



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CSS e page layout

Fabio Vitali
Corsi di laurea in Informatica
Alma Mater – Università di Bologna



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Page layout

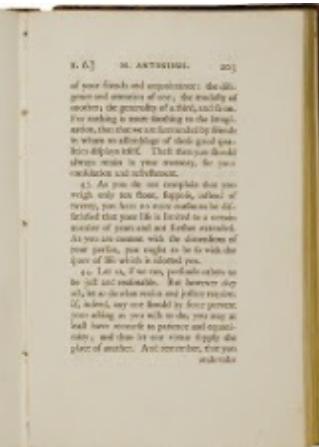
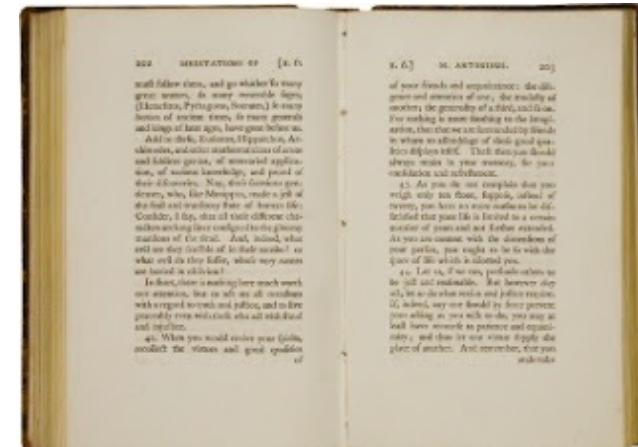
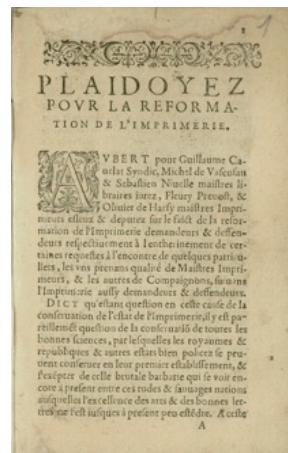
Graphic Design

- Il graphic design è l'arte e il mestiere di creare lo stile e la presentazione visuale di un testo o documento multimediale.
- Il graphic design si è evoluto come disciplina separata dall'attività di scrittura fin dagli anni dell'Impero Romano, divenne un'arte nel Medioevo con la creazione dei cosiddetti codici miniati, e una professione (il tipografo) nel rinascimento, con l'invenzione della stampa a caratteri mobili. Da allora vi sono state numerose epoche e mode e rivoluzioni sempre all'incrocio tra arte, industria e tecnologia.
- Ci sono tre separate arti nel graphic design
 - Tipografia (*typesetting*)
 - ***Organizzazione della pagina*** (*page layout*)
 - Organizzazione iconografica (*visual art curation*)

Page layout

- Page layout è la disposizione armoniosa degli elementi visuali sulla pagina.
- Riguarda l'organizzazione di questi elementi rispetto alle loro dimensioni, alla loro posizione e alla quantità e qualità delle aree vuote intorno ad essi.
- Quando il contenuto richiede più spazio di quello disponibile dalla dimensione della pagina, le pagine vengono moltiplicate e si crea paginazione.

Evoluzione del page layout



Page layout

La paginazione porta con sé alcuni problemi che in tipografia sono sempre citati con attenzione e storie dell'orrore:

- Il *fronte* e il *retro* delle pagine: molta carta non è completamente opaca e le parti nere del retro può trasparire di fronte come vaga area nera e creare uno sgradevole effetto visivo da evitare.
- Le *pagine affiancate* sono spesso guardate allo stesso momento, e possono fornire contrasti interessanti o sgradevoli o ridicoli.

Orrori di pagine affiancate

The image shows two side-by-side newspaper columns from 'The Cyprus Mail'. The left column is a serious news article titled 'SELF-HARMING', while the right column is a light-hearted, humorous page titled 'JUST FOR A LAUGH'.

Left Column: SELF-HARMING

Right Column: JUST FOR A LAUGH

Left Column Content (Self-harm):

Just for a laugh:

Right Column Content (Humor):

Orrori di pagine affiancate



Orrori di pagine affiancate

The Chronicle Herald
Wednesday, March 5, 2014

A9
NEWS

Data sees N.S. at forefront of obesity trend

Study finds need for national strategy to tackle problem

PAUL MCLEOD
OTTAWA BUREAU
pmcleod@herald.ca
[@CH_PaulMcLeod](#)

More Nova Scotia adults are overweight than not and the problem is getting worse, says a new Memorial University of Newfoundland study.

It found that obesity has tripled across Canada from 1985 to 2010 and about one in five Canadian adults are now obese.

But while the numbers are startling across the country, Atlantic Canada fares particularly poorly.

Nova Scotia men are among the largest in the country, but women in the province are rapidly catching up.

The number of overweight women in Nova Scotia over 15 per cent from 2000 to 2009, the study

says do not have to exceed what percentage of the recommended daily intake of sugar they contain, despite sugar being a major contributor to obesity.

Ogilvie is chairman of the Senate's social affairs, science and technology committee, which will be launching a study into obesity this year, with the hopes of having a final report early next year. Food labelling will be one factor it will study.

NDP Leader Thomas Mulcair cast blame on the federal government Tuesday in the House of Commons. He told reporters Prime Minister Stephen Harper has been consulting with down with the provinces and work out a co-ordinated plan.

The Public Health Agency of Canada says it is already working with the provinces and has launched several campaigns to promote Canadians' physical

Mmmmm, pancakes

Photo by TIM KROCHAK • Staff



Volunteer server Sotya Beeler delivers some of the several hundred pancakes and message towels at the annual Shrove Tuesday all-you-can-eat supper at St. Mark's Community Centre in Halifax.

Aspetti del page layout

- Orientamento
- Aspect Ratio
- Dimensioni
- Risoluzione
- Griglie di layout
- La sezione aurea

Orientamento

- Le aree rettangolari hanno due orientamenti naturali:
 - portrait (la dimensione lunga è l'altezza)
 - landscape (la dimensione lunga è la larghezza)
- La tradizione editoriale di libri è stata per lo più orientata al portrait, ma cinema (e TV) sono sempre stati landscape.
- I computer fino ai laptop usavano il layout degli schermi TV e quindi per lo più landscape.
- Gli schermi mobili sono bidirezionali, e il portrait è l'orientamento più frequente negli smartphone (91%) mentre i tablet passano a portrait oltre una certa dimensione (14% a 7", 48% a 8", 82% a 11").



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

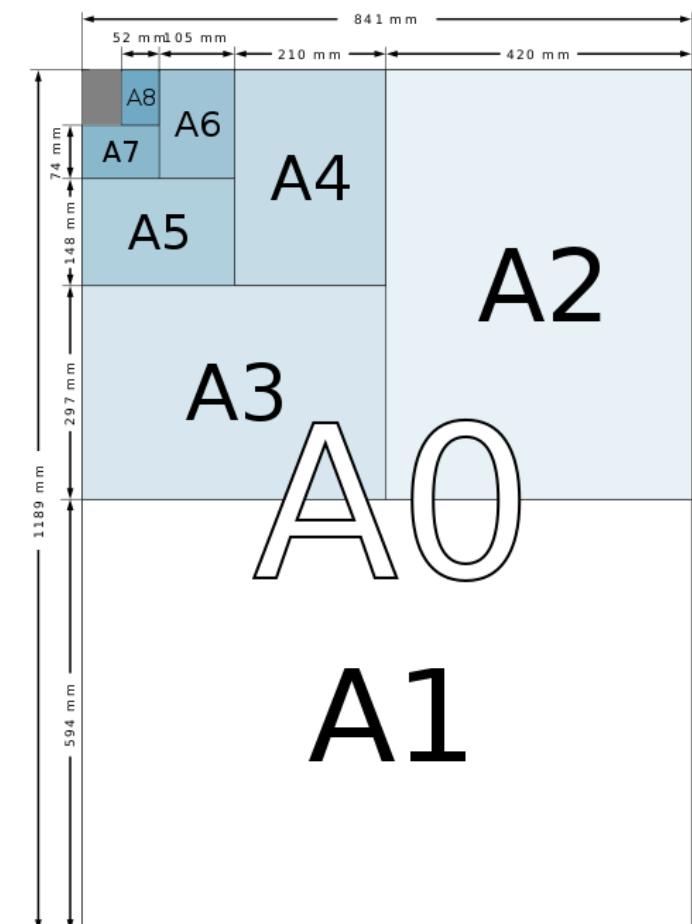
Aspect ratio

- Il rapporto tra altezza e larghezza della pagine/schermo.
- Indipendente dalla dimensione e dall'orientamento usato.
- Aspect ratio comuni:
 - [Quadrato](#) (1.0000) – Non molto usato ma è un punto di partenza
 - [US Letter](#) (1.2941) – Carta da lettere americana
 - [4/3](#) (1.3333) – Trasmissioni televisive analogiche (fino al 2002)
 - [11/8](#) (1.3750) – i film di Hollywood (Academy ratio 35mm)
 - [ISO 216](#) (v2 or 1.4142) – Carta europea (A0, A1, A2, A3, **A4**, A5)
 - [16/10](#) (1.6000) – La maggior parte degli schermi di computer moderni
 - [Regola aurea](#) (1.6180) – Nel mondo classico il rapporto ideale
 - [US Legal](#) (1.6470) – Carta dei documenti legali americani
 - [16/9](#) (1.7777) – Schermi TV digitali e molte generazioni di smartphone
 - [18/9](#) (2.0000) – Nuovi smartphone Android
 - [19.5/9](#) (2.1666) – Nuovi smartphone iPhone
 - [70mm](#) (2.2000) – Film di Hollywood ad alta risoluzione (wide screen)



Dimensioni

- ISO 216 (1975) stabilisce lo standard internazionale sulla dimensione della carta da usare in stampa con le serie A, B e C.
- Europa e Asia si sono allineati a questo standard, mentre gli USA e il resto dell'America sostanzialmente no.
- L'idea di base dei modelli ISO 216 è di piegare e tagliare la carta a metà del lato lungo, ottenendo due fogli con lo stesso aspect ratio del foglio di partenza.
- Questo è possibile solo con un aspect ratio di $\sqrt{2}$ (1.4142).
- La dimensione A0 è quella di partenza, con area complessiva di 1m².
- Le dimensioni B e C sono intermedie (B0 è a metà tra A0 e A1, mentre C0 è a metà tra B0 e B1. Sono molto meno usate.



Risoluzioni

- In teoria, *la risoluzione* è la densità degli elementi visuali in una unità di area. La risoluzione non è appropriata nella stampa analogica ma è un fattore importante di valutazione nella stampa digitale. Si parla di *dots per inches* (DPI) in stampa, e *pixel per inches* (PPI) negli schermi.
- Con risoluzione negli schermi ci si riferisce erroneamente al numero **totale** di pixel invece che alla densità. Per ovviare a questo è stato introdotto il termine pixel density. I due valori sono interdipendenti una volta stabilita la dimensione dello schermo.
- Gli schermi digitali hanno tipicamente una densità minore delle stampanti, per cui la visualizzazione su schermo è più sfocata e meno dettagliata della stampa.

Stampa	Risoluzione
Stampante dot matrix	60-90 DPI
Stampante inkjet	300-720 DPI
Stampante laser	600-2400 DPI
Typesetter professionale	1200-2800 DPI

Schermo	Risoluzione
Apple standard resolution	72 PPI
Windows standard resolution	96 PPI
High density TV screens	213-240 PPI
Retina display (Mac & iPhones)	220-458 PPI
High end smartphones	534-807 PPI

Griglie

- La griglia è la struttura bidimensionale usata nel graphic design per allineare gli elementi e i componenti della pagina in modo gradevole.
- Le griglie vengono usate per raggruppare, allineare, contrastare e dare rilevanza visuale agli elementi della pagina. I componenti possono occupare uno, due o più celle della griglia, sia orizzontalmente sia verticalmente.
- Una griglia è densa se non c'è spazio tra le celle (cioè: margini assenti nelle celle). Una griglia è sparsa se intere celle vengono lasciate vuote per dare enfasi e *breathing space* alle celle piene.
- Le griglie possono dipendere o contrastare in maniera creativa le dimensioni complessive della pagina/schermo.
- Twitter Bootstrap usa una griglia di 12 colonne orizzontali (non ci sono vincoli verticali) perché 12 può essere diviso da 1, 2, 3, 4, 6, e 12 e questo dà molta flessibilità.



John P. Corrigan

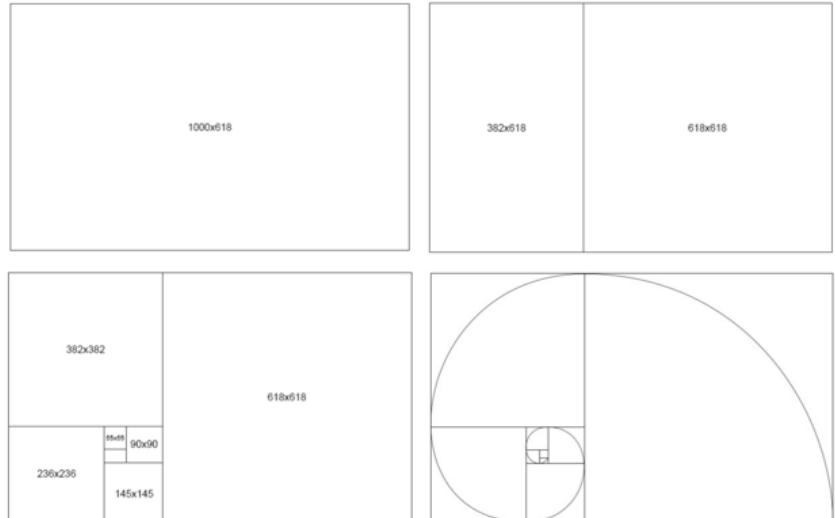
La sezione aurea

Il rapporto aureo, o proporzione divina, o costante di Fidia, o sezione aurea (*golden ratio*) è il rapporto tra due numeri tale per cui il più grande è il medio proporzionale tra il più piccolo e la somma dei due:

- $\phi \stackrel{\text{def}}{=} (a+b) : a = a : b$
- $\phi = 1.6180339887\dots$
- $\phi = \frac{1+\sqrt{5}}{2}$
- ϕ è la soluzione positiva di $\varphi^2 - \varphi - 1 = 0$



© Ancient-Greece.org



Nelle arti grafiche, la sezione aurea è stata considerata fin da egizi e greci, come un rapporto particolarmente piacevole all'occhio. Il rapporto tra base e altezza delle piramidi, dei templi greci, nella disposizione degli elementi in un quadro rinascimentale segue spesso i rapporti progressivamente vincolati della sezione aurea.

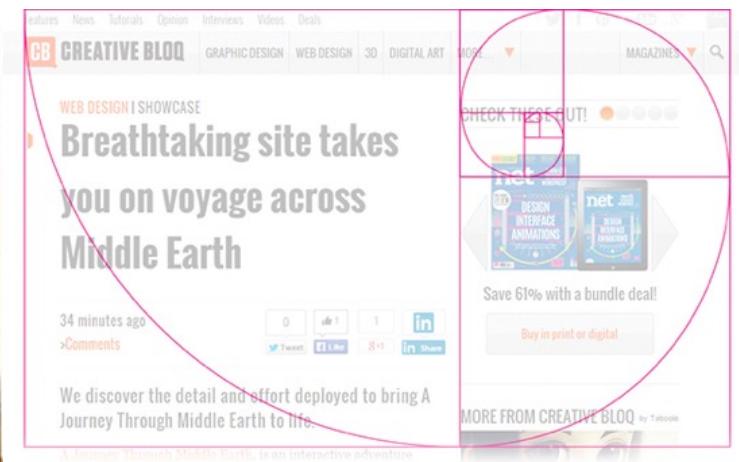
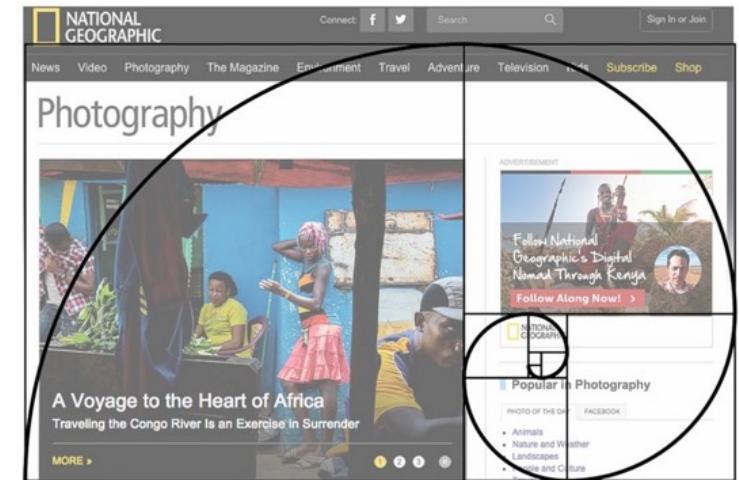
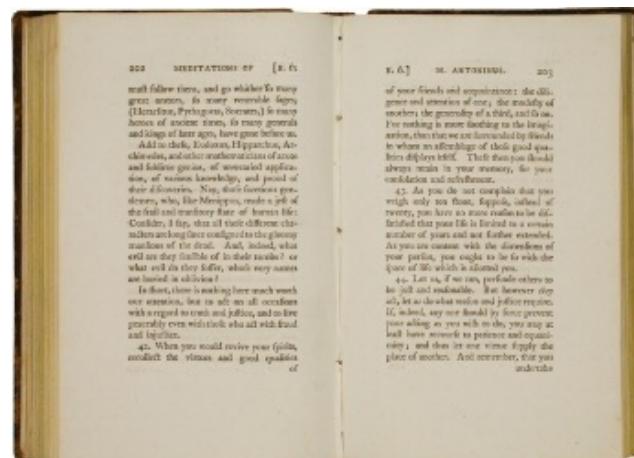
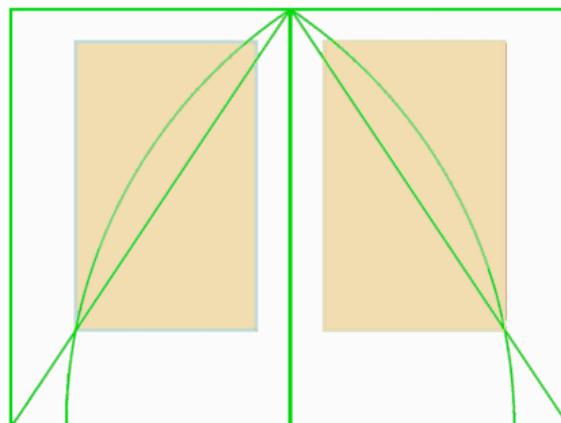


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

La sezione aurea in tipografia

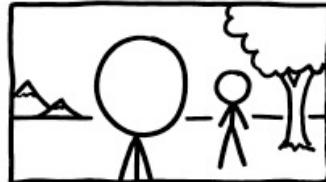
Anche nel page design, fin dal 1600, si considera la sezione aurea come una guida nello stabilire le proporzioni tra altezza e larghezza delle sezioni di una pagina, sia per libri che riviste che pagine web.

Fin dal 1600 i tipografi si basano sulla sezione aurea per identificare le aree grigie di una pagina a stampa. Jan Tschichold (1925) ha proposto l'uso consapevole della sezione aurea nel design. Ci sono numerosi tool online per verificare che gli elementi di una pagina siano organizzati secondo modelli basati sulla sezione aurea.



VIDEO ORIENTATION

HORIZONTAL



VERTICAL



DIAGONAL



PROS

- LOOKS NORMAL TO OLD PEOPLE
- FORMAT USED BY A CENTURY OF CINEMA

CONS

- HUMANS ARE TALLER THAN THEY ARE WIDE
- I'M NOT TURNING MY PHONE SIDEWAYS

- HOW MOST NORMAL PEOPLE SHOOT AND WATCH VIDEO NOW SO WE MAY AS WELL ACCEPT IT

- HUMAN WORLD IS MOSTLY A HORIZONTAL PLANE

- BOLD AND DYNAMIC
- EQUALLY ANNOYING TO ALL VIEWERS
- GOOD COMPROMISE

- NONE



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Cascading Style Sheet II parte

Selettori e regole

Un *selettore* permette di specificare un elemento o una classe di elementi dell'albero HTML (o XML) al fine di associarvi caratteristiche CSS

- Esempi: `h1, #p1, .codice, p.codice, img[alt]`

Una *regola* è un blocco di statement associati ad un elemento attraverso l'uso di un selettore

- Sintassi

```
selettore {  
    statement; statement; ...  
}
```

- Esempio

```
h1 {  
    color: white;  
    background-color: black;  
}
```

Esistono moltissimi selettori diversi. Li usiamo in CSS, ma anche con il DOM e in tante altre occasioni. Attenti alla parola magica "selettore CSS".



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

```

p {
  color: blue;
}

.codice {
  margin: 5mm;
}

#abc1 {
  width: 75%;
}

```

Selettori (1/6): universale, tipo, classe e id

pattern	significato	esempio
*	qualsiasi elemento	*
E	un elemento di tipo <i>E</i>	h1
E.Nomeclasse .nomeclasse	un elemento (di tipo <i>E</i> e) di classe <i>nomeclasse</i>	p.codice .codice
E#ilmiod #ilmiod	un elemento (di tipo <i>E</i> e) con id <i>ilmiod</i>	tr#abc1 #abc1

```

p {
  color: blue;
}

.codice {
  margin: 5mm;
}

#abc1 {
  width: 75%;
}

```

pattern		
*		<pre> <body> <p>Io sono in blu</p> <p class="codice"> Io sono in blu e ho un piccolo margine tutto intorno. </p> <p id="abc1"> Io sono in blu e non occupo tutta la larghezza dello schermo. </p> </body></pre>
E	un elemento di tipo <i>E</i>	h1
E.nomeclasse .nomeclasse	un elemento (di tipo <i>E</i> e) di classe <i>nomeclasse</i>	p.codice .codice
E#ilmiod #ilmiod	un elemento (di tipo <i>E</i> e) con id <i>ilmiod</i>	tr#abc1 #abc1

Selettori (1/6): universale, tipo, classe e id

```

p::first-line {
    color: red;
}

p::first-letter {
    font-size: 300%;
}

q { color: blue; }

q::before {
    content: "«";
}

q::after {
    content: "»";
}

```

Selettori (2/6)

pseudo-elementi

pattern	significato	esempio
E::first-line	la prima riga formattata dell'elemento <i>E</i>	p::first-line
E::first-letter	la prima lettera formattata dell'elemento <i>E</i>	p::first-letter
E::before	contenuto generato prima dell'elemento <i>E</i>	q::before
E::after	contenuto generato dopo l'elemento <i>E</i>	q::after

```
p::first-line {  
    color: red;  
}  
  
p::first-letter {  
    font-size: 300%;  
}  
  
q { color: blue; }  
  
q::before {  
    content: "«";  
}  
  
q::after {  
    content: "»";  
}
```

pattern

E::first-line

E::first-

E::be

E::a

Selettori (2/6) pseudo-elementi

```
<body>  
    <p>Lorem ipsum dolor sit amet,  
    consectetur adipiscing elit. Maecenas  
    rutrum orci eu lorem pulvinar  
    fringilla. In <q>egestas accumsan  
    massa</q>, elementum varius sapien  
    euismod nec. Vivamus ut ullamcorper  
    mi. Sed vehicula vel est sit amet  
    euismod. Ut <q>molestie metus  
    turpis</q>, eu dapibus lacus auctor  
    at.</p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum orci eu lorem pulvinar fringilla. In «egestas accumsan massa», elementum varius sapien euismod nec. Vivamus ut ullamcorper mi. Sed vehicula vel est sit amet euismod. Ut «molestie metus turpis», eu dapibus lacus auctor at.

```

.special span {
  color: blue;
}

.special th {
  font-style: italic;
}

.special + tr {
  background-color: #ffffdd;
}

.special ~ tr {
  color: green;
}

```

Selettori (3/6) prossimità

pattern	significato	esempio
E F	elemento <i>F</i> discendente di un elemento <i>E</i>	table th
E > F	elemento <i>F</i> figlio di un elemento <i>E</i>	tr > th
E + F	elemento <i>F</i> successivo diretto di un elemento <i>E</i>	label + input
E ~ F	elemento <i>F</i> successivo di un elemento <i>E</i>	h1 ~ p

```

.special span {
  color: blue;
}

.special th {
  font-style: italic;
}

.special + tr {
  background-color: yellow;
}

.special ~ tr {
  color: green;
}

```

Selettori (3/6)

Salesman	Jan	Feb	Mar
John Smith THE BEST!	12000	13000	15000
Alice Green the second best!	7000	9000	11000
Hugh Brown	5000	7000	9000

pattern
E F

```

<table border="1" cellpadding="8" cellspacing="0">
<tr>
  <th>Salesman</th><th>Jan</th><th>Feb</th><th>Mar</th>
<tr>
<tr class="special">
  <th>John Smith <span>THE BEST!</span></th>
  <td>12000</td><td>13000</td><td>15000</td>
</tr>
<tr>
  <th>Alice Green <span>the second best!</span></th>
  <td>7000</td><td>9000</td><td>11000</td>
</tr>
<tr>
  <th>Hugh Brown</th>
  <td>5000</td><td>7000</td><td>9000</td>
</tr>
</table>

```

Salesman	Jan	Feb	Mar
John Smith THE BEST!	12000	13000	15000
Alice Green the second best!	7000	9000	11000
Hugh Brown	5000	7000	9000

tr > th

label + input

h1 ~ p

```

a[href] {
  font-style: italic;
}
a[href^='http'] {
  color: red;
}

```

Selettori (4/6)

attributi

pattern	significato	esempio
E[foo]	un elemento <i>E</i> con un attributo <i>foo</i>	img[alt]
E[foo="bar"]	un elemento <i>E</i> con un attributo <i>foo</i> con valore uguale a "bar"	table[border="1"]
E[foo~="bar"]	un elemento <i>E</i> con un attributo <i>foo</i> il che contiene la parola bar	p[class~="codice"]
E[foo^="bar"]	un elemento <i>E</i> con un attributo <i>foo</i> con valore che inizia per "bar"	p[class^="cod"]

```
a[href] {  
    font-style: italic;  
}  
  
a[href^='http'] {  
    color: red;  
}
```

Selettori (4/6)

```
<style>  
    a[href] { color: green; }  
    a[href^="http"] { color: red; }  
    a[href^="http"]::before {  
        content: "🌐 ";  
        font-size: 60%;  
    }  
</style>
```

pattern

[Lorem ipsum](lorem ipsum.html) dolor sit amet, consectetur adipiscing elit. Maecenas rutrum orci eu lorem pulvinar fringilla.

E[foo]

In [egestas accumsan](https://www.lorem ipsum.com) massa, elementum varius sapien euismod nec. Vivamus ut ullamcorper mi. Sed vehicula vel est sit amet euismod. Ut molestie metus turpis, eu dapibus lacus auctor at.

E[foo="bar"]

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum orci eu lorem pulvinar fringilla. In 🌐 egestas accumsan massa, elementum varius sapien euismod nec. Vivamus ut ullamcorper mi. Sed vehicula vel est sit amet euismod. Ut “molestie metus turpis”, eu dapibus lacus auctor at.

ce"]

d"]

```

tr {
  background-color: #ffffee;
}
tr:nth-child(odd) {
  background-color: #dddddd;
}
tr:first-child {
  color: blue;
  background-color: #ffffff;
}

```

Selettori (5/6)

pseudo-classi strutturali

pattern	significato	esempio
E:nth-child(n)	un elemento <i>E</i> che è l' <i>n</i> -simo figlio di suo padre	p:nth-child(odd)
E:nth-last-child(n)	un elemento <i>E</i> che è l' <i>n</i> -simo figlio di suo padre a partire dall'ultimo	p:nth-last-child(1)
E:nth-of-type(n)	un elemento <i>E</i> che è l' <i>n</i> -simo figlio di suo padre di quel tipo	p:nth-of-type(even)
E:first-child	un elemento <i>E</i> che è il primo figlio di suo padre	h1:first-child

```

tr {
  background-color: #ffffee;
}

tr:nth-child(odd) {
  background-color: #e0e0e0;
}

tr:first-child {
  color: blue;
  background-color: #d9eaf7;
}

```

Selettori (5/6)

Salesman	Jan	Feb	Mar
John Smith THE BEST!	12000	13000	15000
Alice Green the second best!	7000	9000	11000
Hugh Brown	5000	7000	9000

pattern

E:nth-child(n)

E:nth-last-child(n)

E:nth-of-type(n)

E:first-child

Salesman	Jan	Feb	Mar
John Smith THE BEST!	12000	13000	15000
Alice Green the second best!	7000	9000	11000
Hugh Brown	5000	7000	9000

```

a:active {
    color:yellow;
}

a:hover {
    cursor:pointer;
}

input:checked {
    border:3px solid;
}

```

Selettori (6/6)

pseudo-classi

pattern	significato	esempio
E:active	un elemento <i>E</i> è attivato dall'utente (es: click)	a:active
E:hover	un elemento <i>E</i> ha il puntatore sopra	a:hover
E:enabled	un elemento <i>E</i> di interfaccia se abilitato	input
E:checked	un elemento <i>E</i> di interfaccia è "checked"	input:checked

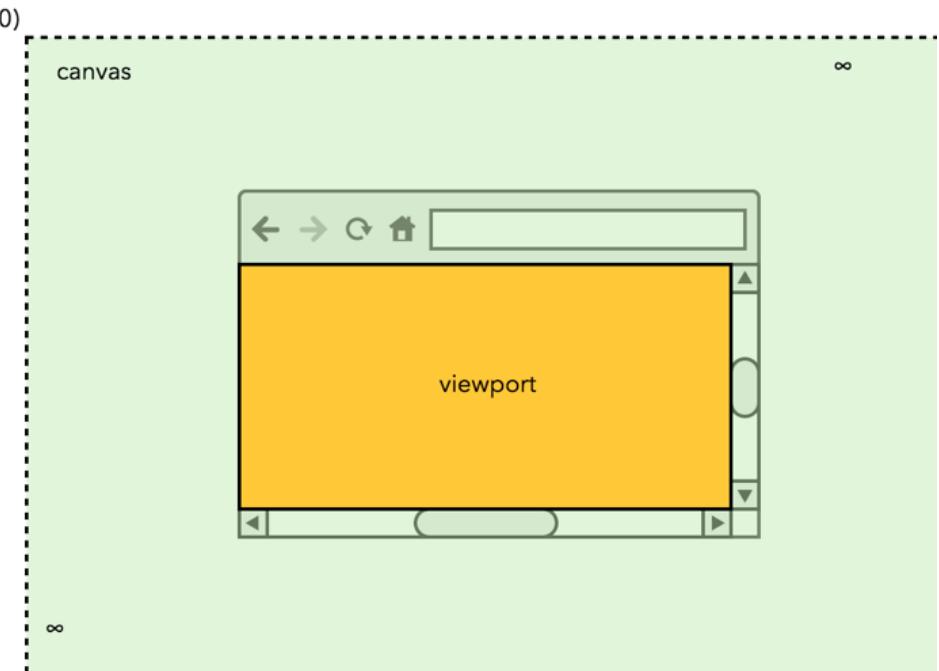


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Proprietà

Canvas e viewport

- Il canvas è l'area virtuale di posizionamento degli elementi del DOM via CSS. E' un piano cartesiano infinito in larghezza e altezza, con l'asse delle x da sinistra a destra e l'asse delle y dall'alto in basso.
- Il viewport è la parte del canvas che è attualmente visibile attraverso lo schermo o la finestra e può muoversi sul canvas in seguito a scrolling.
- Il viewport dipende ovviamente dalla dimensione dello schermo e si muove solo su coordinate positive del canvas.
- Per disegnare fuori schermo basta usare coordinate negative (off-screen drawing) .
- HTML fornisce un elemento <canvas> per disegnare sul canvas indipendentemente dal contenuto testuale del resto del documento HTML.



Unità di misura basate su canvas e viewport (1)

- Pixel (px) è l'unità principale di disegno sul canvas. Il pixel è la più piccola unità indirizzabile sullo schermo. Le dimensioni dei pixel dipendono fortemente dalla dimensione e dalla risoluzione degli schermi e sono molto diversi tra device e device e non sono un'unità affidabile. HTML però usa solo pixel.
- Le stampanti ri-scalano la dimensione dei pixel secondo rapporti che dipendono dalla risoluzione della stampante, con risultati non sempre convincenti.
- I device mobili, a loro volta, si basano sul tag `<meta name="viewport" content="width=XXX, initial-scale=YYY">` per scalare le dimensioni in pixel in maniera coerente, ma è sempre un'operazione rischiosa e poco affidabile.

Suggerimento: usare **sempre** `initial-scale=1` e non usare MAI misure basate su pixel nei propri CSS.

Eccezioni: `0px` e `1px`

Unità di misura basate su canvas e viewport (2)

- *Viewport width (vw)*: una percentuale (1%) della larghezza attuale del viewport. In a viewport that is 500 pixel wide, 3vw = 15px.
- *Viewport height (vh)*: una percentuale (1%) della altezza attuale del viewport.
- *vmin*: il più piccolo tra vw e vh
- *vmax*: il più grande tra vw e vh

Suggerimento: usate unità di viewport per creare layout responsive che si adattano alla dimensione dello schermo o della finestra. Usate vmin e vmax per adattare il layout sia all'uso in modalità portrait sia all'uso in modalità landscape.

L'unità flex (fr)

- Una lunghezza flessibile (flex) è una dimensione con unità fr che rappresenta una frazione dello spazio rimasto nel contenitore.
- Una volta escluse dallo spazio del contenitore le componenti con dimensioni non flessibili, si sommano le unità fr, si divide lo spazio rimasto rispetto a tale somma e si distribuisce lo spazio rimasto alle varie componenti in proporzione alla loro frazione.
- Molto comodo per organizzare rapporti complessi senza dover esplicare le operazioni corrispondenti.

```
.mygrid {  
    display: grid;  
    grid-template-columns: 20% 2fr 1fr;  
}
```

Posizione della scatola

La posizione dipende dalle altre scatole, dallo sfondo (canvas) o dalla finestra (viewport).

- *Posizionamento statico* (default): la scatola viene posta nella posizione di flusso normale che avrebbe naturalmente.
- *Posizionamento assoluto*: la scatola viene posta nella posizione specificata indipendentemente dal flusso (nascondendo ciò che sta sotto).
- *Posizionamento relativo*: la scatola viene spostato di un delta dalla sua posizione naturale
- *Posizionamento fisso*: la scatola viene posta in una posizione assoluta rispetto alla finestra (*viewport*), senza scrollare mai
- *Posizionamento sticky*: la scatola viene posta nella sua posizione naturale all'interno del suo contenitore, ma durante lo scrolling sta fissa rispetto al *viewport* fino a che il contenitore non esce dalla vista.

Ovunque si lavori esplicitamente sulla posizione, si possono indicare fino a quattro proprietà tra *top*, *left*, *right*, *bottom*, *width*, *height*



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

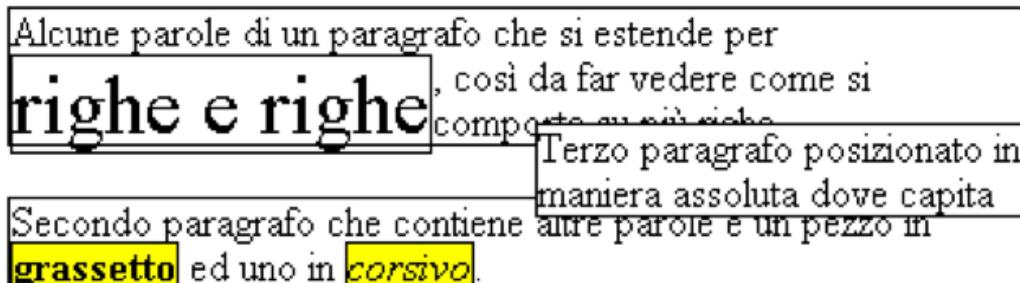
Posizionamento: position, float, coordinate e dimensioni

- **position** (static | relative | absolute | fixed) gestisce il posizionamento rispetto al flusso del documento
- **float** (left | right | none) è una scatola scivolata all'estrema destra o sinistra del contenitore muovendo le altre per farle posto
- **top, bottom, left, right**: coordinate della scatola
- **width, height**: dimensioni usabili invece di *right* e *bottom*

Esempio di posizionamento

```
<p>Alcune parole di un paragrafo  
che si estende per <span  
class="left">righe e righe</span>,  
così da far vedere come si comporta  
su più righe.</p>  
<p>Secondo paragrafo che contiene  
altre parole e un pezzo in  
grassetto ed uno in  
corsivo.</p>  
<p class="abs">Terzo paragrafo  
posizionato in maniera assoluta  
dove capita </p>
```

```
p.abs {  
    position: absolute;  
    top: 40px;  
    left: 210px;  
    width: 190px;  
}  
span.left {  
    float:left;  
    font-size: 200%;  
}
```



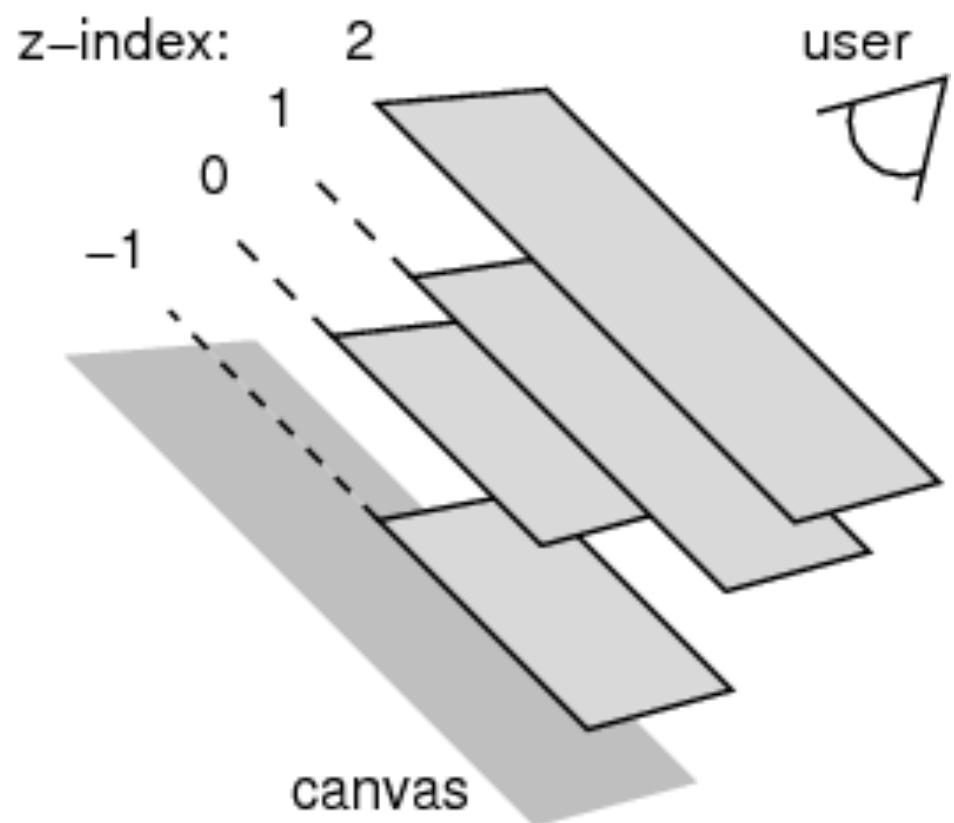
```
p, b, i,span {  
    border-style: solid;  
    border-width: 1px;  
}  
p.abs {  
    background-color: white;  
}  
b, i {  
    background-color: yellow;  
}
```

Posizionamento: z-index

z-index è la posizione nella pila di scatole potenzialmente sovrapposte.

Il valore più alto è più vicino al lettore, e quindi nasconde gli altri.

N.B.: per default il background delle scatole è trasparente.



Posizionamento: overflow

overflow specifica come trattare il contenuto che non sta interamente nella scatola (forse con dimensioni troppo rigide). Valori possibili:

- **visible**: la scatola si espande per contenere tutto il contenuto
- **hidden**: il contenuto extra viene nascosto
- **scroll**: compare una scrollbar per l'accesso al contenuto extra.
- Si possono specificare le proprietà `overflow-x` and `overflow-y` per permettere la comparsa di una sola delle scrollbar.

Visible

Overflow visible example. The container expands to fit its content.

```
overflow: visible;
```

Content: Lorem ipsum dolor amet next level banh mi actually etsy craft beer. Portland meh palo santo pitchfork wayfarers raclette kinfolk try-hard YOLO. Lo-fi cred pork belly, cloud bread artisan heirloom raw denim kombucha. Godard etsy ugh, letterpress roof party fingerstache succulents edison bulb. Iceland disrupt palo santo fixie hella taiyaki celiac green juice

Hidden

Overflow hidden example. Content is truncated.

```
overflow: hidden;
```

Content: Lorem ipsum dolor amet next level banh mi actually etsy craft beer. Portland meh palo santo pitchfork wayfarers raclette kinfolk try-hard YOLO. Lo-fi cred pork belly, cloud bread artisan heirloom raw denim kombucha. Godard etsy ugh, letterpress roof party fingerstache succulents edison bulb. Iceland disrupt palo santo fixie hella taiyaki celiac green juice

Scroll

Overflow scroll example. A scrollbar appears.

```
overflow: scroll;
```

Content: Lorem ipsum dolor amet next level banh mi actually etsy craft beer. Portland meh palo santo pitchfork wayfarers raclette kinfolk try-hard YOLO. Lo-fi cred pork belly, cloud bread artisan heirloom raw denim kombucha. Godard etsy ugh, letterpress roof party fingerstache succulents edison bulb. Iceland disrupt palo santo fixie hella taiyaki celiac green juice

`overflow-x:hidden;`
`overflow-y:scroll;`

Overflow scroll example with horizontal and vertical scrollbars.

```
overflow-x: hidden;  
overflow-y: scroll;
```

Content: Lorem ipsum dolor amet next level banh mi actually etsy craft beer. Portland meh palo santo pitchfork wayfarers raclette kinfolk try-hard YOLO. Lo-fi cred pork belly, cloud bread artisan heirloom raw denim kombucha. Godard etsy ugh, letterpress roof party fingerstache succulents edison bulb. Iceland disrupt palo santo fixie hella taiyaki celiac green juice

Colonne di testo

In CSS3 è stata introdotta la possibilità di gestire colonne multiple
Il contenuto prosegue naturalmente (senza l'uso di tavelle) da una colonna all'altra e il numero delle colonne può variare automaticamente a seconda della dimensione della viewport

```
Ab cde fgh i jkl. Mno
pqr stu vw xyz. A bc
def g hij klm nopqrs
tuv wxy z. Abc de fg
hi jklmno. Pqrstu vw
x yz. Abc def ghi jkl.
M nop qrst uv wx yz.
Ab cde fgh i jkl. Mno
pqr stu vw xyz. A bc
def g hij klm nopqrs
tuv wxy z. Abc de fg
hi jklmno. Pqrstu vw
x yz. Abc def ghi jkl.
M nop qrst uv wx yz.
Ab cde fgh i jkl. Mno
pqr stu vw xyz. A bc
def g hij klm nopqrs
```

```
div {
  column-width: 15em;
  column-gap: 2em;
  column-rule: 4px solid green;
  padding: 5px;
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il layout in CSS

La proprietà `display`

- La proprietà `display` gestisce la natura e l'organizzazione della scatola rispetto a contesto (mondo esterno) e contenuto (mondo interno)
- Ogni elemento HTML ha definita un valore di default per la proprietà `display` che lo caratterizza. Lo cambiamo solo quando serve:
 - `display: block;`
 - `display: inline;`
 - `display: table;`
 - `display: table-row;`
 - `display: table-cell;`
 - `display: list-item;`
- Il modo più semplice e definitivo per nascondere un elemento è:
 - `display: none;`
- Oppure potete fare sparire la scatola esterna tenendo quelle interne:
 - `display: contents;`
- O, infine, ci possiamo fare del layout complesso:
 - `display: grid;`
 - `display: flex;`

Titolo

- 1. primo
- 2. secondo
- 3. terzo

Testo molto lungo. *Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris laoreet. Sed diam dolor, accumsan non, ultricies bibendum, consectetur aliquet, quam. Praesent rutrum.*

Donec ante nulla, placerat in, pretium quis, sodales quis, nibh. Sed tristique velit sed tortor. Maecenas tincidunt semper quam.

- A
- B
- C
- D

Un ulteriore paragrafo di testo.

Il layout

- Il layout è l'organizzazione spaziale delle componenti strutturali più importanti di una pagina web.
- Il layout "naturale" prevede il posizionamento delle scatole secondo il flow di tipo blocco e inline degli elementi HTML di partenza. Questo è raramente ciò che il designer vuole.
- Nel tempo sono state proposte ed usate molte tecniche diverse per ottenere il posizionamento delle scatole secondo le idee del designer. Ne vediamo alcune.

<http://www.fabiovitali.it/TW/2023/layout/>



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Layout 0: no layout

Se non uso niente non posso organizzare sull'asse orizzontale

```
<h1 id="first">Titolo</h1>
<div class="cols">
  <div class="col col1">
    <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
  </div>
  <div class="col col2">
    <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
    <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
  </div>
  <div class="col col3">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
  </div>
</div>
<p id="last">Un ulteriore paragrafo di testo.</p>
```



Layout 0: no layout

Se non uso niente non posso organizzare sull'asse orizzontale

```
<h1 id="first">Titolo</h1>
<div class="cols">
  <div class="col col1">
    <ol><li>primo</li><li>secondo</li><li>
  </div>
  <div class="col col2">
    <p>Testo molto lungo. Lorem ipsum dolor
       adipiscing elit. Mauris. Sed diam dolor
       ultricies bibendum, consectetur , quam
    <p>Donec ante nulla, placerat in, pret
       Sed tristique velit sed tortor. Maecen
    </div>
    <div class="col col3">
      <ul><li>A</li><li>B</li><li>C</li><li>
    </div>
</div>
<p id="last">Un ulteriore paragrafo di testo
```

Titolo

1. primo
2. secondo
3. terzo

Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.

Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.

- A
- B
- C
- D

Un ulteriore paragrafo di testo.

Layout 1: Tabella HTML

Uso di tavelle HTML in maniera un po' creativa. Niente CSS significativo.

```
<table style="height: 300px;" cellpadding="0" cellspacing="0" valign="top">
  <tr>
    <td colspan="4"><h1 id="first">Titolo</h1></td>
  </tr>
  <tr class="middle">
    <td id="left">
      <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
    </td>
    <td id="center">
      <p>Testo molto lungo. Lorem ipsum dolor sit amet,  

        adipiscing elit. Mauris. Sed diam dolor, accumsan ,  

        ultricies bibendum, consectetur , quam. rutrum.</p>
      <p>Donec ante nulla, placerat in, pretium quis, sodales  

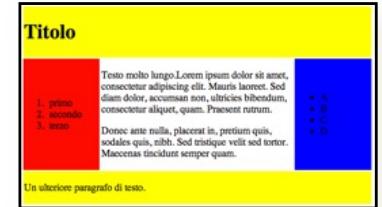
        Sed tristique velit sed tortor. Maecenas semper quam.</p>
    </td>
    <td id="right">
      <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
    </td>
  </tr>
  <tr>
    <td colspan="4"><p id="last">Un ulteriore paragrafo di testo.</p></td>
  </tr>
</table>
```



Nein, nein, nein!

Uso delle HTML in maniera unica e creativa. Il ruolo di CSS significativa.

Layout 2: Float



Uso le proprietà *float* sugli oggetti che debbono muoversi orizzontalmente. Possibile solo fino a tre scatole diverse.

```
<h1 id="first">Titolo</h1>
<div>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
    <p><b>Please note I had to move to second place</b></p>
  </div>
  <div class="col" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
    <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
  </div>
</div>
<p id="last">Un ulteriore paragrafo di testo.</p>
```

```
#left {
  background: red;
  float: left ;
  margin-right: 10px;
}

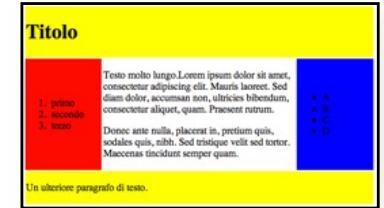
#center {
  width: 100%
}

#right {
  background: blue;
  float: right;
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Layout 3: Positioning



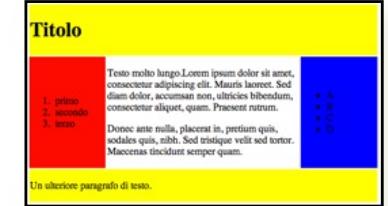
Posiziono in maniera assoluta le scatole nella loro posizione voluta. Possibile ma complesso usare dei valori proporzionali allo spazio disponibile.

```
<h1 id="first">Titolo</h1>
<div>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
    <p><b>Please note I had to move to second place</b></p>
  </div>
  <div class="col" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
    <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
  </div>
</div>
<p id="last">Un ulteriore paragrafo di testo.</p>
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Layout 3: Positioning



Posiziono in maniera assoluta le scatole nella loro posizione voluta. Possibile ma complesso usare dei valori proporzionali allo spazio disponibile.

```
<h1 id="first">Titolo</h1>
<div>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li>
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li>
    <p><b>Please note I had to move to second position</b></p>
  </div>
  <div class="col" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor si...
      adipiscing elit. Mauris. Sed diam dolor, a...
      ultricies bibendum, consectetur , quam. ru...
      <p>Donec ante nulla, placerat in, pretium ...
      Sed tristique velit sed tortor. Maecenas s...
    </div>
  </div>
  <p id="last">Un ulteriore paragrafo di testo.</p>
```

```
#first {
  position: absolute;
  width: 100%;
  top: 0px;
  height: 30px;
  background: yellow;
  margin-top: 10px;
  padding: 10px;
}

#last {
  position: absolute;
  width: 100%;
  top: 370px;
  height: 30px;
  background: yellow;
  margin: 0px;
  padding: 10px;
}
```

```
#left {
  position: absolute;
  top: 60px;
  background: red;
  margin-right: 10px;
}

#center {
  position: absolute;
  top: 60px;
  left: 27%;
  width: 48%;
}

#right {
  position: absolute;
  top: 60px;
  left: 76%;
  background: blue;
```

Layout 4: Tabelle CSS

Titolo	
	Toto molto lungo. Lorem ipsum dolor sit amet, consetetur sadipscing elit. Mauris laoreet. Sed diam dolor, accumsan non, ultricies bibendum, consectetur aliquet, quam. Praesent rutrum.
1. primo	x x
2. secondo	x x
3. terzo	x x
Donec ante nulla, placerat in, pretium quis, sodales quam. Sed tristique velit sed tortor. Maecenas semper quam.	
Un ulteriore paragrafo di testo.	

Uso dei `<div>` come nel posizionamento ma uso la proprietà `display` delle tabelle (`table`, `table-row`, `table-cell`). Piuttosto complicato e non faccio `rowspan` o `colspan`.

```
<h1 id="first">Titolo</h1>
<div class="container">
  <div class="row" id="main">
    <div class="col" id="left">
      <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
    </div>
    <div class="col2" id="center">
      <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
      <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
    </div>
    <div class="col" id="right">
      <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
    </div>
  </div>
</div>
<p id="last">Un ulteriore paragrafo di testo.</p>
```

```
.container {
  display: table;
  table-layout:fixed;
  width: 100%;
}

.row {
  display: table-row;
}

.col {
  display: table-cell;
}

.col2 {
  display: table-cell;
  width: 50%;
}
```

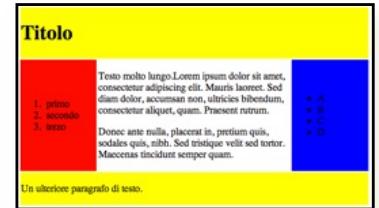
Grid

- Con `display: grid;` si può assegnare ad un elemento un'organizzazione visuale a griglia, organizzata per righe e colonne di altezza e larghezza controllabili.
- I figli dell'elemento con `display: grid;` possono essere assegnati ad una o più elementi della griglia in maniera libera ed indipendente dal document order.
- Posso assegnare altezze diverse ad ogni riga con la proprietà `grid-template-rows` e assegnare larghezze diverse ad ogni riga con la proprietà `grid-template-columns`. Posso creare spazio tra righe e colonne con la proprietà `gap`.
- Ogni elemento figlio dell'elemento `grid` identifica quali colonne e quali righe occupa o per posizione (`grid-row` e `grid-column`) oppure per nome (`grid-area`).



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Layout 5: Grid

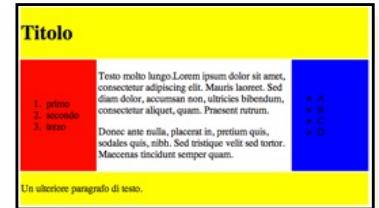


In questo esempio uso le named areas (**grid-template-areas** e **grid-area**) che mi permettono maggior controllo.

```
<div class="container">
  <h1 id="first">Titolo</h1>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
  </div>
  <div class="col2" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
    <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
  </div>
  <p id="last">Un ulteriore paragrafo di testo.</p>
</div>
```



Layout 5: Grid



In questo esempio uso le named areas ([grid-template-areas](#) e [grid-area](#)) che mi permettono ma

```
<div class="container">
  <h1 id="first">Titolo</h1>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li>
  </div>
  <div class="col2" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit ame
    adipiscing elit. Mauris. Sed diam dolor, accums
    ultricies bibendum, consectetur , quam. rutrum.
    <p>Donec ante nulla, placerat in, pretium quis,
    Sed tristique velit sed tortor. Maecenas semper
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></u
  </div>
  <p id="last">Un ulteriore paragrafo di testo.</p>
</div>
```

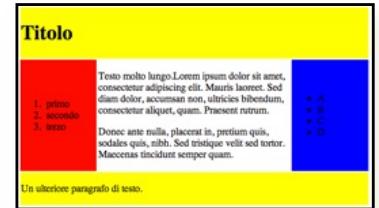
```
.container {
  width: 100%;
  height: 400px;
  display: grid;
  grid-template-areas: "head head head"
                       "left center right"
                       "foot foot foot";
  grid-template-rows: 50px 1fr 50px;
  grid-template-columns: 1fr 2fr 1fr;
}

#first {
  grid-area: head;
  background: yellow;
  padding: 10px;
  margin: 0px;
}

#last {
  grid-area: foot;
  background: yellow;
  padding: 10px;
  margin: 0px;
}
```



Layout 5: Grid



In questo esempio uso le named areas ([grid-template-areas](#) e [grid-area](#)) che mi permettono ma

```
<div class="container">
  <h1 id="first">Titolo</h1>
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li>
  </div>
  <div class="col2" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit ame
    adipiscing elit. Mauris. Sed diam dolor, accums
    ultricies bibendum, consectetur , quam. rutrum.
    <p>Donec ante nulla, placerat in, pretium quis,
    Sed tristique velit sed tortor. Maecenas semper
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></u
  </div>
  <p id="last">Un ulteriore paragrafo di testo.</p>
</div>
```

```
.container {
  width: 100%;
  height: 400px;
  display: grid;
  grid-template-areas: "head head head"
                      "left center right"
                      "foot foot foot";
  grid-template-rows: 50px 1fr 50px;
  grid-template-columns: 1fr 2fr 1fr;
}

#first {
  grid-area: head;
  background: yellow;
  padding: 10px;
  margin: 0px;
}

#left {
  grid-area: left;
  background-color: red;
}

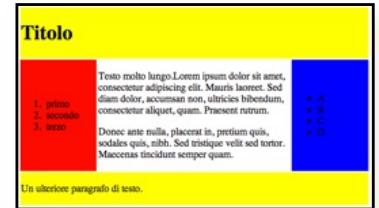
#center {
  grid-area: center;
}

#right {
  grid-area: right;
  background-color: blue;
}
```

Flexbox

- Con **display: flex;** si può assegnare ad un elemento un'organizzazione visuale a contenuti flessibili e che si distribuiscono armonicamente nello spazio disponibile.
- L'elemento con **display flex** impone che i figli siano:
 - organizzati o per righe o per colonne (**flex-direction**),
 - disposti su più righe o colonne separate oppure forzate su una sola (**flex-wrap**)
 - disposti fianco a fianco oppure distribuiti per occupare tutto lo spazio in maniera omogenea (**justify-content**).
- Ogni figlio dell'elemento **flex** può:
 - espandersi o restringersi quanto serve per occupare in maniera controllata lo spazio disponibile (**flex-shrink**, **flex-grow**)
 - cambiare di posizione rispetto al documento (**order**)

Layout 6: Flexbox



In questo esempio organizzo il contenitore "container" per forzare tutto su una riga sola (`wrap: nowrap`), e chiedo alle scatole interne di disporsi su una riga (`direction: row`) e di occupare tutto lo spazio disponibile inserendo spazio vuoto in mezzo tra l'una e le altre (`content: space-between`)

```
<div class="container">
  <h1 id="first">Titolo</h1>
</div>
<div class="container">
  <div class="col" id="left">
    <ol><li>primo</li><li>secondo</li><li>terzo</li></ol>
  </div>
  <div class="col2" id="center">
    <p>Testo molto lungo. Lorem ipsum dolor sit amet, adipiscing elit. Mauris. Sed diam dolor, accumsan , ultricies bibendum, consectetur , quam. rutrum.</p>
    <p>Donec ante nulla, placerat in, pretium quis, sodales Sed tristique velit sed tortor. Maecenas semper quam.</p>
  </div>
  <div class="col" id="right">
    <ul><li>A</li><li>B</li><li>C</li><li>D</li></ul>
  </div>
</div>
<div class="container">
  <p id="last">Un ulteriore paragrafo di testo.</p>
</div>
```

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
  justify-content: space-between;
  width: 100%;
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Altri aspetti di CSS

Altri aspetti di CSS

- Cascata, ereditarietà e `!important`
- Trasformazioni
- `@rules` e media queries
- Animazioni

Ereditarietà

Se non viene specificata una proprietà, CSS assume un valore di default

A parte pochi casi, questo è sempre *inherit*. Questo significa che la proprietà assume lo stesso valore che ha nella scatola contenitore dell'elemento in questione

Qui il contenuto dell'elemento *em* avrà il colore rosso:

```
<p style="color:red;">  
    Qui è <em>in corsivo</em> e qui no.  
</p>
```

Tra i valori non ereditati

display (per HTML è sempre il valore naturale dell'elemento, *block* per *p* o *h1*, *inline* per *b*, *i* o *a*, mentre per XML dipende)

background (sempre *transparent*)

Eccezione alla cascata: !important (1/2)

La keyword **!important** può essere aggiunta in fondo a qualunque statement e aumenta la priorità dello statement anche rispetto a statement successivi attivabili in cascata.

Questo è il primo esempio di cascata, basata su tre fogli di stile, che riportano ciascuno alcune regole:

```
p { font-family: Arial; font-size: 12pt; }
p { color: red; font-size: 11pt; }
p { margin-left: 15pt; color: green; }
```

gli attributi dell'elemento p saranno equivalenti a:

```
p {
  font-family: Arial;
  font-size: 11pt;
  margin-left: 15pt;
  color: green;
}
```

Eccezione alla cascata: !important (2/2)

La keyword **!important** può essere aggiunta in fondo a qualunque statement e aumenta la priorità dello statement anche rispetto a statement successivi attivabili in cascata.

Questo è il primo esempio di cascata, basata su tre fogli di stile, che riportano ciascuno alcune regole. Aggiungendo **!important**:

```
p { font-family: Arial; font-size: 12pt !important; }
p { color: red !important; font-size: 11pt; }
p { margin-left: 15pt; color: green; }
```

gli attributi dell'elemento p saranno equivalenti a:

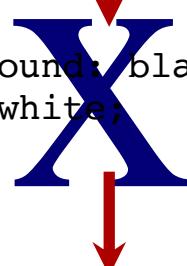
```
p {
  font-family: Arial;
  font-size: 12pt;
  margin-left: 15pt;
  color: red;
}
```

Ma la cascata è davvero una cascata?

```
<html>
<head>
    <title>Esempio CSS</title>
    <style type="text/css">
        @import url(test.css);
    </style>
</head>
<body>
    <h1 class="title">I CSS: questi sconosciuti</h1>
    <p id="p1">Ecco un primo esempio di uso dei CSS.</p>
</body>
</html>
```

accedo via browser

p {
background: black;
color:white;
}



p {
background: gray;
color:blue;
}



I CSS: questi sconosciuti

Ecco un primo esempio di uso dei CSS.

test.css

```
* {
    color: blue !important;
}
p:first-of-type {
    background: gray;
    color: yellow;
}
@media screen {
    p {
        background: black;
        color: white;
    }
}
@media print {
    p {
        width: 8cm;
        margin: 1cm 1.5cm;
    }
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

A cascata sì, ma con calma

L'ordine con cui vengono considerate le varie regole non è dato solo dall'ordine in cui sono posti nei fogli di stile

Infatti, viene applicato un algoritmo di ordinamento sulle dichiarazioni secondo alcuni principi (dal più al meno importante):

- il **media-type** (*print, screen, speech*, ecc.) a cui si riferisce una dichiarazione
- l'**importanza** di una dichiarazione (`! important`)
- l'**origine** della dichiarazione (utente, autore, user agent)
- la **specificità** del selettore della dichiarazione
- l'**ordine** in cui si trovano le dichiarazioni



L'ordine della cascata

La cascata non è ordinata in maniera assoluta sulla base dell'ordine delle dichiarazioni, ma tenendo in considerazione l'importanza e l'origine, con il seguente ordine di precedenza:

1. *dichiarazioni dello user agent*
2. *dichiarazioni dell'utente*
3. *dichiarazioni normali dell'autore*
4. *dichiarazioni importanti dell'autore (!important)*
5. *dichiarazioni importanti dell'utente (!important)*

Le regole della stessa importanza e origine sono ordinate per specificità del selettore, così i selettori più specifici coprono quelli più generali.

Infine regole della stessa importanza e origine e con selettori della stessa specificità sono considerati in *document order*



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

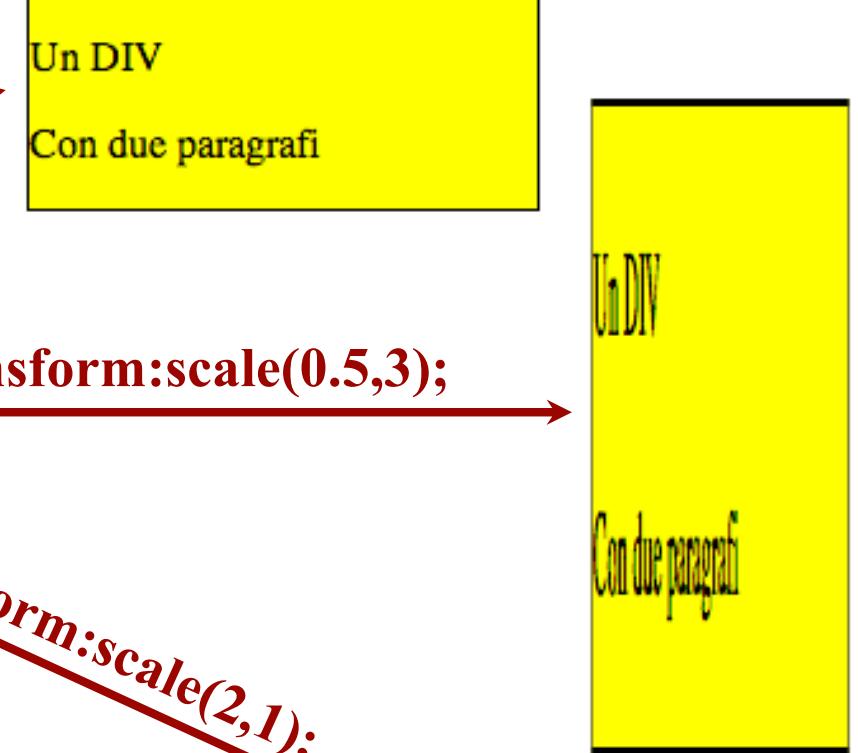
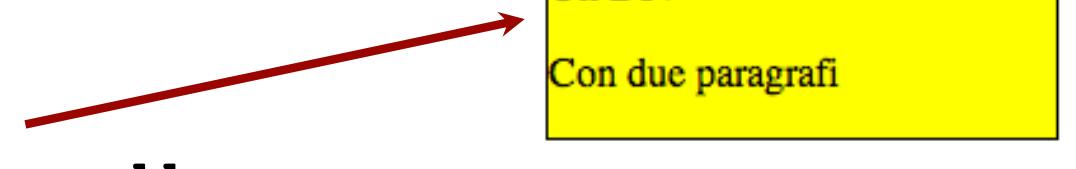
Trasformazioni CSS

- Trasformazioni geometriche sulla scatola una volta che è stata completamente definita e generata.
- Si usa la sintassi **transform: function(parameters);**
- Esempi di funzioni:
 - **translate()**, **translate3d()**: sposta la scatola
 - **scale()**, **scale3d()**: allarga/rimpicciolisce la scatola
 - **rotate()**, **rotate3d()**: ruota la scatola
 - **skew()**: inclina la scatola
 - Ecc.

Trasformazioni CSS: $scale(X,Y)$

```
<style>
div {
    background-color: yellow;
    width: 200px;
    border: 1px solid black;
}
</style>
```

```
<div>
    <p>Un DIV</p>
    <p>Con due paragrafi</p>
</div>
```



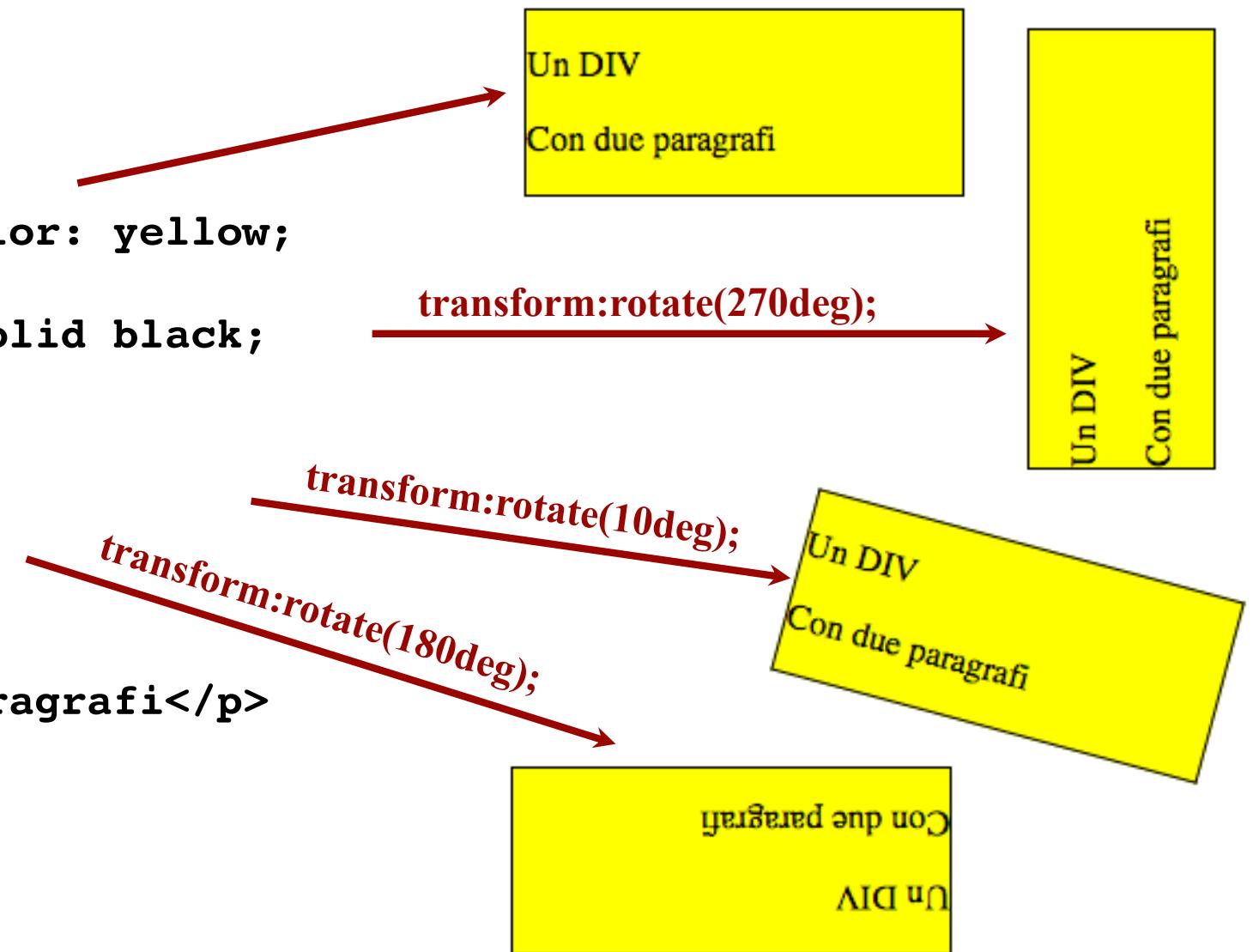
$transform:scale(0.5,3);$

$transform:scale(2,1);$

Trasformazioni CSS: *rotate(deg)*

```
<style>
div {
    background-color: yellow;
    width: 200px;
    border: 1px solid black;
}
</style>
```

```
<div>
    <p>Un DIV</p>
    <p>Con due paragrafi</p>
</div>
```



At rules

Le regole precedute da un "@", comunemente chiamate *at rules*, servono per specificare ambiti o meta-regole del foglio di stile

- **@import**: permette di importare regole da altri stili
- **@charset**: serve per specificare l'encoding (es: UTF-8)
- **@namespace**: permette di definire namespace all'interno di un CSS ed usarli nei selettori
- **@page**: serve per definire caratteristiche di margine dell'intera pagina
- **@font-face**: permette di specificare i font "customizzati" da utilizzare (vengono scaricati automaticamente)
- **@media**: descrive il media-type di destinazione per cui un insieme di statement devono essere applicati
- **@keyframes**: descrive stati iniziali, intermedi e finali di un'animazione in CSS.

@font-face: specificare un font non installato

```
@font-face {  
    font-family: MyFont;  
    src: url("http://www.example.com/font");  
}  
  
h1 {  
    font-family: MyFont, sans-serif;  
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

@media:

specificare regole dipendenti dal device

```
@media screen, projection {  
    html {  
        background: #ffffe0;  
        color: #300;  
    }  
    body {  
        max-width: 35em;  
        margin: 15px;  
    }  
}  
  
@media print {  
    html {  
        background: #fff;  
        color: #000;  
    }  
    body {  
        padding: 1in;  
        border: 0.5pt solid #666;  
    }  
}
```



Media queries

Le *media queries* servono per specificare delle regole particolari che vengono attivate nel caso in cui il supporto usato per visualizzare la pagina Web soddisfa particolari vincoli

Tra le varie espressioni utilizzabili, quelle più comuni permettono di realizzare dei vincoli sulla *larghezza*, *altezza* e *colore* supportati dal dispositivo

Tramite il loro uso, si può adattare la presentazione di una pagina Web a device differenti (pc, smartphone, ecc.) senza cambiare di una virgola il contenuto della pagina

Sintassi:

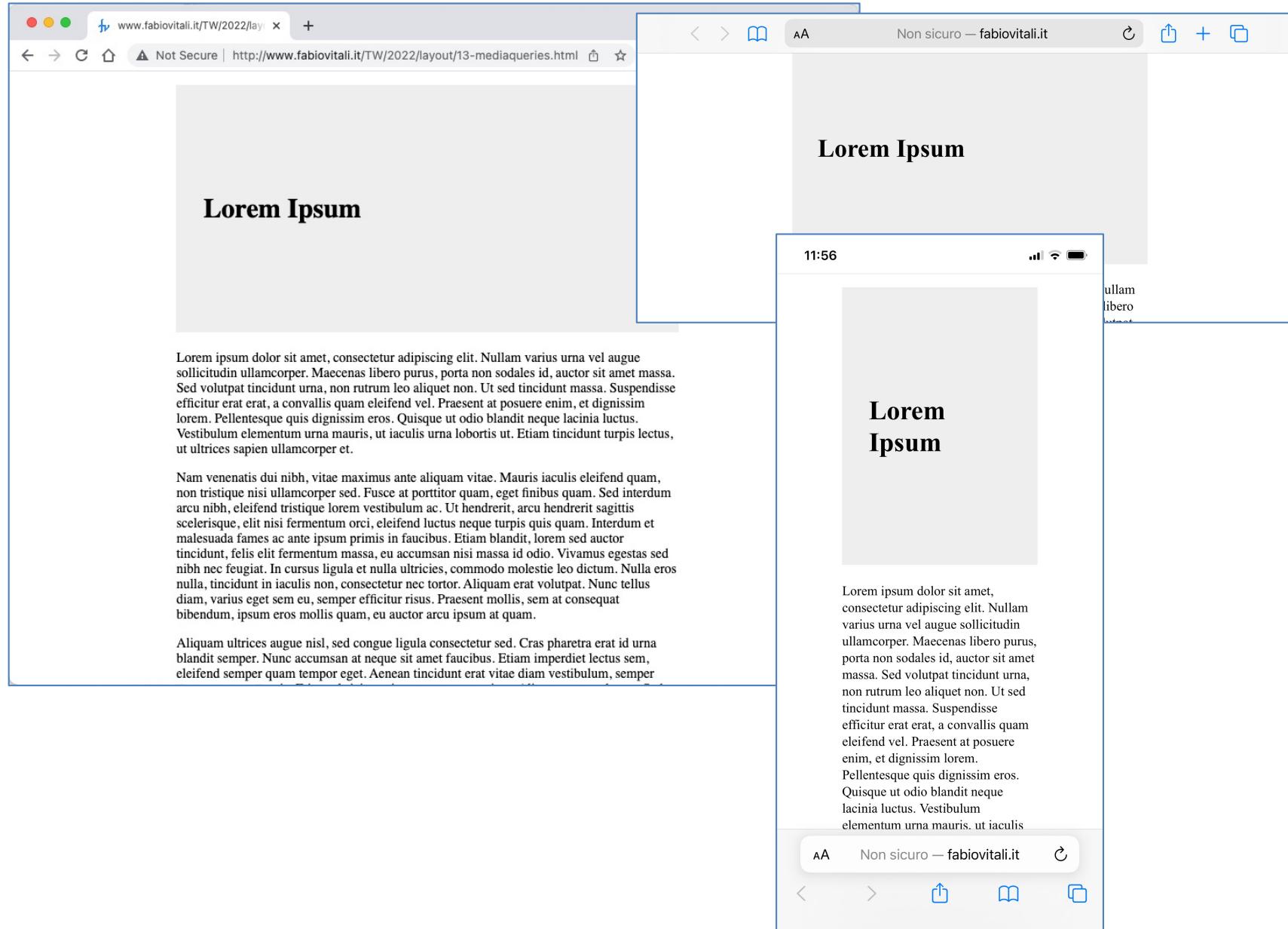
```
@media query {  
    selettore {  
        statement;  
        statement; ...  
    }  
}
```

Media queries

- Il linguaggio per creare media queries è complesso, ed è composto di regole separate, collegate da operatori come and, or, only.
 - **print and screen**
 - **only screen and (max-width: 600px)**
 - **not speech**
- Ci sono più di 30 feature usabili nelle media queries, tra cui any-pointer, aspect-ratio, color, height, hover, max-height, max-width, min-height, min-width, monochrome, orientation, resolution, width, ecc.

Esempio di uso di media query

<http://www.fabiovitali.it/TW/2023/layout/13-mediaqueries.html>



Esempio di uso di media query

<http://www.fabiovitali.it/TW/2023/layout/13-mediaqueries.html>

The image shows a desktop browser window and a mobile browser window side-by-side, both displaying the same HTML content. The desktop browser shows a wide layout with a large header and a main content area. The mobile browser shows a much narrower layout, likely a mobile phone screen, where the content is adapted to fit.

Desktop Browser Content:

```
body {  
    margin: 0px;  
    padding: 1em 20%;  
}  
h1 {  
    background-color: #eeeeee;  
    padding: 4em 1em;  
}  
  
nibh nec feugiat. In cursus ligula et nulla ultricies, commodo molestie leo dictum. Nulla eros nulla, tincidunt in iaculis non, consectetur nec tortor. Aliquam erat volutpat. Nunc tellus diam, varius eget sem eu, semper efficitur risus. Praesent mollis, sem at consequat bibendum, ipsum eros mollis quam, eu auctor arcu ipsum at quam.  
  
Aliquam ultrices augue nisl, sed congue ligula consectetur sed. Cras pharetra erat id urna blandit semper. Nunc accumsan at neque sit amet faucibus. Etiam imperdiet lectus sem, eleifend semper quam tempor eget. Aenean tincidunt erat vitae diam vestibulum, semper
```

Mobile Browser Content:

The mobile browser displays the same content but with a much narrower width, demonstrating how the CSS media queries have adjusted the layout. The header is smaller, and the main content area is also narrower, reflecting the constraints of a mobile device screen.

Esempio di uso di media query

<http://www.fabiovitali.it/TW/2023/layout/13-mediaqueries.html>

The screenshot shows a web browser window with a code editor on the left and two mobile device emulators on the right.

Code Editor:

```
body {  
    margin: 0px;  
    padding: 1em 20%;  
}  
h1 {  
    background-color: #eeeeee;  
    padding: 4em 1em;  
}  
  
@media (max-width: 500px) {  
  
    body {  
        padding: 1em 1em;  
    }  
    h1 {  
        color: #ffffff;  
        background-color: #999999;  
    }  
}
```

Mobile Emulators:

- Emulator 1 (Left):** Shows a desktop-like view of the page with "Lorem Ipsum" text and a background color of #eeeeee.
- Emulator 2 (Middle):** Shows a mobile view of the page with "Lorem Ipsum" text and a background color of #999999. This view is crossed out with a large red 'X'.
- Emulator 3 (Right):** Shows a mobile view of the page with "Lorem Ipsum" text and a background color of #eeeeee.

Page Content:

The page displays "Lorem Ipsum" text and some Latin placeholder text. The main content area has a light gray background. The mobile emulators show the same content but with different styling based on the media query.

Animazioni in CSS

- @keyframes è un blocco di regole per specificare lo stato iniziale (from), lo stato finale (to) ed eventuali stati intermedi (percentuali) di una o più proprietà numeriche CSS.
- Le proprietà più importanti dell'animazione:
 - **animation-name**: blocco keyframes da utilizzare.
 - **animation-delay**: ritardo di partenza
 - **animation-duration**: durata dell'animazione
 - **animation-iteration-count**: numero ripetizioni
 - **animation-timing-function**: la curva di accelerazione.
- La proprietà **animation** è una forma abbreviata di queste proprietà.

Un esempio di animazione

<http://www.fabiovitali.it/TW/2023/layout/11-animation.html>

```
div {  
    padding: 25px;  
    font-size: 130% ;  
    position: relative;  
    animation-name: mymove ;  
    animation-duration: 2000s ;  
    animation-iteration-count: 2;  
    animation-fill-mode: forwards;  
}  
  
@keyframes mymove {  
    from {  
        left: 0%;  
        background-color: #ff4444;  
    }  
    to {  
        left: 75%;  
        background-color: #0044ff;  
    }  
}
```

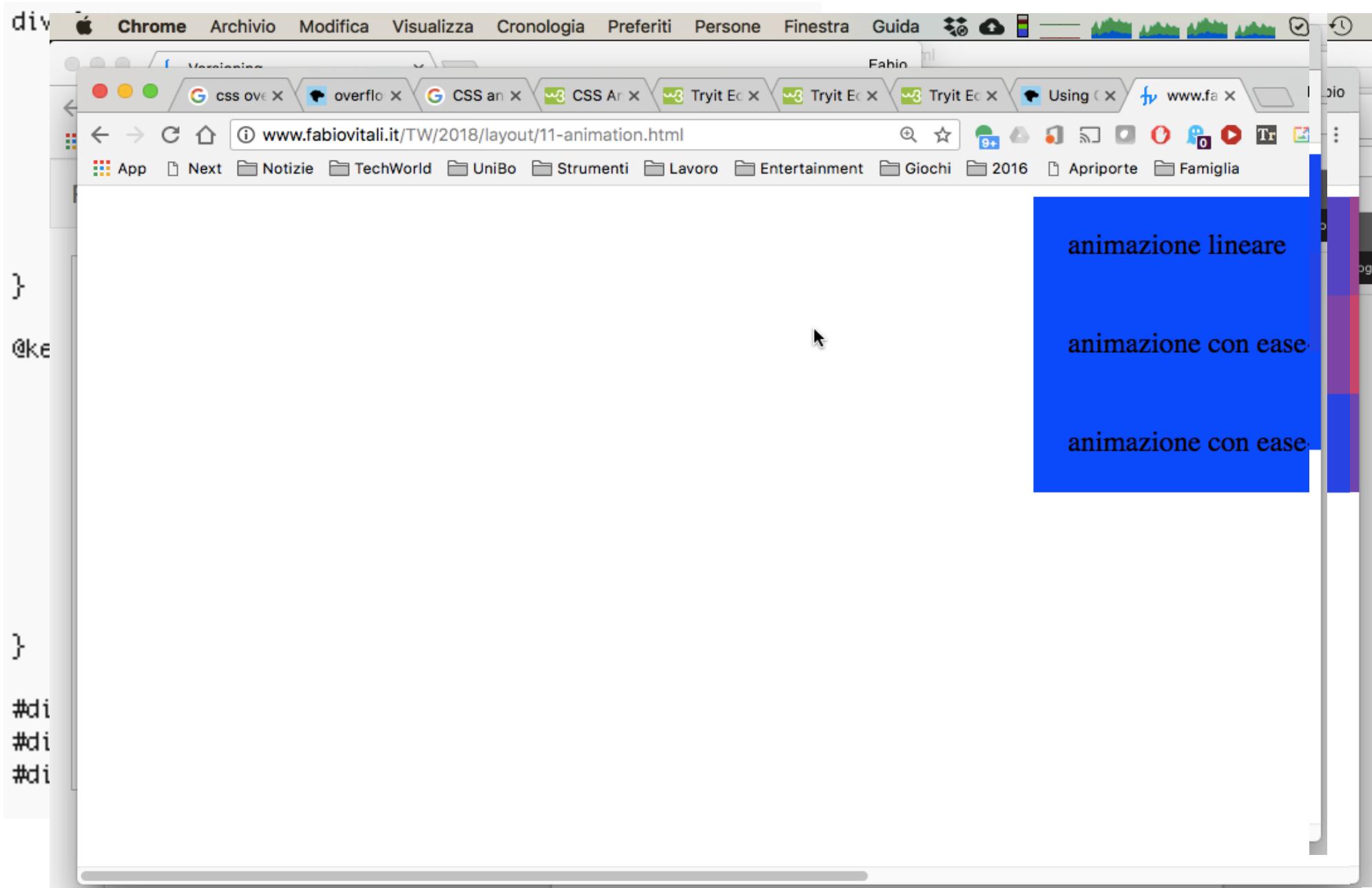
```
#div1 {animation-timing-function: linear;}  
#div2 {animation-timing-function: ease-in;}  
#div3 {animation-timing-function: ease-out;}
```

```
<body>  
    <div id="div1">animazione lineare</div>  
    <div id="div2">animazione con ease-in</div>  
    <div id="div3">animazione con ease-out</div>  
</body>
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Un esempio di animazione



Limiti del CSS

Il CSS ha noti limiti di flessibilità, perché le regole non condividono valori e vanno sempre posti individualmente.

- Sarebbe utile definire lo stesso colore o lo stesso font per molte regole in una volta sola
- Sarebbe utile definire regole sulla base di altre regole
- Sarebbe utile introdurre formule aritmetiche

A partire dal 2009 sono state proposti linguaggi che estendono CSS in questa direzione, sotto forma di pre-processori CSS:

- Attraverso compilazione (cioè il sorgente viene trasformato per generare CSS tradizionale)
- Attraverso interpretazione (cioè il sorgente viene affidato ad uno script client-side – Javascript – per la esecuzione)

Con Level 4 alcuni di questi concetti sono transitati nel CSS standard

LESS, SASS, SCSS

I due (tre) preprocessori più noti.
Ammettono sia compilazione sia
interpretazione.

Feature principali:

- **Variabili**: valori ripetuti più volte;
- **Mix-in**: frammenti di regola utilizzabili in più contesti. Ammettono anche parametri, come le funzioni.
- **Aritmetica**: espressioni numeriche anche complesse

Variabili

```
@color: #4D926F;  
  
#header {  
    color: @color;  
}  
  
h2 {  
    color: @color;  
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

LESS, SASS, SCSS

I due (tre) preprocessori più noti.
Ammettono sia compilazione sia
interpretazione.

Feature principali:

- **Variabili**: valori ripetuti più volte;
- **Mix-in**: frammenti di regola utilizzati in contesti. Ammettono anche parametri e funzioni.
- **Aritmetica**: espressioni numeriche complesse

Variabili

```
@color: #4D926F;
```

Mixin

```
@mixin bordo($color, $width) {  
    border: {  
        color: $color;  
        width: $width;  
        style: dashed;  
    }  
}  
  
p {  
    @include bordo(blue, 15px);  
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

LESS, SASS, SCSS

I due (tre) preprocessori più noti.
Ammettono sia compilazione sia
interpretazione.

Feature principali:

- **Variabili**: valori ripetuti più volte
- **Mix-in**: frammenti di regola utilizzati in contesti. Ammettono anche parametri e funzioni.
- **Aritmetica**: espressioni numeriche complesse

Variabili

```
@color: #4D926F;
```

Mixin

Aritmetica

```
@the-border: 1px;
@base-color: #111111;

#header {
    border-left: @the-border;
    border-right: @the-border * 2;
    color: @base-color * 3;
}
#footer {
    color: @base-color + #003300;
}
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Variabili e calcoli aritmetici in CSS

- **custom property**: una proprietà ad hoc con un valore qualunque (usata come variabile locale). Inizia sempre con '--'.
`--the-border: 1px;
--base-color: #111111;`

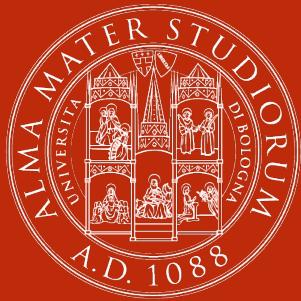
- **:root** : è un selettore per la pseudo classe del document element di HTML, che ha scope globale e non viene mai sovrascritto

```
:root {  
  --the-border: 1px;  
  --base-color: #111111;  
}
```

- **var()** : accede al valore di una custom property in qualunque parte del CSS
- **calc()**: permette calcoli aritmetici (abbastanza limitati)

```
#header {  
  border-left: var(--the-border);  
  border-right: calc( var(--the-border) * 2 );  
  color: calc( var(--base-color) * 3 );  
}
```

N.B.: Poiché il carattere '-' è usato nei nomi CSS, gli operatori aritmetici vanno sempre separati con spazi



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Twitter Bootstrap

Framework CSS

Librerie già pronte con regole associate ad elementi e classi, pronte per essere utilizzate nei documenti HTML con poche o zero modifiche.

Hanno vari pregi:

- Gestiscono le differenze di implementazione delle funzionalità CSS tra i vari browser
- Forniscono accesso facilitato ad effetti speciali (in particolare le animazioni)
- Gestiscono layout responsive (che si adattano alla dimensione dello schermo)
- Forniscono un look integrato, omogeneo e professionale

Limiti

- Tendono ad uniformare pesantemente il look dei siti web



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Twitter Bootstrap

In assoluto il più famoso dei framework CSS. Usato da centinaia di migliaia di siti.

Fornisce :

- Una suite integrata di classi CSS ben omogeneizzata per testi, immagini, form, ecc.
- Gestione di caratteristiche responsive del layout
- Feature di presentazione come tab, barre di navigazione, ecc.
- Un modello posizionale delle scatole basato su divisorie del 12 (così da permettere 2, 3, 4, 6 e 12 scatole ben disposte in un contenitore).
- Classi Javascript standardizzate facilmente utilizzabili anche da autori HTML/CSS.



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Responsive web design

La progettazione di una pagina web in modo da ottimizzare l'esperienza di utilizzo su tutti i device su cui viene presentato.

- Facilità di lettura
- Minimizzazione di ridimensionamento, scrolling e pannellazione

Elementi di un sito web responsive

- Griglia fluida e organizzata su quantità proporzionali alla dimensione dello schermo (dimensioni in % e non in pixel o cm)
- Immagini flessibili (sempre dimensionate dal contenitore e non assolute)
- Uso di @media query (per presentare diversamente la pagina su device e display diversi,

La griglia di Bootstrap

Bootstrap definisce una griglia virtuale divisa in dodicesimi. Ogni elemento occupa una porzione di 1, 2, ... 12 dodicesimi del contenitore, dall'intera finestra fino al più umile elementino.

Inoltre Bootstrap definisce quattro classi di schermi predefiniti (è possibile definirne altri con @media query apposite):

- *xs* (extra small - smartphone), *sm* (small - tablet), *md* (medium - laptop), *lg* (large - schermi Full HD 16/9, 1920x1080 o superiori ecc.)

E' possibile organizzare il layout specificando dimensioni lungo la griglia, e definire dimensioni diverse per schermi diversi.

Sono definite classi per le colonne col-{size}-{12esimi}. Ad esempio col-xs-6 occupa metà della larghezza del contenitore per schermi piccoli.

Due esempi ritrovabili su:

<http://www.fabiovitali.it/TW/2023/bootstrap/>



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Un esempio di griglia

```
<div class="row">
  <div class="col-md-6 col-sm-9 col-xs-12">
    testo
    <div class="row">
      <div class="col-sm-6 col-xs-6">testo</div>
      <div class="col-sm-6 col-xs-6">testo</div>
    </div>
  </div>
  <div class="col-md-6 col-sm-3 col-xs-12">
    testo
  </div>
</div>
```

Un esempio di griglia

Un test sulle griglie

Ridimensiona la dimensione della finestra per vedere l'effetto

<di

Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, due terzi (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.



Sempre metà finché ci sta

Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, un terzo (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.

Il raschietto si vede solo su schermi grandi, medi e piccoli.



```
</div>  
  
<div class="col-md-6 col-sm-3 col-xs-12">  
    testo  
</div>  
</div>
```

Un esempio di griglia

```
<div class="row">
  <div class="col-md-6 col-sm-9 col-xs-12">
    testo
    <div class="col-md-6 col-sm-3 col-xs-12">
      Un test sulle griglie
      RIdimensiona la dimensione della finestra per vedere l'effetto
      <div class="col-md-6 col-sm-3 col-xs-12">
        Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, due terzi (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.
      <div class="col-md-6 col-sm-3 col-xs-12">
        Sempre metà finché ci sta
        Il raschietto si vede solo su schermi grandi, medi e piccoli.
      </div>
    </div>
    <div class="col-md-6 col-sm-3 col-xs-12">
      testo
    </div>
  </div>
</div>
```



Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, un terzo (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.

Un esempio di griglia

```
<div class="row">
  <div class="col-md-6 col-sm-9 col-xs-12">
    testo
    <div class="row">
      <div class="col-sm-6 col-xs-6">testo</div>
      <div class="col-sm-6 col-xs-6">testo</div>
    </div>
  </div>
  <div class="col-md-6 col-sm-3 col-xs-12">
    testo
  </div>
</div>
```

Un test sulle griglie

Ridimensiona la dimensione della finestra per vedere l'effetto

Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, due terzi (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.

Sempre metà finché ci sta

Il raschietto si vede solo su schermi grandi, medi e piccoli.

Questo div occupa metà della larghezza (6/12) su schermi di dimensione media, un terzo (9/12) su schermi piccoli, e l'intera larghezza su schermi molto piccoli.

Navbar e finestre modali

Bootstrap fornisce anche un lungo elenco di servizi grafici, come tabulatori, navbar, finestre modali, pannelli.

- Funzionano su tutti i browser
- Di alta qualità grafica e funzionale
- Usano un po' di Javascript ma sono richiamabili interamente via markup.



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Un esempio di navbar

Applicazione Primo Secondo Terzo A destra

Un test sulle navbar e le modal

Ridimensiona la dimensione della finestra per vedere l'effetto

Intro Capitolo 2 Capitolo 3 ▾

Introduzione

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In in sem nulla. Nulla convallis auctor dictum. Pellentesque finibus lacus vel mauris cursus venenatis. Sed metus risus, ornare sit amet est ut, molestie laoreet risus. Proin ac urna a felis rutrum accumsan at id massa. Etiam rutrum enim eget diam cursus euismod. Vestibulum scelerisque finibus euismod. Duis erat libero, efficitur quis fermentum tempus, bibendum vitae lorem. Duis viverra sem et maximus tincidunt. Quisque ut sapien congue sem bibendum hendrerit non semper sapien. Ut mattis dapibus nibh at sollicitudin. Suspendisse et mauris at ex posuere ultrices.

Ridimensiona la dimensione della finestra per vedere l'effetto

Intro Capitolo 2 Capitolo 3 ▾

Introduzione

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In in sem nulla. Nulla convallis auctor dictum. Pellentesque finibus lacus vel mauris cursus venenatis. Sed metus risus, ornare sit amet est ut, molestie laoreet risus. Proin ac urna a felis rutrum accumsan at id massa. Etiam rutrum enim eget diam cursus euismod. Vestibulum scelerisque finibus euismod. Duis erat libero, efficitur quis fermentum tempus, bibendum vitae lorem. Duis viverra sem et maximus tincidunt. Quisque ut sapien congue sem bibendum hendrerit non semper sapien. Ut mattis dapibus nibh at sollicitudin. Suspendisse et mauris at ex posuere ultrices.

Tailwind

- Una recente (2021) libreria CSS che
 - abbandona il concetto di **componenti** semanticamente caratterizzate (ad esempio "container", "row", "modal", "button"), e
 - introduce le **utility**, semplicissime classi CSS in rapporto 1-1 con proprietà e valori CSS da usare in grande numero direttamente dentro l'HTML.
- La maggior parte delle classi introducono un'unica proprietà ed un unico valore, e molto hanno un prefisso che introduce una media query sulla dimensione dello schermo.

Tailwind

- Una recente (2021) librerie

```
<style>
  .card {
    border-radius: 0.375rem;
    border: 1px solid rgb(115, 115, 115);
    background-color: rgb(64, 64, 64);
    color: rgb(245, 245, 245);
    display: flex;
    flex-direction: row;
    align-items: flex-start;
    cursor: pointer;
  }

  .card:hover {
    background-color: rgb(82, 82, 82);
  }

  .card-img {
    padding: 1rem 0rem 1rem 1rem;
  }

  .card-contents {
    margin: 0.75rem 1rem;
  }
</style>
```

```
<section class="card">
  
  <p class="card-contents">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
  </p>
</section>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Tailwind



 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

21) libreri

```
<style>
  .card {
    border-radius: 0.375rem;
    border: 1px solid rgb(115, 115, 115);
    background-color: rgb(64, 64, 64);
    color: rgb(245, 245, 245);
    display: flex;
    flex-direction: row;
    align-items: flex-start;
    cursor: pointer;
  }
  .card:hover {
    background-color: rgb(82, 82, 82);
  }
  .card-img {
    padding: 1rem 0rem 1rem 1rem;
  }
  .card-contents {
    margin: 0.75rem 1rem;
  }
</style>
```

```
<section class="card">
  
  <p class="card-contents">Lorem ipsum
    dolor sit amet, consectetur adipiscing
    elit, sed do eiusmod tempor incididunt ut
    labore et dolore magna aliqua. Ut enim ad
    minim veniam, quis nostrud exercitation
    ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  </p>
</section>

<section class="rounded-md border border-neutral-500 bg-neutral-700 hover:bg-neutral-600 text-neutral-100 flex flex-row items-start cursor-pointer">
  
  <p class="mx-4 my-3"> Lorem ipsum
    dolor sit amet, consectetur adipiscing
    elit, sed do eiusmod tempor incididunt ut
    labore et dolore magna aliqua. Ut enim ad
    minim veniam, quis nostrud exercitation
    ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  </p>
</section>
```

Due esercizi

- Prendere l'esercizio Tropical Islands
<https://www.fabiovitali.it/TW/2023/tropical/>
che usa Bootstrap 4.0.0
- Farne una versione aggiornata all'ultima versione di Bootstrap
- Creare una versione con Tailwind

Bibliografia

- Cascading Style Sheets, level 1, W3C Recommendation 17 Dec 1996, revised 11 Apr 2008, <http://www.w3.org/TR/CSS1/>
- Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, W3C Working Recommendation 07 June 2011, <http://www.w3.org/TR/CSS2/>
- Selectors Level 3, W3C W3C Recommendation 29 September 2011, <http://www.w3.org/TR/css3-selectors/>
- CSS Color Module Level 3, W3C W3C Recommendation 07 June 2011, <http://www.w3.org/TR/css3-color/>
- CSS Namespaces Module, W3C W3C Recommendation 29 September 2011, <http://www.w3.org/TR/css3-namespace/>
- CSS Multi-column Layout Module, W3C Candidate Recommendation 12 April 2011, <http://www.w3.org/TR/css3-multicol/>
- Media Queries, W3C Recommendation 19 June 2012, <http://www.w3.org/TR/css3-mediaqueries/>



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

www.unibo.it