per il menù di navigazione;
- colonna per i contenuti;
- piè di pagina..

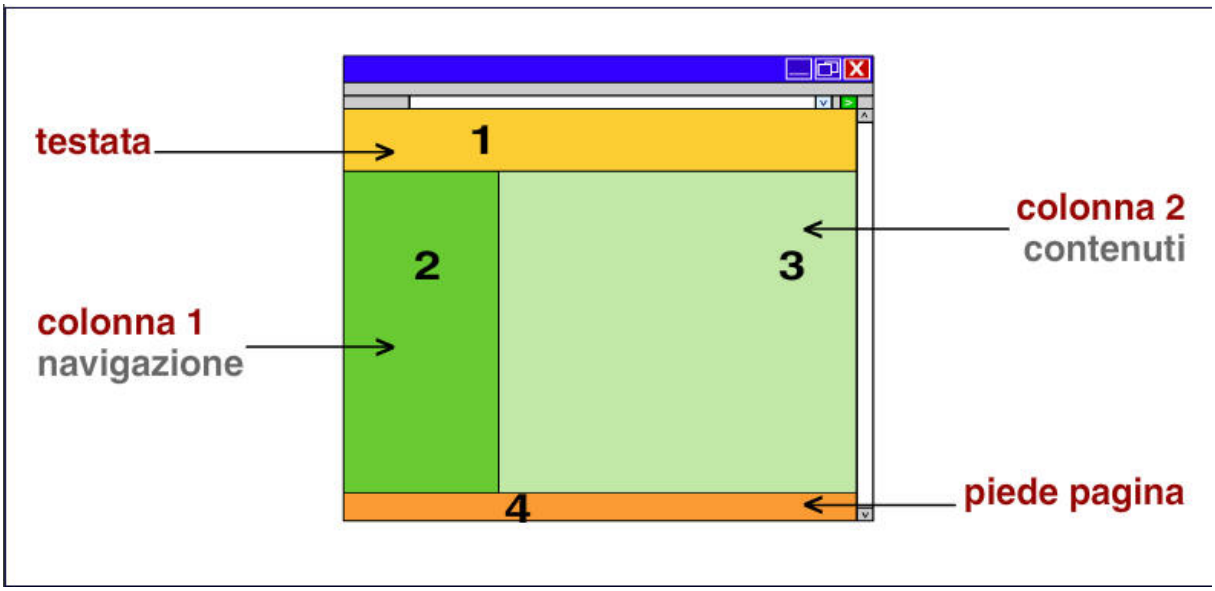
Per contenere le sezioni principali utilizziamo un contenitore per tutta la pagina **( )**che ci permette di gestirne l'estensione orizzontale e gli sfondi delle colonne. Aggiungendo un po' di contenuti otteniamo questa pagina, che risulta senza stile perché non abbiamo ancora scritto il file **foglio_di_stile.css**.



Codice XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
<head>
<title>esempio pratico</title>
<link rel="stylesheet" type="text/css" href="foglio_di_stile.css" />
</head>

<body>
    <div id="pagina">
        <div id="testata">
        <h1>IL LAYOUT A 2 COLONNE con i float</h1>
        </div>
        <div id="colonna1_navigazione">
           <ul>
                 <li><a href="#">voce1</a></li>
               <li><a href="#">voce2</a></li>
               <li><a href="#">voce3</a></li>
           </ul>
        </div>
        <div id="colonna2_contenuti">
              <p>
             Quasta è la sezione dei contenuti.<br />
              Qui andranno inserite tutte le informazioni.</p>
              <p>
              testo da inserire<br />
              testo da inserire<br />
              testo da inserire<br />
              testo da inserire<br />
              testo da inserire<br />
              </p>
       </div>
       <div id="piede_pagina">
          <p>
              esempio realizzato per il corso 7
          </p>
       </div>
    </div>
</body>

</html>
```

Procediamo scrivendo il foglio di stile, con le regole necessarie per ottenere questo tipo di layout.

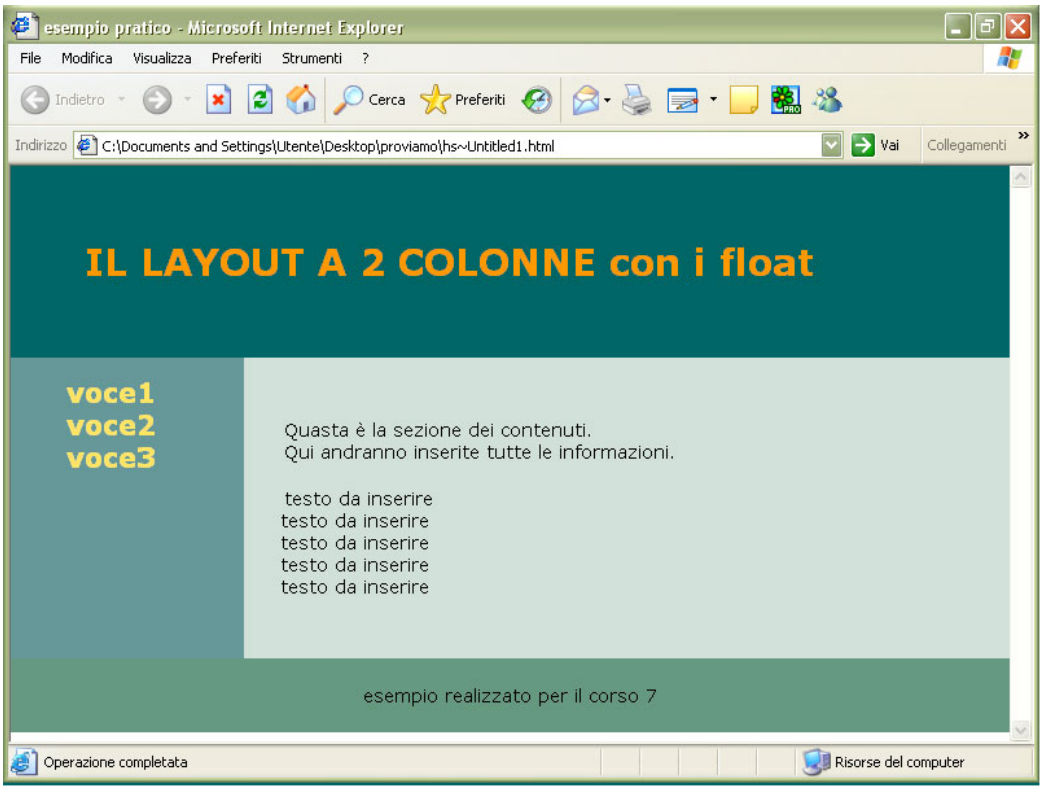


Ecco il foglio di stile **CSS** che definisce la visualizzazione della nostra pagina:

```
/*stili per il layout fluido*/
html,body {
   margin: 0;
   padding:0;
   }
body    {
   font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
   font-size: 76%
   }
#pagina {
    background-color: #669899;
   }

/*stili per la testata*/
#testata{
   background-color: #006666;
   color: #FF9900;
   }
#testata h1   {
   margin: 0;
   padding: 2em;
   font-size: 2em;
   font-weight: bold;
   }

/*stili per la colonna per il menu di navigazione*/
#colonna1_navigazione    {
```

```
   float:left;
   width: 180px;
   }

/*stili per la colonna per i contenuti*/
#colonna2_contenuti{
   margin-left: 180px;
   margin-top: 0;
   padding: 1em;
   background-color: #D1E1D9;
   }

/*stili per la il piede pagina*/
#piede_pagina  {
       clear: left;
       margin: 0;
   padding: 1px;
   background-color: #669980;
   color: Black;
   font-size: 1em;
   text-align: center;
   }
/*stili per il menu di navigazione*/
#colonna1_navigazione ul {
   margin: 1em 0 1em 3em;
   padding: 0;
   list-style-type: none;
   }
#colonna1_navigazione li {
   margin: 1em;
   padding: 0
   }
#colonna1_navigazione li a {
   font-weight: bold;
   font-size: 1.5em;
   color: #FFE566;
   text-decoration: none
   }
#colonna1_navigazione li a:visited {
   font-weight: bold;
   font-size: 1.5em;
   color: #AAAB39;
   text-decoration: underline
   }
#colonna1_navigazione li a:hover {
   font-weight: bold;
   font-size: 1.5em;
   color: #99CC66;
   text-decoration: none
   }
```

Ed ecco il risultato



Analizziamo il foglio di stile per capire come abbiamo impostato il layout:

```
html,body {
   margin: 0;
   padding:0;
   }
body    {
   font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
   font-size: 76%
   }
#pagina {
    background-color: #669899;
   }
```

Per ottenerne uno fluido, non abbiamo specificato in nessun modo la larghezza del contenitore ** ** e abbiamo impostato le dimensioni del carattere (**font**) del **<body>** con una misura relativa (**76%**). Così il font è ridimensionabile tramite le opzioni del browser.

Abbiamo inoltre assegnato un colore di sfondo al contenitore ** ** e nessuno al ** ** che eredita il colore di sfondo del ** ** dando l'impressione di estendersi fino al ** ** .

```
#colonna1_navigazione    {
   float:left;
   width: 180px;
   }

#colonna2_contenuti{
   margin-left: 180px;
   margin-top: 0;
   padding: 1em;
   background-color: #D1E1D9;
   }
#piede_pagina  {
       clear: left;
       margin: 0;
   padding: 1px;
   background-color: #669980;
   color: Black;
   font-size: 1em;
   text-align: center;
   }
```

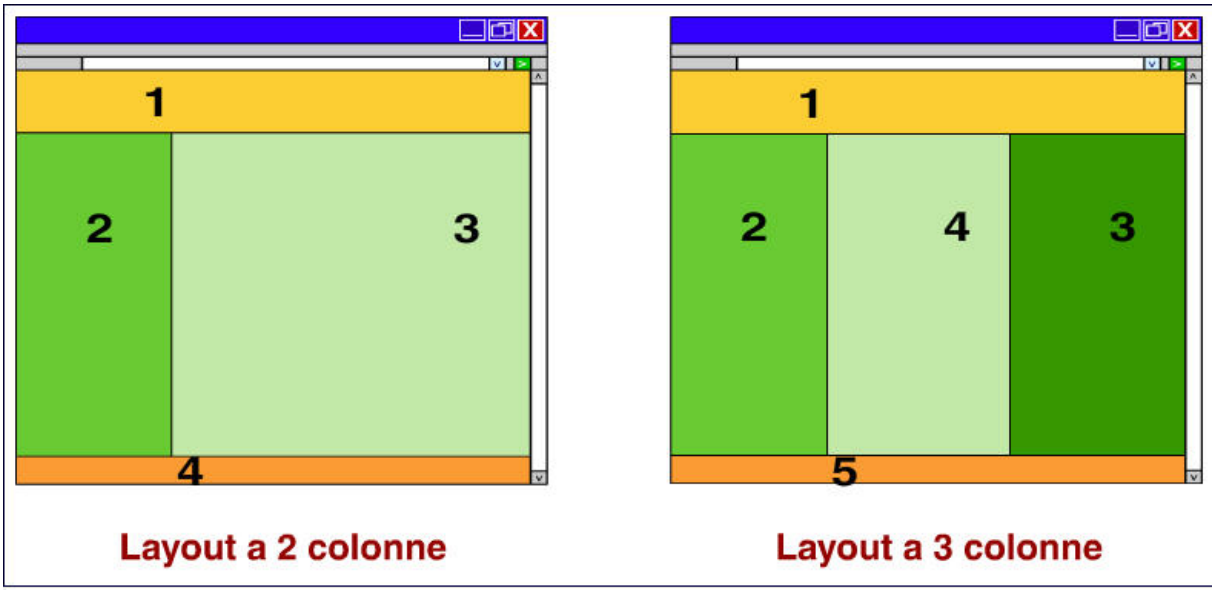
Abbiamo applicato la dichiarazione **float:left** al ** ** e gli abbiamo definito una larghezza di 180px.

Abbiamo definito la proprietà **margin-left** del ** ** con una valore di 180px.

Abbiamo applicato la dichiarazione **clear:left al ** per impedire che la colonna ** ** possa sovrapporsi al ** .**

Lezione 5: Esempi di modelli xhtml/css

Tutti questi esempi sono a disposizione sotto la voce allegati.



.