

# Introduzione a CSS - Parte 3

**Lo stilista colpisce ancora**

Prof. Federico Dossena



# Parte 3

- Cosa vediamo?
  - Font personalizzati
  - Oggetti fluttuanti
  - Il box model

# Font personalizzati

- Tutti i browser moderni supportano il caricamento di font personalizzati nei formati standard **TTF**, **OTF**, **WOFF** e **WOFF2**
- Sul web si possono trovare gratuitamente e a pagamento font per tutti i gusti su siti come questi:
  - <https://www.1001fonts.com/>
  - <https://www.dafont.com/it/>
  - <https://www.fontspace.com/>
- L'importante è **scegliere font adatti al contesto**

# Font personalizzati

- Una volta scaricato il font, lo si può includere aggiungendo una definizione nel CSS, poi lo si può usare liberamente nelle regole
- Bisogna scegliere un nome per il font, non necessariamente lo stesso del file (usiamo qualcosa di comodo)

```
@font-face {  
    font-family: "Elegantissimo";  
    src: url("keukenhof.woff2");  
}
```

# Font personalizzati

- I font caricati in questo modo possono essere utilizzati normalmente nelle regole, con la proprietà font-family
- Esempio:

```
h1,h2,h3,h4,h5,h6{  
    font-family: "Elegantissimo";  
}
```

Ricordarsi le virgolette sui  
nomi dei font personalizzati

```
body{  
    font-family: "Roboto";  
}
```

# Font personalizzati

- È sempre bene specificare un font di default come alternativa qualora il font personalizzato non possa essere caricato
- Si fa specificando più font nella proprietà font-family, separati da una virgola

```
body{  
    font-family: "Roboto", sans-serif;  
}
```

# Font personalizzati

- Attenzione:
  - **Non tutti i font supportano tutti i caratteri** UTF-8, specialmente lettere accentate e caratteri stranieri
  - Non tutti i font includono varianti come grassetto, corsivo, lighter, ecc.
  - Se si tenta di visualizzare un carattere non supportato dal font, di solito viene visualizzato un quadrato o il font di default



# Font personalizzati

- Font più grandi sono spesso distribuiti con molte varianti (Regular per il testo normale, Bold per il grassetto, Italic per il corsivo, ecc.)
- Se carichiamo un font di questo tipo, dobbiamo caricare tutte le varianti che ci servono utilizzando lo stesso nome per il font ma specificando che versione abbiamo caricato

Name	Size
Roboto-Black.ttf	161.1 KiB
Roboto-BlackItalic.ttf	162.4 KiB
Roboto-Bold.ttf	159.6 KiB
Roboto-BoldCondensed.ttf	157.5 KiB
Roboto-BoldCondensedItalic.ttf	159.8 KiB
Roboto-BoldItalic.ttf	162.0 KiB
Roboto-Condensed.ttf	154.6 KiB
Roboto-CondensedItalic.ttf	158.0 KiB
Roboto-Italic.ttf	157.0 KiB
Roboto-Light.ttf	158.8 KiB
Roboto-LightItalic.ttf	159.4 KiB
Roboto-Medium.ttf	156.9 KiB
Roboto-MediumItalic.ttf	160.1 KiB
Roboto-Regular.ttf	154.9 KiB



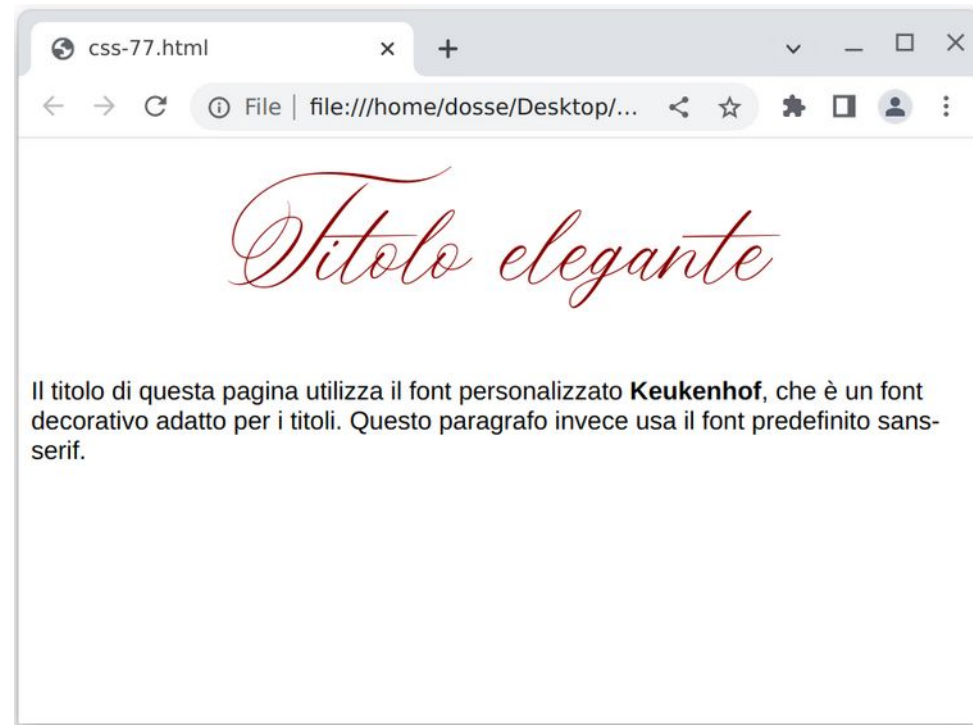
# Font personalizzati

```
@font-face {  
  font-family: "Roboto";  
  src: url("Roboto-Regular.ttf");  
  font-weight:normal;  
  font-style:normal;  
}  
@font-face {  
  font-family: "Roboto";  
  src: url("Roboto-Bold.woff2");  
  font-weight:bold;  
  font-style:normal;  
}  
@font-face {  
  font-family: "Roboto";  
  src: url("Roboto-Italic.woff2");  
  font-weight:normal;  
  font-style:italic;  
}  
...
```

Name	Size
Roboto-Black.ttf	161.1 KiB
Roboto-BlackItalic.ttf	162.4 KiB
Roboto-Bold.ttf	159.6 KiB
Roboto-BoldCondensed.ttf	157.5 KiB
Roboto-BoldCondensedItalic.ttf	159.8 KiB
Roboto-BoldItalic.ttf	162.0 KiB
Roboto-Condensed.ttf	154.6 KiB
Roboto-CondensedItalic.ttf	158.0 KiB
Roboto-Italic.ttf	157.0 KiB
Roboto-Light.ttf	158.8 KiB
Roboto-LightItalic.ttf	159.4 KiB
Roboto-Medium.ttf	156.9 KiB
Roboto-MediumItalic.ttf	160.1 KiB
Roboto-Regular.ttf	154.9 KiB

# Esempio

```
@font-face {  
  font-family: "Elegantissimo";  
  src: url("font/keukenhof.woff2");  
}  
h1, h2, h3, h4, h5, h6 {  
  font-family: "Elegantissimo", cursive;  
  font-weight: normal;  
  text-align: center;  
  color: darkred;  
}  
h1 {  
  font-size: 4rem;  
}  
body {  
  font-family: sans-serif;  
}
```



# Oggetti fluttuanti

- In CSS possiamo far sì che il testo „avvolga“ alcuni elementi, come nell'esempio quì sotto



# Oggetti fluttuanti

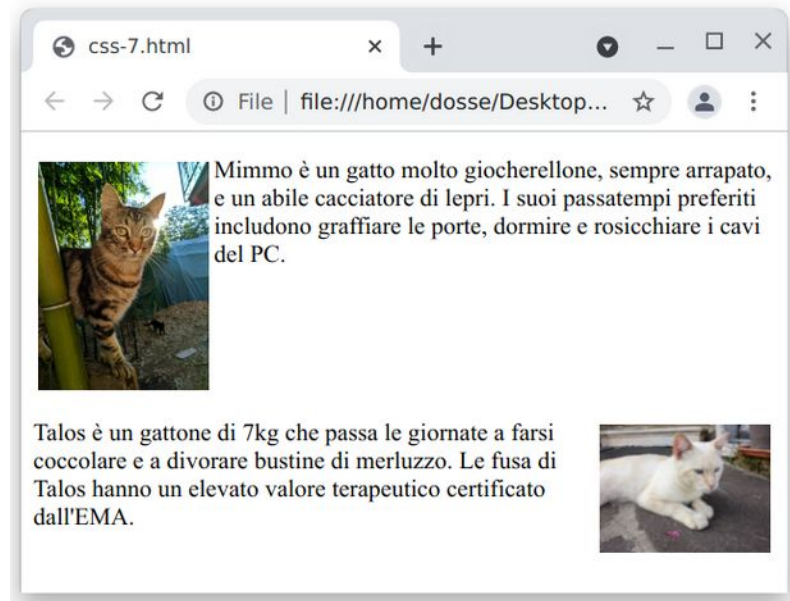
- La proprietà float permette di specificare se l'oggetto deve „fluttuare“ a destra (right) o a sinistra (left)

```
img.piccola{  
  width:7rem;  
  height:auto;  
}  
.fluttuaSX{  
  float:left;  
}  
.fluttuaDX{  
  float:right;  
}
```

```
<p>  
    
  Mimmo è un gatto molto giocherellone, sempre arrapato, e un abile  
  cacciatore di lepri. I suoi passatempi preferiti includono graffiare  
  le porte, dormire e rosicchiare i cavi del PC. Lorem ipsum dolor sit  
  amet, consectetur adipiscing elit. Vestibulum euismod mi ut dictum  
  aliquet. Class aptent taciti sociosqu ad litora torquent per conubia  
  nostra, per inceptos himenaeos. Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit. Vestibulum euismod mi ut dictum aliquet.  
  Class aptent taciti sociosqu ad litora torquent per conubia nostra,  
  per inceptos himenaeos.  
</p>
```

# Oggetti fluttuanti

- Se si inseriscono più oggetti „fluttuanti“ è necessario aggiungere delle barriere per evitare che vadano dove non dovrebbero, come nell'esempio a sinistra





# Oggetti fluttuanti

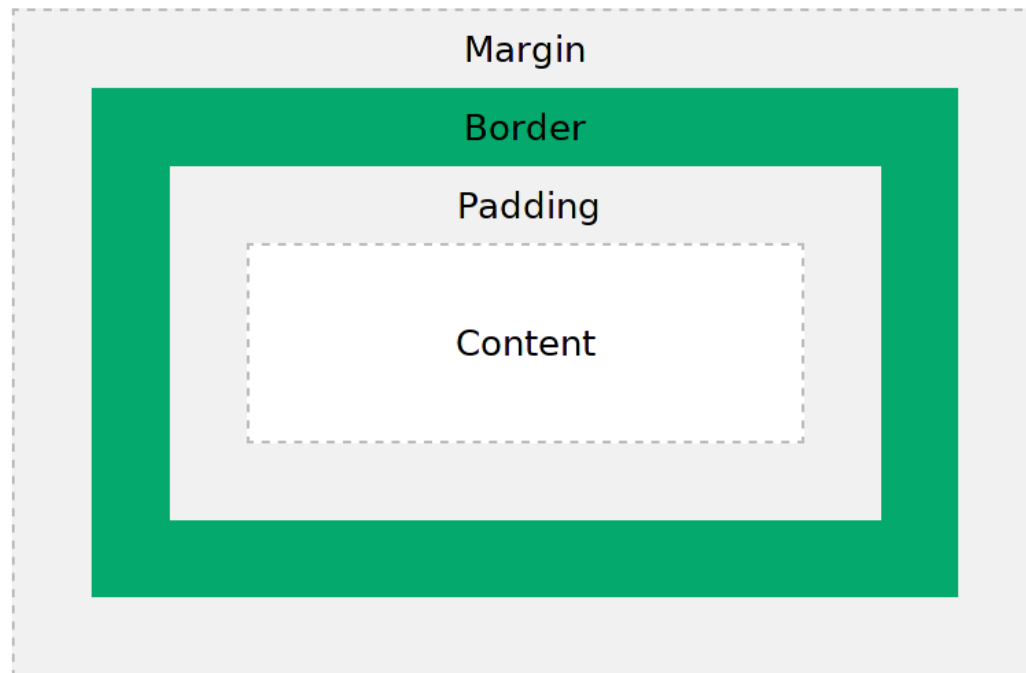
- Una barriera è semplicemente un elemento con `clear:both`;

```
img.piccola{  
  width:7rem;  
  height:auto;  
}  
.fluttuaSX{  
  float:left;  
}  
.fluttuaDX{  
  float:right;  
}  
.barrieraFloat{  
  clear:both;  
}
```

```
<p>  
    
  Mimmo è un gatto molto giocherellone, sempre arrapato, e un abile  
  cacciatore di lepri. I suoi passatempi preferiti includono graffiare  
  le porte, dormire e rosicchiare i cavi del PC.  
</p>  
<div class="barrieraFloat"></div>  
<p>  
    
  Talos è un gattone di 7kg che passa le giornate a farsi coccolare e a  
  divorare bustine di merluzzo. Le fusa di Talos hanno un elevato  
  valore terapeutico certificato dall'EMA.  
</p>
```

# Il box model

- In CSS, tutti gli elementi rispettano il cosiddetto **box model**, illustrato quì sotto
- **Content**: il contenuto dell'elemento (testo, immagini, ecc.). Le proprietà width e height fanno riferimento alla larghezza e altezza di questo
- **Padding**: spazio vuoto tra il contenuto e il bordo (gli si applica lo stesso sfondo del contenuto)
- **Border**: cornice, bordo
- **Margin**: spazio vuoto tra l'elemento e altri elementi

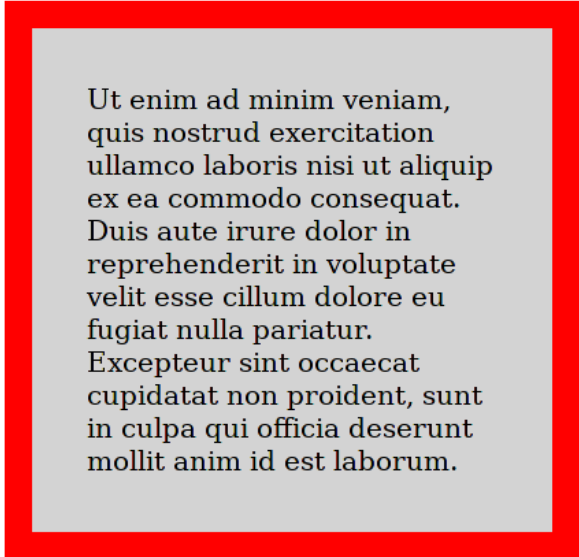


# Esempio

```
#esempio {  
  background-color: lightgrey;  
  width: 15em;  
  border: 1em solid red;  
  padding: 2em;  
  margin: 4em;  
}
```

## Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.



Ut enim ad minim veniam,  
quis nostrud exercitation  
ullamco laboris nisi ut aliquip  
ex ea commodo consequat.  
Duis aute irure dolor in  
reprehenderit in voluptate  
velit esse cillum dolore eu  
fugiat nulla pariatur.  
Excepteur sint occaecat  
cupidatat non proident, sunt  
in culpa qui officia deserunt  
mollit anim id est laborum.



# Il box model

- Il box model ci permette di calcolare le dimensioni degli elementi sulla pagina
- Ad esempio, per calcolare lo spazio orizzontale occupato dall'elemento #esempio devo sommare le varie parti:
  - 15em: larghezza contenuto
  - 2em (x2): padding sx e dx
  - 1em (x2): bordo sx e dx
  - 4em (x2): margine sx e dx
  - Totale: 29em

```
#esempio {  
  background-color: lightgrey;  
  width: 15em;  
  border: 1em solid red;  
  padding: 2em;  
  margin: 4em;  
}
```

# Il box model

- Non sempre si vuole impostare il margin, il padding o il border su tutti i 4 lati dell'elemento. È possibile impostarli singolarmente per ogni lato utilizzando proprietà come margin-top, margin-right, margin-bottom, margin-left, ecc.
- Ad esempio, questo codice toglie il margine da tutti i lati e poi lo imposta a 2rem sotto l'elemento

```
#introduzione{  
    margin:0;  
    margin-bottom:2rem;  
}
```

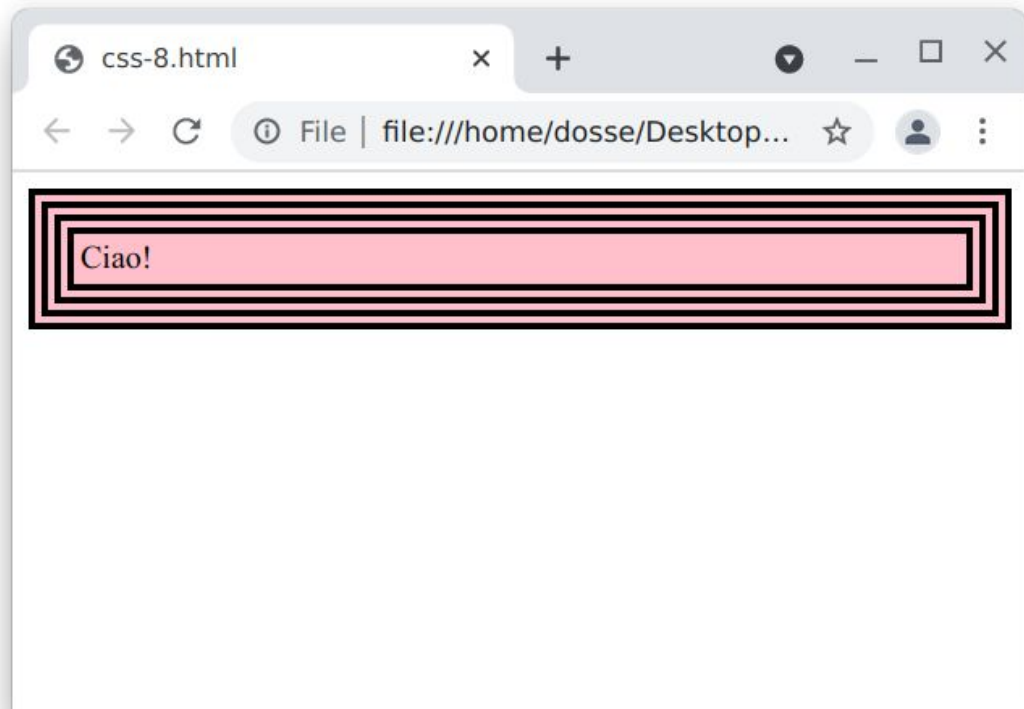
# Il box model

- Le proprietà padding, border e margin permettono di manipolare l'aspetto del „box“ dell'elemento
- A volte, è comodo aggiungere questa regola al proprio CSS:

```
*{  
    box-sizing: border-box;  
}
```
- Aggiungendola, width e height includono anche border e padding, anzichè solo il contenuto, e risolve molte situazioni in cui un oggetto „sfora“ il suo contenitore

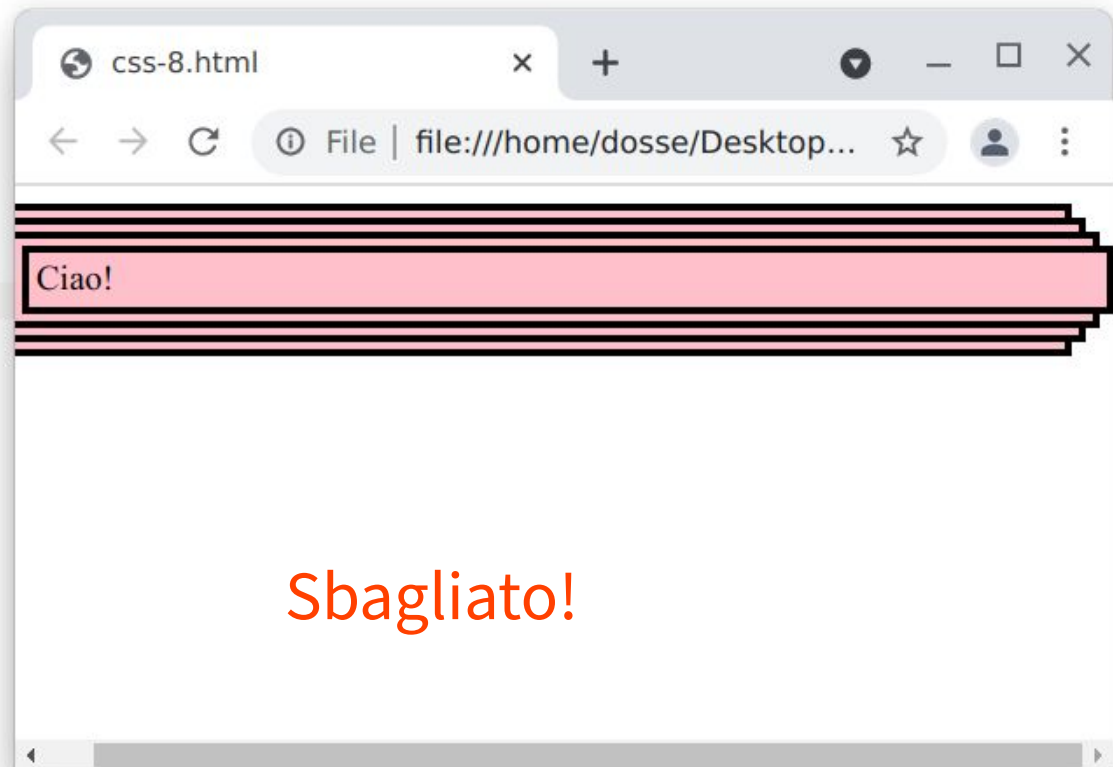
# Esempio

```
<html>
<head>
  <style>
    div.cornice{
      background:pink;
      width:100%;
      border:0.2em solid black;
      padding:0.2em;
    }
    *{
      box-sizing:border-box;
    }
  </style>
</head>
<body>
  <div class="cornice">
    <div class="cornice">
      <div class="cornice">
        <div class="cornice">
          Ciao!
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```



# Esempio

```
<html>
<head>
  <style>
    div.cornice{
      background:pink;
      width:100%;
      border:0.2em solid black;
      padding:0.2em;
    }
  </style>
</head>
<body>
  <div class="cornice">
    <div class="cornice">
      <div class="cornice">
        <div class="cornice">
          Ciao!
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```



Sbagliato!

# Il box model

- Difficile? Non sei da solo!
- Non capire il box model è **uno degli errori più comuni in CSS**
- Errori anche madornali sono visibili a un occhio esperto in molti siti popolari, inclusi Amazon, Facebook, e altri

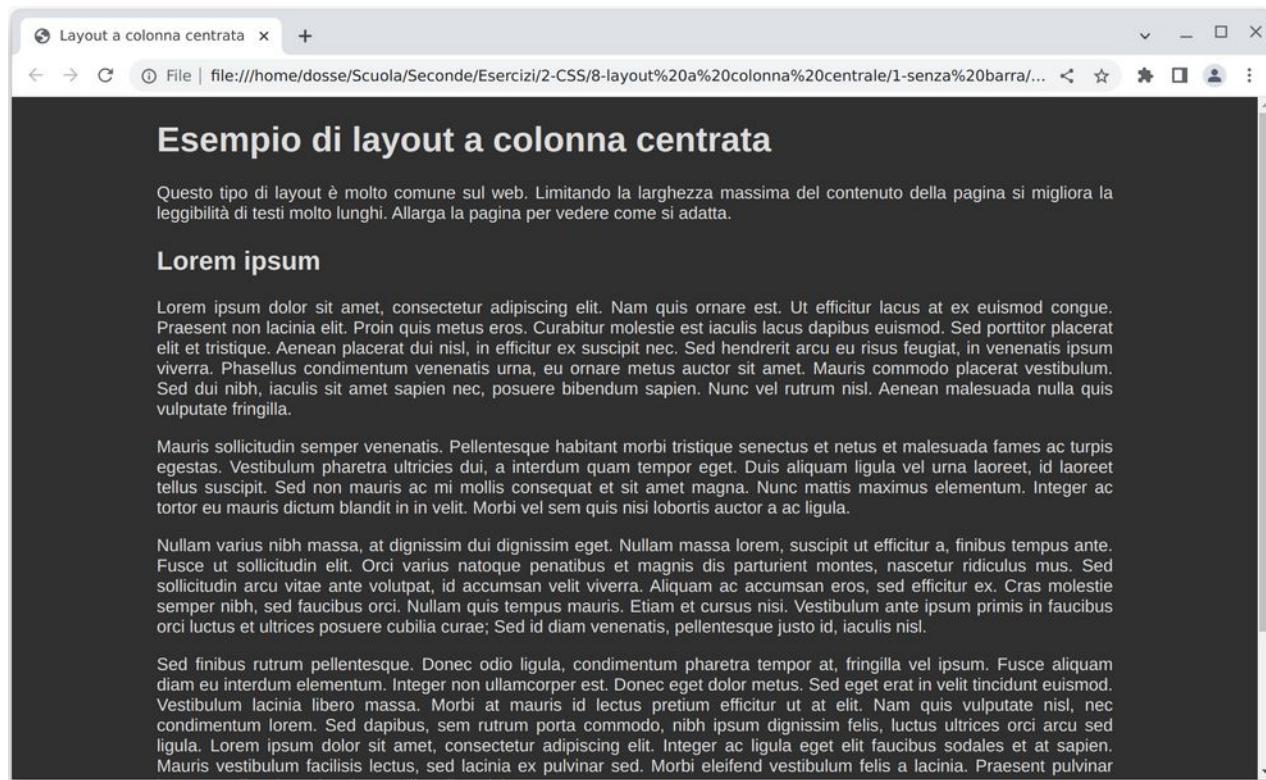
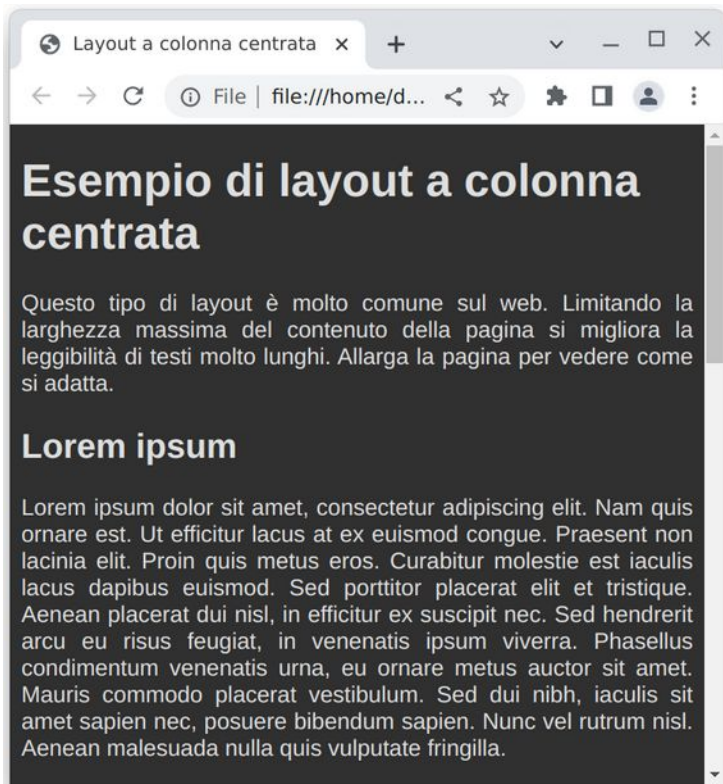
# Esempio

- Un layout molto comune sul web è quello in cui il contenuto della pagina è in una colonna centrale che si allarga fino a un limite massimo
- Si può fare facilmente inserendo tutto il contenuto della pagina in un div che fa da „contenitore“ e applicando una semplice regola

```
<body>
  <div id="principale">|
    <h1>Esempio di layout a colonna centrata</h1>
    <p>
      Questo tipo di layout è molto comune su
      massima del contenuto della pagina si n
      molto lunghi. Allarga la pagina per vec
    </p>
  </div>
</body>
```

```
#principale{
  max-width:55rem;
  margin:0 auto;
}
```

# Esempio





# To be continued

- Scarica il quarto pacco di slide per continuare