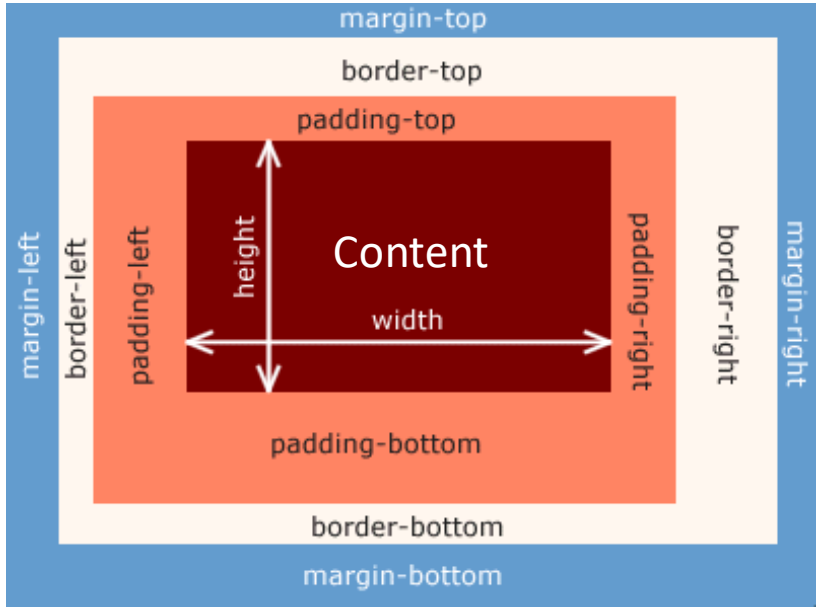
Box Model

Box model

Tutti gli elementi HTML sono rappresentati all'interno di rettangoli (box), che ne determinano l'ingombro e la dimensione.

Ogni box ha una serie di proprietà, utilizzabili per organizzare il layout, come riportato nella figura che segue:



Box model

Attorno al contenuto del tag HTML sono identificabili 3 zone distinte:

- Border: Linea di riquadro configurabile in termini di spessore, colore e tipo di linea (proprietà border già vista per le tabelle).
- Padding: Area trasparente posta tra il contenuto e il bordo, configurabile solamente in termini di spessore.
- Margin: Area trasparente posta all'esterno del contenuto e configurabile solamente in termini di spessore.

Le proprietà da settare sono border, padding e margin, oppure border-left, border-right….

Box model – dimensioni del contenuto

La larghezza totale di un box, così come la larghezza del solo contenuto, si calcolano in maniera differente in base alla situazione in cui ci si trova. Semplificando:

- Se è stata impostata la proprietà width allora la larghezza del contenuto sarà pari a tale valore.
- Se non è stata impostata la proprietà width:
  - Nel caso di tag block level la larghezza totale è pari alla larghezza del contenuto del tag padre
  - Nel caso di tag inline la larghezza del contenuto è pari alla dimensione necessaria per contenerlo

NB: Il tag body ha una larghezza di default pari alla larghezza della finestra del browser.

In ogni caso vale il seguente legame tra larghezza del contenuto (width) e larghezza totale:

LarghTotale = width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Box model – dimensioni del contenuto

L'altezza totale di un box, così come l'altezza del solo contenuto, si calcolano in maniera analoga a quanti visto per la larghezza. Semplificando:

- Se è stata impostata la proprietà height allora l'altezza del contenuto sarà pari a tale valore.
- Se non è stata impostata la proprietà height allora l'altezza del contenuto è pari alla dimensione necessaria per contenerlo

In ogni caso vale il seguente legame tra altezza del contenuto (height) e altezza totale:

AltTotale = height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

Box model – dimensioni del contenuto

E' possibile modificare il calcolo delle dimensioni visto precedentemente.

E' possibile fare in modo che width e heigth si riferiscano alle dimensioni totali del box, piuttosto che alle dimensioni del contenuto. Il contenuto sarà comunque calcolato con la solita forma.

```
.element  { box-sizing: border-box; }
```

Oppure, per applicare a tutti gli elementi:

```
*  { box-sizing: border-box; }
```

Box model – ereditarietà e automargini

Come già detto in precedenza l'ereditarietà degli stili non si applica a tutti. In particolare non si applica a border, margin e padding.

Se necessaria va attivata manualmente tramite la parola chiave inherit. Per esempio:

`margin: inherit;` permette di ereditare il valore dei margini dal tag padre.

Il margine destro e sinistro possono essere utilizzati per centrare orizzontalmente un elemento rispetto al suo contenitore. Per semplificare la gestione in caso di dimensioni di contenuto e contenitore variabili e/o non note a priori è possibile utilizzare la parola chiave auto:

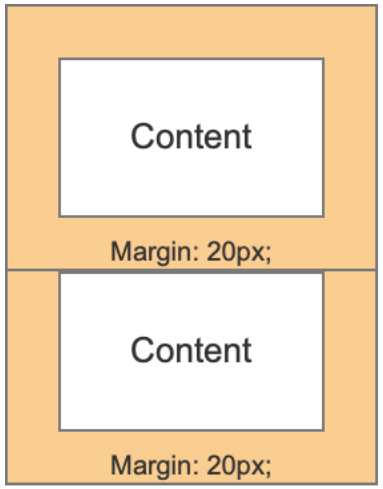
```
margin-left: auto;
```

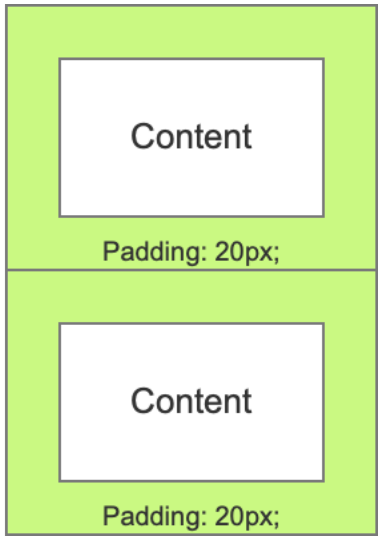
```
margin-right: auto;
```

Box model – margin collapsing

Due box differenti, posti uno vicino all'altro, vedranno i propri margini collassare uno sull'altro.

Il loro posizionamento non avrà una distanza pari alla somma dei due margini ma pari al maggiore tra i due margini. Somma che invece si ottiene utilizzando il padding.





Altri stili per box model

Stile	Significato
`min-width`	Specifica la larghezza minima. Il dimensionamento del box seguirà le regole precedentemente viste senza scendere sotto questa misura
`max-width`	Specifica la larghezza massima. Il dimensionamento del box seguirà le regole precedentemente viste senza salire sopra questa misura
`min-height`	Specifica l'altezza minima. Il dimensionamento del box seguirà le regole precedentemente viste senza scendere sotto questa misura
`max-height`	Specifica l'altezza massima Il dimensionamento del box seguirà le regole precedentemente viste senza salire sopra questa misura
`box-shadow`	Specifica un'ombra all'esterno del box

Unità di misura relative

Solo pixel?

Fino a questo momento abbiamo espresso larghezze e altezze in pixel, ovvero utilizzando una unità di misura assoluta.

Ciò non è sufficiente ne a realizzare pagine complesse, ne a realizzare pagine che si adattano a varie dimensioni e risoluzioni di schermo. Per esempio:

- Realizzare una tabella larga quanto tutta la pagina
- Legare la dimensione del font con l'altezza della riga
- Realizzare una pagina "proporzionale", nella quale dimensioni e spazi aumentano all'aumentare della dimensione del font.

In tutti questi casi è necessario utilizzare una unità di misura relativa, il cui valore non è direttamente espresso in pixel ma indirettamente riferito ad un'altra dimensione.

Unità di misura in css

Css prevede numerose unità di misura. Le più frequentemente utilizzate sono le seguenti:

- px: misura assoluta in pixel
- %: misura relativa alla dimensione (larghezza) del contenuto dell'elemento padre
- em: misura relativa alla dimensione del font prevista per l'elemento corrente (ereditata dal padre)
- rem: misura relativa alla dimensione del font prevista per l'elemento radice (html)
- vh: misura % relativa alla altezza della viewport (della finestra de browser)
- vw: misura % relativa alla larghezza della viewport (della finestra de browser)

Le unità di misura relative possono essere utilizzate in ogni punto in cui è necessario specificare una dimensione. Per esempio width, height, margin, padding, font-size…

Posizionamento degli elementi

Posizionamento degli elementi

Il posizionamento degli elementi può essere cambiato tramite la proprietà position. Sono possibili 4 valori:

- `static`: E' il posizionamento di default, quello che abbiamo visto fino ad ora. Prevede che l'elemento rimanga nella posizione standard.
- `relative`: L'elemento rimane nella sua posizione standard, dalla quale può però essere spostato tramite le proprietà top, bottom, left e right
- `absolute`: L'elemento non rimane più nella sua posizione standard. Viene rimosso dal flusso originale della pagina e viene posizionato con riferimento all'antenato più prossimo con posizionamento diverso da static (se non ne esistono il posizionamento è rispetto al body). Rispetto a tale posizione può essere spostato con top, bottom, left e right
- `fixed`: L'elemento non rimane più nella sua posizione standard. Viene rimosso dal flusso originale della pagina e viene posizionato con riferimento all'intera pagina. Rispetto a tale posizione può essere spostato con top, bottom, left e right

Top, bottom, left, right

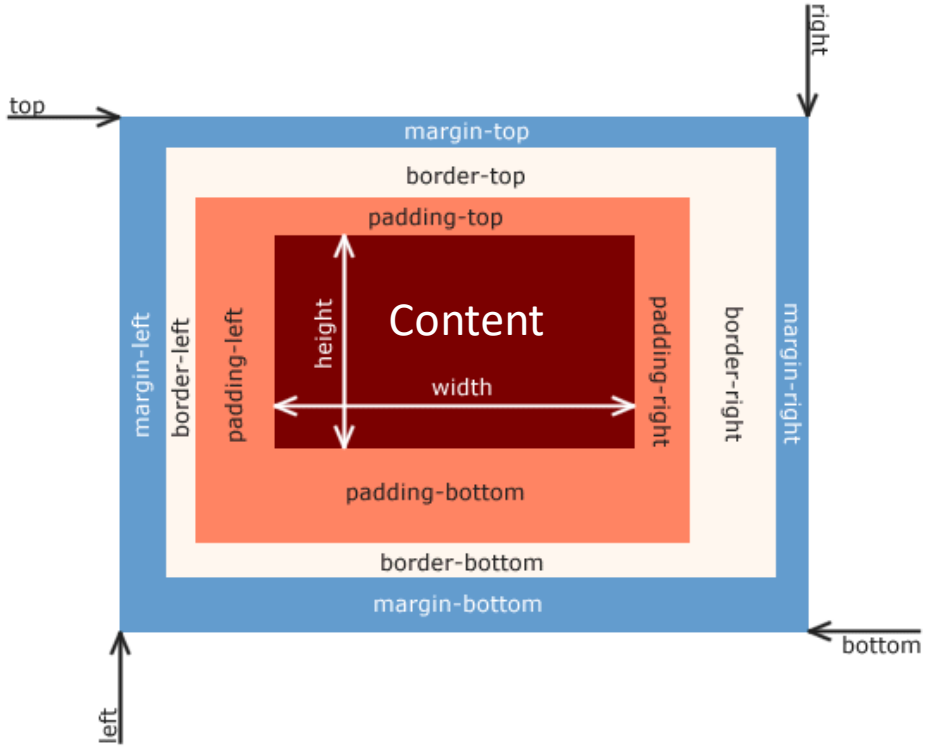
Quando la proprietà position vale relative, absolute o fixed è possibile specificare un offset tramite le proprietà top, bottom, right e left le quali inseriscono l'offset nel lato identificato e portano il box a spostarsi verso il lato opposto. Per esempio:

`top:100px` inserisce un offset sopra al box

portandolo a spostarsi verso il basso.

`right:100px` inserisce l'offset a destra del box porta

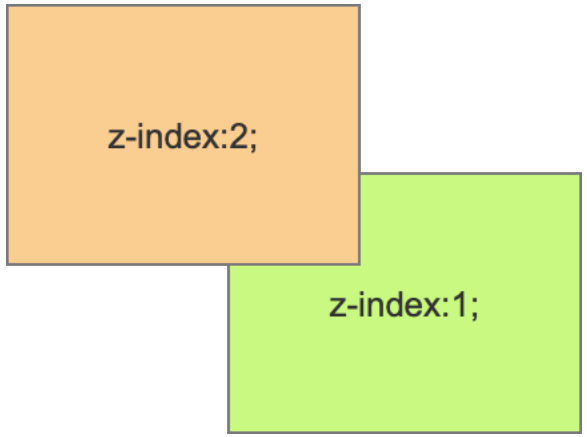
Spostarsi verso sinistra.



Z-index

La proprietà z-index è numerica e permette di modificare il posizionamento di elementi che si sovrappongono.

L'elemento che ha la proprietà z-index maggiore verrà visualizzato come sovrapposto all'altro.



Display

Tramite la proprietà display è possibile modificare il posizionamento degli elementi rimuovendone alcuni e/o specificando se devono essere considerati come block level o inline. Prevede tre valori:

- `none`: l'elemento è completamente rimosso, come se non fosse presente nell'html. Esiste anche la proprietà `visibility:hidden` che nasconde l'elemento ma non lo rimuove.
- `inline`: l'elemento viene rappresentato come un elemento inline. Le proprietà width e height vengono ignorate e l'elemento assume le dimensioni necessarie a contenerne il contenuto.
- `block`: l'elemento viene rappresentato come un elemento block level. Viene cioè rappresentato in una nuova riga ed occupa l'intera larghezza. Anche l'elemento successivo sarà rappresentato in una nuova linea.
- `flex`: I figli dell'elemento vengono organizzati in modo orizzontale o verticale secondo le regole di flexbox (le vedremo tra poco). L'elemento con la proprietà flex settata continua ad essere un block element

Float

Il normale posizionamento degli elementi può essere cambiato anche tramite la proprietà `float`.

La proprietà float permette di affiancare elementi block level che normalmente sarebbero rappresentati uno sotto l'altro. Prevede tre valori:

- `none`: E' il comportamento di default, quello che abbiamo visto fino ad ora.
- `left`: L'elemento viene spostato alla sinistra del suo contenitore permettendo agli elementi che seguono di riempire lo spazio rimanente sulla destra e di attorniarlo.
- `right`: L'elemento viene spostato alla destra del suo contenitore permettendo agli elementi che seguono di riempire lo spazio rimanente sulla sinistra e di attorniarlo.

L'effetto di float può essere ulteriormente modificato e "annullato" con la proprietà `clear`.

NB: L'utilizzo di left e right comporta che la larghezza dell'elemento sia calcolata basandosi sul contenuto.

Overflow

Quando le dimensioni di un elemento sono fisse è possibile che queste siano insufficienti per contenere l'intero contenuto. La proprietà `overflow` permette di stabilire come gestire questi casi. Prevede 4 valori:

- `visible`: è il valore di default e prevede che il contenuto ecceda le dimensioni del contenitore e venga rappresentato anche fuori dal suo box
- `hidden`: prevede che il contenuto eccedente venga nascosto. Sarà cioè visibile solamente il contenuto che rientra nel box
- `scroll`: prevede che il contenuto eccedente venga nascosto ma che venga aggiunta anche una scrollbar per potervi accedere.
- `auto`: prevede che il contenuto eccedente venga nascosto e che sia il browser a stabilire se deve essere aggiunta una scrollbar

Esistono le proprietà `overflow-x` e `overflow-y` per controllare lungo i due assi.

Flexbox

Flexbox

Flexbox, flexible box layout, o anche solamente flex è una modalità di organizzazione di elementi in una sola dimensione, in righe o colonne.

E' più flessibile e semplice da utilizzare rispetto al floating.

Flexbox è attivato, all'interno di un contenitore, con la proprietà: display: flex;

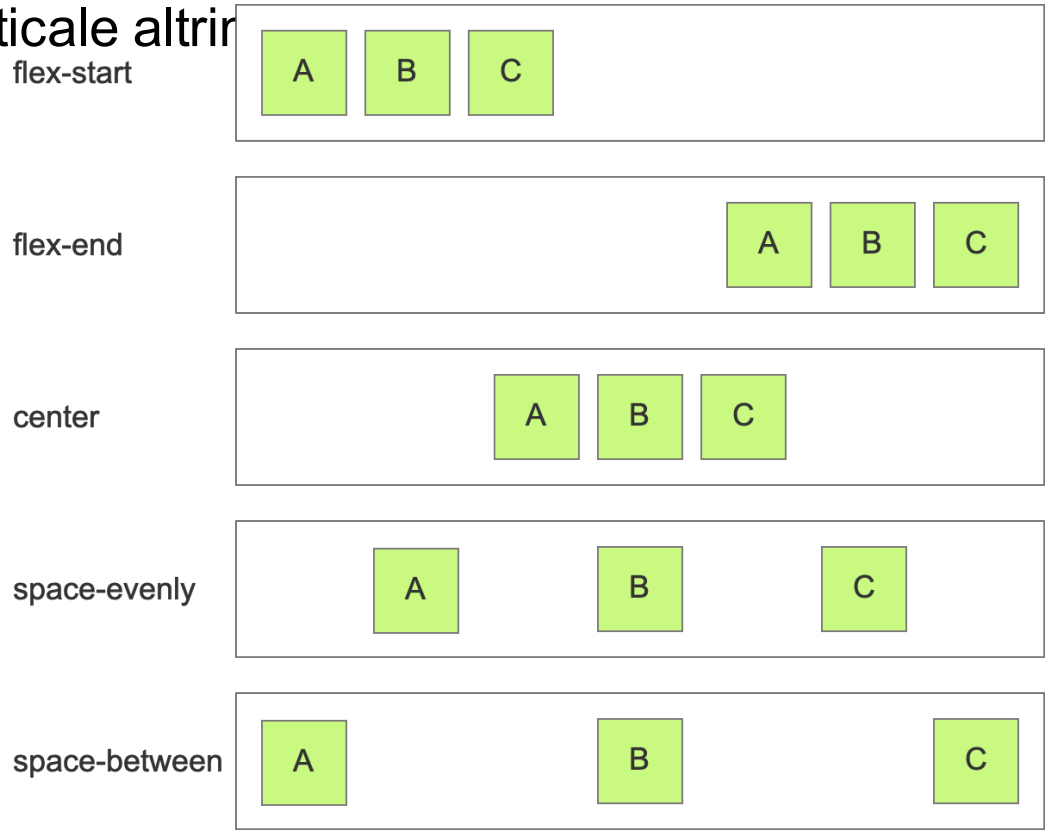
Gli elementi interni al contenitore vengono organizzati orizzontalmente o verticalmente in base al valore di flex-direction:

- Row: gli elementi sono organizzati in senso orizzontale, da sinistra a destra. E' il default e può essere omesso
- Column: gli elementi sono organizzati in senso verticale, dall'alto al basso
- Row-reverse: gli elementi sono organizzati in senso orizzontale, da destra a sinistra
- Column-reverse: gli elementi sono organizzati in senso verticale, dal basso all'alto

Flexbox

Oltre alla direzione gli elementi possono anche essere allineati, tramite l'uso di due proprietà:

- `justify-content`: Allineamento lungo l'asse principale (orizzontale per flex-direction=row*, verticale altrir



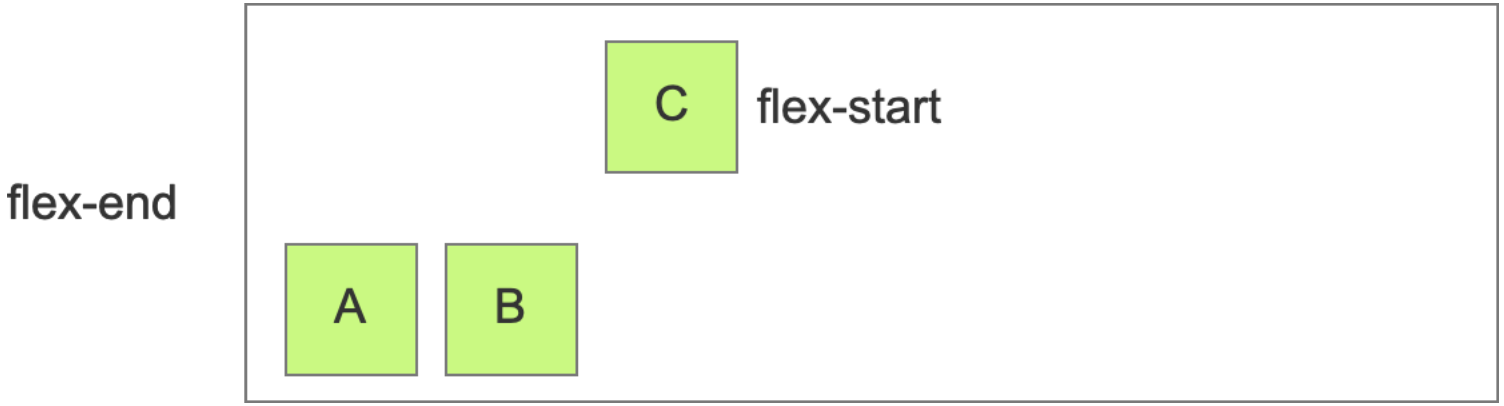
Flexbox

Oltre alla direzione gli elementi possono anche essere allineati, tramite l'uso di due proprietà:

- `Align-items`: Allineamento lungo l'asse secondario (verticale per flex-direction=row*, orizzontale altrimenti):

Flexbox

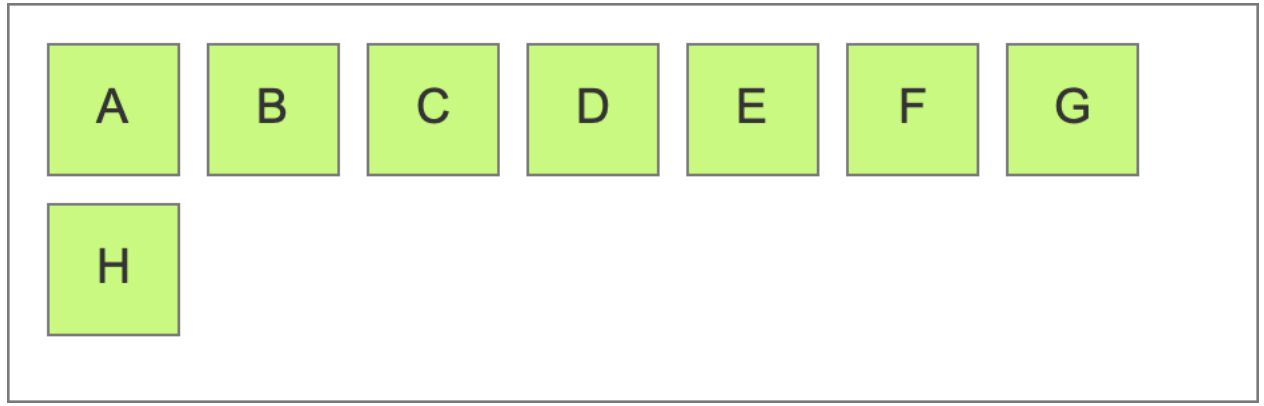
Sugli elementi interni è possibile settare la proprietà `align-self` per modificare l'allineamento rispetto a quanto previsto sul contenitore. Utilizza gli stessi valori previsti per align-items.



Flexbox

Il comportamento di default di flex prevede che tutti gli elementi rimangano in una sola riga/colonna senza andare capo. Se necessario vengono ridotte le dimensioni degli elementi.

E' possibile modificare il comportamento tramite la proprietà `flex-wrap:wrap` per fare in modo che gli elementi vadano "a capo" senza essere ridimensionati.



Media queries

@media queries

Tramite la direttiva @media è possibile utilizzare regole css differenti in base alle caratteristiche specifiche del dispositivo.

Le media queries sono alla base del responsive design, ovvero di siti web che si adattano alle dimensioni dello schermo, e dell'approccio mobile first.

E' possibile combinare due tipologie di caratteristiche:

- Media types: rappresentano la macro tipologia di dispositivo, schermo o stampa (considereremo solamente il primo)
- Media features: rappresentano caratteristiche specifiche del dispositivo, come la larghezza, l'orientamento, la risoluzione... (useremo solamente la larghezza)

Media types e media features possono essere combinati con operatori logici.

@media queries - esempio

```
@media screen and (max-width: 320px)
{
     Selettore
     {
         proprieta1: valore1;
     }
}
```

Le regole css inserite all'interno della @media query verranno eseguite solamente su schermo e fintanto che la larghezza dello schermo è <= 320px. Per larghezze superiori le regole sono ignorate.

@media queries - esempio

```
@media screen and (min-width: 600px)
{
     Selettore
     {
         proprieta1: valore1;
     }
}
```

Le regole css inserite all'interno della @media query verranno eseguite solamente su schermo e fintanto che la larghezza dello schermo è >= 600px. Per larghezze inferiori le regole sono ignorate.

@media queries - esempio

Le regole css continuano ad essere valutate in ordine di scrittura anche in presenza di @media queries.

```
H1{ color:black;   }
@media screen and (min-width: 400px)
{
     H1{ color:green; }
}
@media screen and (min-width: 800px)
{
     H1{ color:red;   }
}
```

@media queries

All'interno delle @media queries è possibile inserire qualsiasi selettore css ed è possibile settare qualsiasi proprietà.

Un caso molto frequente però è quello di ridefinire solamente alcune proprietà necessarie per adattare la pagina alle specifiche dimensioni del dispositivo. Per esempio:

- Modificare le larghezze % di elementi floating
- Modificare l'altezza del font del body lasciando che il resto si ridimensioni grazie e em e rem.
- Nascondere elementi aggiungendo display:none dove necessario

@media queries

E' possibile utilizzare media query anche all'interno del tag link per importare file css differenti in base al dispositivo e alle sue caratteristiche. E' sufficiente aggiungere l'attributo media valorizzato con la condizione di caricamento:

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="desktop.css">

<link rel="stylesheet" media="screen and (max-width: 600px)" href="tablet.css">
```