



Versione stampabile della guida disponibile su
<http://css.html.it/guide/leggi/3/guida-layout-dei-siti-con-i-css/>

Introduzione

Questa guida nasce con l'intento di introdurre il lettore all'impaginazione (layout) dei siti web **senza l'uso di tabelle**. Vedremo nel dettaglio i più comuni layout, le tecniche per realizzarli, i campi d'applicazione, i pro e i contro di ciascuna soluzione. I layout sono stati tutti pensati con la filosofia *semplice è meglio*. La lettura attenta e totale della guida dovrebbe consentire al lettore di **creare** oltre che di capire o usare.

La guida prevede l' introduzione al layout a colonna singola, due e tre colonne. In seguito verrà spiegato come impaginare i contenuti e come preparare un layout per la stampa. A fine guida verranno presentati un indice completo e un pacchetto zip contenente tutti i layout visti nel corso delle lezioni.

Le lezioni presuppongono una discreta padronanza di HTML/xHTML e dei fogli di stile. La codifica del layout è, infatti, una parte fondamentale, ma minima rispetto a quella di un intero sito. Per approfondimenti e richiami ad argomenti specifici e attinenti, sono consigliate le seguenti letture su HTML.it:

- Guida HTML
- Guida XHTML
- Guida ai CSS
- Guida al Web Design
- Guida ai siti ad elevata accessibilità
- Articoli sui CSS della sezione PRO.HTML.it

L' importanza dell' impaginazione

L'impaginazione è uno degli aspetti fondamentali di un sito internet. Dalla diffusione dei fogli di stile, e in seguito alle raccomandazioni del [w3c](#), la tendenza di creare layout senza l'uso di tabelle si è andata man mano consolidando, fino a diventare una realtà in continua espansione.

Ci pare importante chiarire un punto. Sovente, la conoscenza (seppur approfondita) dei fogli di stile non basta per la creazione di un layout robusto senza l'uso delle tabelle. C'è da dire che in rete si trovano moltissimi layout già pronti, alcuni decisamente validi e altri meno. Il loro utilizzo è semplicissimo, e spesso ci si limita ad usarli senza capirli. Ma un buon appassionato di fogli di stile o un webdesigner professionista dovrebbe essere in grado di creare da zero un layout, oppure avere conoscenze tali da poter valutare se un layout già pronto sia valido o meno e se sia quello di cui ha davvero bisogno per realizzare un sito.

Tabelle contro CSS per il layout

Il dibattito tra i sostenitori di tabelle e quelli dei fogli di stile per il layout è ancora molto acceso. Ci sono molti argomenti validi da entrambe le parti, ma attualmente si può affermare quasi con certezza che l'ago della bilancia pende verso i fogli di stile. Vediamo brevemente alcuni aspetti di entrambe le soluzioni:

A favore delle tabelle

- facilità di codifica:
 - Alcuni layout sono molto più semplici da realizzare con le tabelle
 - non necessitano conoscenze particolari dei fogli di stile, basta l'html
- robustezza su praticamente tutti i browsers, anche i più obsoleti
- possibilità di creare facilmente layout anche con editor visuali

A favore dei layout basati sui css

- le tabelle servono per dati tabellari, non per creare layout
- i fogli di stile garantiscono una migliore separazione tra contenuti e presentazione e quindi:
 - alleggerimento del codice html
 - aumento della velocità di caricamento
 - minori spese per la banda
 - restyling e modifiche di layout sono molto più semplici per un sito ben strutturato
- l'uso delle tabelle determina con forza l'ordine del codice html
- i fogli di stile aiutano a mantenere la consistenza di un intero sito web
- fare siti attenenti agli standard e con layout table-less, anche se più difficile, attualmente garantisce la competenza e la competitività di un professionista
- migliore accessibilità e visibilità verso media alternativi: palmari, cellulari, screen readers, text browsers e stampanti

Dalle tabelle al layout table-less

Uno degli aspetti più difficili per i webdesigner è il passaggio a layout senza l'uso di tabelle. Questo passaggio implica sovente un cambiamento radicale nell'approccio di stesura di una pagina html. In questa guida vedremo come rendere questo passaggio il più naturale possibile.

Chi si è avvicinato da poco ai fogli di stile, e più in generale alla creazione dei siti web, troverà facile e logico l'uso dei css per la creazione dei layout in questa guida, a tal punto che le tabelle serviranno solo per dati tabellari e, in fase di creazione di un layout, queste saranno l'ultima alternativa da considerare.

Contenuto, struttura e presentazione

La definizione più conosciuta e diffusa dei fogli di stile è: *sono la separazione tra contenuto e presentazione*. Il contenuto è nella pagina html, mentre la presentazione è riservata ai fogli di stile. Ma c'è una terza variabile che non compare in questa definizione, una variabile di cui molti appassionati ed esperti di webdesign devono tenere conto: la **struttura**. La struttura è inseparabile dal contenuto, ed è l'unico modo per consentirne la presentazione. Per esempio, si immagini una guida di programmi tv in cui le pagine non siano divise per giorni e le programmazioni non suddivise per canale e orario. Una guida ai programmi televisivi con le stesse informazioni testuali, ma presentate in ordine sparso e senza forma. La potreste consultare?

Div, questo misterioso

Un layout table-less si realizza attraverso l'uso dei **div**. La sua definizione è *generic block-level element* ossia contenitore generico block level. Il fatto che sia un elemento block-level ci garantisce il fatto che possa contenere qualsiasi tipo di elemento html. Inoltre, la sua presentazione naturale (quindi senza fogli di stile) è totalmente neutra: infatti questo elemento si presenta di default senza margini, bordi o padding. È quindi il contenitore per eccellenza per realizzare layout senza l'uso di tabelle: **ad ogni div portante verrà associata una sezione della pagina.**

Le sezioni logiche di una pagina web

Il primo passo da fare nella realizzazione di un sito web è pianificare i contenuti. Strutturare logicamente i contenuti di una pagina, creare un **template** (modello) e pensare che tutti i contenuti di un sito saranno facilmente inseribili da subito e in futuri aggiornamenti e mantenimenti del sito, è una fase fondamentale. Vediamo le principali sezioni logiche di un sito web.

L' header (*testata*)

Se volessimo paragonare un sito web ad un libro, l'**header** ne sarebbe la copertina. Generalmente l'header si estende orizzontalmente per tutto lo spazio a disposizione del layout. In verticale l'header si dovrebbe estendere per circa 80, 100 o massimo 150 pixels. È fondamentale anche in questa sezione il contenuto: che sia grafico o testuale, l'header dovrebbe riportare il **nome del sito** e una sorta di **descrizione** o sottotitolo.

È poi diffusa la buona pratica che l'header, oltre che a essere un titolo, sia anche un link che punti alla home page, cosicché da qualsiasi pagina interna del sito, oltre che dal menu di navigazione, sia possibile con un solo click ritornare alla pagina iniziale.

La navigazione

La navigazione, anche detta **menu** è una sezione indispensabile di ogni sito, in quanto permette di accedere ai contenuti. La navigazione principale dovrebbe essere ben visibile, leggibile e distinguibile dai contenuti, e un buon sito internet dovrebbe poter consentire di accedere *da ogni pagina a tutte le altre pagine* senza troppi click e soprattutto senza l'uso dei tasti "indietro" e "avanti" del browsers. È importante ricordare all'utente dove si trova, evidenziando in qualche modo la pagina o sezione corrente.

Per ogni area di contenuti del sito è possibile creare eventualmente una **navigazione secondaria** che si aggiunge o che sostituisce la prima, mentre per pagine con molte sezioni interne distinguibili, come ad esempio una reference, è consigliata l'adozione di una navigazione interna, cioè una sorta di tabella dei contenuti. Per siti con molti link organizzati per aree, si può pensare di disporre la navigazione nella colonna dei contenuti. Infine, i **link esterni al sito** dovrebbero essere facilmente distinguibili, spazialmente e a livello di *look and feel*, dai link interni.

La sezione dei contenuti

È questa la parte principale di un sito. Se è vero che un buon layout e una buona grafica possono fare la differenza al primo impatto, quello che davvero attira un visitatore dopo un primo sguardo sono i contenuti.

Il footer (*pie' di pagina*)

Il footer è generalmente una piccola sezione disposta a fondo pagina e contiene informazioni sullo sviluppatore del sito, sul copyright, i contatti di posta elettronica ed eventualmente indirizzo e numero di telefono se il sito riguarda un'azienda. Il footer dovrebbe essere presente in ogni pagina, ben distinguibile e discreto. A mio avviso, non

dovrebbe essere spazialmente più grande dell'header.

La sezione extra

Questa è la sezione più varia e a seconda della tipologia del sito può essere più o meno estesa o addirittura assente. Ecco cosa può contenere:

- Articoli, pagine o sezioni in evidenza
- News
- Link esterni al sito, sponsor e banner
- Motore di ricerca interno e, facoltativamente, esterno al sito
- Sondaggi e risultati
- form di sottoscrizione a newsletters o servizi
- E, in generale, quello che logicamente non può far parte della navigazione principale o dei contenuti

I più comuni layout table-less

Alle sezioni logiche presentate nella lezione precedente, in ogni sito dovrebbero corrispondere le relative sezioni fisiche: è da qui che nasce l' esigenza del layout. A seconda della portata di informazioni di un sito web nel suo complesso, (o di una pagina tipo) sarà necessario pensare al layout da utilizzare. Ecco i principali layout che vedremo in questa guida:

- **Layout monolitico o a colonna singola**
- **Layout a due colonne**
- **Layout a tre colonne**

Di tutte le tipologie presenteremo diverse varianti e vedremo passo passo come realizzarli. Prima di addentrarci oltre, ci sono ancora due questioni da considerare.

Estensione orizzontale di un layout

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

- **layout fisso**
- **layout fluido**
- **layout elastico**

Le regole del foglio di stile coinvolte sono al massimo un paio per ogni possibile estensione, che vedremo brevemente assumendo che il contenitore principale sia di questo tipo: `<div id="container">`

Layout fluido

Rientrano in questa tipologia tutti i layout che variano larghezza al variare della larghezza della finestra del browser. L'effetto fluido si può ottenere in svariati modi:

- non specificare in nessun modo la larghezza del contenitore principale nè usare margini, bordi o padding, e in tal caso si ha un layout fluido totale;
- attribuire del padding orizzontale al body;
- specificare una larghezza percentuale al contenitore principale;
- attribuire margini, padding o border (fissi o percentuali) al contenitore principale.

Non saprei dare percentuali precise, ma posso affermare che una percentuale significativa di siti web moderni e con impaginazione table-less adotta un layout fluido totale: questo consente infatti di adattare la pagina alle varie risoluzioni e alla larghezza della finestra del browser. Inoltre, un'impostazione fluida consente una stampa migliore di una pagina web: sovente infatti ci si trova a stampare pagine che adottano layout fissi e il risultato della stampa spesso è tagliato su uno o entrambi i lati.

Layout fisso

In un layout fisso, il contenitore principale ha la larghezza dimensionata in pixels. Un layout fisso è generalmente **studiato per una dimensione standard** e vi sono due principali larghezze utilizzate per questo tipo di layout, a seconda delle due più diffuse risoluzioni del monitor:

- 800x600 pixels
- 1024x768 pixels

Generalmente è uso lasciare circa 40-50 pixels come margine di tolleranza, quindi, le due dimensioni utilizzate sono all' incirca **760 pixels** oppure **960 pixels**. Sulla scelta della larghezza da utilizzare, c'è da dire che un layout pensato per una determinata risoluzione risulta decisamente **poco professionale** se visualizzato con una risoluzione minore.

Provate a visualizzare a risoluzione 800x600 un qualsiasi sito che riporta la dicitura *"Sito ottimizzato per una risoluzione di 1024x768"*: la pagina per essere consultata necessiterà di uno scrolling orizzontale, che personalmente ritengo molto fastidioso.

A questo punto la soluzione migliore, se proprio si deve pensare ad un layout fisso, è impostare la larghezza del contenitore principale a 760 pixels o meno. Ecco come ottenere un contenitore centrato e largo 760 pixels:

```
body{
  text-align: center; /*centra in IE 5.x */
}

div#container{
  width: 760px;
  margin: 0px auto; /*centra negli altri browsers*/
  text-align: left; /*ripristina l' allineamento*/
}
```

Seppure, con il passare del tempo si stia procedendo ad un progressivo aumento delle

risoluzioni dei monitor, un sito quale [A List Apart](#), uno tra i più autorevoli a livello internazionale in materia di webdesign, ha adottato un **layout fisso a larghezza 600 pixels**.

Posso solo ipotizzare i due motivi che abbiano spinto [Jeffrey Zeldman](#), fondatore del sito e co-fondatore del [Web Standard Project](#) a una scelta del genere:

- è adatto a qualsiasi risoluzione del monitor dell'utente, anche a 640x480 pixels;
- consente una maggiore leggibilità: è provato infatti che, sia sul web che sulla carta, leggere linee lunghe è più faticoso.

Ma torneremo ampiamente su questi aspetti in altri punti della guida.

Layout elastico

Il layout elastico usa il dimensionamento in **em** non solo per il testo ma anche per la larghezza degli elementi principali della pagina, cosicché solo ridimensionando il carattere del browser è possibile agire sulla larghezza della pagina, ottenendo una sorta di effetto "zoom". Credo che l'idea di un layout elastico sia abbastanza recente, a me è capitato per la prima volta di vederlo presentato [su questo sito](#).

Vediamo brevemente come ottenere un layout elastico. Per prima cosa, si dimensiona il **font-size** del body con una misura relativa di modo che sia possibile ridimensionare il font tramite le opzioni del browser. Una misura percentuale accettabile universalmente è stata suggerita, in un articolo ormai diventato "storico", da Owen Briggs: [Sane css typography](#) ed è **76%**. Questa percentuale empirica dice al browser: metti il font-size al 76% della tua misura di default (*con un font size del 76% e un carattere medio impostato via browser, il risultato è all'incirca pari a 12 pixels*).

A questo punto si dimensiona il container principale, e le sue eventuali colonne, solo in em, consentendo così, reimpostando le dimensioni del carattere del browser, di avere un effetto zoom. Vediamo il css base di un layout elastico e centrato:

```
body{
  font-size: 76%; /*dimensionamento percentuale del font */
  text-align: center; /*centra in IE 5.x */
}

div#container{
  width: 60em; /*dimensionamento in em del container principale */
  margin: 0px auto; /*centra negli altri browsers*/
  text-align: left; /*ripristina l' allineamento*/
}
```

Come realizzare un sito table-less

Si potrebbe pensare, dato che il layout è la struttura portante di un sito internet, che sia la prima cosa di cui ci si debba preoccupare durante la fase di progettazione. La prima fase dovrebbe invece essere l'organizzazione dei contenuti, ancora meglio se con carta e penna: basterà semplicemente individuare le principali sezioni della home page o di una pagina tipo. Fatto questo, si potrà procedere alla decisione del layout da utilizzare, pensando a come si potrebbero disporre queste sezioni all'interno di un layout. A parere mio, ogni webdesigner dovrebbe seguire fin da subito una strategia di sviluppo per un sito table-less.

Una volta pianificati i contenuti, si penserà al layout da utilizzare, considerando che layout a due e tre colonne implicano differenti codifiche html a seconda che siano realizzate con i float o i posizionamenti assoluti. Fatto ciò si può procedere a scrivere l'html minimale senza contenuti testuali, ossia quello che definisce la struttura portante del sito. Questo file verrà salvato come **modello**, di modo da poter essere utilizzato per tutte le pagine del sito.

Successivamente si può procedere a scrivere la parte del foglio di stile relativa al layout. Infine resta solo da riempire le pagine con i contenuti e aggiungere le regole del foglio di stile necessarie. In questa parte la strategia da utilizzare è a discrezione del webdesigner: si potrà adottare una codifica parallela di html e css, oppure una codifica sequenziale (prima l'html, poi il css).

La fase di test di un layout

Uno degli svantaggi di un layout con i css è dovuto, purtroppo, alle differenze di interpretazione dei fogli di stile. Mentre le tabelle garantiscono una certa solidità, i css sono purtroppo un terreno instabile a causa dei browsers. Attualmente, credo ancora per almeno un paio di anni, **Internet Explorer 5.x e 6** sono i browsers più diffusi. Il loro supporto per i css, e in particolare per i css di livello 2, è piuttosto carente ed è veramente limitante per un webdesigner. Sebbene sia senz'altro molto più diffusa la versione 6, un sito web andrebbe testato anche su Internet Explorer 5.0 e 5.5, in particolare per l'errata interpretazione del Box Model di queste due versioni. Nel mio articolo "Capire il Box Model" (Guida CSS) viene spiegato come porre risolvere questo problema relativo a IE 5.x.

Se avete installata una versione di windows che sia 2000 o XP, potete installare diverse versioni di Internet Explorer sulla stessa macchina, il metodo viene spiegato nell'articolo "Tutti gli Explorer insieme su Windows" di Cesare Lamanna.

Attualmente potremmo non preoccuparci più del supporto dei css veramente scarso di **Netscape 4.7** e possiamo nascondere il foglio di stile a questo browser attraverso la regola **@import**.

C'è una notevole sensibilizzazione agli standard del web, e un layout va testato anche con i due principali browsers standard compliant: [Opera](#) e [Mozilla](#) giunti rispettivamente nel momento in cui scrivo, alla versione 7.23 e 1.5. Di questi due browsers, c'è da considerare che chi li ha installati sul proprio sistema lo fa per il piacere della navigazione, dell'attinenza agli standard e delle notevoli funzionalità messe a disposizione dell'utente, e che quindi generalmente tiene installata una versione recente di questi due browsers.

È importante considerare che, anche se maggior parte degli utilizzatori del web ha un personal computer con sistema Windows, ci sono altri sistemi operativi e di conseguenza altri browsers. tra questi, i più importanti sono Internet Explorer per Mac, Safari per Mac e Konqueror per Linux.

La **fase di test multi-browsers** dovrebbe proseguire di pari passo con la fase di codifica di html e css. Sovente infatti ci si trova, a sito finito, ad accorgersi che questo non rende bene su alcuni browsers, e ciò implica una quasi revisione totale e spesso una ricodifica notevole del foglio di stile, con conseguente notevole perdita di tempo per il webdesigner.

Dopo questa lunga, e spero utile introduzione, inizia la parte centrale della guida: la creazione di layout table-less grazie ai fogli di stile.

Layout monolitico

Questa è la forma più elementare di layout. Può andare bene per siti di piccole dimensioni (generalmente con un ordine di pagine sotto la decina), per siti con menu dropdown o per siti con macrosezioni (sottositi). Come anticipato, per la sua semplicità non necessita di posizionamenti o float nella versione elementare.

Il layout monolitico si compone di quattro sezioni fondamentali:

1. **header**
2. **navigazione**
3. **contenuti**
4. **footer**

Generalmente la navigazione è disposta orizzontalmente sotto l'header, ma può anche essere disposta a fianco dei contenuti, ma senza estendersi verticalmente come o più di quest' ultima (si otterrebbe in questo caso un layout a due colonne).

Una **nota importante**: in un layout monolitico si dovrebbe sempre specificare e dichiarare esplicitamente la larghezza dell'area principale. Non dichiarando la larghezza esplicitamente, il layout si adatta alla larghezza della finestra del browser. Questo potrebbe costringere un utente con monitor ad alte o altissime risoluzioni a stringere la finestra del browser per aumentare la leggibilità del sito. Come detto in precedenza, infatti, linee di testo troppo lunghe causano difficoltà di lettura.

Vedremo comunque le tre varianti di questo layout (fisso, fluido ed elastico) più una quarta decisamente particolare. Iniziamo a costruire il nostro layout passo passo, partendo dal codice HTML. Nell'immagine un esempio tipo di un layout monolitico:



Il codice HTML

Ecco il codice HTML di base, per ora senza contenuti:

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="content"></div>
```

```
<div id="footer"></div>
</div>
</body>
```

Come si può notare, la struttura della nostra pagina rispecchia le sezioni logiche di cui abbiamo parlato in apertura. Per contenere le sezioni principali, ho dichiarato un div con id **container**. È fondamentale: il suo scopo è infatti modificare l'estensione orizzontale in maniera facile attraverso poche e semplici regole CSS, come spiegato nella parte introduttiva della guida.

Aggiungiamo ora un po' di contenuto: un semplice header testuale, alcuni elementi per la navigazione tramite una lista, un h2 per il titolo della pagina, due paragrafi nel corpo centrale e due righe nel footer: ecco la [versione senza stile](#) che andremo a trasformare con i CSS.

Layout monolitico

Iniziamo ad elaborare le regole CSS necessarie per realizzare un layout a colonna singola fluido. Per prima cosa, agiremo sui tag **html** e **body**, eliminando margini e padding. Questo perchè tutti i browsers, come spiegato nel mio articolo "Capire il Box Model", assegnano di default margini e/o padding ad alcuni elementi, inclusi **html** e **body**.

Poi agiremo sul **font-size**: personalmente preferisco un carattere proporzionale, e il **76%** è la misura ottimale. Poi il **font-family** della pagina: basterà dichiararlo nel **body**. In questo caso ho indicato un carattere *arial* (che è uno tra i più diffusi) e, cosa importantissima, la famiglia generica a cui appartiene, ovvero *sans serif*, nel caso in cui questo font non fosse disponibile sul sistema dell'utente.

Aggiungeremo un **background-color** all'header, alla navigazione e al footer. Infine, alcune proprietà di formattazione: padding, margin e text-align delle varie sezioni. Ecco il css minimale:

```
html,body{margin:0; padding: 0}
body{font-family: arial,sans-serif; font-size: 76%}
div#container{ /*nessuna regola necessaria*/ }
div#header{background: url(sfondo.gif)}
h1{margin: 0;padding:0}
h2{color:#4078B8; font-size:1.5em; text-align:center}
div#navigation{background-color:#000; color: #fff}
div#content{padding:0 1.5em}
div#content p{line-height:1.3em}
div#footer{padding:0.5em; background-color:#B0D0E8; color:#000; text-align:center}
```

A queste regole vanno ad aggiungersi quelle per la navigazione e la pagina è pronta. Spiegare come presentare le liste di navigazione esula dallo scopo di questa guida, ma ho pensato comunque di aggiungere un po' di stile alla pagina, giusto per renderla un po' più accattivante: [questo è il risultato](#). [Scarica l'esempio](#). In questa, come in tutte le pagine di esempio che presenteremo, è possibile visualizzare il codice CSS utilizzato.

Layout monolitico fisso

trasformare un layout fluido in uno a larghezza fissa è molto semplice a questo punto:

basterà aggiungere qualche regola CSS al **body** e al **div** con id **container**. Per delimitare spazialmente il contenitore è bene usare in questo caso dei bordi. Questo è il [layout fisso](#): come si può notare la lettura risulta molto più agevole e restringendo la finestra si genera la scrollbar orizzontale, impedendo ad esempio che la navigazione si disponga su più righe, garantendo quindi una maggiore solidità [Scarica l'esempio](#). Una **nota importante**: è importante controllare che la stampa di una pagina che lo utilizza venga bene. Sovente, infatti, i contenuti risultano tagliati. Il modo per evitare questo problema è definire una regola per il CSS di stampa. Questa la regola da aggiungere nel foglio di stile per la stampa:

```
<style type="text/css" media="print">
div#container{
  width: auto;
}
</style>
```

In questo modo la larghezza del container in fase di stampa verrà calcolata dal browser per occupare in larghezza solo la pagina di stampa.

Layout monolitico elastico

Questo tipo di layout è un po' particolare. Iniziamo con il dire che tutte le sezioni di testo di un sito con layout elastico devono essere dimensionate *relativamente*, così come le eventuali larghezze degli elementi. Per un layout monolitico, il contenitore principale viene dimensionato in **em** così da consentire, modificando la larghezza del carattere tramite le opzioni del browser, di ottenere una sorta di effetto zoom. Ecco il codice CSS:

```
/*stili per il layout elastico*/
html,body{margin: 0;padding:0}
body{font-family: arial,sans-serif;font-size: 76%;text-align: center}
div#container{width: 60em;margin: 0 auto;text-align: left;
border-left: 2px solid #36c;border-right: 2px solid #36c}

/*stili generici, su header e footer*/
div#header{background-color:#36C;color: #ff0}
h1,h2{margin: 0;padding:0}
h1{padding-left:0.5em;font: bold 2.3em/80px arial,serif}
h2{color: #999;font-size: 1.5em}
div#footer{text-align:center;padding: 0.5em;
background-color: #69c; color: #000}
div#footer a{color: #fff;font-weight: bold;text-decoration: underline}
```

```
/*stili specifici per il layout*/
div#navigation{background-color:#9cf}
div#content{padding: 1em}
```

E l'esempio realizzato di [layout a colonna singola elastico](#). [Scarica l'esempio](#). Anche qui, il mio consiglio è di fare le prove di stampa.

Il problema del footer

Questo è l'ultimo esempio che vedremo per i layout a colonna singola. Mi è capitato spesso di leggere di persone che volevano realizzare questo effetto: avere il footer sempre in fondo alla pagina. La tecnica che presenterò si può adattare anche ad alcuni layout a due e tre colonne. Vediamo come operare nel foglio di stile, usando i posizionamenti assoluti.

Per prima cosa, aggiungiamo una regola per estendere il **body** su tutta la finestra del browser. Questa regola serve per i browser moderni quali Opera e Mozilla, che attribuiscono come altezza effettiva degli elementi **body** e **html** solo l'altezza dei contenuti. A questo punto specifichiamo per il container un posizionamento relativo (**position: relative**): questa semplice regola ci consentirà di posizionare il footer rispetto al bottom del container, tramite **position: absolute**. Basterà specificare, per il footer, la proprietà **bottom:0** e assegnare una larghezza (**width**) pari al 100%. La larghezza effettiva degli elementi posizionati assolutamente è quella necessaria al contenuto, a differenza degli elementi block level non posizionati.

L'ideale sarebbe, ora, assegnare un'**altezza minima al container**. Purtroppo, Internet Explorer non supporta la proprietà **min-height**. Per fortuna, però, su questo browser gli

elementi, se la lunghezza della pagina supera quella della finestra del browser, vengono comunque resi all'interno del container stesso. Opera e Mozilla in caso di lunghezza maggiore assumono invece il comportamento corretto: il footer resta sul fondo della finestra del browser, e si dispone sopra i contenuti.

Opera e Mozilla supportano **min-height** e, a differenza di Internet Explorer, anche il **child selector** che useremo per dichiarare una regola ad hoc non vista da Internet Explorer. Infine, se la pagina è più lunga della finestra del browser, succede che il footer si dispone sopra le ultime righe del corpo centrale. Questo può essere evitato aggiungendo un padding-bottom al main pari o superiore all'altezza del footer. Ecco quindi la parte del css necessario per avere un footer sempre al fondo:

```
html,body{margin:0; padding: 0;height: 100%}  
div#container{position: relative; height:100%}  
body>div#container{height: auto; min-height: 100%}  
div#content{padding: 0 1.5em 5em}  
div#footer{position:absolute;bottom: 0;width: 100%}
```

Ed ecco [questa variante applicata ad un layout fisso a colonna singola](#). Ecco [come rende](#) nel caso in cui i contenuti siano superiori alla pagina. [Scarica l'esempio](#) Ci tengo a precisare che Opera 7.23 presenta una piccola differenza di comportamento rispetto a Mozilla 1.5 e Internet Explorer 5.5 e 6. Il footer, in caso di contenuti che non generano scrolling verticale, allungando la finestra del browser dopo averla accorciata, non segue il suo fondo, se non dopo un aggiorna. Il footer resta comunque dopo i contenuti, sia che la pagina web sia più corta della finestra del browser che nel caso contrario.

Layout a due e tre colonne: introduzione

Ci sono due modi per realizzare layout a due o più colonne:

- L'uso del **posizionamento**, in particolare assoluto e relativo.
- L'uso della proprietà **float** e dei margini.

In quanto alla scelta della tecnica da usare, c'è da dire che:

- la tendenza attualmente più diffusa è quella che poggia sulla proprietà float (consente una maggiore flessibilità e facilità di codifica);
- l'uso del posizionamento assoluto permette un maggior controllo sulla disposizione e separazione dei contenuti.

Di ogni layout vedremo le due versioni e molte varianti, e affronteremo pro e contro delle due tecniche nello specifico della tipologia.

La disposizione delle sezioni in ordine di codice html

È importante tenere presente, come vedremo in seguito, che l'uso dei float e dei posizionamenti assoluti per layout a due o tre colonne condiziona *a monte* la codifica del codice HTML. Un buon webdesigner dovrebbe sempre considerare l'**accessibilità** di un sito e codificarlo di modo che risulti favorito nell'indicizzazione dai **motori di ricerca**.

Sulla questione dell'accessibilità, ho chiesto un parere a Michele Diodati, mentre ho preso spunto dall'articolo "Progettare un sito gradito ai motori di ricerca" di Francesco de

Francesco per quanto riguarda i motori di ricerca. La mia curiosità: *Quanto influisce l'ordine del codice HTML in cui è prevista la navigazione **dopo** i contenuti?*

Michele Diodati, del sito Diodati.org e autore della guida "Costruire siti ad elevata accessibilità" mi ha risposto:

...Se sono favorevole oppure no a soluzioni di impaginazione in cui tramite CSS si faccia apparire il menu di navigazione in un posto qualsiasi della pagina, mentre nel codice HTML si trova dopo il testo principale?

Se questa è la tua domanda, la mia idea è che tale soluzione vada adottata con estrema attenzione, perché fa uso quasi inevitabilmente di posizionamenti assoluti, che non si prestano bene ai ridimensionamenti. Un'impaginazione accessibile deve tenere sempre conto, infatti, della possibilità che l'utente ingrandisca il testo e, quando ciò avviene, nei limiti del possibile non devono avvenire sovrapposizioni di contenuti.

Quanto al fatto di mettere i contenuti prima del menu, questa è una richiesta pressante dei non vedenti, che sono veramente infastiditi da pagine fatte da chilometrici elenchi di link, da percorrere prima di arrivare ai contenuti veri e propri. Nè sono aiutati dai link di salto, che odiano visceralmente. Per loro i link di salto sono appunto salti nel buio: non sanno cosa c'è tra il link e il contenuto.

In effetti non siamo ancora riusciti a trovare un sistema veramente valido che metta d'accordo le abitudini dei vedenti (menu di navigazione a nord, contenuti a sud), con le esigenze dei non vedenti (i contenuti prima di tutto).

Francesco de Francesco, nel suo recente articolo apparso nella sezione PRO di HTML.it afferma:

In considerazione del fatto che tutti i motori danno significato alla posizione del testo nell'ambito del file della pagina e alla posizione delle parole nell'ambito del testo, è evidente che differenti impaginazioni, anche per testi uguali, producono risultati diversi in termini di posizione sui risultati.

Vari motori evidenziano come descrizione del sito trovato le prime parti incontrate nel body. Avere il menù nella parte alta, se necessario per l'usabilità del sito, è spesso fattore negativo per il posizionamento.

Un buon schema d'impaginazione può consentire decisi miglioramenti del posizionamento. Le tecniche di sviluppo, inoltre, permettono spesso di superare i vincoli di posizione.

Un'altro problema da affrontare preliminarmente è quello concernente la lunghezza e la larghezza delle colonne. È l'argomento della prossima lezione.

Lunghezza e larghezza delle colonne

La scelta della strategia da utilizzare, oltre a condizionare tematiche quali accessibilità e indicizzazione sui motori di ricerca di cui abbiamo parlato nella lezione precedente, è dettata anche da questioni di lunghezza relativa delle colonne.

C'è da dire, infatti, che un buon layout a due o tre colonne realizzato con i float consente, a differenza della versione con i posizionamenti assoluti, di non imporre vincoli.

In sostanza, la versione con posizionamenti assoluti introduce un limite che può risultare in

certi casi significativo: se vogliamo un layout con footer, infatti, dobbiamo tener presente che **la colonna dei contenuti deve essere più lunga delle colonne secondarie.**

La larghezza delle colonne secondarie

Premetto che con il termine colonne secondarie intendo genericamente la colonna di navigazione in un layout a due colonne così come le due colonne laterali in un layout a tre colonne.

Una questione che si pone in fase di progettazione di un sito è la scelta del dimensionamento della larghezza delle colonne secondarie. Ci sono tre principali approcci:

- colonne secondarie a **larghezza percentuale**;
- colonne secondarie a **larghezza fissa**;
- colonne secondarie **elastiche**.

L'approccio basato su colonne secondarie a larghezza percentuale è decisamente **poco utilizzato** per svariati motivi, primo fra tutti quello che impedisce un buon controllo della pagina nel suo complesso. Infatti, alcuni browser hanno dei problemi di arrotondamento delle percentuali, e può capitare che 100% significhi *un po' più di 100%*, generando una fastidiosa scrollbar orizzontale.

L'approccio a larghezza fissa (in pixel) è decisamente **il più utilizzato**.

Ci sono comunque delle questioni piuttosto rilevanti da considerare: se si usa un carattere fisso questa strategia è ottima. Se si sceglie un carattere proporzionale bisognerà accertarsi che, all'aumento della dimensione del font tramite le impostazioni del browser, i contenuti delle colonne secondarie degradino bene. Per esempio: possiamo accettare che il testo di qualche link si disponga su due righe?

Dichiarare colonne secondarie elastiche (e quindi **dimensionarle in em**) ha senso **solo quando vogliamo rendere i contenuti totalmente ridimensionabili**. Ha lo svantaggio notevole di non consentire una buona precisione. Con questa strategia non è possibile realizzare layout aiutandosi con immagini di background.

Layout a due colonne: elementi di base

Questa tipologia di layout è adeguata per siti di dimensioni medie. Le sezioni fondamentali di un layout a due colonne sono:

- **header**
- **navigazione**
- **contenuti**
- **footer**

Navigazione a destra o a sinistra?

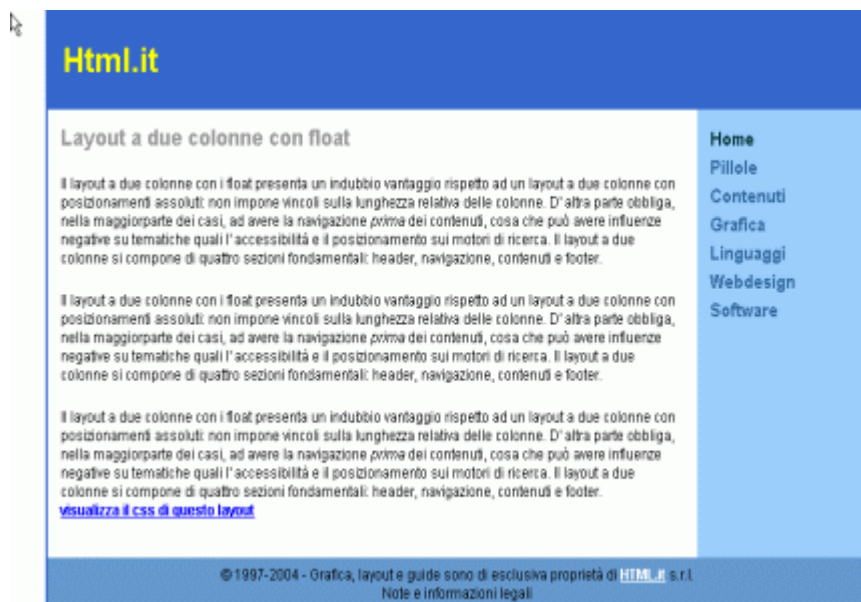
Nell'introduzione al layout a due e tre colonne abbiamo trattato le principali problematiche di progettazione per questi due tipi di impaginazione. Ma c'è una questione che riguarda esclusivamente il layout a due colonne: **meglio avere la navigazione a destra o a sinistra?**

Chiariamo subito che non si tratta di privilegiare una tecnica rispetto ad un'altra. Possiamo ottenere il risultato desiderato sia usando i float, sia il posizionamento assoluto. La scelta è

solo una questione di gusto del progettista e/o di richiesta del committente.

Sebbene la navigazione a sinistra sia senza dubbio la più diffusa, c'è una questione da considerare. In occidente leggiamo dall'alto verso il basso e da sinistra verso destra. Avere la navigazione sulla destra favorisce così i contenuti, che hanno un privilegio visivo.

Gli esempi di layout a due colonne che proporremo, saranno tutti realizzati con la colonna secondaria a destra. Si tenga presente, comunque, che le modifiche al foglio di stile per avere la navigazione a sinistra sono davvero minime. Vediamo uno schema tipo di un layout a due colonne:



Layout a due colonne con il float

Vedremo in questa lezione come creare un layout a due colonne con i float. Per realizzarlo bisogna tener presente che nel codice HTML **la navigazione deve precedere i contenuti**. Ecco l'HTML essenziale, che risulta identico a quello del layout a colonna singola:

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="content"></div>
    <div id="footer"></div>
  </div>
</body>
```

È importante considerare che, con un simile ordine di codice, se si vuole ottenere una buona accessibilità, bisognerà provvedere **accesskey** e/o link di "salta navigazione" per agevolare gli utenti ipovedenti che utilizzano screen reader per navigare.

Aggiungiamo anche in questo caso un po' di contenuti, ottenendo [questa pagina](#), per ora senza stile.

Procediamo aggiungendo gli stili necessari per ottenere il layout. Due sono i passi preliminari da compiere:

- rendere la sezione di navigazione **float**;
- separare verticalmente i contenuti da essa, attraverso l'uso dei margini dichiarati sul **div** principale.

Ecco le regole del foglio di stile che andranno aggiunte alla presentazione dell'header, dei contenuti e del footer:

```
div#navigation{
  float: right;
  width: 13em
} div#content{
  margin-right: 13em
}
```

Semplice vero? C'è da aggiungere che volendo una colonna secondaria a larghezza fissa **basterà dichiarare le dimensioni in pixel anziché in em**. Questo è il [risultato](#), che utilizza un layout fisso, con un po' di regole aggiuntive per presentare l'esempio. [Scarica il file zip con l'esempio](#)

Una regola indispensabile per un layout stabile

Arrivati a questo punto bisogna considerare **la lunghezza relativa** delle colonne. Infatti, se la colonna di navigazione risulta più lunga si ottiene che il footer si dispone appena sotto i contenuti e non sotto la colonna secondaria, causando un effetto non proprio piacevole: **la navigazione si sovrappone al footer**.

Per sistemare le cose basta una semplice regola, si fa in modo che il footer non abbia elementi float alla sua destra attraverso la proprietà **clear**:

```
div#footer{
  clear: right
}
```

Con l'aggiunta di questa semplice riga di codice abbiamo sistemato le cose.

Layout a due colonne con il float

Il layout che abbiamo visto non dà vincoli sulla lunghezza relativa delle due colonne. Se sappiamo a priori che in tutte le pagine in cui useremo un layout a due colonne con i float *la colonna dei contenuti sarà più lunga della colonna di navigazione*, possiamo facilmente separare le due colonne. Le due soluzioni che possiamo adottare sono:

- separare le due colonne **con un bordo fine**;
- separare le due colonne **con l'uso del colore**.

Vedremo qui di seguito come poter avere le due colonne di colore distinto, e soprattutto avere l'effetto che la colonna di navigazione si estenda fino al footer. Allo stesso modo, è possibile assegnare solo un **border-right** al **div** dei contenuti.

Layout a due colonne cromatiche

Vogliamo assegnare un colore diverso alle due colonne, ed avere l'idea che la colonna secondaria si estende cromaticamente fino al footer. Se ci avete già provato, avrete visto

che giocare con l'altezza non ha effetto. Il trucco è semplice: invece di assegnare un colore di sfondo alla navigazione, **assegniamo un colore di sfondo al container e al div principale**. In questo caso la navigazione erediterà il colore del container e darà l'impressione di estendersi fino al footer. Ecco allora il [layout con due colonne di colore diverso](#).

Un limite che si può superare

I problemi però non mancano. [Ecco come rende](#) il layout dell'esempio precedente nel caso in cui la colonna secondaria sia la più lunga.

C'è un'alternativa, che si rivela efficace in entrambi i casi: usare un'immagine di background per il contenitore principale. L'idea è semplice: usare un piccola immagine alta pochi pixel da ripetere verticalmente su tutto il container. Questa immagine deve essere larga quanto la navigazione, e implicitamente introduce una condizione: *la colonna di navigazione dovrà per forza essere dimensionata in pixel*.

Ecco come ottenere un background che si ripete verticalmente lungo tutto il container:

```
div#container{
  background: url(rightcol.jpg) repeat-y top right
}
```

Applicare questa soluzione è semplicissimo: basterà realizzare una piccola immagine del colore che vogliamo con un qualsiasi programma di grafica. Ed ecco il layout a [due colonne con background](#), anche nel caso in cui [la colonna di navigazione sia la più lunga](#).

Scarica lo [zip](#) di questa lezione

Layout a due colonne con posizionamento assoluto

Il layout a due colonne con posizionamenti assoluti presenta dei significati vantaggi esterni rispetto all'analogo con i float. Questi sono l'accessibilità e delle migliori chance di indicizzazione e posizionamento sui motori di ricerca. Ha però uno svantaggio: non consente di avere la colonna di navigazione più lunga della colonna dei contenuti se vogliamo avere un footer a larghezza piena. Se siamo disposti ad accettare questa condizione, a parer mio il layout con posizionamenti assoluti è da preferire.

Anche in questo caso le sezioni fondamentali sono quattro: header, navigazione, contenuti e footer. Cambia il codice HTML, in quanto **la navigazione deve essere preceduta dai contenuti**. Ecco quindi l'HTML elementare di questo layout:

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="content"></div>
    <div id="navigation"></div>
    <div id="footer"></div>
  </div>
</body>
```

Ora vediamo quali sono le regole CSS minimali per realizzare un layout a due colonne con i

posizionamenti assoluti:

```
div#container{  
  position: relative;  
}
```

```
div#content{  
  margin-right: 200px;  
}
```

```
div#navigation{  
  position: absolute;  
  top: 80px; /*l' altezza dell' header*/  
  right: 0;  
  width: 180px; /*un po' di meno per evitare l' uso del padding*/  
}
```

In sostanza il meccanismo è questo: il container viene dichiarato `position: relative` per consentire il posizionamento al suo interno. Il **div** dei contenuti lascia spazio sulla destra per la navigazione e allo stesso tempo, implicitamente, permette al footer di disporsi a fondo pagina riprendendosi il flusso degli elementi; infine la navigazione è posizionata assolutamente.

Anche qui vediamo [il risultato](#).

Allo stesso modo con cui abbiamo realizzato il layout a due colonne cromatiche con i `float`, possiamo ottenere il layout a due colonne cromatiche con il [colore](#) e con l' [immagine di background](#).

Scarica il file [zip](#) di questa lezione

Layout a due colonne fluidi

Gli esempi di layout a due colonne che abbiamo visto finora sono a larghezza fissa. Ecco invece le versioni fluide, entrambe con l'immagine di background:

- [Layout a due colonne con i float](#)
- [Layout a due colonne con i posizionamenti assoluti](#)

Le versioni fluide presentano un indubbio vantaggio: sono già pronte per la stampa. Provate a fare anche solo un anteprima di stampa di uno dei due layout fluidi o di uno dei qualsiasi layout fissi visti in precedenza. Hanno uno svantaggio rispetto alle versioni fisse: stringendo al limite la finestra del browser il testo delle due colonne si sovrappone, fino a far scivolare la navigazione parzialmente sotto i contenuti nel caso della versione con la proprietà **float**.

Purtroppo la proprietà **min-width**

non è supportata da Internet Explorer, nè nella versione 5.x nè nella 6. Ha effetto invece sui browser Gecko e Opera. La dichiarazione è semplice:

```
div#container{  
  min-width: 600px;  
}
```

Per Internet Explorer una soluzione ci sarebbe... ma anticipo subito che è una soluzione proprietaria e **non standard**. Quindi una soluzione simile non passa al vaglio del [validatore](#).

Ci tengo a precisare che non posso garantire di quanto letto in rete su un forum e che personalmente non adotterei mai una soluzione simile, ma dato che Internet Explorer detiene una percentuale maggioritaria dei browsers attualmente usati... questo è la regola css **proprietaria per Internet Explorer**:

```
div#container{  
  width:expression(document.body.clientWidth < 600 ? "600px" : "auto" );  
}
```

La regola sopra è un'espressione condizionale ternaria che dice ad Internet Explorer: se la larghezza della finestra del browser è maggiore di 600 pixels, dai al container tutta la larghezza possibile, altrimenti assegna al container una larghezza di 600 pixels.

Scarica il file [zip](#) di questa lezione

Layout a tre colonne: introduzione

Il layout a tre colonne è uno dei più diffusi e permette di gestire siti di dimensioni medio-alte. Si compone delle seguenti sezioni fondamentali:

- **header**
- **navigazione**
- **contenuti**
- **sezione extra**
- **footer**

In quanto alla non meglio specificabile *sezione extra*, questa può contenere, come abbiamo visto nell'introduzione, svariate funzionalità del sito: news, sondaggi, banner, navigazione supplementare e quant'altro. Dovrebbe comunque essere facilmente consultabile, non troppo scarna, nè troppo densa.

Questo tipo di layout, proprio per la presenza di una sezione logica (e fisica) aggiuntiva, è indicatissimo per impaginare siti dinamici attraverso linguaggi lato server quali PHP o ASP e comunque siti con molti contenuti e/o aggiornamenti frequenti. Vediamo un esempio tipo di un layout a tre colonne:



Lunghezza relativa e larghezza delle colonne

Anche in questo caso, come per i layout a due colonne, apriamo una piccola parentesi sulla lunghezza relativa delle colonne. Resta l'idea che la sezione dei contenuti dovrebbe essere la più corposa, e quindi **la colonna centrale la più lunga**. Come per il layout a due colonne, la versione a tre colonne con i float rispetto all'analogo con posizionamenti ha il pregio di non imporre vincoli sulla lunghezza relativa delle colonne in presenza del footer. D'altra parte, vincola l'ordine di codifica dell'HTML.

In quanto alla larghezza, la tendenza è avere **le due colonne laterali a larghezza fissa e la colonna centrale fluida**. Le due colonne generalmente si aggirano attorno ai 150-200 pixels di larghezza ciascuna, e di solito hanno la stessa larghezza per dare un'idea di simmetria.

Per ottenere colonne laterali elastiche, basterà dimensionare il **font-size** del **body** in percentuale e dimensionare le colonne laterali e i margini della colonna centrale in **em**: questa tecnica non può andar bene, però, con layout che usano immagini di background.

Per la scelta della larghezza totale, c'è da dire che un layout a tre colonne si presenta generalmente fluido, a larghezza totale, occupa cioè in ampiezza tutta la larghezza della finestra del browser.

Dettagli da non trascurare

I layout che stiamo per trattare sono di una complessità maggiore rispetto a quelli visti in precedenza. È importante allora fissare alcuni dettagli sull'uso dei font, dei margini e dei padding degli elementi di una pagina.

È buona norma dare all'utente la possibilità di ridimensionare il testo di una pagina web. Mentre Opera, Mozilla e Safari consentono di personalizzare la dimensione dei caratteri per qualsiasi pagina a prescindere dall'unità di misura scelta per il dimensionamento dei caratteri, Internet Explorer non lo consente per caratteri dimensionati in pixels. Usare

quindi una misura relativa è una scelta più attenta verso l'utente. Una scelta che però comporta alcune questioni in fase di progettazione.

Il dimensionamento in em accordato con il dimensionamento relativo dei font consente, come abbiamo visto, un ingrandimento non solo del testo, ma anche della sezione che lo contiene. Bisogna tener presente in questo caso che un em equivale all'incirca alla larghezza di una lettera con un font non proporzionale (detto comunemente *monospaced*).

Però, scegliere di dimensionare le colonne in pixel e usare al contempo un carattere dimensionato relativamente, comporta test aggiuntivi, quali ad esempio la visualizzazione della pagina modificando la dimensione del font attraverso le opzioni del browser, per verificare che questa renda comunque bene.

Per quanto riguarda margini e padding orizzontali di un elemento, questi dovrebbero essere concordi con l'unità di misura della sua dimensione orizzontale: se per esempio abbiamo dimensionato le colonne laterali in pixel, margini e padding dovrebbero essere espressi in pixel.

Infine, due parole sul padding: come visto nell'introduzione e nel mio articolo "Capire il box model", Internet Explorer 5.x (e Internet Explorer 6 senza un corretto doctype) interpretano erroneamente il padding come parte della dimensione effettiva di un elemento. Finora negli esempi ho sempre cercato di evitare di dimensionare un elemento e specificarne il padding, attribuendo invece dei margini agli elementi in esso contenuti, ma arrivati a questo punto è bene adottare una soluzione che sia più agevole: replicare margini orizzontali per tutti gli elementi contenuti in una colonna decisamente non è una soluzione comoda o snella. È importante, quindi, avvalersi di una tecnica che aggiri queste differenze di interpretazione, e in tal caso il modo più semplice è senz'altro il **box model hack semplificato**.

Layout a tre colonne con float

In questa lezione vedremo come realizzare un layout a tre colonne con i **float**. Le due colonne, come abbiamo detto, devono precedere il contenuto nel codice HTML, vediamo il codice minimale:

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="navigation"></div>
    <div id="extra"></div>
    <div id="content"></div>
    <div id="footer"></div>
  </div>
</body>
```

Anche in questo caso, vediamo il CSS di base. Per quasi tutti gli esempi che seguiranno, le due colonne saranno larghe 170 pixel totali (incluso il padding). Ci avverremo, quindi, del box model hack semplificato:

```
div#navigation{
float:left;
/* Inizia il Simplified Box Model Hack */
width:160px; /* Dimensioni per Opera 5 */
```

```
width:170px; /* Dimensioni per Explorer 5.x */
width:160px; /* Dimensioni per browser standard compliant */
padding:1em 0 0 10px}
```

```
div#extra{
float:right;
width: 160px;
width:170px;
width:160px;
padding:1em 5px}
```

```
div#content{
margin:0 170px;
padding:1em 10px}
```

```
div#footer{clear:both}
```

La colonna dei contenuti principali verrà distanziata da quelle laterali grazie all'uso dei margini laterali di misura uguale alla larghezza totale delle colonne laterali. In pratica, se la colonna laterale misura 170px, imposterò un margine sinistro o destro su quella centrale uguale a 170px.

Infine, il footer avrà un **clear: both** di modo da impedire che le colonne secondarie vadano a coprirlo (nel caso in cui queste ultime siano più lunghe della colonna centrale).

Aggiungiamo un po' di contenuto e stile alle varie sezioni e [questo è il risultato](#). Ecco invece come rende lo stesso layout applicato però a una pagina HTML con la [colonna di navigazione più lunga](#) oppure in cui la [sezione extra è la più lunga](#).

Scarica il file [zip](#) di questa lezione

Layout a tre colonne con float: separare le colonne

Per delimitare spazialmente o cromaticamente le colonne si può ricorrere ai bordi e al background-color. Per separarle grazie ai bordi, basterà **aggiungere i due bordi laterali alla colonna centrale**. Ecco il codice da aggiungere:

```
div#content{
  /* ... */
border-width: 0 1px;
border-style:solid;
border-color: #ccc;
}
```

Ecco quindi [il layout a tre colonne con i bordi](#).

Mentre per il colore le regole aggiuntive saranno:

```
div#container{background-color: #ccc}
div#content{ /*...*/ background-color: #fff}
```

Anche qui vediamo [il risultato](#).

Lo svantaggio principale di queste due soluzioni è che *impongono il limite* sulla lunghezza relativa delle colonne: infatti la colonna centrale dovrà essere la più lunga. La soluzione è usare le immagini di background. Sarà l'argomento della prossima lezione.

Scarica lo [zip](#) di questa lezione

3cf.zip

Layout a tre colonne con float e background

Per dare l'idea che le due colonne laterali si estendono fino al footer indipendentemente dalla loro lunghezza rispetto alla colonna centrale, si possono usare le immagini di background: a colore pieno oppure bianche con bordo fine o sfumato, tutto sta alla creatività. Usare immagini di background richiede però l'aggiunta di un contenitore supplementare nel codice HTML, almeno in certi casi. Ecco il codice HTML base per questo tipo di layout:

```
<body>
  <div id="container">
    <div id="container2">
      <div id="header"></div>
      <div id="navigation"></div>
      <div id="extra"></div>
      <div id="content"></div>
      <div id="footer"></div>
    </div>
  </div>
</body>
```

La questione è che è possibile assegnare una sola immagine di background ad un elemento: i due elementi a cui assegneremo l'immagine di background sono quindi *container* e *container2*. In realtà sotto certe condizioni l'uso di un container aggiuntivo è superfluo: una delle due immagini di background si potrebbe assegnare al **body** solo nel caso in cui la pagina si estende almeno per tutta la lunghezza della finestra e si usi un layout fluido a larghezza totale. La scelta dell'aggiunta di un contenitore supplementare, d'altra parte, consente maggiore possibilità di personalizzazione del layout attraverso l'uso del background: è infatti possibile usare layout semifluidi, fissi o elastici senza dover ritoccare il codice HTML. Ho preparato come esempio una versione fissa, che potrete riadattare a elastica, fluida o semifluida con poche modifiche del CSS. Rivediamo tutte le regole fondamentali del foglio di stile per questo layout:

*/*stili per il layout fisso con background*/*

```
html,body{margin: 0;padding:0}
body{font-family: arial,sans-serif;
font-size: 76%;
text-align:center
}
```

```
div#container{width:760px;
margin:0 auto;
```

```
border-width: 0 1px;
border-style:solid;
border-color: #ccc;
background: url(3rightcol.jpg) repeat-y top right
}
```

```
div#container2{background: url(3leftcol.jpg) repeat-y top left;
text-align: left
}
```

*/*stili specifici per il layout*/*

```
div#navigation{float: left;
width: 120px;
width:130px;
width:120px;
padding: 1em 0 0 10px
}
```

```
div#extra{float: right;
width: 190px;
width:200px;
width:190px;
padding: 1em 5px
}
```

```
div#content{margin: 0 200px 0 130px;
padding: 1em 10px
}
div#footer{clear:both;
text-align:center;
padding: 5px 0;
background-color: #69c;
color: #000
}
```

Vediamo [il risultato](#) anche nel caso in cui sia più lunga la [colonna di sinistra](#) o [quella di destra](#).

Scarica l'esempio in formato [zip](#) di questa lezione

Layout a tre colonne con i posizionamenti assoluti

Per il layout a tre colonne con i posizionamenti assoluti nel codice HTML la sezione dei contenuti deve precedere le due colonne laterali, e deve essere la più lunga.

Ritengo che siano una caratteristica e un limite che si compensano a vicenda: migliore accessibilità e chance di indicizzazione nei motori di ricerca per un limite accettabilissimo. Infatti, è raro trovare siti in cui i contenuti delle colonne laterali eccedono rispetto alla colonna centrale, e per tali situazioni si può sempre pensare ad uno snellimento o una

riorganizzazione (spostando i contenuti nella colonna centrale) dei contenuti delle colonne laterali.

Sull'ordine di codifica HTML delle due colonne laterali non ci sono vincoli: si può quindi scegliere se disporre prima la navigazione oppure la colonna extra. Dovendo effettuare una scelta apparentemente ininfluyente, si può optare per avere la seguente disposizione di codice:

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="content"></div>
    <div id="extra"></div>
    <div id="navigation"></div>
    <div id="footer"></div>
  </div>
</body>
```

Vediamo ora gli stili della versione base:

```
/*stili per il layout fluido*/
```

```
html,body{margin: 0;padding:0}
```

```
body{font-family: arial,sans-serif; font-size: 76%}
```

```
div#container{position: relative}
```

```
/*stili specifici per il layout*/
```

```
div#navigation{position: absolute;
top: 80px;
left: 0;
width: 160px;
width:170px;
width:160px;
padding: 1em 0 0 10px
}
```

```
div#extra{position: absolute;
top: 80px;
right: 0;
width: 160px;
width:170px;
width:160px;
padding: 1em 5px
}
```

```
div#content{margin: 0 170px;
padding: 1em 10px
}
```

```
div#footer{text-align:center;
padding: 5px 0;
```

```
background-color: #69c;
color: #000
}
```

Ed ecco il risultato del [layout a tre colonne con posizionamenti assoluti in versione base](#).

Anche qui con poche aggiunte al foglio di stile otteniamo un layout a tre colonne con posizionamenti assoluti con [colonne separate da bordi fini](#) oppure [in colore](#).

Scarica il file [zip](#) di esempio di questa lezione

Una variante del layout con posizionamenti assoluti

Abbiamo già visto come ottenere le due colonne laterali in tinta: si assegna un colore al container e il bianco al div principale. Il risultato è che le colonne laterali danno l'impressione di estendersi fino al footer, dato che è la colonna centrale a consentire il flusso di elementi.

Finora, inoltre, abbiamo visto che la soluzione per distanziare la colonna centrale dai lati della finestra è aggiungere i margini, in modo da lasciare spazio per il posizionamento delle colonne centrali. L'idea alternativa è abbastanza semplice: usare un bordo spesso quanto i margini. Ecco le regole necessarie alla colonna centrale (il resto rimane invariato):

```
div#content{
  border-style:solid;
  border-width:0 170px;
  border-color:#000 #BDDEFF #000 #C3FFFF;
  padding: 1em 10px
}
```

I bordi spessi hanno due funzioni: separare la colonna centrale per fare spazio a quelle laterali (sostituendo quindi i margini) e dare un colore alle due colonne secondarie. Ecco [il risultato](#). Semplice vero? Ci sono due questioni però riguardo a questo layout. La prima è che Internet Explorer 5.x e 6 danno problemi di allineamento del testo del footer *nel caso in cui i bordi spessi non siano larghi uguali*: per qualche strano motivo sembra che il testo del footer soffra di una "traslazione laterale opposta" pari alla differenza di larghezza dei due bordi.

È per questo che non ho proposto una variante con bordo spesso per il layout a due colonne, dato che la traslazione del testo del footer è proprio pari alla colonna secondaria. Sul layout a tre colonne, se i due bordi spessi sono larghi uguali non c'è problema: questi possono essere dimensionati in pixel, oppure in em per avere colonne laterali elastiche. La seconda questione è che mentre il background-color o le immagini di background non vengono stampate di default dai browsers, i bordi sì. Per ovviare a questo ci sono diverse possibili soluzioni, che comunque comportano lo specificare di un foglio di stile per la stampa.

Scarica l'esempio [zip](#) di questa lezione

Layout a tre colonne con immagine di background

Come per la versione analoga con i float, anche in questo caso è consigliato l'uso di un container aggiuntivo. Ecco quindi il codice HTML base:

```
<body>
  <div id="container">
    <div id="container2">
      <div id="header"></div>
      <div id="content"></div>
      <div id="extra"></div>
      <div id="navigation"></div>
      <div id="footer"></div>
    </div>
  </div>
</body>
```

Le regole CSS da aggiungere sono semplici:

```
div#container{position: relative;
background:url(3rightcol.jpg) repeat-y top right
}
```

```
div#container2{background: url(3leftcol.jpg) repeat-y top left}
```

Anche qui basta semplicemente scegliere il tipo di layout. Ho preparato la [versione con layout fisso](#).

Scarica il file [zip](#) di esempio di questa lezione

Rompere la simmetria

La simmetria è un aspetto ormai molto diffuso in un layout a tre colonne. Personalmente ritengo che la colonna extra dovrebbe essere più larga della colonna di navigazione, soprattutto nel caso in cui questa presenti molto testo. Una scelta simile accompagnata da buoni contenuti può essere senz'altro molto più gradevole e usabile.

Tutti gli esempi visti finora hanno le due colonne laterali pari a 170pixels ciascuna. Una teoria di tipografia e design degli anni 30 elaborata da Jan Tschichold sostiene che una tipografia asimmetrica può risultare decisamente più effettiva, impressiva e funzionale. I principi della tipografia cartacea sono per lo più sconosciuti ai webdesigner, ma al di là delle varie teorie tipografie, credo sia il tempo di sperimentare, soprattutto per fare la differenza da un trend ormai quasi consolidato.

Ecco quindi un [layout a tre colonne](#) asimmetrico ed elastico con uso del background.

Scarica l'esempio [zip](#) di questa lezione

L'impaginazione dei contenuti centrali

Finora abbiamo visto i principali layout e le loro diverse varianti. Tutti gli esempi hanno una

formattazione e dei contenuti semplici per non appesantire troppo i fogli di stile, e soprattutto non distogliere l'attenzione dallo scopo principale di questa guida: l'impaginazione. Non potevo però trascurare una sezione dedicata al "micro-layout" cioè all'impaginazione dei contenuti centrali di una pagina web. Vedremo brevemente come gestire immagini, titoli e paragrafi, ma allo stesso modo si può procedere per qualsiasi elemento.

Float e clear

Prima di addentrarci in esempi pratici, è necessaria una brevissima introduzione sulle due proprietà fondamentali che useremo in questa parte della guida.

La proprietà **float** (vedi l'esposizione e gli esempi dalla Guida ai CSS di HTML.it) rende un elemento "fluttuante" a destra o a sinistra (*right* o *left*) del suo contenitore. Gli elementi adiacenti vengono affiancati sul suo lato libero e continuano disponendosi sotto. Gli elementi float, a differenza degli elementi posizionati, non vengono rimossi dal flusso della pagina, ma solo *traslati*.

La proprietà **clear** consente di impedire ad un elemento di avere elementi float su uno o entrambi i lati. I possibili valori sono *right*, *left* o *both*. La proprietà clear è spesso molto utile per controllare la disposizione degli elementi adiacenti ad elementi float. Il suo uso, spesso necessario in un layout che usa float, può purtroppo avere effetti non desiderati.

Per esempio, su un layout a tre colonne con il float, un **clear:both** applicato ad un qualsiasi elemento nella colonna centrale fa sì che questo si disponga verticalmente al di sotto delle due colonne laterali. Spesso, ma non sempre, si può ricorrere a soluzioni che non usano il clear: in caso contrario, e per avere meno limiti sull'impaginazione dei contenuti, il mio consiglio è di usare layout con posizionamenti assoluti per layout a tre colonne se prevediamo di usare la proprietà clear sulla colonna centrale. È anche ipotizzabile un layout a tre colonne combinato, in cui la colonna extra è float e quella di navigazione è posizionata assolutamente. Questo avrebbe due vantaggi significativi: nessun vincolo di lunghezza tra la colonna centrale e quella extra e la possibilità di usare il clear da un lato sulla colonna centrale.

Per i layout a due colonne il discorso è un po' più semplice. Generalmente gli elementi come le immagini nella sezione dei contenuti vengono resi float a sinistra. Definire la colonna di navigazione fluttuante a destra ci consente di usare il clear:left per la sezione principale senza avere interferenze di layout con la colonna di navigazione.

Testo e immagini

Vedremo brevemente come sia possibile ottenere una sezione composta da un'immagine, un titolo e un paragrafo. Questa soluzione può essere adottata per esempio per news accompagnate da immagini o per descrivere prodotti commerciali.

Ecco il codice HTML minimale:

```
<div class="textimage">  
  
<h2>Layout</h2>  
<p>Qui il testo...</p>  
<div class="clearer">&nbsp;&nbsp;&nbsp;</div>
```

```
</div>
```

Ed ecco il foglio di stile:

```
div.textimage{  
width: 400px;margin-bottom: 10px;  
border: 1px solid #000;background-color: #eee  
}
```

```
div.textimage img{  
float:left;  
margin: 10px  
}
```

```
div.textimage h2, div.textimage p{  
margin:0 10px;  
padding: 0  
}
```

```
div.textimage h2{  
margin-top: 10px  
}
```

```
div.clearer{clear: left}
```

Ecco il [risultato](#). Il meccanismo è questo: l'immagine viene resa float e distanziata dai bordi del suo contenitore con l'uso dei margini. Vengono impostati i margini del paragrafo e del titolo e il resto viene sistemato di conseguenza. L'uso del div con classe **clearer** non ha motivo strutturale ma solo di presentazione: nel caso in cui il testo sia troppo corto e non riesce a coprire l'altezza dell'immagine è necessario **contenere il float**. La tecnica usata qui è presentata da Eric Meyer nell'articolo [Containing float](#).

Questo esempio, immerso nella colonna centrale di un layout con i float, può risultare disastroso per l'uso del clear. Come evitare problemi? In questo caso si può ad esempio fissare l'altezza del contenitore, avendo cura di controllare che il testo ci stia. Sarà però necessario dimensionare testo e altezza del contenitore in maniera *concorde*: per il testo ridimensionabile, entrambi andranno dimensionati in em, oppure in maniera più sicura e semplice sarà possibile renderli fissi e dimensionarli in pixels.

Dividere una sezione in due colonne

Il supporto di testo su più colonne è previsto solo con i CSS3. Quello che vedremo qui è come ottenere dei paragrafi distinti su più colonne.

Questa impostazione è tipica di molti siti dedicati all'informazione, per avere una stretta somiglianza con i quotidiani e per questioni legate alla leggibilità. Nel 99% dei casi, questi siti adottano le tabelle per organizzare contenuti su più colonne nella sezione centrale. Una nota va fatta per quanto riguarda il dimensionamento dei font: una misura fissa è anche in questo caso la scelta più semplice e più stabile, ma meno accessibile.

Vedremo in questo paragrafo come realizzare un div suddiviso in due sezioni di pari larghezza. Con questa tecnica è anche possibile suddividere la colonna centrale di un layout a tre colonne ottenendo facilmente un layout a quattro colonne.

Ecco il codice HTML:

```
<div class="split2">
  <div>
    <h2>Paragrafo 1</h2>
    <p>Testo del paragrafo 1..</p>
  </div>
  <div>
    <h2>Paragrafo 2</h2>
    <p>Testo del paragrafo 2..</p>
  </div>
  <div class="clearer">&nbsp;</div>
</div>
```

Ora aggiungiamo un tocco di stile..

```
div.split2 div{
  float: left;
  width: 45%;
  width: 49%;
  width: 45%;
  padding: 0 2%
}
```

```
div.clearer{
  float: none; clear: left
}
```

Per ora non sono stati previsti stili sul contenitore, ma questi potrebbero coinvolgere larghezza, margini o padding. Il punto centrale di questo approccio è l'uso del **selettore discendente**: i due div contenuti all'interno del contenitore identificato dalla classe *split2* saranno resi float e avranno una larghezza effettiva (padding incluso) pari al 49%. Si è usata la versione semplificata del box model hack per rendere il comportamento uguale tra i vari browsers.

Quello che risulta è un codice HTML minimale e un'unica regola CSS abbastanza snella. Anche in questo caso l'uso del **clear** è necessario per impedire agli elementi successivi alle due colonne di subirne il float. Per non aggiungere div non strutturali è possibile dare il clear sull'elemento successivo al div con classe *split2*. È il caso dell'[esempio](#), in cui il clear viene dato sul div successivo.

Dividere una sezione in tre colonne

Allo stesso modo dell'esempio precedente è possibile suddividere un div in tre o più colonne. Vediamo brevemente il codice HTML:

```
<div class="split3">
  <div>
    <h2>Paragrafo 1</h2>
    <p>Testo del paragrafo 1..</p>
  </div>
  <div>
    <h2>Paragrafo 2</h2>
    <p>Testo del paragrafo 2..</p>
  </div>
  <div>
    <h2>Paragrafo 3</h2>
    <p>Testo del paragrafo 3..</p>
  </div>
  </div>
  <div class="clearer">&nbsp;</div>
</div>
```

Ed ecco il css minimale:

```
div.split3 div{
  float: left;
  width: 29%;
  width: 33%;
  width: 29%;
  padding: 0 2%
}
```

Ecco il [risultato](#). Sia questo esempio che il precedente usano un contenitore semi-fluido per contenere le colonne. Stringendo la finestra del browser tuttavia si noterà che le due colonne hanno comportamenti totalmente differenti sui vari browsers: scivolano in verticale una sopra l'altra (in IE), si sovrappongono l'un l'altra (Opera) oppure vengono tagliate dalla finestra del browser senza generare scrolling (Mozilla). Per evitare queste enormi differenze di comportamento in casi limite, è consigliabile disporre le due o tre colonne in un contenitore a larghezza fissa o elastico, che in tutti i browsers genererà scrolling se la finestra del browser è troppo stretta.

Esempio finale di microlayout

Ho preparato una [pagina completa](#) per mostrare come le tecniche qui presentate si possono utilizzare per impaginare i contenuti anche in pagine complesse e dense di contenuti quali portali e siti di quotidiani.

Si tratta di un layout fisso a tre colonne asimmetriche. In questo caso, a differenza di tutti i layout visti finora, si è usato un dimensionamento dei caratteri in pixel. C'è da notare come le mini-colonne centrali, anziché essere dimensionate in percentuale come negli esempi sopra, siano dimensionate in pixels: Internet explorer 5.5 dava infatti problemi con

il dimensionamento in percentuale, anche con il *box model hack*.

Definire il layout solo per lo schermo

Abbiamo presentato una buona quantità di layout, ma c'è ancora una questione da trattare: come preparare una pagina per la stampa. Provate a fare un anteprima di stampa dei siti web che visitate abitualmente. Risulterà che una buona percentuale di siti viene stampata decisamente male. È un aspetto importante e che merita un approfondimento.

Il punto centrale da considerare è che la larghezza *sicura* per la stampa di una pagina web è 560 pixels. Ciò significa, per fare un esempio, che una pagina realizzata con layout fisso largo 760pixels rischia di essere tagliata da uno o entrambi i lati. In questa prima sezione vedremo velocemente come preparare il layout alla stampa. Per ulteriori approfondimenti vi rimando agli articoli [Css design: going to print \(in inglese\)](#) di Eric Meyer e all'articolo "Stampe perfette di pagine web con i CSS".

Ci sono diversi modi per definire le regole per la stampa, li vedremo nel dettaglio spiegando come applicarli.

Definire il layout solo per lo schermo

Il modo più semplice per consentire una buona stampa è *non definire* regole per la stampa, e cioè lasciare che il foglio di stile venga usato solo a schermo. Ecco come fare:

```
<head>
  <title>Html</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
  <style type="text/css" media="screen">
    @import url("stilisito.css");
  </style>
</head>
```

Specificare il foglio di stile in questo modo ha due vantaggi fondamentali:

- i browser moderni applicheranno il foglio di stile solo a schermo;
- la pagina di stampa verrà quindi resa senza stili associati.

Per la stampa verranno usati gli stili di default del browser. Uno degli svantaggi di questa tecnica è che gli elementi della pagina verranno quindi stampati tutti, e nell'ordine di codice HTML.

Specificare un CSS ad hoc per la stampa

Una migliore alternativa a quella vista nella precedente lezione consiste nello specificare un foglio di stile solo per la stampa:

```
<head>
  <title>Html</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
  <style type="text/css" >
    @import url("stilisito.css") screen;
    @import url("stilistampa.css") print;
  </style>
</head>
```

In questo caso, il foglio di stile per la stampa potrà essere molto semplice. Ecco per esempio un CSS per la stampa di un qualsiasi layout a tre colonne in cui non si stampano le due colonne laterali:

```
body{ font: 10pt arial, sans-serif}
h1,h2,h3,h4,h5,h6{font-family: georgia,serif}
p{line-height: 1.2em}
a{font-weight:bold}
div#navigation, div#extra{display: none} /*non stampa le due colonne laterali*/
```

Bisognerà provvedere alcuni stili di natura tipografica (per la stampa il miglior dimensionamento dei font è in punti, unità di misura che si rivela totalmente inappropriato per lo schermo) ed eventualmente nascondere le sezioni aggiuntive quali la navigazione e la colonna extra. La pagina in questo modo risulterà pulita e ben stampata.

Ridefinire il foglio di stile per lo schermo

La terza soluzione che si può adottare è quella di ridefinire nel foglio di stile le regole CSS per lo schermo. Ecco come fare attraverso le dichiarazioni della sezione head:

```
<head>
  <title>Html</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
  <style type="text/css" >
    @import url("stilisito.css");
    @import url("stilistampa.css") print;
  </style>
</head>
```

Notare come, se non viene specificata l'applicabilità di un foglio di stile, le sue regole verranno applicate a tutti i dispositivi possibili, quindi anche la stampa. In tal caso, quindi, nel foglio di stile per la stampa (che avrà priorità maggiore) andremo a ridefinire solo le regole necessarie che differiscono da quelle del CSS generico.

Questa soluzione è senz'altro la più elaborata, e può andar bene se si vuole mantenere nella stampa una forte somiglianza con la versione a schermo. C'è da dire che i layout fluidi si adattano perfettamente alla larghezza della pagina di stampa, e in tal caso se la stampa è buona e se vogliamo mantenere la stessa impaginazione o le scelte tipografiche usate a schermo si può anche non definire un insieme di regole per la stampa.

Se abbiamo realizzato un layout fisso o elastico, renderlo fluido per la stampa è molto semplice, ecco la regola da aggiungere al CSS per la stampa:

```
div#container{width: auto}
```

Non specificare fogli di stile per la stampa può andare bene per layout fluidi a una e due colonne, mentre per il layout fluidi a tre colonne bisognerà considerare che la larghezza dedicata alla colonna centrale sarà proporzionale a 560 pixels (è infatti questa la larghezza totale dell'area sicura per la stampa) meno la larghezza delle due colonne. Se per esempio le due colonne sono larghe 170pixels l'una, la larghezza utile alla colonna centrale sarà il 40% della larghezza utile di stampa. Una percentuale a parer mio troppo piccola. In questo caso si può pensare di nascondere dalla stampa, per esempio, la colonna di

navigazione.

Ecco quindi il foglio di stile minimale per la stampa che va bene per layout a tre colonne con float e posizionamenti, siano essi fluidi, liquidi o elastici:

```
div#container{ width: auto} /*rende il layout a larghezza di stampa*/  
div#navigation{display: none} /*non stampa la navigazione*/  
div#content{margin-left: 0} /*elimina il margine sinistro*/
```

Se volessimo lasciare tutta la larghezza di stampa per la colonna centrale, la colonna extra si può rendere **static** e a seconda dell'ordine di codice HTML (e quindi della strategia usata: float o posizionamenti assoluti) questa verrà stampata in orizzontale sopra o sotto i contenuti. Ecco il codice da utilizzare in questo caso:

```
div#container{width: auto}  
div#navigation{display: none}  
div#content{margin: 1em 0}  
div#extra{width: auto; float: none; position: static} /*rende static la colonna extra*/
```

Stampa di sfondi e bordi

Abbiamo visto velocemente come preparare una stampa migliore per una pagina web. Ci sono altre questioni di cui parlare brevemente: il colore e le immagini di sfondo non vengono stampate dal browser secondo le impostazioni predefinite. L' unica cosa che viene stampata per default sono i bordi, che sarebbe meglio eliminare in fase di stampa. Ecco il codice:

```
div#container{ border-width: 0}
```

Indice dei layout presentati nella guida

In questa lezione vedremo tutti i layout presentati, con i loro pro e contro. I layout sono facilmente riadattabili alle proprie esigenze. Sarà possibile renderli fluidi, elastici o fissi; con colonne elastiche o fisse; con carattere fisso o relativo. Attraverso questa pagina è possibile consultare e/o scaricare il codice HTML e CSS di ogni layout.

Layout Monolitico

Questa è la forma più elementare di layout, e può andare bene per siti di piccole dimensioni (generalmente con un ordine di pagine sotto la decina), per siti con menu dropdown o per siti con macrosezioni (sottositi). Per la sua semplicità non necessita di posizionamenti o float nella versione elementare.

Layout Monolitico

Questa è la forma più elementare di layout, e può andare bene per siti di piccole dimensioni (generalmente con un ordine di pagine sotto la decina), per siti con menu dropdown o per siti con macrosezioni (sottositi). Per la sua semplicità non necessita di posizionamenti o float nella versione elementare.

Layout monolitico fluido

Pro:

- Estrema facilità di codifica, adatto a tutte le risoluzioni.

Contro:

- All'apparenza risulta molto scarno, e i contenuti sono di difficile lettura ad alte risoluzioni

Layout monolitico fisso

Pro:

- Facilità di codifica, i contenuti risultano maggiormente consultabili

Contro:

- Richiede un foglio di stile per la stampa

Layout monolitico elastico

Pro:

- Consente una proporzionalità tra larghezza della pagina e dimensione del font

Contro:

- Richiede il dimensionamento relativo del font

Layout monolitico con altezza minima

Pro:

- La finestra risulta totalmente impegnata anche con pochi contenuti

Contro:

- Aggiungere i contenuti è una soluzione migliore piuttosto che avere troppo spazio bianco

Layout a due colonne

Adatto per siti di dimensioni medie, consente una migliore organizzazione dei contenuti. Il layout a due colonne con i float presenta un indubbio vantaggio rispetto ad un layout a due colonne con posizionamenti assoluti: non impone vincoli sulla lunghezza relativa delle colonne in presenza di footer. D'altra parte obbliga, nella maggior parte dei casi, ad avere la navigazione prima dei contenuti, cosa che può avere influenze negative su tematiche quali l'accessibilità e il posizionamento sui motori di ricerca.

Layout a due colonne con float - versione base

Pro:

- Nessun vincolo sulla lunghezza relativa delle due colonne

Contro:

- La navigazione precede in ordine di codice HTML i contenuti

Layout a due colonne di colore diverso con float

Pro:

- Le due colonne vengono separate cromaticamente ed è più facile focalizzare i contenuti

Contro:

- I contenuti devono essere più lunghi della colonna secondaria
- La navigazione precede in ordine di codice html i contenuti

Layout a due colonne con float e background

Pro:

- Le due colonne vengono separate cromaticamente ed è più facile focalizzare i contenuti

- Nessun vincolo sulla lunghezza relativa delle due colonne

Contro:

- La navigazione precede i contenuti
- La colonna di navigazione deve essere dimensionata in pixels

Layout a due colonne con posizionamenti assoluti - versione base

Pro:

- Ordine di codifica migliore per accessibilità e motori di ricerca
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- I contenuti devono essere più lunghi della colonna secondaria in presenza del footer

Layout a due colonne cromatiche con posizionamenti assoluti

Pro:

- Le due colonne vengono separate cromaticamente ed è più facile focalizzare i contenuti
- Ordine di codifica migliore per l'accessibilità e motori di ricerca
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- I contenuti devono essere più lunghi della colonna secondaria in presenza del footer

Layout a due colonne con posizionamenti assoluti e immagine di background

Pro:

- Le due colonne vengono separate cromaticamente ed è più facile focalizzare i contenuti
- Ordine di codifica migliore per accessibilità e motori di ricerca
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- I contenuti devono essere più lunghi della colonna secondaria in presenza del footer
- La colonna di navigazione deve essere dimensionata in pixels

Layout a tre colonne

Il layout a tre colonne è uno dei più diffusi e permette di gestire siti di dimensioni medio-alte, fino ad arrivare ai portali. La colonna aggiuntiva può contenere svariate funzionalità del sito: news, sondaggi, banner, navigazione supplementare e quant'altro.

Layout a tre colonne con float versione base

Pro:

- Nessun vincolo sulla lunghezza relativa delle due colonne

Contro:

- La colonna centrale è l'ultima in ordine nel codice HTML

Layout a tre colonne con float e bordi

Pro:

- Le colonne vengono separate visivamente

Contro:

- La colonna centrale è l'ultima in ordine nel codice HTML
- La colonna centrale deve essere la più lunga

Layout a tre colonne con float e colore

Pro:

- Le colonne vengono separate visivamente

Contro:

- La colonna centrale è l'ultima in ordine nel codice HTML
- La colonna centrale deve essere la più lunga
- Le due colonne laterali sono dello stesso colore

Layout a tre colonne con float e background

Pro:

- Le colonne vengono separate visivamente
- Nessun vincolo sulla lunghezza relativa delle tre colonne

Contro:

- La colonna centrale è l'ultima in ordine del codice HTML
- Le due colonne laterali devono essere dimensionate in pixel

[Layout a tre colonne con posizionamenti - Versione base](#)**Pro:**

- La colonna centrale è la prima in ordine nel codice HTML
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- La colonna centrale deve essere la più lunga

[Layout a tre colonne separate da bordi fini e posizionamenti](#)**Pro:**

- Le colonne vengono separate visivamente
- La colonna centrale è la prima in ordine nel codice HTML
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- La colonna centrale deve essere la più lunga

[Layout a tre colonne con colore e posizionamenti assoluti](#)**Pro:**

- Le colonne vengono separate visivamente
- La colonna centrale è la prima in ordine nel codice HTML
- Possibilità di usare float e clear senza interferenze con il layout.

Contro:

- La colonna centrale deve essere la più lunga

[Layout a tre colonne con bordi spessi e posizionamenti assoluti](#)**Pro:**

- Le colonne vengono separate visivamente

- Le colonne laterali possono avere due colori diversi
- La colonna centrale è la prima in ordine nel codice HTML
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- La colonna centrale deve essere la più lunga
- Il colore delle due colonne verrà stampato se non si utilizza un foglio di stile per la stampa
- Le due colonne laterali devono avere la stessa larghezza

[Layout a tre colonne con posizionamenti assoluti e immagini di background](#)

Pro:

- Le colonne vengono separate visivamente
- Le colonne laterali possono avere due colori diversi
- La colonna centrale è la prima in ordine nel codice HTML
- Possibilità di usare float e clear senza interferenze con il layout

Contro:

- La colonna centrale deve essere la più lunga
- Le due colonne laterali devono essere dimensionate in pixel

[Layout a tre colonne asimmetrico ed elastico con background e colonne fisse](#)

Le due colonne sono dimensionate in pixels e per una migliore disposizione dei contenuti hanno larghezza differente. La larghezza dell' intero layout cambia intervenendo sulla dimensione del carattere attraverso le opzioni del browser

[Layout con contenuti impaginati](#)

In questo esempio si usano i posizionamenti assoluti per il layout, e i contenuti nella colonna centrale vengono impaginati su mini-colonne attraverso float e clear

Gli esempi completi

[Scarica gli esempi completi](#) presentati nella guida. Il codice CSS è incorporato nei vari documenti HTML.