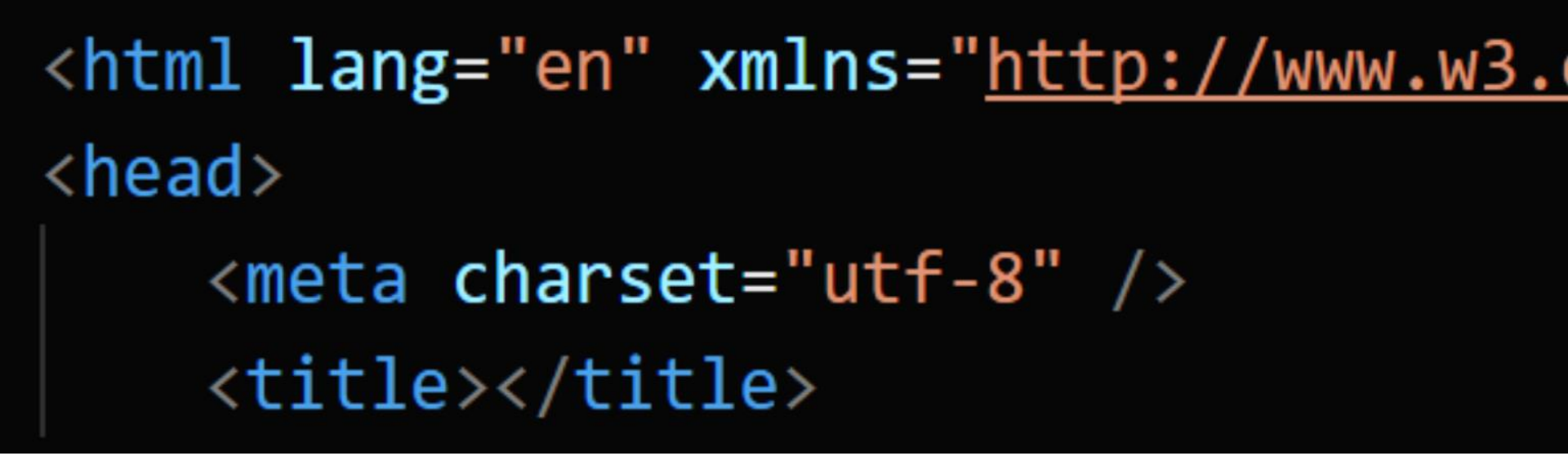


Web Coding

Prototipazione di ipertesti e siti web in HTML
ed introduzione alla creazione di stili grafici con fogli di stile CSS.

Unità Didattica UD04: impostare margini e bordi, posizionamento e visibilità; impaginazione

prof. Giovanni Borga

Margini e bordi

Proprietà singole per i margini

Le proprietà **margin-bottom**, **margin-left**, **margin-top**, **margin-right** definiscono la distanza tra i quattro lati di un elemento e gli elementi adiacenti. Si applicano a tutti i tag e non sono ereditate.

Valori ammessi:

- **un valore numerico con unità di misura**. Il valore è espresso in termini assoluti.
- **un valore in percentuale**. Il valore è calcolato come percentuale rispetto alla larghezza/altezza del blocco contenitore.
- **auto**. Distribuisce il margine equamente nel senso orizzontale e/o verticale. *(molto utilizzato per allineare al centro della finestra un div a larghezza fissa)*

Esempi:

p {margin-bottom: 10px;} - **h1 {margin-left: 20%;}**

NB: si ricordi che i margini adiacenti NON si sommano: se ad esempio per un elemento si imposta un margine inferiore ed esso si trova sopra ad un altro elemento che abbia impostato un margine superiore, la distanza tra i due sarà quella maggiore e non data dalla somma delle due proprietà (meccanismo del margin collapsing).

Sintassi abbreviata per i margini

La proprietà a sintassi abbreviata per i margini è **margin.** Con essa è possibile specificare i valori per tutti e quattro i lati di un elemento.
Si applica a tutti gli elementi e non è ereditata.

Valori ammessi:

- un ***valore numerico*** con unità di misura.
- un ***valore in percentuale***.
- **auto**. *(Per la proprietà margin il valore auto significa che la distanza sarà automaticamente calcolata rispetto alla larghezza dell'elemento contenitore).*

Esempio di utilizzo tipico con tutti e quattro i valori:

div {margin: 10px 15px 10px 20px;}

L'ordine di lettura va inteso in senso orario. *(il primo valore si riferisce al lato superiore, il secondo a quello destro, il terzo al lato inferiore, il quarto a quello sinistro)*.

In pratica la sintassi vista nell'esempio equivale alla seguente:

div {margin-top: 10px; margin-right: 15px; margin-bottom: 10px; margin-left: 20px; }

Esempi di utilizzo della proprietà margin

Nella definizione dei valori è possibile **mischiare percentuali con valori assoluti** in unità di misura.

Un ulteriore abbreviazione della sintassi si può ottenere usando **tre, due o un solo valore**.
Queste le regole da seguire:

- se si usano **tre valori**, il primo si riferisce al margine superiore, il secondo a quelli sinistro e destro, il terzo a quello inferiore
- se si usano **due valori**, il primo si riferisce ai lati superiore e inferiore, il secondo al sinistro e al destro
- se si usa **un solo valore**, un uguale distanza sarà ai quattro lati.
  (Un uso interessante del valore auto per i lati sinistro e destro è quello che consente di centrare in tal modo un elemento rispetto alla pagina o al box contenitore).

Esempi:

```
p {margin: 20px 10px;}

div {margin: 20px;}

h1 {margin: 10px auto;}
```

Margini interni (padding)

I margini interni sono le distanze tra il contenuto di un elemento e il suo bordo. Per gestirli si utilizzano le proprietà del padding. Come per margin, nella sintassi di padding abbiamo 4 proprietà singole per i 4 lati:

padding-top, padding-right, padding-bottom, padding-left

e una a sintassi abbreviata: **padding**.

Anche il padding si applica a tutti gli elementi e non è ereditata.

Valori ammessi

- un **valore numerico** con unità di misura. Il valore è espresso in termini assoluti.
- un **valore in percentuale**. Il valore è calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore.

Esempi:

```
p.box {padding-bottom: 10px;}
p {padding-left: 10px;}
```

NB: per gestire il margine tra finestra e contenuto in HTML valida più o meno per tutti i browser si utilizzava:

```
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
```

Con i CSS invece è sufficiente: `body { margin: 0px; padding: 0px; }`

Proprietà per i bordi

I tre aspetti che caratterizzano un bordo sono: COLORE, STILE, SPESSORE.

Combinando i **tre aspetti** con i **quattro lati** di un blocco arriviamo a definire:

12 proprietà singole:

border-top-color, border-top-style, border-top-width,
border-bottom-color, border-bottom-style, border-bottom-width,
border-right-color, border-right-style, border-right-width,
border-left-color, border-left-style, border-left-width

Combinando invece la **proprietà generale** (border) prima con **i 4 lati** e poi con **i 3 aspetti** otteniamo:

8 proprietà a sintassi abbreviata:

border,
border-bottom, border-top, border-right, border-left,
border-color, border-style, border-width

Proprietà colore per i bordi

La proprietà per gestire il colore dei bordi è **border-color**; utilizzata anche come proprietà singola riferita ad un lato (es. border-top-color).

I valori possibili per il colore sono:

- ***un qualsiasi colore*** *(secondo le modalità già viste per il testo)*
- la parola chiave **inherit** *(colore ereditato dall'elemento parente)*

Proprietà stile per i bordi

La proprietà per gestire lo stile dei bordi è **border-style**, utilizzata anche come proprietà singola riferita ad un lato (es. border-top-style).

Lo stile di un bordo può essere espresso con una delle seguenti parole chiave:

- **none**. *(l'elemento non presenta alcun bordo e lo spessore equivale a 0)*.
- **hidden**. *(equivalente a none)*
- **dotted** *(puntinato)*
- **dashed** *(tratteggiato)*
- **solid** *(continuo)*
- **double**. *(continuo doppio)*
- **groove** *(scanalato)*
- **ridge** *(cornice)*
- **inset** *(incassato)*
- **outset** *(in rilievo)*

Proprietà spessore per i bordi

Infine la proprietà per gestire lo spessore dei bordi è **border-width**, utilizzata anche come proprietà singola riferita ad un lato (es. border-top-width).

Valori ammessi:

- ***un valore numerico*** con unità di misura
- **thin**. *(Bordo sottile)*.
- **medium**. *(Bordo di medio spessore)*.
- **thick**. *(Bordo di largo spessore)*.

Esempi di regole per i bordi

Un primo esempio definisce tutti gli aspetti di stile per il **bordo sinistro**:

```
div {
  border-left-color: black;
  border-left-style: solid;
  border-left-width: 1px;
}
```

Ma risulta molto più comodo scrivere così: `div {border-left: 1px solid black;}`

Se invece vogliamo definire **tutti gli aspetti per tutti i bordi** possiamo usare questo modo:

```
div {
  border-width: 1px 2px 1px 2px;
  border-style: solid;
  border-color: black red black red;
}
```

Per ciascuna di queste proprietà è possibile definire da uno a quattro valori, uno per lato.

(ricordiamo che se ne usiamo quattro l'ordine di lettura è: top, right, bottom, left; se invece ne impostiamo uno, due o tre valgono le regole viste per i margini)

Esempio di utilizzo della proprietà border

L'ultima proprietà a sintassi abbreviata è **border**. Con essa possiamo definire con una sola regola le impostazioni per i quattro bordi.

Il suo uso è però limitato a un solo caso, peraltro molto comune: quello in cui **i quattro bordi abbiano tutti lo stesso colore, lo stesso stile e lo stesso spessore**.

Esempio:

```
div {border: 2px solid black;}
```

Posizionamento degli elementi

Le proprietà speciali Display, Float e Clear

Nei fogli di stile, abbiamo tre proprietà speciali che possono modificare radicalmente la presentazione del documento.

Sono, dunque, strumenti **potenti** ma da utilizzare con molta cautela padroneggiando bene la sintassi e le regole e verificando le eventuali differenze di rendering tra i **diversi browser**.

Queste tre proprietà sono :

Display
Float
Clear

Display

Abbiamo visto in precedenza la fondamentale distinzione tra elementi blocco, inline e lista che è alla base di (X)HTML. Con la proprietà **display** possiamo superare la rigidità di queste regole associando i diversi comportamenti agli elementi che si desidera, indipendentemente dalla loro natura HTML.
La proprietà è ereditata.

I valori ammessi per display sono numerosi, in alcuni casi possono non essere pienamente supportati; i più utilizzati sono i seguenti:

- **inline**. *(L'elemento assume le caratteristiche degli elementi inline)*.
- **block**. *(L'elemento viene trattato come un elemento blocco)*.
- **inline-block**. *(Elemento inline ma con la possibilità di impostare larghezza e altezza)*.
- **list-item.** *(elemento formattato come <li>)*
- **none**. *(L'elemento non viene mostrato. O meglio: è come se non fosse nemmeno presente nel documento, in quanto non genera alcun box. Diversa la proprietà visibility:hidden, che invece nasconde l'elemento)*.
- **initial.** *(Ripristina il default)*
- **inherit.** *(Eredita la proprietà dall'elemento genitore)*

Impieghi della proprietà Display

Potrebbe non essere chiaro il motivo dell'esistenza di una proprietà come display. In realtà in molti casi la flessibilità offerta risulta piuttosto utile.

Le immagini, per esempio, sono per loro natura elementi inline, si inseriscono nel testo ed è talvolta complicato gestirne il posizionamento. Se volessi mostrarle in una riga tutta per loro mi basterebbe impostare il display su block, così:

img {display: block;}

Sul valore none invece; se si volesse trasformare velocissimamente una pagina rendendola solo di testo basterebbe associare un CSS alternativo contenente la regola:

img {display: none;} .

Per controllare con migliore efficacia le dimensioni di un elemento inline si usa invece molto spesso regole come:

span.elemento {display: inline-block; width: 100px}

Float

Float è una proprietà importantissima quando si impostano regole CSS per definire il layout del documento.

Con questa proprietà è possibile **rimuovere un elemento dal normale flusso del documento** e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore. Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore di float. (Il floating in HTML era gestibile solo per le immagini tramite la proprietà align).

La proprietà non è ereditata.

I valori ammessi di float sono tre:

- **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 di default in mancanza di dichiarazione esplicita. L'elemento mantiene la sua posizione normale)*.

NB: Molto importante: I DIV a cui si applica il float manifestano un comportamento non intuitivo dovuto al fatto che la loro dimensione dipende dal contenuto. Per ottenere un layout specifico può essere necessario **impostare esplicitamente una dimensione orizzontale con la proprietà width** per evitare che si sovrappongano o vengano posti sotto ad altri elementi.

Esempio:

```
#div1 {width: 200px; float:right;}
```

Clear

La proprietà **clear** serve a impedire che al fianco di un elemento compaiano altri elementi con il float.
Si applica solo agli elementi blocco e non è ereditata.

La logica di clear è la seguente: il float sposta un elemento dal flusso normale del documento; è possibile che esso venga a trovarsi in posizioni non desiderate, magari al fianco di altri elementi che vogliamo invece tenere separati. clear risolve questo problema.

Valori ammessi per clear:

- **none**. *(Gli elementi con float possono stare a destra e sinistra dell'elemento).*
- **left**. *(Si impedisce il posizionamento a sinistra).*
- **right**. *(Si impedisce il posizionamento a destra).*
- **both**. *(Si impedisce il posizionamento su entrambi i lati).*

Esempio:

```
#titolo1 {float: left; clear: right;}
```

In questo esempio, il titolo, pur essendo impostato con floating a sinistra, non potrà avere altri elementi alla sua destra, nemmeno se questi sono impostati a loro volta con float a sinistra.

Posizionamento degli elementi

Il posizionamento con i CCS è molto complesso ma altrettanto potente.

Affrontiamo l'argomento con la proprietà **position**, ovvero la proprietà fondamentale per la gestione della posizione degli elementi. Position si applica a tutti gli elementi ma non è ereditata.

I valori ammessi di position sono quattro:

- **static**
- **absolute**
- **fixed**
- **relative**

Ciascuno di questi valori merita un spiegazione approfondita da cui emergeranno i concetti di base che governano le regole sul posizionamento.

Valori di Position

Static (è il valore di default)
E' il valore predefinito per tutti gli elementi non posizionati secondo un altro metodo.

Absolute
L'elemento, o meglio, il box dell'elemento viene rimosso dal flusso del documento ed è posizionato in base alle coordinate fornite con le proprietà top, left, right o bottom. Il posizionamento avviene sempre ***rispetto al contenitore dell'elemento*** che è il primo elemento antenato (ancestor) con posizionamento diverso da static. Se tale elemento non esiste il posizionamento avviene in base all'elemento radice HTML (nella maggioranza dei casi l'area che contiene il documento e che ha inizio dall'angolo superiore sinistro). Un elemento posizionato in modo assoluto scorre insieme al resto del documento.

Fixed
Anche in questo caso il box dell'elemento viene sottratto al normale flusso del documento. A differenza di absolute, il box contenitore è sempre la finestra principale del browser, ovvero l'area del contenuto (viewport). Inoltre un box posizionato con fixed non scorre con il resto del documento.

Relative
L'elemento viene posizionato ***relativamente alla posizione che l'elemento avrebbe occupato nel normale flusso del documento.***
La posizione viene sempre impostata con le proprietà top, left, bottom, right. Tuttavia non indicano un punto preciso, ma l'ammontare dello spostamento in senso orizzontale e verticale rispetto alla posizione iniziale.

Gestione della posizione

La posizione precisa degli elementi si indica con quattro proprietà il cui significato è fin troppo esplicito:

Top
Left
Bottom
Right

I valori ammessi per il posizionamento sono:

- un ***valore numerico*** con unità di misura
- un ***valore in percentuale*** La percentuale è relativa all'altezza dell'elemento contenitore.
- **auto**

Se il significato è intuitivo, le regole di comportamento delle quattro direzioni non lo sono altrettanto.

Infatti il **funzionamento di queste proprietà cambia secondo l'impostazione di position**, come possiamo vedere dalla slide successiva…

Top, Left, Bottom, Right

Top: es: **#div1 {top: 10px;}** - **#div1 {top: 10%;}**

- ***Position: absolute / fixed*** : top è la distanza verticale rispetto al bordo superiore dell'elemento contenitore.
- ***Position: relative*** : top è lo spostamento rispetto al lato superiore della posizione originaria.
  Usando valori positivi di top l'elemento viene spostato in basso, mentre con valori negativi verso l'alto.

Left: es: **#div1 {left: 30px;}**

- ***Position: absolute / fixed*** : left è la distanza dal bordo sinistro del box contenitore.
- ***Position: relative*** : left è lo spostamento rispetto al lato sinistro della posizione originaria.
  Valori positivi di left spostano l'elemento verso destra, valori negativi verso sinistra.

Bottom: es: **#div1 {bottom: 50px;}**

- ***Position: absolute / fixed*** : bottom è la distanza verticale rispetto al bordo inferiore dell'elemento contenitore.
- ***Position: relative*** : bottom è lo spostamento rispetto al lato inferiore della posizione originaria.
  Valori positivi di bottom spostano l'elemento verso l'alto, mentre valori negativi verso il basso.

Right. es: **#div1 {right: 50px;}**

- ***Position: absolute / fixed*** : right è la distanza dal bordo destro del box contenitore.
- ***Position: relative*** : right è lo spostamento rispetto al lato destro della posizione originaria.
  Valori positivi di right spostano l'elemento verso sinistra, valori negativi verso destra.

NB: in ogni caso, essendo la posizione sempre definita da due coordinate, si usano per lo più le proprietà top e left per posizionare con precisione gli elementi del documento.

Visibilità e relazioni tra box

Visibility, z-index sono due proprietà che influiscono sull'aspetto visuale dei box definendone visibilità e relazione con gli altri box presenti nella pagina.

Visibility determina se un elemento debba essere visibile o nascosto. Si applica a tutti gli elementi e non è ereditata. I valori ammessi sono i seguenti:

- **visible**. *(L'elemento è visibile. Valore di default)*.
- **hidden**. *(L'elemento è nascosto, ma mantiene il suo posto nel layout dove apparirà come una zona vuota. In ciò è diverso dal valore none della proprietà display)*.

Esempi: **div#box {visibility: hidden;}** - **p#par1 {visibility: visible;}**

Z-index è invece una proprietà che imposta l'ordine di posizionamento dei vari elementi sulla base di una scala di livelli. E' utile nel contesto del posizionamento dinamico (cioè con position absolute o relative); in seguito ad un posizionamento, infatti, è possibile che un elemento si sovrapponga ad un altro rendendolo illeggibile. Impostando lo z-index è possibile modificare l'ordine di «impilamento» degli elementi.

I valori ammassi per z-index sono:

- **auto**. *(L'ordine dei livelli è uguale per tutti gli elementi)*.
- ***un valore numerico***. *(Un valore superiore indica un livello superiore)*.

Esempio: **div#box1 {z-index: 34;}**

Impaginazione e layout della pagina

Le sezioni «logiche» di un documento

INTESTAZIONE
(o HEADER)

Si estende orizzontalmente per tutto lo spazio a disposizione del layout. In verticale generalmente si estende per una frazione dell'altezza (es. 80 - 150 pixel).
L'intestazione riporta il **nome del sito, la descrizione** o un sottotitolo e un link che rimanda 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.

NAVIGAZIONE
(o MENU)

Molto frequente nei siti web, è un **indice** che permette di navigare tra i contenuti. Il menu principale deve essere ben visibile, leggibile e distinguibile dai contenuti. E' opportuno 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 browser. È inoltre importante ricordare all' utente dove si trova, evidenziando in qualche modo la pagina o sezione corrente.

CONTENUTO
(o CONTAINER)

I contenuti principali che si distinguono dal resto **occupano la maggior parte dello spazio**. E' quasi sempre la parte centrale della pagina e contiene sia testo, sia immagini, video e altri elementi interattivi o multimediali.

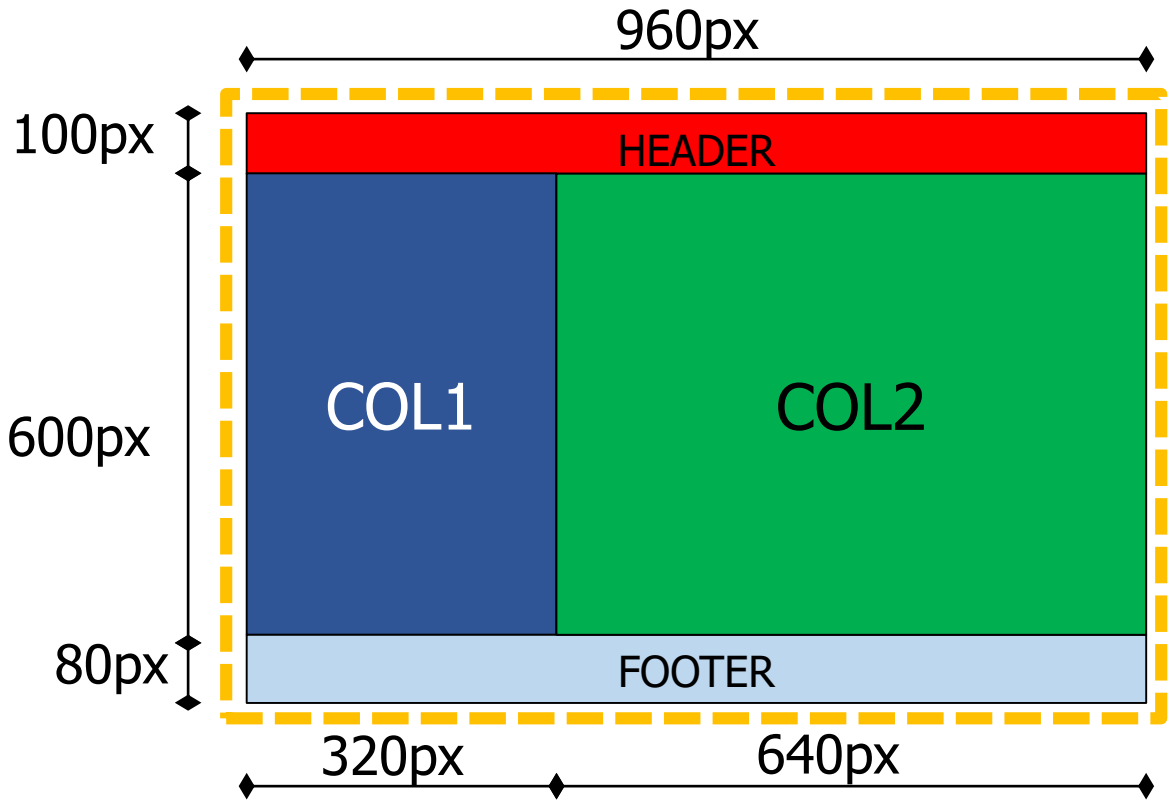
PIE' DI PAGINA
(o FOOTER)

Il piè di pagina è una sezione disposta a fondo pagina e contiene informazioni generali come copyright, contatti ecc.. E' opportuno che sia anch'esso presente in ogni pagina, ben distinguibile anche se in secondo piano rispetto al resto.

Il classico layout fisso a due colonne

Per un layout fisso standard occorrono almeno 5 DIV:

1.CONTAINER
2.HEADER
3.COLONNA1
4.COLONNA2
5.FOOTER



Costruzione passo-passo del layout fisso a due colonne - 1/9

CSS

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```

Il markup (HTML) contiene solo gli elementi di impaginazione e di contenuto. Dimensioni, posizioni e formati vanno definiti nel CSS.
E' **indispensabile assegnare degli ID** a ciascun elemento di impaginazione.

Costruzione passo-passo del layout fisso a due colonne - 2/9

CSS

```
<style>

    #container{
       width:960px;}

    #header{
        background-color: red;}

    #col1{
        background-color: blue;}

    #col2{
        background-color: green;}

    #footer{
        background-color:#B0D0E8;}

 </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```

Costruzione passo-passo del layout fisso a due colonne - 3/9

CSS

```
<style>

   #container{
      width:960px;}

   #header{
      background-color: red; width:960px;}

   #col1{
      background-color: blue; width:320px;}

   #col2{
      background-color: green; width:640px;}

   #footer{
      background-color:#B0D0E8; width:960px;}

 </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```

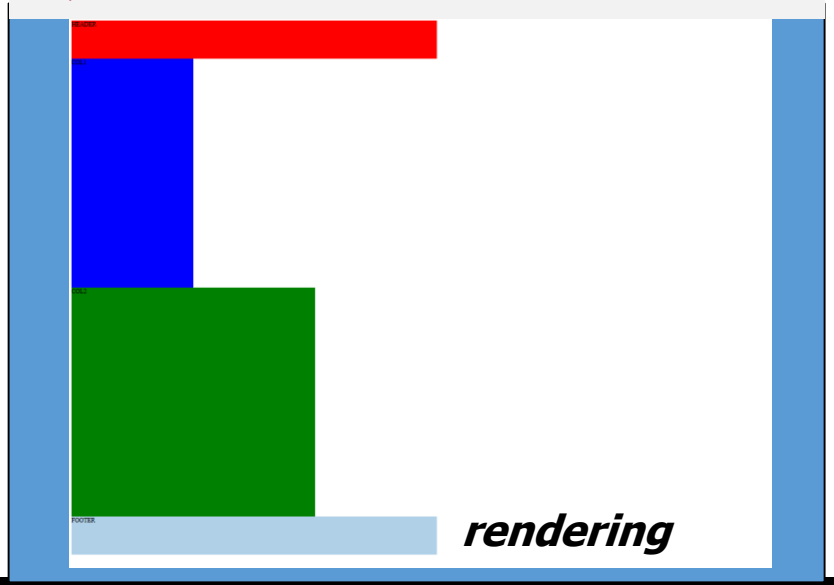
Costruzione passo-passo del layout fisso a due colonne - 4/9

CSS

```
<style>
   #container{
         width:960px;}

   #header{
         background-color: red;
         width:960px;height:100px;}

   #col1{
         background-color: blue;
         width:320px;height:600px;}

   #col2{
         background-color: green;
         width:640px;height:600px;}

   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



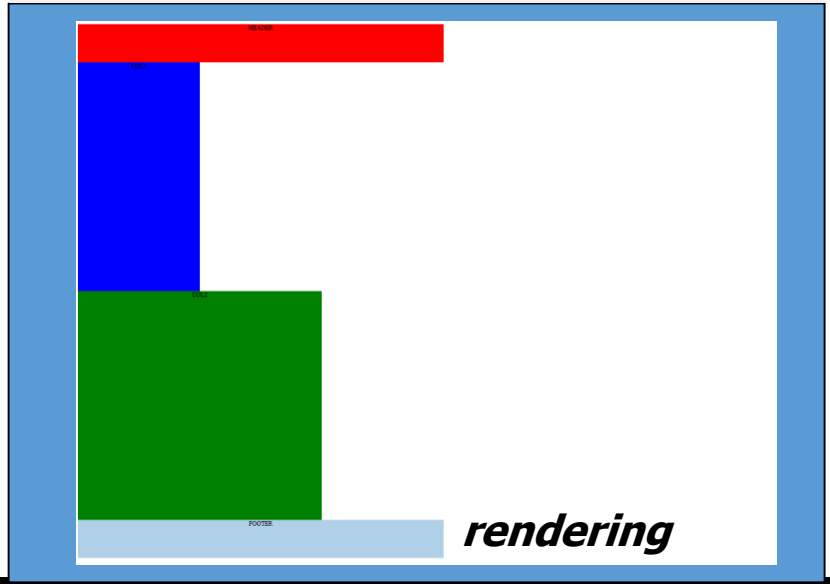
Costruzione passo-passo del layout fisso a due colonne - 5/9

CSS

```
<style>
   #container{
         width:960px;}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



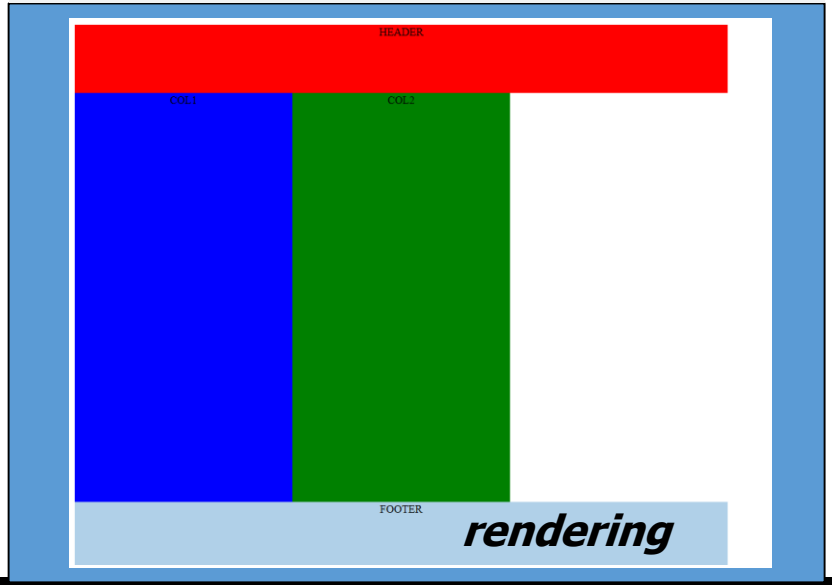
Costruzione passo-passo del layout fisso a due colonne - 6/9

CSS

```
<style>
   #container{
         width:960px;}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center; float:left}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



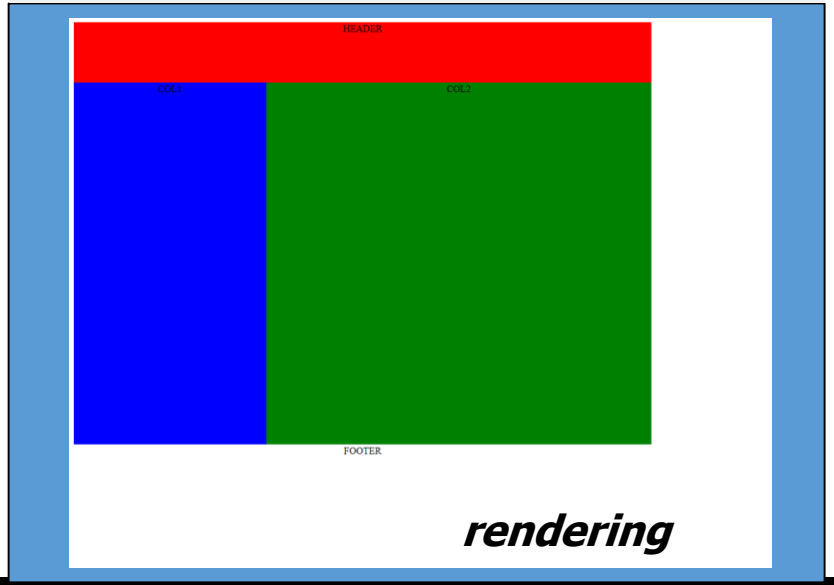
Costruzione passo-passo del layout fisso a due colonne - 7/9

CSS

```
<style>
   #container{
         width:960px;}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center; float:left}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center; float:left}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



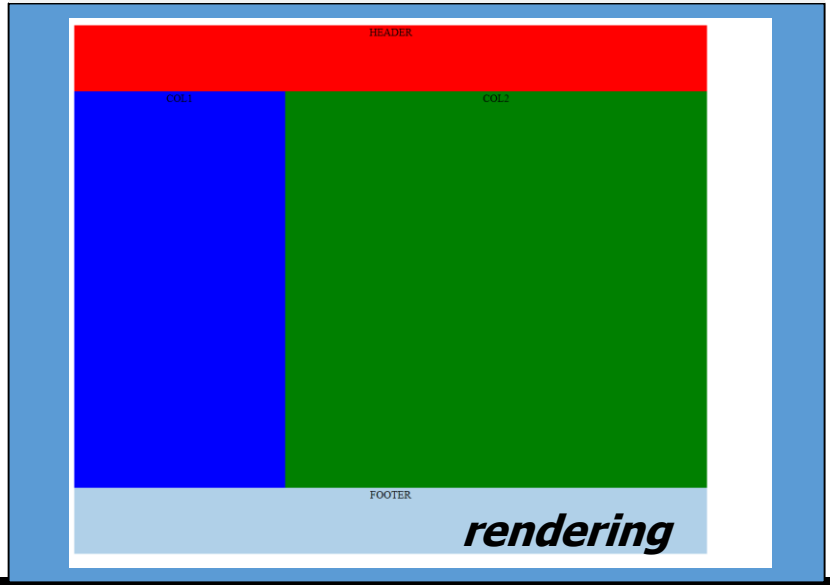
Costruzione passo-passo del layout fisso a due colonne - 8/9

CSS

```
<style>
   #container{
         width:960px;}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center; float:left}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center; float:left}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center; clear:left}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



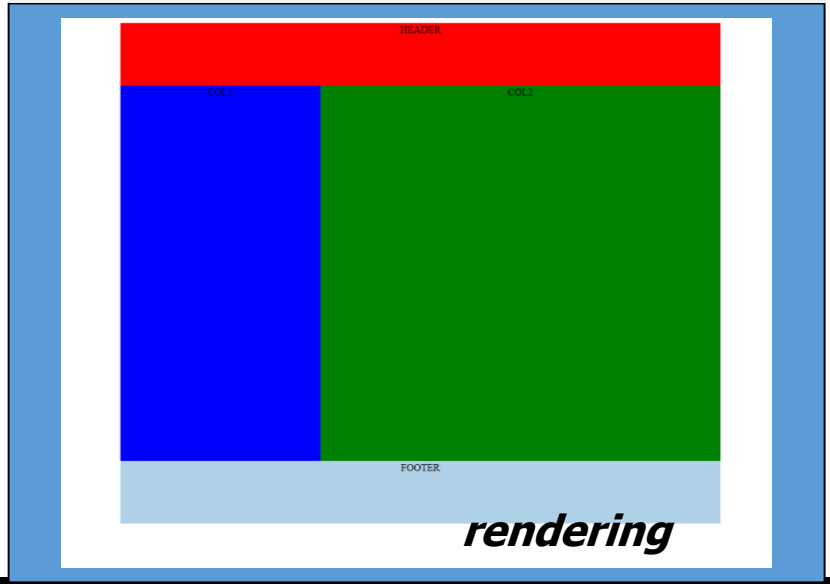
Costruzione passo-passo del layout fisso a due colonne - 9/9

CSS

```
<style>
   #container{
         width:960px; margin:auto}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center; float:left}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center; float:left}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center; clear:left}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



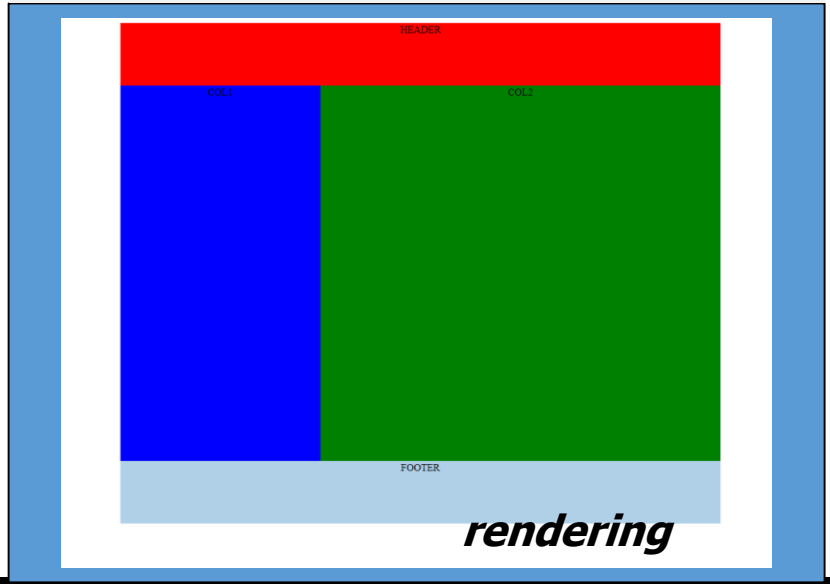
Riepilogo punti chiave del layout fisso a due colonne

CSS

```
<style>
   #container{
         width:960px; margin:auto}
   #header{
         background-color: red;
         width:960px;height:100px;
         text-align:center}
   #col1{
         background-color: blue;
         width:320px;height:600px;
         text-align:center; float:left}
   #col2{
         background-color: green;
         width:640px;height:600px;
         text-align:center; float:left}
   #footer{
         background-color:#B0D0E8;
         width:960px;height:100px;
         text-align:center; clear:left}
   </style>
```

HTML

```
<div id="container">
    <div id="header">HEADER</div>
    <div id="col1">COL1</div>
    <div id="col2">COL2</div>
    <div id="footer">FOOTER</div>
</div>
```



Esercizio n.4

Costruzione di layout

Creare 9 pagine con i layout descritti di seguito e le seguenti impostazioni di base:

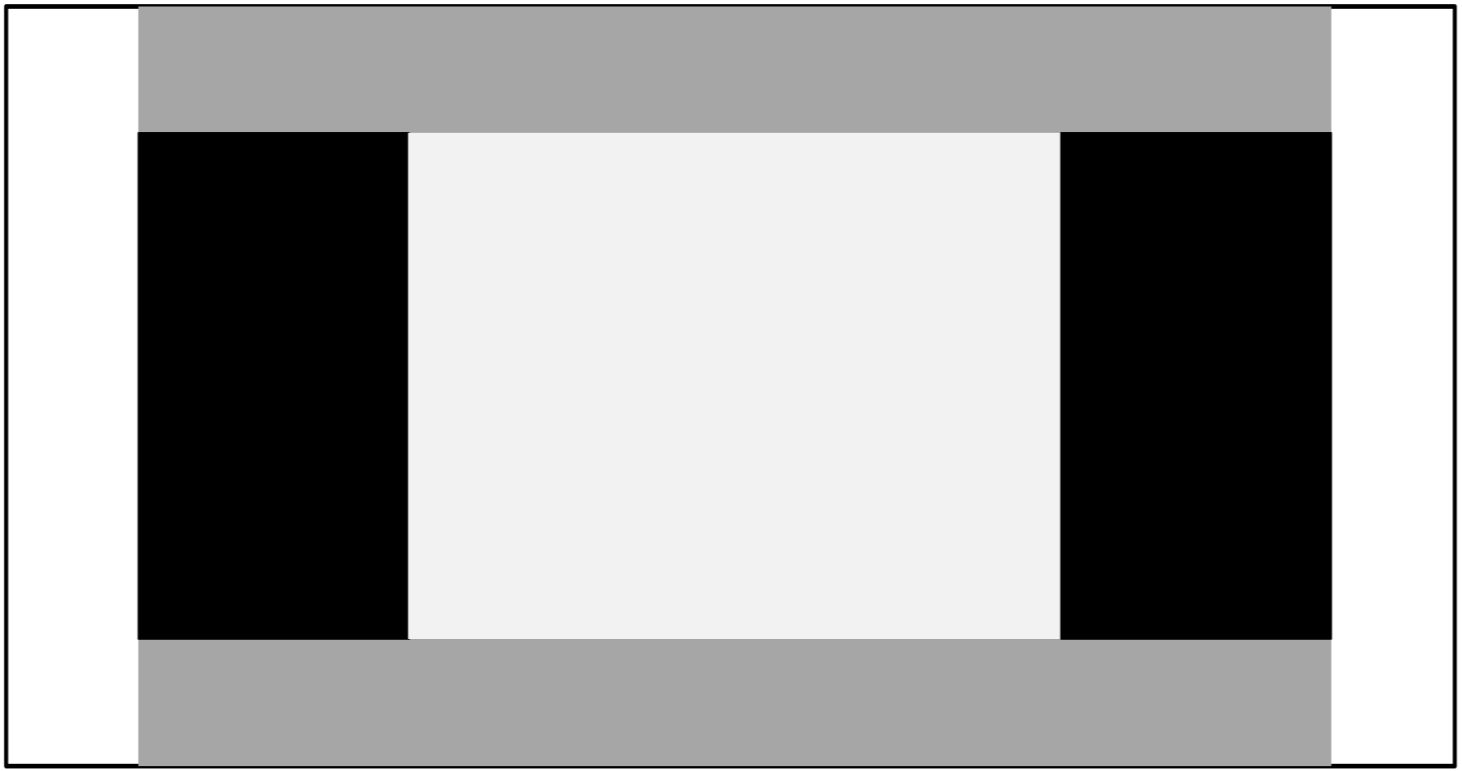
- **Larghezza** fissa 960px centrato nel browser.
- Tutti i box con **DIV** senza bordi (usiamo colore di sfondo per distinguere i blocchi) e dimensioni in pixel.
- **In via eccezionale**, visto che non abbiamo i contenuti, impostiamo le altezze dei tre blocchi principali in modo esplicito: intestazione 100px, corpo 400px, piè di pagina 100px (vedi schema):

Negli schemi a seguire, i **box azzurri barrati** indicano la presenza di un'immagine. Scegliere un'immagine a piacere (anche la stessa ripetuta) e inserirla nella pagina.

Pagina 1 (ipotetica home page)

Pagina 2 (layout standard a 2 colonne)

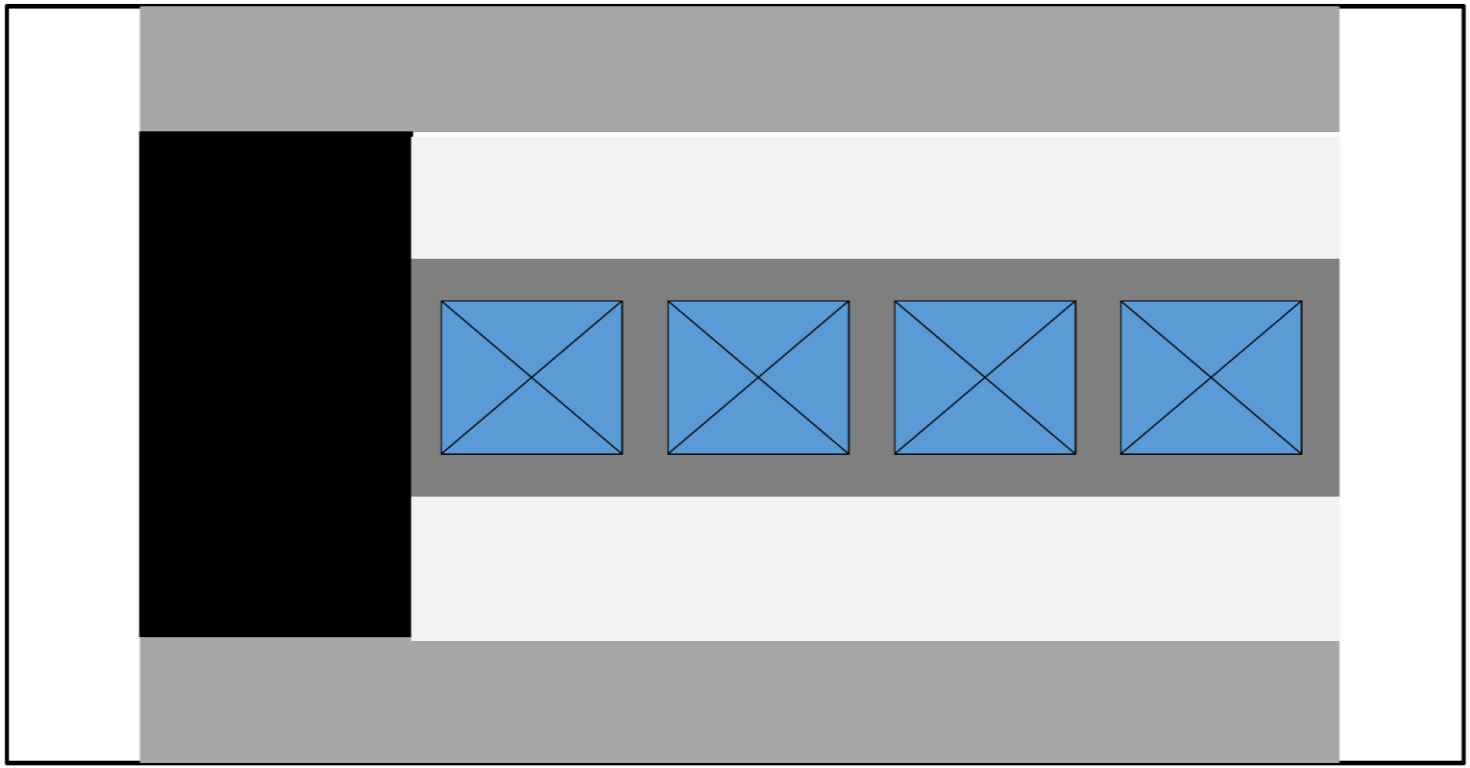
Pagina 3 (layout standard a 3 colonne)



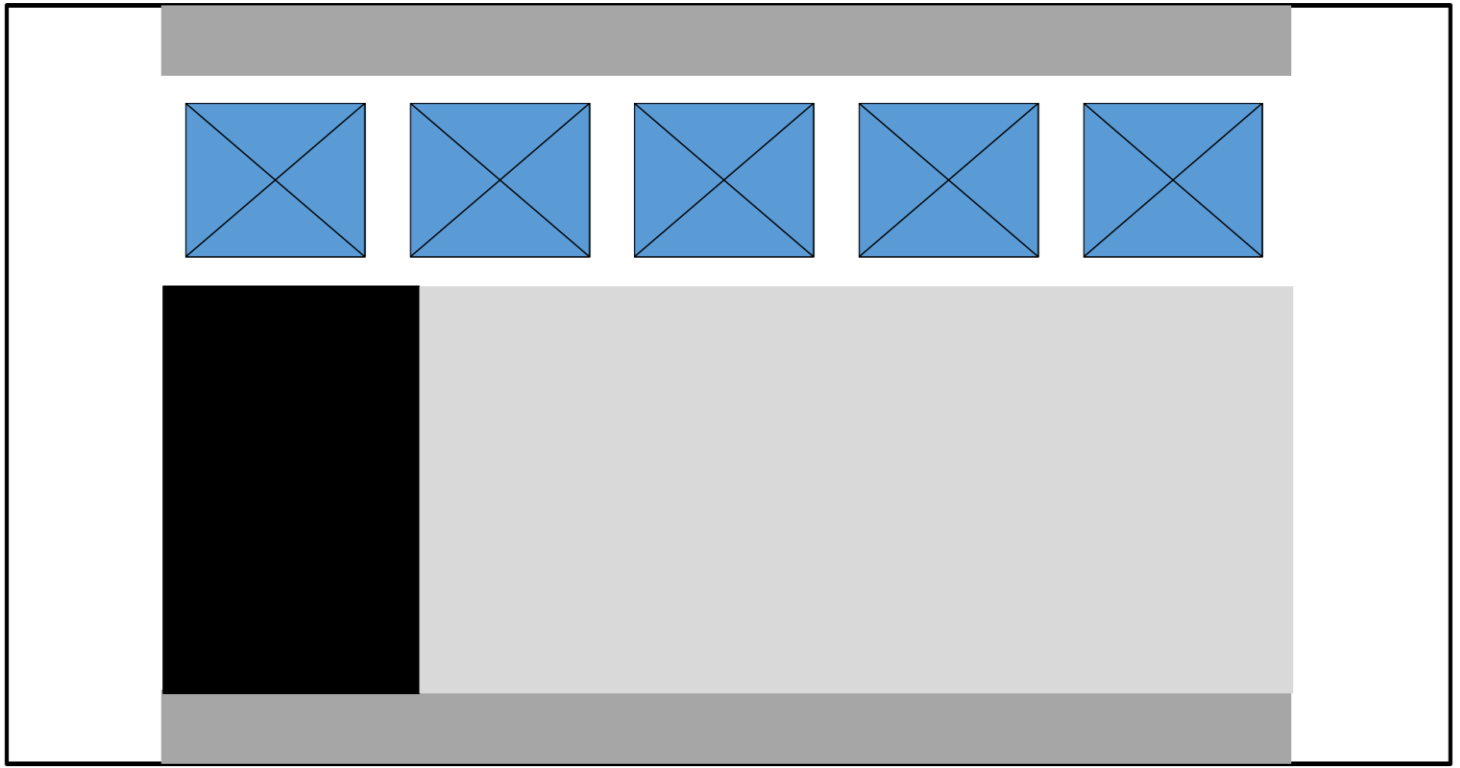
Pagina 4 (layout a 2 colonne con header a 3 bande orizzontali)

Pagina 5 (layout a 2 colonne con corpo a 2 sezioni orizzontali)

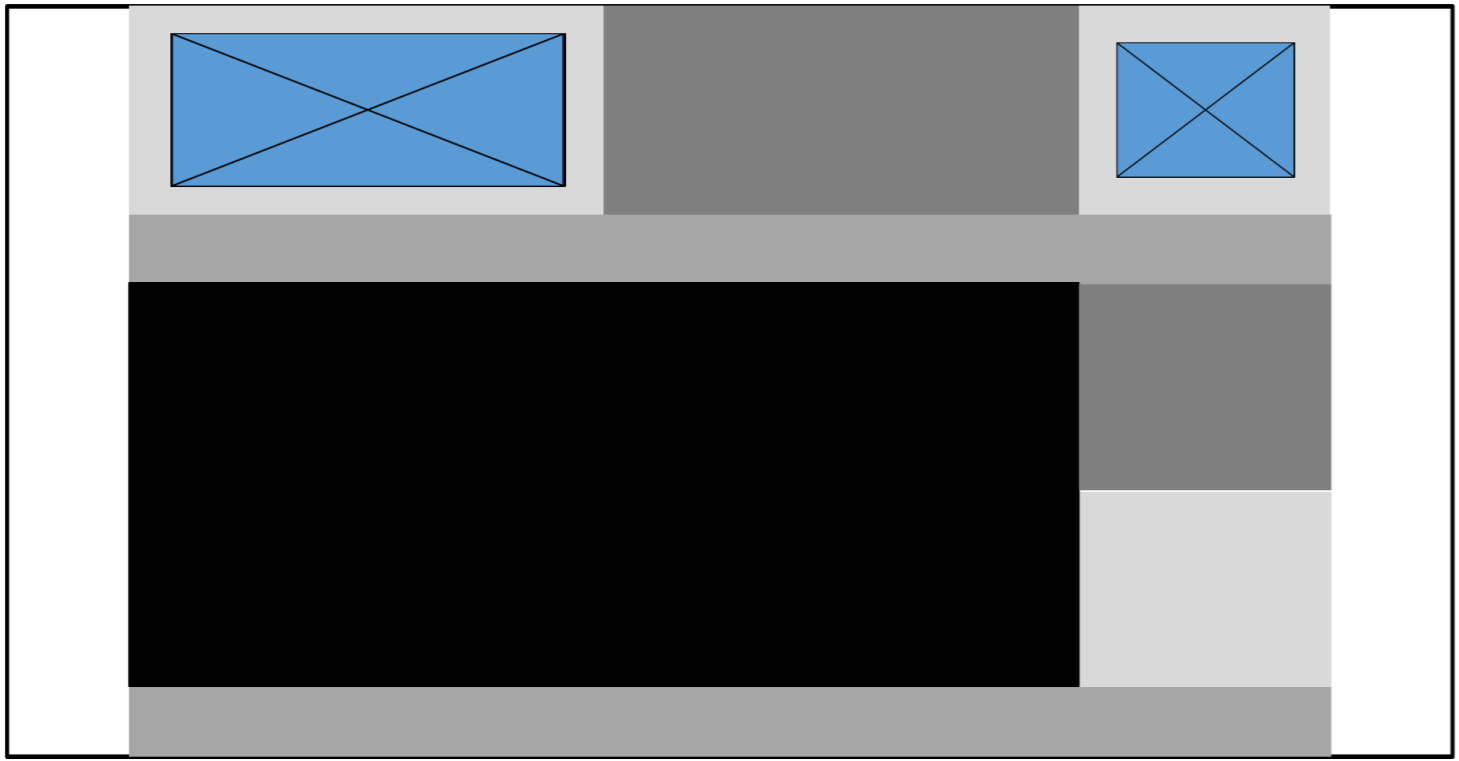
Pagina 6 (layout a 2 colonne, corpo a 3 sezioni e galleria immagini)



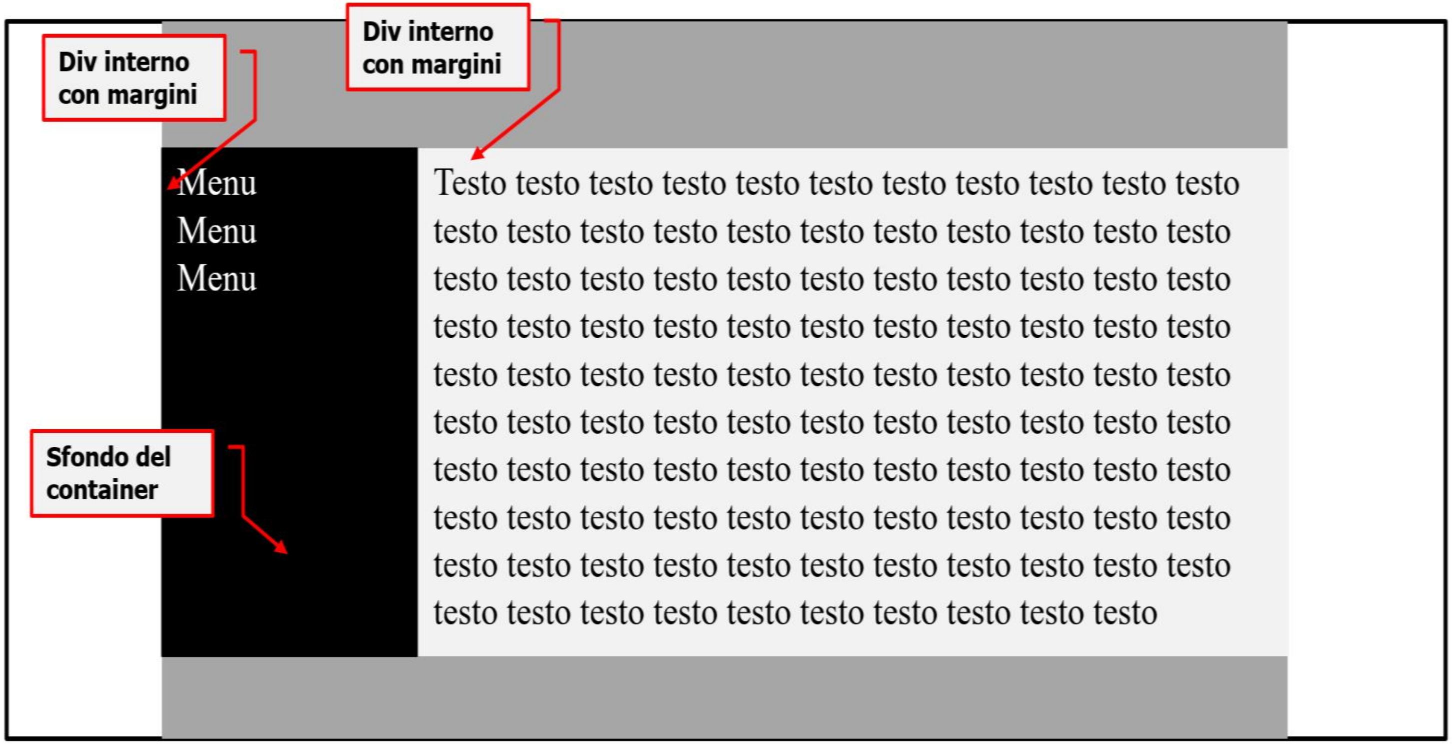
Pagina 7 (layout 2 colonne, header a 2 bande con galleria immagini)



Pagina 8 (layout 2 colonne, menu a dx a 2 sezioni, header a sezioni)



Pagina 9 (inserimento contenuti, margini, menu e corpo ad altezza dinamica anziché fissa a 400px)



Utilizzare uno qualsiasi dei layout precedenti